Computational Natural Language Semantics
320441–Fall 2015
Lecture Notes

Michael Kohlhase

School of Engineering & Science
Jacobs University, Bremen Germany
m.kohlhase@jacobs-university.de

April 28, 2015

Preface

This Document

This document contains the course notes for the course Computational Semantics of Natural Language held at Jacobs University Bremen in the fall semesters 2006/08/10/12.

Contents: The document mixes the slides presented in class with comments of the instructor to give students a more complete background reference.

Caveat: This document is made available for the students of this course only. It is still an early draft, and will develop over the course of the course. It will be developed further in coming academic years.

Licensing: This document is licensed under a Creative Commons license that requires attribution, forbids commercial use, and allows derivative works as long as these are licensed under the same license.

Knowledge Representation Experiment:

This document is also an experiment in knowledge representation. Under the hood, it uses the sTeX package [Koh08, Koh15], a TeX/LaTeX extension for semantic markup, which allows to export the contents into the eLearning platform PantaRhei.

Comments and extensions are always welcome, please send them to the author.

Comments: Comments and extensions are always welcome, please send them to the author.

Acknowledgments

Materials: Some of the material in this course is based on a course "Formal Semantics of Natural Language" held jointly with Prof. Mandy Simons at Carnegie Mellon University in 2001.

ComSem Students: The following students have submitted corrections and suggestions to this and earlier versions of the notes: Bastian Laubner, Ceorgi Chulkov, Stefan Anca and Elena Digor.

Contents

1 Getting Started with "Computational Semantics of Natural Language"

The Course 320441 "Computational Semantics of Natural Language" is a one-semester introductory course that provides an overview over logic-based semantics of natural language. It follows the "method of fragments" introduced by Richard Montague, and builds a sequence of fragments of English with increasing coverage and a sequence of logics that serve as target representation formats. The course can be seen as both a course on semantics and as a course on applied logics.

1.1 Administrativa

We will now go through the ground rules for the course. This is a kind of a social contract between the instructor and the students. Both have to keep their side of the deal to make the acquaintance with research in Natural Language Semantics as efficient and painless as possible.

1.1.1 Grades, Credits, Retaking

Now we come to a topic that is always interesting to the students: the grading scheme.

Prerequisites, Requirements, Grades

- Prerequisites: Motivation, Interest, Curiosity, hard work (mainly,...)
  - experience in (some) logics (catch up with CompLog notes if needed)

  You can do this course if you want! (even without those, but they help)
- Grades: (plan your work involvement carefully)

Course		Lab	
Attendance and Wakefulness	10%	Attendance and Wakefulness	10%
Homework Assignments	60%	Graded Lab Assignments	50%
Quizzes	30%		
No Midterm Exam	–	Intersession Project	20%
No Final Exam	–	Discussion	20%

  In particular, no midterm, and no final in the Lab, but attendance is mandatory! (excuses possible)
- Note that for the grades, the percentages of achieved points are added with the weights above, and only then the resulting percentage is converted to a grade.

SOME RIGHTS RESERVED ©: Michael Kohlhase 1 JACOBS UNIVERSITY

Our main motivation in this grading scheme is to entice you to study continuously. This means that you will have to stay involved, do all your homework assignments, and keep abreast with the course. This also means that your continued involvement may be higher than other (graduate) courses, but you are free to concentrate on these during exam week.

1.1.2 Homeworks, Submission, and Cheating

Homework assignments

- Goal: Reinforce and apply what is taught in class.
- Homeworks: will be small individual problem/programming/proof assignments (but take time to solve) group submission if and only if explicitly permitted
- Admin: To keep things running smoothly
  - Homeworks will be posted on PantaRhei
  - Homeworks are handed in electronically in `JGrader`(plain text, Postscript, PDF,...)
  - go to the tutorials, discuss with your TA (they are there for you!)
  - materials: sometimes posted ahead of time; then read before class, prepare questions, bring printout to class to take notes
- Homework Discipline:
  - start early! (many assignments need more than one evening's work)
  - Don't start by sitting at a blank screen
  - Humans will be trying to understand the text/code/math when grading it.

SOME RIGHTS RESERVED ©: Michael Kohlhase 2 JACOBS UNIVERSITY

Homework assignments are a central part of the course, they allow you to review the concepts covered in class, and practice using them. They are usually directly based on concepts covered in the lecture, so reviewing the course notes often helps getting started.

Homework Submissions, Grading, Tutorials

- Submissions: We use Heinrich Stamerjohanns' `JGrader` system
  - submit all homework assignments electronically to `https://jgrader.de`.
  - you can login with your Jacobs account and password. (should have one!)
  - feedback/grades to your submissions
  - get an overview over how you are doing! (do not leave to midterm)
- Tutorials: select a tutorial group and actually go to it regularly
  - to discuss the course topics after class (lectures need pre/postparation)
  - to discuss your homework after submission (to see what was the problem)
  - to find a study group (probably the most determining factor of success)

SOME RIGHTS RESERVED ©: Michael Kohlhase 3 JACOBS UNIVERSITY

The next topic is very important, you should take this very seriously, even if you think that this is just a self-serving regulation made by the faculty.

All societies have their rules, written and unwritten ones, which serve as a social contract among its members, protect their interestes, and optimize the functioning of the society as a whole. This is also true for the community of scientists worldwide. This society is special, since it balances intense cooperation on joint issues with fierce competition. Most of the rules are largely unwritten; you are expected to follow them anyway. The code of academic integrity at Jacobs is an attempt to put some of the aspects into writing.

It is an essential part of your academic education that you learn to behave like academics, i.e. to function as a member of the academic community. Even if you do not want to become a scientist in the end, you should be aware that many of the people you are dealing with have gone through an academic education and expect that you (as a graduate of Jacobs) will behave by these rules.

The Code of Academic Integrity

- Jacobs has a "Code of Academic Integrity"
  - this is a document passed by the Jacobs community (our law of the university)
  - you have signed it during enrollment (we take this seriously)
- It mandates good behaviors from both faculty and students and penalizes bad ones:
  - honest academic behavior (we don't cheat/falsify)
  - respect and protect the intellectual property of others (no plagiarism)
  - treat all Jacobs members equally (no favoritism)
- this is to protect you and build an atmosphere of mutual respect
  - academic societies thrive on reputation and respect as primary currency
- The Reasonable Person Principle (one lubricant of academia)
  - we treat each other as reasonable persons
  - the other's requests and needs are reasonable until proven otherwise
  - but if the other violates our trust, we are deeply disappointed (severe uncompromising consequences)

SOME RIGHTS RESERVED ©: Michael Kohlhase 4 JACOBS UNIVERSITY

To understand the rules of academic societies it is central to realize that these communities are driven by economic considerations of their members. However, in academic societies, the primary good that is produced and consumed consists in ideas and knowledge, and the primary currency involved is academic reputation[1]. Even though academic societies may seem as altruistic — scientists share their knowledge freely, even investing time to help their peers understand the concepts more deeply — it is useful to realize that this behavior is just one half of an economic transaction. By publishing their ideas and results, scientists sell their goods for reputation. Of course, this can only work if ideas and facts are attributed to their original creators (who gain reputation by being cited). You will see that scientists can become quite fierce and downright nasty when confronted with behavior that does not respect other's intellectual property.

1.1.3 Resources

[1]Of course, this is a very simplistic attempt to explain academic societies, and there are many other factors at work there. For instance, it is possible to convert reputation into money: if you are a famous scientist, you may get a well-paying job at a good university,. . .

Textbooks, Handouts and Information, Forum

- No required textbook, but course notes, posted slides
- Course notes in PDF will be posted at `http://kwarc.info/teaching/ComSem.html`
- Everything will be posted on PantaRhei (Notes+assignments+course forum)
  - announcements, contact information, course schedule and calendar
  - discussion among your fellow students(careful, I will occasionally check for academic integrity!)
  - `http://panta.kwarc.info` (follow instructions there)

SOME RIGHTS RESERVED ©: Michael Kohlhase 5 JACOBS UNIVERSITY

No Textbook: Due to the special circumstances discussed above, there is no single textbook that covers the course. Instead we have a comprehensive set of course notes (this document). They are provided in two forms: as a large PDF that is posted at the course web page and on the PantaRhei system. The latter is actually the preferred method of interaction with the course materials, since it allows to discuss the material in place, to play with notations, to give feedback, etc. The PDF file is for printing and as a fallback, if the PantaRhei system, which is still under development develops problems.

Infrastructure

- There are two text books for the course.
  - Patrick Blackburn & Johan Bos *Representation and Inference for Natural Language A First Course in Computational Semantics* [BB05]
  - L.T.F. Gamut. *Logic, Language and Meaning: Volume 2, Intensional logic and Logical Grammar*. University of Chicago Press [Gam91b]. (Volume 1 [Gam91a] is a nice recap!)
- both are/will be in the library, the first one can be bought for Euro 26.50 at `http://www.blackburnbos.org`

SOME RIGHTS RESERVED ©: Michael Kohlhase 6 JACOBS UNIVERSITY

Both books will be useful references for you. The Blackburn&Bos book covers the semantics construction. The Gamut book is a good reference, and will give you some of the same content from a different perspective, with more emphasis on the logic.

1.2 An Introduction to Natural Language Semantics

In this subsection we will introduce the topic of this course and situate it in the larger field of natural language understanding. But before we do that, let us briefly step back and marvel at the wonders of natural language, perhaps one of the most human of abilities.

Fascination of Language

▷ Even more so than thinking, language is a skill that only humans have.

▷ It is a miracle that we can express complex thoughts in a sentence in a matter of seconds.

▷ It is no less miraculous that a child can learn tens of thousands of words and a complex grammar in a matter of a few years.

©: Michael Kohlhase 7

With this in mind, we will embark on the intellectual journey of building artificial systems that can process (and possibly understand) natural language as well.

Language and Information

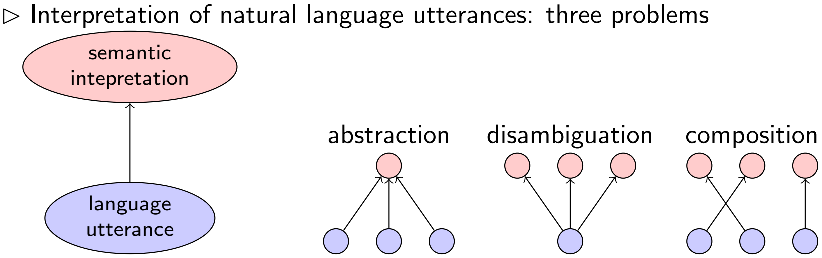
▷ humans use words (sentences, texts) in natural languages to represent information

▷ but:

▷ what really counts is not the words themselves, but the meaning information they carry.

▷ for questions/answers, it would be very useful to find out what words (sentences/texts) mean.

▷ Interpretation of natural language utterances: three problems



©: Michael Kohlhase 8

Meaning of Natural Language; e.g. Machine Translation

▷ Idee: Machine Translation is very simple! (we have good lexica)

▷ **Example 1.1** *Peter liebt Maria.* $\leadsto$ *Peter loves Mary.*

▷ ⚠ this only works for simple examples

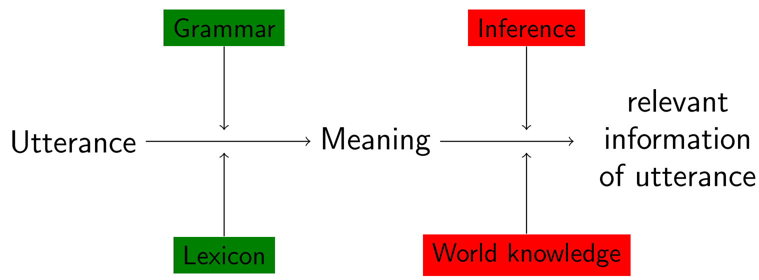
▷ **Example 1.2** *Wirf der Kuh das Heu über den Zaun.*
$\not\leadsto$ *Throw the cow the hay over the fence.* (differing grammar)

▷ ⚠ Grammar is not the only problem

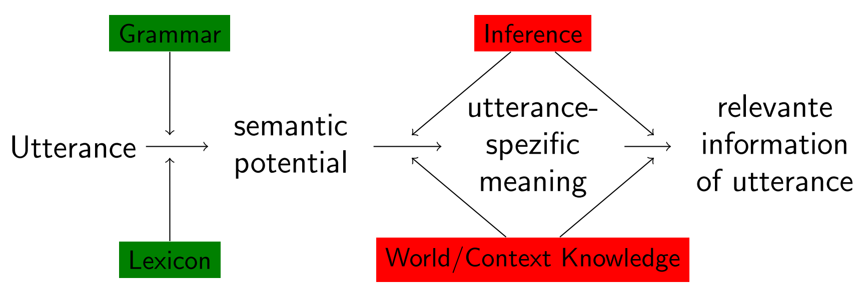
- ▷ *Der Geist ist willig, aber das Fleisch ist schwach!*
- ▷ *Der Schnaps ist gut, aber der Braten ist verkocht!*

▷ We have to understand the meaning!

▷ *The lecture begins at 11:00 am .*



▷ *it starts at eleven.*



Semantics is not a Cure-It-All!

How many animals of each species did Moses take onto the ark?

▷ Actually, it was Noah (But you understood the question anyways)

▷ The only thing that currently really helps is a restricted domain

- ▷ restricted vocabulary
- ▷ restricted world model

▷ Demo: Bahnauskunft unter 0241-604020

▷ Demo: DBPedia `http://wikipedia.aksw.org/`

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

1.2.1 Natural Language Understanding as Engineering

Even though this course concentrates on computational aspects of natural language semantics, it is useful to see it in the context of the field of natural language processing.

Language Technology

▷ Language Assistance

- ▷ written language: Spell-/grammar-/style-checking
- ▷ spoken language: dictation systems and screen readers
- ▷ multilingual text: machine-supported text and dialog translation, eLearning

▷ Dialog Systems

- ▷ Information Systems: at airport, tele-banking, e-commerce, call centers
- ▷ Dialog interfaces for computers, robots, cars

▷ Information management:

- ▷ search and classification of documents
- ▷ information extraction, question answering.

SOME RIGHTS RESERVED ©: Michael Kohlhase 13 JACOBS UNIVERSITY

The field of natural language processing (NLP) is an engineering field at the intersection of computer science, artificial intelligence, and linguistics which is concerned with the interactions between computers and human (natural) languages. Many challenges in NLP involve natural language understanding– that is, enabling computers to derive meaning (representations) from human or natural language input; this is the wider setting in our course. The dual side of NLP: natural language generation which aims at generating natural language or speech from meaning representation requires similar foundations, but different techniques is less relevant for the purposes of this course.[1] EdN:1

What is Natural Language Processing?

▷ Generally: Studying of natural languages and development of systems that can use/generate these.

▷ Here: Understanding natural language (but also generation: other way around)

0) speech processing: acoustic signal $\rightsquigarrow$ word net
1) syntactic processing: word sequence $\rightsquigarrow$ phrase structure
2) semantics construction: phrase structure $\rightsquigarrow$ (quasi-)logical form
3) semantic-pragmatic analysis:
   (quasi-)logical form $\rightsquigarrow$ knowledge representation
4) problem solving: using the generated knowledge (application-specific)

▷ In this course: steps 2) and 3).

SOME RIGHTS RESERVED ©: Michael Kohlhase 14 JACOBS UNIVERSITY

The waterfall model shown above is of course only an engineering-centric model of natural language understanding and not to be confused with a cognitive model; i.e. an account of what happens in human cognition. Indeed, there is a lot of evidence that this simple sequential processing model is not adequate, but it is the simplest one to implement and can therefore serve as a background reference to situating the processes we are interested in.

There are currently two[2] EdN:2

What is the State of the Art In NLU?

▷ Two avenues of of attack for the problem: knowledge-based and statistical techniques (they are complementary)

[1] EdNote: mark up the keywords below with links.
[2] EdNote: continue; give more detailed overview

Deep	Knowledge-based We are here	Not there yet cooperation?
Shallow	no-one wants this	Statistical Methods applications
Analysis ↑ vs. Coverage →	narrow	wide

▷ We will cover foundational methods of deep processing in the course and a mixture of deep and shallow ones in the lab.

SOME RIGHTS RESERVED ©: Michael Kohlhase 15 JACOBS UNIVERSITY

1.2.2 Computational Semantics as a Natural Science

Overview: Formal natural language semantics is an approach to the study of meaning in natural language which utilizes the tools of logic and model theory. Computational semantics adds to this the task of representing the role of inference in interpretation. By combining these two different approaches to the study of linguistic interpretation, we hope to expose you (the students) to the best of both worlds.

Computational Semantics as a Natural Science

▷ In a nutshell: Logic studies formal languages, their relation with the world (in particular the truth conditions). Computational logic adds the question about the computational behavior of the relevant functions of the formal languages.

▷ This is almost the same as the task of natural language semantics!

▷ It is one of the key ideas that logics are good scientific models for natural languages, since they simplify certain aspects so that they can be studied in isolation. In particular, we can use the general scientific method of

1. observing
2. building formal theories for an aspect of reality,
3. deriving the consequences of the assumptions about the world in the theories
4. testing the predictions made by the model against the real-world data. If the model predicts the data, then this confirms the model, if not, we refine the model, starting the process again at 2.

SOME RIGHTS RESERVED ©: Michael Kohlhase 16 JACOBS UNIVERSITY

Excursion: In natural sciences, this is established practice; e.g. astronomers observe the planets, and try to make predictions about the locations of the planets in the future. If you graph the location over time, it appears as a complicated zig-zag line that is difficult to understand. In 1609 Johannes Kepler postulated the model that the planets revolve around the sun in ellipses, where the sun is in one of the focal points. This model made it possible to predict the future whereabouts of the planets with great accuracy by relatively simple mathematical computations. Subsequent observations have confirmed this theory, since the predictions and observations match.

Later, the model was refined by Isaac Newton, by a theory of gravitation; it replaces the Keplerian assumptions about the geometry of planetary orbits by simple assumptions about gravitational forces (gravitation decreases with the inverse square of the distance) which entail the geometry.

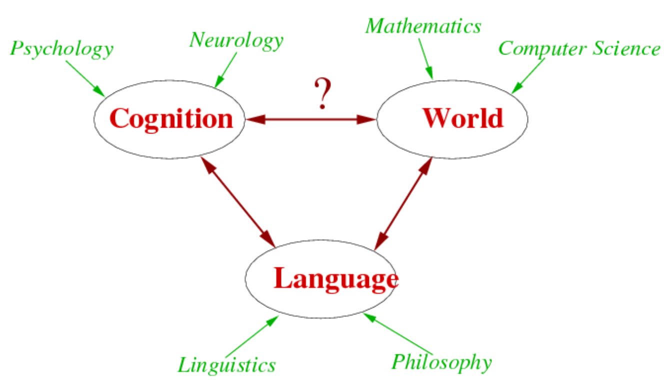
Even later, the Newtonian theory of celestial mechanics was replaced by Einstein's relativity theory, which makes better predictions for great distances and high-speed objects.

All of these theories have in common, that they build a mathematical model of the physical reality, which is simple and precise enough to compute/derive consequences of basic assumptions, that can be tested against observations to validate or falsify the model/theory.

The study of natural language (and of course its meaning) is more complex than natural sciences, where we only observe objects that exist independently of ourselves as observers. Language is an inherently human activity, and deeply interdependent with human cognition (it is arguably one of its motors and means of expression). On the other hand, language is used to communicate about phenomena in the world around us, the world in us, and about hypothetical worlds we only imagine.

Therefore, natural language semantics must necessarily be an intersective discipiline and a trans-disciplinary endeavor, combining methods, results and insights from various disciplines.

NL Semantics as an Intersective Discipline



©: Michael Kohlhase 17

1.2.3 Looking at Natural Language

The next step will be to make some observations about natural language and its meaning, so that we get and intuition of what problems we will have to overcome on the way to modeling natural language.[3] EdN:3

[3] EdNote: introduce meaning by truth conditions and consequences as an analysis tool.

Fun with Diamonds (are they real?) [Dav67b]

- ▷ *This is a blue diamond* ($\models$ *diamond*, $\models$ *blue*)
- ▷ *This is a big diamond* ($\models$ *diamond*, $\not\models$ *big*)
- ▷ *This is a fake diamond* ($\not\models$ *diamond*)
- ▷ *This is a fake blue diamond* ($\models$ *blue*?, $\models$ *diamond*?)
- ▷ *Mary knows that this is a diamond* ($\models$ *diamond*)
- ▷ *Mary believes that this is a diamond* ($\not\models$ *diamond*)

SOME RIGHTS RESERVED ©: Michael Kohlhase 18 JACOBS UNIVERSITY

Logical analysis vs. conceptual analysis: These examples — Mostly borrowed from [Dav67b]— help us to see the difference between logical analysis and conceptual analysis. We observed that from *This is a big diamond.* we cannot conclude *This is big*. Now consider the sentence *Jane is a beautiful dancer*. Similarly, it does not follow from this that Jane is beautiful, but only that she dances beautifully. Now, what it is to be beautiful or to be a beautiful dancer is a complicated matter. To say what these things are is a problem of conceptual analysis. The job of semantics is to uncover the logical form of these sentences. Semantics should tell us that the two sentences have the same logical forms; and ensure that these logical forms make the right predictions about the entailments and truth conditions of the sentences, specifically, that they don't entail that the object is big or that Jane is beautiful. But our semantics should provide a distinct logical form for sentences of the type: *This is a fake diamond.* From which it follows that the thing is fake, but not that it is a diamond.

Ambiguity (It could mean more than one thing)

- ▷ *John went to the bank* (river or financial?)
- ▷ *You should have seen the bull we got from the pope* (three-way!)
- ▷ *I saw her duck* (animal or action?)
- ▷ *John chased the gangster in the red sports car* (three-way too!)

SOME RIGHTS RESERVED ©: Michael Kohlhase 19 JACOBS UNIVERSITY

One way to think about the examples of ambiguity on the previous slide is that they illustrate a certain kind of indeterminacy in sentence meaning. But really what is indeterminate here is what sentence is represented by the physical realization (the written sentence or the phonetic string). The symbol *duck* just happens to be associated with two different things, the noun and the verb. Figuring out how to interpret the sentence is a matter of deciding which item to select. Similarly for the syntactic ambiguity represented by PP attachment. Once you, as interpreter, have selected one of the options, the interpretation is actually fixed. (This doesn't mean, by the way, that as an interpreter you necessarily do select a particular one of the options, just that you can.)

A brief digression: Notice that this discussion is in part a discussion about compositionality, and gives us an idea of what a non-compositional account of meaning could look like. The Radical Pragmatic View is a non-compositional view: it allows the information content of a sentence to be fixed by something that has no linguistic reflex.

To help clarify what is meant by compositionality, let me just mention a couple of other ways in which a semantic account could fail to be compositional.

- Suppose your syntactic theory tells you that S has the structure $[a[bc]]$ but your semantics computes the meaning of S by first combining the meanings of a and b and then combining the result with the meaning of c. This is non-compositional.

- Recall the difference between:[4] EdN:4

  1. Jane knows that George was late.
  2. Jane believes that George was late.

  Sentence 1entails that George was late; sentence 2doesn't. We might try to account for this by saying that in the environment of the verb *believe*, a clause doesn't mean what it usually means, but something else instead. Then the clause *that George was late* is assumed to contribute different things to the informational content of different sentences. This is a non-compositional account.

Quantifiers, Scope and Context

- ▷ *Every man loves a woman* (Keira Knightley or his mother!)
- ▷ *Every car has a radio* (only one reading!)
- ▷ **Example 1.3** *Some student in every course sleeps in every class at least some of the time* (how many readings?)
- ▷ **Example 1.4** *The president of the US is having an affair with an intern* (2002 or 2000?)
- ▷ **Example 1.5** *Everyone is here* (who is everyone?)

SOME RIGHTS RESERVED ©: Michael Kohlhase 20 JACOBS UNIVERSITY

Observation: If we look at the first sentence, then we see that it has two readings[5]:[6] EdN:5 EdN:6

1. there is one woman who is loved by every man.
2. for each man there is one woman whom he loves.

These correspond to distinct situations (or possible worlds) that make the sentence true.

Observation: For the second example we only get one reading: the analogue of 2. The reason for this lies not in the logical structure of the sentence, but in concepts involved. We interpret the meaning of the word *has*[7] as the relation "has as physical part", which in our world carries a certain uniqueness condition: If a is a physical part of b, then it cannot be a physical part of c, unless b is a physical part of c or vice versa. This makes the structurally possible analogue to 1 impossible in our world and we discard it. EdN:7

Observation:

In the examples above, we have seen that (in the worst case), we can have one reading for every ordering of the quantificational phrases in the sentence. So, in the third example, we have four of them, we would get $4! = 12$ readings. It should be clear from introspection[8] that we (humans) do not entertain 12 readings when we understand and process this sentence. Our models should account for such effects as well. EdN:8

[4] EdNote: restore label/ref when that works again
[5] EdNote: explain the term "reading" somewhere
[6] EdNote: restore label/ref when this works again
[7] EdNote: fix the nlex macro, so that it can be used to specify which example a fragment has been taken from.
[8] EdNote: explain somewhere and reference

Context and Interpretation: It appears that the last two sentences have different informational content on different occasions of use. Suppose I say *Everyone is here.* at the beginning of class. Then I mean that everyone who is meant to be in the class is here. Suppose I say it later in the day at a meeting; then I mean that everyone who is meant to be at the meeting is here. What shall we say about this? Here are three different kinds of solution:

Radical Semantic View On every occasion of use, the sentence literally means that everyone in the world is here, and so is strictly speaking false. An interpreter recognizes that the speaker has said something false, and uses general principles to figure out what the speaker actually meant.

Radical Pragmatic View What the semantics provides is in some sense incomplete. What the sentence means is determined in part by the context of utterance and the speaker's intentions. The differences in meaning are entirely due to extra-linguistic facts which have no linguistic reflex.

The Intermediate View The logical form of sentences with the quantifier *every* contains a slot for information which is contributed by the context. So extra-linguistic information is required to fix the meaning; but the contribution of this information is mediated by linguistic form.

More Context: Anaphora

- *John is a bachelor. His wife is very nice.* (Uh, what?, who?)
- *John likes his dog Spiff even though he bites him sometimes.* (who bites?)
- *John likes Spiff. Peter does too.* (what to does Peter do?)
- *John loves his wife. Peter does too.* (whom does Peter love?)
- *John loves golf, and Mary too.* (who does what?)

SOME RIGHTS RESERVED ©: Michael Kohlhase 21 JACOBS UNIVERSITY

Context is Personal and keeps changing

- *The king of America is rich.* (true or false?)
- *The king of America isn't rich.* (false or true?)
- *If America had a king, the king of America would be rich.* (true or false!)
- *The king of Buganda is rich.* (Where is Buganda?)
- *. . . Joe Smith. . . The CEO of Westinghouse announced budget cuts.*(CEO=J.S.!)

SOME RIGHTS RESERVED ©: Michael Kohlhase 22 JACOBS UNIVERSITY

1.2.4 Preview of the Course

Plot of this Course

- Today: Motivation and find out what you already know

▷ What is Natural Language Processing/Understanding/Semantics?
▷ quick walk through the topics (landscape of semantics/pragmatics)

▷ What is logic/Semantics

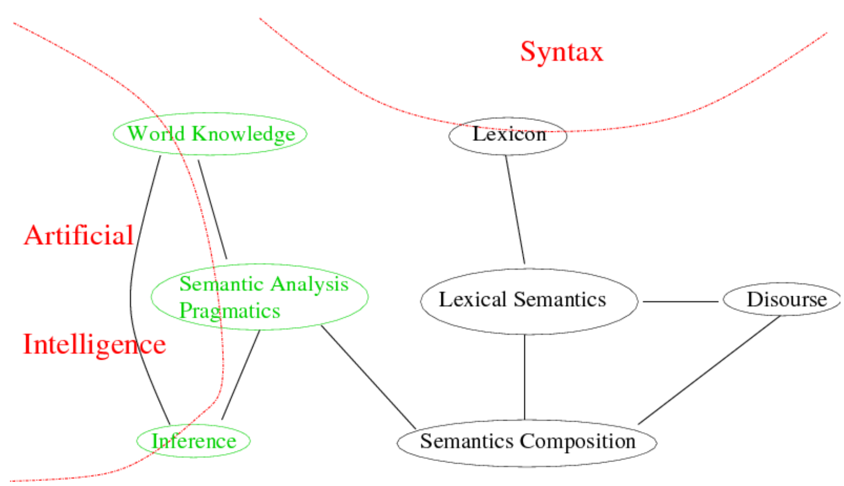
▷ improving models for NL Semantics/Pragmatics (repeat until over)

▷ Syntax (introduce a fragment)
▷ Semantics (construct the logical form)
▷ Pragmatics (construct/refine the interpretation)
▷ Evaluation of the model

▷ Recap, What have we learned (have a happy summer)

SOME RIGHTS RESERVED

A Landscape of Formal Semantics



SOME RIGHTS RESERVED

2 The Method of Fragments: Fragment 1

2.1 Logic as a Tool for Modeling NL Semantics

In this section we will briefly introduce formal logic and motivate how we will use it as a tool for developing precise theories about natural language semantics.[9] EdN:9

2.1.1 What is Logic?

What is Logic?

- formal languages, inference and their relation with the world
  - Formal language $\mathcal{FL}$: set of formulae ($2+3/7$, $\forall x.x+y=y+x$)
  - Formula: sequence/tree of symbols ($x, y, f, g, p, 1, \pi, \in, \neg, \wedge \forall, \exists$)
  - Models: things we understand (e.g. number theory)
  - Interpretation: maps formulae into models ($[\![\text{three plus five}]\!] = 8$)
  - Validity: $\mathcal{M} \models \mathbf{A}$, iff $[\![\mathbf{A}]\!]^{\mathcal{M}} = \mathsf{T}$ (five greater three is valid)
  - Entailment: $\mathbf{A} \models \mathbf{B}$, iff $\mathcal{M} \models \mathbf{B}$ for all $\mathcal{M} \models \mathbf{A}$. (generalize to $\mathcal{H} \models \mathbf{A}$)
  - Inference: rules to transform (sets of) formulae ($\mathbf{A}, \mathbf{A} \Rightarrow \mathbf{B} \vdash \mathbf{B}$)
- Syntax: formulae, inference (just a bunch of symbols)
- Semantics: models, interpr., validity, entailment (math. structures)
- Important Question: relation between syntax and semantics?

SOME RIGHTS RESERVED ©: Michael Kohlhase 25 JACOBS UNIVERSITY

So logic is the study of formal representations of objects in the real world, and the formal statements that are true about them. The insistence on a *formal language* for representation is actually something that simplifies life for us. Formal languages are something that is actually easier to understand than e.g. natural languages. For instance it is usually decidable, whether a string is a member of a formal language. For natural language this is much more difficult: there is still no program that can reliably say whether a sentence is a grammatical sentence of the English language.

We have already discussed the meaning mappings (under the monicker "semantics"). Meaning mappings can be used in two ways, they can be used to understand a formal language, when we use a mapping into "something we already understand", or they are the mapping that legitimize a representation in a formal language. We understand a formula (a member of a formal language) $\mathbf{A}$ to be a representation of an object $\mathcal{O}$, iff $[\![\mathbf{A}]\!] = \mathcal{O}$.

However, the game of representation only becomes really interesting, if we can do something with the representations. For this, we give ourselves a set of syntactic rules of how to manipulate the formulae to reach new representations or facts about the world.

Consider, for instance, the case of calculating with numbers, a task that has changed from a difficult job for highly paid specialists in Roman times to a task that is now feasible for young children. What is the cause of this dramatic change? Of course the formalized reasoning procedures for arithmetic that we use nowadays. These *calculi* consist of a set of rules that can be followed purely syntactically, but nevertheless manipulate arithmetic expressions in a correct and fruitful way. An essential prerequisite for syntactic manipulation is that the objects are given in a formal

[9] EdNote: also talk about Cresswell's most certain principle of semantics

language suitable for the problem. For example, the introduction of the decimal system has been instrumental to the simplification of arithmetic mentioned above. When the arithmetical calculi were sufficiently well-understood and in principle a mechanical procedure, and when the art of clock-making was mature enough to design and build mechanical devices of an appropriate kind, the invention of calculating machines for arithmetic by Wilhelm Schickard (1623), Blaise Pascal (1642), and Gottfried Wilhelm Leibniz (1671) was only a natural consequence.

We will see that it is not only possible to calculate with numbers, but also with representations of statements about the world (propositions). For this, we will use an extremely simple example; a fragment of propositional logic (we restrict ourselves to only one logical connective) and a small calculus that gives us a set of rules how to manipulate formulae.

In computational semantics, the picture is slightly more complicated than in Physics. Where Physics considers mathematical models, we build logical models, which in turn employ the term "model". To sort this out, let us briefly recap the components of logics, we have seen so far.[10] EdN:10

Logics make good (scientific[2]) models for natural language, since they are mathematically precise and relatively simple.

Formal languages simplify natural languages, in that problems of grammaticality no longer arise. Well-formedness can in general be decided by a simple recursive procedure.

Semantic models simplify the real world by concentrating on (but not restricting itself to) mathematically well-understood structures like sets or numbers. The induced semantic notions of validity and logical consequence are precisely defined in terms of semantic models and allow us to make predictions about truth conditions of natural language.

The only missing part is that we can conveniently compute the predictions made by the model. The underlying problem is that the semantic notions like validity and semantic consequence are defined with respect to *all* models, which are difficult to handle.

Therefore, logics typically have a third part, an inference system, or a calculus, which is a syntactic counterpart to the semantic notions. Formally, a calculus is just a set of rules (called inference rules) that transform (sets of) formulae (the assumptions) into other (sets of) formulae (the conclusions). A sequence of rule applications that transform the empty set of assumptions into a formula $\mathbf{T}$, is called a proof of $\mathbf{A}$. To make these assumptions clear, let us look at a very simple example.

2.1.2 Formal Systems

To prepare the ground for the particular developments coming up, let us spend some time on recapitulating the basic concerns of formal systems.

Logical Systems The notion of a logical system is at the basis of the field of logic. In its most abstract form, a logical system consists of a formal language, a class of models, and a satisfaction relation between models and expressions of the formal lanugage. The satisfaction relation tells us when an expression is deemed true in this model.

Logical Systems

▷ **Definition 2.1** A logical system is a triple $\mathcal{S} := \langle \mathcal{L}, \mathcal{K}, \models \rangle$, where $\mathcal{L}$ is a formal language, $\mathcal{K}$ is a set and $\models \subseteq \mathcal{K} \times \mathcal{L}$. Members of $\mathcal{L}$ are called formula e of $\mathcal{S}$, members of $\mathcal{K}$ model s for $\mathcal{S}$, and $\models$ the satisfaction relation.

[10] EdNote: adapt notation

[2] Since we use the word "model" in two ways, we will sometimes explicitly label it by the attribute "scientific" to signify that a whole logic is used to model a natural language phenomenon and with the attribute "semantic" for the mathematical structures that are used to give meaning to formal languages

▷ **Definition 2.2** Let $\mathcal{S} := \langle \mathcal{L}, \mathcal{K}, \models \rangle$ be a logical system, $\mathcal{M} \in \mathcal{K}$ be a model and $\mathbf{A} \in \mathcal{L}$ a formula, then we call $\mathbf{A}$

- ▷ satisfied by $\mathcal{M}$, iff $\mathcal{M} \models \mathbf{A}$
- ▷ falsified by $\mathcal{M}$, iff $\mathcal{M} \not\models \mathbf{A}$
- ▷ satisfiable in $\mathcal{K}$, iff $\mathcal{M} \models \mathbf{A}$ for some model $\mathcal{M} \in \mathcal{K}$.
- ▷ valid in $\mathcal{K}$ (write $\models \mathcal{M}$), iff $\mathcal{M} \models \mathbf{A}$ for all models $\mathcal{M} \in \mathcal{K}$
- ▷ falsifiable in $\mathcal{K}$, iff $\mathcal{M} \not\models \mathbf{A}$ for some $\mathcal{M} \in \mathcal{K}$.
- ▷ unsatisfiable in $\mathcal{K}$, iff $\mathcal{M} \not\models \mathbf{A}$ for all $\mathcal{M} \in \mathcal{K}$.

▷ **Definition 2.3** Let $\mathcal{S} := \langle \mathcal{L}, \mathcal{K}, \models \rangle$ be a logical system, then we define the entailment relation $\models \subseteq \mathcal{L} \times \mathcal{L}$. We say that $\mathbf{A}$ entails $\mathbf{B}$ (written $\mathbf{A} \models \mathbf{B}$), iff we have $\mathcal{M} \models \mathbf{B}$ for all models $\mathcal{M} \in \mathcal{K}$ with $\mathcal{M} \models \mathbf{A}$.

▷ **Observation 2.4** $\mathbf{A} \models \mathbf{B}$ *and* $\mathcal{M} \models \mathbf{A}$ *imply* $\mathcal{M} \models \mathbf{B}$.

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

Example 2.5 (First-Order Logic as a Logical System) Let $\mathcal{L} := \mathit{wff}_o(\Sigma)$, $\mathcal{K}$ be the class of first-order models, and $\mathcal{M} \models \mathbf{A} :\Leftrightarrow \mathcal{I}_\varphi(\mathbf{A}) = \mathsf{T}$, then $\langle \mathcal{L}, \mathcal{K}, \models \rangle$ is a logical system in the sense of Definition 2.1.

Note that central notions like the entailment relation (which is central for understanding reasoning processes) can be defined independently of the concrete compositional setup we have used for first-order logic, and only need the general assumptions about logical systems.

Let us now turn to the syntactical counterpart of the entailment relation: derivability in a calculus. Again, we take care to define the concepts at the general level of logical systems.

Calculi, Derivations, and Proofs The intuition of a calculus is that it provides a set of syntactic rules that allow to reason by considering the form of propositions alone. Such rules are called inference rules, and they can be strung together to derivations — which can alternatively be viewed either as sequences of formulae where all formulae are justified by prior formulae or as trees of inference rule applications. But we can also define a calculus in the more general setting of logical systems as an arbitrary relation on formulae with some general properties. That allows us to abstract away from the homomorphic setup of logics and calculi and concentrate on the basics.

Derivation Systems and Inference Rules

▷ **Definition 2.6** Let $\mathcal{S} := \langle \mathcal{L}, \mathcal{K}, \models \rangle$ be a logical system, then we call a relation $\vdash \subseteq \mathcal{P}(\mathcal{L}) \times \mathcal{L}$ a derivation relation for $\mathcal{S}$, if it

- ▷ is proof-reflexive, i.e. $\mathcal{H} \vdash \mathbf{A}$, if $\mathbf{A} \in \mathcal{H}$;
- ▷ is proof-transitive, i.e. if $\mathcal{H} \vdash \mathbf{A}$ and $\mathcal{H}' \cup \{\mathbf{A}\} \vdash \mathbf{B}$, then $\mathcal{H} \cup \mathcal{H}' \vdash \mathbf{B}$;
- ▷ admits weakening, i.e. $\mathcal{H} \vdash \mathbf{A}$ and $\mathcal{H} \subseteq \mathcal{H}'$ imply $\mathcal{H}' \vdash \mathbf{A}$.

▷ **Definition 2.7** We call $\langle \mathcal{L}, \mathcal{K}, \models, \vdash \rangle$ a formal system, iff $\mathcal{S} := \langle \mathcal{L}, \mathcal{K}, \models \rangle$ is a logical system, and $\vdash$ a derivation relation for $\mathcal{S}$.

▷ **Definition 2.8** Let $\mathcal{L}$ be a formal language, then an inference rule over $\mathcal{L}$

$$\frac{\mathbf{A}_1 \quad \cdots \quad \mathbf{A}_n}{\mathbf{C}}\mathcal{N}$$

where $\mathbf{A}_1, \ldots, \mathbf{A}_n$ and $\mathbf{C}$ are formula schemata for $\mathcal{L}$ and $\mathcal{N}$ is a name.
The $\mathbf{A}_i$ are called assumption s, and $\mathbf{C}$ is called conclusion.

▷ **Definition 2.9** An inference rule without assumptions is called an axiom (schema).

▷ **Definition 2.10** Let $\mathcal{S} := \langle \mathcal{L}, \mathcal{K}, \models \rangle$ be a logical system, then we call a set $\mathcal{C}$ of inference rules over $\mathcal{L}$ a calculus for $\mathcal{S}$.

SOME RIGHTS RESERVED ©: Michael Kohlhase 27

With formula schemata we mean representations of sets of formulae, we use boldface uppercase letters as (meta)-variables for formulae, for instance the formula schema $\mathbf{A} \Rightarrow \mathbf{B}$ represents the set of formulae whose head is $\Rightarrow$.

Derivations and Proofs

▷ **Definition 2.11** Let $\mathcal{S} := \langle \mathcal{L}, \mathcal{K}, \models \rangle$ be a logical system and $\mathcal{C}$ a calculus for $\mathcal{S}$, then a $\mathcal{C}$-derivation of a formula $\mathbf{C} \in \mathcal{L}$ from a set $\mathcal{H} \subseteq \mathcal{L}$ of hypotheses (write $\mathcal{H} \vdash_{\mathcal{C}} \mathbf{C}$) is a sequence $\mathbf{A}_1, \ldots, \mathbf{A}_m$ of $\mathcal{L}$-formulae, such that

- ▷ $\mathbf{A}_m = \mathbf{C}$, (derivation culminates in $\mathbf{C}$)
- ▷ for all $1 \leq i \leq m$, either $\mathbf{A}_i \in \mathcal{H}$, or (hypothesis)
- ▷ there is an inference rule $\frac{\mathbf{A}_{l_1} \quad \cdots \quad \mathbf{A}_{l_k}}{\mathbf{A}_i}$ in $\mathcal{C}$ with $l_j < i$ for all $j \leq k$.(rule application)

Observation: We can also see a derivation as a tree, where the $\mathbf{A}_{l_j}$ are the children of the node $\mathbf{A}_k$.

▷ **Example 2.12** In the propositional Hilbert calculus $\mathcal{H}^0$ we have the derivation $P \vdash_{\mathcal{H}^0} Q \Rightarrow P$: the sequence is $P \Rightarrow Q \Rightarrow P, P, Q \Rightarrow P$ and the corresponding tree on the right.

$$\frac{\dfrac{}{P \Rightarrow Q \Rightarrow P}K \qquad P}{Q \Rightarrow P}MP$$

▷ **Observation 2.13** *Let $\mathcal{S} := \langle \mathcal{L}, \mathcal{K}, \models \rangle$ be a logical system and $\mathcal{C}$ a calculus for $\mathcal{S}$, then the $\mathcal{C}$-derivation relation $\vdash_{\mathcal{D}}$ defined in Definition 2.11 is a derivation relation in the sense of Definition 2.6.*[11]

▷ **Definition 2.14** We call $\langle \mathcal{L}, \mathcal{K}, \models, \mathcal{C} \rangle$ a formal system, iff $\mathcal{S} := \langle \mathcal{L}, \mathcal{K}, \models \rangle$ is a logical system, and $\mathcal{C}$ a calculus for $\mathcal{S}$.

▷ **Definition 2.15** A derivation $\emptyset \vdash_{\mathcal{C}} \mathbf{A}$ is called a proof of $\mathbf{A}$ and if one exists (write $\vdash_{\mathcal{C}} \mathbf{A}$) then $\mathbf{A}$ is called a $\mathcal{C}$-theorem.

▷ **Definition 2.16** an inference rule $\mathcal{I}$ is called admissible in $\mathcal{C}$, if the extension of $\mathcal{C}$ by $\mathcal{I}$ does not yield new theorems.

SOME RIGHTS RESERVED ©: Michael Kohlhase 28

[k]EdNote: MK: this should become a view!

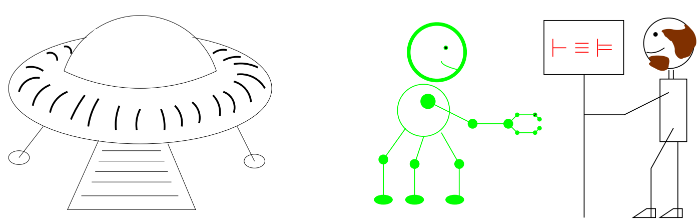
Inference rules are relations on formulae represented by formula schemata (where boldface, uppercase letters are used as meta-variables for formulae). For instance, in Example 2.12 the inference rule $\frac{\mathbf{A} \Rightarrow \mathbf{B} \quad \mathbf{A}}{\mathbf{B}}$ was applied in a situation, where the meta-variables $\mathbf{A}$ and $\mathbf{B}$ were instantiated by the formulae P and $Q \Rightarrow P$.

As axioms do not have assumptions, they can be added to a derivation at any time. This is just what we did with the axioms in Example 2.12.

Properties of Calculi In general formulae can be used to represent facts about the world as propositions; they have a semantics that is a mapping of formulae into the real world (propositions are mapped to truth values.) We have seen two relations on formulae: the entailment relation and the deduction relation. The first one is defined purely in terms of the semantics, the second one is given by a calculus, i.e. purely syntactically. Is there any relation between these relations?

Soundness and Completeness

- ▷ **Definition 2.17** Let $\mathcal{S} := \langle \mathcal{L}, \mathcal{K}, \models \rangle$ be a logical system, then we call a calculus $\mathcal{C}$ for $\mathcal{S}$
  - ▷ sound (or correct), iff $\mathcal{H} \models \mathbf{A}$, whenever $\mathcal{H} \vdash_{\mathcal{C}} \mathbf{A}$, and
  - ▷ complete, iff $\mathcal{H} \vdash_{\mathcal{C}} \mathbf{A}$, whenever $\mathcal{H} \models \mathbf{A}$.
- ▷ Goal: $\vdash \mathbf{A}$ iff $\models \mathbf{A}$ (provability and validity coincide)
  - ▷ To TRUTH through PROOF (CALCULEMUS [Leibniz ~1680])



SOME RIGHTS RESERVED ©: Michael Kohlhase 29 JACOBS UNIVERSITY

Ideally, both relations would be the same, then the calculus would allow us to infer all facts that can be represented in the given formal language and that are true in the real world, and only those. In other words, our representation and inference is faithful to the world.

A consequence of this is that we can rely on purely syntactical means to make predictions about the world. Computers rely on formal representations of the world; if we want to solve a problem on our computer, we first represent it in the computer (as data structures, which can be seen as a formal language) and do syntactic manipulations on these structures (a form of calculus). Now, if the provability relation induced by the calculus and the validity relation coincide (this will be quite difficult to establish in general), then the solutions of the program will be correct, and we will find all possible ones.

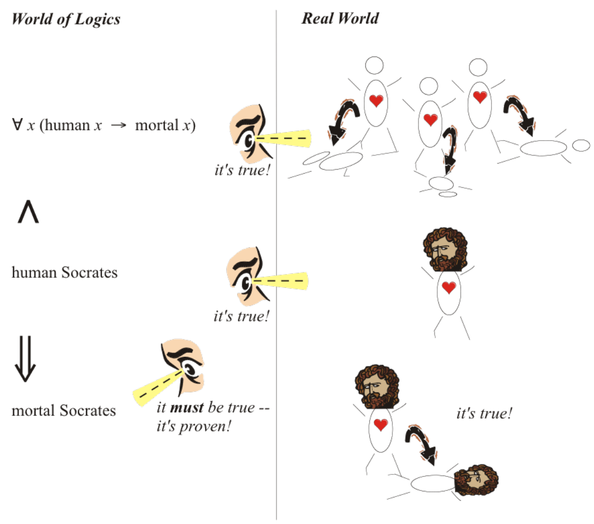
Of course, the logics we have studied so far are very simple, and not able to express interesting facts about the world, but we will study them as a simple example of the fundamental problem of Computer Science: How do the formal representations correlate with the real world.

Within the world of logics, one can derive new propositions (the *conclusions*, here: *Socrates is mortal*) from given ones (the *premises*, here: *Every human is mortal* and *Sokrates is human*). Such derivations are *proofs*.

In particular, logics can describe the internal structure of real-life facts; e.g. individual things, actions, properties. A famous example, which is in fact as old as it appears, is illustrated in the slide below.

The miracle of logics

- ▷ Purely formal derivations are true in the real world!



©: Michael Kohlhase 30

If a logic is correct, the conclusions one can prove are true (= hold in the real world) whenever the premises are true. This is a miraculous fact (think about it!)

2.1.3 Using Logic to Model Meaning of Natural Language

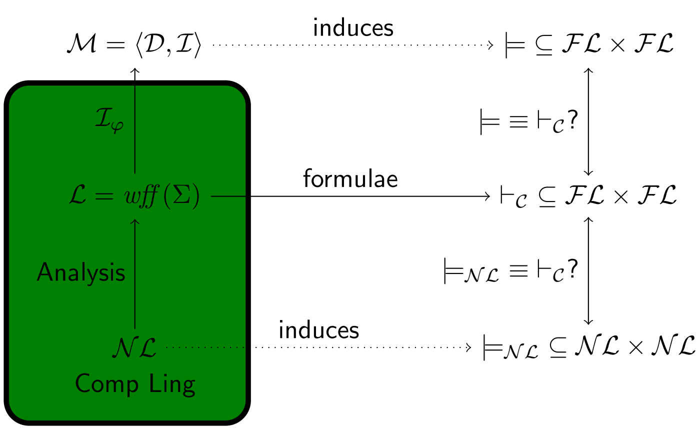
Modeling Natural Language Semantics

- ▷ Problem: Find formal (logic) system for the meaning of natural language
- ▷ History of ideas
  - ▷ Propositional logic [ancient Greeks like Aristotle]
    **Every human is mortal*
  - ▷ First-Order Predicate logic [Frege $\leq$ 1900]
    **I believe, that my audience already knows this.*
  - ▷ Modal logic [Lewis18, Kripke65]
    **A man sleeps. He snores.* $((\exists X.\mathsf{man}(X) \wedge \mathsf{sleep}(X))) \wedge \mathsf{snore}(X)$
  - ▷ Various dynamic approaches (e.g. DRT, DPL)
    **Most men wear black*
  - ▷ Higher-order Logic, e.g. generalized quantifiers
  - ▷ ...

©: Michael Kohlhase 31

Let us now reconcider the role of all of this for natural language semantics. We have claimed that the goal of the course is to provide you with a set of methods to determine the meaning of natural language. If we look back, all we did was to establish translations from natural languages into formal languages like first-order or higher-order logic (and that is all you will find in most semantics papers and textbooks). Now, we have just tried to convince you that these are actually syntactic entities. So, *where is the semantics?*.

Natural Language Semantics?



©: Michael Kohlhase 32

As we mentioned, the green area is the one generally covered by natural language semantics. In the analysis process, the natural language utterances (viewed here as formulae of a language $\mathcal{NL}$) are translated to a formal language $\mathcal{FL}$ (a set $wff(\Sigma)$ of well-formed formulae). We claim that this is all that is needed to recapture the semantics even it this is not immediately obvious at first: Theoretical Logic gives us the missing pieces.

Since $\mathcal{FL}$ is a formal language of a logical systems, it comes with a notion of model and an interpretation function $\mathcal{I}_\varphi$ that translates $\mathcal{FL}$ formulae into objects of that model. This induces a notion of logical consequence[3] as explained in[12]. It also comes with a calculus $\mathcal{C}$ acting on EdN:12 $\mathcal{FL}$-formulae, which (if we are lucky) is correct and complete (then the mappings in the upper rectangle commute).

What we are really interested in in natural language semantics is the truth conditions and natural consequence relations on natural language utterances, which we have denoted by $\models_{\mathcal{NL}}$. If the calculus $\mathcal{C}$ of the logical system $\langle \mathcal{FL}, \mathcal{K}, \models \rangle$ is adequate (it might be a bit presumptious to say sound and complete), then it is a model of the relation $\models_{\mathcal{NL}}$. Given that both rectangles in the diagram commute, then we really have a model for truth-conditions and logical consequence for natural language utterances, if we only specify the analysis mapping (the green part) and the calculus.

Logic-Based Knowledge Representation for NLP

- ▷ Logic (and related formalisms) allow to integrate world knowledge
  - ▷ explicitly (gives more understanding than statistical methods)
  - ▷ transparently (symbolic methods are monotonic)

[3]Relations on a set S are subsets of the cartesian product of S, so we use $R \in (S^*)S$ to signify that R is a (n-ary) relation on X.

[12]EdNote: crossref

▷ systematically (we can prove theorems about our systems)

▷ Signal + World knowledge makes more powerful model

▷ Does not preclude the use of statistical methods to guide inference

▷ Problems with logic-based approaches

▷ Where does the world knowledge come from? (Ontology problem)

▷ How to guide search induced by log. calculi (combinatorial explosion)

One possible answer: Description Logics. (next couple of times)

SOME RIGHTS RESERVED ©: Michael Kohlhase 33 JACOBS UNIVERSITY

2.2 The Method of Fragments

We will proceed by the "method of fragments", introduced by Richard Montague in [Mon70], where he insists on specifying a complete syntax and semantics for a specified subset ("fragment") of a language, rather than writing rules for the a single construction while making implicit assumptions about the rest of the grammar.

▷ *In the present paper I shall accordingly present a precise treatment, culminating in a theory of truth, of a formal language that I believe may be reasonably regarded as a fragment of ordinary English.* source=R. Montague 1970 [Mon70], p.188

The first step in defining a fragment of natural language is to define which sentences we want to consider. We will do this by means of a context-free grammar. This will do two things: act as an oracle deciding which sentences (of natural language) are OK, and secondly to build up syntax trees, which we will later use for semantics construction.

Natural Language Fragments

▷ Idea: Formally identify a set (NL) sentences we want to study by a context-free grammar.

▷ Idea: Use non-terminals to classify NL phrases

▷ **Definition 2.18** We call a non-terminal of a context-free grammar a syntactical category. We distinguish two kinds of rules

structural rule s $\mathcal{L}$: $H \to c_1, \ldots, c_n$ with head H, label $\mathcal{L}$, and a sequence of phrase categories c_i.

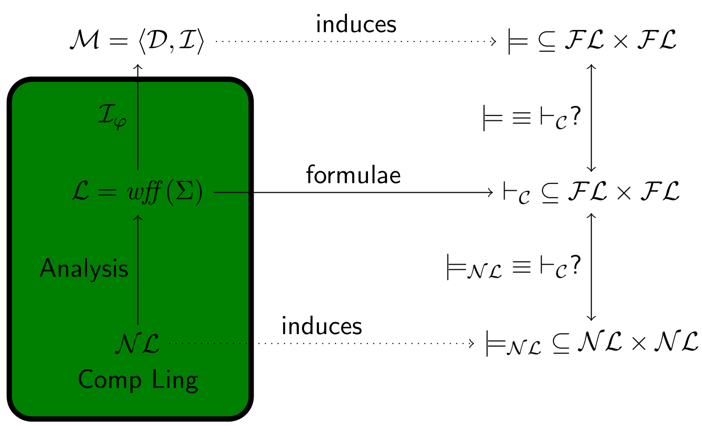
lexical rule s $\mathcal{L}$: $H \to t_1|\ldots|t_n$, where the t_i are terminals (i.e. NL phrases)

SOME RIGHTS RESERVED ©: Michael Kohlhase 34 JACOBS UNIVERSITY

We distinguish two grammar fragments: the structural grammar rules and the lexical rules, because they are guided by differing intuitions. The former set of rules govern how NL phrases can be composed to sentences (and later even to discourses). The latter rules are a simple representation of a lexicon, i.e. a structure which tells us about words (the terminal objects of language): their syntactical categories, their meaning, etc.

Formal Natural Language Semantics with Fragments

▷ Idea: We will follow the picture we have discussed before



Choose a target logic $\mathcal{L}$ and specify a translation from syntax trees to formulae!

©: Michael Kohlhase 35

Semantics by Translation

▷ Idea: We translate sentences by translating their syntax trees via tree node translation rules.

▷ **Definition 2.19** We represent a node α in a syntax tree with children $\beta_1, \ldots, \beta_n$ by $[X_{1\beta_1}, \ldots, X_{n\beta_n}]_\alpha$ and write a translation rule as

$$\mathcal{L}\colon [X_{1\beta_1}, \ldots, X_{n\beta_n}]_\alpha \rightsquigarrow \Phi({X_1}', \ldots, {X_n}')$$

if the translation of the node α can be computed from those of the β_i via a semantical function Φ.

▷ **Definition 2.20** For a natural language utterance A, we will use $\langle \mathrm{A} \rangle$ for the result of translating A.

▷ **Definition 2.21 (Default Rule)** For every word w in the fragment we assue a constant w' in the logic $\mathcal{L}$ and the "pseudo-rule" $t1\colon w \rightsquigarrow w'$. (if no other translation rule applies)

©: Michael Kohlhase 36

[13] EdN:13

2.3 The First Fragment: Setting up the Basics

The first fragment will primarily be used for setting the stage, and introducing the method itself. The coverage of the fragment is too small to do anything useful with it, but it will allow us to

[13] EdNote: Move discussion on compositionality here

discuss the salient features of the method, the particular setup of the grammars and semantics before graduating to more useful fragments.

2.3.1 Natural Language Syntax

Structural Grammar Rules

▷ **Definition 2.22** Fragment 1 knowls the following eight syntactical categories

S	sentence	NP	noun phrase
N	noun	N_{pr}	proper name
V^i	intransitive verb	V^t	transitive verb
conj	connective	Adj	adjective

▷ **Definition 2.23** We have the following grammar rule s in fragment 1.

S1.	S	$\to$	NP V^i
S2.	S	$\to$	NP V^t NP
N1.	NP	$\to$	N_{pr}
N2.	NP	$\to$	the$N$
S3.	S	$\to$	Itisnotthecasethat S
S4.	S	$\to$	S conj S
S5.	S	$\to$	NP is NP
S6.	S	$\to$	NP is Adj.

SOME RIGHTS RESERVED ©: Michael Kohlhase 37 JACOBS UNIVERSITY

Lexical insertion rules for Fragment 1

▷ **Definition 2.24** We have the following lexical insertion rule s in Fragment 1.

L1.	N_{pr}	$\to$	{Prudence, Ethel, Chester, Jo, Bertie, Fiona}
L2.	N	$\to$	{book, cake, cat, golfer, dog, lecturer, student, singer}
L3.	V^i	$\to$	{ran, laughed, sang, howled, screamed}
L4.	V^t	$\to$	{read, poisoned, ate, liked, loathed, kicked}
L5.	conj	$\to$	{and, or}
L6.	Adj	$\to$	{happy, crazy, messy, disgusting, wealthy}

▷ Note: We will adopt the convention that new lexical insertion rules can be generated spontaneously as needed.

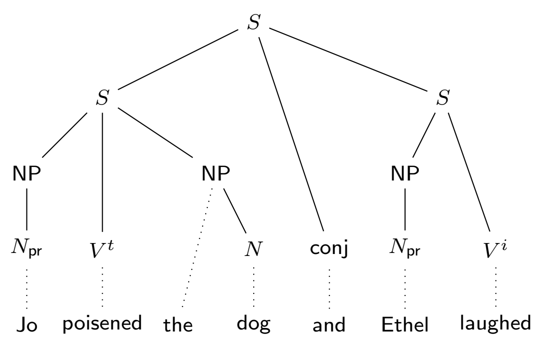
SOME RIGHTS RESERVED ©: Michael Kohlhase 38 JACOBS UNIVERSITY

These rules represent a simple lexicon, they specify which words are accepted by the grammar and what their syntactical categories are.

Syntax Example: *Jo poisoned the dog and Ethel laughed*

▷ **Observation 2.25** *Jo poisoned the dog and Ethel laughed is a sentence of fragment 1*

▷ We can construct a syntax tree for it!



©: Michael Kohlhase 39

The next step will be to introduce the logical model we will use for Fragment 1: Predicate Logic without Quantifiers. Syntactically, this logic is a fragment of first-order logic, but it's expressivity is equivalent to Propositional Logic. Therefore, we will introduce the syntax of full first-order logic (with quantifiers since we will need if for Fragment 4 later), but for the semantics stick with a setup without quantifiers. We will go into the semantic difficulties that they pose later (in Subsection 4.0 and Section 4).

2.3.2 Predicate Logic Without Quantifiers

First-Order Predicate Logic (PL^1)

▷ Coverage: We can talk about (*All humans are mortal*)

- ▹ individual things and denote them by variables or constants
- ▹ properties of individuals, (e.g. being human or mortal)
- ▹ relations of individuals, (e.g. $sibling\_of$ relationship)
- ▹ functions on individuals, (e.g. the $father\_of$ function)

We can also state the existence of an individual with a certain property, or the universality of a property.

▷ But we cannot state assertions like

- ▹ *There is a surjective function from the natural numbers into the reals.*

▷ First-Order Predicate Logic has many good properties (complete calculi, compactness, unitary, linear unification,. . .)

▷ But too weak for formalizing: (at least directly)

- ▹ natural numbers, torsion groups, calculus, . . .
- ▹ generalized quantifiers (*most, at least three, some,. . .*)

©: Michael Kohlhase 40

The formulae of first-order logic is built up from the signature and variables as terms (to represent individuals) and propositions (to represent propositions). The latter include the propositional connectives, but also quantifiers.

PL^1 Syntax (Formulae)

- **Definition 2.26** term s: $\mathbf{A} \in wff_\iota(\Sigma_\iota)$ (denote individuals: type ι)
  - $\mathcal{V}_\iota \subseteq wff_\iota(\Sigma_\iota)$,
  - if $f \in \Sigma_k^f$ and $\mathbf{A}^i \in wff_\iota(\Sigma_\iota)$ for $i \leq k$, then $f(\mathbf{A}^1, \ldots, \mathbf{A}^k) \in wff_\iota(\Sigma_\iota)$.
- **Definition 2.27** proposition s: $\mathbf{A} \in wff_o(\Sigma)$ (denote truth values: type o)
  - if $p \in \Sigma_k^p$ and $\mathbf{A}^i \in wff_\iota(\Sigma_\iota)$ for $i \leq k$, then $p(\mathbf{A}^1, \ldots, \mathbf{A}^k) \in wff_o(\Sigma)$,
  - if $\mathbf{A}, \mathbf{B} \in wff_o(\Sigma)$, then $T, \mathbf{A} \wedge \mathbf{B}, \neg\mathbf{A}, \forall X.\mathbf{A} \in wff_o(\Sigma)$.
- **Definition 2.28** We define the connectives $F, \vee, \Rightarrow, \Leftrightarrow$ via the abbreviations $\mathbf{A} \vee \mathbf{B} := \neg(\neg\mathbf{A} \wedge \neg\mathbf{B})$, $\mathbf{A} \Rightarrow \mathbf{B} := \neg\mathbf{A} \vee \mathbf{B}$, $(\mathbf{A} \Leftrightarrow \mathbf{B}) := (\mathbf{A} \Rightarrow \mathbf{B}) \wedge (\mathbf{B} \Rightarrow \mathbf{A})$, and $F := \neg T$. We will use them like the primary connectives $\wedge$ and $\neg$
- **Definition 2.29** We use $\exists X.\mathbf{A}$ as an abbreviation for $\neg(\forall X.\neg\mathbf{A})$.(existential quantifier)
- **Definition 2.30** Call formulae without connectives or quantifiers atomic else complex.

SOME RIGHTS RESERVED ©: Michael Kohlhase 41 JACOBS UNIVERSITY

Note: that we only need e.g. conjunction, negation, and universal quantification, all other logical constants can be defined from them (as we will see when we have fixed their interpretations).

Semantic Models for PL_{NQ}: What the semantics of PL_{NQ} will do is allow us to determine, for any given sentence of the language, whether it is true or false. Now, in general, to know whether a sentence in a language is true or false, we need to know what the world is like. The same is true for PL_{NQ}. But to make life easier, we don't worry about the real world; we define a situation, a little piece of the world, and evaluate our sentences relative to this situation. We do this using a structure called a *model*.

What we need to know about the world is:

- What objects there are in the world.
- Which predicates are true of which objects, and which objects stand in which relations to each other.

Definition 2.31 A model for PL_{NQ} is an ordered pair $\langle \mathcal{D}, \mathcal{I} \rangle$ where:

- $\mathcal{D}$ is the domain, which specifies what objects there are in the model. (All kinds of things can be objects.)
- $\mathcal{I}$ is an interpretation function. (Can uses the terms "denotation assignment function" and "naming function.")

An interpretation function for a language is a function whose arguments are the non-logical constants of the language, and which give back as value a *denotation* or *reference* for the constant. Specifically:

- To an individual constant, the interpretation function assigns an object from the model. I.e. the interpretation function tells us which objects from the model are named by each of the constants. (Note that the interpretation function can assign the same object to more than one constant; but to each constant, it can assign at most one object as value.)

- To a one-place predicate, the interpretation function assigns a set of objects from the model. Intuitively, these objects are the objects in the model of which the predicate is true.
- To a two-place predicate, the interpretation function assigns a set of *pairs* of objects from the model. Intuitively, these pairs are the pairs of which the predicate is true. (Generalizing: To an n-place predicate, the interpretation function assigns a set of n-tuples of objects from the model.)

Example 2.32 Let $L := \{a, b, c, d, e, P, Q, R, S\}$, we set the domain $\mathcal{D} := \{\mathrm{TRIANGLE}, \mathrm{SQUARE}, \mathrm{CIRCLE}, \mathrm{DIAMOND}\}$ and the interpretation function $\mathcal{I}$ by setting

- $a \mapsto \mathrm{TRIANGLE}$, $b \mapsto \mathrm{SQUARE}$, $c \mapsto \mathrm{CIRCLE}$, $d \mapsto \mathrm{DIAMOND}$, and $e \mapsto \mathrm{DIAMOND}$ for individual constants,
- $P \mapsto \{\mathrm{TRIANGLE}, \mathrm{SQUARE}\}$ and $Q \mapsto \{\mathrm{SQUARE}, \mathrm{DIAMOND}\}$, for unary predicate constants.
- $R \mapsto \{\langle \mathrm{CIRCLE}, \mathrm{DIAMOND}\rangle, \langle \mathrm{DIAMOND}, \mathrm{CIRCLE}\rangle\}$, and
- $S \mapsto \{\langle \mathrm{DIAMOND}, \mathrm{SQUARE}\rangle, \langle \mathrm{SQUARE}, \mathrm{TRIANGLE}\rangle\}$ for binary predicate constants.

The valuation function, $[[\cdot]]^M$, fixes the value (for our purposes, the truth value) of sentences of the language relative to a given model. The valuation function, as you'll notice, is not itself part of the model. The valuation function is the same for any model for a language based on $\mathrm{PL}_{\mathrm{NQ}}$.

Definition 2.33 Let $\langle \mathcal{D}, \mathcal{I}\rangle$ be a model for a language $L \subseteq \mathrm{PL}_{\mathrm{NQ}}$.

1. For any non-logical constant c of L, $\mathcal{I}_\varphi(c) = \mathcal{I}(c)$.
2. Atomic formulas: Let P be an n-place predicate constant, and $t_1, \ldots, t_n$ be individual constants. Then $\mathcal{I}_\varphi(P(t_1, \ldots, t_n)) = \mathsf{T}$ iff $\langle \mathcal{I}_\varphi(t_1), \ldots, \mathcal{I}_\varphi(t_n)\rangle \in \mathcal{I}(P)$.
3. Complex formulas: Let φ and ψ be sentences. Then:
   (a) $\mathcal{I}_\varphi(\neg(\mathbf{A})) = \mathsf{T}$ iff $\mathcal{I}_\varphi(\mathbf{A}) = \mathsf{F}$.
   (b) $\mathcal{I}_\varphi(\mathbf{A} \wedge \mathbf{B}) = \mathsf{T}$ iff $\mathcal{I}_\varphi(\mathbf{A}) = \mathsf{T}$ and $\mathcal{I}_\varphi(\mathbf{B}) = \mathsf{T}$.
   (c) $\mathcal{I}_\varphi(\mathbf{A} \vee \mathbf{B}) = \mathsf{T}$ iff $\mathcal{I}_\varphi(\mathbf{A}) = \mathsf{T}$ or $\mathcal{I}_\varphi(\mathbf{B}) = \mathsf{T}$.
   (d) $\mathcal{I}_\varphi(\mathbf{A} \Rightarrow \mathbf{B}) = \mathsf{T}$ iff $\mathcal{I}_\varphi(\mathbf{A}) = \mathsf{F}$ or $\mathcal{I}_\varphi(\mathbf{B}) = \mathsf{T}$.

$\mathrm{PL}_{\mathrm{NQ}}$: Predicate Logic without variables and functions

- Idea: Study the fragment of first-order Logic without Quantifiers and functions
- Universe s $\mathcal{D}_o = \{\mathsf{T}, \mathsf{F}\}$ of truth value s and $\mathcal{D}_\iota \neq \emptyset$ of individuals
- interpretation $\mathcal{I}$ assigns values to constants, e.g.
  - $\mathcal{I}(\neg)\colon \mathcal{D}_o \to \mathcal{D}_o; \mathsf{T} \mapsto \mathsf{F}; \mathsf{F} \mapsto \mathsf{T}$ and $\mathcal{I}(\wedge) = \ldots$ (as in PL^0)
  - $\mathcal{I}\colon \Sigma_0^f \to \mathcal{D}_\iota$ (interpret individual constants as individuals)
  - $\mathcal{I}\colon \Sigma_k^p \to \mathcal{P}({\mathcal{D}_\iota}^k)$ (interpret predicates as arbitrary relations)
- The value function $\mathcal{I}\colon \mathit{wff}_o(\Sigma) \to \mathcal{D}_o$ assigns values to formulae (recursively)
  - e.g. $\mathcal{I}(\neg\mathbf{A}) = \mathcal{I}(\neg)(\mathcal{I}(\mathbf{A}))$ (just as in PL^0)
  - $\mathcal{I}(p(\mathbf{A}^1, \ldots, \mathbf{A}^k)) := \mathsf{T}$, iff $\langle \mathcal{I}(\mathbf{A}^1), \ldots, \mathcal{I}(\mathbf{A}^k)\rangle \in \mathcal{I}(p)$
- Model: $\mathcal{M} = \langle \mathcal{D}_\iota, \mathcal{I}\rangle$ varies in $\mathcal{D}_\iota$ and $\mathcal{I}$.

▷ **Theorem 2.34** *PL_{NQ} is isomorphic to PL^0* (*interpret atoms as prop. variables*)

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

Now that we have the target logic we can complete the analysis arrow in figure[14]. We do this again, by giving transformation rules. EdN:14

2.3.3 Natural Language Semantics via Translation

Translation rules for non-basic expressions (NP and S)

▷ **Definition 2.35** We have the following translation rules for internal nodes of the syntax tree

T1.	$[X_{\mathsf{NP}}, Y_{V^i}]_S$	$\Longrightarrow$	$Y'(X')$
T2.	$[X_{\mathsf{NP}}, Y_{V^t}, Z_{\mathsf{NP}}]_S$	$\Longrightarrow$	$Y'(X', Z')$
T3.	$[X_{N_{\mathsf{pr}}}]_{\mathsf{NP}}$	$\Longrightarrow$	X'
T4.	$[\mathsf{the}, X_N]_{\mathsf{NP}}$	$\Longrightarrow$	the X'
T5.	$[\mathsf{Itisnotthecasethat} X_S]_S$	$\Longrightarrow$	$\neg(X')$
T6.	$[X_S, Y_{\mathsf{conj}}, Z_S]_S$	$\Longrightarrow$	$Y'(X', Z')$
T7.	$[X_{\mathsf{NP}}, \mathsf{is}, Y_{\mathsf{NP}}]_S$	$\Longrightarrow$	$X' = Y'$
T8.	$[X_{\mathsf{NP}}, \mathsf{is} Y_{\mathsf{Adj}}]_S$	$\Longrightarrow$	$Y'(X')$

Read e.g. $[Y, Z]_X$ as a node with label X in the syntax tree with daughters X and Y. Read X' as the translation of X via these rules.

▷ Note that we have exactly one tranlation per syntax rule.

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

Translation rule for basic lexical items

▷ **Definition 2.36** The target logic for $\mathcal{F}_1$ is $\mathrm{PL}_{\mathrm{NQ}}$, the fragment of PL^1 without quantifiers.

▷ Lexical Translation Rules for $\mathcal{F}_1$ Categories:

- ▷ If w is a proper name, then $w' \in \Sigma_0^f$. (individual constant)
- ▷ If w is an intransitive verb, then $w' \in \Sigma_1^p$. (one-place predicate)
- ▷ If w is a transitive verb, $w' \in \Sigma_2^p$. (two-place predicate)
- ▷ If w is a noun phrase, then $w' \in \Sigma_0^f$. (individual constant)

▷ Semantics by Translation: We translate sentences by translating their syntax trees via tree node translation rules.

▷ For any non-logical word w, we have the "pseudo-rule" $t1 \colon w \rightsquigarrow w'$.

▷ Note: This rule does not apply to the syncategorematic items is and the.

[14]EdNote: reference

▷ Translations for logical connectives

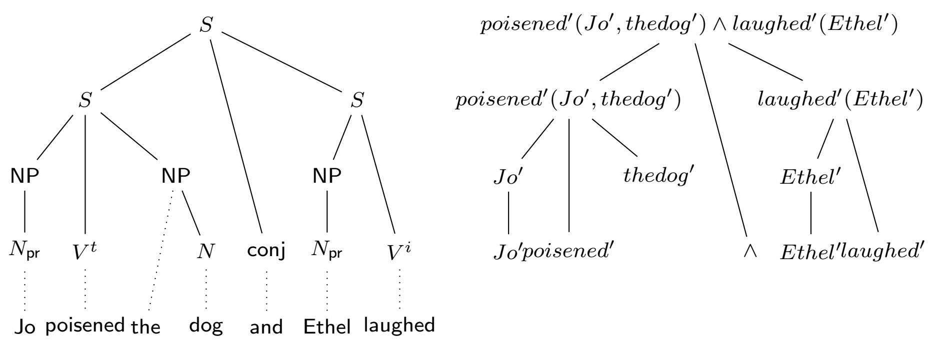
t2.	and	$\Longrightarrow$	$\wedge$
t3.	or	$\Longrightarrow$	$\vee$
t4.	itisnotthecasethat	$\Longrightarrow$	$\neg$

SOME RIGHTS RESERVED ©: Michael Kohlhase 44 JACOBS UNIVERSITY

Translation Example

▷ **Observation 2.37** *Jo poisened the dog and Ethel laughed is a sentence of fragment 1*

▷ We can construct a syntax tree for it!



SOME RIGHTS RESERVED ©: Michael Kohlhase 45 JACOBS UNIVERSITY

2.4 Calculi for Automated Theorem Proving: Analytical Tableaux

In this section we will introduce tableau calculi for propositional logics. To make the reasoning procedure more interesting, we will use first-order predicate logic without variables, function symbols and quantifiers as a basis. This logic (we will call it $\mathrm{PL_{NQ}}$) allows us express simple natural language sentences and to re-use our grammar for experimentation, without introducing the whole complications of first-order inference.

The logic $\mathrm{PL_{NQ}}$ is equivalent to propositional logic in expressivity: atomic formulae[15] take the role of propositional variables. EdN:15

Instead of deducing new formulae from axioms (and hypotheses) and hoping to arrive at the desired theorem, we try to deduce a contradiction from the negation of the theorem. Indeed, a formula **A** is valid, iff ¬**A** is unsatisfiable, so if we derive a contradiction from ¬**A**, then we have proven **A**. The advantage of such "test-calculi" (also called negative calculi) is easy to see. Instead of finding a proof that ends in **A**, we have to find any of a broad class of contradictions. This makes the calculi that we will discuss now easier to control and therefore more suited for mechanization.

[15] EdNote: introduced?, tie in with the stuff before

2.4.1 Analytical Tableaux

Before we can start, we will need to recap some nomenclature on formulae.

Recap: Atoms and Literals

- ▷ **Definition 2.38** We call a formula atomic, or an atom, iff it does not contain connectives. We call a formula complex, iff it is not atomic.
- ▷ **Definition 2.39** We call a pair $\mathbf{A}^\alpha$ a labeled formula, if $\alpha \in \{\mathsf{T}, \mathsf{F}\}$. A labeled atom is called literal.
- ▷ **Definition 2.40** Let Φ be a set of formulae, then we use $\Phi^\alpha := \{\mathbf{A}^\alpha \mid \mathbf{A} \in \Phi\}$.

SOME RIGHTS RESERVED ©: Michael Kohlhase 46 JACOBS UNIVERSITY

The idea about literals is that they are atoms (the simplest formulae) that carry around their intended truth value.

Now we will also review some propositional identities that will be useful later on. Some of them we have already seen, and some are new. All of them can be proven by simple truth table arguments.

Test Calculi: Tableaux and Model Generation

- ▷ Idea: instead of showing $\emptyset \vdash Th$, show $\neg Th \vdash trouble$ (use $\bot$ for trouble)
- ▷ **Example 2.41** Tableau Calculi try to construct models.

Tableau Refutation (Validity)	Model generation (Satisfiability)
$\models P \wedge Q \Rightarrow Q \wedge P$	$\models P \wedge (Q \vee \neg R) \wedge \neg Q$
$P \wedge Q \Rightarrow Q \wedge P^{\mathsf{f}}$ $P \wedge Q^{\mathsf{t}}$ $Q \wedge P^{\mathsf{f}}$ P^{t} Q^{t} $P^{\mathsf{f}} \mid Q^{\mathsf{f}}$ $\bot \mid \bot$	$P \wedge (Q \vee \neg R) \wedge \neg Q^{\mathsf{t}}$ $P \wedge (Q \vee \neg R)^{\mathsf{t}}$ $\neg Q^{\mathsf{t}}$ Q^{f} P^{t} $Q \vee \neg R^{\mathsf{t}}$ $Q^{\mathsf{t}} \mid \neg R^{\mathsf{t}}$ $\bot \mid R^{\mathsf{f}}$
No Model	Herbrand Model $\{P^{\mathsf{t}}, Q^{\mathsf{f}}, R^{\mathsf{f}}\}$ $\varphi := \{P \mapsto \mathsf{T}, Q \mapsto \mathsf{F}, R \mapsto \mathsf{F}\}$

Algorithm: Fully expand all possible tableaux, (no rule can be applied)

- ▷ ▷ Satisfiable, iff there are open branches (correspond to models)

SOME RIGHTS RESERVED ©: Michael Kohlhase 47 JACOBS UNIVERSITY

Tableau calculi develop a formula in a tree-shaped arrangement that represents a case analysis on when a formula can be made true (or false). Therefore the formulae are decorated with exponents that hold the intended truth value.

On the left we have a refutation tableau that analyzes a negated formula (it is decorated with the intended truth value F). Both branches contain an elementary contradiction $\bot$.

On the right we have a model generation tableau, which analyzes a positive formula (it is decorated with the intended truth value T. This tableau uses the same rules as the refutation tableau, but makes a case analysis of when this formula can be satisfied. In this case we have a closed branch and an open one, which corresponds a model).

Now that we have seen the examples, we can write down the tableau rules formally.

Analytical Tableaux (Formal Treatment of $\mathcal{T}_0$)

- ▷ formula is analyzed in a tree to determine satisfiability
- ▷ branches correspond to valuations (models)
- ▷ one per connective

$$\frac{\mathbf{A}\wedge\mathbf{B}^{\mathsf{t}}}{\begin{array}{c}\mathbf{A}^{\mathsf{t}}\\\mathbf{B}^{\mathsf{t}}\end{array}}\mathcal{T}_0\wedge \qquad \frac{\mathbf{A}\wedge\mathbf{B}^{\mathsf{f}}}{\mathbf{A}^{\mathsf{f}}\,\Big|\,\mathbf{B}^{\mathsf{f}}}\mathcal{T}_0\vee \qquad \frac{\neg\mathbf{A}^{\mathsf{t}}}{\mathbf{A}^{\mathsf{f}}}\mathcal{T}_0\neg^{\mathsf{T}} \qquad \frac{\neg\mathbf{A}^{\mathsf{f}}}{\mathbf{A}^{\mathsf{t}}}\mathcal{T}_0\neg^{\mathsf{F}} \qquad \frac{\begin{array}{c}\mathbf{A}^{\alpha}\\\mathbf{A}^{\beta}\end{array}\quad \alpha\neq\beta}{\bot}\mathcal{T}_0\mathsf{cut}$$

- ▷ Use rules exhaustively as long as they contribute new material
- ▷ **Definition 2.42** Call a tableau saturated, iff no rule applies, and a branch closed, iff it ends in $\bot$, else open. (open branches in saturated tableaux yield models)
- ▷ **Definition 2.43 ($\mathcal{T}_0$-Theorem/Derivability)** $\mathbf{A}$ is a $\mathcal{T}_0$-theorem ($\vdash_{\mathcal{T}_0} \mathbf{A}$), iff there is a closed tableau with $\mathbf{A}^{\mathsf{F}}$ at the root.

