Computational Logic
320441 — Fall 2009

MICHAEL KOHLHASE

School of Engineering & Science
Jacobs University, Bremen Germany
m.kohlhase@jacobs-university.de
office: Room 62, Research 1, phone: x3140

This document contains the course notes for the graduate course "Computational Logic" held at Jacobs University, Bremen in the Fall Semsesters 2005, 2007, and 2009. The document mixes the slides presented in class with comments of the instructor to give students a more complete background reference.

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.

This document is also an experiment in knowledge representation. Under the hood, it uses the STEX package, a TEX/LATEX extension for semantic markup. Eventually, this will enable to export the contents into eLearning platforms like Connexions (see http://cnx.rice.edu) or ActiveMath (see http://www.activemath.org).

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

Acknowledgments: The following students have submitted corrections and suggestions to this and earlier versions of the notes: Rares Ambrus, Florian Rabe, Deyan Ginev.

Contents

Outline

Contents

1 Administrativa

We will now go through the ground rules for the course. This is a kind of a social contract between the instructor and the students. Both have to keep their side of the deal to make learning and becoming Computer Scientists as efficient and painless as possible.

Textbooks, Handouts and Information, Forum

- No required textbook, but course notes, posted slides
- Course notes will be posted at `http://kwarc.info/teaching/GenCS1.html`
- Everything will be posted on Panta Rhei (Notes+assignments+course forum)
  - announcements, contact information, course schedule and calendar
  - discussion among your fellow students(careful, I will occasionally check for academic integrity!)
  - `http://panta-rhei.kwarc.info` (use your precourse login and pwd)
  - if there are problems send e-mail to `c.mueller@jacobs-university.de`
- The EECS Seminar (If you want to take a peek into EECS research)

  see details at `http://www.eecs.jacobs-university.de/seminar/`

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

The EECS seminar: The EECS seminar is the colloquium of the EECS&Logistics group at Jacobs University. The seminar features talks by graduate students, Jacobs faculty and external research collaborators. Even though much of the material covered in the talks will be beyond understanding for most first-year students, the speakers usually give a general introduction which shows students which research directions are currently being discussed. For students that want to get involved into research early this is a valuable source of orientation.

Homework assignments

- ▷ Goal: Reinforce and apply what is taught in class.
- ▷ homeworks will be small individual problem/programming/proof assignments (but take time to solve)
- ▷ admin: To keep things running smoothly
  - ▷ Homeworks will be posted on Panta Rhei
  - ▷ Homeworks are handed in electronically in `grader` (plain text, Postscript, PDF,...)
  - ▷ go to the recitations, discuss with your TA (they are there for you!)
- ▷ Homework discipline
  - ▷ start early! (many assignments need more than one evening's work)
  - ▷ Don't start by sitting at a blank screen
  - ▷ Humans will be trying to understand the text/code/math when grading it.

Homework assignments are a central part of the course, they allow you to review the concepts covered in class, and practice using them.

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

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

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

The Code of Academic Integrity

- ▷ Jacobs has a "Code of Academic Integrity"
  - ▹ this is a document passed by the faculty (our law of the university)
  - ▹ you have signed it last week (we take this seriously)
- ▷ It mandates good behavior and penalizes bad from both faculty and students
  - ▹ honest academic behavior (we don't cheat)
  - ▹ respect and protect the intellectual property of others (no plagiarism)
  - ▹ treat all Jacobs members equally (no favoritism)
- ▷ this is to protect you and build an atmosphere of mutual respect
  - ▹ academic societies thrive on reputation and respect as primary currency
- ▷ The Reasonable Person Principle (one lubricant of academia)
  - ▹ we treat each other as reasonable persons
  - ▹ the other's requests and needs are reasonable until proven otherwise

 5 JACOBS UNIVERSITY

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

One special case of academic rules that affects students is the question of cheating, which we will cover next.

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

Cheating [adapted from CMU:15-211 (P. Lee, 2003)]

- There is no need to cheat in this course!! (hard work will do)
- cheating prevents you from learning (you are cutting your own flesh)
- if you are in trouble, come and talk to me (I am here to help you)
- We expect you to know what is useful collaboration and what is cheating
  - you will be required to hand in your own original code/text/math for all assignments
  - you may discuss your homeworks with others, but if doing so impairs your ability to write truly original code/text/math, you will be cheating
  - copying from peers, books or the Internet is plagiarism unless properly attributed (even if you change most of the actual words)
  - more on this as the semester goes on ...
- ⚠ There are data mining tools that monitor the originality of text/code. ⚠

 6 JACOBS UNIVERSITY

We are fully aware that the border between cheating and useful and legitimate collaboration is difficult to find and will depend on the special case. Therefore it is very difficult to put this into firm rules. We expect you to develop a firm intuition about behavior with integrity over the course of stay at Jacobs.

Software/Hardware tools

- You will need computer access for this course (come see me if you do not have a computer of your own)
- we recommend the use of standard software tools
  - the `emacs` and `vi` text editor (powerful, flexible, available, free)
  - `UNIX` (`linux`, `MacOSX`, `cygwin`) (prevalent in CS)
  - `FireFox` (just a better browser)
- learn how to touch-type NOW (reap the benefits earlier, not later)

 7 JACOBS UNIVERSITY

Touch-typing: You should not underestimate the amount of time you will spend typing during your studies. Even if you consider yourself fluent in two-finger typing, touch-typing will give you a factor two in speed. This ability will save you at least half an hour per day, once you master it. Which can make a crucial difference in your success.

Touch-typing is very easy to learn, if you practice about an hour a day for a week, you will re-gain your two-finger speed and from then on start saving time. There are various free typing tutors on the network. At `http://typingsoft.com/all_typing_tutors.htm` you can find about programs, most for windows, some for linux. I would probably try `Ktouch` or `TuxType`

Darko Pesikan recommends the `TypingMaster` program. You can download a demo version from `http://www.typingmaster.com/index.asp?go=tutordemo`

You can find more information by googling something like "learn to touch-type". (goto `http://www.google.com` and type these search terms).

Next we come to a special project that is going on in parallel to teaching the course. I am using the coures materials as a research object as well. This gives you an additional resource, but may affect the shape of the coures materials (which now server double purpose). Of course I can use all the help on the research project I can get.

Experiment: E-Learning with *OMDoc*/ActiveMath/`SWiM`

- My research area: deep representation formats for (mathematical) knowledge
- Application: E-learning systems (represent knowledge to transport it)
- Experiment: Start with this course (Drink my own medicine)
  - Re-Represent the slide materials in *OMDoc* (Open Math Documents)
  - Feed it into the ActiveMath system (`http://www.activemath.org`)
  - Try it on you all (to get feedback from you)
- Tasks (Unfortunately, I cannot pay you for this; maybe later)
  - help me complete the material on the slides (what is missing/would help?)
  - I need to remember "what I say", examples on the board. (take notes)
- Benefits for you (so why should you help?)
  - you will be mentioned in the acknowledgements (for all that is worth)
  - you will help build better course materials (think of next-year's freshmen)

2 First-Order Logic

2.1 First-Order Logic

History of Ideas (abbreviated): Propositional Logic

- General Logic ([ancient Greece, e.g. Aristoteles])
  - \+ conceptual separation of syntax and semantics
  - \+ system of inference rules ("Syllogisms")
  - – no formal language, no formal semantics
- Propositional Logic [Boole ~ 1850]
  - \+ functional structure of formal language (propositions + connectives)
  - \+ mathematical semantics ($\leadsto$ Boolean Algebra)
  - – abstraction from internal structure of propositions

History of Ideas (continued): Predicate Logic

▷ Begriffsschrift [Frege 1879]

+ functional structure of formal language(terms, atomic formulae, connectives, quantifiers)

– weird graphical syntax, no mathematical semantics

– paradoxes e.g. Russell's Paradox [R. 1901] (the set of sets that do not contain themselves)

▷ modern form of predicate logic [Peano $\sim$ 1889]

+ modern notation for predicate logic $(\vee, \wedge, \Rightarrow, \forall, \exists)$

 10

History of Ideas (continued): First-Order Predicate Logic

▷ Types ([Russell 1908])

– restriction to well-types expression

+ paradoxes cannot be written in the system

+ Principia Mathematica ([Whitehead, Russell 1910])

▷ Identification of first-order Logic ([Skolem, Herbrand, Gödel $\sim$ 1920 – '30])

– quantification only over individual variables (cannot write down induction principle)

+ correct, complete calculi, semi-decidable

+ set-theoretic semantics ([Tarski 1936])

 11

History of Ideas (continued): Foundations of Mathematics

▷ Hilbert's Programme: find logical system and calculus, ([Hilbert $\sim$ 1930])

- ▷ that formalizes all of mathematics
- ▷ that admits correct and complete calculi
- ▷ whose consistence is provable in the system itself

▷ Hilbert's Programm is impossible! ([Gödel 1931])

Let $\mathcal{L}$ be a logical system that formalizes arithmetics $(\mathbb{N}, +, *)$,

- ▷ then $\mathcal{L}$ is incomplete
- ▷ then the consistence of $\mathcal{L}$ cannot be proven in $\mathcal{L}$.

 12

History of Ideas (continued): λ-calculus, set theory

- simply typed λ-calculus ([Church 1940])
  - + simplifies Russel's types, λ-operator for functions
  - + comprehension as β-equality (can be mechanized)
  - + simple type-driven semantics (standard semantics $\rightsquigarrow$ incompleteness)
- Axiomatic set theory
  - +– type-less representation (all objects are sets)
  - + first-order logic with axioms
  - + restricted set comprehension (no set of sets)
  - – functions and relations are derived objects

 13 JACOBS UNIVERSITY

First-Order Predicate Logic (PL1)

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

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

 14 JACOBS UNIVERSITY

PL^1 Syntax (Signature and Variables)

- PL1 talks about two kinds of objects: (so we have two kinds of symbols)
  - o for truth values (like in PL0)
  - Type ι for individuals (numbers, foxes, Pokémon,...)
- Definition 2.1: A first-order signature consists of (all disjoint; $k \in \mathbb{N}$)
  - connectives: $\Sigma^o = \{T, F, \neg, \vee, \wedge, \Rightarrow, \Leftrightarrow, \ldots\}$ (functions on truth values)
  - function constants: $\Sigma^f_k = \{f, g, h, \ldots\}$ (functions on individuals)
  - predicate constants: $\Sigma^p_k = \{p, q, r \ldots\}$ (relations among inds.)
  - (Skolem constants: $\Sigma^{sk}_k = \{f^k_1, f^k_2, \ldots\}$) (witness constants; countably ∞)
  - We take the signature Σ to all of these together: $\Sigma := \Sigma^o \cup \Sigma^f \cup \Sigma^p \cup \Sigma^{sk}$, where $\Sigma^* := \bigcup_{k \in \mathbb{N}} \Sigma^*_k$.
- Definition 2.2: For first-order Logic (PL^1), we assue a set of
  - individual variables: $\mathcal{V}_\iota = \{X_\iota, Y_\iota, Z, X^1_\iota, X^2, \ldots\}$ (countably ∞)

PL^1 Syntax (Formulae)

- Definition 2.3: terms: $\mathbf{A}_\iota \in \mathit{wff}_\iota(\Sigma_\iota)$ (denote individuals: type ι)
  - $\mathcal{V}_\iota \subseteq \mathit{wff}_\iota(\Sigma_\iota)$,
  - if $f \in \Sigma^f_k$ and $\mathbf{A}^i \in \mathit{wff}_\iota(\Sigma_\iota)$ for $i \leq k$, then $f(\mathbf{A}^1, \ldots, \mathbf{A}^k) \in \mathit{wff}_\iota(\Sigma_\iota)$.
- Definition 2.4: propositions: $\mathbf{A}_o \in \mathit{wff}_o(\Sigma)$ (denote truth values: type o)
  - if $p \in \Sigma^p_k$ and $\mathbf{A}^i \in \mathit{wff}_\iota(\Sigma_\iota)$ for $i \leq k$, then $p(\mathbf{A}^1, \ldots, \mathbf{A}^k) \in \mathit{wff}_o(\Sigma)$
  - If $\mathbf{A}, \mathbf{B} \in \mathit{wff}_o(\Sigma)$, then $T, (\mathbf{A} \wedge \mathbf{B}), \neg\mathbf{A}, (\forall X_\iota.\mathbf{A}) \in \mathit{wff}_o(\Sigma)$
- Definition 2.5: We define the connectives $F, \vee, \Rightarrow, \Leftrightarrow$ via the abbreviations $\mathbf{A} \vee \mathbf{B} := \neg(\neg\mathbf{A} \wedge \neg\mathbf{B})$, $\mathbf{A} \Rightarrow \mathbf{B} := \neg\mathbf{A} \vee \mathbf{B}$, $\mathbf{A} \Leftrightarrow \mathbf{B} := (\mathbf{A} \Rightarrow \mathbf{B}) \wedge (\mathbf{B} \Rightarrow \mathbf{A})$, and $F := \neg\mathbf{A} \wedge \mathbf{A}$. We will use them like the primary connectives $\wedge$ and $\neg$
- Definition 2.6: We use $\exists X_\iota.\mathbf{A}$ as an abbreviation for $\neg(\forall X_\iota.\neg\mathbf{A})$(existential quantifier)
- Definition 2.7: Formulae without connectives or quantifiers are called atomic else complex.

Semantics (PL1)

- ▷ Universe $\mathcal{D}_o = \{\mathsf{T}, \mathsf{F}\}$ of truth values
- ▷ Arbitrary universe $\mathcal{D}_\iota \neq \emptyset$ of individuals
- ▷ interpretation $\mathcal{I}$ assigns values to constants, e.g.
  - ▷ $\mathcal{I}(\neg)\colon \mathcal{D}_o \to \mathcal{D}_o$ with $\mathsf{T} \mapsto \mathsf{F}$, $\mathsf{F} \mapsto \mathsf{T}$, and $\mathcal{I}(\wedge) = \ldots$ (as in PL0)
  - ▷ $\mathcal{I}\colon \Sigma_k^f \to \mathcal{F}({\mathcal{D}_\iota}^k; \mathcal{D}_\iota)$ (interpret function symbols as arbitrary functions)
  - ▷ $\mathcal{I}\colon \Sigma_k^p \to \mathcal{P}({\mathcal{D}_\iota}^k)$ (interpret predicates as arbitrary relations)
- ▷ Definition 2.8: variable assignment $\varphi\colon \mathcal{V}_\iota \to \mathcal{D}_\iota$ assigns values to variables.

Semantics (PL1 continued)

- ▷ The value function $\mathcal{I}_\varphi$ recursively defined
  - ▷ $\mathcal{I}_\varphi\colon \mathit{wff}_\iota(\Sigma_\iota) \to \mathcal{D}_\iota$ assigns values to terms.
    - ▷ $\mathcal{I}_\varphi(X_\iota) := \varphi(X_\iota)$ and
    - ▷ $\mathcal{I}_\varphi(f(\mathbf{A}_1, \ldots, \mathbf{A}_k)) := \mathcal{I}(f)(\mathcal{I}_\varphi(\mathbf{A}_1), \ldots, \mathcal{I}_\varphi(\mathbf{A}_k))$
  - ▷ $\mathcal{I}_\varphi\colon \mathit{wff}_o(\Sigma) \to \mathcal{D}_o$ assigns values to formulae
    - ▷ e.g. $\mathcal{I}_\varphi(\neg\mathbf{A}) = \mathcal{I}(\neg)(\mathcal{I}_\varphi(\mathbf{A}))$ (just as in Pl0)
    - ▷ $\mathcal{I}_\varphi(p(\mathbf{A}^1, \ldots, \mathbf{A}^k)) := \mathsf{T}$, iff $\langle \mathcal{I}_\varphi(\mathbf{A}^1), \ldots, \mathcal{I}_\varphi(\mathbf{A}^k) \rangle \in \mathcal{I}(p)$
    - ▷ $\mathcal{I}_\varphi(\forall X_\iota.\mathbf{A}) := \mathsf{T}$, iff $\mathcal{I}_{\varphi,[a/X]}(\mathbf{A}) = \mathsf{T}$ for all $a \in \mathcal{D}_\iota$.
- ▷ Model: $\mathcal{M} = \langle \mathcal{D}_\iota, \mathcal{I} \rangle$ varies in $\mathcal{D}_\iota$ and $\mathcal{I}$.

Free and Bound Variables

- ▷ Definition 2.9: We call an occurrence of a variable X bound in a formula $\mathbf{A}$, iff it is in a subterm $\forall X.\mathbf{B}$ of $\mathbf{A}$. We call a variable occurrence free otherwise.

  For a formula $\mathbf{A}$, we will use $\mathbf{BVar}(\mathbf{A})$ (and $\mathbf{free}(\mathbf{A})$) for the set of bound (free) variables of $\mathbf{A}$, i.e. variables that have a free/bound occurrence in $\mathbf{A}$.
- ▷ Definition 2.10: We can inductively define the set $\mathbf{free}(\mathbf{A})$ of free variables of a formula $\mathbf{A}$:

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

- ▷ Definition 2.11: We call a formula $\mathbf{A}$ closed or ground, iff $\mathbf{free}(\mathbf{A}) = \emptyset$. We call a closed proposition a sentence, and denote the set of all ground terms with $\mathit{cwff}_\iota(\Sigma_\iota)$ and the set of sentences with $\mathit{cwff}_o(\Sigma_\iota)$.

2.2 First-Order Substitutions

Substitutions

- Intuition: If $\mathbf{B}$ is a term and X is a variable, then we denote the result of systematically replacing all variables in a term $\mathbf{A}$ by $\mathbf{B}$ with $[\mathbf{B}/X]\mathbf{A}$.
- Problem: What about $[Z/Y], [Y/X]X$, is that Y or Z?
- Folklore: $[Z/Y], [Y/X]X = Y$, but $[Z/Y][Y/X](X) = Z$ of course. (Parallel application)
- Definition 2.12: We call $\sigma\colon wff_\iota(\Sigma_\iota) \to wff_\iota(\Sigma_\iota)$ a substitution, iff $\sigma f(\mathbf{A}_1, \ldots, \mathbf{A}_n) = f(\sigma\mathbf{A}_1, \ldots, \sigma\mathbf{A}_n)$ and the support $\mathbf{supp}(\sigma) := (\{X \mid \sigma X \neq X\})$ of σ is finite.
- Notation 2.13: Note that a substitution σ is determined by its values on variables alone, thus we can write σ as $\sigma|_{\mathcal{V}_\iota} = (\{[\sigma X/X] \mid X \in \mathbf{supp}(\sigma)\})$.
- Example 2.14: $[a/x], [f(b)/y], [a/z]$ instantiates $g(x, y, h(z))$ to $g(a, f(b), h(a))$.
- Definition 2.15: We call $\mathbf{intro}(\sigma) := \bigcup_{X \in \mathbf{supp}(\sigma)} \mathbf{free}(\sigma X)$ the set of variables introduced by σ.

 20 JACOBS UNIVERSITY

Substitution Extension

- Notation 2.16: (Substitution Extension)

  Let σ be a substitution, then we denote with $\sigma, [\mathbf{A}/X]$ the function $(\{\langle Y, \mathbf{A}\rangle \in \sigma \mid Y \neq X\}) \cup \{\langle X, \mathbf{A}\rangle\}$.
  ($\sigma, [\mathbf{A}/X]$ coincides with σ off X, and gives the result $\mathbf{A}$ there.)
- Note: If σ is a substitution, then $\sigma, [\mathbf{A}/X]$ is also a substitution.
- Definition 2.17: If σ is a substitution, then we call $\sigma, [\mathbf{A}/X]$ the extension of σ by $[\mathbf{A}/X]$.

 21 JACOBS UNIVERSITY

Substitutions on Propositions

- ▷ Problem: We want to extend substitutions to propositions, in particular to quantified formulae: What is $\sigma(\forall X.\mathbf{A})$?
- ▷ Idea: σ should not instantiate bound variables:
- ▷ Definition 2.18: $\sigma(\forall X.\mathbf{A}) := (\forall X.\sigma_{-X}\mathbf{A})$, where $\sigma_{-X} := \sigma, [X/X]$
- ▷ Problem: This can lead to variable capture: $[f(X)/Y](\forall X.p(X, Y))$ would evaluate to $\forall X.p(X, f(X))$, where the second occurrence of X is bound after instantiation, whereas it was free before.
- ▷ Definition 2.19: Let $\mathbf{B} \in wff_\iota(\Sigma_\iota)$ and $\mathbf{A} \in wff_o(\Sigma)$, then we call $\mathbf{B}$ substitutible for X in $\mathbf{A}$, iff $\mathbf{A}$ has no occurrence of X in a subterm $\forall Y.\mathbf{C}$ with $Y \in \mathbf{free}(\mathbf{B})$.
- ▷ Solution: forbid substitution $[\mathbf{B}/X]\mathbf{A}$, when $\mathbf{B}$ is not substitutible for X in $\mathbf{A}$
- ▷ Better Solution: rename away the bound variable X in $\forall X.p(X, Y)$ before applying the substitution. (see alphabetic renaming later.)

 22

Substitution Value Lemma for Terms

- ▷ Lemma 2.20: *Let $\mathbf{A}$ and $\mathbf{B}$ be terms, then $\mathcal{I}_\varphi([\mathbf{B}/X]\mathbf{A}) = \mathcal{I}_\psi(\mathbf{A})$, where $\psi = \varphi, [\mathcal{I}_\varphi(\mathbf{B})/X]$*
- ▷ Proof: by induction on the depth of $\mathbf{A}$:

P.1.1 depth=0:

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

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

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

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

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

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

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

by inductive hypothesis

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

□

 23

Substitution Value Lemma for Propositions

▷ Lemma 2.21: *Let* $\mathbf{B} \in wff_\iota(\Sigma_\iota)$ *be substitutible for* X *in* $\mathbf{A} \in wff_o(\Sigma)$*, then* $\mathcal{I}_\varphi([\mathbf{B}/X]\mathbf{A}) = \mathcal{I}_\psi(\mathbf{A})$*, where* $\psi = \varphi, [\mathcal{I}_\varphi(\mathbf{B})/X]$

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

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

P.1.2 $n > 0$ and $\mathbf{A} = \forall X.\mathbf{C}$: then $\mathcal{I}_\psi(\mathbf{A}) = \mathcal{I}_\psi(\forall X.\mathbf{C}) = \mathsf{T}$, iff $\mathcal{I}_{\psi,[a/X]}(\mathbf{C}) = \mathcal{I}_{\varphi,[a/X]}(\mathbf{C}) = \mathsf{T}$, for all $a \in \mathcal{D}_\iota$, which is the case, iff $\mathcal{I}_\varphi(\forall X.\mathbf{C}) = \mathcal{I}_\varphi([\mathbf{B}/X]\mathbf{A}) = \mathsf{T}$. □

P.1.3 $n > 0$ and $\mathbf{A} = \forall Y.\mathbf{C}$ where $X \neq Y$: then $\mathcal{I}_\psi(\mathbf{A}) = \mathcal{I}_\psi(\forall Y.\mathbf{C}) = \mathsf{T}$, iff $\mathcal{I}_{\psi,[a/Y]}(\mathbf{C}) = \mathcal{I}_{\varphi,[a/Y]}([\mathbf{B}/X]\mathbf{C}) = \mathsf{T}$, by inductive hypothesis. So $\mathcal{I}_\psi(\mathbf{A}) = \mathcal{I}_\varphi(\forall Y.[\mathbf{B}/X]\mathbf{C}) = \mathcal{I}_\varphi([\mathbf{B}/X](\forall Y.\mathbf{C})) = \mathcal{I}_\varphi([\mathbf{B}/X]\mathbf{A})$ □

P.1.4 $n > 0$ and $\mathbf{A} = \neg\mathbf{B}$ or $\mathbf{A} = \mathbf{C} \circ \mathbf{D}$:

P.1.4.1 Here we argue like in the inductive case of the term lemma □

□

  JACOBS UNIVERSITY

2.3 Alpha-Renaming for First-Order Logic

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

Alphabetic Renaming

▷ Lemma 2.22: *Bound variables can be renamed: If* Y *is substitutible for* X *in* $\mathbf{A}$*, then* $\mathcal{I}_\varphi(\forall X.\mathbf{A}) = \mathcal{I}_\varphi(\forall Y.[Y/X]\mathbf{A})$

▷ Proof: by the definitions:

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

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

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

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

▷ Definition 2.23: We call two formulae $\mathbf{A}$ and $\mathbf{B}$ alphabetical variants (or α-equal; write $\mathbf{A} =_\alpha \mathbf{B}$), iff $\mathbf{A} = \forall X.\mathbf{C}$ and $\mathbf{B} = \forall Y.[Y/X]\mathbf{C}$ for some variables X and Y.

  JACOBS UNIVERSITY

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

Avoiding Variable Capture by Built-in α-renaming

- Idea: Given alphabetic renaming, we will consider alphabetical variants as identical
- So: Bound variable names in formulae are just as a representational device (we rename bound variables wherever necessary)
- Formally: Take $cwff_o(\Sigma_\iota)$ (new) to be the quotient set of $cwff_o(\Sigma_\iota)$ (old) modulo $=_\alpha$. (formulae as syntactic representatives of equivalence classes)
- Definition 2.24: (Capture-Avoiding Substitution Application)

  Let σ be a substitution, $\mathbf{A}$ a formula, and $\mathbf{A}'$ an alphabetical variant of $\mathbf{A}$, such that $\mathbf{intro}(\sigma) \cap \mathbf{BVar}(\mathbf{A}) = \emptyset$. Then $[\mathbf{A}]_{=_\alpha} = [\mathbf{A}']_{=_\alpha}$ and we can define $\sigma[\mathbf{A}]_{=_\alpha} := [\sigma(\mathbf{A}')]_{=_\alpha}$.
- Notation 2.25: After we have understood the quotient construction, we will neglect making it explicit and write formulae with and substitutions with the understanding that they act on quotients.

2.4 Recap: General Properties of Logics and Calculi

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

Logical Systems

- Definition 2.26: logical system is a triple $\mathcal{S} := \langle \mathcal{L}, \mathcal{K}, \models \rangle$, where $\mathcal{L}$ is a formal language, $\mathcal{K}$ is a set and $\models \subseteq \mathcal{K} \times \mathcal{L}$. Member of $\mathcal{L}$ are called formulae of $\mathcal{S}$, members of $\mathcal{K}$ models for $\mathcal{S}$, and $\models$ the satisfaction relation
- Definition 2.27: Let $\mathcal{S} := \langle \mathcal{L}, \mathcal{K}, \models \rangle$ be a logical system, $\mathcal{M} \in \mathcal{K}$ be a model and $\mathbf{A} \in \mathcal{L}$ a formula, then we call $\mathbf{A}$
  - satisfied by $\mathcal{M}$, iff $\mathcal{M} \models \mathbf{A}$
  - falsified by $\mathcal{M}$, iff $\mathcal{M} \not\models \mathbf{A}$
  - satisfiable in $\mathcal{K}$, iff $\mathcal{M} \models \mathbf{A}$ for some model $\mathcal{M} \in \mathcal{K}$.
  - valid in $\mathcal{K}$, iff $\mathcal{M} \models \mathbf{A}$ for all models $\mathcal{M} \in \mathcal{K}$
  - falsifiable in $\mathcal{K}$, iff $\mathcal{M} \not\models \mathbf{A}$ for some $\mathcal{M} \in \mathcal{K}$.
  - unsatisfiable in $\mathcal{K}$, iff $\mathcal{M} \not\models \mathbf{A}$ for all $\mathcal{M} \in \mathcal{K}$.
- Definition 2.28: Let $\mathcal{S} := \langle \mathcal{L}, \mathcal{K}, \models \rangle$ be a logical system, then we define the entailment relation $\models \subseteq \mathcal{L} \times \mathcal{L}$. We say that $\mathbf{A}$ entails $\mathbf{B}$ (written $\mathbf{A} \models \mathbf{B}$), iff we have $\mathcal{M} \models \mathbf{B}$ for all models $\mathcal{M} \in \mathcal{K}$ with $\mathcal{M} \models \mathbf{A}$.
- Theorem 2.29: $\mathbf{A} \models \mathbf{B}$ *and* $\mathcal{M} \models \mathbf{A}$ *imply* $\mathcal{M} \models \mathbf{B}$.

Inference Rules and Calculi

▷ Definition 2.30: Let $\mathcal{S} := \langle \mathcal{L}, \mathcal{K}, \models \rangle$ be a logical system, then we call a relation $\vdash \subseteq \mathcal{P}(\mathcal{L}) \times \mathcal{L}$ a calculus for $\mathcal{S}$, if it

- ▷ is proof-reflexive, i.e. $\mathcal{H} \vdash \mathbf{A}$, if $\mathbf{A} \in \mathcal{H}$,
- ▷ is proof-transitive, i.e. $(\mathcal{H}' \cup \mathcal{H}'') \vdash \mathcal{A}$, if $\mathcal{H} \vdash \mathbf{A}$ and $\mathcal{H}' \vdash \mathbf{A}'$, where $\mathbf{A}' \in \mathcal{H}$ and $\mathcal{H}'' = \mathcal{H} \backslash \{\mathbf{A}\}$.
- ▷ admits weakening, i.e. $\mathcal{H} \vdash \mathbf{A}$ and $\mathcal{H} \subseteq \mathcal{H}'$ imply $\mathcal{H}' \vdash \mathbf{A}$.

▷ Definition 2.31: A calculus $\mathcal{C}$ is usually given as a set of inference rules $\frac{\mathbf{A}_1, \ldots, \mathbf{A}_n}{\mathbf{C}} \mathcal{N}$, where $\mathbf{A}_1, \ldots, \mathbf{A}_n$ and $\mathbf{C}$ are formula schemata and $\mathcal{N}$ is a name.
The $\mathbf{A}_i$ are called assumptions, and $\mathbf{C}$ is called conclusion.

▷ Definition 2.32: An inference rule without assumptions is called an axiom (schema).

Derivations and Proofs

▷ Definition 2.33: A derivation of a formula $\mathbf{C}$ from a set $\mathcal{H}$ of hypotheses (write $\mathcal{H} \vdash \mathbf{C}$) is a sequence $\mathbf{A}_1, \ldots, \mathbf{A}_m$ of formulae, such that

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

▷ Example 2.34: $\mathbf{A} \vdash (\mathbf{B} \Rightarrow \mathbf{A})$

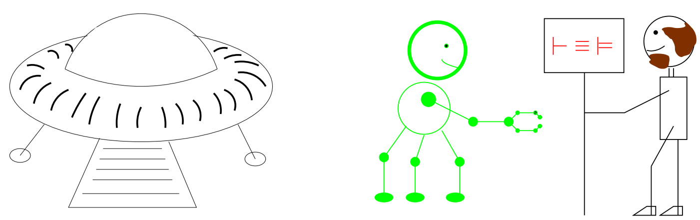
$$\dfrac{\dfrac{}{\mathbf{A} \Rightarrow \mathbf{B} \Rightarrow \mathbf{A}} Ax \qquad \mathbf{A}}{\mathbf{B} \Rightarrow \mathbf{A}} \Rightarrow E$$

▷ Definition 2.35: A derivation $\emptyset \vdash_{\mathcal{C}} \mathbf{A}$ is called a proof of $\mathbf{A}$ and if one exists ($\vdash_{\mathcal{C}} \mathbf{A}$) then $\mathbf{A}$ is called a $\mathcal{C}$-theorem.

