Logic-Based Natural Language Processing
WS 2024/25

Lecture Notes

Prof. Dr. Michael Kohlhase

Knowledge Representation and -Processing
Computer Science, FAU Erlangen-Nürnberg
Michael.Kohlhase@FAU.de

2025-02-06

0.1 Preface

0.1.1 This Document

This document contains the lecture notes for the course "Logic-Based Natural Language Processing" (Logik-Basierte Sprachverarbeitung) held at FAU Erlangen-Nürnberg in the Winter Semesters 2017/18 ff.

This course 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.

As this course is predominantly about modeling natural language and not about the theoretical aspects of the logics themselves, we give the discussion about these as a "suggested readings" section part in ??. This material can safely be skipped (thus it is in the appendix), but contains the missing parts of the "bridge" from logical forms to truth conditions and textual entailment.

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.

Knowledge Representation Experiment: This document is also an experiment in knowledge representation. Under the hood, it uses the sTEX package [Koh08; sTeX], a TEX/LATEX extension for semantic markup, which allows to export the contents into active documents that adapt to the reader and can be instrumented with services based on the explicitly represented meaning of the documents.

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

0.1.2 Acknowledgments

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

ComSem Students: The course is based on a series of courses "Computational Natural Language Semantics" held at Jacobs University Bremen and shares a lot of material with these. The following students have submitted corrections and suggestions to this and earlier versions of the notes: Bastian Laubner, Ceorgi Chulkov, Stefan Anca, Elena Digor, Xu He, and Frederik Schäfer.

LBS Students: The following students have submitted corrections and suggestions to this and earlier versions of the notes: Maximilian Lattka, Frederik Schaefer, Navid Roux.

0.2 Recorded Syllabus

The recorded syllabus – a record the progress of the course in the WS 2024/25 – is in the course page in the ALEA system at `https://courses.voll-ki.fau.de/course-home/lbs`. The table of contents in the LBS lecture notes at `https://kwarc.info/teaching/LBS` indicates the material covered to date in yellow.

For the topics planned for this course, see **??**.

Contents

Elevator Pitch for LBS

- **Mission:** In this course we will
  - explore how to model the *meaning of natural language* via transformation into *logical systems*
  - use *logical inference* there to unravel the missing pieces; the information that is *not linguistically realized*, but is conveyed anyways.
- **Warning:** This course is only for you if you like logic!

  You are going to get lots of it and we are going to introduce our own logics, usually a new facet every week or fortnight.
- **Theory in this course:** We wild do so in an abstract, mathematical fashion, but concrete enough that we could implement all moving parts – NL grammars, semantics construction, and inference systems – in meta-grammatical/logical systems.
- **Practice in PSNLP Project:** We will implement them in the meta-grammatical/logical GLIF system (based on GF, MMT, and ELPI) in the Symbolic NLP Project (5 ECTS; lab work). (see me if you are interested)

Chapter 1

Preliminaries

In this chapter, we want to get all the organizational matters out of the way, so that we can get into the discussion of natural language semantics unencumbered. We will talk about the necessary administrative details, go into how students can get most out of the course, talk about where the various resources provided with the course can be found, and finally introduce the ALeA system, an experimental – using AI methods – learning support system for the LBS course.

1.1 Administrative Ground Rules

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 learning as efficient and painless as possible.

Prerequisites for LBS

- **Content Prerequisites:** The mandatory courses in CS@FAU; Sem 1-4, in particular:
  - course "Grundlagen der Logik in der Informatik" (GLOIN)
  - some of the CS Math courses "Mathematik C1-4" (IngMath1-4) (math tolerance)
  - algorithms and data structures (programming/complexity)
  - AI-1 ("Artificial Intelligence I") (for the logic part)
- **Intuition:** (take them with a kilo of salt)
  - This is what I assume you know! (I have to assume something)
  - In many cases, the dependency of LBS on these is partial and "in spirit".
  - If you have not taken these courses (or do not remember),
    - read up on them as needed! (preferred, do it in a group)
    - We can cover them in class (if there are more of you)
- **The real Prerequisite:** Motivation, interest, curiosity, hard work. (LBS is non-trivial)

▷ You can do this course if you want! (We will help you)

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

Assessment, Grades

▷ **Overall (Module) Grade:**

- ▷ Grade via the exam (Klausur) $\rightsquigarrow$ 100% of the grade.
- ▷ Up to 10% bonus on-top for an exam with $\geq 50\%$ points.($< 50\%$ $\rightsquigarrow$ no bonus)
- ▷ Bonus points $\widehat{=}$ percentage sum of the best 10 prepquizzes divided by 100.

▷ **Exam:** 60 minutes exam conducted in presence on paper ($\sim$ April 1. 2025)

▷ **Retake Exam:** 60 min exam six months later ($\sim$ October 1. 2025)

▷ ⚠ You have to register for exams in `https://campo.fau.de` in the first month of classes.

▷ **Note:** You can de-register from an exam on campo up to three working days before.

Preparedness Quizzes

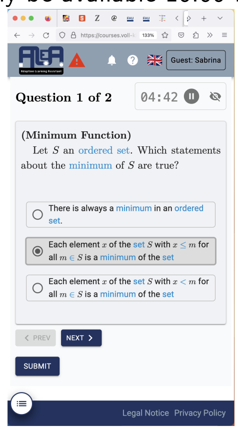
▷ **PrepQuizzes:** Before every lecture we offer a 10 min online quiz – the PrepQuiz – about the material from the previous week. (10:00-10:10; starts in week 3)

▷ **Motivations:** We do this to

- ▷ keep you prepared and working continuously. (primary)
- ▷ bonus points if the exam has $\geq 50\%$ points (potential part of your grade)
- ▷ update the ALeA learner model. (fringe benefit)

▷ The prepquiz will be given in the ALeA system

- `https://courses.voll-ki.fau.de/quiz-dash/lbs`
- You have to be logged into ALeA! (via FAU IDM)
- You can take the prepquiz on your laptop or phone, ...
- ...in the lecture or at home ...
- ...via WLAN or 4G Network. (do not overload)
- Prepquizzes will only be available 10:00-16:10!



Next Week: Pretest

- Next week we will try out the prepquiz infrastructure with a pretest!
  - **Presence**: bring your laptop or cellphone.
  - **Online**: you can and should take the pretest as well.
  - Have a recent firefox or chrome (chrome: younger than March 2023)
  - Make sure that you are logged into ALeA (via FAU IDM; see below)
- **Definition 1.1.1.** A pretest is an assessment for evaluating the preparedness of learners for further studies.
- **Concretely:** This pretest
  - establishes a baseline for the competency expectations in AI-1 and
  - tests the ALeA quiz infrastructure for the prepquizzes.
- Participation in the pretest is optional; it will not influence grades in any way.
- The pretest covers the prerequisites of AI-1 and some of the material that may have been covered in other courses.
- The test will be also used to refine the ALeA learner model, which may make learning experience in ALeA better. (see below)

1.2 Getting Most out of LBS

In this section we will discuss a couple of measures that students may want to consider to get most out of the LBS course.

None of the things discussed in this section – homeworks, tutorials, study groups, and attendance – are mandatory (we cannot force you to do them; we offer them to you as learning opportunities), but most of them are very clearly correlated with success (i.e. passing the exam and getting a good grade), so taking advantage of them may be in your own interest.

LBS Homework Assignments

- **Goal:** Homework assignments reinforce what was taught in lectures.
- **Homework Assignments:** Small individual problem/programming/proof task
  - but take time to solve (at least read them directly ⇝ questions)
- **Didactic Intuition:** Homework assignments give you material to test your understanding and show you how to apply it.
- ⚠ Homeworks give no points, but without trying you are unlikely to pass the exam.
- Homeworks will be mainly peer-graded in the ALeA system.
- **Didactic Motivation:** Through peer grading students are able to see mistakes in their thinking and can correct any problems in future assignments. By grading assignments, students may learn how to complete assignments more accurately and how to improve their future results. (not just us being lazy)

FAU Michael Kohlhase: LBS 6 2025-02-06

It is very well-established experience that without doing the homework assignments (or something similar) on your own, you will not master the concepts, you will not even be able to ask sensible questions, and take very little home from the course. Just sitting in the course and nodding is not enough!

LBS Homework Assignments – Howto

- **Homework Workflow:** in ALeA (see below)
  - Homework assignments will be published on thursdays: see `https://courses.voll-ki.fau.de/hw/lbs`
  - Submission of solutions via the ALeA system in the week after
  - Peer grading/feedback (and master solutions) via answer classes.
- **Quality Control:** TAs and instructors will monitor and supervise peer grading.
- **Experiment:** Can we motivate enough of you to make peer assessment self-sustaining?
  - I am appealing to your sense of community responsibility here ...
  - You should only expect other's to grade your submission if you grade their's (cf. Kant's "Moral Imperative")
  - **Make no mistake**: The grader usually learns at least as much as the gradee.

- **Homework/Tutorial Discipline:**
  - Start early! (many assignments need more than one evening's work)
  - Don't start by sitting at a blank screen (talking & study groups help)
  - Humans will be trying to understand the text/code/math when grading it.
  - Go to the tutorials, discuss with your TA! (they are there for you!)

If you have questions please make sure you discuss them with the instructor, the teaching assistants, or your fellow students. There are three sensible venues for such discussions: online in the lectures, in the tutorials, which we discuss now, or in the course forum – see below. Finally, it is always a very good idea to form study groups with your friends.

Collaboration

- **Definition 1.2.1.** Collaboration (or cooperation) is the process of groups of agents acting together for common, mutual benefit, as opposed to acting in competition for selfish benefit. In a collaboration, every agent contributes to the common goal and benefits from the contributions of others.
- In learning situations, the benefit is "better learning".
- **Observation:** In collaborative learning, the overall result can be significantly better than in competitive learning.
- **Good Practice:** Form study groups. (long- or short-term)
  1. ⚠ those learners who work most, learn most!
  2. ⚠ freeloaders – individuals who only watch – learn very little!
- It is OK to collaborate on homework assignments in LBS! (no bonus points)
- Choose your study group well! (We will (eventually) help via ALeA)

As we said above, almost all of the components of the LBS course are optional. That even applies to attendance. But make no mistake, attendance is important to most of you. Let me explain, . . .

Do I need to attend the LBS Lectures

- Attendance is not mandatory for the LBS course. (official version)
- **Note:** There are two ways of learning: (both are OK, your mileage may vary)
  - Approach B: Read a book/papers (here: lecture notes)
  - Approach I: come to the lectures, be involved, interrupt the instructor whenever you have a question.

  The only advantage of I over B is that books/papers do not answer questions
- Approach S: come to the lectures and sleep does not work!

▷ The closer you get to research, the more we need to discuss!

1.3 Learning Resources for AI-1

But what if you are not in a lecture or tutorial and want to find out more about the LBS topics?

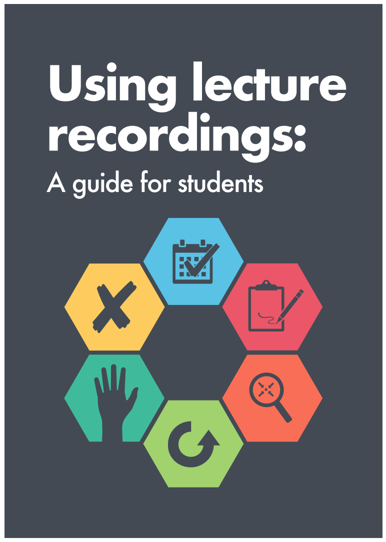
Textbook, Handouts and Information, Forums, Videos

- ▷ **(No) Textbook:** Lecture notes at `http://kwarc.info/teaching/LBS`
  - ▷ I mostly prepare them as we go along (semantically preloaded ⇝ research resource)
  - ▷ Please e-mail me any errors/shortcomings you notice. (improve for group)
- ▷ **For GLIF:** Frederik's Master's Thesis [Sch20]
- ▷ **Classical Semantics/Pragmatics:** (in the FAU Library)
  - ▷ Primary reference for LBS: [CKG09] (in the FAU Library)
  - ▷ also: [HHS07; Bir13; Rie10; ZS13; Sta14; Sae03; Por04; Kea11; Jac83; Cru11; Ari10]
- ▷ **Computational Semantics:** [BB05; EU10]
- ▷ **StudOn Forum:** `https://www.studon.fau.de/crs4625835.html` for
  - ▷ announcements, homeworks (my view on the forum)
  - ▷ questions, discussion among your fellow students (your forum too, use it!)
- ▷ **Course Videos:** at `https://www.fau.tv/course/id/4076.html`

FAU has issued a very insightful guide on using lecture videos. It is a good idea to heed these recommendations, even if they seem annoying at first.

Practical recommendations on Lecture Videos

- ▷ **Excellent Guide:** [Nor+18a] (German version at [Nor+18b])



Attend lectures.

Take notes.

Be specific.

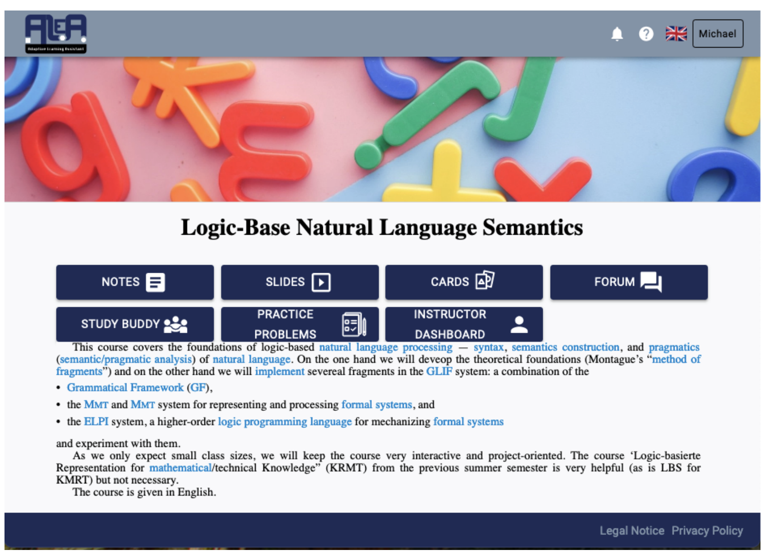
Catch up.

Ask for help.

Don't cut corners.

ALeA in LBS

- ▷ We assume that you already know the ALeA system from AI-1/2



- ▷ Use it for
  - ▷ lecture notes (notes- vs slides-oriented)
  - ▷ flashcards (drill yourself on the LBS jargon/concepts)
  - ▷ course forum (questions, discussions and error reporting)
  - ▷ solving and peer-grading homework assignments
  - ▷ finding study groups (you need not endure LBS alone)
  - ▷ practicing with targeted problems (e.g. from old exams)

- ▷ doing the prepquizzes (before each lecture)

Excursion: We will recap an introduction to ALeA system in??.

Chapter 2

An Introduction to Natural Language Semantics

In this chapter 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 (Natural) Language

- **Definition 2.0.1.** A natural language is any form of spoken or signed means of communication that has evolved naturally in humans through use and repetition without conscious planning or premeditation.
- **In other words:** the language you use all day long, e.g. English, German, ...
- **Why Should we care about natural 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 complex syntax in a matter of a few years.

FAU Michael Kohlhase: LBS 13 2025-02-06

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.

2.1 Natural Language and its Meaning

Before we embark on the journey into understanding the meaning of natural language, let us get an overview over what the concept of "semantics" or "meaning" means in various disciplines.

What is Natural Language Semantics? A Difficult Question!

- **Question:** What is "Natural Language Semantics"?
- **Definition 2.1.1 (Generic Answer).** Semantics is the study of reference, meaning,

or truth.

▷ **Definition 2.1.2.** A sign is anything that communicates a meaning that is not the sign itself to the interpreter of the sign. The meaning can be intentional, as when a word is uttered with a specific meaning, or unintentional, as when a symptom is taken as a sign of a particular medical condition

Meaning is a relationship between signs and the objects they intend, express, or signify.

▷ **Definition 2.1.3.** Reference is a relationship between objects in which one object (the name) designates, or acts as a means by which to refer to – i.e. to connect to or link to – another object (the referent).

▷ **Definition 2.1.4.** Truth is the property of being in accord with reality in a/the mind-independent world. An object ascribed truth is called true, iff it is, and false, if it is not.

▷ **Definition 2.1.5.** For natural language semantics, the signs are usually utterances and names are usually phrases.

▷ That is all very abstract and general, can we make this more concrete?

▷ Different (academic) disciplines find different concretizations.

What is (NL) Semantics? Answers from various Disciplines!

▷ **Observation:** Different (academic) disciplines specialize the notion of semantics (of natural language) in different ways.

▷ **Philosophy:** has a long history of trying to answer it, e.g.

- ▹ Platon ⇝ cave allegory, Aristotle ⇝ Syllogisms.
- ▹ Frege/Russell ⇝ sense vs. referent. (*Michael Kohlhase* vs. *Odysseus*)

▷ **Linguistics/Language Philosophy:** We need semantics e.g. in translation
Der Geist ist willig aber das Fleisch ist schwach! vs.
Der Schnaps ist gut, aber der Braten ist verkocht! (meaning counts)

▷ **Psychology/Cognition:** Semantics $\widehat{=}$ "what is in our brains" (⇝ mental models)

▷ **Mathematics** has driven much of modern logic in the quest for foundations.

- ▹ Logic as "foundation of mathematics" solved as far as possible
- ▹ In daily practice syntax and semantics are not differentiated (much).

▷ **Logic@AI/CS** tries to define meaning and compute with them. (applied semantics)

- ▹ makes syntax explicit in a formal language (formulae, sentences)
- ▹ defines truth/validity by mapping sentences into "world" (interpretation)

▷ gives rules of truth-preserving reasoning (inference)

FAU Michael Kohlhase: LBS 15 2025-02-06

A good probe into the issues involved in natural language understanding is to look at translations between natural language utterances – a task that arguably involves understanding the utterances first.

Meaning of Natural Language; e.g. Machine Translation

- ▷ **Idea:** Machine translation is very simple! (we have good lexica)
- ▷ **Example 2.1.6.** *Peter liebt Maria.* $\rightsquigarrow$ *Peter loves Mary.*
- ▷ ⚠ this only works for simple examples!
- ▷ **Example 2.1.7.** *Wirf der Kuh das Heu über den Zaun.* $\not\rightsquigarrow$ *Throw the cow the hay over the fence.* (differing grammar; Google Translate)
- ▷ **Example 2.1.8.** ⚠ Grammar is not the only problem
  - ▷ *Der Geist ist willig, aber das Fleisch ist schwach!*
  - ▷ *Der Schnaps ist gut, aber der Braten ist verkocht!*
- ▷ **Observation 2.1.9.** *We have to understand the meaning for high-quality translation!*

FAU Michael Kohlhase: LBS 16 2025-02-06

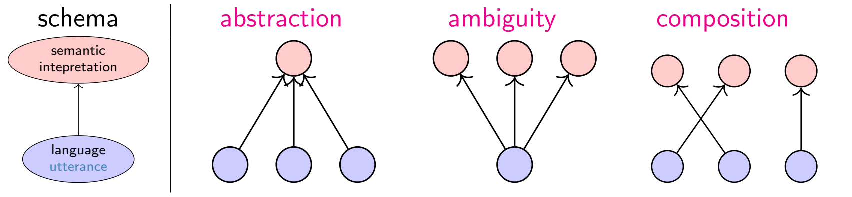
If it is indeed the meaning of natural language, we should look further into how the form of the utterances and their meaning interact.

Language and Information

- ▷ **Observation:** Humans use words (sentences, texts) in natural languages to represent and communicate information.
- ▷ **But:** What really counts is not the words themselves, but the meaning information they carry.
- ▷ **Example 2.1.10 (Word Meaning).**

Newspaper $\rightsquigarrow$ 


- ▷ For questions/answers, it would be very useful to find out what words (sentences/texts) mean.
- ▷ **Definition 2.1.11.** Interpretation of natural language utterances: three problems

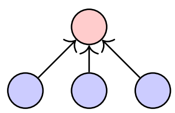


Michael Kohlhase: LBS 17 2025-02-06

Let us support the last claim a couple of initial examples. We will come back to these phenomena again and again over the course of the course and study them in detail.

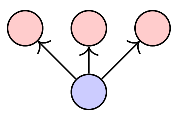
Language and Information (Examples)

▷ **Example 2.1.12 (Abstraction).**



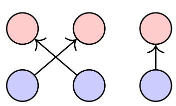
Car and *automobile* have the same meaning.

▷ **Example 2.1.13 (Ambiguity).**



A *bank* can be a financial institution or a geographical feature.

▷ **Example 2.1.14 (Composition).**



Every student sleeps $\leadsto \forall x.student(x) \Rightarrow sleep(x)$

Michael Kohlhase: LBS 18 2025-02-06

But there are other phenomena that we need to take into account when compute the meaning of NL utterances.

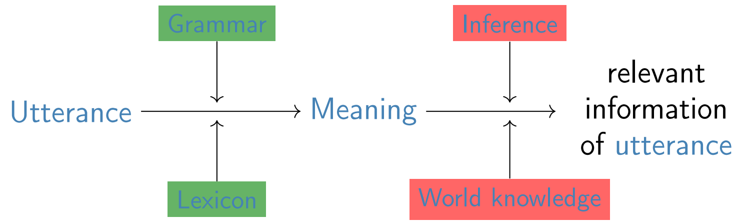
Context Contributes to the Meaning of NL Utterances

▷ **Observation:** Not all information conveyed is linguistically realized in an utterance.

▷ **Example 2.1.15.** *The lecture begins at 11:00 am.* What lecture? Today?

▷ **Definition 2.1.16.** We call a piece i of information linguistically realized in an utterance U, iff, we can trace i to a fragment of U.

▷ **Definition 2.1.17 (Possible Mechanism).** Inferring the missing pieces from the context and world knowledge:



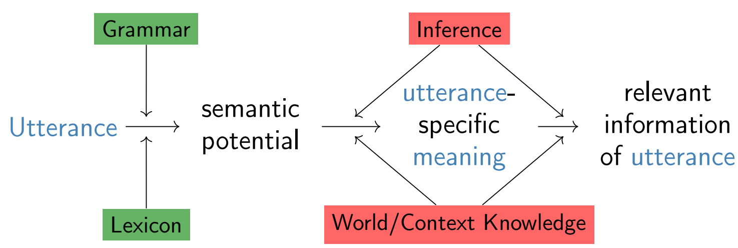
We call this process semantic/pragmatic analysis.

Michael Kohlhase: LBS 19 2025-02-06

We will look at another example, that shows that the situation with semantic/pragmatic analysis is even more complex than we thought. Understanding this is one of the prime objectives of the LBS lecture.

Context Contributes to the Meaning of NL Utterances

- ▷ **Example 2.1.18.** *It starts at eleven.* What starts?
- ▷ Before we can resolve the time, we need to resolve the anaphor *it*.
- ▷ **Possible Mechanism:** More Inference!



⇝ Semantic/pragmatic analysis is quite complex! (prime topic of LBS)

Michael Kohlhase: LBS 20 2025-02-06

?? is also a very good example for the claim Observation 3.10 (Natural Language and its Meaning) in the LBS lecture notes that even for high-quality (machine) translation we need semantics. We end this very high-level introduction with a caveat.

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)

But Semantics works in some cases

- The only thing that currently really helps is a restricted domain:
  - I. e. a restricted vocabulary and world model.
- **Demo:**

  DBPedia `http://dbpedia.org/snorql/`
  Query: Soccer players, who are born in a country with more than 10 million inhabitants, who played as goalkeeper for a club that has a stadium with more than 30.000 seats and the club country is different from the birth country

But Semantics works in some cases

- **Answer:**

  (is computed by DBPedia from a SPARQL query)

```
SELECT distinct ?soccerplayer ?countryOfBirth ?team ?countryOfTeam ?stadiumcapacity
{
?soccerplayer a dbo:SoccerPlayer ;
   dbo:position|dbp:position <http://dbpedia.org/resource/Goalkeeper_(association_football)> ;
   dbo:birthPlace/dbo:country* ?countryOfBirth ;
   #dbo:number 13 ;
   dbo:team ?team .
   ?team dbo:capacity ?stadiumcapacity ; dbo:ground ?countryOfTeam .
   ?countryOfBirth a dbo:Country ; dbo:populationTotal ?population .
   ?countryOfTeam a dbo:Country .
FILTER (?countryOfTeam != ?countryOfBirth)
FILTER (?stadiumcapacity > 30000)
FILTER (?population > 10000000)
} order by ?soccerplayer
```

Results: Browse Go! Reset

SPARQL results:

soccerplayer	countryOfBirth	team	countryOfTeam	stadiumcapacity
:Abdesslam_Benabdellah	:Algeria	:Wydad_Casablanca	:Morocco	67000
:Airton_Moraes_Michellon	:Brazil	:FC_Red_Bull_Salzburg	:Austria	31000
:Alain_Gouaméné	:Ivory_Coast	:Raja_Casablanca	:Morocco	67000
:Allan_McGregor	:United_Kingdom	:Beşiktaş_J.K.	:Turkey	41903
:Anthony_Scribe	:France	:FC_Dinamo_Tbilisi	:Georgia_(country)	54549
:Brahim_Zaari	:Netherlands	:Raja_Casablanca	:Morocco	67000
:Bréiner_Castillo	:Colombia	:Deportivo_Táchira	:Venezuela	38755
:Carlos_Luis_Morales	:Ecuador	:Club_Atlético_Independiente	:Argentina	48069
:Carlos_Navarro_Montoya	:Colombia	:Club_Atlético_Independiente	:Argentina	48069
:Cristián_Muñoz	:Argentina	:Colo-Colo	:Chile	47000
:Daniel_Ferreyra	:Argentina	:FBC_Melgar	:Peru	60000
:David_Bičík	:Czech_Republic	:Karşıyaka_S.K.	:Turkey	51295
:David_Loria	:Kazakhstan	:Karşıyaka_S.K.	:Turkey	51295
:Denys_Boyko	:Ukraine	:Beşiktaş_J.K.	:Turkey	41903
:Eddie_Gustafsson	:United_States	:FC_Red_Bull_Salzburg	:Austria	31000
:Emilian_Dolha	:Romania	:Lech_Poznań	:Poland	43269
:Eusebio_Acasuzo	:Peru	:Club_Bolívar	:Bolivia	42000
:Faryd_Mondragón	:Colombia	:Real_Zaragoza	:Spain	34596
:Faryd_Mondragón	:Colombia	:Club_Atlético_Independiente	:Argentina	48069
:Federico_Vilar	:Argentina	:Club_Atlas	:Mexico	54500
:Fernando_Martinuzzi	:Argentina	:Real_Garcilaso	:Peru	45000
:Fábio_André_da_Silva	:Portugal	:Servette_FC	:Switzerland	30084
:Gerhard_Tremmel	:Germany	:FC_Red_Bull_Salzburg	:Austria	31000
:Gift_Muzadzi	:United_Kingdom	:Lech_Poznań	:Poland	43269
:Günay_Güvenç	:Germany	:Beşiktaş_J.K.	:Turkey	41903
:Hugo_Marques	:Portugal	:C.D._Primeiro_de_Agosto	:Angola	48500
:Héctor_Landazuri	:Colombia	:La_Paz_F.C.	:Bolivia	42000

Michael Kohlhase: LBS 23 2025-02-06

Even if we can get a perfect grasp of the semanticss (aka. meanings) of NL utterances, their structure and context dependency – we will try this in this lecture, but of course fail, since the issues are much too involved and complex for just one lecture – then we still cannot account for all the human mind does with language. But there is hope, for limited and well-understood domains, we can to amazing things. This is what this course tries to show, both in theory as well as in practice.

2.2 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.
- ▷ Information management:
  - ▷ search and classification of documents, (e.g. Google/Bing)
  - ▷ information extraction, question answering. (e.g. http://ask.com)
- ▷ Dialog Systems/Interfaces:

- ▷ information systems: at airport, tele-banking, e-commerce, call centers,
- ▷ dialog interfaces for computers, robots, cars. (e.g. Siri/Alexa)

▷ **Observation:** The earlier technologies largely rely on pattern matching, the latter ones need to compute the meaning of the input utterances, e.g. for database lookups in information systems.

The general context of LBS is natural language processing (NLP), and in particular natural language understanding (NLU). The dual side of NLU: natural language generation (NLG) requires similar foundations, but different techniques is less relevant for the purposes of this course.

What is Natural Language Processing?

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

▷ **Definition 2.2.1.** 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. Most challenges in NLP involve:

- ▷ Natural language understanding (NLU) that is, enabling computers to derive meaning (representations) from human or natural language input.
- ▷ Natural language generation (NLG) which aims at generating natural language or speech from meaning representation.

▷ For communication with/among humans we need both NLU and NLG.

What is the State of the Art In NLU?

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

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

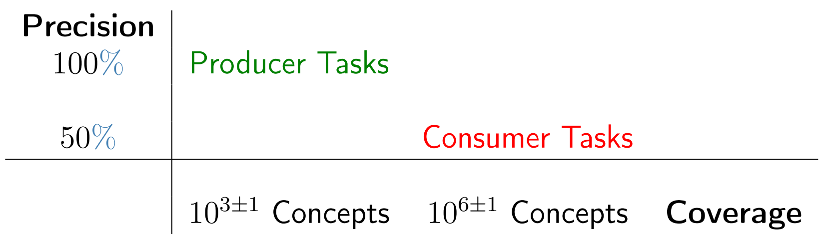
▷ We will cover foundational methods of deep processing in the course and a mixture of deep and shallow ones in the lab.

FAU Michael Kohlhase: LBS 26 2025-02-06

On the last slide we have classified the two main approaches to NLU. In the last 10 years the community has almost entirely concentrated on statistical- and machine-learning based methods, because that has led to applications like google translate, Siri, and the likes. We will now borrow an argument by Aarne Ranta to show that there are (still) interesting applications for knowledge-based methods in NLP, even if they are less visible.

Environmental Niches for both Approaches to NLU

- **Definition 2.2.2.** There are two kinds of applications/tasks in NLU:
  - Consumer tasks: consumer grade applications have tasks that must be fully generic and wide coverage. (e.g. machine translation like Google Translate)
  - Producer tasks: producer grade applications must be high-precision, but can be domain-specific (e.g. multilingual documentation, machinery-control, program verification, medical technology)



after Aarne Ranta [Ran17].

- **Example 2.2.3.** Producing/managing machine manuals in multiple languages across machine variants is a critical producer task for machine tool company.
- A producer domain I am interested in: mathematical/technical documents.

FAU Michael Kohlhase: LBS 27 2025-02-06

An example of a producer task – indeed this is where the name comes from – is the case of a machine tool manufacturer T, which produces digitally programmed machine tools worth multiple million Euro and sells them into dozens of countries. Thus T must also comprehensive machine operation manuals, a non-trivial undertaking, since no two machines are identical and they must be translated into many languages, leading to hundreds of documents. As those manual share a lot of semantic content, their management should be supported by NLP techniques. It is critical that these NLP maintain a high precision, operation errors can easily lead to very costly machine damage and loss of production. On the other hand, the domain of these manuals is quite restricted. A machine tool has a couple of hundred components only that can be described by a comple of thousand attribute only.

Indeed companies like T employ high-precision NLP techniques like the ones we will cover in this course successfully; they are just not so much in the public eye as the consumer tasks.

NLP for NLU: The Waterfall Model

- **Definition 2.2.4 (The NLU Waterfall).** NL understanding is often modeled as a simple linear process: the NLU waterfall consists of five consecutive steps:

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

▷ **Definition 2.2.5.** We call any formalization of an utterance as a logical formula a logical form. A quasi-logical form (QLF) is a representation which can be turned into a logical form by further computation.2

▷ **In this course:** steps 1), 2) and 3).

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.

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 an intuition of what problems we will have to overcome on the way to modeling natural language.

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

▷ **Example 2.3.1.** We study the truth conditions of adjectival complexes:

- *This is a diamond.* ($\models$ *diamond*)
- *This is a blue diamond.* ($\models$ *diamond*, $\models$ *blue*)
- *This is a big diamond.* ($\models$ *diamond*, $\not\models$ *big*)
- *This is a fake diamond.* ($\models \neg$*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*)

Logical analysis vs. conceptual analysis: These examples — mostly borrowed from Davidson:tam67 — 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: The dark side of Meaning

▷ **Definition 2.3.2.** We call an utterance ambiguous, iff it has multiple meanings, which we call readings.

▷ **Example 2.3.3.** All of the following sentences are ambiguous:

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

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:

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

  Sentence 1. entails that George was late; sentence 2. doesn'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

▷ **Example 2.3.4.** *Every man loves a woman.* (Keira Knightley or his mother!)

▷ **Example 2.3.5.** *Every car has a radio.* (only one reading!)

▷ **Example 2.3.6.** *Some student in every course sleeps in every class at least some of the time.* (how many readings?)

▷ **Example 2.3.7.** *The president of the US is having an affair with an intern.* (2002 or 2000?)

▷ **Example 2.3.8.** *Everyone is here.* (who is everyone?)

Observation: If we look at the first sentence, then we see that it has two readings:

1. there is one woman who is loved by every man.
2. for each man there is one woman whom that man 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* 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.
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 $4! = 24$ readings. It should be clear from introspection that we (humans) do not entertain 12 readings when we understand and process this sentence. Our models should account for such effects as well.
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.

We now come to a phenomenon of natural language, that is a paradigmatic challenge for pragmatic analysis: anaphora – the practice of replacing a (complex) reference with a mere pronoun.

More Context: Anaphora – Challenge for Pragmatic Analysis

▷ **Example 2.3.9 (Anaphoric References).**

▹ *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?)

▷ **Definition 2.3.10.** A word or phrase is called anaphoric (or an anaphor), if its interpretation depends upon another phrase in context. In a narrower sense, an anaphor refers to an earlier phrase (its antecedent), while a cataphor to a later one (its postcedent).

Definition 2.3.11. The process of determining the antecedent or postcedent of an anaphoric phrase is called anaphor resolution.

Definition 2.3.12. An anaphoric connection between anaphor and its antecedent or postcedent is called direct, iff it can be understood purely syntactically. An anaphoric connection is called indirect or a bridging reference if additional knowledge is needed.

▷ Anaphora are another example, where natural languages use the inferential capabilities of the hearer/reader to "shorten" utterances.

▷ Anaphora challenge pragmatic analysis, since they can only be resolved from the context using world knowledge.

Anaphora are also interesting for pragmatic analysis, since they introduce (often initially massive amoungs of) ambiguity that needs to be taken care of in the language understanding process.
We now come to another challenge to pragmatic analysis: presuppositions. Instead of just being subject to the context of the readers/hearers like anaphora, they even have the potential to change the context itself or even affect their world knowledge.

Context is Personal and Keeps Changing

▷ **Example 2.3.13.** Consider the following sentences involving definite description:

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

How do the interact with your context and world knowledge?

▷ The interpretation or whether they make sense at all dep

▷ **Note:** Last two examples feed back into the context or even world knowledge:

- ▹ If 4. is uttered by an Africa expert, we add "*Buganda exists and is a monarchy* to our world knowledge
- ▹ We add *Joe Smith is the CEO of Westinghouse to the context/world knowledge* (happens all the time in newpaper articles)

2.4 A Taste of Language Philosophy

We will now discuss some concerns from language philosophy as they pertain to the LBS course. Note that this discussion is only intended to give our discussion on natural language semantics some perspective; in particular, it is in no way a complete introduction to language philosophy, or does the discussion there full justice.
We start out our tour through language philosophy with some examples – as linguists and philosophers often to – to obtain an intuition of the phenomena we want to understand.

What is the Meaning of Natural Language Utterances?

- ▷ **Question:** What is the meaning of the word *chair*?
- ▷ **Answer:** "the set of all chairs" (difficult to delineate, but more or less clear)
- ▷ **Question:** What is the meaning of the word *Michael Kohlhase*?
- ▷ **Answer:** The word refers to an object in the real world: the instructor of LBS.
- ▷ **Alternatively:** The singleton with that object (as for "set of chairs" above)
- ▷ **Question:** What about *Michael Kohlhase sits on a chair*?
- ▷ **Towards an Answer:** We have to combine the two sets, via the meaning of "sits".
- ▷ **Question:** What is the meaning of the word *John F. Kennedy* or *Odysseus*?
- ▷ **Problem:** There are no objects in the real worlds, so the meaning of both is $\emptyset$ and thus equal ☹.

The main intuition we get is that meaning is more complicated than we may have thought in the beginning.

2.4.1 Epistemology: The Philosphy of Science

We start out by looking at the foundations of epistemology, which sets the basis for modern (empirical) science. Our presentation here is modeled on Karl Popper's work on the theory of science. Naturally, our account here is simplified to fit the occasion, see [Pop34; Pop59] for the full story.

Note that like any foundational account of complex concepts like knowledge, belief, rationality, and their justification, we have to base our philosophy on some concepts we take at face value. Here these are natural and formal languages, worlds, situations, etc. which will stay very general in the current foundational setting.

We will later instantiate these by more concrete notions as we go along in the LBS course.

Epistemology – Propositions & Observations

- ▷ **Definition 2.4.1.** Epistemology is the branch of philosophy concerned with studying nature of knowledge, its justification, the rationality of belief, scientific theories and predictions, and various related issues.

- ▷ **Definition 2.4.2.** A proposition is a sentence about the actual world or a class of worlds deemed possible whose meaning can be expressed as being true or false in a specific world.
- ▷ **Definition 2.4.3.** A belief is a proposition φ that an agent a holds true about a class of worlds. This is a characterizing feature of the agent.
- ▷ **Definition 2.4.4 (Knowledge - The JTB Account).** Knowledge is justified, true belief.
- ▷ **Problem:** How can an agent justify a belief to obtain knowledge.
- ▷ **Definition 2.4.5.** Given a world w, the observed value (or just value, i.e. true or false) of a proposition (in w) can be determined by observations, that is an agent, the observer, either observes (experiences) that φ is true in w or conducts a deliberate, systematic experiment that determines φ to be true in w.

The crucial intuition here is that we express belief and possibly knowledge about the world using language. But we can only access truth in the world by observation, a possibly flawed operation. So we will never be able to ascertain the "true belief" part, and need to work all the harder on the "justified" part.

Epistemology – Reproducibility & Phenomena

- ▷ **Problem:** Observations are sometimes unreliable, e.g. observer o perceives φ to be true, while it is false or vice versa.
- ▷ **Idea:** Repeat the observations to raise the probability of getting them right.
- ▷ **Definition 2.4.6.** An observation φ is said to be reproducible, iff φ can observed by different observers in different situations.
- ▷ **Definition 2.4.7.** A phenomenon φ is a proposition that is reproducibly observable to be true in a class of worlds.
- ▷ **Problem:** We would like to verify a phenomenon φ, i.e. observe φ in all worlds, But relevant world classes are too large to make this practically feasible.
- ▷ **Definition 2.4.8.** A world w is a counterexample to a proposition φ, if φ is observably false in w.
- ▷ **Intuition:** The absence of counterexamples is the best we can hope for in general for accepting phenomena.
- ▷ **Intuition:** The phenomena constitute the "world model" of an agent.
- ▷ **Problem:** It is impossible/inefficient (for an agent) to know all phenomena.
- ▷ **Idea:** An agent could retain only a small subset of known propositions, from this all phenomena can be derived.

We will pursue this last idea. The (small) subset of propositions from which the phenomena that are relevant to an agent can be derived will become the beliefs of the agent. An agent will make strive to justify these beliefs to succeed in the world. This is where our notion of knowledge comes from.

Epistemology – Explanations & Hypotheses

- **Definition 2.4.9.** A proposition ψ follows from a proposition φ, iff ψ is true in any world where φ is.
- **Definition 2.4.10.** An explanation of a phenomenon φ is a set Φ of propositions, such that φ follows from Φ.
- **Example 2.4.11.** $\{\varphi\}$ is a (rather useless) explanation for φ.
- **Intuition:** We prefer explanations Φ that explain more than just φ.
- **Observation:** This often coincides with explanations that are in some sense "simpler" or "more elementary" than φ. ($\rightsquigarrow$ Occam's razor)
- **Definition 2.4.12.** A proposition is called falsifiable, iff counterexamples are theoretically possible and the observation of a reproducible series of counterexample is practically feasible.
- **Definition 2.4.13.** A hypothesis is a proposed explanation of a phenomenon that is falsifiable.

We insist that a hypothesis be falsifiable, because we cannot hope to verify it and indeed the absence of counterexamples is the best we can hope for. But if finding counterexamples is hopeless, it is not even worth bothering with a hypothesis.
This gives rise to a very natural strategy of accumulating propositions to represent (what could) knowledge about the world.

Epistemology – Scientific Theories

- **Knowledge Strategy:** Collect hypotheses about the world, drop those with counterexamples and those that can be explained themselves.
- **Definition 2.4.14.** A hypothesis φ can be tested in world/situation w by observing the value of φ in w. If the value is true, then we say that the observation o supports φ or is evidence for φ. If it is false then o falsifies φ.
- **Definition 2.4.15.** A (scientific) theory for a collection Φ of phenomena is a set Θ of hypotheses that
  - has been tested extensively and rigorously without finding counterexamples, and
  - is minimal in the sense that no sub-collection of Θ explains Φ.
- **Definition 2.4.16.** We call any proposition φ that follows from a theory Φ a prediction of Φ.
- **Note:** To falsify a theory Φ, it is sufficient to falsify any prediction. Any observation of a prediction φ of Φ supports Φ.

Indeed the epistemological approach described in this subsection has become the predominant one in modern science. We will introduce both on very simple examples next.

2.4.2 Meaning Theories

If the meaning of natural language is indeed complicated, then we should really admit to that and instead of directly answering the question, allow for multiple opinions and embark on a regime of testing them against reality. We review some concepts from language philosophy towards that end.
We now specialize the general epistemology for natural language the "world" we try to model empirically.

Theories of Meaning

- ▷ **The Central Question:** What is the meaning of natural language?
- ▷ This is difficult to answer definitely, ...
- ▷ **But** we can form meaning theory that make predictions that we can test.
- ▷ **Definition 2.4.17.** A semantic meaning theory assigns semantic contents to expressions of a language.
- ▷ **Definition 2.4.18.** A foundational meaning theory tries to explain why language expressions have the meanings they have; e.g. in terms of mental states of individuals and groups.
- ▷ It is important to keep these two notions apart.
- ▷ We will concentrate on semantic meaning theories in this course.

In [Spe17], an excellent survey on meaning theories, the author likens the difference between semantic and foundational theories of meaning to the differing tasks of an anthropologist trying to fully document the table manner of a distant tribe ($\widehat{=}$ semantic meaning theory) or to explain why the table manners evolve ($\widehat{=}$ foundational meaning theory).
Let us fortify our intuition about semantic meaning theories by showing one that can deal with the meaning of names we started our subsection with.

The Meaning of Singular Terms

- ▷ Let's see a semantic meaning theory in action.
- ▷ **Definition 2.4.19.** A singular term is a phrase that purports to denote or designate a particular individual person, place, or other object.
- ▷ **Example 2.4.20.** *Michael Kohlhase* and *Odysseus* are singular terms.
- ▷ **Definition 2.4.21.** In [Fre92], Gottlob Frege distinguishes between sense (Sinn) and referent (Bedeutung) of singular terms.
- ▷ **Example 2.4.22.** Even though *Odysseus* does not have a referent, it has a very real sense. (but what is a sense?)

▷ **Example 2.4.23.** The ancient greeks knew the planets *Hesperos* (the evening star) and *Phosphoros* (the morning star). These words have different senses, but the – as we now know – same referent: the planet Venus.

▷ **Remark:** Bertrand Russell views singular terms as disguised definite descriptions – *Hesperos* as "the brightest heavenly body that sometimes rises in the evening". Frege's sense can often be conflated with Russell's descriptions.(there can be more than one definite description)

We think of Frege's conceptualization as a semantic meaning theory, since it assigns semantic content – the pair of sense and referent, whatever they might concretely be – to singular terms.

Cresswell's "Most Certain Principle" and Truth Conditions

▷ **Problem:** How can we test meaning theories in practice?

▷ **Definition 2.4.24.** Cresswell's (1982) most certain principle (MCP): [Cre82]

> *I'm going to begin by telling you what I think is the most certain thing I think about meaning. Perhaps it's the only thing. It is this. If we have two sentences A and B, and A is true and B is false, then A and B do not mean the same.*

▷ **Definition 2.4.25.** The truth conditions of a sentence are the conditions of the world under which it is true. These conditions must be such that if all obtain, the sentence is true, and if one doesn't obtain, the sentence is false.

▷ **Observation:** Meaning determines truth conditions and vice versa.

▷ **In Fregean terms** The sense of a sentence (a thought) determines its referent (a truth value).

This principle sounds trivial – and indeed it is, if you think about it – but gives rise to the notion of truth conditions, which form the most important way of finding out about the meaning of sentences: the determinations of truth conditions.

Truth Conditions in Practice

▷ **Idea:** To test/determine the truth conditions of a sentence S in practice, we tell little stories that describe situations/worlds that embed S.

▷ **Example 2.4.26.** Consider the ambiguous sentence from Example 3.27 (Looking at Natural Language) in the LBS lecture notes:
John chased the gangster in the red sports car.
For each of three readings there is story $\widehat{=}$ truth conditions

- ▷ John drives the red sports car and chases the gangster.
- ▷ John chases the gangster who drives the red sports car.
- ▷ John chases the gangster on the back seat of a (very very big) red sports car.

All of these stories correspond to different worlds, so by the MCP there must be at least three readings!

Compositionality

▷ **Definition 2.4.27.** A meaning theory T is compositional, iff the meaning of an expression is a function of the meanings of its parts. We say that T obeys the compositionality principle or simply compositionality if it is.

▷ To compute the meaning of an expression, look up the meanings of the basic expressions forming it and successively compute the meanings of larger parts until a meaning for the whole expression is found.

▷ **Example 2.4.28 (Compositionality at work in arithmetic).** To compute the value of $(x+y)/(z \cdot u)$, look up the values of x, y, z, and u, then compute $x+y$ and $z \cdot u$, and finally compute the value of the whole expression.

▷ Many philosophers and linguists hold that compositionality is at work in ordinary language too.

Why Compositionality is Attractive

▷ Compositionality gives a nice building block for a meaning theory:

▷ **Example 2.4.29.** *[Expressions [are [built [from [words [that [combine [into [[larger [and larger]] subexpressions]]]]]]]]]*

▷ **Consequence:** To compute the meaning of an expression, look up the meanings of its words and successively compute the meanings of larger parts until a meaning for the whole expression is found.

▷ Compositionality explains how people can easily understand sentences they have never heard before, even though there are an infinite number of sentences any given person at any given time has not heard before.

Compositionality and the Congruence Principle

▷ Given reasonable assumptions compositionality entails the

▷ **Definition 2.4.30.** The congruence principle states that whenever A is part of B and A' means just the same as A, replacing A by A' in B will lead to a result that means just the same as B.

▷ **Example 2.4.31.** Consider the following (complex) sentences:

1. *blah blah blah such and such blah blah*
2. *blah blah blah so and so blah blah*

If *such and such* and *so and so* mean the same thing, then 1. and 2. mean the same too.

▷ **Conversely:** if 1. and 2. do not mean the same, then *such and such* and *so and so* do not either.

A Test for Synonymity

▷ Suppose we accept the most certain principle (difference in truth conditions implies difference in meaning) and the congruence principle (replacing words by synonyms results in a synonymous utterance). Then we have a diagnostics for synonymy: Replacing utterances by synonyms preserves truth conditions, or equivalently

▷ **Definition 2.4.32.** The following is called the truth conditional synonymy test:

> If replacing A by B in some sentence C does not preserve truth conditions, then A and B are not synonymous.

▷ We can use this as a test for the question of individuation: when are the meanings of two words the same – when are they synonymous?

▷ **Example 2.4.33 (Unsurprising Results).** The following sentences differ in truth conditions.

1. *The cat is on the mat.*
2. *The dog is on the mat.*

Hence *cat* and *dog* are not synonymous. The converse holds for

1. *John is a Greek.*
2. *John is a Hellene.*

In this case there is no difference in truth conditions.

▷ But there might be another context that does give a difference.

Contentious Cases of Synonymy Test

▷ **Example 2.4.34 (Problem).** The following sentences differ in truth values:

1. *Mary believes that John is a Greek*
2. *Mary believes that John is a Hellene*

So *Greek* is not synonymous to *Hellene*. The same holds in the classical example:

1. *The Ancients knew that Hesperus was Hesperus*
2. *The Ancients knew that Hesperus was Phosphorus*

In these cases most language users do perceive a difference in truth conditions while some philosophers vehemently deny that the sentences under 1. could be true in situations where the 2. sentences are false.

▷ It is important here of course that the context of substitution is within the scope of a verb of propositional attitude. (maybe later!)

A better Synonymy Test

▷ **Definition 2.4.35 (Synonymy).** The following is called the truth conditional synonymy test:

If replacing A by B in some sentence C does not preserve truth conditions in a compositional part of C, then A and B are not synonymous.

Testing Truth Conditions with Logic

▷ **Definition 2.4.36.** A logical language model $\mathcal{M}$ for a natural language L consists of a logical system $\langle\mathcal{L},\mathcal{K},\models\rangle$ and a function φ from L sentences to $\mathcal{L}$-formulae.

▷ **Problem:** How do we find out whether $\mathcal{M}$ models L faithfully?

▷ **Idea:** Test truth conditions of sentences against the predictions $\mathcal{M}$ makes.

▷ **Problem:** The truth conditions for a sentence S in L can only be formulated and verified by humans that speak L.

▷ **In Practice:** Truth conditions are expressed as "stories" that specify salient situations. Native speakers of L are asked to judge whether they make S true/false.

▷ **Observation 2.4.37.** *A logical language model $\mathcal{M} := \langle L,\mathcal{L},\varphi\rangle$ can be tested:*

1. *Select a sentence S and a situation W that makes S true in W.* *(according to humans)*
2. *Translate S in to an $\mathcal{L}$-formula $S' := \varphi(S)$.*
3. *Express W as a set Φ of $\mathcal{L}$-formulae.* *($\Phi \mathrel{\widehat{=}}$ truth conditions)*
4. *$\mathcal{M}$ is supported if $\Phi \models S'$, falsified if $\Phi \not\models S'$.*

▷ **Corollary 2.4.38.** *A logical language model constitutes a semantic meaning theory.*

2.5 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:** Formal 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 aspects 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 hypotheses about the world in the theories
  4. testing the predictions made by the theory against the real-world data. If the theory predicts the data, then this supports the theory, if not, we refine the theory, starting the process again at 2.

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.

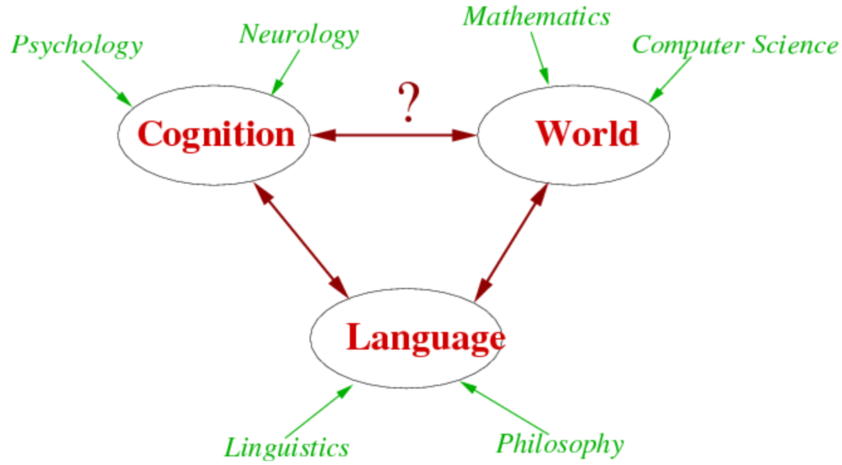
Even later, the Newtonian theory of celestial mechanics was replaced by Einstein's relativity theory, which makes better predictions for great distances and high-speed objects.

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 discipline and a trans-disciplinary endeavour, combining methods, results and insights from various disciplines.

NL Semantics as an Intersective Discipline


Psychology
Neurology
Mathematics
Computer Science
Cognition
?
World
Language
Linguistics
Philosophy

Part I

English as a Formal Language: The Method of Fragments

Chapter 3

Logic as a Tool for Modeling NL Semantics

In this chapter we will briefly introduce formal logic and motivate how we will use it as a tool for developing precise theories about natural language semantics.

We want to build a compositional, semantic meaning theory based on truth conditions, so that we can directly model the truth conditional synonymy test. We will see how this works in detail in **??** after we have recapped the necessary concepts about logic.

3.1 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 natural language, rather than writing rules for the a single construction while making implicit assumptions about the rest of the grammar. [Mon70]

> 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. 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 parse trees, which we will later use for semantics construction.

Natural Language Fragments

- ▷ **Methodological Problem:** How to organize the scientific method for natural language?
- ▷ **Delineation Problem:** What is natural language, e.g. English?
  Which aspects do we want to study?
- ▷ **Idea:** Select a subset (NL) sentences we want to study by a grammar!
  $\leadsto$ Richard Montague's method of fragments (1972).
- ▷ **Definition 3.1.1.** The language L of a context-free grammar is called a fragment of a natural language N, iff $L \subseteq N$.

- **Scientific Fiction:** We can exhaust English with ever-increasing fragments, develop a semantic meaning theory for each.

So far so good, these are nice ideas, but what does this mean in practice?