  $\Phi \subseteq \mathit{wff}_o(\mathcal{V}_o)$ derives $\mathbf{A}$ in $\mathcal{T}_0$ ($\Phi \vdash_{\mathcal{T}_0} \mathbf{A}$), iff there is a closed tableau starting with $\mathbf{A}^{\mathsf{F}}$ and Φ^{T}.

SOME RIGHTS RESERVED ©: Michael Kohlhase 48 JACOBS UNIVERSITY

These inference rules act on tableaux have to be read as follows: if the formulae over the line appear in a tableau branch, then the branch can be extended by the formulae or branches below the line. There are two rules for each primary connective, and a branch closing rule that adds the special symbol $\bot$ (for unsatisfiability) to a branch.

We use the tableau rules with the convention that they are only applied, if they contribute new material to the branch. This ensures termination of the tableau procedure for propositional logic (every rule eliminates one primary connective).

Definition 2.44 We will call a closed tableau with the signed formula $\mathbf{A}^{\alpha}$ at the root a tableau refutation for $\mathcal{A}^{\alpha}$.

The saturated tableau represents a full case analysis of what is necessary to give $\mathbf{A}$ the truth value α; since all branches are closed (contain contradictions) this is impossible.

Definition 2.45 We will call a tableau refutation for $\mathbf{A}^{\mathsf{f}}$ a tableau proof for $\mathbf{A}$, since it refutes the possibility of finding a model where $\mathbf{A}$ evaluates to F. Thus $\mathbf{A}$ must evaluate to T in all models, which is just our definition of validity.

Thus the tableau procedure can be used as a calculus for propositional logic. In contrast to the calculus in section ?sec.hilbert? it does not prove a theorem $\mathbf{A}$ by deriving it from a set of axioms, but it proves it by refuting its negation. Such calculi are called negative or test calculi. Generally negative calculi have computational advantages over positive ones, since they have a built-in sense of direction.

We have rules for all the necessary connectives (we restrict ourselves to $\wedge$ and $\neg$, since the others can be expressed in terms of these two via the propositional identities above. For instance, we can write $\mathbf{A}\vee\mathbf{B}$ as $\neg(\neg\mathbf{A}\wedge\neg\mathbf{B})$, and $\mathbf{A}\Rightarrow\mathbf{B}$ as $\neg\mathbf{A}\vee\mathbf{B}$,. . . .)

We will now look at an example. Following our introduction of propositional logic in in ?impsem-ex? we look at a formulation of propositional logic with fancy variable names. Note that love(mary, bill) is just a variable name like P or X, which we have used earlier.

A Valid Real-World Example

▷ **Example 2.46** *If Mary loves Bill and John loves Mary, then John loves Mary*

$$
\begin{array}{c}
\mathsf{love}(\mathsf{mary},\mathsf{bill}) \wedge \mathsf{love}(\mathsf{john},\mathsf{mary}) \Rightarrow \mathsf{love}(\mathsf{john},\mathsf{mary})^{\mathsf{f}}\\
\neg(\neg\neg(\mathsf{love}(\mathsf{mary},\mathsf{bill}) \wedge \mathsf{love}(\mathsf{john},\mathsf{mary})) \wedge \neg\mathsf{love}(\mathsf{john},\mathsf{mary}))^{\mathsf{f}}\\
\neg\neg(\mathsf{love}(\mathsf{mary},\mathsf{bill}) \wedge \mathsf{love}(\mathsf{john},\mathsf{mary})) \wedge \neg\mathsf{love}(\mathsf{john},\mathsf{mary})^{\mathsf{t}}\\
\neg\neg(\mathsf{love}(\mathsf{mary},\mathsf{bill}) \wedge \mathsf{love}(\mathsf{john},\mathsf{mary}))^{\mathsf{t}}\\
\neg(\mathsf{love}(\mathsf{mary},\mathsf{bill}) \wedge \mathsf{love}(\mathsf{john},\mathsf{mary}))^{\mathsf{f}}\\
\mathsf{love}(\mathsf{mary},\mathsf{bill}) \wedge \mathsf{love}(\mathsf{john},\mathsf{mary})^{\mathsf{t}}\\
\neg\mathsf{love}(\mathsf{john},\mathsf{mary})^{\mathsf{t}}\\
\mathsf{love}(\mathsf{mary},\mathsf{bill})^{\mathsf{t}}\\
\mathsf{love}(\mathsf{john},\mathsf{mary})^{\mathsf{t}}\\
\mathsf{love}(\mathsf{john},\mathsf{mary})^{\mathsf{f}}\\
\bot
\end{array}
$$

This is a closed tableau, so the love(mary, bill) ∧ love(john, mary) ⇒ love(john, mary) is a $\mathcal{T}_0$-theorem.

As we will see, $\mathcal{T}_0$ is sound and complete, so love(mary, bill) ∧ love(john, mary) ⇒ love(john, mary) is valid.

SOME RIGHTS RESERVED ©: Michael Kohlhase 49 JACOBS UNIVERSITY

We could have used the entailment theorem (?entl-thm-cor?) here to show that *If Mary loves Bill and John loves Mary* entails *John loves Mary*. But there is a better way to show entailment: we directly use derivability in $\mathcal{T}_0$

Deriving Entailment in $\mathcal{T}_0$

▷ **Example 2.47** *Mary loves Bill* and *John loves Mary* together entail that *John loves Mary*

$$
\begin{array}{c}
\mathsf{love}(\mathsf{mary},\mathsf{bill})^{\mathsf{t}}\\
\mathsf{love}(\mathsf{john},\mathsf{mary})^{\mathsf{t}}\\
\mathsf{love}(\mathsf{john},\mathsf{mary})^{\mathsf{f}}\\
\bot
\end{array}
$$

This is a closed tableau, so the $\{\mathsf{love}(\mathsf{mary},\mathsf{bill}), \mathsf{love}(\mathsf{john},\mathsf{mary})\} \vdash_{\mathcal{T}_0} \mathsf{love}(\mathsf{john},\mathsf{mary})$, again, as $\mathcal{T}_0$ is sound and complete we have $\{\mathsf{love}(\mathsf{mary},\mathsf{bill}), \mathsf{love}(\mathsf{john},\mathsf{mary})\} \models \mathsf{love}(\mathsf{john},\mathsf{mary})$

SOME RIGHTS RESERVED ©: Michael Kohlhase 50 JACOBS UNIVERSITY

Note: that we can also use the tableau calculus to try and show entailment (and fail). The nice thing is that the failed proof, we can see what went wrong.

A Falsifiable Real-World Example

▷ **Example 2.48** **If Mary loves Bill or John loves Mary, then John loves*

Mary
Try proving the implication (this fails)

$$\begin{array}{c}(\mathsf{love}(\mathsf{mary},\mathsf{bill}) \lor \mathsf{love}(\mathsf{john},\mathsf{mary})) \Rightarrow \mathsf{love}(\mathsf{john},\mathsf{mary})^{\mathsf{f}}\\ \neg(\neg\neg(\mathsf{love}(\mathsf{mary},\mathsf{bill}) \lor \mathsf{love}(\mathsf{john},\mathsf{mary})) \land \neg\mathsf{love}(\mathsf{john},\mathsf{mary}))^{\mathsf{f}}\\ \neg\neg(\mathsf{love}(\mathsf{mary},\mathsf{bill}) \lor \mathsf{love}(\mathsf{john},\mathsf{mary})) \land \neg\mathsf{love}(\mathsf{john},\mathsf{mary})^{\mathsf{t}}\\ \neg\mathsf{love}(\mathsf{john},\mathsf{mary})^{\mathsf{t}}\\ \mathsf{love}(\mathsf{john},\mathsf{mary})^{\mathsf{f}}\\ \neg\neg(\mathsf{love}(\mathsf{mary},\mathsf{bill}) \lor \mathsf{love}(\mathsf{john},\mathsf{mary}))^{\mathsf{t}}\\ \neg(\mathsf{love}(\mathsf{mary},\mathsf{bill}) \lor \mathsf{love}(\mathsf{john},\mathsf{mary}))^{\mathsf{f}}\\ \mathsf{love}(\mathsf{mary},\mathsf{bill}) \lor \mathsf{love}(\mathsf{john},\mathsf{mary})^{\mathsf{t}}\\ \begin{array}{c|c}\mathsf{love}(\mathsf{mary},\mathsf{bill})^{\mathsf{t}} & \mathsf{love}(\mathsf{john},\mathsf{mary})^{\mathsf{t}}\\ & \bot\end{array}\end{array}$$

Indeed we can make $\mathcal{I}_\varphi(\mathsf{love}(\mathsf{mary},\mathsf{bill})) = \mathsf{T}$ but $\mathcal{I}_\varphi(\mathsf{love}(\mathsf{john},\mathsf{mary})) = \mathsf{F}$.

SOME RIGHTS RESERVED ©: Michael Kohlhase 51 JACOBS UNIVERSITY

Obviously, the tableau above is saturated, but not closed, so it is not a tableau proof for our initial entailment conjecture. We have marked the literals on the open branch green, since they allow us to read of the conditions of the situation, in which the entailment fails to hold. As we intuitively argued above, this is the situation, where Mary loves Bill. In particular, the open branch gives us a variable assignment (marked in green) that satisfies the initial formula. In this case, *Mary loves Bill*, which is a situation, where the entailment fails.

Again, the derivability version is much simpler

Testing for Entailment in $\mathcal{T}_0$

▷ **Example 2.49** Does *Mary loves Bill or John loves Mary* entail that *John loves Mary*?

$$\begin{array}{c}\mathsf{love}(\mathsf{mary},\mathsf{bill}) \lor \mathsf{love}(\mathsf{john},\mathsf{mary})^{\mathsf{t}}\\ \mathsf{love}(\mathsf{john},\mathsf{mary})^{\mathsf{f}}\\ \begin{array}{c|c}\mathsf{love}(\mathsf{mary},\mathsf{bill})^{\mathsf{t}} & \mathsf{love}(\mathsf{john},\mathsf{mary})^{\mathsf{t}}\\ & \bot\end{array}\end{array}$$

This saturated tableau has an open branch that shows that the interpretation with $\mathcal{I}_\varphi(\mathsf{love}(\mathsf{mary},\mathsf{bill})) = \mathsf{T}$ but $\mathcal{I}_\varphi(\mathsf{love}(\mathsf{john},\mathsf{mary})) = \mathsf{F}$ falsifies the derivability/entailment conjecture.

SOME RIGHTS RESERVED ©: Michael Kohlhase 52 JACOBS UNIVERSITY

2.4.2 Practical Enhancements for Tableaux

Propositional Identities

▷ **Definition 2.50** Let T and F be new logical constants with $\mathcal{I}(T) = \mathsf{T}$ and $\mathcal{I}(F) = \mathsf{F}$ for all assignments φ.

▷ We have to following identities:

Name	for $\wedge$	for $\vee$
Idenpotence	$\varphi \wedge \varphi = \varphi$	$\varphi \vee \varphi = \varphi$
Identity	$\varphi \wedge T = \varphi$	$\varphi \vee F = \varphi$
Absorption I	$\varphi \wedge F = F$	$\varphi \vee T = T$
Commutativity	$\varphi \wedge \psi = \psi \wedge \varphi$	$\varphi \vee \psi = \psi \vee \varphi$
Associativity	$\varphi \wedge (\psi \wedge \theta) = (\varphi \wedge \psi) \wedge \theta$	$\varphi \vee (\psi \vee \theta) = (\varphi \vee \psi) \vee \theta$
Distributivity	$\varphi \wedge (\psi \vee \theta) = \varphi \wedge \psi \vee \varphi \wedge \theta$	$\varphi \vee \psi \wedge \theta = (\varphi \vee \psi) \wedge (\varphi \vee \theta)$
Absorption II	$\varphi \wedge (\varphi \vee \theta) = \varphi$	$\varphi \vee \varphi \wedge \theta = \varphi$
De Morgan's Laws	$\neg(\varphi \wedge \psi) = \neg\varphi \vee \neg\psi$	$\neg(\varphi \vee \psi) = \neg\varphi \wedge \neg\psi$
Double negation	$\neg\neg\varphi = \varphi$	
Definitions	$\varphi \Rightarrow \psi = \neg\varphi \vee \psi$	$\varphi \Leftrightarrow \psi = (\varphi \Rightarrow \psi) \wedge (\psi \Rightarrow \varphi)$

SOME RIGHTS RESERVED ©: Michael Kohlhase 53 JACOBS UNIVERSITY

We have seen in the examples above that while it is possible to get by with only the connectives $\vee$ and $\neg$, it is a bit unnatural and tedious, since we need to eliminate the other connectives first. In this section, we will make the calculus less frugal by adding rules for the other connectives, without losing the advantage of dealing with a small calculus, which is good making statements about the calculus.

The main idea is to add the new rules as derived rules, i.e. inference rules that only abbreviate deductions in the original calculus. Generally, adding derived inference rules does not change the derivability relation of the calculus, and is therefore a safe thing to do. In particular, we will add the following rules to our tableau system.

We will convince ourselves that the first rule is a derived rule, and leave the other ones as an exercise.

Derived Rules of Inference

▷ **Definition 2.51** Let $\mathcal{C}$ be a calculus, a rule of inference $\frac{\mathbf{A}_1 \quad \cdots \quad \mathbf{A}_n}{\mathbf{C}}$ is called a derived inference rule in $\mathcal{C}$, iff there is a $\mathcal{C}$-proof of $\mathbf{A}_1, \ldots, \mathbf{A}_n \vdash \mathbf{C}$.

▷ **Definition 2.52** We have the following derived rules of inference

$$\frac{\mathbf{A} \Rightarrow \mathbf{B}^t}{\mathbf{A}^f \mid \mathbf{B}^t} \qquad \frac{\mathbf{A} \Rightarrow \mathbf{B}^f}{\begin{array}{c}\mathbf{A}^t\\ \mathbf{B}^f\end{array}} \qquad \frac{\begin{array}{c}\mathbf{A}^t\\ \mathbf{A} \Rightarrow \mathbf{B}^t\end{array}}{\mathbf{B}^t}$$

$$\frac{\mathbf{A} \vee \mathbf{B}^t}{\mathbf{A}^t \mid \mathbf{B}^t} \qquad \frac{\mathbf{A} \vee \mathbf{B}^f}{\begin{array}{c}\mathbf{A}^f\\ \mathbf{B}^f\end{array}} \qquad \frac{\mathbf{A} \Leftrightarrow \mathbf{B}^t}{\begin{array}{c|c}\mathbf{A}^t & \mathbf{A}^f\\ \mathbf{B}^t & \mathbf{B}^f\end{array}} \qquad \frac{\mathbf{A} \Leftrightarrow \mathbf{B}^f}{\begin{array}{c|c}\mathbf{A}^t & \mathbf{A}^f\\ \mathbf{B}^f & \mathbf{B}^t\end{array}}$$

$$\begin{array}{c|c}
\multicolumn{2}{c}{\mathbf{A}^t}\\
\multicolumn{2}{c}{\mathbf{A} \Rightarrow \mathbf{B}^t}\\
\multicolumn{2}{c}{\neg\mathbf{A} \vee \mathbf{B}^t}\\
\multicolumn{2}{c}{\neg(\neg\neg\mathbf{A} \wedge \neg\mathbf{B})^t}\\
\multicolumn{2}{c}{\neg\neg\mathbf{A} \wedge \neg\mathbf{B}^f}\\
\neg\neg\mathbf{A}^f & \neg\mathbf{B}^f\\
\neg\mathbf{A}^t & \mathbf{B}^t\\
\mathbf{A}^f & \\
\bot &
\end{array}$$

SOME RIGHTS RESERVED ©: Michael Kohlhase 54 JACOBS UNIVERSITY

With these derived rules, theorem proving becomes quite efficient. With these rules, the tableau (?tab:firsttab?) would have the following simpler form:

Tableaux with derived Rules (example)

Example 2.53

$$\begin{array}{c}\mathsf{love(mary,bill)} \wedge \mathsf{love(john,mary)} \Rightarrow \mathsf{love(john,mary)}^{\mathsf{f}}\\ \mathsf{love(mary,bill)} \wedge \mathsf{love(john,mary)}^{\mathsf{t}}\\ \mathsf{love(john,mary)}^{\mathsf{f}}\\ \mathsf{love(mary,bill)}^{\mathsf{t}}\\ \mathsf{love(john,mary)}^{\mathsf{t}}\\ \bot\end{array}$$

SOME RIGHTS RESERVED ©: Michael Kohlhase 55 JACOBS UNIVERSITY

Another thing that was awkward in (?tab:firsttab?) was that we used a proof for an implication to prove logical consequence. Such tests are necessary for instance, if we want to check consistency or informativity of new sentences[16]. Consider for instance a discourse $\Delta = \mathbf{D}^1, \ldots, \mathbf{D}^n$, where n is large. To test whether a hypothesis $\mathcal{H}$ is a consequence of Δ ($\Delta \models \mathbf{H}$) we need to show that $\mathbf{C} := (\mathbf{D}^1 \wedge \ldots) \wedge \mathbf{D}^n \Rightarrow \mathbf{H}$ is valid, which is quite tedious, since $\mathcal{C}$ is a rather large formula, e.g. if Δ is a 300 page novel. Moreover, if we want to test entailment of the form ($\Delta \models \mathbf{H}$) often, – for instance to test the informativity and consistency of every new sentence $\mathbf{H}$, then successive Δs will overlap quite significantly, and we will be doing the same inferences all over again; the entailment check is not incremental. EdN:16

Fortunately, it is very simple to get an incremental procedure for entailment checking in the model-generation-based setting: To test whether $\Delta \models \mathbf{H}$, where we have interpreted Δ in a model generation tableau $\mathcal{T}$, just check whether the tableau closes, if we add $\neg\mathbf{H}$ to the open branches. Indeed, if the tableau closes, then $\Delta \wedge \neg\mathbf{H}$ is unsatisfiable, so $\neg((\Delta \wedge \neg\mathbf{H}))$ is valid[17], but this is equivalent to $\Delta \Rightarrow \mathbf{H}$, which is what we wanted to show. EdN:17

Example 2.54 Consider for instance the following entailment in natural language.

Mary loves Bill. John loves Mary $\models$ *John loves Mary*

[18] We obtain the tableau EdN:18

$$\begin{array}{c}\mathrm{love(mary,bill)}^{\mathsf{t}}\\ \mathrm{love(john,mary)}^{\mathsf{t}}\\ \neg(\mathrm{love(john,mary)})^{\mathsf{t}}\\ \mathrm{love(john,mary)}^{\mathsf{f}}\\ \bot\end{array}$$

which shows us that the conjectured entailment relation really holds.

2.4.3 Soundness and Termination of Tableaux

As always we need to convince ourselves that the calculus is sound, otherwise, tableau proofs do not guarantee validity, which we are after. Since we are now in a refutation setting we cannot just show that the inference rules preserve validity: we care about unsatisfiability (which is the dual notion to validity), as we want to show the initial labeled formula to be unsatisfiable. Before we can do this, we have to ask ourselves, what it means to be (un)-satisfiable for a labeled formula or a tableau.

Soundness (Tableau)

- ▷ Idea: A test calculus is sound, iff it preserves satisfiability and the goal formulae are unsatisfiable.

[16] EdNote: add reference to presupposition stuff
[17] EdNote: Fix precedence of negation
[18] EdNote: need to mark up the embedding of NL strings into Math

▷ **Definition 2.55** A labeled formula $\mathbf{A}^\alpha$ is valid under φ, iff $\mathcal{I}_\varphi(\mathbf{A}) = \alpha$.

▷ **Definition 2.56** A tableau $\mathcal{T}$ is satisfiable, iff there is a satisfiable branch $\mathcal{P}$ in $\mathcal{T}$, i.e. if the set of formulae in $\mathcal{P}$ is satisfiable.

▷ **Lemma 2.57** *Tableau rules transform satisfiable tableaux into satisfiable ones.*

▷ **Theorem 2.58 (Soundness)** *A set Φ of propositional formulae is valid, if there is a closed tableau $\mathcal{T}$ for Φ^{f}.*

▷ Proof: by contradiction: Suppose Φ is not valid.

P.1 then the initial tableau is satisfiable (Φ^{f} satisfiable)

P.2 so $\mathcal{T}$ is satisfiable, by Lemma 3.98.

P.3 there is a satisfiable branch (by definition)

P.4 but all branches are closed ($\mathcal{T}$ closed)

□

SOME RIGHTS RESERVED ©: Michael Kohlhase

Thus we only have to prove Lemma 3.98, this is relatively easy to do. For instance for the first rule: if we have a tableau that contains $\mathbf{A} \wedge \mathbf{B}^{\mathsf{t}}$ and is satisfiable, then it must have a satisfiable branch. If $\mathbf{A} \wedge \mathbf{B}^{\mathsf{t}}$ is not on this branch, the tableau extension will not change satisfiability, so we can assue that it is on the satisfiable branch and thus $\mathcal{I}_\varphi(\mathbf{A} \wedge \mathbf{B}) = \mathsf{T}$ for some variable assignment φ. Thus $\mathcal{I}_\varphi(\mathbf{A}) = \mathsf{T}$ and $\mathcal{I}_\varphi(\mathbf{B}) = \mathsf{T}$, so after the extension (which adds the formulae $\mathbf{A}^{\mathsf{t}}$ and $\mathbf{B}^{\mathsf{t}}$ to the branch), the branch is still satisfiable. The cases for the other rules are similar.

The next result is a very important one, it shows that there is a procedure (the tableau procedure) that will always terminate and answer the question whether a given propositional formula is valid or not. This is very important, since other logics (like the often-studied first-order logic) does not enjoy this property.

Termination for Tableaux

▷ **Lemma 2.59** *The tableau procedure terminates, i.e. after a finite set of rule applications, it reaches a tableau, so that applying the tableau rules will only add labeled formulae that are already present on the branch.*

▷ Let us call a labeled formulae $\mathbf{A}^\alpha$ worked off in a tableau $\mathcal{T}$, if a tableau rule has already been applied to it.

▷ Proof:

P.1 It is easy to see tahat applying rules to worked off formulae will only add formulae that are already present in its branch.

P.2 Let $\mu(\mathcal{T})$ be the number of connectives in a labeled formulae in $\mathcal{T}$ that are not worked off.

P.3 Then each rule application to a labeled formula in $\mathcal{T}$ that is not worked off reduces $\mu(\mathcal{T})$ by at least one. (inspect the rules)

P.4 at some point the tableau only contains worked off formulae and literals.

P.5 since there are only finitely many literals in $\mathcal{T}$, so we can only apply the tableau cut rule a finite number of times. □

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

The Tableau calculus basically computes the disjunctive normal form: every branch is a disjunct that is a conjunct of literals. The method relies on the fact that a DNF is unsatisfiable, iff each monomial is, i.e. iff each branch contains a contradiction in form of a pair of complementary literals.

For proving completeness of tableaux we will use the abstract consistency method introduced by Raymond Smullyan — a famous logician who also wrote many books on recreational mathematics and logic (most notably one is titled "What is the name of this book?") which you may like.

2.4.4 Abstract Consistency and Model Existence

We will now come to an important tool in the theoretical study of reasoning calculi: the "abstract consistency"/"model existence" method. This method for analyzing calculi was developed by Jaako Hintikka, Raymond Smullyann, and Peter Andrews in 1950-1970 as an encapsulation of similar constructions that were used in completeness arguments in the decades before.[19] EdN:19

The basic intuition for this method is the following: typically, a logical system $\mathcal{S} = \langle \mathcal{L}, \mathcal{K}, \models \rangle$ has multiple calculi, human-oriented ones like the natural deduction calculi and machine-oriented ones like the automated theorem proving calculi. All of these need to be analyzed for completeness (as a basic quality assurance measure).

A completeness proof for a calculus $\mathcal{C}$ for $\mathcal{S}$ typically comes in two parts: one analyzes $\mathcal{C}$-consistency (sets that cannot be refuted in $\mathcal{C}$), and the other construct $\mathcal{K}$-models for $\mathcal{C}$-consistent sets.

In this situtation the "abstract consistency"/"model existence" method encapsulates the model construction process into a meta-theorem: the "model existence" theorem. This provides a set of syntactic ("abstract consistency") conditions for calculi that are sufficient to construct models.

With the model existence theorem it suffices to show that $\mathcal{C}$-consistency is an abstract consistency property (a purely syntactic task that can be done by a $\mathcal{C}$-proof transformation argument) to obtain a completeness result for $\mathcal{C}$.

Model Existence (Overview)

- ▷ Definition: Abstract consistency
- ▷ Definition: Hintikka set (maximally abstract consistent)
- ▷ Theorem: Hintikka sets are satisfiable
- ▷ Theorem: If Φ is abstract consistent, then Φ can be extended to a Hintikka set.
- ▷ Corollary: If Φ is abstract consistent, then Φ is satisfiable
- ▷ Application: Let $\mathcal{C}$ be a calculus, if Φ is $\mathcal{C}$-consistent, then Φ is abstract consistent.
- ▷ Corollary: $\mathcal{C}$ is complete.

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

The proof of the model existence theorem goes via the notion of a Hintikka set, a set of formulae with very strong syntactic closure properties, which allow to read off models. Jaako Hintikka's

[19] EdNote: cite the original papers!

original idea for completeness proofs was that for every complete calculus $\mathcal{C}$ and every $\mathcal{C}$-consistent set one can induce a Hintikka set, from which a model can be constructed. This can be considered as a first model existence theorem. However, the process of obtaining a Hintikka set for a set $\mathcal{C}$-consistent set Φ of sentences usually involves complicated calculus-dependent constructions.

In this situation, Raymond Smullyann was able to formulate the sufficient conditions for the existence of Hintikka sets in the form of "abstract consistency properties" by isolating the calculus-independent parts of the Hintikka set construction. His technique allows to reformulate Hintikka sets as maximal elements of abstract consistency classes and interpret the Hintikka set construction as a maximizing limit process.

To carry out the "model-existence"/"abstract consistency" method, we will first have to look at the notion of consistency.

Consistency and refutability are very important notions when studying the completeness for calculi; they form syntactic counterparts of satisfiability.

Consistency

- ▷ Let $\mathcal{C}$ be a calculus
- ▷ **Definition 2.60** Φ is called $\mathcal{C}$-refutable, if there is a formula $\mathbf{B}$, such that $\Phi \vdash_{\mathcal{C}} \mathbf{B}$ and $\Phi \vdash_{\mathcal{C}} \neg\mathbf{B}$.
- ▷ **Definition 2.61** We call a pair $\mathbf{A}$ and $\neg\mathbf{A}$ a contradiction.
- ▷ So a set Φ is $\mathcal{C}$-refutable, if $\mathcal{C}$ can derive a contradiction from it.
- ▷ **Definition 2.62** Φ is called $\mathcal{C}$-consistent, iff there is a formula $\mathbf{B}$, that is not derivable from Φ in $\mathcal{C}$.
- ▷ **Definition 2.63** We call a calculus $\mathcal{C}$ reasonable, iff implication elimination and conjunction introduction are admissible in $\mathcal{C}$ and $\mathbf{A} \wedge \neg\mathbf{A} \Rightarrow \mathbf{B}$ is a $\mathcal{C}$-theorem.
- ▷ **Theorem 2.64** *$\mathcal{C}$-inconsistency and $\mathcal{C}$-refutability coincide for reasonable calculi*

SOME RIGHTS RESERVED ©: Michael Kohlhase 59 JACOBS UNIVERSITY

It is very important to distinguish the syntactic $\mathcal{C}$-refutability and $\mathcal{C}$-consistency from satisfiability, which is a property of formulae that is at the heart of semantics. Note that the former specify the calculus (a syntactic device) while the latter does not. In fact we should actually say $\mathcal{S}$-satisfiability, where $\mathcal{S} = \langle \mathcal{L}, \mathcal{K}, \models \rangle$ is the current logical system.

Even the word "contradiction" has a syntactical flavor to it, it translates to "saying against each other" from its latin root.

Abstract Consistency

- ▷ **Definition 2.65** Let ∇ be a family of sets. We call ∇ closed under subset s, iff for each $\Phi \in \nabla$, all subsets $\Psi \subseteq \Phi$ are elements of ∇.
- ▷ **Notation 2.66** We will use $\Phi * \mathbf{A}$ for $\Phi \cup \{\mathbf{A}\}$.
- ▷ **Definition 2.67** A family ∇ of sets of propositional formulae is called an abstract consistency class, iff it is closed under subsets, and for each $\Phi \in \nabla$

  ∇_c) $P \notin \Phi$ or $\neg P \notin \Phi$ for $P \in \mathcal{V}_o$

$\nabla_\neg$) $\neg\neg\mathbf{A} \in \Phi$ implies $\Phi * \mathbf{A} \in \nabla$

$\nabla_\vee$) $(\mathbf{A} \vee \mathbf{B}) \in \Phi$ implies $\Phi * \mathbf{A} \in \nabla$ or $\Phi * \mathbf{B} \in \nabla$

$\nabla_\wedge$) $\neg(\mathbf{A} \vee \mathbf{B}) \in \Phi$ implies $(\Phi \cup \{\neg\mathbf{A}, \neg\mathbf{B}\}) \in \nabla$

▷ **Example 2.68** The empty set is an abstract consistency class

▷ **Example 2.69** The set $\{\emptyset, \{Q\}, \{P \vee Q\}, \{P \vee Q, Q\}\}$ is an abstract consistency class

▷ **Example 2.70** The family of satisfiable sets is an abstract consistency class.

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

So a family of sets (we call it a family, so that we do not have to say "set of sets" and we can distinguish the levels) is an abstract consistency class, iff if fulfills five simple conditions, of which the last three are closure conditions.

Think of an abstract consistency class as a family of "consistent" sets (e.g. $\mathcal{C}$-consistent for some calculus $\mathcal{C}$), then the properties make perfect sense: They are naturally closed under subsets — if we cannot derive a contradiction from a large set, we certainly cannot from a subset, furthermore,

∇_c) If both $P \in \Phi$ and $\neg P \in \Phi$, then Φ cannot be "consistent".

$\nabla_\neg$) If we cannot derive a contradiction from Φ with $\neg\neg\mathbf{A} \in \Phi$ then we cannot from $\Phi * \mathbf{A}$, since they are logically equivalent.

The other two conditions are motivated similarly.

Compact Collections

▷ **Definition 2.71** We call a collection ∇ of sets compact, iff for any set Φ we have
$\Phi \in \nabla$, iff $\Psi \in \nabla$ for every finite subset Ψ of Φ.

▷ **Lemma 2.72** *If ∇ is compact, then ∇ is closed under subsets.*

▷ Proof:

P.1 Suppose $S \subseteq T$ and $T \in \nabla$.

P.2 Every finite subset A of S is a finite subset of T.

P.3 As ∇ is compact, we know that $A \in \nabla$.

P.4 Thus $S \in \nabla$. □

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

The property of being closed under subsets is a "downwards-oriented" property: We go from large sets to small sets, compactness (the interesting direction anyways) is also an "upwards-oriented" property. We can go from small (finite) sets to large (infinite) sets. The main application for the compactness condition will be to show that infinite sets of formulae are in a family ∇ by testing all their finite subsets (which is much simpler).

We will carry out the proof here, since it gives us practice in dealing with the abstract consistency properties.

We now come to a very technical condition that will allow us to carry out a limit construction in the Hintikka set extension argument later.

Compact Collections

▷ **Definition 2.73** We call a collection ∇ of sets compact, iff for any set Φ we have
$\Phi \in \nabla$, iff $\Psi \in \nabla$ for every finite subset Ψ of Φ.

▷ **Lemma 2.74** *If ∇ is compact, then ∇ is closed under subsets.*

▷ Proof:

P.1 Suppose $S \subseteq T$ and $T \in \nabla$.

P.2 Every finite subset A of S is a finite subset of T.

P.3 As ∇ is compact, we know that $A \in \nabla$.

P.4 Thus $S \in \nabla$. □

SOME RIGHTS RESERVED ©: Michael Kohlhase 62 JACOBS UNIVERSITY

The property of being closed under subsets is a "downwards-oriented" property: We go from large sets to small sets, compactness (the interesting direction anyways) is also an "upwards-oriented" property. We can go from small (finite) sets to large (infinite) sets. The main application for the compactness condition will be to show that infinite sets of formulae are in a family ∇ by testing all their finite subsets (which is much simpler).

The main result here is that abstract consistency classes can be extended to compact ones. The proof is quite tedious, but relatively straightforward. It allows us to assume that all abstract consistency classes are compact in the first place (otherwise we pass to the compact extension).

Compact Abstract Consistency Classes

▷ **Lemma 2.75** *Any abstract consistency class can be extended to a compact one.*

▷ Proof:

P.1 We choose $\nabla' := \{\Phi \subseteq \mathit{wff}_o(\mathcal{V}_o) \,|\, \text{every finite subset of } \Phi \text{ is in } \nabla\}$.

P.2 Now suppose that $\Phi \in \nabla$. ∇ is closed under subsets, so every finite subset of Φ is in ∇ and thus $\Phi \in \nabla'$. Hence $\nabla \subseteq \nabla'$.

P.3 Next let us show that each ∇' is compact.

P.3.1 Suppose $\Phi \in \nabla'$ and Ψ is an arbitrary finite subset of Φ.

P.3.2 By definition of ∇' all finite subsets of Φ are in ∇ and therefore $\Psi \in \nabla'$.

P.3.3 Thus all finite subsets of Φ are in ∇' whenever Φ is in ∇'.

P.3.4 On the other hand, suppose all finite subsets of Φ are in ∇'.

P.3.5 Then by the definition of ∇' the finite subsets of Φ are also in ∇, so $\Phi \in \nabla'$. Thus ∇' is compact.

P.4 Note that ∇' is closed under subsets by the Lemma above.

P.5 Now we show that if ∇ satisfies ∇_*, then ∇' satisfies ∇_*.

P.5.1 To show ∇_c, let $\Phi \in \nabla'$ and suppose there is an atom $\mathbf{A}$, such that $\{\mathbf{A}, \neg\mathbf{A}\} \subseteq \Phi$. Then $\{\mathbf{A}, \neg\mathbf{A}\} \in \nabla$ contradicting ∇_c.

P.5.2 To show $\nabla_\neg$, let $\Phi \in \nabla'$ and $\neg\neg\mathbf{A} \in \Phi$, then $\Phi * \mathbf{A} \in \nabla'$.

P.5.2.1 Let Ψ be any finite subset of $\Phi * \mathbf{A}$, and $\Theta := (\Psi \backslash \{\mathbf{A}\}) * \neg\neg\mathbf{A}$.

P.5.2.2 Θ is a finite subset of Φ, so $\Theta \in \nabla$.

P.5.2.3 Since ∇ is an abstract consistency class and $\neg\neg\mathbf{A} \in \Theta$, we get $\Theta * \mathbf{A} \in \nabla$ by $\nabla_\neg$.

P.5.2.4 We know that $\Psi \subseteq \Theta * \mathbf{A}$ and ∇ is closed under subsets, so $\Psi \in \nabla$.

P.5.2.5 Thus every finite subset Ψ of $\Phi * \mathbf{A}$ is in ∇ and therefore by definition $\Phi * \mathbf{A} \in \nabla'$.

P.5.3 the other cases are analogous to $\nabla_\neg$. □

SOME RIGHTS RESERVED ©: Michael Kohlhase 63 JACOBS UNIVERSITY

Hintikka sets are sets of sentences with very strong analytic closure conditions. These are motivated as maximally consistent sets i.e. sets that already contain everything that can be consistently added to them.

∇-Hintikka Set

▷ **Definition 2.76** Let ∇ be an abstract consistency class, then we call a set $\mathcal{H} \in \nabla$ a ∇-Hintikka Set, iff $\mathcal{H}$ is maximal in ∇, i.e. for all $\mathbf{A}$ with $\mathcal{H} * \mathbf{A} \in \nabla$ we already have $\mathbf{A} \in \mathcal{H}$.

▷ **Theorem 2.77 (Hintikka Properties)** *Let ∇ be an abstract consistency class and $\mathcal{H}$ be a ∇-Hintikka set, then*

$\mathcal{H}_c$) For all $\mathbf{A} \in wff_o(\mathcal{V}_o)$ we have $\mathbf{A} \notin \mathcal{H}$ or $\neg\mathbf{A} \notin \mathcal{H}$

$\mathcal{H}_\neg$) If $\neg\neg\mathbf{A} \in \mathcal{H}$ then $\mathbf{A} \in \mathcal{H}$

$\mathcal{H}_\vee$) If $(\mathbf{A} \vee \mathbf{B}) \in \mathcal{H}$ then $\mathbf{A} \in \mathcal{H}$ or $\mathbf{B} \in \mathcal{H}$

$\mathcal{H}_\wedge$) If $\neg(\mathbf{A} \vee \mathbf{B}) \in \mathcal{H}$ then $\neg\mathbf{A}, \neg\mathbf{B} \in \mathcal{H}$

Proof:

▷ **P.1** We prove the properties in turn

P.1.1 $\mathcal{H}_c$: by induction on the structure of $\mathbf{A}$

P.1.1.1.1 $\mathbf{A} \in \mathcal{V}_o$: Then $\mathbf{A} \notin \mathcal{H}$ or $\neg\mathbf{A} \notin \mathcal{H}$ by ∇_c.

P.1.1.1.2 $\mathbf{A} = \neg\mathbf{B}$:

P.1.1.1.2.1 Let us assume that $\neg\mathbf{B} \in \mathcal{H}$ and $\neg\neg\mathbf{B} \in \mathcal{H}$,

P.1.1.1.2.2 then $\mathcal{H} * \mathbf{B} \in \nabla$ by $\nabla_\neg$, and therefore $\mathbf{B} \in \mathcal{H}$ by maximality.

P.1.1.1.2.3 So both $\mathbf{B}$ and $\neg\mathbf{B}$ are in $\mathcal{H}$, which contradicts the inductive hypothesis. □

P.1.1.1.3 $\mathbf{A} = \mathbf{B} \vee \mathbf{C}$: similar to the previous case: □

□

P.1.2 We prove $\mathcal{H}_\neg$ by maximality of $\mathcal{H}$ in ∇.:

P.1.2.1 If $\neg\neg\mathbf{A} \in \mathcal{H}$, then $\mathcal{H} * \mathbf{A} \in \nabla$ by $\nabla_\neg$.

P.1.2.2 The maximality of $\mathcal{H}$ now gives us that $\mathbf{A} \in \mathcal{H}$. □

P.1.3 other $\mathcal{H}_*$ are similar:

The following theorem is one of the main results in the "abstract consistency"/"model existence" method. For any abstract consistent set Φ it allows us to construct a Hintikka set $\mathcal{H}$ with $\Phi \in \mathcal{H}$.

Extension Theorem

▷ **Theorem 2.78** *If ∇ is an abstract consistency class and $\Phi \in \nabla$, then there is a ∇-Hintikka set $\mathcal{H}$ with $\Phi \subseteq \mathcal{H}$.*

▷ Proof:

P.1 Wlog. we assume that ∇ is compact (otherwise pass to compact extension)

P.2 We choose an enumeration $\mathbf{A}^1, \mathbf{A}^2, \ldots$ of the set $wff_o(\mathcal{V}_o)$

P.3 and construct a sequence of sets H^i with $H^0 := \Phi$ and

$$H^{n+1} := \begin{cases} H^n & \text{if } H^n * \mathbf{A}^n \notin \nabla \\ H^n * \mathbf{A}^n & \text{if } H^n * \mathbf{A}^n \in \nabla \end{cases}$$

P.4 Note that all $H^i \in \nabla$, choose $\mathcal{H} := \bigcup_{i\in\mathbb{N}} H^i$

P.5 $\Psi \subseteq \mathcal{H}$ finite implies there is a $j \in \mathbb{N}$ such that $\Psi \subseteq H^j$,

P.6 so $\Psi \in \nabla$ as ∇ closed under subsets and $\mathcal{H} \in \nabla$ as ∇ is compact.

P.7 Let $\mathcal{H} * \mathbf{B} \in \nabla$, then there is a $j \in \mathbb{N}$ with $\mathbf{B} = \mathbf{A}^j$, so that $\mathbf{B} \in H^{j+1}$ and $H^{j+1} \subseteq \mathcal{H}$

P.8 Thus $\mathcal{H}$ is ∇-maximal □

Note that the construction in the proof above is non-trivial in two respects. First, the limit construction for $\mathcal{H}$ is not executed in our original abstract consistency class ∇, but in a suitably extended one to make it compact — the original would not have contained $\mathcal{H}$ in general. Second, the set $\mathcal{H}$ is not unique for Φ, but depends on the choice of the enumeration of $wff_o(\mathcal{V}_o)$. If we pick a different enumeration, we will end up with a different $\mathcal{H}$. Say if $\mathbf{A}$ and $\neg\mathbf{A}$ are both ∇-consistent[20] with Φ, then depending on which one is first in the enumeration $\mathcal{H}$, will contain that one; with all the consequences for subsequent choices in the construction process. EdN:20

Valuation

▷ **Definition 2.79** A function $\nu\colon wff_o(\mathcal{V}_o) \to \mathcal{D}_o$ is called a valuation, iff

- ▷ $\nu(\neg\mathbf{A}) = \mathsf{T}$, iff $\nu(\mathbf{A}) = \mathsf{F}$
- ▷ $\nu(\mathbf{A} \vee \mathbf{B}) = \mathsf{T}$, iff $\nu(\mathbf{A}) = \mathsf{T}$ or $\nu(\mathbf{B}) = \mathsf{T}$

▷ **Lemma 2.80** *If $\nu\colon wff_o(\mathcal{V}_o) \to \mathcal{D}_o$ is a valuation and $\Phi \subseteq wff_o(\mathcal{V}_o)$ with $\nu(\Phi) = \{\mathsf{T}\}$, then Φ is satisfiable.*

[20] EdNote: introduce this above

▷ Proof Sketch: $\nu|_{\mathcal{V}_o}\colon \mathcal{V}_o \to \mathcal{D}_o$ is a satisfying variable assignment. □

▷ **Lemma 2.81** *If $\varphi\colon \mathcal{V}_o \to \mathcal{D}_o$ is a variable assignment, then $\mathcal{I}_\varphi\colon \mathit{wff}_o(\mathcal{V}_o) \to \mathcal{D}_o$ is a valuation.*

SOME RIGHTS RESERVED ©: Michael Kohlhase 66 JACOBS UNIVERSITY

Now, we only have to put the pieces together to obtain the model existence theorem we are after.

Model Existence

▷ **Lemma 2.82 (Hintikka-Lemma)** *If ∇ is an abstract consistency class and $\mathcal{H}$ a ∇-Hintikka set, then $\mathcal{H}$ is satisfiable.*

▷ Proof:

P.1 We define $\nu(\mathbf{A}) := \mathsf{T}$, iff $\mathbf{A} \in \mathcal{H}$

P.2 then ν is a valuation by the Hintikka properties

P.3 and thus $\nu|_{\mathcal{V}_o}$ is a satisfying assignment. □

▷ **Theorem 2.83 (Model Existence)** *If ∇ is an abstract consistency class and $\Phi \in \nabla$, then Φ is satisfiable.*

Proof:

▷ **P.1** There is a ∇-Hintikka set $\mathcal{H}$ with $\Phi \subseteq \mathcal{H}$ (Extension Theorem)

We know that $\mathcal{H}$ is satisfiable. (Hintikka-Lemma)

In particular, $\Phi \subseteq \mathcal{H}$ is satisfiable. □

SOME RIGHTS RESERVED ©: Michael Kohlhase 67 JACOBS UNIVERSITY

2.4.5 A Completeness Proof for Propositional Tableaux

With the model existence proof we have introduced in the last section, the completeness proof for first-order natural deduction is rather simple, we only have to check that Tableaux-consistency is an abstract consistency property.

We encapsulate all of the technical difficulties of the problem in a technical Lemma. From that, the completeness proof is just an application of the high-level theorems we have just proven.

P.2 P.3

Abstract Completeness for $\mathcal{T}_0$

▷ **Lemma 2.84** $\{\Phi \mid \Phi^{\mathsf{T}}$ has no closed Tableau$\}$ *is an abstract consistency class.*

▷ Proof: Let's call the set above ∇

P.1 We have to convince ourselves of the abstract consistency properties

P.1.1 ∇_c: $P, \neg P \in \Phi$ implies $P^{\mathsf{f}}, P^{\mathsf{t}} \in \Phi^{\mathsf{T}}$. □

P.1.2 $\nabla_\neg$: Let $\neg\neg\mathbf{A} \in \Phi$.

P.1.2.1 For the proof of the contrapositive we assume that $\Phi * \mathbf{A}$ has a closed tableau $\mathcal{T}$ and show that already Φ has one:

P.1.2.2 applying $\mathcal{T}_0\neg$ twice allows to extend any tableau with $\neg\neg\mathbf{B}^\alpha$ by $\mathbf{B}^\alpha$.

P.1.2.3 any path in $\mathcal{T}$ that is closed with $\neg\neg\mathbf{A}^\alpha$, can be closed by $\mathbf{A}^\alpha$. □

P.1.3 $\nabla_\vee$: Suppose $(\mathbf{A} \vee \mathbf{B}) \in \Phi$ and both $\Phi * \mathbf{A}$ and $\Phi * \mathbf{B}$ have closed tableaux

P.1.3.1 consider the tableaux:

$$\begin{array}{c} \Phi^\mathsf{T} \\ \mathbf{A}^\mathsf{t} \\ \mathit{Rest}^1 \end{array} \qquad \begin{array}{c} \Phi^\mathsf{T} \\ \mathbf{B}^\mathsf{t} \\ \mathit{Rest}^2 \end{array} \qquad \begin{array}{c} \Psi^\mathsf{T} \\ \mathbf{A} \vee \mathbf{B}^\mathsf{t} \\ \begin{array}{c|c} \mathbf{A}^\mathsf{t} & \mathbf{B}^\mathsf{t} \\ \mathit{Rest}^1 & \mathit{Rest}^2 \end{array} \end{array}$$

□

P.1.4 $\nabla_\wedge$: suppose, $\neg(\mathbf{A} \vee \mathbf{B}) \in \Phi$ and $\Phi\{\neg\mathbf{A}, \neg\mathbf{B}\}$ have closed tableau $\mathcal{T}$.

P.1.4.1 We consider

$$\begin{array}{c} \Phi^\mathsf{T} \\ \mathbf{A}^\mathsf{f} \\ \mathbf{B}^\mathsf{f} \\ \mathit{Rest} \end{array} \qquad \begin{array}{c} \Psi^\mathsf{T} \\ \mathbf{A} \vee \mathbf{B}^\mathsf{f} \\ \mathbf{A}^\mathsf{f} \\ \mathbf{B}^\mathsf{f} \\ \mathit{Rest} \end{array}$$

where $\Phi = \Psi * \neg(\mathbf{A} \vee \mathbf{B})$. □

□

SOME RIGHTS RESERVED ©: Michael Kohlhase 68 JACOBS UNIVERSITY

Observation: If we look at the completeness proof below, we see that the Lemma above is the only place where we had to deal with specific properties of the tableau calculus.

So if we want to prove completeness of any other calculus with respect to propositional logic, then we only need to prove an analogon to this lemma and can use the rest of the machinery we have already established "off the shelf".

This is one great advantage of the "abstract consistency method"; the other is that the method can be extended transparently to other logics.

Completeness of $\mathcal{T}_0$

▷ **Corollary 2.85** $\mathcal{T}_0$ *is complete.*

▷ Proof: by contradiction

P.1 We assume that $\mathbf{A} \in \mathit{wff}_o(\mathcal{V}_o)$ is valid, but there is no closed tableau for $\mathbf{A}^\mathsf{F}$.

P.2 We have $\{\neg\mathbf{A}\} \in \nabla$ as $\neg\mathbf{A}^\mathsf{T} = \mathbf{A}^\mathsf{F}$.

P.3 so $\neg\mathbf{A}$ is satisfiable by the model existence theorem (which is applicable as ∇ is an abstract consistency class by our Lemma above)

P.4 this contradicts our assumption that $\mathbf{A}$ is valid. □

SOME RIGHTS RESERVED ©: Michael Kohlhase 69 JACOBS UNIVERSITY

2.5 Tableaux and Model Generation

2.5.1 Tableau Branches and Herbrand Models

We have claimed above that the set of literals in open saturated tableau branches corresponds to a models. To gain an intuition, we will study our example above,

Model Generation and Interpretation

- ▷ **Example 2.86 (from above)** In Example 2.49 we claimed that

$$\mathcal{H} := \{\mathsf{love}(\mathsf{john},\mathsf{mary})^{\mathsf{F}},\mathsf{love}(\mathsf{mary},\mathsf{bill})^{\mathsf{T}}\}$$

  constitutes a model

$$\begin{array}{c|c} \multicolumn{2}{c}{\mathsf{love}(\mathsf{mary},\mathsf{bill}) \vee \mathsf{love}(\mathsf{john},\mathsf{mary})^{\mathsf{t}}} \\ \multicolumn{2}{c}{\mathsf{love}(\mathsf{john},\mathsf{mary})^{\mathsf{f}}} \\ \mathsf{love}(\mathsf{mary},\mathsf{bill})^{\mathsf{t}} & \mathsf{love}(\mathsf{john},\mathsf{mary})^{\mathsf{t}} \\ & \bot \end{array}$$

- ▷ Recap: A model $\mathcal{M}$ is a pair $\langle \mathcal{D},\mathcal{I} \rangle$, where $\mathcal{D}$ is a set of individuals, and $\mathcal{I}$ is an interpretation function.

- ▷ Problem: Find $\mathcal{D}$ and $\mathcal{I}$

SOME RIGHTS RESERVED ©: Michael Kohlhase 70 JACOBS UNIVERSITY

So the first task is to find a domain $\mathcal{D}$ of interpretation. Our formula mentions *Mary*, *John*, and *Bill*, which we assume to refer to distinct individuals so we need (at least) three individuals in the domain; so let us take $\mathcal{D} := \{A, B, C\}$ and fix $\mathcal{I}(\mathrm{mary}) = A$, $\mathcal{I}(\mathrm{bill}) = B$, $\mathcal{I}(\mathrm{john}) = C$.

So the only task is to find a suitable interpretation for the predicate love that makes love(john, mary) false and love(mary, bill) true. This is simple: we just take $\mathcal{I}(\mathrm{love}) = \{\langle A, B\rangle\}$. Indeed we have

$$\mathcal{I}_\varphi(\mathrm{love}(\mathrm{mary}, \mathrm{bill}) \vee \mathrm{love}(\mathrm{john}, \mathrm{mary})) = \mathsf{T}$$

but $\mathcal{I}_\varphi(\mathrm{love}(\mathrm{john}, \mathrm{mary})) = \mathsf{F}$ according to the rules in[21]. EdN:21

Model Generation and Models

- ▷ Idea: Choose the Universe $\mathcal{D}$ as the set Σ_0^f of constants, choose $\mathcal{I} = \mathsf{Id}_{\Sigma_0^f}$, interpret $p \in \Sigma_k^p$ via $\mathcal{I}(p) := \{\langle a_1, \ldots, a_k\rangle \,|\, p(a_1, \ldots, a_k) \in \mathcal{H}\}$.

- ▷ **Definition 2.87** We call a model a Herbrand model, iff $\mathcal{D} = \Sigma_0^f$ and $\mathcal{I} = \mathsf{Id}_{\Sigma_0^f}$.

- ▷ **Lemma 2.88** *Let $\mathcal{H}$ be a set of atomic formulae, then setting $\mathcal{I}(p) := \{\langle a_1, \ldots, a_k\rangle \,|\, p(a_1, \ldots, a_k) \in \mathcal{H}\}$. yields a Herbrand Model that satisfies $\mathcal{H}$.* *(proof trivial)*

- ▷ **Corollary 2.89** *Let $\mathcal{H}$ be a consistent (i.e. ∇_c holds) set of atomic formulae, then there is a Herbrand Model that satisfies $\mathcal{H}$.* *(take $\mathcal{H}^{\mathsf{T}}$)*

[21] EdNote: crossref

SOME RIGHTS RESERVED ©: Michael Kohlhase 71 JACOBS UNIVERSITY

In particular, the literals of an open saturated tableau branch $\mathcal{B}$ are a Herbrand model $\mathcal{H}$, as we have convinced ourselves above. By inspection of the inference rules above, we can further convince ourselves, that $\mathcal{H}$ satisfies all formulae on $\mathcal{B}$. We must only check that if $\mathcal{H}$ satisfies the succedents of the rule, then it satisfies the antecedent (which is immediate from the semantics of the principal connectives).

In particular, $\mathcal{H}$ is a model for the root formula of the tableau, which is on $\mathcal{B}$ by construction. So the tableau procedure is also a procedure that generates explicit (Herbrand) models for the root literal of the tableau. Every branch of the tableau corresponds to a (possibly) different Herbrand model. We will use this observation in the next section in an application to natural language semantics.

2.5.2 Using Model Generation for Interpretation

We will now use model generation directly as a tool for discourse interpretation.

Using Model Generation for Interpretation

- ▷ Idea: communication by natural language is a process of transporting parts of the mental model of the speaker into the the mental model of the hearer
- ▷ therefore: the interpretation process on the part of the hearer is a process of integrating the meaning of the utterances of the speaker into his mental model.
- ▷ model discourse understanding as a process of generating Herbrand models for the logical form of an utterance in a discourse by our tableau procedure.
- ▷ Advantage: capture ambiguity by generating multiple models for input logical forms.

SOME RIGHTS RESERVED ©: Michael Kohlhase 72 JACOBS UNIVERSITY

Tableaux Machine

- ▷ takes the logical forms (with salience expressions) as input,
- ▷ adds them to all/selected open branches,
- ▷ performs tableau inferences until some resource criterion is met
- ▷ output is application dependent; some choices are
  - ▷ the preferred model given as all the (positive) literals of the preferred branch;
  - ▷ the literals augmented with all non-expanded formulae (from the discourse); (resource-bound was reached)
  - ▷ machine answers user queries (preferred model $\models$ query?)
- ▷ model generation mode (guided by resources and strategies)
- ▷ theorem proving mode ($\Box$ for side conditions; using tableau rules)

SOME RIGHTS RESERVED ©: Michael Kohlhase 73 JACOBS UNIVERSITY

Model Generation Mode

- each proof rule comes with rule costs.
  - Ultimately we want bounded optimization regime [Russell'91]: expansion as long as expected gain in model quality outweighs proof costs
  - Here: each sentence in the discourse has a fixed inference budget Expansion until budget used up.

  Effect: Expensive rules are rarely applied.

- Warning: Finding appropriate values for the rule costs is a major open problem of our approach.

SOME RIGHTS RESERVED ©: Michael Kohlhase 74 JACOBS UNIVERSITY

Concretely, we treat discourse understanding as an online process that receives as input the logical forms of the sentences of the discourse one by one, and maintains a tableau that represents the current set of alternative models for the discourse. Since we are interested in the internal state of the machine (the current tableau), we do not specify the output of the tableau machine. We also assume that the tableau machine has a mechanism for choosing a preferred model from a set of open branches and that it maintains a set of deferred branches that can be re-visited, if extension of the the preferred model fails.

Upon input, the tableau machine will append the given logical form as a leaf to the preferred branch. (We will mark input logical forms in our tableaux by enclosing them in a box.) The machine then saturates the current tableau branch, exploring the set of possible models for the sequence of input sentences. If the subtableau generated by this saturation process contains open branches, then the machine chooses one of them as the preferred model, marks some of the other open branches as deferred, and waits for further input. If the saturation yields a closed sub-tableau, then the machine backtracks, i.e. selects a new preferred branch from the deferred ones, appends the input logical form to it, saturates, and tries to choose a preferred branch. Backtracking is repeated until successful, or until some termination criterion is met, in which case discourse processing fails altogether.

Two Readings

- **Example 2.90** *Peter loves Mary and Mary sleeps or Peter snores* (syntactically ambigous)

  Reading 1 $\mathrm{love}(\mathrm{peter},\mathrm{mary}) \wedge (\mathrm{sleep}(\mathrm{mary}) \vee \mathrm{snore}(\mathrm{peter}))$

  Reading 2 $\mathrm{love}(\mathrm{peter},\mathrm{mary}) \wedge \mathrm{sleep}(\mathrm{mary}) \vee \mathrm{snore}(\mathrm{peter})$

- Let us first consider the first reading in Example 2.90. Let us furthermore assume that we start out with the empty tableau, even though this is cognitively implausible, since it simplifies the presentation.

$$\begin{array}{c}
\boxed{\mathsf{love}(\mathsf{peter},\mathsf{mary}) \wedge (\mathsf{sleep}(\mathsf{mary}) \vee \mathsf{snore}(\mathsf{peter}))} \\
\mathsf{love}(\mathsf{peter},\mathsf{mary})^{\mathsf{t}} \\
\mathsf{sleep}(\mathsf{mary}) \vee \mathsf{snore}(\mathsf{peter})^{\mathsf{t}} \\
\mathsf{sleep}(\mathsf{mary})^{\mathsf{t}} \mid \mathsf{snore}(\mathsf{peter})^{\mathsf{t}}
\end{array}$$

▷ Observation: We have two models, so we have a case of semantical ambiguity.

SOME RIGHTS RESERVED ©: Michael Kohlhase

We see that model generation gives us two models; in both Peter loves Mary, in the first, Mary sleeps, and in the second one Peter snores. If we get a logically different input, e.g. the second reading in Example 2.90, then we obtain different models.

The other Reading

$$\begin{array}{c|c}
\multicolumn{2}{c}{\boxed{\mathsf{love}(\mathsf{peter},\mathsf{mary}) \wedge \mathsf{sleep}(\mathsf{mary}) \vee \mathsf{snore}(\mathsf{peter})}} \\
\mathsf{love}(\mathsf{peter},\mathsf{mary}) \wedge \mathsf{sleep}(\mathsf{mary})^{\mathsf{t}} & \mathsf{snore}(\mathsf{peter})^{\mathsf{t}} \\
\mathsf{love}(\mathsf{peter},\mathsf{mary})^{\mathsf{t}} & \\
\mathsf{sleep}(\mathsf{mary})^{\mathsf{t}} &
\end{array}$$

SOME RIGHTS RESERVED ©: Michael Kohlhase

In a discourse understanding system, both readings have to considered in parallel, since they pertain to a genuine ambiguity. The strength of our tableau-based proceedure is that it keeps the different readings around, so they can be acted upon later.

Note furthermore, that the overall (syntactical and semantic ambiguity) is not as bad as it looks: the left models of both readings are identical, so we only have three semantic readings not four.

Continuing the Discourse

▷ **Example 2.91** *Peter does not love Mary*
then the second tableau would be extended to

$$\begin{array}{c|c}
\multicolumn{2}{c}{\boxed{\mathsf{love}(\mathsf{peter},\mathsf{mary}) \wedge \mathsf{sleep}(\mathsf{mary}) \vee \mathsf{snore}(\mathsf{peter})}} \\
\mathsf{love}(\mathsf{peter},\mathsf{mary}) \wedge \mathsf{sleep}(\mathsf{mary})^{\mathsf{t}} & \mathsf{snore}(\mathsf{peter})^{\mathsf{t}} \\
\mathsf{love}(\mathsf{peter},\mathsf{mary})^{\mathsf{t}} & \boxed{\neg\mathsf{love}(\mathsf{peter},\mathsf{mary})} \\
\mathsf{sleep}(\mathsf{mary})^{\mathsf{t}} & \\
\boxed{\neg\mathsf{love}(\mathsf{peter},\mathsf{mary})} & \\
\mathsf{love}(\mathsf{peter},\mathsf{mary})^{\mathsf{f}} & \\
\bot &
\end{array}$$

and the first tableau closes altogether.

▷ In effect the choice of models has been reduced to one, which consitutes the intuitively correct reading of the discourse

SOME RIGHTS RESERVED ©: Michael Kohlhase

Model Generation models Discourse Understanding

- ▷ Conforms with psycholinguistic findings:
- ▷ [Zwaan'98]: listeners not only represent logical form, but also models containing referents
- ▷ [deVega'95]: online, incremental process
- ▷ [Singer'94]: enriched by background knowledge
- ▷ [Glenberg'87]: major function is to provide basis for anaphor resolution

SOME RIGHTS RESERVED ©: Michael Kohlhase 78 JACOBS UNIVERSITY

2.5.3 Adding Equality to $\mathcal{F}_1$

We will now extend $\mathrm{PL}_{\mathrm{NQ}}$ by equality, which is a very important relation in natural language. Generally, extending a logic with a new logical constant – equality is counted as a logical constant, since it semantics is fixed in all models – involves extending all three components of the logical system: the language, semantics, and the calculus.

$\mathrm{PL}_{\mathrm{NQ}}^{=}$: Adding Equality to $\mathrm{PL}_{\mathrm{NQ}}$

- ▷ Syntax: Just another binary predicate constant $=$
- ▷ Semantics: fixed as $\mathcal{I}_\varphi(a = b) = \mathsf{T}$, iff $\mathcal{I}_\varphi(a) = \mathcal{I}_\varphi(b)$. (logical symbol)
- ▷ **Definition 2.92 (Tableau Calculus $\mathcal{T}_{NQ}^{=}$)** add two additional inference rules (a positive and a negative) to $\mathcal{T}_0$

$$\frac{a \in \mathcal{H}}{a = a^{\mathsf{T}}}\mathcal{T}_{NQ}^{=}\mathsf{sym} \qquad \frac{\begin{array}{c} a = b^{\mathsf{T}} \\ \mathbf{A}\,[a]_p^\alpha \end{array}}{[b/p]\mathbf{A}^\alpha}\mathcal{T}_{NQ}^{=}\mathsf{rep}$$

where

- ▷ $\mathcal{H} \mathrel{\widehat{=}}$ the Herbrand Base, i.e. the set of constants occurring on the branch
- ▷ we write $\mathbf{C}\,[\mathbf{A}]_p$ to indicate that $\mathbf{C}|_p = \mathbf{A}$ ($\mathbf{C}$ has subterm $\mathbf{A}$ at position p).
- ▷ $[\mathbf{A}/p]\mathbf{C}$ is obtained from $\mathbf{C}$ by replacing the subterm at position p with $\mathbf{A}$.

SOME RIGHTS RESERVED ©: Michael Kohlhase 79 JACOBS UNIVERSITY

If we simplify the translation of definite descriptions, so that the phrase *the teacher* is translates to a concrete individual constant, then we can interpret (**??**) as (**??**).

Example: *Mary is the teacher. Peter likes the teacher.*

- ▷ Interpret as logical forms: $\mathsf{mary} = \mathsf{the\_teacher}$ and $\mathsf{like}(\mathsf{peter}, \mathsf{the\_teacher})$ and feed to tableau machine in turn.

▷ Model generation tableau

$$\begin{array}{c} \boxed{\mathsf{mary} = \mathsf{the\_teacher}^{\mathsf{T}}} \\ \boxed{\mathsf{like}(\mathsf{peter}, \mathsf{the\_teacher})^{\mathsf{T}}} \\ \mathsf{like}(\mathsf{peter}, \mathsf{mary})^{\mathsf{F}} \end{array}$$

▷ test whether this entails that *Peter likes Mary*

$$\begin{array}{c} \boxed{\mathsf{mary} = \mathsf{the\_teacher}^{\mathsf{T}}} \\ \boxed{\mathsf{like}(\mathsf{peter}, \mathsf{the\_teacher})^{\mathsf{T}}} \\ \mathsf{like}(\mathsf{peter}, \mathsf{mary})^{\mathsf{F}} \\ \mathsf{like}(\mathsf{peter}, \mathsf{the\_teacher})^{\mathsf{F}} \\ \bot \end{array}$$

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

Fragment 1

▷ Fragment $\mathcal{F}_1$ of English (defined by grammar + lexicon)

▷ Logic $\mathsf{PL_{NQ}}$ (serves as a mathematical model for $\mathcal{F}_1$)

- ▷ Formal Language (individuals, predicates, $\neg, \wedge, \vee, \Rightarrow$)
- ▷ Semantics $\mathcal{I}_\varphi$ defined recursively on formula structure ($\rightsquigarrow$ validity, entailment)
- ▷ Tableau calculus for validity and entailment (CALCULEMUS!)

▷ Analysis function $\mathcal{F}_1 \rightsquigarrow \mathsf{PL_{NQ}}$ (Translation)

▷ Test the Model by checking predictions (calculate truth conditions)

▷ Coverage: Extremely Boring!(accounts for 0 examples from the intro) but the conceptual setup is fascinating

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

3 Adding Context: Pronouns and World Knowledge

In this section we will extend the model generation system by facilities for dealing with world knowledge and pronouns. We want to cover discourses like *Peter loves Fido. Even though he bites him sometimes*. The idea here is to take the ideas from section [22] seriously and integrate them into the model generation system. As we already observed there, we crucially need a notion of context which determines the meaning of the pronoun. Furthermore, the example shows us that we will need to take into account world knowledge as A way to integrate world knowledge to filter out one interpretation, i.e. *Humans don't bite dogs.* EdN:22

In Subsection 3.0 we define the syntax and semantics of a new natural language fragment $\mathcal{F}_2$ which extends $\mathcal{F}_1$ by pronouns, which are translated to free variables in a suitlable extension of $\mathrm{PL_{NQ}}$

[22] EDNOTE: crossref fol.8

and its inference procedures. But this naive approach does not allow us to model enough world knowledge to do anything bigger. So we forge ahead and introduce first-order logic (Subsection 3.1) and its inference procedures (Subsection 3.2 and Subsection 3.4). This allows us to

3.1 First Attempt: Adding Pronouns and World Knowledge as Variables

3.1.1 Fragment 2: Pronouns and Anaphora

Fragment 2 ($\mathcal{F}_2 = \mathcal{F}_1$ + Pronouns)

- Want to cover: *Peter loves Fido. He bites him.* (almost intro)
  - We need: Translation and interpretation for *he*, *she*, *him*,. . . .
  - Also: A way to integrate world knowledge to filter out one interpretation (i.e. *Humans don't bite dogs.*)
- Idea: Integrate variables into $\mathsf{PL_{NQ}}$ (work backwards from that)
- Logical System: $\mathsf{PL}^{\mathcal{V}}_{\mathsf{NQ}} = \mathsf{PL_{NQ}}$ + variables (Translate pronouns to variables)

SOME RIGHTS RESERVED ©: Michael Kohlhase 82

New Grammar in Fragment 2 (Pronouns)

- **Definition 3.1** We have the following structural grammar rules in fragment 2.

S1.	S	$\to$	NPV^i
S2.	S	$\to$	NPV^tNP
N1.	NP	$\to$	N_{pr}
N2.	NP	$\to$	Pron
N3.	NP	$\to$	the$N$
S3.	S	$\to$	Itisnotthecasethat S
S4.	S	$\to$	S conj S
S5.	S	$\to$	NP is NP
S6.	S	$\to$	NP is Adj.

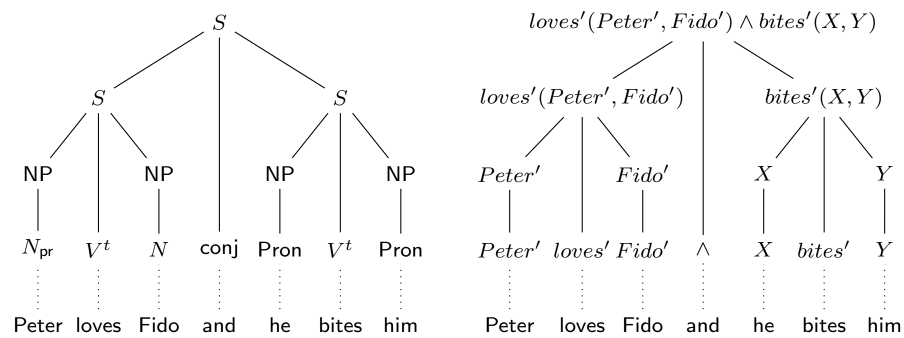
and one additional lexical rule:

L7.	Pron	$\to$	{he, she, it, we, they}

SOME RIGHTS RESERVED ©: Michael Kohlhase 83

Translation for $\mathcal{F}_2$ (first attempt)

- Idea: Pronouns are translated into new variables (so far)
- The syntax/semantic trees for *Peter loves Fido and he bites him.* are straightforward. (almost intro)



©: Michael Kohlhase 84

Predicate Logic with Variables (but no quantifiers)

- Logical System $\mathsf{PL}^{\mathcal{V}}_{\mathsf{NQ}}$: $\mathsf{PL}^{\mathcal{V}}_{\mathsf{NQ}} := \mathsf{PL}_{\mathsf{NQ}} + \text{variables}$
- **Definition 3.2 ($\mathbf{PL}^{\mathcal{V}}_{\mathbf{NQ}}$ Syntax)** category $\mathcal{V} = \{X, Y, Z, X^1, X^2, \ldots\}$ of variables (allow variables wherever individual constants were allowed)
- **Definition 3.3 ($\mathbf{PL}^{\mathcal{V}}_{\mathbf{NQ}}$ Semantics)** Model $\mathcal{M} = \langle \mathcal{D}, \mathcal{I} \rangle$ (need to evaluate variables)
  - variable assignment: $\varphi \colon \mathcal{V}_\iota \to \mathcal{D}$
  - evaluation function $\mathcal{I}_\varphi(X) = \varphi(X)$ (defined like $\mathcal{I}$ elsewhere)
  - call $\mathbf{A} \in \mathit{wff}_o(\Sigma, \mathcal{V}_\mathcal{T})$ valid in $\mathcal{M}$ under φ, iff $\mathcal{I}_\varphi(\mathbf{A}) = \mathsf{T}$,
  - call $\mathbf{A} \in \mathit{wff}_o(\Sigma, \mathcal{V}_\mathcal{T})$ satisfiable in $\mathcal{M}$, iff there is a variable assignment φ, such that $\mathcal{I}_\varphi(\mathbf{A}) = \mathsf{T}$

©: Michael Kohlhase 85

3.1.2 A Tableau Calculus for $PLNQ$ with Free Variables

The main idea here is to extend the fragment of first-order logic we use as a model for natural language to include free variables, and assume that pronouns like *he*, *she*, *it*, and *they* are translated to distinct free variables. Note that we do not allow quantifiers yet – that will come in [23], as quantifiers will pose new problems, and we can already solve some linguistically interesting problems without them. EdN:23

To allow for world knowledge, we generalize the notion of an initial tableau [24]. Instead of allowing only the initial signed formula at the root node, we allow a linear tree whose nodes are labeled with signed formulae representing the world knowledge. As the world knowledge resides in the initial tableau (intuitively before all input), we will also speak of background knowledge. EdN:24

We will use free variables for two purposes in our new fragment. Free variables in the input will stand for pronouns, their value will be determined by random instantiation. Free variables in the

[23] EdNote: crossref
[24] EdNote: crossref

world knowledge allow us to express schematic knowledge. For instance, if we want to express *Humans don't bite dogs.*, then we can do this by the formula $\mathrm{human}(X) \wedge \mathrm{dog}(Y) \Rightarrow \neg\mathrm{bite}(X,Y)$.

Of course we will have to extend our tableau calculus with new inference rules for the new language capabilities.

A Tableau Calculus for $\mathrm{PL}^{\mathcal{V}}_{\mathrm{NQ}}$

▷ **Definition 3.4 (Tableau Calculus for $\mathrm{PL}^{\mathcal{V}}_{\mathrm{NQ}}$)** $\mathcal{T}^p_{\mathcal{V}} = \mathcal{T}_0$ + new tableau rules for formulae with variables

$$\frac{\begin{array}{c}\vdots\\ \mathbf{A}^\alpha \quad c \in \mathcal{H}\\ \vdots\end{array}}{[c/X](\mathbf{A})^\alpha}\,\mathcal{T}^p_{\mathcal{V}}\text{:WK} \qquad \frac{\begin{array}{cc}\vdots & \mathcal{H} = \{a_1, \ldots, a_n\}\\ \boxed{\mathbf{A}^\alpha} & \mathbf{free}(\mathbf{A}) = \{X_1, \ldots, X_m\}\end{array}}{\sigma_1(\mathbf{A})^\alpha \mid \ldots \mid \sigma_{m^n}(\mathbf{A})^\alpha}\,\mathcal{T}^p_{\mathcal{V}}\text{:Ana}$$

$\mathcal{H}$ is the set of ind. constants in the branch above (Herbrand Base)

and the σ_i are substitutions that instantiate the X_j with any combinations of the a_k (there are m^n of them).

▷ the first rule is used for world knowledge (up in the branch)

▷ the second rule is used for input sentences $\boxed{\cdots}$

this rule has to be applied eagerly (while they are still at the leaf)

SOME RIGHTS RESERVED ©: Michael Kohlhase 86 JACOBS UNIVERSITY

Let us look at two examples.

To understand the role of background knowledge we interpret *Peter snores* with respect to the knowledge that *Only sleeping people snore*.

Some Examples in $\mathcal{F}_2$

▷ **Example 3.5 (Peter snores)** (Only sleeping people snore)

$$\begin{array}{c}\mathsf{snore}(X) \Rightarrow \mathsf{sleep}(X)^{\mathsf{T}}\\ \boxed{\mathsf{snore}(\mathsf{peter})^{\mathsf{T}}}\\ \mathsf{snore}(\mathsf{peter}) \Rightarrow \mathsf{sleep}(\mathsf{peter})^{\mathsf{T}}\\ \mathsf{sleep}(\mathsf{peter})^{\mathsf{T}}\end{array}$$

▷ **Example 3.6 (Peter sleeps. John walks. He snores)** (who snores?)

$$\begin{array}{c}\boxed{\mathsf{sleep}(\mathsf{peter})^{\mathsf{T}}}\\ \boxed{\mathsf{walk}(\mathsf{john})^{\mathsf{T}}}\\ \boxed{\mathsf{snore}(X)^{\mathsf{T}}}\\ \mathsf{snore}(\mathsf{peter})^{\mathsf{T}} \mid \mathsf{snore}(\mathsf{john})^{\mathsf{T}}\end{array}$$

SOME RIGHTS RESERVED ©: Michael Kohlhase 87 JACOBS UNIVERSITY

The background knowledge is represented in the schematic formula in the first line of the tableau. Upon receiving the input, the tableau instantiates the schema to line three and uses the chaining rule from [25] to derive the fact that peter must sleep. EdN:25

The third input formula contains a free variable, which is instantiated by all constants in the Herbrand base (two in our case). This gives rise to two models that correspond to the two readings of the discourse.

Let us now look at an example with more realistic background knowledge.

Say we know that birds fly, if they are not penguins. Furthermore, eagles and penguins are birds, but eagles are not penguins. Then we can answer the classic question *Does Tweety fly?* by the following two tableaux.

Does Tweety fly?

Tweety is a bird

$$\begin{array}{c|c}
\multicolumn{2}{c}{\mathsf{bird}(X) \Rightarrow \mathsf{fly}(X) \vee \mathsf{penguin}(X)^{\mathsf{T}}} \\
\multicolumn{2}{c}{\mathsf{eagle}(X) \Rightarrow \mathsf{bird}(X)^{\mathsf{T}}} \\
\multicolumn{2}{c}{\mathsf{eagle}(X) \Rightarrow \neg(\mathsf{penguin}(X))^{\mathsf{T}}} \\
\multicolumn{2}{c}{\mathsf{penguin}(X) \Rightarrow \neg(\mathsf{fly}(X))^{\mathsf{T}}} \\
\multicolumn{2}{c}{\boxed{\mathsf{bird}(tweety)^{\mathsf{T}}}} \\
\multicolumn{2}{c}{\mathsf{fly}(tweety) \vee \mathsf{penguin}(tweety)^{\mathsf{T}}} \\
\mathsf{fly}(tweety)^{\mathsf{T}} & \mathsf{penguin}(tweety)^{\mathsf{T}} \\
 & \neg(\mathsf{fly}(tweety))^{\mathsf{T}} \\
 & \mathsf{fly}(tweety)^{\mathsf{F}}
\end{array}$$

Tweety is an eagle

$$\begin{array}{c|c}
\multicolumn{2}{c}{\mathsf{bird}(X) \Rightarrow \mathsf{fly}(X) \vee \mathsf{penguin}(X)^{\mathsf{T}}} \\
\multicolumn{2}{c}{\mathsf{eagle}(X) \Rightarrow \mathsf{bird}(X)^{\mathsf{T}}} \\
\multicolumn{2}{c}{\mathsf{eagle}(X) \Rightarrow \neg(\mathsf{penguin}(X))^{\mathsf{T}}} \\
\multicolumn{2}{c}{\mathsf{penguin}(X) \Rightarrow \neg(\mathsf{fly}(X))^{\mathsf{T}}} \\
\multicolumn{2}{c}{\boxed{\mathsf{eagle}(tweety)^{\mathsf{T}}}} \\
\multicolumn{2}{c}{\mathsf{fly}(tweety) \vee \mathsf{penguin}(tweety)^{\mathsf{T}}} \\
\mathsf{fly}(tweety)^{\mathsf{T}} & \mathsf{penguin}(tweety)^{\mathsf{T}} \\
 & \neg(\mathsf{eagle}(tweety))^{\mathsf{T}} \\
 & \mathsf{eagle}(tweety)^{\mathsf{F}} \\
 & \bot
\end{array}$$

SOME RIGHTS RESERVED ©: Michael Kohlhase 88 JACOBS UNIVERSITY

3.1.3 Case Study: Peter loves Fido, even though he sometimes bites him

Let us now return to the motivating example from the introduction, and see how our system fares with it (this allows us to test our computational/linguistic theory). We will do this in a completely naive manner and see what comes out.

The first problem we run into immediately is that we do not know how to cope with *even though* and *sometimes*, so we simplify the discourse to *Peter loves Fido and he bites him.*.

Finally: *Peter loves Fido. He bites him.*

▷ Let's try it naively (worry about the problems later.)

$$\begin{array}{c|c|c|c}
\multicolumn{4}{c}{\boxed{l(p,f)^{\mathsf{T}}}} \\
\multicolumn{4}{c}{\boxed{b(X,Y)^{\mathsf{T}}}} \\
b(p,p)^{\mathsf{T}} & b(p,f)^{\mathsf{T}} & b(f,p)^{\mathsf{T}} & b(f,f)^{\mathsf{T}}
\end{array}$$

▷ Problem: We get four readings instead of one!

▷ Idea: We have not specified enough world knowledge

[25] EdNote: crossref

SOME RIGHTS RESERVED ©: Michael Kohlhase 89

The next problem is obvious: We get four readings instead of one (or two)! What has happened? If we look at the models, we see that we did not even specify the background knowledge that was supposed filter out the one intended reading.

We try again with the additional knowledge that *Nobody bites himself* and *Humans do not bite dogs*.

Peter and Fido with World Knowledge

▷ Nobody bites himself, humans do not bite dogs.

$$\begin{array}{c}
\mathsf{dog}(f)^{\mathsf{T}}\\
\mathsf{man}(p)^{\mathsf{T}}\\
b(X,X)^{\mathsf{F}}\\
\mathsf{dog}(X)\wedge\mathsf{man}(Y)\Rightarrow\neg(b(Y,X))^{\mathsf{T}}\\
\boxed{l(p,f)^{\mathsf{T}}}\\
\boxed{b(X,Y)^{\mathsf{T}}}\\
\begin{array}{c|c|c|c}
\begin{array}{c} b(p,p)^{\mathsf{T}}\\ b(p,p)^{\mathsf{F}}\\ \bot \end{array} &
\begin{array}{c} b(p,f)^{\mathsf{T}}\\ \mathsf{dog}(f)\wedge\mathsf{man}(p)\Rightarrow\neg(b(p,f))^{\mathsf{T}}\\ b(p,f)^{\mathsf{F}}\\ \bot \end{array} &
b(f,p)^{\mathsf{T}} &
\begin{array}{c} b(f,f)^{\mathsf{T}}\\ b(f,f)^{\mathsf{F}}\\ \bot \end{array}
\end{array}
\end{array}$$

▷ Observation: Pronoun resolution introduces ambiguities.

▷ Pragmatics: Use world knowledge to filter out impossible readings.

SOME RIGHTS RESERVED ©: Michael Kohlhase 90

We observe that our extended tableau calculus was indeed able to handle this example, if we only give it enough background knowledge to act upon.

But the world knowledge we can express in $\mathrm{PL}^{=}_{\mathrm{NQ}}$ is very limited. We can say that humans do not bite dogs, but we cannot provide the background knowledge to understand a sentence like *Peter was late for class today, the car had a flat tire.*, which needs the

3.1.4 The computational Role of Ambiguities

In the case study, we have seen that pronoun resolution introduces ambiguities, and we can use world knowledge to filter out impossible readings. Generally in the traditional waterfall model of language processing, [4] every processing stage introduces ambiguities that need to be resolved in this stage or later.