▷ Definition 2.36: an inference rule $\mathcal{I}$ is called admissible in $\mathcal{C}$, if the extension of $\mathcal{C}$ by $\mathcal{I}$ does not yield new theorems.

Properties of Calculi (Theoretical Logic)

- ▷ Correctness: (provable implies valid)
  - ▹ $\mathcal{H} \vdash \mathbf{B}$ implies $\mathcal{H} \models \mathbf{B}$ (equivalent: $\vdash \mathbf{A}$ implies $\models \mathbf{B}$)
- ▷ Completeness: (valid implies provable)
  - ▹ $\mathcal{H} \models \mathbf{B}$ implies $\mathcal{H} \vdash \mathbf{B}$ (equivalent: $\models \mathbf{A}$ implies $\vdash \mathbf{B}$)
- ▷ Goal: $\vdash \mathbf{A}$ iff $\models \mathbf{A}$ (provability and validity coincide)
  - ▹ To TRUTH through PROOF (CALCULEMUS [Leibniz ∼1680])



 30 JACOBS UNIVERSITY

2.5 First-Order Calculi

In this section we will introduce two reasoning calculi for first-order logic, both were invented by Gerhard Gentzen in the 1930's and are very much related. The "natural deduction" calculus was created in order to model the natural mode of reasoning e.g. in everyday mathematical practice. This calculus was intended as a counter-approach to the well-known Hilbert'style calculi, which were mainly used as theoretical devices for studying reasoning in principle, not for modeling particular reasoning styles.

The "sequent calculus" was a rationalized version and extension of the natural deduction calculus that makes certain meta-proofs simpler to push through[1]. EdNote(1)

Both calculi have a similar structure, which is motivated by the human-orientation: rather than using a minimal set of inference rules, they provide two inference rules for every connective and quantifier, one "introduction rule" (an inference rule that derives a formula with that symbol at the head) and one "elimination rule" (an inference rule that acts on a formula with this head and derives a set of subformulae).

This allows us to introduce the calculi in two stages, first for the propositional connectives and then extend this to a calculus for first-order logic by adding rules for the quantifiers.

[1] EdNote: say something about cut elimination/analytical calculi somewhere

Calculi: Natural Deduction (ND) [Gentzen'30]

- ▷ tries to mimic human theorem proving behavior (non- minimal)
- ▷ rules for the introduction and elimination of all connectives

Introduction	Elimination	Axiom
$\dfrac{\mathbf{A}\quad\mathbf{B}}{\mathbf{A}\wedge\mathbf{B}}\wedge I$	$\dfrac{\mathbf{A}\wedge\mathbf{B}}{\mathbf{A}}\wedge E_l \quad \dfrac{\mathbf{A}\wedge\mathbf{B}}{\mathbf{B}}\wedge E_r$	$\dfrac{}{\mathbf{A}\vee\neg\mathbf{A}}\, TND$
$\dfrac{\begin{array}{c}[\mathbf{A}]^1\\ \hline\hline \mathbf{B}\end{array}}{\mathbf{A}\Rightarrow\mathbf{B}}\Rightarrow I^1$	$\dfrac{\mathbf{A}\Rightarrow\mathbf{B}\quad\mathbf{A}}{\mathbf{B}}\Rightarrow E$	

- ▷ only in classical logic (otherwise constructive/intuitionistic)

Natural Deduction: Examples

- ▷ Inference with local hypotheses

$$\dfrac{\dfrac{\dfrac{[\mathbf{A}\wedge\mathbf{B}]^1}{\mathbf{B}}\wedge E_r \quad \dfrac{[\mathbf{A}\wedge\mathbf{B}]^1}{\mathbf{A}}\wedge E_l}{\mathbf{B}\wedge\mathbf{A}}\wedge I}{\mathbf{A}\wedge\mathbf{B}\Rightarrow\mathbf{B}\wedge\mathbf{A}}\Rightarrow I^1$$

Natural Deduction in Sequent Calculus Formulation

- ▷ Idea: Explicit representation of hypotheses (lift calculus to judgments)
- ▷ Definition 2.37: A judgment is a meta-statement about the provability of propositions
- ▷ Definition 2.38: A sequent is a judgment of the form $\mathcal{H} \vdash \mathbf{A}$ about the provability of the formula $\mathbf{A}$ from the set $\mathcal{H}$ hypotheses.
- ▷ Idea: Reformulate ND rules so that they act on sequents
- ▷ Example 2.39:

$$\cfrac{\cfrac{\cfrac{[\mathbf{A} \wedge \mathbf{B}] \vdash \mathbf{B}}{[\mathbf{A}, \mathbf{B}] \vdash \mathbf{B}} \wedge E_r \qquad \cfrac{[\mathbf{A} \wedge \mathbf{B}] \vdash \mathbf{A}}{[\mathbf{A}, \mathbf{B}] \vdash \mathbf{A}} \wedge E_l}{[\mathbf{A} \wedge \mathbf{B}] \vdash \mathbf{B} \wedge \mathbf{A}} \wedge I}{\emptyset \vdash \mathbf{A} \wedge \mathbf{B} \Rightarrow \mathbf{B} \wedge \mathbf{A}} \Rightarrow I$$

- ▷ Note: Even though the antecedent of a sequent is written like a sequence, it is actually a set. In particular, we can permute and duplicate members at will.

  JACOBS UNIVERSITY

Sequent-Style Rules for Natural Deduction

- ▷ Definition 2.40: The following inference rules make up the sequent calculus

$$\frac{}{\Gamma, \mathbf{A} \vdash \mathbf{A}} Ax \qquad \frac{\Gamma \vdash \mathbf{B}}{\Gamma, \mathbf{A} \vdash \mathbf{B}} weaken \qquad \frac{}{\Gamma \vdash \mathbf{A} \vee \neg \mathbf{A}} TND$$

$$\frac{\Gamma \vdash \mathbf{A} \quad \Gamma \vdash \mathbf{B}}{\Gamma \vdash \mathbf{A} \wedge \mathbf{B}} \wedge I \qquad \frac{\Gamma \vdash \mathbf{A} \wedge \mathbf{B}}{\Gamma \vdash \mathbf{A}} \wedge E_l \qquad \frac{\Gamma \vdash \mathbf{A} \wedge \mathbf{B}}{\Gamma \vdash \mathbf{B}} \wedge E_r$$

$$\frac{\Gamma \vdash \mathbf{A}}{\Gamma \vdash \mathbf{A} \vee \mathbf{B}} \vee I_l \qquad \frac{\Gamma \vdash \mathbf{B}}{\Gamma \vdash \mathbf{A} \vee \mathbf{B}} \vee I_r \qquad \frac{\Gamma \vdash \mathbf{A} \vee \mathbf{B} \quad \Gamma, \mathbf{A} \vdash \mathbf{C} \quad \Gamma, \mathbf{B} \vdash \mathbf{C}}{\Gamma \vdash \mathbf{C}} \vee E$$

$$\frac{\Gamma, \mathbf{A} \vdash \mathbf{B}}{\Gamma \vdash \mathbf{A} \Rightarrow \mathbf{B}} \Rightarrow I \qquad \frac{\Gamma \vdash \mathbf{A} \Rightarrow \mathbf{B} \quad \Gamma \vdash \mathbf{A}}{\Gamma \vdash \mathbf{B}} \Rightarrow E$$

$$\frac{\Gamma, \mathbf{A} \vdash F}{\Gamma \vdash \neg \mathbf{A}} \neg I \qquad \frac{\Gamma \vdash \neg\neg \mathbf{A}}{\mathbf{A}} \neg E$$

$$\frac{\Gamma \vdash \neg \mathbf{A} \quad \Gamma \vdash \mathbf{A}}{\Gamma \vdash F} FI \qquad \frac{\Gamma \vdash F}{\Gamma \vdash \mathbf{A}} FE$$

  JACOBS UNIVERSITY

First-Order Natural Deduction

- Rules for propositional connectives just as always
- Definition 2.41: (New Quantifier Rules)

$$\frac{\mathbf{A}}{\forall X.\mathbf{A}}\,\forall I^* \qquad \frac{\forall X.\mathbf{A}}{[\mathbf{B}_\iota/X]\mathbf{A}}\,\forall E$$

$$\frac{[\mathbf{B}/X]\mathbf{A}}{\exists X.\mathbf{A}}\,\exists I \qquad \frac{\exists X.\mathbf{A} \quad \begin{array}{c}[[c/X]\mathbf{A}] \\ \hline\hline \mathbf{C}\end{array}}{\mathbf{C}}\,\exists E$$

* means that $\mathbf{A}$ does not depend on any hypothesis in which X is free.

First-Order Natural Deduction in Sequent Formulation

- Rules for propositional connectives just as always
- Definition 2.42: (New Quantifier Rules)

$$\frac{\Gamma \vdash \mathbf{A} \quad X \notin \mathbf{free}(\Gamma)}{\Gamma \vdash \forall X.\mathbf{A}}\,\forall I \qquad \frac{\Gamma \vdash \forall X.\mathbf{A}}{\Gamma \vdash [\mathbf{B}_\iota/X]\mathbf{A}}\,\forall E$$

$$\frac{\Gamma \vdash [\mathbf{B}/X]\mathbf{A}}{\Gamma \vdash \exists X.\mathbf{A}}\,\exists I \qquad \frac{\Gamma \vdash \exists X.\mathbf{A} \quad \Gamma, [c/X]\mathbf{A} \vdash \mathbf{C} \quad c \in \Sigma_0^{sk} \text{ new}}{\Gamma \vdash \mathbf{C}}\,\exists E$$

Linearized Notation for ND [Andrews]

- Linearized version of sequents

$$\begin{array}{llll} 1. & \mathcal{H}_1 \vdash & \mathbf{A}_1 & (\mathcal{J}_1) \\ 2. & \mathcal{H}_2 \vdash & \mathbf{A}_2 & (\mathcal{J}_2) \\ 3. & \mathcal{H}_3 \vdash & \mathbf{A}_3 & (\mathcal{R}1,2) \end{array} \quad \text{corresponds to} \quad \frac{\mathcal{H}_1 \vdash \mathbf{A}_1 \quad \mathcal{H}_2 \vdash \mathbf{A}_2}{\mathcal{H}_3 \vdash \mathbf{A}_3}\,\mathcal{R}$$

- Example:

1.	1	$\vdash$	$\mathbf{A} \wedge \mathbf{B}$	(Hyp)
2.	1	$\vdash$	$\mathbf{B}$	($\wedge E_r$ 1)
3.	1	$\vdash$	$\mathbf{A}$	($\wedge E_l$ 1)
4.	1	$\vdash$	$\mathbf{B} \wedge \mathbf{A}$	($\wedge I$ 2 3)
Thm.		$\vdash$	$\mathbf{A} \wedge \mathbf{B} \Rightarrow \mathbf{B} \wedge \mathbf{A}$	($\Rightarrow I$ 4)

2.6 Abstract Consistency and Model Existence

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

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

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

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

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

Model Existence (Overview)

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

 38 JACOBS UNIVERSITY

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

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

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

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

Consistency

- ▷ Let $\mathcal{C}$ be a calculus
- ▷ Definition 2.43: Φ is called $\mathcal{C}$-refutable, if there is a formula $\mathbf{B}$, such that $\Phi \vdash_{\mathcal{C}} \mathbf{B}$ and $\Phi \vdash_{\mathcal{C}} \neg(\mathbf{B})$.
- ▷ Definition 2.44: We call a pair $\mathbf{A}$ and $\neg\mathbf{A}$ a contradiction.
- ▷ So a set Φ is $\mathcal{C}$-refutable, if $\mathcal{C}$ can derive a contradiction from it.
- ▷ Definition 2.45: Φ is called $\mathcal{C}$-consistent, iff there is a formula $\mathbf{B}$, that is not derivable from Φ in $\mathcal{C}$.
- ▷ Definition 2.46: We call a calculus $\mathcal{C}$ reasonable, iff Modus Ponens is admissible in $\mathcal{C}$ and $\mathbf{A} \wedge \neg\mathbf{A} \Rightarrow \mathbf{B}$ is a $\mathcal{C}$-theorem.
- ▷ Theorem 2.47: *$\mathcal{C}$-inconsistency and $\mathcal{C}$-refutability coincide for reasonable calculi*

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

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

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

Abstract Consistency

- ▷ Definition 2.48: Let ∇ be a family of sets of propositional formulae. We call ∇ closed under subsets, iff for each $\Phi \in \nabla$, all subsets $\Psi \subseteq \Phi$ are elements of ∇.
- ▷ Notation 2.49: We will use $\Phi * \mathbf{A}$ for $\Phi \cup \{\mathbf{A}\}$.
- ▷ Definition 2.50: A family ∇ of sets of formulae is called a (first-order) abstract consistency class, iff it is closed under subsets, and for each $\Phi \in \nabla$

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

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

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

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

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

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

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

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

Compact Collections

▷ Definition 2.51: We call a collection ∇ of sets compact, iff for any set Φ we have $\Phi \in \nabla$, iff $\Psi \in \nabla$ for every finite subset Ψ of Φ.

▷ Lemma 2.52: *If ∇ is compact, then ∇ is closed under subsets.*

▷ Proof:

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

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

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

P.4 Thus $S \in \nabla$. □

 41 JACOBS UNIVERSITY

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

Compact Abstract Consistency Classes

▷ Lemma 2.53: *Any abstract consistency class can be extended to a compact one.*

▷ Proof:

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

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

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

P.4.1 Suppose $\Phi \in \nabla'$ and Ψ is an arbitrary finite subset of Φ.

P.4.2 By definition of ∇' all finite subsets of Φ are in ∇ and therefore $\Psi \in \nabla'$.

P.4.3 Thus all finite subsets of Φ are in ∇' whenever Φ is in ∇'.

P.4.4 On the other hand, suppose all finite subsets of Φ are in ∇'.

P.4.5 Then by the definition of ∇' the finite subsets of Φ are also in ∇, so $\Phi \in \nabla'$. Thus ∇' is compact. □

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

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

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

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

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

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

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

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

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

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

□

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

∇-Hintikka Set

▷ Definition 2.54: Let ∇ be an abstract consistency class, then we call a set $\mathcal{H} \in \nabla$ a ∇-Hintikka Set, iff $\mathcal{H}$ is maximal in ∇, i.e. for all $\mathbf{A}$ with $\mathcal{H} * \mathbf{A} \in \nabla$ we already have $\mathbf{A} \in \mathcal{H}$.

▷ Theorem 2.55: (Hintikka Properties)

Let ∇ be an abstract consistency class and $\mathcal{H}$ be a ∇-Hintikka set, then

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

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

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

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

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

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

▷ Proof:

P.1 We prove the properties in turn

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

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

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

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

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

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

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

□

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

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

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

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

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

Extension Theorem

▷ Theorem 2.56: *If ∇ is an abstract consistency class and $\Phi \in \nabla$ finite, then there is a ∇-Hintikka set $\mathcal{H}$ with $\Phi \subseteq \mathcal{H}$.*

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

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

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

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

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

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

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

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

P.7 Thus $\mathcal{H}$ is ∇-maximal □

 44

Note that the construction in the proof above is non-trivial in two respects. First, the limit construction for $\mathcal{H}$ is not executed in our original abstract consistency class ∇, but in a suitably extended one to make it compact — the original would not have contained $\mathcal{H}$ in general. Second, the set $\mathcal{H}$ is not unique for Φ, but depends on the choice of the enumeration of $cwff_o(\Sigma_\iota)$. If we pick a different enumeration, we will end up with a different $\mathcal{H}$. Say if $\mathbf{A}$ and $\neg\mathbf{A}$ are both ∇-consistent[2] with Φ, then depending on which one is first in the enumeration $\mathcal{H}$, will contain EdNote(2) that one; with all the consequences for subsequent choices in the construction process.

Valuation

▷ Definition 2.57: A function $\nu\colon cwff_o(\Sigma_\iota) \to \mathcal{D}_o$ is called a (first-order) valuation, iff

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

▷ Lemma 2.58: *If $\varphi\colon \mathcal{V}_o \to \mathcal{D}_o$ is a variable assignment, then $\mathcal{I}_\varphi\colon cwff_o(\Sigma_\iota) \to \mathcal{D}_o$ is a valuation.*

▷ Proof: Immediate from the definitions □

 45

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

[2] EdNote: introduce this above

Valuation and Satisfiability

▷ Lemma 2.59: *If $\nu\colon cwff_o(\Sigma_\iota) \to \mathcal{D}_o$ is a valuation and $\Phi \subseteq cwff_o(\Sigma_\iota)$ with $\nu(\Phi) = \{\mathsf{T}\}$, then Φ is satisfiable.*

▷ Proof: We construct a model for Φ.

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

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

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

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

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

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

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

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

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

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

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

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

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

 46 JACOBS UNIVERSITY

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

Model Existence

▷ Theorem 2.60: (Hintikka-Lemma)

If ∇ is an abstract consistency class and $\mathcal{H}$ a ∇-Hintikka set, then $\mathcal{H}$ is satisfiable.

▷ Proof:

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

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

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

▷ Theorem 2.61: (Model Existence)

If ∇ is an abstract consistency class and $\Phi \in \nabla$, then Φ is satisfiable.

▷ Proof:

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

P.2 We know that $\mathcal{H}$ is satisfiable. (Hintikka-Lemma)

P.3 In particular, $\Phi \subseteq \mathcal{H}$ is satisfiable. □

2.7 A Completeness Proof for First-Order ND

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

Consistency, Refutability and Abstract Consistency

▷ Theorem 2.62: (Non-Refutability is an Abstract Consistency Property)
$\Gamma := (\{\Phi \subseteq cwff_o(\Sigma_\iota) \mid \Phi \text{ not } \mathcal{ND}\text{−refutable}\})$ *is an abstract conistency class.*

▷ Proof: We check the properties of an ACC

P.1 If Φ is non-refutable, then any subset is as well, so Γ is closed under subsets.

P.2 We show the abstract consistency conditions ∇_* for $\Phi \in \Gamma$.

P.2.1 ∇_c:

P.2.1.1 We have to show that $\mathbf{A} \notin \Phi$ or $\neg\mathbf{A} \notin \Phi$ for atomic $\mathbf{A} \in wff_o(\Sigma)$.

P.2.1.2 Equivalently, we show the contrapositive: If $\{\mathbf{A}, \neg\mathbf{A}\} \subseteq \Phi$, then $\Phi \notin \Gamma$.

P.2.1.3 So let $\{\mathbf{A}, \neg\mathbf{A}\} \subseteq \Phi$, then Φ is $\mathcal{ND}$-refutable by construction.

P.2.1.4 So $\Phi \notin \Gamma$. □

P.2.2 $\nabla_\neg$: We show the contrapositive again:

P.2.2.1 Let $\neg\neg\mathbf{A} \in \Phi$ and $\Phi * \mathbf{A} \notin \Gamma$

P.2.2.2 Then we have a refutation $\mathcal{D}: \Phi * \mathbf{A} \vdash_{\mathcal{ND}} F$

P.2.2.3 By prepending an application of $\neg E$ for $\neg\neg\mathbf{A}$ to $\mathcal{D}$, we obtain a refutation $\mathcal{D}': \Phi \vdash_{\mathcal{ND}} F$.

P.2.2.4 Thus $\Phi \notin \Gamma$. □

P.2.3 other ∇_* similar: □

  JACOBS UNIVERSITY

Henkin's Theorem

▷ Corollary 2.63: (Henkin's Theorem)
Every $\mathcal{ND}$-consistent set of sentences has a model.

▷ Proof:

P.1 Let Φ be a $\mathcal{ND}$-consistent set of sentencens.

P.2 The class of sets of $\mathcal{ND}$-consistent propositions constitute an abstract consistency class

P.3 Thus the model existence theorem guarantees a countable model for Φ. □

▷ Corollary 2.64: (Löwenheim&Skolem Theorem)
Satisfiable set Φ of first-order sentences has a countable model.

▷ Proof: The model we constructed is countable, since the set of ground terms is. □

  JACOBS UNIVERSITY

Completeness and Compactness

▷ Theorem 2.65: (Completeness Theorem for $\mathcal{ND}$)
If $\Phi \models \mathbf{A}$, then $\Phi \vdash_{\mathcal{ND}} \mathbf{A}$.

▷ Proof:

P.1 If $\mathbf{A}$ is valid in all models of Φ, then $\Phi * \neg\mathbf{A}$ has no model

P.2 Thus $\Phi * \neg\mathbf{A}$ is inconsistent by Henkins Theorem.

P.3 So $\Phi \vdash_{\mathcal{ND}} \neg(\neg\mathbf{A})$[c]

P.4 So $\Phi \vdash_{\mathcal{ND}} \mathbf{A}$ by $\neg E$. □

▷ Theorem 2.66: (Compactness Theorem for first-order logic)
If $\Phi \models \mathbf{A}$, then there is already a finite set $\Psi \subseteq \Phi$ with $\Psi \models \mathbf{A}$.

▷ Proof: This is a direct consequence of the completeness theorem

P.1 We have $\Phi \models \mathbf{A}$, iff $\Phi \vdash_{\mathcal{ND}} \mathbf{A}$.

P.2 As a proof is a finite object, only a finite subset $\Psi \subseteq \Phi$ can appear as leaves in the proof. □

 ©: Michael Kohlhase 50 JACOBS UNIVERSITY

[c]EdNote: show this

2.8 Limits of First-Order Logic

We will now come to the limits of first-order Logic.

Gödel's Incompleteness Theorem

▷ Theorem 2.67: *No logical system that can Peano-Arithmetic ($\mathbb{N}, s, 0, +, *$) admits complete calculi.*

▷ Proof: (Sketch)

P.1 Let $\mathcal{L} := \langle \mathcal{S}, \mathcal{C} \rangle$ be such a system. We show that there is a valid $\mathcal{S}$-sentence $\mathbf{A}_{\mathcal{C}}$, that is no $\mathcal{C}$-theorem.

P.2 Encode the syntax of $\mathcal{S}$ and the $\mathcal{C}$ in Peano-arithmetic

P.3 We can now talk about $\mathcal{S}$ and $\mathcal{C}$ in $\mathcal{S}$ itself.

P.4 E.g. there is a $\mathcal{S}$-sentence $\mathbf{B}$ with the menaning: $\mathbf{A}$ is a $\mathcal{C}$-theorem.

P.5 Choose $\mathbf{A}_{\mathcal{C}}$ as "$\mathbf{A}_{\mathcal{C}}$ is no $\mathcal{C}$-theorem" (cf. Russell's set)

P.6 Obviously: $\mathbf{A}_{\mathcal{C}}$ ist valid in all standard models.

P.7 So $\mathcal{C}$ is either not correct or cannot derive $\mathbf{A}_{\mathcal{C}}$. □

 ©: Michael Kohlhase 51 JACOBS UNIVERSITY

3 First-Order Automated Theorem Proving with Tableaux

3.1 First-Order Tableaux

Test Calculi: Tableaux and Model Generation

- ▷ Idea: instead of showing $\emptyset \vdash Th$, show $\neg Th \vdash trouble$ (use $\bot$ for trouble)
- ▷ Example 3.1:

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

- ▷ Algorithm: Fully expand all possible tableaux, (no rule can be applied)
  - ▷ Satisfiable, iff there are open branches (correspond to models)

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

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

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

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

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

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

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

- Algorithm: Use rules exhaustively as long as they contribute new material
- Definition 3.2: Call a tableau saturated, iff no rule applies, and a branch closed, iff it ends in $\perp$, else open. (open branches in saturated tableaux yield models)
- Definition 3.3: ($\mathcal{T}$-Theorem/Derivability)

  $\mathbf{A}$ is a $\mathcal{T}$-theorem ($\vdash_{\mathcal{T}} \mathbf{A}$), iff there is a closed tableau with $\mathbf{A}^{\mathsf{F}}$ at the root.

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

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

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

Definition 3.4: We will call a closed tableau with the signed formula $\mathbf{A}^{\alpha}$ at the root a tableau refutation for $\mathcal{A}^{\alpha}$.

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

Definition 3.5: We will call a tableau refutation for $\mathbf{A}^{\mathsf{F}}$ a tableau proof for $\mathbf{A}$, since it refutes the possibility of finding a model where $\mathbf{A}$ evaluates to F. Thus $\mathbf{A}$ must evaluate to T in all models, which is just our definition of validity.

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

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

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

First-Order Standard Tableaux ($\mathcal{T}_1$)

▷ Refutation calculus based on trees of labeled formulae

▷ Tableau-Rules: propositional tableau rules plus

$$\frac{\forall X.\mathbf{A}^{\mathsf{T}} \quad \mathbf{C} \in cwff_{\iota}(\Sigma_{\iota})}{[\mathbf{C}/X]\mathbf{A}^{\mathsf{T}}}\,\mathcal{T}_1{:}\forall \qquad \frac{\forall X.\mathbf{A}^{\mathsf{F}} \quad c \in (\Sigma_0^{sk} \backslash \mathcal{H})}{[c/X]\mathbf{A}^{\mathsf{F}}}\,\mathcal{T}_1{:}\exists$$

The rule $\mathcal{T}_1{:}\forall$ rule operationalizes the intuition that a universally quantified formula is true, iff all of the instances of the scope are. To understand the $\mathcal{T}_1{:}\exists$ rule, we have to keep in mind that $\exists X.\mathbf{A}$ abbreviates $\neg(\forall X.\neg\mathbf{A})$, so that we have to read $\forall X.\mathbf{A}^{\mathsf{F}}$ existentially — i.e. as $\exists X.\neg\mathbf{A}^{\mathsf{T}}$, stating that there is an object with property $\neg\mathbf{A}$. In this situation, we can simply give this object a name: c, which we take from our (infinite) set of witness constants Σ_0^{sk}, which we have given ourselves expressly for this purpose when we defined first-order syntax. In other words $[c/X]\neg\mathbf{A}^{\mathsf{T}} = [c/X]\mathbf{A}^{\mathsf{F}}$ holds, and this is just the conclusion of the $\mathcal{T}_1{:}\exists$ rule.

Note that the $\mathcal{T}_1{:}\forall$ rule is computationally extremely inefficient: we have to guess an (i.e. in a search setting to systematically consider all) instance $\mathbf{C} \in wff_{\iota}(\Sigma_{\iota})$ for X. This makes the rule infinitely branching.

3.2 Free Variable Tableaux

In the next calculus we will try to remedy the computational inefficiency of the $\mathcal{T}_1{:}\forall$ rule. We do this by delaying the choice in the universal rule.

Free variable Tableaux ($\mathcal{T}_1^f$)

▷ Refutation calculus based on trees of labeled formulae

▷ Tableau rules

$$\frac{\forall X.\mathbf{A}^{\mathsf{T}} \quad Y \text{ new}}{[Y/X]\mathbf{A}^{\mathsf{T}}}\,\mathcal{T}_1^f{:}\forall \qquad \frac{\forall X.\mathbf{A}^{\mathsf{F}} \quad \mathbf{free}(\forall X.\mathbf{A}) = \{X^1, \ldots, X^k\} \quad f \in \Sigma_k^{sk}}{[f(X^1, \ldots, X^k)/X]\mathbf{A}^{\mathsf{F}}}\,\mathcal{T}_1^f{:}\exists$$

▷ Generalized cut rule $\mathcal{T}_1^f{:}\bot$ instantiates the whole tableau by σ.

$$\frac{\begin{matrix}\mathbf{A}^{\alpha} \\ \mathbf{B}^{\beta}\end{matrix} \quad \alpha \neq \beta \quad \sigma\mathbf{A} = \sigma\mathbf{B}}{\bot}\,\mathcal{T}_1^f{:}\bot$$

▷ Advantage: no guessing necessary in $\mathcal{T}_1^f{:}\forall$-rule

▷ New: find suitable substitution (most general unifier)

Metavariables: Instead of guessing a concrete instance for the universally quantified variable as in the $\mathcal{T}_1$:$\forall$ rule, $\mathcal{T}_1^f$:$\forall$ instantiates it with a new meta-variable Y, which will be instantiated by need in the course of the derivation.

Skolem terms as witnesses: The introduction of meta-variables makes is necessary to extend the treatment of witnesses in the existential rule. Intuitively, we cannot simply invent a new name, since the meaning of the body $\mathbf{A}$ may contain meta-variables introduced by the $\mathcal{T}_1^f$:$\forall$ rule. As we do not know their values yet, the witness for the existential statement in the antecedent of the $\mathcal{T}_1^f$:$\exists$ rule needs to depend on that. So witness it using a witness term, concretely by applying a Skolem function to the meta-variables in $\mathbf{A}$.

Instantiating Metavariables: Finally, the $\mathcal{T}_1^f$:$\perp$ rule completes the treatment of meta-variables, it allows to instantiate the whole tableau in a way that the current branch closes. This leaves us with the problem of finding substitutions that make two terms equal.

3.3 Properties of First-Order Tableaux

Correctness of $\mathcal{T}_1^f$

▷ Lemma 3.6: $\mathcal{T}_1^f$:$\exists$ *transforms satisfiable tableaux into satisfiable ones.*

▷ Proof: Let $\mathcal{T}'$ be obtained by applying $\mathcal{T}_1^f$:$\exists$ to $\forall X.\mathbf{A}^\mathsf{F}$ in $\mathcal{T}$, extending it with $[f(X^1,\ldots,X^n)/X]\mathbf{A}^\mathsf{F}$, where $W := \mathbf{free}(\forall X.\mathbf{A}) = \{X^1,\ldots,X^k\}$

P.1 Let $\mathcal{T}$ be satisfiable in $\mathcal{M} := \langle\mathcal{D},\mathcal{I}\rangle$, then $\mathcal{I}_\varphi(\forall X.\mathbf{A}) = \mathsf{F}$.