Using CFGs for NL Fragments

- **Idea:** Use nonterminals to classify NL phrases.
- **Definition 3.1.2.** We call a nonterminal symbol of a context-free grammar a phrasal category. We distinguish two kinds of rules:

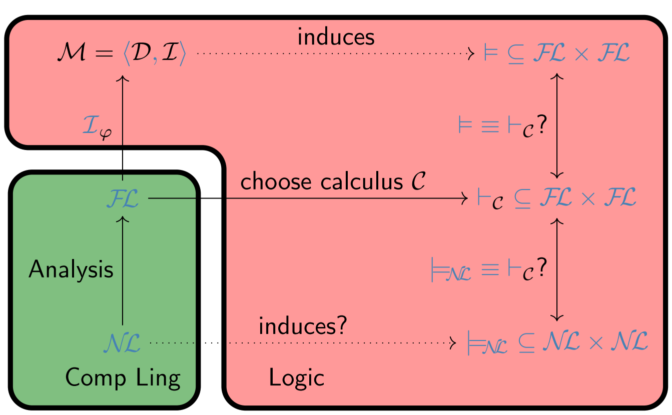
  structural rules: $\mathcal{L}\colon H \to c_1, \ldots, c_n$ with head H, label $\mathcal{L}$, and a sequence of phrasal categories c_i.

  lexical rules: $\mathcal{L}\colon H \to t_1 \mid \ldots \mid t_n$, where the t_i are terminals (i.e. NL phrases)
- **Definition 3.1.3.** In the method of fragments we use a CFG to parse sentences from the fragment into a parse tree (also called abstract syntax tree (AST) for further processing.
- **Todo:** We have to restrict our logical language models to fragments.
- **Definition 3.1.4.** A language fragment model consists of a CFG G, a logical system $\mathcal{L}$, and a semantics construction mapping φ from G-parse trees to $\mathcal{L}$-formulae.

We generically distinguish two parts of a grammar: the structural 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 atomic objects of language): their phrasal categories, their meaning, etc.

Formal Natural Language Semantics with Fragments

- **Idea:** We will follow the picture we have discussed before



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

Semantics by Translation

- **Idea:** We translate sentences by translating their syntax trees via tree node translation rules.
- **Note:** This makes the induced meaning theory compositional.
- **Definition 3.1.5.** 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 3.1.6.** For a natural language utterance or text A, we will use $\langle A \rangle$ for the result of translating A and call it the interpretation of A.
- **Definition 3.1.7 (Default Rule).** For every word w in the fragment we assume a constant w' in the logic $\mathcal{L}$ and the "pseudo-rule" $t1\colon w \rightsquigarrow w'$. (if no other translation rule applies)

3.2 What is Logic?

What is Logic?

- **Definition 3.2.1.** Logic $\widehat{=}$ 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 ($x, y, f, g, p, 1, \pi, \in, \neg, \forall, \exists$)
  - Model: things we understand (e.g. number theory)
  - Interpretation: maps formulae into models ($[\![\text{three plus five}]\!]^{\mathcal{I}} = 8$)
  - Validity: $\mathcal{M} \models \mathbf{A}$, iff $[\![\mathbf{A}]\!]^{\mathcal{I}} = \mathsf{T}$ (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?

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 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 (1623), (1642), and (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 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.
Logics make good (scientific[1]) 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.

[1]As 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

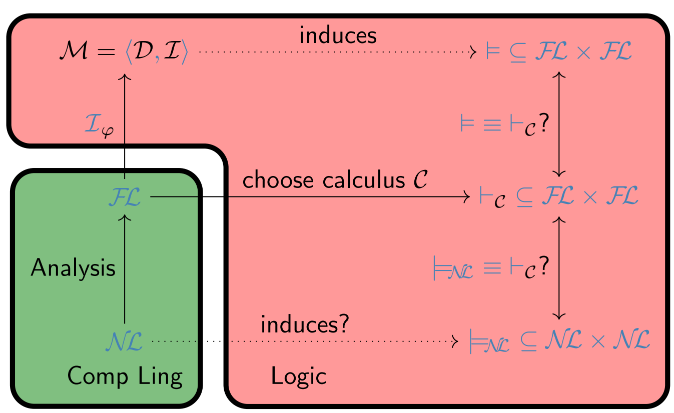
3.3 Using Logic to Model Meaning of Natural Language

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 ≤ 1900]
    * *I believe, that my audience already knows this.*
  - Modal logic [Lewis18, Kripke65]
    * *A man sleeps. He snores.* $((\exists X.\mathrm{man}(X) \wedge \mathrm{sleeps}(X))) \wedge \mathrm{snores}(X)$
  - Various dynamic approaches (e.g. DRT, DPL)
    * *Most men wear black*
  - Higher-order Logic, e.g. generalized quantifiers
  - ...

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 ituisn 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?



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

Since $\mathcal{FL}$ is a formal language of a logical system, it comes with a notion of model and an value function $\mathcal{I}_\varphi$ that translates $\mathcal{FL}$ formulae into objects of that model. This induces a notion of logical consequence[2] as explained in ??. It also comes with a calculus $\mathcal{C}$ acting on $\mathcal{FL}$ formulae, which (if we are lucky) is sound and complete (then the mappings in the upper rectangle commute).

What we are really interested 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 linguistic entailment 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 text/speech fragments, 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)
  - ▷ 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 logical calculi? (combinatorial explosion)
- ▷ **One possible answer:** Description Logics. (Recall the AI-1 lecture?)

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

Chapter 4

Fragment 1

We will now put the ideas from the last chapter into practice in the setting of the Montague's "Method of Fragments". We will introduce a first very simple fragment mostly for the purpose of setting up the conceptual infrastructure and seeing how the various bits and pieces might interact, not so much because the fragment in and of itself is linguistically interesting.

4.1 The First Fragment: Setting up the Basics

The first fragment will primarily be used for setting the stage, and introducing the method of fragments itself. the coverage of the fragment is too small to do anything useful with it, but it will allow us to discuss the salient features of the method, the particular setup of the grammars and semantics before graduating to more useful fragments.

Fragment $\mathcal{F}_1$ Data (Sentences we want to cover)

- ▷ **Fragment $\mathcal{F}_1$ Data:** We delineate the intended fragment by giving examples
  1. *Ethel kicked the cat and Fiona laughted*
  2. *Peter is the teacher*
  3. *The teacher is happy*
  4. *It is not the case that Bertie ran*
  5. *It is not the case that Jo is happy*
- ▷ We can later use these sentences as benchmark tests.

FAU Michael Kohlhase: LBS 60 2025-02-06

Now that we have the target logic we can complete the analysis arrow in slide 55. We do this again, by giving translation rules.

4.1.1 Natural Language Syntax (Fragment 1)

Structural Grammar Rules

- ▷ **Definition 4.1.1.** $\mathcal{F}_1$ uses the following eight phrasal categories

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

▷ **Definition 4.1.2.** We have the following production rules in $\mathcal{F}_1$.
$S1$: $S \to \mathrm{NP}\ V^i$,
$S2$: $S \to \mathrm{NP}\ V^t\ \mathrm{NP}$,
$N1$: $\mathrm{NP} \to N_{\mathrm{pr}}$,
$N2$: $\mathrm{NP} \to$ the N,
$S3$: $S \to$ It is not the case that S,
$S4$: $S \to S\ \mathrm{conj}\ S$,
$S5$: $S \to \mathrm{NP}$ is NP, and
$S6$: $S \to \mathrm{NP}$ is Adj

Lexical insertion rules for Fragment $\mathcal{F}_1$

▷ **Definition 4.1.3.** We have the following lexical insertion rules in fragment $\mathcal{F}_1$.

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

▷ **Definition 4.1.4.** A production rule whose head is a single non-terminal and whose body consists of a single terminal is called lexical or a lexical insertion rule.

▷ **Notation:** Lexical insertion rules are usually written using BNF alternative in the body ⇝ grouping rules with the same head.

▷ **Definition 4.1.5.** The subset of lexical rules of a grammar G is called the lexicon of G and the set of body symbols the vocabulary (or alphabet). The nonterminals in their heads are called lexical categories of G.

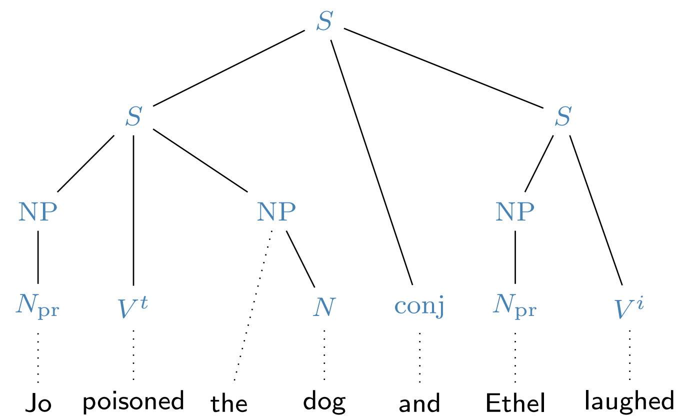
▷ **Note:** We will adopt the convention that new lexical insertion rules can be generated spontaneously as needed.

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

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

▷ **Observation 4.1.6.** *Jo poisoned the dog and Ethel laughed is a sentence of fragment 1*

▷ We can construct a parse tree for it!



FAU Michael Kohlhase: LBS 63 2025-02-06

4.1.2 Predicate Logic without Quantifiers

The next step will be to introduce the logical model we will use for fragment $\mathcal{F}_1$: Predicate Logic without Quantifiers. Syntactically, this logic is a fragment of first-order logic, but it's expressivity is equivalent to propositional logic.

Individuals and their Properties/Relationships

- **Observation:** We want to talk about individuals like Stefan, Nicole, and Jochen and their properties, e.g. being blond, or studying AI and relationships, e.g. that *Stefan loves Nicole*.
- **Idea:** Re-use PL^0, but replace propositional variables with something more expressive! (instead of fancy variable name trick)
- **Definition 4.1.7.** A first-order signature $\langle \Sigma^f, \Sigma^p \rangle$ consists of
  - $\Sigma^f := \bigcup_{k\in\mathbb{N}} \Sigma^f_k$ of function constants, where members of Σ^f_k denote k-ary functions on individuals,
  - $\Sigma^p := \bigcup_{k\in\mathbb{N}} {\Sigma^p}_k$ of predicate constants, where members of ${\Sigma^p}_k$ denote k-ary relations among individuals,

  where Σ^f_k and ${\Sigma^p}_k$ are pairwise disjoint, countable sets of symbols for each $k \in \mathbb{N}$.

  A 0-ary function constant refers to a single individual, therefore we call it a individual constant.

FAU Michael Kohlhase: LBS 64 2025-02-06

A Grammar for $\mathrm{PL}^{\mathrm{nq}}$

▷ **Definition 4.1.8.** The formulae of $\mathrm{PL}^{\mathrm{nq}}$ are given by the following grammar

function constants	f^k	$\in$	Σ_k^f	
predicate constants	p^k	$\in$	$\Sigma^p{}_k$	
terms	t	$::=$	f^0	individualconstant
		$\mid$	$f^k(t_1,\ldots,t_k)$	application
formulae	$\mathbf{A}$	$::=$	$p^k(t_1,\ldots,t_k)$	atomic
		$\mid$	$\neg\mathbf{A}$	negation
		$\mid$	$\mathbf{A}_1\wedge\mathbf{A}_2$	conjunction

$\mathrm{PL}^{\mathrm{nq}}$ Semantics

▷ **Definition 4.1.9.** Domains $\mathcal{D}_0=\{\mathsf{T},\mathsf{F}\}$ of truth values and $\mathcal{D}_\iota\neq\emptyset$ of individuals.

▷ **Definition 4.1.10.** Interpretation $\mathcal{I}$ assigns values to constants, e.g.

- ▷ $\mathcal{I}(\neg)\colon\mathcal{D}_0\to\mathcal{D}_0;\mathsf{T}\mapsto\mathsf{F};\mathsf{F}\mapsto\mathsf{T}$ and $\mathcal{I}(\wedge)=\ldots$ (as in PL^0)
- ▷ $\mathcal{I}\colon\Sigma_0^f\to\mathcal{D}_\iota$ (interpret individual constants as individuals)
- ▷ $\mathcal{I}\colon\Sigma_k^f\to{\mathcal{D}_\iota}^k\to\mathcal{D}_\iota$ (interpret function constants as functions)
- ▷ $\mathcal{I}\colon\Sigma^p{}_k\to\mathcal{P}({\mathcal{D}_\iota}^k)$ (interpret predicate constants as relations)

▷ **Definition 4.1.11.** The value function $\mathcal{I}$ assigns values to formulae: (recursively)

- ▷ $\mathcal{I}(f(\mathbf{A}^1,\ldots,\mathbf{A}^k)):=\mathcal{I}(f)(\mathcal{I}(\mathbf{A}^1),\ldots,\mathcal{I}(\mathbf{A}^k))$
- ▷ $\mathcal{I}(p(\mathbf{A}^1,\ldots,\mathbf{A}^k)):=\mathsf{T}$, iff $\langle\mathcal{I}(\mathbf{A}^1),\ldots,\mathcal{I}(\mathbf{A}^k)\rangle\in\mathcal{I}(p)$
- ▷ $\mathcal{I}(\neg\mathbf{A})=\mathcal{I}(\neg)(\mathcal{I}(\mathbf{A}))$ and $\mathcal{I}(\mathbf{A}\wedge\mathbf{B})=\mathcal{I}(\wedge)(\mathcal{I}(\mathbf{A}),\mathcal{I}(\mathbf{G}))$ (just as in PL^0)

▷ **Definition 4.1.12.** Model: $\mathcal{M}=\langle\mathcal{D}_\iota,\mathcal{I}\rangle$ varies in $\mathcal{D}_\iota$ and $\mathcal{I}$.

▷ **Theorem 4.1.13.** $\mathrm{PL}^{\mathrm{nq}}$ *is isomorphic to* PL^0 *(interpret atoms as prop. variables)*

All of the definitions above are quite abstract, we now look at them again using a very concrete – if somewhat contrived – example: The relevant parts are a universe $\mathcal{D}$ with four elements, and an interpretation that maps the signature into individuals, functions, and predicates over $\mathcal{D}$, which are given as concrete sets.

A Model for $\mathrm{PL}^{\mathrm{nq}}$

▷ **Example 4.1.14.** Let $L:=\{a,b,c,d,e,P,Q,R,S\}$, we set the universe $\mathcal{D}:=\{\clubsuit,\spadesuit,\heartsuit,\diamondsuit\}$, and specify the interpretation function $\mathcal{I}$ by setting

- ▷ $a\mapsto\clubsuit$, $b\mapsto\spadesuit$, $c\mapsto\heartsuit$, $d\mapsto\diamondsuit$, and $e\mapsto\diamondsuit$ for constants,
- ▷ $P\mapsto\{\clubsuit,\spadesuit\}$ and $Q\mapsto\{\spadesuit,\diamondsuit\}$, for unary predicate constants.
- ▷ $R\mapsto\{\langle\heartsuit,\diamondsuit\rangle,\langle\diamondsuit,\heartsuit\rangle\}$, and $S\mapsto\{\langle\diamondsuit,\spadesuit\rangle,\langle\spadesuit,\clubsuit\rangle\}$ for binary predicate constants.

▷ **Example 4.1.15 (Computing Meaning in this Model).**

- $\mathcal{I}(R(a,b) \wedge P(c)) = \mathsf{T}$, iff
- $\mathcal{I}(R(a,b)) = \mathsf{T}$ and $\mathcal{I}(P(c)) = \mathsf{T}$, iff
- $\langle \mathcal{I}(a), \mathcal{I}(b) \rangle \in \mathcal{I}(R)$ and $\mathcal{I}(c) \in \mathcal{I}(P)$, iff
- $\langle \clubsuit, \spadesuit \rangle \in \{\langle \heartsuit, \diamondsuit \rangle, \langle \diamondsuit, \heartsuit \rangle\}$ and $\heartsuit \in \{\clubsuit, \spadesuit\}$

So, $\mathcal{I}(R(a,b) \wedge P(c)) = \mathsf{F}$.

FAU Michael Kohlhase: LBS 67 2025-02-06

The example above also shows how we can compute of meaning by in a concrete model: we just follow the evaluation rules to the letter.
We now come to the central technical result about $\mathrm{PL}^{\mathrm{nq}}$: it is essentially the same as propositional logic (PL^0). We say that the two logic are isomorphic. Technically, this means that the formulae of $\mathrm{PL}^{\mathrm{nq}}$ can be translated to PL^0 and there is a corresponding model translation from the models of PL^0 to those of $\mathrm{PL}^{\mathrm{nq}}$ such that the respective notions of evaluation are assignped to each other.

$\mathrm{PL}^{\mathrm{nq}}$ and PL^0 are Isomorphic

- **Observation:** For every choice of Σ of signature, the set $\mathcal{A}_\Sigma$ of atomic $\mathrm{PL}^{\mathrm{nq}}$ formulae is countable, so there is a $\mathcal{V}_\Sigma \subseteq \mathcal{V}_0$ and a bijection $\theta_\Sigma \colon \mathcal{A}_\Sigma \to \mathcal{V}_\Sigma$.

  θ_Σ can be extended to formulae as $\mathrm{PL}^{\mathrm{nq}}$ and PL^0 share connectives.

- **Lemma 4.1.16.** *For every model $\mathcal{M} = \langle \mathcal{D}_\iota, \mathcal{I} \rangle$, there is a variable assignment $\varphi_\mathcal{M}$, such that $\mathcal{I}_{\varphi_\mathcal{M}}(\mathbf{A}) = \mathcal{I}(\mathbf{A})$.*

- *Proof sketch:* We just define $\varphi_\mathcal{M}(X) := \mathcal{I}(\theta_\Sigma^{-1}(X))$

- **Lemma 4.1.17.** *For every variable assignment $\psi \colon \mathcal{V}_\Sigma \to \{\mathsf{T}, \mathsf{F}\}$ there is a model $\mathcal{M}^\psi = \langle \mathcal{D}^\psi, \mathcal{I}^\psi \rangle$, such that $\mathcal{I}_\psi(\mathbf{A}) = \mathcal{I}^\psi(\mathbf{A})$.*

- *Proof sketch:* see next slide

- **Corollary 4.1.18.** *$\mathrm{PL}^{\mathrm{nq}}$ is isomorphic to PL^0, i.e. the following diagram commutes:*

$$\begin{tikzcd}
\langle \mathcal{D}^\psi, \mathcal{I}^\psi \rangle & \mathcal{V}_\Sigma \to \{\mathsf{T}, \mathsf{F}\} \arrow[l, "\psi \mapsto \mathcal{M}^\psi"'] \\
\mathrm{PL}^{\mathrm{nq}}(\Sigma) \arrow[u, "\mathcal{I}^\psi()"] \arrow[r, "\theta_\Sigma"] & \mathrm{PL}^0(\mathcal{A}_\Sigma) \arrow[u, "\mathcal{I}_{\varphi_\mathcal{M}}()"']
\end{tikzcd}$$

- **Note:** This constellation with a language isomorphism and a corresponding model isomorphism (in converse direction) is typical for a logic isomorphism.

FAU Michael Kohlhase: LBS 68 2025-02-06

The practical upshot of the commutative diagram from ?? is that if we have a way of computing evaluation (or entailment for that matter) in PL^0, then we can "borrow" it for $\mathrm{PL}^{\mathrm{nq}}$ by composing it with the language and model translations. In other words, we can reuse calculi and automated theorem provers from PL^0 for $\mathrm{PL}^{\mathrm{nq}}$.
But we still have to provide the proof for ??, which we do now.

Valuation and Satisfiability

▷ **Lemma 4.1.19.** *For every variable assignment* $\psi\colon \mathcal{V}_\Sigma \to \{\mathsf{T},\mathsf{F}\}$ *there is a model* $\mathcal{M}^\psi = \langle \mathcal{D}^\psi, \mathcal{I}^\psi \rangle$, *such that* $\mathcal{I}_\psi(\mathbf{A}) = \mathcal{I}^\psi(\mathbf{A})$.

▷ *Proof:* We construct $\mathcal{M}^\psi = \langle \mathcal{D}^\psi, \mathcal{I}^\psi \rangle$ and show that it works as desired.

1. Let $\mathcal{D}^\psi$ be the set of $\mathrm{PL}^{\mathrm{nq}}$ terms over Σ, and
   - ▷ $\mathcal{I}^\psi(f)\colon {\mathcal{D}_\iota}^k \to {\mathcal{D}^\psi}^k; \langle \mathbf{A}_1, \ldots, \mathbf{A}_k \rangle \mapsto f(\mathbf{A}_1, \ldots, \mathbf{A}_k)$ for $f \in \Sigma_k^f$
   - ▷ $\mathcal{I}^\psi(p) := \{\langle \mathbf{A}_1, \ldots, \mathbf{A}_k \rangle \,|\, \psi(\theta_\psi^{-1} p(\mathbf{A}_1, \ldots, \mathbf{A}_k)) = \mathsf{T}\}$ for $p \in \Sigma^p$.
2. We show $\mathcal{I}^\psi(\mathbf{A}) = \mathbf{A}$ for terms $\mathbf{A}$ by induction on $\mathbf{A}$
   2.1. If $\mathbf{A} = c$, then $\mathcal{I}^\psi(\mathbf{A}) = \mathcal{I}^\psi(c) = c = \mathbf{A}$
   2.2. If $\mathbf{A} = f(\mathbf{A}_1, \ldots, \mathbf{A}_n)$ then
   $\mathcal{I}^\psi(\mathbf{A}) = \mathcal{I}^\psi(f)(\mathcal{I}(\mathbf{A}_1), \ldots, \mathcal{I}(\mathbf{A}_n)) = \mathcal{I}^\psi(f)(\mathbf{A}_1, \ldots, \mathbf{A}_k) = \mathbf{A}$.
3. For a $\mathrm{PL}^{\mathrm{nq}}$ formula $\mathbf{A}$ we show that $\mathcal{I}^\psi(\mathbf{A}) = \mathcal{I}_\psi(\mathbf{A})$ by induction on $\mathbf{A}$.
   3.1. If $\mathbf{A} = p(\mathbf{A}_1, \ldots, \mathbf{A}_k)$, then $\mathcal{I}^\psi(\mathbf{A}) = \mathcal{I}^\psi(p)(\mathcal{I}(\mathbf{A}_1), \ldots, \mathcal{I}(\mathbf{A}_n)) = \mathsf{T}$, iff $\langle \mathbf{A}_1, \ldots, \mathbf{A}_k \rangle \in \mathcal{I}^\psi(p)$, iff $\psi(\theta_\psi^{-1} \mathbf{A}) = \mathsf{T}$, so $\mathcal{I}^\psi(\mathbf{A}) = \mathcal{I}_\psi(\mathbf{A})$ as desired.
   3.2. If $\mathbf{A} = \neg \mathbf{B}$, then $\mathcal{I}^\psi(\mathbf{A}) = \mathsf{T}$, iff $\mathcal{I}^\psi(\mathbf{B}) = \mathsf{F}$, iff $\mathcal{I}^\psi(\mathbf{B}) = \mathcal{I}_\psi(\mathbf{B})$, iff $\mathcal{I}^\psi(\mathbf{A}) = \mathcal{I}_\psi(\mathbf{A})$.
   3.3. If $\mathbf{A} = \mathbf{B} \wedge \mathbf{C}$ then we argue similarly
4. Hence $\mathcal{I}^\psi(\mathbf{A}) = \mathcal{I}_\psi(\mathbf{A})$ for all $\mathrm{PL}^{\mathrm{nq}}$ formulae and we have concluded the proof.

Now that we have the target logic we can complete the analysis arrow in slide 55. We do this again, by giving translation rules.

4.1.3 Natural Language Semantics via Translation

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

▷ **Definition 4.1.20.** We have the following translation rules for non-leaf node of the syntax tree

$T1$: $[X_{\mathrm{NP}}, Y_{V^i}]_S \rightsquigarrow Y'(X')$
$T2$: $[X_{\mathrm{NP}}, Y_{V^t}, Z_{\mathrm{NP}}]_S \rightsquigarrow Y'(X', Z')$
$T3$: $[X_{N_{\mathrm{pr}}}]_{\mathrm{NP}} \rightsquigarrow X'$
$T4$: $[\text{the}, X_N]_{\mathrm{NP}} \rightsquigarrow theX'$
$T5$: $[\text{It is not the case that} X_S]_S \rightsquigarrow (\neg X')$
$T6$: $[X_S, Y_{\mathrm{conj}}, Z_S]_S \rightsquigarrow Y'(X', Z')$
$T7$: $[X_{\mathrm{NP}}, \text{is}, Y_{\mathrm{NP}}]_S \rightsquigarrow X' = Y'$
$T8$: $[X_{\mathrm{NP}}, \text{is}, Y_{\mathrm{Adj}}]_S \rightsquigarrow Y'(X')$

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

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

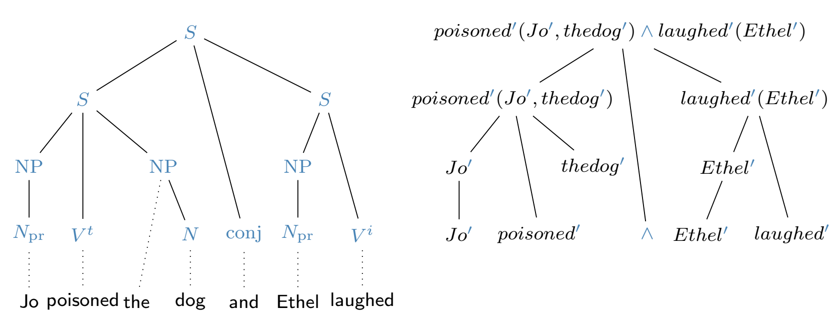
Translation rule for basic lexical items

- **Definition 4.1.21.** 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 $w' \in {\Sigma^p}_1$. (one-place predicate)
  - If w is a transitive verb, $w' \in {\Sigma^p}_2$. (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 lexical item (i.e. 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.
- Translations for logical connectives

$$t2\colon \text{and} \rightsquigarrow \wedge,\ t3\colon \text{or} \rightsquigarrow \vee,\ t4\colon \text{it is not the case that} \rightsquigarrow \neg$$

Translation Example

- **Observation 4.1.22.** *Jo poisoned the dog and Ethel laughed is a sentence of fragment $\mathcal{F}_1$.*
- We can construct a syntax tree for it!



4.2 Testing Truth Conditions via Inference

Now that our language fragment model is complete for fragment $\mathcal{F}_1$, we can test it to see whether it makes the correct predictions.

We use one of the examples from introduction even though we have to somewhat force-fit into fragment $\mathcal{F}_1$. As the fragment was mostly introduced to show the basic setup, this may be forgivable.

Testing Truth Conditions in $\mathrm{PL}^{\mathrm{nq}}$

- **Idea 1:** To test our language model ($\mathcal{F}_1$)
  - Select a sentence S and a situation W that makes S true. (according to humans)
  - Translate S in to a formula S' in $\mathrm{PL}^{\mathrm{nq}}$.
  - Express W as a set Φ of formulae in $\mathrm{PL}^{\mathrm{nq}}$ ($\Phi \mathrel{\hat{=}}$ truth conditions)
  - Our language model is supported if $\Phi \models S'$, falsified if $\Phi \not\models S'$.
- **Example 4.2.1 (John chased the gangster in the red sports car).**
  - We claimed that we have three readings ??
    $R_1 := c(j,g) \wedge in(j,s)$, $R_2 := c(j,g) \wedge in(g,s)$, and $R_3 := c(j,g) \wedge in(j,s) \wedge in(g,s)$
  - So there must be three distinct situations W that make S true
  1. *John is in the red sports car, but the gangster isn't*
     $W_1 := c(j,g) \wedge in(j,s) \wedge \neg in(g,s)$, so $W_1 \models R_1$, but $W_1 \not\models R_2$ and $W_1 \not\models R_3$
  2. *The gangster is in the red sports car, but John isn't*
     $W_2 := c(j,g) \wedge in(g,s) \wedge \neg in(j,s)$, so $W_2 \models R_2$, but $W_2 \not\models R_1$ and $W_2 \not\models R_3$
  3. *Both are in the red sports car*
     $\mathrel{\hat{=}}$ they run around on the back seat of a very big sports car
     $W_3 := c(j,g) \wedge in(j,s) \wedge in(g,s)$, so $W_3 \models R_3$, but $W_3 \not\models R_1$ and $W_3 \not\models R_1$
- **Idea 2:** Use a calculus to model $\models$, e.g. $\mathcal{ND}_0$

4.3 Summary & Evaluation

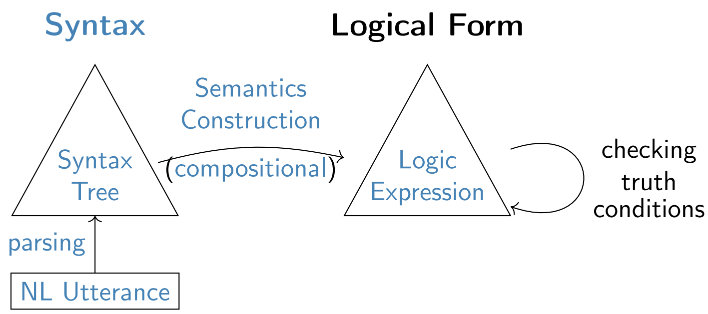
So let us evaluate what we have achieved so far:

Fragment $\mathcal{F}_1$ – Summary

- Fragment $\mathcal{F}_1$ of English (defined by grammar + lexicon)
- Logic $\mathrm{PL}^{\mathrm{nq}}$ (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 $\mathcal{F}_1 \rightsquigarrow \mathrm{PL}^{\mathrm{nq}}$ (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

Summary: The Interpretation Process (so far)

- **The Interpretation Process in $\mathcal{F}_1$:** Can be visualized in the following diagram:



Chapter 5

Fragment 2: Pronouns and World Knowledge $\rightsquigarrow$ Semantic/Pragmatic Analysis

In this chapter we will extend fragment $\mathcal{F}_1$ from last chapter with and pronouns: We want to cover discourses like *Peter loves Fido. Even though he bites him sometimes*. As we already observed there, we crucially need a notion of context to determine the meaning of the pronoun during semantic/pragmatic analysis, which we focus on here.

In particular, 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/reading, i.e. *Humans don't bite dogs.*

For this purpose, we introduce a new concept: the notion of a tableau machine that casts semantic/pragmatic analysis as an inferential process.

5.1 Fragment 2: Pronouns and Anaphora

We start out with the new data we want to cover in this fragment and some ideas of all the things we need to adapt. Actually there that is only one new sentence: The Peter/Fido example from the introduction of LBS.

Fragment $\mathcal{F}_2$ ($\mathcal{F}_2 \mathrel{\widehat{=}} \mathcal{F}_1$ + Anaphoric Pronouns)

- ▷ **Want to cover:** *Peter loves Fido. He bites him.* (almost intro)
  - ▷ **We need**: Translation and interpretation for pronouns like *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 $\mathrm{PL}^{\mathrm{nq}}$ (work backwards from that)
- ▷ **Logical System:** $\mathrm{PL}^{\mathrm{nq}}(\mathcal{V}) = \mathrm{PL}^{\mathrm{nq}} +$ variables (Translate pronouns to variables)

FAU Michael Kohlhase: LBS 76 2025-02-06

For the syntax, the necessary changes are quite minor as well. We need to extend the grammar from fragment 1 by one new phrasal category and one derivation rule:

New Grammar in $\mathcal{F}_2$ (Pronouns)

▷ **Definition 5.1.1.** We have the following structural grammar rules in $\mathcal{F}_2$

$$\begin{gathered} S1\colon S \to \mathrm{NP}, V^i, \\ S2\colon S \to \mathrm{NP}, V^t, \mathrm{NP}, \\ N1\colon \mathrm{NP} \to N_{\mathrm{pr}}, \\ N2\colon \mathrm{NP} \to \mathrm{Pron}, \\ N3\colon \mathrm{NP} \to \text{the}, N, \\ S3\colon S \to \text{it is not the case that}, S, \\ S4\colon S \to S, \mathrm{conj}, S, \\ S5\colon S \to \mathrm{NP}, \text{is}, \mathrm{NP}, \\ S6\colon S \to \mathrm{NP}, \text{is}, \mathrm{Adj} \end{gathered}$$

and one additional lexical rule:

$$L7\colon \mathrm{Pron} \to he \mid she \mid it \mid we \mid they$$

We also have to adapt the logical system we want to translate into, and we do this by adding variables. Recall that variables denote arbitrary individuals and can be instantiated in inference-processes. That makes them seem suitable as a logical counterpart for pronouns.

The main idea here is to extend $\mathrm{PL^{nq}}$ – 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 i.e. every occurrance of a pronoun to a new variable.

The mathematical development of $\mathrm{PL^{nq}}(\mathcal{V})$ itself is rather simple: it extends $\mathrm{PL^{nq}}$, but stays a fragment of first-order logic, so we can get by with the methods developed for that.

Note that we do not allow quantifiers yet that will come in **??**, as quantifiers will pose new problems, and we can already solve some linguistically interesting problems without them.

Predicate Logic with Variables (but no Quantifiers)

▷ **Definition 5.1.2 (Logical System $\mathrm{PL^{nq}}(\mathcal{V})$).** $\mathrm{PL^{nq}}(\mathcal{V}) := \mathrm{PL^{nq}}$ + variables

▷ **Definition 5.1.3 ($\mathrm{PL^{nq}}(\mathcal{V})$ Syntax).**
Category $\mathcal{V} = \{X, Y, Z, X^1, X^2, \ldots\}$ of variables (allow variables wherever individual constants were allowed)

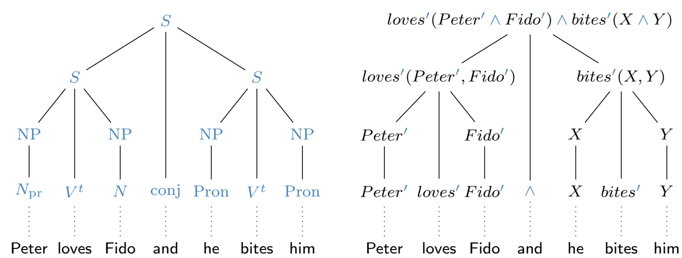
▷ **Definition 5.1.4 ($\mathrm{PL^{nq}}(\mathcal{V})$ Semantics).**
First-order model $\mathcal{M} = \langle \mathcal{D}, \mathcal{I} \rangle$ (need to evaluate variables)

- ▷ variable assignment: $\varphi\colon \mathcal{V}_\iota \to U$
- ▷ value function: $\mathcal{I}_\varphi(X) = \varphi(X)$ (defined like $\mathcal{I}$ elsewhere)
- ▷ call a $\mathrm{PL^{nq}}(\mathcal{V})$ formula $\mathbf{A}$ valid in $\mathcal{M}$ under φ, iff $\mathcal{I}_\varphi(\mathbf{A}) = \mathsf{T}$,
- ▷ call it satisfiable in $\mathcal{M}$, iff there is a variable assignment φ, such that $\mathcal{I}_\varphi(\mathbf{A}) = \mathsf{T}$

And now, the translation to $\mathrm{PL^{nq}}(\mathcal{V})$ is again very simple:

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

- **Idea:** Pronouns are translated into new variables (so far)
- **New Translation Rule:** We translate pronouns by the "rule": $T9$: $[X]_{\mathrm{Pron}} \rightsquigarrow Y_{new}$, where Y_{new} is a new variable.
- The syntax/semantic trees for *Peter loves Fido and he bites him.* are straightforward. (almost intro)



Here we see how the principle of compositionality we impose on semantics construction makes our life easy: for every syntax rule, we need exactly one translation rule – here the one above.

5.2 Inference with World Knowledge and Free Variables – A Case Study

In $\mathcal{F}_1$ we did not have a dedicated semantic/pragmatic analysis phase, but in $\mathcal{F}_2$ we have anaphoric pronouns which need to be resolved. So we will start experimenting with model generation tableaux to see where this will go with respect to anaphor resolution.

5.2.1 Pragmatics via Model Generation Tableaux?

As we have established (see **??**) that $\mathrm{PL}^{\mathrm{nq}}$ is isomorphic to PL^0, we can directly use the propositional tableau calculus for deciding entailment in $\mathrm{PL}^{\mathrm{nq}}$. For $\mathrm{PL}^{\mathrm{nq}}(\mathcal{V})$, we have to do more, especially, if we want to deal with anaphora and the world knowledge we have to use to process them. In particular we will have to extend our tableau calculus with new inference rules for the new language capabilities.

A Tableau Calculus for $\mathrm{PL}^{\mathrm{nq}}(\mathcal{V})$

- **Definition 5.2.1 (Tableau Calculus for $\mathrm{PL}^{\mathrm{nq}}(\mathcal{V})$).** $\mathcal{T}_\mathcal{V}^p = \mathcal{T}_0$ + new tableau rules for formulae with variables

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

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

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

- the first rule is used for world knowledge (up in the branch)
- the second rule is used for input logical forms $\boxed{\cdots}$
  this rule has to be applied eagerly (while they are still at the leaf)

We use free variables for two purposes in our new fragment: Free variables in the input stand for pronouns, their value will be determined by random instantiation. Free variables in the 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 $\mathrm{human}(X) \wedge \mathrm{dog}(Y) \Rightarrow \neg\mathrm{bites}(X,Y)$.
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*.
To allow for world knowledge, we generalize the notion of an initial tableau. Instead of allowing only the initial labeled formula at the root node, we allow a linear tree whose nodes are labeled formulae with positive 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.

Some Examples in $\mathcal{F}_2$

- **Example 5.2.2 (Peter snores).** (Only sleeping people snore)

$$(\mathrm{snores}(X) \Rightarrow \mathrm{sleeps}(X))^{\mathsf{T}}$$
$$\boxed{\mathrm{snores}(\mathrm{peter})}$$
$$(\mathrm{snores}(\mathrm{peter}) \Rightarrow \mathrm{sleeps}(\mathrm{peter}))^{\mathsf{T}}$$
$$\mathrm{sleeps}(\mathrm{peter})^{\mathsf{T}}$$

- **Example 5.2.3 (Peter sleeps. John walks. He snores).** (who snores?)

$$\boxed{\mathrm{sleeps}(\mathrm{peter})}$$
$$\boxed{\mathrm{walks}(\mathrm{john})}$$
$$\boxed{\mathrm{snores}(X)}$$
$$\mathrm{snores}(\mathrm{peter})^{\mathsf{T}} \mid \mathrm{snores}(\mathrm{john})^{\mathsf{T}}$$

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 **??** to derive the fact that Peter must sleep.
The third input formula contains a free variable, which is instantiated by all constant in the Herbrand universe (two in our case). This gives rise to two Herbrand 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? The everlasting Question in AI

▷ **Example 5.2.4.**

Tweety is a bird

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

Tweety is an eagle

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

▷ For the second we need to add more world knowledge.

5.2.2 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, and worry about the theory in the next subsection.
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.)

$$\boxed{l(p,f)}$$
$$\boxed{b(X,Y)}$$
$$b(p,p)^{\mathsf{T}} \mid b(p,f)^{\mathsf{T}} \mid b(f,p)^{\mathsf{T}} \mid b(f,f)^{\mathsf{T}}$$

▷ **Problem:** We get four readings instead of one!

▷ **Idea:** We have not specified enough world knowledge.

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.

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

- ▷ **Observation:** Anaphor resolution introduces ambiguities.
- ▷ **Pragmatics:** Use world knowledge to filter out impossible readings.

We observe that our extended tableaucalculus 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 $\mathrm{PL}^{\mathrm{nq}}(\mathcal{V})$ 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 knowledge like *Every car has wheels, which have a tire.* and *if a tire is flat, the car breaks down.*, which is outside the realm of $\mathrm{PL}^{\mathrm{nq}}(\mathcal{V})$.

5.2.3 The Computational Role of Ambiguities

In the case study above we have seen that anaphor resolution introduces ambiguities, and we can use world knowledge to filter out impossible readings. Generally in the traditional waterfall model of language processing – which posits that NL understanding is a process that analyzes the input in stages: syntax, semantics construction, pragmatics – 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)
  - ▹ 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)
   - ⇝ multiple syntactic representations (pragmatics as filter)
2. resolve the pronouns in the pragmatics (our model here)
   - ⇝ need underspecified syntactic representations (e.g. variables)
   - ⇝ 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 9 (Case Study: Peter loves Fido, even though he sometimes bites him) in the LBS lecture notes to *Peter loves Mary. She loves him.*.

Translation for Fragment $\mathcal{F}_2$ Reconsidered

- **Idea:** Pronouns are translated into *new variables*. (so far)
- **Problem:** *Peter loves Mary. She loves him.*

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

- **Idea:** Attach world knowledge to pronouns. (just as with Peter and Fido)
  - Use the world knowledge to distinguish (linguistic) gender by predicates masc and fem.
- **Problem:** Properties of
  - proper names are given in the model,
  - pronouns must be given by the syntax-semantics interface.
- **In particular:** How to generate $\mathrm{loves}(X,Y)\wedge\mathrm{masc}(X)\wedge\mathrm{fem}(Y)$ compositionally?

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 9 (Case Study: Peter loves Fido, even though he sometimes bites him) in the LBS lecture notes 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 woulad need to generate $\mathrm{loves}(X,Y) \wedge \mathrm{masc}(X) \wedge \mathrm{fem}(Y)$ 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 and extend the logic, so that compositionality becomes simpler to achieve. We will explore the latter option in the next section. 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 $\mathcal{D}$ is subdivided into subsets $\mathcal{D}_{\mathbb{A}} \subseteq \mathcal{D}$, 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 5.2.5 (Sorted Logics).** (in our case $PL^1_{\mathcal{S}}$)
  Assume a set of sorts $\mathcal{S} := \{\mathbb{A}, \mathbb{B}, \mathbb{C}, \ldots\}$, annotate every syntactic and semantic structure with them. Make all constructions and operations well worted:
  - 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 $\mathcal{D}$ into subsets $\mathcal{D}_{\mathbb{A}} \subseteq \mathcal{D}$
    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) For every sort $\mathbb{A}$, we introduce a first-order predicate $\mathcal{R}_{\mathbb{A}}$ and
  - 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 $\mathcal{R}_{\mathbb{A}}(X) \wedge \mathcal{R}_{\mathbb{C}}(Z) \wedge R(X,Y) \wedge \neg P(Z)$ in input.
  - Meaning is preserved, but translation is non-compositional!

FAU Michael Kohlhase: LBS 87 2025-02-06

5.3 Tableaux and Model Generation

Now that we have seen that using tableaux in model generation mode – i.e. decorate the initial formula with T and see what branches develop – let us supply some of the theory after the fact, and clean up all the details that have been missing.

The main result of this section is the a tableau machine – an online inferential process for natural language interpretation – that we will develop further as a model for semantic/pragmatic analysis in this course.

5.3.1 Tableau Branches and Herbrand Models

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

Model Generation and Interpretation

▷ **Example 5.3.1 (from above).** In ?? we claimed that the set

$$\mathcal{B} := \{\mathrm{loves}(\mathrm{john}, \mathrm{mary})^{\mathsf{F}}, \mathrm{loves}(\mathrm{mary}, \mathrm{bill})^{\mathsf{T}}\}$$

of literals on the open branch of the tableau $\mathcal{T}$ below

$$\begin{array}{c|c} \multicolumn{2}{c}{(\mathrm{loves}(\mathrm{mary}, \mathrm{bill}) \vee \mathrm{loves}(\mathrm{john}, \mathrm{mary}))^{\mathsf{T}}} \\ \multicolumn{2}{c}{\mathrm{loves}(\mathrm{john}, \mathrm{mary})^{\mathsf{F}}} \\ \mathrm{loves}(\mathrm{mary}, \mathrm{bill})^{\mathsf{T}} & \mathrm{loves}(\mathrm{john}, \mathrm{mary})^{\mathsf{T}} \\ & \bot \end{array}$$

constitutes a "model". (it can be conveniently read off)

▷ **Recap:** A first-order 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 $\mathcal{I}$ based on $\mathcal{B}$.

FAU Michael Kohlhase: LBS 88 2025-02-06

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 loves that makes $\mathrm{loves}(\mathrm{john}, \mathrm{mary})$ false and $\mathrm{loves}(\mathrm{mary}, \mathrm{bill})$ true. This is simple: we just take $\mathcal{I}(\mathrm{loves}) = \{\langle A, B \rangle\}$. Indeed we have

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

but $\mathcal{I}_\varphi(\mathrm{loves}(\mathrm{john}, \mathrm{mary})) = \mathsf{F}$ according to the rules in[1].

Model Generation and Models

▷ **Idea:** Choose the universe $\mathcal{D}$ as the set Σ_0^f of constants, choose $\mathcal{I} = \mathrm{Id}_{\Sigma_0^f}$, interpret $p \in \Sigma^p{}_k$ as $\mathcal{R}_{\mathcal{B}}(p) := \{\langle a_1, \ldots, a_k \rangle \mid p(a_1, \ldots, a_k)^{\mathsf{T}} \in \mathcal{B}\}$.

▷ **Definition 5.3.2.** We call a model a Herbrand model, iff $\mathcal{D} = \Sigma_0^f$ and $\mathcal{I} = \mathrm{Id}_{\Sigma_0^f}$.

▷ **Definition 5.3.3.** Let $\mathcal{H}$ be a set of atomic propositions such that $\mathbf{A}^{\mathsf{F}} \not\in \mathcal{H}$, if $\mathbf{A}^{\mathsf{T}} \in \mathcal{H}$, then we call $\mathcal{H}$ a Herbrand valuation.

▷ **Lemma 5.3.4.** *Let $\mathcal{H}$ be a Herbrand valuation, then setting $\mathcal{I}(p) := \mathcal{R}_{\mathcal{H}}(p)$ yields a Herbrand model that satisfies $\mathcal{H}$.* (*proof trivial*)

▷ **Corollary 5.3.5.** *Let $\mathcal{H}$ be a Herbrand valuation, then there is a Herbrand model that satisfies $\mathcal{H}$.* (*use $\mathcal{R}_{\mathcal{H}}$*)

FAU Michael Kohlhase: LBS 89 2025-02-06

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

[1] EdNote: crossref

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.

5.3.2 Using Model Generation for Interpretation

We will now use model generation directly as a tool for discourse interpretation. To do so, we will have to go beyond just looking at model generation calculi and extend this to an inference-driven processing model: the tableau machine. But first we look for the motivation for this from cognitive science.

Using Model Generation for Interpretation

- ▷ **Definition 5.3.6.** Mental model theory [JL83; JLB91] posits that humans form mental models of the world, i.e. (neural) representations of possible states of the world that are consistent with the perceptions up to date and use them to reason about the world.
- ▷ **So** communication by natural language is a process of transporting parts of the mental model of the speaker into the mental model of the hearer.
- ▷ **Therefore** the NL 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.
- ▷ **Idea:** We can model discourse understanding as a process of generating Herbrand models for the logical form of an utterance in a discourse by a tableau based model generation procedure.
- ▷ **Advantage:** Capturing ambiguity by generating multiple models for input logical forms.

Tableau Machine

- ▷ **Definition 5.3.7.** The tableau machine is an inferential cognitive model for incremental natural language understanding that implements mental model theory via tableau based model generation over a sequence of input sentences.

  It iterates the following process for every input sentence staring with the empty tableau:

  1. add the logical form of the input sentence S_i to the selected branch,
  2. perform tableau inferences below S_i until saturated or some resource criterion is met
  3. if there are open branches choose a "preferred branch", otherwise backtrack to previous tableau for S_j with $j < i$ and open branches, then re-process $S_{j+1}, \ldots, S_i$ if possible, else fail.

  The output is application-dependent; some choices are

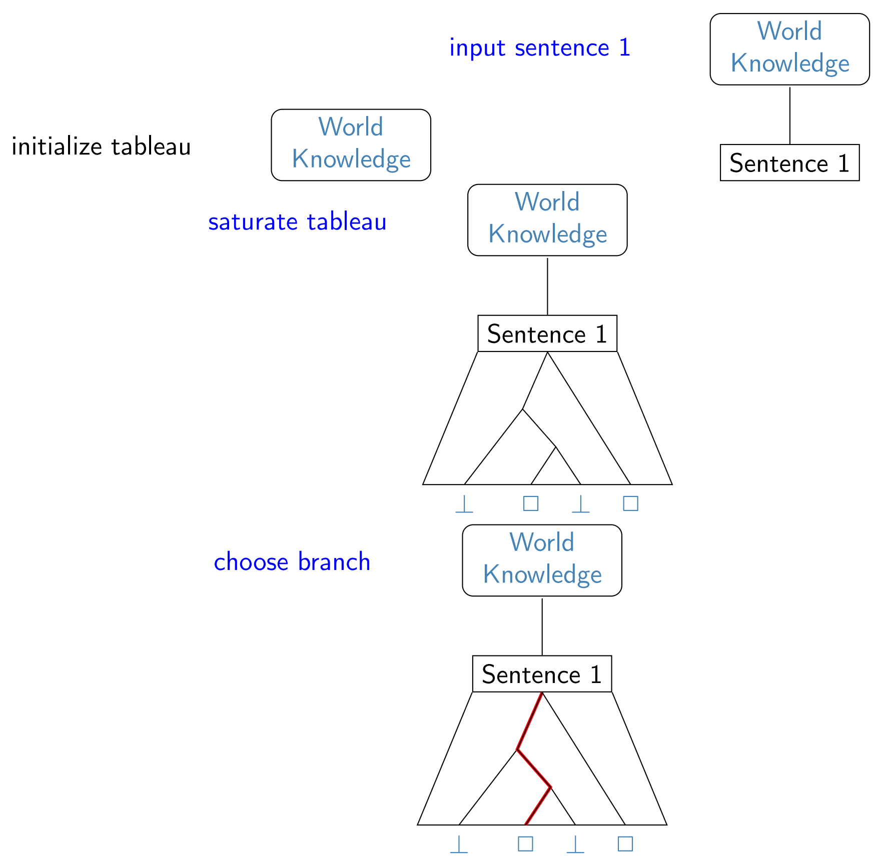
- ▹ the Herbrand model for the preferred branch ⇝ preferred interpretation;
- ▹ the literals augmented with all non-expanded formulae (from the discourse); (resource-bound was reached)
- ▹ Tableau machine answers user queries (preferred model $\models$ query?)

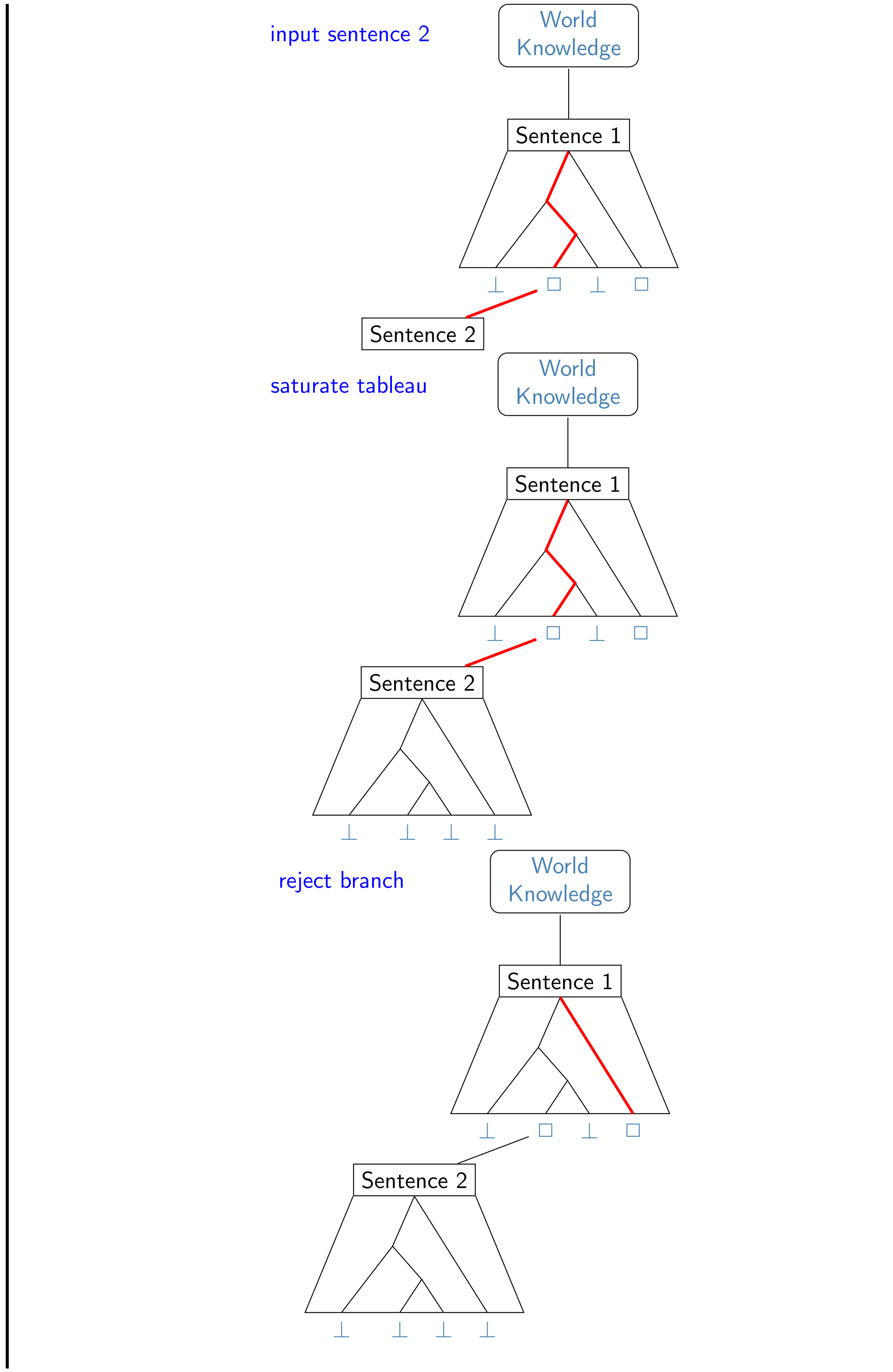
- ▷ Interpretation mode via model generation (guided by resources and strategies)
- ▷ Query mode by refutation theorem proving (□ for side conditions; using tableau rules)

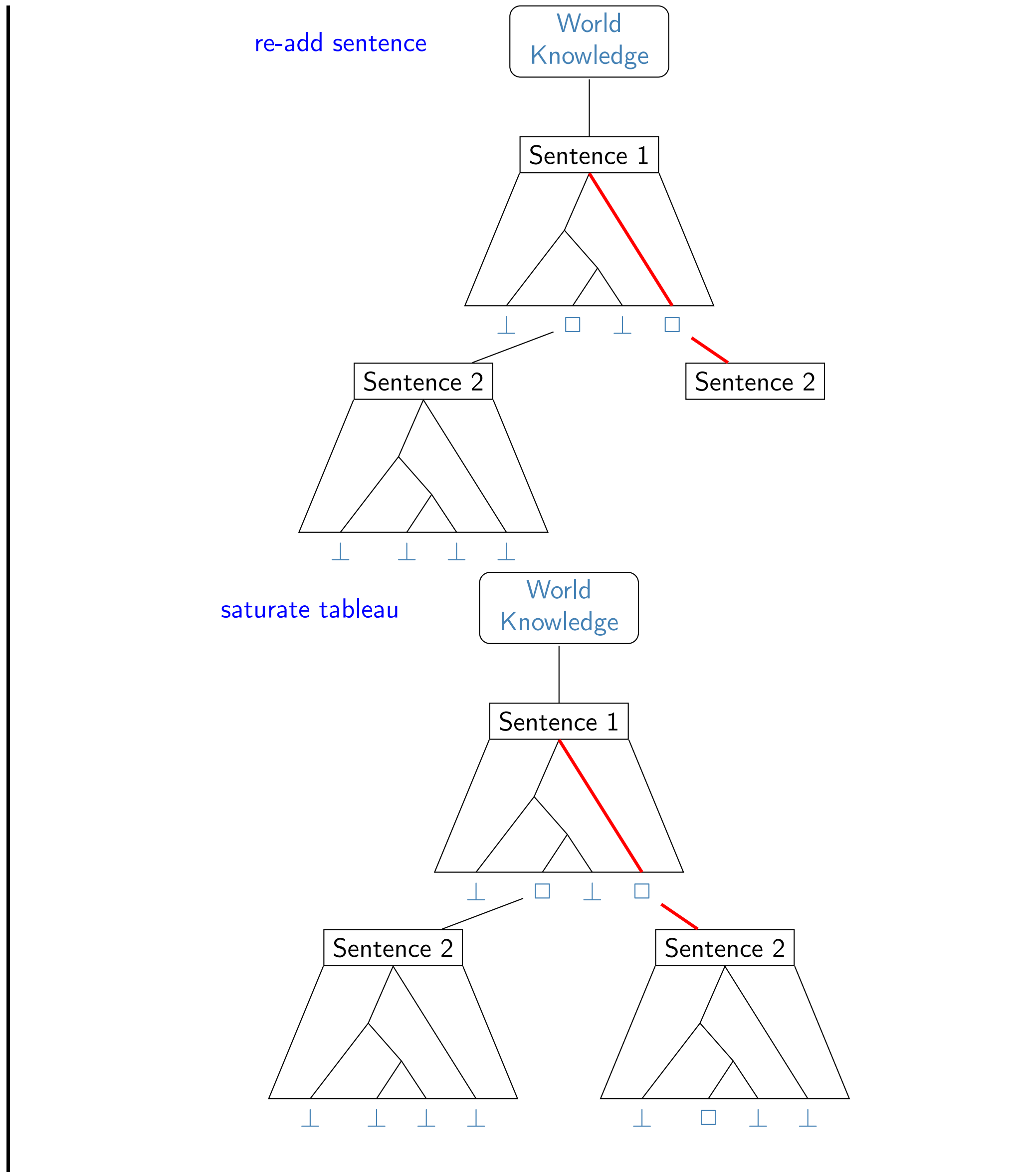
Concretely, we treat discourse understanding as an online process that receives as input the logical forms of the sentences of the discourse one by one, and maintains a tableau that represents the current set of alternative models for the discourse. Since we are interested in the internal state of the 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 preferred model fails.