The computational Role of Ambiguities

▷ Observation: (in the traditional waterfall model) Every processing stage introduces ambiguities that need to be resolved.

- ▷ Syntax: e.g. *Peter chased the man in the red sports car* (attachment)
- ▷ Semantics: e.g. *Peter went to the bank* (lexical)

[4] which posits that NL understanding is a process that analyzes the input in stages: syntax, semantics composition, pragmatics

▷ Pragmatics: e.g. *Two men carried two bags* (collective vs. distributive)

▷ Question: Where does pronoun-ambiguity belong? (much less clear)

▷ Answer: we have freedom to choose

1. resolve the pronouns in the syntax (generic waterfall model)
   $\rightsquigarrow$ multiple syntactic representations (pragmatics as filter)
2. resolve the pronouns in the pragmatics (our model here)
   $\rightsquigarrow$ need underspecified syntactic representations (e.g. variables)
   $\rightsquigarrow$ pragmatics needs ambiguity treatment (e.g. tableaux)

For pronoun ambiguities, this is much less clear. In a way we have the freedom to choose. We can

1. resolve the pronouns in the syntax as in the generic waterfall model, then we arrive at multiple syntactic representations, and can use pragmatics as filter to get rid of unwanted readings
2. resolve the pronouns in the pragmatics (our model here) then we need underspecified syntactic representations (e.g. variables) and pragmatics needs ambiguity treatment (in our case the tableaux).

We will continue to explore the second alternative in more detail, and refine the approach. One of the advantages of treating the anaphoric ambiguities in the syntax is that syntactic agreement information like gender can be used to disambiguate. Say that we vary the example from section **??** to *Peter loves Mary. She loves him.*.

Translation for $\mathcal{F}_2$

▷ Idea: Pronouns are translated into new variables (so far)

▷ Problem: *Peter loves Mary. She loves him.*

$$\boxed{\mathsf{love}(\mathsf{peter},\mathsf{mary})^{\mathsf{T}}}$$
$$\boxed{\mathsf{love}(X,Y)^{\mathsf{T}}}$$
$$\mathsf{love}(\mathsf{peter},\mathsf{peter})^{\mathsf{T}} \mid \mathsf{love}(\mathsf{peter},\mathsf{mary})^{\mathsf{T}} \mid \mathsf{love}(\mathsf{mary},\mathsf{peter})^{\mathsf{T}} \mid \mathsf{love}(\mathsf{mary},\mathsf{mary})^{\mathsf{T}}$$

▷ Idea: attach world knowledge to pronouns (just as with Peter and Fido)

▷ use the world knowledge to distinguish gender by predicates masc and fem

▷ Idea: attach world knowledge to pronouns (just as with Peter and Fido)

▷ Problem: properties of

▷ proper names are given in the model,

▷ pronouns must be given by the syntax/semantics interface

▷ How to generate $\mathsf{love}(X, Y) \wedge (\mathsf{masc}(X) \wedge \mathsf{fem}(Y))$ compositionally?

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

The tableau (over)-generates the full set of pronoun readings. At first glance it seems that we can fix this just like we did in section **??** by attaching world knowledge to pronouns, just as with Peter and Fido. Then we could use the world knowledge to distinguish gender by predicates, say masc and fem.

But if we look at the whole picture of building a system, we can see that this idea will not work. The problem is that properties of proper names like Fido are given in the background knowledge, whereas the relevant properties of *pronouns* must be given by the syntax/semantics interface. Concretely, we would need to generate $\mathsf{love}(X, Y) \wedge (\mathsf{masc}(X) \wedge \mathsf{fem}(Y))$ for *She loves him*. How can we do such a thing compositionally?

Again we basically have two options, we can either design a clever syntax/semantics interface, or we can follow the lead of Montague semantics[26] and extend the logic, so that compositionality becomes simpler to achieve. We will explore the latter option in the next section. EdN:26

The problem we stumbled across in the last section is how to associate certain properties (in this case agreement information) with variables compositionally. Fortunately, there is a ready-made logical theory for it. Sorted first-order logic. Actually there are various sorted first-order logics, but we will only need the simplest one for our application at the moment.

Sorted first-order logic extends the language with a set $\mathcal{S}$ of sorts $\mathbb{A}, \mathbb{B}, \mathbb{C}, \ldots$, which are just special symbols that are attached to all terms in the language.

Syntactically, all constants, and variables are assigned sorts, which are annotated in the lower index, if they are not clear from the context. Semantically, the universe $\mathcal{D}_\iota$ is subdivided into subsets $\mathcal{D}_\mathbb{A} \subseteq \mathcal{D}_\iota$, which denote the objects of sort $\mathbb{A}$; furthermore, the interpretation function $\mathcal{I}$ and variable assignment φ have to be well-sorted. Finally, on the calculus level, the only change we have to make is to restrict instantiation to well-sorted substitutions:

Sorts refine World Categories

▷ **Definition 3.7 (Sorted Logics)** (in our case $PL^1_\mathcal{S}$)

assume a set of sorts $\mathcal{S} := \{\mathbb{A}, \mathbb{B}, \mathbb{C}, \ldots\}$ (everything well-sorted)

- ▷ Syntax: variables and constants are sorted $X_\mathbb{A}, Y_\mathbb{B}, Z^1_{\mathbb{C}!1} \ldots, a_\mathbb{A}, b_\mathbb{A}, \ldots$
- ▷ Semantics: subdivide the Universe $\mathcal{D}_\iota$ into subsets $\mathcal{D}_\mathbb{A} \subseteq \mathcal{D}_\iota$
  Interpretation $\mathcal{I}$ and variable assignment φ have to be well-sorted $\mathcal{I}(a_\mathbb{A}), \varphi(X_\mathbb{A}) \in \mathcal{D}_\mathbb{A}$.
- ▷ Calculus: substitutions must be well-sorted $[a_\mathbb{A}/X_\mathbb{A}]$ OK, $[a_\mathbb{A}/X_\mathbb{B}]$ not.

▷ Observation: Sorts do not add expressivity in principle (just practically)

- ▷ Translate $R(X_\mathbb{A}) \wedge \neg(P(Z_\mathbb{C}))$ to $\mathcal{R}_\mathbb{A}(X) \wedge \mathcal{R}_\mathbb{C}(Z) \Rightarrow R(X) \wedge \neg(P(Z))$ in world knowledge.
- ▷ Translate $R(X_\mathbb{A}) \wedge \neg(P(Z_\mathbb{C}))$ to $\mathcal{R}_\mathbb{A}(X) \wedge \mathcal{R}_\mathbb{C}(Z) \wedge R(X \wedge Y) \wedge \neg(P(Z))$ in input.
- ▷ Meaning is preserved, but translation is compositional!

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

[26] EdNote: crossref

3.2 First-Order Logic

First-order logic is the most widely used formal system for modelling knowledge and inference processes. It strikes a very good bargain in the trade-off between expressivity and conceptual and computational complexity. To many people first-order logic is "the logic", i.e. the only logic worth considering, its applications range from the foundations of mathematics to natural language semantics.

First-Order Predicate Logic (PL^1)

- Coverage: We can talk about (*All humans are mortal*)
  - individual things and denote them by variables or constants
  - properties of individuals, (e.g. being human or mortal)
  - relations of individuals, (e.g. *sibling_of* relationship)
  - functions on individuals, (e.g. the *father_of* function)

  We can also state the existence of an individual with a certain property, or the universality of a property.
- But we cannot state assertions like
  - *There is a surjective function from the natural numbers into the reals.*
- First-Order Predicate Logic has many good properties (complete calculi, compactness, unitary, linear unification,...)
- But too weak for formalizing: (at least directly)
  - natural numbers, torsion groups, calculus, ...
  - generalized quantifiers (*most, at least three, some,...*)

SOME RIGHTS RESERVED ©: Michael Kohlhase 94 JACOBS UNIVERSITY

We will now introduce the syntax and semantics of first-order logic. This introduction differs from what we commonly see in undergraduate textbooks on logic in the treatment of substitutions in the presence of bound variables. These treatments are non-syntactic, in that they take the renaming of bound variables (α-equivalence) as a basic concept and directly introduce capture-avoiding substitutions based on this. But there is a conceptual and technical circularity in this approach, since a careful definition of α-equivalence needs substitutions.

In this subsection we follow Peter Andrews' lead from [And02] and break the circularity by introducing syntactic substitutions, show a substitution value lemma with a substitutability condition, use that for a soundness proof of α-renaming, and only then introduce capture-avoiding substitutions on this basis. This can be done for any logic with bound variables, we go through the details for first-order logic here as an example.

3.2.1 First-Order Logic: Syntax and Semantics

The syntax and semantics of first-order logic is systematically organized in two distinct layers: one for truth values (like in propositional logic) and one for individuals (the new, distinctive feature of first-order logic).

The first step of defining a formal language is to specify the alphabet, here the first-order signatures and their components.

PL^1 Syntax (Signature and Variables)

- ▷ **Definition 3.8** First-order logic (PL^1), is a formal logical system extensively used in mathematics, philosophy, linguistics, and computer science. It combines propositional logic with the ability to quantify over individuals.
- ▷ PL^1 talks about two kinds of objects: (so we have two kinds of symbols)
  - ▷ truth value s; sometimes annotated by type o (like in PL^0)
  - ▷ individual s; sometimes annotated by type ι (numbers, foxes, Pokémon,...)
- ▷ **Definition 3.9** A first-order signature consists of (all disjoint; $k \in \mathbb{N}$)
  - ▷ connective s: $\Sigma^o = \{T, F, \neg, \vee, \wedge, \Rightarrow, \Leftrightarrow, \ldots\}$ (functions on truth values)
  - ▷ function constant s: $\Sigma^f_k = \{f, g, h, \ldots\}$ (functions on individuals)
  - ▷ predicate constant s: $\Sigma^p_k = \{p, q, r, \ldots\}$ (relations among inds.)
  - ▷ (Skolem constant s: $\Sigma^{sk}_k = \{f^k_1, f^k_2, \ldots\}$) (witness constructors; countably ∞)
  - ▷ We take the signature Σ to be all of these together: $\Sigma := \Sigma^o \cup \Sigma^f \cup \Sigma^p \cup \Sigma^{sk}$, where $\Sigma^* := \bigcup_{k\in\mathbb{N}} \Sigma^*_k$.
- ▷ We assume a set of individual variable s: $\mathcal{V}_\iota = \{X_\iota, Y_\iota, Z, X^1{}_\iota, X^2\}$ (countably ∞)

SOME RIGHTS RESERVED ©: Michael Kohlhase 95 JACOBS UNIVERSITY

We make the deliberate, but non-standard design choice here to include Skolem constants into the signature from the start. These are used in inference systems to give names to objects and construct witnesses. Other than the fact that they are usually introduced by need, they work exactly like regular constants, which makes the inclusion rather painless. As we can never predict how many Skolem constants we are going to need, we give ourselves countably infinitely many for every arity. Our supply of individual variables is countably infinite for the same reason.

The formulae of first-order logic is built up from the signature and variables as terms (to represent individuals) and propositions (to represent propositions). The latter include the propositional connectives, but also quantifiers.

PL^1 Syntax (Formulae)

- ▷ **Definition 3.10** term s: $\mathbf{A} \in wff_\iota(\Sigma_\iota)$ (denote individuals: type ι)
  - ▷ $\mathcal{V}_\iota \subseteq wff_\iota(\Sigma_\iota)$,
  - ▷ if $f \in \Sigma^f_k$ and $\mathbf{A}^i \in wff_\iota(\Sigma_\iota)$ for $i \leq k$, then $f(\mathbf{A}^1, \ldots, \mathbf{A}^k) \in wff_\iota(\Sigma_\iota)$.
- ▷ **Definition 3.11** proposition s: $\mathbf{A} \in wff_o(\Sigma)$ (denote truth values: type o)
  - ▷ if $p \in \Sigma^p_k$ and $\mathbf{A}^i \in wff_\iota(\Sigma_\iota)$ for $i \leq k$, then $p(\mathbf{A}^1, \ldots, \mathbf{A}^k) \in wff_o(\Sigma)$,
  - ▷ if $\mathbf{A}, \mathbf{B} \in wff_o(\Sigma)$, then $T, \mathbf{A} \wedge \mathbf{B}, \neg\mathbf{A}, \forall X.\mathbf{A} \in wff_o(\Sigma)$.

▷ **Definition 3.12** We define the connectives $F, \vee, \Rightarrow, \Leftrightarrow$ via the abbreviations $\mathbf{A} \vee \mathbf{B} := \neg(\neg\mathbf{A} \wedge \neg\mathbf{B})$, $\mathbf{A} \Rightarrow \mathbf{B} := \neg\mathbf{A} \vee \mathbf{B}$, $(\mathbf{A} \Leftrightarrow \mathbf{B}) := (\mathbf{A} \Rightarrow \mathbf{B}) \wedge (\mathbf{B} \Rightarrow \mathbf{A})$, and $F := \neg T$. We will use them like the primary connectives $\wedge$ and $\neg$

▷ **Definition 3.13** We use $\exists X.\mathbf{A}$ as an abbreviation for $\neg(\forall X.\neg\mathbf{A})$.(existential quantifier)

▷ **Definition 3.14** Call formulae without connectives or quantifiers atomic else complex.

SOME RIGHTS RESERVED ©: Michael Kohlhase

Note: that we only need e.g. conjunction, negation, and universal quantification, all other logical constants can be defined from them (as we will see when we have fixed their interpretations).

The introduction of quantifiers to first-order logic brings a new phenomenon: variables that are under the scope of a quantifiers will behave very differently from the ones that are not. Therefore we build up a vocabulary that distinguishes the two.

Free and Bound Variables

▷ **Definition 3.15** We call an occurrence of a variable X bound in a formula $\mathbf{A}$, iff it occurs in a sub-formula $\forall X.\mathbf{B}$ of $\mathbf{A}$. We call a variable occurrence free otherwise.

For a formula $\mathbf{A}$, we will use $\mathbf{BVar}(\mathbf{A})$ (and $\mathbf{free}(\mathbf{A})$) for the set of bound (free) variables of $\mathbf{A}$, i.e. variables that have a free/bound occurrence in $\mathbf{A}$.

▷ **Definition 3.16** We define the set $\mathbf{free}(\mathbf{A})$ of free variable s of a formula $\mathbf{A}$ inductively:

$$\begin{aligned}
&\mathbf{free}(X) := \{X\}\\
&\mathbf{free}(f(\mathbf{A}_1, \ldots, \mathbf{A}_n)) := \textstyle\bigcup_{1 \leq i \leq n} \mathbf{free}(\mathbf{A}_i)\\
&\mathbf{free}(p(\mathbf{A}_1, \ldots, \mathbf{A}_n)) := \textstyle\bigcup_{1 \leq i \leq n} \mathbf{free}(\mathbf{A}_i)\\
&\mathbf{free}(\neg\mathbf{A}) := \mathbf{free}(\mathbf{A})\\
&\mathbf{free}(\mathbf{A} \wedge \mathbf{B}) := \mathbf{free}(\mathbf{A}) \cup \mathbf{free}(\mathbf{B})\\
&\mathbf{free}(\forall X.\mathbf{A}) := \mathbf{free}(\mathbf{A}) \backslash \{X\}
\end{aligned}$$

▷ **Definition 3.17** We call a formula $\mathbf{A}$ closed or ground, iff $\mathbf{free}(\mathbf{A}) = \emptyset$. We call a closed proposition a sentence, and denote the set of all ground terms with $cwff_\iota(\Sigma_\iota)$ and the set of sentences with $cwff_o(\Sigma_\iota)$.

SOME RIGHTS RESERVED ©: Michael Kohlhase

We will be mainly interested in (sets of) sentences – i.e. closed propositions – as the representations of meaningful statements about individuals. Indeed, we will see below that free variables do not gives us expressivity, since they behave like constants and could be replaced by them in all situations, except the recursive definition of quantified formulae. Indeed in all situations where variables occur freely, they have the character of meta-variables, i.e. syntactic placeholders that can be instantiated with terms when needed in an inference calculus.

The semantics of first-order logic is a Tarski-style set-theoretic semantics where the atomic syntactic entities are interpreted by mapping them into a well-understood structure, a first-order universe that is just an arbitrary set.

Semantics of PL^1 (Models)

- We fix the Universe $\mathcal{D}_o = \{\mathsf{T}, \mathsf{F}\}$ of truth value s.
- We assume an arbitrary universe $\mathcal{D}_\iota \neq \emptyset$ of individuals (this choice is a parameter to the semantics)
- **Definition 3.18** An interpretation $\mathcal{I}$ assigns values to constants, e.g.
  - $\mathcal{I}(\neg)\colon \mathcal{D}_o \to \mathcal{D}_o$ with $\mathsf{T} \mapsto \mathsf{F}$, $\mathsf{F} \mapsto \mathsf{T}$, and $\mathcal{I}(\wedge) = \ldots$ (as in PL^0)
  - $\mathcal{I}\colon \Sigma_k^f \to \mathcal{F}({\mathcal{D}_\iota}^k; \mathcal{D}_\iota)$ (interpret function symbols as arbitrary functions)
  - $\mathcal{I}\colon \Sigma_k^p \to \mathcal{P}({\mathcal{D}_\iota}^k)$ (interpret predicates as arbitrary relations)
- **Definition 3.19** A variable assignment $\varphi\colon \mathcal{V}_\iota \to \mathcal{D}_\iota$ maps variables into the universe.
- A first-order Model $\mathcal{M} = \langle \mathcal{D}_\iota, \mathcal{I} \rangle$ consists of a universe $\mathcal{D}_\iota$ and an interpretation$\mathcal{I}$.

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

We do not have to make the universe of truth values part of the model, since it is always the same; we determine the model by choosing a universe and an interpretation function.

Given a first-order model, we can define the evaluation function as a homomorphism over the construction of formulae.

Semantics of PL^1 (Evaluation)

- Given a model $\langle \mathcal{D}, \mathcal{I} \rangle$, the value function $\mathcal{I}_\varphi$ is recursively defined:(two parts: terms & propositions)
  - $\mathcal{I}_\varphi\colon wff_\iota(\Sigma_\iota) \to \mathcal{D}_\iota$ assigns values to terms.
    - $\mathcal{I}_\varphi(X) := \varphi(X)$ and
    - $\mathcal{I}_\varphi(f(\mathbf{A}_1, \ldots, \mathbf{A}_k)) := \mathcal{I}(f)(\mathcal{I}_\varphi(\mathbf{A}_1), \ldots, \mathcal{I}_\varphi(\mathbf{A}_k))$
  - $\mathcal{I}_\varphi\colon wff_o(\Sigma) \to \mathcal{D}_o$ assigns values to formulae:
    - $\mathcal{I}_\varphi(T) = \mathcal{I}(T) = \mathsf{T}$, $\mathcal{I}_\varphi(\neg\mathbf{A}) = \mathcal{I}(\neg)(\mathcal{I}_\varphi(\mathbf{A}))$ $\mathcal{I}_\varphi(\mathbf{A} \wedge \mathbf{B}) = \mathcal{I}(\wedge)(\mathcal{I}_\varphi(\mathbf{A}), \mathcal{I}_\varphi(\mathbf{B}))$ (just as in PL^0)
    - $\mathcal{I}_\varphi(p(\mathbf{A}^1, \ldots, \mathbf{A}^k)) := \mathsf{T}$, iff $\langle \mathcal{I}_\varphi(\mathbf{A}^1), \ldots, \mathcal{I}_\varphi(\mathbf{A}^k) \rangle \in \mathcal{I}(p)$
    - $\mathcal{I}_\varphi(\forall X.\mathbf{A}) := \mathsf{T}$, iff $\mathcal{I}_{\varphi,[a/X]}(\mathbf{A}) = \mathsf{T}$ for all $a \in \mathcal{D}_\iota$.

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

The only new (and interesting) case in this definition is the quantifier case, there we define the value of a quantified formula by the value of its scope – *but with an extended variable assignment*. Note that by passing to the scope $\mathbf{A}$ of $\forall x.\mathbf{A}$, the occurrences of the variable x in $\mathbf{A}$ that were bound in $\forall x.\mathbf{A}$ become free and are amenable to evaluation by the variable assignment $\psi := \varphi, [a/X]$. Note that as an extension of φ, the assignment ψ supplies exactly the right value for x in $\mathbf{A}$. This variability of the variable assignment in the definition value function justifies the somewhat complex setup of first-order evaluation, where we have the (static) interpretation function for the symbols from the signature and the (dynamic) variable assignment for the variables.

Note furthermore, that the value $\mathcal{I}_\varphi(\exists x.\mathbf{A})$ of $\exists x.\mathbf{A}$, which we have defined to be $\neg(\forall x.\neg\mathbf{A})$ is true, iff it is not the case that $\mathcal{I}_\varphi(\forall x.\neg\mathbf{A}) = \mathcal{I}_\psi(\neg\mathbf{A}) = \mathsf{F}$ for all $\mathsf{a} \in \mathcal{D}_\iota$ and $\psi := \varphi, [a/X]$. This is the case, iff $\mathcal{I}_\psi(\mathbf{A}) = \mathsf{T}$ for some $\mathsf{a} \in \mathcal{D}_\iota$. So our definition of the existential quantifier yields the appropriate semantics.

3.2.2 First-Order Substitutions

We will now turn our attention to substitutions, special formula-to-formula mappings that operationalize the intuition that (individual) variables stand for arbitrary terms.

Substitutions on Terms

- ▷ Intuition: If $\mathbf{B}$ is a term and X is a variable, then we denote the result of systematically replacing all occurrences of X in a term $\mathbf{A}$ by $\mathbf{B}$ with $[\mathbf{B}/X](\mathbf{A})$.
- ▷ Problem: What about $[Z/Y], [Y/X](X)$, is that Y or Z?
- ▷ Folklore: $[Z/Y], [Y/X](X) = Y$, but $[Z/Y]([Y/X](X)) = Z$ of course. (Parallel application)
- ▷ **Definition 3.20** We call $\sigma\colon \mathit{wff}_\iota(\Sigma_\iota) \to \mathit{wff}_\iota(\Sigma_\iota)$ a substitution, iff $\sigma(f(\mathbf{A}_1, \ldots, \mathbf{A}_n)) = f(\sigma(\mathbf{A}_1), \ldots, \sigma(\mathbf{A}_n))$ and the support $\mathbf{supp}(\sigma) := \{X \mid \sigma(X) \neq X\}$ of σ is finite.
- ▷ **Observation 3.21** *Note that a substitution σ is determined by its values on variables alone, thus we can write σ as $\sigma|_{\mathcal{V}_\iota} = \{[\sigma(X)/X] \mid X \in \mathbf{supp}(\sigma)\}$.*
- ▷ **Notation 3.22** We denote the substitution σ with $\mathbf{supp}(\sigma) = \{x^i \mid 1 \leq i \leq n\}$ and $\sigma(x^i) = \mathbf{A}_i$ by $[\mathbf{A}_1/x^1], \ldots, [\mathbf{A}_n/x^n]$.
- ▷ **Example 3.23** $[a/x], [f(b)/y], [a/z]$ instantiates $g(x, y, h(z))$ to $g(a, f(b), h(a))$.
- ▷ **Definition 3.24** We call $\mathbf{intro}(\sigma) := \bigcup_{X \in \mathbf{supp}(\sigma)} \mathbf{free}(\sigma(X))$ the set of variables introduced by σ.

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

The extension of a substitution is an important operation, which you will run into from time to time. Given a substitution σ, a variable x, and an expression $\mathbf{A}$, $\sigma, [\mathbf{A}/x]$ extends σ with a new value for x. The intuition is that the values right of the comma overwrite the pairs in the substitution on the left, which already has a value for x, even though the representation of σ may not show it.

Substitution Extension

- ▷ **Notation 3.25 (Substitution Extension)** Let σ be a substitution, then we denote with $\sigma, [\mathbf{A}/X]$ the function $\{\langle Y, \mathbf{A}\rangle \in \sigma \mid Y \neq X\} \cup \{\langle X, \mathbf{A}\rangle\}$. ($\sigma, [\mathbf{A}/X]$ coincides with σ of X, and gives the result $\mathbf{A}$ there.)
- ▷ Note: If σ is a substitution, then $\sigma, [\mathbf{A}/X]$ is also a substitution.
- ▷ **Definition 3.26** If σ is a substitution, then we call $\sigma, [\mathbf{A}/X]$ the extension of σ by $[\mathbf{A}/X]$.
- ▷ We also need the dual operation: removing a variable from the support

▷ **Definition 3.27** We can discharge a variable X from a substitution σ by $\sigma_{-X} := \sigma, [X/X]$.

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

Note that the use of the comma notation for substitutions defined in Notation 3.22 is consistent with substitution extension. We can view a substitution $[a/x], [f(b)/y]$ as the extension of the empty substitution (the identity function on variables) by $[f(b)/y]$ and then by $[a/x]$. Note furthermore, that substitution extension is not commutative in general.

For first-order substitutions we need to extend the substitutions defined on terms to act on propositions. This is technically more involved, since we have to take care of bound variables.

Substitutions on Propositions

▷ Problem: We want to extend substitutions to propositions, in particular to quantified formulae: What is $\sigma(\forall X.\mathbf{A})$?

▷ Idea: σ should not instantiate bound variables. ($[\mathbf{A}/X](\forall X.\mathbf{B}) = \forall \mathbf{A}.\mathbf{B}'$ ill-formed)

▷ **Definition 3.28** $\sigma(\forall X.\mathbf{A}) := (\forall X.\sigma_{-X}(\mathbf{A}))$.

▷ Problem: This can lead to variable capture: $[f(X)/Y](\forall X.p(X,Y))$ would evaluate to $\forall X.p(X, f(X))$, where the second occurrence of X is bound after instantiation, whereas it was free before.

▷ **Definition 3.29** Let $\mathbf{B} \in wff_\iota(\Sigma_\iota)$ and $\mathbf{A} \in wff_o(\Sigma)$, then we call $\mathbf{B}$ substitutable for X in $\mathbf{A}$, iff $\mathbf{A}$ has no occurrence of X in a subterm $\forall Y.\mathbf{C}$ with $Y \in \mathbf{free}(\mathbf{B})$.

▷ Solution: Forbid substitution $[\mathbf{B}/X]\mathbf{A}$, when $\mathbf{B}$ is not substitutable for X in $\mathbf{A}$.

▷ Better Solution: Rename away the bound variable X in $\forall X.p(X,Y)$ before applying the substitution. (see alphabetic renaming later.)

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

Here we come to a conceptual problem of most introductions to first-order logic: they directly define substitutions to be capture-avoiding by stipulating that bound variables are renamed in the to ensure subsitutability. But at this time, we have not even defined alphabetic renaming yet, and cannot formally do that without having a notion of substitution. So we will refrain from introducing capture-avoiding substitutions until we have done our homework.

We now introduce a central tool for reasoning about the semantics of substitutions: the "substitution-value Lemma", which relates the process of instantiation to (semantic) evaluation. This result will be the motor of all soundness proofs on axioms and inference rules acting on variables via substitutions. In fact, any logic with variables and substitutions will have (to have) some form of a substitution-value Lemma to get the meta-theory going, so it is usually the first target in any development of such a logic.

We establish the substitution-value Lemma for first-order logic in two steps, first on terms, where it is very simple, and then on propositions, where we have to take special care of substitutability.

Substitution Value Lemma for Terms

▷ **Lemma 3.30** *Let* $\mathbf{A}$ *and* $\mathbf{B}$ *be terms, then* $\mathcal{I}_\varphi([\mathbf{B}/X]\mathbf{A}) = \mathcal{I}_\psi(\mathbf{A})$, *where* $\psi = \varphi, [\mathcal{I}_\varphi(\mathbf{B})/X]$.

▷ Proof: by induction on the depth of $\mathbf{A}$:

P.1.1 depth=0:

P.1.1.1 Then $\mathbf{A}$ is a variable (say Y), or constant, so we have three cases

P.1.1.1.1 $\mathbf{A} = Y = X$: then $\mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{A})) = \mathcal{I}_\varphi([\mathbf{B}/X](X)) = \mathcal{I}_\varphi(\mathbf{B}) = \psi(X) = \mathcal{I}_\psi(X) = \mathcal{I}_\psi(\mathbf{A})$.

P.1.1.1.2 $\mathbf{A} = Y \neq X$: then $\mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{A})) = \mathcal{I}_\varphi([\mathbf{B}/X](Y)) = \mathcal{I}_\varphi(Y) = \varphi(Y) = \psi(Y) = \mathcal{I}_\psi(Y) = \mathcal{I}_\psi(\mathbf{A})$.

P.1.1.1.3 $\mathbf{A}$ is a constant: analogous to the preceding case ($Y \neq X$)

P.1.1.2 This completes the base case (depth = 0). □

P.1.2 depth> 0: then $\mathbf{A} = f(\mathbf{A}_1, \ldots, \mathbf{A}_n)$ and we have

$$\begin{aligned}\mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{A})) &= \mathcal{I}(f)(\mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{A}_1)), \ldots, \mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{A}_n))) \\ &= \mathcal{I}(f)(\mathcal{I}_\psi(\mathbf{A}_1), \ldots, \mathcal{I}_\psi(\mathbf{A}_n)) \\ &= \mathcal{I}_\psi(\mathbf{A}).\end{aligned}$$

by inductive hypothesis

P.1.2.2 This completes the inductive case, and we have proven the assertion □

□

We now come to the case of propositions. Note that we have the additional assumption of substitutability here.

Substitution Value Lemma for Propositions

▷ **Lemma 3.31** *Let* $\mathbf{B} \in wf\!f_\iota(\Sigma_\iota)$ *be substitutable for* X *in* $\mathbf{A} \in wf\!f_o(\Sigma)$, *then* $\mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{A})) = \mathcal{I}_\psi(\mathbf{A})$, *where* $\psi = \varphi, [\mathcal{I}_\varphi(\mathbf{B})/X]$.

▷ Proof: by induction on the number n of connectives and quantifiers in $\mathbf{A}$

P.1.1 $n = 0$: then $\mathbf{A}$ is an atomic proposition, and we can argue like in the inductive case of the substitution value lemma for terms.

P.1.2 $n > 0$ and $\mathbf{A} = \neg\mathbf{B}$ or $\mathbf{A} = \mathbf{C} \circ \mathbf{D}$: Here we argue like in the inductive case of the term lemma as well.

P.1.3 $n > 0$ and $\mathbf{A} = \forall X.\mathbf{C}$: then $\mathcal{I}_\psi(\mathbf{A}) = \mathcal{I}_\psi(\forall X.\mathbf{C}) = \mathsf{T}$, iff $\mathcal{I}_{\psi,[a/X]}(\mathbf{C}) = \mathcal{I}_{\varphi,[a/X]}(\mathbf{C}) = \mathsf{T}$, for all $a \in \mathcal{D}_\iota$, which is the case, iff $\mathcal{I}_\varphi(\forall X.\mathbf{C}) = \mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{A})) = \mathsf{T}$.

P.1.4 $n > 0$ and $\mathbf{A} = \forall Y.\mathbf{C}$ where $X \neq Y$: then $\mathcal{I}_\psi(\mathbf{A}) = \mathcal{I}_\psi(\forall Y.\mathbf{C}) = \mathsf{T}$, iff $\mathcal{I}_{\psi,[a/Y]}(\mathbf{C}) = \mathcal{I}_{\varphi,[a/Y]}([\mathbf{B}/X](\mathbf{C})) = \mathsf{T}$, by inductive hypothesis. So $\mathcal{I}_\psi(\mathbf{A}) = \mathcal{I}_\varphi(\forall Y.[\mathbf{B}/X](\mathbf{C})) = \mathcal{I}_\varphi([\mathbf{B}/X](\forall Y.\mathbf{C})) = \mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{A}))$ □

To understand the proof full, you should look out where the substitutability is actually used.

Armed with the substitution value lemma, we can now define alphabetic renaming and show it to be sound with respect to the semantics we defined above. And this soundness result will justify the definition of capture-avoiding substitution we will use in the rest of the course.

3.2.3 Alpha-Renaming for First-Order Logic

Armed with the substitution value lemma we can now prove one of the main representational facts for first-order logic: the names of bound variables do not matter; they can be renamed at liberty without changing the meaning of a formula.

Alphabetic Renaming

▷ **Lemma 3.32** *Bound variables can be renamed: If Y is substitutable for X in $\mathbf{A}$, then $\mathcal{I}_\varphi(\forall X.\mathbf{A}) = \mathcal{I}_\varphi(\forall Y.[Y/X](\mathbf{A}))$*

▷ Proof: by the definitions:

P.1 $\mathcal{I}_\varphi(\forall X.\mathbf{A}) = \mathsf{T}$, iff

P.2 $\mathcal{I}_{\varphi,[a/X]}(\mathbf{A}) = \mathsf{T}$ for all $a \in \mathcal{D}_\iota$, iff

P.3 $\mathcal{I}_{\varphi,[a/Y]}([Y/X](\mathbf{A})) = \mathsf{T}$ for all $a \in \mathcal{D}_\iota$, iff (by substitution value lemma)

P.4 $\mathcal{I}_\varphi(\forall Y.[Y/X](\mathbf{A})) = \mathsf{T}$. □

▷ **Definition 3.33** We call two formulae $\mathbf{A}$ and $\mathbf{B}$ alphabetical variants (or α-equal; write $\mathbf{A} =_\alpha \mathbf{B}$), iff $\mathbf{A} = \forall X.\mathbf{C}$ and $\mathbf{B} = \forall Y.[Y/X](\mathbf{C})$ for some variables X and Y.

We have seen that naive substitutions can lead to variable capture. As a consequence, we always have to presuppose that all instantiations respect a substitutability condition, which is quite tedious. We will now come up with an improved definition of substitution application for first-order logic that does not have this problem.

Avoiding Variable Capture by Built-in α-renaming

▷ Idea: Given alphabetic renaming, we will consider alphabetical variants as identical

▷ So: Bound variable names in formulae are just a representational device (we rename bound variables wherever necessary)

▷ Formally: Take $cwff_o(\Sigma_\iota)$ (new) to be the quotient set of $cwff_o(\Sigma_\iota)$ (old) modulo $=_\alpha$. (formulae as syntactic representatives of equivalence classes)

▷ **Definition 3.34 (Capture-Avoiding Substitution Application)** Let σ be a substitution, $\mathbf{A}$ a formula, and $\mathbf{A}'$ an alphabetical variant of $\mathbf{A}$, such that $\mathbf{intro}(\sigma) \cap \mathbf{BVar}(\mathbf{A}) = \emptyset$. Then $[\mathbf{A}]_{=_\alpha} = [\mathbf{A}']_{=_\alpha}$ and we can define $\sigma([\mathbf{A}]_{=_\alpha}) := [\sigma(\mathbf{A}')]_{=_\alpha}$.

▷ **Notation 3.35** After we have understood the quotient construction, we will neglect making it explicit and write formulae and substitutions with the understanding that they act on quotients.

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

3.3 Abstract Consistency and Model Existence

We will now come to an important tool in the theoretical study of reasoning calculi: the "abstract consistency"/"model existence" method. This method for analyzing calculi was developed by Jaako Hintikka, Raymond Smullyann, and Peter Andrews in 1950-1970 as an encapsulation of similar constructions that were used in completeness arguments in the decades before.[27] EdN:27

The basic intuition for this method is the following: typically, a logical system $\mathcal{S} = \langle \mathcal{L}, \mathcal{K}, \models \rangle$ has multiple calculi, human-oriented ones like the natural deduction calculi and machine-oriented ones like the automated theorem proving calculi. All of these need to be analyzed for completeness (as a basic quality assurance measure).

A completeness proof for a calculus $\mathcal{C}$ for $\mathcal{S}$ typically comes in two parts: one analyzes $\mathcal{C}$-consistency (sets that cannot be refuted in $\mathcal{C}$), and the other construct $\mathcal{K}$-models for $\mathcal{C}$-consistent sets.

In this situtation the "abstract consistency"/"model existence" method encapsulates the model construction process into a meta-theorem: the "model existence" theorem. This provides a set of syntactic ("abstract consistency") conditions for calculi that are sufficient to construct models.

With the model existence theorem it suffices to show that $\mathcal{C}$-consistency is an abstract consistency property (a purely syntactic task that can be done by a $\mathcal{C}$-proof transformation argument) to obtain a completeness result for $\mathcal{C}$.

Model Existence (Overview)

▷ Definition: Abstract consistency

▷ Definition: Hintikka set (maximally abstract consistent)

▷ Theorem: Hintikka sets are satisfiable

▷ Theorem: If Φ is abstract consistent, then Φ can be extended to a Hintikka set.

▷ Corollary: If Φ is abstract consistent, then Φ is satisfiable

▷ Application: Let $\mathcal{C}$ be a calculus, if Φ is $\mathcal{C}$-consistent, then Φ is abstract consistent.

▷ Corollary: $\mathcal{C}$ is complete.

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

The proof of the model existence theorem goes via the notion of a Hintikka set, a set of formulae with very strong syntactic closure properties, which allow to read off models. Jaako Hintikka's original idea for completeness proofs was that for every complete calculus $\mathcal{C}$ and every $\mathcal{C}$-consistent set one can induce a Hintikka set, from which a model can be constructed. This can be considered

[27] EdNote: cite the original papers!

as a first model existence theorem. However, the process of obtaining a Hintikka set for a set $\mathcal{C}$-consistent set Φ of sentences usually involves complicated calculus-dependent constructions.

In this situation, Raymond Smullyann was able to formulate the sufficient conditions for the existence of Hintikka sets in the form of "abstract consistency properties" by isolating the calculus-independent parts of the Hintikka set construction. His technique allows to reformulate Hintikka sets as maximal elements of abstract consistency classes and interpret the Hintikka set construction as a maximizing limit process.

To carry out the "model-existence"/"abstract consistency" method, we will first have to look at the notion of consistency.

Consistency and refutability are very important notions when studying the completeness for calculi; they form syntactic counterparts of satisfiability.

Consistency

▷ Let $\mathcal{C}$ be a calculus

▷ **Definition 3.36** Φ is called $\mathcal{C}$-refutable, if there is a formula $\mathbf{B}$, such that $\Phi \vdash_{\mathcal{C}} \mathbf{B}$ and $\Phi \vdash_{\mathcal{C}} \neg\mathbf{B}$.

▷ **Definition 3.37** We call a pair $\mathbf{A}$ and $\neg\mathbf{A}$ a contradiction.

▷ So a set Φ is $\mathcal{C}$-refutable, if $\mathcal{C}$ can derive a contradiction from it.

▷ **Definition 3.38** Φ is called $\mathcal{C}$-consistent, iff there is a formula $\mathbf{B}$, that is not derivable from Φ in $\mathcal{C}$.

▷ **Definition 3.39** We call a calculus $\mathcal{C}$ reasonable, iff implication elimination and conjunction introduction are admissible in $\mathcal{C}$ and $\mathbf{A} \wedge \neg\mathbf{A} \Rightarrow \mathbf{B}$ is a $\mathcal{C}$-theorem.

▷ **Theorem 3.40** *$\mathcal{C}$-inconsistency and $\mathcal{C}$-refutability coincide for reasonable calculi*

SOME RIGHTS RESERVED ©: Michael Kohlhase 108 JACOBS UNIVERSITY

It is very important to distinguish the syntactic $\mathcal{C}$-refutability and $\mathcal{C}$-consistency from satisfiability, which is a property of formulae that is at the heart of semantics. Note that the former specify the calculus (a syntactic device) while the latter does not. In fact we should actually say $\mathcal{S}$-satisfiability, where $\mathcal{S} = \langle \mathcal{L}, \mathcal{K}, \models \rangle$ is the current logical system.

Even the word "contradiction" has a syntactical flavor to it, it translates to "saying against each other" from its latin root.

The notion of an "abstract consistency class" provides the a calculus-independent notion of "consistency": A set Φ of sentences is considered "consistent in an abstract sense", iff it is a member of an abstract consistency class ∇.

Abstract Consistency

▷ **Definition 3.41** Let ∇ be a family of sets. We call ∇ closed under subsets, iff for each $\Phi \in \nabla$, all subsets $\Psi \subseteq \Phi$ are elements of ∇.

▷ **Notation 3.42** We will use $\Phi * \mathbf{A}$ for $\Phi \cup \{\mathbf{A}\}$.

▷ **Definition 3.43** A family $\nabla \subseteq wff_o(\Sigma)$ of sets of formulae is called a (first-order) abstract consistency class, iff it is closed under subsets, and for each

$\Phi \in \nabla$

∇_c) $\mathbf{A} \notin \Phi$ or $\neg\mathbf{A} \notin \Phi$ for atomic $\mathbf{A} \in \mathit{wff}_o(\Sigma)$.

$\nabla_\neg$) $\neg\neg\mathbf{A} \in \Phi$ implies $\Phi * \mathbf{A} \in \nabla$

$\nabla_\wedge$) $(\mathbf{A} \wedge \mathbf{B}) \in \Phi$ implies $(\Phi \cup \{\mathbf{A}, \mathbf{B}\}) \in \nabla$

$\nabla_\vee$) $\neg(\mathbf{A} \wedge \mathbf{B}) \in \Phi$ implies $\Phi * \neg\mathbf{A} \in \nabla$ or $\Phi * \neg\mathbf{B} \in \nabla$

$\nabla_\forall$) If $(\forall X.\mathbf{A}) \in \Phi$, then $\Phi * [\mathbf{B}/X](\mathbf{A}) \in \nabla$ for each closed term $\mathbf{B}$.

$\nabla_\exists$) If $\neg(\forall X.\mathbf{A}) \in \Phi$ and c is an individual constant that does not occur in Φ, then $\Phi * \neg[c/X](\mathbf{A}) \in \nabla$

SOME RIGHTS RESERVED ©: Michael Kohlhase 109 JACOBS UNIVERSITY

The conditions are very natural: Take for instance ∇_c, it would be foolish to call a set Φ of sentences "consistent under a complete calculus", if it contains an elementary contradiction. The next condition $\nabla_\neg$ says that if a set Φ that contains a sentence $\neg\neg\mathbf{A}$ is "consistent", then we should be able to extend it by $\mathbf{A}$ without losing this property; in other words, a complete calculus should be able to recognize $\mathbf{A}$ and $\neg\neg\mathbf{A}$ to be equivalent.

We will carry out the proof here, since it gives us practice in dealing with the abstract consistency properties.

Actually we are after abstract consistency classes that have an even stronger property than just being closed under subsets. This will allow us to carry out a limit construction in the Hintikka set extension argument later.

Compact Collections

▷ **Definition 3.44** We call a collection ∇ of sets compact, iff for any set Φ we have
$\Phi \in \nabla$, iff $\Psi \in \nabla$ for every finite subset Ψ of Φ.

▷ **Lemma 3.45** *If ∇ is compact, then ∇ is closed under subsets.*

▷ Proof:

P.1 Suppose $S \subseteq T$ and $T \in \nabla$.

P.2 Every finite subset A of S is a finite subset of T.

P.3 As ∇ is compact, we know that $A \in \nabla$.

P.4 Thus $S \in \nabla$. □

SOME RIGHTS RESERVED ©: Michael Kohlhase 110 JACOBS UNIVERSITY

The property of being closed under subsets is a "downwards-oriented" property: We go from large sets to small sets, compactness (the interesting direction anyways) is also an "upwards-oriented" property. We can go from small (finite) sets to large (infinite) sets. The main application for the compactness condition will be to show that infinite sets of formulae are in a family ∇ by testing all their finite subsets (which is much simpler).

The main result here is that abstract consistency classes can be extended to compact ones. The proof is quite tedious, but relatively straightforward. It allows us to assume that all abstract consistency classes are compact in the first place (otherwise we pass to the compact extension).

Compact Abstract Consistency Classes

▷ **Lemma 3.46** *Any first-order abstract consistency class can be extended to a compact one.*

▷ Proof:

P.1 We choose $\nabla' := \{\Phi \subseteq cwff_o(\Sigma_\iota) \,|\, \text{every finite subset of } \Phi \text{is in } \nabla\}$.

P.2 Now suppose that $\Phi \in \nabla$. ∇ is closed under subsets, so every finite subset of Φ is in ∇ and thus $\Phi \in \nabla'$. Hence $\nabla \subseteq \nabla'$.

P.3 Let us now show that each ∇' is compact.

P.3.1 Suppose $\Phi \in \nabla'$ and Ψ is an arbitrary finite subset of Φ.

P.3.2 By definition of ∇' all finite subsets of Φ are in ∇ and therefore $\Psi \in \nabla'$.

P.3.3 Thus all finite subsets of Φ are in ∇' whenever Φ is in ∇'.

P.3.4 On the other hand, suppose all finite subsets of Φ are in ∇'.

P.3.5 Then by the definition of ∇' the finite subsets of Φ are also in ∇, so $\Phi \in \nabla'$. Thus ∇' is compact.

P.4 Note that ∇' is closed under subsets by the Lemma above.

P.5 Next we show that if ∇ satisfies ∇_*, then ∇' satisfies ∇_*.

P.5.1 To show ∇_c, let $\Phi \in \nabla'$ and suppose there is an atom $\mathbf{A}$, such that $\{\mathbf{A}, \neg\mathbf{A}\} \subseteq \Phi$. Then $\{\mathbf{A}, \neg\mathbf{A}\} \in \nabla$ contradicting ∇_c.

P.5.2 To show $\nabla_\neg$, let $\Phi \in \nabla'$ and $\neg\neg\mathbf{A} \in \Phi$, then $\Phi * \mathbf{A} \in \nabla'$.

P.5.2.1 Let Ψ be any finite subset of $\Phi * \mathbf{A}$, and $\Theta := (\Psi \backslash \{\mathbf{A}\}) * \neg\neg\mathbf{A}$.

P.5.2.2 Θ is a finite subset of Φ, so $\Theta \in \nabla$.

P.5.2.3 Since ∇ is an abstract consistency class and $\neg\neg\mathbf{A} \in \Theta$, we get $\Theta * \mathbf{A} \in \nabla$ by $\nabla_\neg$.

P.5.2.4 We know that $\Psi \subseteq \Theta * \mathbf{A}$ and ∇ is closed under subsets, so $\Psi \in \nabla$.

P.5.2.5 Thus every finite subset Ψ of $\Phi * \mathbf{A}$ is in ∇ and therefore by definition $\Phi * \mathbf{A} \in \nabla'$.

P.5.3 the other cases are analogous to $\nabla_\neg$. □

SOME RIGHTS RESERVED ©: Michael Kohlhase 111 JACOBS UNIVERSITY

Hintikka sets are sets of sentences with very strong analytic closure conditions. These are motivated as maximally consistent sets i.e. sets that already contain everything that can be consistently added to them.

∇-Hintikka Set

▷ **Definition 3.47** Let ∇ be an abstract consistency class, then we call a set $\mathcal{H} \in \nabla$ a ∇-Hintikka Set, iff $\mathcal{H}$ is maximal in ∇, i.e. for all $\mathbf{A}$ with $\mathcal{H} * \mathbf{A} \in \nabla$ we already have $\mathbf{A} \in \mathcal{H}$.

▷ **Theorem 3.48 (Hintikka Properties)** *Let ∇ be an abstract consistency class and $\mathcal{H}$ be a ∇-Hintikka set, then*

$\mathcal{H}_c$) *For all $\mathbf{A} \in wff_o(\Sigma)$ we have $\mathbf{A} \notin \mathcal{H}$ or $\neg\mathbf{A} \notin \mathcal{H}$.*

$\mathcal{H}_\neg$) *If* $\neg\neg\mathbf{A} \in \mathcal{H}$ *then* $\mathbf{A} \in \mathcal{H}$.

$\mathcal{H}_\wedge$) *If* $(\mathbf{A} \wedge \mathbf{B}) \in \mathcal{H}$ *then* $\mathbf{A}, \mathbf{B} \in \mathcal{H}$.

$\mathcal{H}_\vee$) *If* $\neg(\mathbf{A} \wedge \mathbf{B}) \in \mathcal{H}$ *then* $\neg\mathbf{A} \in \mathcal{H}$ *or* $\neg\mathbf{B} \in \mathcal{H}$.

$\mathcal{H}_\forall$) *If* $(\forall X.\mathbf{A}) \in \mathcal{H}$, *then* $[\mathbf{B}/X](\mathbf{A}) \in \mathcal{H}$ *for each closed term* $\mathbf{B}$.

$\mathcal{H}_\exists$) *If* $\neg(\forall X.\mathbf{A}) \in \mathcal{H}$ *then* $\neg[\mathbf{B}/X](\mathbf{A}) \in \mathcal{H}$ *for some term closed term* $\mathbf{B}$.

Proof:

▷ **P.1** We prove the properties in turn

$\mathcal{H}_c$ goes by induction on the structure of $\mathbf{A}$

P.2.1 $\mathbf{A}$ atomic: Then $\mathbf{A} \notin \mathcal{H}$ or $\neg\mathbf{A} \notin \mathcal{H}$ by ∇_c.

P.2.2 $\mathbf{A} = \neg\mathbf{B}$:

P.2.2.1 Let us assume that $\neg\mathbf{B} \in \mathcal{H}$ and $\neg\neg\mathbf{B} \in \mathcal{H}$,

P.2.2.2 then $\mathcal{H} * \mathbf{B} \in \nabla$ by $\nabla_\neg$, and therefore $\mathbf{B} \in \mathcal{H}$ by maximality.

P.2.2.3 So $\{\mathbf{B}, \neg\mathbf{B}\} \subseteq \mathcal{H}$, which contradicts the inductive hypothesis. □

P.2.3 $\mathbf{A} = \mathbf{B} \vee \mathbf{C}$: similar to the previous case

We prove $\mathcal{H}_\neg$ by maximality of $\mathcal{H}$ in ∇.

P.3.1 If $\neg\neg\mathbf{A} \in \mathcal{H}$, then $\mathcal{H} * \mathbf{A} \in \nabla$ by $\nabla_\neg$.

P.3.2 The maximality of $\mathcal{H}$ now gives us that $\mathbf{A} \in \mathcal{H}$.

The other $\mathcal{H}_*$ are similar □

SOME RIGHTS RESERVED ©: Michael Kohlhase 112 JACOBS UNIVERSITY

The following theorem is one of the main results in the "abstract consistency"/"model existence" method. For any abstract consistent set Φ it allows us to construct a Hintikka set $\mathcal{H}$ with $\Phi \in \mathcal{H}$.

P.4 Extension Theorem

▷ **Theorem 3.49** *If* ∇ *is an abstract consistency class and* $\Phi \in \nabla$ *finite, then there is a* ∇*-Hintikka set* $\mathcal{H}$ *with* $\Phi \subseteq \mathcal{H}$.

▷ Proof: Wlog. assume that ∇ compact (else use compact extension)

P.1 Choose an enumeration $\mathbf{A}^1, \mathbf{A}^2, \ldots$ of $\mathit{cwff}_o(\Sigma_\iota)$ and $c^1, c^2, \ldots$ of Σ_0^{sk}.

P.2 and construct a sequence of sets H^i with $H^0 := \Phi$ and

$$H^{n+1} := \begin{cases} H^n & \text{if } H^n * \mathbf{A}^n \notin \nabla \\ H^n \cup \{\mathbf{A}^n, \neg[c^n/X](\mathbf{B})\} & \text{if } H^n * \mathbf{A}^n \in \nabla \text{ and } \mathbf{A}^n = \neg(\forall X.\mathbf{B}) \\ H^n * \mathbf{A}^n & \text{else} \end{cases}$$

P.3 Note that all $H^i \in \nabla$, choose $\mathcal{H} := \bigcup_{i \in \mathbb{N}} H^i$

P.4 $\Psi \subseteq \mathcal{H}$ finite implies there is a $j \in \mathbb{N}$ such that $\Psi \subseteq H^j$,

P.5 so $\Psi \in \nabla$ as ∇ closed under subsets and $\mathcal{H} \in \nabla$ as ∇ is compact.

P.6 Let $\mathcal{H} * \mathbf{B} \in \nabla$, then there is a $j \in \mathbb{N}$ with $\mathbf{B} = \mathbf{A}^j$, so that $\mathbf{B} \in H^{j+1}$ and $H^{j+1} \subseteq \mathcal{H}$

P.7 Thus $\mathcal{H}$ is ∇-maximal □

SOME RIGHTS RESERVED ©: Michael Kohlhase 113 JACOBS UNIVERSITY

Note that the construction in the proof above is non-trivial in two respects. First, the limit construction for $\mathcal{H}$ is not executed in our original abstract consistency class ∇, but in a suitably extended one to make it compact — the original would not have contained $\mathcal{H}$ in general. Second, the set $\mathcal{H}$ is not unique for Φ, but depends on the choice of the enumeration of $cwff_o(\Sigma_\iota)$. If we pick a different enumeration, we will end up with a different $\mathcal{H}$. Say if $\mathbf{A}$ and $\neg\mathbf{A}$ are both ∇-consistent[28] with Φ, then depending on which one is first in the enumeration $\mathcal{H}$, will contain that one; with all the consequences for subsequent choices in the construction process. EdN:28

Valuation

▷ **Definition 3.50** A function $\nu\colon cwff_o(\Sigma_\iota) \to \mathcal{D}_o$ is called a (first-order) valuation, iff

- ▷ $\nu(\neg\mathbf{A}) = \mathsf{T}$, iff $\nu(\mathbf{A}) = \mathsf{F}$
- ▷ $\nu(\mathbf{A} \wedge \mathbf{B}) = \mathsf{T}$, iff $\nu(\mathbf{A}) = \mathsf{T}$ and $\nu(\mathbf{B}) = \mathsf{T}$
- ▷ $\nu(\forall X.\mathbf{A}) = \mathsf{T}$, iff $\nu([\mathbf{B}/X](\mathbf{A})) = \mathsf{T}$ for all closed terms $\mathbf{B}$.

▷ **Lemma 3.51** *If $\varphi\colon \mathcal{V}_\iota \to \mathcal{D}$ is a variable assignment, then $\mathcal{I}_\varphi\colon cwff_o(\Sigma_\iota) \to \mathcal{D}_o$ is a valuation.*

▷ Proof Sketch: Immediate from the definitions □

SOME RIGHTS RESERVED ©: Michael Kohlhase 114 JACOBS UNIVERSITY

Thus a valuation is a weaker notion of evaluation in first-order logic; the other direction is also true, even though the proof of this result is much more involved: The existence of a first-order valuation that makes a set of sentences true entails the existence of a model that satisfies it.[29] EdN:29

Valuation and Satisfiability

▷ **Lemma 3.52** *If $\nu\colon cwff_o(\Sigma_\iota) \to \mathcal{D}_o$ is a valuation and $\Phi \subseteq cwff_o(\Sigma_\iota)$ with $\nu(\Phi) = \{\mathsf{T}\}$, then Φ is satisfiable.*

▷ Proof: We construct a model for Φ.

P.1 Let $\mathcal{D}_\iota := cwff_\iota(\Sigma_\iota)$, and

- ▷ $\mathcal{I}(f)\colon {\mathcal{D}_\iota}^k \to \mathcal{D}_\iota; \langle \mathbf{A}_1, \ldots, \mathbf{A}_k \rangle \mapsto f(\mathbf{A}_1, \ldots, \mathbf{A}_k)$ for $f \in \Sigma^f$
- ▷ $\mathcal{I}(p)\colon {\mathcal{D}_\iota}^k \to \mathcal{D}_o; \langle \mathbf{A}_1, \ldots, \mathbf{A}_k \rangle \mapsto \nu(p(\mathbf{A}_1, \ldots, \mathbf{A}_n))$ for $p \in \Sigma^p$.

P.2 Then variable assignments into $\mathcal{D}_\iota$ are ground substitutions.

P.3 We show $\mathcal{I}_\varphi(\mathbf{A}) = \varphi(\mathbf{A})$ for $\mathbf{A} \in wff_\iota(\Sigma_\iota)$ by induction on $\mathbf{A}$

P.3.1 $\mathbf{A} = X$: then $\mathcal{I}_\varphi(\mathbf{A}) = \varphi(X)$ by definition.

P.3.2 $\mathbf{A} = f(\mathbf{A}_1, \ldots, \mathbf{A}_n)$: then $\mathcal{I}_\varphi(\mathbf{A}) = \mathcal{I}(f)(\mathcal{I}_\varphi(\mathbf{A}_1), \ldots, \mathcal{I}_\varphi(\mathbf{A}_n)) = \mathcal{I}(f)(\varphi(\mathbf{A}_1), \ldots, \varphi(\mathbf{A}_n)) = f(\varphi(\mathbf{A}_1), \ldots, \varphi(\mathbf{A}_n)) = \varphi(f(\mathbf{A}_1, \ldots, \mathbf{A}_n)) = \varphi(\mathbf{A})$

P.4 We show $\mathcal{I}_\varphi(\mathbf{A}) = \nu(\varphi(\mathbf{A}))$ for $\mathbf{A} \in wff_o(\Sigma)$ by induction on $\mathbf{A}$

[28] EdNote: introduce this above

[29] EdNote: I think that we only get a semivaluation, look it up in Andrews.

P.4.1 $\mathbf{A} = p(\mathbf{A}_1, \ldots, \mathbf{A}_n)$: then $\mathcal{I}_\varphi(\mathbf{A}) = \mathcal{I}(p)(\mathcal{I}_\varphi(\mathbf{A}_1), \ldots, \mathcal{I}_\varphi(\mathbf{A}_n)) = \mathcal{I}(p)(\varphi(\mathbf{A}_1), \ldots, \varphi(\mathbf{A}_n)) = \nu(p(\varphi(\mathbf{A}_1), \ldots, \varphi(\mathbf{A}_n))) = \nu(\varphi(p(\mathbf{A}_1, \ldots, \mathbf{A}_n))) = \nu(\varphi(\mathbf{A}))$

P.4.2 $\mathbf{A} = \neg\mathbf{B}$: then $\mathcal{I}_\varphi(\mathbf{A}) = \mathsf{T}$, iff $\mathcal{I}_\varphi(\mathbf{B}) = \nu(\varphi(\mathbf{B})) = \mathsf{F}$, iff $\nu(\varphi(\mathbf{A})) = \mathsf{T}$.

P.4.3 $\mathbf{A} = \mathbf{B} \wedge \mathbf{C}$: similar

P.4.4 $\mathbf{A} = \forall X.\mathbf{B}$: then $\mathcal{I}_\varphi(\mathbf{A}) = \mathsf{T}$, iff $\mathcal{I}_\psi(\mathbf{B}) = \nu(\psi(\mathbf{B})) = \mathsf{T}$, for all $\mathbf{C} \in \mathcal{D}_\iota$, where $\psi = \varphi, [\mathbf{C}/X]$. This is the case, iff $\nu(\varphi(\mathbf{A})) = \mathsf{T}$.

P.5 Thus $\mathcal{I}_\varphi(\mathbf{A}) = \nu(\varphi(\mathbf{A})) = \nu(\mathbf{A}) = \mathsf{T}$ for all $\mathbf{A} \in \Phi$.

P.6 Hence $\mathcal{M} \models \mathbf{A}$ for $\mathcal{M} := \langle \mathcal{D}_\iota, \mathcal{I} \rangle$. □

SOME RIGHTS RESERVED ©: Michael Kohlhase 115 JACOBS UNIVERSITY

Now, we only have to put the pieces together to obtain the model existence theorem we are after.

Model Existence

▷ **Theorem 3.53 (Hintikka-Lemma)** *If ∇ is an abstract consistency class and $\mathcal{H}$ a ∇-Hintikka set, then $\mathcal{H}$ is satisfiable.*

▷ Proof:

P.1 we define $\nu(\mathbf{A}) := \mathsf{T}$, iff $\mathbf{A} \in \mathcal{H}$,

P.2 then ν is a valuation by the Hintikka set properties.

P.3 We have $\nu(\mathcal{H}) = \{\mathsf{T}\}$, so $\mathcal{H}$ is satisfiable. □

▷ **Theorem 3.54 (Model Existence)** *If ∇ is an abstract consistency class and $\Phi \in \nabla$, then Φ is satisfiable.*

Proof:

▷ **P.1** There is a ∇-Hintikka set $\mathcal{H}$ with $\Phi \subseteq \mathcal{H}$ (Extension Theorem)

We know that $\mathcal{H}$ is satisfiable. (Hintikka-Lemma)

In particular, $\Phi \subseteq \mathcal{H}$ is satisfiable. □

SOME RIGHTS RESERVED ©: Michael Kohlhase 116 JACOBS UNIVERSITY

3.4 First-Order Inference with Tableaux

3.4.1 First-Order Tableaux

P.2 P.3

Test Calculi: Tableaux and Model Generation

▷ Idea: instead of showing $\emptyset \vdash Th$, show $\neg Th \vdash trouble$ (use $\perp$ for trouble)

▷ **Example 3.55** Tableau Calculi try to construct models.

Tableau Refutation (Validity)	Model generation (Satisfiability)
$\models P \wedge Q \Rightarrow Q \wedge P$	$\models P \wedge (Q \vee \neg R) \wedge \neg Q$
$P \wedge Q \Rightarrow Q \wedge P^{\mathsf{f}}$ $P \wedge Q^{\mathsf{t}}$ $Q \wedge P^{\mathsf{f}}$ P^{t} Q^{t} $P^{\mathsf{f}} \mid Q^{\mathsf{f}}$ $\bot \mid \bot$	$P \wedge (Q \vee \neg R) \wedge \neg Q^{\mathsf{t}}$ $P \wedge (Q \vee \neg R)^{\mathsf{t}}$ $\neg Q^{\mathsf{t}}$ Q^{f} P^{t} $Q \vee \neg R^{\mathsf{t}}$ $Q^{\mathsf{t}} \mid \neg R^{\mathsf{t}}$ $\bot \mid R^{\mathsf{f}}$
No Model	Herbrand Model $\{P^{\mathsf{t}}, Q^{\mathsf{f}}, R^{\mathsf{f}}\}$ $\varphi := \{P \mapsto \mathsf{T}, Q \mapsto \mathsf{F}, R \mapsto \mathsf{F}\}$

Algorithm: Fully expand all possible tableaux, (no rule can be applied)

▷ ▹ Satisfiable, iff there are open branches (correspond to models)

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

Tableau calculi develop a formula in a tree-shaped arrangement that represents a case analysis on when a formula can be made true (or false). Therefore the formulae are decorated with exponents that hold the intended truth value.

On the left we have a refutation tableau that analyzes a negated formula (it is decorated with the intended truth value F). Both branches contain an elementary contradiction $\bot$.

On the right we have a model generation tableau, which analyzes a positive formula (it is decorated with the intended truth value T. This tableau uses the same rules as the refutation tableau, but makes a case analysis of when this formula can be satisfied. In this case we have a closed branch and an open one, which corresponds a model).

Now that we have seen the examples, we can write down the tableau rules formally.

Analytical Tableaux (Formal Treatment of $\mathcal{T}_0$)

▷ formula is analyzed in a tree to determine satisfiability

▷ branches correspond to valuations (models)

▷ one per connective

$$\frac{\mathbf{A} \wedge \mathbf{B}^{\mathsf{t}}}{\begin{array}{c}\mathbf{A}^{\mathsf{t}}\\ \mathbf{B}^{\mathsf{t}}\end{array}}\mathcal{T}_0\wedge \qquad \frac{\mathbf{A} \wedge \mathbf{B}^{\mathsf{f}}}{\mathbf{A}^{\mathsf{f}} \mid \mathbf{B}^{\mathsf{f}}}\mathcal{T}_0\vee \qquad \frac{\neg\mathbf{A}^{\mathsf{t}}}{\mathbf{A}^{\mathsf{f}}}\mathcal{T}_0\neg^{\mathsf{T}} \qquad \frac{\neg\mathbf{A}^{\mathsf{f}}}{\mathbf{A}^{\mathsf{t}}}\mathcal{T}_0\neg^{\mathsf{F}} \qquad \frac{\begin{array}{c}\mathbf{A}^{\alpha}\\ \mathbf{A}^{\beta}\end{array} \quad \alpha \neq \beta}{\bot}\mathcal{T}_0\mathsf{cut}$$

▷ Use rules exhaustively as long as they contribute new material

▷ **Definition 3.56** Call a tableau saturated, iff no rule applies, and a branch closed, iff it ends in $\bot$, else open. (open branches in saturated tableaux yield models)

▷ **Definition 3.57 ($\mathcal{T}_0$-Theorem/Derivability)** $\mathbf{A}$ is a $\mathcal{T}_0$-theorem ($\vdash_{\mathcal{T}_0} \mathbf{A}$), iff there is a closed tableau with $\mathbf{A}^{\mathsf{F}}$ at the root.

$\Phi \subseteq \mathit{wff}_o(\mathcal{V}_o)$ derives $\mathbf{A}$ in $\mathcal{T}_0$ ($\Phi \vdash_{\mathcal{T}_0} \mathbf{A}$), iff there is a closed tableau starting with $\mathbf{A}^{\mathsf{F}}$ and Φ^{T}.

SOME RIGHTS RESERVED ©: Michael Kohlhase 118 JACOBS UNIVERSITY

These inference rules act on tableaux have to be read as follows: if the formulae over the line appear in a tableau branch, then the branch can be extended by the formulae or branches below the line. There are two rules for each primary connective, and a branch closing rule that adds the special symbol $\bot$ (for unsatisfiability) to a branch.

We use the tableau rules with the convention that they are only applied, if they contribute new material to the branch. This ensures termination of the tableau procedure for propositional logic (every rule eliminates one primary connective).

Definition 3.58 We will call a closed tableau with the signed formula $\mathbf{A}^\alpha$ at the root a tableau refutation for $\mathcal{A}^\alpha$.

The saturated tableau represents a full case analysis of what is necessary to give $\mathbf{A}$ the truth value α; since all branches are closed (contain contradictions) this is impossible.

Definition 3.59 We will call a tableau refutation for $\mathbf{A}^{\mathsf{f}}$ a tableau proof for $\mathbf{A}$, since it refutes the possibility of finding a model where $\mathbf{A}$ evaluates to F. Thus $\mathbf{A}$ must evaluate to T in all models, which is just our definition of validity.

Thus the tableau procedure can be used as a calculus for propositional logic. In contrast to the calculus in section ?sec.hilbert? it does not prove a theorem $\mathbf{A}$ by deriving it from a set of axioms, but it proves it by refuting its negation. Such calculi are called negative or test calculi. Generally negative calculi have computational advantages over positive ones, since they have a built-in sense of direction.

We have rules for all the necessary connectives (we restrict ourselves to $\wedge$ and $\neg$, since the others can be expressed in terms of these two via the propositional identities above. For instance, we can write $\mathbf{A} \vee \mathbf{B}$ as $\neg(\neg\mathbf{A} \wedge \neg\mathbf{B})$, and $\mathbf{A} \Rightarrow \mathbf{B}$ as $\neg\mathbf{A} \vee \mathbf{B}$,....)

We will now extend the propositional tableau techiques to first-order logic. We only have to add two new rules for the universal quantifiers (in positive and negative polarity).

First-Order Standard Tableaux ($\mathcal{T}_1$)

- ▷ Refutation calculus based on trees of labeled formulae
  - ▷ Tableau-Rules: $\mathcal{T}_0$ (propositional tableau rules) plus

$$\frac{\forall X.\mathbf{A}^{\mathsf{t}} \quad \mathbf{C} \in cwff_\iota(\Sigma_\iota)}{[\mathbf{C}/X](\mathbf{A})^{\mathsf{t}}}\mathcal{T}_1{:}\forall \qquad \frac{\forall X.\mathbf{A}^{\mathsf{f}} \quad c \in (\Sigma_0^{sk} \backslash \mathcal{H})}{[c/X](\mathbf{A})^{\mathsf{f}}}\mathcal{T}_1{:}\exists$$

SOME RIGHTS RESERVED ©: Michael Kohlhase 119 JACOBS UNIVERSITY

The rule $\mathcal{T}_1$:$\forall$ rule operationalizes the intuition that a universally quantified formula is true, iff all of the instances of the scope are. To understand the $\mathcal{T}_1$:$\exists$ rule, we have to keep in mind that $\exists X.\mathbf{A}$ abbreviates $\neg(\forall X.\neg\mathbf{A})$, so that we have to read $\forall X.\mathbf{A}^{\mathsf{F}}$ existentially — i.e. as $\exists X.\neg\mathbf{A}^{\mathsf{T}}$, stating that there is an object with property $\neg\mathbf{A}$. In this situation, we can simply give this object a name: c, which we take from our (infinite) set of witness constants Σ_0^{sk}, which we have given ourselves expressly for this purpose when we defined first-order syntax. In other words $[c/X](\neg\mathbf{A})^{\mathsf{T}} = [c/X](\mathbf{A})^{\mathsf{F}}$ holds, and this is just the conclusion of the $\mathcal{T}_1$:$\exists$ rule.

Note that the $\mathcal{T}_1$:$\forall$ rule is computationally extremely inefficient: we have to guess an (i.e. in a search setting to systematically consider all) instance $\mathbf{C} \in wff_\iota(\Sigma_\iota)$ for X. This makes the rule infinitely branching.

3.4.2 Free Variable Tableaux

In the next calculus we will try to remedy the computational inefficiency of the $\mathcal{T}_1$:$\forall$ rule. We do this by delaying the choice in the universal rule.

Free variable Tableaux ($\mathcal{T}_1^f$)

- Refutation calculus based on trees of labeled formulae
  - $\mathcal{T}_0$ (propositional tableau rules) plus
  - Quantifier rules:

$$\frac{\forall X.\mathbf{A}^{\mathsf{t}} \quad Y new}{[Y/X](\mathbf{A})^{\mathsf{t}}}\mathcal{T}_1^f{:}\forall \qquad \frac{\forall X.\mathbf{A}^{\mathsf{f}} \quad \mathbf{free}(\forall X.\mathbf{A}) = \{X^1, \ldots, X^k\} \quad f \in \Sigma_k^{sk}}{[f(X^1, \ldots, X^k)/X](\mathbf{A})^{\mathsf{f}}}\mathcal{T}_1^f{:}\exists$$

  - Generalized cut rule $\mathcal{T}_1^f{:}\bot$ instantiates the whole tableau by σ.

$$\frac{\begin{array}{l}\mathbf{A}^{\alpha} \\ \mathbf{B}^{\beta}\end{array} \quad \alpha \neq \beta \quad \sigma(\mathbf{A}) = \sigma(\mathbf{B})}{\bot}\mathcal{T}_1^f{:}\bot$$

Advantage: no guessing necessary in $\mathcal{T}_1^f$:$\forall$-rule

- New: find suitable substitution (most general unifier)

SOME RIGHTS RESERVED ©: Michael Kohlhase 120 JACOBS UNIVERSITY

Metavariables: Instead of guessing a concrete instance for the universally quantified variable as in the $\mathcal{T}_1$:$\forall$ rule, $\mathcal{T}_1^f$:$\forall$ instantiates it with a new meta-variable Y, which will be instantiated by need in the course of the derivation.

Skolem terms as witnesses: The introduction of meta-variables makes is necessary to extend the treatment of witnesses in the existential rule. Intuitively, we cannot simply invent a new name, since the meaning of the body $\mathbf{A}$ may contain meta-variables introduced by the $\mathcal{T}_1^f$:$\forall$ rule. As we do not know their values yet, the witness for the existential statement in the antecedent of the $\mathcal{T}_1^f$:$\exists$ rule needs to depend on that. So witness it using a witness term, concretely by applying a Skolem function to the meta-variables in $\mathbf{A}$.

Instantiating Metavariables: Finally, the $\mathcal{T}_1^f$:$\bot$ rule completes the treatment of meta-variables, it allows to instantiate the whole tableau in a way that the current branch closes. This leaves us with the problem of finding substitutions that make two terms equal.

Multiplicity in Tableaux

- **Observation 3.60** *All $\mathcal{T}_1^f$ rules except $\mathcal{T}_1^f$:$\forall$ only need to be applied once.*
- **Example 3.61** A tableau proof for $(p(a) \vee p(b)) \Rightarrow (\exists x.p(x))$.

Start, close branch	use $\mathcal{T}_1^f{:}\forall$ again
$\begin{array}{c} (p(a) \vee p(b)) \Rightarrow (\exists x.p(x))^{\mathsf{f}} \\ p(a) \vee p(b)^{\mathsf{t}} \\ \exists x.p(x)^{\mathsf{f}} \\ \forall x.\neg p(x)^{\mathsf{t}} \\ \neg p(y)^{\mathsf{t}} \\ p(y)^{\mathsf{f}} \\ \begin{array}{c\|c} p(a)^{\mathsf{t}} & p(b)^{\mathsf{t}} \\ \bot : [a/x] & \end{array} \end{array}$	$\begin{array}{c} (p(a) \vee p(b)) \Rightarrow (\exists x.p(x))^{\mathsf{f}} \\ p(a) \vee p(b)^{\mathsf{t}} \\ \exists x.p(x)^{\mathsf{f}} \\ \forall x.\neg p(x)^{\mathsf{t}} \\ \neg p(a)^{\mathsf{t}} \\ p(a)^{\mathsf{f}} \\ \begin{array}{c\|c} p(a)^{\mathsf{t}} & p(b)^{\mathsf{t}} \\ \bot & \neg p(z)^{\mathsf{t}} \\ & p(z)^{\mathsf{f}} \\ & \bot : [b/z] \end{array} \end{array}$

- ▷ **Definition 3.62** Let $\mathcal{T}$ be a tableau for $\mathbf{A}$, and a positive occurrence of $\forall x.\mathbf{B}$ in $\mathbf{A}$, then we call the number of applications of $\mathcal{T}_1^f{:}\forall$ to $\forall x.\mathbf{B}$ its multiplicity.
- ▷ **Observation 3.63** *Given a prescribed multiplicity for each positive $\forall$, saturation with $\mathcal{T}_1^f$ terminates.*
- ▷ Proof Sketch: All $\mathcal{T}_1^f$ rules reduce the number of connectives and negative $\forall$ or the multiplicity of positive $\forall$. □
- ▷ **Theorem 3.64** $\mathcal{T}_1^f$ *is only complete with unbounded multiplicities.*
- ▷ Proof Sketch: Otherwise validity in PL^1 would be decidable. □

SOME RIGHTS RESERVED

JACOBS UNIVERSITY

Treating $\mathcal{T}_1^f{:}\bot$

- ▷ The $\mathcal{T}_1^f{:}\bot$ rule instantiates the whole tableau.
- ▷ There may be more than one $\mathcal{T}_1^f{:}\bot$ opportunity on a branch
- ▷ **Example 3.65** Choosing which matters – this tableau does not close!

$$\begin{array}{c} \exists x.(p(a) \wedge p(b) \Rightarrow p(x)) \wedge (q(b) \Rightarrow q(x))^{\mathsf{f}} \\ (p(a) \wedge p(b) \Rightarrow p(y)) \wedge (q(b) \Rightarrow q(y))^{\mathsf{f}} \\ \begin{array}{c|c} p(a) \Rightarrow p(b) \Rightarrow p(y)^{\mathsf{f}} & q(b) \Rightarrow q(y)^{\mathsf{f}} \\ p(a)^{\mathsf{t}} & q(b)^{\mathsf{t}} \\ p(b)^{\mathsf{t}} & q(y)^{\mathsf{f}} \\ p(y)^{\mathsf{f}} & \\ \bot : [a/y] & \end{array} \end{array}$$

  choosing the other $\mathcal{T}_1^f{:}\bot$ in the left branch allows closure.
- ▷ Two ways of systematic proof search in $\mathcal{T}_1^f$:
  - ▷ backtracking search over $\mathcal{T}_1^f{:}\bot$ opportunities
  - ▷ saturate without $\mathcal{T}_1^f{:}\bot$ and find spanning matings (later)

SOME RIGHTS RESERVED

JACOBS UNIVERSITY

Spanning Matings for $\mathcal{T}_1^f{:}\bot$

- **Observation 3.66** $\mathcal{T}_1^f$ *without* $\mathcal{T}_1^f{:}\bot$ *is terminating and confluent for given multiplicities.*
- Idea: Saturate without $\mathcal{T}_1^f{:}\bot$ and treat all cuts at the same time.
- **Definition 3.67** Let $\mathcal{T}$ be a $\mathcal{T}_1^f$ tableau, then we call a unification problem $\mathcal{E} := (\mathbf{A}_1 =^? \mathbf{A}_1) \wedge \ldots \wedge (\mathbf{A}_n =^? \mathbf{B}_n)$ a mating for $\mathcal{T}$, iff $\mathbf{A}_i^{\mathsf{t}}$ and $\mathbf{B}_i^{\mathsf{f}}$ occur in $\mathcal{T}$.

  We say that $\mathcal{E}$ is a spanning mating, if $\mathcal{E}$ is unifiable and every branch $\mathcal{B}$ of $\mathcal{T}$ contains $\mathbf{A}_i^{\mathsf{t}}$ and $\mathbf{B}_i^{\mathsf{f}}$ for some i.
- **Theorem 3.68** *A* $\mathcal{T}_1^f$*-tableau with a spanning mating induces a closed* $\mathcal{T}_1$*-tableau.*
- Proof Sketch: Just apply the unifier of the spanning mating. □
- Idea: Existence is sufficient, we do not need to compute the unifier
- Implementation: Saturate without $\mathcal{T}_1^f{:}\bot$, backtracking search for spanning matings with $\mathcal{DU}$, adding pairs incrementally.

SOME RIGHTS RESERVED ©: Michael Kohlhase 123 JACOBS UNIVERSITY

3.4.3 First-Order Unification

We will now look into the problem of finding a substitution σ that make two terms equal (we say it unifies them) in more detail. The presentation of the unification algorithm we give here "transformation-based" this has been a very influential way to treat certain algorithms in theoretical computer science.

A transformation-based view of algorithms: The "transformation-based" view of algorithms divides two concerns in presenting and reasoning about algorithms according to Kowalski's slogan[30] EdN:30

computation = logic + control

The computational paradigm highlighted by this quote is that (many) algorithms can be thought of as manipulating representations of the problem at hand and transforming them into a form that makes it simple to read off solutions. Given this, we can simplify thinking and reasoning about such algorithms by separating out their "logical" part, which deals with is concerned with how the problem representations can be manipulated in principle from the "control" part, which is concerned with questions about when to apply which transformations.

It turns out that many questions about the algorithms can already be answered on the "logic" level, and that the "logical" analysis of the algorithm can already give strong hints as to how to optimize control.

In fact we will only concern ourselves with the "logical" analysis of unification here.

The first step towards a theory of unification is to take a closer look at the problem itself. A first set of examples show that we have multiple solutions to the problem of finding substitutions that make two terms equal. But we also see that these are related in a systematic way.

[30] EdNote: find the reference, and see what he really said

Unification (Definitions)

- ▷ Problem: For given terms $\mathbf{A}$ and $\mathbf{B}$ find a substitution σ, such that $\sigma(\mathbf{A}) = \sigma(\mathbf{B})$.
- ▷ **Notation 3.69** We write term pairs as $\mathbf{A} =^? \mathbf{B}$ e.g. $f(X) =^? f(g(Y))$
- ▷ Solutions (e.g. $[g(a)/X], [a/Y]$, $[g(g(a))/X], [g(a)/Y]$, or $[g(Z)/X], [Z/Y]$) are called unifier s, $\mathbf{U}(\mathbf{A} =^? \mathbf{B}) := \{\sigma \mid \sigma(\mathbf{A}) = \sigma(\mathbf{B})\}$
- ▷ Idea: find representatives in $\mathbf{U}(\mathbf{A} =^? \mathbf{B})$, that generate the set of solutions
- ▷ **Definition 3.70** Let σ and θ be substitutions and $W \subseteq \mathcal{V}_\iota$, we say that a substitution σ is more general than θ (on W write $\sigma \leq \theta[W]$), iff there is a substitution ρ, such that $\theta = \rho \circ \sigma[W]$, where $\sigma = \rho[W]$, iff $\sigma(X) = \rho(X)$ for all $X \in W$.
- ▷ **Definition 3.71** σ is called a most general unifier of $\mathbf{A}$ and $\mathbf{B}$, iff it is minimal in $\mathbf{U}(\mathbf{A} =^? \mathbf{B})$ wrt. $\leq [\mathbf{free}(\mathbf{A}) \cup \mathbf{free}(\mathbf{B})]$.

SOME RIGHTS RESERVED ©: Michael Kohlhase 124 JACOBS UNIVERSITY

The idea behind a most general unifier is that all other unifiers can be obtained from it by (further) instantiation. In an automated theorem proving setting, this means that using most general unifiers is the least committed choice — any other choice of unifiers (that would be necessary for completeness) can later be obtained by other substitutions.