P.2 We need to find a model $\mathcal{M}'$ that satisfies $\mathcal{T}'$ (find interpretation for f)

P.3 By definition $\mathcal{I}_{\varphi,[a/X]}(\mathbf{A}) = \mathsf{F}$ for some $a \in \mathcal{D}$ (depends on $\varphi|_W$)

P.4 Let $g\colon \mathcal{D}^n \to \mathcal{D}$ be defined by $g(a_1,\ldots,a_k) := a$, if $\varphi(X^i) = a_i$

P.5 choose $\mathcal{M}' = \langle\mathcal{D},\mathcal{I}'\rangle$ with $\mathcal{I}' := \mathcal{I},[g/f]$, then by subst. value lemma

$$\mathcal{I}'_\varphi([f(X^1,\ldots,X^k)/X]\mathbf{A}) = \mathcal{I}'_{\varphi,[\mathcal{I}'_\varphi(f(X^1,\ldots,X^k))/X]}(\mathbf{A}) = \mathcal{I}'_{\varphi,[a/X]}(\mathbf{A}) = \mathsf{F}$$

P.6 So $[f(X^1,\ldots,X^k)/X]\mathbf{A}^\mathsf{F}$ satisfiable in $\mathcal{M}'$ □

Correctness of $\mathcal{T}_1^f$

▷ Lemma 3.7: *Tableau rules transform satisfiable tableaux into satisfiable ones.*

▷ Proof: we examine the tableau rules in turn

P.1.1 propositional rules: as in propositional tableaux □

P.1.2 $\mathcal{T}_1^f{:}\exists$: in the Lemma above □

P.1.3 $\mathcal{T}subst$: by substitution value lemma □

P.1.4 $\mathcal{T}_1^f{:}\forall$:

P.1.4.1 $\mathcal{I}_\varphi(\forall X.\mathbf{A}) = \mathsf{T}$, iff $\mathcal{I}_\psi(\mathbf{A}) = \mathsf{T}$ for all $a \in \mathcal{D}_\iota$

P.1.4.2 so in particular for some $a \in \mathcal{D}_\iota \neq \emptyset$. □

▷ Corollary 3.8: $\mathcal{T}_1^f$ *is correct.*

Completeness of ($\mathcal{T}_1^f$)

▷ Theorem 3.9: $\mathcal{T}_1^f$ *is refutation complete.*

▷ Proof: We show that $\nabla := (\{\Phi \mid \Phi^\mathsf{T} \text{ has no closed Tableau}\})$ is an abstract consistency class

P.1 (∇_*, $\nabla_\neg$, $\nabla_\vee$, and $\nabla_\neg$) as for propositional case.

P.2 ($\nabla_\forall$) by the lifting lemma below

P.3 ($\nabla_\exists$) Let $\mathcal{T}$ be a closed tableau for $\neg(\forall X.\mathbf{A}) \in \Phi$ and $\Phi^\mathsf{T} * [c/X](\mathbf{A})^\mathsf{F} \in \nabla$.

$$\begin{array}{c} \Psi^\mathsf{T} \\ \forall X.\mathbf{A}^\mathsf{F} \\ [c/X]\mathbf{A}^\mathsf{F} \\ Rest \end{array} \qquad \begin{array}{c} \Psi^\mathsf{T} \\ \forall X.\mathbf{A}^\mathsf{F} \\ [f(X^1,\ldots,X^k)/X]\mathbf{A}^\mathsf{F} \\ [f(X^1,\ldots,X^k)/c]Rest \end{array}$$

□

Tableau-Lifting

- **Theorem 3.10:** *If $\mathcal{T}_\theta$ is a closed tableau for a st $\theta\Phi$ of formulae, then there is a closed tableau $\mathcal{T}$ for Φ.*
- Proof: by induction over the structure of $\mathcal{T}_\theta$ we build an isomorphic tableau $\mathcal{T}$, and a tableau-isomorphism $\omega\colon \mathcal{T} \to \mathcal{T}_\theta$, such that $\omega(\mathbf{A}) = \theta\mathbf{A}$.

  P.1 only the tableau-substitution rule is interesting.

  P.2 Let $\theta{\mathbf{A}^i}^{\mathsf{T}}$ and $\theta{\mathbf{B}^i}^{\mathsf{F}}$ cut formulae in the branch Θ^i_θ of $\mathcal{T}_\theta$

  P.3 there is a joint unifier σ of $\theta(\mathbf{A}^1)=^?\theta(\mathbf{B}^1) \wedge \ldots \wedge \theta(\mathbf{A}^n)=^?\theta(\mathbf{B}^n)$

  P.4 thus $\sigma \circ \theta$ is a unifier of $\mathbf{A}$ and $\mathbf{B}$

  P.5 hence there is a most general unifier ρ of $\mathbf{A}^1=^?\mathbf{B}^1 \wedge \ldots \wedge \mathbf{A}^n=^?\mathbf{B}^n$

  P.6 so Θ is closed $\square$

4 Higher-Order Logic and λ-Calculus

4.1 Higher-Order Predicate Logic

Higher-Order Predicate Logic ($PL\Omega$)

- Quantification over functions and Predicates: $\forall P.\exists F.P(a) \vee \neg P(F(a))$
- Comprehension: (Existence of Functions)
  $\exists F.\forall X.FX = \mathbf{A}$ e.g. $f(x) = 3x^+5x - 7$
- Extensionality: (Equality of functions and truth values)
  $\forall F.\forall G.(\forall X.FX = GX) \Rightarrow F = G$
  $\forall P.\forall Q.(P \Leftrightarrow Q) \Leftrightarrow P = Q$
- Leibniz-Equality: (Indiscernability)
  $\mathbf{A} = \mathbf{B}$ for $\forall P.P\mathbf{A} \Rightarrow P\mathbf{B}$
- Problem: Russell's Antinomy ($\forall Q.\mathcal{M}(Q) \Leftrightarrow \neg Q(Q)$)
  - the set $\mathcal{M}$ of all sets that do not contain themselves
  - Question: Is $\mathcal{M} \in \mathcal{M}$? Answer: $\mathcal{M} \in \mathcal{M}$ iff $\mathcal{M} \notin \mathcal{M}$.
- What has happened? the predicate Q has been applied to itself
- Solution for this course: Forbid self-applications by types!!
  - ι, o (type of individuals, truth values), $\alpha \to \beta$ (function type)
  - right associative bracketing: $\alpha \to \beta \to \gamma$ abbreviates $\alpha \to (\beta \to \gamma)$
  - vector notation: $\overline{\alpha_n} \to \beta$ abbreviates $\alpha_1 \to \ldots \to \alpha_n \to \beta$
- Well-typed formulae (prohibits paradoxes like $\forall Q.\mathcal{M}(Q) \Leftrightarrow \neg Q(Q)$)
- Other solution: Give it a non-standard semantics (Domain-Theory [Scott])

Well-Typed Formulae ($PL\Omega$)

- signature $\Sigma = \bigcup_{\alpha \in \mathcal{T}} \Sigma_\alpha$ with
- connectives: $\neg \in \Sigma_{(o \to o)}$ $\{\vee, \wedge, \Rightarrow, \Leftrightarrow \ldots\} \subseteq \Sigma_{o \to o \to o}$
- variables $\mathcal{V}_\mathcal{T} = \bigcup_{\alpha \in \mathcal{T}} \mathcal{V}_\alpha$, such that every $\mathcal{V}_\alpha$ countably infinite.
- well-typed formulae $\mathit{wff}_\alpha(\Sigma, \mathcal{V}_\mathcal{T})$ of type α
  - $\mathcal{V}_\alpha \cup \Sigma_\alpha \subseteq \mathit{wff}_\alpha(\Sigma, \mathcal{V}_\mathcal{T})$
  - If $\mathbf{C} \in \mathit{wff}_{(\alpha \to \beta)}(\Sigma, \mathcal{V}_\mathcal{T})$ and $\mathbf{A} \in \mathit{wff}_\alpha(\Sigma, \mathcal{V}_\mathcal{T})$, then $(\mathbf{CA}) \in \mathit{wff}_\beta(\Sigma, \mathcal{V}_\mathcal{T})$
  - If $\mathbf{A} \in \mathit{wff}_o(\Sigma, \mathcal{V}_\mathcal{T})$, then $(\forall X_\alpha.\mathbf{A}) \in \mathit{wff}_o(\Sigma, \mathcal{V}_\mathcal{T})$
- first-order terms have type ι, formulae (propositions) the type o.
- there is no type annotation such that $\forall Q.\mathcal{M}(Q) \Leftrightarrow \neg Q(Q)$ is well-typed.
  Q needs type α as well as $\alpha \to o$.

Standard Semantics

- Definition 4.1: The universe of discourse (carrier)
  - arbitrary set of individuals $\mathcal{D}_\iota$ fixed truth values $\mathcal{D}_o = \{\mathsf{T}, \mathsf{F}\}$
  - function universes $\mathcal{D}_{\alpha \to \beta} = \mathcal{F}(\mathcal{D}_\alpha; \mathcal{D}_\beta)$
- interpretation of constants: typed mapping $\mathcal{I} \colon \Sigma \to \mathcal{D}$ (i.e. $\mathcal{I}(\Sigma_\alpha) \subseteq \mathcal{D}_\alpha$)
- variable assignment: typed mapping $\varphi \colon \mathcal{V}_\mathcal{T} \to \mathcal{D}$
- Definition 4.2: value function: typed mapping $\mathcal{I}_\varphi \colon \mathit{wff}_\mathcal{T}(\Sigma, \mathcal{V}_\mathcal{T}) \to \mathcal{D}$
  - $\mathcal{I}_\varphi|_{\mathcal{V}_\mathcal{T}} = \varphi$ $\mathcal{I}_\varphi|_{\Sigma_\mathcal{T}} = \mathcal{I}$
  - $\mathcal{I}_\varphi(\mathbf{AB}) = \mathcal{I}_\varphi(\mathbf{A})(\mathcal{I}_\varphi(\mathbf{B}))$
  - $\mathcal{I}_\varphi(\forall X_\alpha.\mathbf{A}) = \mathsf{T}$, iff $\mathcal{I}_{\varphi,[\mathsf{a}/X]}(\mathbf{A}) = \mathsf{T}$ for all $\mathsf{a} \in \mathcal{D}_\alpha$.
- $\mathbf{A}_o$ valid under φ, iff $\mathcal{I}_\varphi(\mathbf{A}) = \mathsf{T}$.

Example: Peano Axioms for the Natural Numbers

- $\Sigma = \{[\mathbb{N}\colon \iota \to o], [0\colon \iota], [s\colon \iota \to \iota]\}$
- $\mathbb{N}0$ (0 is a natural number)
- $\forall X_\iota.\mathbb{N}X \Rightarrow \mathbb{N}(sX)$ (the successor of a natural number is natural)
- $\neg(\exists X_\iota.\mathbb{N}X \wedge sX = 0)$ (0 has no predecessor)
- $\forall X_\iota.\forall Y_\iota.(sX = sY) \Rightarrow X = Y$ (the successor function is injective)
- $\forall P_{\iota\to o}.P0 \Rightarrow (\forall X_\iota.\mathbb{N}X \Rightarrow PXP(sX)) \Rightarrow (\forall Y_\iota.\mathbb{N}Y \Rightarrow P(Y))$
  induction axiom: all properties P, that hold of 0, and with every n for its successor $s(n)$, hold on all $\mathbb{N}$

Expressive Formalism for Mathematics

- Example 4.3: (Cantor's Theorem)
  The cardinality of a set is smaller than that of its power set.
  - smaller − card$(M, N) := \neg(\exists F.$surjective$(F, M, N))$
  - surjective$(F, M, N) := (\forall X \in M.\exists Y \in N.FY = X)$

  Simplified Formalization: $\neg\exists F_{\iota\to\iota\to\iota}.\forall G_{\iota\to\iota}.\exists J_\iota.FJ = G$
- Standard-Benchmark for higher-order theorem provers
- can be proven by TPS and LEO (see below)

Hilbert-Calculus

- Definition 4.4: ($\mathcal{H}_\Omega$ Axioms)
  - $\forall P_o, Q_o.P \Rightarrow Q \Rightarrow P$
  - $\forall P_o, Q_o, R_o.P \Rightarrow Q \Rightarrow R \Rightarrow P \Rightarrow Q \Rightarrow P \Rightarrow R$
  - $\forall P_o, Q_o.\neg P \Rightarrow \neg Q \Rightarrow P \Rightarrow Q$
- Definition 4.5: ($\mathcal{H}_\Omega$ Inference rules)

$$\frac{\mathbf{A}_o \Rightarrow \mathbf{B}_o \quad \mathbf{A}}{\mathbf{B}} \qquad \frac{\forall X_\alpha.\mathbf{A}}{[\mathbf{B}/X_\alpha]\mathbf{A}} \qquad \frac{\mathbf{A}}{\forall X_\alpha.\mathbf{A}} \qquad \frac{X \notin \mathbf{free}(\mathbf{A}) \quad \forall X_\alpha.\mathbf{A} \wedge \mathbf{B}}{\mathbf{A} \wedge (\forall X_\alpha.\mathbf{B})}$$

- Theorem 4.6: *Correct wrt. standard semantics*
- Also Complete?

Hilbert-Calculus $\mathcal{H}_\Omega$ (continued)

- valid sentences that are not $\mathcal{H}_\Omega$-theorems:
  - Cantor's Theorem:
    $\neg(\exists F_{\iota\iota\to\iota}.\forall G_{\iota\to\iota}.(\forall K_\iota.\mathbb{N}K \Rightarrow \mathbb{N}(GK)) \Rightarrow (\exists J_\iota.(\mathbb{N}J) \wedge FJ = G))$
    (There is no surjective mapping from $\mathbb{N}$ into the set $\mathcal{F}(\mathbb{N};,)\mathbb{N}$ of natural number sequences)
  - proof attempt fails at the subgoal $\exists G_{\iota\to\iota}.\forall X_\iota.GX = s(fXX)$
- new axiom schema: Comprehension: $\exists F_{\overline{\alpha_n}\to\beta}.\forall X_\alpha.FX = \mathbf{A}_\beta$
  (for every variable X_α and every term $\mathbf{A} \in \mathit{wff}_\beta(\Sigma, \mathcal{V}_\mathcal{T})$)
- new axiom schemata: extensionality
  $\mathbf{Ext}^{\alpha\beta}$ $\forall F_{\alpha\to\beta}.\forall G_{\alpha\to\beta}.(\forall X_\alpha.FX = GX) \Rightarrow F = G$
  $\mathbf{Ext}^{\mathbf{o}}$ $\forall F_o.\forall G_o.(F \Leftrightarrow G) \Leftrightarrow F = G$
- correct! complete? cannot be!! [Gödel 1931]

  JACOBS UNIVERSITY

Way Out: Henkin-Semantics

- Gödel's incompleteness theorem only holds for standard semantics
- find generalization that admits complete calculi:
- Idea: generalize so that the carrier only contains those functions that are requested by the comprehension axioms.
- Theorem 4.7: (Henkin 1950)

  $\mathcal{H}_\Omega$ is complete wrt. this semantics.
- Proof: Idea

  more models $\leadsto$ less valid sentences (these are $\mathcal{H}_\Omega$-theorems)

  □
- Henkin-models induce sensible measure of completeness for higher-order logic.

  JACOBS UNIVERSITY

Equality

- ▷ "Leibniz equality" (Indiscernability) $\mathbf{Q}^\alpha \mathbf{A}_\alpha \mathbf{B}_\alpha = \forall P_{\alpha\to o}.P\mathbf{A} \Leftrightarrow P\mathbf{B}$
- ▷ not that $\forall P_{\alpha\to o}.P\mathbf{A} \Rightarrow P\mathbf{B}$(get the other direction by instantiating P with Q, where $QX \Leftrightarrow \neg PX$)
- ▷ Theorem 4.8: *If $\mathcal{M} = \langle \mathcal{D}, \mathcal{I} \rangle$ is a standard model, then $\mathcal{I}_\varphi(\mathbf{Q}^\alpha)$ is the identity relation on $\mathcal{D}_\alpha$.*
- ▷ Notation 4.9: We write $\mathbf{A} = \mathbf{B}$ for $\mathbf{QAB}$($\mathbf{A}$ and $\mathbf{B}$ are equal, iff there is no property P that can tell them apart.)
- ▷ Proof:

P.1 $\mathcal{I}_\varphi(\mathbf{QAB}) = \mathcal{I}_\varphi(\forall P.P\mathbf{A} \Rightarrow P\mathbf{B}) =$ T, iff $\mathcal{I}_{\varphi,[r/P]}(PX \Rightarrow PY) =$ T for all $r \in \mathcal{D}_{\alpha\to o}$.

P.2 For $\mathbf{A} = \mathbf{B}$ we have $\mathcal{I}_{\varphi,[r/P]}(P\mathbf{A}) = r(\mathcal{I}_\varphi(\mathbf{A})) =$ F or $\mathcal{I}_{\varphi,[r/P]}(P\mathbf{A}) = r(\mathcal{I}_\varphi(\mathbf{A})) =$ T.

P.3 Thus $\mathcal{I}_\varphi(\mathbf{QAB}) =$ T.

P.4 Let $\mathcal{I}_\varphi(\mathbf{A}) \neq \mathcal{I}_\varphi(\mathbf{B})$ and $r = \{\mathcal{I}_\varphi(\mathbf{A})\}$

P.5 so $r(\mathcal{I}_\varphi(\mathbf{A})) =$ T and $r(\mathcal{I}_\varphi(\mathbf{B})) =$ F

P.6 $\mathcal{I}_\varphi(\mathbf{QAB}) =$ F, as $\mathcal{I}_{\varphi,[r/P]}(P\mathbf{A} \Rightarrow P\mathbf{B}) =$ F, since $\mathcal{I}_{\varphi,[r/P]}(P\mathbf{A}) = r(\mathcal{I}_\varphi(\mathbf{A})) =$ T and $\mathcal{I}_{\varphi,[r/P]}(P\mathbf{B}) = r(\mathcal{I}_\varphi(\mathbf{B})) =$ F. □

 68 JACOBS UNIVERSITY

4.2 Simply Typed λ-Calculus

In this section we will present a logic that can deal with functions – the simply typed λ-calculus. It is a typed logic, so everything we write down is typed (even if we do not always write the types down).

Simply typed λ-Calculus (Syntax)

- ▷ Signature $\Sigma = \bigcup_{\alpha\in\mathcal{T}} \Sigma_\alpha$
- ▷ $\mathcal{V}_\mathcal{T} = \bigcup_{\alpha\in\mathcal{T}} \mathcal{V}_\alpha$, such that $\mathcal{V}_\alpha$ are countably infinite
- ▷ Definition 4.10: We call the set $\mathit{wff}_\alpha(\Sigma, \mathcal{V}_\mathcal{T})$ defined by the rules
  - ▷ $\mathcal{V}_\alpha \cup \Sigma_\alpha \subseteq \mathit{wff}_\alpha(\Sigma, \mathcal{V}_\mathcal{T})$
  - ▷ If $\mathbf{C} \in \mathit{wff}_{(\alpha\to\beta)}(\Sigma, \mathcal{V}_\mathcal{T})$ and $\mathbf{A} \in \mathit{wff}_\alpha(\Sigma, \mathcal{V}_\mathcal{T})$, then $(\mathbf{CA}) \in \mathit{wff}_\beta(\Sigma, \mathcal{V}_\mathcal{T})$
  - ▷ If $\mathbf{A} \in \mathit{wff}_\alpha(\Sigma, \mathcal{V}_\mathcal{T})$, then $(\lambda X_\beta.\mathbf{A}) \in \mathit{wff}_{(\beta\to\alpha)}(\Sigma, \mathcal{V}_\mathcal{T})$

  the set of well-typed formulae of type α over the signature Σ and use $\mathit{wff}_\mathcal{T}(\Sigma, \mathcal{V}_\mathcal{T}) := \bigcup_{\alpha\in\mathcal{T}} \mathit{wff}_\alpha(\Sigma, \mathcal{V}_\mathcal{T})$ for the set of all well-typed formulae.
- ▷ Definition 4.11: 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 $\mathbf{B}$.

 69 JACOBS UNIVERSITY

Simply typed λ-Calculus (Notations)

▷ Notation 4.12: (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: 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 $\mathbf{A}(\mathbf{B}\mathbf{C})$.

▷ Notation 4.13: (Abstraction is right-associative)

We abbreviate $\lambda X^1.\lambda X^2.\cdots\lambda X^n.\mathbf{A}\cdots$ with $\lambda X^1\ldots X^n.\mathbf{A}$ eliding brackets, and further to $\lambda\overline{X_n}.\mathbf{A}$ in a kind of vector notation.

▷ Notation 4.14: (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 $\mathbf{B}$.[4] EdNote(4)

The intuitions about functional structure of λ-terms and about free and bound variables are encoded into three transformation rules $\Lambda^{\rightarrow}$:

$\alpha\beta\eta$-Equality (Overview)

▷ reduction with $\left\{\begin{array}{l}\beta : (\lambda X.\mathbf{A})\mathbf{B} \rightarrow_\beta [\mathbf{B}/X](\mathbf{A}) \\ \eta : \lambda X.\mathbf{A}X \rightarrow_\eta \mathbf{A}\end{array}\right.$ under α : $\begin{array}{c}(\lambda X.\mathbf{A}) \\ =_\alpha \\ (\lambda Y.[Y/X](\mathbf{A}))\end{array}$

▷ Theorem 4.15: *$\beta\eta$-reduction is well-typed, correct, terminating and confluent in the presence of α-conversion.*

▷ Consequence: Unique β-normal form $\lambda X^1\ldots X^k.h\mathbf{A}^1\ldots\mathbf{A}^n$ where

- ▷ h constant or variable (the head symbol)
- ▷ $h\mathbf{A}^1\ldots\mathbf{A}^n$ (the matrix) $\lambda X^1\ldots X^k.$ (the Binder)
- ▷ the subterms $\mathbf{A}^i$ are of the same form.

▷ Definition 4.16: Head Reduction always has a unique β redex

$$\lambda\overline{X_n}.(\lambda Y.\mathbf{A})\mathbf{B}^1\ldots\mathbf{B}^n \rightarrow_\beta^h \lambda\overline{X_n}.[\mathbf{B}^1/Y](\mathbf{A})\mathbf{B}^2\ldots\mathbf{B}^n$$

▷ Definition 4.17: Long $\beta\eta$-normal form, iff β-NF and matrix has base type.

▷ Definition 4.18: η-Expansion: $\eta\left[\lambda X^1\ldots X^n.\mathbf{A}\right] := \lambda X^1\ldots X^nY^1\ldots Y^m.\mathbf{A}Y^1\ldots Y^m$

The first rule (α-conversion) just says that we can rename bound variables as we like. The β-reduction rule codifies the intuition behind function application by replacing bound variables with argument. The third rule 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

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

the same argument – say $\mathbf{B}$ and β-reduce the left side, then we arrive at the right hand side $\lambda X_\alpha.\mathbf{A}X\mathbf{B} \rightarrow_\beta \mathbf{AB}$.

The semantics of $\Lambda^{\rightarrow}$ is structured around the types. Like the models we discussed before, a model $\mathcal{M}$ 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 4.19**: We call a collection $\mathcal{D}_\mathcal{T} := (\{\mathcal{D}_\alpha \mid \alpha \in \mathcal{T}\})$ a typed collection (of sets) and a collection $f_\mathcal{T}\colon \mathcal{D}_\mathcal{T} \rightarrow \mathcal{E}_\mathcal{T}$, a typed function, iff $f_\alpha\colon \mathcal{D}_\alpha \rightarrow \mathcal{E}_\alpha$.
- **Definition 4.20**: A typed collection $\mathcal{D}_\mathcal{T}$ is called a frame, iff $\mathcal{D}_{\alpha\rightarrow\beta} \subseteq \mathcal{D}_\alpha \rightarrow \mathcal{D}_\beta$
- **Definition 4.21**: Given a frame $\mathcal{D}_\mathcal{T}$, and a typed function $\mathcal{I}\colon \Sigma \rightarrow \mathcal{D}$, then we call $\mathcal{I}_\varphi\colon \mathit{wff}_\mathcal{T}(\Sigma,\mathcal{V}_\mathcal{T}) \rightarrow \mathcal{D}$ the value function induced by $\mathcal{I}$, iff
  - $\mathcal{I}_\varphi|_{\mathcal{V}_\mathcal{T}} = \varphi$, $\quad \mathcal{I}_\varphi|_\Sigma = \mathcal{I}$
  - $\mathcal{I}_\varphi(\mathbf{AB}) = \mathcal{I}_\varphi(\mathbf{A})(\mathcal{I}_\varphi(\mathbf{B}))$
  - $\mathcal{I}_\varphi(\lambda X_\alpha.\mathbf{A})$ is that function $f \in \mathcal{D}_{(\alpha\rightarrow\beta)}$, such that $f(a) = \mathcal{I}_{\varphi,[a/X]}(\mathbf{A})$ for all $a \in \mathcal{D}_\alpha$
- **Definition 4.22**: We call a pair $\langle\mathcal{D},\mathcal{I}\rangle$ a Σ-model, iff $\mathcal{I}_\varphi\colon \mathit{wff}_\mathcal{T}(\Sigma,\mathcal{V}_\mathcal{T}) \rightarrow \mathcal{D}$ is total. (comprehension-closed)

  Such modes are also called generalized models ([Henkin 1950])

4.3 Simply Typed λ Calculus

Simply typed λ-Calculus

- arbitrary (for now) set $\mathcal{BT}$ of base types.
- Signature $\Sigma_\mathcal{T} = \bigcup_{\alpha\in\mathcal{T}} \Sigma_\alpha$
- $\mathcal{V}_\mathcal{T} = \bigcup_{\alpha\in\mathcal{T}} \mathcal{V}_\alpha$, such that $\mathcal{V}_\alpha$ countably infinite
- well-typed formulae $\mathit{wff}_\alpha(\Sigma,\mathcal{V}_\mathcal{T})$ of type α
  - $\mathcal{V}_\alpha, \Sigma_\alpha \subseteq \mathit{wff}_\alpha(\Sigma,\mathcal{V}_\mathcal{T})$
  - If $\mathbf{C} \in \mathit{wff}_{(\alpha\rightarrow\beta)}(\Sigma,\mathcal{V}_\mathcal{T})$ and $\mathbf{A} \in \mathit{wff}_\alpha(\Sigma,\mathcal{V}_\mathcal{T})$, then $(\mathbf{CA}) \in \mathit{wff}_\beta(\Sigma,\mathcal{V}_\mathcal{T})$
  - If $\mathbf{A} \in \mathit{wff}_\alpha(\Sigma,\mathcal{V}_\mathcal{T})$, then $(\lambda X_\beta.\mathbf{A}) \in \mathit{wff}_{(\beta\rightarrow\alpha)}(\Sigma,\mathcal{V}_\mathcal{T})$
- α-equality: $\lambda X_\alpha.\mathbf{A} =_\alpha \lambda Y_\alpha.[Y/X](\mathbf{A})$ if $Y \notin \mathbf{free}(\mathbf{A})$
- β-equality: $(\lambda X_\alpha.\mathbf{A})\mathbf{B} =_\beta [\mathbf{B}/X](\mathbf{A})$

Types

- Types are semantic annotations for terms that prevent antinomies
- Definition 4.23: 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 types over $\mathcal{BT}$.
- Definition 4.24: (iotypes.def)

  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 $\overline{\alpha_n} \to \beta$.