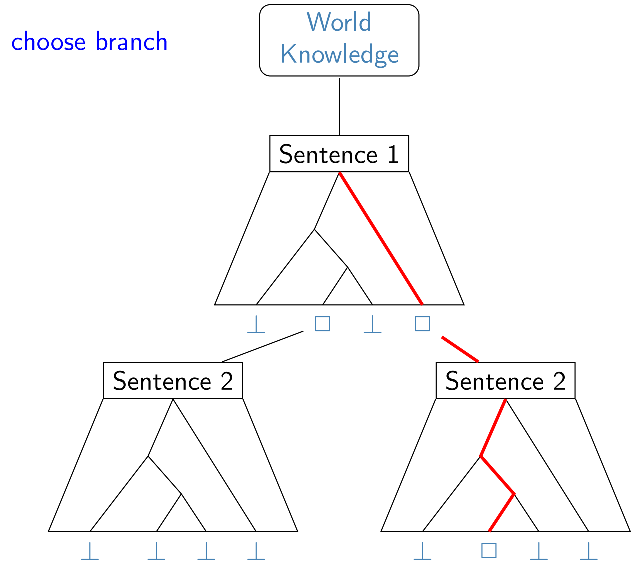
The Tableau Machine in Model Generation Mode

▷ **Example 5.3.8.** The tableau machine in action (query mode on two sentences).




input sentence 2
World Knowledge
Sentence 1
Sentence 2
saturate tableau
World Knowledge
Sentence 1
Sentence 2
reject branch
World Knowledge
Sentence 1
Sentence 2


re-add sentence
World Knowledge
Sentence 1
Sentence 2
Sentence 2
saturate tableau
World Knowledge
Sentence 1
Sentence 2
Sentence 2



Upon input, the tableau machine appends the given logical form as a leaf to the preferred branch. 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.

After discussing the general operation of the tableau machine, let us now come to a concrete linguistic example to see whether it behaves as we expect from a semantic/pragmatic analysis method.

The example we consider below is challenging for most NLU pipelines, since it combines syntactic and pragmatic ambiguity.

Two (Syntactical) Readings

▷ **Example 5.3.9 (A syntactically ambiguous sentence).**
Peter loves Mary and Mary sleeps or Peter snores.

Reading 1: $\mathrm{loves}(\mathrm{peter},\mathrm{mary}) \wedge (\mathrm{sleeps}(\mathrm{mary}) \vee \mathrm{snores}(\mathrm{peter}))$

Reading 2: $\mathrm{loves}(\mathrm{peter},\mathrm{mary}) \wedge \mathrm{sleeps}(\mathrm{mary}) \vee \mathrm{snores}(\mathrm{peter})$

Consider the first reading, start out with the empty tableau for simplicity, even though this is cognitively implausible.

$$\boxed{\mathrm{loves}(\mathrm{peter},\mathrm{mary}) \wedge (\mathrm{sleeps}(\mathrm{mary}) \vee \mathrm{snores}(\mathrm{peter}))}$$
$$\mathrm{loves}(\mathrm{peter},\mathrm{mary})^{\mathsf{T}}$$
$$(\mathrm{sleeps}(\mathrm{mary}) \vee \mathrm{snores}(\mathrm{peter}))^{\mathsf{T}}$$
$$\mathrm{sleeps}(\mathrm{mary})^{\mathsf{T}} \mid \mathrm{snores}(\mathrm{peter})^{\mathsf{T}}$$

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

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 **??**, then we obtain different models.

The other (Syntactical) Reading

$$\begin{array}{c}
\boxed{\mathrm{loves}(\mathrm{peter},\mathrm{mary})\wedge\mathrm{sleeps}(\mathrm{mary})\vee\mathrm{snores}(\mathrm{peter})}\\
\begin{array}{c|c}
(\mathrm{loves}(\mathrm{peter},\mathrm{mary})\wedge\mathrm{sleeps}(\mathrm{mary}))^{\mathsf{T}} & \mathrm{snores}(\mathrm{peter})^{\mathsf{T}}\\
\mathrm{loves}(\mathrm{peter},\mathrm{mary})^{\mathsf{T}} & \\
\mathrm{sleeps}(\mathrm{mary})^{\mathsf{T}} &
\end{array}
\end{array}$$

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 procedure 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 5.3.10.** *Peter does not love Mary.*
Then the second tableau would be extended to

$$\begin{array}{c}
\boxed{\mathrm{loves}(\mathrm{peter},\mathrm{mary})\wedge\mathrm{sleeps}(\mathrm{mary})\vee\mathrm{snores}(\mathrm{peter})}\\
\begin{array}{c|c}
(\mathrm{loves}(\mathrm{peter},\mathrm{mary})\wedge\mathrm{sleeps}(\mathrm{mary}))^{\mathsf{T}} & \mathrm{snores}(\mathrm{peter})^{\mathsf{T}}\\
\mathrm{loves}(\mathrm{peter},\mathrm{mary})^{\mathsf{T}} & \boxed{\neg\mathrm{loves}(\mathrm{peter},\mathrm{mary})}\\
\mathrm{sleeps}(\mathrm{mary})^{\mathsf{T}} & \\
\boxed{\neg\mathrm{loves}(\mathrm{peter},\mathrm{mary})} & \\
\mathrm{loves}(\mathrm{peter},\mathrm{mary})^{\mathsf{F}} & \\
\bot &
\end{array}
\end{array}$$

and the first tableau closes altogether.

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

5.3.3 Adding Equality to PLNQ for Fragment 1

We will now extend $\mathrm{PL}^{\mathrm{nq}}$ by equality, which is a very important relation in natural language – and a liability from $\mathcal{F}_1$: remember the translation rule

$$T7\colon\ [X_{\mathrm{NP}},\mathsf{is},Y_{\mathrm{NP}}]_S \rightsquigarrow X' = Y'$$

which we conveniently forgot because $\mathrm{PL}^{\mathrm{nq}}$ did not have equality? We fix this now.
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 constant)
- **Definition 5.3.11 (Tableau Calculus $\mathcal{T}^{=}_{\mathrm{NQ}}$).** Add two additional inference rules (a positive and a negative) to $\mathcal{T}_0$

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

  where

  - $\mathcal{H} \mathrel{\widehat{=}}$ the Herbrand universe, 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 $\mathbf{A}$.

- **Note:** We could have equivalently written $\mathcal{T}^{=}_{\mathrm{NQ}}\mathrm{sym}$ as $\frac{a = a^{\mathsf{F}}}{\perp}$:

  With $\mathcal{T}^{=}_{\mathrm{NQ}}\mathrm{sym}$ conjure $a = a^{\mathsf{T}}$ from thin air, use it to close $a = a^{\mathsf{F}}$.
- **So, ...** $\mathcal{T}^{=}_{\mathrm{NQ}}\mathrm{sym}$ and $\mathcal{T}^{=}_{\mathrm{NQ}}\mathrm{rep}$ follow the pattern of having a T and a F rule per logical constant.

If we use the simple translation of definite descriptions from fragment $\mathcal{F}_1$, where the phrase *the teacher* translates to a concrete individual constant, then we can interpret (**??**) as (**??**).

Reading Comprehension Example: Mini TOEFL test

- **Example 5.3.12 (Reading Comprehension).** If you hear/read *Mary is the teacher. Peter likes the teacher.*, do you know whether *Peter likes Mary*?
- **Idea:** Interpret via tableau machine (interpretation mode) and test entailment in query mode.
- **Interpretation:** Feed $\Phi_1 := \mathrm{mary} = \mathrm{the\_teacher}$ and $\Phi_2 := \mathrm{likes}(\mathrm{peter}, \mathrm{the\_teacher})$ to the tableau machine in turn. Model generation tableau (nothing to do on these inputs)

$$\boxed{\mathrm{mary} = \mathrm{the\_teacher}}$$
$$\boxed{\mathrm{likes}(\mathrm{peter}, \mathrm{the\_teacher})}$$

- **Question Answering:** Use the tableau machine in query mode for an "entailment test": Label $\varphi := \mathrm{likes}(\mathrm{peter}, \mathrm{mary})$ with F and saturate.

$$\begin{array}{c} \boxed{\mathrm{mary} = \mathrm{the\_teacher}} \\ \boxed{\mathrm{likes}(\mathrm{peter}, \mathrm{the\_teacher})} \\ \left\| \begin{array}{c} \mathrm{likes}(\mathrm{peter}, \mathrm{mary})^{\mathsf{F}} \\ \mathrm{likes}(\mathrm{peter}, \mathrm{the\_teacher})^{\mathsf{F}} \\ \bot \end{array} \right\| \end{array}$$

Indeed, it closes, so $\Phi_1, \Phi_2 \vDash \varphi \rightsquigarrow$ *yes, Peter likes Mary*.

▷ **Note:** The part marked in double vertical lines is removed from the tableau after answering. (do not mess up the tree/models)

5.4 Summary & Evaluation

So let us evaluate what we have achieved in the new, extended fragment.

Fragment $\mathcal{F}_2$ – Summary

▷ Fragment $\mathcal{F}_2$ extends $\mathcal{F}_1$ by pronouns.

▷ Logic/translation extended correspondingly:

- ▷ Equality (actually already needed for $\mathcal{F}_1$)
- ▷ Variables as underspecified representations for anaphoric pronouns.

▷ New NLU component: semantic/pragmatic analysis

- ▷ Tableau machine as an inferential model for pronoun resolution.
- ▷ Uses world knowledge to augment/prune models.

▷ **Coverage:** Still relatively limited (accounts for 1 example from the intro)

Model Generation models Discourse Understanding

▷ The tableau machine algorithm conforms with psycholinguistic findings:

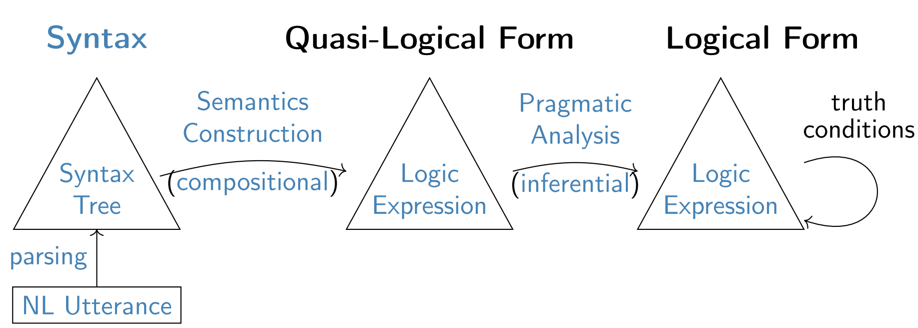
- ▷ Zwaan& Radvansky [ZR98]: listeners not only represent logical form, but also models containing referents.
- ▷ deVega [de 95]: online, incremental process.
- ▷ Singer [Sin94]: enriched by background knowledge.
- ▷ Glenberg et al. [GML87]: major function is to provide basis for anaphor resolution.

Towards a Performance Model for NLU

- **Problem:** The tableau machine is only a competence model.
- **Definition 5.4.1.** A competence model is a meaning theory that delineates a space of possible discourses. A performance model delineates the discourses actually used in communication. (after [Cho65])
- **Idea:** We need to guide the tableau machine in which inferences and branch choices it performs.
- **Idea:** Each tableau rule comes with rule costs.
  - **Here**: each sentence in the discourse has a fixed inference budget.
    Expansion until budget used up.
  - **Ultimately** we want bounded optimization regime [Rus91]:
    Expansion as long as expected gain in model quality outweighs proof costs
- **Effect:** Expensive rules are rarely applied. (only if the promise great rewards)
- ⚠ Finding appropriate values for rule costs and model quality is an open problem.

Summary: The Full Interpretation Process

- **Full Interpretation Process:** In $\mathcal{F}_2$ we have extended the interpretation process by semantic/pragmatic analysis, so we arrive at:



Chapter 6

Fragment 3: Complex Verb Phrases

With the setup of the method of fragments in fragment $\mathcal{F}_2$ and its tableau machine for semantic/pragmatic analysis complete, we now extend it to cover more interesting syntactical structures. The main new feature will be to significantly extend the logical system so that it can cope with the composition problem identified in ??.

6.1 Fragment 3 (Handling Verb Phrases)

$\mathcal{F}_3$: New Data (Verb Phrases)

▷ New Data: in $\mathcal{F}_3$.

1. *Ethel howled and screamed.*
2. *Ethel kicked the dog and poisoned the cat.*
3. *Fiona liked Jo and loathed Ethel and tolerated Prudence.*
4. *Fiona kicked the cat and laughed.*
5. *Bertie didn't laugh.*
6. *Bertie didn't laugh and didn't scream.*
7. *Bertie didn't laugh or scream.*
8. *Bertie didn't laugh or kick the dog.*
9. * *Bertie didn't didn't laugh.*

▷ We extend $\mathcal{F}_2$. (no feature interaction)

The main extension of the fragment is the introduction of the new phrasal category *VP*, we have to interpret.

New Grammar in Fragment $\mathcal{F}_3$ (Verb Phrases)

▷ To account for the syntax we come up with the concept of a verb phrase (VP)

▷ **Definition 6.1.1.** A verb phrase is any phrase that can be used (syntactially) whereever a verb can be.

▷ **Example 6.1.2.** The phrase *tolerated Prudence* is like *slept* (syntactially)

▷ **Idea:** Allow verb phrases (VP in the grammar wherever we had intransitive verbs (V^i) before.

▷ **Problem:** The obvious rule $VP \rightarrow$ didn't VP over-generates: it accepts * *Bertie didn't didn't laugh.* (note the infinitive)

▷ **Definition 6.1.3.** A verb is called finite, iff it contextually complements either an explicit subject or – in the imperative mood – an implicit subject.

▷ **Observation:** Finite verbs are inflected.

▷ **Definition 6.1.4.** Non-finite verbs, are verb forms that do not show tense, person, or number.

▷ **Idea:** We will use features +fin for finite, -fin for non-finite in grammar rules, and ±fin for schemata.

Intuitively, verb phrases denote functions that can be applied to the NP meanings (rule 1 below). Complex VP functions can be constructed from simpler ones by NL coordinators acting as functional operators.

New Grammar in Fragment $\mathcal{F}_3$ (Verb Phrases)

▷ **Definition 6.1.5.** $\mathcal{F}_3$ has the following rules:

S1.	S	!:	$\rightarrow$	NP $VP_{+\text{fin}}$
S2.	S	!:	$\rightarrow$	S conj S
V1.	$VP_{\pm\text{fin}}$	!:	$\rightarrow$	$V^i_{\pm\text{fin}}$
V2.	$VP_{\pm\text{fin}}$	!:	$\rightarrow$	$V^t_{\pm\text{fin}}$ NP
V3.	$VP_{\pm\text{fin}}$	!:	$\rightarrow$	$VP_{\pm\text{fin}}$ conj $VP_{\pm\text{fin}}$
V4.	$VP_{+\text{fin}}$	!:	$\rightarrow$	$BE_{=}$ NP
V5.	$VP_{+\text{fin}}$	!:	$\rightarrow$	BE_{pred} Adj.
V6.	$VP_{+\text{fin}}$	!:	$\rightarrow$	didn't $VP_{-\text{fin}}$

N1.	NP	$\rightarrow$	N_{pr}
N2.	NP	$\rightarrow$	Pron
N3.	NP	$\rightarrow$	the N
L8.	$BE_{=}$	$\rightarrow$	is
L9.	BE_{pred}	$\rightarrow$	is
L10.	$V^i_{-\text{fin}}$	$\rightarrow$	run, laugh,...
L11.	$V^t_{-\text{fin}}$	$\rightarrow$	read, poison,...

▷ **Remark:** The ±fin feature solves the "didn't" over-generation problem.

▷ **Remark:** Many machine-oriented grammars have extensive feature systems like our ±fin.

▷ **Limitations of $\mathcal{F}_3$:**

- ▷ $\mathcal{F}_3$ does not allow coordination of transitive verbs (problematic anyways)
  Prudence kicked and scratched Ethel.

Testing the Grammar on an Example

▷ **Example 6.1.6.**

N_{pr} $V^i_{+\text{fin}}$ conj $V^i_{+\text{fin}}$

Ethel howled and screamed

Towards a Semantics for $\mathcal{F}_3$

- ▷ **Recall:** So far we have mapped intransitive verb (V^i) to predicates which could be applied to NP meanings (individuals).
- ▷ **So:** VP meanings are functions from individuals to truth values
- ▷ **And:** conj meanings are functionals that map functions to functions.
- ▷ In logic we distinguish such objects (individuals and functions of various kinds) by assigning them types.
- ▷ Let's make this formal $\rightsquigarrow$ develop a suitable logic!

6.2 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

- ▷ **Intuition:** Types are semantic annotations for terms that prevent antinomies.
- ▷ **Definition 6.2.1.** Given a set $\mathcal{BT}$ of base types, construct function types: $\alpha \to \beta$ 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 simple types over $\mathcal{BT}$.
- ▷ **Definition 6.2.2.** We will use ι for the type of individuals and o for the type of truth values.
- ▷ **Right Associativity:** The type constructor is used as a right-associative operator, i.e. we use $\alpha \to \beta \to \gamma$ as an abbreviation for $\alpha \to (\beta \to \gamma)$
- ▷ **Vector Notation:** We will use a kind of vector notation for function types, abbreviating $\alpha_1 \to \ldots \to \alpha_n \to \beta$ with $\overline{\alpha}_n \to \beta$.

To strengthen our intuition about the way types can work, we look at the canonical example: Russell's paradox.

What can happen without Types as a Safety-Net

- **Definition 6.2.3.** The unrestricted comprehension principle states that for any sufficiently well-defined property P, there is the set of all and only the objects that have property P.
- **Definition 6.2.4.** Russell's paradox (also known as Russell's antinomy) is a set-theoretic paradox that shows that every set theory that contains an unrestricted comprehension principle leads to contradictions.
- **Definition 6.2.5.** The Russell set R is the set of all sets that are not members of themselves.
- **Observation:** If R is assumed to exist (e.g. by the unrestricted comprehension principle), then we end up with an antinomy:
  - Suppose $R \in R$, then then we must have $R \notin R$, since we have explicitly taken out the set that contain themselves.
  - Suppose $R \notin R$, then have $R \in R$, since all other sets are elements.

  So $R \in R$ iff $R \notin R$, which is a contradiction! (Russell's Antinomy [Rus03])
- **Does Logic help?:**
  - No, if untyped: $R := \{m \mid m \notin m\}$ or equivalently: $R := \{m \mid m\, m\}$.
  - Yes, if typed: $m(m)$ cannot be well-typed with simple types, so we can not define R.
- **Generally:** Simple types prevent self-application: If we type $m(m)$ as $m_\alpha(m_\beta)$, then we must have $\alpha = \beta \to \gamma$ for the function application to work but also $\alpha = \beta$ to have consistent typing.

Here we see the isomorphism between characteristic functions and sets at work again. In the argumentation about preventing harmful self-application.

But let us come back to the work types can do in FragmentThree. In anticipation of a typed target logic, we can associate types to the syntactic categories. Note that different categories can have the same type, which can look confusing at first. But we should take this as a sign that the syntactic analysis of natural language is finer-grained than is needed in knowledge representation and inference for the semantic-pragmatic analysis.

Syntactical Categories and Types

- Now, we can assign types to syntactic 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 $S2$: $S \to S$ conj S

▷ $(\iota \to o) \to (\iota \to o) \to (\iota \to o)$ for VP-coordination in $V3$: $VP \to VP$ conj VP.

▷ **Note:** Computational Linguistics, often uses a different notation for types: e (entity) for ι, t (truth value) for o, and $\langle\alpha,\beta\rangle$ for $\alpha \to \beta$ (no bracket elision convention).

So the type for VP-coordination has the form $\langle\langle e,t\rangle,\langle\langle e,t\rangle,\langle e,t\rangle\rangle\rangle$

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 they 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 $\exists F_{\alpha\to\beta}.\forall X_\alpha.F\ X = \mathbf{A}_\beta$

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 $\mathbf{A} \in wf\!f_\beta(\Sigma_\mathcal{T},\mathcal{V}_\mathcal{T})$ (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 $\mathbf{B} \in wf\!f_\alpha(\Sigma_\mathcal{T},\mathcal{V}_\mathcal{T})$.

▷ **Definition 6.2.6.** Axiom of β equality: $(\lambda X_\alpha.\mathbf{A})\ \mathbf{B} = [\mathbf{B}/X](\mathbf{A}_\beta)$

▷ **Idea:** Introduce a new class of formulae (λ-calculus [Chu40])

In a similar way we can treat (functional) extensionality.

From Extensionality to η-Conversion

- **Definition 6.2.7.** 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 6.2.8.** We say that $\mathbf{A}$ and $\lambda X_\alpha.\mathbf{A}\ X$ are η-equal, (write $\mathbf{A}_{\alpha\to\beta} =_\eta \lambda X_\alpha.\mathbf{A}\ X$), iff $X \notin \mathrm{free}(\mathbf{A})$.
- **Theorem 6.2.9.** *η-equality and Extensionality are equivalent*
- *Proof:* We show that η-equality is special case of extensionality; the converse direction is trivial
  1. Let $\forall X_\alpha.\mathbf{A}X = \mathbf{B}X$, thus $\mathbf{A}X = \mathbf{B}X$ with $\forall E$
  2. $\lambda X_\alpha.\mathbf{A}X = \lambda X_\alpha.\mathbf{B}X$, therefore $\mathbf{A} = \mathbf{B}$ with η
  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: $\forall F_o.\forall G_o.FG \Leftrightarrow F = G$ 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?

6.3 Simply Typed λ-Calculus

In this section we will present a logical system 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)

- **Definition 6.3.1.** Signature $\Sigma_\mathcal{T} = \bigcup_{\alpha\in\mathcal{T}}\Sigma_\alpha$ (includes countably infinite signatures Σ_α^{Sk} of Skolem contants).
- $\mathcal{V}_\mathcal{T} = \bigcup_{\alpha\in\mathcal{T}}\mathcal{V}_\alpha$, such that $\mathcal{V}_\alpha$ are countably infinite.
- **Definition 6.3.2.** We call the set $\mathit{wff}_\alpha(\Sigma_\mathcal{T}, \mathcal{V}_\mathcal{T})$ defined by the rules
  - $\mathcal{V}_\alpha \cup \Sigma_\alpha \subseteq \mathit{wff}_\alpha(\Sigma_\mathcal{T}, \mathcal{V}_\mathcal{T})$
  - If $\mathbf{C} \in \mathit{wff}_{\alpha\to\beta}(\Sigma_\mathcal{T}, \mathcal{V}_\mathcal{T})$ and $\mathbf{A} \in \mathit{wff}_\alpha(\Sigma_\mathcal{T}, \mathcal{V}_\mathcal{T})$, then $\mathbf{C}\ \mathbf{A} \in \mathit{wff}_\beta(\Sigma_\mathcal{T}, \mathcal{V}_\mathcal{T})$
  - If $\mathbf{A} \in \mathit{wff}_\alpha(\Sigma_\mathcal{T}, \mathcal{V}_\mathcal{T})$, then $\lambda X_\beta.\mathbf{A} \in \mathit{wff}_{\beta\to\alpha}(\Sigma_\mathcal{T}, \mathcal{V}_\mathcal{T})$

  the set of well typed formulae of type α over the signature $\Sigma_\mathcal{T}$ and use $\mathit{wff}_\mathcal{T}(\Sigma_\mathcal{T}, \mathcal{V}_\mathcal{T}) := \bigcup_{\alpha\in\mathcal{T}}\mathit{wff}_\alpha(\Sigma_\mathcal{T}, \mathcal{V}_\mathcal{T})$ for the set of all well-typed formulae.

▷ **Definition 6.3.3.** 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. $\sigma(X_\alpha) \in \mathit{wff}_\alpha(\Sigma_\mathcal{T}, \mathcal{V}_\mathcal{T})$ and capture-avoiding.

▷ **Definition 6.3.4 (Simply Typed λ-Calculus).** The simply typed λ calculus $\Lambda^\rightarrow$ over a signature $\Sigma_\mathcal{T}$ has the formulae $\mathit{wff}_\mathcal{T}(\Sigma_\mathcal{T}, \mathcal{V}_\mathcal{T})$ (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: $\lambda X.\mathbf{A}\ X =_\eta \mathbf{A}$ if $X \notin \mathrm{free}(\mathbf{A})$.

The intuitions about functional structure of λ-terms and about free and bound variables are encoded into three transformation rules $\Lambda^\rightarrow$: 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 arguments. 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 $\lambda X_\alpha.\mathbf{A}\ X\mathbf{B} =_\beta \mathbf{A}\ \mathbf{B}$.
We will use a set of bracket elision rules that make the syntax of $\Lambda^\rightarrow$ more palatable. This makes $\Lambda^\rightarrow$ 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)

▷ **Application is left-associative:** We abbreviate $\mathbf{F}\ \mathbf{A}^1\ \mathbf{A}^2\ \ldots\ \mathbf{A}^n$ with $\mathbf{F}\ \mathbf{A}^1 \ldots \mathbf{A}^n$ eliding the brackets and further with $\mathbf{F}\ \overline{\mathbf{A}^n}$ in a kind of vector notation.

▷ **Andrews' dot Notation:** A . stands for a left bracket whose partner is as far right as is consistent with existing brackets; i.e. $\mathbf{A}\ .\mathbf{B}\ \mathbf{C}$ abbreviates $\mathbf{A}\ (\mathbf{B}\ \mathbf{C})$.

▷ **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.

▷ **Outer brackets:** Finally, we allow ourselves to elide outer brackets where they can be inferred.

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 $\mathbf{B}$.[2] In this presentation of the simply typed λ-calculus we build-in $=_\alpha$-equality and use capture-avoiding substitution directly. A clean introduction would followed the steps in section 7 (First-Order Logic) in the LBS lecture notes by introducing substitutions with a substitutability condition like the one in Definition 7.26 (First-Order Substitutions) in the LBS lecture notes, then establishing the soundness of $=_\alpha$ conversion, and only then postulating defining capture-avoiding substitution application as in **??**. 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.

[2] EdNote: rationalize the semantic macros for syntax!

Computationally, λ-calculi 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 $\Lambda^{\to}$ as we will see below.

$=_{\alpha\beta\eta}$-Equality (Overview)

▷ **Definition 6.3.5.**

Reduction with $\left\{\begin{array}{ll} =_\beta : & (\lambda X.\mathbf{A})\,\mathbf{B} \to_\beta [\mathbf{B}/X](\mathbf{A}) \\ =_\eta : & \lambda X.\mathbf{A}\,X \to_\eta \mathbf{A}\end{array}\right.$ under $=_\alpha$: $\begin{array}{c}\lambda X.\mathbf{A} \\ =_\alpha \\ \lambda Y.([Y/X](\mathbf{A}))\end{array}$

The treductions can be applied at top-level (as above), but also in subterms:

If $\mathbf{A} \to_{\alpha\beta\eta} \mathbf{B}$, then $\mathbf{C}\,\mathbf{A} \to_{\alpha\beta\eta} \mathbf{C}\,\mathbf{B}$, $\mathbf{A}\,\mathbf{C} \to_{\alpha\beta\eta} \mathbf{B}\,\mathbf{C}$, and $\lambda X.\mathbf{A} \to_{\alpha\beta\eta} \lambda X.\mathbf{B}$.

▷ **Theorem 6.3.6.** *β-reduction is well-typed, terminating and confluent in the presence of α-conversion.*

▷ **Definition 6.3.7 (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 6.3.8.** *$=_{\beta\eta}$-reduction yields unique normal forms (up to $=_\alpha$-equivalence).*

We will now introduce some terminology to be able to talk about λ terms and their parts.

Syntactic Parts of λ-Terms

▷ **Definition 6.3.9 (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

- ▷ $\mathbf{H}$ 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 6.3.10.** 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 6.3.11.** *The syntactic heads of β-normal forms are constant or variables.*

▷ **Definition 6.3.12.** 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 $\mathrm{head}(\mathbf{A})$. We call a λ-term a j-projection, iff its head is the j^{th} bound variable.

▷ **Definition 6.3.13.** We call a λ-term a η long form, iff its matrix has base type.

▷ **Definition 6.3.14.** η Expansion makes η long forms

$$\eta[\lambda X^1 \ldots X^n.\mathbf{A}] := \lambda X^1 \ldots X^n.\lambda Y^1 \ldots Y^m.\mathbf{A}\,Y^1 \ldots Y^m$$

▷ **Definition 6.3.15.** Long $\beta\eta$ normal form, iff it is β normal and η-long.

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 $\lambda \overline{X^n_\alpha}.\mathbf{B}$, 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.
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 6.3.16.** 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 6.3.17.** 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 6.3.18.** Given a frame $\mathcal{D}_\mathcal{T}$, and a typed function $\mathcal{I} \colon \Sigma \to \mathcal{D}$, we call $\mathcal{I}_\varphi \colon \mathit{wff}_\mathcal{T}(\Sigma_\mathcal{T}, \mathcal{V}_\mathcal{T}) \to \mathcal{D}$ the value function induced by $\mathcal{I}$, iff

1. $\mathcal{I}_\varphi|_{\mathcal{V}_\mathcal{T}} = \varphi,\qquad \mathcal{I}_\varphi|_{\Sigma_\mathcal{T}} = \mathcal{I}$,
2. $\mathcal{I}_\varphi(\mathbf{A}\ \mathbf{B}) = \mathcal{I}_\varphi(\mathbf{A})(\mathcal{I}_\varphi(\mathbf{B}))$, and
3. $\mathcal{I}_\varphi(\lambda X_\alpha.\mathbf{A})$ is that function $f \in \mathcal{D}_{\alpha\to\beta}$, such that $f(a) = \mathcal{I}_{\varphi,[a/X]}(\mathbf{A})$ for all $a \in \mathcal{D}_\alpha$.

▷ **Note:** Not every λ-term has a $\mathcal{I}_\varphi$-value as we have only required $\mathcal{D}_{\alpha\to\beta} \subseteq \mathcal{D}_\alpha \to \mathcal{D}_\beta$ for frames. (there might not be enough functions)

▷ **Definition 6.3.19.** We call $\langle \mathcal{D}, \mathcal{I} \rangle$, where $\mathcal{D}$ is a frame and $\mathcal{I}$ is a typed function comprehension closed or a $\Sigma_\mathcal{T}$-algebra, iff $\mathcal{I}_\varphi \colon \mathit{wff}_\mathcal{T}(\Sigma_\mathcal{T}, \mathcal{V}_\mathcal{T}) \to \mathcal{D}$ is total.

▷ **Theorem 6.3.20.** *$=_{\alpha\beta\eta}$ (seen as a calculus) is sound and complete for Σ-algebras.*

▷ **Upshot for LBS:** $\Lambda^{\to}$ is *the* logical system for reasoning about functions!

The definition of the semantics in **??** is surprisingly simple. The only part that is new at all is the third clause, and there we already know the trick with treating binders by extending the variable assignment from quantifiers in first-order logic.

The real subtlety is in the definition of frames, where instead of requiring $\mathcal{D}_{\alpha\to\beta} = \mathcal{D}_\alpha \to \mathcal{D}_\beta$ (full function universes we have only required $\mathcal{D}_{\alpha\to\beta} \subseteq \mathcal{D}_\alpha \to \mathcal{D}_\beta$, which necessitates the post-hoc definition of a $\Sigma_\mathcal{T}$-algebra. But the added complexity gives us thm.abe-sound-complete.
Excursion: We will discuss the semantics, computational properties, and a more modern presentation of the λ calculus in **??**.

6.4 A Logical System for Fragment 3

Higher-Order Logic without Quantifiers ($\mathrm{HOL}^{\mathrm{nq}}$)

▷ **Problem:** Need a logic like $\mathrm{PL}^{\mathrm{nq}}$, but with λ-terms to interpret $\mathcal{F}_3$ into.

▷ **Idea:** Re-use the syntactical framework of $\Lambda^{\rightarrow}$.

▷ **Definition 6.4.1.** Let $\mathrm{HOL}^{\mathrm{nq}}$ be an instance of $\Lambda^{\rightarrow}$, with $\mathcal{BT} = \{\iota, o\}$, $\wedge \in \Sigma_{o\to o\to o}$, $\neg \in \Sigma_{o\to o}$, and $= \in \Sigma_{\alpha\to\alpha\to o}$ for all types α.

▷ **Idea:** To extend this to a semantics for $\mathrm{HOL}^{\mathrm{nq}}$, we only have to say something about the base type o, and the logical constants $\neg_{o\to o}$, $\wedge_{o\to o\to o}$, and $=_{\alpha\to\alpha\to o}$.

▷ **Definition 6.4.2.** We define the semantics of $\mathrm{HOL}^{\mathrm{nq}}$ by setting

1. $\mathcal{D}_o = \{\mathsf{T}, \mathsf{F}\}$; the set of truth values
2. $\mathcal{I}(\neg) \in \mathcal{D}_{o\to o}$, is the function $\{\mathsf{F} \mapsto \mathsf{T}, \mathsf{T} \mapsto \mathsf{F}\}$
3. $\mathcal{I}(\wedge) \in \mathcal{D}_{o\to o\to o}$ 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. $\mathcal{I}(=) \in \mathcal{D}_{\alpha\to\alpha\to o}$ 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 $\mathrm{PL}^{\mathrm{nq}}$ as our target logic, 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 target logic, $\mathrm{HOL}^{\mathrm{nq}}$, 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 6.4.3 (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 $\mathcal{D}_{\iota\to o}$ 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 $\mathcal{D}_{\iota\to\iota\to o}$. 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.

$\mathrm{HOL}^{\mathrm{nq}}$ is an expressive logical system

▷ $\mathrm{HOL}^{\mathrm{nq}}$ is an expressive logical system

▷ **Example 6.4.4.** We can express set union as a $\mathrm{HOL}^{\mathrm{nq}}$ term:

$$\cup := \lambda P_{\iota\to o}.\lambda Q_{\iota\to o}.\lambda X_\iota.P\ X \vee Q\ X$$

Let us test whether $\{1,2\} \cup \{2,3\}$ really is $\{1,2,3\}$.

Note that we can represent (the characteristic function of) $\{1,2\}$ as the $\mathrm{HOL}^{\mathrm{eq}}$ term $\lambda Z_\iota.Z = 1 \vee Z = 2$. (and the other sets analogously)

So lets represent $\{1,2\} \cup \{2,3\}$ as a $\mathrm{HOL}^{\mathrm{eq}}$ term and β-reduce:

$$\begin{array}{ll} & (\lambda P_{\iota\to o}.\lambda Q_{\iota\to o}.\lambda X_\iota.P\ X \vee Q\ X)\ (\lambda Z_\iota.Z = 1 \vee Z = 2)\ (\lambda Z_\iota.Z = 2 \vee Z = 3) \\ \to_\beta & (\lambda Q_{\iota\to o}.\lambda X_\iota.(\lambda Z_\iota.Z = 1 \vee Z = 2)\ X \vee Q\ X)\ (\lambda Z_\iota.Z = 2 \vee Z = 3) \\ \to_\beta & \lambda X_\iota.(\lambda Z_\iota.Z = 1 \vee Z = 2)\ X \vee (\lambda Z_\iota.Z = 2 \vee Z = 3)\ X \\ \to_\beta & \lambda X_\iota.X = 1 \vee X = 2 \vee X = 2 \vee X = 3 \\ \Leftrightarrow & \lambda X_\iota.X = 1 \vee X = 2 \vee X = 3 \end{array}$$

?? shows the characteristic strength of $\mathrm{HOL}^{\mathrm{eq}}$ as a logical system: The ability of constructing functions via the λ operator allows us to define many of the operators and relations that we would have to declare in a first-order signature in e.g. a first-order logic and then axiomatize so that we can reason about them. The logical connectives and equality we would normally use in the axioms, we can directly use in the operator definitions directly. When these λ-defined operators are applied to arguments, the substitution from β-reduction brings them into the right positions.

6.5 Translation for Fragment 3

Now that we have done the heavy lifting by building our target logic $\mathrm{HOL}^{\mathrm{eq}}$, the translation for $\mathcal{F}_3$ is relatively straightforward. We just have to deal with verb phrases and VP coordination. The first works just as for intransitive verbs in $\mathcal{F}_1$ and for the latter we define custom operators as denotations for the coordinators.

Translations for Fragment $\mathcal{F}_3$

- We will look at the new translation rules: (the rest from $\mathcal{F}_2$ stay the same)

$$\begin{array}{l} T1\colon [X_{\mathrm{NP}}, Y_{VP}]_S \rightsquigarrow VP'(\mathrm{NP}'), \\ T3\colon [X_{VP}, Y_{\mathrm{conj}}, Z_{VP}]_{VP} \rightsquigarrow \mathrm{conj}'(VP', VP'), \\ T4\colon [X_{V^t}, Y_{\mathrm{NP}}]_{VP} \rightsquigarrow V^{t\prime}(\mathrm{NP}') \end{array}$$

- **Note:** We can get away with this because $\mathrm{PL}^{\mathrm{nq}} \subseteq \mathrm{HOL}^{\mathrm{eq}}$ in the target logic.
- The lexical insertion rules will give us two items each for *is*, *and*, and *or*, corresponding to the two types we have given them above.

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	$(\iota \to o) \to \iota \to o$	$\lambda P_{\iota\to o}X_\iota.\neg P\ X$	

- **Note:** All words are translated to $\mathrm{HOL}^{\mathrm{eq}}$ formulae.
- **BTW:** The translation of *or* in VP-coordination is just set union $\widehat{=}$ disjunction lifted to sets. (analogous with *and*, conjunction and intersection)

Translation Example

- It only remains to test $\mathcal{F}_3$ on an example from the original data!
- **Example 6.5.1.** *Ethel howled and screamed* to

$$\begin{aligned} & (\lambda F_{\iota\to o} G_{\iota\to o} X_\iota . F(X) \wedge G(X)) \text{ howls screams ethel} \\ \to_\beta\ & (\lambda G_{\iota\to o} X_\iota . \text{howls}(X) \wedge G(X)) \text{ screams ethel} \\ \to_\beta\ & (\lambda X_\iota . \text{howls}(X) \wedge \text{screams}(X)) \text{ ethel} \\ \to_\beta\ & \text{howls}(\text{ethel}) \wedge \text{screams}(\text{ethel}) \end{aligned}$$

6.6 Summary & Evaluation

So let us evaluate what we have achieved in the new, extended fragment.

Fragment $\mathcal{F}_3$ – Summary

- Fragment $\mathcal{F}_3$ extends $\mathcal{F}_2$ by verb phrases.
- We need a completely new idea for the logic ⇝ need functions to express translation
- Logical system: $\mathrm{HOL}^{\mathrm{nq}} \mathrel{\widehat{=}} \Lambda^{\to} + \mathrm{PL}^0$.
  - $\Lambda^{\to}$ contributes the simple types and functions
  - PL^0 contributes type o and connectives.
- **Coverage:** Better: we can do verb phrase coordination.

6.6.1 Overview/Summary so far

Where we started: A *VP*-less fragment and $\mathrm{PL}^{\mathrm{nq}}$.:

$\mathrm{PL}^{\mathrm{nq}}$	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 $\mathrm{PL}^{\mathrm{nq}}$ 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 $\mathrm{HOL}^{\mathrm{nq}}$.: 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 coordinators which connect *VP*s and other expression types. At this point, the resources of $\mathrm{PL}^{\mathrm{nq}}$ no longer sufficed to provide adequate compositional translations of the fragment. So we introduced a new translation language, $\mathrm{HOL}^{\mathrm{nq}}$. However, the general picture of the table above does not change; only the target logic 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 target logic.
- 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 construct appropriate translations for the differently-typed variants.

Some issues about translations: $\mathrm{HOL}^{\mathrm{nq}}$ provides multiple syntactically and semantically equivalent versions of many of its expressions. For example:

1. Let runs be an $\mathrm{HOL}^{\mathrm{nq}}$ constant of type $\iota \to o$. Then $\mathrm{runs} = \lambda X.\mathrm{runs}(X)$
2. Let loves be an $\mathrm{HOL}^{\mathrm{nq}}$ constant of type $\iota \to \iota \to o$. Then $\mathrm{loves} = \lambda X.\lambda Y.\mathrm{loves}(X, Y)$
3. Similarly, $\mathrm{loves}(a) = \lambda Y.\mathrm{loves}(a, Y)$
4. And $\mathrm{loves}(\mathrm{jane}, \mathrm{george}) = (\lambda X.\lambda Y.\mathrm{loves}(X, Y))\ \mathrm{jane}(\mathrm{george})$

Logically, both sides of the equations are considered equal, since $=_\eta$-equality (remember $\lambda X.\mathbf{A}\ X \to_\eta \mathbf{A}$, if $X \notin \mathrm{free}(\mathbf{A})$) is built into $\mathrm{HOL}^{\mathrm{nq}}$. In fact all the right-hand sides are $=_\eta$-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 coordinator), 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 inference can be performed.

Chapter 7

Fragment 4: Noun Phrases and Quantification

In this chapter 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.

$\mathcal{F}_4$ will require further enrichments of the translation language. Our next steps are:

- Analysis of NP.
- Treatment of adjectives and adverbs.
- Quantification and definite description

7.1 Fragment 4

As always we start off a new fragment by looking at the new data we want to cover.

New Data in Fragment $\mathcal{F}_4$ (more Noun Phrases)

▷ In $\mathcal{F}_4$ we want to extend $\mathcal{F}_3$ so it can deal with the following sentences: (without the "the-NP" trick)

1. *Peter loved the cat.*, but not * *Peter loved the the cat.*
2. *John killed a cat with a white tail.*
3. *Peter chased the gangster in the red sportscar.*
4. *Peter loves every cat.*
5. *Every man loves a woman.*
6. *The quick brown fox jumps over the lazy dog.*
7. *The very heavy boat sank quickly.*

FAU Michael Kohlhase: LBS 122 2025-02-06

The first example sugests that we need a full and uniform treatment of determiners like *the*, *a*, and *every*. The second and third introduces a new phenomenon: prepositional phrases like *with a hammer/mouse*; these are essentially nominal phrases that modify the meaning of other phrases via a preposition like *with*, *in*, *on*, *at*. These two show that the prepositional phrase can modify the verb or the object.

New Grammar in Fragment $\mathcal{F}_4$ (Common Noun Phrases)

▷ To account for the syntax we extend the functionality of noun phrases from $\mathcal{F}_1$.

▷ **Definition 7.1.1.** $\mathcal{F}_4$ adds the rules on the right to $\mathcal{F}_3$ (on the left):

$S1\colon S \to \mathrm{NP}\ VP_{+\mathsf{fin}}$,	$S3\colon S \to S\ \mathrm{PP}$,
$S2\colon S \to S\ \mathrm{conj}\ S$,	$N3\colon \mathrm{NP} \to \mathrm{Det}\ \mathrm{CNP}$,
$V1\colon VP_{\pm\mathsf{fin}} \to V^i_{\pm\mathsf{fin}}$,	$N4\colon \mathrm{CNP} \to N$,
$V2\colon VP_{\pm\mathsf{fin}} \to V^t_{\pm\mathsf{fin}}\ \mathrm{NP}$,	$N5\colon \mathrm{CNP} \to \mathrm{CNP}\ \mathrm{PP}$,
$V4\colon VP_{+\mathsf{fin}} \to BE_{=}\ \mathrm{NP}$,	$N6\colon \mathrm{CNP} \to \mathrm{Adj}\ \mathrm{CNP}$,
$V5\colon VP_{+\mathsf{fin}} \to BE_{pred}\ \mathrm{Adj}$,	$P1\colon \mathrm{PP} \to P\ \mathrm{NP}$,
$V6\colon VP_{+\mathsf{fin}} \to \text{didn't}\ VP_{-\mathsf{fin}}$,	$V3'\colon VP_{\pm\mathsf{fin}} \to VP_{\pm\mathsf{fin}}\ \mathrm{VPconj}_{\pm\mathsf{fin}}$,
$N1\colon \mathrm{NP} \to N_{\mathrm{pr}}$,	$V7\colon \mathrm{VPconj}_{\pm\mathsf{fin}} \to \mathrm{conj}\ VP_{\pm\mathsf{fin}}$,
$N2\colon \mathrm{NP} \to \mathrm{Pron}$	$V8\colon VP_{+\mathsf{fin}} \to VP_{+\mathsf{fin}}\ \mathrm{Adv}$,
	$V9\colon VP_{\pm\mathsf{fin}} \to VP_{\pm\mathsf{fin}}\ \mathrm{PP}$,
	$L1\colon P \to \text{with} \mid \text{of} \mid \ldots$

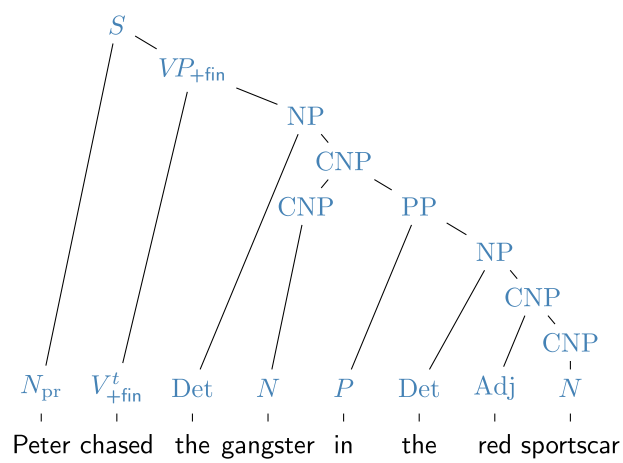
▷ **Definition 7.1.2.** A common noun is a noun that describes a type, for example *woman*, or *philosophy* rather than an token, such as *Amelia Earhart* (proper name).

Note: Again, we assume appropriate lexical insertion rules without specification.

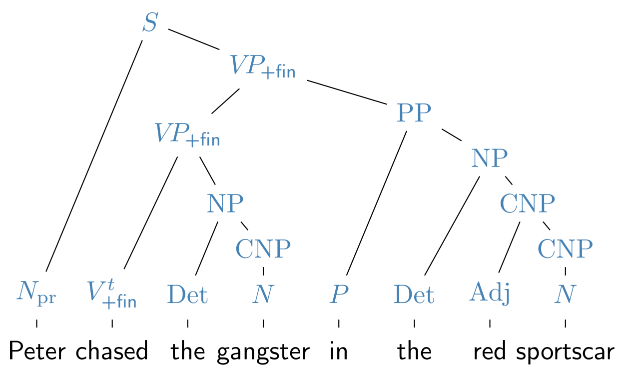
Testing the $\mathcal{F}_4$ Syntax on an example

▷ **Example 7.1.3.** Can we capture the (syntactic) attachment ambiguity in

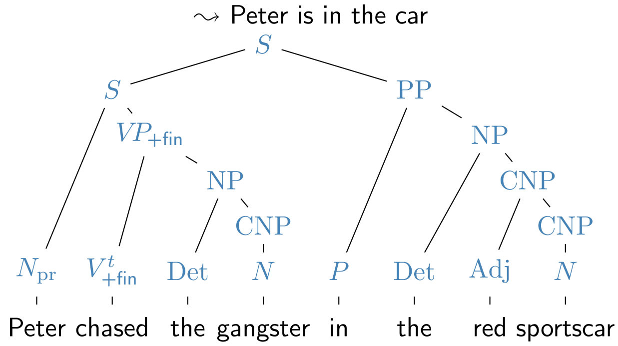
Peter chased the gangster in the red sportscar.



⇝ The gangster is in the car



$\rightsquigarrow$ Peter is in the car



$\rightsquigarrow$ Both Peter and the gangster are in the car

7.2 A Target Logic for Fragment 4

Now that we have fixed $\mathcal{F}_4$ and have an idea of the syntactical categories, we have to take a look at the target logic. We will first take stock of what we need and then develop the necessary logic technology.

Higher-Order Logic with Descriptions

- **Plan:** We need to extend $\mathrm{HOL}^{\mathrm{nq}}$ with
  - quantifiers so we can treat *Every student sleeps*
  - a logical operator for definite descriptions, e.g. *the teacher sleeps*

  We will call this logic Higher-Order Logic with Descriptions (quantifiers taken for granted)
- **Note:** Quantifiers can be added to any logic: Extend the
  - syntax by variables and a new binding symbol (language-level)
  - semantics by a new clause for the value function
  - calculi by new quantifier introduction/elimination rules

  Quite tedious compared to simply adding a new logical constant!

▷ **Note:** The description operator will have to have type $(\iota \to o) \to \iota$, as the denotation of *teacher* has type $\iota \to o$ and *the teacher* has type ι. (like *Mary*)

7.2.1 Quantifiers and Equality in Higher-Order Logic

As a first step towards our target logic, we will now introduce a higher-order logic with quantifiers and equality building on $\mathrm{HOL}^{\mathrm{nq}}$ as a logical system without concern for linguistic issues. We will call this system $\mathrm{HOL}^{\to}$.

Actually, there are two (equivalent) ways of developing $\mathrm{HOL}^{\to}$: we can either add quantifiers and define equality using them or we can take equality as primitive and define all connectives and quantifiers from that. The latter shows that $\mathrm{HOL}^{\mathrm{nq}}$ and $\mathrm{HOL}^{\to}$ are equally expressive – and the extension does not add anything in theory.
There is a more elegant way to treat quantifiers than extending language, semantics, and inference systems in $\mathrm{HOL}^{\to}$. It builds on the realization that the λ-abstraction is the only 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 binding operator: λ, use that to treat quantification.

▷ **Definition 7.2.1.** We add two new logical constants Π^α and Σ^α for each type α:

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)

▷ **Definition 7.2.2.** Regain traditional quantifiers as abbreviations:

$$(\forall X_\alpha.\mathbf{A}) := \Pi^\alpha\ (\lambda X_\alpha.\mathbf{A}) \qquad (\exists X_\alpha.\mathbf{A}) := \Sigma^\alpha\ (\lambda X_\alpha.\mathbf{A})$$

▷ **Observation:** Indeed: $\mathcal{I}_\varphi(\forall X_\iota.\mathbf{A}) = \mathcal{I}_\varphi(\Pi^\iota\ (\lambda X_\iota.\mathbf{A})) = \mathcal{I}(\Pi^\iota)(\mathcal{I}_\varphi(\lambda X_\iota.\mathbf{A})) = \mathsf{T}$ iff $\mathcal{I}_\varphi(\lambda X_\iota.\mathbf{A})(a) = \mathcal{I}_{[a/X],\varphi}(\mathbf{A}) = \mathsf{T}$ for all $a \in \mathcal{D}_\alpha$.

▷ **Definition 7.2.3.** We call this approach to binding operators higher-order abstract syntax (HOAS).

In $\mathrm{HOL}^{\to}$, where we have quantifiers, we can define an operator for equality using Leibniz' indiscernibility criterion. According to this, two objects are equal, iff they do not have any properties that can be used to tell them apart. As we can quantify over properties – which can be expressed as variables of type $\alpha \to o$ – in $\mathrm{HOL}^{\to}$ we can directly express the principle and β-abstract it into a predicate.