Note that there is a subtlety in the definition of the ordering on substitutions: we only compare on a subset of the variables. The reason for this is that we have defined substitutions to be total on (the infinite set of) variables for flexibility, but in the applications (see the definition of a most general unifiers), we are only interested in a subset of variables: the ones that occur in the initial problem formulation. Intuitively, we do not care what the unifiers do off that set. If we did not have the restriction to the set W of variables, the ordering relation on substitutions would become much too fine-grained to be useful (i.e. to guarantee unique most general unifiers in our case).

Now that we have defined the problem, we can turn to the unification algorithm itself. We will define it in a way that is very similar to logic programming: we first define a calculus that generates "solved forms" (formulae from which we can read off the solution) and reason about control later. In this case we will reason that control does not matter.

Unification (Equational Systems)

- ▷ Idea: Unification is equation solving.
- ▷ **Definition 3.72** We call a formula $\mathbf{A}^1 =^? \mathbf{B}^1 \wedge \ldots \wedge \mathbf{A}^n =^? \mathbf{B}^n$ an equational system iff $\mathbf{A}^i, \mathbf{B}^i \in \mathit{wff}_\iota(\Sigma_\iota, \mathcal{V}_\iota)$.
- ▷ We consider equational systems as sets of equations ($\wedge$ is ACI), and equations as two-element multisets ($=^?$ is C).

SOME RIGHTS RESERVED ©: Michael Kohlhase 125 JACOBS UNIVERSITY

In principle, unification problems are sets of equations, which we write as conjunctions, since all of them have to be solved for finding a unifier. Note that it is not a problem for the "logical view" that the representation as conjunctions induces an order, since we know that conjunction is associative, commutative and idempotent, i.e. that conjuncts do not have an intrinsic order or multiplicity, if we consider two equational problems as equal, if they are equivalent as propositional formulae. In the same way, we will abstract from the order in equations, since we know that the equality relation is symmetric. Of course we would have to deal with this somehow in the implementation (typically, we would implement equational problems as lists of pairs), but that belongs into the "control" aspect of the algorithm, which we are abstracting from at the moment.

Solved forms and Most General Unifiers

- **Definition 3.73** We call a pair $\mathbf{A} =^? \mathbf{B}$ solved in a unification problem $\mathcal{E}$, iff $\mathbf{A} = X$, $\mathcal{E} = X =^? \mathbf{A} \wedge \mathcal{E}$, and $X \notin (\mathbf{free}(\mathbf{A}) \cup \mathbf{free}(\mathcal{E}))$. We call an unification problem $\mathcal{E}$ a solved form, iff all its pairs are solved.
- **Lemma 3.74** *Solved forms are of the form $X^1 =^? \mathbf{B}^1 \wedge \ldots \wedge X^n =^? \mathbf{B}^n$ where the X^i are distinct and $X^i \notin \mathbf{free}(\mathbf{B}^j)$.*
- **Definition 3.75** Any substitution $\sigma = [\mathbf{B}^1/X^1], \ldots, [\mathbf{B}^n/X^n]$ induces a solved unification problem $\mathcal{E}_\sigma := (X^1 =^? \mathbf{B}^1) \wedge \ldots \wedge (X^n =^? \mathbf{B}^n)$.
- **Lemma 3.76** *If $\mathcal{E} = X^1 =^? \mathbf{B}^1 \wedge \ldots \wedge X^n =^? \mathbf{B}^n$ is a solved form, then $\mathcal{E}$ has the unique most general unifier $\sigma_\mathcal{E} := [\mathbf{B}^1/X^1], \ldots, [\mathbf{B}^n/X^n]$.*
- Proof: Let $\theta \in \mathbf{U}(\mathcal{E})$

  P.1 then $\theta(X^i) = \theta(\mathbf{B}^i) = \theta \circ \sigma_\mathcal{E}(X^i)$

  P.2 and thus $\theta = \theta \circ \sigma_\mathcal{E}[\mathbf{supp}(\sigma)]$. □

  Note: we can rename the introduced variables in most general unifiers!

SOME RIGHTS RESERVED ©: Michael Kohlhase 126 JACOBS UNIVERSITY

It is essential to our "logical" analysis of the unification algorithm that we arrive at equational problems whose unifiers we can read off easily. Solved forms serve that need perfectly as Lemma 3.76 shows.

Given the idea that unification problems can be expressed as formulae, we can express the algorithm in three simple rules that transform unification problems into solved forms (or unsolvable ones).

▷ Unification Algorithm

- **Definition 3.77** Inference system $\mathcal{U}$

$$\frac{\mathcal{E} \wedge f(\mathbf{A}^1, \ldots, \mathbf{A}^n) =^? f(\mathbf{B}^1, \ldots, \mathbf{B}^n)}{\mathcal{E} \wedge \mathbf{A}^1 =^? \mathbf{B}^1 \wedge \ldots \wedge \mathbf{A}^n =^? \mathbf{B}^n} \mathcal{U}\mathrm{dec} \qquad \frac{\mathcal{E} \wedge \mathbf{A} =^? \mathbf{A}}{\mathcal{E}} \mathcal{U}\mathrm{triv}$$

$$\frac{\mathcal{E} \wedge X =^? \mathbf{A} \quad X \notin \mathbf{free}(\mathbf{A}) \quad X \in \mathbf{free}(\mathcal{E})}{[\mathbf{A}/X](\mathcal{E}) \wedge X =^? \mathbf{A}} \mathcal{U}\mathrm{elim}$$

- **Lemma 3.78** *$\mathcal{U}$ is correct: $\mathcal{E} \vdash_\mathcal{U} \mathcal{F}$ implies $\mathbf{U}(\mathcal{F}) \subseteq \mathbf{U}(\mathcal{E})$*
- **Lemma 3.79** *$\mathcal{U}$ is complete: $\mathcal{E} \vdash_\mathcal{U} \mathcal{F}$ implies $\mathbf{U}(\mathcal{E}) \subseteq \mathbf{U}(\mathcal{F})$*
- **Lemma 3.80** *$\mathcal{U}$ is confluent: the order of derivations does not matter*

▷ **Corollary 3.81** *First-Order Unification is unitary: i.e. most general unifiers are unique up to renaming of introduced variables.*

▷ Proof Sketch: the inference system $\mathcal{U}$ is trivially branching □

SOME RIGHTS RESERVED ©: Michael Kohlhase 127 JACOBS UNIVERSITY

The decomposition rule $\mathcal{U}$dec is completely straightforward, but note that it transforms one unification pair into multiple argument pairs; this is the reason, why we have to directly use unification problems with multiple pairs in $\mathcal{U}$.

Note furthermore, that we could have restricted the $\mathcal{U}$triv rule to variable-variable pairs, since for any other pair, we can decompose until only variables are left. Here we observe, that constant-constant pairs can be decomposed with the $\mathcal{U}$dec rule in the somewhat degenerate case without arguments.

Finally, we observe that the first of the two variable conditions in $\mathcal{U}$elim (the "occurs-in-check") makes sure that we only apply the transformation to unifiable unification problems, whereas the second one is a termination condition that prevents the rule to be applied twice.

The notion of completeness and correctness is a bit different than that for calculi that we compare to the entailment relation. We can think of the "logical system of unifiability" with the model class of sets of substitutions, where a set satisfies an equational problem $\mathcal{E}$, iff all of its members are unifiers. This view induces the soundness and completeness notions presented above.

The three meta-properties above are relatively trivial, but somewhat tedious to prove, so we leave the proofs as an exercise to the reader.

We now fortify our intuition about the unification calculus by two examples. Note that we only need to pursue one possible $\mathcal{U}$ derivation since we have confluence.

Unification Examples

Example 3.82 Two similar unification problems:

$$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{f(g(x,x),h(a)) =^? f(g(a,z),h(z))}{g(x,x) =^? g(a,z) \wedge h(a) =^? h(z)}\,\mathcal{U}\mathrm{dec}}{x =^? a \wedge x =^? z \wedge h(a) =^? h(z)}\,\mathcal{U}\mathrm{dec}}{x =^? a \wedge x =^? z \wedge a =^? z}\,\mathcal{U}\mathrm{dec}}{x =^? a \wedge a =^? z \wedge a =^? z}\,\mathcal{U}\mathrm{elim}}{x =^? a \wedge z =^? a \wedge a =^? a}\,\mathcal{U}\mathrm{elim}}{x =^? a \wedge z =^? a}\,\mathcal{U}\mathrm{triv}$$	$$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{f(g(x,x),h(a)) =^? f(g(b,z),h(z))}{g(x,x) =^? g(b,z) \wedge h(a) =^? h(z)}\,\mathcal{U}\mathrm{dec}}{x =^? b \wedge x =^? z \wedge h(a) =^? h(z)}\,\mathcal{U}\mathrm{dec}}{x =^? b \wedge x =^? z \wedge a =^? z}\,\mathcal{U}\mathrm{dec}}{x =^? b \wedge b =^? z \wedge a =^? z}\,\mathcal{U}\mathrm{elim}}{x =^? a \wedge z =^? a \wedge a =^? b}\,\mathcal{U}\mathrm{elim}$$
MGU: $[a/x], [a/z]$	$a =^? b$ not unifiable

SOME RIGHTS RESERVED ©: Michael Kohlhase 128 JACOBS UNIVERSITY

We will now convince ourselves that there cannot be any infinite sequences of transformations in $\mathcal{U}$. Termination is an important property for an algorithm.

The proof we present here is very typical for termination proofs. We map unification problems into a partially ordered set $\langle S, \prec\rangle$ where we know that there cannot be any infinitely descending sequences (we think of this as measuring the unification problems). Then we show that all transformations in $\mathcal{U}$ strictly decrease the measure of the unification problems and argue that if there were an infinite transformation in $\mathcal{U}$, then there would be an infinite descending chain in S, which contradicts our choice of $\langle S, \prec\rangle$.

The crucial step in in coming up with such proofs is finding the right partially ordered set. Fortunately, there are some tools we can make use of. We know that $\langle\mathbb{N}, <\rangle$ is terminating, and there are some ways of lifting component orderings to complex structures. For instance it is well-known that the lexicographic ordering lifts a terminating ordering to a terminating ordering on finite-dimensional Cartesian spaces. We show a similar, but less known construction with multisets for our proof.

Unification (Termination)

▷ **Definition 3.83** Let S and T be multisets and $\prec$ a partial ordering on $S \cup T$. Then we define $(S \prec^m T)$, iff $S = C \uplus T'$ and $T = C \uplus \{t\}$, where $s \prec t$ for all $s \in S'$. We call $\prec^m$ the multiset ordering induced by $\prec$.

▷ **Lemma 3.84** *If $\prec$ is total/terminating on S, then $\prec^m$ is total/terminating on $\mathcal{P}(S)$.*

▷ **Lemma 3.85** *$\mathcal{U}$ is terminating* *(any $\mathcal{U}$-derivation is finite)*

▷ Proof: We prove termination by mapping $\mathcal{U}$ transformation into a Noetherian space.

P.1 Let $\mu(\mathcal{E}) := \langle n, \mathcal{N}\rangle$, where

- ▷ m is the number of unsolved variables in $\mathcal{E}$
- ▷ $\mathcal{N}$ is the multiset of term depths in $\mathcal{E}$

P.2 The lexicographic order $\prec$ on pairs $\mu(\mathcal{E})$ is decreased by all inference rules.

P.2.1 $\mathcal{U}$dec and $\mathcal{U}$triv decrease the multiset of term depths without increasing the unsolved variables

P.2.2 $\mathcal{U}$elim decreases the number of unsolved variables (by one), but may increase term depths. □

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

But it is very simple to create terminating calculi, e.g. by having no inference rules. So there is one more step to go to turn the termination result into a decidability result: we must make sure that we have enough inference rules so that any unification problem is transformed into solved form if it is unifiable.

Unification (decidable)

▷ **Definition 3.86** We call an equational problem $\mathcal{E}$ $\mathcal{U}$-reducible, iff there is a $\mathcal{U}$-step $\mathcal{E} \vdash_{\mathcal{U}} \mathcal{F}$ from $\mathcal{E}$.

▷ **Lemma 3.87** *If $\mathcal{E}$ is unifiable but not solved, then it is $\mathcal{U}$-reducible*

▷ Proof: We assume that $\mathcal{E}$ is unifiable but unsolved and show the $\mathcal{U}$ rule that applies.

P.1 There is an unsolved pair $\mathbf{A} =^? \mathbf{B}$ in $\mathcal{E} = \mathcal{E}' \wedge \mathbf{A} =^? \mathbf{B}$.

P.2 we have two cases

P.2.1 $\mathbf{A}, \mathbf{B} \notin \mathcal{V}_\iota$: then $\mathbf{A} = f(\mathbf{A}^1 \ldots \mathbf{A}^n)$ and $\mathbf{B} = f(\mathbf{B}^1 \ldots \mathbf{B}^n)$, and thus $\mathcal{U}$dec is applicable

P.2.2 $\mathbf{A} = X \in \mathbf{free}(\mathcal{E})$: then $\mathcal{U}$elim (if $\mathbf{B} \neq X$) or $\mathcal{U}$triv (if $\mathbf{B} = X$) is applicable. □

▷ **Corollary 3.88** *Unification is decidable in PL^1.*

▷ Proof Idea: $\mathcal{U}$-irreducible sets of equations can be obtained in finite time by Lemma 3.85 and are either solved or unsolvable by Lemma 3.87, so they provide the answer. □

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

3.4.4 Efficient Unification

Complexity of Unification

▷ Observation: Naive unification is exponential in time and space.

▷ consider the terms

$$\begin{aligned} s_n &= f(f(x_0,x_0),f(f(x_1,x_1),f(\ldots,f(x_{n-1},x_{n-1}))\ldots)) \\ t_n &= f(x_1,f(x_2,f(x_3,f(\ldots,x_n)\ldots))) \end{aligned}$$

▷ The most general unifier of s_n and t_n is
$[f(x_0,x_0)/x_1],[f(f(x_0,x_0),f(x_0,x_0))/x_2],[f(f(f(x_0,x_0),f(x_0,x_0)),f(f(x_0,x_0),f(x_0,x_0)))/x_3],\ldots$

▷ it contains $\sum_{i=1}^{n} 2^i = 2^{n+1} - 2$ occurrences of the variable x_0. (exponential)

▷ Problem: the variable x_0 has been copied too often

▷ Idea: Find a term representation that re-uses subterms

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

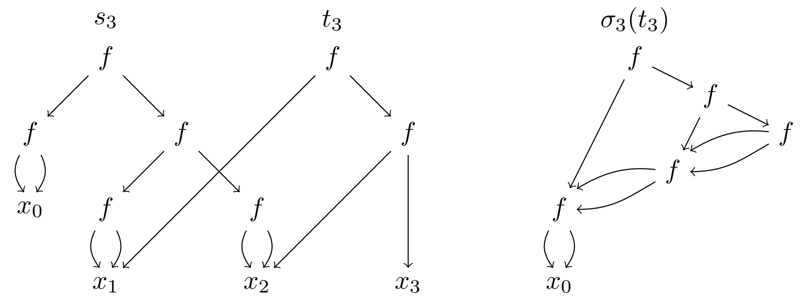
Directed Acyclic Graphs (DAGs)

▷ use directed acyclic graphs for the term representation

- ▷ variables my only occur once in the DAG
- ▷ subterms can be referenced multiply

▷ **Observation 3.89** *Terms can be transformed into DAGs in linear time*

▷ **Example 3.90** $s_3, t_3, \sigma_3(s_3)$



SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

DAG Unification Algorithm

- **Definition 3.91** We say that $X^1 =^? \mathbf{B}^1 \wedge \ldots \wedge X^n =^? \mathbf{B}^n$ is a DAG solved form, iff the X^i are distinct and $X^i \notin \mathbf{free}(\mathbf{B}^j)$ for $i \leq j$
- **Definition 3.92** The inference system $\mathcal{DU}$ contains rules $\mathcal{U}$dec and $\mathcal{U}$triv from $\mathcal{U}$ plus the following:

$$\frac{\mathcal{E} \wedge X =^? \mathbf{A} \wedge X =^? \mathbf{B} \quad \mathbf{A}, \mathbf{B} \notin \mathcal{V}_\iota \quad |\mathbf{A}| \leq |\mathbf{B}|}{\mathcal{E} \wedge X =^? \mathbf{A} \wedge \mathbf{A} =^? \mathbf{B}} \mathcal{DU}\text{merge}$$

$$\frac{\mathcal{E} \wedge X =^? Y \quad X \neq Y \quad X, Y \in \mathbf{free}(\mathcal{E})}{[Y/X](\mathcal{E}) \wedge X =^? Y} \mathcal{DU}\text{evar}$$

where $|\mathbf{A}|$ is the number of symbols in $\mathbf{A}$.

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

Unification by DAG-chase

- Idea: Extend the Input-DAGs by edges that represent unifiers.
- write $n.a$, if a is the symbol of node n.
- auxiliary procedures: (all linear or constant time)
  - $\mathtt{find}(n)$ follows the path from n and returns the end node
  - $\mathtt{union}(n, m)$ adds an edge between n and m.
  - $\mathtt{occur}(n, m)$ determines whether $n.x$ occurs in the DAG with root m.

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

Algorithm unify

- Input: symmetric pairs of nodes in DAGs

```
fun unify(n,n) = true
  | unify(n.x,m) = if occur(n,m) then true else union(n,m)
```

```
  | unify(n.f,m.g) = if g!=f then false
                     else forall (i,j) => unify(find(i),find(j)) (chld m,chld n)
                     end
```

- linear in space, since no new nodes are created, and at most one link per variable.
- consider terms $f(s_n, f(t'_n, x_n)), f(t_n, f(s'_n, y_n)))$, where $s'_n = [y_i/x_i](s_n)$ und $t'_n = [y_i/x_i](t_n)$.
- `unify` needs exponentially many recursive calls to unify the nodes x_n and y_n. (they are unified after n calls, but checking needs the time)
- Idea: Also bind the function nodes, if the arguments are unified.

```
  unify(n.f,m.g) = if g!=f then false
          else union(n,m);
                     forall (i,j) => unify(find(i),find(j)) (chld m,chld n)
          end
```

- this only needs linearly many recursive calls as it directly returns with true or makes a node inaccessible for `find`.
- linearly many calls to linear procedures give quadratic runtime.

SOME RIGHTS RESERVED

Spanning Matings for $\mathcal{T}_1^f{:}\bot$

- **Observation 3.93** *$\mathcal{T}_1^f$ without $\mathcal{T}_1^f{:}\bot$ is terminating and confluent for given multiplicities.*
- Idea: Saturate without $\mathcal{T}_1^f{:}\bot$ and treat all cuts at the same time.
- **Definition 3.94** Let $\mathcal{T}$ be a $\mathcal{T}_1^f$ tableau, then we call a unification problem $\mathcal{E} := (\mathbf{A}_1 =^? \mathbf{A}_1) \wedge \ldots \wedge (\mathbf{A}_n =^? \mathbf{B}_n)$ a mating for $\mathcal{T}$, iff $\mathbf{A}_i^{\mathsf{t}}$ and $\mathbf{B}_i^{\mathsf{f}}$ occur in $\mathcal{T}$.

  We say that $\mathcal{E}$ is a spanning mating, if $\mathcal{E}$ is unifiable and every branch $\mathcal{B}$ of $\mathcal{T}$ contains $\mathbf{A}_i^{\mathsf{t}}$ and $\mathbf{B}_i^{\mathsf{f}}$ for some i.
- **Theorem 3.95** *A $\mathcal{T}_1^f$-tableau with a spanning mating induces a closed $\mathcal{T}_1$-tableau.*
- Proof Sketch: Just apply the unifier of the spanning mating. □
- Idea: Existence is sufficient, we do not need to compute the unifier
- Implementation: Saturate without $\mathcal{T}_1^f{:}\bot$, backtracking search for spanning matings with $\mathcal{DU}$, adding pairs incrementally.

SOME RIGHTS RESERVED

Now that we understand basic unification theory, we can come to the meta-theoretical properties of the tableau calculus, which we now discuss to make the understanding of first-order inference complete.

3.4.5 Soundness and Completeness of First-Order Tableaux

For the soundness result, we recap the definition of soundness for test calculi from the propositional case.

Soundness (Tableau)

- ▷ Idea: A test calculus is sound, iff it preserves satisfiability and the goal formulae are unsatisfiable.
- ▷ **Definition 3.96** A labeled formula $\mathbf{A}^{\alpha}$ is valid under φ, iff $\mathcal{I}_{\varphi}(\mathbf{A}) = \alpha$.
- ▷ **Definition 3.97** A tableau $\mathcal{T}$ is satisfiable, iff there is a satisfiable branch $\mathcal{P}$ in $\mathcal{T}$, i.e. if the set of formulae in $\mathcal{P}$ is satisfiable.
- ▷ **Lemma 3.98** *Tableau rules transform satisfiable tableaux into satisfiable ones.*
- ▷ **Theorem 3.99 (Soundness)** *A set Φ of propositional formulae is valid, if there is a closed tableau $\mathcal{T}$ for Φ^{f}.*
- ▷ Proof: by contradiction: Suppose Φ is not valid.

P.1 then the initial tableau is satisfiable (Φ^{f} satisfiable)

P.2 so $\mathcal{T}$ is satisfiable, by Lemma 3.98.

P.3 there is a satisfiable branch (by definition)

P.4 but all branches are closed ($\mathcal{T}$ closed)

□

SOME RIGHTS RESERVED ©: Michael Kohlhase 137 JACOBS UNIVERSITY

Thus we only have to prove Lemma 3.98, this is relatively easy to do. For instance for the first rule: if we have a tableau that contains $\mathbf{A} \wedge \mathbf{B}^{\mathsf{t}}$ and is satisfiable, then it must have a satisfiable branch. If $\mathbf{A} \wedge \mathbf{B}^{\mathsf{t}}$ is not on this branch, the tableau extension will not change satisfiability, so we can assue that it is on the satisfiable branch and thus $\mathcal{I}_{\varphi}(\mathbf{A} \wedge \mathbf{B}) = \mathsf{T}$ for some variable assignment φ. Thus $\mathcal{I}_{\varphi}(\mathbf{A}) = \mathsf{T}$ and $\mathcal{I}_{\varphi}(\mathbf{B}) = \mathsf{T}$, so after the extension (which adds the formulae $\mathbf{A}^{\mathsf{t}}$ and $\mathbf{B}^{\mathsf{t}}$ to the branch), the branch is still satisfiable. The cases for the other rules are similar.

The soundness of the first-order free-variable tableaux calculus can be established a simple induction over the size of the tableau.

Soundness of $\mathcal{T}_1^f$

- ▷ **Lemma 3.100** *Tableau rules transform satisfiable tableaux into satisfiable ones.*
- ▷ Proof:

P.1 we examine the tableau rules in turn

P.1.1 propositional rules: as in propositional tableaux

P.1.2 $\mathcal{T}_1^f{:}\exists$: by Lemma 3.102

P.1.3 $\mathcal{T}_1^f{:}\bot$: by Lemma 3.31 (substitution value lemma)

P.1.4 $\mathcal{T}_1^f{:}\forall$:

P.1.4.1 $\mathcal{I}_\varphi(\forall X.\mathbf{A}) = \mathsf{T}$, iff $\mathcal{I}_\psi(\mathbf{A}) = \mathsf{T}$ for all $a \in \mathcal{D}_\iota$

P.1.4.2 so in particular for some $a \in \mathcal{D}_\iota \neq \emptyset$. □

□

▷ **Corollary 3.101** $\mathcal{T}_1^f$ *is correct.*

SOME RIGHTS RESERVED ©: Michael Kohlhase 138 JACOBS UNIVERSITY

The only interesting steps are the cut rule, which can be directly handled by the substitution value lemma, and the rule for the existential quantifier, which we do in a separate lemma.

Soundness of $\mathcal{T}_1^f{:}\exists$

▷ **Lemma 3.102** $\mathcal{T}_1^f{:}\exists$ *transforms satisfiable tableaux into satisfiable ones.*

▷ Proof: Let $\mathcal{T}'$ be obtained by applying $\mathcal{T}_1^f{:}\exists$ to $\forall X.\mathbf{A}^\mathsf{f}$ in $\mathcal{T}$, extending it with $[f(X^1,\ldots,X^n)/X](\mathbf{A})^\mathsf{f}$, where $W := \mathbf{free}(\forall X.\mathbf{A}) = \{X^1,\ldots,X^k\}$

P.1 Let $\mathcal{T}$ be satisfiable in $\mathcal{M} := \langle\mathcal{D},\mathcal{I}\rangle$, then $\mathcal{I}_\varphi(\forall X.\mathbf{A}) = \mathsf{F}$.

P.2 We need to find a model $\mathcal{M}'$ that satisfies $\mathcal{T}'$ (find interpretation for f)

P.3 By definition $\mathcal{I}_{\varphi,[a/X]}(\mathbf{A}) = \mathsf{F}$ for some $a \in \mathcal{D}$ (depends on $\varphi|_W$)

P.4 Let $g\colon \mathcal{D}^k \to \mathcal{D}$ be defined by $g(a_1,\ldots,a_k) := a$, if $\varphi(X^i) = a_i$

P.5 choose $\mathcal{M}' = \langle\mathcal{D},\mathcal{I}'\rangle$ with $\mathcal{I}' := \mathcal{I},[g/f]$, then by subst. value lemma

$$\begin{aligned}\mathcal{I}'_\varphi([f(X^1,\ldots,X^k)/X](\mathbf{A})) &= \mathcal{I}'_{\varphi,[\mathcal{I}'_\varphi(f(X^1,\ldots,X^k))/X]}(\mathbf{A})\\ &= \mathcal{I}'_{\varphi,[a/X]}(\mathbf{A}) = \mathsf{F}\end{aligned}$$

P.6 So $[f(X^1,\ldots,X^k)/X](\mathbf{A})^\mathsf{f}$ satisfiable in $\mathcal{M}'$ □

SOME RIGHTS RESERVED ©: Michael Kohlhase 139 JACOBS UNIVERSITY

This proof is paradigmatic for soundness proofs for calculi with Skolemization. We use the axiom of choice at the meta-level to choose a meaning for the Skolem function symbol.

Armed with the Model Existence Theorem for first-order logic (Theorem 3.54), the completeness of first-order tableaux is similarly straightforward. We just have to show that the collection of tableau-irrefutable sentences is an abstract consistency class, which is a simple proof-transformation exercise in all but the universal quantifier case, which we postpone to its own Lemma.

Completeness of ($\mathcal{T}_1^f$)

▷ **Theorem 3.103** $\mathcal{T}_1^f$ *is refutation complete.*

▷ Proof: We show that $\nabla := \{\Phi \,|\, \Phi^\mathsf{T} \text{ has no closed Tableau}\}$ is an abstract consistency class

P.1 (∇_c, $\nabla_\neg$, $\nabla_\vee$, and $\nabla_\wedge$)as for propositional case.

P.2 ($\nabla_\forall$)by the lifting lemma below

P.3 ($\nabla_\exists$)Let $\mathcal{T}$ be a closed tableau for $\neg(\forall X.\mathbf{A}) \in \Phi$ and $\Phi^{\mathsf{T}} * [c/X](\mathbf{A})^{\mathsf{F}} \in \nabla$.

$$\begin{array}{c} \Psi^{\mathsf{T}} \\ \forall X.\mathbf{A}^{\mathsf{f}} \\ [c/X](\mathbf{A})^{\mathsf{f}} \\ Rest \end{array} \qquad \begin{array}{c} \Psi^{\mathsf{T}} \\ \forall X.\mathbf{A}^{\mathsf{f}} \\ [f(X^1,\ldots,X^k)/X](\mathbf{A})^{\mathsf{f}} \\ [f(X^1,\ldots,X^k)/c](Rest) \end{array}$$

□

So we only have to treat the case for the universal quantifier. This is what we usually call a "lifting argument", since we have to transform ("lift") a proof for a formula $\theta(\mathbf{A})$ to one for $\mathbf{A}$. In the case of tableaux we do that by an induction on the tableau refutation for $\theta(\mathbf{A})$ which creates a tableau-isomorphism to a tableau refutation for $\mathbf{A}$.

Tableau-Lifting

▷ **Theorem 3.104** *If $\mathcal{T}_\theta$ is a closed tableau for a st $\theta(\Phi)$ of formulae, then there is a closed tableau $\mathcal{T}$ for Φ.*

▷ Proof: by induction over the structure of $\mathcal{T}_\theta$ we build an isomorphic tableau $\mathcal{T}$, and a tableau-isomorphism $\omega\colon \mathcal{T} \to \mathcal{T}_\theta$, such that $\omega(\mathbf{A}) = \theta(\mathbf{A})$.

P.1 only the tableau-substitution rule is interesting.

P.2 Let $\theta(\mathbf{A}^i)^{\mathsf{t}}$ and $\theta(\mathbf{B}^i)^{\mathsf{f}}$ cut formulae in the branch Θ_θ^i of $\mathcal{T}_\theta$

P.3 there is a joint unifier σ of $\theta(\mathbf{A}^1) =^? \theta(\mathbf{B}^1) \wedge \ldots \wedge \theta(\mathbf{A}^n) =^? \theta(\mathbf{B}^n)$

P.4 thus $\sigma \circ \theta$ is a unifier of $\mathbf{A}$ and $\mathbf{B}$

P.5 hence there is a most general unifier ρ of $\mathbf{A}^1 =^? \mathbf{B}^1 \wedge \ldots \wedge \mathbf{A}^n =^? \mathbf{B}^n$

P.6 so Θ is closed □

Again, the "lifting lemma for tableaux" is paradigmatic for lifting lemmata for other refutation calculi.

3.5 Model Generation with Quantifiers

Since we have introduced new logical constants, we have to extend the model generation calculus by rules for these. To keep the calculus simple, we will treat $\exists X.\mathbf{A}$ as an abbreviation of $\neg(\forall X.\neg\mathbf{A})$. Thus we only have to treat the universal quantifier in the rules.

Model Generation (RM Calculus [Konrad'98])

▷ Idea: Try to generate domain-minimal (i.e. fewest individuals) models(for NL interpretation)

▷ Problem: Even one function symbol makes Herbrand base infinite (solution: leave them out)

▷ **Definition 3.105** Add ground quantifier rules to these

$$\frac{\forall X.\mathbf{A}^{\mathsf{t}} \quad c \in \mathcal{H}}{[c/X](\mathbf{A})^{\mathsf{t}}}\,\mathsf{RM{:}\forall} \qquad \frac{\forall X.\mathbf{A}^{\mathsf{f}} \quad \mathcal{H} = \{a_1, \ldots, a_n\} \quad w \notin \mathcal{H}\ \text{new}}{[a_1/X](\mathbf{A})^{\mathsf{f}} \;\big|\; \ldots \;\big|\; [a_n/X](\mathbf{A})^{\mathsf{f}} \;\big|\; [w/X](\mathbf{A})^{\mathsf{f}}}\,\mathsf{RM{:}\exists}$$

- ▷ RM:∃ rule introduces new witness constant w to Herbrand base $\mathcal{H}$ of branch
- ▷ Apply RM:∀ exhaustively (for new w reapply all RM:∀ rules on branch!)

SOME RIGHTS RESERVED ©: Michael Kohlhase 142 JACOBS UNIVERSITY

The rule RM:∀ allows to instantiate the scope of the quantifier with all the instances of the Herbrand base, whereas the rule RM:∃ makes a case distinction between the cases that the scope holds for one of the already known individuals (those in the Herbrand base) or a currently unknown one (for which it introduces a witness constant $w \in \Sigma_0^{sk}$).

Note that in order to have a complete calculus, it is necessary to apply the RM:∀ rule to all universal formulae in the tree with the new constant w. With this strategy, we arrive at a complete calculus for (finite) satisfiability in first-order logic, i.e. if a formula has a (finite) Model, then this calculus will find it. Note that this calculus (in this simple form) does not necessarily find minimal models.

Generating infinite models (Natural Numbers)

▷ We have to re-apply the RM:∀ rule for any new constant

▷ **Example 3.106** This leads to the generation of infinite modesl

$$\begin{array}{c}
\forall x.\neg x > x \wedge \ldots^{\mathsf{t}}\\
N(0)^{\mathsf{t}}\\
\forall x.N(x) \Rightarrow (\exists y.y > x)^{\mathsf{t}}\\
N(0) \Rightarrow (\exists y.y > 0)^{\mathsf{t}}\\
\begin{array}{c|c}
\begin{array}{c} N(0)^{\mathsf{f}}\\ \bot \end{array} &
\begin{array}{c}
\exists y.y > 0^{\mathsf{t}}\\
\begin{array}{c|c}
\begin{array}{c} 0 > 0^{\mathsf{t}}\\ 0 > 0^{\mathsf{f}}\\ \bot \end{array} &
\begin{array}{c}
1^{\mathsf{t}} > 0\\
N(1) \Rightarrow (\exists y.y > 1)^{\mathsf{t}}\\
\begin{array}{c|c}
\begin{array}{c} N(1)^{\mathsf{f}}\\ \vdots\\ \bot \end{array} &
\begin{array}{c}
\exists y.y > 1^{\mathsf{t}}\\
\begin{array}{c|c|c}
\begin{array}{c} 0 > 1^{\mathsf{t}}\\ \vdots\\ \bot \end{array} &
\begin{array}{c} 1 > 1^{\mathsf{t}}\\ 1 > 2^{\mathsf{t}}\\ \bot \end{array} &
\begin{array}{c} 2 > 1^{\mathsf{t}}\\ \vdots \end{array}
\end{array}
\end{array}
\end{array}
\end{array}
\end{array}
\end{array}
\end{array}
\end{array}$$

SOME RIGHTS RESERVED ©: Michael Kohlhase 143 JACOBS UNIVERSITY

The rules RM:∀ and RM:∃ may remind you of the rules we introduced for $\mathrm{PL}_{\mathrm{NQ}}^{\mathcal{V}}$. In fact the rules mainly differ in their scoping behavior. We will use RM:∀ as a drop-in replacement for the world-knowledge rule $\mathcal{T}_{\mathcal{V}}^{p}$:WK, and express world knowledge as universally quantified sentences. The rules $\mathcal{T}_{\mathcal{V}}^{p}$:Ana and RM:∃ differ in that the first may only be applied to input formulae and does not introduce a witness constant. (It should not, since variables here are anaphoric). We need the rule RM:∃ to deal with rule-like world knowledge.

Example: *Peter is a man. No man walks*

without sorts

$\boxed{\mathsf{man}(\mathsf{peter})}$

$\boxed{\neg(\exists X.\mathsf{man}(X) \wedge \mathsf{walk}(X))}$

$\exists X.\mathsf{man}(X) \wedge \mathsf{walk}(X)^{\mathsf{f}}$

$\mathsf{man}(\mathsf{peter}) \wedge \mathsf{walk}(\mathsf{peter})^{\mathsf{f}}$

$\mathsf{man}(\mathsf{peter})^{\mathsf{f}} \mid \mathsf{walk}(\mathsf{peter})^{\mathsf{f}}$

$\bot$

problem: 1000 men

$\Rightarrow$

1000 closed branches

with sort $\mathbb{M}\mathtt{ale}$

$\boxed{\mathsf{man}(\mathsf{peter})}$

$\boxed{\neg(\exists X_{\mathbb{M}\mathtt{ale}}.\mathsf{walk}(X))}$

$\exists X_{\mathbb{M}\mathtt{ale}}.\mathsf{walk}(X)^{\mathsf{f}}$

$\mathsf{walk}(\mathsf{peter})^{\mathsf{f}}$

▷ Herbrand-model

$$\{\mathsf{man}(\mathsf{peter})^{\mathsf{t}}, \mathsf{walk}(\mathsf{peter})^{\mathsf{f}}\}$$

Anaphor resolution *A man sleeps. He snores*

$\boxed{\exists X.\mathsf{man}(X) \wedge \mathsf{sleep}(X)}$

$\mathsf{man}(c^1_{\mathbb{M}\mathrm{an}})^{\mathsf{t}}$

$\mathsf{sleep}(c^1_{\mathbb{M}\mathrm{an}})^{\mathsf{t}}$

$\boxed{\exists Y_{\mathbb{M}\mathrm{an}}.\mathsf{snore}(Y)}$

$\swarrow \qquad \searrow$

$\mathsf{snore}(c^1_{\mathbb{M}\mathrm{an}})^{\mathsf{t}}$ | $\mathsf{snore}(c^2_{\mathbb{M}\mathrm{an}})^{\mathsf{t}}$

minimal | deictic

In a situation without men (but maybe thousands of women)

Anaphora with World Knowledge

▷ *Mary is married to Jeff. Her husband is not in town.*

▷ $\exists U_{\mathbb{F}\mathtt{emale}}, V_{\mathbb{M}\mathtt{ale}}.U = \mathsf{mary} \wedge \mathsf{married}(U,V) \wedge V = \mathsf{jeff} \wedge (\exists W_{\mathbb{M}\mathtt{ale}}, W'_{\mathbb{F}\mathtt{emale}}.\mathsf{hubby}(W,W')\neg\mathsf{intown}(W))$

▷ World knowledge

- ▹ if woman X is married to man Y, then Y is the only husband of X.
- ▹ $\forall X_{\mathbb{F}\mathtt{emale}}, Y_{\mathbb{M}\mathtt{ale}}.\mathsf{married}(X,Y) \Rightarrow \mathsf{hubby}(Y,X) \wedge (\forall Z.\mathsf{hubby}(Z,X) \Rightarrow Z \doteq Y)$

▷ model generation gives tableau, all branches contain

$$\{\mathsf{married}(\mathsf{mary},\mathsf{jeff})^{\mathsf{t}}, \mathsf{hubby}(\mathsf{jeff},\mathsf{mary})^{\mathsf{t}}\mathsf{intown}(\mathsf{jeff})^{\mathsf{f}}\}$$

▷ Differences: additional negative facts e.g. $\mathsf{married}(\mathsf{mary},\mathsf{mary})^{\mathsf{f}}$.

A branch without world knowledge

$$\begin{array}{c}
\exists X_{\mathtt{Female}}, Y_{\mathtt{Male}}. X \doteq \mathsf{mary} \wedge \mathsf{married}(X,Y) \wedge Y \doteq \mathsf{jeff}^{\mathsf{t}} \\
\exists Y.\mathsf{mary} \doteq \mathsf{mary} \wedge \mathsf{married}(\mathsf{mary},Y) \wedge Y \doteq \mathsf{jeff}^{\mathsf{t}} \\
\mathsf{mary} \doteq \mathsf{mary}, \mathsf{married}(\mathsf{mary},\mathsf{jeff}), \mathsf{jeff} \doteq \mathsf{jeff}^{\mathsf{t}} \\
\mathsf{mary} \doteq \mathsf{mary}^{\mathsf{t}} \\
\mathsf{married}(\mathsf{mary},\mathsf{jeff})^{\mathsf{t}} \\
\mathsf{jeff} \doteq \mathsf{jeff}^{\mathsf{t}} \\
\exists Z_{\mathtt{Male}}, Z'_{\mathtt{Female}}.\mathsf{hubby}(Z,Z') \wedge \neg\mathsf{intown}(Z)^{\mathsf{t}} \\
\exists Z'.\mathsf{hubby}(c^1_{\mathtt{Male}},Z') \wedge \neg\mathsf{intown}(c^1_{\mathtt{Male}})^{\mathsf{t}} \\
\mathsf{hubby}(c^1_{\mathtt{Male}},\mathsf{mary}) \wedge \neg\mathsf{intown}(c^1_{\mathtt{Male}})^{\mathsf{t}} \\
\mathsf{hubby}(c^1_{\mathtt{Male}},\mathsf{mary})^{\mathsf{t}} \\
\neg\mathsf{intown}(c^1_{\mathtt{Male}})^{\mathsf{t}} \\
\mathsf{intown}(c^1_{\mathtt{Male}})^{\mathsf{f}}
\end{array}$$

- ▷ Problem: Bigamy
- ▷ $c^1_{\mathtt{Male}}$ and jeff husbands of *Mary*!

©: Michael Kohlhase 147

4 Fragment 3: Complex Verb Phrases

4.1 Fragment 3 (Handling Verb Phrases)

New Data (Verb Phrases)

- ▷ *Ethel howled and screamed.*
- ▷ *Ethel kicked the dog and poisoned the cat.*
- ▷ *Fiona liked Jo and loathed Ethel and tolerated Prudence.*
- ▷ *Fiona kicked the cat and laughed.*
- ▷ *Prudence kicked and scratched Ethel.*
- ▷ *Bertie didn't laugh.*
- ▷ *Bertie didn't laugh and didn't scream.*
- ▷ *Bertie didn't laugh or scream.*
- ▷ *Bertie didn't laugh or kick the dog.*

©: Michael Kohlhase 148

New Grammar in Fragment 3 (Verb Phrases)

- ▷ To account for the syntax we come up with the concept of a verb-phrase (VP)
- ▷ **Definition 4.1** $\mathcal{F}_3$ has the following rules:

S1.	S	$\to$	NPVP_{+fin}
S2.	S	$\to$	Sconj$S$
V1.	$VP_{\pm fin}$	$\to$	${V^i}_{\pm fin}$
V2.	$VP_{\pm fin}$	$\to$	${V^t}_{\pm fin}$, NP
V3.	$VP_{\pm fin}$	$\to$	$VP_{\pm fin}$, conj, $VP_{\pm fin}$
V4.	VP_{+fin}	$\to$	$BE_{=}$, NP
V5.	VP_{+fin}	$\to$	BE_{pred}, Adj.
V6.	VP_{+fin}	$\to$	didn't VP_{-fin}
N1.	NP	$\to$	N_{pr}
N2.	NP	$\to$	Pron
N3.	NP	$\to$	the N

L8.	$BE_{=}$	$\to$	{is}
L9.	BE_{pred}	$\to$	{is}
L10.	${V^i}_{-fin}$	$\to$	{run, laugh, sing,...}
L11.	${V^t}_{-fin}$	$\to$	{read, poison,eat,...}

▷ Limitations of $\mathcal{F}_3$:

- ▷ The rule for *didn't* over-generates: **John didn't didn't run* (need tense for that)
- ▷ $\mathcal{F}_3$ does not allow coordination of transitive verbs (problematic anyways)

SOME RIGHTS RESERVED ©: Michael Kohlhase 149 JACOBS UNIVERSITY

The main extension of the fragment is the introduction of the new category *VP*, we have to interpret. Intuitively, *VP*s denote functions that can be applied to the NP meanings (rule 1). Complex *VP* functions can be constructed from simpler ones by NL connectives acting as functional operators.

Given the discussion above, we have to deal with various kinds of functions in the semantics. NP meanings are individuals, *VP* meanings are functions from individuals to individuals, and conj meanings are functionals that map functions to functions. It is a tradition in logic to distinguish such objects (individuals and functions of various kinds) by assigning them types.

4.2 Dealing with Functions in Logic and Language

So we need to have a logic that can deal with functions and functionals (i.e. functions that construct new functions from existing ones) natively. This goes beyond the realm of first-order logic we have studied so far. We need two things from this logic:

1. a way of distinguishing the respective individuals, functions and functionals, and
2. a way of constructing functions from individuals and other functions.

There are standard ways of achieving both, which we will combine in the following to get the "simply typed lambda calculus" which will be the workhorse logic for $\mathcal{F}_3$.

The standard way for distinguishing objects of different levels is by introducing types, here we can get by with a very simple type system that only distinguishes functions from their arguments

Types

▷ Types are semantic annotations for terms that prevent antinomies

▷ **Definition 4.2** Given a set $\mathcal{BT}$ of base type s, construct function type s: $\alpha \to \beta$ is the type of functions with domain type α and range type β. We call the closure $\mathcal{T}$ of $\mathcal{BT}$ under function types the set of type s over $\mathcal{BT}$.

▷ **Definition 4.3** We will use ι for the type of individuals and o for the type of truth values.

▷ The type constructor is used as a right-associative operator, i.e. we use $\alpha \to \beta \to \gamma$ as an abbreviation for $\alpha \to (\beta \to \gamma)$

▷ We will use a kind of vector notation for function types, abbreviating $\alpha_1 \to \ldots \to \alpha_n \to \beta$ with $\overline{\alpha_n} \to \beta$.

SOME RIGHTS RESERVED ©: Michael Kohlhase 150 JACOBS UNIVERSITY

Syntactical Categories and Types

▷ Now, we can assign types to syntactical categories.

Cat	Type	Intuition
S	o	truth value
NP	ι	individual
N_{pr}	ι	individuals
VP	$\iota \to o$	property
V^i	$\iota \to o$	unary predicate
V^t	$\iota \to \iota \to o$	binary relation

▷ For the category conj, we cannot get by with a single type. Depending on where it is used, we need the types

- ▷ $o \to o \to o$ for S-coordination in rule $S2$: $S \to S, \mathsf{conj}, S$
- ▷ $(\iota \to o) \to (\iota \to o) \to (\iota \to o)$ for VP-coordination in $V3$: $VP \to VP, \mathsf{conj}, VP$.
- ▷ Note: Computational Linguistics, often uses a different notation for types: e for ι, t for o, and $\langle \alpha, \beta \rangle$ for $\alpha \to \beta$ (no bracket elision convention). So the type for VP-coordination has the form $\langle \langle e, t \rangle, \langle \langle e, t \rangle, \langle e, t \rangle \rangle \rangle$

SOME RIGHTS RESERVED ©: Michael Kohlhase 151 JACOBS UNIVERSITY

For a logic which can really deal with functions, we have to have two properties, which we can already read off the language of mathematics (as the discipine that deals with functions and funcitonals professionally): We

1. need to be able to construct functions from expressions with variables, as in $f(x) = 3x^2 + 7x + 5$, and
2. consider two functions the same, iff the return the same values on the same arguments.

In a logical system (let us for the moment assume a first-order logic with types that can quantify over functions) this gives rise to the following axioms:

Comprehension $\exists F_{\alpha \to \beta}. \forall X_\alpha. FX = \mathbf{A}_\beta$

Extensionality $\forall F_{\alpha \to \beta}. \forall G_{\alpha \to \beta}. (\forall X_\alpha. FX = GX) \Rightarrow F = G$

The comprehension axioms are computationally very problematic. First, we observe that they are equality axioms, and thus are needed to show that two objects of PLΩ are equal. Second we observe that there are countably infinitely many of them (they are parametric in the term $\mathbf{A}$, the type α and the variable name), which makes dealing with them difficult in practice. Finally, axioms with both existential and universal quantifiers are always difficult to reason with.

Therefore we would like to have a formulation of higher-order logic without comprehension axioms. In the next slide we take a close look at the comprehension axioms and transform them into a form without quantifiers, which will turn out useful.

From Comprehension to β-Conversion

▷ $\exists F_{\alpha\to\beta}.\forall X_\alpha.FX = \mathbf{A}_\beta$ for arbitrary variable X_α and term $\mathbf{A} \in \mathit{wff}_\beta(\Sigma, \mathcal{V}_\mathcal{T})$ (for each term $\mathbf{A}$ and each variable X there is a function $f \in \mathcal{D}_{\alpha\to\beta}$, with $f(\varphi(X)) = \mathcal{I}_\varphi(\mathbf{A})$)

- ▷ schematic in α, β, X_α and $\mathbf{A}_\beta$, very inconvenient for deduction

▷ Transformation in $\mathcal{H}_\Omega$

- ▷ $\exists F_{\alpha\to\beta}.\forall X_\alpha.FX = \mathbf{A}_\beta$
- ▷ $\forall X_\alpha.(\lambda X_\alpha.\mathbf{A})X = \mathbf{A}_\beta$ ($\exists E$)
  Call the function F whose existence is guaranteed "$(\lambda X_\alpha.\mathbf{A})$"
- ▷ $(\lambda X_\alpha.\mathbf{A})\mathbf{B} = [\mathbf{B}/X]\mathbf{A}_\beta$ ($\forall E$), in particular for $\mathbf{B} \in \mathit{wff}_\alpha(\Sigma, \mathcal{V}_\mathcal{T})$.

▷ **Definition 4.4** Axiom of β-equality: $(\lambda X_\alpha.\mathbf{A})\mathbf{B} = [\mathbf{B}/X](\mathbf{A}_\beta)$

▷ new formulae (λ-calculus [Church 1940])

In a similar way we can treat (functional) extensionality.

From Extensionality to η-Conversion

▷ **Definition 4.5** Extensionality Axiom: $\forall F_{\alpha\to\beta}.\forall G_{\alpha\to\beta}.(\forall X_\alpha.FX = GX) \Rightarrow F = G$

▷ Idea: Maybe we can get by with a simplified equality schema here as well.

▷ **Definition 4.6** We say that $\mathbf{A}$ and $\lambda X_\alpha.\mathbf{A}X$ are η-equal, (write $\mathbf{A}_{\alpha\to\beta} =_\eta \lambda X_\alpha.\mathbf{A}X$, if), iff $X \notin \mathbf{free}(\mathbf{A})$.

▷ **Theorem 4.7** *η-equality and Extensionality are equivalent*

▷ Proof: We show that η-equality is special case of extensionality; the converse entailment is trivial

P.1 Let $\forall X_\alpha.\mathbf{A}X = \mathbf{B}X$, thus $\mathbf{A}X = \mathbf{B}X$ with $\forall E$

P.2 $\lambda X_\alpha.\mathbf{A}X = \lambda X_\alpha.\mathbf{B}X$, therefore $\mathbf{A} = \mathbf{B}$ with η

P.3 Hence $\forall F_{\alpha\to\beta}.\forall G_{\alpha\to\beta}.(\forall X_\alpha.FX = GX) \Rightarrow F = G$ by twice $\forall I$. ☐

▷ Axiom of truth values: $\forall F_o.\forall G_o.(F \Leftrightarrow G) \Leftrightarrow F = G$ unsolved.

The price to pay is that we need to pay for getting rid of the comprehension and extensionality axioms is that we need a logic that systematically includes the λ-generated names we used in the transformation as (generic) witnesses for the existential quantifier. Alonzo Church did just that with his "simply typed λ-calculus" which we will introduce next.

This is all very nice, but what do we actually translate into?

4.3 Translation for Fragment 3

4.3.1 Translation from $\mathcal{F}_3$ into $\Lambda^{\to}$

Translations for Fragment 3

▷ We will look at the new translation rules (the rest stay the same).

T1	$[X_{\mathsf{NP}}, Y_{VP}]_S$	$\Longrightarrow$	$VP'(\mathsf{NP}')$
T3	$[X_{VP}, Y_{\mathsf{conj}}, Z_{VP}]_{VP}$	$\Longrightarrow$	$\mathsf{conj}'(VP', VP')$
T4	$[X_{V^t}, Y_{\mathsf{NP}}]_{VP}$	$\Longrightarrow$	$V^{t\prime}(\mathsf{NP}')$

▷ The lexical insertion rules will give us two items each for *is*, *and*, and *or*, corresponding to the two types we have given them.

word	type	term	case
BE_{pred}	$(\iota \to o) \to \iota \to o$	$\lambda P_{\iota \to o}.P$	adjective
BE_{eq}	$\iota \to \iota \to o$	$\lambda X_\iota Y_\iota.X = Y$	verb
and	$o \to o \to o$	$\wedge$	S-coord.
and	$(\iota \to o) \to (\iota \to o) \to \iota \to o$	$\lambda F_{\iota \to o} G_{\iota \to o} X_\iota.F(X) \wedge G(X)$	VP-coord.
or	$o \to o \to o$	$\vee$	S-coord.
or	$(\iota \to o) \to (\iota \to o) \to \iota \to o$	$\lambda F_{\iota \to o} G_{\iota \to o} X_\iota.F(X) \vee G(X)$	VP-coord.
didn't	$(\iota \to o) \to \iota \to o$	$\lambda P_{\iota \to o} X_\iota.\neg(PX)$	

Need to assume the logical connectives as constants of the λ-calculus.

▷ Note: With these definitions, it is easy to restrict ourselves to binary branching in the syntax of the fragment.

SOME RIGHTS RESERVED ©: Michael Kohlhase 154 JACOBS UNIVERSITY

- **Definition 4.8 (Translation of non-branching nodes)** If φ is a non-branching node with daughter ψ, then the translation φ' of φ is given by the translation ψ' of ψ.

- **Definition 4.9 (Translation of branching nodes (Function Application))** If φ is a branching node with daughters ψ and θ, where ψ' is an expression of type $\alpha \to \beta$ and θ' is an expression of type α, then $\varphi' = \psi'\theta'$.

- Note on notation: We now have higher-order constants formed using words from the fragment, which are not (or are not always) translations of the words from which they are formed. We thus need some new notation to represent the translation of an expression from the fragment. We will use the notation introduced above, i.e. $john'$ is the translation of the word *John*. We will continue to use primes to indicate that something is an expression (e.g. john). Words of the fragment of English should be either underlined or italicized.

Translation Example

▷ **Example 4.10** *Ethel howled and screamed* to

$$
\begin{array}{ll}
 & (\lambda F_{\iota\to o} G_{\iota\to o} X_\iota.F(X) \wedge G(X))\mathsf{howl}\,\mathsf{scream}\,\mathsf{ethel} \\
\to_\beta & (\lambda G_{\iota\to o} X_\iota.\mathsf{howl}(X) \wedge G(X))\mathsf{scream}\,\mathsf{ethel} \\
\to_\beta & (\lambda X_\iota.\mathsf{howl}(X) \wedge \mathsf{scream}(X))\mathsf{ethel} \\
\to_\beta & \mathsf{howl}(\mathsf{ethel}) \wedge \mathsf{scream}(\mathsf{ethel})
\end{array}
$$

Higher-Order Logic without Quantifiers (HOL_{NQ})

▷ Problem: Need a logic like PL_{NQ}, but with λ-terms to interpret $\mathcal{F}_3$ into.

▷ Idea: Re-use the syntactical framework of $\Lambda^{\rightarrow}$.

▷ **Definition 4.11** Let HOL_{NQ} be an instance of $\Lambda^{\rightarrow}$, with $\mathcal{BT} = \{\iota, o\}$, $\wedge \in \Sigma_{o\rightarrow o\rightarrow o}$, $\neg \in \Sigma_{o\rightarrow o}$, and $= \in \Sigma_{\alpha\rightarrow\alpha\rightarrow o}$ for all types α.

▷ Idea: To extend this to a semantics for HOL_{NQ}, we only have to say something about the base type o, and the logical constants $\neg_{o\rightarrow o}$, $\wedge_{o\rightarrow o\rightarrow o}$, and $=_{\alpha\rightarrow\alpha\rightarrow o}$.

▷ **Definition 4.12** We define the semantics of HOL_{NQ} by setting

1. $\mathcal{D}_o = \{\mathsf{T}, \mathsf{F}\}$; the set of truth values
2. $\mathcal{I}(\neg) \in \mathcal{D}_{(o\rightarrow o)}$, is the function $\{\mathsf{F} \mapsto \mathsf{T}, \mathsf{T} \mapsto \mathsf{F}\}$
3. $\mathcal{I}(\wedge) \in \mathcal{D}_{(o\rightarrow o\rightarrow o)}$ is the function with $\mathcal{I}(\wedge)@\langle \mathsf{a}, \mathsf{b}\rangle = \mathsf{T}$, iff $\mathsf{a} = \mathsf{T}$ and $\mathsf{b} = \mathsf{T}$.
4. $\mathcal{I}(=) \in \mathcal{D}_{(\alpha\rightarrow\alpha\rightarrow o)}$ is the identity relation on $\mathcal{D}_\alpha$.

You may be worrying that we have changed our assumptions about the denotations of predicates. When we were working with PL_{NQ} as our translation language, we assumed that one-place predicates denote sets of individuals, that two-place predicates denote sets of pairs of individuals, and so on. Now, we have adopted a new translation language, HOL_{NQ}, which interprets all predicates as functions of one kind or another.

The reason we can do this is that there is a systematic relation between the functions we now assume as denotations, and the sets we used to assume as denotations. The functions in question are the *characteristic functions* of the old sets, or are curried versions of such functions.

Recall that we have characterized sets extensionally, i.e. by saying what their members are. A characteristic function of a set A is a function which "says" which objects are members of A. It does this by giving one value (for our purposes, the value 1) for any argument which is a member of A, and another value, (for our purposes, the value 0), for anything which is not a member of the set.

Definition 4.13 (Characteristic function of a set) f_S is the characteristic function of the set S iff $f_S(a) = \mathsf{T}$ if $\mathsf{a} \in S$ and $f_S(a) = \mathsf{F}$ if $\mathsf{a} \notin S$.

Thus any function in $\mathcal{D}_{\iota\rightarrow o}$ will be the characteristic function of some set of individuals. So, for example, the function we assign as denotation to the predicate *run* will return the value T for some arguments and F for the rest. Those for which it returns T correspond exactly to the individuals which belonged to the set *run* in our old way of doing things.

Now, consider functions in $\mathcal{D}_{\iota\rightarrow\iota\rightarrow o}$. Recall that these functions are equivalent to two-place relations, i.e. functions from pairs of entities to truth values. So functions of this kind are characteristic functions of sets of pairs of individuals.

In fact, any function which ultimately maps an argument to $\mathcal{D}_o$ is a characteristic function of some set. The fact that many of the denotations we are concerned with turn out to be characteristic functions of sets will be very useful for us, as it will allow us to go backwards and forwards between "set talk" and "function talk," depending on which is easier to use for what we want to say.

4.4 Simply Typed λ-Calculus

In this section we will present a logic that can deal with functions – the simply typed λ-calculus. It is a typed logic, so everything we write down is typed (even if we do not always write the types down).

Simply typed λ-Calculus (Syntax)

- Signature $\Sigma = \bigcup_{\alpha \in \mathcal{T}} \Sigma_\alpha$ (includes countably infinite Signatures Σ_α^{Sk} of Skolem contant s).
- $\mathcal{V}_\mathcal{T} = \bigcup_{\alpha \in \mathcal{T}} \mathcal{V}_\alpha$, such that $\mathcal{V}_\alpha$ are countably infinite
- **Definition 4.14** We call the set $wff_\alpha(\Sigma, \mathcal{V}_\mathcal{T})$ defined by the rules
  - $\mathcal{V}_\alpha \cup \Sigma_\alpha \subseteq wff_\alpha(\Sigma, \mathcal{V}_\mathcal{T})$
  - If $\mathbf{C} \in wff_{\alpha\to\beta}(\Sigma, \mathcal{V}_\mathcal{T})$ and $\mathbf{A} \in wff_\alpha(\Sigma, \mathcal{V}_\mathcal{T})$, then $(\mathbf{CA}) \in wff_\beta(\Sigma, \mathcal{V}_\mathcal{T})$
  - If $\mathbf{A} \in wff_\alpha(\Sigma, \mathcal{V}_\mathcal{T})$, then $(\lambda X_\beta.\mathbf{A}) \in wff_{\beta\to\alpha}(\Sigma, \mathcal{V}_\mathcal{T})$

  the set of well-typed formula e of type α over the signature Σ and use $wff_\mathcal{T}(\Sigma, \mathcal{V}_\mathcal{T}) := \bigcup_{\alpha \in \mathcal{T}} wff_\alpha(\Sigma, \mathcal{V}_\mathcal{T})$ for the set of all well-typed formulae.
- **Definition 4.15** We will call all occurrences of the variable X in $\mathbf{A}$ bound in $\lambda X.\mathbf{A}$. Variables that are not bound in $\mathbf{B}$ are called free in $\mathbf{B}$.
- Substitutions are well-typed, i.e. $\sigma(X_\alpha) \in wff_\alpha(\Sigma, \mathcal{V}_\mathcal{T})$ and capture-avoiding.
- **Definition 4.16 (Simply Typed λ-Calculus)** The simply typed λ-calculus $\Lambda^\to$ over a signature Σ has the formulae $wff_\mathcal{T}(\Sigma, \mathcal{V}_\mathcal{T})$ (they are called λ-terms) and the following equalities:
  - α conversion: $\lambda X.\mathbf{A} =_\alpha \lambda Y.[Y/X](\mathbf{A})$
  - β conversion: $(\lambda X.\mathbf{A})\mathbf{B} =_\beta [\mathbf{B}/X](\mathbf{A})$
  - η conversion: $\lambda X.\mathbf{A}X =_\eta \mathbf{A}$

SOME RIGHTS RESERVED ©: Michael Kohlhase 157 JACOBS UNIVERSITY

The intuitions about functional structure of λ-terms and about free and bound variables are encoded into three transformation rules $\Lambda^\to$: The first rule (α-conversion) just says that we can rename bound variables as we like. β-conversion codifies the intuition behind function application by replacing bound variables with argument. The equality relation induced by the η-reduction is a special case of the extensionality principle for functions ($f = g$ iff $f(a) = g(a)$ for all possible arguments a): If we apply both sides of the transformation to the same argument – say $\mathbf{B}$ and then we arrive at the right hand side, since $\lambda X_\alpha.\mathbf{A}X\mathbf{B} =_\beta \mathbf{AB}$.

We will use a set of bracket elision rules that make the syntax of $\Lambda^\to$ more palatable. This makes $\Lambda^\to$ expressions look much more like regular mathematical notation, but hides the internal structure. Readers should make sure that they can always reconstruct the brackets to make sense of the syntactic notions below.

Simply typed λ-Calculus (Notations)

- **Notation 4.17 (Application is left-associative)** We abbreviate $(((\mathbf{FA}^1)\mathbf{A}^2)\ldots)\mathbf{A}^n$ with $\mathbf{FA}^1\ldots\mathbf{A}^n$ eliding the brackets and further with $\mathbf{F}\overline{\mathbf{A}^n}$ in a kind of vector notation.

▷ A . stands for a left bracket whose partner is as far right as is consistent with existing brackets; i.e. $\mathbf{A}.\mathbf{BC}$ abbreviates $\mathbf{A}(\mathbf{BC})$.

▷ **Notation 4.18 (Abstraction is right-associative)** We abbreviate $\lambda X^1.\lambda X^2.\cdots\lambda X^n.\mathbf{A}\cdots$ with $\lambda X^1\ldots X^n.\mathbf{A}$ eliding brackets, and further to $\lambda\overline{X^n}.\mathbf{A}$ in a kind of vector notation.

▷ **Notation 4.19 (Outer brackets)** Finally, we allow ourselves to elide outer brackets where they can be inferred.

SOME RIGHTS RESERVED ©: Michael Kohlhase 158 JACOBS UNIVERSITY

Intuitively, $\lambda X.\mathbf{A}$ is the function f, such that $f(\mathbf{B})$ will yield $\mathbf{A}$, where all occurrences of the formal parameter X are replaced by $\mathbf{B}$.[31] EdN:31

In this presentation of the simply typed λ-calculus we build-in α-equality and use capture-avoiding substitutions directly. A clean introduction would followed the steps in Subsection 3.1 by introducing substitutions with a substitutability condition like the one in Definition 3.29, then establishing the soundness of α conversion, and only then postulating defining capture-avoiding substitution application as in Definition 3.34. The development for $\Lambda^{\rightarrow}$ is directly parallel to the one for PL^1, so we leave it as an exercise to the reader and turn to the computational properties of the λ-calculus.

Computationally, the λ-calculus obtains much of its power from the fact that two of its three equalities can be oriented into a reduction system. Intuitively, we only use the equalities in one direction, i.e. in one that makes the terms "simpler". If this terminates (and is confluent), then we can establish equality of two λ-terms by reducing them to normal forms and comparing them structurally. This gives us a decision procedure for equality. Indeed, we have these properties in $\Lambda^{\rightarrow}$ as we will see below.

$\alpha\beta\eta$-Equality (Overview)

▷ reduction with $\left\{\begin{array}{l}\beta : (\lambda X.\mathbf{A})\mathbf{B} \rightarrow_{\beta} [\mathbf{B}/X](\mathbf{A}) \\ \eta : \lambda X.\mathbf{A}X \rightarrow_{\eta} \mathbf{A}\end{array}\right.$ under $=_{\alpha}$: $\begin{array}{c}\lambda X.\mathbf{A} \\ =_{\alpha} \\ \lambda Y.[Y/X](\mathbf{A})\end{array}$

▷ **Theorem 4.20** *$\beta\eta$-reduction is well-typed, terminating and confluent in the presence of $=_{\alpha}$-conversion.*

▷ **Definition 4.21 (Normal Form)** We call a λ-term $\mathbf{A}$ a normal form (in a reduction system $\mathcal{E}$), iff no rule (from $\mathcal{E}$) can be applied to $\mathbf{A}$.

▷ **Corollary 4.22** *$\beta\eta$-reduction yields unique normal forms (up to α-equivalence).*

SOME RIGHTS RESERVED ©: Michael Kohlhase 159 JACOBS UNIVERSITY

We will now introduce some terminology to be able to talk about λ-terms and their parts.

Syntactic Parts of λ-Terms

▷ **Definition 4.23 (Parts of λ-Terms)** We can always write a λ-term in the form $\mathbf{T} = \lambda X^1\ldots X^k.\mathbf{H}\mathbf{A}^1\ldots\mathbf{A}^n$, where $\mathbf{H}$ is not an application. We call

[31] EdNote: rationalize the semantic macros for syntax!

- ▷ $\mathbf{H}$ the syntactic head of $\mathbf{T}$
- ▷ $\mathbf{H}\mathbf{A}^1 \ldots \mathbf{A}^n$ the matrix of $\mathbf{T}$, and
- ▷ $\lambda X^1 \ldots X^k.$ (or the sequence $X_1, \ldots, X_k$) the binder of $\mathbf{T}$

▷ **Definition 4.24** Head Reduction always has a unique β redex

$$\lambda\overline{X^n}.(\lambda Y.\mathbf{A})\mathbf{B}^1 \ldots \mathbf{B}^n \to_\beta^h \lambda\overline{X^n}.[\mathbf{B}^1/Y](\mathbf{A})\mathbf{B}^2 \ldots \mathbf{B}^n$$

▷ **Theorem 4.25** *The syntactic heads of β-normal forms are constant or variables.*

▷ **Definition 4.26** Let $\mathbf{A}$ be a λ-term, then the syntactic head of the β-normal form of $\mathbf{A}$ is called the head symbol of $\mathbf{A}$ and written as head$(\mathbf{A})$. We call a λ-term a j-projection, iff its head is the j^{th} bound variable.

▷ **Definition 4.27** We call a λ-term a η-long form, iff its matrix has base type.

▷ **Definition 4.28** η-Expansion makes η-long forms

$$\eta\big[\lambda X^1 \ldots X^n.\mathbf{A}\big] := \lambda X^1 \ldots X^n.\lambda Y^1 \ldots Y^m.\mathbf{A}Y^1 \ldots Y^m$$

▷ **Definition 4.29** Long $\beta\eta$-normal form, iff it is β-normal and η-long.

SOME RIGHTS RESERVED ©: Michael Kohlhase 160 JACOBS UNIVERSITY

η long forms are structurally convenient since for them, the structure of the term is isomorphic to the structure of its type (argument types correspond to binders): if we have a term $\mathbf{A}$ of type $\overline{\alpha_n} \to \beta$ in η-long form, where $\beta \in \mathcal{BT}$, then $\mathbf{A}$ must be of the form $\lambda\overline{X_\alpha}^n.\mathbf{B}$, where $\mathbf{B}$ has type β. Furthermore, the set of η-long forms is closed under β-equality, which allows us to treat the two equality theories of $\Lambda^\to$ separately and thus reduce argumentational complexity.

4.5 Computational Properties of λ-Calculus

As we have seen above, the main contribution of the λ-calculus is that it casts the comprehension and (functional) extensionality axioms in a way that is more amenable to automation in reasoning systems, since they can be oriented into a confluent and terminating reduction system. In this subsection we prove the respective properties. We start out with termination, since we will need it later in the proof of confluence.

4.5.1 Termination of β-reduction

We will use the termination of β reduction to present a very powerful proof method, called the "logical relations method", which is one of the basic proof methods in the repertoire of a proof theorist, since it can be extended to many situations, where other proof methods have no chance of succeeding.

Before we start into the termination proof, we convince ourselves that a straightforward induction over the structure of expressions will not work, and we need something more powerful.

Termination of β-Reduction

▷ only holds for the typed case
$(\lambda X.XX)(\lambda X.XX) \to_\beta (\lambda X.XX)(\lambda X.XX)$

▷ **Theorem 4.30 (Typed β-Reduction terminates)** *For all $\mathbf{A} \in \mathit{wff}_\alpha(\Sigma, \mathcal{V}_\mathcal{T})$, the chain of reductions from $\mathbf{A}$ is finite.*

▷ proof attempts:

- ▷ Induction on the structure $\mathbf{A}$ must fail, since this would also work for the untyped case.
- ▷ Induction on the type of $\mathbf{A}$ must fail, since β-reduction conserves types.

▷ combined induction on both: Logical Relations [Tait 1967]

SOME RIGHTS RESERVED ©: Michael Kohlhase 161 JACOBS UNIVERSITY

The overall shape of the proof is that we reason about two relations: $\mathcal{SR}$ and $\mathcal{LR}$ between λ-terms and their types. The first is the one that we are interested in, $\mathcal{LR}(\mathbf{A}, \alpha)$ essentially states the property that $\beta\eta$ reduction terminates at $\mathbf{A}$. Whenever the proof needs to argue by induction on types it uses the "logical relation" $\mathcal{LR}$, which is more "semantic" in flavor. It coincides with $\mathcal{SR}$ on base types, but is defined via a functionality property.

Relations $\mathcal{SR}$ and $\mathcal{LR}$

▷ **Definition 4.31** $\mathbf{A}$ is called strongly reducing at type α (write $\mathcal{SR}(\mathbf{A}, \alpha)$), iff each chain β-reductions from $\mathbf{A}$ terminates.

▷ We define a logical relation$\mathcal{LR}$ inductively on the structure of the type

- ▷ α base type: $\mathcal{LR}(\mathbf{A}, \alpha)$, iff $\mathcal{SR}(\mathbf{A}, \alpha)$
- ▷ $\mathcal{LR}(\mathbf{C}, \alpha \to \beta)$, iff $\mathcal{LR}(\mathbf{CA}, \beta)$ for all $\mathbf{A} \in \mathit{wff}_\alpha(\Sigma, \mathcal{V}_\mathcal{T})$ with $\mathcal{LR}(\mathbf{A}, \alpha)$.

Proof: Termination Proof

▷ **P.1** $\mathcal{LR} \subseteq \mathcal{SR}$ (Lemma 4.33 b))

$\mathbf{A} \in \mathit{wff}_\alpha(\Sigma, \mathcal{V}_\mathcal{T})$ implies $\mathcal{LR}(\mathbf{A}, \alpha)$ (Theorem 4.37 with $\sigma = \emptyset$)

thus $\mathcal{SR}(\mathbf{A}, \alpha)$. □

P.2 **P.3**▷ **Lemma 4.32 ($\mathcal{SR}$ is closed under subterms)** *If $\mathcal{SR}(\mathbf{A}, \alpha)$ and $\mathbf{B}_\beta$ is a subterm of $\mathbf{A}$, then $\mathcal{SR}(\mathbf{B}, \beta)$.*

▷ Proof Idea: Every infinite β-reduction from $\mathbf{B}$ would be one from $\mathbf{A}$. □

SOME RIGHTS RESERVED ©: Michael Kohlhase 162 JACOBS UNIVERSITY

The termination proof proceeds in two steps, the first one shows that $\mathcal{LR}$ is a sub-relation of $\mathcal{SR}$, and the second that $\mathcal{LR}$ is total on λ-terms. Togther they give the termination result.

The next result proves two important technical side results for the termination proofs in a joint induction over the structure of the types involved. The name "rollercoaster lemma" alludes to the fact that the argument starts with base type, where things are simple, and iterates through the two parts each leveraging the proof of the other to higher and higher types.

$\mathcal{LR} \subseteq \mathcal{SR}$ (Rollercoaster Lemma)

▷ **Lemma 4.33 (Rollercoaster Lemma)**

a) *If h is a constant or variable of type $\overline{\alpha_n} \to \alpha$ and $\mathcal{SR}(\mathbf{A}^i, \alpha^i)$, then $\mathcal{LR}(h\overline{\mathbf{A}^n}, \alpha)$.*

b) *$\mathcal{LR}(\mathbf{A}, \alpha)$ implies $\mathcal{SR}(\mathbf{A}, \alpha)$.*

Proof: we prove both assertions by simultaneous induction on α

▷ **P.1.1** α base type:

P.1.1.1.1 a): $h\overline{\mathbf{A}^n}$ is strongly reducing, since the $\mathbf{A}^i$ are (brackets!)

P.1.1.1.1.2 so $\mathcal{LR}(h\overline{\mathbf{A}^n}, \alpha)$ as α is a base type ($\mathcal{SR} = \mathcal{LR}$) □

P.1.1.1.2 b): by definition □

$\alpha = \beta \to \gamma$:

P.1.2.1.1 a): Let $\mathcal{LR}(\mathbf{B}, \beta)$.

P.1.2.1.1.2 by IH b) we have $\mathcal{SR}(\mathbf{B}, \beta)$, and $\mathcal{LR}((h\overline{\mathbf{A}^n})\mathbf{B}, \gamma)$ by IH a)

P.1.2.1.1.3 so $\mathcal{LR}(h\overline{\mathbf{A}^n}, \alpha)$ by definition. □

P.1.2.1.2 b): Let $\mathcal{LR}(\mathbf{A}, \alpha)$ and $X_\beta \notin \mathbf{free}(\mathbf{A})$.

P.1.2.1.2.2 $\mathcal{LR}(X, \beta)$ by IH a) with $n = 0$, thus $\mathcal{LR}(\mathbf{A}X, \gamma)$ by definition.

P.1.2.1.2.3 By IH b) we have $\mathcal{SR}(\mathbf{A}X, \gamma)$ and by Lemma 4.32 $\mathcal{SR}(\mathbf{A}, \alpha)$. □

□

□

SOME RIGHTS RESERVED ©: Michael Kohlhase 163 JACOBS UNIVERSITY

The part of the rollercoaster lemma we are really interested in is part b). But part a) will become very important for the case where $n = 0$; here it states that constants and variables are $\mathcal{LR}$.

The next step in the proof is to show that all well-formed formulae are $\mathcal{LR}$. For that we need to prove closure of $\mathcal{LR}$ under $=_\beta$ expansion

β-Expansion Lemma

▷ **Lemma 4.34** *If $\mathcal{LR}([\mathbf{B}/X](\mathbf{A}), \alpha)$ and $\mathcal{LR}(\mathbf{B}, \beta)$ for $X_\beta \notin \mathbf{free}(\mathbf{B})$, then $\mathcal{LR}((\lambda X_\alpha.\mathbf{A})\mathbf{B}, \alpha)$.*

▷ Proof:

P.1 Let $\alpha = \overline{\gamma_i} \to \delta$ where δ base type and $\mathcal{LR}(\mathbf{C}^i, \gamma^i)$

P.2 It is sufficient to show that $\mathcal{SR}(((\lambda X.\mathbf{A})\mathbf{B})\overline{\mathbf{C}}, \delta)$, as δ base type

P.3 We have $\mathcal{LR}([\mathbf{B}/X](\mathbf{A})\overline{\mathbf{C}}, \delta)$ by hypothesis and definition of $\mathcal{LR}$.

P.4 thus $\mathcal{SR}([\mathbf{B}/X](\mathbf{A})\overline{\mathbf{C}}, \delta)$, as δ base type.

P.5 in particular $\mathcal{SR}([\mathbf{B}/X](\mathbf{A}), \alpha)$ and $\mathcal{SR}(\mathbf{C}^i, \gamma^i)$ (subterms)

P.6 $\mathcal{SR}(\mathbf{B}, \beta)$ by hypothesis and Lemma 4.33

P.7 So an infinite reduction from $((\lambda X.\mathbf{A})\mathbf{B})\overline{\mathbf{C}}$ cannot solely consist of redexes from $[\mathbf{B}/X](\mathbf{A})$ and the $\mathbf{C}^i$.

P.8 so an infinite reduction from $((\lambda X.\mathbf{A})\mathbf{B})\overline{\mathbf{C}}$ must have the form

$$\begin{array}{rcl} ((\lambda X.\mathbf{A})\mathbf{B})\overline{\mathbf{C}} & \rightarrow_\beta^* & ((\lambda X.\mathbf{A}')\mathbf{B}')\overline{\mathbf{C}'} \\ & \rightarrow_\beta^1 & [\mathbf{B}'/X](\mathbf{A}')\overline{\mathbf{C}'} \\ & \rightarrow_\beta^* & \ldots \end{array}$$

where $\mathbf{A} \rightarrow_\beta^* \mathbf{A}'$, $\mathbf{B} \rightarrow_\beta^* \mathbf{B}'$ and $\mathbf{C}^i \rightarrow_\beta^* \mathbf{C}^{i'}$

P.9 so we have $[\mathbf{B}/X](\mathbf{A}) \rightarrow_\beta^* [\mathbf{B}'/X](\mathbf{A}')$

P.10 so we have the infinite reduction

$$\begin{array}{rcl} [\mathbf{B}/X](\mathbf{A})\overline{\mathbf{C}} & \rightarrow_\beta^* & [\mathbf{B}'/X](\mathbf{A}')\overline{\mathbf{C}'} \\ & \rightarrow_\beta^* & \ldots \end{array}$$

which contracts our assumption □

▷ **Lemma 4.35 ($\mathcal{LR}$ is closed under β-expansion)**
If $\mathbf{C} \rightarrow_\beta \mathbf{D}$ and $\mathcal{LR}(\mathbf{D},\alpha)$, so is $\mathcal{LR}(\mathbf{C},\alpha)$.

SOME RIGHTS RESERVED ©: Michael Kohlhase 164 JACOBS UNIVERSITY

Note that this Lemma is one of the few places in the termination proof, where we actually look at the properties of $=_\beta$ reduction.

We now prove that every well-formed formula is related to its type by $\mathcal{LR}$. But we cannot prove this by a direct induction. In this case we have to strengthen the statement of the theorem – and thus the inductive hypothesis, so that we can make the step cases go through. This is common for non-trivial induction proofs. Here we show instead that *every instance* of a well-formed formula is related to its type by $\mathcal{LR}$; we will later only use this result for the cases of the empty substitution, but the stronger assertion allows a direct induction proof.

$\mathbf{A} \in wff_\alpha(\Sigma, \mathcal{V}_\mathcal{T})$ implies $\mathcal{LR}(\mathbf{A},\alpha)$

▷ **Definition 4.36** We write $\mathcal{LR}(\sigma)$ if $\mathcal{LR}(\sigma(X_\alpha),\alpha)$ for all $X \in \mathbf{supp}(\sigma)$.

▷ **Theorem 4.37** *If $\mathbf{A} \in wff_\alpha(\Sigma, \mathcal{V}_\mathcal{T})$, then $\mathcal{LR}(\sigma(\mathbf{A}),\alpha)$ for any substitution σ with $\mathcal{LR}(\sigma)$.*

▷ Proof: by induction on the structure of $\mathbf{A}$

P.1.1 $\mathbf{A} = X_\alpha \in \mathbf{supp}(\sigma)$: then $\mathcal{LR}(\sigma(\mathbf{A}),\alpha)$ by assumption

P.1.2 $\mathbf{A} = X \notin \mathbf{supp}(\sigma)$: then $\sigma(\mathbf{A}) = \mathbf{A}$ and $\mathcal{LR}(\mathbf{A},\alpha)$ by Lemma 4.33 with $n = 0$.

P.1.3 $\mathbf{A} \in \Sigma$: then $\sigma(\mathbf{A}) = \mathbf{A}$ as above

P.1.4 $\mathbf{A} = \mathbf{BC}$: by IH $\mathcal{LR}(\sigma(\mathbf{B}),\gamma \rightarrow \alpha)$ and $\mathcal{LR}(\sigma(\mathbf{C}),\gamma)$

P.1.4.2 so $\mathcal{LR}(\sigma(\mathbf{B})\sigma(\mathbf{C}),\alpha)$ by definition of $\mathcal{LR}$. □

P.1.5 $\mathbf{A} = \lambda X_\beta.\mathbf{C}_\gamma$: Let $\mathcal{LR}(\mathbf{B},\beta)$ and $\theta := \sigma,[\mathbf{B}/X]$, then θ meets the conditions of the IH.

P.1.5.2 Moreover $\sigma(\lambda X_\beta.\mathbf{C}_\gamma)\mathbf{B} \rightarrow_\beta \sigma,[\mathbf{B}/X](\mathbf{C}) = \theta(\mathbf{C})$.

P.1.5.3 Now, $\mathcal{LR}(\theta(\mathbf{C}),\gamma)$ by IH and thus $\mathcal{LR}(\sigma(\mathbf{A})\mathbf{B},\gamma)$ by Lemma 4.35.