Substitution Value Lemma for λ-Terms

- Lemma 4.25: (Substitution Value Lemma)

  Let $\mathbf{A}$ and $\mathbf{B}$ be terms, then $\mathcal{I}_\varphi([\mathbf{B}/X]\mathbf{A}) = \mathcal{I}_\psi(\mathbf{A})$, where $\psi = \varphi, [\mathcal{I}_\varphi(\mathbf{B})/X]$
- Proof: by induction on the depth of $\mathbf{A}$

  P.1 we have five cases

  P.1.1 $\mathbf{A} = X$: Then $\mathcal{I}_\varphi([\mathbf{B}/X]\mathbf{A}) = \mathcal{I}_\varphi([\mathbf{B}/X]X) = \mathcal{I}_\varphi(\mathbf{B}) = \psi(X) = \mathcal{I}_\psi(X) = \mathcal{I}_\psi(\mathbf{A})$. □

  P.1.2 $\mathbf{A} = Y \neq X$ and $Y \in \mathcal{V}_\mathcal{T}$: then $\mathcal{I}_\varphi([\mathbf{B}/X]\mathbf{A}) = \mathcal{I}_\varphi([\mathbf{B}/X]Y) = \mathcal{I}_\varphi(Y) = \varphi(Y) = \psi(Y) = \mathcal{I}_\psi(Y) = \mathcal{I}_\psi(\mathbf{A})$. □

  P.1.3 $\mathbf{A} \in \Sigma$: This is analogous to the last case. □

  P.1.4 $\mathbf{A} = \mathbf{CD}$: then $\mathcal{I}_\varphi([\mathbf{B}/X]\mathbf{A}) = \mathcal{I}_\varphi([\mathbf{B}/X]\mathbf{CD}) = \mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{C})[\mathbf{B}/X](\mathbf{D})) = \mathcal{I}_\varphi([\mathbf{B}/X]\mathbf{C})(\mathcal{I}_\varphi([\mathbf{B}/X]\mathbf{D})) = \mathcal{I}_\psi(\mathbf{C})(\mathcal{I}_\psi(\mathbf{D})) = \mathcal{I}_\psi(\mathbf{CD}) = \mathcal{I}_\psi(\mathbf{A})$ □

  P.1.5 $\mathbf{A} = \lambda Y_\alpha.\mathbf{C}$:

  P.1.5.1 We can assume that $X \neq Y$ and $Y \notin \mathbf{free}(\mathbf{B})$

  P.1.5.2 Thus for all $a \in \mathcal{D}_\alpha$ we have $\mathcal{I}_\varphi([\mathbf{B}/X]\mathbf{A})(a) = \mathcal{I}_\varphi([\mathbf{B}/X]\lambda Y.\mathbf{C})(a) = \mathcal{I}_\varphi(\lambda Y.[\mathbf{B}/X](\mathbf{C}))(a) = \mathcal{I}_{\varphi,[a/Y]}([\mathbf{B}/X]\mathbf{C}) = \mathcal{I}_{\psi,[a/Y]}(\mathbf{C}) = \mathcal{I}_\psi(\lambda Y.\mathbf{C})(a) = \mathcal{I}_\psi(\mathbf{A})(a)$ □

  □

Correctness of $\alpha\beta\eta$-Equality

▷ Theorem 4.26: *Let $\mathcal{A} := \langle\mathcal{D}, \mathcal{I}\rangle$ be a Henkin model and $Y \notin \mathbf{free}(\mathbf{A})$, then $\mathcal{I}_\varphi(\lambda X.\mathbf{A}) = \mathcal{I}_\varphi(\lambda Y.[Y/X]\mathbf{A})$ for all assignments φ.*

▷ Proof: by substitution value lemma

P.1

$$\begin{aligned} \mathcal{I}_\varphi(\lambda Y.[Y/X]\mathbf{A})@\mathsf{a} &= \mathcal{I}_{\varphi,[a/Y]}([Y/X]\mathbf{A}) \\ &= \mathcal{I}_{\varphi,[a/X]}(\mathbf{A}) \\ &= \mathcal{I}_\varphi(\lambda X.\mathbf{A})@\mathsf{a} \end{aligned}$$

□

▷ Theorem 4.27: *If $\mathcal{A} := \langle\mathcal{D}, \mathcal{I}\rangle$ is a Henkin model 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

P.1

$$\begin{aligned} \mathcal{I}_\varphi((\lambda X.\mathbf{A})\mathbf{B}) &= \mathcal{I}_\varphi(\lambda X.\mathbf{A})@\mathcal{I}_\varphi(\mathbf{B}) \\ &= \mathcal{I}_{\varphi,[\mathcal{I}_\varphi(\mathbf{B})/X]}(\mathbf{A}) \\ &= \mathcal{I}_\varphi([\mathbf{B}/X]\mathbf{A}) \end{aligned}$$

□

 76 JACOBS UNIVERSITY

Correctness of $\alpha\beta\eta$ (continued)

▷ Theorem 4.28: *If $X \notin \mathbf{free}(\mathbf{A})$, then $\mathcal{I}_\varphi(\lambda X.\mathbf{A}X) = \mathcal{I}_\varphi(\mathbf{A})$ for all φ.*

▷ Proof:

$$\begin{aligned} \mathcal{I}_\varphi(\lambda X.\mathbf{A}X)@\mathsf{a} &= \mathcal{I}_{\varphi,[\mathsf{a}/X]}(\mathbf{A}X) \\ &= \mathcal{I}_\varphi(\mathbf{A})@\mathcal{I}_{\varphi,[\mathsf{a}/X]}(X) \\ &= \mathcal{I}_\varphi(\mathbf{A})@\mathsf{a} \end{aligned}$$

as $X \notin \mathbf{free}(\mathbf{A})$. □

▷ Theorem 4.29: *$\alpha\beta\eta$-equality is correct wrt. Henkin models. (if $\mathbf{A} =_{\alpha\beta\eta} \mathbf{B}$, then $\mathcal{I}_\varphi(\mathbf{A}) = \mathcal{I}_\varphi(\mathbf{B})$ for all assignments φ)*

 77 JACOBS UNIVERSITY

Simply Typed λ-Calculus as an Inference System

▷ Judgment $\Gamma \vdash_\Sigma \mathbf{A} \colon \alpha$ (formula $\mathbf{A}$ has type α given the type assumptions in Γ)

▷ $\mathbf{A} \in \mathit{wff}_\alpha(\Sigma, \mathcal{V}_\mathcal{T})$, iff $\Gamma \vdash_\Sigma \mathbf{A} \colon \alpha$ derivable in

$$\frac{c \in \Sigma_\alpha}{\Gamma \vdash_\Sigma c \colon \alpha}\,\text{wff:const} \qquad \frac{\Gamma \vdash_\Sigma \mathbf{A} \colon (\beta \to \alpha) \quad \Gamma \vdash_\Sigma \mathbf{B} \colon \beta}{\Gamma \vdash_\Sigma (\mathbf{A}\mathbf{B}) \colon \alpha}\,\text{wff:app}$$

$$\frac{\Gamma(X) = \alpha}{\Gamma \vdash_\Sigma X \colon \alpha}\,\text{wff:var} \qquad \frac{\Gamma, [X \colon \alpha] \vdash_\Sigma \mathbf{A} \colon \alpha}{\Gamma \vdash_\Sigma (\lambda X_\beta.\mathbf{A}) \colon (\beta \to \alpha)}\,\text{wff:abs}$$

▷ Oops: this looks surprisingly like a natural deduction calculus. ($\Rightarrow$ Curry Howard Isomorphism)

$\beta\eta$-Equality by Inference Rules: One-Step Reduction

▷ One-step Reduction ($+ \in \{\alpha, \beta, \eta\}$)

$$\frac{}{\vdash ((\lambda X.\mathbf{A})\mathbf{B}) \to^1_\beta [\mathbf{B}/X](\mathbf{A})}\,\text{wff}\beta\text{:top} \qquad \frac{X \notin \mathbf{free}(\mathbf{A})}{\vdash (\lambda X.\mathbf{A}X) \to^1_\beta \mathbf{A}}\,\text{wff}\eta\text{:top}$$

$$\frac{\vdash \mathbf{A} \to^1_+ \mathbf{B}}{\vdash (\mathbf{A}\mathbf{C}) \to^1_+ (\mathbf{B}\mathbf{C})}\,\text{tr:app}fn \qquad \frac{\vdash \mathbf{A} \to^1_+ \mathbf{B}}{\vdash (\mathbf{C}\mathbf{A}) \to^1_+ (\mathbf{C}\mathbf{B})}\,\text{tr:app}arg$$

$$\frac{\vdash \mathbf{A} \to^1_+ \mathbf{B}}{\vdash (\lambda X.\mathbf{A}) \to^1_+ (\lambda X.\mathbf{B})}\,\text{tr:abs}$$

$\beta\eta$-Equality by Inference Rules: Multi-Step Reduction

▷ Multi-Step-Reduction ($+ \in \{\alpha, \beta, \eta\}$)

$$\frac{\vdash \mathbf{A} \rightarrow^1_+ \mathbf{B}}{\vdash \mathbf{A} \rightarrow^*_+ \mathbf{B}}\,\text{ms:start} \qquad \frac{}{\vdash \mathbf{A} \rightarrow^*_+ \mathbf{A}}\,\text{ms:ref}$$

$$\frac{\vdash \mathbf{A} \rightarrow^*_+ \mathbf{B} \quad \vdash \mathbf{B} \rightarrow^*_+ \mathbf{C}}{\vdash \mathbf{A} \rightarrow^*_+ \mathbf{C}}\,\text{ms:trans}$$

▷ Congruence Relation

$$\frac{\vdash \mathbf{A} \rightarrow^*_+ \mathbf{B}}{\vdash \mathbf{A} =_+ \mathbf{B}}\,\text{eq:start}$$

$$\frac{\vdash \mathbf{A} =_+ \mathbf{B}}{\vdash \mathbf{B} =_+ \mathbf{A}}\,\text{eq:sym} \qquad \frac{\vdash \mathbf{A} =_+ \mathbf{B} \quad \vdash \mathbf{B} =_+ \mathbf{C}}{\vdash \mathbf{A} =_+ \mathbf{C}}\,\text{eq:trans}$$

  JACOBS UNIVERSITY

4.4 Computational Properties of λ-Calculus

From Extensionality to η-Conversion

▷ Definition 4.30: Extensionality Axiom: $\forall F_{\alpha\rightarrow\beta}.\forall G_{\alpha\rightarrow\beta}.(\forall X_\alpha.FX = GX) \Rightarrow F = G$

▷ Theorem 4.31: *η-equality and Extensionality are equivalent*

▷ Proof: We show that η-equality ($\mathbf{A}_{\alpha\rightarrow\beta} =_\eta \lambda X_\alpha.\mathbf{A}X$, if $X \notin \mathbf{free}(\mathbf{A})$) is special case of extensionality; the converse entailment is trivial

P.1 Let $\forall X_\alpha.\mathbf{A}X = \mathbf{B}X$, thus $\mathbf{A}X = \mathbf{B}X$ with $\forall E$

P.2 $\lambda X_\alpha.\mathbf{A}X = \lambda X_\alpha.\mathbf{B}X$, therefore $\mathbf{A} = \mathbf{B}$ with η

P.3 Hence $\forall F_o.\forall G_o.(F \Leftrightarrow G) \Leftrightarrow F = G$ □

▷ Axiom of truth values: $\forall F_o.\forall G_o.(F \Leftrightarrow G) \Leftrightarrow F = G$ unsolved.

  JACOBS UNIVERSITY

η-Reduction ist terminating and confluent

- ▷ Lemma 4.32: *η-Reduction ist terminating*
- ▷ Proof: by a simple counting argument □
- ▷ Lemma 4.33: *η-Reduction ist confluent*
- ▷ Proof: by diagram chase □

$\beta\eta$ is confluent

- ▷ Lemma 4.34: *$\rightarrow^*_\beta$ and $\rightarrow^*_\eta$ commute.*
- ▷ Proof: diagram chase □

4.4.1 Termination of β-reduction

The second result is that β reduction terminates. We will use this 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.

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 4.35: (Typed β-Reduction terminates)
  For all $\mathbf{A} \in \mathit{wff}_\alpha(\Sigma, \mathcal{V}_\mathcal{T})$, the chain of reductions from $\mathbf{A}$ is finite.
- ▷ proof attempts:
  - ▷ Induction on the structure $\mathbf{A}$ must fail, since this would also work for the untyped case.
  - ▷ Induction on the type of $\mathbf{A}$ must fail, since β-reduction conserves types.
- ▷ combined induction on both: Logical Relations [Tait 1967]

Relations $\mathcal{SR}$ and $\mathcal{LR}$

- ▷ **Definition 4.36:** $\mathbf{A}$ is called strongly reducing at type α (write $\mathcal{SR}(\mathbf{A},\alpha)$), iff each chain β-reductions from $\mathbf{A}$ terminates.
- ▷ **Lemma 4.37:** (Lemma 1)

  If $\mathcal{SR}(\mathbf{C},\alpha)$ and $\mathbf{B}_\beta$ is a subterm of $\mathbf{A}$, then $\mathcal{SR}(\mathbf{B},\beta)$.
- ▷ **Proof Idea:** Every infinite β-reduction from $\mathbf{B}$ would be one from $\mathbf{A}$. □
- ▷ We define a logical relation $\mathcal{LR}$ inductively on the structure of the type
  - ▷ α base type: $\mathcal{LR}(\mathbf{A},\alpha)$, iff $\mathcal{SR}(\mathbf{A},\alpha)$
  - ▷ $\mathcal{LR}(\mathbf{C},\alpha\to\beta)$, iff $\mathcal{LR}(\mathbf{CA},\beta)$ for all $\mathbf{A}\in wff_\alpha(\Sigma,\mathcal{V}_\mathcal{T})$ with $\mathcal{LR}(\mathbf{A},\alpha)$.
- ▷ **Proof:** Termination Proof

  P.1 $\mathcal{LR}\subseteq\mathcal{SR}$ (Lemma 2b)

  P.2 $\mathbf{A}\in wff_\alpha(\Sigma,\mathcal{V}_\mathcal{T})$ implies $\mathcal{LR}(\mathbf{A},\alpha)$

  P.3 also $\mathcal{SR}(\mathbf{A},\alpha)$ □

 85 JACOBS UNIVERSITY

$\mathcal{LR}\subseteq\mathcal{SR}$ (Rollercoaster Lemma)

- ▷ **Lemma 4.38:** (Lemma 2)

  a) If h is a constant or variable of type $\overline{\alpha_n}\to\beta$ and $\mathcal{SR}(\mathbf{A}^i,\alpha^i)$, then $\mathcal{LR}(h\overline{\mathbf{A}^n},\beta)$.

  b) $\mathcal{LR}(\mathbf{A},\alpha)$ implies $\mathcal{SR}(\mathbf{A},\alpha)$.
- ▷ **Proof:** we prove both assertions by simultaneous induction on α

  P.1.1 α base type:

  P.1.1.1.1 a): $h\overline{\mathbf{A}^n}$ is strongly reducing, since the $\mathbf{A}^i$ are (brackets!)

  P.1.1.1.1.2 so $\mathcal{LR}(h\overline{\mathbf{A}^n},\beta)$ as α is a base type ($\mathcal{SR}=\mathcal{LR}$) □

  P.1.1.1.2 b): by definition □

  □

  P.1.2 $\alpha=\beta\to\gamma$:

  P.1.2.1.1 a): Let $\mathcal{LR}(\mathbf{B},\beta)$.

  P.1.2.1.1.2 by the second IH we have $\mathcal{SR}(\mathbf{B},\beta)$, and $\mathcal{LR}(h\overline{\mathbf{A}^n}\mathbf{B},\gamma)$ by the first IH

  P.1.2.1.1.3 so $\mathcal{LR}(h\overline{\mathbf{A}^n},\beta)$ by definition. □

  P.1.2.1.2 b): Let $\mathcal{LR}(\mathbf{A},\alpha)$ and $X_\beta\notin\mathbf{free}(\mathbf{A})$.

  P.1.2.1.2.2 by the first IH (with $n=0$) we have $\mathcal{LR}(X,\beta)$, thus $\mathcal{LR}(\mathbf{A}X,\gamma)$ by definition.

  P.1.2.1.2.3 By the second IH we have $\mathcal{SR}(\mathbf{A}X,\gamma)$ and by Lemma 1 $\mathcal{SR}(\mathbf{A},\alpha)$. □

  □

  □

 86 JACOBS UNIVERSITY

β-Expansion-Lemma

▷ Lemma 4.39: *If* $\mathcal{LR}([\mathbf{B}/X]\mathbf{A},\alpha)$ *and* $\mathcal{LR}(\mathbf{B},\beta)$ *for* $X_\beta \notin \mathbf{free}(\mathbf{A})$, *then* $\mathcal{LR}((\lambda X_\alpha.\mathbf{A})\mathbf{B},\alpha)$.

▷ Proof:

P.1 Let $\alpha = \overline{\gamma_i} \to \delta$ where δ base type and $\mathcal{LR}(\mathbf{C}^i,\gamma^i)$

P.2 It is sufficient to show that $\mathcal{SR}((\lambda X.\mathbf{A})\mathbf{B}\overline{\mathbf{C}},\delta)$, as δ base type

P.3 We have $\mathcal{LR}([\mathbf{B}/X](\mathbf{A})\overline{\mathbf{C}},\delta)$ by hypothesis and definition of $\mathcal{LR}$.

P.4 thus $\mathcal{SR}([\mathbf{B}/X](\mathbf{A})\overline{\mathbf{C}},\delta)$, as δ base type.

P.5 in particular $\mathcal{SR}([\mathbf{B}/X]\mathbf{A},\alpha)$ and $\mathcal{SR}(\mathbf{C}^i,\gamma^i)$ (subterms)

P.6 $\mathcal{SR}(\mathbf{B},\beta)$ by hypothesis and Lemma 2

P.7 So an infinite reduction from $(\lambda X.\mathbf{A})\mathbf{B}\overline{\mathbf{C}}$ cannot solely consist of redexes from $[\mathbf{B}/X]\mathbf{A}$ and the $\mathbf{C}^i$.

P.8 so an infinite reduction from $(\lambda X.\mathbf{A})\mathbf{B}\overline{\mathbf{C}}$ must have the form

$$\begin{array}{lll} (\lambda X.\mathbf{A})\mathbf{B}\overline{\mathbf{C}} & \to_\beta^* & (\lambda X.\mathbf{A}')\mathbf{B}'\overline{\mathbf{C}'} \\ & \to_\beta^1 & [\mathbf{B}'/X](\mathbf{A}')\overline{\mathbf{C}'} \\ & \to_\beta^* & \ldots \end{array}$$

where $\mathbf{A} \to_\beta^* \mathbf{A}'$, $\mathbf{B} \to_\beta^* \mathbf{B}'$ and $\mathbf{C}^i \to_\beta^* \mathbf{C}^{i\prime}$

P.9 so we have $[\mathbf{B}/X](\mathbf{A}) \to_\beta^* [\mathbf{B}'/X](\mathbf{A}')$

P.10 so we have the infinite reduction

$$\begin{array}{lll} [\mathbf{B}/X](\mathbf{A})\overline{\mathbf{C}} & \to_\beta^* & [\mathbf{B}'/X](\mathbf{A}')\overline{\mathbf{C}'} \\ & \to_\beta^* & \ldots \end{array}$$

which contradicts our assumption □

 87 JACOBS UNIVERSITY

Closure under β-Expansion (weakly reducing)

▷ Lemma 4.40: (Lemma 3)
If $\mathbf{C} \to_\beta^h \mathbf{D}$ *and* $\mathcal{LR}(\mathbf{D}, \alpha)$*, so is* $\mathcal{LR}(\mathbf{C}, \alpha)$.

▷ Proof: by induction over the structure of α

P.1.1 α base type:

P.1.1.1 we have $\mathcal{SR}(\mathbf{D}, \alpha)$ by definition

P.1.1.2 so $\mathcal{SR}(\mathbf{C}, \alpha)$, since head reduction is unique

P.1.1.3 and thus $\mathcal{LR}(\mathbf{C}, \alpha)$. □

▷ **P.1.2** $\alpha = \beta \to \gamma$:

P.1.2.1 Let $\mathcal{LR}(\mathbf{B}, \beta)$, by definition we have $\mathcal{LR}(\mathbf{DB}, \gamma)$.

P.1.2.2 but $\mathbf{CB} \to_\beta^h \mathbf{DB}$, so $\mathcal{LR}(\mathbf{CB}, \gamma)$ by IH

P.1.2.3 and $\mathcal{LR}(\mathbf{C}, \alpha)$ by definition. □

□

Note: This Lemma only holds for weak reduction (any chain of β head reductions terminates) for strong reduction we need a stronger Lemma.

 88

$\mathbf{A} \in wff_\alpha(\Sigma, \mathcal{V}_\mathcal{T})$ implies $\mathcal{LR}(\mathbf{A}, \alpha)$

▷ Theorem 4.41: *If* $\mathcal{LR}(\sigma X_\alpha, \alpha)$ *for all* $X \in \mathbf{supp}(\sigma)$ *and* $\mathbf{A} \in wff_\alpha(\Sigma, \mathcal{V}_\mathcal{T})$*, then* $\mathcal{LR}(\sigma\mathbf{A}, \alpha)$.

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

P.1.1 $\mathbf{A} = X_\alpha \in \mathbf{supp}(\sigma)$: then $\mathcal{LR}(\sigma\mathbf{A}, \alpha)$ by assumption □

P.1.2 $\mathbf{A} = X \notin \mathbf{supp}(\sigma)$: then $\sigma\mathbf{A} = \mathbf{A}$ and $\mathcal{LR}(\mathbf{A}, \alpha)$ by Lemma 2 with $n = 0$. □

P.1.3 $\mathbf{A} \in \Sigma$: then $\sigma\mathbf{A} = \mathbf{A}$ as above □

P.1.4 $\mathbf{A} = \mathbf{BC}$: by IH $\mathcal{LR}(\sigma\mathbf{B}, \gamma \to \alpha)$ and $\mathcal{LR}(\sigma\mathbf{C}, \gamma)$

P.1.4.2 so $\mathcal{LR}(\sigma(\mathbf{B})\sigma(\mathbf{C}), \alpha)$ by definition of $\mathcal{LR}$. □

P.1.5 $\mathbf{A} = \lambda X_\beta.\mathbf{C}_\gamma$: Let $\mathcal{LR}(\mathbf{B}, \beta)$ and $\theta := \sigma, [\mathbf{B}/X]$, then θ meets the conditions of the IH.

P.1.5.2 Moreover $\sigma(\lambda X_\beta.\mathbf{C}_\gamma)\mathbf{B} \to_\beta \sigma, [\mathbf{B}/X](\mathbf{C}) = \theta(\mathbf{C})$.

P.1.5.3 Now, $\mathcal{LR}(\theta\mathbf{C}, \gamma)$ by IH and thus $\mathcal{LR}(\sigma(\mathbf{A})\mathbf{B}, \gamma)$ by Lemma 3.

P.1.5.4 So $\mathcal{LR}(\sigma\mathbf{A}, \alpha)$ by definition of $\mathcal{LR}$. □

□

 89

4.5 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.

Normal Forms in the simply typed λ-calculus

- ▷ Definition 4.42: We call a term $\mathbf{A} \in \mathit{wff}_{\mathcal{T}}(\Sigma, \mathcal{V}_{\mathcal{T}})$ a β normal form iff there is no $\mathbf{B} \in \mathit{wff}_{\mathcal{T}}(\Sigma, \mathcal{V}_{\mathcal{T}})$ with $\mathbf{A} \to_\beta \mathbf{B}$.
  We call $\mathbf{N}$ a β normal form of $\mathbf{A}$, iff $\mathbf{N}$ is a β-normal form and $\mathbf{A} \to_\beta \mathcal{N}$.
  We denote the set of β-normal forms with $\mathit{wff}_{\mathcal{T}}(\Sigma, \mathcal{V}_{\mathcal{T}})\big\downarrow_\beta$.
- ▷ We have just proved that $\beta\eta$-reduction is terminating and confluent, so we have
- ▷ Corollary 4.43: (Normal Forms)
  Every $\mathbf{A} \in \mathit{wff}_{\mathcal{T}}(\Sigma, \mathcal{V}_{\mathcal{T}})$ has a unique β normal form ($\beta\eta$, long $\beta\eta$ normal form), which we denote by $\mathbf{A}\downarrow_\beta$ ($\mathbf{A}\downarrow_{\beta\eta}$ $\mathbf{A}\downarrow^l_{\beta\eta}$)

A Herbrand Model for $\Lambda^\to$

- ▷ Definition 4.44: (Term Structures for Σ)
  Let $\mathcal{T}_{\beta\eta} := \langle c\mathit{wff}_{\mathcal{T}}(\Sigma, \mathcal{V}_{\mathcal{T}})\big\downarrow_\beta, \mathcal{I}^{\beta\eta}\rangle$, where $c\mathit{wff}_{\mathcal{T}}(\Sigma, \mathcal{V}_{\mathcal{T}})\big\downarrow_\beta$ is the set of ground $\beta\eta$-normal λ-terms and $\mathcal{I}^{\beta\eta}(c)@\mathbf{A} := c\mathbf{A}\downarrow_\beta$. We call $\mathcal{T}_{\beta\eta}$ the β-term structure for Σ.
- ▷ Let φ be an assignment into $c\mathit{wff}_{\mathcal{T}}(\Sigma, \mathcal{V}_{\mathcal{T}})\big\downarrow_\beta$. Note that $\sigma := \varphi|_{\mathbf{free}(\mathbf{A})}$ is a substitution, since $\mathbf{free}(\mathbf{A})$ is finite. We have $\mathcal{I}^{\beta\eta}_\varphi(\mathbf{A}) = \sigma\mathbf{A}\downarrow_\beta$.
- ▷ The name term structure in the previous definition is justified by the following lemma.
- ▷ Lemma 4.45: *$\mathcal{T}_{\beta\eta}$ is a Σ-model*

We can see that $\alpha\beta\eta$-equality is complete for the class of Σ-models, i.e. if the equation $\mathbf{A} = \mathbf{B}$ is valid, then $\mathbf{A} =_{\alpha\beta\eta} \mathbf{B}$. Thus $\alpha\beta\eta$ equivalence fully characterizes equality in the class of all Σ-models, while additional η-equality characterizes functionality.

Completetness of $\alpha\beta\eta$-Equality

▷ Theorem 4.46: $\mathbf{A} = \mathbf{B}$ *is valid in the class of* Σ*-models, iff* $\mathbf{A} =_{\alpha\beta\eta} \mathbf{B}$.

▷ Proof: This is a simple consequence of the fact that $\mathcal{T}_{\beta\eta}$ is a Σ-model.

P.1 For closed equations the proof goes like this: if $\mathbf{A} = \mathbf{B}$ is valid in all Σ-models, it must be in $\mathcal{T}_{\beta\eta}$ and in particular $\mathbf{A}{\downarrow_\beta} = \mathcal{I}(\mathbf{A}) = \mathcal{I}(\mathbf{B}) = \mathbf{B}{\downarrow_\beta}$ and therefore $\mathbf{A} =_{\alpha\beta\eta} \mathbf{B}$.

P.2 If the equation is not closed, then the argument is a little more subtle due to the presence of free variables.

P.3 We extend Σ with constant c_α^i for each type α and each $i \in \mathbb{N}$.

P.4 Since we have assumed countably many variables per type, there is a bijection between the set of variables and the set of constants in Σ, which induces a variable assignment φ_Σ into $\mathit{cwff}_\mathcal{T}(\Sigma, \mathcal{V}_\mathcal{T}){\downarrow_\beta}$ (each variable X_α is mapped to its associated constant $c_\alpha^i \in \mathit{cwff}_\mathcal{T}(\Sigma, \mathcal{V}_\mathcal{T}){\downarrow_\beta}$).

P.5 Thus $\mathcal{I}_{\varphi_\Sigma}(\mathbf{A}) = \mathcal{I}_{\varphi_\Sigma}(\mathbf{B})$ is the long $\beta\eta$-normal form of $\varphi_\Sigma(\mathbf{A})$ and $\varphi_\Sigma(\mathbf{B})$.

P.6 Since φ_Σ is a structure preserving homomorphism on well-formed formulae, $\varphi_\Sigma^{-1}(\mathcal{I}_{\varphi_\Sigma}(\mathbf{A}))$ is the is the long $\beta\eta$-normal form of both $\mathbf{A}$ and $\mathbf{B}$ and thus $\mathbf{A} =_{\alpha\beta\eta} \mathbf{B}$. □

4.6 λ-Calculus Properties

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. On the other hand, it can be shown that $\alpha\beta\eta$-reduction is complete for this model class, i.e. if $\mathcal{I}_\varphi(\mathbf{A}) = \mathcal{I}_\varphi(\mathbf{B})$, for all $\mathcal{D}$ and φ, then $\mathbf{A} =_{\alpha\beta\eta} \mathbf{B}$.

4.7 The Curry-Howard Isomorphism

The Curry-Howard Iso. for minimal $\Rightarrow$-Logic

▷ make the structural similarity between λ-calculus and propositional ND explicit

- ▷ $\rightarrow$ resembles $\Rightarrow$
- ▷ types resemble formulae ("propositions as types")
- ▷ λ-terms resemble proofs ("proof terms"', "proofs as programs")
- ▷ wff:app resembles $\Rightarrow E$, wff:abs resembles $\Rightarrow I$

▷ $\mathbf{A}$ provable, iff α non-empty (e.g. for the Hilbert-axioms)

- ▷ $\lambda X_\alpha.\lambda Y_\beta.X_\alpha$ has type $\alpha \rightarrow \beta \rightarrow \alpha$
- ▷ $\lambda X_{\alpha\rightarrow\beta\rightarrow\gamma}.\lambda Y_{\alpha\rightarrow\gamma}.\lambda Z_\gamma.X Z\,(Y Z)$ has type $(\alpha \rightarrow \beta \rightarrow \gamma) \rightarrow (\alpha \rightarrow \beta) \rightarrow \alpha \rightarrow \gamma$

▷ works well for $\rightarrow$ and $\Rightarrow$

Types for Conjunctions

- new type constructor: $\times$ (product type $\alpha \times \beta$)
- new term constructors: $\langle\cdot,\cdot\rangle$, π_1 and π_2
- new type inference rules

$$\frac{\Gamma \vdash_\Sigma \mathbf{A}\colon \alpha \quad \Gamma \vdash_\Sigma \mathbf{B}\colon \beta}{\Gamma \vdash_\Sigma \langle \mathbf{A}, \mathbf{B}\rangle\colon \alpha \times \beta}\,\text{wff:pair} \qquad \frac{\Gamma \vdash_\Sigma \mathbf{A}\colon \alpha \times \beta}{\Gamma \vdash_\Sigma \pi_1(\mathbf{A})\colon \alpha}\,\text{wff:}\pi_l \qquad \frac{\Gamma \vdash_\Sigma \mathbf{A}\colon \alpha \times \beta}{\Gamma \vdash_\Sigma \pi_2(\mathbf{A})\colon \beta}\,\text{wff:}\pi_r$$

- new reductions (gives canonical reduction system)

$$(\pi_1(\langle \mathbf{A}, \mathbf{B}\rangle) \to^1_\beta \mathbf{A}) \qquad (\pi_2(\langle \mathbf{A}, \mathbf{B}\rangle) \to^1_\beta \mathbf{B}) \qquad (\langle \pi_1(\mathbf{A}), \pi_2(\mathbf{A})\rangle \to^1_\eta \mathbf{A})$$

- Others: disjoint sum for disjunction, complement for negation,...

Example (Conjunction)

$$\cfrac{\cfrac{\cfrac{[\mathbf{A} \wedge \mathbf{B}]^X}{\mathbf{B}}\wedge E_r \quad \cfrac{[\mathbf{A} \wedge \mathbf{B}]^X}{\mathbf{A}}\wedge E_l}{\mathbf{B} \wedge \mathbf{A}}\wedge I}{\mathbf{A} \wedge \mathbf{B} \Rightarrow \mathbf{B} \wedge \mathbf{A}} \Rightarrow I^X$$

corresponds to $\lambda X_{\alpha\times\beta}.\langle \pi_2(X), \pi_1(X)\rangle$

- Normalization

$$\cfrac{\cfrac{\begin{array}{c}\mathcal{D}\\ \overline{\overline{\mathbf{A}}}\end{array} \quad \begin{array}{c}\mathcal{E}\\ \overline{\overline{\mathbf{B}}}\end{array}}{\mathbf{A} \wedge \mathbf{B}}\wedge I}{\mathbf{A}}\wedge E_l \qquad \begin{array}{c}\mathcal{D}\\ \overline{\overline{\mathbf{A}}}\end{array}$$

since $(\pi_1(\langle \mathbf{M}, \mathbf{N}\rangle) \to^1_\beta \mathbf{M})$

- analogous for the other reductions

The Curry-Howard Isomorphism by Example

▷ Example 4.47: (Deriving the S-Axiom)

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma \vdash \quad \alpha \to \beta \to \gamma \quad \Gamma \vdash \quad \alpha \qquad \dfrac{\Gamma \vdash \quad \alpha \to \beta \quad \Gamma \vdash \quad \alpha}{\Gamma \vdash \quad \beta}}{\Gamma \vdash \quad \gamma}}{\alpha \to \beta \to \gamma\,, \quad \alpha \to \beta \vdash \quad \alpha \to \gamma}}{\alpha \to \beta \to \gamma \vdash \quad (\alpha \to \beta) \to \alpha \to \gamma}}{\emptyset \vdash \quad (\alpha \to \beta \to \gamma) \to (\alpha \to \beta) \to \alpha \to \gamma}$$

where $\Gamma = \quad \alpha \to \beta \to \gamma\,, \quad \alpha \to \beta\,, \quad \alpha$

The Curry-Howard Isomorphism by Example

▷ Example 4.48: (Deriving the S-Axiom)

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma \vdash_\Sigma X : \alpha \to \beta \to \gamma \quad \Gamma \vdash_\Sigma Z : \alpha \qquad \dfrac{\Gamma \vdash_\Sigma Y : \alpha \to \beta \quad \Gamma \vdash_\Sigma Z : \alpha}{\Gamma \vdash_\Sigma Y Z : \beta}}{\Gamma \vdash_\Sigma X Z (Y Z) : \gamma}}{[X : \alpha \to \beta \to \gamma], [Y : \alpha \to \beta] \vdash_\Sigma \lambda Z.X Z (Y Z) : \alpha \to \gamma}}{[X : \alpha \to \beta \to \gamma] \vdash_\Sigma \lambda Y Z.X Z (Y Z) : (\alpha \to \beta) \to \alpha \to \gamma}}{\emptyset \vdash_\Sigma \lambda X Y Z.X Z (Y Z) : (\alpha \to \beta \to \gamma) \to (\alpha \to \beta) \to \alpha \to \gamma}$$

where $\Gamma = [X : \alpha \to \beta \to \gamma], [Y : \alpha \to \beta], [Z : \alpha]$

What next for the Curry Howard Isomorphism?