Equality

▷ **Definition 7.2.4 (Leibniz equality).** $\mathbf{Q}^\alpha \mathbf{A}_\alpha \mathbf{B}_\alpha = \forall P_{\alpha \to o}.P\mathbf{A} \Leftrightarrow P\mathbf{B}$ (Leibniz' indiscernibility of identicals)

- **Note:** $\forall P_{\alpha\to o}.P\mathbf{A} \Rightarrow P\mathbf{B}$ (get the other direction by instantiating P with Q, where $QX \Leftrightarrow \neg PX$)
- **Theorem 7.2.5.** *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 $\mathcal{D}_\alpha$.*
- **Definition 7.2.6 (Notation).** 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:*
  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 $r \in \mathcal{D}_{\alpha\to o}$.
  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 $\mathcal{I}_{\varphi,[r/P]}(P\mathbf{B}) = r(\mathcal{I}_\varphi(\mathbf{B})) = \mathsf{T}$.
  3. Thus $\mathcal{I}_\varphi(\mathbf{QAB}) = \mathsf{T}$.
  4. Let $\mathcal{I}_\varphi(\mathbf{A}) \neq \mathcal{I}_\varphi(\mathbf{B})$ and $r = \{\mathcal{I}_\varphi(\mathbf{A})\} \in \mathcal{D}_{\alpha\to o}$ (exists in a standard model)
  5. so $r(\mathcal{I}_\varphi(\mathbf{A})) = \mathsf{T}$ and $r(\mathcal{I}_\varphi(\mathbf{B})) = \mathsf{F}$
  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 $\mathcal{I}_{\varphi,[r/P]}(P\mathbf{B}) = r(\mathcal{I}_\varphi(\mathbf{B})) = \mathsf{F}$.

As we can see, we can even prove that the denotation of Leibniz equality expressed in $\mathrm{HOL}^{\to}$ is the identity relation on the respective universe.

Alternative: HOL^{∞}

- **Definition 7.2.7.** There is only one logical constant in HOL^{∞}: $q^\alpha \in \Sigma_{\alpha\to\alpha\to o}$ with $\mathcal{I}(q^\alpha)(a,b) = \mathsf{T}$, iff $a = b$.

  We define the rest as below: 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.

In a way, this development of higher-order logic is more foundational, especially in the context of Henkin semantics. There, **??** does not hold (see [And72] for details). Indeed the proof of **??** needs the existence of singletons, which can be shown to be equivalent to the existence of the identity relation. In other words, Leibniz equality only denotes the equality relation, if we have an equality relation in the models. However, the only way of enforcing this (remember that Henkin models only guarantee functions that can be explicitly written down as λ-terms) is to add a logical constant for equality to the signature.

7.2.2 A Logic for Definite Descriptions

The next extension is a description operator. Again, we will develop the target logic from a logical systems perspective before we come to linguistic or inferential aspects.

Semantics of Definite Descriptions

- **Problem:** We need the meaning 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 7.2.8.** 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 $\{a\}$, and is otherwise undefined. (remember that we can interpret predicates as sets)
- **Axioms for ι:**

$$\forall X_\alpha . X = \iota \;=\; X$$
$$\forall P, Q . Q(\iota\; P) \wedge (\forall X, Y . P(X) \wedge P(Y) \Rightarrow X = Y) \Rightarrow (\forall . P(Z) \Rightarrow Q(Z))$$

Note: The first axiom is an equational characterization of ι. It uses the fact that the singleton with member X can be written as $=\; X$ (or $\lambda Y. = XY$, which is $=_\eta$-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 $\mathrm{HOL}^{\to}$.
Actually, the description operator is just one of a set of similar operators. We will look at them together to get a better intution.

More Operators and Axioms for $\mathrm{HOL}^{\to}$

- **Definition 7.2.9.** The unary conditional $\mathbf{w}^\alpha \in \Sigma_{o\to\alpha\to\alpha}$
  $\mathbf{w}\;(\mathbf{A}_o)\mathbf{B}_\alpha$ means: "If $\mathbf{A}$, then $\mathbf{B}$".
- **Definition 7.2.10.** The binary conditional $\mathbf{if}^\alpha \in \Sigma_{o\to\alpha\to\alpha\to\alpha}$
  $\mathbf{if}\;(\mathbf{A}_o)\;(\mathbf{B}_\alpha)\;(\mathbf{C}_\alpha)$ means: "if $\mathbf{A}$, then $\mathbf{B}$ else $\mathbf{C}$".
- **Definition 7.2.11.** The description operator $\iota^\alpha \in \Sigma_{(\alpha\to o)\to\alpha}$
  if $\mathbf{P}$ is a singleton set, then $\iota\;(\mathbf{P}_{\alpha\to o})$ is the (unique) element in $\mathbf{P}$.
- **Definition 7.2.12.** The choice operator $\gamma^\alpha \in \Sigma_{(\alpha\to o)\to\alpha}$
  if $\mathbf{P}$ is non-empty, then $\gamma\;(\mathbf{P}_{\alpha\to o})$ is an arbitrary element from $\mathbf{P}$.
- **Definition 7.2.13 (Axioms for these Operators).**

- unary conditional: $\forall \varphi_o.\forall X_\alpha.\varphi \Rightarrow \mathbf{w}\ \varphi X = X$
- binary conditional: $\forall \varphi_o.\forall X_\alpha, Y_\alpha, Z_\alpha.(\varphi \Rightarrow \mathbf{if}\ \varphi\ X\ Y = X) \wedge (\neg\varphi \Rightarrow \mathbf{if}\ \varphi\ Z\ X = X)$
- description operator $\forall P_{\alpha\to o}.(\exists^1 X_\alpha.PX) \Rightarrow (\forall Y_\alpha.PY \Rightarrow \iota\ P = Y)$
- choice operator $\forall P_{\alpha\to o}.(\exists X_\alpha.PX) \Rightarrow (\forall Y_\alpha.PY \Rightarrow \gamma\ P = Y)$

▷ **Idea:** These operators ensure a much larger supply of functions in Henkin models.

More on the Description Operator

▷ ι is a weak form of the choice operator. (only works on singletons)

▷ Alternative Axiom of Descriptions: $\forall X_\alpha.\iota^\alpha\ = \ X = X$.

- use that $\mathcal{I}_{[a/X]}(=\ X) = \{a\}$
- we only need this for base types $\neq o$
- Define $\iota^o\ :==\ (\lambda X_o.X)$ or $\iota^o\ := \lambda G_{o\to o}.G\ T$ or $\iota^o\ :== =\ T$
- $\iota^{(\alpha\to\beta)}\ := \lambda H_{(\alpha\to\beta)\to o} X_\alpha.\iota^\beta\ (\lambda Z_\beta.(\exists F_{\alpha\to\beta}.H\ F \wedge F\ X = Z))$

7.3 Translation for Fragment 4

Now we can finally come to the linguistic aspects of $\mathcal{F}_4$ and in particular the translation.
If we assume that $\forall X.\mathrm{boy}(X) \Rightarrow \mathrm{runs}(X)$ is an adequate translation of *Every boy runs*, and $\exists X.\mathrm{boy}(X) \wedge \mathrm{runs}(X)$ one for *Some boy runs*, then we obtain the translations of the determiners by straightforward $=_\beta$-expansion.

Translation of Determiners and Quantifiers

▷ **Idea:** We establish the meaning of quantifying determiners by $=_\beta$-expansion.

1. assume that we are translating into a λ-calculus with quantifiers and that
   - $\forall X.\mathrm{boy}(X) \Rightarrow \mathrm{runs}(X)$ translates *Every boy runs*, and
   - $\exists X.\mathrm{boy}(X) \wedge \mathrm{runs}(X)$ for *Some boy runs*
2. $\forall\!\!\!\forall := \lambda P_{\iota\to o} Q_{\iota\to o}.(\forall.P(X) \Rightarrow Q(X))$ for *every*. (subset relation)
3. $\exists\!\!\!\exists := \lambda P_{\iota\to o} Q_{\iota\to o}.(\exists.P(X) \wedge Q(X))$ for *some*. (non-empty 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)

▷ **Definition 7.3.1.**

We translate the word 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 7.3.2.** This translates *The pope spoke* to $\tau(\mathrm{pope}, \mathrm{speaks})$, which $=_\beta$-reduces to $\mathrm{speaks}(\iota\ \mathrm{pope})$.

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).

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 $\leadsto$ Adj' or
- ▹ $9'$: Adj $\leadsto$ $(\lambda P_{\iota\to o} X_\iota.P(X) \wedge Adj'(X))$

▷ If *Adj* is a non-intersective adjective, then Adj' is a constant of type $(\iota \to o) \to \iota \to o$ whose denotation is given the interpretation by $\mathcal{I}$ and

- ▹ 10: Adj $\leadsto$ Adj'.

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 description. 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 $(\iota \to o) \to o$ (and are applied to the VP) whereas subject NPs with proper names have type ι. (argument to the VP)

▷ **Idea:** *John runs* translates to $\mathrm{runs}(\mathrm{john})$, where $\mathrm{runs} \in \Sigma_{\iota\to o}$ and $\mathrm{john} \in \Sigma_\iota$.

Now we $=_\beta$-expand over the VP yielding $(\lambda P_{\iota\to o}.P(\mathrm{john}))\ \mathrm{runs}$

$\lambda P_{\iota\to o}.P(\mathrm{john})$ has type $(\iota \to o) \to o$ and can be applied to the VP runs.

▷ **Definition 7.3.3.** If $c \in \Sigma_\alpha$, then type raising c yields $\lambda P_{\alpha\to o}.P\ c$.

Definite NPs

- **Problem:** On our current assumptions, $the' = \iota$, and so for any definite NP *the N*, its translation is $\iota\ N$, an expression of type ι.
- **Idea:** Type lift just as we did with proper names: $\iota\ N$ 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 $\mathsf{a} \in \mathcal{D}_{\iota\to o}$: $\mathcal{I}(the')(\mathsf{a}) = \mathcal{I}(every')(\mathsf{a})$ if $\#(\mathsf{a}) = 1$, undefined otherwise

  So the' is that function in $\mathcal{D}_{(\iota\to o)\to(\iota\to o)\to o}$ such that 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

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*. loves 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).

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

$$\circ\ g\ f \to_\beta \lambda X.g(f(X)) \qquad \text{and} \qquad \circ\ g\ f\ a \to_\beta g(f(a))$$

In our example, we have

$$\begin{aligned} \circ\ (\lambda P.P(\text{john}))\ \text{loves} &=_{Def} (\lambda FGX.F(G(X)))\ (\lambda P.P(\text{john}))\ \text{loves} \\ &\to_\beta (\lambda GX.(\lambda P.P(\text{john}))\ G(X))\ \text{loves} \\ &\to_\beta \lambda X.(\lambda P.P(\text{john}))\ \text{loves}\ X \\ &\to_\beta! \lambda X.\text{loves}(X, \text{john}) \end{aligned}$$

Definition 7.3.4 (Function Composition). Let $f\colon A \to B$ and $g\colon B \to C$ be functions, then we call the function $h\colon A \to C$ 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").

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 ∀ or ∃ 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.

$$\#(\mathcal{I}_\varphi(\text{boy}) \cap \mathcal{I}_\varphi(\text{runs})) > \#(\mathcal{I}_\varphi(\text{boy}) \backslash \mathcal{I}_\varphi(\text{runs}))$$

▷ **Definition 7.3.5.** $\#(A) > \#(B)$, iff there is no surjective function from B to A, so we can define

$$most' := \lambda AB.\neg(\exists F.\forall X.A(X) \wedge \neg B(X) \Rightarrow (\exists.A(Y) \wedge B(Y) \wedge X = F(Y)))$$

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:

$$\{\langle X, Y\rangle \mid \#(X \cap Y) > \#(X \backslash Y)\}$$

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 $\{\langle X, Y\rangle \mid X \subseteq Y\}$
2. *some* denotes $\{\langle X, Y\rangle \mid X \cap Y \neq \emptyset\}$

▷ The denotations can be given in equivalent function terms, as demonstrated above with the denotation of *most*.

7.4 Inference for Fragment 4

In $\mathcal{F}_4$ we have extended the target logic with quantifiers and description operators of any type. But if we look at the results of the results semantics construction on the examples we see that these are first-order with descriptions only.

As a consequence, we can get by with modest extensions of the first-order model generation calculi we have used for the tableau machine in semantic/pragmatic analysis. We will develop these separately for the quantifiers and descriptions now.

7.4.1 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 (The RM Calculus [Kon04])

- **Idea:** Try to generate domain-minimal (i.e. fewest individuals) Herbrand models (for NL interpretation)
- **Problem:** Even one function constant makes Herbrand universe infinite (solution: leave them out)
- **Definition 7.4.1.** RM adds ground quantifier rules to propositional tableau calculus

$$\frac{(\forall X.\mathbf{A})^{\mathsf{T}} \quad c \in \mathcal{H}}{([c/X](\mathbf{A}))^{\mathsf{T}}}\ RM{\forall}$$

$$\frac{(\forall X.\mathbf{A})^{\mathsf{F}} \quad \mathcal{H} = \{a_1, \ldots, a_n\} \quad w \notin \mathcal{H} \text{ new}}{([a_1/X](\mathbf{A}))^{\mathsf{F}} \;\Big|\; \ldots \;\Big|\; ([a_n/X](\mathbf{A}))^{\mathsf{F}} \;\Big|\; ([w/X](\mathbf{A}))^{\mathsf{F}}}\ RM{\exists}$$

  - $RM\exists$ rule introduces new witness constant w to the branch Herbrand universe $\mathcal{H}$: the set of all individual constants on the branch.
  - Apply $RM\forall$ exhaustively (for new w reapply all $RM\forall$ rules on branch!)

The rule $RM\forall$ allows to instantiate the scope of the quantifier with all the instances of the Herbrand universe, whereas the rule $RM\exists$ makes a case distinction between the cases that the scope holds for one of the already known individuals (those in the Herbrand universe) 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 $RM\forall$ 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 $RM\forall$ rule for any new constant

▷ **Example 7.4.2.** This leads to the generation of infinite models

$$
\begin{array}{c}
(\forall x.\neg x > x \wedge \ldots)^{\mathsf{T}} \\
N(0)^{\mathsf{T}} \\
(\forall x.N(x) \Rightarrow (\exists y.N(y) \wedge y > x))^{\mathsf{T}} \\
(N(0) \Rightarrow (\exists y.N(y) \wedge y > 0))^{\mathsf{T}} \\
\begin{array}{c|c}
\begin{array}{c} N(0)^{\mathsf{F}} \\ \bot \end{array} &
\begin{array}{c}
(\exists y.N(y) \wedge y > 0)^{\mathsf{T}} \\
\begin{array}{c|c}
\begin{array}{c} 0 > 0^{\mathsf{T}} \\ N(0)^{\mathsf{T}} \\ 0 > 0^{\mathsf{F}} \\ \bot \end{array} &
\begin{array}{c}
N(1)^{\mathsf{T}} \\
1 > 0^{\mathsf{T}} \\
(N(1) \Rightarrow (\exists y.N(y) \wedge y > 1))^{\mathsf{T}} \\
\begin{array}{c|c}
\begin{array}{c} N(1)^{\mathsf{F}} \\ \bot \end{array} &
\begin{array}{c}
(\exists y.N(y) \wedge y > 1)^{\mathsf{T}} \\
\begin{array}{c|c|c}
\begin{array}{c} N(0)^{\mathsf{T}} \\ 0 > 1^{\mathsf{T}} \\ \vdots \\ \bot \end{array} &
\begin{array}{c} N(1)^{\mathsf{T}} \\ 1 > 1^{\mathsf{T}} \\ 1 > 1^{\mathsf{F}} \\ \bot \end{array} &
\begin{array}{c} N(2)^{\mathsf{T}} \\ 2 > 1^{\mathsf{T}} \\ \vdots \end{array}
\end{array}
\end{array}
\end{array}
\end{array}
\end{array}
\end{array}
\end{array}
\end{array}
$$

Michael Kohlhase: LBS 142 2025-02-06

The rules $RM\,\forall$ and $RM\,\exists$ may remind you of the rules we introduced for $\mathrm{PL}^{\mathrm{nq}}(\mathcal{V})$ in $\mathcal{F}_2$. In fact the rules mainly differ in their scoping behavior. We will use $RM\,\forall$ as a drop-in replacement for the world-knowledge rule $\mathcal{T}^p_{\mathcal{V}}WK$, and express world knowledge as universally quantified sentences. The rules $\mathcal{T}^p_{\mathcal{V}}Ana$ and $RM\,\exists$ 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 $RM\,\exists$ to deal with rule-like world knowledge.

Let us test the new calculus on a couple of linguistically motivated examples. We start very simple: with a discourse of two sentences, where the second has a quantifier.

Example: *Peter is a man. No man walks*

▷ **Example 7.4.3 (Model generation with quantifiers).**
Peter is a man. No man walks

$$
\begin{array}{c}
\boxed{\mathrm{man}(\mathrm{peter})} \\
\boxed{\neg(\exists X.\mathrm{man}(X) \wedge \mathrm{walks}(X))} \\
(\exists X.\mathrm{man}(X) \wedge \mathrm{walks}(X))^{\mathsf{F}} \\
(\forall X.\neg\mathrm{man}(X) \vee \neg\mathrm{walks}(X))^{\mathsf{T}} \\
(\neg\mathrm{man}(\mathrm{peter}) \vee \neg\mathrm{walks}(\mathrm{peter}))^{\mathsf{T}} \\
\begin{array}{c|c}
\neg\mathrm{man}(\mathrm{peter})^{\mathsf{T}} & \neg\mathrm{walks}(\mathrm{peter})^{\mathsf{T}} \\
\mathrm{man}(\mathrm{peter})^{\mathsf{F}} & \mathrm{walks}(\mathrm{peter})^{\mathsf{F}} \\
\bot &
\end{array}
\end{array}
$$

Herbrand valuation: $\{\mathrm{man}(\mathrm{peter})^{\mathsf{T}}, \mathrm{walks}(\mathrm{peter})^{\mathsf{F}}\}$

Michael Kohlhase: LBS 143 2025-02-06

The next example is a bit more interesting: We have an anaphor that needs to be resolved.

Anaphor Resolution *A man sleeps. He snores*

▷ **Example 7.4.4 (Anaphor Resolution).** *A man sleeps. He snores*

$$\boxed{\exists X.\mathrm{man}(X) \wedge \mathrm{sleeps}(X)}$$
$$(\mathrm{man}(c^1) \wedge \mathrm{sleeps}(c^1))^{\mathsf{T}}$$
$$\mathrm{man}(c^1)^{\mathsf{T}}$$
$$\mathrm{sleeps}(c^1)^{\mathsf{T}}$$
$$\boxed{\exists Y.\mathrm{man}(Y) \wedge \mathrm{snores}(Y)}$$

$(\mathrm{man}(c^1) \wedge \mathrm{snores}(c^1))^{\mathsf{T}}$	$(\mathrm{man}(c^2) \wedge \mathrm{snores}(c^2))^{\mathsf{T}}$
$\mathrm{man}(c^1)^{\mathsf{T}}$	$\mathrm{man}(c^2)^{\mathsf{T}}$
$\mathrm{snores}(c^1)^{\mathsf{T}}$	$\mathrm{snores}(c^2)^{\mathsf{T}}$
minimal	*deicbtic*

Anaphora with World Knowledge

- **Example 7.4.5.** *Mary is married to Jeff. Her husband is not in town.* (slightly outside $\mathcal{F}_2$)
  In PL^1: $\mathrm{married}(\mathrm{mary}, \mathrm{jeff})$, and
  $$\exists W_{\mathtt{Male}}, W'_{\mathtt{Female}}.\mathrm{husband}(W, W') \wedge \neg\mathrm{intown}(W)$$
- World knowledge
  - If woman X is married to man Y, then Y is the only husband of X.
  - $\forall X_{\mathtt{Female}}, Y_{\mathtt{Male}}.\mathrm{married}(X, Y) \Rightarrow \mathrm{husband}(Y, X) \wedge (\forall Z.\mathrm{husband}(Z, X) \Rightarrow (Z = Y))$
- Model generation gives tableau where all open branches contain
  $$\{\mathrm{married}(\mathrm{mary}, \mathrm{jeff})^{\mathsf{T}}, \mathrm{husband}(\mathrm{jeff}, \mathrm{mary})^{\mathsf{T}}, \mathrm{intown}(\mathrm{jeff})^{\mathsf{F}}\}$$
- **Differences:** Additional negative facts e.g. $\mathrm{married}(\mathrm{mary}, \mathrm{mary})^{\mathsf{F}}$.

A branch without World Knowledge

$$\mathrm{married}(\mathrm{mary}, \mathrm{jeff})^{\mathsf{T}}$$
$$(\exists Z_{\mathtt{Male}}, Z'_{\mathtt{Female}}.\mathrm{husband}(Z, Z') \wedge \neg\mathrm{intown}(Z))^{\mathsf{T}}$$
$$(\exists Z'.\mathrm{husband}(c^1{}_{\mathtt{Male}}, Z') \wedge \neg\mathrm{intown}(c^1{}_{\mathtt{Male}}))^{\mathsf{T}}$$
$$(\mathrm{husband}(c^1{}_{\mathtt{Male}}, \mathrm{mary}) \wedge \neg\mathrm{intown}(c^1{}_{\mathtt{Male}}))^{\mathsf{T}}$$
$$\mathrm{husband}(c^1{}_{\mathtt{Male}}, \mathrm{mary})^{\mathsf{T}}$$
$$\neg\mathrm{intown}(c^1{}_{\mathtt{Male}})^{\mathsf{T}}$$
$$\mathrm{intown}(c^1{}_{\mathtt{Male}})^{\mathsf{F}}$$

- **Problem:** Bigamy: $c^1{}_{\mathtt{Male}}$ and jeff are husbands of *Mary*!

7.4.2 Model Generation with Definite Descriptions

To obtain a model generation calculus for $\mathrm{HOL}^{\mathrm{nq}}$ 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

▷ **Definition 7.4.6.**

$$\frac{\begin{array}{cc} P(c)^{\mathsf{T}} & \\ Q(\iota\ P)^{\alpha} & \mathcal{H} = \{c, a_1, \ldots, a_n\} \end{array}}{\begin{array}{c} Q(c)^{\alpha} \\ (P(a_1) \Rightarrow c = a_1)^{\mathsf{T}} \\ \vdots \\ (P(a_n) \Rightarrow c = a_n)^{\mathsf{T}} \end{array}} RM\,\iota$$

▷ **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 $RM\,\iota$ forces it to be by making all other members of P equal to c.

Mary owned a lousy computer. The hard drive crashed.

$$(\forall X.\mathrm{computer}(X) \Rightarrow (\exists Y.\mathrm{harddrive}(Y) \wedge \mathrm{partof}(Y, X)))^{\mathsf{T}}$$
$$\boxed{\exists X.\mathrm{computer}(X) \wedge \mathrm{lousy}(X) \wedge \mathrm{own}(\mathrm{mary}, X)}$$
$$\mathrm{computer}(c)^{\mathsf{T}}$$
$$\mathrm{lousy}(c)^{\mathsf{T}}$$
$$\mathrm{own}(\mathrm{mary}, c)^{\mathsf{T}}$$

$$\begin{array}{c|c} \mathrm{harddrive}(c)^{\mathsf{T}} & \mathrm{harddrive}(d)^{\mathsf{T}} \\ \mathrm{partof}(c, c)^{\mathsf{T}} & \mathrm{partof}(d, c)^{\mathsf{T}} \\ \vdots & \boxed{\mathrm{crashes}(\iota\ \mathrm{harddrive})} \\ \bot & \mathrm{crashes}(d)^{\mathsf{T}} \\ & (\mathrm{harddrive}(\mathrm{mary}) \Rightarrow \mathrm{mary} = d)^{\mathsf{T}} \\ & (\mathrm{harddrive}(c) \Rightarrow c = d)^{\mathsf{T}} \end{array}$$

Definition 7.4.7. 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 $RM\,\iota$ 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

$$\begin{array}{c|c} \multicolumn{2}{c}{\mathrm{dog}(\mathrm{fido})^{\mathsf{T}}} \\ \multicolumn{2}{c}{\mathrm{dog}(\mathrm{chester})^{\mathsf{T}}} \\ \multicolumn{2}{c}{\boxed{\mathrm{bark}(\iota\ \mathrm{dog})}} \\ \multicolumn{2}{c}{\mathrm{bark}(\mathrm{fido})^{\mathsf{T}}} \\ \multicolumn{2}{c}{(\mathrm{dog}(\mathrm{chester}) \Rightarrow \mathrm{chester} = \mathrm{fido})^{\mathsf{T}}} \\ \mathrm{dog}(\mathrm{chester})^{\mathsf{F}} & \mathrm{chester} = \mathrm{fido}^{\mathsf{T}} \\ \bot & \end{array} \tag{7.1}$$

▷ 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.

7.4.3 Model Generation with Unique Name Assumptions

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

▷ **Definition 7.4.8.** The unique name assumption (UNA) makes the assumption that names are unique (in the respective context)

▷ **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 universe $\mathcal{H} = \mathcal{U} \cup \mathcal{W}$ into subsets $\mathcal{U}$ for constants with a UNA, and $\mathcal{W}$ without. (treat them differently)

▷ **Definition 7.4.9 (Model Generation with UNA).** We add the following two rules to the $\mathcal{RM}$ calculus to deal with the unique name assumption.

$$\frac{\begin{array}{c} a = b^{\mathsf{T}} \\ \mathbf{A}^{\alpha} \end{array} \quad a \in \mathcal{W} \quad b \in \mathcal{H}}{([b/a](\mathbf{A}))^{\alpha}}\ \mathcal{RM}\,\mathrm{subst} \qquad \frac{a = b^{\mathsf{T}} \quad a, b \in \mathcal{U}}{\bot}\ \mathcal{RM}\,\mathrm{una}$$

In effect we make the equality replacement rule directional; it only allows the substitution for a constant without the unique name assumption. Finally, $\mathcal{RM}$ 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 $\mathcal{RM}$ una, we can close the right branch of tableau (7.1), in accord with our intuition about the discourse.

Solving a Crime with Unique Names

▷ **Example 7.4.10.** Tony has observed (at most) two people. Tony observed a murderer that had black hair. It turns out that Bill and Bob were the two people Tony observed. Bill is blond, and Bob has black hair. (Who was the murderer.)
Let $\mathcal{U} = \{\mathrm{Bill}, \mathrm{Bob}\}$ and $\mathcal{W} = \{\mathrm{murderer}\}$:

$$\begin{array}{c}
(\forall z.\mathrm{observes}(\mathrm{Tony}, z) \Rightarrow (z = \mathrm{Bill} \vee z = \mathrm{Bob}))^{\mathsf{T}}\\
\mathrm{observes}(\mathrm{Tony}, \mathrm{Bill})^{\mathsf{T}}\\
\mathrm{observes}(\mathrm{Tony}, \mathrm{Bob})^{\mathsf{T}}\\
\mathrm{observes}(\mathrm{Tony}, \mathrm{murderer})^{\mathsf{T}}\\
\mathrm{black_hair}(\mathrm{murderer})^{\mathsf{T}}\\
\neg\mathrm{black_hair}(\mathrm{Bill})^{\mathsf{T}}\\
\mathrm{black_hair}(\mathrm{Bill})^{\mathsf{F}}\\
\mathrm{black_hair}(\mathrm{Bob})^{\mathsf{T}}\\
(\mathrm{observes}(\mathrm{Tony}, \mathrm{murderer}) \Rightarrow (\mathrm{murderer} = \mathrm{Bill} \vee \mathrm{murderer} = \mathrm{Bob}))^{\mathsf{T}}\\
(\mathrm{murderer} = \mathrm{Bill} \vee \mathrm{murderer} = \mathrm{Bob})^{\mathsf{T}}\\
\begin{array}{c|c}
\mathrm{murderer} = \mathrm{Bill}^{\mathsf{T}} & \mathrm{murderer} = \mathrm{Bob}^{\mathsf{T}}\\
\mathrm{black_hair}(\mathrm{Bill})^{\mathsf{T}} & \\
\bot &
\end{array}
\end{array}$$

FAU Michael Kohlhase: LBS 151 2025-02-06

Rabbits [Gardent & Konrad '99]

▷ Interpret "the" as $\lambda PQ.Q\iota\ P \wedge \mathrm{uniq}(P)$
where $\mathrm{uniq} := \lambda P.(\exists X.P(X) \wedge (\forall Y.P(Y) \Rightarrow X = Y))$
and $\forall\!\!\!\forall := \lambda PQ.(\forall X.P(X) \Rightarrow Q(X))$.

▷ "the rabbit is cute", has logical form $\mathrm{uniq}(\mathrm{rabbit}) \wedge (\mathrm{rabbit} \subseteq \mathrm{cute})$.

▷ RM generates $\{\ldots, \mathrm{rabbit}(c), \mathrm{cute}(c)\}$ in situations with at most 1 rabbit.
(special $RM\,\exists$ rule yields identification and accommodation (c^{new}))

+ At last an approach that takes world knowledge into account!

– tractable only for toy discourses/ontologies
The world cup final was watched on TV by 7 million people.
A rabbit is in the garden.
$\forall X.\mathrm{human}(x)\exists Y.\mathrm{human}(X) \wedge \mathrm{father}(X, Y)$ $\quad\forall X, Y.\mathrm{father}(X, Y) \Rightarrow X \neq Y$

FAU Michael Kohlhase: LBS 152 2025-02-06

More than one Rabbit

▷ **Problem:** What about two rabbits?
Bugs and Bunny are rabbits. Bugs is in the hat. Jon removes the rabbit from the hat.

- **Idea: Uniqueness under Scope [Gardent & Konrad '99]:**
  - refine *the* to $\lambda PRQ.\mathrm{uniq}(P \cap R \wedge \forall\!\!\forall(P \cap R, Q))$
    where R is an "identifying property" (identified from the context and passed as an arbument to *the*)
  - here R is "being in the hat" (by world knowledge about removing)
  - makes Bugs unique (in $P \cap R$) and the discourse acceptable.
- **Idea:** [Hobbs & Stickel&...]:
  - use generic relation rel for "relatedness to context" for P^2.

  ?? Is there a general theory of relatedness?

Michael Kohlhase: LBS 153 2025-02-06

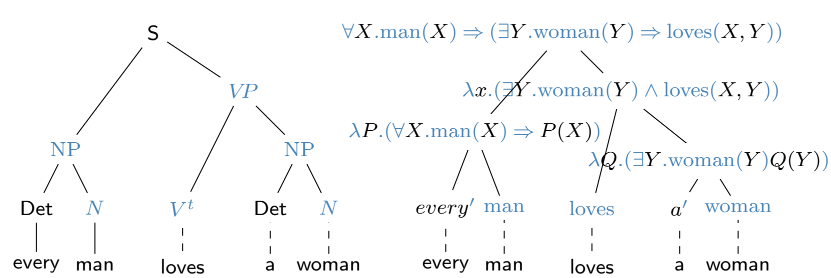
7.5 Quantifier Scope Ambiguity and Underspecification

7.5.1 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.*
- **Definition 7.5.1.** We call these systematic ambiguities quantifyer scope ambiguities
- **Example 7.5.2.** We can represent the "wide-scope" reading with our methods



- **Question:** How to map an unambiguous input structure to multiple translations.

Michael Kohlhase: LBS 154 2025-02-06

This is a correct representation of one of the possible readings 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 readings 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 $=_\beta$-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 $=_\beta$-reduction.

In the VP node for *loves a woman* we have $(\lambda FX.\lambda Q.(\exists Y.\mathrm{woman}(Y) \wedge Q\,Y))\ \mathrm{loves}$ and thus the sentence representation is

$$(\lambda P.(\forall X.\mathrm{man}(X) \Rightarrow P(X)))\ (\lambda FX.\lambda Q.(\exists Y.\mathrm{woman}(Y) \wedge Q\,Y))\ \mathrm{loves}$$

The resulting expression is an application of form $\langle \mathrm{everyman}\rangle(\langle \mathrm{awoman}\rangle(\langle \mathrm{loves}\rangle))$. 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 $=_\beta$-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 $\langle \mathrm{awoman}\rangle(\langle \mathrm{everyman}\rangle(\langle \mathrm{loves}\rangle))$ would do.

Storing and Quantifying In

- **Analysis:** The sentence meaning is of the form $\langle \mathrm{everyman}\rangle(\langle \mathrm{awoman}\rangle(\langle \mathrm{loves}\rangle))$
- **Idea:** Somehow have to move the *a woman* part in front of the *every* to obtain

$$\langle \mathrm{awoman}\rangle(\langle \mathrm{everyman}\rangle(\langle \mathrm{loves}\rangle))$$

- **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* out-scopes *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.

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 semantics 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 7.5.3.** $\mathbf{STORE}(NP, \Phi) \longrightarrow (\Phi, \Sigma{*}NP)$, 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 7.5.4.** $\mathbf{QI}(\langle \Phi, \Sigma{*}NP \rangle) \rightarrow \langle NP \oplus \Phi, \Sigma \rangle$ 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. $\mathbf{STORE}$ 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 $\mathbf{STORE}(B', C')$

  Note that $\mathbf{STORE}$ may be applied whether or not the stores of the constituent nodes are empty.

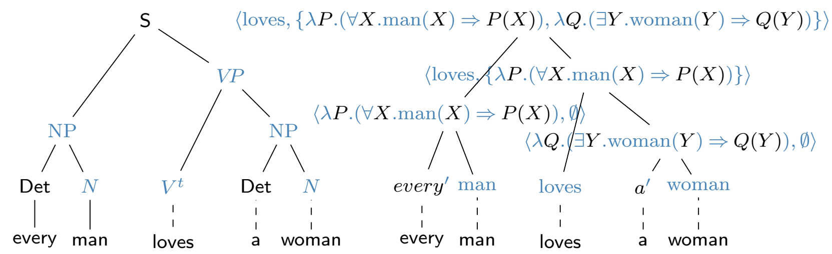
We now have more than one way to translate a branching node, but the choice is partly constrained by whether or not the daughters of the node have empty stores. We have the following two options for translating a branching node. (Note: To simplify the notation, let us adopt the following convention: 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 $\mathbf{STORE}$ must always eventually be followed by application of $\mathbf{QI}$. (Note that $\mathbf{QI}$ is not a translation rule, but a sort of transformation on translations.) But when must $\mathbf{QI}$ be applied? There are two cases:

1. The process of semantics construction 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 $\mathbf{QI}$ 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 processing 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 loves. To compute the semantics of the VP nodes, we use the rule $\mathbf{STORE}$ and obtain $\langle \mathrm{loves}, \{\mathbf{A}\} \rangle$ and similarly $\langle \mathrm{loves}, \{\mathbf{A}, \mathbf{B}\} \rangle$ for the for the S node, thus we have the following semantics tree.

Quantifying in Practice: *Every man loves a woman*

- **Example 7.5.5.**



▷ Continue with **QI** applications: first retrieve $\lambda Q.(\exists Y.\mathrm{woman}(Y) \Rightarrow Q(Y))$

$$
\begin{array}{ll}
 & \langle \mathrm{loves}, \{\lambda P.(\forall X.\mathrm{man}(X) \Rightarrow P(X)), \lambda Q.(\exists Y.\mathrm{woman}(Y) \Rightarrow Q(Y))\}\rangle \\
\to_{QI} & \langle \circ\ (\lambda P.(\forall X.\mathrm{man}(X) \Rightarrow P(X)))\ \mathrm{loves}, \{\lambda Q.(\exists Y.\mathrm{woman}(Y) \Rightarrow Q(Y))\}\rangle \\
\to_{\beta} & \langle \lambda Z.(\lambda P.(\forall X.\mathrm{man}(X) \Rightarrow P(X)))\ \mathrm{loves}\ Z, \{\lambda Q.(\exists Y.\mathrm{woman}(Y) \Rightarrow Q(Y))\}\rangle \\
\to_{\beta} & \langle \lambda Z.(\forall X.\mathrm{man}(X) \Rightarrow \mathrm{loves}\ Z\ X), \{\lambda Q.(\exists Y.\mathrm{woman}(Y) \Rightarrow Q(Y))\}\rangle \\
\to_{QI} & \langle (\lambda Q.(\exists Y.\mathrm{woman}(Y) \Rightarrow Q(Y)))\ (\lambda Z.(\forall X.\mathrm{man}(X) \Rightarrow \mathrm{loves}\ Z\ X)), \emptyset\rangle \\
\to_{\beta} & \langle \exists Y.\mathrm{woman}(Y) \Rightarrow (\lambda Z.(\forall X.\mathrm{man}(X) \Rightarrow \mathrm{loves}\ Z\ X))\ Y, \emptyset\rangle \\
\to_{\beta} & \langle \exists Y.\mathrm{woman}(Y) \Rightarrow (\forall X.\mathrm{man}(X) \Rightarrow \mathrm{loves}\ Y\ X), \emptyset\rangle
\end{array}
$$

FAU Michael Kohlhase: LBS 157 2025-02-06

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.

7.5.2 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*:

$$\lambda PY.P\ (\lambda X.\mathrm{likes}(X,Y))$$

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 [CKG09, pp.178-179])

FAU Michael Kohlhase: LBS 158 2025-02-06

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

▷ **Intuition:** 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 7.5.6.** 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 operators (because we will use them to bind free variables in the core representation).

▷ **Definition 7.5.7.** In the Cooper storage method, QNPs are stored in the store and later retrieved – not necessarily in the order they were stored – to build the 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.

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.

Working with Stores

▷ Working out the translation for *Every student likes some professor.*

$NP_1 \rightarrow \lambda P.(\exists X.\mathrm{prof}(X) \wedge P(X))$ **or** $\langle \lambda Q.Q(X_1), (\lambda P.(\exists X.\mathrm{prof}(X) \wedge P(X)), 1) \rangle$
$V_t \rightarrow \lambda RY.R\ (\lambda Z.\mathrm{likes}(Z, Y))$
$VP \rightarrow$ (Combine core representations by FA; pass store up)*
$\rightarrow \langle \lambda Y.\mathrm{likes}(X_1, Y), (\lambda P.(\exists X.\mathrm{prof}(X) \wedge P(X)), 1) \rangle$
$NP_2 \rightarrow \lambda P.(\forall Z.\mathrm{student}(Z) \Rightarrow P(Z))$ **or** $\langle \lambda R.R(X_2), (\lambda P.(\forall Z.\mathrm{student}(Z) \Rightarrow P(Z)), 2) \rangle$
$S \rightarrow$ (Combine core representations by FA; pass stores up)**
$\rightarrow \langle \mathrm{likes}(X_1, X_2), (\lambda P.(\exists X.\mathrm{prof}(X) \wedge P(X)), 1), (\lambda P.(\forall Z.\mathrm{student}(Z) \Rightarrow P(Z)), 2) \rangle$

* Combining V_t with place holder

1. $(\lambda RY.R\ (\lambda Z.\mathrm{likes}(Z, Y)))\ (\lambda Q.Q(X_1))$
2. $\lambda Y.(\lambda Q.Q(X_1))\ (\lambda Z.\mathrm{likes}(Z, Y))$
3. $\lambda Y.(\lambda Z.\mathrm{likes}(Z, Y))\ X_1$
4. $\lambda Y.\mathrm{likes}(X_1, Y)$

** Combining VP with place holder

1. $(\lambda R.R(X_2))\ (\lambda Y.\mathrm{likes}(X_1, Y))$
2. $(\lambda Y.\mathrm{likes}(X_1, Y))\ X_2$
3. $\mathrm{likes}(X_1, X_2)$

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 $\iota \rightarrow o$.)
3. Apply this λ-term to the QNP translation (which is of type $(\iota \rightarrow o) \rightarrow o$).

Example: *Every student likes some professor.*

1. Retrieve *every student*
   (a) $(\lambda Q.(\forall Z.\mathrm{student}(Z) \Rightarrow Q(Z)))\ (\lambda X_2.\mathrm{likes}(X_1, X_2))$
   (b) $\forall Z.\mathrm{student}(Z) \Rightarrow (\lambda X_2.\mathrm{likes}(X_1, X_2))\ Z$
   (c) $\forall Z.\mathrm{student}(Z) \Rightarrow \mathrm{likes}(X_1, Z)$
2. Retrieve *some professor*
   (a) $(\lambda P.(\exists X.\mathrm{prof}(X) \wedge P(X)))\ (\lambda X_1.(\forall Z.\mathrm{student}(Z) \Rightarrow \mathrm{likes}(X_1, Z)))$
   (b) $\exists X.\mathrm{prof}(X)(\lambda X_1.(\forall Z.\mathrm{student}(Z) \Rightarrow \mathrm{likes}(X_1, Z)))\ X$
   (c) $\exists X.\mathrm{prof}(X) \wedge (\forall Z.\mathrm{student}(Z) \Rightarrow \mathrm{likes}(X, Z))$

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.

7.5.3 Underspecification

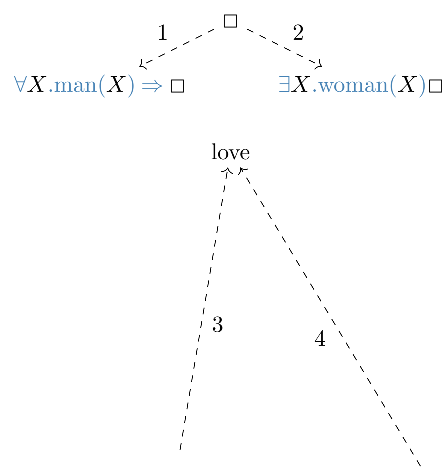
In this subsection we introduce Johan Bos' "Hole Semantics", since this is possibly the simplest underspecification framework around. The main idea is that the result of the translation is a "quasi-logical form" (QLF), i.e. a representation that represents all possible readings. This QLF can then be used for semantic/pragmatic analysis.

7.5.3.1 Unplugging Predicate Logic

The problem we need to solve for our QLF is that regular logical formulae, such as

$$\forall X.\mathrm{man}(X) \Rightarrow (\exists Y.\mathrm{woman}(Y) \Rightarrow \mathrm{loves}(Y, X))$$

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* out-scopes *every man*. So much for the general idea, how can this be represented in logic?

7.5.3.2 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.

7.5.3.3 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	$[\ell_1/h_0], [\ell_2/h_1], [\ell_3/h_2]$	$\forall X.\mathrm{man}(X) \Rightarrow (\exists Y.\mathrm{woman}(Y) \wedge \mathrm{loves}(X, Y))$
2	$[\ell_2/h_0], [\ell_3/h_1], [\ell_1/h_2]$	$\exists Y.\mathrm{woman}(Y) \Rightarrow (\forall X.\mathrm{man}(X) \wedge \mathrm{loves}(X, Y))$

7.6 Summary & Evaluation

So let us evaluate what we have achieved in the new, extended fragment.

Fragment $\mathcal{F}_4$ – Summary

- ▷ Fragment $\mathcal{F}_4$ extends $\mathcal{F}_3$ by noun phrases.
- ▷ **Coverage:** Better:

Chapter 8

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 kills 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)

- **Definition 8.0.1.** In event semantics we extend the argument structure of (action) verbs contains a 'hidden' argument, the event argument, then treat modifiers as predicates (often called roles) over events [Dav67a].
- **Example 8.0.2.**

1. $\exists e.\exists x,y.\mathrm{bathroom}(x) \wedge \mathrm{hammer}(y) \wedge \mathrm{kill}(e,\mathrm{peter},\iota\ \mathrm{cat}) \wedge \mathrm{in}(e,x) \wedge \mathrm{with}(e,y)$
2. $\exists e.\exists x,y.\mathrm{bathroom}(x) \wedge \mathrm{hammer}(y) \wedge \mathrm{kill}(e,\mathrm{peter},\iota\ \mathrm{cat}) \wedge \mathrm{in}(e,x) \wedge \mathrm{with}(e,y) \wedge \mathrm{at}(e,24:00)$

Event semantics: Neo-Davidsonian Systems

- **Idea:** Take apart the Davidsonian predicates even further, add event participants via thematic roles (from [Par90]).
- **Definition 8.0.3.** Neo-Davidsonian semantics extends event semantics by adding two standardized roles: the agent $\mathrm{ag}(e,s)$ and the patient $\mathrm{pat}(e,o)$ for the subject s and direct object d of the event e.

- **Example 8.0.4.** Translate *John killed a cat with a hammer.* as $\exists e.\exists x.\mathrm{hammer}(x) \wedge \mathrm{killing}(e) \wedge \mathrm{ag}(e, \mathrm{peter}) \wedge \mathrm{pat}(e, \iota\ \mathrm{cat}) \wedge \mathrm{with}(e, x)$
- **Further Elaboration:** Events can be broken down into sub-events and modifiers can predicate over sub-events.
- **Example 8.0.5.** 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)

Event Types and Properties of Events

- **Example 8.0.6 (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 8.0.7 (Types of Events).** There are different types of events that go with different temporal modifiers. [Ven57] distinguishes
  1. states: e.g. *know the answer*, *stand in the corner*
  2. processes: e.g. *run*, *eat*, *eat apples*, *eat soup*
  3. accomplishments: e.g. *run a mile*, *eat an apple*, and
  4. achievements: e.g. *reach the summit*
- **Observations:**
  1. processes and accomplishments appear in the progressive (1),
  2. states and achievements do not (2).
- **Definition 8.0.8.** The 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 8.0.9.**
  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*

Part II

Topics in Semantics

Chapter 9

Dynamic Approaches to NL Semantics

In this chapter 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 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 chapter, we will introduce dynamic 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]). section 35 (Discourse Representation Theory) in the LBS lecture notes gives an introduction to dynamic language phenomena and how they can be modeled in DRT. section 37 (Dynamic Logic for Imperative Programs) in the LBS lecture notes relates the linguistically motivated logics to modal logics used for modeling imperative programs and draws conclusions about the role of language in cognition. **??** 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, section 36 (Higher-Order Dynamics) in the LBS lecture notes discusses how dynamic approaches to NL semantics can be combined with ideas Montague Semantics to arrive at a fully compositional approach to discourse semantics.

9.1 Discourse Representation Theory

In this section 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 $\mathcal{F}_4$ (see **??**).

Anaphora and Indefinites revisited (Data)

- ▷ **Observation:** We have concentrated on single sentences so far; let's do better.
- ▷ **Definition 9.1.1.** A discourse is a unit of natural language longer than a single

sentence.

- **New Data:** Discourses interact with anaphora.:
  - *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 if 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)
- We gloss the intended anaphoric reference with the labels in upper and lower indices.

In the first example, we can pick up the subject *Peter* of the first sentence with the anaphoric reference *He* in the second. 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 $\mathcal{F}_2$ (see section 7 (Fragment 2: Pronouns and Anaphora) in the LBS lecture notes) 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 anaphor resolution we will adopt in this section.

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 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 discourse 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 anaphoric potential, but the sentence *Peter does not own every book* does not, even though their translations $\exists x.\mathrm{book}(x) \wedge \neg\mathrm{own}(\mathrm{peter}, x)$ 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.

Dynamic Effects in Natural Language

- **Problem:** E.g. Quantifier Scope
  - * *A man sleeps. He snores.*
  - $(\exists X.\mathrm{man}(X) \wedge \mathrm{sleeps}(X)) \wedge \mathrm{snores}(X)$

- ▷ X is bound in the first conjunct, and free in the second.

▷ **Problem:** Donkey sentence: *If a farmer owns a donkey, he beats it.*
$\forall X, Y.\mathrm{farmer}(X) \wedge \mathrm{donkey}(Y) \wedge \mathrm{own}(X, Y) \Rightarrow \mathrm{beat}(X, Y)$

▷ **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)

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: discourse referents, and manage their 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)

▷ **Definition 9.1.2.** Discourse Representation Theory (DRT) is a logical system, which uses discourse referents to model quantification and pronouns. DRT formulae are called discourse representation structures (DRS); these introduce a set of discourse referents and specify their meaning by conditions which comprise:

- ▷ atomic first-order propositions,
- ▷ dynamic negations $\neg D$,
- ▷ dynamic implications $D \Rrightarrow E$, and
- ▷ dynamic disjunctions $D \vee E$.

▷ **Example 9.1.3.** Discourse referents e.g. in *A student owns a book.*

- ▷ are kept in a dynamic context (⇝ accessibility)
- ▷ are declared e.g. in indefinite nominals
- ▷ specified in conditions via predicates

$$\begin{array}{|l|}\hline X, Y \\ \hline \mathrm{student}(X) \\ \mathrm{book}(Y) \\ \mathrm{own}(X, Y) \\ \hline \end{array}$$

▷ **Example 9.1.4.** Discourse representation structures (DRS)
A student owns a book. He reads it. *If a farmer owns a donkey, he beats it.*

$$\begin{array}{|l|}\hline X, Y, R, S \\ \hline \mathrm{student}(X) \\ \mathrm{book}(Y) \\ \mathrm{own}(X, Y) \\ \mathrm{read}(R, S) \\ X = R \\ Y = S \\ \hline \end{array}
\qquad
\begin{array}{|l|}\hline \\ \hline \begin{array}{|l|}\hline X, Y \\ \hline \mathrm{farmer}(X) \\ \mathrm{donkey}(Y) \\ \mathrm{own}(X, Y) \\ \hline \end{array} \Rrightarrow \begin{array}{|l|}\hline \\ \hline \mathrm{beat}(X, Y) \\ \hline \end{array} \\ \hline \end{array}$$

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?
- ▷ **Solution:** We construct sentence DRSes individually and merge them (DRSes and conditions separately)
- ▷ **Example 9.1.5.** A three-sentence discourse. (not quite Shakespeare)

Mary sees John.	*John kills a cat.*	*Mary calls a cop.*	merge
$\begin{array}{\|l\|}\hline \\ \hline \mathrm{see}(\mathrm{mary},\mathrm{john}) \\ \hline\end{array}$	$\begin{array}{\|l\|}\hline U \\ \hline \mathrm{cat}(U) \\ \mathrm{kills}(\mathrm{john},U) \\ \hline\end{array}$	$\begin{array}{\|l\|}\hline V \\ \hline \mathrm{policeman}(V) \\ \mathrm{calls}(\mathrm{mary},V) \\ \hline\end{array}$	$\begin{array}{\|l\|}\hline U,V \\ \hline \mathrm{see}(\mathrm{mary},\mathrm{john}) \\ \mathrm{cat}(U) \\ \mathrm{kills}(\mathrm{john},U) \\ \mathrm{policeman}(V) \\ \mathrm{calls}(\mathrm{mary},V) \\ \hline\end{array}$

- ▷ Sentence composition via the DRT Merge Operator $\otimes$. (acts on DRSes)

FAU Michael Kohlhase: LBS 171 2025-02-06

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[1] 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.

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)
- ▷ **Example 9.1.6.** *A student owns a book. He reads it.*

[1] We will not go into the sentence semantics construction process here

A student[1] owns a book[2]. He$_1$ reads it$_2$ | merge/resolve | simplify

X, Y
student(X)
book(Y)
own(X, Y)

R, S
read(R, S)

X, Y, R, S
student(X)
book(Y)
own(X, Y)
read(R, S)
$X = R$
$Y = S$

X, Y
student(X)
book(Y)
own(X, Y)
read(X, Y)

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 (more Logic-like Syntax)

- **Definition 9.1.7.** Given a set $\mathcal{DR}$ of discourse referents, discourse representation structure (DRSes) are given by the following grammar:

conditions	$\mathcal{C}::=p(a_1, \ldots, a_n) \mid \mathcal{C}_1 \wedge \mathcal{C}_2 \mid \neg\mathcal{D} \mid \mathcal{D}_1 \Vee \mathcal{D}_2 \mid \mathcal{D}_1 \Rrightarrow \mathcal{D}_2$
DRSes	$\mathcal{D}::=\delta U^1, \ldots, U^n.\mathcal{C} \mid \mathcal{D}_1 \otimes \mathcal{D}_2 \mid \mathcal{D}_1 \,;;\, \mathcal{D}_2$

- $\otimes$ and $;;$ are for sentence composition (⊗ from DRT, ;; from DPL)
- **Example 9.1.8.** $\delta U, V.\text{farmer}(U) \wedge \text{donkey}(V) \wedge \text{own}(U, V) \wedge \text{beat}(U, V)$
- **Definition 9.1.9.** The meaning of $\otimes$ and $;;$ is given operationally by $=_\tau$ equality:

$$\begin{aligned} \delta\mathcal{X}.\mathcal{C}_1 \otimes \delta\mathcal{Y}.\mathcal{C}_2 &\to_\tau \delta\mathcal{X}, \mathcal{Y}.\mathcal{C}_1 \wedge \mathcal{C}_2 \\ \delta\mathcal{X}.\mathcal{C}_1 \,;;\, \delta\mathcal{Y}.\mathcal{C}_2 &\to_\tau \delta\mathcal{X}, \mathcal{Y}.\mathcal{C}_1 \wedge \mathcal{C}_2 \end{aligned}$$

- Discourse referents used instead of bound variables.(specify scoping independently of logic)
- **Idea:** Semantics inherited from first-order logic by a translation mapping.

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 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:** How can we formally define accessibility. (to make predictions)

- **Idea:** Make use of the structural properties of DRT.
- **Definition 9.1.10.** 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 $\mathcal{D}^1 \veebar \mathcal{D}^2$, 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 9.1.11 (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 9.1.12.** We call a DRS static, iff the dynamic potential is empty, and dynamic, if it is not.

Sub DRSes and Accessibility

- **Observation:** Accessibility gives DRSes the flavor of binding structures. (with non-standard scoping!)
- **Idea:** Apply the usual binding heuristics to DRT, e.g.
  - reject DRSes with unbound discourse referents.
- **Questions:** If we view of discourse referents as "nonstandard bound variables"
  - what about renaming referents?