P.1.5.4 So $\mathcal{LR}(\sigma(\mathbf{A}), \alpha)$ by definition of $\mathcal{LR}$. □

□

In contrast to the proof of the roller coaster Lemma above, we prove the assertion here by an induction on the structure of the λ-terms involved. For the base cases, we can directly argue with the first assertion from Lemma 4.33, and the application case is immediate from the definition of $\mathcal{LR}$. Indeed, we defined the auxiliary relation $\mathcal{LR}$ exclusively that the application case – which cannot be proven by a direct structural induction; remember that we needed induction on types in Lemma 4.33– becomes easy.

The last case on λ-abstraction reveals why we had to strengthen the inductive hypothesis: $=_\beta$ reduction introduces a substitution which may increase the size of the subterm, which in turn keeps us from applying the inductive hypothesis. Formulating the assertion directly under all possible $\mathcal{LR}$ substitutions unblocks us here.

This was the last result we needed to complete the proof of termiation of β-reduction.

Remark: If we are only interested in the termination of head reductions, we can get by with a much simpler version of this lemma, that basically relies on the uniqueness of head β reduction.

Closure under Head β-Expansion (weakly reducing)

▷ **Lemma 4.38 ($\mathcal{LR}$ is closed under head β-expansion)** *If $\mathbf{C} \to_\beta^h \mathbf{D}$ and $\mathcal{LR}(\mathbf{D}, \alpha)$, so is $\mathcal{LR}(\mathbf{C}, \alpha)$.*

▷ Proof: by induction over the structure of α

P.1.1 α base type:

P.1.1.1 we have $\mathcal{SR}(\mathbf{D}, \alpha)$ by definition

P.1.1.2 so $\mathcal{SR}(\mathbf{C}, \alpha)$, since head reduction is unique

P.1.1.3 and thus $\mathcal{LR}(\mathbf{C}, \alpha)$. □

P.1.2 $\alpha = \beta \to \gamma$:

P.1.2.1 Let $\mathcal{LR}(\mathbf{B}, \beta)$, by definition we have $\mathcal{LR}(\mathbf{DB}, \gamma)$.

P.1.2.2 but $\mathbf{CB} \to_\beta^h \mathbf{DB}$, so $\mathcal{LR}(\mathbf{CB}, \gamma)$ by IH

P.1.2.3 and $\mathcal{LR}(\mathbf{C}, \alpha)$ by definition. □

□

Note: This result only holds for weak reduction (any chain of β head reductions terminates) for strong reduction we need a stronger Lemma.

For the termination proof of head β-reduction we would just use the same proof as above, just for a variant of $\mathcal{SR}$, where $\mathcal{SR}\mathbf{A}\alpha$ that only requires that the head reduction sequence out of $\mathbf{A}$ terminates. Note that almost all of the proof except Lemma 4.32 (which holds by the same argument) is invariant under this change. Indeed Rick Statman uses this observation in [Sta85] to give a set of conditions when logical relations proofs work.

4.5.2 Confluence of $\beta\eta$ Conversion

We now turn to the confluence for $\beta\eta$, i.e. that the order of reductions is irrelevant. This entails the uniqueness of $\beta\eta$ normal forms, which is very useful.

Intuitively confluence of a relation R means that "anything that flows apart will come together again." – and as a consequence normal forms are unique if they exist. But there is more than one way of formalizing that intuition.

▷ Confluence

▷ **Definition 4.39 (Confluence)** Let $R \subseteq A^2$ be a relation on a set A, then we say that

- ▷ has a diamond property, iff for every $a, b, c \in A$ with $a \to^1_R b$ $a \to^1_R c$ there is a $d \in A$ with $b \to^1_R d$ and $c \to^1_R d$.
- ▷ is confluent, iff for every $a, b, c \in A$ with $a \to^*_R b$ $a \to^*_R c$ there is a $d \in A$ with $b \to^*_R d$ and $c \to^*_R d$.
- ▷ weakly confluent iff for every $a, b, c \in A$ with $a \to^1_R b$ $a \to^1_R c$ there is a $d \in A$ with $b \to^*_R d$ and $c \to^*_R d$.

diamond property	confluent	weakly confluent
$\begin{tikzcd} & a \arrow[dl] \arrow[dr] & \\ b \arrow[dr] & & c \arrow[dl] \\ & d & \end{tikzcd}$	$\begin{tikzcd} & a \arrow[dl, "*"'] \arrow[dr, "*"] & \\ b \arrow[dr, "*"'] & & c \arrow[dl, "*"] \\ & d & \end{tikzcd}$	$\begin{tikzcd} & a \arrow[dl] \arrow[dr] & \\ b \arrow[dr, "*"'] & & c \arrow[dl, "*"] \\ & d & \end{tikzcd}$

©: Michael Kohlhase 167

The diamond property is very simple, but not many reduction relations enjoy it. Confluence is the notion that that directly gives us unique normal forms, but is difficult to prove via a digram chase, while weak confluence is amenable to this, does not directly give us confluence.

We will now relate the three notions of confluence with each other: the diamond property (sometimes also called strong confluence) is stronger than confluence, which is stronger than weak confluence

Relating the notions of confluence

▷ **Observation 4.40** *If a rewrite relation has a diamond property, then it is weakly confluent.*

▷ **Theorem 4.41** *If a rewrite relation has a diamond property, then it is confluent.*

▷ Proof Idea: by a tiling argument, composing 1×1 diamonds to an $n \times m$ diamond. □

▷ **Theorem 4.42 (Newman's Lemma)** *If a rewrite relation is terminating and weakly confluent, then it is also confluent.*

©: Michael Kohlhase 168

Note that Newman's Lemma cannot be proven by a tiling argument since we cannot control the growth of the tiles. There is a nifty proof by Gérard Huet [Hue80] that is worth looking at.

After this excursion into the general theory of reduction relations, we come back to the case at hand: showing the confluence of $\beta\eta$-reduction.

η is very well-behaved – i.e. confluent and terminating

η-Reduction ist terminating and confluent

- ▷ **Lemma 4.43** *η-Reduction ist terminating*
- ▷ Proof: by a simple counting argument □
- ▷ **Lemma 4.44** *η-reduction is confluent.*
- ▷ Proof Idea: We show that η-reduction has the diamond property by diagram chase over

$$\begin{tikzcd}
 & \lambda X.\mathbf{A}X \arrow[dl] \arrow[dr] & \\
\mathbf{A} \arrow[dr] & & \lambda X.\mathbf{A}'X \arrow[dl] \\
 & \mathbf{A}' &
\end{tikzcd}$$

where $\mathbf{A} \rightarrow_\eta \mathbf{A}'$. Then the assertion follows by Theorem 4.41. □

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

For β-reduction the situation is a bit more involved, but a simple diagram chase is still sufficient to prove weak confluence, which gives us confluence via Newman's Lemma

β is confluent

- ▷ **Lemma 4.45** *β-Reduction is weakly confluent.*
- ▷ Proof Idea: by diagram chase over

$$\begin{tikzcd}
 & & (\lambda X.\mathbf{A})\mathbf{B} \arrow[dll] \arrow[d] \arrow[drr] & & \\
(\lambda X.\mathbf{A}')\mathbf{B} \arrow[dr] & & (\lambda X.\mathbf{A})\mathbf{B}' \arrow[dl] \arrow[dr] & & {[\mathbf{B}/X](\mathbf{A})} \arrow[dl, "*"'] \\
 & (\lambda X.\mathbf{A}')\mathbf{B}' \arrow[dr] & & {[\mathbf{B}'/X](\mathbf{A})} \arrow[dl] & \\
 & & {[\mathbf{B}'/X](\mathbf{A}')} & &
\end{tikzcd}$$

□

- ▷ **Corollary 4.46** *β-Reduction is confluent.*
- ▷ Proof Idea: by Newman's Lemma. □

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

There is one reduction in the diagram in the proof of Lemma 4.45 which (note that $\mathbf{B}$ can occur multiple times in $[\mathbf{B}/X](\mathbf{A})$) is not necessary single-step. The diamond property is broken by the outer two reductions in the diagram as well.

We have shown that the β and η reduction relations are terminating and confluent and terminating individually, now, we have to show that $\beta\eta$ is a well. For that we introduce a new concept.

Commuting Relations

▷ **Definition 4.47** Let A be a set, then we say that relations $\mathcal{R} \in A^2$ and $\mathcal{S} \in A^2$ commute, if $X \to_{\mathcal{R}} Y$ and $X \to_{\mathcal{S}} Z$ entail the existence of a $W \in A$ with $Y \to_{\mathcal{S}} W$ and $Z \to_{\mathcal{R}} W$.

$$\begin{tikzcd} & X \arrow[dl, "\mathcal{R}"'] \arrow[dr, "\mathcal{S}"] & \\ Y \arrow[dr, "\mathcal{S}"'] & & Z \arrow[dl, "\mathcal{R}"] \\ & W & \end{tikzcd}$$

▷ **Observation 4.48** *If $\mathcal{R}$ and $\mathcal{S}$ commute, then $\to_{\mathcal{R}}$ and $\to_{\mathcal{S}}$ do as well.*

▷ **Observation 4.49** *$\mathcal{R}$ is confluent, if $\mathcal{R}$ commutes with itself.*

▷ **Lemma 4.50** *If $\mathcal{R}$ and $\mathcal{S}$ are terminating and confluent relations such that $\to^*_{\mathcal{R}}$ and $\to^*_{\mathcal{S}}$ commute, then $\to^*_{\mathcal{R}\cup\mathcal{S}}$ is confluent.*

▷ Proof Sketch: As $\mathcal{R}$ and $\mathcal{S}$ commute, we can reorder any reduction sequence so that all $\mathcal{R}$-reductions precede all $\mathcal{S}$-reductions. As $\mathcal{R}$ is terminating and confluent, the $\mathcal{R}$-part ends in a unique normal form, and as $\mathcal{S}$ is normalizing it must lead to a unique normal form as well. □

SOME RIGHTS RESERVED ©: Michael Kohlhase 171 JACOBS UNIVERSITY

This directly gives us our goal.

$\beta\eta$ is confluent

▷ **Lemma 4.51** *$\to^*_\beta$ and $\to^*_\eta$ commute.*

▷ Proof Sketch: diagram chase □

SOME RIGHTS RESERVED ©: Michael Kohlhase 172 JACOBS UNIVERSITY

4.6 The Semantics of the Simply Typed λ-Calculus

The semantics of $\Lambda^{\to}$ is structured around the types. Like the models we discussed before, a model (we call them "algebras", since we do not have truth values in $\Lambda^{\to}$) is a pair $\langle\mathcal{D},\mathcal{I}\rangle$, where $\mathcal{D}$ is the universe of discourse and $\mathcal{I}$ is the interpretation of constants.

Semantics of $\Lambda^{\to}$

▷ **Definition 4.52** We call a collection $\mathcal{D}_{\mathcal{T}} := \{\mathcal{D}_\alpha \,|\, \alpha \in \mathcal{T}\}$ a typed collection (of sets) and a collection $f_{\mathcal{T}} \colon \mathcal{D}_{\mathcal{T}} \to \mathcal{E}_{\mathcal{T}}$, a typed function, iff $f_\alpha \colon \mathcal{D}_\alpha \to \mathcal{E}_\alpha$.

▷ **Definition 4.53** A typed collection $\mathcal{D}_{\mathcal{T}}$ is called a frame, iff $\mathcal{D}_{\alpha\to\beta} \subseteq \mathcal{D}_\alpha \to \mathcal{D}_\beta$

▷ **Definition 4.54** Given a frame $\mathcal{D}_{\mathcal{T}}$, and a typed function $\mathcal{I} \colon \Sigma \to \mathcal{D}$, then we call $\mathcal{I}_\varphi \colon \mathit{wff}_{\mathcal{T}}(\Sigma, \mathcal{V}_{\mathcal{T}}) \to \mathcal{D}$ the value function induced by $\mathcal{I}$, iff

- ▷ $\mathcal{I}_\varphi|_{\mathcal{V}_{\mathcal{T}}} = \varphi$, $\qquad \mathcal{I}_\varphi|_\Sigma = \mathcal{I}$

- ▷ $\mathcal{I}_\varphi(\mathbf{AB}) = \mathcal{I}_\varphi(\mathbf{A})(\mathcal{I}_\varphi(\mathbf{B}))$
- ▷ $\mathcal{I}_\varphi(\lambda X_\alpha.\mathbf{A})$ is that function $f \in \mathcal{D}_{\alpha\to\beta}$, such that $f(a) = \mathcal{I}_{\varphi,[a/X]}(\mathbf{A})$ for all $a \in \mathcal{D}_\alpha$

▷ **Definition 4.55** We call a frame $\langle\mathcal{D},\mathcal{I}\rangle$ comprehension-closed or a Σ-algebra, iff $\mathcal{I}_\varphi \colon wff_\mathcal{T}(\Sigma,\mathcal{V}_\mathcal{T}) \to \mathcal{D}$ is total. (every λ-term has a value)

SOME RIGHTS RESERVED ©: Michael Kohlhase 173 JACOBS UNIVERSITY

4.6.1 Soundness of the Simply Typed λ-Calculus

We will now show is that $\alpha\beta\eta$-reduction does not change the value of formulae, i.e. if $\mathbf{A} =_{\alpha\beta\eta} \mathbf{B}$, then $\mathcal{I}_\varphi(\mathbf{A}) = \mathcal{I}_\varphi(\mathbf{B})$, for all $\mathcal{D}$ and φ. We say that the reductions are sound. As always, the main tool for proving soundess is a substitution value lemma. It works just as always and verifies that we the definitions are in our semantics plausible.

Substitution Value Lemma for λ-Terms

▷ **Lemma 4.56 (Substitution Value Lemma)** *Let* $\mathbf{A}$ *and* $\mathbf{B}$ *be terms, then* $\mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{A})) = \mathcal{I}_\psi(\mathbf{A})$*, where* $\psi = \varphi, [\mathcal{I}_\varphi(\mathbf{B})/X]$

▷ Proof: by induction on the depth of $\mathbf{A}$

P.1 we have five cases

P.1.1 $\mathbf{A} = X$: Then $\mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{A})) = \mathcal{I}_\varphi([\mathbf{B}/X](X)) = \mathcal{I}_\varphi(\mathbf{B}) = \psi(X) = \mathcal{I}_\psi(X) = \mathcal{I}_\psi(\mathbf{A})$.

P.1.2 $\mathbf{A} = Y \neq X$ and $Y \in \mathcal{V}_\mathcal{T}$: then $\mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{A})) = \mathcal{I}_\varphi([\mathbf{B}/X](Y)) = \mathcal{I}_\varphi(Y) = \varphi(Y) = \psi(Y) = \mathcal{I}_\psi(Y) = \mathcal{I}_\psi(\mathbf{A})$.

P.1.3 $\mathbf{A} \in \Sigma$: This is analogous to the last case.

P.1.4 $\mathbf{A} = \mathbf{CD}$: then $\mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{A})) = \mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{CD})) = \mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{C})[\mathbf{B}/X](\mathbf{D})) = \mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{C}))(\mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{D}))) = \mathcal{I}_\psi(\mathbf{C})(\mathcal{I}_\psi(\mathbf{D})) = \mathcal{I}_\psi(\mathbf{CD}) = \mathcal{I}_\psi(\mathbf{A})$

P.1.5 $\mathbf{A} = \lambda Y_\alpha.\mathbf{C}$:

P.1.5.1 We can assume that $X \neq Y$ and $Y \notin \mathbf{free}(\mathbf{B})$

P.1.5.2 Thus for all $a \in \mathcal{D}_\alpha$ we have $\mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{A}))(a) = \mathcal{I}_\varphi([\mathbf{B}/X](\lambda Y.\mathbf{C}))(a) = \mathcal{I}_\varphi(\lambda Y.[\mathbf{B}/X](\mathbf{C}))(a) = \mathcal{I}_{\varphi,[a/Y]}([\mathbf{B}/X](\mathbf{C})) = \mathcal{I}_{\psi,[a/Y]}(\mathbf{C}) = \mathcal{I}_\psi(\lambda Y.\mathbf{C})(a) = \mathcal{I}_\psi(\mathbf{A})(a)$ □

□

SOME RIGHTS RESERVED ©: Michael Kohlhase 174 JACOBS UNIVERSITY

Soundness of $\alpha\beta\eta$-Equality

▷ **Theorem 4.57** *Let* $\mathcal{A} := \langle\mathcal{D},\mathcal{I}\rangle$ *be a* Σ*-algebra and* $Y \notin \mathbf{free}(\mathbf{A})$*, then* $\mathcal{I}_\varphi(\lambda X.\mathbf{A}) = \mathcal{I}_\varphi(\lambda Y.[Y/X]\mathbf{A})$ *for all assignments* φ*.*

▷ Proof: by substitution value lemma

$$\begin{aligned}\mathcal{I}_\varphi(\lambda Y.[Y/X]\mathbf{A})@\mathsf{a} &= \mathcal{I}_{\varphi,[a/Y]}([Y/X](\mathbf{A}))\\ &= \mathcal{I}_{\varphi,[a/X]}(\mathbf{A})\\ &= \mathcal{I}_\varphi(\lambda X.\mathbf{A})@\mathsf{a}\end{aligned}$$

▷ **Theorem 4.58** *If $\mathcal{A} := \langle\mathcal{D},\mathcal{I}\rangle$ is a Σ-algebra and X not bound in $\mathbf{A}$, then $\mathcal{I}_\varphi((\lambda X.\mathbf{A})\mathbf{B}) = \mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{A}))$.*

▷ Proof: by substitution value lemma again

$$\begin{aligned}\mathcal{I}_\varphi((\lambda X.\mathbf{A})\mathbf{B}) &= \mathcal{I}_\varphi(\lambda X.\mathbf{A})@\mathcal{I}_\varphi(\mathbf{B})\\ &= \mathcal{I}_{\varphi,[\mathcal{I}_\varphi(\mathbf{B})/X]}(\mathbf{A})\\ &= \mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{A}))\end{aligned}$$

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

Soundness of $\alpha\beta\eta$ (continued)

▷ **Theorem 4.59** *If $X \notin \mathbf{free}(\mathbf{A})$, then $\mathcal{I}_\varphi(\lambda X.\mathbf{A}X) = \mathcal{I}_\varphi(\mathbf{A})$ for all φ.*

▷ Proof: by calculation

$$\begin{aligned}\mathcal{I}_\varphi(\lambda X.\mathbf{A}X)@\mathsf{a} &= \mathcal{I}_{\varphi,[\mathsf{a}/X]}(\mathbf{A}X)\\ &= \mathcal{I}_{\varphi,[\mathsf{a}/X]}(\mathbf{A})@\mathcal{I}_{\varphi,[\mathsf{a}/X]}(X)\\ &= \mathcal{I}_\varphi(\mathbf{A})@\mathcal{I}_{\varphi,[\mathsf{a}/X]}(X) && \text{as } X \notin \mathbf{free}(\mathbf{A}).\\ &= \mathcal{I}_\varphi(\mathbf{A})@\mathsf{a}\end{aligned}$$

▷ **Theorem 4.60** *$\alpha\beta\eta$-equality is sound wrt. Σ-algebras. (if $\mathbf{A} =_{\alpha\beta\eta} \mathbf{B}$, then $\mathcal{I}_\varphi(\mathbf{A}) = \mathcal{I}_\varphi(\mathbf{B})$ for all assignments φ)*

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

4.6.2 Completeness of $\alpha\beta\eta$-Equality

We will now show is that $\alpha\beta\eta$-equality is complete for the semantics we defined, i.e. that whenever $\mathcal{I}_\varphi(\mathbf{A}) = \mathcal{I}_\varphi(\mathbf{B})$ for all variable assignments φ, then $\mathbf{A} =_{\alpha\beta\eta} \mathbf{B}$. We will prove this by a model existence argument: we will construct a model $\mathcal{M} := \langle\mathcal{D},\mathcal{I}\rangle$ such that if $\mathbf{A} \neq_{\alpha\beta\eta} \mathbf{B}$ then $\mathcal{I}_\varphi(\mathbf{A}) \neq \mathcal{I}_\varphi(\mathbf{B})$ for some φ.

As in other completeness proofs, the model we will construct is a "ground term model", i.e. a model where the carrier (the frame in our case) consists of ground terms. But in the λ-calculus, we have to do more work, as we have a non-trivial built-in equality theory; we will construct the "ground term model" from sets of normal forms. So we first fix some notations for them.

Normal Forms in the simply typed λ-calculus

▷ **Definition 4.61** We call a term $\mathbf{A} \in wff_\mathcal{T}(\Sigma,\mathcal{V}_\mathcal{T})$ a β normal form iff there is no $\mathbf{B} \in wff_\mathcal{T}(\Sigma,\mathcal{V}_\mathcal{T})$ with $\mathbf{A} \to_\beta \mathbf{B}$.

We call $\mathbf{N}$ a β normal form of $\mathbf{A}$, iff $\mathbf{N}$ is a β-normal form and $\mathbf{A} \to_\beta \mathbf{N}$.

We denote the set of β-normal forms with $\mathit{wff}_{\mathcal{T}}(\Sigma, \mathcal{V}_{\mathcal{T}})\downarrow_{\beta\eta}$.

▷ We have just proved that $\beta\eta$-reduction is terminating and confluent, so we have

▷ **Corollary 4.62 (Normal Forms)** *Every $\mathbf{A} \in \mathit{wff}_{\mathcal{T}}(\Sigma, \mathcal{V}_{\mathcal{T}})$ has a unique β normal form ($\beta\eta$, long $\beta\eta$ normal form), which we denote by $\mathbf{A}\downarrow_{\beta}$ ($\mathbf{A}\downarrow_{\beta\eta}$ $\mathbf{A}\downarrow^{l}_{\beta\eta}$)*

SOME RIGHTS RESERVED ©: Michael Kohlhase

The term frames will be a quotient spaces over the equality relations of the λ-calculus, so we introduce this construction generally.

Frames and Quotients

▷ **Definition 4.63** Let $\mathcal{D}$ be a frame and $\sim$ a typed equivalence relation on $\mathcal{D}$, then we call $\sim$ a congruence on $\mathcal{D}$, iff $f \sim f'$ and $g \sim g'$ imply $f(g) \sim f'(g')$.

▷ **Definition 4.64** We call a congruence $\sim$ functional, iff for all $f, g \in \mathcal{D}_{\alpha\to\beta}$ the fact that $f(a) \sim g(a)$ holds for all $a \in \mathcal{D}_{\alpha}$ implies that $f \sim g$.

▷ **Example 4.65** $=_{\beta}$ ($=_{\beta\eta}$) is a (functional) congruence on $\mathit{cwff}_{\mathcal{T}}(\Sigma)$ by definition.

▷ **Theorem 4.66** *Let $\mathcal{D}$ be a Σ-frame and $\sim$ a functional congruence on $\mathcal{D}$, then the quotient space $\mathcal{D}/_{\sim}$ is a Σ-frame.*

▷ Proof:

P.1 $\mathcal{D}/_{\sim} = \{[f]_{\sim} \mid f \in \mathcal{D}\}$, define $[f]_{\sim}([a]_{\sim}) := [f(a)]_{\sim}$.

P.2 This only depends on equivalence classes: Let $f' \in [f]_{\sim}$ and $a' \in [a]_{\sim}$.

P.3 Then $[f(a)]_{\sim} = [f'(a)]_{\sim} = [f'(a')]_{\sim} = [f(a')]_{\sim}$

P.4 To see that we have $[f]_{\sim} = [g]_{\sim}$, iff $f \sim g$, iff $f(a) = g(a)$ since $\sim$ is functional.

P.5 This is the case iff $[f(a)]_{\sim} = [g(a)]_{\sim}$, iff $[f]_{\sim}([a]_{\sim}) = [g]_{\sim}([a]_{\sim})$ for all $a \in \mathcal{D}_{\alpha}$ and thus for all $[a]_{\sim} \in \mathcal{D}/_{\sim}$. □

SOME RIGHTS RESERVED ©: Michael Kohlhase

To apply this result, we have to establish that $\beta\eta$-equality is a functional congruence.

We first establish $\beta\eta$ as a functional congruence on $\mathit{wff}_{\mathcal{T}}(\Sigma, \mathcal{V}_{\mathcal{T}})$ and then specialize this result to show that is is also functional on $\mathit{cwff}_{\mathcal{T}}(\Sigma)$ by a grounding argument.

$\beta\eta$-Equivalence as a Functional Congruence

▷ **Lemma 4.67** *$\beta\eta$-equality is a functional congruence on $\mathit{wff}_{\mathcal{T}}(\Sigma, \mathcal{V}_{\mathcal{T}})$.*

▷ Proof: Let $\mathbf{AC} =_{\beta\eta} \mathbf{BC}$ for all $\mathbf{C}$ and $X \in (\mathcal{V}_{\gamma} \backslash (\mathbf{free}(\mathbf{A}) \cup \mathbf{free}(\mathbf{B})))$.

P.1 then (in particular) $\mathbf{A}X =_{\beta\eta} \mathbf{B}X$, and

P.2 $\lambda X.\mathbf{A}X =_{\beta\eta} \lambda X.\mathbf{B}X$, since $\beta\eta$-equality acts on subterms.

P.3 By definition we have $\mathbf{A} =_\eta \lambda X_\alpha.\mathbf{A}X =_{\beta\eta} \lambda X_\alpha.\mathbf{B}X =_\eta \mathbf{B}$. □

▷ **Definition 4.68** We call an injective substitution $\sigma\colon \mathbf{free}(\mathbf{C}) \to \Sigma$ a grounding substitution for $\mathbf{C} \in \mathit{wff}_{\mathcal{T}}(\Sigma, \mathcal{V}_{\mathcal{T}})$, iff no $\sigma(X)$ occurs in $\mathbf{C}$.

Observation: They always exist, since all Σ_α are infinite and $\mathbf{free}(\mathbf{C})$ is finite.

⊳ **Theorem 4.69** *$\beta\eta$-equality is a functional congruence on $\mathit{cwff}_{\mathcal{T}}(\Sigma)$.*

▷ Proof: We use Lemma 4.67

P.1 Let $\mathbf{A}, \mathbf{B} \in \mathit{cwff}_{(\alpha\to\beta)}(\Sigma)$, such that $\mathbf{A} \neq_{\beta\eta} \mathbf{B}$.

P.2 As $\beta\eta$ is functional on $\mathit{wff}_{\mathcal{T}}(\Sigma, \mathcal{V}_{\mathcal{T}})$, there must be a $\mathbf{C}$ with $\mathbf{AC} \neq_{\beta\eta} \mathbf{BC}$.

P.3 Now let $\mathbf{C}' := \sigma(\mathbf{C})$, for a grounding substitution σ.

P.4 Any $\beta\eta$ conversion sequence for $\mathbf{AC}' \neq_{\beta\eta} \mathbf{BC}'$ induces one for $\mathbf{AC} \neq_{\beta\eta} \mathbf{BC}$.

P.5 Thus we have shown that $\mathbf{A} \neq_{\beta\eta} \mathbf{B}$ entails $\mathbf{AC}' \neq_{\beta\eta} \mathbf{BC}'$. □

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

Note that: the result for $\mathit{cwff}_{\mathcal{T}}(\Sigma)$ is sharp. For instance, if $\Sigma = \{c_\iota\}$, then $\lambda X.X \neq_{\beta\eta} \lambda X.c$, but $(\lambda X.X)c =_{\beta\eta} c =_{\beta\eta} (\lambda X.c)c$, as $\{c\} = \mathit{cwff}_\iota(\Sigma)$ (it is a relatively simple exercise to extend this problem to more than one constant). The problem here is that we do not have a constant d_ι that would help distinguish the two functions. In $\mathit{wff}_{\mathcal{T}}(\Sigma, \mathcal{V}_{\mathcal{T}})$ we could always have used a variable.

This completes the preparation and we can define the notion of a term algebra, i.e. a Σ-algebra whose frame is made of $\beta\eta$-normal λ-terms.

A Herbrand Model for $\Lambda^\to$

▷ **Definition 4.70** We call $\mathcal{T}_{\beta\eta} := \langle \mathit{cwff}_{\mathcal{T}}(\Sigma){\downarrow_{\beta\eta}}, \mathcal{I}^{\beta\eta}\rangle$ the Σ term algebra, if $\mathcal{I}^{\beta\eta} = \mathrm{Id}_\Sigma$.

▷ The name "term algebra" in the previous definition is justified by the following

▷ **Theorem 4.71** *$\mathcal{T}_{\beta\eta}$ is a Σ-algebra*

▷ Proof: We use the work we did above

P.1 Note that $\mathit{cwff}_{\mathcal{T}}(\Sigma){\downarrow_{\beta\eta}} = \mathit{cwff}_{\mathcal{T}}(\Sigma)/_{=_{\beta\eta}}$ and thus a Σ-frame by Theorem 4.66 and Lemma 4.67.

P.2 So we only have to show that the value function $\mathcal{I}^{\beta\eta} = \mathrm{Id}_\Sigma$ is total.

P.3 Let φ be an assignment into $\mathit{cwff}_{\mathcal{T}}(\Sigma){\downarrow_{\beta\eta}}$.

P.4 Note that $\sigma := (\varphi|_{\mathbf{free}(\mathbf{A})})$ is a substitution, since $\mathbf{free}(\mathbf{A})$ is finite.

P.5 A simple induction on the structure of $\mathbf{A}$ shows that $\mathcal{I}^{\beta\eta}_\varphi(\mathbf{A}) = \sigma(\mathbf{A}){\downarrow_{\beta\eta}}$.

P.6 So the value function is total since substitution application is. □

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

And as always, once we have a term model, showing completeness is a rather simple exercise.

We can see that $\alpha\beta\eta$-equality is complete for the class of Σ-algebras, i.e. if the equation $\mathbf{A} = \mathbf{B}$ is valid, then $\mathbf{A} =_{\alpha\beta\eta} \mathbf{B}$. Thus $\alpha\beta\eta$ equivalence fully characterizes equality in the class of all Σ-algebras.

Completetness of $\alpha\beta\eta$-Equality

▷ **Theorem 4.72** $\mathbf{A} = \mathbf{B}$ *is valid in the class of* Σ*-algebras, iff* $\mathbf{A} =_{\alpha\beta\eta} \mathbf{B}$.

▷ Proof: For $\mathbf{A}$, $\mathbf{B}$ closed this is a simple consequence of the fact that $\mathcal{T}_{\beta\eta}$ is a Σ-algebra.

P.1 If $\mathbf{A} = \mathbf{B}$ is valid in all Σ-algebras, it must be in $\mathcal{T}_{\beta\eta}$ and in particular $\mathbf{A}\downarrow_{\beta\eta} = \mathcal{I}^{\beta\eta}(\mathbf{A}) = \mathcal{I}^{\beta\eta}(\mathbf{B}) = \mathbf{B}\downarrow_{\beta\eta}$ and therefore $\mathbf{A} =_{\alpha\beta\eta} \mathbf{B}$.

P.2 If the equation has free variables, then the argument is more subtle.

P.3 Let σ be a grounding substitution for $\mathbf{A}$ and $\mathbf{B}$ and φ the induced variable assignment.

P.4 Thus ${\mathcal{I}^{\beta\eta}}_{\varphi}(\mathbf{A}) = {\mathcal{I}^{\beta\eta}}_{\varphi}(\mathbf{B})$ is the $\beta\eta$-normal form of $\sigma(\mathbf{A})$ and $\sigma(\mathbf{B})$.

P.5 Since φ is a structure preserving homomorphism on well-formed formulae, $\varphi^{-1}({\mathcal{I}^{\beta\eta}}_{\varphi}(\mathbf{A}))$ is the is the $\beta\eta$-normal form of both $\mathbf{A}$ and $\mathbf{B}$ and thus $\mathbf{A} =_{\alpha\beta\eta} \mathbf{B}$. □

SOME RIGHTS RESERVED ©: Michael Kohlhase 181 JACOBS UNIVERSITY

Theorem 4.72 and Theorem 4.60 complete our study of the sematnics of the simply-typed λ-calculus by showing that it is an adequate logic for modeling (the equality) of functions and their applications.

4.7 Simply Typed λ-Calculus via Inference Systems

Now, we will look at the simply typed λ-calculus again, but this time, we will present it as an inference system for well-typedness jugdments. This more modern way of developing type theories is known to scale better to new concepts.

Simply Typed λ-Calculus as an Inference System: Terms

▷ Idea: Develop the λ-calculus in two steps

- ▷ A context-free grammar for "raw λ-terms" (for the structure)
- ▷ Identify the well-typed λ-terms in that (cook them until well-typed)

▷ **Definition 4.73** A grammar for the raw terms of the simply typed λ-calculus:

$$\begin{array}{lcl} \alpha & :== & c \mid \alpha \to \alpha \\ \Sigma & :== & \cdot \mid \Sigma, [c : \mathsf{type}] \mid \Sigma, [c : \alpha] \\ \Gamma & :== & \cdot \mid \Gamma, [x : \alpha] \\ \mathbf{A} & :== & c \mid X \mid \mathbf{A}^1\mathbf{A}^2 \mid \lambda X_\alpha.\mathbf{A} \end{array}$$

▷ Then: Define all the operations that are possible at the "raw terms level", e.g. realize that signatures and contexts are partial functions to types.

SOME RIGHTS RESERVED ©: Michael Kohlhase 182 JACOBS UNIVERSITY

Simply Typed λ-Calculus as an Inference System: Judgments

▷ **Definition 4.74** Judgment s make statements about complex properties of the syntactic entities defined by the grammar.

▷ **Definition 4.75** Judgments for the simply typed λ-calculus

$\vdash \Sigma : \mathsf{sig}$	Σ is a well-formed signature
$\Sigma \vdash \alpha : \mathsf{type}$	α is a well-formed type given the type assumptions in Σ
$\Sigma \vdash \Gamma : \mathsf{ctx}$	Γ is a well-formed context given the type assumptions in Σ
$\Gamma \vdash_\Sigma \mathbf{A} \colon \alpha$	$\mathbf{A}$ has type α given the type assumptions in Σ and Γ

SOME RIGHTS RESERVED ©: Michael Kohlhase 183 JACOBS UNIVERSITY

Simply Typed λ-Calculus as an Inference System: Rules

▷ $\mathbf{A} \in \mathit{wff}_\alpha(\Sigma, \mathcal{V}_\mathcal{T})$, iff $\Gamma \vdash_\Sigma \mathbf{A} \colon \alpha$ derivable in

$$\frac{\Sigma \vdash \Gamma : \mathsf{ctx} \quad \Gamma(X) = \alpha}{\Gamma \vdash_\Sigma X \colon \alpha}\,\mathsf{wff{:}var} \qquad \frac{\Sigma \vdash \Gamma : \mathsf{ctx} \quad \Sigma(c) = \alpha}{\Gamma \vdash_\Sigma c \colon \alpha}\,\mathsf{wff{:}const}$$

$$\frac{\Gamma \vdash_\Sigma \mathbf{A} \colon \beta \to \alpha \quad \Gamma \vdash_\Sigma \mathbf{B} \colon \beta}{\Gamma \vdash_\Sigma \mathbf{A}\mathbf{B} \colon \alpha}\,\mathsf{wff{:}app} \qquad \frac{\Gamma, [X : \beta] \vdash_\Sigma \mathbf{A} \colon \alpha}{\Gamma \vdash_\Sigma \lambda X_\beta . \mathbf{A} \colon \beta \to \alpha}\,\mathsf{wff{:}abs}$$

Oops: this looks surprisingly like a natural deduction calculus. ($\leadsto$ Curry Howard Isomorphism)

▷ To be complete, we need rules for well-formed signatures, types and contexts

$$\frac{}{\vdash \cdot : \mathsf{sig}}\,\mathsf{sig{:}empty} \qquad \frac{\vdash \Sigma : \mathsf{sig}}{\vdash \Sigma, [\alpha : \mathsf{type}] : \mathsf{sig}}\,\mathsf{sig{:}type}$$

$$\frac{\vdash \Sigma : \mathsf{sig} \quad \Sigma \vdash \alpha : \mathsf{type}}{\vdash \Sigma, [c : \alpha] : \mathsf{sig}}\,\mathsf{sig{:}const}$$

$$\frac{\Sigma \vdash \alpha : \mathsf{type} \quad \Sigma \vdash \beta : \mathsf{type}}{\Sigma \vdash \alpha \to \beta : \mathsf{type}}\,\mathsf{typ{:}fn} \qquad \frac{\vdash \Sigma : \mathsf{sig} \quad \Sigma(\alpha) = \mathsf{type}}{\Sigma \vdash \alpha : \mathsf{type}}\,\mathsf{typ{:}start}$$

$$\frac{\vdash \Sigma : \mathsf{sig}}{\Sigma \vdash \cdot : \mathsf{ctx}}\,\mathsf{ctx{:}empty} \qquad \frac{\Sigma \vdash \Gamma : \mathsf{ctx} \quad \Sigma \vdash \alpha : \mathsf{type}}{\Sigma \vdash \Gamma, [X : \alpha] : \mathsf{ctx}}\,\mathsf{ctx{:}var}$$

SOME RIGHTS RESERVED ©: Michael Kohlhase 184 JACOBS UNIVERSITY

Example: A Well-Formed Signature

▷ Let $\Sigma := [\alpha : \mathsf{type}], [f : \alpha \to \alpha \to \alpha]$, then Σ is a well-formed signature, since we have derivations $\mathcal{A}$ and $\mathcal{B}$

$$\dfrac{\vdash \cdot : \mathsf{sig}}{\vdash [\alpha : \mathsf{type}] : \mathsf{sig}}\,\mathsf{sig{:}type} \qquad \dfrac{\mathcal{A} \quad [\alpha : \mathsf{type}](\alpha) = \mathsf{type}}{[\alpha : \mathsf{type}] \vdash \alpha : \mathsf{type}}\,\mathsf{typ{:}start}$$

and with these we can construct the derivation $\mathcal{C}$

$$\dfrac{\mathcal{A} \quad \dfrac{\mathcal{B} \quad \dfrac{\mathcal{B} \quad \mathcal{B}}{[\alpha : \mathsf{type}] \vdash \alpha \to \alpha : \mathsf{type}}\,\mathsf{typ{:}fn}}{[\alpha : \mathsf{type}] \vdash \alpha \to \alpha \to \alpha : \mathsf{type}}\,\mathsf{typ{:}fn}}{\vdash \Sigma : \mathsf{sig}}\,\mathsf{sig{:}const}$$

SOME RIGHTS RESERVED

Example: A Well-Formed λ-Term

▷ using Σ from above, we can show that $\Gamma := [X : \alpha]$ is a well-formed context:

$$\dfrac{\dfrac{\mathcal{C}}{\Sigma \vdash \cdot : \mathsf{ctx}}\,\mathsf{ctx{:}empty} \quad \dfrac{\mathcal{C} \quad \Sigma(\alpha) = \mathsf{type}}{\Sigma \vdash \alpha : \mathsf{type}}\,\mathsf{typ{:}start}}{\Sigma \vdash \Gamma : \mathsf{ctx}}\,\mathsf{ctx{:}var}$$

We call this derivation $\mathcal{G}$ and use it to show that

▷ $\lambda X_\alpha . f X X$ is well-typed and has type $\alpha \to \alpha$ in Σ. This is witnessed by the type derivation

$$\dfrac{\dfrac{\dfrac{\dfrac{\mathcal{C} \quad \Sigma(f) = \alpha \to \alpha \to \alpha}{\Gamma \vdash_\Sigma f : \alpha \to \alpha \to \alpha}\,\mathsf{wff{:}const} \quad \dfrac{\mathcal{G}}{\Gamma \vdash_\Sigma X : \alpha}\,\mathsf{wff{:}var}}{\Gamma \vdash_\Sigma f X : \alpha \to \alpha}\,\mathsf{wff{:}app} \quad \dfrac{\mathcal{G}}{\Gamma \vdash_\Sigma X : \alpha}\,\mathsf{wff{:}var}}{\Gamma \vdash_\Sigma f X X : \alpha}\,\mathsf{wff{:}app}}{\cdot \vdash_\Sigma \lambda X_\alpha . f X X : \alpha \to \alpha}\,\mathsf{wff{:}abs}$$

SOME RIGHTS RESERVED

$\beta\eta$-Equality by Inference Rules: One-Step Reduction

▷ One-step Reduction ($+ \in \{\alpha, \beta, \eta\}$)

$$\frac{\Gamma \vdash_\Sigma \mathbf{A} \colon \alpha \quad \Gamma \vdash_\Sigma \mathbf{B} \colon \beta}{\Gamma \vdash_\Sigma (\lambda X.\mathbf{A})\mathbf{B} \rightarrow^1_\beta [\mathbf{B}/X](\mathbf{A})} \mathsf{wff}\beta\mathsf{:top}$$

$$\frac{\Gamma \vdash_\Sigma \mathbf{A} \colon \beta \to \alpha \quad X \notin \mathbf{dom}(\Gamma)}{\Gamma \vdash_\Sigma \lambda X.\mathbf{A}X \rightarrow^1_\eta \mathbf{A}} \mathsf{wff}\eta\mathsf{:top}$$

$$\frac{\Gamma \vdash_\Sigma \mathbf{A} \rightarrow^1_+ \mathbf{B} \quad \Gamma \vdash_\Sigma \mathbf{AC} \colon \alpha}{\Gamma \vdash_\Sigma \mathbf{AC} \rightarrow^1_+ \mathbf{BC}} \mathsf{tr{:}app}fn$$

$$\frac{\Gamma \vdash_\Sigma \mathbf{A} \rightarrow^1_+ \mathbf{B} \quad \Gamma \vdash_\Sigma \mathbf{CA} \colon \alpha}{\Gamma \vdash_\Sigma \mathbf{CA} \rightarrow^1_+ \mathbf{CB}} \mathsf{tr{:}app}arg$$

$$\frac{\Gamma, [X : \alpha] \vdash_\Sigma \mathbf{A} \rightarrow^1_+ \mathbf{B}}{\Gamma \vdash_\Sigma \lambda X.\mathbf{A} \rightarrow^1_+ \lambda X.\mathbf{B}} \mathsf{tr{:}abs}$$

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

$\beta\eta$-Equality by Inference Rules: Multi-Step Reduction

▷ Multi-Step-Reduction ($+ \in \{\alpha, \beta, \eta\}$)

$$\frac{\Gamma \vdash_\Sigma \mathbf{A} \rightarrow^1_+ \mathbf{B}}{\Gamma \vdash_\Sigma \mathbf{A} \rightarrow^*_+ \mathbf{B}} \mathsf{ms{:}start} \qquad \frac{\Gamma \vdash_\Sigma \mathbf{A} \colon \alpha}{\Gamma \vdash_\Sigma \mathbf{A} \rightarrow^*_+ \mathbf{A}} \mathsf{ms{:}ref}$$

$$\frac{\Gamma \vdash_\Sigma \mathbf{A} \rightarrow^*_+ \mathbf{B} \quad \Gamma \vdash_\Sigma \mathbf{B} \rightarrow^*_+ \mathbf{C}}{\Gamma \vdash_\Sigma \mathbf{A} \rightarrow^*_+ \mathbf{C}} \mathsf{ms{:}trans}$$

▷ Congruence Relation

$$\frac{\Gamma \vdash_\Sigma \mathbf{A} \rightarrow^*_+ \mathbf{B}}{\Gamma \vdash_\Sigma \mathbf{A} =_+ \mathbf{B}} \mathsf{eq{:}start}$$

$$\frac{\Gamma \vdash_\Sigma \mathbf{A} =_+ \mathbf{B}}{\Gamma \vdash_\Sigma \mathbf{B} =_+ \mathbf{A}} \mathsf{eq{:}sym} \qquad \frac{\Gamma \vdash_\Sigma \mathbf{A} =_+ \mathbf{B} \quad \Gamma \vdash_\Sigma \mathbf{B} =_+ \mathbf{C}}{\Gamma \vdash_\Sigma \mathbf{A} =_+ \mathbf{C}} \mathsf{eq{:}trans}$$

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

5 Fragment 4: Noun Phrases and Quantification

5.1 Overview/Summary so far

Where we started: A *VP*-less fragment and $\mathrm{PL_{NQ}}$.:

$\mathrm{PL_{NQ}}$	Fragment of English
Syntax: Definition of wffs	Syntax: Definition of allowable sentences
Semantics: Model theory	SEMANTICS BY TRANSLATION

What we did:

- Tested the translation by testing predictions: semantic tests of entailment.
- More testing: syntactic tests of entailment. For this, we introduced the model generation calculus. We can make this move from semantic proofs to syntactic ones safely, because we know that $\mathrm{PL_{NQ}}$ is sound and complete.
- Moving beyond semantics: Used model generation to predict interpretations of semantically under-determined sentence types.

Where we are now: A fragment with a VP and HOL_{NQ}.: We expanded the fragment and began to consider data which demonstrate the need for a VP in any adequate syntax of English, and the need for connectives which connect VPs and other expression types. At this point, the resources of $\mathrm{PL_{NQ}}$ no longer sufficed to provide adequate compositional translations of the fragment. So we introduced a new translation language, HOL_{NQ}. However, the general picture of the table above does not change; only the translation language itself changes.

Some discoveries:

- The task of giving a semantics via translation for natural language includes as a subtask the task of finding an adequate translation language.
- Given a typed language, function application is a powerful and very useful tool for modeling the derivation of the interpretation of a complex expression from the interpretations of its parts and their syntactic arrangement. To maintain a transparent interface between syntax and semantics, binary branching is preferable. Happily, this is supported by syntactic evidence.
- Syntax and semantics interact: Syntax forces us to introduce VP. The assumption of compositionality then forces us to translate and interpret this new category.
- We discovered that the "logical operators" of natural language can't always be translated directly by their formal counterparts. Their formal counterparts are all sentence connectives; but English has versions of these connectives for other types of expressions. However, we can use the familiar sentential connectives to derive appropriate translations for the differently-typed variants.

Some issues about translations: HOL_{NQ} provides multiple syntactically and semantically equivalent versions of many of its expressions. For example:

1. Let run be an HOL_{NQ} constant of type $\iota \to o$. Then $\mathrm{run} = \lambda X.\mathrm{run}(X)$
2. Let love be an HOL_{NQ} constant of type $\iota \to \iota \to o$. Then $\mathrm{love} = \lambda X.\lambda Y.\mathrm{love}(X, Y)$
3. Similarly, $\mathrm{love}(a) = \lambda Y.\mathrm{love}(a, Y)$
4. And $\mathrm{love}(\mathrm{jane}, \mathrm{george}) = ((\lambda X.\lambda Y.\mathrm{love}(X, Y))\mathrm{jane})\mathrm{george}$

Logically, both sides of the equations are considered equal, since η-equality (remember $\lambda X.\mathbf{A}X \to_\eta \mathbf{A}$, if $X \notin \mathbf{free}(\mathbf{A})$) is built into HOL_{NQ}. In fact all the right-hand sides are η-expansions of the left-hand sides. So you can use both, as you choose in principle.

But practically, you like to know which to give when you are asked for a translation? The answer depends on what you are using it for. Let's introduce a distinction between *reduced translations* and *unreduced translations*. An unreduced translation makes completely explicit the type assignment of each expression and the mode of composition of the translations of complex expressions, i.e. how the translation is derived from the translations of the parts. So, for example, if you have just offered a translation for a lexical item (say, and as a V^t connective), and now want to demonstrate how this lexical item works in a sentence, give the unreduced translation of the sentence in question and then demonstrate that it reduces to the desired reduced version.

The reduced translations have forms to which the deduction rules apply. So always use reduced translations for input in model generation: here, we are assuming that we have got the translation right, and that we know how to get it, and are interested in seeing what further deductions can be performed.

Where we are going: We will continue to enhance the fragment both by introducing additional types of expressions and by improving the syntactic analysis of the sentences we are dealing with. This will require further enrichments of the translation language. Next steps:

- Analysis of NP.

- Treatment of adjectives.
- Quantification

5.2 Fragment 4

New Data (more Noun Phrases)

▷ We want to be able to deal with the following sentences (without the "the-NP" trick)

1. *Peter loved the cat.*
2. but not **Peter loved the the cat.*
3. *John killed a cat with a hammer.*
4. *Peter loves every cat.*
5. *Every man loves a woman.*

SOME RIGHTS RESERVED ©: Michael Kohlhase 189 JACOBS UNIVERSITY

New Grammar in Fragment 4 (Common Noun Phrases)

▷ To account for the syntax we extend the functionality of noun phrases.

▷ **Definition 5.1** $\mathcal{F}_4$ adds the rules on the right to $\mathcal{F}_3$ (on the left):

S1.	S	$\to$	NP, VP_{+fin}
S2.	S	$\to$	S, Sconj
V1.	$VP_{\pm fin}$	$\to$	$V^i{}_{\pm fin}$
V2.	$VP_{\pm fin}$	$\to$	$V^t{}_{\pm fin}$, CNP
V3.	$VP_{\pm fin}$	$\to$	$VP_{\pm fin}$, VPconj$_{\pm fin}$
V4.	VP_{+fin}	$\to$	$BE_{=}$, NP
V5.	VP_{+fin}	$\to$	BE_{pred}, Adj.
V6.	VP_{+fin}	$\to$	didn't VP_{-fin}
N1.	NP	$\to$	N_{pr}
N2.	NP	$\to$	Pron

N3.	NP	$\to$	(Det) CNP
N4.	CNP	$\to$	N
N5.	CNP	$\to$	PP
P1.	PP	$\to$	P, NP
S3.	Sconj	$\to$	conj, S
V4.	VPconj$_{\pm fin}$	$\to$	conj, $VP_{\pm fin}$
L1.	P	$\to$	{with,of,...}

▷ **Definition 5.2** A common noun is a noun that describes a type, for example *woman*, or *philosophy* rather than an individual, such as *Amelia Earhart* (proper name).

SOME RIGHTS RESERVED ©: Michael Kohlhase 190 JACOBS UNIVERSITY

Notes:

- Parentheses indicate optionality of a constituent.
- We assume appropriate lexical insertion rules without specification.

If we assume that $\forall X.\mathrm{boy}(X) \Rightarrow \mathrm{run}(X)$ is an adequate translation of *Every boy runs*, and $\exists X.\mathrm{boy}(X) \wedge \mathrm{run}(X)$ one for *Some boy runs*, Then we obtain the translations of the determiners by by straightforward β-expansion.

Translation of Determiners and Quantifiers

▷ Idea: We establish the semantics of quantifying determiners by β-expansion.

1. assume that we are translating into a λ-calculus with quantifiers and that $\forall X.\mathsf{boy}(X) \Rightarrow \mathsf{run}(X)$ translates *Every boy runs*, and $\exists X.\mathsf{boy}(X) \wedge \mathsf{run}(X)$ for *Some boy runs*
2. $\forall\!\!\!\forall := \lambda P_{\iota\to o} Q_{\iota\to o}.(\forall X.P(X) \Rightarrow Q(X))$ for *every* (subset relation)
3. $\exists\!\!\!\exists := \lambda P_{\iota\to o} Q_{\iota\to o}.(\exists X.P(X) \wedge Q(X))$ for *some* (nonempty intersection)

▷ Problem: Linguistic Quantifiers take two arguments (restriction and scope), logical ones only one! (in logics, restriction is the universal set)

▷ We cannot treat *the* with regular quantifiers(new logical constant; see below)

▷ We translate the to $\tau := \lambda P_{\iota\to o} Q_{\iota\to o}.Q(\iota P)$, where ι is a new operator that given a set returns its (unique) member.

▷ **Example 5.3** This translates *The pope spoke* to $\tau(\mathsf{pope}, \mathsf{speak})$, which β-reduces to $\mathsf{speak}(\iota\mathsf{pope})$.

SOME RIGHTS RESERVED ©: Michael Kohlhase 191 JACOBS UNIVERSITY

Note that if we interpret objects of type $\iota \to o$ as sets, then the denotations of *boy* and *run* are sets (of boys and running individuals). Then the denotation of *every* is a relation between sets; more specifically the subset relation. As a consequence, *All boys run* is true if the set of boys is a subset of the set of running individuals. For *some* the relation is the non-empty intersection relation, *some boy runs* is true if the intersection of set of boys and the set of running individuals is non-empty.

Note that there is a mismatch in the "arity" of linguistic and logical notions of quantifiers here. Linguistic quantifiers take two arguments, the restriction (in our example *boy*) and the predication (*run*). The logical quantifiers only take one argument, the predication $\mathbf{A}$ in $\forall X.\mathbf{A}$. In a way, the restriction is always the universal set. In our model, we have modeled the linguistic quantifiers by adding the restriction with a connective (implication for the universal quantifier and conjunction for the existential one).

5.3 Quantifiers and Equality in Higher-Order Logic

There is a more elegant way to treat quantifiers in $\mathrm{HOL}^{\to}$. It builds on the realization that the λ-abstraction is the only variable binding operator we need, quantifiers are then modeled as second-order logical constants. Note that we do not have to change the syntax of $\mathrm{HOL}^{\to}$ to introduce quantifiers; only the "lexicon", i.e. the set of logical constants. Since Π^α and Σ^α are logical constants, we need to fix their semantics.

Higher-Order Abstract Syntax

▷ Idea: In $\mathrm{HOL}^{\to}$, we already have variable binder: λ, use that to treat quantification.

▷ **Definition 5.4** We assume logical constants Π^α and Σ^α of type $\alpha \to o \to o$.

Regain quantifiers as abbreviations:

$$(\forall X_\alpha.\mathbf{A}) := \Pi^\alpha(\lambda X_\alpha.\mathbf{A}) \qquad (\exists X_\alpha.\mathbf{A}) := \Sigma^\alpha(\lambda X_\alpha.\mathbf{A})$$

▷ **Definition 5.5** We must fix the semantics of logical constants:

1. $\mathcal{I}(\Pi^\alpha)(p) = \mathsf{T}$, iff $p(a) = \mathsf{T}$ for all a $\in \mathcal{D}_\alpha$ (i.e. if p is the universal set)
2. $\mathcal{I}(\Sigma^\alpha)(p) = \mathsf{T}$, iff $p(a) = \mathsf{T}$ for some a $\in \mathcal{D}_\alpha$ (i.e. iff p is non-empty)

▷ With this, we re-obtain the semantics we have given for quantifiers above:

$$\mathcal{I}_\varphi(\forall X_\iota.\mathbf{A}) = \mathcal{I}_\varphi(\Pi^\iota(\lambda X_\iota.\mathbf{A})) = \mathcal{I}(\Pi^\iota)(\mathcal{I}_\varphi(\lambda X_\iota.\mathbf{A})) = \mathsf{T}$$

iff $\mathcal{I}_\varphi(\lambda X_\iota.\mathbf{A})(a) = \mathcal{I}_{[a/X],\varphi}(\mathbf{A}) = \mathsf{T}$ for all $a \in \mathcal{D}_\alpha$

Equality

▷ "Leibniz equality" (Indiscernability) $\mathbf{Q}^\alpha \mathbf{A}_\alpha \mathbf{B}_\alpha = \forall P_{\alpha\to o}.P\mathbf{A} \Leftrightarrow P\mathbf{B}$

▷ not that $\forall P_{\alpha\to o}.P\mathbf{A} \Rightarrow P\mathbf{B}$ (get the other direction by instantiating P with Q, where $QX \Leftrightarrow \neg PX$)

▷ **Theorem 5.6** *If $\mathcal{M} = \langle\mathcal{D}, \mathcal{I}\rangle$ is a standard model, then $\mathcal{I}_\varphi(\mathbf{Q}^\alpha)$ is the identity relation on $\mathcal{D}_\alpha$.*

▷ **Notation 5.7** We write $\mathbf{A} = \mathbf{B}$ for $\mathbf{QAB}$ ($\mathbf{A}$ and $\mathbf{B}$ are equal, iff there is no property P that can tell them apart.)

▷ Proof:

P.1 $\mathcal{I}_\varphi(\mathbf{QAB}) = \mathcal{I}_\varphi(\forall P.P\mathbf{A} \Rightarrow P\mathbf{B}) = \mathsf{T}$, iff $\mathcal{I}_{\varphi,[r/P]}(P\mathbf{A} \Rightarrow P\mathbf{B}) = \mathsf{T}$ for all $r \in \mathcal{D}_{\alpha\to o}$.

P.2 For $\mathbf{A} = \mathbf{B}$ we have $\mathcal{I}_{\varphi,[r/P]}(P\mathbf{A}) = r(\mathcal{I}_\varphi(\mathbf{A})) = \mathsf{F}$ or $\mathcal{I}_{\varphi,[r/P]}(P\mathbf{B}) = r(\mathcal{I}_\varphi(\mathbf{B})) = \mathsf{T}$.

P.3 Thus $\mathcal{I}_\varphi(\mathbf{QAB}) = \mathsf{T}$.

P.4 Let $\mathcal{I}_\varphi(\mathbf{A}) \neq \mathcal{I}_\varphi(\mathbf{B})$ and $r = \{\mathcal{I}_\varphi(\mathbf{A})\}$

P.5 so $r(\mathcal{I}_\varphi(\mathbf{A})) = \mathsf{T}$ and $r(\mathcal{I}_\varphi(\mathbf{B})) = \mathsf{F}$

P.6 $\mathcal{I}_\varphi(\mathbf{QAB}) = \mathsf{F}$, as $\mathcal{I}_{\varphi,[r/P]}(P\mathbf{A} \Rightarrow P\mathbf{B}) = \mathsf{F}$, since $\mathcal{I}_{\varphi,[r/P]}(P\mathbf{A}) = r(\mathcal{I}_\varphi(\mathbf{A})) = \mathsf{T}$ and $\mathcal{I}_{\varphi,[r/P]}(P\mathbf{B}) = r(\mathcal{I}_\varphi(\mathbf{B})) = \mathsf{F}$. □

Alternative: HOL$^=$

▷ only one logical constant $q^\alpha \in \Sigma_{\alpha\to\alpha\to o}$ with $\mathcal{I}(q^\alpha)(a, b) = \mathsf{T}$, iff $a = b$.

▷ Definitions (D) and Notations (N)

N	$\mathbf{A}_\alpha = \mathbf{B}_\alpha$	for	$q^\alpha \mathbf{A}_\alpha \mathbf{B}_\alpha$
D	T	for	$q^o = q^o$
D	F	for	$\lambda X_o.T = \lambda X_o.X_o$
D	Π^α	for	$q^{(\alpha \to o)}(\lambda X_\alpha.T)$
N	$\forall X_\alpha.\mathbf{A}$	for	$\Pi^\alpha(\lambda X_\alpha.\mathbf{A})$
D	$\wedge$	for	$\lambda X_o.\lambda Y_o.\lambda G_{o\to o\to o}.GTT = \lambda G_{o\to o\to o}.GXY$
N	$\mathbf{A} \wedge \mathbf{B}$	for	$\wedge \mathbf{A}_o \mathbf{B}_o$
D	$\Rightarrow$	for	$\lambda X_o.\lambda Y_o.X = .X \wedge Y$
N	$\mathbf{A} \Rightarrow \mathbf{B}$	for	$\Rightarrow \mathbf{A}_o \mathbf{B}_o$
D	$\neg$	for	$q^o F$
D	$\vee$	for	$\lambda X_o.\lambda Y_o.\neg(\neg X \wedge \neg Y)$
N	$\mathbf{A} \vee \mathbf{B}$	for	$\vee \mathbf{A}_o \mathbf{B}_o$
D	$\exists X_\alpha.\mathbf{A}_o$	for	$\neg(\forall X_\alpha.\neg \mathbf{A})$
N	$\mathbf{A}_\alpha \neq \mathbf{B}_\alpha$	for	$\neg(q^\alpha \mathbf{A}_\alpha \mathbf{B}_\alpha)$

▷ yield the intuitive meanings for connectives and quantifiers.

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

We have managed to deal with the determiners *every* and *some* in a compositional fashion, using the familiar first order quantifiers. However, most natural language determiners cannot be treated so straightforwardly. Consider the determiner *most*, as in:

1. *Most boys run.*

There is clearly no simple way to translate this using $\forall$ or $\exists$ in any way familiar from first order logic. As we have no translation at hand, then, let us consider what the truth conditions of this sentence are.

Generalized Quantifiers

▷ Problem: What about *Most boys run.*: linguistically *most* behaves exactly like *every* or *some*.

▷ Idea: *Most boys run* is true just in case the number of boys who run is greater than the number of boys who do not run.

$$\#(\mathcal{I}_\varphi(\text{boy}) \cap \mathcal{I}_\varphi(\text{run})) > \#(\mathcal{I}_\varphi(\text{boy}) \backslash \mathcal{I}_\varphi(\text{run}))$$

▷ **Definition 5.8** $\#(A) > \#(B)$, iff there is no surjective function from B to A, so we can define

$$most' := \lambda AB.\neg(\exists F.\forall X.A(X) \wedge \neg B(X) \Rightarrow (\exists Y.A(Y) \wedge B(Y) \wedge X = F(Y)))$$

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

The NP *most boys* thus must denote something which, combined with the denotation of a VP, gives this statement. In other words, it is a function from sets (or, equivalently, from functions in $\mathcal{D}_{\iota\to o}$) to truth values which gives true just in case the argument stands in the relevant relation to the denotation of *boy*. This function is itself a characteristic function of a set of sets, namely:

$$\{X \mid \#(\mathcal{I}_\varphi(\text{boy}), X) > \#(\mathcal{I}_\varphi(\text{boy}) \backslash X)\}$$

Note that this is just the same kind of object (a set of sets) as we postulated above for the denotation of *every boy*.

Now we want to go a step further, and determine the contribution of the determiner *most* itself. *most* must denote a function which combines with a CNP denotation (i.e. a set of individuals or, equivalently, its characteristic function) to return a set of sets: just those sets which stand in the appropriate relation to the argument.

The function $most'$ is the characteristic function of a set of pairs:

$$\{\langle X,Y\rangle \,|\, \#(X\cap Y) > \#(X\backslash Y)\}$$

Conclusion: *most* denotes a relation between sets, just as *every* and *some* do. In fact, all natural language determiners have such a denotation. (The treatment of the definite article along these lines raises some issues to which we will return.)

Back to *every* and *some* (set characterization)

- ▷ We can now give an explicit set characterization of *every* and *some*:
  1. *every* denotes $\{\langle X,Y\rangle \,|\, X\subseteq Y\}$
  2. *some* denotes $\{\langle X,Y\rangle \,|\, X\cap Y\neq\emptyset\}$
- ▷ The denotations can be given in equivalent function terms, as demonstrated above with the denotation of *most*.

SOME RIGHTS RESERVED ©: Michael Kohlhase 196 JACOBS UNIVERSITY

5.4 Model Generation with Definite Descriptions

Semantics of Definite Descriptions

- ▷ Problem: We need a semantics for the determiner *the*, as in *the boy runs*
- ▷ Idea (Type): *the boy* behaves like a proper name (e.g. *Peter*), i.e. has type ι. Applying *the* to a noun (type $\iota\to o$) yields ι. So *the* has type $(\alpha\to o)\to\alpha$, i.e. it takes a set as argument.
- ▷ Idea (Semantics): *the* has the fixed semantics that this function returns the single member of its argument if the argument is a singleton, and is otherwise undefined. (new logical constant)
- ▷ **Definition 5.9** We introduce a new logical constant ι. $\mathcal{I}(\iota)$ is the function $f\in\mathcal{D}_{((\alpha\to o)\to\alpha)}$, such that $f(s)=a$, iff $s\in\mathcal{D}_{(\alpha\to o)}$ is the singleton set $\{a\}$, and is otherwise undefined.(remember that we can interpret predicates as sets)
- ▷ Axioms for ι:

$$\forall X_\alpha.X=\iota(=X)$$
$$\forall P,Q.Q((\iota P))\land(\forall X,Y.P(X)\land P(Y)\Rightarrow X=Y)\Rightarrow(\forall Z.P(Z)\Rightarrow Q(Z))$$

SOME RIGHTS RESERVED ©: Michael Kohlhase 197 JACOBS UNIVERSITY

Note: The first axiom is an equational characterization of ι. It uses the fact that the singleton set with member X can be written as $=X$ (or $\lambda Y.=XY$, which is η-equivalent). The second axiom says that if we have $Q(\iota P)$ and P is a singleton (i.e. all $X,Y\in P$ are identical), then Q holds on any member of P. Surprisingly, these two axioms are equivalent in $\mathrm{HOL}^{\to}$.

More Axioms for HOL$^{\to}$

- **Definition 5.10** unary conditional $\mathbf{w} \in \Sigma_{\alpha\to o\to\alpha}$
  $\mathbf{w}\mathbf{A}_o\mathbf{B}_\alpha$ means: "If $\mathbf{A}$, then $\mathbf{B}$"
- **Definition 5.11** binary conditional $\mathbf{if} \in \Sigma_{\alpha\to\alpha\to o\to\alpha}$
  $\mathbf{if}\mathbf{A}_o\mathbf{B}_\alpha\mathbf{C}_\alpha$ means: "if $\mathbf{A}$, then $\mathbf{B}$ else $\mathbf{C}$".
- **Definition 5.12** description operator $\iota \in \Sigma_{(\alpha\to o)\to\alpha}$
  if $\mathbf{P}$ is a singleton set, then $\iota\mathbf{P}_{\alpha\to o}$ is the element in $\mathbf{P}$,
- **Definition 5.13** choice operator $\gamma \in \Sigma_{(\alpha\to o)\to\alpha}$
  if $\mathbf{P}$ is non-empty, then $\gamma\mathbf{P}_{\alpha\to o}$ is an arbitrary element from $\mathbf{P}$
- **Definition 5.14 (Axioms for these Operators)**
  - unary conditional: $\forall\varphi_o.\forall X_\alpha.\varphi \Rightarrow \mathbf{w}\varphi X = X$
  - conditional: $\forall\varphi_o.\forall X_\alpha, Y_\alpha, Z_\alpha.(\varphi \Rightarrow \mathbf{if}\varphi XY = X) \wedge (\neg\varphi \Rightarrow \mathbf{if}\varphi ZX = X)$
  - description $\forall P_{\alpha\to o}.(\exists^1 X_\alpha.PX) \Rightarrow (\forall Y_\alpha.PY \Rightarrow \iota P = Y)$
  - choice $\forall P_{\alpha\to o}.(\exists X_\alpha.PX) \Rightarrow (\forall Y_\alpha.PY \Rightarrow \gamma P = Y)$

  Idea: These operators ensure a much larger supply of functions in Henkin models.

SOME RIGHTS RESERVED ©: Michael Kohlhase 198 JACOBS UNIVERSITY

More on the Description Operator

- ι is a weak form of the choice operator (only works on singleton sets)
- Alternative Axiom of Descriptions: $\forall X_\alpha.\iota^\alpha(=X) = X$.
  - use that $\mathcal{I}_{[\mathsf{a}/X]}(=X) = \{\mathsf{a}\}$
  - we only need this for base types $\neq o$
  - Define $\iota^o := =(\lambda X_o.X)$ or $\iota^o := \lambda G_{o\to o}.GT$ or $\iota^o := =(=T)$
  - $\iota^{\alpha\to\beta} := \lambda H_{(\alpha\to\beta)\to o}X_\alpha.\iota^\beta(\lambda Z_\beta.(\exists F_{\alpha\to\beta}.(HF) \wedge (FX) = Z))$

SOME RIGHTS RESERVED ©: Michael Kohlhase 199 JACOBS UNIVERSITY

To obtain a model generation calculus for HOL_{NQ} with descriptions, we could in principle add one of these axioms to the world knowledge, and work with that. It is better to have a dedicated inference rule, which we present here.

A Model Generation Rule for ι

▷

$$\frac{\begin{array}{c} P(c)^{\mathsf{T}} \\ Q(\iota P)^{\alpha} \end{array} \quad \mathcal{H} = \{c, a_1, \ldots, a_n\}}{\begin{array}{c} Q(c)^{\alpha} \\ P(a_1) \Rightarrow c = a_1{}^{\mathsf{t}} \\ \vdots \\ P(a_n) \Rightarrow c = a_n{}^{\mathsf{t}} \end{array}} \, \mathsf{RM}{:}\iota$$

▷ Intuition: If we have a member c of P and $Q(\iota P)$ is defined (it has truth value $\alpha \in \{\mathsf{T}, \mathsf{F}\}$), then P must be a singleton (i.e. all other members X of P are identical to c) and Q must hold on c. So the rule RM:ι forces it to be by making all other members of P equal to c.

SOME RIGHTS RESERVED ©: Michael Kohlhase 200 JACOBS UNIVERSITY

Mary owned a lousy computer. The hard drive crashed.

$$\begin{array}{c}
\forall X.\mathsf{comp}(X) \Rightarrow (\exists Y.\mathsf{hd}(Y) \wedge \mathsf{part_of}(Y, X))^{\mathsf{t}} \\
\boxed{\exists X.\mathsf{comp}(X) \wedge \mathsf{lousy}(X) \wedge \mathsf{own}(\mathsf{mary}, X)^{\mathsf{t}}} \\
\mathsf{comp}(c)^{\mathsf{t}} \\
\mathsf{lousy}(c)^{\mathsf{t}} \\
\mathsf{own}(\mathsf{mary}, c)^{\mathsf{t}} \\
\mathsf{hd}(d)^{\mathsf{t}} \\
\mathsf{part_of}(d, c)^{\mathsf{t}} \\
\boxed{\mathsf{crash}(\iota \mathsf{hd})^{\mathsf{t}}} \\
\mathsf{crash}(d)^{\mathsf{t}} \\
\mathsf{hd}(\mathsf{mary}) \Rightarrow \mathsf{mary} = d^{\mathsf{t}} \\
\mathsf{hd}(c) \Rightarrow c = d^{\mathsf{t}}
\end{array}$$

SOME RIGHTS RESERVED ©: Michael Kohlhase 201 JACOBS UNIVERSITY

Definition 5.15 In this example, we have a case of what is called a bridging reference, following H. Clark (1977): intuitively, we build an inferential bridge from the computer whose existence is asserted in the first sentence to the hard drive invoked in the second.

By incorporating world knowledge into the tableau, we are able to model this kind of inference, and provide the antecedent needed for interpreting the definite.

Now let us use the RM:ι rule for interpreting *The dog barks* in a situation where there are two dogs: Fido and Chester. Intuitively, this should lead to a closed tableau, since the uniqueness presupposition is violated. Applying the rules, we get the following tableau.

Another Example *The dog barks*

▷ In a situation, where there are two dogs: Fido and Chester

$$\begin{array}{c|c} \multicolumn{2}{c}{\mathsf{dog}(\mathsf{fido})^{\mathsf{t}}} \\ \multicolumn{2}{c}{\mathsf{dog}(\mathsf{chester})^{\mathsf{t}}} \\ \multicolumn{2}{c}{\mathsf{bark}(\iota\mathsf{dog})^{\mathsf{t}}} \\ \multicolumn{2}{c}{\mathsf{bark}(\mathsf{fido})^{\mathsf{t}}} \\ \multicolumn{2}{c}{\mathsf{dog}(\mathsf{chester}) \Rightarrow \mathsf{chester} = \mathsf{fido}^{\mathsf{t}}} \\ \mathsf{dog}(\mathsf{chester})^{\mathsf{f}} & \mathsf{chester} = \mathsf{fido}^{\mathsf{t}} \\ \bot & \end{array} \tag{1}$$

▷ Note that none of our rules allows us to close the right branch, since we do not know that Fido and Chester are distinct. Indeed, they could be the same dog (with two different names). But we can eliminate this possibility by adopting a new assumption.

SOME RIGHTS RESERVED ©: Michael Kohlhase 202 JACOBS UNIVERSITY

5.5 Model Generation with a Unique Name Assumption

Normally (i.e. in natural languages) we have the default assumption that names are unique. In principle, we could do this by adding axioms of the form $n = m^{\mathsf{F}}$ to the world knowledge for all pairs of names n and m. Of course the cognitive plausibility of this approach is very questionable. As a remedy, we can build a Unique-Name-Assumption (UNA) into the calculus itself.

Model Generation with Unique Name Assumption (UNA)

▷ Problem: Names are unique (usually in natural language)

▷ Idea: Add background knowledge of the form $n = m^{\mathsf{F}}$ (n and m names)

▷ Better Idea: Build UNA into the calculus: partition the Herbrand base $\mathcal{H} = \mathcal{U} \cup \mathcal{W}$ into subsets $\mathcal{U}$ for constants with a unique name assumption, and $\mathcal{W}$ without. (treat them differently)

▷ **Definition 5.16 (Model Generation with UNA)** We add the following two rules to the RM calculus to deal with the unique name assumption.

$$\frac{\begin{array}{c} a = b^{\mathsf{t}} \\ \mathbf{A}^{\alpha} \end{array} \quad a \in \mathcal{W} \quad b \in \mathcal{H}}{[b/a](\mathbf{A})^{\alpha}}\,\mathsf{RM{:}subst} \qquad \frac{a = b^{\mathsf{t}} \quad a, b \in \mathcal{U}}{\bot}\,\mathsf{RM{:}una}$$

SOME RIGHTS RESERVED ©: Michael Kohlhase 203 JACOBS UNIVERSITY

In effect we make the $\mathcal{T}_0$subst rule directional; it only allows the substitution for a constant without the unique name assumption. Finally, RM:una mechanizes the unique name assumption by allowing a branch to close if two different constants with unique names are claimed to be equal. All the other rules in our model generation calculus stay the same. Note that with RM:una, we can close the right branch of tableau (1), in accord with our intuition about the discourse.

5.6 Davidsonian Semantics: Treating Verb Modifiers

Event semantics: Davidsonian Systems

- Problem: How to deal with argument structure of (action verbs) and their modifiers
  - *John killed a cat with a hammer.*
- Idea: Just add an argument to kill for express the means
- Problem: But there may be more modifiers
  1. *Peter killed the cat in the bathroom with a hammer.*
  2. *Peter killed the cat in the bathroom with a hammer at midnight.*

  So we would need a lot of different predicates for the verb *killed*.(impractical)
- Idea: Extend the argument structure of (action) verbs contains a 'hidden' argument, the event argument, then tread modifiers as predicates over events [Dav67a].
- **Example 5.17** 1. $\exists e.\exists x, y.\mathsf{br}(x) \wedge \mathsf{hammer}(y) \wedge \mathsf{kill}(e, \mathsf{peter}, \iota\mathsf{cat}) \wedge \mathsf{in}(e, x) \wedge \mathsf{with}(e, y)$
  2. $\exists e.\exists x, y.\mathsf{br}(x) \wedge \mathsf{hammer}(y) \wedge \mathsf{kill}(e, \mathsf{peter}, \iota\mathsf{cat}) \wedge \mathsf{in}(e, x) \wedge \mathsf{with}(e, y) \wedge \mathsf{at}(e, 24:00)$

SOME RIGHTS RESERVED ©: Michael Kohlhase 204 JACOBS UNIVERSITY

Event semantics: Neo-Davidsonian Systems

- Idea: Take apart the Davidsonian predicates even further, add event participants via thematic roles (from [Par90]).
- **Example 5.18** Translate *John killed a cat with a hammer.* as $\exists e.\exists x.\mathsf{hammer}(x) \wedge \mathsf{killing}(e) \wedge \mathsf{ag}(e, \mathsf{peter}) \wedge \mathsf{pat}(e, \iota\mathsf{cat}) \wedge \mathsf{with}(e, x)$
- Further Elaboration: Events can be broken down into sub-events and modifiers can predicate over sub-events.
- **Example 5.19** The "process" of climbing Mt. Everest starts with the "event" of (optimistically) leaving the base camp and culminates with the "achievement" of reaching the summit (being completely exhausted).
- Note: This system can get by without functions, and only needs unary and binary predicates. (well-suited for model generation)

SOME RIGHTS RESERVED ©: Michael Kohlhase 205 JACOBS UNIVERSITY

Event types and properties of events

- **Example 5.20 (Problem)** Some (temporal) modifiers are incompatible with some events, e.g. in English progressive:
  1. *He is eating a sandwich* and *He is pushing the cart.*, but not

2. *_He is being tall._ or *_He is finding a coin._

▷ **Definition 5.21 (Types of Events)** There are different types of events that go with different temporal modifiers. [Ven57]distinguishes

1. state s: e.g. _know the answer_, _stand in the corner_
2. process es: e.g._run_, _eat_, _eat apples_, _eat soup_
3. accomplishment s: e.g. _run a mile_, _eat an apple_, and
4. achievement s: e.g. _reach the summit_

Observations on Example 5.20:

▷ 1. activities and accomplishments appear in the progressive (1),
2. states and achievements do not (2).

The for/in Test:

▷ 1. states and activities, but not accomplishments and achievements are compatible with _for_-adverbials
2. whereas the opposite holds for in-adverbials (5).

▷ **Example 5.22** 1. _run a mile in an hour_ vs. *_run a mile for an hour_, but
2. *_reach the summit for an hour_ vs _reach the summit in an hour_

SOME RIGHTS RESERVED ©: Michael Kohlhase 206 JACOBS UNIVERSITY

6 Quantifier Scope Ambiguity and Underspecification

6.1 Scope Ambiguity and Quantifying-In

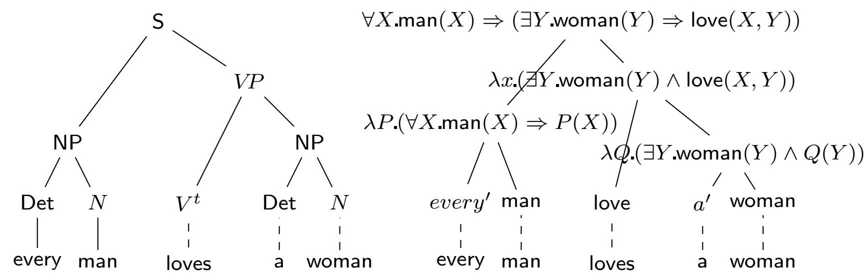
Now that we are able to interpret sentences with quantification objects and subjects, we can address the issue of quantifier scope ambiguities.

Quantifier Scope Ambiguities: Data

▷ Consider the following sentences:

1. _Every man loves a woman_ (Britney Spears or his mother?)
2. _Most Europeans speak two languages._
3. _Some student in every course sleeps in every class at least some of the time._

▷ **Example 6.1** We can represent the "wide-scope" reading with our methods



Question: how to map an unambiguous input structure to multiple translations.

©: Michael Kohlhase 207

This is a correct representation of one of the possible meanings of the sentence - namely the one where the quantifier of the object-NP occurs inside the scope of the quantifier of the subject-NP. We say that the quantifier of the object-NP has narrow scope while the quantifier of the subject-NP has wide scope. But the other reading is not generated here! This means our algorithm doesn't represent the linguistic reality correctly.

What's the problem?: This is because our approach so far constructs the semantics deterministically from the syntactic analysis. Our analysis simply isn't yet able to compute two different meanings for a syntactically unambiguous sentence. The reason why we only get the reading with wide scope for the subject is because in the semantic construction process, the verb semantics is first combined with the object semantics, then with that of the subject. And given the order of the -prefixes in our semantic representations, this eventually transports the object semantics inside the subject's scope.

A Closer Look: To understand why our algorithm produces the reading it does (and not the other alternative), let us have a look at the order of applications in the semantic representation as it is before we start β-reducing. To be able to see the order of applications more clearly, we abbreviate the representations for the determiners. E.g. we write instead of . We will of course have to expand those abbreviations at some point when we want to perform β-reduction.

In the VP node for *loves a woman* we have $(\lambda FX.\lambda Q.(\exists Y.\mathrm{woman}(Y) \wedge (QY)))\mathrm{love}$ and thus the sentence representation is

$$(\lambda P.(\forall X.\mathrm{man}(X) \Rightarrow P(X)))((\lambda FX.\lambda Q.(\exists Y.\mathrm{woman}(Y) \wedge (QY)))\mathrm{love})$$

The resulting expression is an application of form ⟨every man⟩(⟨a woman⟩(⟨loves⟩)). I.e. the universal quantifier occurs in the functor (the translation of the subject NP), and the existential quantifier occurs in the argument (corresponding to the VP). The scope relations in the β-reduced result reflect the structure in this application.

With some imagination we can already guess what an algorithm would have to do in order to produce the second reading we've seen above (where the subject-NP has narrow scope): It would somehow have to move the *a woman* part in front of the *every*. Something like ⟨a woman⟩(⟨every man⟩(⟨loves⟩)) would do.

▷ Storing and Quantifying In

▷ Analyssis: The sentence meaning is of the form ⟨every man⟩(⟨a woman⟩(⟨loves⟩))

▷ Idea: Somehow have to move the *a woman* part in front of the *every* to obtain

⟨a woman⟩(⟨every man⟩(⟨loves⟩))

▷ More concretely: Let's try *A woman - every man loves her.*
In semantics construction, apply *a woman* to *every man loves her*.
So *a woman* outscopes *every man*.