- Additional types for disjunction and negation
- Problem: What about quantifiers?
- Idea 1: Introduce type polymorphism ("Second-Order λ-Calculus")
  - type-variables, type-quantification
  - $\lambda X.X$ has the type $\forall \alpha.\alpha \to \alpha$
- Idea 2: make types dependent on terms ("Edinburgh Logical Framework")
  - typ-constructors "type-families": $vec(7)$ is the type of vectors of length 7
  - $mat(3 \to 5) \to \vec{(3)} \to \vec{(5)}$ is the type of a matrix multiplication operator
  - $pf(\mathbf{A})$ the type of proofs of a formula $\mathbf{A}$.

Example: Proof Normalization

$$\dfrac{\dfrac{\mathcal{D}}{\mathbf{A}} \quad \dfrac{\dfrac{[\mathbf{A}]^X}{\mathbf{B}}\mathcal{E}}{\mathbf{A} \Rightarrow \mathbf{B}} \Rightarrow I^X}{\mathbf{B}} \Rightarrow E$$

- $\mathcal{E}$ corresponds to $\mathbf{M}_\mu$ ($\mathbf{B}$ to β)
- $\mathcal{D}$ corresponds to $\mathbf{N}_\nu$ ($\mathbf{A}$ to α)
- the whole to $(\lambda X_\alpha.\mathbf{M}_\mu)\mathbf{N}_\nu$

- Normalize: $((\lambda X_\alpha.\mathbf{M}_\mu)\mathbf{N}_\nu) \to_\beta^1 [\mathbf{N}/X](\mathbf{M})$ corresponds to $\dfrac{\dfrac{\mathcal{D}}{\mathbf{A}}}{\mathbf{B}}\mathcal{E}$

- Theorem 4.49: (Cut Elimination)
  For each ND-proof of a formula $\mathbf{A}$ there is a $\Rightarrow E$-free proof of $\mathbf{A}$.

λ-calculus and functional Programming

- z.B. LISP (λ-calculus with lists $[a|b|c]$ and conditionals)

```
sort=(λ (L) (if L=[]  then  []else [min(L)|sort(del(min(L),L))]))
del=(λ (X,L) (if L=[]  then  ⊥
             elif  X=first(L) then rest(L)
             else  [first(L)|del(X,rest(L))]))
min=(λ (L)  (if L=[] then  ⊥
             elif  L=[first(L)] then  first(L)
             elif first(L)<min(rest(L)) then  first(L)
             else  min(rest(L))))
```

- Program-Evaluation is β-reduction and list-reduction

```
del c [a|b|c]
        →β1 [first([a|b|c])|del(c,rest([a|b|c]))]→β1 [a|del(c,[b|c])]
        →β1 [a|first([b|c])|del(c,rest([b|c]))]→β1 [a|b|del(c,[c])]
        →β1 [a|b|rest([c])]→β1 [a|b|[]]→β1 [a|b]
```

Proofs as Programs

- Remember: Proofs are λ-terms & λ-terms are programs
- Idea: Then proofs should be programs (well only constructive ones)
- Example 4.50: (Sorting)

  a theory of ordered lists:

  - $\models perm(L, M)$, if M is a permutation of L
  - $\models ord(L)$, if L ordered wrt. $<$
  - $X < L$ if $X < Y$ for all $Y \in L$

  Theorems:

  - $\models min(L) < del(min(L), L)$
  - $\models ord(L) \wedge x < L \Rightarrow ord([x|L])$
  - $\models perm(L \Rightarrow M) \Rightarrow perm([x|L], [x|M])$

- Theorem 4.51: $\forall L.\exists M.ord(L)perm(L, M)$
- Proof: by induction on the structure of the list L

  P.1.1 If $L = []$: choose $M = []$ □

  P.1.2 If $L \neq []$:

  P.1.2.1 by IH there is a list W, such that $ord(W)perm(W, del(min(L), L))$

  P.1.2.2 chose $M = [min(L)|W]$ □

  □

  Programm:

- `sort=(λ L (if L=[] then ⊥ else [min(L)|sort(del(min(L),L))]))`
- Note: the correcness of this program is ensured by the proof
- Note: different proofs yield different programs
- Note: the programs extracted from automatically found proofs are not always efficient (Slowsort!)

 101 JACOBS UNIVERSITY

5 Knowledge Representation

Before we start into the development of description logics, we set the stage by looking into what alternatives for knowledge representation we know.

5.1 Introduction to Knowledge Representation

What is knowledge? Why Representation?

- For the purposes of this course: Knowledge is the information necessary to support intelligent reasoning (during NLP)

representation	can be used to determine
set of words	whether a word is admissible
list of words	the rank of a word
a lexicon	translation or grammatical function
structure	function

- Representation as structure and function.
  - the representation determines the content theory (what is the data?)
  - the function determines the process model (what do we do with the data?)

Knowledge Representation vs. Data Structures

- Why do we use the term "knowledge representation" rather than
  - data structures? (sets, lists, ... above)
  - information representation? (it is information)
- no good reason other than AI practice, with the intuition that
  - data is simple and general (supports many algorithms)
  - knowledge is complex (has distinguished process model)

Some Paradigms for AI/NLP

- GOFAI (good old-fashioned AI)
  - symbolic knowledge representation, process model based on heuristic search
- statistical, corpus-based approaches.
  - symbolic representation, process model based on machine learning
  - knowledge is divided into symbolic- and statistical (search) knowledge
- connectionist approach (not in this course)
  - sub-symbolic representation, process model based on primitive processing elements (nodes) and weighted links
  - knowledge is only present in activation patters, etc.

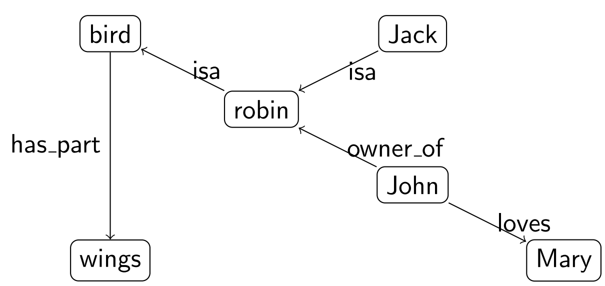
KR Approaches/Evaluation Criteria

- ▷ Expressive Adequacy: What can be represented, what distinctions are supported.
- ▷ Reasoning Efficiency: can the representation support processing that generates results in acceptable speed?
- ▷ Primitives: what are the primitive elements of representation, are they intuitive, cognitively adequate?
- ▷ Meta-representation: knowledge about knowledge
- ▷ Incompleteness: the problems of reasoning with knowledge that is known to be incomplete.

 105

Semantic Networks [e.g. Collins and Quillian '69]

- ▷ Graph structure with for representing knowledge
  - ▷ nodes represent concepts (e.g. bird, John, robin)
  - ▷ links represent relations between these (isa, father_of, belongs_to)

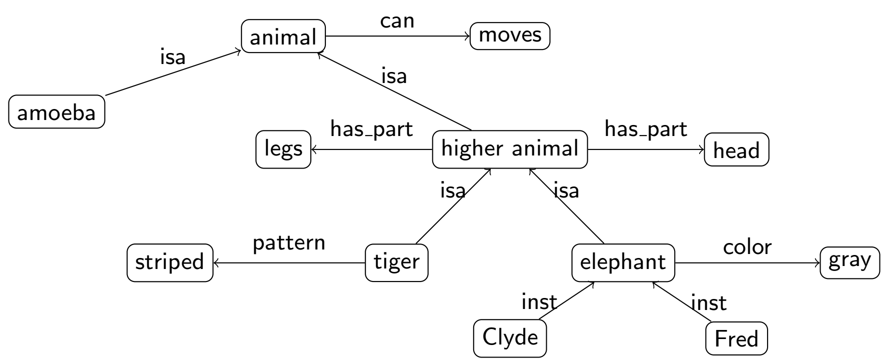


 106

The famous "Isa-Hierarchy"

▷ Idea: encode taxonomic information about concepts and individuals

- ▷ in "isa" links (inclusion of concepts)
- ▷ in "inst" links (concept memberships)
- ▷ use property inheritance in the process model

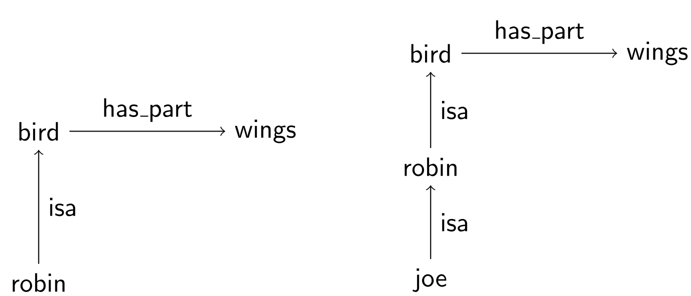


Limitations of Semantic Networks

▷ What is the meaning of a link?

- ▷ link names are very suggestive (misleading for humans)
- ▷ meaning of link types defined in the process model (no denotational semantics)

▷ No division of optional and defining arguments

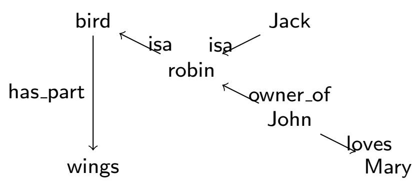


5

[e]EdNote: with a cancel link link to the has link

Another Notation for Semantic Networks

- ▷ use function/argument notation
  - ▷ Interpret nodes as arguments (reification to individuals)
  - ▷ Interpret links as functions (logical relations)



$isa(robin, bird)$
$haspart(bird, wings)$
$isa(Jack, robin)$
$owner\_of(John, robin)$
$loves(John, Mary)$

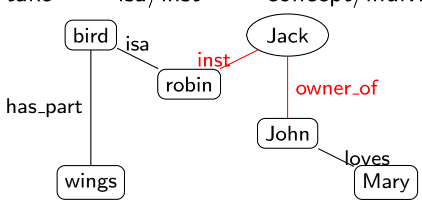
\+ linear notation (equivalent, but better to implement on a computer)

\+ easy to give process model by deduction (e.g. PROLOG)

– worse locality properties (networks are associative)

A Denotational Semantics for Semantic Networks

- ▷ take isa/inst concept/individual distinction into account



$robin \subseteq bird$
$haspart(bird, wings)$
$Jack \in robin$
$owner\_of(John, Jack)$
$loves(John, Mary)$

- ▷ looks like first-order logic, if we take
  - ▷ $A \subseteq B$ to mean $\forall X.A(X) \Rightarrow B(X)$
  - ▷ $a \in S$ to mean $S(a)$
  - ▷ $haspart(A, B)$ to mean $\forall X.A(X) \Rightarrow (\exists Y.B(Y) \wedge part\_of(X \wedge Y))$
- ▷ Take first-order deduction as process model (gives inheritance for free)

Frame Notation as Logic with Locality

▷ Predicate Logic: (where is the locality?)

$catch\_22 \in catch\_object$	There is an instance of catching
$catcher(catch\_22, jack\_2)$	Jack did the catching
$caught(catch\_22, ball\_5)$	He caught a certain ball

▷ Frame Notation (group everything around the object)

```
(catch_object  catch_22
               (catcher jack_2)
               (caught ball_5))
```

+ Once you have decided on a frame, all the information is local

+ easy to define schemes for concepts (aka. types in feature structures)

– how to determine frame, when to choose frame (log/chair)

KR involving Time (Scripts [Shank '77])

▷ Idea: organize typical event sequences, actors and props into representation structure

▷ Example 5.1: getting your hair cut (at a beauty parlor)

▷ props, actors as script variables

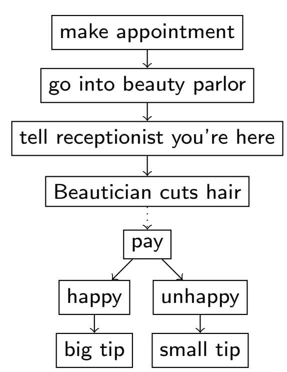
▷ events in a (generalized) sequence

▷ use script material for

▷ anaphors, bridging references

▷ default common ground

▷ to fill in missing material into situations



Other Representation Formats (not covered)

▷ Procedural Representations (production systems)

▷ analogical representations (interesting but not here)

▷ iconic representations (interesting but very difficult to formalize)

▷ If you are interested, come see me off-line

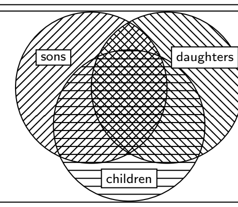
5.2 Logic-Based Knowledge Representation

Logic-Based Knowledge Representation

- Logic (and related formalisms) have a well-defined semantics
  - explicitly (gives more understanding than statistical/neural methods)
  - transparently (symbolic methods are monotonic)
  - systematically (we can prove theorems about our systems)
- Problems with logic-based approaches
  - Where does the world knowledge come from? (Ontology problem)
  - How to guide search induced by log. calculi (combinatorial explosion)
- One possible answer: Description Logics. (next couple of times)

Propositional Logic as Set Description Language

- use propositional logic as a set description language
- variant syntax: $\sqcap \hat{=} \wedge$ (intersection), $\sqcup \hat{=} \vee$ (union), $\overline{\cdot} \hat{=} \neg$ (complement), $\sqsubseteq \hat{=} \Rightarrow$ (subsumption)

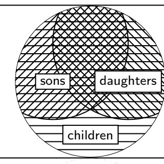
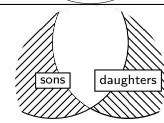
Example	Set Semantics
son $\sqsubseteq$ child daughter $\sqsubseteq$ child $\overline{\text{son} \sqcap \text{daughter}}$ child $\sqsubseteq$ son $\sqcup$ daughter	

- Definition 5.2: (Formal Semantics)
  let $\mathcal{D}$ be a given set (called the domain) and $\varphi\colon \mathcal{V}_o \to \wp(\mathcal{D})$, then
  - $[\![P]\!] := \varphi(P) \subseteq \mathcal{D}$,
  - $[\![\mathbf{A} \sqcup \mathbf{B}]\!] = [\![\mathbf{A}]\!] \cup [\![\mathbf{B}]\!]$ and $[\![\overline{\mathbf{A}}]\!] = \mathcal{D} \backslash [\![\mathbf{A}]\!] \ldots$

Effects of the Axioms in this example

▷ Idea: use logical axioms to describe the world (Axioms restrict the class of admissible domain structures)

▷

Axioms	Effect
son $\sqsubseteq$ child daughter $\sqsubseteq$ child	
$\overline{\text{son} \sqcap \text{daughter}}$ child $\sqsubseteq$ son $\sqcup$ daughter	

Predicate-Logic Formulation

Propositional Logic	Predicate Logic
son $\sqsubseteq$ child	$\forall x.\text{son}(x) \Rightarrow \text{child}(x)$
daughter $\sqsubseteq$ child	$\forall x.\text{daughter}(x) \Rightarrow \text{child}(x)$
$\overline{\text{son} \sqcap \text{daughter}}$	$\forall x.\neg(\text{son}(x) \wedge \text{daughter}(x))$
child $\sqsubseteq$ son $\sqcup$ daughter	$\forall x.\text{child}(x) \Rightarrow \text{son}(x) \vee \text{daughter}(x)$

Set-Theoretic Semantics

▷ Definition 5.3: A model $\mathcal{M} = \langle \mathcal{D}, \mathcal{I} \rangle$ consists of a Interpretation $\mathcal{I}$ over a non-empty domain $\mathcal{D}$ is a mapping $[\![\cdot]\!]$:

	Operator Meaning			formula semantics		
1				$[\![p]\!]$	$\subset$	$\mathcal{D}$
2	$[\![\overline{\cdot}]\!]$	=	complement	$[\![\overline{\mathbf{A}}]\!]$	$=$	$\overline{[\![\mathbf{A}]\!]} := =\mathcal{D}\backslash[\![\mathbf{A}]\!]$
3	$[\![\sqcap]\!]$	=	$\cap$	$[\![\mathbf{A} \sqcap \mathbf{B}]\!]$	$=$	$[\![\mathbf{A}]\!] \cap [\![\mathbf{B}]\!]$
4	$[\![\sqcup]\!]$	=	$\cup$	$[\![\mathbf{A} \sqcup \mathbf{B}]\!]$	$=$	$[\![\mathbf{A}]\!] \cup [\![\mathbf{B}]\!]$
5	$[\![\sqsubseteq]\!]$	=	$\subseteq$	$[\![\mathbf{A} \sqsubseteq \mathbf{B}]\!]$	$=$	$\overline{[\![\mathbf{A}]\!]} \cup [\![\mathbf{B}]\!]$
6	$[\![\equiv]\!]$	=	set equality	$[\![\mathbf{A} \equiv \mathbf{B}]\!]$	$=$	$([\![\mathbf{A}]\!] \cap [\![\mathbf{B}]\!]) \cup \overline{(\overline{[\![\mathbf{A}]\!]} \cup [\![\mathbf{B}]\!])}$

▷ Justification for 5: $\mathbf{A} \Rightarrow \mathbf{B} = \neg\mathbf{A} \vee \mathbf{B}$

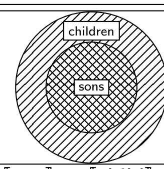
▷ Justification for 6: $\mathbf{A} \Leftrightarrow \mathbf{B} = \mathbf{A} \wedge \mathbf{B} \vee \neg\mathbf{A} \wedge \neg\mathbf{B} = \mathbf{A} \wedge \mathbf{B} \vee \neg(\mathbf{A} \vee \mathbf{B})$

Set-Theoretic Semantics of Axioms

- ▷ Set-Theoretic Semantics of 'true' and 'false' $(\top = \varphi \sqcup \overline{\varphi} \quad \bot = \varphi \sqcap \overline{\varphi})$

$$[\![\bot]\!] = [\![p]\!] \cup [\![\overline{p}]\!] = [\![p]\!] \cup \overline{[\![p]\!]} = \mathcal{D} \qquad \text{Analogously: } [\![\bot]\!] = \emptyset$$

- ▷ Set-Theoretic Semantics of Axioms: $\mathbf{A}$ is true in $\mathcal{M} = \langle \mathcal{D}, \mathcal{I} \rangle$, iff $[\![\mathbf{A}]\!] = \mathcal{D}$

Axioms	Semantics
son $\sqsubseteq$ child is true iff $\overline{[\![\text{son}]\!]} \cup [\![\text{child}]\!] = \mathcal{D}$ iff $[\![\text{son}]\!] \subseteq [\![\text{child}]\!]$	
son $\sqsubseteq$ child	$[\![\text{son}]\!] \subseteq [\![\text{child}]\!]$
daughter $\sqsubseteq$ child	$\overline{[\![\text{daughter}]\!]} \subseteq [\![\text{child}]\!]$
$\overline{\text{son} \sqcap \text{daughter}}$	$\overline{[\![\text{son}]\!] \cap [\![\text{daughter}]\!]} = \mathcal{D}$
child $\sqsubseteq$ son $\sqcup$ daughter	$[\![\text{child}]\!] \subseteq [\![\text{son}]\!] \cup [\![\text{daughter}]\!]$

 119 JACOBS UNIVERSITY

Set-Theoretic Semantics and Predicate Logic

- ▷ use logical operators $\sqcap, \sqcup, \sqsubseteq, \equiv$ instead of $\wedge, \vee, \Rightarrow, \Leftrightarrow$ if we are using PL0 with set-theoretic semantics.
- ▷ Translation into PL1
  - ▷ recursively add argument variable x
  - ▷ change back $\sqcap, \sqcup, \sqsubseteq, \equiv$ to $\wedge, \vee, \Rightarrow, \Leftrightarrow$
  - ▷ universal closure for x at formula level.

$\overline{\mathrm{PL0}}^{fo(x)} = \mathrm{PL1}$	Comment
$\overline{\overline{p}}^{fo(x)} = p(x)$	
$\overline{(\overline{\mathbf{A}})}^{fo(x)} = \neg\overline{\mathbf{A}}^{fo(x)}$	
$\overline{(\mathbf{A} \sqcap \mathbf{B}}^{fo(x)}) = \overline{\mathbf{A}}^{fo(x)} \wedge \overline{\mathbf{B}}^{fo(x)}$	$\wedge$ vs. $\sqcap$
$\overline{(\mathbf{A} \sqcup \mathbf{B}}^{fo(x)} = \overline{\mathbf{A}}^{fo(x)} \vee \overline{\mathbf{B}}^{fo(x)}$	$\vee$ vs. $\sqcup$
$\overline{(\mathbf{A} \sqsubseteq \mathbf{B})}^{fo(x)} = \overline{\mathbf{A}}^{fo(x)} \Rightarrow \overline{\mathbf{B}}^{fo(x)}$	$\Rightarrow$ vs. $\sqsubseteq$
$\overline{(\mathbf{A} = \mathbf{B})}^{fo(x)} = \overline{\mathbf{A}}^{fo(x)} \Leftrightarrow \overline{\mathbf{B}}^{fo(x)}$	$\Leftrightarrow$ vs. $=$
$\overline{\mathbf{A}}^{fo} = \forall x.\overline{\mathbf{A}}^{fo(x)}$	for formulae

 120 JACOBS UNIVERSITY

Translation Examples

▷ Example 5.4:

$$\begin{aligned}
\overline{\mathsf{son} \sqsubseteq \mathsf{child}}^{fo} &= \forall x.\mathsf{son}(x) \Rightarrow \mathsf{child}(x)\\
\overline{\mathsf{daughter} \sqsubseteq \mathsf{child}}^{fo} &= \forall x.\mathsf{daughter}(x) \Rightarrow \mathsf{child}(x)\\
\overline{\overline{(\mathsf{son} \sqsubseteq \mathsf{daughter})}}^{fo} &= \forall x.\mathsf{son}(x) \wedge \overline{\mathsf{daughter}(x)}\\
\overline{\mathsf{child} \sqsubseteq \mathsf{son} \sqcup \mathsf{daughter}}^{fo} &= \forall x.\mathsf{child}(x) \Rightarrow \mathsf{son}(x) \vee \mathsf{daughter}(x)
\end{aligned}$$

▷ What are the advantages of translation to PL1?

- ▹ theoretically: A better understanding of the semantics
- ▹ computationally: NOTHING
  many tests are decidable for PL0, but not for PL1 (Description Logics?)

Kinds of Inference in Description Logics

▷ Consistency test (is a concept definition satisfiable?)

▷ Subsumption test (does a concept subsume another?)

▷ Instance test (is an individual an example of a concept?)

▷ ...

▷ Problem: decidability, complexity, algorithm

Consistency Test

▷ Example 5.5: T-Box

woman	=	person ⊓ $\overline{\text{has\_Y}}$	person without y-chromosome
man	=	person ⊓ has_Y	person with y-chromosome
hermaphrodite	=	man ⊓ woman	man and woman

▷ This specification is inconsistent, i.e. $[\![\mathsf{hermaphrodite}]\!] = \emptyset$ for all $\mathcal{D}$, φ.

▷ Algorithm: propositional satisfiability test (NP-complete) we know how to do this, e.g. tableau, resolution

Subsumption Test

▷ Example 5.6: in this case trivial

Axioms	entailed subsumption relation
woman $=$ person $\sqcap \overline{\text{has\_Y}}$	woman $\sqsubseteq$ person
man $=$ person $\sqcap$ has_Y	man $\sqsubseteq$ person

▷ Reduction to consistency test: (need to implement only one) $Axioms \Rightarrow \mathbf{A} \Rightarrow \mathbf{B}$ is valid iff $Axioms \wedge \mathbf{A} \wedge \neg\mathbf{B}$ is inconsistent.

▷ Definition 5.7: $\mathbf{A}$ subsumes $\mathbf{B}$ (modulo an axiom set $\mathcal{A}$)
iff $[\![\mathbf{B}]\!] \subseteq [\![\mathbf{A}]\!]$ for all interpretations $\mathcal{D}$, that satisfy $\mathcal{A}$
iff $Axioms \Rightarrow \mathbf{B} \Rightarrow \mathbf{A}$ is valid ‘

▷ in our example: person subsumes woman and man

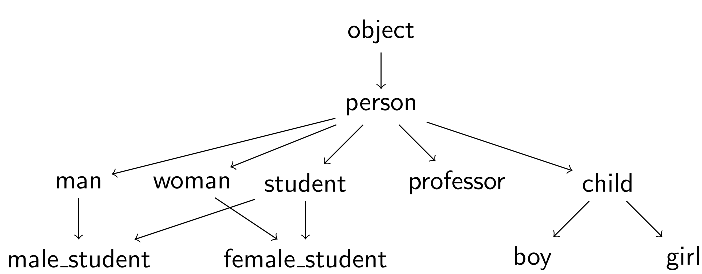
Classification

▷ The subsumption relation among all concepts (subsumption graph)

▷ Visualization of the Subsumption graph for inspection (plausibility)

▷ Definition 5.8: Classification is the computation of the subsumption graph

▷ Example 5.9: (not always so trivial)



Instance Test

▷ Example 5.10: (will explain TBox and ABox with ALC later)

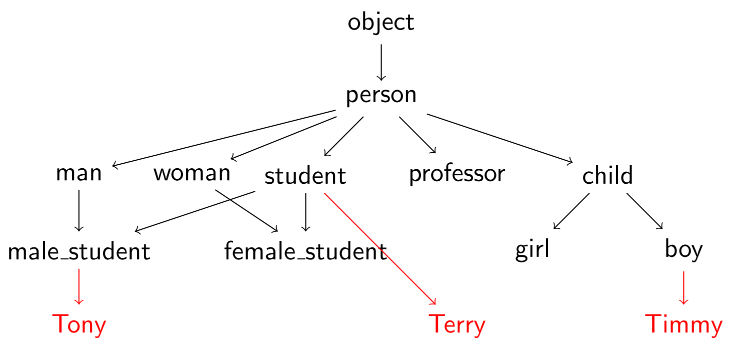
T-Box (terminological Box)	A-Box (assertional Box, data base)	
woman $=$ person $\sqcap \overline{\text{has\_Y}}$	tony: person	Paul is a person
man $=$ person $\sqcap$ has_Y	tony: has_Y	Paul has a y-chromosome

▷ This entails: tony: man (Paul is a man).

Realization

▷ Definition 5.11: Realization is the computation of all instance relations between ABox objects and TBox concepts.

▷ sufficient to remember the lowest concepts in the subsumption graph



▷ if tony: male_student is known, we do not need tony: man.

5.3 A simple Description Logic: ALC

Motivation for $\mathcal{ALC}$ (Prototype Description Logic)

▷ Propositional logic (PL0) is not expressive enough

▷ Example 5.12: "mothers are women that have a child"

▷ Reason: there are no quantifiers in PL0 (existential ($\exists$) and universal ($\forall$))

▷ Idea: use first-order predicate logic (PL1)

$$\forall x.mother(x) \Leftrightarrow woman(x) \wedge (\exists y.has\_child(x, y))$$

▷ Problem: complex algorithms, non-termination (PL1 is too expressive)

▷ Idea: Try to travel the middle ground
more expressive than PL0 (quantifiers) but weaker than PL1 (still tractable)

▷ Technique: Allow only "restricted quantification", where quantified variables only range over values that can be reached via a binary relation like has_child.

Syntax of $\mathcal{ALC}$

- Concepts: (aka. "predicates" in PL1 or "propositional variables" in PL0)
  concepts in DLs name classes of objects like in OOP.
- Special concepts: $\top$ (for "true" or "all") and $\bot$ (for "false" or "none").
- Example 5.13: person, woman, man, mother, professor, student, car, BMW, computer, computer program, heart attack risk, furniture, table, leg of a chair,...
- Roles: name binary relations (like in in PL1)
- Example 5.14: has_child, has_son, has_daughter, loves, hates gives_course, executes_computer_program, has_leg_of_table, has_wheel, has_motor,...