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 9.1.13.** For $=_\tau$-normal (fully merged) DRSes use the translation $\overline{\cdot}$:

$$\begin{array}{rcl}\overline{\delta U^1, \ldots, U^n.\mathcal{C}} & = & \exists U^1, \ldots, U^n.\overline{\mathcal{C}} \\ \overline{\neg\mathcal{D}} & = & \neg\overline{\mathcal{D}} \\ \overline{\mathcal{D}\veebar\mathcal{E}} & = & \overline{\mathcal{D}} \vee \overline{\mathcal{E}} \\ \overline{\mathcal{D}\wedge\mathcal{E}} & = & \overline{\mathcal{D}} \wedge \overline{\mathcal{E}} \\ \overline{(\delta U^1, \ldots, U^n.\mathcal{C}_1) \Rrightarrow (\delta V^1, \ldots, V^n.\mathcal{C}_2)} & = & \forall U^1, \ldots, U^n.\overline{\mathcal{C}_1} \Rightarrow (\exists V^1, \ldots, V^n.\overline{\mathcal{C}_2})\end{array}$$

▷ **Example 9.1.14.** $\begin{array}{|l|}\hline X,Y\\\hline \mathrm{student}(X)\\ \mathrm{book}(Y)\\ \mathrm{own}(X,Y)\\\hline\end{array} = \exists X.\exists Y.\mathrm{student}(X)\wedge\mathrm{book}(Y)\wedge\mathrm{own}(X,Y).$

▷ **Example 9.1.15.**

$$\overline{(\delta U,V.\mathrm{farmer}(U)\wedge\mathrm{donkey}(V)\wedge\mathrm{own}(U,V))\Rrightarrow(\delta W.\mathrm{stick}(W)\wedge\mathrm{beatwith}(U,V,W))}$$
$$=\forall X,Y.\mathrm{farmer}(X)\wedge\mathrm{donkey}(X)\wedge\mathrm{own}(X,Y)\Rightarrow(\exists.\mathrm{stick}(Z)\wedge\mathrm{beatwith}(Z,X,Y))$$

▷ **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).

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 section.

Properties of Dynamic Scope

▷ **Idea:** Test DRT on the data above for the dynamic phenomena

▷ **Example 9.1.16 (Negation Closes Dynamic Potential).**
Peter has no[1] car. * *It[1] is parked outside.*

$$\begin{array}{|l|}\hline \\\hline \neg\begin{array}{|l|}\hline U\\\hline \mathrm{acar}(U)\\ \mathrm{own}(\mathrm{peter},U)\\\hline\end{array}\\\hline\end{array}\otimes\begin{array}{|l|}\hline \\\hline \mathrm{parked}(U)\\\hline\end{array}\qquad\neg(\exists U.\mathrm{acar}(U)\wedge\mathrm{own}(\mathrm{peter},U))\ldots$$

▷ **Example 9.1.17 (Universal Quantification is Static).**
Peter does not own every book[1]. * *It[1] is a novel.*

$$\begin{array}{|l|}\hline \\\hline \neg\begin{array}{|l|}\hline \\\hline \begin{array}{|l|}\hline U\\\hline \mathrm{book}(U)\\\hline\end{array}\Rrightarrow\begin{array}{|l|}\hline \\\hline \mathrm{own}(\mathrm{peter},U)\\\hline\end{array}\\\hline\end{array}\\\hline\end{array}\otimes\begin{array}{|l|}\hline \\\hline \mathrm{novel}(U)\\\hline\end{array}\qquad\neg(\forall U.\mathrm{book}(U)\Rrightarrow\mathrm{own}(\mathrm{peter},U))\ldots$$

▷ **Example 9.1.18 (Existential Quantification is Dynamic).**
There is a book[1] that Peter does not own. *It[1] is a novel.*

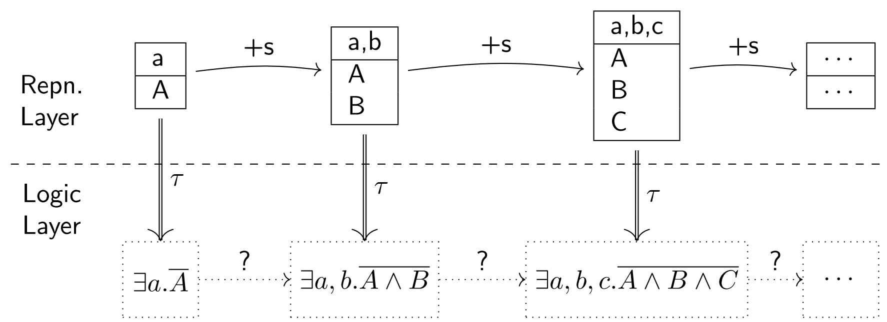
$$\begin{array}{|l|}\hline V\\\hline \mathrm{book}(V)\\ (\neg\mathrm{own}(\mathrm{peter},V))\\\hline\end{array}\otimes\begin{array}{|l|}\hline \\\hline \mathrm{novel}(V)\\\hline\end{array}\qquad\exists U.\mathrm{book}(U)\wedge\neg\mathrm{own}(\mathrm{peter},U)\wedge\mathrm{novel}(U)$$

?? shows that dynamic negation closes off the dynamic potential. Indeed, the referent U is not accessible in the second argument of $\otimes$. **??** predicts the inaccessibility of U for the same reason. In contrast to that, U is accessible in **??**, since it is not under the scope of a dynamic negation.
The examples above, and in particular the difference between Example 35.15 (Discourse Representation Theory) in the LBS lecture notes and Example 35.16 (Discourse Representation Theory) in the LBS lecture notes 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 con-

struction for sentences followed by DRS merging via $=_\tau$ reduction.

DRT as a Representational Level

- ▷ DRT adds a level to the knowledge representation which provides anchors (the discourse referents) for anaphora and the like.
- ▷ Propositional semantics by translation into PL^1. ("+s" adds a sentence)



- ▷ Anaphor resolution works incrementally on the representational level.

FAU Michael Kohlhase: LBS 178 2025-02-06

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 states that satisfy the conditions. 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 9.1.19.** Let $\mathcal{M} = \langle \mathcal{D}, \mathcal{I} \rangle$ be a first-order model, then a state is an assignment from discourse referents into $\mathcal{D}$.
- ▷ **Definition 9.1.20.** Let $\varphi, \psi : \mathcal{DR} \to \mathcal{U}$ be states, 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 $(\mathcal{X}, \mathcal{S})$ ($\mathcal{S}$ set of states)
- ▷ **Definition 9.1.21 (Meaning of complex formulae).** The value function $\mathcal{I}_\varphi$ for DRT is defined with the help of a dynamic value function $\mathcal{I}^\delta_\varphi$ on DRSs: For conditions:
  - ▷ $\mathcal{I}_\varphi(\neg\mathcal{D}) = \mathsf{T}$, if $\mathcal{I}^\delta_\varphi(\mathcal{D})^2 = \emptyset$.
  - ▷ $\mathcal{I}_\varphi(\mathcal{D} \vee \mathcal{E}) = \mathsf{T}$, if $\mathcal{I}^\delta_\varphi(\mathcal{D})^2 \neq \emptyset$ or $\mathcal{I}^\delta_\varphi(\mathcal{E})^2 \neq \emptyset$.
  - ▷ $\mathcal{I}_\varphi(\mathcal{D} \Rrightarrow \mathcal{E}) = \mathsf{T}$, 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 $\psi[\mathcal{I}^\delta_\varphi(\mathcal{E})^1]\tau$.

  For DRSs $\mathcal{D}$ we set $\mathcal{I}_\varphi(\mathcal{D}) = \mathsf{T}$, iff $\mathcal{I}^\delta_\varphi(\mathcal{D})^2 \neq \emptyset$, and define

  - ▷ $\mathcal{I}^\delta_\varphi(\delta\mathcal{X}.\mathbf{C}) = (\mathcal{X}, \{\psi \mid \varphi[\mathcal{X}]\psi$ and $\mathcal{I}_\psi(\mathbf{C}) = \mathsf{T}\})$.
  - ▷ $\mathcal{I}^\delta_\varphi(\mathcal{D} \otimes \mathcal{E}) = \mathcal{I}^\delta_\varphi(\mathcal{D} \,;;\, \mathcal{E}) = (\mathcal{I}^\delta_\varphi(\mathcal{D})^1 \cup \mathcal{I}^\delta_\varphi(\mathcal{E})^1, \mathcal{I}^\delta_\varphi(\mathcal{D})^2 \cap \mathcal{I}^\delta_\varphi(\mathcal{E})^2)$

We use the dynamic value function $\mathcal{I}^{\delta}_{\varphi}(\mathcal{D})$ for DRSs $\mathcal{D}$ that might be continued and (the static $\mathcal{I}_{\varphi}(\mathcal{D})$ for ones that are already final.

We can now fortify our intuition by computing the direct semantics of two sentences, which differ in their dynamic potential. We start out with the simple *Peter owns a car* and then progress to *Peter owns no car*.

Examples (Computing Direct Semantics)

▷ **Example 9.1.22.** *Peter owns a car*

$$\begin{aligned} & \mathcal{I}^{\delta}_{\varphi}(\delta U.\mathrm{acar}(U) \wedge \mathrm{own}(\mathrm{peter}, U)) \\ = \; & (\{U\},\{\psi \,|\, \varphi[U]\psi \text{ and } \mathcal{I}_{\psi}(\mathrm{acar}(U) \wedge \mathrm{own}(\mathrm{peter}, U)) = \mathsf{T}\}) \\ = \; & (\{U\},\{\psi \,|\, \varphi[U]\psi \text{ and } \mathcal{I}_{\psi}(\mathrm{acar}(U)) = \mathsf{T} \text{ and } \mathcal{I}_{\psi}(\mathrm{own}(\mathrm{peter}, U)) = \mathsf{T}\}) \\ = \; & (\{U\},\{\psi \,|\, \varphi[U]\psi \text{ and } \psi(U) \in \mathcal{I}(\mathrm{acar}) \text{ and } (\psi(U),\mathrm{peter}) \in \mathcal{I}(\mathrm{own})\}) \end{aligned}$$

The set of states $[a/U]$, such that a is a car and is owned by Peter

▷ **Example 9.1.23.** For *Peter owns no car* we look at the condition:

$$\begin{aligned} & \mathcal{I}_{\varphi}(\neg(\delta U.\mathrm{acar}(U) \wedge \mathrm{own}(\mathrm{peter}, U))) = \mathsf{T} \\ \Leftrightarrow \; & \mathcal{I}^{\delta}_{\varphi}(\delta U.\mathrm{acar}(U) \wedge \mathrm{own}(\mathrm{peter}, U))^{2} = \emptyset \\ \Leftrightarrow \; & (\{U\},\{\psi \,|\, \varphi[\mathcal{X}]\psi \text{ and } \psi(U) \in \mathcal{I}(\mathrm{acar}) \text{ and } (\psi(U),\mathrm{peter}) \in \mathcal{I}(\mathrm{own})\})^{2} = \emptyset \\ \Leftrightarrow \; & \{\psi \,|\, \varphi[\mathcal{X}]\psi \text{ and } \psi(U) \in \mathcal{I}(\mathrm{acar}) \text{ and } (\psi(U),\mathrm{peter}) \in \mathcal{I}(\mathrm{own})\} = \emptyset \end{aligned}$$

i.e. iff there are no a, that are cars and that are owned by Peter.

The first thing we see in **??** 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 **??**, the interpretation is of the form $(\emptyset,\mathcal{I}^{\delta}_{\varphi}(\mathcal{C}))$, where $\mathcal{C}$ is the condition we compute the truth value of in **??**.

9.2 Dynamic Model Generation

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. This calculus can be seen as a first step towards a tableau machine for DRT and thus as a first step towards semantic-pragmatic analysis for discourses.

Deduction in Dynamic Logics

▷ **Problem:** Mechanize the dynamic entailment relation (with anaphora)

▷ **Idea:** Use dynamic deduction theorem to reduce (dynamic) entailment to (dynamic) satisfiability

▷ **History of Attempts:** Direct Deduction on DRT (or DPL) [Sau93; RG94; MR98]

(++) Specialized Calculi for dynamic representations.

(− −) Needs lots of development until we have efficient implementations.

▷ Translation approach (used in our experiment)

(−) Translate to PL^1.

(++) 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 NLP. (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:** Use translation approach for 1. to test 2. and 3. [Bla+01] (Wow, it works!)

Excursion: Incrementality in Dynamic Calculi

▷ For applications, we need to be able to check for

- ▷ satisfiability ($\exists \mathcal{M}.\mathcal{M} \models \mathbf{A}$), 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 $\mathcal{M}$)

▷ **Theorem 9.2.1 (Entailment Theorem).** *$\mathcal{H}, \mathbf{A} \models \mathbf{B}$, iff $\mathcal{H} \models \mathbf{A} \Rightarrow \mathbf{B}$. (e.g. for first-order logic and DPL)*

▷ **Theorem 9.2.2 (Deduction Theorem).** *For most calculi $\mathcal{C}$ we have $\mathcal{H}, \mathbf{A} \vdash_{\mathcal{C}} \mathbf{B}$, iff $\mathcal{H} \vdash_{\mathcal{C}} \mathbf{A} \Rightarrow \mathbf{B}$. (e.g. for $\mathcal{ND}^1$)*

▷ **Problem:** Analogue $\mathbf{H}_1 \otimes \cdots \otimes \mathbf{H}_n \models \mathbf{A}$ is not equivalent to $\models (\mathbf{H}_1 \otimes \cdots \otimes \mathbf{H}_n) \Rrightarrow \mathbf{A}$ in DRT, as $\otimes$ symmetric.

▷ **Thus** the 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 $\mathcal{M} = \langle\mathcal{D}, \mathcal{I}\rangle$, 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.

Dynamic Herbrand Valuation

▷ **Definition 9.2.3.** We call a model $\mathcal{M} = \langle\mathcal{U}, \mathcal{I}, \mathcal{I}^{\delta}_{\cdot}\rangle$ a dynamic Herbrand interpretation, if $\langle\mathcal{U}, \mathcal{I}\rangle$ is a Herbrand model.

▷ **Question:** Can represent $\mathcal{M}$ as a triple $\langle\mathcal{X}, \mathcal{S}, \mathcal{B}\rangle$, where $\mathcal{B}$ is the Herbrand valuation for $\langle\mathcal{U}, \mathcal{I}\rangle$?

▷ **Definition 9.2.4.** $\mathcal{M}$ is called finite, iff $\mathcal{U}$ is finite.

▷ **Definition 9.2.5.** $\mathcal{M}$ 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 9.2.6.** $\mathcal{M}$ is domain minimal if for all $\mathcal{M}'$ the following holds:

$$\#(\mathcal{U}(\mathcal{M})) \leq \#(\mathcal{U}(\mathcal{M})')$$

Dynamic Model Generation Calculus

▷ **Definition 9.2.7.** We use a tableau framework, extend by state information, and rules for DRSes.

- ▷

$$\frac{(\delta U_{\mathbb{A}}.\mathbf{A})^{\mathsf{T}} \quad \mathcal{H} = \{a_1, \ldots, a_n\} \quad w \notin \mathcal{H} \text{ new}}{\begin{array}{c} [a_1/U] \\ ([a_1/U](\mathbf{A}))^{\mathsf{T}} \end{array} \Bigg| \ldots \Bigg| \begin{array}{c} [a_n/U] \\ ([a_n/U](\mathbf{A}))^{\mathsf{T}} \end{array} \Bigg| \begin{array}{c} [w/U] \\ ([w/U](\mathbf{A}))^{\mathsf{T}} \end{array}} \; RM\,\delta$$

- ▷ Mechanize ;; by adding representation of the second DRS at all leaves. (⇝ tableau machine)
- ▷ Treat conditions by DRT translation

$$\frac{\neg\mathcal{D}}{\overline{\neg\mathcal{D}}} \qquad \frac{\mathcal{D} \Rrightarrow \mathcal{D}'}{\overline{\mathcal{D} \Rrightarrow \mathcal{D}'}} \qquad \frac{\mathcal{D} \vee \mathcal{D}'}{\overline{\mathcal{D} \vee \mathcal{D}'}}$$

Example: *Peter is a man. No man walks*

▷ **Example 9.2.8 (Model Generation).** *Peter is a man. No man walks*

$$\begin{array}{c|c} \multicolumn{2}{c}{\boxed{\mathrm{man}(\mathrm{peter})}} \\ \multicolumn{2}{c}{\mathrm{man}(\mathrm{peter})^{\mathsf{T}}} \\ \multicolumn{2}{c}{\boxed{\neg(\delta U.\mathrm{man}(U) \wedge \mathrm{walks}(U))}} \\ \multicolumn{2}{c}{\neg(\forall U.\mathrm{man}(U) \wedge \mathrm{walks}(U))^{\mathsf{T}}} \\ \multicolumn{2}{c}{(\forall X.\mathrm{man}(X) \wedge \mathrm{walks}(X))^{\mathsf{F}}} \\ \multicolumn{2}{c}{(\mathrm{man}(\mathrm{peter}) \wedge \mathrm{walks}(\mathrm{peter}))^{\mathsf{F}}} \\ \mathrm{man}(\mathrm{peter})^{\mathsf{F}} & \mathrm{walks}(\mathrm{peter})^{\mathsf{F}} \\ \bot & \end{array}$$

Dynamic Herbrand interpretation: $\langle\emptyset, \emptyset, \{\mathrm{man}(\mathrm{peter})^{\mathsf{T}}, \mathrm{walks}(\mathrm{peter})^{\mathsf{F}}\}\rangle$

Example: Anaphor Resolution *A man sleeps. He snores*

▷ **Example 9.2.9 (Anaphor Resolution).** *A man sleeps. He snores*

$$\begin{array}{c|c} \multicolumn{2}{c}{\boxed{\delta U_{\mathrm{Man}}.\mathrm{man}(U) \wedge \mathrm{sleeps}(U)}} \\ \multicolumn{2}{c}{[c^1_{\mathrm{Man}}/U_{\mathrm{Man}}]} \\ \multicolumn{2}{c}{\mathrm{man}(c^1_{\mathrm{Man}})^{\mathsf{T}}} \\ \multicolumn{2}{c}{\mathrm{sleeps}(c^1_{\mathrm{Man}})^{\mathsf{T}}} \\ \multicolumn{2}{c}{\boxed{\delta V_{\mathrm{Man}}.\mathrm{snores}(V)}} \\ [c^1_{\mathrm{Man}}/V_{\mathrm{Man}}] & [c^2_{\mathrm{Man}}/V_{\mathrm{Man}}] \\ \mathrm{snores}(c^1_{\mathrm{Man}})^{\mathsf{T}} & \mathrm{snores}(c^2_{\mathrm{Man}})^{\mathsf{T}} \\ \text{minimal} & \text{deictic} \end{array}$$

Anaphora with World Knowledge

▷ **Example 9.2.10 (Anaphora with World Knowledge).**

- ▷ *Mary is married to Jeff. Her husband is not in town.*
- ▷ $\delta U_{\mathbb{F}}, V_{\mathbb{M}}.U = \mathrm{mary} \wedge \mathrm{married}(U, V) \wedge V = \mathrm{jeff} \,;;\, \delta W_{\mathbb{M}}, W'_{\mathbb{F}}.\mathrm{husband}(W, W') \wedge \neg\mathrm{intown}(W)$
- ▷ World knowledge
  - ▷ If a female X is married to a male Y, then Y is X's only husband.
  - ▷ $\leadsto \forall X_{\mathbb{F}}, Y_{\mathbb{M}}.\mathrm{married}(X, Y) \Rightarrow \mathrm{husband}(Y, X) \wedge (\forall Z.\mathrm{husband}(Z, X) \Rightarrow Z = Y)$

▷ Model generation yields saturated tableau, all branches contain

$$\langle\{U, V, W, W'\}, \{[\mathrm{mary}/U], [\mathrm{jeff}/V], [\mathrm{jeff}/W], [\mathrm{mary}/W']\}, \mathcal{H}\rangle$$

with

$$\mathcal{H} = \{\mathrm{married}(\mathrm{mary}, \mathrm{jeff})^{\mathsf{T}}, \mathrm{husband}(\mathrm{jeff}, \mathrm{mary})^{\mathsf{T}}, \neg\mathrm{intown}(\mathrm{jeff})^{\mathsf{T}}\}$$

▷ They only differ in additional negative facts, e.g. $\mathrm{married}(\mathrm{mary}, \mathrm{mary})^{\mathsf{F}}$.

FAU Michael Kohlhase: LBS 189 2025-02-06

Model Generation models Discourse Understanding

▷ The tableau machine algorithm conforms with psycholinguistic findings:

- ▷ Zwaan& Radvansky [ZR98]: listeners not only represent logical form, but also models containing referents.
- ▷ deVega [de 95]: online, incremental process.
- ▷ Singer [Sin94]: enriched by background knowledge.
- ▷ Glenberg et al. [GML87]: major function is to provide basis for anaphor resolution.

FAU Michael Kohlhase: LBS 190 2025-02-06

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.

Excursion: We will discuss such a dynamic λ calculus in??.

Chapter 10

Propositional Attitudes and Modalities

10.1 Introduction

Modalities and Propositional Attitudes

- **Definition 10.1.1.** Modality is a feature of language that allows for communicating things about, or based on, situations which need not be actual. A sentence is called modal, if it involves a modality
- **Definition 10.1.2.** Modality is signaled by phrases (called moods) that express a speaker's general intentions and commitment to how believable, obligatory, desirable, or actual an expressed proposition is.
- **Example 10.1.3.** Data on modalities (moods in red)
  - $\mathbf{A}$ *probably holds*, (possibilistic)
  - *it has always been the case that* $\mathbf{A}$, (temporal)
  - *it is well-known that* $\mathbf{A}$, (epistemic)
  - $\mathbf{A}$ *is allowed/prohibited*, (deontic)
  - $\mathbf{A}$ *is provable*, (provability)
  - $\mathbf{A}$ *holds after the program* P *terminates*, (program)
  - $\mathbf{A}$ *hods during the execution of* P. (dito)
  - *it is necessary that* $\mathbf{A}$, (aletic)
  - *it is possible that* $\mathbf{A}$, (dito)

Michael Kohlhase: LBS 191 2025-02-06

Modeling Modalities and Propositional Attitudes

- **Example 10.1.4.** Again, the pattern from above:
  - *it is necessary that Peter knows logic* ($\mathbf{A}$ = Peter knows logic)

- *it is possible that John loves logic*, ($\mathbf{A}$ = John loves logic)

- **Observation:** All of the red parts above modify the clause/sentence $\mathbf{A}$. We call them modalities.
- **Definition 10.1.5 (A related Concept from Philosophy).** A propositional attitude is a mental state held by an agent toward a proposition.
- **Question:** But how to model this in logic?
- **Idea:** New sentence-to-sentence operators for *necessary* and *possible*. (extend existing logics with them.)
- **Observation:** $\mathbf{A}$ *is necessary*, iff $\neg\mathbf{A}$ *is impossible*.
- **Definition 10.1.6.** A modal logic is a logical system that has logical constants that model modalities.

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*
- Clarence Irving Lewis 1918 [Lew18] (Systems $S1, \ldots, S5$)
  - strict implication $I(\mathbf{A} \wedge \mathbf{B})$ (I for "impossible")
- Kurt Gödel 1932: Modal logic of provability (S4) [Göd32]
- Saul Kripke 1959-63: Possible worlds semantics [Kri63]
- Vaugham Pratt 1976: Dynamic Program Logic [Pra76]
- $\vdots$

Basic Modal Logics (ML^0 and ML^1)

- **Definition 10.1.7.** 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 10.1.8.** First-order modal logic ML^1 extends first-order logic with two new logical constants: $\Box$ for necessity and $\Diamond$ for possibility.

▷ **Example 10.1.9.** We interpret

1. *Necessarily, every mortal will die.* as $\Box(\forall X.\mathrm{mortal}(X) \Rightarrow \mathrm{willdie}(X))$
2. *Possibly, something is immortal.* as $\Diamond(\exists X.\neg\mathrm{mortal}(X))$

▷ **Questions:** What do $\Box$ and $\Diamond$ mean? How do they behave?

Epistemic and Doxastic Modality

▷ **Definition 10.1.10.** Modal sentences can convey information about the speaker's state of knowledge (epistemic state) or belief (doxastic state).

▷ **Example 10.1.11.** We might paraphrase sentence (2) as (3):

1. A: *Where's John?*
2. B: *He might be in the library.*
3. B': *It is consistent with the speaker's knowledge that John is in the library.*

▷ **Definition 10.1.12.** We way that a world w is an epistemic possibility for an agent B if it could be consistent with B's knowledge.

▷ **Definition 10.1.13.** An epistemic logic is one that models the epistemic state of a speaker. Doxastic logic does the same for the doxastic state.

▷ **Definition 10.1.14.** In deontic logic, we interpret the accessibility relation $\mathcal{R}$ as epistemic accessibility:

- ▷ With this $\mathcal{R}$, represent B's utterance as $\Diamond\mathrm{inlib}(j)$.
- ▷ Similarly, represent *John must be in the library.* as $\Box\mathrm{inlib}(j)$.

▷ **Question:** If $\mathcal{R}$ is epistemic accessibility, what properties should it have?

To determine the properties of epistemic accessibility we ask ourselves, 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?

Deontic Modality

▷ **Definition 10.1.15.** Deontic modality is a modality that indicates how the world ought to be according to certain norms, expectations, speaker desire, etc.

▷ **Definition 10.1.16.** Deontic modality has the following subcategories

- ▷ Commissive modality (the speaker's commitment to do something, like a promise or threat): e.g. *I shall help you.*

- Directive modality (commands, requests, etc.): e.g. *Come!*, *Let's go!*, *You've got to taste this curry!*
- Volitive modality (wishes, desires, etc.): *If only I were rich!*

▷ **Question:** If we want to interpret $\Box\mathrm{runs}(j)$ as *It is required that John runs* (or, more idiomatically, as *John must run*), what formulae should be valid on this interpretation of the operators? (This is for homework!)

10.2 Semantics 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 ?? 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[1]. 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 10.2.1.** We use a set $\mathcal{W}$ of possible worlds, and a accessibility relation $\mathcal{R} \subseteq \mathcal{W} \times \mathcal{W}$: if $\mathcal{R}(v,w)$, then we say that w is accessible from v.

▷ **Example 10.2.2.** $\mathcal{W} = \mathbb{N}$ with $\mathcal{R} = \{\langle n, n+1\rangle \mid n \in \mathbb{N}\}$. (temporal logic)

▷ **Definition 10.2.3.** Variable assignment $\varphi\colon \mathcal{V}_0 \times \mathcal{W} \to \mathcal{D}_0$ assigns values to variables in a given possible world.

▷ **Definition 10.2.4.** Value function $\mathcal{I}_{\cdot}^{\cdot}\colon \mathcal{W} \times \mathit{wff}_0(\mathcal{V}_0) \to \mathcal{D}_0$ (assigns values to formulae in a possible world)

- $\mathcal{I}_\varphi^w(V) = \varphi(w,V)$ for $.V \in \mathcal{V}_0$
- $\mathcal{I}_\varphi^w(\neg\mathbf{A}) = \mathsf{T}$, iff $\mathcal{I}_\varphi^w(\mathbf{A}) = \mathsf{F}$. ($\wedge$ analogous)
- $\mathcal{I}_\varphi^w(\Box\mathbf{A}) = \mathsf{T}$, iff $\mathcal{I}_\varphi^{w'}(\mathbf{A}) = \mathsf{T}$ for all $w' \in \mathcal{W}$ with $w\mathcal{R}w'$.

▷ **Definition 10.2.5.** We call a triple $\mathcal{M} := \langle\mathcal{W}, \mathcal{R}, \mathcal{I}\rangle$ a Kripke model.

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 proposition, 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 proposition, 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:

[1]Note the impossibility of avoiding modal language in the paraphrase!

1. **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 butg 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.

2. **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 temporal 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.

3. **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).

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:

Accessibility Relations. E.g. for Temporal Modalities

▷ **Example 10.2.6 (Temporal Worlds with Ordering).** Let $\langle\mathcal{W},\circ,<,\subseteq\rangle$ an interval time structure, then we can use $\langle\mathcal{W},<\rangle$ as a Kripke models. Then PAST becomes a modal operator.

▷ **Example 10.2.7.** Suppose we have $i<j$ and $j<k$. Then intuitively, if *Jane is laughing* is true at i, then *Jane laughed* should be true at j and at k, i.e. $\mathcal{I}_\varphi^w(j)\mathrm{PAST}(\mathrm{laughs}(j))$ and $\mathcal{I}_\varphi^w(k)\mathrm{PAST}(\mathrm{laughs}(j))$.
But this holds only if "$<$" is transitive. (which it is!)

▷ **Example 10.2.8.** Here is a clearly counter-intuitive claim: For any time i and any sentence $\mathbf{A}$, if $\mathcal{I}_\varphi^w(i)\mathrm{PRES}(\mathbf{A})$ then $\mathcal{I}_\varphi^w(i)\mathrm{PAST}(\mathbf{A})$.
(For example, the truth of *Jane is at the finish line* at i implies the truth of *Jane was at the finish line* at i.)
But we would get this result if we allowed $<$ to be reflexive. ($<$ is irreflexive)

▷ Treating tense modally, we obtain reasonable truth conditions.

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 $w_g(\Box\mathbf{A}) = \mathsf{T}$ implies that $w_g(\mathbf{A}) = \mathsf{T}$ 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 world 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: $\mathbf{A} \Rightarrow \Diamond\mathbf{A}$

Whatever is, is necessarily possible:

1. This also seems like a reasonable slogan. Hence, we want to guarantee the validity of:

$$\mathbf{A} \Rightarrow \Box\Diamond\mathbf{A}$$

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.

Modal Axioms (Propositional Logic)

- **Definition 10.2.9.** Necessitation: $\dfrac{\mathbf{A}}{\Box\mathbf{A}}\,N$
- **Definition 10.2.10 (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

$\mathbb{K}$ Theorems

- **Observation 10.2.11.** $\Box(\mathbf{A} \wedge \mathbf{B}) \models \Box\mathbf{A} \wedge \Box\mathbf{B}$ *in* $\mathbb{K}$.
- **Observation 10.2.12.** $\mathbf{A} \Rightarrow \mathbf{B} \models \Box\mathbf{A} \Rightarrow \Box\mathbf{B}$ *in* $\mathbb{K}$.
- **Observation 10.2.13.** $\mathbf{A} \Rightarrow \mathbf{B} \models \Diamond\mathbf{A} \Rightarrow \Diamond\mathbf{B}$ *in* $\mathbb{K}$.

Translation to First-Order Logic

- **Question:** Is modal logic more expressive than predicate logic?
- **Answer:** Very rarely! (usually can be translated)

▷ **Definition 10.2.14.** Translation τ from ML into PL^1, (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 models in PL^1. (diagram is simple consequence)

▷ **Definition 10.2.15.** 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 10.2.16.** The standard translation τ_w from modal logics to first-order logic is given by the following process:

- ▷ Extend all function constants by a "world argument": $\overline{f} \in \Sigma_{k+1}^f$ for every $f \in \Sigma_k^f$
- ▷ for predicate constants accordingly.
- ▷ insert the "translation world" there: e.g. $\tau_w(f(a,b)) = \overline{f}(w, \overline{a}(w), \overline{b}(w))$.
- ▷ New predicate constant $\mathcal{R}$ for the accessibility relation.
- ▷ New constant s for the "start world".
- ▷ $\tau_w(\Box \mathbf{A}) = \forall w'.w\mathcal{R}w' \Rightarrow \tau_{w'}(\mathbf{A})$.
- ▷ Use all axioms from the respective correspondence theory.

▷ **Definition 10.2.17 (Alternative).** Functional translations, if $\mathcal{R}$ associative:

- ▷ New function constant $f_\mathcal{R}$ for the accessibility relation.
- ▷ Revise the standard translation by one of the following
  - ▷ $\tau_w(\Box \mathbf{A}) = \forall w'.w = f_\mathcal{R}(w') \Rightarrow \tau_w(\mathbf{A})$. (naive solution)
  - ▷ $\tau_{f_\mathcal{R}(w)}(\Box \mathbf{A}) = \tau_w(\mathbf{A})$ (better for mechanizing [Ohl88])

Translation (continued)

▷ **Theorem 10.2.18.** $\tau_s\colon \mathrm{ML}^0 \to \mathrm{PL}^0$ *is correct and complete.*

▷ *Proof:* show that $\exists \mathcal{M}.\mathcal{M} \models \Phi$ iff $\exists \mathcal{M}'.\mathcal{M}' \models \tau_s(\Phi)$

1. Let $\mathcal{M} = \langle \mathcal{W}, \mathcal{R}, \varphi \rangle$ with $\mathcal{M} \models \mathbf{A}$
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 $\mathcal{I}(r) = \mathcal{R}$.

we prove $\mathcal{M} \models_\psi \tau_w(\mathbf{A})'$ *for* $\psi = \mathrm{Id}_\mathcal{W}$ *by structural induction over* $\mathbf{A}$.

3. $\mathbf{A} = P$
   3.1. $\mathcal{I}_\psi(\tau_w(\mathbf{A})) = \mathcal{I}_\psi(\overline{p}(w)) = \mathcal{I}(\overline{p}(w)) = \varphi(P, w) = \mathsf{T}$
4. $\mathbf{A} = \neg\mathbf{B}$, $\mathbf{A} = \mathbf{B} \wedge \mathbf{C}$ *trivial by IH.*
5. $\mathbf{A} = \Box\mathbf{B}$
   5.1. $\mathcal{I}_\psi(\tau_w(\mathbf{A})) = \mathcal{I}_\psi(\forall w.r(w,v) \Rightarrow \tau_v(\mathbf{B})) = \mathsf{T}$, if $\mathcal{I}_\psi(r(w,v)) = \mathsf{F}$ or $\mathcal{I}_\psi(\tau_v(\mathbf{B})) = \mathsf{T}$ for all $v \in \mathcal{W}$.
   5.2. $\mathcal{M} \models_\psi \tau_{v'}(\mathbf{B})$ so by IH $\mathcal{M} \models^v \mathbf{B}$.
   5.3. 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) [HC84].
- David Harel: *Dynamic Logic*, Handbook of Philosophical Logic, D. Gabbay, Hrsg. Reidel (1984) [Har84].
- Johan van Benthem: *Language in Action, Categories, Lambdas and Dynamic Logic*, North Holland (1991) [Ben91].
- Reinhard Muskens, Johan van Benthem, Albert Visser, *Dynamics*, in *Handbook of Logic and Language*, Elsevier, (1995) [MBV95].
- Blackburn, DeRijke, Vedema: *Modal Logic*; 2001 [BRV01]. look at the chapter "Guide to the literature" in the end.

Excursion: We discuss a model existence theorem that can be the basis of completeness proofs for modal logics in??.

10.3 A Multiplicity of Modalities $\rightsquigarrow$ Multimodal Logic

The epistemic and deontic modality 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 modalities for natural language must have a means of representing this relativization.

A Multiplicity of Modalities

- Epistemic (knowledge and belief) modalities must be relativized to an individual
  - *Peter knows that Trump is lying habitually.*
  - *John believes that Peter knows that Trump is lying habitually.*
  - *You must take the written drivers' exam to be admitted to the practical test.*

- Similarly, we find in natural language expressions of necessity and possibility relative to many different kinds of things.
- Consider the deontic (obligatory/permissible) modalities
  - *[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].*
- Many different sorts of modality, sentences are multiply ambiguous towards which one.

In a series of papers beginning with her 1978 dissertation (in German), Angelika Kratzer proposed an account of the semantics of natural language 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: [Kra12; Lew73; Sta68].

Multimodal Logics

- **Definition 10.3.1.** A multimodal logic provides operators for multiple modalities: $[1], [2], [3], \ldots, \langle 1\rangle, \langle 2\rangle, \ldots$
- **Definition 10.3.2.** Multimodal Kripke models provide multiple accessibility relations $\mathcal{R}_1, \mathcal{R}_2, \ldots \subseteq \mathcal{W} \times \mathcal{W}$.
- **Definition 10.3.3.** The value function in multimodal logic generalizes the clause for $\Box$ in ML^0 to
  - $\mathcal{I}_\varphi^w([i]\mathbf{A}) = \mathsf{T}$, iff $\mathcal{I}_\varphi^{w'}(\mathbf{A}) = \mathsf{T}$ for all $w' \in \mathcal{W}$ with $w\mathcal{R}_i w'$.
- **Example 10.3.4 (Epistemic Logic: talking about knowing/believing).** $[peter]\langle klaus\rangle\mathbf{A}$ (Peter knows that Klaus considers $\mathbf{A}$ possible)
- **Example 10.3.5 (Program Logic: talking about programs).** $[X{:=}\mathbf{A}][Y{:=}\mathbf{A}]X = Y$ (after assignments, the values of X and Y are equal)

We will now contrast DRT (see section 35 (Discourse Representation Theory) in the LBS lecture notes) 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.

10.4 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 10.4.1 (Fibonacci).** Consider the following (imperative) program that computes $\mathrm{Fib}(X)$ as the value of Z:
  $\alpha := \langle Y, Z\rangle{:=}\langle 1, 1\rangle \mathbin{;} \textbf{while } X \neq 0 \textbf{ do } \langle X, Y, Z\rangle{:=}\langle X-1, Z, Y+Z\rangle \textbf{ end}$
  - States for the "input" $X = 4$: $\langle 4, \_, \_\rangle, \langle 4, 1, 1\rangle, \langle 3, 1, 2\rangle, \langle 2, 2, 3\rangle, \langle 1, 3, 5\rangle, \langle 0, 5, 8\rangle$
  - **Correctness**? For positive X, running α with input $\langle X, \_, \_\rangle$ we end with $\langle 0, \mathrm{Fib}(X-1), \mathrm{Fib}(X)\rangle$
  - Termination? α does not terminate on input $\langle -1, \_, \_\rangle$.

Multi-Modal Logic fits well

- **Observation:** Multi modal logic fits well
  - States as possible worlds, program statements as accessibility relations.
  - Two syntactic categories: programs α and formulae $\mathbf{A}$.
  - Interpret $[\alpha]\mathbf{A}$ as *If α terminates, then $\mathbf{A}$ holds afterwards*
  - Interpret $\langle\alpha\rangle\mathbf{A}$ as *α terminates and $\mathbf{A}$ holds afterwards*.
- **Example 10.4.2.** Assertions about Fibonacci number (α)
  - $\forall X, Y.[\alpha]Z = \mathrm{Fib}(X)$
  - $\forall X, Y.(X \geq 0) \Rightarrow \langle\alpha\rangle Z = \mathrm{Fib}(X)$

Levels of Description in Dynamic Program Logic

- Propositional dynamic logic (DL^0) (independent of variable assignments)
  - $\models [\alpha]\mathbf{A} \wedge [\alpha]\mathbf{B} \Leftrightarrow [\alpha](\mathbf{A} \wedge \mathbf{B})$
  - $\models [\textbf{while } \mathbf{A} \vee \mathbf{B} \textbf{ do } \alpha \textbf{ end}]\mathbf{C} \Leftrightarrow [\textbf{while } \mathbf{A} \textbf{ do } \alpha \textbf{ end} \mathbin{;} \textbf{while } \mathbf{B} \textbf{ do } \alpha \mathbin{;} \textbf{while } \mathbf{A} \textbf{ do } \alpha \textbf{ end end}]\mathbf{C}$
- First-order program logic (DL^1) (function, predicates uninterpreted)
  - $\models p(f(X)) \Rightarrow g(Y, f(X)) \Rightarrow \langle Z{:=}f(X)\rangle p(Z, g(Y, Z))$
  - $\models Z = Y \wedge (\forall X.f(g(X)) = X) \Rightarrow [\textbf{while } p(Y) \textbf{ do } Y{:=}g(Y) \textbf{ end}]\langle\textbf{while } Y \neq Z \textbf{ do } Y{:=}f(Y) \textbf{ end}\rangle T$
- DL^1 with interpreted functions, predicates (maybe some other time)

- $\forall X.\langle \textbf{while } X \neq 1 \textbf{ do if } even(X) \textbf{ then} X{:=}\frac{X}{2} \textbf{ else } X{:=}3X+1 \textbf{ end}\rangle T$

- **Definition 10.4.3.** We collectively call these dynamic program logics.

DL^0 Syntax

- **Definition 10.4.4.** Propositional dynamic logic (DL^0) is PL^0 extended by
  - program variables $\mathcal{V}_\pi = \{\alpha, \beta, \gamma, \ldots\}$,
  - modalities $[\alpha], \langle\alpha\rangle$.
  - program constructors $\Sigma^\pi = \{;, \cup, *, ?\}$ (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 stop	test

- **Idea:** Standard program primitives as derived concepts

Construct	as
$\textbf{if } \mathbf{A} \textbf{ then} \alpha \textbf{ else } \beta$	$(\mathbf{A}?\,;\alpha) \cup (\neg\mathbf{A}?\,;\beta)$
$\textbf{while } \mathbf{A} \textbf{ do } \alpha \textbf{ end}$	$*(\mathbf{A}?\,;\alpha)\,;\neg\mathbf{A}?$
$\textbf{repeat } \alpha \textbf{ until } \mathbf{A} \textbf{ end}$	$*(\alpha\,;\neg\mathbf{A}?)\,;\mathbf{A}?$

DL^0 Semantics

- **Definition 10.4.5.** A model for DL^0 consists of a set $\mathcal{W}$ of possible worlds called states for DL^0.
- **Definition 10.4.6.** DL^0 variable assignments come in two parts:
  - $\varphi\colon \mathcal{V}_0 \times \mathcal{W} \to \mathcal{D}_0$ (for propositional variables)
  - $\pi\colon \mathcal{V}_\pi \to \mathcal{P}(\mathcal{W} \times \mathcal{W})$ (maps program variables to accessibility relations)
- **Definition 10.4.7.** The meaning of complex formulae is given by the following value function $\mathcal{I}^w_{\varphi,\pi}\colon wff_0(\mathcal{V}_0) \to \mathcal{D}_0$ on formulae:
  - $\mathcal{I}^w_{\varphi,\pi}(V) = \varphi(w, V)$ for $V \in \mathcal{V}_0$.
  - $\mathcal{I}^w_{\varphi,\pi}(\neg\mathbf{A}) = \mathsf{T}$ 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 $w\mathcal{I}_{\varphi,\pi}(\alpha)w'$.

  And $\mathcal{I}_{\varphi,\pi}\colon wff_0(\mathcal{V}_0) \to \mathcal{P}(\mathcal{W} \times \mathcal{W})$ on programs: (independent of $w \in \mathcal{W}$)
  - $\mathcal{I}_{\varphi,\pi}(\alpha) = \pi(\alpha)$. (program variable by assignment)
  - $\mathcal{I}_{\varphi,\pi}(\alpha\,;\beta) = \mathcal{I}_{\varphi,\pi}(\beta) \circ \mathcal{I}_{\varphi,\pi}(\alpha)$ (sequence by composition)

- $\mathcal{I}_{\varphi,\pi}(\alpha \cup \beta) = \mathcal{I}_{\varphi,\pi}(\alpha) \cup \mathcal{I}_{\varphi,\pi}(\beta)$ (distribution by union)
- $\mathcal{I}_{\varphi,\pi}(*\alpha) = \mathcal{I}_{\varphi,\pi}(\alpha)^*$ (iteration by reflexive transitive closure)
- $\mathcal{I}_{\varphi,\pi}(\mathbf{A}?) = \{\langle w, w\rangle \mid \mathcal{I}^w_{\varphi,\pi}(\mathbf{A}) = \mathsf{T}\}$ (test by subset of identity relation)

First-Order Program Logic (DL^1)

- **Observation:** Imperative programs uses variables, function and predicate constants (uninterpreted), but no program variables. The main operation is variable assignment.
- **Idea:** Make a multimodal logic in the spirit of DL^0 that features all of these for a deeper understanding.
- **Definition 10.4.8.** First-order program logic (DL^1) combines the features of PL^1, DL^0 without program variables, with the following two assignment operators:
  - nondeterministic assignment $X{:=}?$
  - deterministic assignment $X{:=}\mathbf{A}$
- **Example 10.4.9.** $\models p(f(X)) \Rightarrow g(Y, f(X)) \Rightarrow \langle Z{:=}f(X)\rangle p(Z, g(Y, Z))$ in DL^1.
- **Example 10.4.10.** In DL^1 we have
  $\models Z = Y \wedge (\forall X.p(f(g(X)) = X)) \Rightarrow [\mathbf{while}\ p(Y)\ \mathbf{do}\ Y{:=}g(Y)\ \mathbf{end}]\langle \mathbf{while}\ Y \neq Z\ \mathbf{do}\ Y{:=}f(Y)\ \mathbf{end}\rangle T$

DL^1 Semantics

- **Definition 10.4.11.** Let $\mathcal{M} = \langle \mathcal{D}, \mathcal{I}\rangle$ be a first-order model then the states (possible worlds) are variable assignments: $\mathcal{W} = \{\varphi \mid \varphi\colon \mathcal{V}_\iota \to \mathcal{D}\}$
- **Definition 10.4.12.** For a set $\mathcal{X}$ of variables, write $\varphi[\mathcal{X}]\psi$, iff $\varphi(X) = \psi(X)$ for all $X \notin \mathcal{X}$.
- **Definition 10.4.13.** The meaning of complex formulae is given by the following value function $\mathcal{I}^w_\varphi\colon \mathit{wff}_o(\Sigma, \mathcal{V}_\iota) \to \mathcal{D}_0$
  - $\mathcal{I}^w_\varphi(\mathbf{A}) = \mathcal{I}_\varphi(\mathbf{A})$ if $\mathbf{A}$ term or atom.
  - $\mathcal{I}^w_\varphi(\neg\mathbf{A}) = \mathsf{T}$ iff $\mathcal{I}^w_\varphi(\mathbf{A}) = \mathsf{F}$
  - $\vdots$
  - $\mathcal{I}_\varphi(X{:=}?) = \{\langle\varphi, \psi\rangle \mid \varphi[X]\psi\}$
  - $\mathcal{I}_\varphi(X{:=}\mathbf{A}) = \{\langle\varphi, \psi\rangle \mid \varphi[X]\psi \text{ and } \psi(X) = \mathcal{I}_\varphi(\mathbf{A})\}$.
- **Observation 10.4.14 (Substitution and Quantification).** *We have*
  - $\mathcal{I}_\varphi([X{:=}\mathbf{A}]\mathbf{B}) = \mathcal{I}_{\varphi,[\mathcal{I}_\varphi(\mathbf{A})/X]}(\mathbf{B})$
  - $\forall X.\mathbf{A} = [X{:=}?]\mathbf{A}$.

▷ Thus substitutions and quantification are definable in DL^1.

Natural Deduction for DL^0

▷ **Definition 10.4.15.** The natural deduction calculus $\mathcal{DND}_0$ for DL^0 contains the inference rules from $\mathcal{ND}_0$ plus:

Introduction

$$\frac{[\alpha][\beta]\mathbf{A}}{[\alpha\,;\,\beta]\mathbf{A}}\;\mathcal{DND}_0;I$$

$$\frac{[\alpha]\mathbf{A}\quad[\beta]\mathbf{A}}{[\alpha\cup\beta]\mathbf{A}}\;\mathcal{DND}_0\cup I$$

$$\frac{[\alpha]^0\mathbf{A}\quad\ldots\quad[\alpha]^n\mathbf{A}\quad\text{for all } n\in\mathbb{N}}{[*\alpha]\mathbf{A}}\;\mathcal{DND}_0*I$$

$$\frac{\begin{array}{c}[\mathbf{A}]^1\\ \hline\hline \mathbf{B}\end{array}}{[\mathbf{A}?]\mathbf{B}}\;\mathcal{DND}_0?I^1$$

Elimination

$$\frac{[\alpha\,;\,\beta]\mathbf{A}}{[\alpha][\beta]\mathbf{A}}\;\mathcal{DND}_0;E$$

$$\frac{[\alpha\cup\beta]\mathbf{A}}{[\alpha]\mathbf{A}}\;\mathcal{DND}_0\cup E_l\qquad\frac{[\alpha\cup\beta]\mathbf{A}}{[\beta]\mathbf{A}}\;\mathcal{DND}_0\cup E_r$$

$$\frac{[*\alpha]\mathbf{A}\quad n\in\mathbb{N}}{[\alpha]^n\mathbf{A}}\;\mathcal{DND}_0*E$$

$$\frac{[\mathbf{A}?]\mathbf{B}\quad\mathbf{A}}{\mathbf{B}}\;\mathcal{DND}_0?E$$

For details see [HM95].

Natural Deduction for DL^1

▷ **Definition 10.4.16.** The natural deduction calculus $\mathcal{DND}_1$ for DL^1 contains the inference rules from $\mathcal{ND}^1$ and $\mathcal{DND}_0$ plus:

$$\frac{[\mathbf{A}/X](\mathbf{B})\quad X\notin(\mathrm{free}(\mathbf{A})\cup\mathrm{free}(\mathbf{B}))}{[X{:=}\mathbf{A}]\mathbf{B}}\;\mathcal{DND}_0{:=}I$$

$$\frac{[X{:=}\mathbf{A}]\mathbf{B}\qquad\begin{array}{c}[[\mathbf{A}/X](\mathbf{B})]^1\\ \hline\hline \mathbf{C}\end{array}}{\mathbf{C}}\;\mathcal{DND}_0{:=}E$$

For details see [HM95].

▷ **Observation:** No inference rules for $:=?$ needed as $\forall X.\mathbf{A} = [X{:=}?]\mathbf{A}$ $\leadsto$ $\mathcal{ND}^1\forall I$ and $\mathcal{ND}^1\forall E$ suffice.

Natural Language as Programming Languages

▷ **Question:** Why is dynamic program logic interesting in a natural language semantics course?

▷ **Answer:** There are fundamental relations between dynamic (discourse) logics and dynamic program logics.

▷ **David Israel:** "*Natural languages are programming languages for mind*" [Isr93]

Chapter 11

Some Issues in the Semantics of Tense

Tense as a Deictic Element

- **Goal:** Capturing 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.*
- **Observation 11.0.1.** *Tense is a deictic element, i.e. its interpretation requires reference to something outside the sentence itself.*
- **Remark:** Often, in particular in the case of monoclausal sentences occurring in isolation, as in our examples, this "something" is the speech time.
- **Idea:** make use of the reference time *now*:
  - *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.

Michael Kohlhase: LBS 217 2025-02-06

A Simple Semantics for Tense

- **Problem:** The meaning of *Jane saw George* and *Jane will see George* is defined in terms of *Jane sees George*.

  $\leadsto$ We need the truth conditions of the present tense sentence.
- **Idea:** *Jane sees George* is true at a time iff Jane sees George at that time.
- **Implementation:** Postulate temporal operator as sentential operators (expressions of type $o \to o$). Interpret

1. *Jane saw George* as $\mathrm{PAST}(\mathrm{see}(g,j))$,
2. *Jane sees George* as $\mathrm{PRES}(\mathrm{see}(g,j))$, and
3. *Jane wil see George* as $\mathrm{FUT}(\mathrm{see}(g,j))$.

Some notes:

- Most treatments of the semantics of tense invoke some notion of a tenseless proposition/formula for the base case, just like we do. The idea here is that markers of past, present and future all operate on an underlying un-tenseed expression, which can be evaluated for truth at a time.
- Note that we 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, we 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.

Models and Evaluation for a Tensed Language

- **Problem:** The interpretations of constants vary over time.
- **Idea:** Introduce times into our models, and let the interpretation function give values of constants at a time. Relativize the valuation function to times
- **Idea:** We will consider temporal structures, where denotations are constant on intervals.
- **Definition 11.0.2.** Let $I \subseteq \{[i,j] \mid i,j \in \mathbb{R}\}$ be a set of real intervals, then we call $\langle I, \circ, <, \subseteq \rangle$ an interval time structure, where for intervals $i := [i_l,i_l]$ and $j := [l_l,j_r]$ we say that
  - i and j overlap (written $i \circ j$), iff $l_l \leq ir$,
  - i precedes j (written $i < j$), iff $ir \leq l_l$, and
  - i is contained in j (written $i \subseteq j$), iff $l_l \leq i_l$ and $ir \leq j_r$.
- **Definition 11.0.3.** A temporal model is a triple $\langle \mathcal{D}, \mathbb{I}, \mathcal{I} \rangle$, where
  - $\mathcal{D}$ is a set called the domain,
  - $\mathbb{I}$ is an interval time structure, and
  - $\mathcal{I} \colon \mathbb{I} \times \Sigma_{\mathcal{T}} \to \mathcal{D}$ an interpretation function.

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 intuitively time is given by as 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 $<$.

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 in our interval time structure. 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.

Interpretation rules for the temporal operators

- **Definition 11.0.4.** For the value function $\mathcal{I}_\varphi^i(\cdot)$ we only redefine the clause for constants:
  - $\mathcal{I}_\varphi^i(c):=\mathcal{I}^i(c)$
  - $\mathcal{I}_\varphi^i(X):=\varphi(X)$
  - $\mathcal{I}_\varphi^i(\mathbf{FA}):=\mathcal{I}_\varphi^i(\mathbf{F})(\mathcal{I}_\varphi^i(\mathbf{A}))$.
- **Definition 11.0.5.** We define the meaning of the temporal operators:
  1. $\mathcal{I}_\varphi^i(\mathrm{PRES}(\Phi)) = \mathsf{T}$, iff $\mathcal{I}_\varphi^i(\Phi) = \mathsf{T}$.
  2. $\mathcal{I}_\varphi^i(\mathrm{PAST}(\Phi)) = \mathsf{T}$ iff there is an interval $j \in \mathbb{I}$ with $j < i$ and $\mathcal{I}_\varphi^j(\Phi) = \mathsf{T}$.
  3. $\mathcal{I}_\varphi^i(\mathrm{FUT}(\Phi)) = \mathsf{T}$ iff there is an interval $j \in \mathbb{I}$ with $i < j$ and $\mathcal{I}_\varphi^j(\Phi) = \mathsf{T}$.

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)*.

Perfective vs. imperfective

- **Data:**
  - *Jane left.*
  - *Jane was leaving.*
- **Question:** 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.*

Future Readings of Present Tense

▷ **New Data:**

1. *Jane leaves tomorrow.*
2. *Jane is leaving tomorrow.*
3. ?? *It rains tomorrow.*
4. ?? *It is raining tomorrow.*
5. ?? *The dog barks tomorrow.*
6. ??*The dog is barking tomorrow.*

▷ Future readings of present tense appear to arise only when the event described is planned, or planable, either by the subject of the sentence, the speaker, or a third party.