▷ Problem: How to represent pronouns and link them to their antecedents

▷ **STORE** is an alternative translation rule. Given a node with an NP daughter, we can translate the node by passing up to it the translation of its non-NP daughter, and putting the translation of the NP into a store, for later use.

▷ The **QI** rule allows us to empty out a non-empty store.

SOME RIGHTS RESERVED ©: Michael Kohlhase 208 JACOBS UNIVERSITY

To make the second analysis work, one has to think of a representation for the pronoun, and one must provide for linking the pronoun to its antecedent "a woman" later in the semantic construction process. Intuitively, the pronoun itself is semantically empty. Now Montague's idea essentially was to choose a new variable to represent the pronoun. Additionally, he had to secure that this variable ends up in the right place after -reduction.

Storing and Quantifying In (Technically)

▷ **Definition 6.2 STORE**$(NP, \Phi) \longrightarrow (\Phi, \Sigma * NP)$, where $\Sigma * NP$ is the result of adding NP to Σ, i.e. $\Sigma * NP = \Sigma \cup \{NP\}$; we will assume that NP is not already in Σ, when we use the $*$ operator.

▷ **Definition 6.3 QI**$(\langle \Phi, \Sigma * NP \rangle) \rightarrow \langle NP \oplus \Phi, \Sigma \rangle$ where $\oplus$ is either function application or function composition.

▷ Nondeterministic Semantics Construction: Adding rules gives us more choice

1. **Rule C (simple combination)** If A is a node with daughters B and C, and the translations of B and of C have empty stores, then A translates to $B' \oplus C'$. Choice of rule is determined by types.
2. **STORE** If A is a node with daughters B and C, where:
   - ▷ B is an NP with translation B' and
   - ▷ C translates to (C', Σ)

   then A may translate to **STORE**(B', C')

Note that **STORE** may be applied whether or not the stores of the constituent nodes are empty.

SOME RIGHTS RESERVED ©: Michael Kohlhase 209 JACOBS UNIVERSITY

We now have more than one way to translate a branching node, but the choice is partly constrained by whether or not the daughters of the node have empty stores. We have the following two options for translating a branching node. (Note: To simplify the notation, let us adopt the following convention: If the translation of A has an empty store, we omit reference to the store in representing the translation of A, $\mathbf{A}$.)

Application of **STORE** must always eventually be followed by application of **QI**. (Note that **QI** is not a translation rule, but a sort of transformation on translations.) But when must **QI** be applied? There are two cases:

1. The process of semantic composition must conclude with an empty store.

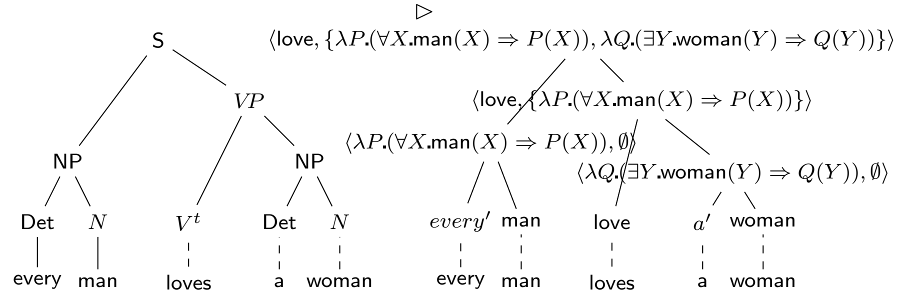
2. If A is a branching node one of whose daughters is a conjunction (i.e. *and* or *or*, the translation of A is given by Rule $\mathbf{C}$).

The first of these rules has the effect that if the initial translation of S has a non-empty store, we must apply **QI** as many times as needed to empty the store. The second rule has the effect of requiring the same thing where *and* attaches to any constituent.

We assume that our syntax returned the syntax tree on the left. Just as before; the only difference is that we have a different syntax-semantics interface. The NP nodes get their semantics $\mathbf{A} := \lambda P.(\forall X.\mathrm{man}(X) \Rightarrow P(X))$ and $\mathbf{B} := \lambda Q.(\exists Y.\mathrm{woman}(Y) \Rightarrow Q(Y))$ as before. Similarly, the V^t node has the value love. To compute the semantics of the VP nodes, we use the rule **STORE** and obtain $\langle \mathrm{love}, \{\mathbf{A}\}\rangle$ and similarly $\langle \mathrm{love}, \{\mathbf{A}, \mathbf{B}\}\rangle$ for the for the S node, thus we have the following semantics tree

Quantifying in Practice: *Every man loves a woman*

▷



▷ Continue with **QI** applications: first retrieve $\lambda Q.(\exists Y.\mathrm{woman}(Y) \Rightarrow Q(Y))$

$$\begin{array}{ll}
 & \langle \mathrm{love}, \{\lambda P.(\forall X.\mathrm{man}(X) \Rightarrow P(X)), \lambda Q.(\exists Y.\mathrm{woman}(Y) \Rightarrow Q(Y))\}\rangle \\
\rightarrow_{QI} & \langle \circ(\lambda P.(\forall X.\mathrm{man}(X) \Rightarrow P(X)))\mathrm{love}, \{\lambda Q.(\exists Y.\mathrm{woman}(Y) \Rightarrow Q(Y))\}\rangle \\
\rightarrow_{\beta} & \langle \lambda Z.(\lambda P.(\forall X.\mathrm{man}(X) \Rightarrow P(X)))(\mathrm{love}Z), \{\lambda Q.(\exists Y.\mathrm{woman}(Y) \Rightarrow Q(Y))\}\rangle \\
\rightarrow_{\beta} & \langle \lambda Z.(\forall X.\mathrm{man}(X) \Rightarrow \mathrm{love}ZX), \{\lambda Q.(\exists Y.\mathrm{woman}(Y) \Rightarrow Q(Y))\}\rangle \\
\rightarrow_{QI} & \langle (\lambda Q.(\exists Y.\mathrm{woman}(Y) \Rightarrow Q(Y)))(\lambda Z.(\forall X.\mathrm{man}(X) \Rightarrow \mathrm{love}ZX)), \emptyset\rangle \\
\rightarrow_{\beta} & \langle \exists Y.\mathrm{woman}(Y) \Rightarrow (\lambda Z.(\forall X.\mathrm{man}(X) \Rightarrow \mathrm{love}ZX))Y, \emptyset\rangle \\
\rightarrow_{\beta} & \langle \exists Y.\mathrm{woman}(Y) \Rightarrow (\forall X.\mathrm{man}(X) \Rightarrow \mathrm{love}YX), \emptyset\rangle
\end{array}$$

SOME RIGHTS RESERVED ©: Michael Kohlhase 210 JACOBS UNIVERSITY

This reading corresponds to the wide scope reading for *a woman*. If we had used the **QI** rules the other way around, first extracting *a woman* and then *every man*, we would have gotten the reading with wide scope for *every man* in the same way.

6.2 Type Raising for non-quantificational NPs

There is now a discrepancy in the type assigned to subject NPs with quantificational determiners, and subject NPs consisting of a proper name or a definite NP. This corresponds to a discrepancy in the roles of the NP and VP in interpretation: where the NP is quantificational, it takes the VP as argument; where the NP is non-quantificational, it constitutes the argument of the VP. This discrepancy can be resolved by type raising.

Proper names

▷ Problem: Subject NPs with quantificational determiners have type $(\iota \rightarrow o) \rightarrow$

o (and are applied to the VP) whereas subject NPs with proper names have type ι (argument to the VP)

▷ Idea: *John runs* translates to run(john), where run $\in \Sigma_{\iota\to o}$ and john $\in \Sigma_\iota$.
Now we β-expand over the VP yielding $(\lambda P_{\iota\to o}.P(\text{john}))$run
$\lambda P_{\iota\to o}.P(\text{john})$ has type $(\iota \to o) \to o$ and can be applied to the VP run.

▷ **Definition 6.4** If $c \in \Sigma_\alpha$, then type-raising c yields $\lambda P_{\alpha\to o}.Pc$.

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

Definite NPs

▷ Problem: On our current assumptions, $the' = \iota$, and so for any definite NP *the N*, its translation is ιN, an expression of type ι.

▷ Idea: Type lift just as we did with proper names: ιN type lifts to $\lambda P.P(\iota N)$, so $the' = \lambda PQ.Q(\iota P)$

▷ Advantage: This is a "generalized quantifier treatment": the' treated as denoting relations between sets.

▷ Solution by Barwise&Cooper 1981: For any a $\in \mathcal{D}_{(\iota\to o)}$: $\mathcal{I}(the')(\mathsf{a}) = \mathcal{I}(every')(\mathsf{a})$ if $\#(\mathsf{a}) = 1$, undefined otherwise
So the' is that function in $\mathcal{D}_{(\iota\to o)\to(\iota\to o)\to o}$ s.t for any $A, B \in \mathcal{D}_{(\iota\to o)}$ if $\#(A) = 1$ then $the'(A, B) = \mathsf{T}$ if $A \subseteq B$ and $the'(A, B) = \mathsf{F}$ if $A \not\subseteq B$ otherwise undefined

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

This treatment of *the* is completely equivalent to the ι treatment, guaranteeing that, for example, the sentence *The dog barked* has the value true if there is a unique dog and that dog barked, the value false if there is a unique dog and that dog did not bark, and, if there is no dog or more than one dog, has an undefined value. So we can indeed treat *the* as a generalized quantifier.

However, there are two further considerations.

1. The function characterized above cannot straightforwardly be represented as a relation on sets. We might try the following:

$$\{\langle X, Y\rangle \mid \#(X) = 1 \;\&\; X \subseteq Y\}$$

   Now, consider a pair $\langle X, Y\rangle$ which is not a member of the set. There are two possibilities: either $\#(X) \neq 1$ or $\#(X) = 1$ and $X \not\subseteq Y$. But we want to treat these two cases differently: the first leads to undefinedness, and the second to falsity. But the relation does not capture this difference.

2. If we adopt a generalized quantifier treatment for the definite article, then we must always treat it as an expression of type $\iota \to o \to o$. If we maintain the ι treatment, we can choose, for any given case, whether to treat a definite NP as an expression of type ι, or to type lift the NP to $\iota \to o \to o$. This flexibility will be useful (particularly for purposes of model generation). Consequently, we will maintain the ι treatment.

   These considerations may appear purely technical in nature. However, there is a significant philosophical literature on definite descriptions, much of which focuses on the question of whether these expressions are referential or quantificational. Many have the view that definite descriptions are ambiguous between a referential and a quantificational interpretation, which in fact differentiates them from other NPs, and which is captured to some extent by our proposed treatment.

Our discussion of quantification has led us to a treatment of quantified NPs as expressions of type $(\iota \to o) \to o$. Moreover, we now have the option of treating proper names and definite descriptions as expressions of this higher type too. This change in the type of NPs causes no difficulties with composition in the intransitive sentences considered so far, although it requires us to take the translation of the VP as argument to the subject NP.

Problems with Type raised NPs

- ▷ Problem: We have type-raised NPs, but consider transitive verbs as in *Mary loves most cats*. love is of type $\iota \to \iota \to o$ while the object NP is of type $(\iota \to o) \to o$ (application?)
- ▷ Another Problem: We encounter the same problem in the sentence *Mary loves John* if we choose to type-lift the NPs.
- ▷ Idea: Change the type of the transitive verb to allow it to "swallow" the higher-typed object NP.
- ▷ Better Idea: adopt a new rule for semantic composition for this case
- ▷ Remember: $loves'$ is a function from individuals (e.g. *John*) to properties (in the case of the VP *loves John*, the property *X loves John* of X).

SOME RIGHTS RESERVED ©: Michael Kohlhase 213 JACOBS UNIVERSITY

In our type-raised semantics, the denotation of NPs is a function f from properties to truth values. So if we compose an NP denotation with a transitive verb denotation, we obtain a function from individuals to truth values, i.e. a property.

Type raised NPs and Function Composition

- ▷ We can extend $\mathrm{HOL}^{\to}$ by a constant $\circ_{(\beta\to\gamma)\to(\alpha\to\beta)\to\alpha\to\gamma}$ by setting $\circ := \lambda FGX.F(G(X))$ thus

$$\circ g f \to_\beta \lambda X.g(f(X)) \quad \text{and} \quad (\circ g f)a \to_\beta g(f(a))$$

In our example, we have

$$\begin{aligned} \circ(\lambda P.P(\mathrm{john}))\mathrm{love} &=_{Def} (\lambda FGX.F(G(X)))(\lambda P.P(\mathrm{john}))\mathrm{love} \\ &\to_\beta (\lambda GX.(\lambda P.P(\mathrm{john}))G(X))\mathrm{love} \\ &\to_\beta \lambda X.(\lambda P.P(\mathrm{john}))(\mathrm{love}X) \\ &\to_\beta \lambda X.\mathrm{love}(X,\mathrm{john}) \end{aligned}$$

SOME RIGHTS RESERVED ©: Michael Kohlhase 214 JACOBS UNIVERSITY

Definition 6.5 (Function Composition) Let $f: A \to B$ and $g: B \to C$ be functions, then we call the function $h: A \to C$ such that $h(a) = g(f(a))$ for all $a \in A$ the composition of g and f and write it as gf (read this as " g after f").

6.3 Dealing with Quantifier Scope Ambiguity: Cooper Storage

Type raising transitive verbs

- We need transitive verbs to combine with quantificational objects of type $(\iota \to o) \to o$ but ...
- We still ultimately want their "basic" translation to be type $\iota \to \iota \to o$, i.e. something that denotes a relation between individuals.
- We do this by starting with the basic translation, and raising its type. Here is what we'll end up with, for the verb *like*:

  $$\lambda PY.P(\lambda X.\mathrm{like}(X,Y))$$

  where P is a variable of type $(\iota \to o) \to o$ and X, Y are variables of type ι. (For details on how this is derived, see Cann pp.178-179.)

SOME RIGHTS RESERVED ©: Michael Kohlhase 215 JACOBS UNIVERSITY

We have already seen the basic idea that we will use here. We will proceed with compositional translation in the familiar way. But when we encounter a QNP, we will put its translation aside, in a *store*. To make sure we know where it came from, we will put a "place holder" in the translation, and co-index the stored NP with its place holder. When we get to the S node, we will have a representation which we can re-combine with each of the stored NPs in turn. The order in which we re-combine them will determine the scopal relations among them.

Cooper Storage

- A *store* consists of a "core" semantic representation, computed in the usual way, plus the representations of quantifiers encountered in the composition so far.
- **Definition 6.6** A store is an n-place sequence. The first member of the sequence is the core semantic representation. The other members of the sequence (if any) are pairs (β, i) where:
  - β is a QNP translation and
  - i is an index, which will associate the NP translation with a free variable in the core semantic translation.

  We call these pairs binding operator s (because we will use them to bind free variables in the core representation).
- The elements in the store are written enclosed in angled brackets. However, we will often have a store which consists of only one element, the core semantic representation. This is because QNPs are the only things which add elements beyond the core representation to the store. So we will adopt the convention that when the store has only one element, the brackets are omitted.

SOME RIGHTS RESERVED ©: Michael Kohlhase 216 JACOBS UNIVERSITY

How we put QNPs in the store

- ▷ **Storage Rule**
  If the store $\langle\varphi,(\beta,j),\ldots,(\gamma,k)\rangle$ is a possible translation for a QNP, then the store
  $$\langle\lambda P.P(X_i)(\varphi,i)(\beta,j),\ldots,(\gamma,k)\rangle$$
  where i is a new index, is also a possible translation for that QNP.
- ▷ This rule says: if you encounter a QNP with translation φ, you can replace its translation with an indexed place holder of the same type, $\lambda P.P(X_i)$, and add φ to the store, paired with the index i. We will use the place holder translation in the semantic composition of the sentence.

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

Working with stores

- ▷ Working out the translation for *Every student likes some professor.*

$$\begin{array}{lll}
NP_1 & \rightarrow & \lambda P.(\exists X.\mathsf{prof}(X)\wedge P(X)) \textbf{ or } \langle\lambda Q.Q(X_1),(\lambda P.(\exists X.\mathsf{prof}(X)\wedge P(X)),1)\rangle\\
V_t & \rightarrow & \lambda RY.R(\lambda Z.\mathsf{like}(Z,Y))\\
VP & \rightarrow & \text{(Combine core representations by FA; pass store up)*}\\
 & \rightarrow & \langle\lambda Y.\mathsf{like}(X_1,Y),(\lambda P.(\exists X.\mathsf{prof}(X)\wedge P(X)),1)\rangle\\
NP_2 & \rightarrow & \lambda P.(\forall Z.\mathsf{stud}(Z)\Rightarrow P(Z)) \textbf{ or } \langle\lambda R.R(X_2),(\lambda P.(\forall Z.\mathsf{stud}(Z)\Rightarrow P(Z)),2)\rangle\\
S & \rightarrow & \text{(Combine core representations by FA; pass stores up)**}\\
 & \rightarrow & \langle\mathsf{like}(X_1,X_2),(\lambda P.(\exists X.\mathsf{prof}(X)\wedge P(X)),1),(\lambda P.(\forall Z.\mathsf{stud}(Z)\Rightarrow P(Z)),2)\rangle
\end{array}$$

* Combining V_t with place holder

1. $(\lambda RY.R(\lambda Z.\mathsf{like}(Z,Y)))(\lambda Q.Q(X_1))$
2. $\lambda Y.(\lambda Q.Q(X_1))(\lambda Z.\mathsf{like}(Z,Y))$
3. $\lambda Y.(\lambda Z.\mathsf{like}(Z,Y))X_1$
4. $\lambda Y.\mathsf{like}(X_1,Y)$

** Combining VP with place holder

1. $(\lambda R.R(X_2))(\lambda Y.\mathsf{like}(X_1,Y))$
2. $(\lambda Y.\mathsf{like}(X_1,Y))X_2$
3. $\mathsf{like}(X_1,X_2)$

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

Retrieving NPs from the store

- ▷ Retrieval: Let σ_1 and σ_2 be (possibly empty) sequences of binding operators. If the store $\langle\varphi,\sigma_1,\sigma_2,(\beta,i)\rangle$ is a translation of an expression of category S, then the store $\langle\beta(\lambda X_1.\varphi),\sigma_1,\sigma_2\rangle$ is also a translation of it.
- ▷ What does this say?: It says: suppose you have an S translation consisting of a core representation (which will be of type o) and one or more indexed QNP translations. Then you can do the following:

  1. Choose one of the QNP translations to retrieve.

2. Rewrite the core translation, λ-abstracting over the variable which bears the index of the QNP you have selected. (Now you will have an expression of type $\iota \to o$.)
3. Apply this λ expression to the QNP translation (which is of type $(\iota \to o) \to o$).

SOME RIGHTS RESERVED ©: Michael Kohlhase 219 JACOBS UNIVERSITY

Example: *Every student likes some professor.*

1. Retrieve *every student*
   (a) $(\lambda Q.(\forall Z.\mathsf{stud}(Z) \Rightarrow Q(Z)))(\lambda X_2.\mathsf{like}(X_1, X_2))$
   (b) $\forall Z.\mathsf{stud}(Z) \Rightarrow (\lambda X_2.\mathsf{like}(X_1, X_2))Z$
   (c) $\forall Z.\mathsf{stud}(Z) \Rightarrow \mathsf{like}(X_1, Z)$
2. Retrieve *some professor*
   (a) $(\lambda P.(\exists X.\mathsf{prof}(X) \wedge P(X)))(\lambda X_1.(\forall Z.\mathsf{stud}(Z) \Rightarrow \mathsf{like}(X_1, Z)))$
   (b) $\exists X.\mathsf{prof}(X)(\lambda X_1.(\forall Z.\mathsf{stud}(Z) \Rightarrow \mathsf{like}(X_1, Z)))X$
   (c) $\exists X.\mathsf{prof}(X) \wedge (\forall Z.\mathsf{stud}(Z) \Rightarrow \mathsf{like}(X, Z))$

SOME RIGHTS RESERVED ©: Michael Kohlhase 220 JACOBS UNIVERSITY

The Cooper storage approach to quantifier scope ambiguity basically moved the ambiguity problem into the syntax/semantics interface: from a single syntactic tree, it generated multiple unambiguous semantic representations. We will now come to an approach, which does not force the system to commit to a particular reading so early.

6.4 Underspecification

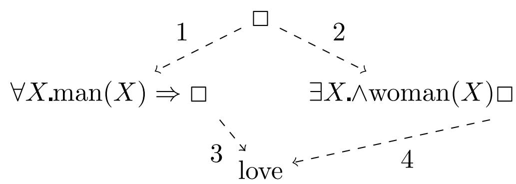
In this subsection we introcude Johan Bos' "Hole Semantics", since this is possibly the simplest underspecification framework around. The main idea is that the result of the translation is a "quasi-logical form" (QLF), i.e. a representation that represents all possible readings. This QLF can then be used for semantic/pragmatic analysis.

6.4.1 Unplugging Predicate Logic

The problem we need to solve for our QLF is that regular logical formulae, such as

$$\forall X.\mathrm{man}(X) \Rightarrow (\exists Y.\mathrm{woman}(Y) \Rightarrow \mathrm{love}(Y, X))$$

fully specifies the scope relation between the quantifiers. The idea behind "hole semantics" (and most other approaches to quantifier scope underspecification) is to "unplug" first-order logic, i.e. to take apart logical formulae into smaller parts, and add constraints on how the parts can be plugged together again. To keep track of where formulae have to be plugged together again, "hole semantics" uses the notion of "holes". Our example *Every man loves a woman* now has the following form:



The meaning of the dashed arrows is that the holes (depicted by □) can be filled by one of the formulas that are pointed to. The hole at the top of the graph serves as the representation of the whole sentence.

We can disambiguate the QLF by choosing an arc for every hole and plugging the respective formulae into the holes, collapsing the graph into a single logical formula. If we act on arcs 1 and 4, we obtain the wide-scope reading for *every man*, if we act on 2 and 3, we obtain the reading, where *a woman* outscopes *every man*. So much for the general idea, how can this be represented in logic?

6.4.2 PL_H a first-order logic with holes

The main idea is to label the holes and formulae, and represent the arcs as pairs of labels. To do this, we add holes to first-order logic, arriving at a logic PL_H. This can simply be done by reserving a lexical category $\mathcal{H} = \{h_0, h_1, h_2, \ldots\}$ of holes, and adding them as possible atomic formulae, so that $\forall X.\mathrm{man}(X) \Rightarrow h_1$ is a PL_H formula.

Using this, a QLF is a triple $\langle F, C \rangle$, where F is a set of labeled formulae of the form $\ell_i \colon \mathbf{A}_1$, where ℓ_i is taken from a set $\mathcal{L} = \{\ell_0, \ell_1, \ldots\}$ of labels, and $\mathbf{A}_i$ is a PL_H formula, and C is a set constraints of the form $\ell_i \leq h_j$. The underspecified representation above now has the form

$$\langle \{\ell_1 \colon \forall X.\mathrm{man}(X) \Rightarrow h_1, \ell_2 \colon \forall Y.\mathrm{woman}(Y) \Rightarrow h_2\}, \{\ell_1 \leq h_0, \ell_2 \leq h_0, \ell_3 \leq h_1, \ell_3 \leq h_2\} \rangle$$

Note that we always reserve the hole h_0 for the top-level hole, that represents the sentence meaning.

6.4.3 Plugging and Chugging

A plugging p for a QLF $\mathcal{Q}$ is now a mapping from the holes in $\mathcal{Q}$ to the labels in $\mathcal{Q}$ that satisfies the constraint C of $\mathcal{Q}$, i.e. for all holes h in $\mathcal{Q}$ we have $h \leq p(h) \in C$. Note that the set of admissible pluggings can be computed from the constraint alone in a straightforward manner. Acting on the pluggings yields a logical formula. In our example, we have two pluggings that give us the intended readings of the sentence.

#	plugging	logical form
1	$[\ell_1/h_0], [\ell_2/h_1], [\ell_3/h_2]$	$\forall X.\mathrm{man}(X) \Rightarrow (\exists Y.\mathrm{woman}(Y) \wedge \mathrm{love}(X,Y))$
2	$[\ell_2/h_0], [\ell_3/h_1], [\ell_1/h_2]$	$\exists Y.\mathrm{woman}(Y) \Rightarrow (\forall X.\mathrm{man}(X) \wedge \mathrm{love}(X,Y))$

7 Fragment 5: Adjectives

A Grammar fragment for Adjectives and Adverbs

▷ We want to deal with the following data.

1. *The quick brown fox jumps over the lazy dog.*
2. *The very heavy boat sank.*

SOME RIGHTS RESERVED ©: Michael Kohlhase 221 JACOBS UNIVERSITY

New Grammar in Fragment 5 (Adjectives)

▷ To account for the syntax we come up with the concept of a verb-phrase (VP)

▷ **Definition 7.1** $\mathcal{F}_4$ adds the rules on the right to $\mathcal{F}_3$ (on the left)

S1.	S	$\to$	NP, VP_{+fin}				
S2.	S	$\to$	S, Sconj				
V1.	$VP_{\pm fin}$	$\to$	${V^i}_{\pm fin}$	N3.	NP	$\to$	(Det) CNP
V2.	$VP_{\pm fin}$	$\to$	${V^t}_{\pm fin}$, NP	N4.	CNP	$\to$	N
V3.	$VP_{\pm fin}$	$\to$	$VP_{\pm fin}$VPconj$_{\pm fin}$	N5.	CNP	$\to$	PP
V4.	$VP_{\pm fin}$	$\to$	V^tNP$_{\pm fin}$	P1.	PP	$\to$	P, NP
V5.	VP_{+fin}	$\to$	$BE_=$, NP	S3.	Sconj	$\to$	conj, S
V6.	VP_{+fin}	$\to$	BE_{pred}, Adj.	V4.	VPconj$_{\pm fin}$	$\to$	conj, $VP_{\pm fin}$
V7.	VP	$\to$	didn't VP_{-fin}	L1.	P	$\to$	{with,of,...}
N1.	NP	$\to$	N_{pr}				
N2.	NP	$\to$	Pron				

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

Notes:

- Parentheses indicate optionality of a constituent.
- We assume appropriate lexical insertion rules without specification.

Translation of Special lexical items and classes

▷ If *Adj* is an intersective adjective and Adj' is an constant of type $\iota \to o$, then 9: Adj $\rightsquigarrow Adj'$ or 9′: Adj $\rightsquigarrow \lambda P_{\iota\to o} X_\iota . P(X) \wedge Adj'(X)$

▷ If *Adj* is a nonintersective adjective, then 10: Adj $\rightsquigarrow Adj'$, where Adj' is a constant of type $(\iota \to o) \to \iota \to o$ whose denotation is given by $\mathcal{I}$.

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

8 Some issues in the semantics of tense

8.1 Truth conditions of tensed sentences

Our goal is to capture the truth conditions and the logical form of sentences of English. Clearly, the following three sentences have different truth conditions.

1. *Jane saw George.*
2. *Jane sees George.*
3. *Jane will see George.*

Tense is a *deictic* element, i.e. its interpretation requires reference to something outside the sentence itself. Often, in particular in the case of monoclausal sentences occuring in isolation, as in our examples, this “something” is the speech time.

As a first pass, it is plausible to say something like the following:

- *Jane saw George* is true at a time iff *Jane sees George* was true at some point in time before now.
- *Jane will see George* is true at a time iff *Jane sees George* will be true at some point in time after now.

But here, we've tried to formulate the truth conditions of past and future in terms of the present tense sentence *Jane sees George*. But we have to give the truth conditions of the present tense sentence too. Let's try this:

- *Jane sees George* is true at a time iff Jane sees George at that time.

Now what we have on the right hand side of the statement does not mention any sentence, so we are doing better.

In fact, most treatments of the semantics of tense invoke some notion of a tenseless proposition/formula for the base case. The idea here is that markers of past, present and future all operate on an underlying untensed expression, which can be evaluated for truth at a time.

In the account that we will look at, tenses are treated as sentential operators (expressions of type $o \rightarrow o$). Our example sentences will thus be given the following translations:

1. $PAST(\text{see}(g, j))$
2. $PRES(\text{see}(g, j))$
3. $FUT(\text{see}(g, j))$

Some notes:

- Notice that I have made no attempt to show how these translations would be derived from the natural language syntax. Giving a compositional semantics for tense is a complicated business – for one thing, it requires us to first establish the syntax of tense – so we set this goal aside in this brief presentation.
- Here, I have implicitly assumed that the English modal *will* is simply a tense marker. This is indeed assumed by some. But others consider that it is no accident that *will* has the syntax of other modals like *can* and *must*, and believe that *will* is also semantically a modal.

8.2 Models and Evaluation for a tensed language

To evaluate a tensed sentence we need:

- To represent the fact that the extensions of constants vary over time.
- To be able to talk about the time of evaluation

To do this, we have to:

- Introduce times into our models, and let the interpretation function give values of constants at a time.
- Relativize the valuation function to times

8.2.1 Models with times

Our models will now be 4-tuples $\langle \mathcal{D}, \mathcal{I}, \mathcal{T}, < \rangle$ where $\mathcal{D}$ is a domain and $\mathcal{I}$ an interpretation function (as before), $\mathcal{T}$ is a set of moments of time and $<$ is an ordering on the members of $\mathcal{T}$.

The ordering relation: The ordering relation $<$ is needed to make sure that our models represent temporal relations in an intuitively correct way. Whatever the truth may be about time, as language users we have rather robust intuitions that time goes in one direction along a straight line, so that every moment of time is either before, after or identical to any other moment; and no moment of time is both before and after another moment. If we think of the set of times as the set of natural numbers, then the ordering relation $<$ is just the relation *less than* on that set.

Intervals: Although $\mathcal{T}$ is a set of moments of time, we will adopt here (following Cann, who follows various others) an *interval semantics*, in which expressions are evaluated relative to intervals of time. Intervals are defined in terms of moments, as a continuous set of moments ordered by $<$. Unlike moments, intervals may overlap (represented $i \circ j$); and one interval may be contained in another ($i \subseteq j$). As with moments, one interval may precede another ($i < j$).

The new interpretation function: In models without times, the interpretation function $\mathcal{I}$ assigned an extension to every constant. Now, we want it to assign an extension to each constant relative to each interval definable in terms of the $\mathcal{T}$. We thus redefine $\mathcal{I}$ as follows:

$$\mathcal{I} \colon \Sigma_\alpha \to \{\langle i, \Delta \rangle \,|\, i \in \mathcal{I} \wedge \Delta \in \mathcal{D}_\alpha\}$$

i.e. the interpretation function associates each constant with a pair consisting of an interval and an appropriate extension, interpreted as the extension at that interval. This set of pairs is, of course, equivalent to a function from intervals to extensions.

We add a requirement that the intervals for which any predicate is defined include all moments in $\mathcal{T}$, to ensure that there are no "gaps" in the definition of the constant.

Relativizing the valuation function to times:

- In the beginning, we evaluated sentences relative to a model: $[\![\cdot]\!]^M$.
- Then, we evaluated sentences relative to a model and an assignment function: $[\![\cdot]\!]^M_g$.
- Now, we will evaluate sentences relative to a model, an assignment function and an interval: $[\![\cdot]\!]^{M,i}_g$.

Finally, we need to revise the base clauses of the model theory.

Definition 8.1 (Revised base clause for semantic rules) Given a model $\mathcal{M} = \langle \mathcal{D}, \mathcal{I}, \mathcal{T}, < \rangle$, if α is a constant, then $[\![\alpha]\!]^{M,i}_g = \mathcal{I}(\alpha)(i)$.

Rules for interpreting variables and function/argument constructions remain unchanged, except for use of new valuation function.

8.2.2 Interpretation rules for the temporal operators

1. $[\![PRES(\Phi)]\!]^{M,i}_g = \mathsf{T}$, iff $[\![\Phi]\!]^{M,i}_g = \mathsf{T}$.
2. $[\![PAST(\Phi)]\!]^{M,i}_g = \mathsf{T}$ iff there is an interval j s.t. $j < i$ and $[\![\Phi]\!]^{M,j}_g = \mathsf{T}$.
3. $[\![FUT(\Phi)]\!]^{M,i}_g = \mathsf{T}$ iff there is an interval j s.t. $i < j$ and $[\![\Phi]\!]^{M,j}_g = \mathsf{T}$.

8.3 Complex tenses in English

How do we use this machinery to deal with complex tenses in English?

- Past of past (pluperfect): Jane had left (by the time I arrived).
- Future perfect: Jane will have left (by the time I arrive).
- Past progressive: Jane was going to leave (when I arrived).

8.4 Perfective vs. imperfective

- Jane left.
- Jane was leaving.

How do the truth conditions of these sentences differ?

Standard observation: Perfective indicates a completed action, imperfective indicates an incomplete or ongoing action. This becomes clearer when we look at the "creation predicates" like *build a house* or *write a book*

- *Jane built a house.* entails: *There was a house that Jane built.*
- *Jane was building a house.* does not entail that *there was a house that Jane built.*

8.5 Future readings of present tense

1. *Jane leaves tomorrow.*
2. *Jane is leaving tomorrow.*

But compare:

1. ?? *It rains tomorrow.*
2. ?? *It is raining tomorrow.*
3. ?? *The dog barks tomorrow.*
4. ??*The dog is barking tomorrow.*

Future readings of present tense appear to arise only when the event described is planned, or plannable, either by the subject of the sentence, the speaker, or a third party.

8.6 Sequence of tense

George said that Jane was laughing.

- Reading 1: George said "*Jane is laughing.*" I.e. saying and laughing co-occur. So past tense in subordinate clause is past of utterance time, but not of main clause reference time.
- Reading 2: George said "*Jane was laughing.*" I.e. laughing preceds saying. So past tense in subordinate clause is past of utterance time and of main clause reference time.

George saw the woman who was laughing.

- How many readings?

George will say that Jane is laughing.

- Reading 1: George will say "*Jane is laughing.*" Saying and laughing co-occur, but both saying and laughing are future of utterance time. So present tense in subordinate clause indicates futurity relative to utterance time, but not to main clause reference time.
- Reading 2: Laughing overlaps utterance time and saying (by George). So present tense in subordinate clause is present relative to utterance time *and* main clause reference time.

George will see the woman who is laughing.

- How many readings?

Note that in all of the above cases, the predicate in the subordinate clause describes an event that is extensive in time. Consider readings when subordinate event is punctual.

George said that Mary fell.

- Falling must precede George's saying.

George saw the woman who fell.

- Same three readings as before: falling must be past of utterance time, but could be past, present or future relative to seeing (i.e main clause reference time).

And just for fun, consider past under present...
George will claim that Mary hit Bill.

- Reading 1: hitting is past of utterance time (therefore past of main clause reference time).
- Reading 2: hitting is future of utterance time, but past of main clause reference time.

And finally...

1. *A week ago, John decided that in ten days at breakfast he would tell his mother that they were having their last meal together. Abusch 1988*
2. *John said a week ago that in ten days he would buy a fish that was still alive. Ogihara 1996*

8.7 Interpreting tense in discourse

Example 8.2 (Ordering and Overlap) *A man walked into the bar. He sat down and ordered a beer. He was wearing a nice jacket and expensive shoes, but he asked me if I could spare a buck.*

Example 8.3 (Tense as anaphora?)

1. Said while driving down the NJ turnpike *I forgot to turn off the stove.*
2. *I didn't turn off the stove.*

9 Propositional Attitudes and Modalities

9.1 Semantics of Modals

9.1.1 A semantics for necessity and possibility

In Kripke semantics, the intuitions about the truth conditions of modals sentences are expressed as follows:

- A sentence of the form $\Box\mathbf{A}$, where $\mathbf{A}$ is a well-formed formula of type o, is true at w iff $\mathbf{A}$ is true at *every possible world accessible from* w.
- A sentence of the form $\Diamond\mathbf{A}$, where $\mathbf{A}$ is a well-formed formula of type o, is true at w iff $\mathbf{A}$ is true at *some possible world accessible from* w.

You might notice that these truth conditions are parallel in certain ways to the truth conditions for tensed sentence. In fact, the semantics of tense is itself a modal semantics which was developed on analogy to Kripke's modal semantics. Here are the relevant similarities:

Relativization of evaluation A tensed sentence must be evaluated for truth relative to a given time. A tensed sentence may be true at one time but false at another. Similarly, we must evaluate modal sentences relative to a possible world, for a modal sentence may be true at one world (i.e. relative to one possible state of affairs) but false at another.

Truth depends on value of embedded formula at another world The truth of a tensed sentence at a time t depends on the truth of the formula embedded under the tense operator at some relevant time (possibly) different from t. Similarly, the truth of a modal sentence at w depends on the truth of the formula embedded under the modal operator at some world or worlds possibly different from w.

Accessibility You will notice that the world at which the embedded formula is to be evaluated is required to be *accessible* from the world of evaluation. The accessibility relation on possible worlds is a generalization of the ordering relation on times that we introduced in our temporal semantics. (We will return to this momentarily).

You can see, then, that a model theory for a modal language requires additions which are parallel to the additions we made for our model theory for a temporal language.

- We need to add to our models a set of possible worlds, and an ordering (accessibility relation) on the members of that set.
- We need to allow the value of constants to vary from one world to another.
- We need our valuation function to be relativized to possible worlds.

Relativization of the valuation function is straightforward. Here is the definition of a model for a modal predicate logic. (Note that there are a number of variants on the system presented here. This one has all of the vital ingredients, however.)

A model for a modal type theoretic language is a tuple $\langle \mathcal{D}, \mathcal{I}, \mathcal{W}, \mathcal{R} \rangle$ where:

- $\mathcal{D}$ is a domain. For simplicity, we will assume that the same domain is shared by all worlds w in the model. Alternatively, one might want a system which relativizes domains to worlds.
- $\mathcal{I}$ is an interpretation function. This function must allow the value of constants to vary from world to world. $\mathcal{I}$ maps a constant $c \in \Sigma_\alpha$ to a function $[\![c]\!]$ which itself maps possible worlds to elements in $\mathcal{D}_\alpha$ i.e.

$$\mathcal{I}\colon \Sigma_\alpha \to \mathcal{W} \to \mathcal{D}_\alpha$$

- $\mathcal{W}$ is a non-empty set (of possible worlds).
- $\mathcal{R}$ is an accessibility relation on $\mathcal{W}$

9.1.2 Accessibility relations

It will be helpful to start by thinking again about the ordering relation on times introduced in temporal models. This ordering relation is in fact one sort of accessibility relation.

Why did we need the ordering relation? We needed it in order to ensure that our temporal semantics makes intuitively correct predictions about the truth conditions of tensed sentences and about entailment relations between them. Here are two illustrative examples:

Example 9.1 Suppose we have $t_i < t_j$ and $t_j < t_k$. Then intuitively, if *Jane is laughing* is true at (the unit interval) t_i, then *Jane laughed* should be true at t_j and at t_k, i.e. $[\![PAST[\mathrm{laugh}(j)]]\!]_g^{t_j}$ and $[\![PAST[\mathrm{laugh}(j)]]\!]_g^{t_k}$. But this holds only if "$<$" is transitive, i.e. iff for all i, j, k, if $i < j$ and $j < k$ then also $i < k$. (And of course, as defined, "$<$" is transitive.)

Example 9.2 Here is a clearly counter-intuitive claim: For any time t_i and any sentence $\mathbf{A}$, if $[\![PRES[\mathbf{A}]]\!]_g^{t_i}$ then $[\![PAST[\mathbf{A}]]\!]_g^{t_i}$. (For example, the truth of *Jane is at the finish line* at t_i implies the truth of *Jane was at the finish line* at t_i.) But we would get this result if we allowed "$<$" to be reflexive. (Of course, as defined, "$<$" is irreflexive.)

Thus, by ordering the times in our model in accord with our intuitions about time, we can ensure correct predictions about truth conditions and entailment relations for tensed sentences.

In the modal domain, we do not have intuitions about how possible worlds should be ordered. But we do have intuitions about truth conditions and entailment relations among modal sentences. So we need to set up an accessibility relation on the set of possible worlds in our model which, in combination with the truth conditions for $\Box$ and $\Diamond$ given above, will produce intuitively correct claims about entailment.

One of the prime occupations of modal logicians is to look at the sets of validities which are obtained by imposing various different constraints on the accessibility relation. We will here consider just two examples.

What must be, is:

1. It seems intuitively correct that if it is necessarily the case that $\mathbf{A}$, then $\mathbf{A}$ is true, i.e. that $[\![\Box\mathbf{A}]\!]_g^w = \mathsf{T}$ implies that $[\![\mathbf{A}]\!]_g^w = \mathsf{T}$ or, more simply, that the following formula is valid:

$$\Box\mathbf{A} \Rightarrow \mathbf{A}$$

2. To guarantee this implication, we must ensure that any world w is among the worlds accessible from w, i.e. we must make $\mathcal{R}$ reflexive.

3. Note that this also guarantees, among other things, that the following is valid: $\mathbf{A} \implies \Diamond\mathbf{A}$

Whatever is, is necessarily possible:

1. This also seems like a reasonable slogan. Hence, we want to guarantee the validity of:

$$\mathbf{A} \implies \Box\Diamond\mathbf{A}$$

2. To do this, we must guarantee that if $\mathbf{A}$ is true at a some world w, then for every world w' accessible from w, there is at least one $\mathbf{A}$ world accessible from w'. To do this, we can guarantee that every world w is accessible from every world which is accessible from it, i.e. make $\mathcal{R}$ symmetric.

9.2 Different kinds of modality

So far, we've treated $\Box$ and $\Diamond$ as translations of a particular sense of the English expressions *it is necessary that* and *it is possible that*. This is what is called the *alethic* sense of the expressions, the sense having to do with strict logical necessity and possibility. However, now that we have a logical language containing these operators, we are free to interpret them in different ways, and so can use them to represent other senses of the modal expressions of natural language.

9.2.1 Epistemic and Doxastic Modality

In the following discourse, the modal sentence is naturally interpreted as saying something about the speaker's state of knowledge (epistemic state) or belief (doxastic state).

A *Where's John?*

B *He might be in the library.*

We might paraphrase sentence (9.2.1) as follows:
It is consistent with the speaker's knowledge that John is in the library.

Now, suppose we interpret the accessibility relation $\mathcal{R}$ in the following way: if w' is accessible from w then w' is an epistemic possibility for the agent B at w, i.e. w' represents a way the world could be consistent with the knowledge that B has in w. With this interpretation of $\mathcal{R}$, it is natural to represent B's utterance as $\Diamond(\text{inlib}j)$

Similarly, the truth conditions of: *John must be in the library.* are given by: $\Box \text{in\_lib}(j)$

An important question now arises for the modal logician. If $\mathcal{R}$ is interpreted as epistemic accessibility, what properties should it have? That is, what statements involving $\Box$ and $\Diamond$ should be valid on the epistemic interpretation of the operators, and how do we fix the accessibility relation to guarantee this?

9.2.2 Deontic modality

Another type of modality that has been investigated extensively is deontic modality, the semantics of obligation and permission.

Question: If we want to interpret $\Box\text{run}(j)$ as *It is required that John runs* (or, more idiomatically, as *John must run*), then what interpretation should we give to the accessibility relation?

What formulae should be valid on this interpretation of the operators? (This is for homework!)

9.3 A multiplicity of modalities?

The epistemic and deontic modalities differ from alethic, or logical, modality in that they must be relativized to an individual. Although we can choose to abstract away from this, it is clear that what is possible relative to John's set of beliefs may not be possible relative to Jane's, or that what is obligatory for Jane may not be obligatory for John. A theory of modality for natural language must have a means of representing this relativization.

Similarly, we find in natural language expressions of necessity and possibility relative to many different kinds of things. Consider:

- *[Given the university's rules] Jane can take that class.*
- *[Given her intellectual ability] Jane can take that class.*
- *[Given her schedule] Jane can take that class.*
- *[Given my desires] I must meet Henry.*
- *[Given the requirements of our plan] I must meet Henry.*
- *[Given the way things are] I must meet Henry (every day and not know it).*

These examples show not only that there are many different sorts of modality, but also that a given sentence may be multiply ambiguous with respect to the sort of modality it expresses.

In a series of papers beginning with her 1978 dissertation (in German), Angelika Kratzer proposed an account of the semantics of natural language models which accommodates this ambiguity. (The ambiguity is treated not as a semantic ambiguity, but as context dependency.) Kratzer's account, which is now the standard view in semantics and (well-informed) philosophy of language, adopts central ingredients from Kripke semantics – the basic possible world framework and the notion of an accessibility relation – but puts these together in a novel way. Kratzer's account of modals incorporates an account of natural language conditionals; this account has been influenced by, and been influential for, the accounts of conditionals developed by David Lewis and Robert Stalnaker. These also are now standardly accepted (at least by those who accept the possible worlds framework).

Some references: [**?**, Lew73, Sta68].

9.4 Basic Modal Logic

9.4.1 Propositional Attitues and Modalities

Modalities and Propositional Attitudes

- Modalities (also called "propositional attitudes")
  - *it is necessary that* $\mathbf{A}$,
  - *it is possible that* $\mathbf{A}$,
  - $\mathbf{A}$ *probably holds*,
  - *it has always been the case that* $\mathbf{A}$,
  - *it is well-known that* $\mathbf{A}$,
  - $\mathbf{A}$ *is allowed/prohibited*,
  - $\mathbf{A}$ *is provable*,
  - $\mathbf{A}$ *holds after the program* P *terminates*,
  - $\mathbf{A}$ *hods during the execution of* P.

  Observation: All of the red parts above modify the sentence $\mathbf{A}$.

- Idea: New sentence-to-sentence operators for *necessary* and *possible*.(extend existing logics with them.)

SOME RIGHTS RESERVED ©: Michael Kohlhase 224 JACOBS UNIVERSITY

Various logicians and philosophers looked at ways to use possible worlds, or similar theoretical entities, to give a semantics for modal sentences (specifically, for a modal logic), including Descartes and Leibniz. In the modern era, Carnap, Montague and Hintikka pursued formal developments of this idea. But the semantics for modal logic which became the basis of all following work on the topic was developed by Kripke 1963. This kind of semantics is often referred to as *Kripke semantics*.

History of Modal Logic

- Aristoteles studies the logic of necessity and possibility
- Diodorus: temporal modalities
  - possible: *is true or will be*
  - necessary: *is true and will never be false*
- C. Lewis [JSL18] (Systems $S1, \ldots, S5$)
  - strict implication $I(\mathbf{A} \wedge \mathbf{B})$ (I for "impossible")
- [Gödel 1932]: Modal logic of provability (S4)
- [Kripke 59-63] Possible Worlds Semantics
- [Pratt 76] Dynamic Logic
- $\vdots$

SOME RIGHTS RESERVED ©: Michael Kohlhase 225 JACOBS UNIVERSITY

Propositional Modal Logic (ML_0)

- **Definition 9.3** Propositional modal logic ML_0 extends propositional logic with two new logical constants: $\Box$ for necessity and $\Diamond$ for possibility. ($\Diamond\mathbf{A} = \neg(\Box\neg\mathbf{A})$)
- Observation: Nothing hinges on the fact that we use propositional logic

  Definition 9.4 First-Order modal logic ML_1 extends first-order logic with two new logical constants: $\Box$ for necessity and $\Diamond$ for possibility.
- **Example 9.5** We interpret
  1. *Necessarily, every mortal will die.* as $\Box(\forall X.\mathsf{mortal}(X) \Rightarrow \mathsf{will\_die}(X))$
  2. *Possibly, something is immortal.* as $\Diamond(\exists X.\neg\mathsf{mortal}(X))$

  Questions: What do $\Box$ and $\Diamond$ mean? How do they behave?

SOME RIGHTS RESERVED ©: Michael Kohlhase 226 JACOBS UNIVERSITY

9.4.2 Semantics and Deduction for Modal Logics

Basic ideas The fundamental intuition underlying the semantics for modality is that modal statements are statements about *how things might be*, statements about possible states of affairs. According to this intuition, sentence (1) in Example 9.5 says that in every possible state of affairs – every way that things might be – every mortal will die, while sentence (2) says that there is some possible state of affairs – some way that things might be – in which something is mortal[5]. What is needed in order to express this intuition in a model theory is some kind of entity which will stand for possible states of affairs, or ways things might be. The entity which serves this purpose is the infamous *possible world*.

Semantics of ML_0

- **Definition 9.6** A set $\mathcal{W}$ of possible worlds, accessibility relation $\mathcal{R} \subseteq \mathcal{W} \times \mathcal{W}$
- **Example 9.7** $\mathcal{W} = \mathbb{N}$ with $\mathcal{R} = \{\langle n, n+1\rangle \,|\, n \in \mathbb{N}\}$ (time logic)
- **Definition 9.8** variable assignment $\varphi \colon \mathcal{V}_o \times \mathcal{W} \to \mathcal{D}_o$ assigns values to propositional variables in a given world.
- **Definition 9.9** value function $\mathcal{I}_\varphi^w \colon \mathit{wff}_o(\mathcal{V}_o) \to \mathcal{D}_o$ (assigns values to formulae in world)
  - $\mathcal{I}_\varphi^w(V) = \varphi(w, V)$
  - $\mathcal{I}_\varphi^w(\neg\mathbf{A}) = \mathsf{T}$, iff $\mathcal{I}_\varphi^w(\mathbf{A}) = \mathsf{F}$
  - $\mathcal{I}_\varphi^w(\Box\mathbf{A}) = \mathsf{T}$, iff $\mathcal{I}_\varphi^{w'}(\mathbf{A}) = \mathsf{T}$ for all $w' \in \mathcal{W}$ with $w\mathcal{R}w'$.
- **Definition 9.10** We call a triple $\mathcal{M} := \langle \mathcal{W}, \mathcal{R}, \mathcal{I}\rangle$ a Kripke model.

[5]Note the impossibility of avoiding modal language in the paraphrase!

SOME RIGHTS RESERVED ©: Michael Kohlhase 227 JACOBS UNIVERSITY

Modal Axioms (Propositional Logic)

▷ **Definition 9.11** Necessitation: $\dfrac{\mathbf{A}}{\Box\mathbf{A}}N$

▷ **Definition 9.12 (Normal Modal Logics)**

System	Axioms	Accessibility Relation
$\mathbb{K}$	$\Box(\mathbf{A} \Rightarrow \mathbf{B}) \Rightarrow \Box\mathbf{A} \Rightarrow \Box\mathbf{B}$	general
$\mathbb{T}$	$\mathbb{K} + \Box\mathbf{A} \Rightarrow \mathbf{A}$	reflexive
$\mathbb{S}4$	$\mathbb{T} + \Box\mathbf{A} \Rightarrow \Box\Box\mathbf{A}$	reflexive + transitive
$\mathbb{B}$	$\mathbb{T} + \Diamond\Box\mathbf{A} \Rightarrow \mathbf{A}$	reflexive + symmetric
$\mathbb{S}5$	$\mathbb{S}4 + \Diamond\mathbf{A} \Rightarrow \Box\Diamond\mathbf{A}$	equivalence relation

SOME RIGHTS RESERVED ©: Michael Kohlhase 228 JACOBS UNIVERSITY

$\mathbb{K}$ Theorems

▷ $\Box(\mathbf{A} \wedge \mathbf{B}) \models (\Box\mathbf{A} \wedge \Box\mathbf{B})$

▷ $\mathbf{A} \Rightarrow \mathbf{B} \models (\Box\mathbf{A} \Rightarrow \Box\mathbf{B})$

▷ $\mathbf{A} \Rightarrow \mathbf{B} \models (\Diamond\mathbf{A} \Rightarrow \Diamond\mathbf{B})$

▷ Substitutivity of equivalence : $\dfrac{\mathbf{A} \Leftrightarrow \mathbf{B} \quad \mathbf{C}\left[\mathbf{A}\right]_p}{\mathbf{C}\left[\mathbf{B}\right]_p} \Leftrightarrow$

SOME RIGHTS RESERVED ©: Michael Kohlhase 229 JACOBS UNIVERSITY

Translation to First-Order Logic

▷ Question: Is modal logic more expressive than predicate logic?

▷ Answer: Very rarely (usually can be translated)

▷ **Definition 9.13** Translation τ from ML into FOL, (so that the diagram commutes)

$$\begin{tikzcd}
\text{Kripke-Sem.} \arrow[r, "\overline{\tau}"] & \text{Tarski-Sem.} \\
\text{modal logic} \arrow[u, "\mathcal{I}_\varphi^w"] \arrow[r, "\tau"] & \text{predicate logic} \arrow[u, "\mathcal{I}_\varphi"]
\end{tikzcd}$$

▷ Idea: Axiomatize Kripke-Semantics in PL^1 (diagram is simple consequence)

▷ **Definition 9.14** A logic morphism $\Theta\colon \mathcal{L} \to \mathcal{L}'$ is called

- correct, iff $\exists \mathcal{M}.\mathcal{M} \models \Phi$ implies $\exists \mathcal{M}'.\mathcal{M}' \models' \Theta(\Phi)$
- complete, iff $\exists \mathcal{M}'.\mathcal{M}' \models' \Theta(\Phi)$ implies $\exists \mathcal{M}.\mathcal{M} \models \Phi$

Modal Logic Translation (formal)

- **Definition 9.15 (Standard Translation)**
  - Extend all functions and predicates by a "world argument"
  - $\tau_w(f(a, b)) = \overline{f}(w, \overline{a}(w), \overline{b}(w))$
  - New relation constant $\mathcal{R}$ for the accessibility relation
  - New constant s for the "start world"
  - $\tau_w(\square\mathbf{A}) = \forall w'.w\mathcal{R}w' \Rightarrow \tau_{w'}(\mathbf{A})$
  - Use all axioms from the respective correspondence theory
- **Definition 9.16 (Alternative)** functional translation, if $\mathcal{R}$ associative:
  - new function constant $f_{\mathcal{R}}$ for the accessibility relation
  - $\tau_w(\square\mathbf{A}) = \forall w'.w = f_{\mathcal{R}}(w') \Rightarrow \tau_w(\mathbf{A})$ (or even)
  - $\tau_{f_{\mathcal{R}}(w)}(\square\mathbf{A}) = \tau_w(\mathbf{A})$ (better for mechanizing [Ohlbach '90])

Translation (continued)

- **Theorem 9.17** $\tau_s \colon ML_0 \to PL^0$ *is correct and complete*
- Proof: show that $\exists \mathcal{M}.\mathcal{M} \models \Phi$ iff $\exists \mathcal{M}'.\mathcal{M}' \models \tau_s(\Phi)$

P.1 Let $\mathcal{M} = \langle \mathcal{W}, \mathcal{R}, \varphi \rangle$ with $\mathcal{M} \models \mathbf{A}$

P.2 chose $\mathcal{M}' = \langle \mathcal{W}, \mathcal{I} \rangle$, such that $\mathcal{I}(\overline{p}) = \varphi(p) \colon \mathcal{W} \to \{\mathsf{T}, \mathsf{F}\}$ and $\mathcal{I}(r) = \mathcal{R}$.

P.3 we prove $\mathcal{M}' \models_\psi \tau_w(\mathbf{A})$ for $\psi = \mathsf{Id}_{\mathcal{W}}$ by structural induction over $\mathbf{A}$.

P.3.1 $\mathbf{A} = P$: $\mathcal{I}_\psi(\tau_w(\mathbf{A}))$

P.3.2 $\mathbf{A} = \neg\mathbf{B}$, $\mathbf{A} = \mathbf{B} \wedge \mathbf{C}$: trivial by IH. □

P.3.3 $\mathbf{A} = \square\mathbf{B}$:

P.3.3.1 $\mathcal{I}_\psi(\tau_w(\mathbf{A})) = \mathcal{I}_\psi(\forall w.r(w \Rightarrow v) \Rightarrow \tau_v(\mathbf{B})) = \mathsf{T}$, if $\mathcal{I}_\psi(r(w, v)) = \mathsf{F}$ or $\mathcal{I}_\psi(\tau_v(\mathbf{B})) = \mathsf{T}$ for all $v \in \mathcal{W}$

P.3.3.2 $\mathcal{M}' \models_\psi \tau_v(\mathbf{B})$ so by IH $\mathcal{M} \models^v \mathbf{B}$.

P.3.3.3 so $\mathcal{M}' \models_\psi \tau_w(\mathbf{A})$. □

□

Modal Logic (References)

- G. E. Hughes und M. M. Cresswell: *A companion to Modal Logic*, University Paperbacks, Methuen (1984).
- David Harel: *Dynamic Logic*, Handbook of Philosophical Logic, D. Gabbay, Hrsg. Reidel (1984).
- Johan van Benthem: *Language in Action, Categories, Lambdas and Dynamic Logic*, North Holland (1991).
- Reinhard Muskens, Johan van Benthem, Albert Visser, Dynamics, Handbook of Logic and Language, Elsevier, (1995).
- Blackburn, DeRijke, Vedema: *Modal Logic* `http://turing.wins.uva.nl/~mdr/Publications/modal-logic.html`
  look at the chapter "Guide to the literature" in the end.

SOME RIGHTS RESERVED ©: Michael Kohlhase 233 JACOBS UNIVERSITY

9.5 Model Existence and Completeness for Modal Logic

Abstract Consistency for ML_0

- **Definition 9.18** If Φ is a set of propositions, then

$$\Box^{-}(\Phi) := \{\mathbf{A} \,|\, \Box\mathbf{A} \in \Phi\}$$

- **Definition 9.19** A family ∇ of sets of ML_0-formulae is called abstract consistency class for ML_0, it if is closed under subsets and for all $\Phi \in \nabla$ we have

  ∇_c) $P \notin \Phi$ or $\neg P \notin \Phi$ for $P \in \mathcal{V}_o$

  $\vdots$

  $\nabla_\wedge$) $\neg(\mathbf{A} \vee \mathbf{B}) \in \Phi$ implies $(\Phi \cup \{\neg\mathbf{A}, \neg\mathbf{B}\}) \in \nabla$

  $\nabla_\Box$) $\Diamond\mathbf{A} \in \Phi$ implies $\Box^{-}(\Phi) * \mathbf{A} \in \nabla$

SOME RIGHTS RESERVED ©: Michael Kohlhase 234 JACOBS UNIVERSITY

∇-Hintikka Set

- **Definition 9.20** If ∇ abstract consistency class for ML_0, then we call $\mathcal{H}$ a ∇-Hintikka set, if $\mathcal{H}$ maximal in ∇, i.e. for all $\mathbf{A}$ with $\mathcal{H} * \mathbf{A} \in \nabla$ we already have $\mathbf{A} \in \mathcal{H}$.
- **Theorem 9.21 (Extension Theorem)** *If ∇ is an abstract consistency class for ML and $\Phi \in \nabla$, then there is a ∇-Hintikka set $\mathcal{H}$ with $\Phi \subseteq \mathcal{H}$.*

  Proof:

  P.1 chose an enumeration $\mathbf{A}^1, \mathbf{A}^2, \ldots$ of $\mathit{wff}_o(\mathcal{V}_o)$

P.2 construct sequence of sets H^i with $H^0 := \Phi$ and

- $H^{n+1} := H^n$, if $H^n * \mathbf{A}^n \notin \nabla$
- $H^{n+1} := H^n * \mathbf{A}^n$, if $H^n * \mathbf{A}^n \in \nabla$

P.3 All $H^i \in \nabla$, so choose $\mathcal{H} := \bigcup_{i \in \mathbb{N}} H^i$

P.4 $\Psi \subseteq \mathcal{H}$ finite implies that there is a $j \in \mathbb{N}$ with $\Psi \subseteq H^j$, so $\Psi \in \nabla$ as ∇ closed under subsets.

P.5 $\mathcal{H} \in \nabla$ since ∇ compact.

P.6 let $\mathcal{H} * \mathbf{B} \in \nabla$, then there is a $j \in \mathbb{N}$ with $\mathbf{B} = \mathbf{A}^j$

P.7 $\mathbf{B} \in H^{j+1} \subseteq \mathcal{H}$, so $\mathcal{H}$ ∇-maximal. □

Canonical ∇-Model

- **Definition 9.22** If ∇ is an abstract consistency class, for ML_0, then we call $\mathcal{M}_\nabla := \langle \mathcal{W}_\nabla, \mathcal{R}_\nabla, \varphi_\nabla \rangle$ the canonical ∇-model, iff
  - $\mathcal{W}_\nabla = \{\mathcal{H} \,|\, \mathcal{H} \in \nabla\text{maximal}\}$
  - $(v\mathcal{R}_\nabla w)$ iff $\Box^-(v) \subseteq w$
  - $\varphi(P, w) = \mathsf{T}$ iff $P \in w$
- **Lemma 9.23** *If $w \in \mathcal{W}_\nabla$ and $\Diamond\mathbf{A} \in w$, then there is a $w' \in \mathcal{W}_\nabla$ with $(w\mathcal{R}_\nabla w')$ and $\mathbf{A} \in w'$.*

  Proof: Let $\Diamond\mathbf{A} \in w$
- **P.1** thus $\Box^-(w) * \mathbf{A} \in \nabla$

  by the extension theorem there is a $w' \in \mathcal{W}_\nabla$ with $\Box^-(w) * \mathbf{A} \subseteq w'$

  so $\Box^-(w) \subseteq w'$ and thus $(w\mathcal{R}_\nabla w')$.

  on the other and we have $\mathbf{A} \in w'$. □

P.2 P.3 P.4

Model existence for ML_0

- **Lemma 9.24** *If $w \in \mathcal{W}_\nabla$, then $\mathcal{I}^w_{\varphi_\nabla}(\mathbf{A}) = \mathsf{T}$ iff $\mathbf{A} \in w$.*
- Proof: Induction on the structure of $\mathbf{A}$

P.1.1 If $\mathbf{A}$ is a variable: then we get the assertion by the definition of φ_∇. □

P.1.2 If $\mathbf{A} = \neg\mathbf{B}$ and $\mathbf{A} \in w$: then $\mathbf{B} \notin w$, thus $\mathcal{I}^w_{\varphi_\nabla}(\mathbf{B}) = \mathsf{F}$, and thus $\mathcal{I}^w_{\varphi_\nabla}(\mathbf{A}) = \mathsf{T}$. □

P.1.3 $\mathbf{A} = \mathbf{B} \wedge \mathbf{C}$: analog □

P.1.4 $\mathbf{A} = \Box\mathbf{B}$:

P.1.4.1 Let $\mathbf{A} \in w$ and $w\mathcal{R}_\nabla w'$

P.1.4.2 thus $\Box^{-}(w) \subseteq w'$ and thus $\mathbf{B} \in w'$

P.1.4.3 so (IH) $\mathcal{I}_{\varphi_\nabla}^{w'}(\mathbf{B}) = \mathsf{T}$ for any such w'.

P.1.4.4 and finally $\mathcal{I}_{\varphi_\nabla}^{w}(\mathbf{A}) = \mathsf{T}$ □

P.1.5 $\mathbf{A} = \Diamond\mathbf{B}$:

P.1.5.1 Let $\mathbf{A} \notin w$

P.1.5.2 so $\neg\mathbf{A} = \Diamond\neg\mathbf{B} \notin w$

P.1.5.3 and thus $\neg\mathbf{B} \in w'$ for some $w\mathcal{R}_\nabla w'$ by (Lemma1)

P.1.5.4 so $\mathbf{B} \in w'$ and thus $\mathcal{I}_{\varphi_\nabla}^{w'}(\mathbf{B}) = \mathsf{T}$ by IH and finally $\mathcal{I}_{\varphi_\nabla}^{w}(\mathbf{A}) = \mathsf{T}$. □

□

▷ **Theorem 9.25 (Model existence)** *If ∇ is an abstract consistency class for ML_0 and $\Phi \in \nabla$, then there is a world $w \in \mathcal{W}_\nabla$ with $\mathcal{M}_\nabla \models^w \Phi$.*

Proof:

▷ **P.1** there is a ∇-Hintikka set $\mathcal{H} = w$ with $w \in \mathcal{W}_\nabla$ and $\Phi \subseteq \mathcal{H}$.

by Lemma 2 we have $\mathcal{I}_{\varphi}^{w}(\mathbf{A}) = \mathsf{T}$ for all $\mathbf{A} \in \Phi$. □

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

P.2 Completeness

▷ **Theorem 9.26** *$\mathbb{K}$-consistency is an abstract consistency class for ML_0*

▷ Proof: Let $\Diamond\mathbf{A} \in \Phi$

P.1 To show: $\Box^{-}(\Phi) * \mathbf{A}$ is $\mathbb{K}$-consistent if Φ is $\mathbb{K}$-consistent

P.2 converse: Φ is $\mathbb{K}$-inconsistent if $\Box^{-}(\Phi) * \mathbf{A}$ $\mathbb{K}$-inconsistent.

P.3 There is a finite subset $\Psi \subseteq \Box^{-}(\Phi)$ with $(\Psi \vdash_{\mathbb{K}} \neg\mathbf{A})$

P.4 $(\Box\Psi \vdash_{\mathbb{K}} \Box\neg\mathbf{A})$ (distributivity of $\Box$)

P.5 $(\Phi \vdash_{\mathbb{K}} \Box\neg\mathbf{A}) = \neg(\Diamond\mathbf{A})$ since $\Box\Psi \subseteq \Phi$

P.6 thus Φ is **K**-inconsistent. □

▷ **Corollary 9.27** *$\mathbb{K}$ is complete wrt. Kripke models*

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

Further Completeness Theorems

▷ **Theorem 9.28** *$\mathbb{T}$-consistency is an abstract consistency class for ML_0 and $\mathcal{R}_{\mathbb{T}}$ is reflexive.*

▷ Proof: Let $\mathbf{A} \in \Box^{-}(w)$

P.1 then $\Box\mathbf{A} \in w$ by definition

P.2 with $\mathbb{T}$ $(\Box\mathbf{A} \Rightarrow \mathbf{A})$ and Modus Ponens we have $\mathbf{A} \in w$.

P.3 Thus $\Box^{-}(w) \subseteq w$ and $w\mathcal{R}_{\mathbb{T}}w$ for all $w \in \mathcal{W}_{\mathbb{T}}$. $\Box$

▷ **Theorem 9.29** *$\mathbb{S}4$-consistency is an abstract consistency class for ML_0 and $\mathcal{R}_{\mathbb{S}4}$ is transitive.*

Proof: Let $w_1\mathcal{R}_{\mathbb{S}4}w_2\mathcal{R}_{\mathbb{S}4}w_3$ and $\Box\mathbf{A} \in w$.

▷ **P.1** by $\mathbb{S}4$ $(\Box\mathbf{A} \Rightarrow \Box\Box\mathbf{A})$ and Modus Ponens we have $\Box\Box\mathbf{A} \in w_1$.

and thus $\Box\mathbf{A} \in w_2 = \Box^{-}(w_1)$ and $\mathbf{A} \in w_3 = \Box^{-}(w_2)$.

Thus $\Box^{-}(w_1) \subseteq w_3$ and $w_1\mathcal{R}_{\mathbb{S}4}w_3$. $\Box$

P.2 **P.3** ▷ **Corollary 9.30** *$\mathbb{T}$ ($\mathbb{S}4$) is complete wrt. reflexive (reflexive transitive) Kripke-models*

Multimodal Logics

▷ Multiple Modalities $[1], [2], [3], \ldots, \langle 1\rangle, \langle 2\rangle, \ldots$

▷ Models with multiple accessibility relations $\mathcal{R}_1, \mathcal{R}_2, \ldots \subseteq \mathcal{W} \times \mathcal{W}$

▷ **Definition 9.31** value function: $\mathcal{I}_\varphi^w([i]\mathbf{A}) = \mathsf{T}$, iff $\mathcal{I}_\varphi^{w'}(\mathbf{A}) = \mathsf{T}$ for all $w' \in \mathcal{W}$ with $w\mathcal{R}_iw'$.

▷ **Example 9.32 (Epistemic Logic: talking about knowing/believing)** $[peter]\langle klaus\rangle\mathbf{A}$ (Peter knows that Klaus considers $\mathbf{A}$ possible)

▷ **Example 9.33 (Program Logic: talking about programs)** $[X \leftarrow \mathbf{A}][(Y \leftarrow \mathbf{A})]X = Y)$ (after assignments the values of X and Y are equal)

10 Dynamic Approaches to NL Semantics

In this section we tackle another level of language, the discourse level, where we look especially at the role of cross-sentential anaphora. This is an aspect of natural language that cannot (compositionally) be modeled in first-order logic, due to the strict scoping behavior of quantifiers. This has led to the developments of dynamic variants of first-order logic: the "file change semantics" [Hei82] by Irene Heim and (independently) "discourse representation theory" (DRT [Kam81]) by Hans Kamp, which solve the problem by re-interpreting indefinites to introduce representational objects – called "discourse referents in DRT" – that are not quantificationally bound variables and can therefore have a different scoping behavior. These approaches have been very influential in the representation of discourse – i.e. multi-sentence – phenomena.

In this section, we will introduce discourse logics taking DRT as a starting point since it was adopted more widely than file change semantics and the later "dynamic predicate logics" (DPL [GS91]). Subsection 10.0 gives an introduction to dynamic language phenomena and how they can be modeled in DRT. Subsection 10.2 relates the linguistically motivated logics to modal logics used for modeling imperative programs and draws conclusions about the role of language in cognition. Subsection 10.3 extends our primary inference system – model generation – to DRT and relates the concept of discourse referents to Skolem constants. Dynamic model generation also establishes a natural system of "direct deduction" for dynamic semantics. Finally Subsection 10.1 discusses how dynamic approaches to NL semantics can be combined with ideas Montague Semantics to arrive at a fully compositional approach to discourse semantics.

10.1 Discourse Representation Theory

In this subsection we introduce Discourse Representation Theory as the most influential framework for aproaching dynamic phenomena in natural language. We will only cover the basic ideas here and leave the coverage of larger fragments of natural language to [KR93].

Let us look at some data about effects in natural languages that we cannot really explain with our treatment of indefinite descriptions in fragment 4 (see Section 4).

Anaphora and Indefinites revisited (Data)

- ▷ *Peter*[1] *is sleeping. He*$_1$ *is snoring.* (normal anaphoric reference)
- ▷ *A man*[1] *is sleeping. He*$_1$ *is snoring.* (Scope of existential?)
- ▷ *Peter has a car*[1]*. It*$_1$ *is parked outside.* (even it this worked)
- ▷ **Peter has no car*[1]*. It*$_1$ *is parked outside.* (what about negation?)
- ▷ *There is a book*[1] *that Peter does not own. It*$_1$ *is a novel.* (OK)
- ▷ **Peter does not own every book*[1]*. It*$_1$ *is a novel.* (equivalent in PL^1)
- ▷ *If a farmer*[1] *owns a donkey*$_2$*, he*$_1$ *beats it*$_2$*.* (even inside sentences)

SOME RIGHTS RESERVED ©: Michael Kohlhase 241 JACOBS UNIVERSITY

In the first example, we can pick up the subject *Peter* of the first sentence with the anaphoric reference *He* in the second. We gloss the intended anaphoric reference with the labels in upper and lower indices. And indeed, we can resolve the anaphoric reference in the semantic representation by translating *He* to (the translation of) *Peter*. Alternatively we can follow the lead of fragment 2 (see Subsubsection 3.1.0) and introduce variables for anaphora and adding a conjunct that equates the respective variable with the translation of *Peter*. This is the general idea of anaphora resolution we will adopt in this subsection.

Dynamic Effects in Natural Language

- ▷ Problem: E.g. Quantifier Scope
  - ▷ **A man sleeps. He snores.*
  - ▷ $(\exists X.\mathsf{man}(X) \wedge \mathsf{sleep}(X)) \wedge \mathsf{snore}(X)$
  - ▷ X is bound in the first conjunct, and free in the second.
- ▷ Problem: Donkey sentence: *If a farmer owns a donkey, he beats it.*
  $\forall X, Y.\mathsf{farmer}(X) \wedge \mathsf{donkey}(Y) \wedge \mathsf{own}(X, Y) \Rightarrow \mathsf{beat}(X, Y)$
- ▷ Ideas:
  - ▷ composition of sentences by conjunction inside the scope of existential quantifiers (non-compositional, ...)

- ▷ Extend the scope of quantifiers dynamically (DPL)
- ▷ Replace existential quantifiers by something else (DRT)

SOME RIGHTS RESERVED ©: Michael Kohlhase 242 JACOBS UNIVERSITY

Intuitively, the second example should work exactly the same – it should not matter, whether the subject NP is given as a proper name or an indefinite description. The problem with the indefinite descriptions is that that they are translated into existential quantifiers and we cannot refer to the bound variables – see below. Note that this is not a failure of our envisioned treatment of anaphora, but of our treatment of indefinite descriptions; they just do not generate the objects that can be referred back to by anaphoric references (we will call them "referents"). We will speak of the "anaphoric potential" for this the set of referents that can be anaphorically referred to.

The second pair of examples is peculiar in the sense that if we had a solution for the indefinite description in *Peter has a car*, we would need a solution that accounts for the fact that even though *Peter has a car* puts a car referent into the anaphoric potential *Peter has no car* – which we analyze compositionally as *It is not the case that Peter has a car* does not. The interesting effect is that the negation closes the anaphoric potential and excludes the car referent that *Peter has a car* introduced.

The third pair of sentences shows that we need more than PL^1 to represent the meaning of quantification in natural language while the sentence *There is a book that peter does not own.* induces a book referent in the dynamic potential, but the sentence *Peter does not own every book* does not, even though their translations $\exists x.\wedge \mathrm{book}(x), \neg \mathrm{own}(\mathrm{peter}, x)$ and $\neg(\forall x.\mathrm{book}(x) \Rightarrow \mathrm{own}(\mathrm{peter}, x))$ are logically equivalent.

The last sentence is the famous "donkey sentence" that shows that the dynamic phenomena we have seen above are not limited to inter-sentential anaphora.

The central idea of Discourse Representation Theory (DRT), is to eschew the first-order quantification and the bound variables it induces altogether and introduce a new representational device: a discourse referents, and manage its visibility (called accessibility in DRT) explicitly.

We will introduce the traditional, visual "box notation" by example now before we turn to a systematic definition based on a symbolic notation later.

Discourse Representation Theory (DRT)

- ▷ Discourse referents
  - ▷ are kept in a dynamic context (Accessibility)
  - ▷ are declared e.g. in indefinite nominals
  - ▷ *A student owns a book.*

$$\begin{array}{|l|}\hline X, Y \\ \hline \mathsf{stud}(X) \\ \mathsf{book}(Y) \\ \mathsf{own}(X,Y) \\ \hline \end{array}$$

- ▷ Discourse representation structures (DRS)

A student owns a book. He reads it. *If a farmer owns a donkey, he beats it.*

$$\begin{array}{|l|}\hline X, Y, R, S \\ \hline \mathsf{stud}(X) \\ \mathsf{book}(Y) \\ \mathsf{own}(X,Y) \\ \mathsf{read}(R,S) \\ X = R \\ Y = S \\ \hline \end{array}
\qquad
\begin{array}{|c|}\hline \\ \hline \begin{array}{|l|}\hline X, Y \\ \hline \mathsf{farmer}(X) \\ \mathsf{donkey}(Y) \\ \mathsf{own}(X,Y) \\ \hline \end{array} \Rrightarrow \begin{array}{|l|}\hline \\ \hline \mathsf{beat}(Y,X) \\ \hline \end{array} \\ \hline \end{array}$$

©: Michael Kohlhase 243

These examples already show that there are three kinds of objects in DRT: The meaning of sentences is given as DRSes, which are denoted as "file cards" that list the discourse referents (the participants in the situation described in the DRS) at the top of the "card" and state a couple of conditions on the discourse referents. The conditions can contain DRSes themselves, e.g. in conditional conditions.

With this representational infrastructure in place we can now look at how we can construct discourse DRSes – i.e. DRSes for whole discourses. The sentence composition problem was – after all – the problem that led to the development of DRT since we could not compositionally solve it in first-order logic.

Discourse DRS Construction

- ▷ Problem: How do we construct DRSes for multi-sentence discourses?
- ▷ title=Solution We construct sentence DRSes individually and merge them (DRSes and conditions separately)
- ▷ **Example 10.1** A three-sentence discourse. (not quite Shakespeare)

Mary sees John

see(mary, john)

John kills a cat

U
cat(U) kill(john, U)

Mary calls a cop

V
cop(V) calls(mary, V)

merge

U, V
see(mary, john) cat(U) kill(john, U) cop(V) calls(mary, V)

Sentence composition via the DRT Merge Operator $\otimes$. (acts on DRSes)

©: Michael Kohlhase 244

Note that – in contrast to the "smuggling-in"-type solutions we would have to dream up for first-order logic – sentence composition in DRT is compositional: We construct sentence DRSes[6] and merge them. We can even introduce a "logic operator" for this: the merge operator $\otimes$, which can be thought of as the "full stop" punctuation operator.

Now we can have a look at anaphor resolution in DRT. This is usually considered as a separate process – part of semantic-pragmatic analysis. As we have seen, anaphora are

Anaphor Resolution in DRT

- ▷ Problem: How do we resolve anaphora in DRT?
- ▷ Solution: Two phases
  - ▷ translate pronouns into discourse referents (semantics construction)
  - ▷ identify (equate) coreferring discourse referents, (maybe) simplify (semantic/pragmatic analysis)

[6] We will not go into the sentence semantics construction process here

▷ **Example 10.2** *A student owns a book. He reads it.*

A student[1] owns a book[2]. He$_1$ reads it$_2$

	resolution	simplify
$\begin{array}{\|l\|}\hline X,Y,R,S \\ \hline \mathsf{stud}(X) \\ \mathsf{book}(Y) \\ \mathsf{read}(R,S) \\ \hline \end{array}$	$\begin{array}{\|l\|}\hline X,Y,R,S \\ \hline \mathsf{stud}(X) \\ \mathsf{book}(Y) \\ \mathsf{read}(R,S) \\ X=R \\ Y=S \\ \hline \end{array}$	$\begin{array}{\|l\|}\hline X,Y \\ \hline \mathsf{stud}(X) \\ \mathsf{book}(Y) \\ \mathsf{read}(X,Y) \\ \hline \end{array}$

SOME RIGHTS RESERVED ©: Michael Kohlhase 245 JACOBS UNIVERSITY

We will sometime abbreviate the anaphor resolution process and directly use the simplified version of the DRSes for brevity.

Using these examples, we can now give a more systematic introduction of DRT using a more symbolic notation. Note that the grammar below over-generates, we still need to specify the visibility of discourse referents.

DRT (Syntax)

▷ **Definition 10.3** Given a set $\mathcal{DR}$ of discourse referent s, discourse representation structure s (DRS es) are given by the following grammar:

condition s	$\mathcal{C} \to p(a_1,\ldots,a_n) \mid (\mathcal{C}_1 \wedge \mathcal{C}_2) \mid \neg\mathcal{D} \mid (\mathcal{D}_1 \vee\!\!\vee \mathcal{D}_2) \mid (\mathcal{D}_1 \Rrightarrow \mathcal{D}_2)$
DRSes	$\mathcal{D} \to (\delta U^1,\ldots,U^n\mathcal{C}) \mid (\mathcal{D}_1 \otimes \mathcal{D}_2) \mid (\mathcal{D}_1 ;; \mathcal{D}_2)$

▷ $\otimes$ and ;; are for sentence composition ($\otimes$ from DRT, ;; from DPL)

▷ **Example 10.4** $\delta U,V.\mathsf{farmer}(U) \wedge \mathsf{donkey}(V) \wedge \mathsf{own}(U,V) \wedge \mathsf{beat}(U,V)$

▷ **Definition 10.5** The meaning of $\otimes$ and ;; is given operationally by τ-Equality:

$$\begin{array}{rcl} \delta\mathcal{X}\mathcal{C}_1 \otimes \delta\mathcal{Y}\mathcal{C}_2 & \to_\tau & \delta\mathcal{X},\mathcal{Y}\mathcal{C}_1 \wedge \mathcal{C}_2 \\ \delta\mathcal{X}\mathcal{C}_1 ;; \delta\mathcal{Y}\mathcal{C}_2 & \to_\tau & \delta\mathcal{X},\mathcal{Y}\mathcal{C}_1 \wedge \mathcal{C}_2 \end{array}$$

▷ Discourse Referents used instead of bound variables (specify scoping independently of logic)

▷ Idea: Semantics by mapping into first-order Logic.

SOME RIGHTS RESERVED ©: Michael Kohlhase 246 JACOBS UNIVERSITY

We can now define the notion of accessibility in DRT, which in turn determines the (predicted) dynamic potential of a DRS: A discourse referent has to be accessible in order to be picked up by an anaphoric reference.

We will follow the classical exposition and introduce accessibility as a derived concept induced by a non-structural notion of sub-DRS.

Sub-DRSes and Accessibility

▷ Problem: Formally define accessibility (to make predictions)

▷ Idea: make use of the structural properties of DRT

▷ **Definition 10.6** A referent is accessible in all sub-DRS of the declaring DRS.