Syntax of $\mathcal{ALC}$: Formulae $F_{\mathcal{ALC}}$

- Grammar: $F_{\mathcal{ALC}} ::= C \mid \top \mid \bot \mid \overline{F_{\mathcal{ALC}}} \mid F_{\mathcal{ALC}} \sqcap F_{\mathcal{ALC}} \mid F_{\mathcal{ALC}} \sqcup F_{\mathcal{ALC}} \mid (\exists \mathrm{R}.F_{\mathcal{ALC}}) \mid (\forall \mathrm{R}.F_{\mathcal{ALC}})$
- Example 5.15:
  - person $\sqcap$ ($\exists$has_child.student) (parents of students)
    (The set of persons that have a child which is a student)
  - person $\sqcap$ ($\exists$has_child.$\exists$has_child.student) (grandparents of students)
  - person $\sqcap$ ($\exists$has_child.$\exists$has_child.student $\sqcup$ teacher)(grandparents of students or teachers)
  - person $\sqcap$ ($\forall$has_child.student) (parents whose children are all students)
  - person $\sqcap$ ($\forall$haschild.$\exists$has_child.student) (grandparents, whose children all have at least one child that is a student)

More Examples

- car $\sqcap$ ($\exists$has_part.$\exists$made_in.$\overline{\text{EU}}$)(cars that have at least one part that has not been made in the EU)
- student $\sqcap$ ($\forall$audits_course.graduatelevelcourse)(students, that only audit graduate level courses)
- house $\sqcap$ ($\forall$has_parking.off_street) (houses with off-street parking)
- Note: $p \sqsubseteq q$ can still be used as an abbreviation for $\overline{p} \sqcup q$.
- student $\sqcap$ ($\forall$audits_course.($\exists$hasrecitation.$\top$) $\sqsubseteq$ ($\forall$has_TA.woman))
  (students that only audit courses that either have no recitation or recitations that are TAed by women)

$\mathcal{ALC}$ Concept Definitions

- ▷ Define new concepts from known ones: $(KD_{\mathcal{ALC}} :== C = F_{\mathcal{ALC}})$

		Definition	rec
man	=	person ⊓ (∃has_chrom.Y_chrom)	−
woman	=	person ⊓ (∀has_chrom.$\overline{\text{Y\_chrom}}$)	−
mother	=	woman ⊓ (∃has_child.person)	−
father	=	man ⊓ (∃has_child.person)	−
grandparent	=	person ⊓ (∃has_child.mother ⊔ father)	−
german	=	person ⊓ (∃has_parents.german)	+
number_list	=	empty_list ⊔ (∃is_first.number) ⊓ (∃is_rest.number_list)	+

Concept Axioms

- ▷ Definition 5.16: General DL formulae that are not concept definitions are called Concept Axioms.
- ▷ They normally contain additional information about concepts
- ▷ Example 5.17:
  - ▷ $\overline{\text{person} \sqcap \text{car}}$ (persons and cars are disjoint)
  - ▷ car ⊑ motor_vehicle (cars are motor vehicles)
  - ▷ motor_vehicle ⊑ car ⊔ truck ⊔ motorcycle(motor vehicles are cars, trucks, or motorcycles)

TBoxes: "terminological Box"

- ▷ Definition 5.18: finite set of concept definitions + finite set of concept axioms
- ▷ Definition 5.19: Acyclic TBox (mostly treated)
  TBox does not contain recursive definitions
- ▷ Definition 5.20: Normalized wrt. TBox (convenient)
  A formula $\mathbf{A}$ is called normalized wrt. $\mathcal{T}$, iff it does not contain concept names defined in $\mathcal{T}$.
- ▷ Algorithm: (Input: A formula $\mathbf{A}$ and a TBox $\mathcal{T}$.) (for arbitrary DLs)
  - ▷ While [$\mathbf{A}$ contains concept name c and $\mathcal{T}$ concept definition $c = \mathbf{C}$]
    - ▷ substitute c by $\mathbf{C}$ in $\mathbf{A}$.
- ▷ Lemma 5.21: *this algorithm terminates for acyclic TBoxes*

Normalization Example (normalizing grandparent)

grandparent
$\mapsto$ person $\sqcap$ ($\exists$has_child.mother $\sqcup$ father)
$\mapsto$ person $\sqcap$ ($\exists$has_child.woman $\sqcap$ ($\exists$has_child.person), man, $\exists$has_child.person)
$\mapsto$ person $\sqcap$ ($\exists$has_child.person $\sqcap$ ($\exists$has_chrom.Y_chrom) $\sqcap$ ($\exists$has_child.person) $\sqcap$ person $\sqcap$ ($\exists$has_chrom.Y_chrom) $\sqcap$ ($\exists$has_child.person))

- ▷ Observation: normalization result can be exponential and redundant
- ▷ Observation: need not terminate on cyclic TBoxes

german $\mapsto$ person $\sqcap$ ($\exists$has_parents.german)
$\mapsto$ person $\sqcap$ ($\exists$has_parents.person $\sqcap$ ($\exists$has_parents.german))
$\mapsto$ ...

Semantics of $\mathcal{ALC}$

- ▷ $\mathcal{ALC}$ semantics is an extension of the set-semantics of propositional logic.
- ▷ Definition 5.22: An Interpretation $\mathcal{I}$ over a non-empty domain $\mathcal{D}$ is a mapping $[\![\cdot]\!]$:

Op.	formula semantics		
	$[\![c]\!] \subseteq \mathcal{D} = [\![\top]\!]$	$[\![\bot]\!] = \emptyset$	$[\![r]\!] \subseteq \mathcal{D} \times \mathcal{D}$
$\overline{\cdot}$	$[\![\overline{\varphi}]\!]$	$=$	$\overline{[\![\varphi]\!]} = \mathcal{D}\backslash[\![\varphi]\!]$
$\sqcap$	$[\![\varphi \sqcap \psi]\!]$	$=$	$[\![\varphi]\!] \cap [\![\psi]\!]$
$\sqcup$	$[\![\varphi \sqcup \psi]\!]$	$=$	$[\![\varphi]\!] \cup [\![\psi]\!]$
$\exists$R.	$[\![\exists \mathrm{R}.\varphi]\!]$	$=$	$(\{x \in \mathcal{D} \mid \exists y.\langle x, y\rangle \in [\![\mathrm{R}]\!]$ and $y \in [\![\varphi]\!]\})$
$\forall$R.	$[\![\forall \mathrm{R}.\varphi]\!]$	$=$	$(\{x \in \mathcal{D} \mid \forall y.$ if $\langle x, y\rangle \in [\![\mathrm{R}]\!]$ then $y \in [\![\varphi]\!]\})$

Propositional Identities

Name	for $\sqcap$	for $\sqcup$
Idenpot.	$\varphi \sqcap \varphi = \varphi$	$\varphi \sqcup \varphi = \varphi$
Identity	$\varphi \sqcap \top = \varphi$	$\varphi \sqcup \bot = \varphi$
Absorpt.	$\varphi \sqcup \top = \top$	$\varphi \sqcap \bot = \bot$
Commut.	$\varphi \sqcap \psi = \psi \sqcap \varphi$	$\varphi \sqcup \psi = \psi \sqcup \varphi$
Assoc.	$\varphi \sqcap (\psi \sqcap \theta) = (\varphi \sqcap \psi) \sqcap \theta$	$\varphi\psi \sqcup \theta = (\varphi \sqcup \psi) \sqcup \theta$
Distrib.	$\varphi \sqcap (\psi \sqcup \theta) = \varphi \sqcap \psi \sqcup \varphi \sqcap \theta$	$\varphi \sqcup \psi \sqcap \theta = (\varphi \sqcup \psi) \sqcap (\varphi \sqcup \theta)$
Absorpt.	$\varphi \sqcap (\varphi \sqcup \theta) = \varphi$	$\varphi \sqcup \varphi \sqcap \theta = \varphi \sqcap \theta$
Morgan	$\overline{\varphi \sqcap \psi} = \overline{\varphi} \sqcup \overline{\psi}$	$\overline{\varphi \sqcup \psi} = \overline{\varphi} \sqcap \overline{\psi}$
dneg	$\overline{\overline{\varphi}} = \varphi$	

More $\mathcal{ALC}$ Identities

▷

$\overline{\exists \mathsf{R}.\varphi}$	$=$	$\forall \mathsf{R}.\overline{\varphi}$	$\overline{\forall \mathsf{R}.\varphi}$	$=$	$\exists \mathsf{R}.\overline{\varphi}$
$\forall \mathsf{R}.\varphi \sqcap \psi$	$=$	$\forall \mathsf{R}.\varphi \sqcap (\forall \mathsf{R}.\psi)$	$\exists \mathsf{R}.\varphi \sqcup \psi$	$=$	$\exists \mathsf{R}.\varphi \sqcup (\exists \mathsf{R}.\psi)$

▷ Proof of 1

$$\begin{aligned}
[\![\overline{\exists \mathsf{R}.\varphi}]\!] = \mathcal{D}\backslash[\![(\exists \mathsf{R}.\varphi)]\!] &= \mathcal{D}\backslash(\{x \in \mathcal{D} \mid \exists y.\langle x,y\rangle \in [\![\mathsf{R}]\!] \text{ and } y \in [\![\varphi]\!]\})\\
&= (\{x \in \mathcal{D} \mid \text{not } \exists y.\langle x,y\rangle \in [\![\mathsf{R}]\!] \text{ and } y \in [\![\varphi]\!]\})\\
&= (\{x \in \mathcal{D} \mid \forall y.\text{if } \langle x,y\rangle \in [\![\mathsf{R}]\!] \text{ then } y \notin [\![\varphi]\!]\})\\
&= (\{x \in \mathcal{D} \mid \forall y.\text{if } \langle x,y\rangle \in [\![\mathsf{R}]\!] \text{ then } y \in (\mathcal{D}\backslash[\![\varphi]\!])\})\\
&= (\{x \in \mathcal{D} \mid \forall y.\text{if } \langle x,y\rangle \in [\![\mathsf{R}]\!] \text{ then } y \in [\![\overline{\varphi}]\!]\})\\
&= [\![\forall \mathsf{R}.\overline{\varphi}]\!]
\end{aligned}$$

Negation Normal Form

▷ Definition 5.23: (NNF)

$\overline{\cdot}$ directly in front of concept names in $\mathcal{ALC}$ formulae

▷ Use the $\mathcal{ALC}$ rules to compute it. (in linear time)

example	by rule
$\overline{\exists \mathsf{R}.(\forall \mathsf{S}.e) \sqcap \overline{(\forall \mathsf{S}.d)}}$	
$\mapsto \forall \mathsf{R}.\overline{(\forall \mathsf{S}.e) \sqcap \overline{(\forall \mathsf{S}.d)}}$	$\overline{\exists \mathsf{R}.\varphi} \mapsto \forall \mathsf{R}.\overline{\varphi}$
$\mapsto \forall \mathsf{R}.\overline{(\forall \mathsf{S}.e)} \sqcup \overline{\overline{(\forall \mathsf{S}.d)}}$	$\overline{\varphi \sqcap \psi} \mapsto \overline{\varphi} \sqcup \overline{\psi}$
$\mapsto \forall \mathsf{R}.(\exists \mathsf{S}.\overline{e}) \sqcup \overline{\overline{(\forall \mathsf{S}.d)}}$	$\overline{\forall \mathsf{R}.\varphi} \mapsto \forall \mathsf{R}.\overline{\varphi}$
$\mapsto \forall \mathsf{R}.(\exists \mathsf{S}.\overline{e}) \sqcup (\forall \mathsf{S}.d)$	$\overline{\overline{\varphi}} \mapsto \varphi$

$\mathcal{T}_{ALC}$: A Tableau-Calculus for $\mathcal{ALC}$

$$\frac{\begin{array}{c}x\colon c\\ x\colon \overline{c}\end{array}}{*}\bot \qquad \frac{x\colon \varphi \sqcap \psi}{\begin{array}{c}x\colon \varphi\\ x\colon \psi\end{array}}\sqcap \qquad \frac{x\colon \varphi \sqcup \psi}{x\colon \varphi \mid x\colon \psi}\sqcup \qquad \frac{\begin{array}{c}x\colon \forall \mathsf{R}.\varphi\\ x\ \mathsf{R}\ y\end{array}}{y\colon \varphi}\forall \qquad \frac{x\colon \exists \mathsf{R}.\varphi}{\begin{array}{c}x\ \mathsf{R}\ y\\ y\colon \varphi\end{array}}\exists$$

▷ tableau calculus acts constraints of the form

- ▷ $x\colon \varphi$ (x variable and $\varphi \in \mathcal{ALC}$) ($x$ is in the set φ)
- ▷ $x\ \mathsf{R}\ y$, (x, y variables, and R role name) (x and y are in relation R)

Examples

1	$x\colon \forall \mathsf{has\_child}.\mathsf{man} \sqcap \exists \mathsf{has\_child}.\overline{\mathsf{man}}$	initial	$x\colon \forall \mathsf{has\_child}.\mathsf{man} \sqcap \exists \mathsf{has\_child}.\mathsf{man}$	initial
2	$x\colon \forall \mathsf{has\_child}.\mathsf{man}$	$\sqcap_l$	$x\colon \forall \mathsf{has\_child}.\mathsf{man}$	$\sqcap_l$
3	$x\colon \exists \mathsf{has\_child}.\overline{\mathsf{man}}$	$\sqcap_r$	$x\colon \exists \mathsf{has\_child}.\mathsf{man}$	$\sqcap_r$
4	x has_child y	$\exists_r$	x has_child y	$\exists_r$
5	$y\colon \overline{\mathsf{man}}$	$\exists_s$	$y\colon \mathsf{man}$	$\exists_s$
6	$y\colon \mathsf{man}$	$\forall$	open	
7	$*$	$\bot$		
	inconsistent			

The right tableau has a model: there are two persons, x and y. y is the only child of x, y is a man

Another Example

1	$x\colon (\forall \mathsf{has\_child}.\mathsf{ugrad} \sqcup \mathsf{grad}) \sqcap (\exists \mathsf{has\_child}.\overline{\mathsf{ugrad}})$	
2	$x\colon \forall \mathsf{has\_child}.\mathsf{ugrad} \sqcup \mathsf{grad}$	$\sqcap_l$
3	$x\colon \exists \mathsf{has\_child}.\overline{\mathsf{ugrad}}$	$\sqcap_r$
4	x has_child y	$\exists_s$
5	$y\colon \overline{\mathsf{ugrad}}$	$\exists_r$
6	$y\colon \mathsf{ugrad} \sqcup \mathsf{grad}$	$\forall$
7	$y\colon \mathsf{ugrad}$ $y\colon \mathsf{grad}$	$\sqcup$
8	$*$ open	

The left branch is closed, the right one represents a model: y is a child of x, y is a graduate student, x hat exactly one child: y.

Properties of Tableau Calculi

- ▷ We study the following properties of a tableau calculus $\mathcal{C}$:

  Termination there are no infinite sequences of rule applications.

  Correctness If φ is consistent, then $\mathcal{C}$ terminates with an open branch.

  Completeness If φ is in consistent, then $\mathcal{C}$ terminates and all branches are closed.

  Complexity of the algorithm

  Complexity of the satisfiability itself

Termination

- ▷ Theorem 5.24: *The Tableau Algorithm for ALC terminates* To prove termination of a tableau algorithm, find a well-founded measure (function) that is decreased by all rules
- ▷ Proof: Sketch (full proof very technical)

P.1 any rule except $\forall$ can only be applied once to $x\colon \psi$.

P.2 rule $\forall$ applicable to $x\colon \forall \mathrm{R}.\psi$ at most as the number of R-successors of x. (those y with x R y above)

P.3 the R-successors are generated by $x\colon \exists \mathrm{R}.\theta$ above, (number bounded by size of input formula)

P.4 every rule application to $x\colon \psi$ generates constraints $z\colon \psi'$, where ψ' a proper sub-formula of ψ. □

Correctness

- ▷ Lemma 5.25: *If φ consistent, then $\mathcal{T}$ terminates on $x\colon \varphi$ with open branch.*
- ▷ Proof: Let $\mathcal{M}$ be a model for φ and $w \in [\![\varphi]\!]$.

P.1 we define $[\![x]\!] := w$ and
$$\begin{array}{lll} \Im \models x\colon \psi & \text{iff} & [\![x]\!] \in [\![\psi]\!] \\ \Im \models x \mathrel{\mathrm{R}} y & \text{iff} & \langle x, y\rangle \in [\![\mathrm{R}]\!] \\ \Im \models S & \text{iff} & \Im \models c \text{ for all } c \in S \end{array}$$

P.2 This gives us $\Im \models x\colon \varphi$ (base case)

P.3 case analysis: if branch consistent, then either

- ▷ no rule applicable to leaf (open branch)
- ▷ or rule applicable and one new branch satisfiable (green inductive case)

P.4 consequence: there must be an open branch (by termination) □

Case analysis on the rules

$\sqcap$ **applies** , then $\Im \models x\colon \varphi \sqcap \psi$, i.e. $[\![x]\!] \in [\![(\varphi \sqcap \psi)]\!]$
so $[\![x]\!] \in [\![\varphi]\!]$ and $[\![x]\!] \in [\![\psi]\!]$, thus $\Im \models x\colon \varphi$ and $\Im \models x\colon \psi$.

$\sqcup$ **applies** , then $\Im \models x\colon \varphi \sqcup \psi$, i.e $[\![x]\!] \in [\![(\varphi \sqcup \psi)]\!]$
so $[\![x]\!] \in [\![\varphi]\!]$ or $[\![x]\!] \in [\![\psi]\!]$, thus $\Im \models x\colon \varphi$ or $\Im \models x\colon \psi$,
wlog. $\Im \models x\colon \varphi$.

$\forall$ **applies** , then $\Im \models x\colon \forall \mathrm{R}.\varphi$ and $\Im \models x \mathrel{\mathrm{R}} y$, i.e. $[\![x]\!] \in [\![(\forall \mathrm{R}.\varphi)]\!]$ and $\langle x, y\rangle \in [\![\mathrm{R}]\!]$, so $[\![y]\!] \in [\![\varphi]\!]$

$\exists$ **applies** , then $\Im \models x\colon \exists \mathrm{R}.\varphi$, i.e $[\![x]\!] \in [\![(\exists \mathrm{R}.\varphi)]\!]$,
so there is a $v \in D$ with $\langle [\![x]\!], v\rangle \in [\![\mathrm{R}]\!]$ and $v \in [\![\varphi]\!]$.
We define $[\![y]\!] := v$, then $\Im \models x \mathrel{\mathrm{R}} y$ and $\Im \models y\colon \varphi$

Completeness of the Tableau Calculus

▷ lemma 5.26: *Open saturated tableau branches for φ induce models for φ.*

▷ Proof: construct a model for the branch and verify for φ

P.1 (Model Construction) Let $\mathcal{B}$ be an open saturated branch

▷ we define
$$\begin{aligned} D &:= (\{x \mid x\colon \psi \in \mathcal{B} \text{ or } z \mathrel{\mathsf{R}} x \in \mathcal{B}\}) \\ [\![c]\!] &:= (\{x \mid x\colon c \in \mathcal{B}\}) \\ [\![\mathsf{R}]\!] &:= (\{\langle x, y\rangle \mid x \mathrel{\mathsf{R}} y \in S_n\}) \end{aligned}$$

▷ well-defined since never $x\colon c, x\colon \overline{c} \in \mathcal{B}$ (otherwise $\bot$ applies)

▷ $\Im$ satisfies all constraints $x\colon c$, $x\colon \overline{c}$ and $x \mathrel{\mathsf{R}} y$, (by construction)

P.2 (Induction) $\Im \models y\colon \psi$, for all $y\colon \psi \in \mathcal{B}$ (on $k = size(\psi)$ next slide)

P.3 (Consequence) $\Im \models x\colon \varphi$. □

JACOBS UNIVERSITY

Case Analysis for Induction

case $y\colon \psi = y\colon \psi_1 \sqcap \psi_2$ Then $\{y\colon \psi_1, y\colon \psi_2\} \subseteq \mathcal{B}$ ($\sqcap$-rule, saturation)

so $\Im \models y\colon \psi_1$ und $\Im \models y\colon \psi_2$ and $\Im \models y\colon \psi_1 \sqcap \psi_2$ (IH, Definition)

case $y\colon \psi = y\colon \psi_1 \sqcup \psi_2$ Then $y\colon \psi_1 \in \mathbf{B}$ or $y\colon \psi_2 \in \mathbf{B}$ ($\sqcup$-rule, saturation)

so $\Im \models y\colon \psi_1$ or $\Im \models y\colon \psi_2$ and $\Im \models y\colon \psi_1 \sqcup \psi_2$ (IH, Definition)

case $y\colon \psi = y\colon \exists \mathsf{R}.\theta$ then $\{y \mathrel{\mathsf{R}} z, z\colon \theta\} \subseteq \mathbf{B}$ (z new variable) ($\exists_*$-rules, saturation)

so $\Im \models z\colon \theta$ and $\Im \models y \mathrel{\mathsf{R}} z$, thus $\Im \models y\colon \exists \mathsf{R}.\theta$. (IH, Definition)

case $y\colon \psi = y\colon \forall \mathsf{R}.\theta$ Let $\langle [\![y]\!], v\rangle \in [\![\mathsf{R}]\!]$ for some $r \in \Im_D$

then $v = z$ for some variable z with $y \mathrel{\mathsf{R}} z \in \mathbf{B}$ (construction of $[\![\mathsf{R}]\!]$)

So $z\colon \theta \in \mathcal{B}$ and $\Im \models z\colon \theta$. ($\forall$-rule, saturation, Def)

Since v was arbitrary we have $\Im \models y\colon \forall \mathsf{R}.\theta$.

JACOBS UNIVERSITY

Complexity

- Idea: We can organize the tableau procedure, so that the branches are worked off one after the other. Therefore the size of the branches is relevant of the (space)-complexity of the procedure.
- The size of the branches is *polynomial* in the size of the input formula (same reasons as for termination)
  - every rule except $\forall$ is only applied to a constraint $x\colon\psi$.
  - The $\forall$ is applied to constraints of the form $x\colon\forall\mathsf{R}.\psi$ at most as often as there are R-successors of x.
  - The R-successors of x are generated by constraints $x\colon\exists\mathsf{R}.\theta$, whose number is bounded by the size of the input formula.
  - Each application to a constraint $x\colon\psi$ generates constraints $z\colon\psi'$ where ψ' is a proper subformula of ψ.

  The total size is the size of the input formula plus number of $\exists$-formulae times number of $\forall$-formulae.
- Theorem 5.27: *The consisteny problem for $\mathcal{ALC}$ is in* $\mathbf{PSPACE}$.
- Theorem 5.28: *The consistency problem for $\mathcal{ALC}$ is PSPACE-Complete.*
- Proof: reduce a PSPACE-complete problem to $\mathcal{ALC}$-consistency

  □
- Theorem 5.29: (Time Complexity)
  The $\mathcal{ALC}$-consistency problem is in $\mathbf{EXPTIME}$
- Proof: Sketch: There can be exponentially many branches(already for propositional logic)

  □

The functional Algorithm for $\mathcal{ALC}$

▷ Observation: leads to treatment for $\exists$

- ▷ the $\exists$-rule generates the constraints $x \mathsf{R} y$ and $y\colon \psi$ from $x\colon \exists \mathsf{R}.\psi$
- ▷ this triggers the $\forall$-rule for $x\colon \forall \mathsf{R}.\theta_i$, which generate $y\colon \theta_1, \ldots, y\colon \theta_n$
- ▷ for y we have $y\colon \psi$ and $y\colon \theta_1, \ldots y\colon \theta_n$. (do all of this in a single step)
- ▷ we are only interested in non-emptyness, not in the particular witnesses (leave them out)

$consistent(S)$ =
if $\{c, \overline{c}\} \subseteq S$ **then false** (inconsistent) **elseif** '$(\varphi \sqcap \psi)$' $\in S$ **and** ('φ' $\notin S$ **or** 'ψ' $\notin S$)
then $consistent(S \cup \{\varphi, \psi\})$
elseif '$(\varphi \sqcup \psi)$' $\in S$ **and** $\{\varphi, \psi\} \notin S$
then $consistent(S \cup \{\varphi\})$ **or** $consistent(S \cup \{\psi\})$
elseif forall '$(\exists \mathsf{R}.\psi)$' $\in S$
$consistent(\{\psi\} \cup (\{\theta \mid \text{'}(\forall \mathsf{R}.\theta)\text{'} \in S\}))$
else true (consistent)

▷ relatively simple to implement (good implementations optimized)

▷ but: this is restricted to $\mathcal{ALC}$. (extension to other DL difficult)

Extending the Tableau Algorithm by Concept Axioms

▷ Concept axioms, e.g. child $\sqsubseteq$ son $\sqcup$ daughter could not be handled in tableau calculi

▷ Idea: Whenever a new variable y is introduced (by $\exists$-rule) add the information that axioms hold for y.

▷
- ▷ initialize tableau with $\{x\colon \varphi\} \cup \mathcal{CA}$ ($\mathcal{CA}\colon =$ set of concept axioms)
- ▷ new $\exists$-rule: $\dfrac{x\colon \exists \mathsf{R}.\varphi \quad \alpha \in \mathcal{CA}}{y\colon \alpha}\exists_{\mathcal{CA}}$ (apply-co-exhaustively to $\exists$)

Problem: $\mathcal{CA} := \{\exists \mathsf{R}.c\}$ and start tableau with $x\colon d$ (non-termination)

Non-Termination of Tableau with Concept Axioms

$x\colon d$	start
$x\colon \exists \mathsf{R}.c$	$\in \mathcal{CA}$
$x\ \mathsf{R}\ y_1$	$\exists$
$y_1\colon c$	$\exists$
$y_1\colon \exists \mathsf{R}.c$	$\exists_{\mathcal{CA}}$
$y_1\ \mathsf{R}\ y_2$	$\exists$
$y_2\colon c$	$\exists$
$y_2\colon \exists \mathsf{R}.c$	$\exists_{\mathcal{CA}}$
$\ldots$	

Solution: Loop-Check:

- ▷ instead of a new variable y take an old variable z, if we can guarantee that whatever holds for y already holds for z.
- ▷ we can only do this, iff the $\forall$-rule has been exhaustively applied.

ABoxes (Database Component of DL)

▷ Formula: $a\colon \varphi$ (a is a φ) | $a\mathsf{R}b$ (a stands in relation R to b)

property	example
internally inconsistent	$\mathsf{tony}\colon \mathsf{student}, \mathsf{tony}\colon \overline{\mathsf{student}}$
inconsistent with a TBox	$TBox:\ \overline{\mathsf{student} \sqcap \mathsf{prof}}$ $ABox:\ \mathsf{tony}\colon \mathsf{student}, \mathsf{tony}\colon \mathsf{prof}$
implicit info that is not explicit	$Abox:\ \mathsf{tony}\colon \forall \mathsf{has\_grad}.\mathsf{genius}$ $\mathsf{tony}\mathsf{has\_grad}\mathsf{mary}$ $\models \mathsf{mary}\colon \mathsf{genius}$
info that can be combined with TBox info	$TBox:\ \mathsf{cont\_prof} = \mathsf{prof} \sqcap (\forall \mathsf{has\_grad}.\mathsf{genius})$ $ABox:\ \mathsf{tony}\colon \mathsf{cont\_prof}, \mathsf{tony}\mathsf{has\_grad}\mathsf{mary}$ $\models \mathsf{mary}\colon \mathsf{genius}$

Tableau-based Instance Test and Realization

- Query: do the ABox and TBox together entail $a\colon\varphi$ $(a \in \varphi?)$
- Algorithm: test $a\colon\overline{\varphi}$ for consistency with ABox and TBox.[6] (use our tableau)
- necessary changes: (no big deal)
  - Normalize ABox wrt. TBox (definition expansion)
  - initialize the tableau with ABox in NNF (so it can be used)

Example: add mary: $\overline{\text{genius}}$ to determine $ABox, TBox \models$ mary: genius			
TBox	cont_prof = prof ⊓ (∀has_grad.genius)	tony: prof ⊓ (∀has_grad.genius) tonyhas_gradmary tony: prof tony: ∀has_grad.genius mary: $\overline{\text{genius}}$ *	$Norm$ $Norm$ ⊓ ⊓ ∀ *
ABox	tony: cont_prof tonyhas_gradmary		

- Note: The instance test is the base for the realization (remember?)
- extend to more complex ABox queries: (give me all instances of φ)

[6] EdNote: need to unify abox and tbox judgments.

5.4 ALC Extensions

Language Extensions

- $\mathcal{ALC}$ is much more expressive than propositional logic, (still not enough)
- Idea: study more expressive extensions
- Need to study:
  - which new operators? (are some definable)
  - translation into predicate logic
  - are the inference problems decidable? (consistency, subsumption, instance test,...)
  - what is the complexity of the decision problem?
  - what do the algorithms look like?

5.4.1 Functional Roles and Number Restrictions

Functional Roles

▷ Example 5.30: CSR $\widehat{=}$ Car with glass sun roof

- ▷ In $\mathcal{ALC}$: CSR $=$ car $\sqcap$ ($\exists$has_sun_roof.glass)
- ▷ potential unwanted interpretation: more than one sun roof.
- ▷ Problem: has_sun_roof is a relation in $\mathcal{ALC}$ (no partial function)

▷ Example 5.31: Humans have exactly one father and mother.

- ▷ in $\mathcal{ALC}$: human $\sqsubseteq$ ($\exists$has_father.human) $\sqcap$ ($\exists$has_mother.human)
- ▷ Problem: has_father should be a total function (on the set of humans)

Solution: Number Restrictions (see next slide)

▷ Example 5.32: Teenager $=$ human between 13 and 19

- ▷ teenager $=$ human $\sqcap$ (age < 20)age > 12 (not covered by $\mathcal{ALC}$)
- ▷ Solution: concrete domains (outside the scope of this course)

  JACOBS UNIVERSITY

Number Restrictions

▷ Example 5.33: Car $=$ vehicle with at least four wheels

▷ Trick: In $\mathcal{ALC}$: model car using two new distinguishing concepts p_1 and p_2
vehicle $\sqcap$ ($\exists$has_wheel.$p_1 \sqcap p_2$) $\sqcap$ ($\exists$has_wheel.$\overline{p_1} \sqcap p_2$) $\sqcap$ ($\exists$has_wheel.$p_1 \sqcap \overline{p_2}$) $\sqcap$ ($\exists$has_wheel.$\overline{p_1} \sqcap \overline{p_2}$)

▷ Problem: city $=$ town with at least 1,000,000 inhabitants (oh boy)

▷ Alternative: Operators for number restrictions.

  JACOBS UNIVERSITY

(Unqualified) Number Restrictions

▷ $\mathcal{ALC}$ plus operators $\exists^n_\geq$R and $\forall^n_\leq$R (R role, $n \in \mathbb{N}$)

▷ Example 5.34:

$$\begin{array}{rcl} \text{car} & = & \text{vehicle} \sqcap \exists^4_\geq \text{has\_wheel} \\ \text{city} & = & \text{town} \sqcap \exists^{1,000,000}_\geq \text{has\_inhabitants} \\ \text{small\_family} & = & \text{family} \sqcap \forall^2_\leq \text{has\_child} \end{array}$$

▷ Semantics:

$$\begin{array}{rcl} [\![\exists^n_\geq \mathsf{R}]\!] & = & (\{x \in \mathcal{D} \mid \#(\{y \mid \langle x,y\rangle \in [\![\mathsf{R}]\!]\}) \geq n\}) \\ [\![\forall^n_\leq \mathsf{R}]\!] & = & (\{x \in \mathcal{D} \mid \#(\{y \mid \langle x,y\rangle \in [\![\mathsf{R}]\!]\}) \leq n\}) \end{array}$$

▷ Intuitively: $\exists^n_\geq$R is the set of objects that have at least n R-successors.

▷ Example 5.35: $\exists^{1,000,000}_\geq$has_inhabitants is the set of objects that have at least 1,000,000 inhabitants.

  JACOBS UNIVERSITY

Translation into Predicate Logic

▷ Two extra rules for number restrictions: (very cumbersome)

$\overline{\exists^n_{\geq} \mathsf{R}}^{fo(x)}$	$\overline{\forall^n_{\leq} \mathsf{R}}^{fo(x)}$
$\exists y_1.\mathsf{R}(x,y_1)\wedge \quad \ldots \quad \wedge\exists y_n.\mathsf{R}(x,y_n)$	$\neg\exists y.\mathsf{R}(x,y)\vee$
$\wedge y_1 \neq y_2\wedge \quad \ldots \quad \wedge y_1 \neq y_n$	$(\exists y_1.\mathsf{R}(x,y_1)\wedge\ldots\wedge\exists y_n.\mathsf{R}(x,y_n))$
$\wedge \quad y_2 \neq y_3\wedge \quad \ldots\wedge y_2 \neq y_n\wedge$	$\forall y.\mathsf{R}(x,y)\Rightarrow(y=y_1\vee\ldots\vee y=y_n))$
$y_{n-1}\neq y_n$	

▷ Definable Operator: $=$

▷ $\overset{n}{=}\mathsf{R} := \exists^n_{\geq}\mathsf{R}\sqcap\forall^n_{\leq}\mathsf{R}$
defines the set of objects that have exactly n R-successors.