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 precedes 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.

Sequence of Tense (continued)

- ▷ *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)

Interpreting Tense in Discourse

- ▷ **Example 11.0.6 (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 11.0.7 (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.*

Chapter 12

Quantifier Scope Ambiguity and Underspecification

12.1 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.

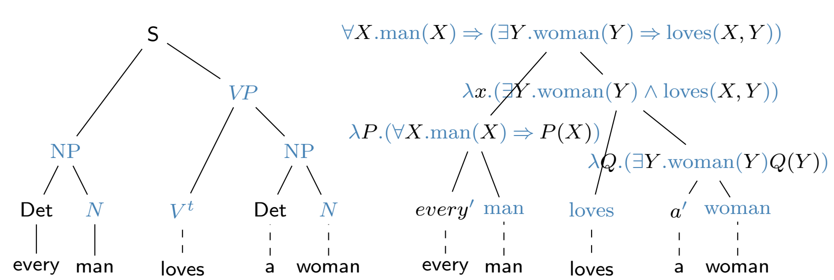
Quantifier Scope Ambiguities: Data

▷ 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.*

▷ **Definition 12.1.1.** We call these systematic ambiguities quantifyer scope ambiguities

▷ **Example 12.1.2.** We can represent the "wide-scope" reading with our methods



▷ **Question:** How to map an unambiguous input structure to multiple translations.

This is a correct representation of one of the possible readings 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 readings 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 $=_\beta$-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 $=_\beta$-reduction.

In the VP node for *loves a woman* we have $(\lambda F X.\lambda Q.(\exists Y.\mathrm{woman}(Y) \wedge Q\,Y))\ \mathrm{loves}$ and thus the sentence representation is

$$(\lambda P.(\forall X.\mathrm{man}(X) \Rightarrow P(X)))\ (\lambda F X.\lambda Q.(\exists Y.\mathrm{woman}(Y) \wedge Q\,Y))\ \mathrm{loves}$$

The resulting expression is an application of form $\langle \mathrm{everyman}\rangle(\langle \mathrm{awoman}\rangle(\langle \mathrm{loves}\rangle))$. 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 $=_\beta$-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 $\langle \mathrm{awoman}\rangle(\langle \mathrm{everyman}\rangle(\langle \mathrm{loves}\rangle))$ would do.

Storing and Quantifying In

- **Analysis:** The sentence meaning is of the form $\langle \mathrm{everyman}\rangle(\langle \mathrm{awoman}\rangle(\langle \mathrm{loves}\rangle))$
- **Idea:** Somehow have to move the *a woman* part in front of the *every* to obtain

$$\langle \mathrm{awoman}\rangle(\langle \mathrm{everyman}\rangle(\langle \mathrm{loves}\rangle))$$

- **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* out-scopes *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.

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 semantics 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 12.1.3.** **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 12.1.4.** **QI**$(\langle \Phi, \Sigma * NP \rangle) \to \langle NP \oplus \Phi, \Sigma \rangle$ 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.

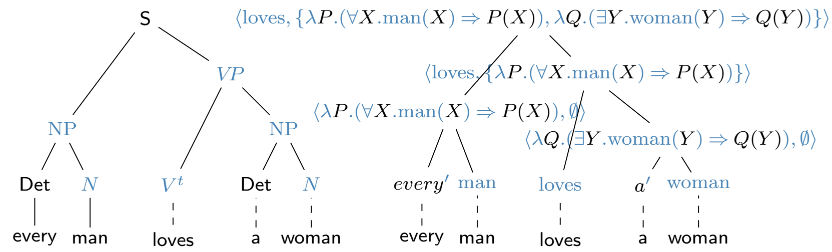
We now have more than one way to translate a branching node, but the choice is partly constrained by whether or not the daughters of the node have empty stores. We have the following two options for translating a branching node. (Note: To simplify the notation, let us adopt the following convention: 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 semantics construction 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 **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 processing 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 loves. To compute the semantics of the VP nodes, we use the rule **STORE** and obtain $\langle \mathrm{loves}, \{\mathbf{A}\} \rangle$ and similarly $\langle \mathrm{loves}, \{\mathbf{A}, \mathbf{B}\} \rangle$ for the for the S node, thus we have the following semantics tree.

Quantifying in Practice: *Every man loves a woman*

▷ **Example 12.1.5.**



▷ Continue with **QI** applications: first retrieve $\lambda Q.(\exists Y.\mathrm{woman}(Y) \Rightarrow Q(Y))$

$$
\begin{array}{ll}
 & \langle \mathrm{loves}, \{\lambda P.(\forall X.\mathrm{man}(X) \Rightarrow P(X)), \lambda Q.(\exists Y.\mathrm{woman}(Y) \Rightarrow Q(Y))\}\rangle \\
\to_{QI} & \langle \circ\ (\lambda P.(\forall X.\mathrm{man}(X) \Rightarrow P(X)))\ \mathrm{loves}, \{\lambda Q.(\exists Y.\mathrm{woman}(Y) \Rightarrow Q(Y))\}\rangle \\
\to_{\beta} & \langle \lambda Z.(\lambda P.(\forall X.\mathrm{man}(X) \Rightarrow P(X)))\ \mathrm{loves}\ Z, \{\lambda Q.(\exists Y.\mathrm{woman}(Y) \Rightarrow Q(Y))\}\rangle \\
\to_{\beta} & \langle \lambda Z.(\forall X.\mathrm{man}(X) \Rightarrow \mathrm{loves}\ Z\ X), \{\lambda Q.(\exists Y.\mathrm{woman}(Y) \Rightarrow Q(Y))\}\rangle \\
\to_{QI} & \langle (\lambda Q.(\exists Y.\mathrm{woman}(Y) \Rightarrow Q(Y)))\ (\lambda Z.(\forall X.\mathrm{man}(X) \Rightarrow \mathrm{loves}\ Z\ X)), \emptyset\rangle \\
\to_{\beta} & \langle \exists Y.\mathrm{woman}(Y) \Rightarrow (\lambda Z.(\forall X.\mathrm{man}(X) \Rightarrow \mathrm{loves}\ Z\ X))\ Y, \emptyset\rangle \\
\to_{\beta} & \langle \exists Y.\mathrm{woman}(Y) \Rightarrow (\forall X.\mathrm{man}(X) \Rightarrow \mathrm{loves}\ Y\ X), \emptyset\rangle
\end{array}
$$

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.

12.2 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 description. 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 $(\iota \to o) \to o$ (and are applied to the VP) whereas subject NPs with proper names have type ι. (argument to the VP)

▷ **Idea:** *John runs* translates to $\mathrm{runs}(\mathrm{john})$, where $\mathrm{runs} \in \Sigma_{\iota \to o}$ and $\mathrm{john} \in \Sigma_{\iota}$.

Now we $=_{\beta}$-expand over the VP yielding $(\lambda P_{\iota \to o}.P(\mathrm{john}))\ \mathrm{runs}$

$\lambda P_{\iota \to o}.P(\mathrm{john})$ has type $(\iota \to o) \to o$ and can be applied to the VP runs.

▷ **Definition 12.2.1.** If $c \in \Sigma_{\alpha}$, then type raising c yields $\lambda P_{\alpha \to o}.P\ c$.

Definite NPs

▷ **Problem:** On our current assumptions, $the' = \iota$, and so for any definite NP *the*

N, its translation is $\iota\ N$, an expression of type ι.

▷ **Idea:** Type lift just as we did with proper names: $\iota\ N$ 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 $a \in \mathcal{D}_{\iota\to o}$: $\mathcal{I}(the')(a) = \mathcal{I}(every')(a)$ if $\#(a) = 1$, undefined otherwise

So the' is that function in $\mathcal{D}_{(\iota\to o)\to(\iota\to o)\to o}$ such that 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

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:

   $$\{\langle X, Y\rangle \,|\, \#(X) = 1 \ \&\ X \subseteq Y\}$$

   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*. loves 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).

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 $\mathrm{HOL}^{\rightarrow}$ by a constant $\circ_{(\beta\rightarrow\gamma)\rightarrow(\alpha\rightarrow\beta)\rightarrow\alpha\rightarrow\gamma}$ by setting $\circ := \lambda FGX.F(G(X))$ thus

$$\circ\; g\; f \rightarrow_\beta \lambda X.g(f(X)) \qquad \text{and} \qquad \circ\; g\; f\; a \rightarrow_\beta g(f(a))$$

In our example, we have

$$\begin{aligned}
\circ\;(\lambda P.P(\mathrm{john}))\;\mathrm{loves} &=_{Def} (\lambda FGX.F(G(X)))\;(\lambda P.P(\mathrm{john}))\;\mathrm{loves}\\
&\rightarrow_\beta (\lambda GX.(\lambda P.P(\mathrm{john}))\;G(X))\;\mathrm{loves}\\
&\rightarrow_\beta \lambda X.(\lambda P.P(\mathrm{john}))\;\mathrm{loves}\;X\\
&\rightarrow_\beta! \lambda X.\mathrm{loves}(X,\mathrm{john})
\end{aligned}$$

Definition 12.2.2 (Function Composition). Let $f\colon A\rightarrow B$ and $g\colon B\rightarrow C$ be functions, then we call the function $h\colon A\rightarrow C$ 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").

12.3 Dealing with Quantifier Scope Ambiguity: Cooper Storage

Type raising transitive verbs

- We need transitive verbs to combine with quantificational objects of type $(\iota\rightarrow o)\rightarrow o$ but ...
- We still ultimately want their "basic" translation to be type $\iota\rightarrow\iota\rightarrow o$, 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*:

$$\lambda PY.P\;(\lambda X.\mathrm{likes}(X,Y))$$

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 [CKG09, pp.178-179])

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

- **Intuition:** 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 12.3.1.** 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 operators (because we will use them to bind free variables in the core representation).
- **Definition 12.3.2.** In the Cooper storage method, QNPs are stored in the store and later retrieved – not necessarily in the order they were stored – to build the 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.

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.

Working with Stores

▷ Working out the translation for *Every student likes some professor.*

$$
\begin{array}{lll}
NP_1 & \rightarrow & \lambda P.(\exists X.\mathrm{prof}(X) \wedge P(X)) \text{ or } \langle \lambda Q.Q(X_1), (\lambda P.(\exists X.\mathrm{prof}(X) \wedge P(X)), 1)\rangle \\
V_t & \rightarrow & \lambda RY.R\ (\lambda Z.\mathrm{likes}(Z,Y)) \\
VP & \rightarrow & \text{(Combine core representations by FA; pass store up)*} \\
 & \rightarrow & \langle \lambda Y.\mathrm{likes}(X_1,Y), (\lambda P.(\exists X.\mathrm{prof}(X) \wedge P(X)), 1)\rangle \\
NP_2 & \rightarrow & \lambda P.(\forall Z.\mathrm{student}(Z) \Rightarrow P(Z)) \text{ or } \langle \lambda R.R(X_2), (\lambda P.(\forall Z.\mathrm{student}(Z) \Rightarrow P(Z)), 2)\rangle \\
S & \rightarrow & \text{(Combine core representations by FA; pass stores up)**} \\
 & \rightarrow & \langle \mathrm{likes}(X_1,X_2), (\lambda P.(\exists X.\mathrm{prof}(X) \wedge P(X)), 1), (\lambda P.(\forall Z.\mathrm{student}(Z) \Rightarrow P(Z)), 2)\rangle
\end{array}
$$

* Combining V_t with place holder

1. $(\lambda RY.R\ (\lambda Z.\mathrm{likes}(Z,Y)))\ (\lambda Q.Q(X_1))$
2. $\lambda Y.(\lambda Q.Q(X_1))\ (\lambda Z.\mathrm{likes}(Z,Y))$
3. $\lambda Y.(\lambda Z.\mathrm{likes}(Z,Y))\ X_1$
4. $\lambda Y.\mathrm{likes}(X_1,Y)$

** Combining VP with place holder

1. $(\lambda R.R(X_2))\ (\lambda Y.\mathrm{likes}(X_1,Y))$
2. $(\lambda Y.\mathrm{likes}(X_1,Y))\ X_2$
3. $\mathrm{likes}(X_1,X_2)$

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 $\iota \rightarrow o$.)
3. Apply this λ-term to the QNP translation (which is of type $(\iota \rightarrow o) \rightarrow o$).

Example: *Every student likes some professor.*

1. Retrieve *every student*

(a) $(\lambda Q.(\forall Z.\text{student}(Z) \Rightarrow Q(Z)))\ (\lambda X_2.\text{likes}(X_1, X_2))$

(b) $\forall Z.\text{student}(Z) \Rightarrow (\lambda X_2.\text{likes}(X_1, X_2))\ Z$

(c) $\forall Z.\text{student}(Z) \Rightarrow \text{likes}(X_1, Z)$

2. Retrieve *some professor*

(a) $(\lambda P.(\exists X.\text{prof}(X) \wedge P(X)))\ (\lambda X_1.(\forall Z.\text{student}(Z) \Rightarrow \text{likes}(X_1, Z)))$

(b) $\exists X.\text{prof}(X)(\lambda X_1.(\forall Z.\text{student}(Z) \Rightarrow \text{likes}(X_1, Z)))\ X$

(c) $\exists X.\text{prof}(X) \wedge (\forall Z.\text{student}(Z) \Rightarrow \text{likes}(X, Z))$

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.

12.4 Underspecification

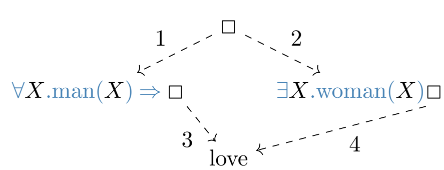
In this section we introduce Johan Bos' "Hole Semantics", since this is possibly the simplest underspecification framework around. The main idea is that the result of the translation is a "quasi-logical form" (QLF), i.e. a representation that represents all possible readings. This QLF can then be used for semantic/pragmatic analysis.

12.4.1 Unplugging Predicate Logic

The problem we need to solve for our QLF is that regular logical formulae, such as

$$\forall X.\text{man}(X) \Rightarrow (\exists Y.\text{woman}(Y) \Rightarrow \text{loves}(Y, X))$$

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 $\Box$) 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* out-scopes *every man*. So much for the general idea, how can this be represented in logic?

12.4.2 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.

12.4.3 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	$[\ell_1/h_0], [\ell_2/h_1], [\ell_3/h_2]$	$\forall X.\mathrm{man}(X) \Rightarrow (\exists Y.\mathrm{woman}(Y) \wedge \mathrm{loves}(X, Y))$
2	$[\ell_2/h_0], [\ell_3/h_1], [\ell_1/h_2]$	$\exists Y.\mathrm{woman}(Y) \Rightarrow (\forall X.\mathrm{man}(X) \wedge \mathrm{loves}(X, Y))$

Chapter 13

Higher-Order Unification and NL Semantics Reconstruction

13.1 Introduction

Application of HOL in NL Semantics: Ellipsis

- **Example 13.1.1.** *John loves his wife. George does too*
  - $\mathrm{loves}(\mathrm{john}, \mathrm{wifeof}(\mathrm{john})) \wedge Q(\mathrm{george})$
  - "*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:** $Q(\mathrm{john}) =_{\alpha\beta\eta} \mathrm{loves}(\mathrm{john}, \mathrm{wifeof}(\mathrm{john}))$
- **Solutions (computed by HOU):**
  - $Q = \lambda z.\mathrm{loves}(z, \mathrm{wifeof}(z))$ and $Q = \lambda z.\mathrm{loves}(z, \mathrm{wifeof}(\mathrm{john}))$
  - * $Q = \lambda z.\mathrm{loves}(\mathrm{john}, \mathrm{wifeof}(z))$ and $Q = \lambda z.\mathrm{loves}(\mathrm{john}, \mathrm{wifeof}(\mathrm{john}))$
- **Readings:** *George loves his own wife.* and *George loves John's wife.*
- **Erraneous HOU Predictions:** * *John loves George's wife.* and * *John loves John's wife.*

FAU Michael Kohlhase: LBS 241 2025-02-06

Higher-Order Unification (HOU)

- **Intuitively:** Equation solving in the simply typed λ-calculus (modulo the built-in $\alpha\beta\eta$-equality)
- **Formally:** Given formulae $\mathbf{A}, \mathbf{B} \in wff_\alpha(\Sigma_{\mathcal{T}}, \mathcal{V}_{\mathcal{T}})$, find a substitution σ with $\sigma(\mathbf{A}) =_{\alpha\beta\eta} \sigma(\mathbf{B})$.
- **Definition 13.1.2.**

  We call $\mathcal{E} := \mathbf{A}_1 =^? \mathbf{B}_1 \wedge \ldots \wedge \mathbf{A}_n =^? \mathbf{B}_n$ a unification problem. The set $\mathbf{U}(\mathcal{E}) =$

$\{\sigma \mid \sigma(\mathbf{A}_i) =_{\alpha\beta\eta} \sigma(\mathbf{B}_i)\}$ is called the set of unifiers for $\mathcal{E}$ and any of its members a unifier.

▷ **Example 13.1.3.** the unification problem $F(fa)=^{?}f(Fa)$ where $F, f{:}\alpha \rightarrow \alpha$ 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?)

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"

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.*

Discourse Relations (The General Case)

▷ We need inferences to discover them

▷ General conditions [Hobbs 1990]

Relation	Requirements	Particle
Parallel	$a_i \sim b_i, p \mathrel{=\!\!|\!\!=} q$	*and*
Contrast	$a_i \sim b_i$, $p \models \neg q$ or $\neg p \models q$ a_i, b_i contrastive	*but*
Exempl.	$p \mathrel{=\!|} q$, $a_i \in \vec{b}$ or $a_i \mathrel{=\!|} b_i$	*for example*
Generl.	$p \mathrel{=\!|} q$, $b_i \in \vec{a}$ or $b_i \mathrel{=\!|} a_i$	*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.

13.2 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.

13.2.1 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 **??** 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 13.2.1.** $\sigma{=_{\beta\eta}}\rho[W]$, iff $\sigma(X) =_{\alpha\beta\eta} \rho(X)$ for all $X \in W$. $\sigma{=_{\beta\eta}}\rho[\mathcal{E}]$ iff $\sigma{=_{\beta\eta}}\rho[\mathrm{free}(\mathcal{E})]$
- **Definition 13.2.2.** 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 13.2.3.** $\Psi \subseteq \mathbf{U}(\mathcal{E})$ 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 13.2.4.** If $\Psi \subseteq \mathbf{U}(\mathcal{E})$ is complete, then $\leq_\beta$-minimal elements $\sigma \in \Psi$ are most general unifier of $\mathcal{E}$.
- **Theorem 13.2.5.** *The set* $\{[\lambda uv.h\ u/F]\} \cup \{\sigma_i \,|\, i \in \mathbb{N}\}$ *where*

  $$\sigma_i{:=}[\lambda uv.g_n\ u\ u\ h_1^n\ u\ v\ \ldots\ u\ h_n^n\ u\ v/F], [\lambda v.z/X]$$

  is a complete set of unifiers for the equation $F\ X\ (a_\iota){=^?}F\ X\ (b_\iota)$, *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]

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 13.2.6.** $X^1{=^?}\mathbf{B}^1 \wedge \ldots \wedge X^n{=^?}\mathbf{B}^n$ is in solved form, if the X^i are distinct free variables $X^i \notin \mathrm{free}(\mathbf{B}^j)$ and $\mathbf{B}^j$ does not contain Skolem constants for all j.
- **Lemma 13.2.7.** *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:*

1. $\sigma(X^i) =_{\alpha\beta\eta} \sigma(\mathbf{B}^i)$, so $\sigma \in \mathbf{U}(\mathcal{E})$
2. Let $\theta \in \mathbf{U}(\mathcal{E})$, then $\theta(X^i) =_{\alpha\beta\eta} \theta(\mathbf{B}^i) = \theta \circ \sigma(X^i)$
3. so $\theta \leq_{\beta\eta} (\theta \circ \sigma)[\mathcal{E}]$.

13.2.2 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 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 13.2.8.** The higher order simplification transformations $\mathcal{SIM}$ consist of the rules below.

$$\frac{(\lambda X_\alpha.\mathbf{A}) =^? (\lambda Y_\alpha.\mathbf{B}) \wedge \mathcal{E} \quad s \in \Sigma_\alpha^{Sk} \text{new}}{([s/X](\mathbf{A})) =^? ([s/Y](\mathbf{B})) \wedge \mathcal{E}} \mathcal{SIM}{:}\alpha$$

$$\frac{(\lambda X_\alpha.\mathbf{A}) =^? \mathbf{B} \wedge \mathcal{E} \quad s \in \Sigma_\alpha^{Sk} \text{new}}{([s/X](\mathbf{A})) =^? \mathbf{B}s \wedge \mathcal{E}} \mathcal{SIM}{:}\eta$$

$$\frac{(h\ \overline{\mathbf{U}^n}) =^? (h\ \overline{\mathbf{V}^n}) \wedge \mathcal{E} \quad h \in (\Sigma \cup \Sigma^{Sk})}{\mathbf{U}_1 =^? \mathbf{V}_1 \wedge \ldots \wedge \mathbf{U}_n =^? \mathbf{V}_n \wedge \mathcal{E}} \mathcal{SIM}{:}\text{dec}$$

$$\frac{\mathcal{E} \wedge X =^? \mathbf{A} \quad X \notin \text{free}(\mathbf{A}) \quad \mathbf{A} \cap \Sigma^{Sk} = \emptyset \quad X \in \text{free}(\mathcal{E})}{[\mathbf{A}/X](\mathcal{E}) \wedge X =^? \mathbf{A}} \mathcal{SIM}{:}\text{elim}$$

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 $\alpha\beta\eta$-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 $\mathcal{SIM}{:}\text{elim}$ 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 13.2.9 (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 13.2.10.** $\mathbf{U}(\mathcal{E} \wedge \mathcal{E}_\sigma) = \mathbf{U}(\sigma(\mathcal{E}) \wedge \mathcal{E}_\sigma)$.
- *Proof:* by the definitions
  1. If $\theta \in \mathbf{U}(\mathcal{E} \wedge \mathcal{E}_\sigma)$, then $\theta \in (\mathbf{U}(\mathcal{E}) \cap \mathbf{U}(\mathcal{E}_\sigma))$.
  2. So $\theta =_{\beta\eta} (\theta \circ \sigma)[\mathrm{supp}(\sigma)]$,
  3. and thus $\theta \circ \sigma \in \mathbf{U}(\mathcal{E})$, iff $\theta \in \mathbf{U}(\sigma(\mathcal{E}))$.
- **Theorem 13.2.11.** *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

  We the $\mathcal{SIM}$ rules individually for the base case
  1. $\mathcal{SIM}{:}\alpha$ *by $\alpha\beta\eta$-conversion*
  2. $\mathcal{SIM}{:}\eta$ *By η-conversion in the presence of $\mathcal{SIM}{:}\alpha$*
  3. $\mathcal{SIM}{:}\mathrm{dec}$ *The head $h \in (\Sigma \cup \Sigma^{Sk})$ cannot be instantiated.*
  4. $\mathcal{SIM}{:}\mathrm{elim}$ *By* **??**.
  5. The step case goes directly by induction hypothesis and transitivity of the derivation relation.

Now that we have simplifiation 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.
- **Definition 13.2.12 (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.

  is called the general binding of type α for the head h.
- **Observation 13.2.13.**

General bindings are unique up to choice of names for H^i.

▷ **Definition 13.2.14.** If the head h is j^{th} bound variable in $\mathbf{G}_\alpha^h(\Sigma)$, call $\mathbf{G}_\alpha^h(\Sigma)$ j-projection binding (and write $\mathbf{G}_\alpha^j(\Sigma)$) else imitation binding

▷ clearly $\mathbf{G}_\alpha^h(\Sigma) \in wff_\alpha(\Sigma_\mathcal{T}, \mathcal{V}_\mathcal{T})$ and $\mathrm{head}(\mathbf{G}_\alpha^h(\Sigma)) = h$

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 13.2.15. The following general bindings may be helpful: $\mathbf{G}_{(\iota\to\iota)}^{(a_\iota)}(\Sigma) = \lambda X_\iota.a$, $\mathbf{G}_{(\iota\to\iota\to\iota)}^{(a_\iota)}(\Sigma) = \lambda X_\iota Y_\iota.a$, and $\mathbf{G}_{(\iota\to\iota\to\iota)}^{(a_{\iota\to\iota})}(\Sigma) = \lambda X_\iota Y_\iota.a(HXY)$, where H is of type $\iota\to\iota\to\iota$

We will now show that the general bindings defined in Definition 16.14 (Higher-Order Unification Transformations) in the LBS lecture notes are indeed the most general λ-terms given their type and head symbol.

Approximation Theorem

▷ **Theorem 13.2.16.** *If $\mathbf{A} \in wff_\alpha(\Sigma_\mathcal{T}, \mathcal{V}_\mathcal{T})$ with $\mathrm{head}(\mathbf{A}) = h$, then there is a general binding $\mathbf{G} = \mathbf{G}_\alpha^h(\Sigma)$ and a substitution ρ with $\rho(\mathbf{G}) =_{\alpha\beta\eta} \mathbf{A}$ and $\mathrm{dp}\rho < \mathrm{dp}\mathbf{A}$.*

▷ *Proof:* We analyze the term structure of $\mathbf{A}$

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 $\lambda \overline{X_\alpha^k}.h\,\overline{\mathbf{U}^n}$.
2. $\mathbf{G} = \mathbf{G}_\alpha^h(\Sigma) = \lambda \overline{X_\alpha^k}.h(H_1\,\overline{X})\ldots(H_n\,\overline{X})$ for some variables $H_i{:}\overline{\alpha_k} \to \gamma_i$.
3. Choose $\rho := [\lambda \overline{X_\alpha^k}.\mathrm{U}_1/H_1], \ldots, [\lambda \overline{X_\alpha^k}.\mathrm{U}_n/H_n]$.
4. Then we have $$\begin{aligned}\rho(\mathbf{G}) &= \lambda \overline{X_\alpha^k}.h(\lambda \overline{X_\alpha^k}.\mathrm{U}_1\,\overline{X})\ldots(\lambda \overline{X_\alpha^k}.\mathrm{U}_n\,\overline{X})\\ &=_{\beta\eta} \lambda \overline{X_\alpha^k}.h\,\overline{\mathbf{U}^n}\\ &=_{\beta\eta} \mathbf{A}\end{aligned}$$
5. The depth condition can be read off as $\mathrm{dp}(\lambda \overline{X_\alpha^k}.\mathrm{U}_1) \leq \mathrm{dp}\mathbf{A} - 1$.

With this result we can state the higher-order unification transformations.

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 13.2.17.** Let $\mathbf{G} = \mathbf{G}_\alpha^h(\Sigma)$ (imitation) or $\mathbf{G} \in \{\mathbf{G}_\alpha^j(\Sigma) \mid 1 \leq j \leq n\}$, then the calculus $\mathcal{HOU}$ for higher-order unification consists of the transformations (always reduce to $\mathcal{SIM}$ normal form)

▷ Rule for flex/rigid pairs: $$\frac{(F_\alpha\,\overline{\mathbf{U}}) =^? (h\,\overline{\mathbf{V}}) \land \mathcal{E}}{F =^? \mathbf{G} \land (F\,\overline{\mathbf{U}}) =^? (h\,\overline{\mathbf{V}}) \land \mathcal{E}}\,\mathcal{HOU}{:}\mathrm{fr}$$

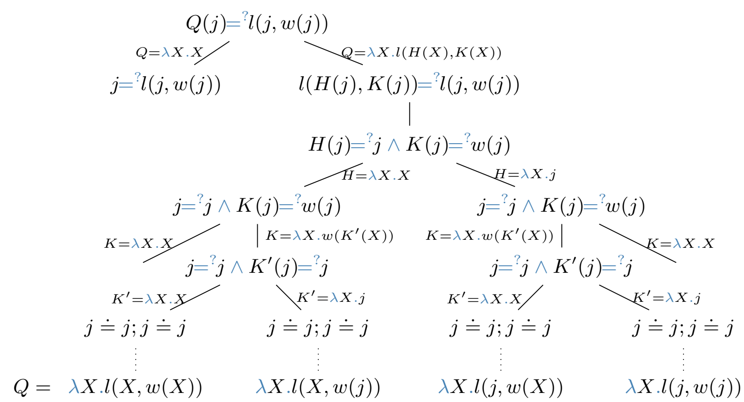
▷ Rules for flex/flex pairs:

$$\frac{(F_\alpha\,\overline{\mathbf{U}})=^?(H\,\overline{\mathbf{V}})\wedge\mathcal{E}}{F=^?\mathbf{G}\wedge(F\,\overline{\mathbf{U}})=^?(H\,\overline{\mathbf{V}})\wedge\mathcal{E}}\ \mathcal{HOU}{:}\mathrm{ff}$$

Let us now fortify our intuition with a simple example.

$\mathcal{HOU}$ Example

Example 13.2.18. Let $Q, w{:}\iota \to \iota$, $l{:}\iota \to \iota \to \iota$, and $j{:}\iota$, then we have the following derivation tree in $\mathcal{HOU}$.



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 arithmetic into $\Lambda^{\to}$ is useful for testing our unification algorithm.

A Test Generator for Higher-Order Unification

▷ **Definition 13.2.19 (Church Numerals).** We define closed λ-terms of type $\nu := (\alpha \to \alpha) \to \alpha \to \alpha$

- ▷ Numbers: Church numerals: (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 13.2.20.** *Subtraction and (integer) division on Church numberals can be automted via higher-order unification.*

▷ **Example 13.2.21.**

$5-2$ 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.

13.2.3 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 undecidability 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 13.2.22.** *Second-order unification is undecidable (Goldfarb '82 [Gol81])*

▷ *Proof sketch:* Reduction to Hilbert's tenth problem (solving Diophantine equations) (known to be undecidable)

▷ **Definition 13.2.23.**

We call an equation a Diophantine equation, if it is of the form

- ▷ $x_ix_j=x_k$
- ▷ $x_i+x_j=x_k$
- ▷ $x_i=c_j$ 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. **??**).

▷ **Theorem 13.2.24.** *The general solution for sets of Diophantine equations is undecidable. (Matijasevič 1970 [Mat70])*

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 13.2.25.** *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 13.2.26.** *$\mathcal{HOU}$ is correct: If $\mathcal{E}\vdash_{\mathcal{HOU}}\mathcal{E}'$, then $\mathbf{U}(\mathcal{E}')\subseteq\mathbf{U}(\mathcal{E})$.*

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 ??) 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 13.2.27 ($\mathcal{HOU}$ derives Complete Set of Unifiers).** *If $\theta\in\mathbf{U}(\mathcal{E})$, 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 $\mathcal{F}$.

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 13.2.28.** 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 $X\in\mathrm{supp}(\theta)$.
- ▷ $\mu_2(\mathcal{E})$ 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 13.2.29.** *$\prec$ is well-founded.* (by construction)

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 **??** and look at their consequences before we proove it.

Completeness of $\mathcal{HOU}$ (μ-Prescription)

▷ **Theorem 13.2.30.** *If $\mathcal{E}$ is unsolved and $\theta \in \mathbf{U}(\mathcal{E})$, then there is a unification problem $\mathcal{E}$ with $\mathcal{E}\vdash_{\mathcal{HOU}}\mathcal{E}'$ and a substitution $\theta' \in \mathbf{U}(\mathcal{E}')$, such that*

- ▹ $\theta =_{\beta\eta} \theta'[\mathcal{E}]$
- ▹ $\mu(\mathcal{E}, \theta' 0) \prec \mu(\mathcal{E}, \theta')$.

we call such a $\mathcal{HOU}$-step a μ-prescribed

▷ **Corollary 13.2.31.** *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.

We now come to the proof of **??**, which is a relatively simple consequence of **??**.

Proof of ??

▷ *Proof:*

1. Let $\mathbf{A} =^? \mathbf{B}$ be an unsolved pair of the form $(\mathbf{F}\,\overline{\mathbf{U}}) =^? (\mathbf{G}\,\overline{\mathbf{V}})$ in $\mathcal{F}$.
2. $\mathcal{E}$ is a $\mathcal{SIM}$ normal form, so $\mathbf{F}$ and $\mathbf{G}$ must be constants or variables,
3. but not the same constant, since otherwise $\mathcal{SIM}$:dec would be applicable.
4. By **??** 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 $\mathrm{head}(\mathbf{G}) \notin \mathrm{supp}(\theta)$, then $\mathcal{HOU}$:fr is applicable,
   - ▹ if $\mathrm{head}(\mathbf{G}) \in \mathrm{supp}(\theta)$, then $\mathcal{HOU}$:ff is applicable.
5. Choose $\theta' := \theta \cup \rho$. Then $\theta =_{\beta\eta} \theta'[\mathcal{E}]$ and $\theta' \in \mathbf{U}(\mathcal{E}')$ by correctness.
6. $\mathcal{HOU}$:ff and $\mathcal{HOU}$:fr solve $F \in \mathrm{supp}(\theta)$ and replace F by $\mathrm{supp}(\rho)$ in the set of unsolved variable of $\mathcal{E}$.
7. so $\mu_1(\mathcal{E}, \theta') \prec \mu_1(\mathcal{E}, \theta)'$ and thus $\mu(\mathcal{E}, \theta') \prec \mu(\mathcal{E}, \theta')$.

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 13.2.32.** *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 $\mathcal{D}\colon \mathcal{E}\vdash_{\mathcal{HOU}}\mathcal{F}$ a maximal μ-prescribed $\mathcal{HOU}$-derivation from $\mathcal{E}$.

1. This must be finite, since $\prec$ is well-founded (ind. over length n of $\mathcal{D}$)
2. If $n = 0$, then $\mathcal{E}$ is solved and $\sigma_{\mathcal{E}}$ most general unifier
3. thus $\sigma_{\mathcal{E}} \leq_{\beta\eta} \theta[\mathcal{E}]$

4. If $n > 0$, then there is a μ-prescribed step $\mathcal{E}\vdash_{\mathcal{HOU}}\mathcal{E}'$ and a substitution θ as in ??.
5. by IH there is a $\mathcal{HOU}$-derivation $\mathcal{E}'\vdash_{\mathcal{HOU}}\mathcal{F}$ with $\sigma_{\mathcal{F}}\leq_{\beta\eta}\theta'[\mathcal{E}']$.
6. by correctness $\sigma_{\mathcal{F}} \in \mathbf{U}(\mathcal{E}') \subseteq \mathbf{U}(\mathcal{E})$.
7. rules of $\mathcal{HOU}$ only expand free variables, so $\sigma_{\mathcal{F}}\leq_{\beta\eta}\theta'[\mathcal{E}']$.
8. Thus $\sigma_{\mathcal{F}}\leq_{\beta\eta}\theta'[\mathcal{E}]$,
9. This completes the proof, since $\theta'=_{\beta\eta}\theta[\mathcal{E}]$ by ??.

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.
- ▷ $\mathcal{HOU}{:}\mathrm{ff}$ 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), ...

13.2.4 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

- ▷ $\mathcal{HOU}{:}\mathrm{ff}$ 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 13.2.33.** *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 13.2.34 (Pre-Unification).** For given terms $\mathbf{A}, \mathbf{B} \in wff_\alpha(\Sigma_\mathcal{T}, \mathcal{V}_\mathcal{T})$ 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 13.2.35.** *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 13.2.36.** A unification problem is a pre solved form, iff all of its pairs are solved or flex/flex

▷ **Lemma 13.2.37.** *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!

▷ **Definition 13.2.38.** The higher-order pre-unification calculus $\mathcal{HOPU}$ only consists of $\mathcal{SIM}$ and $\mathcal{HOU}$:fr.

▷ **Theorem 13.2.39.** *$\mathcal{HOPU}$ is a correct and complete pre-unification algorithm*

▷ *Proof sketch:* with exactly the same methods as higher-order unification

▷ **Theorem 13.2.40.** *Higher-order pre-unification is infinitary, i.e. a unification problem can have infinitely many unifiers.* (Huet 76' [Hue76])

▷ **Example 13.2.41.** $Y\ (\lambda X_\iota.X)\ a =^? a$, 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.

13.2.5 Applications of Higher-Order Unification

Application of HOL in NL Semantics: Ellipsis

▷ **Example 13.2.42.** *John loves his wife. George does too*

- ▷ $\mathrm{loves}(\mathrm{john}, \mathrm{wifeof}(\mathrm{john})) \wedge Q(\mathrm{george})$
- ▷ "*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:** $Q(\mathrm{john}) =_{\alpha\beta\eta} \mathrm{loves}(\mathrm{john}, \mathrm{wifeof}(\mathrm{john}))$

▷ **Solutions (computed by HOU):**

- ▷ $Q = \lambda z.\mathrm{loves}(z, \mathrm{wifeof}(z))$ and $Q = \lambda z.\mathrm{loves}(z, \mathrm{wifeof}(\mathrm{john}))$
- * $Q = \lambda z.\mathrm{loves}(\mathrm{john}, \mathrm{wifeof}(z))$ and $Q = \lambda z.\mathrm{loves}(\mathrm{john}, \mathrm{wifeof}(\mathrm{john}))$

▷ **Readings:** *George loves his own wife.* and *George loves John's wife.*

▷ **Erraneous HOU Predictions:** * *John loves George's wife*. and * *John loves John's wife*.

13.3 Linguistic Applications of Higher-Order Unification

George does too (HOU)

$Q(j) =^? l(j, w(j))$

$\xrightarrow{Q=\lambda X.X}$ $j =^? l(j, w(j))$

$\xrightarrow{Q=\lambda X.l(H(X),K(X))}$ $l(H(j), K(j)) =^? l(j, w(j))$

$H(j) =^? j \wedge K(j) =^? w(j)$

$\xrightarrow{H=\lambda X.X}$ $j =^? j \wedge K(j) =^? w(j)$

- $\xrightarrow{K=\lambda X.X}$ … $\xrightarrow{K'=\lambda X.X}$ $j \doteq j; j \doteq j$ … $Q = \lambda X.l(X, w(X))$
- $\xrightarrow{K=\lambda X.w(K'(X))}$ $j =^? j \wedge K'(j) =^? j$
  - $\xrightarrow{K'=\lambda X.X}$ $j \doteq j; j \doteq j$ … $\lambda X.l(X, w(X))$
  - $\xrightarrow{K'=\lambda X.j}$ $j \doteq j; j \doteq j$ … $\lambda X.l(X, w(j))$

$\xrightarrow{H=\lambda X.j}$ $j =^? j \wedge K(j) =^? w(j)$

- $\xrightarrow{K=\lambda X.w(K'(X))}$ $j =^? j \wedge K'(j) =^? j$
  - $\xrightarrow{K'=\lambda X.X}$ $j \doteq j; j \doteq j$ … $\lambda X.l(j, w(X))$
  - $\xrightarrow{K'=\lambda X.j}$ $j \doteq j; j \doteq j$ … $\lambda X.l(j, w(j))$
- $\xrightarrow{K=\lambda X.X}$ …

$Q = \quad \lambda X.l(X, w(X)) \quad \lambda X.l(X, w(j)) \quad \lambda X.l(j, w(X)) \quad \lambda X.l(j, w(j))$

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 13.3.1.**

- ▹ $\mathrm{loves}(\underline{\mathrm{john}}, \mathrm{wifeof}(\mathrm{john})) = Q(\mathrm{george})$
- ▹ $Q = \lambda x.\mathrm{loves}(x, \mathrm{wifeof}(\mathrm{john}))$
- ▹ $Q = \lambda x.\mathrm{loves}(\underline{\mathrm{john}}, \mathrm{wifeof}(\mathrm{john}))$

▷ Use the colored λ-calculus for general theory

Colored λ-calculus [HK00]

- Developed for inductive theorem proving (Rippling with Metavariable)
- **Definition 13.3.2.** Symbol occurrences can be annotated with colors (variables α, β, γ, ... and constants a, b,...)
- Bound variables are uncolored ($\beta\eta$ conversion just as usual)
- **Definition 13.3.3.** 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)

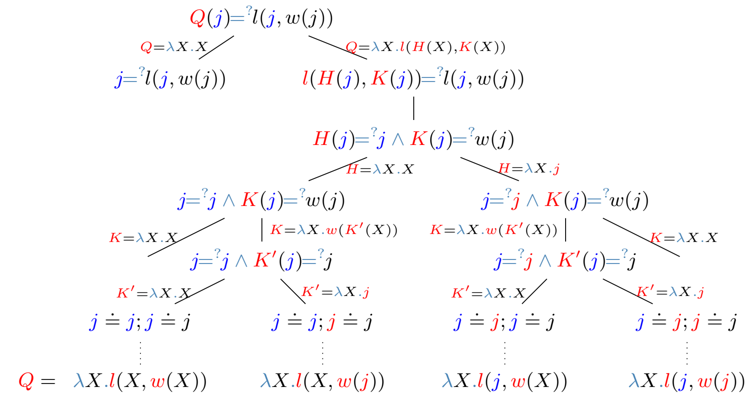
Colored HO-Unification

- HOCU has only two differences wrt. general HOU

$$\frac{f_f(t^1,\ldots,t^n) =^? f_f(s^1,\ldots,s^n)}{\mathsf{a} =^? \mathsf{b} \wedge t^1 =^? s^1 \wedge t^n =^? s^n} \qquad \frac{X_X =^? \mathbf{A} \wedge \mathcal{E}}{X =^? \mathbf{A} \wedge [\mathbf{A}/X](\mathcal{E})}$$

- Decomposition must consider colors
- Elimination ensures Monochromicity and Consistency
  - $X =^? \mathbf{A} := X_X =^? \mathbf{A}_\mathbf{A} \wedge X_X =^? \mathbf{A}_\mathbf{A}$
  - $[\mathbf{A}/X] := [\mathbf{A}_\mathbf{A}/X_X],\ldots,[\mathbf{A}_\mathbf{A}/X_X]$ propagates color information

George does too (HOCU)



The Original Motivation: First-Order Rippling

Example 13.3.4. Proving: $\forall x, y : list.\mathrm{rev}(\mathrm{app}(\mathrm{rev}(x), y)) = \mathrm{app}(\mathrm{rev}(y), x)$

$$\mathrm{rev}(\mathrm{app}(\mathrm{rev}(\mathrm{cons}(h, x)), y)) = \mathrm{app}(\mathrm{rev}(y), \mathrm{cons}(h, x))$$
$$\downarrow$$
$$\mathrm{rev}(\mathrm{app}(\mathrm{app}(\mathrm{rev}(x), \mathrm{cons}(h, \mathrm{nil})), y)) = \mathrm{app}(\mathrm{rev}(y), \mathrm{cons}(h, x))$$
$$\mathrm{app}_\alpha(X_X, \mathrm{cons}(Y, Z_Z)) = \mathrm{app}_\alpha(F_1(X_X, Y, Z), Z_Z) \downarrow$$
$$\mathrm{rev}(\mathrm{app}(\mathrm{app}(\mathrm{rev}(x), \mathrm{cons}(h, \mathrm{nil})), y)) = \mathrm{app}(F_1(\mathrm{rev}(y), h, x), x)$$
$$\mathrm{app}(\mathrm{rev}_\alpha(Y_Y), \mathrm{cons}(X, \mathrm{nil})) = \mathrm{rev}_\alpha(\mathrm{cons}(X, Y_Y)) \downarrow$$
$$\mathrm{rev}(\mathrm{app}(\mathrm{app}(\mathrm{rev}(x), \mathrm{cons}(h, \mathrm{nil})), y)) = \mathrm{app}(\mathrm{rev}(\mathrm{cons}(h, y)), x)$$
$$\downarrow$$
$$\mathrm{rev}(\mathrm{app}(\mathrm{rev}(x), \mathrm{cons}(h, y))) = \mathrm{app}(\mathrm{rev}(\mathrm{cons}(h, y)), x)$$

The Higher-Order Case: Schematic Rippling

Example 13.3.5 (Synthesizing Induction Orderings). $\forall x.\exists y.f(g(y)) \leq x$
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 13.3.6 (A Unification Problem).**

$$
\begin{array}{c}
F(\mathrm{rev}(y),h,x) =^? rev_\alpha(Y_\beta)\ \mathrm{cons}(X,\mathrm{nil}) \\
\downarrow\ [\lambda UVW.\mathrm{app}(H(U,V,W),K(U,V,W))/F] \\
H(\mathrm{rev}(u),h,v) =^? \mathrm{rev}_\alpha(Y_Y) \wedge K(\mathrm{rev}(u),h,v) =^? \mathrm{cons}(X,\mathrm{nil}) \\
\downarrow\ [\lambda UVW.\mathrm{cons}(M(U,V,W),N(U,V,W))/K],\ [\lambda UVW.U/H] \\
\mathrm{rev}(u) =^? \mathrm{rev}_\alpha(Y_Y) \wedge \mathrm{cons}(M(\mathrm{rev}(u),h,v),N(\mathrm{rev}(u),h,v)) =^? \mathrm{cons}(X,\mathrm{nil}) \\
\downarrow \\
\alpha =^? \blacksquare \wedge u =^? Y_Y \wedge X =^? M(\mathrm{rev}(u),h,v) \wedge N(\mathrm{rev}(u),h,v) =^? nil \\
\downarrow \\
h =^? h \wedge \mathrm{nil} =^? \mathrm{nil}
\end{array}
$$

Result: $[\lambda UVW.\mathrm{app}(U,\mathrm{cons}(V,\mathrm{nil}))/F], [u/Y_Y], [h/X], [\blacksquare/\alpha]$

Linguistic Application: Focus/Ground Structures

▷ **Example 13.3.7.** *John only likes MARY.*

▷ **Analysis:** $\mathrm{likes}(\mathrm{john},\mathrm{mary}) \wedge (\forall x.G(x)) \Rightarrow x = \mathrm{mary}$.

▷ **Equation:** $\mathrm{likes}(\mathrm{john},\mathrm{mary}) =_{\alpha\beta\eta} G(\mathrm{mary})$.

- ▹ Variable G for (back)ground (Focus is prosodically marked)

▷ **Solution:** $G = \lambda z.\mathrm{likes}(\mathrm{john},z)$

▷ **Semantics:** $\mathrm{likes}(\mathrm{john},\mathrm{mary}) \wedge (\forall x.\mathrm{likes}(\mathrm{john},x) \Rightarrow x = \mathrm{mary})$.

▷ **Linguistic Coverage:** Prosodically unmarked focus, sentences with multiple focus operators
[Gardent & Kohlhase'96]

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

Interaction of Constraints via Extended Colors

- **Example 13.3.8.** *John likes MARY and Peter does too*
  - Ellipsis: $l(j_j, s_s) = R_R(j_j)$
  - Focus: $R_R(p) = G_G(F_F)$
  - ¬pe forbids only pe ¬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:** $G_G = \lambda x.l(p_p, x)$ $F_F = m_m$

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 John* (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.

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)

$\{R_R \leftarrow \lambda z.l(z, j)\}$	Strict Reading
$\{R_R \leftarrow \lambda z.l(z, p_p)\}$	Sloppy Reading

▷ 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

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)

13.4 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

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

$$\mathbf{G}^{+}_{\mathbb{E}}() = \left\{\begin{array}{l}+Z_{\mathbb{E}}W_{\mathbb{E}},\\ +Z_{\mathbb{O}}W_{\mathbb{O}},\\ +Z_{\mathbb{N}}Z_{\mathbb{N}}\end{array}\right\}$$

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

$$\begin{array}{l}[+{:}(\mathbb{R}\to\mathbb{R}\to\mathbb{R})], [*{:}(\mathbb{R}\to\mathbb{R}\to\mathbb{R})], [(\lambda X.*XX){:}(\mathbb{R}\to\mathbb{R}^+)],\\ [\mathbb{R}^+\sqsubset\mathbb{R}], [\mathbb{M}\sqsubset\mathbb{P}], [\mathbb{P}\sqsubset\mathbb{M}], [\mathbb{M}\sqsubset\mathbb{P}]\\ [(\lambda X.X){:}\mathbb{M}], [(\lambda XY.Y){:}(\mathbb{R}\to\mathbb{M})],\\ [(\lambda FGX.*(FX)(GX)){:}(\mathbb{M}\to\mathbb{M}\to\mathbb{M})],\\ [(\lambda FGX.+(FX)(GX)){:}(\mathbb{M}\to\mathbb{M}\to\mathbb{P})],\\ [\partial{:}(\mathbb{M}\to\mathbb{P})], [\partial{:}(\mathbb{P}\to\mathbb{P})], [\partial{:}(\mathbb{M}\to\mathbb{M})].\end{array}$$

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)

Abductive Reconstruction of Parallelism (ARP)

- Mix Parallelism with HOCU
- Example (Gapping): *John likes Golf and Mary too.*
- Representation $\mathrm{loves}(\mathrm{john}, \mathrm{golf}) \wedge R(\mathrm{mary})$
- Equation $\mathrm{loves}(\mathrm{john}_{\mathrm{john}}, \mathrm{golf}_{\mathrm{golf}})=^s R^{\neg\mathsf{pe}}_{(\mathbb{W}\mathtt{oman}\to o)}(\mathrm{mary}_{\mathrm{mary}})$
  - 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)

- Initial Equation: $\mathrm{loves}(\mathrm{john}_{\mathrm{john}}, \mathrm{golf}_{\mathrm{golf}})=^? R^{\neg\mathsf{pe}}_{(\mathbb{W}\mathtt{oman}\to o)}(\mathrm{mary}_{\mathrm{mary}})$
  - imitate $R^{\neg\mathsf{pe}}_{(\mathbb{W}\mathtt{oman}\to o)}$ with $\lambda Z.\mathrm{loves}(H_H Z, K_K Z)$
  - H, K new variables of sort $\mathbb{W}\mathtt{oman}\to\mathbb{H}\mathtt{uman}$

- $\mathrm{loves}(\mathrm{john}_{\mathrm{john}}, \mathrm{golf}_{\mathrm{golf}}) =^{?} \mathrm{loves}(H_H(\mathrm{mary}_{\mathrm{mary}}), K_K \mathrm{mary}_{\mathrm{mary}})$
- $H_H \mathrm{mary}_{\mathrm{mary}} =^{?} \mathrm{john}_{\mathrm{john}} \wedge K_K \mathrm{mary}_{\mathrm{mary}} =^{?} \mathrm{golf}_{\mathrm{golf}}$
- Two possible continuations:

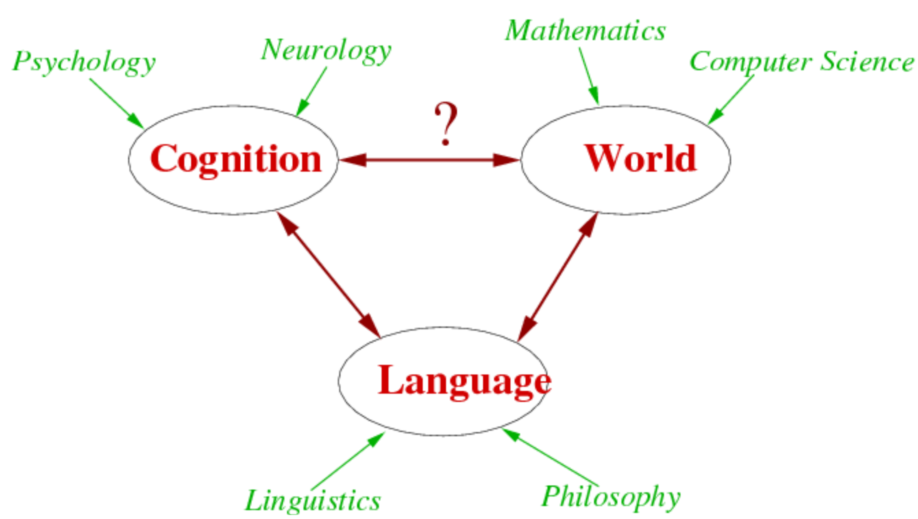
▷ project $H = \lambda Z.Z$ (so $A =^{?}$ pe)	▷ project $K = \lambda Z.Z$ (so $\neg A =^{?}$ pe)
▷ imitate $K = \lambda Z.\mathrm{golf}_{\mathrm{golf}}$	▷ imitate $H = \lambda Z.\mathrm{john}_{\mathrm{john}}$
▷ then $\boxed{\begin{array}{l}\mathrm{mary}_{\mathrm{mary}} =^{?} \mathrm{john}_{\mathrm{john}} \\ \mathrm{golf}_{\mathrm{golf}} =^{?} \mathrm{golf}_{\mathrm{golf}}\end{array}}$	▷ then $\boxed{\begin{array}{l}\mathrm{john}_{\mathrm{john}} =^{?} \mathrm{john}_{\mathrm{john}} \\ \mathrm{mary}_{\mathrm{mary}} =^{?} \mathrm{golf}_{\mathrm{golf}}\end{array}}$
▷ *Mary likes Golf* (*preferred*)	▷ John *likes Mary*

Chapter 14

Conclusion

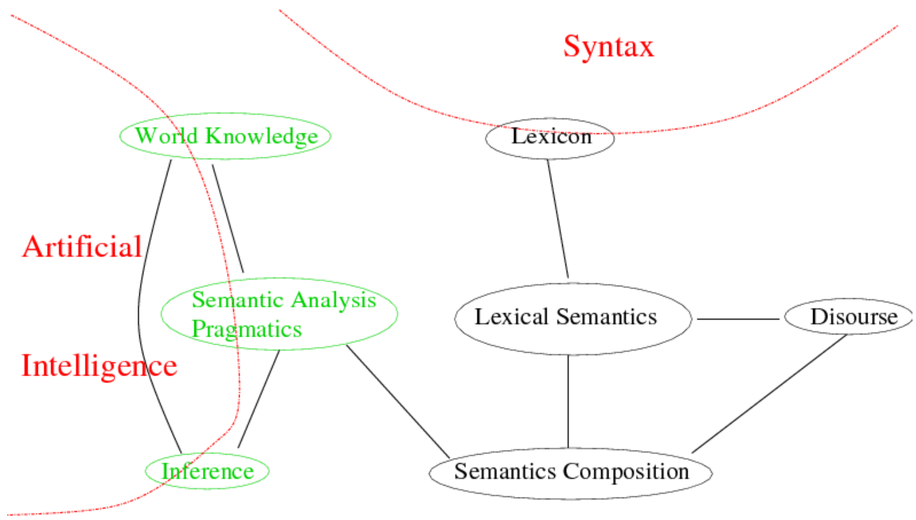
14.1 A Recap in Diagrams

NL Semantics as an Intersective Discipline



Michael Kohlhase: LBS 291 2025-02-06

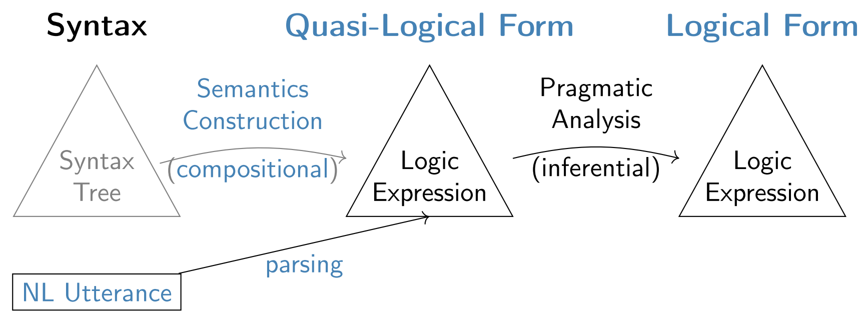
A landscape of formal semantics



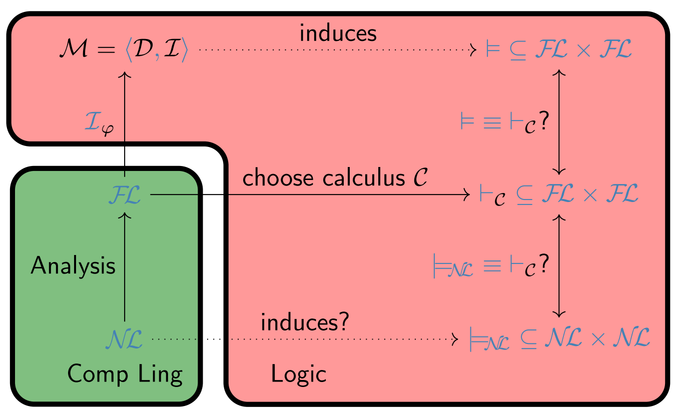
Modeling Natural Language Semantics

- **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.* $((\exists X.\mathrm{man}(X) \wedge \mathrm{sleeps}(X))) \wedge \mathrm{snores}(X)$
  - Various dynamic approaches (e.g. DRT, DPL)
    * *Most men wear black*
  - Higher-order Logic, e.g. generalized quantifiers
  - ...