- ▹ If $\mathcal{D} = \delta U^1, \ldots, U^n \mathcal{C}$, then any sub-DRS of $\mathcal{C}$ is a sub-DRS of $\mathcal{D}$.
- ▹ If $\mathcal{D} = \mathcal{D}_1 \otimes \mathcal{D}_2$, then $\mathcal{D}_1$ is a sub-DRS of $\mathcal{D}_2$ and vice versa.
- ▹ If $\mathcal{D} = \mathcal{D}_1 ;; \mathcal{D}_2$, then $\mathcal{D}_2$ is a sub-DRS of $\mathcal{D}_1$.
- ▹ If $\mathcal{C}$ is of the form $\mathcal{C}_1 \wedge \mathcal{C}_2$, or $\neg\mathcal{D}$, or $\mathcal{D}_1 \mathbin{\mathbb{V}} \mathcal{D}_2$, or $\mathcal{D}_1 \Rrightarrow \mathcal{D}_2$, then any sub-DRS of the $\mathcal{C}_i$, and the $\mathcal{D}_i$ is a sub-DRS of $\mathcal{C}$.
- ▹ If $\mathcal{D} = \mathcal{D}_1 \Rrightarrow \mathcal{D}_2$, then $\mathcal{D}_2$ is a sub-DRS of $\mathcal{D}_1$

▷ **Definition 10.7 (Dynamic Potential)** (which referents can be picked up?)

A referent U is in the dynamic potential of a DRS $\mathcal{D}$, iff it is accessible in $\mathcal{D} \otimes \begin{array}{|c|}\hline \\ \hline p(U) \\ \hline\end{array}$

▷ **Definition 10.8** We call a DRS static, iff the dynamic potential is empty, and dynamic, if it is not.

Observation: Accessibility gives DRSes the flavor of binding structures. (with non-standard scoping!)

⊳ Idea: Apply the usual heuristics binding heuristics to DRT, e.g.

- ▹ reject DRSes with unbound discourse referents.

▷ Questions: if view of discourse referents as "nonstandard bound variables"

- ▹ what about renaming referents?

SOME RIGHTS RESERVED ©: Michael Kohlhase 247 JACOBS UNIVERSITY

The meaning of DRSes is (initially) given by a translation to PL[1]. This is a convenient way to specify meaning, but as we will see, it has its costs, as we will see.

Translation from DRT to FOL

▷ **Definition 10.9** For τ-normal (fully merged) DRSes use

$$\begin{array}{rcl}
\overline{\delta U^1, \ldots, U^n \mathcal{C}} & = & \exists U^1, \ldots, U^n \overline{\mathcal{C}} \\
\overline{\neg\mathcal{D}} & = & \neg\overline{\mathcal{D}} \\
\overline{\mathcal{D} \mathbin{\mathbb{V}} \mathcal{E}} & = & \overline{\mathcal{D}} \vee \overline{\mathcal{E}} \\
\overline{\mathcal{D} \wedge \mathcal{E}} & = & \overline{\mathcal{D}} \wedge \overline{\mathcal{E}} \\
\overline{(\delta U^1, \ldots, U^n \mathcal{C}_1) \Rrightarrow (\delta V^1, \ldots, V^m \mathcal{C}_2)} & = & \forall U^1, \ldots, U^n \overline{\mathcal{C}_1} \Rightarrow (\exists V^1, \ldots, V^l \overline{\mathcal{C}_2})
\end{array}$$

▷ **Example 10.10** $\exists X.\mathsf{man}(X) \wedge \mathsf{sleep}(X) \wedge \mathsf{snore}(X)$

▷ Consequence: Validity of DRSes can be checked by translation.

▷ Question: Why not use first-order logic directly?

▷ Answer: Only translate at the end of a discourse (translation closes all dynamic contexts: frequent re-translation).

SOME RIGHTS RESERVED

We can now test DRT as a logical system on the data and see whether it makes the right predictions about the dynamic effects identified at the beginning of the subsection.

Properties of Dynamic Scope

▷ Idea: Test DRT on the data above for the dynamic phenomena

▷ **Example 10.11 (Negation Closes Dynamic Potential)**
Peter has no[1] car. **It*$_1$ *is parked outside.*

$$\begin{array}{|c|}\hline \\ \hline \neg\;\begin{array}{|l|}\hline U\\ \hline \mathsf{car}(U)\\ \mathsf{own}(\mathsf{peter},U)\\ \hline\end{array} \\ \hline\end{array} \otimes \begin{array}{|l|}\hline \\ \hline \mathsf{park}(U)\\ \hline\end{array} \qquad \neg(\exists U.\mathsf{car}(U)\wedge\mathsf{own}(\mathsf{peter},U))\ldots$$

▷ **Example 10.12 (Universal Quantification is Static)**
Peter does not own every book[1]. **It*$_1$ *is a novel.*

$$\begin{array}{|c|}\hline \\ \hline \neg\;\begin{array}{|c|}\hline \\ \hline \begin{array}{|l|}\hline U\\ \hline \mathsf{book}(U)\\ \hline\end{array} \Rrightarrow \begin{array}{|l|}\hline \\ \hline \mathsf{own}(\mathsf{peter},U)\\ \hline\end{array}\\ \hline\end{array} \\ \hline\end{array} \otimes \begin{array}{|l|}\hline \\ \hline \mathsf{novel}(U)\\ \hline\end{array} \qquad \neg(\forall U.\mathsf{book}(U)\Rightarrow\mathsf{own}(\mathsf{peter},U))\ldots$$

▷ **Example 10.13 (Existential Quantification is Dynamic)**
There is a book[1] that Peter does not own. *It*$_1$ *is a novel.*

$$\begin{array}{|l|}\hline V\\ \hline \mathsf{book}(V)\\ \neg\mathsf{own}(\mathsf{peter},V)\\ \hline\end{array} \otimes \begin{array}{|l|}\hline \\ \hline \mathsf{novel}(V)\\ \hline\end{array} \qquad \exists U.\mathsf{book}(U)\wedge\neg\mathsf{own}(\mathsf{peter},U)\wedge\mathsf{novel}(U)$$

SOME RIGHTS RESERVED

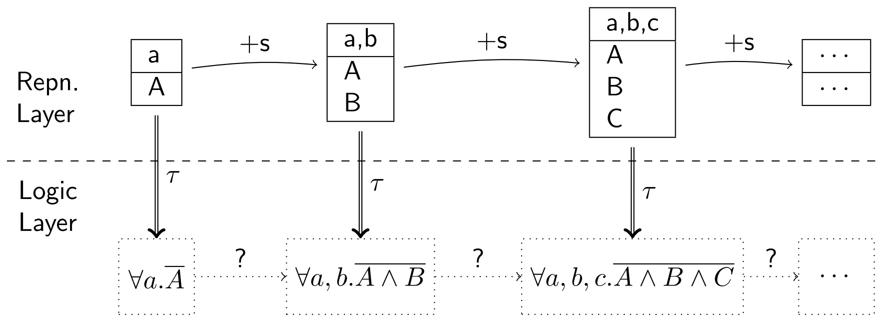
Example 10.11 shows that negation closes off the dynamic potential. Indeed, the referent U is not accessible in the second argument of $\otimes$. Example 10.12 makes predicts the inaccessibility of U for the same reason. In contrast to that, U is accessible in Example 10.13, since it is not under the scope of a dynamic negation. Incidentally, the

The examples above, and in particular the difference between Example 10.12 and Example 10.13 show that DRT forms a representational level above – recall that we can translate down – PL^1, which serves as the semantic target language. Indeed DRT makes finer distinctions than PL^1, and supports an incremental process of semantics construction: DRS construction for sentences followed by DRS merging via τ-reduction.

DRT as a Representational Level

▷ DRT adds a level to the knowledge representation which provides anchors (the discourse referents) for anaphors and the like

▷ propositional semantics by translation into PL^1



▷ Anaphor resolution works incrementally on the representational level.

SOME RIGHTS RESERVED ©: Michael Kohlhase 250 JACOBS UNIVERSITY

We will now introduce a "direct semantics" for DRT: a notion of "model" and an evaluation mapping that interprets DRSes directly – i.e. not via a translation of first-order logic. The main idea is that atomic conditions and conjunctions are interpreted largely like first-order formulae, while DRSes are interpreted as sets of assignments to discourse referents that make the conditions true. A DRS is satisfied by a model, if that set is non-empty.

A Direct Semantics for DRT (Dyn. Interpretation $\mathcal{I}^\delta_\varphi$)

▷ **Definition 10.14** Let $\mathcal{M} = \langle \mathcal{U}, \mathcal{I} \rangle$ be a FO Model and $\varphi, \psi \colon \mathcal{DR} \to \mathcal{U}$ be referent assignment s, then we say that ψ extends φ on $\mathcal{X} \subseteq \mathcal{DR}$ (write $\varphi\,[\mathcal{X}]\,\psi$), if $\varphi(U) = \psi(U)$ for all $U \notin \mathcal{X}$.

▷ Idea: Conditions as truth values; DRSes as pairs $\langle \mathcal{X}, \mathcal{S} \rangle$ ($\mathcal{S}$ set of states)

▷ **Definition 10.15 (Meaning of complex formulae)** ▷ $\mathcal{I}^\delta_\varphi(p(a_1, \ldots, a_n))$ as always.

- ▷ $\mathcal{I}^\delta_\varphi(\mathbf{A} \wedge \mathbf{B}) = \mathsf{T}$, iff $\mathcal{I}^\delta_\varphi(\mathbf{A}) = \mathsf{T}$ and $\mathcal{I}^\delta_\varphi(\mathbf{B}) = \mathsf{T}$.
- ▷ $\mathcal{I}^\delta_\varphi(\neg\mathcal{D}) = \mathsf{T}$, if $\mathcal{I}^\delta_\varphi(\mathcal{D})^2 = \emptyset$.
- ▷ $\mathcal{I}^\delta_\varphi(\mathcal{D} \Cup \mathcal{E}) = \mathsf{T}$, if $\mathcal{I}^\delta_\varphi(\mathcal{D})^2 \neq \emptyset$ or $\mathcal{I}^\delta_\varphi(\mathcal{E})^2 \neq \emptyset$.
- ▷ $\mathcal{I}^\delta_\varphi(\mathcal{D} \twoheadrightarrow \mathcal{E}) = \mathsf{T}$, if for all $\psi \in \mathcal{I}^\delta_\varphi(\mathcal{D})^2$ there is a $\tau \in \mathcal{I}^\delta_\varphi(\mathcal{E})^2$ with $\psi\,\left[\mathcal{I}^\delta_\varphi(\mathcal{E})^1\right]\,\tau$.
- ▷ $\mathcal{I}^\delta_\varphi(\delta\mathcal{X}.\mathbf{C}) = \langle \mathcal{X}, \{\psi \mid \varphi\,[\mathcal{X}]\,\psi \text{ and } \mathcal{I}^\delta_\psi(\mathbf{C}) = \mathsf{T}\} \rangle$.
- ▷ $\mathcal{I}^\delta_\varphi(\mathcal{D} \otimes \mathcal{E}) = \mathcal{I}^\delta_\varphi(\mathcal{D} ;; \mathcal{E}) = \langle \mathcal{I}^\delta_\varphi(\mathcal{D})^1 \cup \mathcal{I}^\delta_\varphi(\mathcal{E})^1, \mathcal{I}^\delta_\varphi(\mathcal{D})^2 \cap \mathcal{I}^\delta_\varphi(\mathcal{E})^2 \rangle$

SOME RIGHTS RESERVED ©: Michael Kohlhase 251 JACOBS UNIVERSITY

We can now fortify our intuition by computing the direct semantics of two sentences, which differ in their dynamic potential.

Examples (Computing Direct Semantics)

▷ **Example 10.16** *Peter owns a car*

$$\begin{aligned} & \mathcal{I}^\delta_\varphi(\delta U.\mathsf{car}(U) \wedge \mathsf{own}(\mathsf{peter}, U)) \\ = \; & \langle\{U\}, \{\psi \mid \varphi\,[\mathcal{X}]\,\psi \text{ and } \mathcal{I}^\delta_\psi(\mathsf{car}(U) \wedge \mathsf{own}(\mathsf{peter}, U)) = \mathsf{T}\}\rangle \\ = \; & \langle\{U\}, \{\psi \mid \varphi\,[\mathcal{X}]\,\psi \text{ and } \mathcal{I}^\delta_\psi(\mathsf{car}(U)) = \mathsf{T} \text{ and } \mathcal{I}^\delta_\psi(\mathsf{own}(\mathsf{peter}, U)) = \mathsf{T}\}\rangle \\ = \; & \langle\{U\}, \{\psi \mid \varphi\,[\mathcal{X}]\,\psi \text{ and } \psi(U) \in \mathsf{car} \text{ and } \langle\psi(U), \mathsf{peter}\rangle \in \mathsf{own}\}\rangle \end{aligned}$$

The set of states $[a/U]$, such that a is a car and is owned by Peter

▷ **Example 10.17** For *Peter owns no car* we look at the condition:

$$\begin{aligned} & \mathcal{I}^\delta_\varphi(\neg(\delta U.\mathsf{car}(U) \wedge \mathsf{own}(\mathsf{peter}, U))) = \mathsf{T} \\ \Leftrightarrow \; & \mathcal{I}^\delta_\varphi(\delta U.\mathsf{car}(U) \wedge \mathsf{own}(\mathsf{peter}, U))^2 = \emptyset \\ \Leftrightarrow \; & \langle\{U\}, \{\psi \mid \varphi\,[\mathcal{X}]\,\psi \text{ and } \psi(U) \in \mathsf{car} \text{ and } \langle\psi(U), \mathsf{peter}\rangle \in \mathsf{own}\}\rangle^2 = \emptyset \\ \Leftrightarrow \; & \{\psi \mid \varphi\,[\mathcal{X}]\,\psi \text{ and } \psi(U) \in \mathsf{car} \text{ and } \langle\psi(U), \mathsf{peter}\rangle \in \mathsf{own}\} = \emptyset \end{aligned}$$

i.e. iff there are no a, that are cars and that are owned by Peter.

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

The first thing we see in Example 10.62 is that the dynamic potential can directly be read off the direct interpretation of a DRS: it is the domain of the states in the first component. In Example 10.63, the interpretation is of the form $\langle\emptyset, \mathcal{I}^\delta_\varphi(\mathcal{C})\rangle$, where $\mathcal{C}$ is the condition we compute the truth value of in Example 10.63.

The cost we had to pay for being able to deal with discourse phenomena is that we had to abandon the compositional treatment of natural language we worked so hard to establish in fragments 3 and 4. To have this, we would have to have a dynamic λ calculus that would allow us to raise the respective operators to the functional level. Such a logical system is non-trivial, since the interaction of structurally scoped λ-bound variables and dynamically bound discourse referents is non-trivial.

10.2 Higher-Order Dynamics

In this subsection we will develop a typed λ calculus that extend DRT-like dynamic logics like the simply typed λ calculus extends first-order logic.

10.2.1 Introduction

We start out our development of a Montague-like compositional treatment of dynamic semantics construction by naively "adding λs" to DRT and deriving requirements from that.

Making Montague Semantics Dynamic

▷ **Example 10.18** *A man sleeps.*

$\mathsf{a\_man} = \lambda Q.(\begin{array}{|c|}\hline U \\ \hline \mathsf{man}(U) \\ \hline\end{array} \otimes Q(U))$

$\mathsf{sleep} = \lambda X.\begin{array}{|c|}\hline \\ \hline \mathsf{sleep}(X) \\ \hline\end{array}$

Application and β-reduction:

$$\mathsf{a\_man\_sleep} = \mathsf{a\_man(sleep)}$$

$$\rightarrow_\beta \begin{array}{|l|}\hline U \\ \hline \mathsf{man}(U) \\ \hline \end{array} \otimes \begin{array}{|l|}\hline \\ \hline \mathsf{sleep}(U) \\ \hline \end{array} \rightarrow_\tau \begin{array}{|l|}\hline U \\ \hline \mathsf{man}(U) \\ \mathsf{sleep}(U) \\ \hline \end{array}$$

SOME RIGHTS RESERVED ©: Michael Kohlhase 253 JACOBS UNIVERSITY

At the sentence level we just disregard that we have no idea how to interpret λ-abstractions over DRSes and just proceed as in the static (first-order) case. Somewhat surprisingly, this works rather well, so we just continue at the discourse level.

Coherent Text (Capturing Discourse Referents)

▷ **Example 10.19** *A man*[1] *sleeps. He*$_1$ *snores.*

$$(\lambda PQ.(P \otimes Q))\ \mathsf{a\_man\_sleep}\ \mathsf{he\_snore}$$

$$\rightarrow_\beta \left(\lambda Q. \begin{array}{|l|}\hline U \\ \hline \mathsf{man}(U) \\ \mathsf{sleep}(U) \\ \hline \end{array} \otimes Q\right) \begin{array}{|l|}\hline \\ \hline \mathsf{snore}(U) \\ \hline \end{array}$$

$$\rightarrow_\tau \begin{array}{|l|}\hline U \\ \hline \mathsf{man}(U) \\ \mathsf{sleep}(U) \\ \hline \end{array} \otimes \begin{array}{|l|}\hline \\ \hline \mathsf{snore}(U) \\ \hline \end{array} \rightarrow_\tau \begin{array}{|l|}\hline U \\ \hline \mathsf{man}(U) \\ \mathsf{sleep}(U) \\ \mathsf{snore}(U) \\ \hline \end{array}$$

▷ **Example 10.20 (Linear notation)**
$(\lambda Q.(\delta U.\mathsf{man}(U) \wedge \mathsf{sleep}(U) \wedge Q(U)))\mathsf{he\_snore} \longrightarrow_{\beta\tau} \delta U.\mathsf{man}(U) \wedge \mathsf{sleep}(U) \wedge \mathsf{snore}(U)$

SOME RIGHTS RESERVED ©: Michael Kohlhase 254 JACOBS UNIVERSITY

Here we have our first surprise: the second β reduction seems to capture the discourse referent U: intuitively it is "free" in $\delta.\mathrm{snore}()U$ and after β reduction it is under the influence of a δ declaration. In the λ-calculus tradition variable capture is the great taboo, whereas in our example, it seems to drive/enable anaphor resolution.

Considerations like the ones above have driven the development of many logical systems attempting the compositional treatment of dynamic logics. All were more or less severely flawed.

Compositional Discourse Representation Theories

▷ Many logical systems

- ▷ Compositional DRT (Zeevat, 1989 [Zee89]
- ▷ Dynamic Montague Grammar (DMG Gronendijk/Stokhof 1990 [GS90])
- ▷ CDRT (Muskens 1993/96 [Mus96])
- ▷ λ-DRT (Kohlhase/Kuschert/Pinkal 1995 [KKP96])
- ▷ TLS (van Eijck 1996 [vE97])

Problem: Difficult to tell the differences or make predictions!

⊳ One Answer: Dynamic λ-calculus [Kohlhase&Kuschert&Müller'96,98]

- ⊳ Augment type system by information on referents: a meta-logic that models different forms of accessibility as a parameter.

SOME RIGHTS RESERVED ©: Michael Kohlhase 255 JACOBS UNIVERSITY

Here we will look at a system that makes the referent capture the central mechanism using an elaborate type system to describe referent visibility and thus accessibility. This generalization allows to understand and model the interplay of λ-bound variables and discourse referents without being distracted by linguistic modeling questions (which are relegated to giving appropriate types to the operators).

Another strong motivation for a higher-order treatment of dynamic logics is that maybe the computational semantic analysis methods based on higher-order features (mostly higher-order unification) can be analogously transferred to the dynamic setting.

Motivation for the Future

- ⊳ Higher-Order Unification Analyses of
  - ⊳ Ellipsis (Dalrymple/Shieber/Pereira 1991 [DSP91])
  - ⊳ Focus (Pulman 1994 [Pul94], Gardent/Kohlhase 1996 [GK96])
  - ⊳ Corrections (Gardent/Kohlhase/van Leusen 1996 [GKvL96])
  - ⊳ Underspecification (Pinkal 1995 [Pin96])
- ⊳ are based on static type theory [Mon74]
- ⊳ Higher-Order Dynamic Unification needed for dynamic variants of these

SOME RIGHTS RESERVED ©: Michael Kohlhase 256 JACOBS UNIVERSITY

To set the stage for the development of a higher-order system for dynamic logic, let us remind ourselves of the setup of the static system

Recap: Simple Type Theory

- ⊳ Structural layer: simply typed λ-calculus
  - ⊳ types, well-formed formulae, λ-abstraction
  - ⊳ Theory: $\alpha\beta\eta$-conversion, Operational: Higher-Order Unification
- ⊳ Logical layer: higher-order logic
  - ⊳ special types ι, o
  - ⊳ logical constants $\wedge_{o\to o\to o}, \Rightarrow, \forall, \ldots$ with fixed semantics
  - ⊳ Theory: logical theory, Operational: higher-order theorem proving

Goal: Develop two-layered approach to compositional discourse theories.

⊳ Application: Dynamic Higher-Order Unification (DHOU) with structural layer only.

This separation of concerns: structural properties of functions vs. a propositional reasoning level has been very influential in modeling static, intra-sentential properties of natural language, therefore we want to have a similar system for dynamic logics as well. We will use this as a guiding intuition below.

10.2.2 Setting Up Higher-Order Dynamics

To understand what primitives a language for higher-order dynamics should provide, we will analyze one of the attempts – λ-DRT – to higher-order dynamics

λ-DRT is a relatively straightforward (and naive) attempt to "sprinkle λs over DRT" and give that a semantics. This is mirrored in the type system, which had a primitive types for DRSes and "intensions" (mappings from states to objects). To make this work we had to introduce "intensional closure", a semantic device akin to type raising that had been in the folklore for some time. We will not go into intensions and closure here, since this did not lead to a solution and refer the reader to [KKP96] and the references there.

Recap: λ-DRT (simplified)

- Types: ι (individuals), o (conditions), t (DRSes), $\alpha \to \beta$ (functions), $s \to \alpha$ (intensions)
- Syntax: if U_ι a referent and $\mathbf{A}$ an expression of type o, then $\delta U_\iota.\mathbf{A}$ a DRS (type t).
- $\alpha\beta\eta$-reduction for the λ-calculus part, and further:
  - $(\delta\mathcal{X}.\mathbf{A} \otimes \delta\mathcal{Y}.\mathbf{B}) \to_\tau (\delta\mathcal{X} \cup \mathcal{Y}.\mathbf{A} \wedge \mathbf{B})$
  - ${}^{\vee\wedge}\mathbf{A} \to_\mu \mathbf{A}$

  Observations:
- - complex interaction of λ and δ
  - alphabetical change for δ-bound "variables" (referents)?
  - need intensional closure for $\beta\eta$-reduction to be correct

In hindsight, the contribution of λ-DRT was less the proposed semantics – this never quite worked beyond correctness of $\alpha\beta\eta$ equality – but the logical questions about types, reductions, and the role of states it raised, and which led to further investigations.

We will now look at the general framework of "a λ-calculus with discourse referents and δ-binding" from a logic-first perspective and try to answer the questions this raises. The questions of modeling dynamic phenomena of natural language take a back-seat for the moment.

Finding the right Dynamic Primitives

- Need to understand Merge Reduction: ($\to_\tau$-reduction)

- Why do we have $(\delta U.\mathbf{A} \otimes \mathbf{B}) \rightarrow_\tau (\delta U.\mathbf{A} \wedge \mathbf{B})$
- but not $(\delta U.\mathbf{A}) \Rrightarrow \mathbf{B} \rightarrow_\tau (\delta U.\mathbf{A} \Rrightarrow \mathbf{B})$

▷ and Referent Scoping: (ρ-equivalence)

- When are the meanings of $\mathbf{C}\,[\delta U.\mathbf{A}]_\pi$ and $\mathbf{C}\,[\delta V.[V/U](\mathbf{A})]_\pi$ equal?
- OK for $\mathbf{C} = \neg$ and $\mathbf{C} = \lambda P.(\delta W.\mathbf{A} \Rrightarrow P)$
- Not for $\mathbf{C} = \lambda P.P$ and $\mathbf{C} = \lambda P.P \wedge \neg P$.

Observation: There must be a difference of $\otimes, \neg, \lambda P.(\delta W.\mathbf{A} \Rrightarrow P), \lambda P.P \wedge \neg P$ wrt. the behavior on referents

⊳ Intuitively: $\otimes, \lambda P.(\delta W.\mathbf{A} \Rrightarrow P)$ transport U, while $\neg, \lambda P.P \wedge \neg P$ do not

▷ Idea: Model this in the types (rest of the talk/lecture)

SOME RIGHTS RESERVED

A particularly interesting phenomenon is that of referent capture as the motor or anaphor resolution, which have already encountered above.

Variable/Referent Capture

▷ **Example 10.21 (Anaphor Resolution Revisited)** Let us revisit ?anaphor-resolution.ex?

A student[1] owns a book[2]. He$_1$ reads it$_2$ resolution simplify

$$\begin{array}{|l|}\hline X,Y\\\hline \mathrm{stud}(X)\\ \mathrm{book}(Y)\\\hline\end{array} \otimes \begin{array}{|l|}\hline R,S\\\hline \mathrm{read}(R,S)\\\hline\end{array} \quad \begin{array}{|l|}\hline X,Y\\\hline \mathrm{stud}(X)\\ \mathrm{book}(Y)\\\hline\end{array} \otimes \begin{array}{|l|}\hline R,S\\\hline \mathrm{read}(R,S)\\ R=X\\ S=Y\\\hline\end{array} \quad \begin{array}{|l|}\hline X,Y\\\hline \mathrm{stud}(X)\\ \mathrm{book}(Y)\\ \mathrm{read}(X,Y)\\\hline\end{array}$$

▷ **Example 10.22** $(\lambda P.\begin{array}{|l|}\hline U\\\hline \neg\ P\\\hline\end{array})\begin{array}{|l|}\hline \\\hline r(U)\\\hline\end{array}$ (functor has dynamic binding power)

▷ Variable capture (or rather referent capture)

- is the motor of dynamicity
- is a structural property

Idea: Code the information for referent capture in the type system

SOME RIGHTS RESERVED

In Example 10.21 we see that with the act of anaphor resolution, the discourse referents induced by the anaphoric pronouns get placed under the influence of the dynamic binding in the first DRS – which is OK from an accessibility point of view, but from a λ-calculus perspective this constitutes a capturing event, since the binding relation changes. This becomes especially obvious, if we look at the simplified form, where the discourse referents introduced in the translation of the pronouns have been eliminated altogether.

In Example 10.22 we see that a capturing situation can occur even more explicitly, if we allow λs – and $\alpha\beta\eta$ equality – in the logic. We have to deal with this, and again, we choose to model it in the type system.

With the intuitions sharpened by the examples above, we will now start to design a type system that can take information about referents into account. In particular we are interested in the capturing behavior identified above. Therefore we introduce information about the "capturing status" of discourse referents in the respective expressions into the types.

▷ Types in $\mathcal{DLC}$

- ▷ Requirements: In the types we need information about
  - ▷ δ-bound referents (they do the capturing)
  - ▷ free referents (they are liable to be captured)
- ▷ **Definition 10.23** New type (moded type) $\Gamma\#\alpha$ where
  - ▷ mode $\Gamma = V^-, U^+, \ldots$ (V is a free and U a capturing referent)
  - ▷ term type α (type in the old sense)
- ▷ What about functional types? (Look at example)

SOME RIGHTS RESERVED ©: Michael Kohlhase 261 JACOBS UNIVERSITY

To see how our type system for $\mathcal{DLC}$ fares in real life, we see whether we can capture the referent dynamics of λ-DRT. Maybe this also tells us what we still need to improve.

Rational Reconstruction of λ-DRT (First Version)

- ▷ Two-level approach
  - ▷ model structural properties (e.g. accessibility relation) in the types
  - ▷ leave logical properties (e.g. negation flips truth values) for later
- ▷ Types: $\iota, o, \alpha \to \beta$ only. $\Gamma\#o$ is a DRS.
- ▷ Idea: Use mode constructors $\downarrow$ and $\uplus$ to describe the accessibility relation.
- ▷ **Definition 10.24** $\downarrow$ closes off the anaphoric potential and makes the referents classically bound ($\downarrow U^+, V^+ = U^\circ, V^\circ$)
- ▷ **Definition 10.25** The prioritized union operator combines two modes by letting $+$ overwrite $-$. ($U^+, V^- \uplus U^-, V^+ = U^+, V^+$)
- ▷ **Example 10.26 (DRT Operators)** Types of DRT connectives (indexed by Γ, Δ):
  - ▷ $\neg$ has type $\Gamma\#o \to \downarrow\Gamma\#o$ (intuitively like $t \to o$)
  - ▷ $\otimes$ has type $\Gamma\#o \to \Delta\#o \to \Gamma \uplus \Delta\#o$ (intuitively like $t \to t \to t$)
  - ▷ $\vee$ has type $\Gamma\#o \to \Delta\#o \to \downarrow\Gamma \uplus \downarrow\Delta\#o$
  - ▷ $\Rrightarrow$ has type $\Gamma\#o \to \Delta\#o \to \downarrow\Gamma \uplus \downarrow\Delta\#o$

SOME RIGHTS RESERVED ©: Michael Kohlhase 262 JACOBS UNIVERSITY

We can already see with the experiment of modeling the DRT operators that the envisioned type system gives us a way of specifying accessibility and how the dynamic operators handle discourse referents. So we indeed have the beginning of a structural level for higher-order dynamics, and at the same time a meta-logic flavor, since we can specify other dynamic logics in a λ-calculus.

10.2.3 A Type System for Referent Dynamics

We will now take the ideas above as the basis for a type system for $\mathcal{DLC}$.

The types above have the decided disadvantage that they mix mode information with information about the order of the operators. They also need free mode variables, which turns out to be a problem for designing the semantics. Instead, we will employ two-dimensional types, where the mode part is a function on modes and the other a normal simple type.

Types in $\mathcal{DLC}$ (Final Version)

- Problem: A type like $\Gamma\#o \to \Gamma^-\#o$ mixes mode information with simple type information.
- Alternative formulation: $\downarrow\#o \to o$ (use a mode operator for the mode part)
- **Definition 10.27** $\mathcal{DLC}$ types are pairs $\mathbf{A}\#\alpha$, where
  - $\mathbf{A}$ is a mode specifier, α is a simple type; $\mathbf{A}$ is functional, iff α is.

  Idea: Use the simply typed λ-calculus for mode specifiers
- Other connectives (new version)
  - $\neg$ gets type $\lambda F.\downarrow F\#o \to o$
  - $\otimes$ gets type $\uplus\#o \to o \to o$
  - $\Rrightarrow$ gets type $\lambda FG.\downarrow(F \uplus \downarrow G)\#o \to o \to o$

SOME RIGHTS RESERVED ©: Michael Kohlhase 263 JACOBS UNIVERSITY

With this idea, we can re-interpret the DRT types from Example 10.26

A λ-Calculus for Mode Specifiers

- **Definition 10.28** New base type μ for modes; $\widetilde{\alpha}$ is α with ι, o replaced by μ.
- mode specifiers $\mathbb{A}, \mathbb{B}, \mathbb{C}$ are simply typed λ-terms built up from mode variables $F, G, F^1, \ldots$ and
- **Definition 10.29 (Mode constants)**
  - the empty mode $\emptyset$ of type μ
  - the elementary mode s U^+, U^- and U° of type μ for all referents $U \in \mathcal{R}$
  - the mode functions $\cdot^+$, $\cdot^-$, $\downarrow\cdot$, $+\cdot$, and $-\cdot$ of type $\mu \to \mu$, and
  - the mode function $\uplus$ of type $\mu \to \mu \to \mu$.

▷ Theory of mode equality specifies the meaning of mode constants (e.g. $(U^+, V^-, W^- \uplus U^-, V^+) \to_\mu U^+, V^+, W^-$)

SOME RIGHTS RESERVED ©: Michael Kohlhase 264 JACOBS UNIVERSITY

Type Inference for $\mathcal{DLC}$ (two dimensions)

▷ **Definition 10.30**

$$\frac{c \in \Sigma_\alpha}{\mathcal{A} \vdash_\Sigma c\colon \alpha} \qquad \frac{\mathcal{A}(X) = F\#\alpha \quad \mathcal{A}(F) = \widetilde{\alpha}}{\mathcal{A} \vdash_\Sigma X\colon F\#\alpha} \qquad \frac{U \in \mathcal{R}_\alpha \quad \mathcal{A}(U) = \emptyset\#\alpha}{\mathcal{A} \vdash_\Sigma U\colon U^-\#\alpha}$$

$$\frac{\mathcal{A}, [X : F\#\beta], [F : \widetilde{\beta}] \vdash_\Sigma \mathbf{A}\colon \mathbb{A}\#\alpha}{\mathcal{A} \vdash_\Sigma \lambda X_{F\#\beta}.\mathbf{A}\colon \lambda F.\mathbb{A}\#\beta \to \alpha} \qquad \frac{\mathcal{A} \vdash_\Sigma \mathbf{A}\colon \mathbb{A}\#\beta \to \gamma \quad \mathcal{A} \vdash_\Sigma \mathbf{B}\colon \mathbb{B}\#\beta}{\mathcal{A} \vdash_\Sigma \mathbf{AB}\colon \mathbb{AB}\#\gamma}$$

$$\frac{\mathcal{A} \vdash_\Sigma \mathbf{A}\colon \mathbb{A}\#\alpha \quad \mathcal{A} \vdash_\Sigma \beta\eta\mu =_{\mathbb{A}} \mathbb{B}}{\mathcal{A} \vdash_\Sigma \mathbf{A}\colon \mathbb{B}\#\alpha} \qquad \frac{\mathcal{A} \vdash_\Sigma \mathbf{A}\colon \lambda F.\mathbb{A}\#\alpha \quad \mathcal{A} \vdash_\Sigma \mathbb{A}\colon \mu}{\mathcal{A} \vdash_\Sigma \delta U_\beta.\mathbf{A}\colon \lambda F.(U^+ \uplus \mathbb{A})\#\alpha}$$

where $\mathcal{A}$ is a variable context mapping variables and referents to types

SOME RIGHTS RESERVED ©: Michael Kohlhase 265 JACOBS UNIVERSITY

Example (Identity)

▷ We have the following type derivation for the identity.

$$\frac{\overline{[F : \widetilde{\alpha}], [X : F\#\alpha] \vdash_\Sigma X\colon F\#\alpha}}{\vdash_\Sigma \lambda X_{F\#\alpha}.X\colon \lambda F_{\widetilde{\alpha}}.F\#\alpha \to \alpha}$$

▷ $(\lambda X_{F\#\alpha \to \alpha}.X)(\lambda X_{G\#\alpha}.X)$ has type $\mathcal{A} \vdash_\Sigma (\lambda F_{\mu \to \mu}.F)(\lambda G_\mu.G)\#\alpha \to \alpha =_{\beta\eta\mu} \lambda G_\mu.G\#\alpha \to \alpha$

▷ **Theorem 10.31 (Principal Types)** *For any given variable context $\mathcal{A}$ and formula $\mathbf{A}$, there is at most one type $\mathbb{A}\#\alpha$ (up to mode $\beta\eta\mu$-equality) such that $\mathcal{A} \vdash_\Sigma \mathbf{A}\colon \mathbf{A}\#\alpha$ is derivable in $\mathcal{DLC}$.*

SOME RIGHTS RESERVED ©: Michael Kohlhase 266 JACOBS UNIVERSITY

Linguistic Example

▷ **Example 10.32** *No man sleeps.*

Assume $U \in \mathcal{R}_\iota$ and man, sleep $\in \mathcal{R}_{\lambda F.F\#\iota\to o}$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{\mathcal{A}\vdash_\Sigma \mathsf{man}(U)\colon U^-\#o}}{\mathcal{A}\vdash_\Sigma \delta U.\mathsf{man}(U)\colon U^+\#o} \quad \dfrac{\vdots}{\mathcal{A}\vdash_\Sigma \mathsf{sleep}(U)\colon U^-\#o}}{\mathcal{A}\vdash_\Sigma \delta U.\mathsf{man}(U)\wedge \mathsf{sleep}(U)\colon U^+\uplus U^-\#o}}{\mathcal{A}\vdash_\Sigma \neg(\delta U.\mathsf{man}(U)\wedge \mathsf{sleep}(U))\colon {\downarrow}(U^+\uplus U^-)\#o}}{\mathcal{A}\vdash_\Sigma \neg(\delta U.\mathsf{man}(U)\wedge \mathsf{sleep}(U))\colon U^\circ\#o}$$

A Further (Tricky) Example: $\mathbf{A}_\neg := \lambda X.X \wedge \neg X$

- a referent declaration in the argument of $\mathbf{A}_\neg$ will be copied, and the two occurrences will have a different status
  $\mathbf{A}_\neg(\delta U.\mathsf{man}(U)) \to_\beta (\delta U.\mathsf{man}(U) \wedge \neg(\delta U.\mathsf{man}(U)))$
- assuming $\mathcal{A}(X) = F\#o$ gives

$$\dfrac{\dfrac{\mathcal{A}\vdash_\Sigma X\colon F\#o \quad \dfrac{\mathcal{A}\vdash_\Sigma X\colon F\#o}{\mathcal{A}\vdash_\Sigma \neg X\colon {\downarrow}F\#o}}{\mathcal{A}\vdash_\Sigma X\wedge\neg X\colon F\uplus{\downarrow}F\#o}}{\mathcal{A}\vdash_\Sigma \lambda X.X\wedge\neg X\colon \lambda F.(F\uplus{\downarrow}F)\#o\to o}$$

- thus, assuming $\mathcal{A}\vdash_\Sigma \delta U.\mathsf{man}(U)\colon U^+\#o$, we derive $\mathcal{A}\vdash_\Sigma \mathbf{A}_\neg(\delta U.\mathsf{man}(U))\colon U^+, U^\circ\#o$

A Further Example: Generalised Coordination

- We may define a generalised *and*:
  $\lambda R^1 \ldots R^n.\lambda X^1 \ldots X^m.(R^1 X^1 \ldots X^m \otimes \ldots \otimes R^n X^1 \ldots X^m)$
  with type
  $\lambda F^1 \ldots F^n.(F^1 \uplus \ldots \uplus F^n)\#\overline{(\overline{\beta_m} \to o) \to (\overline{\beta_m} \to o)}$
- thus from john $:= \lambda P.(\delta U.U = j \otimes P(U))$
  and mary $:= \lambda P.(\delta V.V = m \otimes P(V))$
- we get johnandmary $= \lambda P.(\delta U.U = j \otimes P(U) \otimes \delta V.V = m \otimes P(V))$
- combine this with *own a donkey*:

$\lambda X.(\delta W.\mathsf{donkey}(W) \otimes \mathsf{own}(W, X) \otimes \delta U.U = j \otimes \delta W.\mathsf{donkey}(W) \otimes \mathsf{own}(W, U) \otimes \delta V.V = m \otimes \delta W.\mathsf{donkey}(W) \otimes$ o

10.2.4 Modeling Higher-Order Dynamics

Discourse Variants $=_\delta$

- The order and multiplicity of introduction of discourse referents is irrelevant
  - $(\delta U.\delta V.\mathbf{A}) =_\delta (\delta V.\delta U.\mathbf{A})$
  - $(\delta U.\delta U.\mathbf{A}) =_\delta (\delta U.\mathbf{A})$.
  - This is needed to model DRT, where discourse referents appear in sets.
- functional and dynamic binding can be interchanged
  - $\lambda X.(\delta U.\mathbf{A}) =_\delta (\delta U.\lambda X.\mathbf{A})$
  - This is useful for convenient η-long-forms (DHOU).

Renaming of Discourse Referents?

- Consider $\mathbf{A} := (\lambda XY.Y)(\delta U.U)$
  - δU cannot have any effect on the environment, since it can be deleted by β-reduction.
  - $\mathbf{A}$ has type $\lambda F.F\#\alpha \to \alpha$ (U does not occur in it).

Idea: Allow to rename U in $\mathbf{A}$, if "$\mathbf{A}$ is independent of U"

- Similar effect for $\mathbf{B} := \neg(\delta U.\mathrm{man}(U))$, this should equal $\neg(\delta V.\mathrm{man}(V))$
- **Definition 10.33** $=_\rho$-renaming is induced by the following inference rule:

$$\frac{V \in \mathcal{R}_\beta \text{ fresh} \quad U_\beta \notin \mathcal{DP}(\mathbf{A})}{\mathbf{A} =_\rho \mathcal{C}_U^V(\mathbf{A})}$$

Where $\mathcal{C}_U^V(\mathbf{A})$ is the result of replacing all referents U by V.

Dynamic Potential

- The binding effect of an expression $\mathbf{A}$ can be read off its modality $\mathbf{A}$
- A modality $\mathbf{A}$ may be simplified by $\beta\eta\mu$-reduction (where μ-equality reflects the semantics of the mode functions, e.g. $U^+ \uplus U^- =_\mu U^+$).
- **Definition 10.34** The dynamic binding potential of $\mathbf{A}$: $\mathcal{DP}(\mathbf{A}) := \{U \mid U^+ \in \mathrm{occ}(\mathbf{A}') \text{ or } U^- \in \mathrm{occ}(\mathbf{A}')\}$, where $\mathbf{A}'$ is the $\beta\eta\mu$-normal form of $\mathbf{A}$.

▷ **Definition 10.35** If $U \notin \mathcal{DP}(\mathbf{A})$, then U is called independent of $\mathbf{A}$.

SOME RIGHTS RESERVED ©: Michael Kohlhase 272 JACOBS UNIVERSITY

Some Examples for Dynamic Potential

▷ **Example 10.36**

Formula	Modality	$\mathcal{DP}$
$\delta U.P$	U^+	$\{U\}$
$\lambda P.(\delta U.P)$	$\lambda F.(U^+ \uplus F)$	$\{U\}$
$\neg(\delta U.\mathrm{man}(U))$	U°	$\emptyset$
$\lambda P.\neg(\delta U.P)$	$\lambda F.\downarrow(U^+), F$	$\{U\}$
$\lambda X.U$	$\lambda F.U^-$	$\{U\}$
$(\lambda X.X)U$	$(\lambda F.F)U^-$	$\{U\}$
$\lambda P.\mathrm{man}(U) \wedge P$	$\lambda F.(F \uplus U^-)$	$\{U\}$
$\lambda P.P$	$\lambda F.F$	$\emptyset$
$\lambda XY.Y$	$\lambda FG.G$	$\emptyset$
$(\lambda XY.Y)(\delta U.U)$	$\lambda G.G$	$\emptyset$
$\lambda P.P(\lambda Q.\neg(\delta U.Q))(\lambda R.(\delta U.R))$		$\{U\}$

SOME RIGHTS RESERVED ©: Michael Kohlhase 273 JACOBS UNIVERSITY

Reductions

▷ $\beta\eta$-reduction: $\frac{}{(\lambda X.\mathbf{A})\mathbf{B} \to_\beta [\mathbf{B}/X](\mathbf{A})}$ and $\frac{X \notin \mathbf{free}(\mathbf{A})}{\lambda X.\mathbf{A}X \to_\eta \mathbf{A}}$

▷ Dynamic Reduction: $\frac{\mathcal{A} \vdash_\Sigma \mathbf{A}\colon \mathbb{A}\#\alpha \quad U^+ \in \mathbf{Trans}(\mathbb{A})}{\mathbf{A}(\delta U.\mathbf{B}) \to_\tau (\delta U.\mathbf{AB})}$

▷ **Example 10.37** Merge-Reduction $(\delta U.\mathbf{A} \otimes \delta V.\mathbf{B}) \to_\tau (\delta U.\delta V.(\mathbf{A} \otimes \mathbf{B}))$

▷ Intuition: The merge operator is just dynamic conjunction!

▷ Observation: Sequential merge ;; of type $\overrightarrow{\uplus}\#o \to o \to o$ does not transport V in the second argument.

SOME RIGHTS RESERVED ©: Michael Kohlhase 274 JACOBS UNIVERSITY

10.2.5 Direct Semantics for Dynamic λ Calculus

Higher-Order Dynamic Semantics (Static Model)

▷ Frame $\mathcal{D} = \{\mathcal{D}_\alpha \mid \alpha \in \mathcal{T}\}$

- ▷ $\mathcal{D}_\mu$ is the set of modes (mappings from variables to signs)
- ▷ $\mathcal{D}_o$ is the set of truth values $\{\mathsf{T}, \mathsf{F}\}$.
- ▷ $\mathcal{D}_\iota$ is an arbitrary universe of individuals.
- ▷ $\mathcal{D}_{\alpha\to\beta} \subseteq \mathcal{F}(\mathcal{D}_\alpha; \mathcal{D}_\beta)$

▷ Interpretation $\mathcal{I}$ of constants, assignment φ of variables.

- ▷ $\mathcal{I}_\varphi(c) = \mathcal{I}(c)$, for a constant c
- ▷ $\mathcal{I}_\varphi(X) = \varphi(X)$, for a variable X
- ▷ $\mathcal{I}_\varphi(\mathbf{AB}) = \mathcal{I}_\varphi(\mathbf{A})(\mathcal{I}_\varphi(\mathbf{B})))$
- ▷ $\mathcal{I}_\varphi(\lambda X.\mathbf{B})(\mathsf{a}) = \mathcal{I}_{\varphi,[\mathsf{a}/X]}(\mathbf{B})$.

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

Dynamic Semantics (Frames)

▷ Two approaches: "Dynamic" (Amsterdam) and "Static" (Saarbrücken)

- ▷ Will show that they are equivalent (later)
- ▷ Use the static semantics for $\mathcal{DLC}$ now.

▷ What is the denotation of a dynamic object?

- ▷ "Static Semantics": essentially a set of states (considers only type o) (equivalently function from states to $\mathcal{D}_o$: characteristic function)
- ▷ generalize this to arbitrary base type: $\mathcal{D}_\alpha^\Gamma = \mathcal{F}(\mathcal{B}_\Gamma; \mathcal{D}_\alpha)$, where $\mathcal{B}_\Gamma$ is the set of Γ-states

▷ Γ-states: well-typed referent assignments $s\colon \mathbf{Dom}^\pm(\Gamma) \to \mathcal{D}$
$s|\Delta$ is s coerced into a Δ-state.

▷ For expressions of functional type: $\mathcal{D}_{\alpha\to\beta}^\Phi = \bigcup_{\Psi\in\mathcal{D}_{\tilde{\alpha}}} \mathcal{F}(\mathcal{D}_\alpha^\Psi; \mathcal{D}_\beta^{\Phi(\Psi)})$

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

Dynamic Semantics (Evaluation)

▷ Standard Tool: Intensionalization (guards variables by delaying evaluation of current state)

▷ Idea: Ideal for semantics of variable capture

- ▷ guard all referents
- ▷ make this part of the semantics (thus implicit in syntax)

▷ Evaluation of variables and referents

- ▷ If $X \in \mathcal{V}$, then $\mathcal{I}_\varphi(X) = \varphi(X)$
- ▷ If $U \in \mathcal{R}$, then $\mathcal{I}_\varphi(U) = \Lambda s \in \mathcal{B}_{U^-}.s(U)$ (implicit intensionalization!)
- ▷ $\mathcal{I}_\varphi(\delta U.\mathbf{B}_{\mathbb{B}\#\beta}) = \Lambda s \in \mathcal{B}_{(\mathcal{I}_\varphi(\mathbb{B}_\mu)\uplus U^+)}.\mathcal{I}_\varphi(\mathbf{B})s|\mathcal{I}_\varphi(\mathbb{B}_\mu)$.
- ▷ $\mathcal{I}_\varphi(\mathbf{BC}) = \mathcal{I}_\varphi(\mathbf{B})(\mathcal{I}_\varphi(\mathbf{C}))$.
- ▷ $\mathcal{I}_\varphi(\lambda X_\gamma.\mathbf{B}) = \Lambda^\Phi \mathsf{a} \in \mathcal{D}_\gamma^\Phi.\mathcal{I}_{(\varphi,[\mathsf{a}/X])}(\mathbf{B})$

▷ Referent names crucial in dynamic objects

▷ Well actually: $\mathcal{I}_\varphi(\delta U.\mathcal{B}_{\Lambda\overline{F_n}.\mathbb{B}_\mu\#\beta}) = \Lambda\overline{\mathfrak{a}_n}.(\Lambda s \in \mathcal{B}_{(\mathcal{I}_\varphi(\mathbb{B}_\mu)\uplus U+)}.\mathcal{I}_\varphi(\mathbf{B})s|\mathcal{I}_\varphi(\mathcal{B}_\mu))$.

Metatheoretic Results

▷ **Theorem 10.38 (Normalization)** *$\beta\eta\tau$-Reduction is terminating and confluent (modulo $\alpha\rho\delta$).*

▷ **Theorem 10.39 (Substitution is type-preserving)** *If $X \notin \mathbf{dom}(\mathcal{A})$, then $\mathcal{A}, [X : F\#\beta] \vdash_\Sigma \mathbf{A} \colon \mathbb{A}\#\alpha$ and $\mathcal{A} \vdash_\Sigma \mathbf{B} \colon \mathbb{B}\#\beta$ imply $\mathcal{A} \vdash_\Sigma [\mathbf{B}/X](\mathbf{A}) \colon [\mathbf{B}/F](\mathbb{A})\#\alpha$.*

▷ **Theorem 10.40 (Subject Reduction)** *If $\mathcal{A} \vdash_\Sigma \mathbf{A} \colon \mathbb{A}\#\alpha$ and $\mathcal{A} \vdash_\Sigma \mathbf{A} =_{\beta\eta\tau} \mathbf{B}$, then $\mathcal{A} \vdash_\Sigma \mathbf{B} \colon \mathbb{A}\#\alpha$.*

▷ **Theorem 10.41 (Soundness of Reduction)** *If $\mathcal{A} \vdash_\Sigma \mathbf{A} =_{\alpha\beta\delta\eta\tau\rho} \mathbf{B}$, then $\mathcal{I}_\varphi(\mathbf{A}) = \mathcal{I}_\varphi(\mathbf{B})$.*

▷ **Conjecture 10.42 (Completeness)** *If $\mathcal{I}_\varphi(\mathbf{A}) = \mathcal{I}_\varphi(\mathbf{B})$, then $\mathcal{A} \vdash_\Sigma \mathbf{A} =_{\alpha\beta\delta\eta\tau\rho} \mathbf{B}$ (just needs formalisation of equality of logical operators.)*

10.2.6 Dynamic λ Calculus outside Linguistics

Conclusion

▷ Basis for compositional discourse theories

- ▷ two-layered approach (only use theorem proving where necessary)
- ▷ functional and dynamic information can be captured structurally
- ▷ comprehensive equality theory (interaction of func. and dyn. part)

▷ In particular

- ▷ new dynamic primitives (explain others)
- ▷ simple semantics (compared to other systems)

▷ This leads to

- ▷ dynamification of existing linguistic analyses (DHOU)
- ▷ rigorous comparison of different dynamic systems (Meta-Logic)

Future Directions

- Generalize $\mathcal{DLC}$ to a true mode calculus:
  - turn δ into a logical constant δ_U: (use type declaration and application)

$$\frac{\mathcal{A} \vdash_\Sigma \mathbf{A} \colon \mathbb{A}\#\alpha}{\mathcal{A} \vdash_\Sigma \delta U_\beta \mathbf{.A} \colon U^+ \uplus \mathbb{A}_\mu\#\alpha} \qquad \frac{\vdash_\Sigma \delta_U \colon \lambda F.(U^+ \uplus F)\#\alpha \to \alpha \quad \mathcal{A} \vdash_\Sigma \mathbf{A} \colon \mathbb{A}\#\alpha}{\mathcal{A} \vdash_\Sigma \delta_U \mathbf{A} \colon U^+ \uplus \mathbb{A}_\mu\#\alpha}$$

  - this allows for more than one δ-like operator
- Better still (?) go for a dependent type discipline (implement in LF?)
- δ of type $\lambda UF.(U^+ \uplus F)\#\alpha \to \alpha$ yields $\delta(U) \widehat{=} \delta_U$

Use $\mathcal{DLC}$ as a model for Programming

- Remember dynamic binding in LISP?

```
((lambda (F) (let ((U 1)) (F 1)))(lambda (X) (+ X U))→ 2
((lambda (F) (let ((U 0)) (F 1)))(lambda (X) (+ X U))→ 1
```

- Ever wanted to determine the \$PRINTER environment variable in a Java applet? (sorry forbidden, since the semantics of dynamic binding are unclear.)
- $\mathcal{DLC}$ is ideal for that (about time too!)
- **Example 10.43 (LISP)** give $\mathtt{let}_U$ the type $\lambda F.F{\Uparrow}^\circ_U$, where $(\mathbb{A}, U^-){\Uparrow}^\circ_U = \mathbb{A}, U^\circ$. (no need for U^+ in LISP)
- **Example 10.44 (Java)** If you want to keep your \$EDITOR variable private (pirated?)

  only allow applets of type $\mathbb{A}\#\alpha$, where $\mathtt{\$EDITOR} \notin \mathcal{DP}(\mathbb{A})$.
- It is going to be a lot of fun!

We will now contrast DRT with a modal logic for modeling imperative programs – incidentally also called "dynamic logic". This will give us new insights into the nature of dynamic phenomena in natural language.

10.3 Dynamic Logic for Imperative Programs

Dynamic Program Logic (DL)

- Modal logics for argumentation about imperative, non-deterministic programs
- Idea: Formalize the traditional argumentation about program correctness:

tracing the variable assignments (state) across program statements

▷ **Example 10.45 (Fibonacci)**
$\alpha := \langle Y, Z\rangle \hookleftarrow \langle 1, 1\rangle; \mathbf{while}\ X \neq 0\ \mathbf{do}\ \langle X, Y, Z\rangle \hookleftarrow \langle X-1, Z, Y+Z\rangle\ \mathbf{end}$

▷ States: $\langle 4, \_, \_\rangle, \langle 4, 1, 1\rangle, \langle 3, 1, 2\rangle, \langle 2, 2, 3\rangle, \langle 1, 3, 5\rangle, \langle 0, 5, 8\rangle$

▷ Assertions:

- ▹ Correctness: for positive X, running α with input $\langle X, \_, \_\rangle$ we end with $\langle 0, Fib(X-1), Fib(X)\rangle$
- ▹ Termination: α does not terminate on input $\langle -1, \_, \_\rangle$.

SOME RIGHTS RESERVED ©: Michael Kohlhase

Multi-Modal Logic fits well

▷ States as possible worlds, program statements as accessibility relation

▷ two syntactic categories: programs α and formulae $\mathbf{A}$.

▷ $[\alpha]\mathbf{A}$ as *If α terminates, then $\mathbf{A}$ holds afterwards*

▷ $\langle\alpha\rangle\mathbf{A}$ as *α terminates and $\mathbf{A}$ holds afterwards*.

▷ **Example 10.46** Assertions about Fibonacci (α)

- ▹ $\forall X, Y.[\alpha]Z = Fib(X)$
- ▹ $\forall X, Y.(X \geq 0) \Rightarrow \langle\alpha\rangle Z = Fib(X)$

SOME RIGHTS RESERVED ©: Michael Kohlhase

Levels of Description in Dynamic Logic

▷ **Definition 10.47** Propositional Dynamic Logic ($DL0$) (independent of variable assignments)

- ▹ $\models [\alpha]\mathbf{A} \wedge [\alpha]\mathbf{B} \Leftrightarrow [\alpha](\mathbf{A} \wedge \mathbf{B})$
- ▹ $\models [\mathbf{while}\ \mathbf{A} \vee \mathbf{B}\ \mathbf{do}\ \alpha\ \mathbf{end}]\mathbf{C} \Leftrightarrow [\mathbf{while}\ \mathbf{A}\ \mathbf{do}\ \alpha\ \mathbf{end}; \mathbf{while}\ \mathbf{B}\ \mathbf{do}\ \alpha; \mathbf{while}\ \mathbf{A}\ \mathbf{do}\ \alpha\ \mathbf{end}\ \mathbf{end}]\mathbf{C}$

first-order uninterpreted dynamic logic (DL1) (function, predicates uninterpreted)

▷ ▹ **Example 10.48** $\models p(f(X)) \Rightarrow g(Y, f(X)) \Rightarrow \langle Z \hookleftarrow f(X)\rangle p(Z, g(Y, Z))$
▹ **Example 10.49** $\models Z = Y \wedge (\forall X.f(g(X)) = X) \Rightarrow [\mathbf{while}\ p(Y)\ \mathbf{do}\ Y \hookleftarrow g(Y)\ \mathbf{end}]\langle\mathbf{while}\ Y \neq Z\ \mathbf{do}\ Y \hookleftarrow f(Y)\ \mathbf{end}\rangle T$

interpreted first-order dynamic logic (functions, predicates interpreted)

▷ ▹ $\forall X.\langle\mathbf{while}\ X \neq 1\ \mathbf{do}\ \mathbf{if}\ even(X)\ \mathbf{then}\ X \hookleftarrow \frac{X}{2}\ \mathbf{else} X \hookleftarrow 3X+1\ \mathbf{end}\ \mathbf{end}\rangle T$

SOME RIGHTS RESERVED ©: Michael Kohlhase

$DL0$ Syntax

▷ **Definition 10.50** Propositional Dynamic Logic ($DL0$) is PL^0 extended by

- ▷ program variable s $\mathcal{V}^\pi = \{\alpha, \beta, \gamma, \ldots\}$, modality $[\alpha]$, $\langle\alpha\rangle$.
- ▷ program constructor s $\Sigma^\pi = \{;, \cup, *, ?\}$ (minimal set)

$\alpha; \beta$	execute first α, then β	sequence
$\alpha \cup \beta$	execute (non-deterministically) either α or β	distribution
$*\alpha$	(non-deterministically) repeat α finitely often	iteration
$\mathbf{A}?$	proceed if $\models \mathbf{A}$, else error	test

▷ standard program primitives as derived concepts

Construct	as
if A then α **else**β **end**	$(\mathbf{A}?; \alpha) \cup (\neg\mathbf{A}?; \beta)$
while A do α **end**	$*(\mathbf{A}?; \alpha); \neg\mathbf{A}?$
repeat α **until A end**	$*(\alpha; \neg\mathbf{A}?); \mathbf{A}?$

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

$DL0$ Semantics

▷ **Definition 10.51** A model for $DL0$ consists of a set $\mathcal{W}$ of state s (possible worlds)

▷ **Definition 10.52** $DL0$ variable assignment s come in two parts:

- ▷ $\varphi \colon \mathcal{V}_o \times \mathcal{W} \to \{\mathsf{T}, \mathsf{F}\}$ (for propositional variables)
- ▷ $\pi \colon \mathcal{V}_o \to \mathcal{P}(\mathcal{W} \times \mathcal{W})$ (for program variables)

▷ **Definition 10.53** The meaning of complex formulae is given by the following value function $\mathcal{I}^w_\varphi \colon \mathit{wff}_o(\mathcal{V}_o) \to \mathcal{D}_o$

- ▷ $\mathcal{I}^w_{\varphi,\pi}(V) = \varphi(w, V)$ for $V \in \mathcal{V}_o$ and $\mathcal{I}^w_{\varphi,\pi}(V) = \pi(V)$ for $V \in \mathcal{V}^\pi$.
- ▷ $\mathcal{I}^w_{\varphi,\pi}(\neg\mathbf{A}) = \mathsf{T}$ iff $\mathcal{I}^w_{\varphi,\pi}(\mathbf{A}) = \mathsf{F}$
- ▷ $\mathcal{I}^w_{\varphi,\pi}([\alpha]\mathbf{A}) = \mathsf{T}$ iff $\mathcal{I}^{w'}_{\varphi,\pi}(\mathbf{A}) = \mathsf{T}$ for all $w' \in \mathcal{W}$ with $w\mathcal{I}^w_{\varphi,\pi}(\alpha)w'$.
- ▷ $\mathcal{I}^w_{\varphi,\pi}(\alpha; \beta) = \mathcal{I}^w_{\varphi,\pi}(\alpha) \circ \mathcal{I}^w_{\varphi,\pi}(\beta)$ (sequence)
- ▷ $\mathcal{I}^w_{\varphi,\pi}(\alpha \cup \beta) = \mathcal{I}^w_{\varphi,\pi}(\alpha) \cup \mathcal{I}^w_{\varphi,\pi}(\beta)$ (choice)
- ▷ $\mathcal{I}^w_{\varphi,\pi}(*\alpha) = \mathcal{I}^w_{\varphi,\pi}(\alpha)^*$ (transitive closure)
- ▷ $\mathcal{I}^w_{\varphi,\pi}(\mathbf{A}?) = \{\langle w, w\rangle \mid \mathcal{I}^w_{\varphi,\pi}(\mathbf{A}) = \mathsf{T}\}$ (test)

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

First-Order Program Logic (DL1)

▷ logic variables, constants, functions and predicates (uninterpreted), but no program variables

▷ **Definition 10.54 (Assignments)** ▷ nondeterministic assignment $X := ?$

▷ deterministic assignment $X := \mathbf{A}$

▷ **Example 10.55** $\models p(f(X)) \Rightarrow g(Y, f(X)) \Rightarrow \langle Z := f(X)\rangle p(Z, g(Y, Z))$

▷ **Example 10.56** $\models Z = Y \wedge (\forall X. p(f(g(X)) = X)) \Rightarrow [\textbf{while } p(Y) \textbf{ do } Y := g(Y) \textbf{ end}]\langle \textbf{while } Y \neq Z \textbf{ do } Y := f(Y) \textbf{ end}\rangle T$

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

DL1 Semantics

▷ **Definition 10.57** Let $\mathcal{M} = \langle \mathcal{D}, \mathcal{I}\rangle$ be a first-order model then we take the States (possible worlds) are variable assignments: $\mathcal{W} = \{\varphi \,|\, \varphi \colon \mathcal{V}_\iota \to \mathcal{D}\}$

▷ **Definition 10.58** Write $\varphi\,[\mathcal{X}]\,\psi$, iff $\varphi(X) = \psi(X)$ for all $X \notin \mathcal{X}$.

▷ **Definition 10.59** The meaning of complex formulae is given by the following value function $\mathcal{I}^w_\varphi \colon \mathit{wff}_o(\Sigma) \to \mathcal{D}_o$

▷ $\mathcal{I}^w_\varphi(\mathbf{A}) = \mathcal{I}_\varphi(\mathbf{A})$ if $\mathbf{A}$ term or atom.

▷ $\mathcal{I}^w_\varphi(\neg\mathbf{A}) = \mathsf{T}$ iff $\mathcal{I}^w_\varphi(\mathbf{A}) = \mathsf{F}$

▷ $\vdots$

▷ $\mathcal{I}^w_\varphi(X := ?) = \{\langle\varphi, \psi\rangle \,|\, \varphi\,[X]\,\psi\}$

▷ $\mathcal{I}^w_\varphi(X := \mathbf{A}) = \{\langle\varphi, \psi\rangle \,|\, \varphi\,[X]\,\psi$ and $\psi(X) = \mathcal{I}_\varphi(\mathbf{A})\}$.

▷ $\mathcal{I}_\varphi([X := \mathbf{A}]\mathbf{B}) = \mathcal{I}_{\varphi,[\mathcal{I}_\varphi(\mathbf{A})/X]}(\mathbf{B})$

▷ $\forall X.\mathbf{A}$ abbreviates $[X := ?]\mathbf{A}$

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

We will now establish a method for direct deduction on DRT, i.e. deduction at the representational level of DRT, without having to translate – and retranslate – before deduction.

10.4 Dynamic Model Generation

Deduction in Dynamic Logics

▷ Mechanize the dynamic entailment relation (with anaphora)

▷ Use dynamic deduction theorem to reduce (dynamic) entailment to (dynamic) satisfiability

▷ Direct Deduction on DRT (or DPL) [Saurer'93, Gabbay& Reyle'94, Monz& deRijke'98,...]

(++) Specialized Calculi for dynamic representations

(– –) Needs lots of development until we have efficient implementations

▷ Translation approach (used in our experiment)

(−) Translate to FOL

(++) Use off-the-shelf theorem prover (in this case MathWeb)

An Opportunity for Off-The-Shelf ATP?

▷ Pro: ATP is mature enough to tackle applications

- ▷ Current ATP are highly efficient reasoning tools
- ▷ Full automation is needed for NL processing (ATP as an oracle)
- ▷ ATP as logic engines is one of the initial promises of the field

▷ Contra: ATP are general logic systems

1. NLP uses other representation formalisms (DRT, Feature Logic,. . .)
2. ATP optimized for mathematical (combinatorially complex) proofs
3. ATP (often) do not terminate

Experiment: [Blackburn & Bos & Kohlhase & Nivelle'98]
Use translation approach for 1. to test 2. and 3. (Wow, it works!) Play with `http://www.coli.uni-sb.de/~bos/doris`

▷ Excursion: Incrementality in Dynamic Calculi

▷ For applications, we need to be able to check for

- ▷ consistency ($\exists \mathcal{M}.\mathcal{M} \models \mathbf{A}$), validity ($\forall \mathcal{M}.\mathcal{M} \models \mathbf{A}$) and
- ▷ entailment ($\mathcal{H} \models \mathbf{A}$, iff $\mathcal{M} \models \mathcal{H}$ implies $\mathcal{M} \models \mathbf{A}$ for all $\mathcal{M}$)

Deduction Theorem: $\mathcal{H} \models \mathbf{A}$, iff $\models \mathcal{H} \Rightarrow \mathbf{A}$. (valid for first-order Logic and DPL)

⧐ Problem: Analogue $\mathbf{H}_1 \otimes \cdots \otimes \mathbf{H}_n \models \mathbf{A}$ is not equivalent to $\models (\mathbf{H}_1 \otimes \cdots \otimes \mathbf{H}_n) \Rightarrow$ $\Rightarrow \mathbf{A}$ in DRT, since $\otimes$ symmetric.

▷ Thus: validity check cannot be used for entailment in DRT.

▷ Solution: Use sequential merge ;; (from DPL) for sentence composition

Model Generation for Dynamic Logics

- ▷ Problem: Translation approach is not incremental
  - ▷ For each check, the DRS for the whole discourse has to be translated
  - ▷ Can become infeasible, once discourses get large (e.g. novel)
  - ▷ This applies for all other approaches for dynamic deduction too
- ▷ Idea: Extend model generation techniques instead!
  - ▷ Remember: A DRS $\mathcal{D}$ is valid in $\mathcal{M} = \langle \mathcal{D}, \mathcal{I}^\delta \rangle$, iff $\mathcal{I}^\delta_\emptyset(\mathcal{D})^2 \neq \emptyset$
  - ▷ Find a model $\mathcal{M}$ and state φ, such that $\varphi \in \mathcal{I}^\delta_\emptyset(\mathcal{D})^2$.
  - ▷ Adapt first-order model generation technology for that

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

We will now introduce a "direct semantics" for DRT: a notion of "model" and an evaluation mapping that interprets DRSes directly – i.e. not via a translation of first-order logic. The main idea is that atomic conditions and conjunctions are interpreted largely like first-order formulae, while DRSes are interpreted as sets of assignments to discourse referents that make the conditions true. A DRS is satisfied by a model, if that set is non-empty.

A Direct Semantics for DRT (Dyn. Interpretation $\mathcal{I}^\delta_\varphi$)

- ▷ **Definition 10.60** Let $\mathcal{M} = \langle \mathcal{U}, \mathcal{I} \rangle$ be a FO Model and $\varphi, \psi \colon \mathcal{DR} \to \mathcal{U}$ be referent assignments, then we say that ψ extends φ on $\mathcal{X} \subseteq \mathcal{DR}$ (write $\varphi\,[\mathcal{X}]\,\psi$), if $\varphi(U) = \psi(U)$ for all $U \notin \mathcal{X}$.
- ▷ Idea: Conditions as truth values; DRSes as pairs $\langle \mathcal{X}, \mathcal{S} \rangle$ ($\mathcal{S}$ set of states)
- ▷ **Definition 10.61 (Meaning of complex formulae)** ▷ $\mathcal{I}^\delta_\varphi(p(a_1, \ldots, a_n))$ as always.
  - ▷ $\mathcal{I}^\delta_\varphi(\mathbf{A} \wedge \mathbf{B}) = \mathsf{T}$, iff $\mathcal{I}^\delta_\varphi(\mathbf{A}) = \mathsf{T}$ and $\mathcal{I}^\delta_\varphi(\mathbf{B}) = \mathsf{T}$.
  - ▷ $\mathcal{I}^\delta_\varphi(\neg\mathcal{D}) = \mathsf{T}$, if $\mathcal{I}^\delta_\varphi(\mathcal{D})^2 = \emptyset$.
  - ▷ $\mathcal{I}^\delta_\varphi(\mathcal{D} \veebar \mathcal{E}) = \mathsf{T}$, if $\mathcal{I}^\delta_\varphi(\mathcal{D})^2 \neq \emptyset$ or $\mathcal{I}^\delta_\varphi(\mathcal{E})^2 \neq \emptyset$.
  - ▷ $\mathcal{I}^\delta_\varphi(\mathcal{D} \Rrightarrow \mathcal{E}) = \mathsf{T}$, if for all $\psi \in \mathcal{I}^\delta_\varphi(\mathcal{D})^2$ there is a $\tau \in \mathcal{I}^\delta_\varphi(\mathcal{E})^2$ with $\psi \left[\mathcal{I}^\delta_\varphi(\mathcal{E})^1\right] \tau$.
  - ▷ $\mathcal{I}^\delta_\varphi(\delta\mathcal{X}.\mathbf{C}) = \langle \mathcal{X}, \{\psi \mid \varphi\,[\mathcal{X}]\,\psi \text{ and } \mathcal{I}^\delta_\psi(\mathbf{C}) = \mathsf{T}\} \rangle$.
  - ▷ $\mathcal{I}^\delta_\varphi(\mathcal{D} \otimes \mathcal{E}) = \mathcal{I}^\delta_\varphi(\mathcal{D};; \mathcal{E}) = \langle \mathcal{I}^\delta_\varphi(\mathcal{D})^1 \cup \mathcal{I}^\delta_\varphi(\mathcal{E})^1, \mathcal{I}^\delta_\varphi(\mathcal{D})^2 \cap \mathcal{I}^\delta_\varphi(\mathcal{E})^2 \rangle$

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

We can now fortify our intuition by computing the direct semantics of two sentences, which differ in their dynamic potential.

Examples (Computing Direct Semantics)

▷ **Example 10.62** *Peter owns a car*

$$\begin{aligned}
& \mathcal{I}^\delta_\varphi(\delta U.\mathsf{car}(U) \wedge \mathsf{own}(\mathsf{peter}, U)) \\
= \; & \langle\{U\}, \{\psi \mid \varphi\,[\mathcal{X}]\,\psi \text{ and } \mathcal{I}^\delta_\psi(\mathsf{car}(U) \wedge \mathsf{own}(\mathsf{peter}, U)) = \mathsf{T}\}\rangle \\
= \; & \langle\{U\}, \{\psi \mid \varphi\,[\mathcal{X}]\,\psi \text{ and } \mathcal{I}^\delta_\psi(\mathsf{car}(U)) = \mathsf{T} \text{ and } \mathcal{I}^\delta_\psi(\mathsf{own}(\mathsf{peter}, U)) = \mathsf{T}\}\rangle \\
= \; & \langle\{U\}, \{\psi \mid \varphi\,[\mathcal{X}]\,\psi \text{ and } \psi(U) \in \mathsf{car} \text{ and } \langle\psi(U), \mathsf{peter}\rangle \in \mathsf{own}\}\rangle
\end{aligned}$$

The set of states $[a/U]$, such that a is a car and is owned by Peter

▷ **Example 10.63** For *Peter owns no car* we look at the condition:

$$\begin{aligned}
& \mathcal{I}^\delta_\varphi(\neg(\delta U.\mathsf{car}(U) \wedge \mathsf{own}(\mathsf{peter}, U))) = \mathsf{T} \\
\Leftrightarrow \; & \mathcal{I}^\delta_\varphi(\delta U.\mathsf{car}(U) \wedge \mathsf{own}(\mathsf{peter}, U))^2 = \emptyset \\
\Leftrightarrow \; & \langle\{U\}, \{\psi \mid \varphi\,[\mathcal{X}]\,\psi \text{ and } \psi(U) \in \mathsf{car} \text{ and } \langle\psi(U), \mathsf{peter}\rangle \in \mathsf{own}\}\rangle^2 = \emptyset \\
\Leftrightarrow \; & \{\psi \mid \varphi\,[\mathcal{X}]\,\psi \text{ and } \psi(U) \in \mathsf{car} \text{ and } \langle\psi(U), \mathsf{peter}\rangle \in \mathsf{own}\} = \emptyset
\end{aligned}$$

i.e. iff there are no a, that are cars and that are owned by Peter.

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

The first thing we see in Example 10.62 is that the dynamic potential can directly be read off the direct interpretation of a DRS: it is the domain of the states in the first component. In Example 10.63, the interpretation is of the form $\langle\emptyset, \mathcal{I}^\delta_\varphi(\mathcal{C})\rangle$, where $\mathcal{C}$ is the condition we compute the truth value of in Example 10.63.

Dynamic Herbrand Interpretation

▷ **Definition 10.64** We call a dynamic interpretation $\mathcal{M} = \langle\mathcal{U}, \mathcal{I}, \mathcal{I}^\delta_\varphi\rangle$ a dynamic Herbrand interpretation, if $\langle\mathcal{U}, \mathcal{I}\rangle$ is a Herbrand model.

▷ Can represent $\mathcal{M}$ as a triple $\langle\mathcal{X}, \mathcal{S}, \mathcal{B}\rangle$, where $\mathcal{B}$ is the Herbrand base for $\langle\mathcal{U}, \mathcal{I}\rangle$.

▷ **Definition 10.65** $\mathcal{M}$ is called finite, iff $\mathcal{U}$ is finite.

▷ **Definition 10.66** $\mathcal{M}$ is minimal, iff for all $\mathcal{M}'$ the following holds: $(\mathcal{B}(\mathcal{M}') \subseteq \mathcal{B}(\mathcal{M})) \Rightarrow \mathcal{M}' = \mathcal{M}$.

▷ **Definition 10.67** $\mathcal{M}$ is domain minimal if for all $\mathcal{M}'$ the following holds: $\#(\mathcal{U}(\mathcal{M})) \leq \#(\mathcal{U}(\mathcal{M}'))$.

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

Sorted DRT=DRT^{++} (Syntax)

▷ Two syntactic categories

Conditions $\quad \mathcal{C} \to p(a_1, \ldots, a_n) | (\mathcal{C}_1 \wedge \mathcal{C}_2) | \neg\mathcal{D} | (\mathcal{D}_1 \mathbin{\mathbb{V}} \mathcal{D}_2) | (\mathcal{D}_1 \Rrightarrow \mathcal{D}_2)$

DRSes $\quad \mathcal{D} \to (\delta U^1_{\mathbb{A}_1}, \ldots, U^n_{\mathbb{A}_n}.\mathcal{C}) | (\mathcal{D}_1)\mathcal{D}_2 | (\mathcal{D}_1)\mathcal{D}_2$

▷ **Example 10.68** $\delta U_{\mathbb{H}}, V_{\mathbb{N}}.\mathsf{farmer}(U) \wedge \mathsf{donkey}(V) \wedge \mathsf{own}(U, V) \wedge \mathsf{beat}(U, V)$

▷ τ-Equality:

$$\begin{array}{rcl} \delta\mathcal{X}\mathcal{C}_1 \otimes \delta\mathcal{Y}\mathcal{C}_2 & \to_\tau & \delta\mathcal{X},\mathcal{Y}\mathcal{C}_1 \wedge \mathcal{C}_2 \\ \delta\mathcal{X}\mathcal{C}_1;;\delta\mathcal{Y}\mathcal{C}_2 & \to_\tau & \delta\mathcal{X},\mathcal{Y}\mathcal{C}_1 \wedge \mathcal{C}_2 \end{array}$$

▷ Discourse Referents used instead of bound variables (specify scoping independently of logic)

▷ Idea: Semantics by mapping into sorted first-order Logic

Dynamic Model Generation Calculus

▷ Use a tableau framework, extend by state information and rules for DRSes.

$$\frac{\delta U_{\mathbb{A}}\mathbf{.A}^{\mathsf{t}} \quad \mathcal{H} = \{a^1,\ldots,a^n\} \quad w \notin \mathcal{H} \text{ new}}{\begin{array}{c}[a_1/U]\\ \neg[a_1/U](\mathbf{A})^{\mathsf{t}}\end{array} \Bigg| \ldots \Bigg| \begin{array}{c}[a_n/U]\\ \neg[a_n/U](\mathbf{A})^{\mathsf{t}}\end{array} \Bigg| \begin{array}{c}[w/U]\\ \neg[w/U](\mathbf{A})^{\mathsf{t}}\end{array}} \mathrm{RM}{:}\delta$$

▷ Mechanize ;; by adding representation at all leaves

▷ Treat conditions by translation

$$\frac{\neg\mathcal{D}}{\neg\mathcal{D}} \qquad \frac{\mathcal{D} \Rrightarrow \mathcal{D}'}{\mathcal{D} \Rrightarrow \mathcal{D}'} \qquad \frac{\mathcal{D} \vee\!\!\vee \mathcal{D}'}{\mathcal{D} \vee\!\!\vee \mathcal{D}'}$$

Example: *Peter is a man. No man walks*

without sorts

$\boxed{\mathsf{man}(\mathsf{peter})}$

$\boxed{\neg(\delta U\mathbf{.}\mathsf{man}(U) \wedge \mathsf{walk}(U))}$

$\forall X\mathbf{.}\mathsf{man}(X) \wedge \mathsf{walk}(X)^{\mathsf{f}}$

$\mathsf{man}(\mathsf{peter}) \wedge \mathsf{walk}(\mathsf{peter})^{\mathsf{f}}$

$\mathsf{man}(\mathsf{peter})^{\mathsf{f}}$ | $\mathsf{walk}(\mathsf{peter})^{\mathsf{f}}$

$\bot$

problem: 1000 men

$\Rightarrow$

1000 closed branches

with sort $\mathbb{M}\mathtt{ale}$

$\boxed{\mathsf{man}(\mathsf{peter})}$

$\boxed{\neg(\delta U_{\mathbb{M}\mathtt{ale}}\mathbf{.}\mathsf{walk}(U))}$

$\exists X_{\mathbb{M}\mathtt{ale}}\mathbf{.}\mathsf{walk}(X)^{\mathsf{f}}$

$\mathsf{walk}(\mathsf{peter})^{\mathsf{f}}$

▷ Dynamic Herbrand Interpretation:

$$\langle\{U_{\mathbb{A}}\},\{[\mathsf{peter}/U_{\mathbb{A}}]\},\{\mathsf{man}(\mathsf{peter})^{\mathsf{t}},\mathsf{walk}(\mathsf{peter})^{\mathsf{f}}\}\rangle$$

Example: Anaphora Resolution

A man sleeps. He snores

$$\boxed{\delta U_{\mathbb{Man}}.\mathsf{man}(U) \wedge \mathsf{sleep}(U)}$$
$$[c^1_{\mathbb{Man}}/U_{\mathbb{Man}}]$$
$$\mathsf{man}(c^1_{\mathbb{Man}})^{\mathsf{t}}$$
$$\mathsf{sleep}(c^1_{\mathbb{Man}})^{\mathsf{t}}$$
$$\boxed{\delta V_{\mathbb{Man}}.\mathsf{snore}(V)}$$

$[c^1_{\mathbb{Man}}/V_{\mathbb{Man}}]$	$[c^2_{\mathbb{Man}}/V_{\mathbb{Man}}]$
$\mathsf{snore}(c^1_{\mathbb{Man}})^{\mathsf{t}}$	$\mathsf{snore}(c^2_{\mathbb{Man}})^{\mathsf{t}}$
minimal	deictic

SOME RIGHTS RESERVED

Anaphora with World Knowledge

▷ *Mary is married to Jeff. Her husband is not in town.*

▷ $\delta U_{\mathbb{F}}, V_{\mathbb{M}}.U = \mathsf{mary} \wedge \mathsf{married}(U, V) \wedge V = \mathsf{jeff};; \delta W_{\mathbb{M}}, W'_{\mathbb{F}}.\mathsf{hubby}(W, W') \wedge \neg\mathsf{intown}(W)$

▷ World knowledge

- ▹ if a female X is married to a male Y, then Y is X's only husband
- ▹ $\forall X_{\mathbb{F}}, Y_{\mathbb{M}}.\mathsf{married}(X, Y) \Rightarrow \mathsf{hubby}(Y, X) \wedge (\forall Z.\mathsf{hubby}(Z, X) \Rightarrow Z = Y)$

▷ Model generation yields tableau, all branches contain

$$\langle \{U, V, W, W'\}, \{[\mathsf{mary}/U], [\mathsf{jeff}/V], [\mathsf{jeff}/W], [\mathsf{mary}/W']\}, \mathcal{H} \rangle$$

with

$$\mathcal{H} = \{\mathsf{married}(\mathsf{mary}, \mathsf{jeff})^{\mathsf{t}}, \mathsf{hubby}(\mathsf{jeff}, \mathsf{mary})^{\mathsf{t}}, \neg\mathsf{intown}(\mathsf{jeff})^{\mathsf{t}}\}$$

▷ they only differ in additional negative facts, e.g. $\mathsf{married}(\mathsf{mary}, \mathsf{mary})^{\mathsf{f}}$.