▷ Example 5.36: car $=$ vehicle $\sqcap \overset{n}{=}$ has_wheel (vehicles with exactly 4 wheels)

Functional Roles

▷ Example 5.37: CSR $=$ car $\sqcap \overset{1}{=}$ has_sun_roof (CSR $=$ car with sun roof)

has_sun_roof is a relation, but restricted to CSR it is a total function.

▷ Partial functions: Chd $=$ computer $\sqcap\forall^1_{\leq}$has_hd (computer with at most one hard drive)
has_hd is a partial function on the set Chd

▷ Intuition: number restrictions can be used to encode partial and total functions, but not to specify the range type.

Negation Rules

▷ Observation: to compute the negation normal form, need the rules for the new operators

$\overline{\exists^n_{\geq}\mathsf{R}} \mapsto \forall^{n-1}_{\leq}\mathsf{R}$	$\overline{\forall^n_{\leq}\mathsf{R}} \mapsto \exists^{n+1}_{\geq}\mathsf{R}$

▷ Proof: by the semantics of the operators

□

▷ Example 5.38:

$$1: \overline{\exists^5_{\geq}\text{has\_child}} = \forall^4_{\leq}\text{has\_child}$$
$$2: \overline{\forall^5_{\leq}\text{has\_child}} = \exists^6_{\geq}\text{has\_child}$$

Tableaux Rules (without ABox Information)

$$\frac{x\;\mathsf{R}\;a_1 \quad \vdots \quad x\;\mathsf{R}\;a_{n-k} \quad x\colon \exists^{n}_{\geq}\mathsf{R}}{x\;\mathsf{R}\;y_1 \quad \vdots \quad x\;\mathsf{R}\;y_k}\; y_1,\ldots,y_k \text{ new} \qquad \frac{x\;\mathsf{R}\;a_1 \quad \vdots \quad x\;\mathsf{R}\;a_m \quad x\colon \forall^{n}_{\leq}\mathsf{R}}{[a_j/a_i] \text{ everywhere}}\; \begin{array}{c} m>n \\ 1\leq i,j\leq m \end{array}$$

▷ Basic Intuition (but when do we fail? Can we always identify)

- ▹ $\exists^{n}_{\geq}\mathsf{R}$: Introduce as many R-successors as necessary
- ▹ $\forall^{n}_{\leq}\mathsf{R}$: Identify two R-successors if there are too many (repeat as needed)

  JACOBS UNIVERSITY

5.4.2 Unique Names

Unique Name Assumption

▷ Problem: assuming UNA for ABox constants (but not always)

▷ Definition 5.39: (Unique Name Assumption) (UNA)

Different names for objects denote different objects, (cannot be equated)

▷ Example 5.40:

Bob : gardener
Bob : gardener
UNAbomber : gardener

- ▹ Bill and Bob are different
- ▹ but the UNAbomber can be Bill or Bob or someone else.

▷ Assumption: mark every ABox constant with 'UNA' or '$\overline{\text{UNA}}$'

  JACOBS UNIVERSITY

Tableau Rules (with ABox Information)

$$\frac{\begin{array}{c} x\;\mathsf{R}\;a_1 \\ \vdots \\ x\;\mathsf{R}\;a_{n-k} \\ x\colon \exists^n_{\geq}\mathsf{R} \end{array} \quad \begin{array}{c} y_1,\ldots,y_k\colon \mathsf{UNA}\;\;\text{new} \\ a_1,\ldots,a_{n-k}\colon \mathsf{UNA} \end{array}}{\begin{array}{c} x\;\mathsf{R}\;y_1 \\ \vdots \\ x\;\mathsf{R}\;y_k \end{array}}$$

$$\frac{\begin{array}{c} x\;\mathsf{R}\;a_1 \vdots \\ x\;\mathsf{R}\;a_m \\ x\colon \forall^n_{\leq}\mathsf{R} \end{array} \quad \begin{array}{c} m > n \\ 1 \leq i,j \leq m \\ a_i\colon \overline{\mathsf{UNA}} \end{array}}{[a_j/a_i]\;\;\text{everywhere}}$$

$$\frac{\begin{array}{c} x\;\mathsf{R}\;a_1 \\ \vdots \\ x\;\mathsf{R}\;a_m \\ x\colon \forall^n_{\leq}\mathsf{R} \end{array} \quad \begin{array}{c} m > n \\ a_1,\ldots,a_m\colon \mathsf{UNA} \end{array}}{\bot}$$

Example: Solving a Crime with Number Restrictions

▷ Example 5.41: 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.)

Bill: UNA, Bob: UNA, tony: UNA, muderer: $\overline{\mathsf{UNA}}$

$$\begin{array}{c} \mathsf{tony}\colon \forall^2_{\leq}\mathsf{observes} \\ \mathsf{tony\ observes\ Bill} \\ \mathsf{tony\ observes\ Bob} \\ \mathsf{tony\ observes\ muderer} \\ \mathsf{muderer}\colon \mathsf{black\_hair} \\ \mathsf{Bill}\colon \overline{\mathsf{black\_hair}} \\ \mathsf{Bob}\colon \mathsf{black\_hair} \\ \begin{array}{c|c} \mathsf{tony\ observes\ Bill} & \mathsf{tony\ observes\ Bob} \\ \mathsf{Bill}\colon \mathsf{black\_hair} & \mathsf{Bob}\colon \mathsf{black\_hair} \\ * & \end{array} \end{array}$$

5.4.3 Qualified Number Restrictions

Qualified Number Restrictions

- $\mathcal{ALC}$ plus operators $\exists^n_{\geq}\mathsf{R}.\varphi$ and $\forall^n_{\leq}\mathsf{R}.\varphi$ (R role, $n \in \mathbb{N}$, φ formula)
- Example 5.42: $\mathsf{person} \sqcap \forall^2_{\leq}\mathsf{has\_child.blond}$‘ (persons with ≤ 2 blond kids)
- Example 5.43: $\mathsf{comp} \sqcap \exists^5_{\geq}\mathsf{has\_client.car\_comp}$ (company with at least 5 clients in the automobile industry)
- Special case: Unqualified Number restrictions ($\exists^n_{\geq}\mathsf{R}.\top$, $\forall^n_{\leq}\mathsf{R}.\top$.)
- Semantics: $[\![\exists^n_{\geq}\mathsf{R}.\varphi)]\!] = (\{x \in \mathcal{D} \mid \#(\{y \mid \langle x,y\rangle \in [\![\mathsf{R}]\!] \text{ and } y \in [\![\varphi]\!]\}) \geq n\})$
  $[\![\forall^n_{\leq}\mathsf{R}.\varphi)]\!] = (\{x \in \mathcal{D} \mid \#(\{y \mid \langle x,y\rangle \in [\![\mathsf{R}]\!] \text{ and } y \in [\![\varphi]\!]\}) \leq n\})$

Negation and Quantifier Elimination

- Negation rules: $\boxed{\overline{\exists^n_{\geq}\mathsf{R}.\varphi} = \forall^{n-1}_{\leq}\mathsf{R}.\varphi \;\Big|\; \overline{\forall^n_{\leq}\mathsf{R}.\varphi} = \exists^{n+1}_{\geq}\mathsf{R}.\varphi}$
- Example 5.44: $\overline{\exists^3_{\geq}\mathsf{has\_child.teacher}} = \forall^2_{\leq}\mathsf{has\_child.teacher}$
- Example 5.45: $\overline{\forall^3_{\leq}\mathsf{has\_child.teacher}} = \exists^4_{\geq}\mathsf{has\_child.teacher}$
- Quantifier elimination (regular quantifiers no longer necessary)
  - $\exists\mathsf{R}.\varphi = \exists^1_{\geq}\mathsf{R}.\varphi$
  - $\forall\mathsf{R}.\varphi = \overline{\exists\mathsf{R}.\overline{\varphi}} = \overline{\exists^1_{\geq}\mathsf{R}.\overline{\varphi}} = \forall^0_{\leq}\mathsf{R}.\overline{\varphi}$

Optimizied Tableau Rules [Tob00]

- Definition 5.46: $\mathcal{T}_{ALC}$ rules plus:

$$\frac{\begin{array}{c}\mathcal{B}\\ x\colon \exists^n_{\geq}r.\varphi\end{array} \quad \#(\{y \mid x\;\mathsf{R}\;y, y\colon \varphi \in \mathcal{B}\}) < n \quad y \text{ new}}{\begin{array}{c}x\;\mathsf{R}\;y\\ y\colon \varphi\\ y\colon \xi_1\\ \vdots\\ y\colon \xi_k\end{array}}$$

where $\{\psi_1, \ldots, \psi_k\} = (\{\psi \mid x\colon \exists^m_{\geq}\mathsf{R}.\psi \in \mathcal{B} \text{ or } x\colon \forall^m_{\leq}\mathsf{R}.\psi \in \mathcal{B}\})$ and $\xi_i = \psi$ or $\xi = \overline{\psi}$.

$$\frac{\begin{array}{c}\mathcal{B}\\ x\colon \forall^n_{\leq}r.\varphi\end{array} \quad \#(\{y \mid x\;\mathsf{R}\;y, y\colon \varphi \in \mathcal{B}\}) > n}{*}$$

Example Tableau

▷ Example 5.47:

$$
\begin{array}{c}
x\colon \exists^{3}_{\geq}\mathsf{R}.\varphi \sqcap \forall^{1}_{\leq}\mathsf{R}.\psi \sqcap \forall^{1}_{\leq}\mathsf{R}.\overline{\psi}\\
x\colon \exists^{3}_{\geq}\mathsf{R}.\varphi\\
x\colon \forall^{1}_{\leq}\mathsf{R}.\psi\\
x\colon \forall^{1}_{\leq}\mathsf{R}.\overline{\psi}\\
x\;\mathsf{R}\;y_1\\
y_1\colon \varphi\\
\begin{array}{c|c}
\begin{array}{c}
y_1\colon \psi\\
x\;\mathsf{R}\;y_2\\
y_2\colon \varphi\\
\begin{array}{c|c}
\begin{array}{c} y_2\colon \psi\\ * \end{array} &
\begin{array}{c}
y_2\colon \overline{\psi}\\
x\;\mathsf{R}\;y_3\\
y_3\colon \varphi\\
\begin{array}{c|c}
\begin{array}{c} y_3\colon \psi\\ * \end{array} & \begin{array}{c} y_3\colon \overline{\psi}\\ * \end{array}
\end{array}
\end{array}
\end{array}
\end{array}
&
\begin{array}{c}
y_1\colon \overline{\psi}\\
x\;\mathsf{R}\;y_2\\
y_2\colon \varphi\\
\begin{array}{c|c}
\begin{array}{c}
y_2\colon \psi\\
x\;\mathsf{R}\;y_3\\
y_2\colon \varphi\\
\begin{array}{c|c}
\begin{array}{c} y_3\colon \psi\\ * \end{array} & \begin{array}{c} y_3\colon \overline{\psi}\\ * \end{array}
\end{array}
\end{array}
&
\begin{array}{c} y_2\colon \overline{\psi}\\ * \end{array}
\end{array}
\end{array}
\end{array}
\end{array}
$$

▷ Problem: Naive Implementation: exponential path lengths

Implementation by "Traces"

▷ Algorithm SAT(φ) = sat($x_0, \{x_0\colon \varphi\}$)

$sat(x, S)$:

allocate counter $\#r^S(x,\psi) := 0$ for all roles R and positive or negative subformulae ψ in S.

apply rules $\sqcap$ and $\sqcup$ as long as possible

If S contains an inconsistency, RETURN $*$.

`while`($\mapsto_{\geq}$ is appliccable to x) `do`:

$S_{neu} := \{\mathcal{T}_{ALC}\mathsf{R}xy, y\colon \varphi, y\colon \xi_1, \ldots y\colon \xi_k\}$
`where`
y is a new variable,
$x\colon \exists^{n}_{\geq}\mathsf{R}.\varphi$ triggers rule $\mapsto_{\geq}$,
$\{\psi_1,\ldots,\psi_k\} = (\{\psi \mid x\colon \exists^{m}_{\geq}\mathsf{R}.\psi \in \mathcal{B}$ or $x\colon \forall^{m}_{\leq}\mathsf{R}.\psi \in \mathcal{B}\})$ and
$\xi_i = \psi$ oder $\xi = \neg\psi$.

For each $y\colon \psi \in S_{new}$: $\#r^S(x,\psi) + = 1$ If $x\colon \forall^{m}_{\leq}\mathsf{R}.\psi \in \mathcal{B}$ and $\#r^S(x,\psi) > m$ RETURN $*$

If $sat(y, S_{neu}) = *$ RETURN $*$ od

RETURN "'consistent"'.

Analysis

- ▷ Idea: Each R-successor of x triggers a recursive call of sat.
- ▷ There may be exponentially many R-successor, but they are treated one-by-one, so their space can be re-used.
- ▷ The chains of R-successors are at most as long as the nesting depth of operators(linear)
- ▷ Lemma 5.48: *Space consumption is polynomial.*
- ▷ Lemma 5.49: *This algorithm is complete.*
- ▷ Proof: Sketch: The global counters $\#r^S(x,\psi)$ count the R-successors and trigger rule $\mapsto_{\leq}$.

□

- ▷ Theorem 5.50: *The algorithm is correct, complete and terminating, and PSPACE (no worse than $\mathcal{ALC}$)*

 171 JACOBS UNIVERSITY

5.4.4 Role Operators

The DL-Zoo: Operator Types

- ▷ Operators on role names (construct roles on the fly)
- ▷ role hierarchy and role axioms (knowledge about roles)
- ▷ nominals (names for domain elements)
- ▷ features (partial functions)
- ▷ concrete domains (e.g. $\mathbb{N}, \mathbb{Z}, trees$)
- ▷ external data structures (for programming)
- ▷ epistemic operators (belief,...)
- ▷ ...

 172 JACOBS UNIVERSITY

Role Hierarchies

- ▷ Idea: specification of subset relations among relations.
- ▷ Example 5.51: role hierarchy as a directed graph $\mathcal{R}$

has_daughter ⊑ has_child
has_son ⊑ has_child
talks_to ⊑ communicates_with
calls ⊑ communicates_with
buys ⊑ obtains
steals ⊑ obtains

 173 JACOBS UNIVERSITY

$\mathcal{ALC}$ with Role hierarchies (without role operators)

▷ Definition 5.52: $\mathcal{T}_{ALC}$ + complex roles instead of role names

$$\frac{x\colon \exists \mathsf{R}.\varphi}{\begin{array}{c} x\ \mathsf{R}\ y \\ y\colon \varphi \end{array}}\ \exists \qquad \frac{\begin{array}{c} x\ \mathsf{S}\ y \\ x\colon \forall \mathsf{R}.\varphi \end{array} \quad \mathsf{S} \sqsubseteq \mathsf{R} \in \mathcal{R}}{y\colon \varphi}\ \forall_{\sqsubseteq}$$

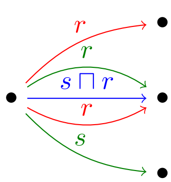
The $\exists$ rule is the same as before

Operators on Roles: Role Conjunction

▷ Example 5.53: person $\sqcap$ ($\exists$(has_teacher $\sqcap$ has_friend).swiss)
(persons that have a Swiss teacher that is also their friend)

▷ Example 5.54: com $\sqcap$ ($\exists$(has_employee $\sqcap$ has_attorney).lawyer)
(companies that have an employed attorney that is a lawyer)

▷ Semantics: $[\![\mathsf{R} \sqcap \mathsf{S}]\!] = [\![\mathsf{R}]\!] \cap [\![\mathsf{S}]\!] = (\{\langle x,y\rangle \in \mathcal{D} \mid \langle x,y\rangle \in [\![\mathsf{R}]\!] \text{ and } \langle x,y\rangle \in [\![\mathsf{S}]\!]\})$



Inference Rules

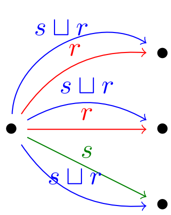
$$(\forall \mathsf{R} \sqcap \mathsf{S}.\varphi) \sqsubseteq (\forall \mathsf{R}.\varphi) \sqcap (\forall \mathsf{S}.\varphi)$$
$$(\exists \mathsf{R} \sqcap \mathsf{S}.\varphi) \sqsubseteq (\exists \mathsf{R}.\varphi) \sqcap (\exists \mathsf{S}.\varphi)$$
$$\exists^n_{\geq} \mathsf{R} \sqcap \mathsf{S}.\varphi \sqsubseteq \exists^n_{\geq} \mathsf{R}.\varphi \sqcap \exists^n_{\geq} \mathsf{S}.\varphi$$
$$\forall^{n+m}_{\leq} \mathsf{R} \sqcap \mathsf{S}.\varphi \sqsubseteq \exists^n_{\geq} \mathsf{R}.\varphi \sqcap \exists^m_{\geq} \mathsf{S}.\varphi$$

Role Disjunction $\sqcup$

▷ Example 5.55: person $\sqcap$ ($\forall$has_child $\sqcup$ has_friend.teacher)
(persons whose children and friends are all teachers)

▷ Example 5.56: com $\sqcap$ ($\exists$has_employee $\sqcup$ has_consultant.member_of_congress)
(companies with an employee or consultant who is member of congress)

▷ Semantics: $[\![\mathsf{R} \sqcup \mathsf{S}]\!] = [\![\mathsf{R}]\!] \cup [\![\mathsf{S}]\!] = (\{\langle x,y\rangle \in \mathcal{D} \mid \langle x,y\rangle \in [\![\mathsf{R}]\!] \text{ or } \langle x,y\rangle \in [\![\mathsf{S}]\!]\})$



Inference Rules

$$\forall \mathsf{R} \sqcup \mathsf{S}.\varphi = (\forall \mathsf{R}.\varphi) \sqcup (\forall \mathsf{S}.\varphi)$$
$$\exists \mathsf{R} \sqcup \mathsf{S}.\varphi = (\exists \mathsf{R}.\varphi) \sqcup (\exists \mathsf{S}.\varphi)$$
$$\exists^n_{\geq} \mathsf{R} \sqcup \mathsf{S}.\varphi = ??$$
$$\forall^n_{\leq} \mathsf{R} \sqcup \mathsf{S}.\varphi \sqsubseteq \forall^n_{\leq} \mathsf{R}.\varphi \sqcap \forall^n_{\leq} \mathsf{S}.\varphi$$
$$\forall^{n+m}_{\leq} \mathsf{R} \sqcup \mathsf{S}.\varphi \sqsubseteq \forall^n_{\leq} \mathsf{R}.\varphi \sqcap \forall^m_{\leq} \mathsf{S}.\varphi$$
$$\exists^{max(n,m)}_{\geq} \mathsf{R} \sqcup \mathsf{S}.\varphi \sqsubseteq \exists^n_{\geq} \mathsf{R}.\varphi \sqcup \exists^m_{\geq} \mathsf{S}.\varphi$$

Role Complement $\bar{\cdot}$

- ▷ Example 5.57: univ ⊓ (∀has_employee ⊓ $\overline{\text{has\_prof}}$.unionized) (universities whose employees that are not professors are unionized)
- ▷ Example 5.58: house ⊓ (∃resident ⊓ $\overline{\text{owner}}$.swiss)(houses whose residents that are not owners are Swiss)
- ▷ $[\![\overline{\mathsf{R}}]\!] = \mathcal{D}^2 \backslash [\![\mathsf{R}]\!] = (\{\langle x, y \rangle \in \mathcal{D}^2 \mid \langle x, y \rangle \notin [\![\mathsf{R}]\!]\})$
- ▷ Observation: $\sqcap, \sqcup, \bar{\cdot}$ is a Boolean algebra (propositional logic)

  We can compute with role terms built up from $\sqcap, \sqcup, \bar{\cdot}$ exactly like with propositional formulae built up from $\wedge, \vee, \neg$.
- ▷ Example 5.59: $\forall \overline{\mathsf{R} \sqcap \mathsf{S}}.\varphi = \forall \overline{\mathsf{R}} \sqcup \overline{\mathsf{S}}.\varphi$
- ▷ more rules: if $\mathsf{R} \sqsubseteq \mathsf{S}$ is a tautology, then $(\forall \mathsf{S}.\varphi) \sqsubseteq (\forall \mathsf{R}.\varphi)$ and $(\exists \mathsf{R}.\varphi) \sqsubseteq (\exists \mathsf{S}.\varphi)$

 177

Special Relations 0 and 1

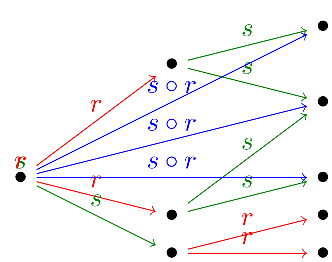
$\mathsf{R} \sqcap \overline{\mathsf{R}} = 0$	empty relation
$\mathsf{R} \sqcup \overline{\mathsf{R}} = 1$	universal relation

- ▷ Question: what does $\forall 1.\varphi$ mean?

 178

Role composition ∘

- ▷ Example 5.60: person ⊓ (∃has_child ∘ has_child.prof) (persons that have grandchild that is a professor)
- ▷ Example 5.61: univ ⊓ (∀has_student ∘ has_Partner ∘ lives_in).Texas) (universities whose students all have partners that live in Texas)
- ▷ Semantics: $[\![\mathsf{R} \circ \mathsf{S}]\!] = [\![\mathsf{R}]\!][\![\mathsf{S}]\!] = (\{\langle x, z \rangle \in \mathcal{D}^2 \mid \exists y.\langle x, y \rangle \in [\![\mathsf{S}]\!] \text{ and } \langle y, z \rangle \in [\![\mathsf{R}]\!]\})$



 179

Converse Roles $(\cdot^{-1})$

▷ Example 5.62: (set of objects whose parents are teachers)

$$\begin{aligned}
[\![\forall \mathsf{has_child}^{-1}.\mathsf{teacher}]\!] &= (\{x \mid \forall y.\langle x,y\rangle \in [\![\mathsf{has_child}^{-1}]\!] \Rightarrow y \in [\![\mathsf{teacher}]\!]\}) \\
&= (\{x \mid \forall y.\langle y,x\rangle \in [\![\mathsf{has_child}]\!] \Rightarrow y \in [\![\mathsf{teacher}]\!]\}) \\
&= (\{x \mid \forall y.\langle x,y\rangle \in [\![\mathsf{has_parents}]\!] \Rightarrow y \in [\![\mathsf{teacher}]\!]\})
\end{aligned}$$

▷ Definition 5.63: $[\![\mathsf{R}^{-1}]\!] = [\![\mathsf{R}]\!]^{-1} = (\{\langle y,x\rangle \in \mathcal{D}^2 \mid \langle x,y\rangle \in [\![\mathsf{R}]\!]\})$

▷ Example 5.64:

$$\begin{aligned}
\mathsf{has_child}^{-1} &= \mathsf{has_parents} \\
\mathsf{is_part_of}^{-1} &= \mathsf{contains_as_part} \\
\mathsf{owns}^{-1} &= \mathsf{belongs_to} \\
\ldots
\end{aligned}$$

 180

Translation of Role Terms

$\mathsf{tr}^{x,y}(\mathsf{R}) = \mathsf{R}(x,y)$	$=$
$\mathsf{tr}^{x,y}(\mathsf{R} \sqcap \mathsf{S}) = \mathsf{tr}^{x,y}(\mathsf{R}) \wedge \mathsf{tr}^{x,y}(\mathsf{S})$	$\mathsf{tr}^{x,y}(\mathsf{R} \sqcup \mathsf{S}) = \mathsf{tr}^{x,y}(\mathsf{R}) \vee \mathsf{tr}^{x,y}(\mathsf{S})$
$\mathsf{tr}^{x,y}(\mathsf{R} \sqsubseteq \mathsf{S}) = \mathsf{tr}^{x,y}(\mathsf{R}) \Rightarrow \mathsf{tr}^{x,y}(\mathsf{S})$ $\mathsf{tr}^{x,y}(\mathsf{R}^{-1}) = \mathsf{tr}^{y,x}(\mathsf{R})$	$\mathsf{tr}^{x,y}(\mathsf{R} \circ \mathsf{S}) = (\exists z.\mathsf{tr}^{x,z}(\mathsf{R}), \mathsf{tr}^{z,y}(\mathsf{S}))$ $\mathsf{tr}^{x,y}(\overline{\mathsf{R}}) = \neg \mathsf{tr}^{x,y}(\mathsf{R})$
$\overline{\forall \mathsf{R}.\varphi}^{fo(x)} = (\forall y.\mathsf{tr}^{x,y}(\mathsf{R})) \Rightarrow \overline{\varphi}^{fo(y)}$	$\overline{\exists \mathsf{R}.\varphi}^{fo(x)} = (\exists y.\mathsf{tr}^{x,y}(\mathsf{R}), \overline{\varphi}^{fo(y)})$

▷ Example 5.65:

$$\begin{aligned}
&\overline{\forall \mathsf{R} \circ \mathsf{S} \sqcap \mathsf{T}^{-1}.c}^{fo(x)} \\
&= \forall y.\mathsf{tr}^{x,y}(\overline{\mathsf{R} \circ \mathsf{S} \sqcap \mathsf{T}^{-1}}) \Rightarrow \overline{c}^{fo(y)} \\
&= \forall y.\neg \mathsf{tr}^{x,y}(\mathsf{R} \circ \mathsf{S} \sqcap \mathsf{T}^{-1}) \Rightarrow c(y) \\
&= \forall y.\neg(\exists z.\mathsf{R}(x \wedge z) \wedge \mathsf{tr}^{z,y}(\mathsf{S} \sqcap \mathsf{T}^{-1})) \Rightarrow c(y) \\
&= \forall y.\neg(\exists z.\mathsf{R}(x \wedge z) \wedge \mathsf{tr}^{y,z}(\mathsf{S} \sqcap \mathsf{T})) \Rightarrow c(y) \\
&= \forall y.\neg(\exists z.\mathsf{R}(x \wedge z) \wedge \mathsf{S}(y \wedge z) \wedge \mathsf{T}(y \wedge z)) \Rightarrow c(y)
\end{aligned}$$

 181

Connection to dynamic Logic

▷ Dynamic Logic is used for specification and verification of imperative programs (including non-deterministic, parallel)

▷ Similar to $\mathcal{ALC}$ with role terms (role terms as program fragments)

▷ Domain of interpretation of a DynL formula is the set of states of the processes ($[\![\forall \mathsf{R}.\varphi]\!] =$ "in all states after executing R, φ holds")

$\mathsf{R} \sqcap \mathsf{S}$	parallel execution of R and S
$\mathsf{R} \sqcup \mathsf{S}$	execution of R or S (nondeterministically)
$\mathsf{R} \circ \mathsf{S}$	execution of S after R
$\overline{\mathsf{R}}$	execution of a program that is not R
R^{-1}	execution of an undo operation
$?\psi$	test whether ψ holds (not in $\mathcal{ALC}$)

 182

Tableaux Calculus: $\mathcal{ALC}$ + Role Terms

▷ Definition 5.66: complex roles instead of role names

$$\frac{\begin{array}{c} x\colon \exists \mathsf{R}.\varphi \\ x\;\mathsf{R}\;z \end{array}}{\begin{array}{c} x\;\mathsf{R}\;y \\ y\colon \varphi \end{array}}\;\exists \qquad \frac{\begin{array}{c} \mathcal{B} \\ x\colon \forall \mathsf{R}.\varphi \end{array} \quad \mathcal{B} \models x\;\mathsf{R}\;y}{y\colon \varphi}\;\forall_{\mathsf{R}}$$

▷ Problem: What is $\mathcal{B} \models x\;\mathsf{R}\;y$ ($\mathcal{B}$ is the current branch)

▷ Simple case: no role composition $\circ$ and no converse roles $\cdot^{-1}$.

- then $\mathcal{B} \models x\;\mathsf{R}\;y$, iff $(\{\mathsf{S} \mid x\;\mathsf{S}\;y \in \mathcal{B}\}) \cup \{\overline{\mathsf{R}}\}$ inconsistent in PL0 (decidable)

▷ General case: $\mathcal{B} \models x\;\mathsf{R}\;y$, iff $(\{\mathsf{tr}^{u,v}\mathsf{S} \mid u\;\mathsf{S}\;v \in \mathcal{B}\}) \cup \{\mathsf{tr}^{x,y}(\overline{\mathsf{R}})\}$ inconsistent in PL1 (undecidable in general)

 183

Special Cases for $\mathcal{B} \models x\;\mathsf{R}\;y$

▷ no role composition $\circ$ (decidable)

- then $\mathcal{B} \models x\;\mathsf{R}\;y$, iff $(\{\mathsf{tr}^{x,y}\mathsf{S} \mid x\;\mathsf{S}\;y \in \mathcal{B}\}) \cup \{\mathsf{tr}^{x,y}(\overline{\mathsf{R}})\}$ inconsistent in PL1 (as set of ground formulae).

▷ role complement only for role names (decidable)

- then $(\{\mathsf{tr}^{u,v}\mathsf{S} \mid u\;\mathsf{S}\;v \in \mathcal{B}\})$ is a set of ground formulae and $\mathsf{tr}^{x,y}(\overline{\mathsf{R}})$ only contains constants and variables in the clause normal form.