A Semantic Processing Pipeline based on LF



Natural Language Semantics?



14.2 Where to From Here

Where to from here?

- ▷ We can continue the exploration of semantics in two different ways:
  - ▷ Look around for additional logical/formal 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...

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)

Reciprocals

▷ What's required to make these true?

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.

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.

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?)

Bibliography

[And72] Peter B. Andrews. “General Models and Extensionality”. In: *Journal of Symbolic Logic* 37.2 (1972), pp. 395–397.

[Ari10] Mira Ariel. *Defining Pragmatics*. Research Surveys in Linguistics. Cambridge University Press, 2010.

[BB05] Patrick Blackburn and Johan Bos. *Representation and Inference for Natural Language. A First Course in Computational Semantics*. CSLI, 2005.

[Ben91] Johan van Benthem. *Language in Action, Categories, Lambdas and Dynamic Logic*. Vol. 130. Studies in Logic and Foundation of Mathematics. North Holland, 1991.

[Bir13] Betty J. Birner. *Introduction to Pragmatics*. Wiley-Blackwell, 2013.

[Bla+01] Patrick Blackburn et al. “Inference and Computational Semantics”. In: *Computing Meaning (Volume 2)*. Ed. by Harry Bunt et al. Kluwer Academic Publishers, 2001, pp. 11–28.

[BRV01] Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal Logic*. New York, NY, USA: Cambridge University Press, 2001. ISBN: 0-521-80200-8.

[Cho65] Noam Chomsky. *Aspects of the Theory of Syntax*. MIT Press, 1965.

[Chu40] Alonzo Church. “A Formulation of the Simple Theory of Types”. In: *Journal of Symbolic Logic* 5 (1940), pp. 56–68.

[CKG09] Ronnie Cann, Ruth Kempson, and Eleni Gregoromichelaki. *Semantics – An Introduction to Meaning in Language*. Cambridge University Press, 2009. ISBN: 0521819628.

[Cre82] M. J. Cresswell. “The Autonomy of Semantics”. In: *Processes, Beliefs, and Questions: Essays on Formal Semantics of Natural Language and Natural Language Processing*. Ed. by Stanley Peters and Esa Saarinen. Springer, 1982, pp. 69–86. DOI: 10.1007/978-94-015-7668-0_2.

[Cru11] Alan Cruse. *Meaning in Language: An Introduction to Semantics and Pragmatics*. Oxford Textbooks in Linguistics. 2011.

[Dav67a] Donald Davidson. “The logical form of action sentences”. In: *The logic of decision and action*. Ed. by N. Rescher. Pittsburgh: Pittsburgh University Press, 1967, pp. 81–95.

[Dav67b] Donald Davidson. “Truth and Meaning”. In: *Synthese* 17 (1967).

[de 95] Manuel de Vega. “Backward updating of mental models during continuous reading of narratives”. In: *Journal of Experimental Psychology: Learning, Memory, and Cognition* 21 (1995), pp. 373–385.

[DSP91] Mary Dalrymple, Stuart Shieber, and Fernando Pereira. “Ellipsis and Higher-Order Unification”. In: *Linguistics & Philosophy* 14 (1991), pp. 399–452.

[Eij97] Jan van Eijck. “Type Logic with States”. In: *Logic Journal of the IGPL* 5.5 (Sept. 1997).

[EU10] Jan van Eijck and Christina Unger. *Computational Semantics with Functional Programming*. Cambridge University Press, 2010.

[Fre92] Gottlob Frege. “Über Sinn und Bedeutung”. In: *Zeitschrift für Philosophie und philosophische Kritik* 100 (1892), pp. 25–50.

[GK96] Claire Gardent and Michael Kohlhase. “Focus and Higher–Order Unification”. In: *Proceedings of the 16th International Conference on Computational Linguistics*. Copenhagen, 1996, pp. 268–279. URL: https://kwarc.info/kohlhase/papers/coling96.pdf.

[GKL96] Claire Gardent, Michael Kohlhase, and Noor van Leusen. “Corrections and Higher-Order Unification”. In: *Proceedings of KONVENS’96*. Bielefeld, Germany: De Gruyter, 1996, pp. 268–279. URL: https://kwarc.info/kohlhase/papers/konvens96.pdf.

[GML87] A. M. Glenberg, M. Meyer, and K. Lindem. “Mental models contribute to foregrounding during text comprehension”. In: *Journal of Memory and Language* 26 (1987), pp. 69–83.

[Göd32] Kurt Gödel. “Zum Intuitionistischen Aussagenkalkül”. In: *Anzeiger der Akademie der Wissenschaften in Wien* 69 (1932), pp. 65–66.

[Gol81] Warren D. Goldfarb. “The Undecidability of the Second-Order Unification Problem”. In: *Theoretical Computer Science* 13 (1981), pp. 225–230.

[GS90] Jeroen Groenendijk and Martin Stokhof. “Dynamic Montague Grammar”. In: *Papers from the Second Symposium on Logic and Language*. Ed. by L. Kálmán and L. Pólos. Akadémiai Kiadó, Budapest, 1990, pp. 3–48.

[GS91] Jeroen Groenendijk and Martin Stokhof. “Dynamic Predicate Logic”. In: *Linguistics & Philosophy* 14 (1991), pp. 39–100.

[Har84] D. Harel. “Dynamic Logic”. In: *Handbook of Philosophical Logic*. Ed. by D. Gabbay and F. Günthner. Vol. 2. Reidel, Dordrecht, 1984, pp. 497–604.

[HC84] G. E. Hughes and M. M. Cresswell. *A companion to Modal Logic*. University Paperbacks. Methuen, 1984.

[Hei82] Irene Heim. “The Semantics of Definite and Indefinite Noun Phrases”. PhD thesis. University of Massachusetts, 1982.

[HHS07] James R. Hurford, Brendan Heasley, and Michael B. Smith. *Semantics: A coursebook*. 2nd. Cambridge University Press, 2007.

[HK00] Dieter Hutter and Michael Kohlhase. “Managing Structural Information by Higher-Order Colored Unification”. In: *Journal of Automated Reasoning* 25.2 (2000), pp. 123–164. URL: https://kwarc.info/kohlhase/papers/jar00.pdf.

[HM95] Furio Honsell and Marino Miculan. “A natural deduction approach to dynamic logic”. In: *Types for Proofs and Programs TYPES ’95*. Ed. by Stefano Berardi and Mario Coppo. 1995, pp. 165–182. ISBN: 978-3-540-61780-8. DOI: 10.1007/3-540-61780-9_69.

[Hue76] Gérard P. Huet. “Résolution d’Équations dans des Langages d’ordre 1,2,...,w.” Thèse d‘État. Unif-bib: Université de Paris VII, 1976.

[Hue80] Gérard Huet. “Confluent Reductions: Abstract Properties and Applications to Term Rewriting Systems”. In: *Journal of the ACM (JACM)* 27.4 (1980), pp. 797–821.

[Isr93] David J. Israel. “The Very Idea of Dynamic Semantics”. In: *Proceedings of the Ninth Amsterdam Colloquium*. 1993. URL: https://arxiv.org/pdf/cmp-lg/9406026.pdf.

[Jac83] Ray Jackendoff. *Semantics and Cognition*. MIT Press, 1983.

[JL83] P. N. Johnson-Laird. *Mental Models*. Cambridge University Press, 1983.

[JLB91] P. N. Johnson-Laird and Ruth M. J. Byrne. *Deduction*. Lawrence Erlbaum Associates Publishers, 1991.

[Kam81] Hans Kamp. "A Theory of Truth and Semantic Representation". In: *Formal Methods in the Study of Language*. Ed. by J. Groenendijk, Th. Janssen, and M. Stokhof. Amsterdam, Netherlands: Mathematisch Centrum Tracts, 1981, pp. 277–322.

[Kea11] Kate Kearns. *Semantics*. 2nd. Palgrave Macmillan, 2011.

[KKP96] Michael Kohlhase, Susanna Kuschert, and Manfred Pinkal. "A type-theoretic semantics for λ-DRT". In: *Proceedings of the 10^{th} Amsterdam Colloquium*. Ed. by P. Dekker and M. Stokhof. ILLC. Amsterdam, 1996, pp. 479–498. URL: https://kwarc.info/kohlhase/papers/amscoll95.pdf.

[Koh08] Michael Kohlhase. "Using LaTeX as a Semantic Markup Format". In: *Mathematics in Computer Science* 2.2 (2008), pp. 279–304. URL: https://kwarc.info/kohlhase/papers/mcs08-stex.pdf.

[Kon04] Karsten Konrad. *Model Generation for Natural Language Interpretation and Analysis*. Vol. 2953. LNCS. Springer, 2004. ISBN: 3-540-21069-5. DOI: 10.1007/b95744.

[KR93] Hans Kamp and Uwe Reyle. *From Discourse to Logic: Introduction to Model-Theoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory*. Dordrecht: Kluwer, 1993.

[Kra12] Angelika Kratzer. *Modals and Conditionals. New and Revised Perspectives*. Oxford Studies in Theoretical Linguistics. Oxford University Press, 2012.

[Kri63] Saul Kripke. "Semantical Considerations on Modal Logic". In: *Acta Philosophica Fennica* (1963), pp. 83–94.

[Lew18] Clarence Irving Lewis. *A Survey of Symbolic Logic*. University of California Press, 1918. URL: http://hdl.handle.net/2027/hvd.32044014355028.

[Lew73] David K. Lewis. *Counterfactuals*. Blackwell Publishers, 1973.

[Mat70] Ju. V. Matijasevič. "Enumerable sets are diophantine". In: *Soviet Math. Doklady* 11 (1970), pp. 354–358.

[MBV95] Reinhard Muskens, Johan van Benthem, and Albert Visser. "Dynamics". In: ed. by Johan van Benthem and Ter Meulen. Elsevier Science B.V., 1995.

[Mon70] R. Montague. "English as a Formal Language". In: Reprinted in [Tho74], 188–221. Edizioni di Communita, Milan, 1970, pp. 189–224.

[Mon74] Richard Montague. "The Proper Treatment of Quantification in Ordinary English". In: *Formal Philosophy. Selected Papers*. Ed. by R. Thomason. New Haven: Yale University Press, 1974.

[MR98] C. Monz and M. de Rijke. "A Resolution Calculus for Dynamic Semantics". In: *Logics in Artificial Intelligence. European Workshop JELIA '98*. LNAI 1489. Springer Verlag, 1998.

[Mus96] Reinhard Muskens. "Combining Montague Semantics and Discourse Representation". In: *Linguistics & Philosophy* 14 (1996), pp. 143 –186.

[Nor+18a] Emily Nordmann et al. *Lecture capture: Practical recommendations for students and lecturers*. 2018. URL: https://osf.io/huydx/download.

[Nor+18b] Emily Nordmann et al. *Vorlesungsaufzeichnungen nutzen: Eine Anleitung für Studierende*. 2018. URL: https://osf.io/e6r7a/download.

[Ohl88] Hans Jürgen Ohlbach. "A Resolution Calculus for Modal Logics". PhD thesis. Universität Kaiserslautern, 1988.

[Par90] Terence Parsons. *Events in the Semantics of English: A Study in Subatomic Semantics*. Vol. 19. Current Studies in Linguistics. MIT Press, 1990.

[Pin96] Manfred Pinkal. "Radical underspecification". In: *Proceedings of the 10^{th} Amsterdam Colloquium*. Ed. by P. Dekker and M. Stokhof. ILLC. Amsterdam, 1996, pp. 587–606.

[Pop34] Karl Popper. *Logik der Forschung*. Springer Verlag, 1934.

[Pop59] Karl Popper. *Logic of Scientific Discovery*. Basic Books, 1959.

[Por04] Paul Portner. *What is Meaning? Fundamentals of Formal Semantics*. Blackwell, 2004.

[Pra76] V. Pratt. "Semantical considerations of Floyd-Hoare logic". In: *Proceedings of the 17th Symposium on Foundations of Computer Science*. 1976, pp. 109–121.

[Pul94] Stephen G. Pulman. *Higher Order Unification and the Interpretation of Focus*. Tech. rep. CRC-049. SRI Cambridge, UK, 1994.

[Ran17] Aarne Ranta. *Automatic Translation for Consumers and Producers*. Presentation given at the Chalmers Initiative Seminar. 2017. URL: https://www.grammaticalframework.org/~aarne/mt-digitalization-2017.pdf.

[RG94] Uwe Reyle and Dov M. Gabbay. "Direct Deductive computation on Discourse Representation Structures". In: *Linguistics & Philosophy* 17 (1994), pp. 343–390.

[Rie10] Nick Riemer. *Introducing Semantics*. Cambridge Introductions to Language and Linguistics. Cambridge University Press, 2010.

[Rus03] Bertrand Russell. *The Principles of Mathematics*. Cambridge University Press, 1903.

[Rus91] Stuart J. Russell. "An Architecture for Bounded Rationality". In: *SIGART Bulletin* 2.4 (1991), pp. 146–150.

[Sae03] John I. Saeed. *Semantics*. 2nd. Blackwell, 2003.

[Sau93] Werner Saurer. "A Natural Deduction System for Discourse Representation Theory". In: *Journal of Philosophical Logic* 22 (1993).

[Sch20] Jan Frederik Schaefer. "Prototyping NLU Pipelines – A Type-Theoretical Framework". Master's Thesis. Informatik, FAU Erlangen-Nürnberg, 2020. URL: https://gl.kwarc.info/supervision/MSc-archive/blob/master/2020/Schaefer_Jan_Frederik.pdf.

[Sin94] M. Singer. "Discourse Inference Processes". In: *Handbook of Psycholinguistics*. Ed. by M. A. Gernsbacher. Academic Press, 1994, pp. 479–515.

[Spe17] Jeff Speaks. "Theories of Meaning". In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Fall 2017. Metaphysics Research Lab, Stanford University, 2017. URL: https://plato.stanford.edu/archives/fall2017/entries/meaning/.

[Sta14] Robert Stalnaker. *Context*. Oxford University Press, 2014.

[Sta68] Robert C. Stalnaker. "A Theory of Conditionals". In: *Studies in Logical Theory, American Philosophical Quarterly*. Blackwell Publishers, 1968, pp. 98–112.

[Sta85] Rick Statman. "Logical relations and the typed lambda calculus". In: *Information and Computation* 65 (1985).

[sTeX] *sTeX: A semantic Extension of TeX/LaTeX*. URL: https://github.com/sLaTeX/sTeX (visited on 05/11/2020).

[Tho74] R. Thomason, ed. *Formal Philosophy: selected Papers of Richard Montague*. Yale University Press, New Haven, CT, 1974.

[Ven57] Zeno Vendler. "Verbs and times". In: *Philosophical Review* 56 (1957), pp. 143–160.

[Zee89] Henk Zeevat. "A Compositional Approach to DRT". In: *Linguistics & Philosophy* 12 (1989), pp. 95–131.

[ZR98] R. A. Zwaan and G. A. Radvansky. "Situation models in language comprehension and memory". In: *Psychological Bulletin* 123 (1998), pp. 162–185.

[ZS13] Thomas Ede Zimmermann and Wolfgang Sternefeld. *Introduction to Semantics*. de Gruyter Mouton, 2013.

Index

Part III

Excursions

As this course is predominantly about modeling natural language and not about the theoretical aspects of the logics themselves, we give the discussion about these as a “suggested readings” section part here.

Appendix A

AI-Supported Learning

In this chapter we introduce the ALeA (Adaptive Learning Assistant) system, a learning support system we will use to support students in LBS.

ALeA: Adaptive Learning Assistant

- ▷ **Idea:** Use AI methods to help teach/learn AI (AI4AI)
- ▷ **Concretely:** Provide HTML versions of the LBS slides/lecture notes and embed learning support services into them. (for pre/postparation of lectures)
- ▷ **Definition A.0.1.** Call a document active, iff it is interactive and adapts to specific information needs of the readers. (lecture notes on steroids)
- ▷ **Intuition:** ALeA serves active course materials. (PDF mostly inactive)
- ▷ **Goal:** Make ALeA more like a instructor + study group than like a book!
- ▷ **Example A.0.2 (Course Notes).** $\widehat{=}$ Slides + Comments

$\rightsquigarrow$ yellow parts in table of contents (left) already covered in lectures.

Michael Kohlhase: LBS 304 2025-02-06

The central idea in the AI4AI approach – using AI to support learning AI – and thus the ALeA system is that we want to make course materials – i.e. what we give to students for preparing and postparing lectures – more like teachers and study groups (only available 24/7) than like static books.

VoLL-KI Portal at `https://courses.voll-ki.fau.de`

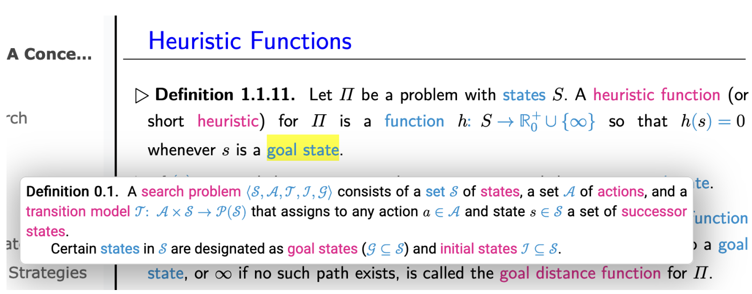
- **Portal for ALeA Courses:** `https://courses.voll-ki.fau.de`

- **LBS in ALeA:** `https://courses.voll-ki.fau.de/course-home/lbs`
  - All details for the course.
  - recorded syllabus (keep track of material covered in course)
  - syllabus of the last semesters (for over/preview)
- **ALeA Status:** The ALeA system is deployed at FAU for over 1000 students taking eight courses
  - (some) students use the system actively (our logs tell us)
  - reviews are mostly positive/enthusiastic (error reports pour in)

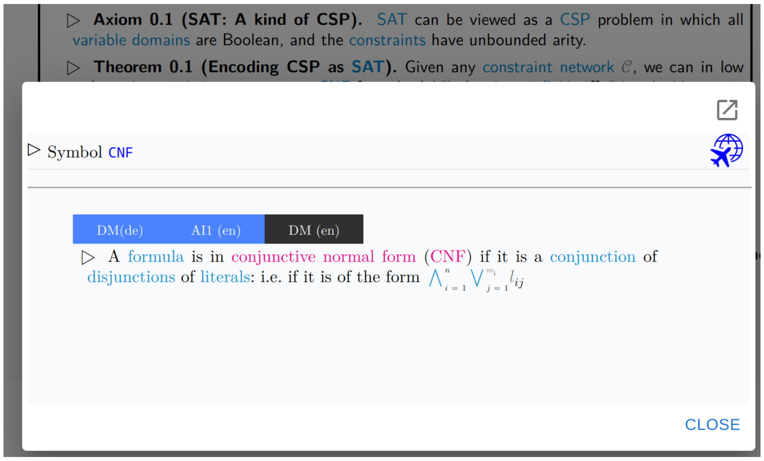
The ALeA LBS page is the central entry point for working with the ALeA system. You can get to all the components of the system, including two presentations of the course contents (notes- and slides-centric ones), the flashcards, the localized forum, and the quiz dashboard.
We now come to the heart of the ALeA system: its learning support services, which we will now briefly introduce. Note that this presentation is not really sufficient to understand what you may be getting out of them, you will have to try them, and interact with them sufficiently that the learner model can get a good estimate of your competencies to adapt the results to you.

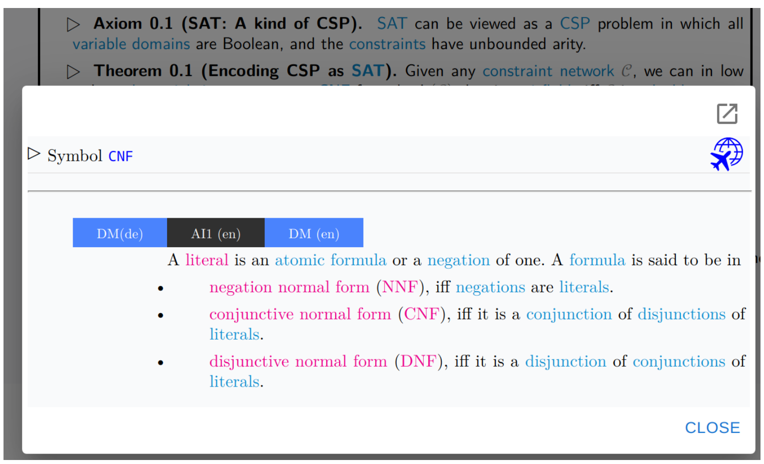
Learning Support Services in ALeA

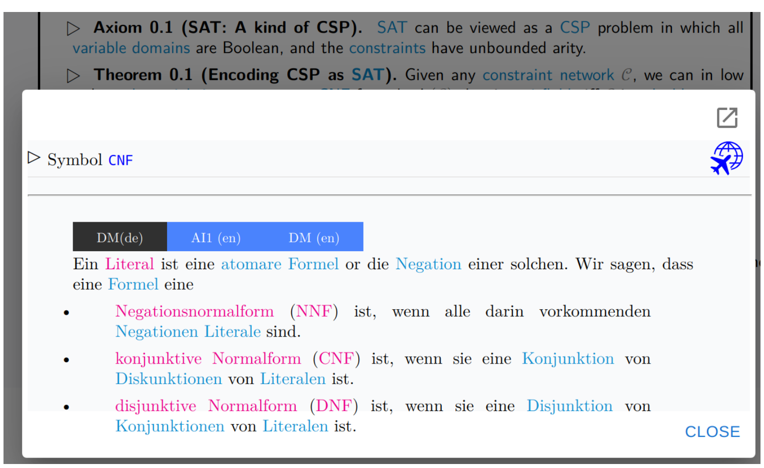
- **Idea:** Embed learning support services into active course materials.
- **Example A.0.3 (Definition on Hover).** Hovering on a (cyan) term reference reminds us of its definition. (even works recursively)



▷ **Example A.0.4 (More Definitions on Click).** Clicking on a (cyan) term reference shows us more definitions from other contexts.







▷ **Example A.0.5 (Guided Tour).** A guided tour for a concept c assembles definitions/etc. into a self-cont

c = countable $\leadsto$

▷ . . . your idea here . . . (the sky is the limit)

Note that this is only an initial collection of learning support services, we are constantly working on additional ones. Look out for feature notifications () on the upper right hand of the ALeA screen.

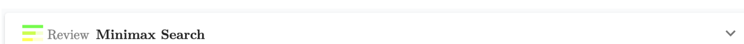
(Practice/Remedial) Problems Everywhere

▷ **Problem:** Learning requires a mix of understanding and test-driven practice.

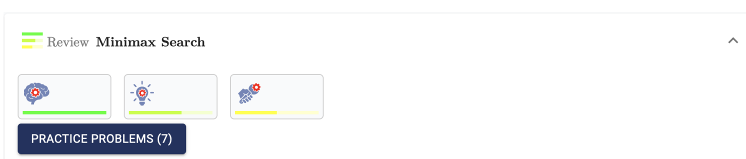
▷ **Idea:** ALeA supplies targeted practice problems everywhere.

▷ **Concretely:** Revision markers at the end of sections.

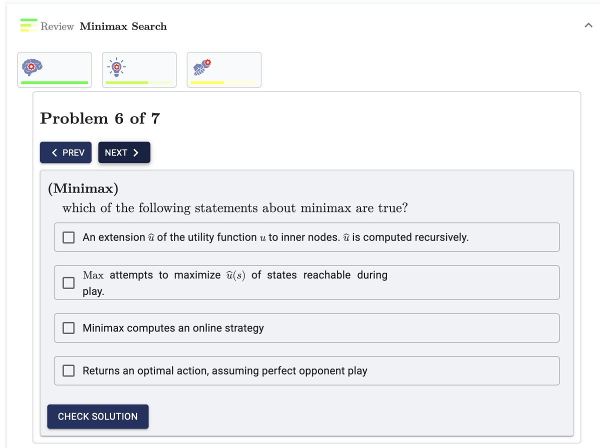
- ▷ A relatively non-intrusive overview over competency



- ▷ Click to extend it for details.



- ▷ Practice problems as usual. (targeted to your specific competency)



While the learning support services up to now have been adressed to individual learners, we now turn to services addressed to communities of learners, ranging from study groups with three learners, to whole courses, and even – eventually – all the alumni of a course, if they have not de-registered from ALeA.

Currently, the community aspect of ALeA only consists in localized interactions with the course materials.

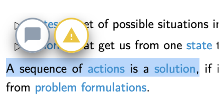
The ALeA system uses the semantic structure of the course materials to localize some interactions that are otherwise often from separate applications. Here we see two:

1. one for reporting content errors – and thus making the material better for all learners – and‘‘

2. a localized course forum, where forum threads can be attached to learning objects.

Localized Interactions with the Community

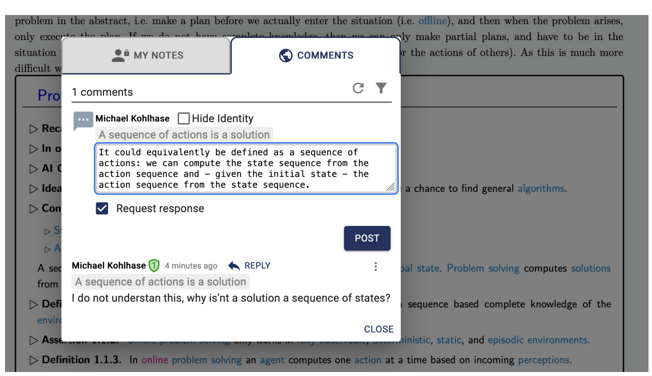
▷ Selecting text brings up localized – i.e. anchored on the selection – interactions:



▷ post a (public) comment or take (private) note

▷ report an error to the course authors/instructors

▷ Localized comments induce a thread in the ALEA forum (like the StudOn Forum, but targeted towards specific learning objects.)



▷ Answering questions gives karma $\widehat{=}$ a public measure of user helpfulness.

▷ Notes can be anonymous (⇝ generate no karma)

Let us briefly look into how the learning support services introduced above might work, focusing on where the necessary information might come from. Even though some of the concepts in the discussion below may be new to LBS students, it is worth looking into them. Bear with us as we try to explain the AI components of the ALeA system.

ALeA $\widehat{=}$ Data-Driven & AI-enabled Learning Assistance

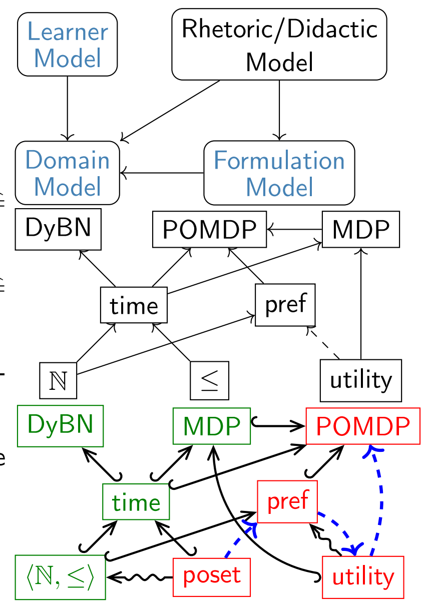
▷ **Idea:** Do what a teacher does! Use/maintain four models:

▷ **Ingredient 1:** Domain model $\widehat{=}$ knowledge/theory graph

▷ **Ingredient 2:** Learner model $\widehat{=}$ adding competency estimations

▷ **Ingredient 3:** A collection of ready-formulated learning objects

▷ **Ingredient 4:** Educational dialogue planner ⇝ guided tours



(Good) teachers

▷ understand the objects and their properties they are talking about

▷ have readimade formulations how to convey them best

▷ and understand how these best work together

▷ model what the learners already know/understand and adapts them accordingly

A theory graph provides (modular representation of the domain)

▷ symbols with URIs for all concepts, objects, and relations

▷ definitions, notations, and verbalizations for all symbols

▷ "object-oriented inheritance" and views between theories.

The learner model is a function from learner IDs $\times$ symbol URIs to competency values

▷ competency comes in six cognitive dimensions: `remember`, `understand`, `analyze`, `evaluate`, `apply`, and `create`.

▷ ALeA logs all learner interactions (keeps data learner-private)

- ▷ each interaction updates the learner model function.

Learning objects are the text fragments learners see and interact with; they are structured by

- ▷ didactic relations, e.g. tasks have prerequisites and learning objectives
- ▷ rhetoric relations, e.g. introduction, elaboration, and transition

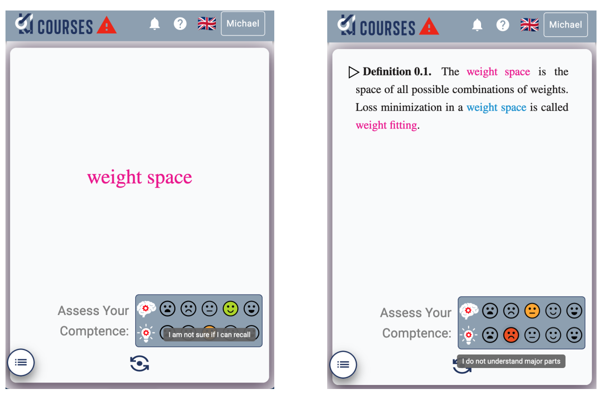
The dialogue planner assembles learning objects into active course material using

- ▷ the domain model and didactic relations to determine the order of LOs
- ▷ the learner model to determine what to show
- ▷ the rhetoric relations to make the dialogue coherent

We can use the same four models discussed in the space of guided tours to deploy additional learning support services, which we now discuss.

New Feature: Drilling with Flashcards

- ▷ Flashcards challenge you with a task (term/problem) on the front...



...and the definition/answer is on the back.

- ▷ Self-assessment updates the learner model (before/after)
- ▷ **Idea:** Challenge yourself to a card stack, keep drilling/assessing flashcards until the learner model eliminates all.
- ▷ **Bonus:** Flashcards can be generated from existing semantic markup (educational equivalent to free beer)

We have already seen above how the learner model can drive the drilling with flashcards. It can also be used for the configuration of card stacks by configuring a domain e.g. a section in the course materials and a competency threshold. We now come to a very important issue that we always face when we do AI systems that interface with humans. Most web technology companies that take one the approach "the user pays for the services with their personal data, which is sold on" or integrate advertising for renumeration. Both are not acceptable in university setting.

But abstaining from monetizing personal data still leaves the problem how to protect it from intentional or accidental misuse. Even though the GDPR has quite extensive exceptions for research, the ALeA system – a research prototype – adheres to the principles and mandates of the GDPR. In particular it makes sure that personal data of the learners is only used in learning support services directly or indirectly initiated by the learners themselves.

Learner Data and Privacy in ALEA

- **Observation:** Learning support services in ALEA use the learner model; they
  - need the learner model data to adapt to the invidivual learner!
  - collect learner interaction data (to update the learner model)
- **Consequence:** You need to be logged in (via your FAU IDM credentials) for useful learning support services!
- **Problem:** Learner model data is highly sensitive personal data!
- **ALeA Promise:** The ALEA team does the utmost to keep your personal data safe. (SSO via FAU IDM/eduGAIN, ALEA trust zone)
- **ALeA Privacy Axioms:**
  1. ALEA only collects learner models data about logged in users.
  2. Personally identifiable learner model data is only accessible to its subject (delegation possible)
  3. Learners can always query the learner model about its data.
  4. All learner model data can be purged without negative consequences (except usability deterioration)
  5. Logging into ALEA is completely optional.
- **Observation:** Authentication for bonus quizzes are somewhat less optional, but you can always purge the learner model later.

Michael Kohlhase: LBS 311 2025-02-06

So, now that you have an overview over what the ALEA system can do for you, let us see what you have to concretely do to be able to use it.

Concrete Todos for ALeA

- **Recall:** You will use ALeA for the prepquizzes (or lose bonus points)
  All other use is optional. (but AI-supported pre/postparation can be helpful)
- To use the ALeA system, you will have to log in via SSO: (do it now)
  - go to `https://courses.voll-ki.fau.de/course-home/lbs`,
  - in the upper right hand corner you see 
,
  - log in via your FAU IDM credentials. (you should have them by now)
  - You get access to your personal ALeA profile via 

    (plus feature notifications, manual, and language chooser)

▷ **Problem:** Most ALeA services depend on the learner model. (to adapt to you)

▷ **Solution:** Initialize your learner model with your educational history!

- ▹ **Concretely**: enter taken CS courses (FAU equivalents) and grades.
- ▹ ALeA uses that to estimate your CS/AI competencies. (for your benefit)
- ▹ then ALeA knows about you; I don't! (ALeA trust zone)

Even if you did not understand some of the AI jargon or the underlying methods (yet), you should be good to go for using the ALeA system in your day-to-day work.

Appendix B

Properties of the Simply Typed λ Calculus

B.1 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 section we prove the respective properties. We start out with termination, since we will need it later in the proof of confluence.

B.1.1 Termination of β-reduction

We will use the termination of $=_\beta$ 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
  $(\lambda X.XX)\ (\lambda X.XX) \rightarrow_\beta (\lambda X.XX)\ (\lambda X.XX)$
- **Theorem B.1.1 (Typed β-Reduction terminates).** *For all $\mathbf{A} \in wff_\alpha(\Sigma_\mathcal{T}, \mathcal{V}_\mathcal{T})$, 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]

FAU Michael Kohlhase: LBS 313 2025-02-06

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 B.1.2.** $\mathbf{A}$ is called strongly reducing at type α (write $\mathcal{SR}(\mathbf{A},\alpha)$), iff each chain β-reductions from $\mathbf{A}$ terminates.

▷ **Definition B.1.3.** 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 $\mathcal{LR}(\mathbf{C}\ \mathbf{A},\beta)$ for all $\mathbf{A}\in \mathit{wff}_\alpha(\Sigma_\mathcal{T},\mathcal{V}_\mathcal{T})$ with $\mathcal{LR}(\mathbf{A},\alpha)$.

▷ *Proof:* Termination Proof

1. $\mathcal{LR}\subseteq\mathcal{SR}$ (?? b))
2. $\mathbf{A}\in \mathit{wff}_\alpha(\Sigma_\mathcal{T},\mathcal{V}_\mathcal{T})$ implies $\mathcal{LR}(\mathbf{A},\alpha)$ (?? with $\sigma=\emptyset$)
3. thus $\mathcal{SR}(\mathbf{A},\alpha)$.

▷ **Lemma B.1.4 ($\mathcal{SR}$ 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 sketch: Every infinite β reduction from $\mathbf{B}$ would be one from $\mathbf{A}$.

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 B.1.5 (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

1. α base type
   1.1. a)
      1.1.1. $h\ \overline{\mathbf{A}^n}$ is strongly reducing, since the $\mathbf{A}^i$ are (brackets!)
      1.1.2. so $\mathcal{LR}(h\ \overline{\mathbf{A}^n},\alpha)$ as α is a base type ($\mathcal{SR}=\mathcal{LR}$)
   1.2. b)*by definition*
2. $\alpha=\beta\to\gamma$
   2.1. a)
      2.1.1. Let $\mathcal{LR}(\mathbf{B},\beta)$.
      2.1.2. by IH b) we have $\mathcal{SR}(\mathbf{B},\beta)$, and $\mathcal{LR}((h\ \overline{\mathbf{A}^n})\ \mathbf{B},\gamma)$ by IH a)
      2.1.3. so $\mathcal{LR}(h\ \overline{\mathbf{A}^n},\alpha)$ by definition.

2.2. b)
 2.2.1. Let $\mathcal{LR}(\mathbf{A},\alpha)$ and $X_\beta \notin \mathrm{free}(\mathbf{A})$.
 2.2.2. $\mathcal{LR}(X,\beta)$ by IH a) with $n=0$, thus $\mathcal{LR}(\mathbf{A}\ X,\gamma)$ by definition.
 2.2.3. By IH b) we have $\mathcal{SR}(\mathbf{A}X,\gamma)$ and by **??** $\mathcal{SR}(\mathbf{A},\alpha)$.

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 B.1.6.** *If $\mathcal{LR}([\mathbf{B}/X](\mathbf{A}),\alpha)$ and $\mathcal{LR}(\mathbf{B},\beta)$ for $X_\beta \notin \mathrm{free}(\mathbf{B})$, then $\mathcal{LR}((\lambda X_\alpha.\mathbf{A})\ \mathbf{B},\alpha)$.*

▷ *Proof:*

1. Let $\alpha = \overline{\gamma}_i \rightarrow \delta$ where δ base type and $\mathcal{LR}(\mathrm{C}^i,\gamma^i)$
2. It is sufficient to show that $\mathcal{SR}((\lambda X.\mathbf{A})\ \mathbf{B}\ \overline{\mathbf{C}},\delta)$, as δ base type
3. We have $\mathcal{LR}(([\mathbf{B}/X](\mathbf{A}))\ \overline{\mathbf{C}},\delta)$ by hypothesis and definition of $\mathcal{LR}$.
4. thus $\mathcal{SR}(([\mathbf{B}/X](\mathbf{A}))\ \overline{\mathbf{C}},\delta)$, as δ base type.
5. in particular $\mathcal{SR}([\mathbf{B}/X](\mathbf{A}),\alpha)$ and $\mathcal{SR}(\mathrm{C}^i,\gamma^i)$ (subterms)
6. $\mathcal{SR}(\mathbf{B},\beta)$ by hypothesis and **??**
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 C^i.
8. so an infinite reduction from $(\lambda X.\mathbf{A})\ \mathbf{B}\ \overline{\mathbf{C}}$ must have the form
$$\begin{aligned}(\lambda X.\mathbf{A})\ \mathbf{B}\ \overline{\mathbf{C}} &\rightarrow^*_\beta (\lambda X.\mathbf{A}')\ \mathbf{B}'\ \overline{\mathbf{C}'}\\ &\rightarrow^1_\beta ([\mathbf{B}'/X](\mathbf{A}'))\ \overline{\mathbf{C}'}\\ &\rightarrow^*_\beta \ldots\end{aligned}$$
where $\mathbf{A}\rightarrow^*_\beta\mathbf{A}'$, $\mathbf{B}\rightarrow^*_\beta\mathbf{B}'$ and $\mathrm{C}^i\rightarrow^*_\beta\mathrm{C}^{i'}$
9. so we have $[\mathbf{B}/X](\mathbf{A})\rightarrow^*_\beta[\mathbf{B}'/X](\mathbf{A}')$
10. so we have the infinite reduction
$$\begin{aligned}([\mathbf{B}/X](\mathbf{A}))\ \overline{\mathbf{C}} &\rightarrow^*_\beta ([\mathbf{B}'/X](\mathbf{A}'))\ \overline{\mathbf{C}'}\\ &\rightarrow^*_\beta \ldots\end{aligned}$$
which contradicts our assumption

▷ **Lemma B.1.7 ($\mathcal{LR}$ is closed under β-expansion).** *If $\mathbf{C}\rightarrow_\beta\mathbf{D}$ and $\mathcal{LR}(\mathbf{D},\alpha)$, so is $\mathcal{LR}(\mathbf{C},\alpha)$.*

Note that this Lemma is one of the few places in the termination proof, where we actually look at the properties of β 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 induction 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.

$\mathbf{A} \in wff_\alpha(\Sigma_\mathcal{T}, \mathcal{V}_\mathcal{T})$ implies $\mathcal{LR}(\mathbf{A}, \alpha)$

▷ **Definition B.1.8.** We write $\mathcal{LR}(\sigma)$ if $\mathcal{LR}(\sigma(X_\alpha), \alpha)$ for all $X \in \mathrm{supp}(\sigma)$.

▷ **Theorem B.1.9.** *If $\mathbf{A} \in wff_\alpha(\Sigma_\mathcal{T}, \mathcal{V}_\mathcal{T})$, then $\mathcal{LR}(\sigma(\mathbf{A}), \alpha)$ for any substitution σ with $\mathcal{LR}(\sigma)$.*

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

1. $\mathbf{A} = X_\alpha \in \mathrm{supp}(\sigma)$
   1.1. then $\mathcal{LR}(\sigma(\mathbf{A}), \alpha)$ by assumption
2. $\mathbf{A} = X \notin \mathrm{supp}(\sigma)$
   2.1. then $\sigma(\mathbf{A}) = \mathbf{A}$ and $\mathcal{LR}(\mathbf{A}, \alpha)$ by **??** with $n = 0$.
3. $\mathbf{A} \in \Sigma_\mathcal{T}$
   3.1. then $\sigma(\mathbf{A}) = \mathbf{A}$ as above
4. $\mathbf{A} = \mathbf{BC}$
   4.1. by IH $\mathcal{LR}(\sigma(\mathbf{B}), \gamma \to \alpha)$ and $\mathcal{LR}(\sigma(\mathbf{C}), \gamma)$
   4.2. so $\mathcal{LR}((\sigma(\mathbf{B}))\ (\sigma(\mathbf{C})), \alpha)$ by definition of $\mathcal{LR}$.
5. $\mathbf{A} = \lambda X_\beta.\mathbf{C}_\gamma$
   5.1. Let $\mathcal{LR}(\mathbf{B}, \beta)$ and $\theta := \sigma,[\mathbf{B}/X]$, then θ meets the conditions of the IH.
   5.2. Moreover $(\sigma(\lambda X_\beta.\mathbf{C}_\gamma))\ \mathbf{B} \to_\beta \sigma,[\mathbf{B}/X](\mathbf{C}) = \theta(\mathbf{C})$.
   5.3. Now, $\mathcal{LR}(\theta(\mathbf{C}), \gamma)$ by IH and thus $\mathcal{LR}((\sigma(\mathbf{A}))\ \mathbf{B}, \gamma)$ by **??**.
   5.4. So $\mathcal{LR}(\sigma(\mathbf{A}), \alpha)$ by definition of $\mathcal{LR}$.

FAU Michael Kohlhase: LBS 317 2025-02-06

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 **??**, 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 **??**– becomes easy.

The last case on λ-abstraction reveals why we had to strengthen the induction hypothesis: β reduction introduces a substitution which may increase the size of the subterm, which in turn keeps us from applying the induction 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 $=_\beta$-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 $=_\beta$ reduction.

Closure under Head β-Expansion (weakly reducing)

▷ **Lemma B.1.10 ($\mathcal{LR}$ 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

1. α base type
   1.1. we have $\mathcal{SR}(\mathbf{D}, \alpha)$ by definition

1.2. so $\mathcal{SR}(\mathbf{C}, \alpha)$, since head reduction is unique
1.3. and thus $\mathcal{LR}(\mathbf{C}, \alpha)$.

2. $\alpha = \beta \rightarrow \gamma$
   2.1. Let $\mathcal{LR}(\mathbf{B}, \beta)$, by definition we have $\mathcal{LR}(\mathbf{DB}, \gamma)$.
   2.2. but $\mathbf{C}\ \mathbf{B} \rightarrow_{\beta}^{h} \mathbf{D}\ \mathbf{B}$, so $\mathcal{LR}(\mathbf{CB}, \gamma)$ by IH
   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 $=_{\beta}$-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 **??** (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.

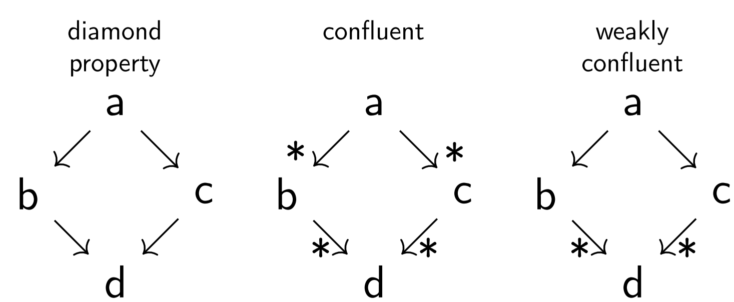
B.1.2 Confluence of $\beta\eta$ Conversion

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 B.1.11 (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 $a \rightarrow^1{}_R b$ $a \rightarrow^1{}_R c$ there is a $d \in A$ with $b \rightarrow^1{}_R d$ and $c \rightarrow^1{}_R d$.
- ▹ is confluent, iff for every $a, b, c \in A$ with $a \rightarrow^*{}_R b$ $a \rightarrow^*{}_R c$ there is a $d \in A$ with $b \rightarrow^*{}_R d$ and $c \rightarrow^*{}_R d$.
- ▹ weakly confluent iff for every $a, b, c \in A$ with $a \rightarrow^1{}_R b$ $a \rightarrow^1{}_R c$ there is a $d \in A$ with $b \rightarrow^*{}_R d$ and $c \rightarrow^*{}_R d$.



The diamond property is very simple, but not many reduction relations enjoy it. Confluence is the notion 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 B.1.12.** *If a rewrite relation has a diamond property, then it is weakly confluent.*
- ▷ **Theorem B.1.13.** *If a rewrite relation has a diamond property, then it is confluent.*
- ▷ *Proof sketch:* by a tiling argument, composing 1×1 diamonds to an $n \times m$ diamond.
- ▷ **Theorem B.1.14 (Newman's Lemma).** *If a rewrite relation is terminating and weakly confluent, then it is also confluent.*

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.
$\rightarrow_\eta^*$ is very well-behaved – i.e. confluent and terminating

η-Reduction ist terminating and confluent

- ▷ **Lemma B.1.15.** *η-Reduction ist terminating*
- ▷ *Proof sketch:* by a simple counting argument
- ▷ **Lemma B.1.16.** *η-reduction is confluent.*
- ▷ *Proof sketch:* 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 **??**.

For $=_\beta$-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 **??**

$=_\beta$ is confluent

- ▷ **Lemma B.1.17.** *$=_\beta$-Reduction is weakly confluent.*
- ▷ *Proof sketch:* by diagram chase over

$$\begin{tikzcd}[column sep=tiny]
& & (\lambda X.\mathbf{A})\,\mathbf{B} \arrow[dll] \arrow[d] \arrow[drr] & & \\
(\lambda X.\mathbf{A}')\,\mathbf{B} \arrow[dr] & & (\lambda X.\mathbf{A})\,\mathbf{B}' \arrow[dl] \arrow[dr] & & [\mathbf{B}/X](\mathbf{A}) \arrow[dl, "*"] \\
& (\lambda X.\mathbf{A}')\,\mathbf{B}' \arrow[dr] & & [\mathbf{B}'/X](\mathbf{A}) \arrow[dl] & \\
& & [\mathbf{B}'/X](\mathbf{A}') & &
\end{tikzcd}$$

▷ **Corollary B.1.18.** *$=_\beta$-Reduction is confluent.*

Proof sketch: by Newman's Lemma.

There is one reduction in the diagram in the proof of ?? 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 $=_\beta$ and $=_\eta$ 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 B.1.19.** 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 $Z \to_\mathcal{R} W$.

$$\begin{tikzcd}[column sep=small]
& X \arrow[dl, "\mathcal{R}"'] \arrow[dr, "\mathcal{S}"] & \\
Y \arrow[dr, "\mathcal{S}"'] & & Z \arrow[dl, "\mathcal{R}"] \\
& W &
\end{tikzcd}$$

▷ **Observation B.1.20.** *If $\mathcal{R}$ and $\mathcal{S}$ commute, then $\to_\mathcal{R}$ and $\to_\mathcal{S}$ do as well.*

▷ **Observation B.1.21.** *$\mathcal{R}$ is confluent, if $\mathcal{R}$ commutes with itself.*

▷ **Lemma B.1.22.** *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.

This directly gives us our goal.

$\to^*_{\beta\eta}$ is confluent

▷ **Lemma B.1.23.** *$\to^*_\beta$ and $\to^*_\eta$ commute.*

▷ *Proof sketch:* diagram chase

B.2 The Semantics of the Simply Typed λ-Calculus

The semantics of $\Lambda^{\rightarrow}$ 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 $\Lambda^{\rightarrow}$) 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 $\Lambda^{\rightarrow}$

▷ **Definition B.2.1.** We call a collection $\mathcal{D}_{\mathcal{T}} := \{\mathcal{D}_\alpha \mid \alpha \in \mathcal{T}\}$ 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 B.2.2.** 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 B.2.3.** Given a frame $\mathcal{D}_{\mathcal{T}}$, and a typed function $\mathcal{I} \colon \Sigma \to \mathcal{D}$, we call $\mathcal{I}_\varphi \colon \mathit{wff}_{\mathcal{T}}(\Sigma_{\mathcal{T}}, \mathcal{V}_{\mathcal{T}}) \to \mathcal{D}$ the value function induced by $\mathcal{I}$, iff

1. $\mathcal{I}_\varphi|_{\mathcal{V}_{\mathcal{T}}} = \varphi$, $\quad \mathcal{I}_\varphi|_{\Sigma_{\mathcal{T}}} = \mathcal{I}$,
2. $\mathcal{I}_\varphi(\mathbf{A}\,\mathbf{B}) = \mathcal{I}_\varphi(\mathbf{A})(\mathcal{I}_\varphi(\mathbf{B}))$, and
3. $\mathcal{I}_\varphi(\lambda X_\alpha.\mathbf{A})$ is that function $f \in \mathcal{D}_{\alpha \to \beta}$, such that $f(a) = \mathcal{I}_{\varphi,[a/X]}(\mathbf{A})$ for all $a \in \mathcal{D}_\alpha$.

▷ **Note:** Not every λ-term has a $\mathcal{I}_\varphi$-value as we have only required $\mathcal{D}_{\alpha \to \beta} \subseteq \mathcal{D}_\alpha \to \mathcal{D}_\beta$ for frames. (there might not be enough functions)

▷ **Definition B.2.4.** We call $\langle \mathcal{D}, \mathcal{I} \rangle$, where $\mathcal{D}$ is a frame and $\mathcal{I}$ is a typed function comprehension closed or a $\Sigma_{\mathcal{T}}$-algebra, iff $\mathcal{I}_\varphi \colon \mathit{wff}_{\mathcal{T}}(\Sigma_{\mathcal{T}}, \mathcal{V}_{\mathcal{T}}) \to \mathcal{D}$ is total.

▷ **Theorem B.2.5.** *$=_{\alpha\beta\eta}$ (seen as a calculus) is sound and complete for Σ-algebras.*

▷ **Upshot for LBS:** $\Lambda^{\rightarrow}$ is *the* logical system for reasoning about functions!

The definition of the semantics in **??** is surprisingly simple. The only part that is new at all is the third clause, and there we already know the trick with treating binders by extending the variable assignment from quantifiers in first-order logic.

The real subtlety is in the definition of frames, where instead of requiring $\mathcal{D}_{\alpha \to \beta} = \mathcal{D}_\alpha \to \mathcal{D}_\beta$ (full function universes we have only required $\mathcal{D}_{\alpha \to \beta} \subseteq \mathcal{D}_\alpha \to \mathcal{D}_\beta$, which necessitates the post-hoc definition of a $\Sigma_{\mathcal{T}}$-algebra. But the added complexity gives us thm.abe-sound-complete.

B.2.1 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 B.2.6 (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 $\mathbf{A}$

we have five cases

1. $\mathbf{A} = X$

1.1. Then $\mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{A})) = \mathcal{I}_\varphi([\mathbf{B}/X](X)) = \mathcal{I}_\varphi(\mathbf{B}) = \psi(X) = \mathcal{I}_\psi(X) = \mathcal{I}_\psi(\mathbf{A})$.