SOME RIGHTS RESERVED

Model Generation models Discourse Understanding

▷ Conforms with psycholinguistic findings:

▷ [Zwaan'98]: listeners not only represent logical form, but also models containing referents

▷ [deVega'95]: online, incremental process

▷ [Singer'94]: enriched by background knowledge

▷ [Glenberg'87]: major function is to provide basis for anaphor resolution

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

11 Higher-Order Unification and NL Semantics Reconstruction

11.1 Introduction

Application of HOL in NL Semantics: Ellipsis

▷ **Example 11.1** *John loves his wife. George does too*

- ▷ $\mathsf{love}(\mathsf{john}, \mathsf{wife\_of}(\mathsf{john})) \wedge Q(\mathsf{george})$
- ▷ "*George* has property some Q, which we still have to determine"

Idea: If *John* has property Q, then it is that he *loves his wife*.

▷ Equation: $Q(\mathsf{john}) =_{\alpha\beta\eta} \mathsf{love}(\mathsf{john}, \mathsf{wife\_of}(\mathsf{john}))$

▷ Solutions (computed by HOU):

- ▷ $Q = \lambda z.\mathsf{love}(z, \mathsf{wife\_of}(z))$ and $Q = \lambda z.\mathsf{love}(z, \mathsf{wife\_of}(\mathsf{john}))$
- * $Q = \lambda z.\mathsf{love}(\mathsf{john}, \mathsf{wife\_of}(z))$ and $Q = \lambda z.\mathsf{love}(\mathsf{john}, \mathsf{wife\_of}(\mathsf{john}))$

▷ Readings: *George loves his own wife.* and *George loves Johns wife.*

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

Higher-Order Unification (HOU)

▷ Intuitively: Equation solving in the simply typed λ-calculus (modulo the built-in $\alpha\beta\eta$-equality)

▷ Formally: given formulae $\mathbf{A}, \mathbf{B} \in wff_\alpha(\Sigma, \mathcal{V}_\mathcal{T})$, find a substitution σ with $\sigma(\mathbf{A}) =_{\alpha\beta\eta} \sigma(\mathbf{B})$.

▷ **Definition 11.2** We call $\mathcal{E} := (\mathbf{A}^1 =^? \mathbf{B}^1) \wedge \ldots \wedge (\mathbf{A}^n =^? \mathbf{B}^n)$ a unification problem. The set $\mathbf{U}(\mathcal{E}) = \{\sigma \,|\, \sigma(\mathbf{A}^i) =_{\alpha\beta\eta} \sigma(\mathbf{B}^i)\}$ is called the set of unifiers for $\mathcal{E}$ and any of its members a unifier.

▷ **Example 11.3** the unification problem $F(fa) =^? f(Fa)$ where $F, f : \alpha \to \alpha$ and $\vdash_\Sigma a : \alpha$ has unifiers $[f/F], [\lambda X_\alpha.f(fX)/F], [\lambda X_\alpha.f(f(fX))/F], \ldots$

▷ find Representatives that induce all of $\mathbf{U}(\mathcal{E})$(are there most general unifiers?)

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

Discourse Coherence

▷ Meaning of a Discourse is more than just the conjunction of sentences

▷ Coherence is prerequisite for well-formedness (not just pragmatics)

A John killed Peter.

B^1 No, John killed BILL!

*B^2 *No, John goes hiking!*

B^3 No, PETER died in that fight!

▷ Coherence in a discourse is achieved by Discourse Relations

- ▷ in this case "Contrastive Parallelism"

SOME RIGHTS RESERVED

Discourse Relations (Examples)

▷ Parallel: *John organized rallies for Clinton, and Fred distributed pamphlets for him.*

▷ Contrast: *John supported Clinton, but Mary opposed him.*

▷ Exemplification: *Young aspiring politicians often support their party's presidential candidate. For instance John campaigned hard for Clinton in 1996.*

▷ Generalization: *John campaigned hard for Clinton in 1996. Young aspiring politicians often support their party's presidential candidate.*

▷ Elaboration: *A young aspiring politician was arrested in Texas today. John Smith, 34, was nabbed in a Houston law firm while attempting to embezzle funds for his campaign.*

SOME RIGHTS RESERVED

Discourse Relations (The General Case)

▷ We need inferences to discover them

▷ General conditions [Hobbs 1990]

Relation	Requirements	Particle
Parallel	$a_i \sim b_i, p \mathrel{=\!\mid\!\models} q$	*and*
Contrast	$a_i \sim b_i$, $p \models \neg q$ or $\neg p \models q$ a_i, b_i contrastive	*but*
Exempl.	$p \mathrel{=\!\mid} q$, $a_i \in \vec{b}$ or $a_i \mathrel{=\!\mid} b_i$	*for example*
Generl.	$p \mathrel{=\!\mid} q$, $b_i \in \vec{a}$ or $b_i \mathrel{=\!\mid} a_i$	*in general*
Elabor.	$q \simeq p$, $a_i \sim b_i$	*that is*

Source semantics $p(a_1, \ldots, a_n)$, Target semantics $q(a1, \ldots, a_m)$

▷ Need theorem proving methods for general case.

Underspecification/Ellipsis

- ▷ Natural language is economic
- ▷ Use the hearer's inferential capabilities to reduce communication costs.
- ▷ Makes use of Discourse Coherence for reconstruction (here: Parallelism)
  - ▷ *Jon loves his wife. Bill does too.* [love his/Bill's wife]
  - ▷ *Mary wants to go to Spain and Fred wants to go to Peru, but because of limited resources, only one of them can.* [go where he/she wants to go]
- ▷ Anaphora give even more Coherence. (here: Elaboration)
  - ▷ *I have a new car. It is in the parking lot downstairs.* [My new car]
- ▷ Discourse Relation determines the value of underspecified element.

Analyses based on Parallelism

- ▷ HOU Analyses (the structural level)
  - ▷ Ellipsis [DSP'91, G&K'96, DSP'96, Pinkal, et al'97]
  - ▷ Focus [Pulman'95, G&K96]
  - ▷ Corrections [G&K& v. Leusen'96]
  - ▷ Deaccenting, Sloppy Interpretation [Gardent, 1996]
- ▷ Discourse Theories (the general case, needs deduction!)
  - ▷ Literature and Cognition [Hobbs, CSLI Notes'90]
  - ▷ Cohesive Forms [Kehler, PhD'95]

Problem: All assume parallelism structure: given a pair of parallel utterances, the parallel elements are taken as given.

11.2 Higher-Order Unification

We now come to a very important (if somewhat non-trivial and under-appreciated) algorithm: higher-order unification, i.e. unification in the simply typed λ-calculus, i.e. unification modulo $\alpha\beta\eta$ equality.

11.2.1 Higher-Order Unifiers

Before we can start solving the problem of higher-order unification, we have to become clear about the terms we want to use. It turns out that "most general $\alpha\beta\eta$ unifiers may not exist – as Theorem 11.8 shows, there may be infinitely descending chains of unifiers that become more an more general. Thus we will have to generalize our concepts a bit here.

▷ HOU: Complete Sets of Unifiers

▷ Question: Are there most general higher-order Unifiers?

▷ Answer: What does that mean anyway?

▷ **Definition 11.4** $\sigma =_{\beta\eta} \rho[W]$, iff $\sigma(X) =_{\alpha\beta\eta} \rho(X)$ for all $X \in W$. $\sigma =_{\beta\eta} \rho[\mathcal{E}]$ iff $\sigma =_{\beta\eta} \rho[\mathbf{free}(\mathcal{E})]$

▷ **Definition 11.5** σ is more general than θ on W ($\sigma \leq_{\beta\eta} \theta[W]$), iff there is a substitution ρ with $\theta =_{\beta\eta} \rho \circ \sigma[W]$.

▷ **Definition 11.6** $\Psi \subseteq \mathbf{U}(\mathcal{E})$ is a complete set of unifiers, iff for all unifiers $\theta \in \mathbf{U}(\mathcal{E})$ there is a $\sigma \in \Psi$, such that $\sigma \leq_{\beta\eta} \theta[\mathcal{E}]$.

▷ **Definition 11.7** If $\Psi \subseteq \mathbf{U}(\mathcal{E})$ is complete, then $\leq_{\beta\eta}$-minimal elements $\sigma \in \Psi$ are most general unifiers of $\mathcal{E}$.

▷ **Theorem 11.8** *The set* $\{[\lambda uv.hu/F]\} \cup \{\sigma_i \mid i \in \mathbb{N}\}$ *where*

$$\sigma_i := [\lambda uv.g_n u(u(h_1^n uv))\ldots(u(h_n^n uv))/F], [\lambda v.z/X]$$

is a complete set of unifiers for the equation $FXa_\iota =^? FXb_\iota$, *where* F *and* X *are variables of types* $(\iota \to \iota) \to \iota \to \iota$ *and* $\iota \to \iota$

Furthermore, σ_{i+1} *is more general than* σ_i.

▷ Proof Sketch: [Hue76, Theorem 5] □

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

The definition of a solved form in $\Lambda^{\to}$ is just as always; even the argument that solved forms are most general unifiers works as always, we only need to take $\alpha\beta\eta$ equality into account at every level.

Unification

▷ **Definition 11.9** $X^1 =^? \mathbf{B}^1 \wedge \ldots \wedge X^n =^? \mathbf{B}^n$ is in solved form, if the X^i are distinct free variables $X^i \notin \mathbf{free}(\mathbf{B}^j)$ and $\mathbf{B}^j$ does not contain Skolem constants for all j.

▷ **Lemma 11.10** *If* $\mathcal{E} = X^1 =^? \mathbf{B}^1 \wedge \ldots \wedge X^n =^? \mathbf{B}^n$ *is in solved form, then* $\sigma_{\mathcal{E}} := [\mathbf{B}^1/X^1], \ldots, [\mathbf{B}^n/X^n]$ *is the unique most general unifier of* $\mathcal{E}$

▷ Proof:

P.1 $\sigma(X^i) =_{\alpha\beta\eta} \sigma(\mathbf{B}^i)$, so $\sigma \in \mathbf{U}(\mathcal{E})$

P.2 Let $\theta \in \mathbf{U}(\mathcal{E})$, then $\theta(X^i) =_{\alpha\beta\eta} \theta(\mathbf{B}^i) = \theta \circ \sigma(X^i)$

P.3 so $\theta \leq_{\beta\eta} \theta \circ \sigma[\mathcal{E}]$. □

11.2.2 Higher-Order Unification Transformations

We are now in a position to introduce the higher-order unifiation transformations. We proceed just like we did for first-order unification by casting the unification algorithm as a set of unification inference rules, leaving the control to a second layer of development.

We first look at a group of transformations that are (relatively) well-behaved and group them under the concept of "simplification", since (like the first-order transformation rules they resemble) have good properties. These are usually implemented in a group and applied eagerly.

Simplification $\mathcal{SIM}$

▷ **Definition 11.11** The higher-order simplification transformations $\mathcal{SIM}$ consist of the rules below.

$$\frac{\lambda X_\alpha.\mathbf{A} =^? \lambda Y_\alpha.\mathbf{B} \wedge \mathcal{E} \quad s \in \Sigma_\alpha^{Sk}\text{new}}{[s/X](\mathbf{A}) =^? [s/Y](\mathbf{B}) \wedge \mathcal{E}}\,\mathcal{SIM}{:}\alpha$$

$$\frac{\lambda X_\alpha.\mathbf{A} =^? \mathbf{B} \wedge \mathcal{E} \quad s \in \Sigma_\alpha^{Sk}\text{new}}{[s/X](\mathbf{A}) =^? \mathbf{B}s \wedge \mathcal{E}}\,\mathcal{SIM}{:}\eta$$

$$\frac{h\overline{\mathbf{U}^n} =^? h\overline{\mathbf{V}^n} \wedge \mathcal{E} \quad h \in (\Sigma \cup \Sigma^{Sk})}{\mathbf{U}^n =^? \mathbf{V}^n, \mathcal{E} \wedge \ldots \wedge \mathbf{U}^n =^? \mathbf{V}^n \wedge \mathcal{E}}\,\mathcal{SIM}{:}\text{dec}$$

$$\frac{\mathcal{E} \wedge X =^? \mathbf{A} \quad X \notin \mathbf{free}(\mathbf{A}) \quad \mathbf{A} \cap \Sigma^{Sk} = \emptyset \quad X \in \mathbf{free}(\mathcal{E})}{[\mathbf{A}/X](\mathcal{E}) \wedge X =^? \mathbf{A}}\,\mathcal{SIM}{:}\text{elim}$$

After rule applications all λ-terms are reduced to head normal form.

The main new feature of these rules (with respect to their first-order counterparts) is the handling of λ-binders. We eliminate them by replacing the bound variables by Skolem constants in the bodies: The $\mathcal{SIM}:\alpha$ standardizes them to a single one using α-equality, and $\mathcal{SIM}:\eta$ first η-expands the right-hand side (which must be of functional type) so that $\mathcal{SIM}:\alpha$ applies. Given that we are setting bound variables free in this process, we need to be careful that we do not use them in the $\mathcal{SIM}$:elim rule, as these would be variable-capturing.

Consider for instance the higher-order unification problem $\lambda X.X =^? \lambda Y.W$, which is unsolvable (the left hand side is the identity function and the right hand side some constant function – whose value is given by W). So after an application of $\mathcal{SIM}:\alpha$, we have $c =^? W$, which looks like it could be a solved pair, but the elimination rule prevents that by insisting that instances may not contain Skolem Variables.

Conceptually, $\mathcal{SIM}$ is a direct generalization of first-order unification transformations, and shares it properties; even the proofs go correspondingly.

Properties of Simplification

▷ **Lemma 11.12 (Properties of $\mathcal{SIM}$)** *$\mathcal{SIM}$ generalizes first-order unification.*

- ▷ *$\mathcal{SIM}$ is terminating and confluent up to α-conversion*
- ▷ *Unique $\mathcal{SIM}$ normal forms exist (all pairs have the form $h\overline{\mathbf{U}^n} =^? k\overline{\mathbf{V}^m}$)*

▷ **Lemma 11.13** $\mathbf{U}(\mathcal{E} \wedge \mathcal{E}_\sigma) = \mathbf{U}(\sigma(\mathcal{E}) \wedge \mathcal{E}_\sigma)$.

Proof: by the definitions

▷ **P.1** If $\theta \in \mathbf{U}(\mathcal{E} \wedge \mathcal{E}_\sigma)$, then $\theta \in (\mathbf{U}(\mathcal{E}) \cap \mathbf{U}(\mathcal{E}_\sigma))$.

So $\theta =_{\beta\eta} \theta \circ \sigma[\mathbf{supp}(\sigma)]$,

and thus $(\theta \circ \sigma) \in \mathbf{U}(\mathcal{E})$, iff $\theta \in \mathbf{U}(\sigma(\mathcal{E}))$. □

P.2 P.3▷ **Theorem 11.14** *If $\mathcal{E} \vdash_{\mathcal{SIM}} \mathcal{F}$, then $\mathbf{U}(\mathcal{E}) \leq_{\beta\eta} \mathbf{U}(\mathcal{F})[\mathcal{E}]$. (correct, complete)*

Proof: By an induction over the length of the derivation

P.1 We the $\mathcal{SIM}$ rules individually for the base case

P.1.1 $\mathcal{SIM}{:}\alpha$: by α-conversion □

P.1.2 $\mathcal{SIM}{:}\eta$: By η-conversion in the presence of $\mathcal{SIM}{:}\alpha$ □

P.1.3 $\mathcal{SIM}$:dec: The head $h \in (\Sigma \cup \Sigma^{Sk})$ cannot be instantiated. □

P.1.4 $\mathcal{SIM}$:elim: By Lemma 11.13. □

P.2 The step case goes directly by inductive hypothesis and transitivity of derivation. □

SOME RIGHTS RESERVED ©: Michael Kohlhase 312 JACOBS UNIVERSITY

Now that we have simplification out of the way, we have to deal with unification pairs of the form $h\overline{\mathbf{U}^n} =^? k\overline{\mathbf{V}^m}$. Note that the case where both h and k are contstants is unsolvable, so we can assume that one of them is a variable. The unification problem $F_{\alpha\to\alpha}a =^? a$ is a particularly simple example; it has solutions $[\lambda X_\alpha.a/F]$ and $[\lambda X_\alpha.X/F]$. In the first, the solution comes by instantiating F with a λ-term of type $\alpha \to \alpha$ with head a, and in the second with a 1-projection term of type $\alpha \to \alpha$, which projects the head of the argument into the right position. In both cases, the solution came from a term with a given type and an appropriate head. We will look at the problem of finding such terms in more detail now.

General Bindings

▷ Problem: Find all formulae of given type α and head h.

▷ sufficient: long $\beta\eta$ head normal form, most general

▷ General Bindings: $\mathbf{G}^h_\alpha(\Sigma) := \lambda\overline{X^k_\alpha}.h(H^1\overline{X})\ldots(H^n\overline{X})$

- ▷ where $\alpha = \overline{\alpha_k} \to \beta$, $h : \overline{\gamma_n} \to \beta$ and $\beta \in \mathcal{BT}$
- ▷ and $H^i : \overline{\alpha_k} \to \gamma_i$ new variables.

▷ **Observation 11.15** *General bindings are unique up to choice of names for H^i.*

▷ **Definition 11.16** If the head h is j^{th} bound variable in $\mathbf{G}^h_\alpha(\Sigma)$, call $\mathbf{G}^h_\alpha(\Sigma)$ j-projection binding (and write $\mathbf{G}^j_\alpha(\Sigma)$) else imitation binding

▷ clearly $\mathbf{G}^h_\alpha(\Sigma) \in \mathit{wff}_\alpha(\Sigma, \mathcal{V}_\mathcal{T})$ and $\mathrm{head}(\mathbf{G}^h_\alpha(\Sigma)) = h$

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

For the construction of general bindings, note that their construction is completely driven by the intended type α and the (type of) the head h. Let us consider some examples.

Example 11.17 The following general bindings may be helpful: $\mathbf{G}^{a_\iota}_{\iota\to\iota}(\Sigma) = \lambda X_\iota.a$, $\mathbf{G}^{a_\iota}_{\iota\to\iota\to\iota}(\Sigma) = \lambda X_\iota Y_\iota.a$, and $\mathbf{G}^{a_{\iota\to\iota}}_{\iota\to\iota\to\iota}(\Sigma) = \lambda X_\iota Y_\iota.a(HXY)$, where H is of type $\iota\to\iota\to\iota$

We will now show that the general bindings defined in Definition 11.16 are indeed the most general λ-terms given their type and head.

Approximation Theorem

▷ **Theorem 11.18** *If $\mathbf{A} \in \mathit{wff}_\alpha(\Sigma, \mathcal{V}_\mathcal{T})$ with head$(\mathbf{A}) = h$, then there is a general binding $\mathbf{G} = \mathbf{G}^h_\alpha(\Sigma)$ and a substitution ρ with $\rho(\mathbf{G}) =_{\alpha\beta\eta} \mathbf{A}$ and dp$(\rho) <$ dp$(\mathbf{A})$.*

▷ Proof: We analyze the term structure of $\mathbf{A}$

P.1 If $\alpha = \overline{\alpha_k} \to \beta$ and $h : \overline{\gamma_n} \to \beta$ where $\beta \in \mathcal{BT}$, then the long head normal form of $\mathbf{A}$ must be $\lambda \overline{X^k_\alpha}.h\overline{\mathbf{U}^n}$.

P.2 $\mathbf{G} = \mathbf{G}^h_\alpha(\Sigma) = \lambda \overline{X^k_\alpha}.h(H^1\overline{X}) \ldots (H^n\overline{X})$ for some variables $H^i : \overline{\alpha_k} \to \gamma_i$.

P.3 Choose $\rho := [\lambda \overline{X^k_\alpha}.\mathbf{U}^1/H^1], \ldots, [\lambda \overline{X^k_\alpha}.\mathbf{U}^n/H^n]$.

P.4 Then we have
$$\begin{aligned}\rho(\mathbf{G}) &= \lambda \overline{X^k_\alpha}.h(\lambda \overline{X^k_\alpha}.\mathbf{U}^1\overline{X}) \ldots (\lambda \overline{X^k_\alpha}.\mathbf{U}^n\overline{X})\\ &=_{\beta\eta} \lambda \overline{X^k_\alpha}.h\overline{\mathbf{U}^n}\\ &=_{\beta\eta} \mathbf{A}\end{aligned}$$

P.5 The depth condition can be read off as dp$(\lambda \overline{X^k_\alpha}.\mathbf{U}^1) \leq$ dp$(\mathbf{A}) - 1$. □

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

With this result we can state the higher-order unification transformations.

p Higher-Order Unification ($\mathcal{HOU}$)

▷ Recap: After simplification, we have to deal with pairs where one (flex/rigid) or both heads (flex/flex) are variables

▷ **Definition 11.19** Let $\mathbf{G} = \mathbf{G}^h_\alpha(\Sigma)$ (imitation) or $\mathbf{G} \in \{\mathbf{G}^j_\alpha(\Sigma) \mid 1 \leq j \leq n\}$, then $\mathcal{HOU}$ consists of the transformations (always reduce to $\mathcal{SIM}$ normal form)

▷ Rule for flex/rigid pairs: $$\frac{F_\alpha\overline{\mathbf{U}} =^? h\overline{\mathbf{V}} \wedge \mathcal{E}}{F =^? \mathbf{G} \wedge F\overline{\mathbf{U}} =^? h\overline{\mathbf{V}} \wedge \mathcal{E}}\,\mathcal{HOU}\text{:fr}$$

▷ Rules for flex/flex pairs: $$\frac{F_\alpha\overline{\mathbf{U}} =^? H\overline{\mathbf{V}} \wedge \mathcal{E}}{F =^? \mathbf{G} \wedge F\overline{\mathbf{U}} =^? H\overline{\mathbf{V}} \wedge \mathcal{E}}\,\mathcal{HOU}\text{:ff}$$

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

Let us now fortify our intuition with a simple example.

$\mathcal{HOU}$ Example

Example 11.20 Let $Q, w : \iota \to \iota$, $l : \iota \to \iota \to \iota$, and $j : \iota$, then we have the following derivation tree in $\mathcal{HOU}$.

$Q(j) =^? l(j, w(j))$

- $Q=\lambda X.X$: $j =^? l(j, w(j))$
- $Q=\lambda X.l(H(X),K(X))$: $l(H(j), K(j)) =^? l(j, w(j))$
  - $H(j) =^? j \wedge K(j) =^? w(j)$
    - $H=\lambda X.X$: $j =^? j \wedge K(j) =^? w(j)$
      - $K=\lambda X.X$
      - $K=\lambda X.w(K'(X))$: $j =^? j \wedge K'(j) =^? j$
        - $K'=\lambda X.X$: $j \doteq j; j \doteq j$ ⋮ $Q = \lambda X.l(X, w(X))$
        - $K'=\lambda X.j$: $j \doteq j; j \doteq j$ ⋮ $\lambda X.l(X, w(j))$
    - $H=\lambda X.j$: $j =^? j \wedge K(j) =^? w(j)$
      - $K=\lambda X.w(K'(X))$: $j =^? j \wedge K'(j) =^? j$
        - $K'=\lambda X.X$: $j \doteq j; j \doteq j$ ⋮ $\lambda X.l(j, w(X))$
        - $K'=\lambda X.j$: $j \doteq j; j \doteq j$ ⋮ $\lambda X.l(j, w(j))$
      - $K=\lambda X.X$

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

The first thing that meets the eye is that higher-order unification is branching. Indeed, for flex/-rigid pairs, we have to systematically explore the possibilities of binding the head variable the imitation binding and all projection bindings. On the initial node, we have two bindings, the projection binding leads to an unsolvable unification problem, whereas the imitation binding leads to a unification problem that can be decomposed into two flex/rigid pairs. For the first one of them, we have a projection and an imitation binding, which we systematically explore recursively. Eventually, we arrive at four solutions of the initial problem.

The following encoding of natural number arithmetics into $\Lambda^{\to}$ is useful for testing our unification algorithm

A Test Generator for Higher-Order Unification

▷ **Definition 11.21 (Church Numerals)** We define closed λ-terms of type $\nu := (\alpha \to \alpha) \to \alpha \to \alpha$

▷ Numbers: Church numeral s: (n-fold iteration of arg1 starting from arg2)

$$n := \lambda S_{\alpha\to\alpha}.\lambda O_{\alpha}.\underbrace{S(S\ldots S}_{n}(O)\ldots)$$

▷ Addition (N-fold iteration of S from N)

$$+ := \lambda N_{\nu} M_{\nu}.\lambda S_{\alpha\to\alpha}.\lambda O_{\alpha}.NS(MSO)$$

▷ Multiplication: (N-fold iteration of MS ($=+m$) from O)

$$\cdot := \lambda N_{\nu} M_{\nu}.\lambda S_{\alpha\to\alpha}.\lambda O_{\alpha}.N(MS)O$$

▷ **Observation 11.22** *Subtraction and (integer) division on Church numberals can be automted via higher-order unification.*

▷ **Example 11.23** $5-2$ by solving the unification problem $2+x_{\nu} =^{?} 5$

Equation solving for Church numerals yields a very nice generator for test cases for higher-order unification, as we know which solutions to expect.

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

11.2.3 Properties of Higher-Order Unification

We will now establish the properties of the higher-order unification problem and the algorithms we have introduced above. We first establish the unidecidability, since it will influence how we go about the rest of the properties.

We establish that higher-order unification is undecidable. The proof idea is a typical for undecidable proofs: we reduce the higher-order unification problem to one that is known to be undecidable: here, the solution of Diophantine equations $\mathbb{N}$.

▷ Undecidability of Higher-Order Unification

▷ **Theorem 11.24** *Second-order unification is undecidable* *(Goldfarb '82 [Gol81])*

▷ Proof Sketch: Reduction to Hilbert's tenth problem (solving Diophantine equations) (known to be undecidable)

□

▷ **Definition 11.25** We call an equation a Diophantine equation, if it is of the form

- ▷ $x_i x_j = x_k$
- ▷ $x_i + x_j = x_k$
- ▷ $x_i = c_j$ where $c_j \in \mathbb{N}$

where the variables x_i range over $\mathbb{N}$.

▷ These can be solved by higher-order unification on Church numerals. (cf. Observation 11.22)

.

▷ **Theorem 11.26** *The general solution for sets of Diophantine equations is undecidable.* *(Matijasevič 1970 [Mat70])*

SOME RIGHTS RESERVED ©: Michael Kohlhase 318 JACOBS UNIVERSITY

The argument undecidability proofs is always the same: If higher-order unification were decidable, then via the encoding we could use it to solve Diophantine equations, which we know we cannot by Matijasevič's Theorem.

The next step will be to analyze our transformations for higher-order unification for correctness and completeness, just like we did for first-order unification.

$\mathcal{HOU}$ is Correct

▷ **Lemma 11.27** *If $\mathcal{E} \vdash_{\mathcal{HOU}\mathrm{fr}} \mathcal{E}'$ or $\mathcal{E} \vdash_{\mathcal{HOU}\mathrm{ff}} \mathcal{E}'$, then $\mathbf{U}(\mathcal{E}') \subseteq \mathbf{U}(\mathcal{E})$.*

▷ Proof Sketch: $\mathcal{HOU}$:fr and $\mathcal{HOU}$:ff only add new pair. □

▷ **Corollary 11.28** *$\mathcal{HOU}$ is correct: If $\mathcal{E} \vdash_{\mathcal{HOU}} \mathcal{E}'$, then $\mathbf{U}(\mathcal{E}') \subseteq \mathbf{U}(\mathcal{E})$.*

SOME RIGHTS RESERVED ©: Michael Kohlhase 319 JACOBS UNIVERSITY

Given that higher-order unification is not unitary and undecidable, we cannot just employ the notion of completeness that helped us in the analysis of first-order unification. So the first thing is to establish the condition we want to establish to see that $\mathcal{HOU}$ gives a higher-order unification algorithm.

Completeness of $\mathcal{HOU}$

▷ We cannot expect completeness in the same sense as for first-order unification: "If $\mathcal{E} \vdash_{\mathcal{U}} \mathcal{F}$, then $\mathbf{U}(\mathcal{E}) \subseteq \mathbf{U}(\mathcal{F})$" (see Lemma 3.79) as the rules fix a binding and thus partially commit to a unifier (which excludes others).

▷ We cannot expect termination either, since HOU is undecidable.

▷ For a semi-decision procedure we only need termination on unifiable problems.

▷ **Theorem 11.29 ($\mathcal{HOU}$ derives Complete Set of Unifiers)** *If $\sigma \in \mathbf{U}(\mathcal{E})$, then there is a $\mathcal{HOU}$-derivation $\mathcal{E} \vdash_{\mathcal{HOU}} \mathcal{F}$, such that $\mathcal{F}$ is in solved form, $\sigma_{\mathcal{F}} \in \mathbf{U}(\mathcal{E})$, and $\sigma_{\mathcal{F}}$ is more general than θ.*

▷ Proof Sketch: Given a unifier θ of $\mathcal{E}$, we guide the derivation with a measure μ_θ towards $\mathcal{F}$. □

SOME RIGHTS RESERVED ©: Michael Kohlhase 320 JACOBS UNIVERSITY

So we will embark on the details of the completeness proof. The first step is to define a measure that will guide the $\mathcal{HOU}$ transformation out of a unification problem $\mathcal{E}$ given a unifier θ of cE.

Completeness of $\mathcal{HOU}$ (Measure)

▷ **Definition 11.30** We call $\mu(\mathcal{E},\theta) := \langle \mu_1(\mathcal{E},\theta), \mu_2(\theta) \rangle$ the unification measure for $\mathcal{E}$ and θ, if

- ▷ $\mu_1(\mathcal{E},\theta)$ is the multiset of term depths of $\theta(X)$ for the unsolved $X \in \mathbf{supp}(\theta)$.
- ▷ $\mu_2(\mathcal{E})$ the multiset of term depths in $\mathcal{E}$.
- ▷ $\prec$ is the strict lexicographic order on pairs: $(\langle a,b\rangle \prec \langle c,d\rangle$, if $a<c$ or $a=c$ and $b<d)$
- ▷ Component orderings are multiset orderings: $(M \cup \{m\} < M \cup N$ iff $n < m$ for all $n \in N)$

▷ **Lemma 11.31** *$\prec$ is well-founded.* *(by construction)*

SOME RIGHTS RESERVED ©: Michael Kohlhase 321 JACOBS UNIVERSITY

This measure will now guide the $\mathcal{HOU}$ transformation in the sense that in any step it chooses whether to use $\mathcal{HOU}$:fr or $\mathcal{HOU}$:ff, and which general binding (by looking at what θ would do). We formulate the details in Theorem 11.32 and look at their consequences before we proove it.

Completeness of $\mathcal{HOU}$ (μ-Prescription)

▷ **Theorem 11.32** *If $\theta \in \mathbf{U}(\mathcal{E})$, then there is a unification problem $\mathcal{F}$ with $\mathcal{E} \vdash_{\mathcal{HOU}} \mathcal{E}'$ and a substitution $\theta' \in \mathbf{U}(\mathcal{E}')$, such that*

- ▷ $\theta =_{\beta\eta} \theta'[\mathcal{E}]$
- ▷ $\mu(\mathcal{E}',\theta') \prec \mu(\mathcal{E},\theta)$.

we call such a $\mathcal{HOU}$-step a μ-prescribed

▷ **Corollary 11.33** *If $\mathcal{E}$ is unifiable without μ-prescribed $\mathcal{HOU}$-steps, then $\mathcal{E}$ is solved.*

▷ In other words: μ guides the $\mathcal{HOU}$-transformations to a solved form

SOME RIGHTS RESERVED ©: Michael Kohlhase 322 JACOBS UNIVERSITY

We now come to the proof of Theorem 11.32, which is a relatively simple consequence of Theorem 11.18.

Proof of Theorem 11.32

▷ Proof:

P.1 Let $\mathbf{A} =^? \mathbf{B}$ be an unsolved pair of the form $\mathbf{F}\overline{\mathbf{U}} =^? \mathbf{G}\overline{\mathbf{V}}$ in $\mathcal{F}$.

P.2 $\mathcal{E}$ is a $\mathcal{SIM}$ normal form, so $\mathbf{F}$ and $\mathbf{G}$ must be constants or variables,

P.3 but not the same constant, since otherwise $\mathcal{SIM}$:dec would be applicable.

P.4 We can also exclude $\mathbf{A} =_{\alpha\beta\eta} \mathbf{B}$, as $\mathcal{SIM}$:triv would be be appliccable.

P.5 If $\mathbf{F} = \mathbf{G}$ is a variable not in $\mathbf{supp}(\theta)$, then $\mathcal{SIM}$:dec appliccable. By correctness we have $\theta \in \mathbf{U}(\mathcal{E}')$ and $\mu(\mathcal{E}',\theta) \prec \mu(\mathcal{E},\theta)$, as $\mu_1(\mathcal{E}',\theta) \preceq \mu_1(\mathcal{E},\theta)$ and $\mu_2(\mathcal{E}') \prec \mu_2(\mathcal{E})$.

P.6 Otherwise we either have $\mathbf{F} \neq \mathbf{G}$ or $\mathbf{F} = \mathbf{G} \in \mathbf{supp}(\theta)$.

P.7 In both cases $\mathbf{F}$ or $\mathbf{G}$ is an unsolved variable $F \in \mathbf{supp}(\theta)$ of type α, since $\mathcal{E}$ is unsolved.

P.8 Without loss of generality we choose $F = \mathbf{F}$.

P.9 By Theorem 11.18 there is a general binding $\mathbf{G} = \mathbf{G}^f_\alpha(\Sigma)$ and a substitution ρ with $\rho(\mathbf{G}) =_{\alpha\beta\eta} \theta(F)$. So,

- ▷ if head$(\mathbf{G}) \notin \mathbf{supp}(\theta)$, then $\mathcal{HOU}$:fr is appliccable,
- ▷ if head$(\mathbf{G}) \in \mathbf{supp}(\theta)$, then $\mathcal{HOU}$:ff is appliccable.

P.10 Choose $\theta' := \theta \cup \rho$. Then $\theta =_{\beta\eta} \theta'[\mathcal{E}]$ and $\theta' \in \mathbf{U}(\mathcal{E}')$ by correctness.

P.11 $\mathcal{HOU}$:ff and $\mathcal{HOU}$:fr solve $F \in \mathbf{supp}(\theta)$ and replace F by $\mathbf{supp}(\rho)$ in the set of unsolved variable of $\mathcal{E}$.

P.12 so $\mu_1(\mathcal{E}', \theta') \prec \mu_1(\mathcal{E}, \theta)$ and thus $\mu(\mathcal{E}', \theta') \prec \mu(\mathcal{E}, \theta)$. □

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

We now convince ourselves that if $\mathcal{HOU}$ terminates with a unification problem, then it is either solved – in which case we can read off the solution – or unsolvable.

Terminal $\mathcal{HOU}$-problems are Solved or Unsolvable

▷ **Theorem 11.34** *If $\mathcal{E}$ is a unsolved UP and $\theta \in \mathbf{U}(\mathcal{E})$, then there is a $\mathcal{HOU}$-derivation $\mathcal{E} \vdash_{\mathcal{HOU}} \sigma_\sigma$, with $\sigma \leq_{\beta\eta} \theta[\mathcal{E}]$.*

▷ Proof: Let $\mathcal{D}$: $\mathcal{E} \vdash_{\mathcal{HOU}} \mathcal{F}$ a maximal μ-prescribed $\mathcal{HOU}$-derivation from $\mathcal{E}$.

P.1 This must be finite, since $\prec$ is well-founded (ind. over length n of $\mathcal{D}$)

P.2 If $n = 0$, then $\mathcal{E}$ is solved and $\sigma_\mathcal{E}$ most general unifier

P.3 thus $\sigma_\mathcal{E} \leq_{\beta\eta} \theta[\mathcal{E}]$

P.4 If $n > 0$, then there is a μ-prescribed step $\mathcal{E} \vdash_{\mathcal{HOU}} \mathcal{E}'$ and a substitution θ' as in Theorem 11.32.

P.5 by IH there is a $\mathcal{HOU}$-derivation $\mathcal{E}' \vdash_{\mathcal{HOU}} \mathcal{F}$ with $\sigma_\mathcal{F} \leq_{\beta\eta} \theta'[\mathcal{E}']$.

P.6 by correctness $\sigma_\mathcal{F} \in \mathbf{U}(\mathcal{E}') \subseteq \mathbf{U}(\mathcal{E})$.

P.7 rules of $\mathcal{HOU}$ only expand free variables, so $\sigma_\mathcal{F} \leq_{\beta\eta} \theta'[\mathcal{E}']$

P.8 Thus $\sigma_\mathcal{F} \leq_{\beta\eta} \theta'[\mathcal{E}]$,

P.9 This completes the proof, since $\theta' =_{\beta\eta} \theta[\mathcal{E}]$ by ?prescribed.thm?. □

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

We now recap the properties of higher-order unification (HOU) to gain an overview.

Properties of HO-Unification

▷ HOU is undecidable, HOU need not have most general unifiers

▷ The $\mathcal{HOU}$ transformation induce an algorithm that enumerates a complete set of higher-order unifiers.

▷ $\mathcal{HOU}$:ff gives enormous degree of indeterminism

▷ HOU is intractable in practice consider restricted fragments where it is!

▷ HO Matching (decidable up to order four), HO Patterns (unitary, linear), ...

11.2.4 Pre-Unification

We will now come to a variant of higher-order unification that is used in higher-order theorem proving, where we are only interested in the exgistence of a unifier – e.g. in mating-style tableaux. In these cases, we can do better than full higher-order unification.

Pre-Unification

- ▷ $\mathcal{HOU}$:ff has a giant branching factor in the search space for unifiers. (makes HOU impracticable)
- ▷ In most situations, we are more interested in solvability of unification problems than in the unifiers themselves.
- ▷ More liberal treatment of flex/flex pairs.
- ▷ **Observation 11.35** *flex/flex-pairs $F\overline{\mathbf{U}^n} =^? G\overline{\mathbf{V}^m}$ are always (trivially) solvable by $[\lambda\overline{X^n}.H/F], [\lambda\overline{Y^m}.H/G]$, where H is a new variable*
- ▷ Idea: consider flex/flex-pairs as pre-solved.
- ▷ **Definition 11.36 (Pre-Unification)** For given terms $\mathbf{A}, \mathbf{B} \in wff_\alpha(\Sigma, \mathcal{V}_\mathcal{T})$ find a substitution σ, such that $\sigma(\mathbf{A}) =^p_{\beta\eta} \sigma(\mathbf{B})$, where $=^p_{\beta\eta}$ is the equality theory that is induced by $\beta\eta$ and $F\overline{\mathbf{U}} = G\overline{\mathbf{V}}$.
- ▷ **Lemma 11.37** *A higher-order unification problem is unifiable, iff it is pre-unifiable.*

The higher-order pre-unification algorithm can be obtained from $\mathcal{HOU}$ by simply omitting the offending $\mathcal{HOU}$:ff rule.

Pre-Unification Algorithm $\mathcal{HOPU}$

- ▷ **Definition 11.38** A unification problem is a pre-solved form, iff all of its pairs are solved or flex/flex
- ▷ **Lemma 11.39** *If $\mathcal{E}$ is solved and $\mathcal{P}$ flex/flex, then σ_σ is a most general unifier of a pre-solved form $\mathcal{E} \wedge \mathcal{P}$.*
- ▷ Restrict all $\mathcal{HOU}$ rule so that they cannot be applied to pre-solved pairs.
- ▷ In particular, remove $\mathcal{HOU}$:ff!
- ▷ $\mathcal{HOPU}$ only consists of $\mathcal{SIM}$ and $\mathcal{HOU}$:fr.
- ▷ **Theorem 11.40** *$\mathcal{HOPU}$ is a correct and complete pre-unification algorithm*
- ▷ Proof Sketch: with exactly the same methods as higher-order unification □
- ▷ **Theorem 11.41** *Higher-order pre-unification is infinitary, i.e. a unification problem can have infinitely many unifiers.* *(Huet 76' [Hue76])*

▷ **Example 11.42** $Y(\lambda X_\iota.X)a =^? a$, where a is a constant of type ι and Y a variable of type $(\iota \to \iota) \to \iota \to \iota$ has the most general unifiers $\lambda sz.s^n z$ and $\lambda sz.s^n a$, which are mutually incomparable and thus most general.

SOME RIGHTS RESERVED ©: Michael Kohlhase 327 JACOBS UNIVERSITY

11.2.5 Applications of Higher-Order Unification

Application of HOL in NL Semantics: Ellipsis

▷ **Example 11.43** *John loves his wife. George does too*

- ▷ $\mathsf{love}(\mathsf{john}, \mathsf{wife\_of}(\mathsf{john})) \wedge Q(\mathsf{george})$
- ▷ "*George* has property some Q, which we still have to determine"

Idea: If *John* has property Q, then it is that he *loves his wife*.

▷ Equation: $Q(\mathsf{john}) =_{\alpha\beta\eta} \mathsf{love}(\mathsf{john}, \mathsf{wife\_of}(\mathsf{john}))$

▷ Solutions (computed by HOU):

- ▷ $Q = \lambda z.\mathsf{love}(z, \mathsf{wife\_of}(z))$ and $Q = \lambda z.\mathsf{love}(z, \mathsf{wife\_of}(\mathsf{john}))$
- * $Q = \lambda z.\mathsf{love}(\mathsf{john}, \mathsf{wife\_of}(z))$ and $Q = \lambda z.\mathsf{love}(\mathsf{john}, \mathsf{wife\_of}(\mathsf{john}))$

▷ Readings: *George loves his own wife*. and *George loves Johns wife*.

SOME RIGHTS RESERVED ©: Michael Kohlhase 328 JACOBS UNIVERSITY

11.3 Linguistic Applications of Higher-Order Unification

George does too (HOU)

$$Q(j) =^? l(j, w(j))$$

- $Q = \lambda X.X$: $j =^? l(j, w(j))$
- $Q = \lambda X.l(H(X), K(X))$: $l(H(j), K(j)) =^? l(j, w(j))$
  - $H(j) =^? j \wedge K(j) =^? w(j)$
    - $H = \lambda X.X$: $j =^? j \wedge K(j) =^? w(j)$
      - $K = \lambda X.X$
      - $K = \lambda X.w(K'(X))$: $j =^? j \wedge K'(j) =^? j$
        - $K' = \lambda X.X$: $j \doteq j; j \doteq j$ ⋮ $Q = \lambda X.l(X, w(X))$
        - $K' = \lambda X.j$: $j \doteq j; j \doteq j$ ⋮ $\lambda X.l(X, w(j))$
    - $H = \lambda X.j$: $j =^? j \wedge K(j) =^? w(j)$
      - $K = \lambda X.w(K'(X))$: $j =^? j \wedge K'(j) =^? j$
        - $K' = \lambda X.X$: $j \doteq j; j \doteq j$ ⋮ $\lambda X.l(j, w(X))$
        - $K' = \lambda X.j$: $j \doteq j; j \doteq j$ ⋮ $\lambda X.l(j, w(j))$
      - $K = \lambda X.X$

SOME RIGHTS RESERVED ©: Michael Kohlhase 329 JACOBS UNIVERSITY

Primary Occurrence Restriction

- Problem: HOU over-generates
- Idea: [Dalrymple, Shieber, Pereira]
  Given a labeling of occurrences as either primary or secondary, the POR excludes of the set of linguistically valid solutions, any solution which contains a primary occurrence.
- A primary occurrence is an occurrence that is directly associated with a source parallel element.
- a source parallel element is an element of the source (i.e. antecedent) clause which has a parallel counterpart in the target (i.e. elliptic) clause.
- **Example 11.44** ▹ $\mathsf{love}(\underline{\mathsf{john}}, \mathsf{wife\_of}(\mathsf{john})) = Q(\mathsf{george})$
  - $Q = \lambda x.\mathsf{love}(x, \mathsf{wife\_of}(\mathsf{john}))$
  - $Q = \lambda x.\mathsf{love}(\underline{\mathsf{john}}, \mathsf{wife\_of}(\mathsf{john}))$
- Use the colored λ-calculus for general theory

SOME RIGHTS RESERVED ©: Michael Kohlhase 330 JACOBS UNIVERSITY

Colored λ-calculus[HK00]

- Developed for inductive theorem proving (Rippling with Meta-Variables)
- Symbol occurrences can be annotated with color s
  (variables α, β, γ, ... and constants a, b,...)
- Bound variables are uncolored ($\beta\eta$-conversion just as usual)
- Well-colored substitutions σ
  - Map colored variables X_X to colored formulae.
  - If a and b are different colors, then $|\sigma(X_X)| = |\sigma(X_X)|$: equal color erasures. (Consistency)
  - All color annotations on $\sigma(X_X)$ have to be compatible with those for c. (Monochromacity)

SOME RIGHTS RESERVED ©: Michael Kohlhase 331 JACOBS UNIVERSITY

Colored HO-Unification

- HOCU has only two differences wrt. general HOU

$$\frac{f_f(t^1,\ldots,t^n) =^? f_f(s^1,\ldots,s^n)}{\mathsf{a} =^? \mathsf{b} \wedge t^1 =^? s^1 \wedge \ldots \wedge t^n =^? s^n} \qquad \frac{X_X =^? \mathbf{A} \wedge \mathcal{E}}{X =^? \mathbf{A} \wedge [\mathbf{A}/X](\mathcal{E})}$$

▷ Decomposition must consider colors

▷ Elimination ensures Monochromicity and Consistency

- ▷ $(X =^? \mathbf{A}) := (X_X =^? \mathbf{A_A}) \wedge \ldots \wedge (X_X =^? \mathbf{A_A})$
- ▷ $[\mathbf{A}/X] := [\mathbf{A_A}/X_X], \ldots, [\mathbf{A_A}/X_X]$ propagates color information

SOME RIGHTS RESERVED ©: Michael Kohlhase

George does too (HOCU)

$$Q(j) =^? l(j, w(j))$$

- $\xrightarrow{Q=\lambda X.X}$ $j =^? l(j, w(j))$
- $\xrightarrow{Q=\lambda X.l(H(X),K(X))}$ $l(H(j), K(j)) =^? l(j, w(j))$
  - $H(j) =^? j \wedge K(j) =^? w(j)$
    - $\xrightarrow{H=\lambda X.X}$ $j =^? j \wedge K(j) =^? w(j)$
      - $\xrightarrow{K=\lambda X.X}$
      - $\xrightarrow{K=\lambda X.w(K'(X))}$ $j =^? j \wedge K'(j) =^? j$
        - $\xrightarrow{K'=\lambda X.X}$ $j \doteq j; j \doteq j$ ⋮ $Q = \lambda X.l(X, w(X))$
        - $\xrightarrow{K'=\lambda X.j}$ $j \doteq j; j \doteq j$ ⋮ $\lambda X.l(X, w(j))$
    - $\xrightarrow{H=\lambda X.j}$ $j =^? j \wedge K(j) =^? w(j)$
      - $\xrightarrow{K=\lambda X.w(K'(X))}$ $j =^? j \wedge K'(j) =^? j$
        - $\xrightarrow{K'=\lambda X.X}$ $j \doteq j; j \doteq j$ ⋮ $\lambda X.l(j, w(X))$
        - $\xrightarrow{K'=\lambda X.j}$ $j \doteq j; j \doteq j$ ⋮ $\lambda X.l(j, w(j))$
      - $\xrightarrow{K=\lambda X.X}$

SOME RIGHTS RESERVED ©: Michael Kohlhase

The Original Motivation: First-Order Rippling

Example 11.45 Proving: $\forall x, y : list.\mathsf{rev}(\mathsf{app}(\mathsf{rev}(x), y)) = \mathsf{app}(\mathsf{rev}(y), x)$

$$\mathsf{rev}(\mathsf{app}(\mathsf{rev}(\mathsf{cons}(h, x)), y)) = \mathsf{app}(\mathsf{rev}(y), \mathsf{cons}(h, x))$$
$$\downarrow$$
$$\mathsf{rev}(\mathsf{app}(\mathsf{app}(\mathsf{rev}(x), \mathsf{cons}(h, \mathsf{nil}))y,)) = \mathsf{app}(\mathsf{rev}(y), \mathsf{cons}(h, x))$$
$$\mathsf{app}_\alpha(X_X, \mathsf{cons}(Y, Z_Z)) = \mathsf{app}\alpha(F_1(X_X, Y, Z), Z_Z) \downarrow$$
$$\mathsf{rev}(\mathsf{app}(\mathsf{app}(\mathsf{rev}(x), \mathsf{cons}(h, \mathsf{nil})), y)) = \mathsf{app}(F_1(\mathsf{rev}(y), h, x), x)$$
$$\mathsf{app}(\mathsf{rev}_\alpha(Y_Y), \mathsf{cons}(X, \mathsf{nil})) = \mathsf{rev}_\alpha(\mathsf{cons}(X, Y_Y)) \downarrow$$
$$\mathsf{rev}(\mathsf{app}(\mathsf{app}(\mathsf{rev}(x), \mathsf{cons}(h, \mathsf{nil})), y)) = \mathsf{app}(\mathsf{rev}(\mathsf{cons}(h, y)), x)$$
$$\downarrow$$
$$\mathsf{rev}(\mathsf{app}(\mathsf{rev}(x), \mathsf{cons}(h, y),)) = \mathsf{app}(\mathsf{rev}(\mathsf{cons}(h, y)), x)$$

SOME RIGHTS RESERVED ©: Michael Kohlhase

The Higher-Order Case: Schematic Rippling

Example 11.46 (Synthesizing Induction Orderings) Example: $\forall x.\exists y.f(g(y)) \leq x$
Induction Step: $\forall x.\exists y.f(g(y)) \leq x$ to $\exists y.f(g(y)) \leq F(x)$

$$\begin{array}{rcll} f(g(y)) & \leq & F(x) & \\ f(s(g(y'))) & \leq & F(x) & \\ s(s(f(g(y')))) & \leq & F(x) & \\ s(s(f(g(y')))) & \leq & s(s(x)) & F \leftarrow \lambda X.s(s(X)) \\ f(g(y')) & \leq & x & \end{array}$$

A Unification Problem

▷ **Example 11.47 (A Unification Problem)**

$$F(\mathsf{rev}(y), h, x) =^? (rev_\alpha(Y_\beta), \mathsf{cons}(X, \mathsf{nil}))$$
$$\downarrow [\lambda UVW.\mathsf{app}(H(U,V,W), K(U,V,W))/F]$$
$$H(\mathsf{rev}(u), h, v) =^? \mathsf{rev}_\alpha(Y_Y) \wedge K(\mathsf{rev}(u), h, v) =^? \mathsf{cons}(X, \mathsf{nil})$$
$$\downarrow [\lambda UVW.\mathsf{cons}(M(U,V,W), N(U,V,W))/K], [\lambda UVW.U/H]$$
$$\mathsf{rev}(u) =^? \mathsf{rev}_\alpha(Y_Y) \wedge \mathsf{cons}(M(\mathsf{rev}(u), h, v), N(\mathsf{rev}(u), h, v)) =^? \mathsf{cons}(X, \mathsf{nil})$$
$$\downarrow$$
$$\alpha =^? \blacksquare \wedge u =^? Y_Y \wedge X =^? M(\mathsf{rev}(u), h, v) \wedge N(\mathsf{rev}(u), h, v) =^? nil$$
$$\downarrow$$
$$h =^? h \wedge \mathsf{nil} =^? \mathsf{nil}$$

Result: $[\lambda UVW.\mathsf{app}(U, \mathsf{cons}(V, \mathsf{nil}))/F], [u/Y_Y], [h/X], [\blacksquare/\alpha]$

Linguistic Application: Focus/Ground Structures

▷ **Example 11.48** *John only likes MARY.*

▷ Analysis: $\mathsf{like}(\mathsf{john}, \mathsf{mary}) \wedge (\forall x.G(x) \implies x = \mathsf{mary})$.

▷ Equation: $\mathsf{like}(\mathsf{john}, \mathsf{mary}) =_{\alpha\beta\eta} G(\mathsf{mary})$.

- ▷ Variable G for (back)ground (Focus is prosodically marked)

▷ Solution: $G = \lambda z.\mathsf{like}(\mathsf{john}, z)$

▷ Semantics: $\mathsf{like}(\mathsf{john}, \mathsf{mary}) \wedge (\forall x.\mathsf{like}(\mathsf{john}, x) \Rightarrow x = \mathsf{mary})$.

▷ Linguistic Coverage: Prosodically unmarked focus, sentences with multiple focus operators
[Gardent & Kohlhase'96]

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

Isn't HOCU just a notational variant of DSP's POR?

- HOCU has a *formal*, well–understood foundation which permits a clear assessment of its mathematical and computational properties;
- It is a *general* theory of colors:
- Other Constraints
  - POR for focus
  - Second Occurrence Expressions
  - Weak Crossover Constraints
- Multiple constraints and their interaction are easily handled
  - Use feature constraints as colors

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

Interaction of Constraints via Extended Colors

- *Jon likes MARY and Peter does too*
  - Ellipsis: $l(j_j, s_s) = R_R(j_j)$
  - Focus: $R_R(p) = G_G(F_F)$
  - $\neg$pe forbids only pe $\neg$pf forbids only pf

Derivation:

-
  - Solution $R_R = \lambda x.l(x, s_s)$ to the Ellipsis equation
  - yields Focus equation $l(p, s_s) = G_G(F_F)$

Solution: $G_G = \lambda x.l(p_p, x)$ $F_F = m_m$

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

Featuring even more colors for Interaction

- *John$_1$'s mum loves him$_1$. Peter's mum does too.*
- Two readings:
  - *Peter's mum loves Peter* (sloppy)
  - *Peter's mum loves Jon* (strict)
- Parallelism equations

$$\begin{aligned} C(j) &= l(m(j), j) \\ C(p) &= R(m(p)) \end{aligned}$$

▷ Two solution for the first equation:

$$C = \lambda Z.l(m(Z), j) \text{ (strict)} \quad \text{and} \quad C = \lambda Z.l(m(Z), Z) \text{ (sloppy)}$$

▷ Two versions of the second equation

$$\begin{aligned} l(m(p), j) &= R(m(p)) \\ l(m(p), p) &= R(m(p)) \end{aligned}$$

▷ $R = \lambda Z.l(Z, j)$ solves the first equation (strict reading)

▷ the second equation is unsolvable $R = \lambda Z.l(Z, p)$ is not well-colored.

▷ Idea: Need additional constraint:
VPE may not contain (any part of) it's subject

▷ Need more dimensions of colors to model the interaction

▷ Idea: Extend supply of colors to feature terms.

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

John$_1$'s mum loves him$_1$. Peter's mum does too.

▷ Parallelism Constraints

$$\begin{aligned} C_C(j_j) &= l(m_m(j_j), j) \\ C_C(p_p) &= R_R(m_m(p_p)) \end{aligned}$$

▷ Resolving the first equation yields two possible values for C_C:

$$\lambda z.l(m_m(z), j) \quad \text{and} \quad \lambda z.l(m_m(z), z)$$

▷ Two versions of the second equation

$$\begin{aligned} l(m_m(p_p), j) &= R_R(m_m(p_p)) \\ l(m_m(p_p), p_p) &= R_R(m_m(p_p)) \end{aligned}$$

▷ Two solutions for the ellipsis (for R_R)

$$\begin{array}{ll} \{R_R \leftarrow \lambda z.l(z, j)\} & \text{Strict Reading} \\ \{R_R \leftarrow \lambda z.l(z, p_p)\} & \text{Sloppy Reading} \end{array}$$

▷ Need *dynamic constraints*

- ▷ resulting from the unification of several independent constraints
- ▷ VPE subject is $[e\ \ +]$, while part of is a parallel element ($[p\ \ +]$).
- ▷ Various linguistic modules interact in creating complex constraints

SOME RIGHTS RESERVED ©: Michael Kohlhase JACOBS UNIVERSITY

Computation of Parallelism (The General Case)

- We need inferences to discover discourse relations
- General Conditions [Hobbs 1990]

Relation	Requirements	Particle
Parallel	$a_i \sim b_i$, $p \simeq q$	*and*
Contrast	$a_i \sim b_i$, $p \supset \neg q$ or $\neg p \supset q$ a_i, b_i contrastive	*but*

Source semantics $p(\vec{a})$, Target semantics $q(\vec{b})$

- $a \sim b$, iff $\forall p.p(a) \Rightarrow (\exists q \simeq p.q(b))$ $\qquad$ $p \simeq q$, iff $\forall a.p(a) \Rightarrow (\exists b \sim a.q(b))$
- Need theorem proving methods for general case.
- Idea: use only special properties (Sorts from the Taxonomy)

SOME RIGHTS RESERVED ©: Michael Kohlhase 342 JACOBS UNIVERSITY

11.4 Sorted Higher-Order Unification

Sorted λ-Calculus

- higher-order automated theorem provers are relatively weak
- transfer first-order theorem proving technology to higher-order
- sorts are a particularly efficient refinement
  - separation of sorts and types
  - functional base sorts
  - term declarations as very general mechanism for declaring sort information

SOME RIGHTS RESERVED ©: Michael Kohlhase 343 JACOBS UNIVERSITY

Sorted Unification:

- Example: Signature Σ with

$$\begin{array}{l}[+ :: \mathbb{N} \to \mathbb{N} \to \mathbb{N}] \\ [+ :: \mathbb{E} \to \mathbb{E} \to \mathbb{E}] \\ [+ :: \mathbb{O} \to \mathbb{O} \to \mathbb{E}] \\ [\lambda X.+XX :: \mathbb{N} \to \mathbb{E}]\end{array}$$

- general bindings

$$\mathbf{G}^{+}_{\mathbb{E}}(\Sigma) = \left\{ \begin{array}{l} +Z_{\mathbb{E}}W_{\mathbb{E}}, \\ +Z_{\mathbb{O}}W_{\mathbb{O}}, \\ +Z_{\mathbb{N}}Z_{\mathbb{N}} \end{array} \right\}$$

SOME RIGHTS RESERVED ©: Michael Kohlhase 344 JACOBS UNIVERSITY

Example (Elementary Calculus)

▷ Sorts

- ▷ $\mathbb{R}^+$, $\mathbb{R}$ of type ι: (non-negative) real numbers
- ▷ $\mathbb{M}$, $\mathbb{P}$ of type $\iota \to \iota$: monomials, polynomials
- ▷ $\mathbb{M}$, $\mathbb{P}$ of type $\iota \to \iota$: differentiable and continuous functions

▷ Signature Σ

$$\begin{array}{l}[+ :: \mathbb{R} \to \mathbb{R} \to \mathbb{R}], [* :: \mathbb{R} \to \mathbb{R} \to \mathbb{R}], [\lambda X.{*}XX :: \mathbb{R} \to \mathbb{R}^+],\\ [\mathbb{R}^+ \sqsubset \mathbb{R}], [\mathbb{M} \sqsubset \mathbb{P}], [\mathbb{P} \sqsubset \mathbb{M}], [\mathbb{M} \sqsubset \mathbb{P}]\\ [\lambda X.X :: \mathbb{M}], [\lambda XY.Y :: \mathbb{R} \to \mathbb{M}],\\ [\lambda FGX.{*}(FX)(GX) :: \mathbb{M} \to \mathbb{M} \to \mathbb{M}],\\ [\lambda FGX.{+}(FX)(GX) :: \mathbb{M} \to \mathbb{M} \to \mathbb{P}],\\ [\partial :: \mathbb{M} \to \mathbb{P}], [\partial :: \mathbb{P} \to \mathbb{P}], [\partial :: \mathbb{M} \to \mathbb{M}].\end{array}$$

SOME RIGHTS RESERVED ©: Michael Kohlhase 345 JACOBS UNIVERSITY

Example (continued)

▷ Question: Are there non-negative, differentiable functions?

▷ Unification Problem: $G_{\mathbb{R}\to\mathbb{R}^+} =^? F_{\mathbb{M}}$

- ▷ guess $G_{\mathbb{R}\to\mathbb{R}^+}$ to be $(\lambda X.{*}(H^1_{\mathbb{R}\to\mathbb{R}}X)(H^1X))$:

$$F_{\mathbb{M}} =^? \lambda X.{*}(H^1_{\mathbb{R}\to\mathbb{R}}X)(H^1X)$$

- ▷ imitate with $F_{\mathbb{M}}$ as $\lambda X.{*}(H^2_{\mathbb{M}}X)(H^3_{\mathbb{M}}X)$:

$$H^1_{\mathbb{R}\to\mathbb{R}}Z^0 =^? H^2_{\mathbb{M}}Z^0 \wedge H^1_{\mathbb{R}\to\mathbb{R}}Z^0 =^? H^3_{\mathbb{M}}Z^0$$

- ▷ weaken $H^1_{\mathbb{R}\to\mathbb{R}}$ to $H^4_{\mathbb{M}}$

$$H^4_{\mathbb{M}}Z^0 =^? H^2_{\mathbb{M}}Z^0 \wedge H^4_{\mathbb{M}}Z^0 =^? H^3_{\mathbb{M}}Z^0$$

- ▷ solvable with with $H^4 = H^3 = H^2$

▷ Answer: $F = G = \lambda X_{\mathbb{R}}.{*}(H^4_{\mathbb{M}}X)(H^4_{\mathbb{M}}X))$ (even degree monomial)

SOME RIGHTS RESERVED ©: Michael Kohlhase 346 JACOBS UNIVERSITY

Abductive Reconstruction of Parallelism (ARP)

▷ Mix Parallelism with HOCU

▷ Example (Gapping): *John likes Golf and Mary too.*

▷ Representation love(john, golf) $\wedge$ R(mary)

▷ Equation $\mathsf{love}(\mathsf{john}_{\mathsf{john}}, \mathsf{golf}_{\mathsf{golf}}) =^s R^{\neg\mathsf{pe}}_{(\mathbb{W}\mathtt{oman}\to o)}(\mathsf{mary}_{\mathsf{mary}})$

- R for the missing semantics (of Sort $\mathbb{W}\mathtt{oman} \to o$ and not primary for ellipsis)

- Number Restriction Constraint
  - *Jon* and *golf* might be parallel to *Mary*, but at most one of them can.
  - color variable A: if *Jon* is pe then *golf* isn't, and vice versa
  - Generalizes DSP's Primary Occurrence Restriction (POR)

SOME RIGHTS RESERVED ©: Michael Kohlhase

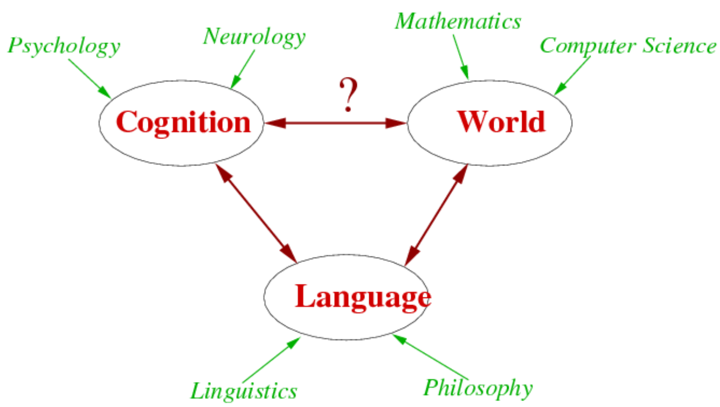
- Initial Equation: $\mathsf{love}(\mathsf{john}_{\mathsf{john}}, \mathsf{golf}_{\mathsf{golf}}) =^? R^{\neg \mathsf{pe}}_{\mathbb{W}\mathtt{oman} \to o}(\mathsf{mary}_{\mathsf{mary}})$
  - imitate $R^{\neg \mathsf{pe}}_{\mathbb{W}\mathtt{oman} \to o}$ with $\lambda Z.\mathsf{love}(H_H Z, K_K Z)$
  - H, K new variables of sort $\mathbb{W}\mathtt{oman} \to \mathbb{H}\mathtt{uman}$
- $\mathsf{love}(\mathsf{john}_{\mathsf{john}}, \mathsf{golf}_{\mathsf{golf}}) =^? \mathsf{love}(H_H(\mathsf{mary}_{\mathsf{mary}}), K_K \mathsf{mary}_{\mathsf{mary}})$
- $H_H \mathsf{mary}_{\mathsf{mary}} =^? \mathsf{john}_{\mathsf{john}} \wedge K_K \mathsf{mary}_{\mathsf{mary}} =^? \mathsf{golf}_{\mathsf{golf}}$
- Two possible continuations:

project $H = \lambda Z.Z$ (so $\mathsf{A} =^? \mathsf{pe}$)	project $K = \lambda Z.Z$ (so $\neg\mathsf{A} =^? \mathsf{pe}$)
imitate $K = \lambda Z.\mathsf{golf}_{\mathsf{golf}}$	imitate $H = \lambda Z.\mathsf{john}_{\mathsf{john}}$
then $\boxed{\begin{array}{l}\mathsf{mary}_{\mathsf{mary}} =^? \mathsf{john}_{\mathsf{john}} \\ \mathsf{golf}_{\mathsf{golf}} =^? \mathsf{golf}_{\mathsf{golf}}\end{array}}$	then $\boxed{\begin{array}{l}\mathsf{john}_{\mathsf{john}} =^? \mathsf{john}_{\mathsf{john}} \\ \mathsf{mary}_{\mathsf{mary}} =^? \mathsf{golf}_{\mathsf{golf}}\end{array}}$
Mary likes Golf (*preferred*)	John *likes Mary*

SOME RIGHTS RESERVED ©: Michael Kohlhase

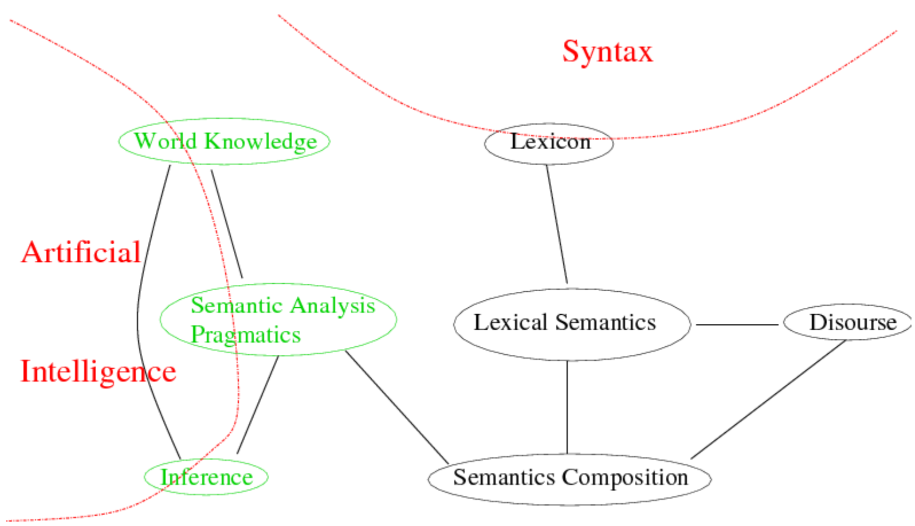
12 Conclusion

NL Semantics as an Intersective Discipline



SOME RIGHTS RESERVED ©: Michael Kohlhase 349 JACOBS UNIVERSITY

A landscape of formal semantics



SOME RIGHTS RESERVED ©: Michael Kohlhase 350 JACOBS UNIVERSITY

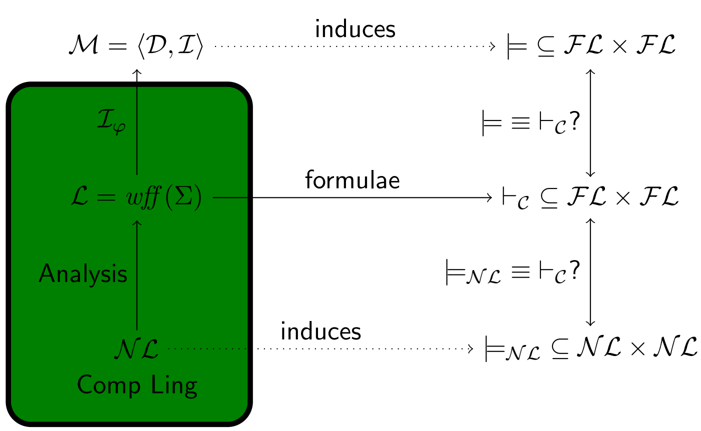
Modeling Natural Language Semantics

- ▷ Problem: Find formal (logic) system for the meaning of natural language
- ▷ History of ideas
  - ▷ Propositional logic [ancient Greeks like Aristotle]
    **Every human is mortal*
  - ▷ First-Order Predicate logic [Frege ≤ 1900]
    **I believe, that my audience already knows this.*
  - ▷ Modal logic [Lewis18, Kripke65]
    **A man sleeps. He snores.* $((\exists X.\mathsf{man}(X) \wedge \mathsf{sleep}(X))) \wedge \mathsf{snore}(X)$
  - ▷ Various dynamic approaches (e.g. DRT, DPL)
    **Most men wear black*
  - ▷ Higher-order Logic, e.g. generalized quantifiers
  - ▷ ...

©: Michael Kohlhase 351

Let us now reconcider the role of all of this for natural language semantics. We have claimed that the goal of the course is to provide you with a set of methods to determine the meaning of natural language. If we look back, all we did was to establish translations from natural languages into formal languages like first-order or higher-order logic (and that is all you will find in most semantics papers and textbooks). Now, we have just tried to convince you that these are actually syntactic entities. So, *where is the semantics?*.

Natural Language Semantics?



©: Michael Kohlhase 352

As we mentioned, the green area is the one generally covered by natural language semantics. In the analysis process, the natural language utterances (viewed here as formulae of a language $\mathcal{NL}$) are translated to a formal language $\mathcal{FL}$ (a set $wff(\Sigma)$ of well-formed formulae). We claim that this is all that is needed to recapture the semantics even it this is not immediately obvious at first: Theoretical Logic gives us the missing pieces.

Since $\mathcal{FL}$ is a formal language of a logical systems, it comes with a notion of model and an interpretation function $\mathcal{I}_\varphi$ that translates $\mathcal{FL}$ formulae into objects of that model. This induces a notion of logical consequence[7] as explained in[32]. It also comes with a calculus $\mathcal{C}$ acting on $\mathcal{FL}$-formulae, which (if we are lucky) is correct and complete (then the mappings in the upper rectangle commute). EdN:32

What we are really interested in in natural language semantics is the truth conditions and natural consequence relations on natural language utterances, which we have denoted by $\models_{\mathcal{NL}}$. If the calculus $\mathcal{C}$ of the logical system $\langle\mathcal{FL}, \mathcal{K}, \models\rangle$ is adequate (it might be a bit presumptious to say sound and complete), then it is a model of the relation $\models_{\mathcal{NL}}$. Given that both rectangles in the diagram commute, then we really have a model for truth-conditions and logical consequence for natural language utterances, if we only specify the analysis mapping (the green part) and the calculus.

Where to from here?

- ▷ We can continue the exploration of semantics in two different ways:
  - ▷ Look around for additional logical systems and see how they can be applied to various linguistic problems. (The logician's approach)
  - ▷ Look around for additional linguistic forms and wonder about their truth conditions, their logical forms, and how to represent them. (The linguist's approach)
- ▷ Here are some possibilities...

SOME RIGHTS RESERVED ©: Michael Kohlhase 353 JACOBS UNIVERSITY

Semantics of Plurals

1. The dogs were barking.
2. Fido and Chester were barking. (What kind of an object do the subject NPs denote?)
3. Fido and Chester were barking. They were hungry.
4. Jane and George came to see me. She was upset. (Sometimes we need to look inside a plural!)
5. Jane and George have two children. (Each? Or together?)
6. Jane and George got married. (To each other? Or to other people?)
7. Jane and George met. (The predicate makes a difference to how we interpret the plural)

SOME RIGHTS RESERVED ©: Michael Kohlhase 354 JACOBS UNIVERSITY

Reciprocals

- ▷ What's required to make these true?

[7] Relations on a set S are subsets of the cartesian product of S, so we use $R \in (S^*)S$ to signify that R is a (n-ary) relation on X.

[32] EdNote: crossref

1. The men all shook hands with one another.
2. The boys are all sitting next to one another on the fence.
3. The students all learn from each other.

Presuppositional expressions

- What are presuppositions?
- What expressions give rise to presuppositions?
- Are all apparent presuppositions really the same thing?
  1. The window in that office is open.
  2. The window in that office isn't open.
  3. George knows that Jane is in town.
  4. George doesn't know that Jane is in town.
  5. It was / wasn't George who upset Jane.
  6. Jane stopped / didn't stop laughing.
  7. George is / isn't late.

Presupposition projection

1. George doesn't know that Jane is in town.
2. Either Jane isn't in town or George doesn't know that she is.
3. If Jane is in town, then George doesn't know that she is.
4. Henry believes that George knows that Jane is in town.

Conditionals

- What are the truth conditions of conditionals?
  1. If Jane goes to the game, George will go.
     - Intuitively, not made true by falsity of the antecedent or truth of consequent independent of antecedent.
  2. If John is late, he must have missed the bus.
- Generally agreed that conditionals are modal in nature. Note presence of modal in consequent of each conditional above.

SOME RIGHTS RESERVED ©: Michael Kohlhase 358 JACOBS UNIVERSITY

Counterfactual conditionals

- And what about these??
  1. If kangaroos didn't have tails, they'd topple over. (David Lewis)
  2. If Woodward and Bernstein hadn't got on the Watergate trail, Nixon might never have been caught.
  3. If Woodward and Bernstein hadn't got on the Watergate trail, Nixon would have been caught by someone else.
- Counterfactuals undoubtedly modal, as their evaluation depends on which alternative world you put yourself in.

SOME RIGHTS RESERVED ©: Michael Kohlhase 359 JACOBS UNIVERSITY

Before and after

- These seem easy. But modality creeps in again...
  1. Jane gave up linguistics after she finished her dissertation. (Did she finish?)
  2. Jane gave up linguistics before she finished her dissertation. (Did she finish? Did she start?)

SOME RIGHTS RESERVED ©: Michael Kohlhase 360 JACOBS UNIVERSITY

References

[And02] Peter B. Andrews. *An Introduction to Mathematical Logic and Type Theory: To Truth Through Proof.* Kluwer Academic Publishers, second edition, 2002.

[BB05] Patrick Blackburn and Johan Bos. *Representation and Inference for Natural Language. A First Course in Computational Semantics.* CSLI, 2005.

[Dav67a] Donald Davidson. The logical form of action sentences. In N. Rescher, editor, *The logic of decision and action*, pages 81–95. Pittsburgh University Press, Pittsburgh, 1967.

[Dav67b] Donald Davidson. Truth and meaning. *Synthese*, 17, 1967.

[DSP91] Mary Dalrymple, Stuart Shieber, and Fernando Pereira. Ellipsis and higher-order unification. *Linguistics & Philosophy*, 14:399–452, 1991.

[Gam91a] L. T. F. Gamut. *Logic, Language and Meaning, Volume I, Introduction to Logic*, volume 1. University of Chicago Press, Chicago, 1991.

[Gam91b] L. T. F. Gamut. *Logic, Language and Meaning, Volume II, Intensional Logic and Logical Grammar*, volume 2. University of Chicago Press, Chicago, 1991.

[GK96] Claire Gardent and Michael Kohlhase. Focus and higher–order unification. In *Proceedings of the 16th International Conference on Computational Linguistics*, pages 268–279, Copenhagen, 1996.

[GKvL96] Claire Gardent, Michael Kohlhase, and Noor van Leusen. Corrections and Higher-Order Unification. In *Proceedings of KONVENS'96*, pages 268–279, Bielefeld, Germany, 1996. De Gruyter.

[Gol81] Warren D. Goldfarb. The undecidability of the second-order unification problem. *Theoretical Computer Science*, 13:225–230, 1981.

[GS90] Jeroen Groenendijk and Martin Stokhof. Dynamic Montague Grammar. In L. Kálmán and L. Pólos, editors, *Papers from the Second Symposium on Logic and Language*, pages 3–48. Akadémiai Kiadó, Budapest, 1990.

[GS91] Jeroen Groenendijk and Martin Stokhof. Dynamic predicate logic. *Linguistics & Philosophy*, 14:39–100, 1991.

[Hei82] Irene Heim. *The Semantics of Definite and Indefinite Noun Phrases*. PhD thesis, University of Massachusetts, 1982.

[HK00] Dieter Hutter and Michael Kohlhase. Managing structural information by higher-order colored unification. *Journal of Automated Reasoning*, 25(2):123–164, 2000.

[Hue76] Gérard P. Huet. *Résolution d'Équations dans des Langages d'ordre 1,2,...,w.* Thèse d'état, Université de Paris VII, 1976.

[Hue80] Gérard Huet. Confluent reductions: Abstract properties and applications to term rewriting systems. *Journal of the ACM (JACM)*, 27(4):797–821, 1980.

[Kam81] Hans Kamp. A theory of truth and semantic representation. In J. Groenendijk, Th. Janssen, and M. Stokhof, editors, *Formal Methods in the Study of Language*, pages 277–322. Mathematisch Centrum Tracts, Amsterdam, Netherlands, 1981.

[KKP96] Michael Kohlhase, Susanna Kuschert, and Manfred Pinkal. A type-theoretic semantics for λ-DRT. In P. Dekker and M. Stokhof, editors, *Proceedings of the 10th Amsterdam Colloquium*, pages 479–498, Amsterdam, 1996. ILLC.

[Koh08] Michael Kohlhase. Using LaTeX as a semantic markup format. *Mathematics in Computer Science*, 2(2):279–304, 2008.

[Koh15] Michael Kohlhase. sTeX: Semantic markup in TeX/LaTeX. Technical report, Comprehensive TeX Archive Network (CTAN), 2015.

[KR93] Hans Kamp and Uwe Reyle. *From Discourse to Logic: Introduction to Model-Theoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory.* Kluwer, Dordrecht, 1993.

[Lew73] David K. Lewis. *Counterfactuals.* Blackwell Publishers, 1973.

[Mat70] Ju. V. Matijasevič. Enumerable sets are diophantine. *Soviet Math. Doklady*, 11:354–358, 1970.

[Mon70] R. Montague. *English as a Formal Language*, chapter Linguaggi nella Societa e nella Tecnica, B. Visentini et al eds, pages 189–224. Edizioni di Communita, Milan, 1970. Reprinted in [Tho74], 188–221.

[Mon74] Richard Montague. The proper treatment of quantification in ordinary English. In R. Thomason, editor, *Formal Philosophy. Selected Papers.* Yale University Press, New Haven, 1974.

[Mus96] Reinhard Muskens. Combining Montague semantics and discourse representation. *Linguistics & Philosophy*, 14:143 – 186, 1996.

[Par90] Terence Parsons. *Events in the Semantics of English: A Study in Subatomic Semantics*, volume 19 of *Current Studies in Linguistics*. MIT Press, 1990.

[Pin96] Manfred Pinkal. Radical underspecification. In P. Dekker and M. Stokhof, editors, *Proceedings of the 10th Amsterdam Colloquium*, pages 587–606, Amsterdam, 1996. ILLC.

[Pul94] Stephen G. Pulman. Higher order unification and the interpretation of focus. Technical Report CRC-049, SRI Cambridge, UK, 1994.

[Sta68] Robert C. Stalnaker. A theory of conditionals. In *Studies in Logical Theory, American Philosophical Quarterly*, pages 98–112. Blackwell Publishers, 1968.

[Sta85] Rick Statman. Logical relations and the typed lambda calculus. *Information and Computation*, 65, 1985.

[Tho74] R. Thomason, editor. *Formal Philosophy: selected Papers of Richard Montague*. Yale University Press, New Haven, CT, 1974.

[vE97] Jan van Eijck. Type logic with states. *Logic Journal of the IGPL*, 5(5), September 1997.

[Ven57] Zeno Vendler. Verbs and times. *Philosophical Review*, 56:143–160, 1957.

[Zee89] Henk Zeevat. A compositional approach to DRT. *Linguistics & Philosophy*, 12:95–131, 1989.

Index