2. $\mathbf{A} = Y \neq X$ and $Y \in \mathcal{V}_\mathcal{T}$

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

3. $\mathbf{A} \in \Sigma_\mathcal{T}$

3.1. This is analogous to the last case.

4. $\mathbf{A} = \mathbf{C}\ \mathbf{D}$

4.1. then $\mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{A})) = \mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{C}\ \mathbf{D})) = \mathcal{I}_\varphi(([\mathbf{B}/X](\mathbf{C}))\ ([\mathbf{B}/X](\mathbf{D}))) = \mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{C}))(\mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{D}))) = \mathcal{I}_\psi(\mathbf{C})(\mathcal{I}_\psi(\mathbf{D})) = \mathcal{I}_\psi(\mathbf{C}\ \mathbf{D}) = \mathcal{I}_\psi(\mathbf{A})$

5. $\mathbf{A} = \lambda Y_\alpha.\mathbf{C}$

5.1. We can assume that $X \neq Y$ and $Y \notin \mathrm{free}(\mathbf{B})$

5.2. Thus for all $a \in \mathcal{D}_\alpha$ we have $\mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{A}))(a) = \mathcal{I}_\varphi([\mathbf{B}/X](\lambda Y.\mathbf{C}))(a) = \mathcal{I}_\varphi(\lambda Y.([\mathbf{B}/X](\mathbf{C})))(a) = \mathcal{I}_{\varphi,[a/Y]}([\mathbf{B}/X](\mathbf{C})) = \mathcal{I}_{\psi,[a/Y]}(\mathbf{C}) = \mathcal{I}_\psi(\lambda Y.\mathbf{C})(a) = \mathcal{I}_\psi(\mathbf{A})(a)$

Soundness of $\alpha\beta\eta$-Equality

▷ **Theorem B.2.7.** *Let $\mathcal{A} := \langle\mathcal{D},\mathcal{I}\rangle$ be a $\Sigma_\mathcal{T}$-algebra and $Y \notin \mathrm{free}(\mathbf{A})$, 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

$$\begin{aligned}\mathcal{I}_\varphi(\lambda Y.[Y/X]\mathbf{A})@a &= \mathcal{I}_{\varphi,[a/Y]}([Y/X](\mathbf{A}))\\ &= \mathcal{I}_{\varphi,[a/X]}(\mathbf{A})\\ &= \mathcal{I}_\varphi(\lambda X.\mathbf{A})@a\end{aligned}$$

▷ **Theorem B.2.8.** *If $\mathcal{A} := \langle\mathcal{D},\mathcal{I}\rangle$ is a $\Sigma_\mathcal{T}$-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

▷

$$\begin{aligned}\mathcal{I}_\varphi((\lambda X.\mathbf{A})\ \mathbf{B}) &= \mathcal{I}_\varphi(\lambda X.\mathbf{A})@\mathcal{I}_\varphi(\mathbf{B})\\ &= \mathcal{I}_{\varphi,[\mathcal{I}_\varphi(\mathbf{B})/X]}(\mathbf{A})\\ &= \mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{A}))\end{aligned}$$

Soundness of $\alpha\beta\eta$ (continued)

▷ **Theorem B.2.9.** *If $X \notin \mathrm{free}(\mathbf{A})$, then $\mathcal{I}_\varphi(\lambda X.\mathbf{A}\ X) = \mathcal{I}_\varphi(\mathbf{A})$ for all φ.*

▷ *Proof:* by calculation

$$
\begin{aligned}
\mathcal{I}_\varphi(\lambda X.\mathbf{A}\ X)@a &= \mathcal{I}_{\varphi,[a/X]}(\mathbf{A}\ X) \\
&= \mathcal{I}_{\varphi,[a/X]}(\mathbf{A})@\mathcal{I}_{\varphi,[a/X]}(X) \\
&= \mathcal{I}_\varphi(\mathbf{A})@\mathcal{I}_{\varphi,[a/X]}(X) \qquad \text{as } X \notin \mathrm{free}(\mathbf{A}). \\
&= \mathcal{I}_\varphi(\mathbf{A})@a
\end{aligned}
$$

▷ **Theorem B.2.10.** *$\alpha\beta\eta$-equality is sound wrt. $\Sigma_\mathcal{T}$-algebras. (if $\mathbf{A} =_{\alpha\beta\eta} \mathbf{B}$, then $\mathcal{I}_\varphi(\mathbf{A}) = \mathcal{I}_\varphi(\mathbf{B})$ for all assignments φ)*

FAU Michael Kohlhase: LBS 329 2025-02-06

B.2.2 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. x

Normal Forms in the simply typed λ-calculus

▷ **Definition B.2.11.** We call a term $\mathbf{A} \in \mathit{wff}_\mathcal{T}(\Sigma_\mathcal{T}, \mathcal{V}_\mathcal{T})$ a β normal form iff there is no $\mathbf{B} \in \mathit{wff}_\mathcal{T}(\Sigma_\mathcal{T}, \mathcal{V}_\mathcal{T})$ with $\mathbf{A} \rightarrow_\beta \mathbf{B}$.

We call $\mathbf{N}$ a β normal form of $\mathbf{A}$, iff $\mathbf{N}$ is a β-normal form and $\mathbf{A} \rightarrow_\beta \mathbf{N}$.

We denote the set of β-normal forms with $wff_\mathcal{T}(\Sigma_\mathcal{T}, \mathcal{V}_\mathcal{T})\big\downarrow_{\beta\eta}$.

The η- and $\beta\eta$ normal forms are definied analogously.

▷ We have just proved that β, η, and $\beta\eta$-reduction are terminating and confluent, so we have

▷ **Corollary B.2.12 (Normal Forms).** *Every $\mathbf{A} \in \mathit{wff}_\mathcal{T}(\Sigma_\mathcal{T}, \mathcal{V}_\mathcal{T})$ has a unique β normal form ($\beta\eta$, long $\beta\eta$ normal form), which we denote by $\mathbf{A}\downarrow_\beta$ ($\mathbf{A}\downarrow_{\beta\eta}$ $\mathbf{A}\downarrow_{\beta\eta^l}$).*

FAU Michael Kohlhase: LBS 330 2025-02-06

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 B.2.13.** 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 B.2.14.** 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 B.2.15.** $=_\beta$ ($=_{\beta\eta}$) is a (functional) congruence on $\mathit{cwff}_\mathcal{T}(\Sigma_\mathcal{T})$ by definition.

▷ **Theorem B.2.16.** *Let $\mathcal{DT}$ be a $\Sigma_\mathcal{T}$-frame and $\sim$ a functional congruence on $\mathcal{D}$, then the quotient space $\mathcal{D}/\sim$ is a $\Sigma_\mathcal{T}$-frame.*

▷ *Proof:*

1. $\mathcal{D}/\sim=\{[f]_\sim \mid f\in\mathcal{D}\}$, define $[f]_\sim([a]_\sim):=[f(a)]_\sim$.
2. This only depends on equivalence classes: Let $f'\in[f]_\sim$ and $a'\in[a]_\sim$.
3. Then $[f(a)]_\sim=[f'(a)]_\sim=[f'(a')]_\sim=[f(a')]_\sim$
4. To see that we have $[f]_\sim=[g]_\sim$, iff $f\sim g$, iff $f(a)=g(a)$ since $\sim$ is functional.
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 $[a]_\sim\in\mathcal{D}/\sim$.

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 $wff_\mathcal{T}(\Sigma_\mathcal{T},\mathcal{V}_\mathcal{T})$ and then specialize this result to show that is is also functional on $cwff_\mathcal{T}(\Sigma_\mathcal{T})$ by a grounding argument.

$\beta\eta$-Equivalence as a Functional Congruence

▷ **Lemma B.2.17.** *$\beta\eta$-equality is a functional congruence on $wff_\mathcal{T}(\Sigma_\mathcal{T},\mathcal{V}_\mathcal{T})$.*

▷ *Proof:* Let $\mathbf{A}\ \mathbf{C}=_{\beta\eta}\mathbf{B}\ \mathbf{C}$ for all $\mathbf{C}$ and $X\in(\mathcal{V}_\gamma\backslash(\mathrm{free}(\mathbf{A})\cup\mathrm{free}(\mathbf{B})))$.

1. then (in particular) $\mathbf{A}\ X=_{\beta\eta}\mathbf{B}\ X$, and
2. $\lambda X.\mathbf{A}\ X=_{\beta\eta}\lambda X.\mathbf{B}\ X$, since $\beta\eta$-equality acts on subterms.
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 B.2.18.** We call an injective substitution $\sigma\colon\mathrm{free}(\mathbf{C})\to\Sigma_\mathcal{T}$ a grounding substitution for $\mathbf{C}\in wff_\mathcal{T}(\Sigma_\mathcal{T},\mathcal{V}_\mathcal{T})$, iff no $\sigma(X)$ occurs in $\mathbf{C}$.

▷ **Observation:** They always exist, since all Σ_α are infinite and $\mathrm{free}(\mathbf{C})$ is finite.

▷ **Theorem B.2.19.** *$\beta\eta$-equality is a functional congruence on $cwff_\mathcal{T}(\Sigma_\mathcal{T})$.*

▷ *Proof:* We use **??**

1. Let $\mathbf{A},\mathbf{B}\in cwff_{(\alpha\to\beta)}(\Sigma_\mathcal{T})$, such that $\mathbf{A}\neq_{\beta\eta}\mathbf{B}$.
2. As $\beta\eta$ is functional on $wff_\mathcal{T}(\Sigma_\mathcal{T},\mathcal{V}_\mathcal{T})$, there must be a $\mathbf{C}$ with $\mathbf{A}\ \mathbf{C}\neq_{\beta\eta}\mathbf{B}\ \mathbf{C}$.
3. Now let $\mathbf{C}':=\sigma(\mathbf{C})$, for a grounding substitution σ.
4. Any $\beta\eta$ conversion sequence for $\mathbf{A}\ \mathbf{C}'\neq_{\beta\eta}\mathbf{B}\ \mathbf{C}'$ induces one for $\mathbf{A}\ \mathbf{C}\neq_{\beta\eta}\mathbf{B}\ \mathbf{C}$.
5. Thus we have shown that $\mathbf{A}\neq_{\beta\eta}\mathbf{B}$ entails $\mathbf{A}\ \mathbf{C}'\neq_{\beta\eta}\mathbf{B}\ \mathbf{C}'$.

Note that: the result for $cwff_\mathcal{T}(\Sigma_\mathcal{T})$ is sharp. For instance, if $\Sigma_\mathcal{T}=\{c_\iota\}$, 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 $\{c\}=cwff_\iota(\Sigma_\mathcal{T})$ (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 $wff_\mathcal{T}(\Sigma_\mathcal{T},\mathcal{V}_\mathcal{T})$ we could always have used a variable.

This completes the preparation and we can define the notion of a term algebra, i.e. a $\Sigma_\mathcal{T}$-algebra whose frame is made of $=_{\beta\eta}$-normal λ-terms.

A Herbrand Model for $\Lambda^{\to}$

▷ **Definition B.2.20.** We call $\mathcal{T}_{\beta\eta} := \langle cwff_{\mathcal{T}}(\Sigma_{\mathcal{T}})\downarrow_{\beta\eta}, \mathcal{I}^{\beta\eta}\rangle$ the Σ term algebra, if $\mathcal{I}^{\beta\eta} = \mathrm{Id}_{\Sigma_{\mathcal{T}}}$.

▷ The name "term algebra" in the previous definition is justified by the following

▷ **Theorem B.2.21.** *$\mathcal{T}_{\beta\eta}$ is a $\Sigma_{\mathcal{T}}$-algebra*

▷ *Proof:* We use the work we did above

1. Note that $cwff_{\mathcal{T}}(\Sigma_{\mathcal{T}})\downarrow_{\beta\eta} = cwff_{\mathcal{T}}(\Sigma_{\mathcal{T}})/{=_{\beta\eta}}$ and thus a $\Sigma_{\mathcal{T}}$-frame by **??** and **??**.
2. So we only have to show that the value function $\mathcal{I}^{\beta\eta} = \mathrm{Id}_{\Sigma_{\mathcal{T}}}$ is total.
3. Let φ be an assignment into $cwff_{\mathcal{T}}(\Sigma_{\mathcal{T}})\downarrow_{\beta\eta}$.
4. Note that $\sigma := \varphi|_{\mathrm{free}(\mathbf{A})}$ is a substitution, since $\mathrm{free}(\mathbf{A})$ is finite.
5. A simple induction on the structure of $\mathbf{A}$ shows that $\mathcal{I}^{\beta\eta}{}_{\varphi}(\mathbf{A}) = (\sigma(\mathbf{A}))\downarrow_{\beta\eta}$.
6. So the value function is total since substitution application is.

FAU Michael Kohlhase: LBS 333 2025-02-06

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 $\Sigma_{\mathcal{T}}$-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 $\Sigma_{\mathcal{T}}$-algebras.

Completetness of $\alpha\beta\eta$-Equality

▷ **Theorem B.2.22.** *$\mathbf{A} = \mathbf{B}$ is valid in the class of $\Sigma_{\mathcal{T}}$-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 $\Sigma_{\mathcal{T}}$-algebra.

1. If $\mathbf{A} = \mathbf{B}$ is valid in all $\Sigma_{\mathcal{T}}$-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 $\mathbf{A} =_{\alpha\beta\eta} \mathbf{B}$.

If the equation has free variables, then the argument is more subtle.

2. Let σ be a grounding substitution for $\mathbf{A}$ and $\mathbf{B}$ and φ the induced variable assignment.
3. 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 $\sigma(\mathbf{B})$.
4. 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 $\mathbf{A} =_{\alpha\beta\eta} \mathbf{B}$.

FAU Michael Kohlhase: LBS 334 2025-02-06

?? and **??** complete our study of the semantics of the simply-typed λ-calculus by showing that it is an adequate logical system for modeling (the equality) of functions and their applications.

B.3 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 B.3.1.** A grammar for the raw terms of the simply typed λ-calculus:

$$\begin{array}{lll} \alpha & ::= & c \mid \alpha \to \alpha \\ \Sigma & ::= & \cdot \mid \Sigma, [c : \mathrm{type}] \mid \Sigma, [c{:}\alpha] \\ \Gamma & ::= & \cdot \mid \Gamma, [x{:}\alpha] \\ \mathbf{A} & ::= & c \mid X \mid \mathbf{A}^1\ \mathbf{A}^2 \mid \lambda X_\alpha.\mathbf{A} \end{array}$$

- **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.

Simply Typed λ-Calculus as an Inference System: Judgments

- **Definition B.3.2.** Judgments make statements about complex properties of the syntactic entities defined by the grammar.
- **Definition B.3.3.** Judgments for the simply typed λ-calculus

$\vdash \Sigma : \mathrm{sig}$	Σ is a well-formed signature
$\Sigma \vdash \alpha : \mathrm{type}$	α is a well-formed type given the type assumptions in Σ
$\Sigma \vdash \Gamma : \mathrm{ctx}$	Γ is a well-formed context given the type assumptions in Σ
$\Gamma \vdash_\Sigma \mathbf{A} : \alpha$	$\mathbf{A}$ has type α given the type assumptions in Σ and Γ

Simply Typed λ-Calculus as an Inference System: Rules

- **Definition B.3.4.** $\mathbf{A} \in wff_\alpha(\Sigma_\mathcal{T}, \mathcal{V}_\mathcal{T})$, iff $\Gamma \vdash_\Sigma \mathbf{A} : \alpha$ derivable in

$$\frac{\Sigma \vdash \Gamma : \mathrm{ctx} \quad \Gamma(X) = \alpha}{\Gamma \vdash_\Sigma X : \alpha}\ \mathrm{wff\ var} \qquad \frac{\Sigma \vdash \Gamma : \mathrm{ctx} \quad \Sigma(c) = \alpha}{\Gamma \vdash_\Sigma c : \alpha}\ \mathrm{wff\ const}$$

$$\frac{\Gamma \vdash_\Sigma \mathbf{A} : \beta \to \alpha \quad \Gamma \vdash_\Sigma \mathbf{B} : \beta}{\Gamma \vdash_\Sigma \mathbf{A}\ \mathbf{B} : \alpha}\ \mathrm{wff\ app} \qquad \frac{\Gamma, [X{:}\beta] \vdash_\Sigma \mathbf{A} : \alpha}{\Gamma \vdash_\Sigma \lambda X_\beta.\mathbf{A} : \beta \to \alpha}\ \mathrm{wff\ abs}$$

- **Oops:** this looks surprisingly like a natural deduction calculus. (⇝ Curry-Howard isomorphism)
- To be complete, we need rules for well-formed signatures, types and contexts
- **Definition B.3.5.**

$$\frac{}{\vdash \cdot : \mathrm{sig}}\ \mathrm{sig\,empty} \qquad \frac{\vdash \Sigma : \mathrm{sig}}{\vdash (\Sigma,[\alpha : \mathrm{type}]) : \mathrm{sig}}\ \mathrm{sig\,type}$$

$$\frac{\vdash \Sigma : \mathrm{sig} \quad \Sigma \vdash \alpha : \mathrm{type}}{\vdash (\Sigma,[c{:}\alpha]) : \mathrm{sig}}\ \mathrm{sig\,const}$$

$$\frac{\Sigma \vdash \alpha : \mathrm{type} \quad \Sigma \vdash \beta : \mathrm{type}}{\Sigma \vdash (\alpha \to \beta) : \mathrm{type}}\ \mathrm{typ\,fn} \qquad \frac{\vdash \Sigma : \mathrm{sig} \quad \Sigma(\alpha) = \mathrm{type}}{\Sigma \vdash \alpha : \mathrm{type}}\ \mathrm{typ\,start}$$

$$\frac{\vdash \Sigma : \mathrm{sig}}{\Sigma \vdash \cdot : \mathrm{ctx}}\ \mathrm{ctx\,empty} \qquad \frac{\Sigma \vdash \Gamma : \mathrm{ctx} \quad \Sigma \vdash \alpha : \mathrm{type}}{\Sigma \vdash (\Gamma,[X{:}\alpha]) : \mathrm{ctx}}\ \mathrm{ctx\,var}$$

FAU Michael Kohlhase: LBS 337 2025-02-06

Example: A Well-Formed Signature

- Let $\Sigma := [\alpha : \mathrm{type}],[f{:}\alpha \to \alpha \to \alpha]$, then Σ is a well-formed signature, since we have derivations $\mathcal{A}$ and $\mathcal{B}$

$$\frac{\vdash \cdot : \mathrm{sig}}{\vdash [\alpha : \mathrm{type}] : \mathrm{sig}}\ \mathrm{sig\,type} \qquad \frac{\mathcal{A} \quad [\alpha : \mathrm{type}](\alpha) = \mathrm{type}}{[\alpha : \mathrm{type}] \vdash \alpha : \mathrm{type}}\ \mathrm{typ\,start}$$

  and with these we can construct the derivation $\mathcal{C}$

$$\dfrac{\mathcal{A} \quad \dfrac{\mathcal{B} \quad \dfrac{\mathcal{B} \quad \mathcal{B}}{[\alpha : \mathrm{type}] \vdash (\alpha \to \alpha) : \mathrm{type}}\ \mathrm{typ\,fn}}{[\alpha : \mathrm{type}] \vdash (\alpha \to \alpha \to \alpha) : \mathrm{type}}\ \mathrm{typ\,fn}}{\vdash \Sigma : \mathrm{sig}}\ \mathrm{sig\,const}$$

FAU Michael Kohlhase: LBS 338 2025-02-06

Example: A Well-Formed λ-Term

- using Σ from above, we can show that $\Gamma := [X{:}\alpha]$ is a well-formed context:

$$\dfrac{\dfrac{\mathcal{C}}{\Sigma \vdash \cdot : \mathrm{ctx}}\ \mathrm{ctx\,empty} \quad \dfrac{\mathcal{C} \quad \Sigma(\alpha) = \mathrm{type}}{\Sigma \vdash \alpha : \mathrm{type}}\ \mathrm{typ\,start}}{\Sigma \vdash \Gamma : \mathrm{ctx}}\ \mathrm{ctx\,var}$$

  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

$$\dfrac{\dfrac{\dfrac{\mathcal{C} \quad \Sigma(f) = \alpha \to \alpha \to \alpha}{\Gamma \vdash_\Sigma f \colon \alpha \to \alpha \to \alpha}\text{wff const} \quad \dfrac{\mathcal{G}}{\Gamma \vdash_\Sigma X \colon \alpha}\text{wff var}}{\Gamma \vdash_\Sigma f\ X \colon \alpha \to \alpha}\text{wff app} \quad \dfrac{\mathcal{G}}{\Gamma \vdash_\Sigma X \colon \alpha}\text{wff var}}{\dfrac{\Gamma \vdash_\Sigma f\ X\ X \colon \alpha}{\cdot \vdash_\Sigma \lambda X_\alpha . f\ X\ X \colon \alpha \to \alpha}\text{wff abs}}\text{wff app}$$

$\beta\eta$-Equality by Inference Rules: One-Step Reduction

▷ **Definition B.3.6.** One-step Reduction ($+ \in \{\alpha, \beta, \eta\}$)

$$\dfrac{\Gamma, [X{:}\beta] \vdash_\Sigma \mathbf{A} \colon \alpha \quad \Gamma \vdash_\Sigma \mathbf{B} \colon \beta}{\Gamma \vdash_\Sigma (\lambda X.\mathbf{A})\ \mathbf{B} \to_\beta^1 [\mathbf{B}/X](\mathbf{A})} wff\beta top$$

$$\dfrac{\Gamma \vdash_\Sigma \mathbf{A} \colon \beta \to \alpha \quad X \notin \mathbf{dom}(\Gamma)}{\Gamma \vdash_\Sigma \lambda X.\mathbf{A}\ X \to_\eta^1 \mathbf{A}} wff\eta top$$

$$\dfrac{\Gamma \vdash_\Sigma \mathbf{A} \to_+^1 \mathbf{B} \quad \Gamma \vdash_\Sigma \mathbf{A}\ \mathbf{C} \colon \alpha}{\Gamma \vdash_\Sigma \mathbf{A}\ \mathbf{C} \to_+^1 \mathbf{B}\ \mathbf{C}} tr\, appfn$$

$$\dfrac{\Gamma \vdash_\Sigma \mathbf{A} \to_+^1 \mathbf{B} \quad \Gamma \vdash_\Sigma \mathbf{C}\ \mathbf{A} \colon \alpha}{\Gamma \vdash_\Sigma \mathbf{C}\ \mathbf{A} \to_+^1 \mathbf{C}\ \mathbf{B}} tr\, apparg$$

$$\dfrac{\Gamma, [X{:}\alpha] \vdash_\Sigma \mathbf{A} \to_+^1 \mathbf{B}}{\Gamma \vdash_\Sigma \lambda X.\mathbf{A} \to_+^1 \lambda X.\mathbf{B}} tr\, abs$$

$\beta\eta$-Equality by Inference Rules: Multi-Step Reduction

▷ **Definition B.3.7.** Multi-Step-Reduction ($+ \in \{\alpha, \beta, \eta\}$)

$$\dfrac{\Gamma \vdash_\Sigma \mathbf{A} \to_+^1 \mathbf{B}}{\Gamma \vdash_\Sigma \mathbf{A} \to_+^* \mathbf{B}} ms\, start \qquad \dfrac{\Gamma \vdash_\Sigma \mathbf{A} \colon \alpha}{\Gamma \vdash_\Sigma \mathbf{A} \to_+^* \mathbf{A}} ms\, ref$$

$$\dfrac{\Gamma \vdash_\Sigma \mathbf{A} \to_+^* \mathbf{B} \quad \Gamma \vdash_\Sigma \mathbf{B} \to_+^* \mathbf{C}}{\Gamma \vdash_\Sigma \mathbf{A} \to_+^* \mathbf{C}} ms\, trans$$

▷ Congruence Relation

$$\dfrac{\Gamma \vdash_\Sigma \mathbf{A} \to_+^* \mathbf{B}}{\Gamma \vdash_\Sigma \mathbf{A} =_+ \mathbf{B}} eq\, start$$

$$\dfrac{\Gamma \vdash_\Sigma \mathbf{A} =_+ \mathbf{B}}{\Gamma \vdash_\Sigma \mathbf{B} =_+ \mathbf{A}} eq\, sym \qquad \dfrac{\Gamma \vdash_\Sigma \mathbf{A} =_+ \mathbf{B} \quad \Gamma \vdash_\Sigma \mathbf{B} =_+ \mathbf{C}}{\Gamma \vdash_\Sigma \mathbf{A} =_+ \mathbf{C}} eq\, trans$$

Type Computation: Manage Types Algorithmically

- **Questions:**
  - type check: Is $\Gamma \vdash_{\Sigma} \mathbf{A} \colon \alpha$?
  - type inference: are there Γ, α, such that $\Gamma \vdash_{\Sigma} \mathbf{A} \colon \alpha$?
  - type reconstruction the above without type annotations at bound variables?
- prenex fragment makes problems decidable (see Curry-Howard isomorphism)
- **Algorithm (Hindley & Milner):**
  - invert inference rules
  - first-order unification,
  - universal generalization, minimization

Example Computation

rule tree	constraint
$$\dfrac{\dfrac{\dfrac{\dfrac{[X{:}\alpha]}{\Gamma, [X{:}\beta]}}{\Gamma, [X{:}\beta] \vdash_{\Sigma} X \colon \alpha} \quad \Gamma \vdash_{\Sigma} \lambda X.X \colon \beta \to \alpha}{\dfrac{\dfrac{[W{:}\delta] \in \Gamma, [Z{:}\gamma]}{\Gamma, [Z{:}\gamma] \vdash_{\Sigma} W \colon \delta}}{\Gamma \vdash_{\Sigma} \lambda Z.W \colon \beta}} \; \Gamma \vdash_{\Sigma} \lambda X.X(\lambda Z.W) \colon \alpha}{}$$	$\alpha = \beta$, $[W{:}\delta] \in \Gamma$, $\beta = \gamma \to \delta$

- unification: $\alpha = \beta = \gamma \to \delta$,
- minimization: $\Gamma = [W{:}\delta]$
- **Solution:** $[W{:}\delta]] \vdash_{\Sigma} \lambda X.X(\lambda Z.W) \colon \forall \gamma.\gamma \to \delta$

Appendix C

Higher-Order Dynamics

In this chapter we will develop a typed λ calculus that extend DRT-like dynamic logics like the simply typed λ calculus extends first-order logic.

C.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 C.1.1.** *A man sleeps.*

$\text{a\_man} = \lambda Q.\left(\begin{array}{|l|}\hline U \\ \hline \text{man}(U) \\ \hline \end{array} \otimes Q(U)\right)$

$\text{sleep} = \lambda X.\begin{array}{|l|}\hline \\ \hline \text{sleeps}(X) \\ \hline \end{array}$

Application and β-reduction:

$$\begin{array}{rcl} \text{a\_man\_sleep} & = & \text{a\_man}(\text{sleep}) \\ & \to_\beta & \begin{array}{|l|}\hline U \\ \hline \text{man}(U) \\ \hline \end{array} \otimes \begin{array}{|l|}\hline \\ \hline \text{sleeps}(U) \\ \hline \end{array} \to_\tau \begin{array}{|l|}\hline U \\ \hline \text{man}(U) \\ \text{sleeps}(U) \\ \hline \end{array} \end{array}$$

Michael Kohlhase: LBS 344 2025-02-06

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 C.1.2.** *A man^1 sleeps. He$_1$ snores.*

$$(\lambda PQ.(P \otimes Q))\ \text{a\_man\_sleep}\ \text{he\_snore}$$

$$\to_{=_\beta} \left(\lambda Q.\begin{array}{|l|}\hline U \\ \hline \mathrm{man}(U) \\ \mathrm{sleeps}(U) \\ \hline \end{array} \otimes Q\right) \begin{array}{|l|}\hline \\ \hline \mathrm{snores}(U) \\ \hline \end{array}$$

$$\to_\tau \begin{array}{|l|}\hline U \\ \hline \mathrm{man}(U) \\ \mathrm{sleeps}(U) \\ \hline \end{array} \otimes \begin{array}{|l|}\hline \\ \hline \mathrm{snores}(U) \\ \hline \end{array} \quad \to_\tau \begin{array}{|l|}\hline U \\ \hline \mathrm{man}(U) \\ \mathrm{sleeps}(U) \\ \mathrm{snores}(U) \\ \hline \end{array}$$

▷ **Example C.1.3 (Linear notation).** $(\lambda Q.(\delta U.\mathrm{man}(U) \wedge \mathrm{sleeps}(U) \wedge Q(U)))\ \text{he\_snore} \longrightarrow_{\beta\tau} \delta U.\mathrm{man}(U) \wedge \mathrm{sleeps}(U) \wedge \mathrm{snores}(U)$

Here we have our first surprise: the second $=_\beta$ reduction seems to capture the discourse referent U: intuitively it is "free" in $\delta U.\mathrm{snores}(U)$ and after $=_\beta$ 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, referent capture 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 [Eij97])
- ▷ **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.

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 [GKL96])
  - Underspecification (Pinkal 1995 [Pin96])
- are based on static type theory [Mon74]
- Higher-Order Dynamic Unification needed for dynamic variants of these

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.

C.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
$\lambda-$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: $\lambda-$DRT (simplified)

- **Definition C.2.1 (Types).** ι (individuals), o (conditions), t (DRSes), $\alpha \rightarrow \beta$ (functions), $s \rightarrow \alpha$ (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).
- **Definition C.2.2.** $=_{\alpha\beta\eta}$-reduction for the λ-calculus part, and further:
  - $(\delta \mathcal{X}.\mathbf{A} \otimes \delta \mathcal{Y}.\mathbf{B}) \rightarrow_\tau (\delta \mathcal{X} \cup \mathcal{Y}.\mathbf{A} \wedge \mathbf{B})$
  - ${}^{\vee\wedge}\mathbf{A} \rightarrow_\mu \mathbf{A}$
- **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 $\lambda-$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: ($\rightarrow_\tau$-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 $\mathbf{C}\left[(\delta U.\mathbf{A})\right]_\pi$ and $\mathbf{C}\left[(\delta V.[V/U](\mathbf{A}))\right]_\pi$ 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:** g $\otimes, \lambda P.(\delta W.\mathbf{A} \Rrightarrow P)$ transport U, while $\neg, \lambda P.P \wedge \neg P$ do not
- **Idea:** Model this in the types (rest of the talk/lecture)

A particularly interesting phenomenon is that of referent capture as the motor or anaphor resolution, which have already encountered Example 35.4 (Discourse Representation Theory) in the LBS lecture notes.

Variable/Referent Capture

▷ **Example C.2.3 (Anaphor Resolution Revisited).** Let us revisit Example 35.4 (Discourse Representation Theory) in the LBS lecture notes

A student[1] owns a book[2]. He[1] reads it[2] — anaphor resolution — simplify

$$\begin{array}{|l|}\hline X, Y \\ \hline \mathrm{student}(X) \\ \mathrm{book}(Y) \\ \hline \end{array} \otimes \begin{array}{|l|}\hline R, S \\ \hline \mathrm{read}(R \\ S) \\ \hline \end{array} \quad \begin{array}{|l|}\hline X, Y \\ \hline \mathrm{student}(X) \\ \mathrm{book}(Y) \\ \hline \end{array} \otimes \begin{array}{|l|}\hline R, S \\ \hline \mathrm{read}(R \\ S) \\ R = X \\ S = Y \\ \hline \end{array} \quad \begin{array}{|l|}\hline X, Y \\ \hline \mathrm{student}(X) \\ \mathrm{book}(Y) \\ \mathrm{read}(X \\ Y) \\ \hline \end{array}$$

▷ **Example C.2.4.** $(\lambda P.\begin{array}{|l|}\hline U \\ \hline (\neg P) \\ \hline \end{array}) \begin{array}{|l|}\hline \\ \hline r(U) \\ \hline \end{array}$ (functor has dynamic binding power)

▷ **Definition C.2.5.** We call this referent capture.

▷ **Note:** Referent capture

- ▷ is the motor of dynamicity
- ▷ is a structural property

▷ **Idea:** Code the information for referent capture in the type system.

▷ **Definition C.2.6.** The dynamic λ calculus extends $\Lambda^{\to}$ with discourse referent and a type system that encodes information about their dynamic status.

In ?? 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 ?? 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 C.2.7.** 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)

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 C.2.8.** $\downarrow$ closes off the dynamic binding potential and makes the referents classically bound
($\downarrow U^+, V^+ = U^\circ, V^\circ$)

▷ **Definition C.2.9.** The prioritized union operator combines two modes by letting $+$ overwrite $-$. ($U^+, V^- \uplus U^-, V^+ = U^+, V^+$)

▷ **Example C.2.10 (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 $\Gamma\#o \to \Delta\#o \to \downarrow(\Gamma \uplus \downarrow\Delta)\#o$

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.

C.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 C.3.1.** $\mathcal{DLC}$ 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 $\uplus\#o \to o \to o$
- ▷ $\vee$ gets type $\lambda FG.(\downarrow F \uplus \downarrow G)\#o \to o \to o$
- ▷ $\Rightarrow$ gets type $\lambda FG.\downarrow(F \uplus \downarrow G)\#o \to o \to o$

With this idea, we can re-interpret the DRT types from Example 36.13 (Setting Up Higher-Order Dynamics) in the LBS lecture notes.

A λ-Calculus for Mode Specifiers

▷ **Definition C.3.2.** New base type μ for modes; $\widetilde{\alpha}$ is α with ι, o replaced by μ.

▷ **Definition C.3.3.** Mode specifiers $\mathbb{A}, \mathbb{B}, \mathbb{C}$ are simply typed λ-terms built up from mode variables $F, G, F^1, \ldots$ and mode constants

- ▷ the empty mode $\emptyset$ of type μ
- ▷ the elementary modes 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 $\mu \to \mu \to \mu$.

▷ **Definition C.3.4.** Theory of mode equality specifies the meaning of mode constants

(e.g. $(U^+, V^-, W^- \uplus U^-, V^+) \to_\mu U^+, V^+, W^-$)

▷ **Definition C.3.5.** For each DLC type α, we define the type $\widetilde{\alpha}$ which is α with all base types replaced by μ.

Summary: DLC Grammar

▷ We summarize the setup in the following context-free grammar

$\alpha ::= \iota \mid o \mid \alpha_1 \to \alpha_2$	simple types
$\gamma ::= \mu \mid \gamma_1 \to \gamma_2$	mode types
$\mathbb{B} ::= \emptyset \mid U^+ \mid U^- \mid U^\circ \mid \mathbb{B}_1, \mathbb{B}_2 \mid \mathbb{B}_1 \uplus \mathbb{B}_2 \mid \downarrow\mathbb{B}$	basic modes
$\mathbb{M} ::= \mathbb{B} \mid \mathbb{M}_1 \mathbb{M}_2 \mid \lambda F_\gamma . \mathbb{M}$	modes (typed via mode types γ)
$\tau ::= \mathbb{M} \# \alpha$	DLC types
$\mathbf{M} ::= U \mid c \mid \mathbf{M}_1 \mathbf{M}_2 \mid \lambda X_\tau . \mathbf{M} \mid \delta U . \mathbf{M}$	DLC terms (typed via DLC types τ)

▷ But not all of these raw terms can be given a meaning ⇝ only use those that can be shown to be well-typed. (up next)

Type Inference for $\mathcal{DLC}$ (two dimensions)

▷ **Definition C.3.6.** The type inference system for $\mathcal{DLC}$ consists of the following rules:

$$\frac{c \in \Sigma_\alpha}{\mathcal{A} \vdash_\Sigma c \colon \alpha} \qquad \frac{\mathcal{A}(X) = F\#\alpha \quad \mathcal{A}(F) = \widetilde{\alpha}}{\mathcal{A} \vdash_\Sigma X \colon F\#\alpha} \qquad \frac{U \in \mathcal{R}_\alpha, \mathcal{A}(U) = \emptyset\#\alpha}{\mathcal{A} \vdash_\Sigma U \colon U^-\#\alpha}$$

$$\frac{\mathcal{A}, [X{:}F\#\beta], [F{:}\widetilde{\beta}] \vdash_\Sigma \mathbf{A} \colon \mathbb{A}\#\alpha}{\mathcal{A} \vdash_\Sigma \lambda X_{F\#\beta} . \mathbf{A} \colon \lambda F . \mathbb{A}\#\beta \to \alpha} \qquad \frac{\mathcal{A} \vdash_\Sigma \mathbf{A} \colon \mathbb{A}\#\beta \to \gamma \quad \mathcal{A} \vdash_\Sigma \mathbf{B} \colon \mathbb{B}\#\beta}{\mathcal{A} \vdash_\Sigma \mathbf{A}\ \mathbf{B} \colon \mathbb{A}(\mathbb{B})\#\gamma}$$

$$\frac{\mathcal{A} \vdash_\Sigma \mathbf{A} \colon \mathbb{A}\#\alpha \quad \mathcal{A} \vdash_\Sigma \mathbb{A} =_{\beta\eta\mu} \mathbb{B}}{\mathcal{A} \vdash_\Sigma \mathbf{A} \colon \mathbb{B}\#\alpha} \qquad \frac{\mathcal{A} \vdash_\Sigma \mathbf{A} \colon \lambda F . \mathbb{A}\#\alpha \quad \mathcal{A} \vdash_\Sigma \mathbb{A} \colon \mu}{\mathcal{A} \vdash_\Sigma \delta U_\beta . \mathbf{A} \colon \lambda F . (U^+ \uplus \mathbb{A})\#\alpha}$$

where $\mathcal{A}$ is a variable context mapping variables and referents to types

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 C.3.7 (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 $\mathcal{DLC}$.*

Linguistic Example

▷ **Example C.3.8.** *No man sleeps.*
Assume $U \in \mathcal{R}_\iota$ and $\mathrm{man}, \mathrm{sleeps} \in \mathcal{R}_{\lambda F.F\#\iota\to o}$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{\mathcal{A}\vdash_\Sigma \mathrm{man}(U) \colon U^-\#o}}{\mathcal{A}\vdash_\Sigma \delta U.\mathrm{man}(U) \colon U^+\#o} \quad \dfrac{\vdots}{\mathcal{A}\vdash_\Sigma \mathrm{sleeps}(U) \colon U^-\#o}}{\mathcal{A}\vdash_\Sigma \delta U.\mathrm{man}(U) \wedge \mathrm{sleeps}(U) \colon U^+ \uplus U^-\#o}}{\mathcal{A}\vdash_\Sigma \neg(\delta U.\mathrm{man}(U) \wedge \mathrm{sleeps}(U)) \colon \downarrow(U^+ \uplus U^-)\#o}}{\mathcal{A}\vdash_\Sigma \neg(\delta U.\mathrm{man}(U) \wedge \mathrm{sleeps}(U)) \colon U^\circ\#o}$$

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
$\mathbf{A}_\neg\ (\delta U.\mathrm{man}(U)) \to_\beta (\delta U.\mathrm{man}(U) \wedge \neg(\delta U.\mathrm{man}(U)))$

▷ 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 $\mathcal{A}\vdash_\Sigma \delta U.\mathrm{man}(U) \colon U^+\#o$, we derive

$$\mathcal{A}\vdash_\Sigma \mathbf{A}_\neg\ (\delta U.\mathrm{man}(U)) \colon U^+, U^\circ\#o$$

A Further Example: Generalized 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 $\mathrm{john} := \lambda P.(\delta U.U = j \otimes P(U))$
  and $\mathrm{mary} := \lambda P.(\delta V.V = m \otimes P(V))$
- we get $\mathrm{johnandmary} = \lambda P.(\delta U.U = j \otimes P(U) \otimes \delta V.V = m \otimes P(V))$
- combine this with *own a donkey*:

$\lambda X.(\delta W.\mathrm{donkey}(W) \otimes \mathrm{own}(W, X) \otimes \delta U.U = j \otimes \delta W.\mathrm{donkey}(W) \otimes \mathrm{own}(W, U) \otimes \delta V.V = m \otimes \delta W.\mathrm{donkey}(W) \otimes \mathrm{own}$

C.4 Modeling Higher-Order Dynamics

Discourse Variants $=_\delta$

- **Definition C.4.1.** We capture "referent renaming" in an equality judgment $=_\delta$.
- 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 $=_\eta$-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 $=_\beta$-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 C.4.2.** ρ renaming is induced by the following inference rule:

$$\frac{V \in \mathcal{R}_\beta \text{ fresh} \quad U_\beta \notin \mathcal{DP}(\mathbf{A})}{\mathbf{A} =_\rho \mathcal{C}^V_U(\mathbf{A})}$$

Where $\mathcal{C}^V_U(\mathbf{A})$ 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 mode specifier $\mathbf{A}$
- A mode specifier $\mathbf{A}$ 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 C.4.3.** 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 C.4.4.** If $U \notin \mathcal{DP}(\mathbf{A})$, then U is called independent of $\mathbf{A}$.

Some Examples for Dynamic Potential

- **Example C.4.5.**

Formula	Mode specifier	$\mathcal{DP}$
$\delta U.P$	U^+	$\{U\}$
$\lambda P.(\delta U.P)$	$\lambda F.(U^+ \uplus F)$	$\{U\}$
$\neg(\delta U.\mathrm{man}(U))$	U°	$\emptyset$
$\lambda P.\neg(\delta U.P)$	$\lambda F.\Downarrow(U^+), F$	$\{U\}$
$\lambda X.U$	$\lambda F.U^-$	$\{U\}$
$(\lambda X.X)\ U$	$(\lambda F.F)\ U^-$	$\{U\}$
$\lambda P.\mathrm{man}(U) \wedge P$	$\lambda F.(F \uplus U^-)$	$\{U\}$
$\lambda P.P$	$\lambda F.F$	$\emptyset$
$\lambda XY.Y$	$\lambda FG.G$	$\emptyset$
$(\lambda XY.Y)\ (\delta U.U)$	$\lambda G.G$	$\emptyset$
$\lambda P.P\ (\lambda Q.\neg(\delta U.Q))\ (\lambda R.(\delta U.R))$		$\{U\}$

Reductions

- $\beta\eta$-reduction: $\dfrac{}{(\lambda X.\mathbf{A})\ \mathbf{B} \to_\beta [\mathbf{B}/X](\mathbf{A})}$ and $\dfrac{X \notin \mathrm{free}(\mathbf{A})}{\lambda X.\mathbf{A}\ X \to_\eta \mathbf{A}}$
- **Definition C.4.6.** Dynamic Reduction: $\dfrac{\mathcal{A} \vdash_\Sigma \mathbf{A} : \mathbb{A}\#\alpha \quad U^+ \in \mathbf{Trans}(\mathbb{A})}{\mathbf{A}\ (\delta U.\mathbf{B}) \to_\tau (\delta U.\mathbf{A}\ \mathbf{B})}$
- **Example C.4.7.** 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.

C.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.
  - $\mathcal{D}_{\alpha\to\beta} \subseteq \mathcal{D}_\alpha \to \mathcal{D}_\beta$
- 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
  - $\mathcal{I}_\varphi(\mathbf{A}\ \mathbf{B}) = \mathcal{I}_\varphi(\mathbf{A})(\mathcal{I}_\varphi(\mathbf{B}))$
  - $\mathcal{I}_\varphi(\lambda X.\mathbf{B})(\mathsf{a}) = \mathcal{I}_{\varphi,[\mathsf{a}/X]}(\mathbf{B})$.

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: $\mathcal{D}_\alpha^\Gamma = \mathcal{B}_\Gamma \to \mathcal{D}_\alpha$, 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: $\mathcal{D}^\Phi_{(\alpha\to\beta)} = \bigcup_{\Psi\in\mathcal{D}^\sim_\alpha} \mathcal{D}^\Psi_\alpha \to \mathcal{D}^{\Phi(\Psi)}_\beta$

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)
- ▷ **Definition C.5.1.** 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!)
  - ▷ $\mathcal{I}_\varphi(\delta U.\mathbf{B}_{\mathbb{B}\#\beta}) = \Lambda s \in \mathcal{B}_{(\mathcal{I}_\varphi(\mathbb{B}_\mu)\uplus U^+)}.\mathcal{I}_\varphi(\mathbf{B})s|\mathcal{I}_\varphi(\mathbb{B}_\mu)$.
  - ▷ $\mathcal{I}_\varphi(\mathbf{B}\ \mathbf{C}) = \mathcal{I}_\varphi(\mathbf{B})(\mathcal{I}_\varphi(\mathbf{C}))$.
  - ▷ $\mathcal{I}_\varphi(\lambda X_\gamma.\mathbf{B}) = \Lambda^\Phi \mathsf{a} \in \mathcal{D}^\Phi_\gamma.\mathcal{I}_{\varphi,[\mathsf{a}/X]}(\mathbf{B})$
- ▷ Referent names crucial in dynamic objects
- ▷ **Well actually:** $\mathcal{I}_\varphi(\delta U.\mathcal{B}_{(\Lambda \overline{F_n}.\mathbb{B}_\mu\#\beta)}) = \Lambda \overline{\mathsf{a}_n}.\Lambda s \in \mathcal{B}_{(\mathcal{I}_\varphi(\mathbb{B}_\mu)\uplus U^+)}.\mathcal{I}_\varphi(\mathbf{B})s|\mathcal{I}_\varphi(\mathcal{B}_\mu)$.

FAU Michael Kohlhase: LBS 369 2025-02-06

Metatheoretic Results

- ▷ **Theorem C.5.2 (Normalization).** *$\beta\eta\tau$-Reduction is terminating and confluent (modulo $\alpha\rho\delta$).*
- ▷ **Theorem C.5.3 (Substitution is type-preserving).** *If $X \notin \mathbf{dom}(\mathcal{A})$, then $\mathcal{A}, [X{:}F\#\beta] \vdash_\Sigma \mathbf{A} : \mathbb{A}\#\alpha$ and $\mathcal{A} \vdash_\Sigma \mathbf{B} : \mathbb{B}\#\beta$ imply*

$$\mathcal{A} \vdash_\Sigma [\mathbf{B}/X](\mathbf{A}) : [\mathbf{B}/F](\mathbb{A})\#\alpha$$

- ▷ **Theorem C.5.4 (Subject Reduction).** *If $\mathcal{A} \vdash_\Sigma \mathbf{A} : \mathbb{A}\#\alpha$ and $\mathcal{A} \vdash_\Sigma \mathbf{A} =_{\beta\eta\tau} \mathbf{B}$, then $\mathcal{A} \vdash_\Sigma \mathbf{B} : \mathbb{A}\#\alpha$.*
- ▷ **Theorem C.5.5 (Soundness of Reduction).** *If $\mathcal{A} \vdash_\Sigma \mathbf{A} =_{\alpha\beta\delta\eta\tau\rho} \mathbf{B}$, then $\mathcal{I}_\varphi(\mathbf{A}) = \mathcal{I}_\varphi(\mathbf{B})$.*
- ▷ 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.)

FAU Michael Kohlhase: LBS 370 2025-02-06

C.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))$\to$ 2 ((lambda (F) (**let** ((U 0)) (F 1)))(lambda (X) (+ X U))$\to$ 1
- Ever wanted to determine the \$PRINTERenvironment 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 C.6.1 (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 C.6.2 (Java).** If you want to keep your \$EDITOR variable private (pirated?) only allow applets of type $\mathbb{A}\#\alpha$, where `$EDITOR` $\notin\mathcal{DP}(\mathbb{A})$.
- It is going to be a lot of fun!

Appendix D

Model Existence and Completeness for Modal Logic

Abstract Consistency for ML^0

▷ **Definition D.0.1.** If Φ is a set of propositions, then

$$\Box^{-}(\Phi){:=}\{\mathbf{A}\,|\,\Box\mathbf{A}\in\Phi\}$$

▷ **Definition D.0.2.** A collection ∇ 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 $P\in\mathcal{V}_0$

$\vdots$

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

$\nabla_\Box$) $\Diamond\mathbf{A}\in\Phi$ implies $\Box^{-}(\Phi)*\mathbf{A}\in\nabla$

FAU Michael Kohlhase: LBS 374 2025-02-06

∇-Hintikka Set

▷ **Definition D.0.3.** If ∇ is an 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 D.0.4 (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:

1. chose an enumeration $\mathbf{A}_1,\mathbf{A}_2\ldots$ of $\mathit{wff}_0(\mathcal{V}_0)$
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 $H_n*\mathbf{A}_n\in\nabla$
3. All $H_i\in\nabla$, so choose $\mathcal{H}:=\bigcup_{i\in\mathbb{N}}H_i$

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.
5. $\mathcal{H} \in \nabla$ since ∇ compact.
6. let $\mathcal{H}{*}\mathbf{B} \in \nabla$, then there is a $j \in \mathbb{N}$ with $\mathbf{B} = \mathbf{A}_j$
7. $\mathbf{B} \in H_{j+1} \subseteq \mathcal{H}$, so $\mathcal{H}$ ∇-maximal.

Canonical ∇-Model

▷ **Definition D.0.5.** 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

- ▷ $\mathcal{W}_\nabla = \{\mathcal{H} \,|\, \mathcal{H} \in \nabla\text{maximal}\}$
- ▷ $\mathcal{R}_\nabla(v, w)$ iff $\Box^-(v) \subseteq w$
- ▷ $\varphi(P, w) = \mathsf{T}$ iff $P \in w$

▷ **Lemma D.0.6.** *If $w \in \mathcal{W}_\nabla$ and $\Diamond\mathbf{A} \in w$, then there is a $w' \in \mathcal{W}_\nabla$ with $\mathcal{R}_\nabla(w, w')$ and $\mathbf{A} \in w'$.*

▷ *Proof:* Let $\Diamond\mathbf{A} \in w$

1. thus $\Box^-(w){*}\mathbf{A} \in \nabla$
2. by the extension theorem there is a $w' \in \mathcal{W}_\nabla$ with $\Box^-(w){*}\mathbf{A} \subseteq w'$
3. so $\Box^-(w) \subseteq w'$ and thus $\mathcal{R}_\nabla(w, w')$.
4. on the other and we have $\mathbf{A} \in w'$.

Model existence for ML^0

▷ **Lemma D.0.7.** *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 $\mathbf{A}$

1. If $\mathbf{A}$ is a variable *then we get the assertion by the definition of φ_∇.*
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* $\mathcal{I}^w_{\varphi_\nabla}(\mathbf{A}) = \mathsf{T}$.
3. $\mathbf{A} = \mathbf{B} \wedge \mathbf{C}$ *analog*
4. $\mathbf{A} = \Box\mathbf{B}$
   - 4.1. Let $\mathbf{A} \in w$ and $w\mathcal{R}_\nabla w'$
   - 4.2. thus $\Box^-(w) \subseteq w'$ and thus $\mathbf{B} \in w'$
   - 4.3. so (IH) $\mathcal{I}^{w'}_{\varphi_\nabla}(\mathbf{B}) = \mathsf{T}$ for any such w'.
   - 4.4. and finally $\mathcal{I}^w_{\varphi_\nabla}(\mathbf{A}) = \mathsf{T}$
5. $\mathbf{A} = \Diamond\mathbf{B}$
   - 5.1. Let $\mathbf{A} \notin w$
   - 5.2. so $\neg\mathbf{A} = \Diamond\neg\mathbf{B} \notin w$
   - 5.3. and thus $\neg\mathbf{B} \in w'$ for some $w\mathcal{R}_\nabla w'$ by (Lemma1)
   - 5.4. so $\mathbf{B} \in w'$ and thus $\mathcal{I}^{w'}_{\varphi_\nabla}(\mathbf{B}) = \mathsf{T}$ by IH and finally $\mathcal{I}^w_{\varphi_\nabla}(\mathbf{A}) = \mathsf{T}$.

▷ **Theorem D.0.8 (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:

▷ 1. there is a ∇-Hintikka set $\mathcal{H} = w$ with $w \in \mathcal{W}_\nabla$ and $\Phi \subseteq \mathcal{H}$.
2. by Lemma 2 we have $\mathcal{I}_\varphi^w(\mathbf{A}) = \mathsf{T}$ for all $\mathbf{A} \in \Phi$.

Completeness

▷ **Theorem D.0.9.** *$\mathbb{K}$-consistency is an abstract consistency class for ML^0*

▷ *Proof:* Let $\Diamond\mathbf{A} \in \Phi$

1. To show: $\Box^-(\Phi)*\mathbf{A}$ is $\mathbb{K}$-consistent if Φ is $\mathbb{K}$-consistent
2. converse: Φ is $\mathbb{K}$-inconsistent if $\Box^-(\Phi)*\mathbf{A}$ $\mathbb{K}$-inconsistent.
3. There is a finite subset $\Psi \subseteq \Box^-(\Phi)$ with $\Psi\vdash_{\mathbb{K}}(\neg\mathbf{A})$
4. $(\Box\Psi)\vdash_{\mathbb{K}}(\Box\neg\mathbf{A})$ (distributivity of $\Box$)
5. $\Phi\vdash_{\mathbb{K}}(\Box\neg\mathbf{A}) = \neg(\Diamond\mathbf{A})$ since $\Box\Psi \subseteq \Phi$
6. thus Φ is $\mathbf{K}$-inconsistent.

▷ **Corollary D.0.10.** *$\mathbb{K}$ is complete wrt. Kripke models*

Further Completeness Theorems

▷ **Theorem D.0.11.** *$\mathbb{T}$-consistency is an abstract consistency class for ML^0 and $\mathcal{R}_{\mathbb{T}}$ is reflexive.*

▷ *Proof:* Let $\mathbf{A} \in \Box^-(w)$

1. then $\Box\mathbf{A} \in w$ by definition
2. with $\mathbb{T}$ $(\Box\mathbf{A} \Rightarrow \mathbf{A})$ and Modus Ponens we have $\mathbf{A} \in w$.
3. Thus $\Box^-(w) \subseteq w$ and $w\mathcal{R}_{\mathbb{T}}w$ for all $w \in \mathcal{W}_{\mathbb{T}}$.

▷ **Theorem D.0.12.** *$\mathbb{S}4$-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 $\Box\mathbf{A} \in w$.

▷ 1. by $\mathbb{S}4$ $(\Box\mathbf{A} \Rightarrow \Box\Box\mathbf{A})$ and Modus Ponens we have $\Box\Box\mathbf{A} \in w_1$.
2. and thus $\Box\mathbf{A} \in w_2 = \Box^-(w_1)$ and $\mathbf{A} \in w_3 = \Box^-(w_2)$.
3. Thus $\Box^-(w_1) \subseteq w_3$ and $w_1\mathcal{R}_{\mathbb{S}4}w_3$.

▷ **Corollary D.0.13.** *$\mathbb{T}$ ($\mathbb{S}4$) is complete wrt. reflexive (reflexive transitive) Kripke-models*