▷ The general case is undecidable, therefore the naive tableau approach is unsuitable

 184

5.4.5 Role Axioms

General Role Axioms

has_daughter $\sqsubseteq$ has_child	daughters are children
has_son $\sqsubseteq$ has_child	sons are children
has_daughter $\sqcap$ has_son	sons and daughters are disjoint
has_child $\sqsubseteq$ has_son $\sqcup$ has_daughter	children are either sons or daughters

▷ Translation of an axiom ρ: $\mathsf{trr}(\rho) = \forall x, y.\mathsf{tr}^{x,y}(\rho)$

$$\begin{aligned} & \mathsf{trr}(\mathsf{has\_child} \sqsubseteq (\mathsf{has\_son} \sqcup \mathsf{has\_daughter})) \\ = \;& \forall x, y.\mathsf{tr}^{x,y}(\mathsf{has\_child} \sqsubseteq \mathsf{has\_son} \sqcup \mathsf{has\_daughter}) \\ = \;& \forall x, y.\mathsf{has\_child}(x \Rightarrow y) \Rightarrow \mathsf{has\_son}(x \vee y) \vee \mathsf{has\_daughter}(x \vee y) \end{aligned}$$

 185

$\mathcal{ALC}$ + Role Terms + Role Axioms ρ

- Idea: Tableau like for $\mathcal{ALC}$ + role terms ($\mathcal{B}, \rho \models x \mathsf{R} y$ instead of $\mathcal{B} \models x \mathsf{R} y$)
- Simple case: no role composition $\circ$ and no converse roles $\cdot^{-1}$. (decidable)
  - then $\mathcal{B}, \rho \models x \mathsf{R} y$ iff $(\{\mathsf{S} \mid x \mathsf{S} y \in \mathcal{B}\}) \cup \rho \cup \{\overline{\mathsf{R}}\}$ inconsistent in PL0
- General case: $\mathcal{B}, \rho \models x \mathsf{R} y$, iff $(\{\mathsf{tr}^{u,v}\mathsf{S} \mid u \mathsf{S} v \in (\mathcal{B} \cup \mathsf{trr}(\rho) \cup \{\mathsf{tr}^{x,y}(\overline{\mathsf{R}})\})\})$ inconsistent in PL1 (undecidable in general)
- no role composition $\circ$ (decidable)
  - then $\mathcal{B}, \rho \models x \mathsf{R} y$, iff $(\{\mathsf{tr}^{x,y}\mathsf{S} \mid x \mathsf{S} y \in (S \cup \mathsf{trr}(\rho) \cup \{\mathsf{tr}^{x,y}(\overline{\mathsf{R}})\})\})$ inconsistent in PL1 (as set of formulae without functions).
- role complement only for role names (decidable)
  - then $(\{\mathsf{tr}^{u,v}\mathsf{S} \mid u \mathsf{S} v \in \mathcal{B}\})$ is a set of ground formulae and both $\mathsf{tr}^{x,y}(\rho)$ and $\mathsf{tr}^{x,y}(\overline{\mathsf{R}})$ only contain constants and variables in CNF

 186 JACOBS UNIVERSITY

5.4.6 Features

$\mathcal{ALCF}$: Features

- Idea: Features are partial functions.
- Idea: $\mathcal{ALCF}$ is $\mathcal{ALC}$ + features + special constraints on feature paths
- Definition 5.67: Let $\mathcal{F} := \{f, g, f_1, \ldots\}$ be a set of features, then we define the $\mathcal{ALCF}$ formulae by
  $F_{\mathcal{ALCF}} ::= F_{\mathcal{ALC}} \mid \mathsf{R}.F_{\mathcal{ALCF}} \mid \pi\uparrow \mid \pi = \pi \mid \pi \neq \pi$ where $\pi ::= f \mid f \circ \pi$
- Definition 5.68: The semantics of the $\mathcal{ALC}$ part is as always.
  1. The meaning of a feature f is a partial function $[\![f]\!] \colon \mathcal{D} \times \mathcal{D} \to \mathcal{D}$.
  2. $[\![f \circ \pi]\!](x) := [\![\pi]\!]([\![f]\!](x))$
  3. $[\![\pi\uparrow]\!] := (\mathcal{D} \backslash \mathbf{dom}([\![\pi]\!]))$
  4. $[\![f.\varphi]\!] := (\{x \in \mathbf{dom}([\![\pi]\!]) \mid [\![f]\!](x) \in [\![\varphi]\!]\})$
  5. $[\![\varphi = \omega]\!] := (\{x \in (\mathbf{dom}([\![\pi]\!]) \cap \mathbf{dom}([\![\omega]\!])) \mid [\![\pi]\!](x) = [\![\omega]\!](x)\})$
  6. $[\![\varphi \neq \omega]\!] := (\{x \in (\mathbf{dom}([\![\pi]\!]) \cap \mathbf{dom}([\![\omega]\!])) \mid [\![\pi]\!](x) \neq [\![\omega]\!](x)\})$

 187 JACOBS UNIVERSITY

Examples

- ▷ Example 5.69: *persons, whose father is a teacher*: person ⊓ had_father.teacher
- ▷ Example 5.70: *persons that have no father*: person ⊓ $\overline{\text{had\_father}\uparrow}$
- ▷ Example 5.71: *companies, whose bosses have no company car*: company ⊓ has_boss ∘ has_comp_car↑
- ▷ Example 5.72: *cars whose exterior color is the same as the interior color*: car ⊓ color_exterior = color_interior
- ▷ Example 5.73: *cars whose exterior color is different from the interior color*: car ⊓ color_exterior ≠ color_interior
- ▷ Example 5.74: *companies whose Bosses and Vice Presidents have the same company car*: company ⊓ has_boss ∘ has_comp_car = has_VP ∘ has_comp_car

Normalization

- ▷ Normalization rules

$$\begin{array}{rcl} \overline{f.\varphi} & \rightarrow & f\uparrow \sqcup f.\overline{\varphi} \\ \overline{\pi = \omega} & \rightarrow & (\pi\uparrow)\omega\uparrow \sqcup \pi \neq \omega \\ \overline{\pi \neq \omega} & \rightarrow & (\pi\uparrow)\omega\uparrow \sqcup \pi = \omega \\ \overline{f \circ \pi\uparrow} & \rightarrow & f\uparrow \sqcup f \circ \pi\uparrow \end{array}$$

- ▷ Example 5.75: (for the last transformation)

$$\overline{\text{has\_boss} \circ \text{has\_comp\_car} \circ \text{has\_sun\_roof}\uparrow} = \ldots$$

i.e. the set of objects that do not have a boss, plus the set of objects whose boss does not have a company car plus the set of objects whose bosses have company cars without sun roofs

Tableau Calculus

▷ Definition 5.76: The calculus is an extension of $\mathcal{T}_{ALC}$.

$$\frac{x\colon f.\varphi}{\begin{array}{c}x\ f\ y\\ y\colon \varphi\end{array}} \qquad \frac{x\colon \pi = \omega}{\begin{array}{c}x\ \pi\ y\\ x\ \omega\ y\end{array}} \qquad \frac{x\colon \pi \neq \omega}{\begin{array}{c}x\ \pi\ y\\ x\ \omega\ z\\ y \neq z\end{array}} \qquad \frac{x\ f\circ\pi\ y}{\begin{array}{c}x\ f\ y\\ z\ \pi\ y\end{array}} \qquad \frac{\begin{array}{ll}\mathcal{B} & \\ x\ f\ y & \neq y, z\\ x\ f\ z & \end{array}}{[y/z]\mathcal{B}}$$

$$\frac{x\colon \bot}{*} \qquad \frac{\begin{array}{c}x\colon c\\ x\colon \overline{c}\end{array}}{*} \qquad \frac{\begin{array}{c}x\ f\ y\\ x\colon f\!\uparrow\end{array}}{*} \qquad \frac{x \neq x}{*}$$

▷ Theorem 5.77: *The calculus is correct, complete and terminating.*

▷ Theorem 5.78: *It can be implmented in PSPACE*

Example

▷ Example 5.79: has_boss $\circ$ has_comp_car$\uparrow$ $\sqcap$ has_boss.has_comp_car.has_sun_roof.$\top$ is inconsistent.

▷ Normalize: (has_boss$\uparrow$ $\sqcup$ has_boss.has_comp_car$\uparrow$) $\sqcap$ has_boss.has_comp_car.has_sun_roof.$\top$

▷ Tableau

$$\begin{array}{c}
x\colon \text{has_boss}\uparrow \sqcup \text{has_boss.has_comp_car}\uparrow\\
x\colon \text{has_boss.has_comp_car.has_sun_roof.}\top\\
x\ \text{has_boss}\ y\\
y\colon \text{has_comp_car.has_sun_roof.}\top\\
y\ \text{has_comp_car}\ z\\
z\colon \text{has_sun_roof.}\top\\
\begin{array}{c|c}
x\colon \text{has_boss}\uparrow & x\colon \text{has_boss.has_comp_car}\uparrow\\
* & x\ \text{has_boss}\ v\\
 & v\colon \text{has_comp_car}\uparrow\\
 & y\colon \text{has_comp_car}\uparrow \quad (y = v)\\
 & *
\end{array}
\end{array}$$

5.4.7 Concrete Domains

$\mathcal{ALC}$ with "concrete Domains" (Examples)

Formula	Concrete Domain
person $\sqcap$ age < 20	real numbers
persons younger than 20	
company $\sqcap$ has_CEO $\circ$ has_comp_car $\circ$ price) $> \$100000$	natural numbers
companies with CEOs with expensive car	
car $\sqcap$ height $>$ width	natural numbers
cars that are higher than wide	
person $\sqcap$ first_name $<$ last_name	strings
persons whose first name is lexicograhpically smaller than their last name	
person $\sqcap$ has_father $\circ$ studiesbefore(has_mother $\circ$ studies	temporal interval logic
persons whose fathers have studied before their mothers	

Concrete Domain

- **Definition 5.80:** A concrete domain is a pair $\langle\mathcal{C},\mathcal{P}\rangle$, where $\mathcal{C}$ is a set and $\mathcal{P}$ a set of predicates.
- **Example 5.81:**
  - $\mathcal{C} = \mathbb{N}$ and $\mathcal{P} = \{=,<,\leq,>,\geq\}$ (natural numbers)
  - $\mathcal{C} = \mathbb{R}$ and $\mathcal{P} = \{=,<,\leq,>,\geq\}$ (real numbers)
  - $\mathcal{C}$ = temporal intervals, $\mathcal{P} = \{\text{before}, \text{after}, \text{overlaps}, \ldots\}$ (Allen's interval logic)
  - $\mathcal{C}$ = facts in a relational data base, $\mathcal{P}$ = SQL relations

Admissible Concrete Domains

- **Idea:** concrete domains are admissible, iff $\mathcal{P}$ is decidable.
- **Definition 5.82:** Let $\{P_1,\ldots,P_n\} \subseteq \mathcal{P}$, then conjunctions $P_1(x_1,\ldots) \wedge \ldots \wedge P_n(x_n,\ldots)$ are called satisfiable, iff there is a satisfying variable assignment $[a_i/x_i]$ with $a_i \in \mathcal{C}$. (the model is fixed in a concrete domain)
- **Example 5.83:** $\mathcal{C}$ = real numbers

$P_1(x,y) = \exists z.(x+z^2=y)$	satisfiable ($z = \sqrt{y-x}$, e.g. $x=y=1, z=0$)
$P_2(x,y) = P_1(x,y) \wedge x > y$	unsatisfiable

- **Definition 5.84:** A concrete domain $\langle\mathcal{C},\mathcal{P}\rangle$ is called admissible, iff
  1. the satisfiability problem for conjunctions is decidable
  2. $\mathcal{P}$ is closed under negation and contains a name for $\mathcal{C}$.

$\mathcal{ALC}(\mathcal{C})$

▷ Syntax: $F_{\mathcal{ALC}(\mathcal{C})}$:== $F_{\mathcal{ALCF}} \mid P(\pi, \ldots, \pi)$

▷ Example 5.85: a female human under 21 can become a woman by having a child

mother = human ⊓ female ⊓ (∃has_child.human)

woman = human ⊓ female ⊓ (mother ⊔ age ≥ 21)

here age $\geq 21 \in F_{\mathcal{ALC}(\mathcal{C})}$, since it is of the form $P(\text{age})$ $(P = \lambda x.x \geq 21)$

▷

▷ Semantics: Semantics of $\mathcal{ALC}(\mathcal{D})$

- $\mathcal{D}$ and $\mathcal{C}$ are disjoint.
- $P(\pi_1, \ldots, \pi_n) = \left\{ x \in \mathcal{D} \,\middle|\, \begin{array}{l} \text{there are } y_1 = [\![\pi_1]\!](x), \ldots, y_n = [\![\pi_n]\!](x) \in \mathcal{C} \\ \text{with } \langle y_1, \ldots, y_n \rangle \in [\![P]\!] \end{array} \right\}$

Warning: $[\![\overline{\varphi}]\!] = \mathcal{D} \backslash [\![\varphi]\!]$, but not $[\![\overline{\varphi}]\!] = \mathcal{D} \cup \mathcal{C} \backslash [\![\varphi]\!]$

 195 JACOBS UNIVERSITY

!

Negation Rules and Tableau Calculus

▷ Let $\top_{\mathcal{C}}$ be the name for the concrete domain (as a set) and $\overline{P}$ the negated predicate for P ($\mathcal{C}$ is admissible)

▷ New negation rule: $\overline{P(\pi_1, \ldots, \pi_n)} \rightarrow \overline{P}(\pi_1, \ldots, \pi_n) \sqcup (\forall \pi_1.\top_{\mathcal{C}}) \sqcup \ldots \sqcup (\forall \pi_n.\top_{\mathcal{C}})$

▷ New tableau rule

$$\frac{\begin{array}{c} P_1(x_{11}, \ldots, x_{1n_1}) \\ \vdots \\ P_k(x_{k1}, \ldots, x_{kn_k}) \end{array} \quad \bigwedge_{1 \leq i \leq k} P_i(x_{i1}, \ldots, x_{in_i}) \quad \text{inconsistent}}{*} \perp^p$$

 196 JACOBS UNIVERSITY

Example: $\mathsf{car} \sqcap \mathsf{height} = 2 \sqcap \mathsf{width} = 1 \sqsubseteq \mathsf{car} \sqcap \mathsf{height} > \mathsf{width}$

$$\begin{array}{c}
x\colon \mathsf{car} \sqcap \mathsf{height} = 2 \sqcap \mathsf{width} = 1 \\
x\colon \overline{\mathsf{car}} \sqcap \mathsf{width} \leq \mathsf{height} \\
x\colon \mathsf{car} \\
x\colon \mathsf{height} = 2 \\
x\colon \mathsf{width} = 1
\end{array}$$

$$\begin{array}{c|c}
x\colon \overline{\mathsf{car}} & x\colon \mathsf{width} \leq \mathsf{height} \\
* & x \mathsf{height}\ y_1 \\
 & y_1 = 2 \\
 & x\colon \mathsf{width} = y_2 \\
 & y_2 = 1 \\
 & x\colon \mathsf{width}\ y_3 \\
 & x\colon \mathsf{height} = y_4 \\
 & y_3 \leq y_4 \\
 & y_1 \leq y_2 \\
 & *
\end{array}$$

 197 JACOBS UNIVERSITY

5.4.8 Nominals

Nominals

- **Definition 5.86:** (Idea)
  nominal are names for domain elements that can be used in the T-Box.
- **Example 5.87:** *Students that study on Bremen or Hamburg*: $\mathsf{student} \sqcap (\exists \mathsf{studies\_in}.\{Bremen, Hamburg\})$
- **Example 5.88:** *Students that have a friend with name Eva*: $\mathsf{student} \sqcap (\exists \mathsf{has\_friend} \circ \mathsf{has\_name}.\{Eva\})$
- **Example 5.89:** *persons that have phoned Bill, Bob, or the murderer*: $\mathsf{person} \sqcap (\exists \mathsf{has\_phoned}.\{Bill, Bob, murderer\})$
- **Example 5.90:** *friends of Eva*: $\mathsf{person} \sqcap \mathsf{has\_friend}\colon Eva$
- **Example 5.91:** *companies whose employees all bank at Sparda Bank*: $\mathsf{company} \sqcap (\forall \mathsf{has\_empl}.\mathsf{has\_bank}\colon Sparda)$
- **Example 5.92:** *employees of Jacobs that bank at Sparda*: $\mathsf{employed\_at}\colon Jacobs \sqcap \mathsf{has\_bank}\colon Sparda$

 198 JACOBS UNIVERSITY

Semantics

▷ Definition 5.93: $[\![\{a_1,\ldots,a_n\}]\!]$ is the set of objects with names $a_1,\ldots a_n$.

▷ Definition 5.94: $[\![\mathsf{R}\colon a]\!]$ is the set of objects that have $[\![a]\!]$ as R-successor

$$\begin{array}{lcl} [\![\{a_1,\ldots,a_n\}]\!] & = & \{[\![a_1]\!],\ldots,[\![a_n]\!]\} \\ [\![\mathsf{R}\colon a]\!] & = & (\{x\in\mathcal{D} \mid \langle x,[\![a]\!]\rangle \in [\![\mathsf{R}]\!]\}) \end{array}$$

▷ Definition 5.95: (Negation Rules)

$$\begin{array}{rcl} \overline{\{a_1,\ldots,a_n\}} & = & \text{invariant} \\ \overline{\mathsf{R}\colon a} & = & \forall\mathsf{R}.\overline{\{a\}} \end{array}$$

▷ Example 5.96: $\overline{\mathsf{had\_friend}\colon Eva}$ (the complement of the set of friends of Eva)

$= \forall\mathsf{had\_friend}.\overline{\{Eva\}}$ (the set of objects that do not have Eva as a friend)

Example Language with Nominals

▷ We consider the following language: $\mathcal{ALC}$ + unquantified number restrictions ($\exists^n_{\geq}\mathsf{R}$, $\forall^n_{\leq}\mathsf{R}$), some role operators ($\sqcap$, $\circ$, $\cdot^{-1}$), $\{a_1,\ldots,a_n\}$, $\mathsf{R}\colon a$

▷ Example 5.97: *persons that have at most two friends among their neighbors and whose neighbors are Bill, Bob, or the gardener* $\mathsf{person} \sqcap \forall^2_{\leq}(\mathsf{has\_friend} \sqcap \mathsf{has\_neighbor}) \sqcap (\forall\mathsf{has\_neighbor}.\{Bill, Bob, Gardener\})$

▷ Example 5.98: *companies with at least 100 employees that have a car and live in Bremen* $\mathsf{company} \sqcap \exists^{100}_{\geq}\mathsf{has\_empl} \circ \mathsf{has\_comp\_car} \sqcap \mathsf{has\_empl} \circ \mathsf{lives\_in}\colon Bremen$

Tableaux Calculus (only T-Box)

▷ Definition 5.99: The calculus consists of the $\mathcal{T}_{ALC}$ rules together with:

$$\frac{a\colon\overline{\{\ldots,a,\ldots\}}}{*} \qquad \frac{\begin{array}{c}\mathcal{B}\\ x\colon\{a_1,\ldots,a_n\}\end{array}}{[x/a_1]\mathcal{B} \;\Big|\; \ldots [x/a_n]\mathcal{B}} \qquad \frac{x\colon\mathsf{R}\colon a}{x\;\mathsf{R}\;a}$$

$$\frac{x\;\mathsf{R}^{-1}\;y}{y\;\mathsf{R}\;x} \qquad \frac{x\;\mathsf{R}\sqcap\mathsf{S}\;y}{\begin{array}{c}x\;\mathsf{R}\;y\\ x\;\mathsf{S}\;y\end{array}} \qquad \frac{x\;\mathsf{R}\circ\mathsf{S}\;y}{\begin{array}{c}x\;\mathsf{R}\;z\\ z\;\mathsf{S}\;y\end{array}}$$

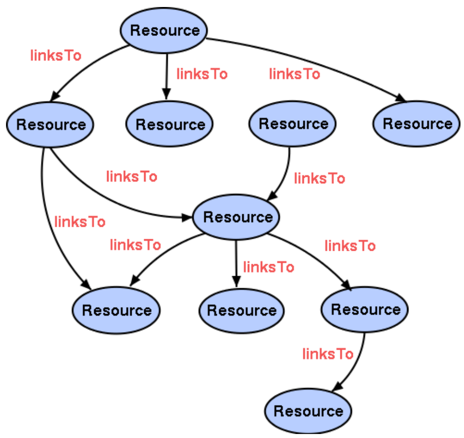
▷ Theorem 5.100: *The calculus is correct, complete, and terminating*

▷ Proof: very technical but not terribly difficult using the techniques developed so far. □

5.5 The Semantic Web

The Current Web

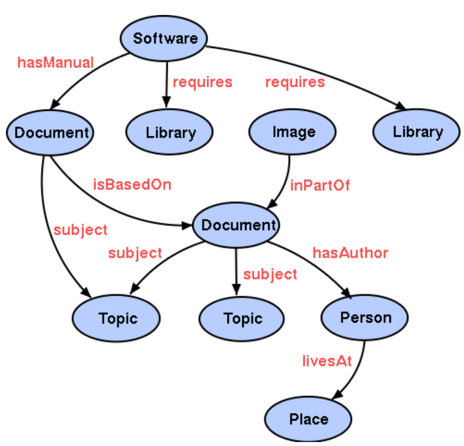
- ▷ Resources: identified by URI's, untyped
- ▷ Links: `href`, `src`, . . . limited, non-descriptive
- ▷ User: Exciting world - semantics of the resource, however, gleaned from content
- ▷ Machine: Very little information available - significance of the links only evident from the context around the anchor.



 202

The Semantic Web

- ▷ Resources: Globally Identified by URI's or Locally scoped (Blank), Extensible, Relational
- ▷ Links: Identified by URI's, Extensible, Relational
- ▷ User: Even more exciting world, richer user experience
- ▷ Machine: More processable information is available (Data Web)
- ▷ Computers and people: Work, learn and exchange knowledge effectively



 203

What is the Information a User sees?

WWW2002
The eleventh international world wide web conference
Sheraton waikiki hotel
Honolulu, hawaii, USA
7-11 may 2002
1 location 5 days learn interact

Registered participants coming from
australia, canada, chile denmark, france, germany, ghana, hong kong, india,
ireland, italy, japan, malta, new zealand, the netherlands, norway,
singapore, switzerland, the united kingdom, the united states, vietnam, zaire

On the 7th May Honolulu will provide the backdrop of the eleventh
international world wide web conference. This prestigious event ?
Speakers confirmed
Tim Berners-Lee: Tim is the well known inventor of the Web, ?
Ian Foster: Ian is the pioneer of the Grid, the next generation internet ?

What the machine sees

𝒲𝒲𝒲∈′′∈
𝒯⟨⌉⌉↕⌉⊑⌉\⊔⟨⟩\⊔⌉∇\⊣⊔⟩≀\⊣↕⊒≀∇↕⌈⊒⟩⌈⌉⊒⌉⌊⌋≀\{⌉∇⌉\⌋⌉
𝒮⟨⌉∇⊣⊔≀\⊒⊣⟩∥⟩∥⟩⟨≀⊔⌉↕
ℋ≀\≀↕⊓↕⊓⇔⟨⊣⊒⊣⟩⟩⇔𝒰𝒮𝒜
↖∞∞⇕⊣†∈′′∈

ℛ⌉}⟩∫⊔⌉∇⌉⌈√⊣∇⊔⟩⌋⟩√⊣\⊔∫⌋≀⇕⟩\}{∇≀⇕
⊣⊓∫⊔∇⊣↕⟩⊣⇔⌋⊣\⊣⌈⊣⇔⌋⟨⟩↕⌉⌈⌉\⇕⊣∇∥⇔{∇⊣\⌋⌉⇔}⌉∇⇕⊣\†⇔}⟨⊣\⊣⇔⟨≀\}∥≀\}⇔⟩\⌈⟩⊣⇔
⟩∇⌉↕⊣\⌈⇔⟩⊔⊣↕†⇔|⊣√⊣\⇔⇕⊣↕⊔⊣⇔\⌉⊒‡⌉⊣↕⊣\⌈⇔⊔⟨⌉\⌉⊔⟨⌉∇↕⊣\⌈∫⇔\≀∇⊒⊣†⇔
∫⟩\}⊣√≀∇⌉⇔∫⊒⟩⊔‡⌉∇↕⊣\⌈⇔⊔⟨⌉⊓\⟩⊔⌉⌈∥⟩\}⌈≀⇕⇔⊔⟨⌉⊓\⟩⊔⌉⌈∫⊔⊣⊔⌉∫⇔⊑⟩⌉⊔\⊣⇕⇔‡⊣⟩∇⌉

𝒪\⊔⟨⌉⊔⟨ℳ⊣†ℋ≀\≀↕⊓↕⊓⊒⟩↕↕√∇≀⊑⟩⌈⌉⊔⟨⌉⌊⊣⌋∥⌈∇≀√≀{⊔⟨⌉⌉↕⌉⊑⌉\⊔⟨
⟩\⊔⌉∇\⊣⊔⟩≀\⊣↕⊒≀∇↕⌈⊒⟩⌈⌉⊒⌉⌊⌋≀\{⌉∇⌉\⌋⌉↙𝒯⟨⟩∫√∇⌉∫⊔⟩}⟩≀⊓∫⌉⊑⌉\⊔⊥

𝒮√⌉⊣∥⌉∇∫⌋≀\{⟩∇⇕⌉⌈
𝒯⟩⇕⌊⌉∇\⌉∇∫↖↕⌉⌉¬𝒯⟩⇕⟩∫⊔⟨⌉⊒⌉↕↕∥\≀⊒\⟩\⊑⌉\⊔≀∇≀{⊔⟨⌉𝒲⌉⌊⇔⊥
ℐ⊣\ℱ≀∫⊔⌉∇¬ℐ⊣\⟩∫⊔⟨⌉√⟩≀\⌉⌉∇≀{⊔⟨⌉𝒢∇⟩⌈⇔⊔⟨⌉\⌉§⊔}⌉\⌉∇⊣⊔⟩≀\⟩\⊔⌉∇\⌉⊔⊥

Solution: XML markup with "meaningful" Tags

<title>[illegible]</title>
<place>[illegible]</place>
<date>[illegible]</date>
<participants>[illegible]</participants>
</introduction>[illegible]</introduction>
<program>[illegible]
<speaker>[illegible]</speaker>
<speaker>[illegible]<speaker></program>

 206 JACOBS UNIVERSITY

What the machine sees of the XML

[illegible]

 207 JACOBS UNIVERSITY

Need to add "Semantics"

- ▷ External agreement on meaning of annotations E.g., Dublin Core
  - ▷ Agree on the meaning of a set of annotation tags
  - ▷ Problems with this approach: Inflexible, Limited number of things can be expressed
- ▷ Use Ontologies to specify meaning of annotations
  - ▷ Ontologies provide a vocabulary of terms
  - ▷ New terms can be formed by combining existing ones
  - ▷ Meaning (semantics) of such terms is formally specified
  - ▷ Can also specify relationships between terms in multiple ontologies

 208 JACOBS UNIVERSITY

5.6 Description Logics and the Semantic Web

Resource Description Framework

- ▷ Definition 5.101: The Resource Description Framework (RDF) is a framework for describing resources on the web. It is a XML vocabulary developed by the W3C.
- ▷ Note: RDF is designed to be read and understood by computers, not to be being displayed to people
- ▷ Example 5.102: RDF can be used for describing
  - ▷ properties for shopping items, such as price and availability
  - ▷ time schedules for web events
  - ▷ information about web pages (content, author, created and modified date)
  - ▷ content and rating for web pictures
  - ▷ content for search engines
  - ▷ electronic libraries

 209 JACOBS UNIVERSITY

Resources and URIs

- ▷ RDF describes resources with properties and property values.
- ▷ RDF uses Web identifiers (URIs) to identify resources.
- ▷ Definition 5.103: A resource is anything that can have a URI, such as `http://www.jacobs-university.de`
- ▷ Definition 5.104: A property is a resource that has a name, such as *author* or *homepage*, and a property value is the value of a property, such as *Michael Kohlhase* or `http://kwarc.info/kohlhase` (a property value can be another resource)
- ▷ Definition 5.105: The combination of a resource, a property, and a property value forms a statement (known as the subject, predicate and object of a statement).
- ▷ Example 5.106: Statement: *The [author]^{pred} of [this slide]^{subj} is [Michael Kohlhase]^{obj}*

 210 JACOBS UNIVERSITY

XML Syntax for RDF

- ▷ RDF is a concrete XML vocabulary for writing statements
- ▷ Example 5.107: The following RDF document could describe the slides as a resource

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22−rdf−syntax−ns#"
      xmlns:dc= "http://purl.org/dc/elements/1.1/">
  <rdf:Description about="https://svn.kwarc.info/.../slides/kr/en/rdf.tex">
    <dc:creator>Michael Kohlhase</dc:creator>
    <dc:source>http://www.w3schools.com/rdf</dc:source>
  </rdf:Description>
</rdf:RDF>
```

▷ This RDF document makes two statements:

- ▷ The subject of both is given in the about attribute of the `rdf:Description` element
- ▷ The predicates are given by the element names of its children
- ▷ The objects are given in the elements as URIs or literal content.

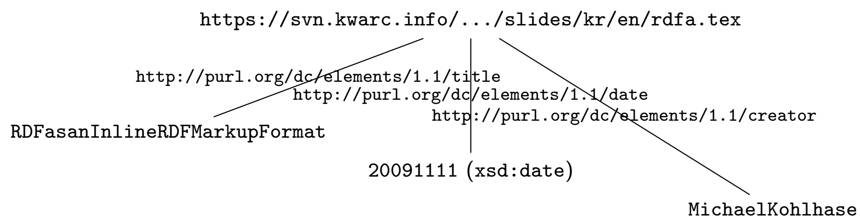
Intuitively: RDF is a way to write down ABox information in a web-scalable way.

 211 JACOBS UNIVERSITY

RDFa as an Inline RDF Markup Format

- ▷ Problem: RDF is a standoff markup format (annotate by URIs pointing into other files)
- ▷ Example 5.108:

```
<div xmlns:dc="http://purl.org/dc/elements/1.1/">
    <h2 property="dc:title">RDF as an Inline RDF Markup Format</h2>
    <h3 property="dc:creator">Michael Kohlhase</h3>
    <em property="dc:date" datatype="xsd:date"
        content="20091111">November 11., 2009</em>
</div>
```



OWL as an Ontology Language for the Semantic Web

- ▷ Idea: Use Description Logics to talk about RDF triples.
- ▷ An RDF triple is an ABox entry for a role contraint $h\mathrm{R}s$
- ▷ Example 5.109: h is the resource for Ian Horrocks, s is the resource for Ulrike Sattler, and R is the the relation “hasColleague” in

```
<rdf:Description about="some.uri/person/ian_horrocks">
  <hasColleague resource="some.uri/person/uli_sattler"/>
</rdf:Description>
```

Idea: Now collect similar resources in classes, and state rules about them in a way, so that we can use inference to make kwnowledge explicit that was implicit before (saves us lots of work!)

- ⊳ Idea: We know how to do this, this is just $\mathcal{ALC}$+!!!

The OWL Language

- Three species of OWL
  - OWL Full is union of OWL syntax and RDF
  - OWL DL restricted to FOL fragment
  - OWL Lite is "easier to implement" subset of OWL DL
- Semantic layering
  - OWL DL $\hat{=}$ OWL Full within DL fragment
  - DL semantics officially definitive
  - OWL DL based on SHIQ Description Logic($\mathcal{ALC}$ + nubmer restrictions, transitive roles, inverse roles, role inclusion
  - OWL DL benefits from many years of DL research
  - Well defined semantics, formal properties well understood (complexity, decidability)
  - Known reasoning algorithms, Implemented systems (highly optimized)

 214 JACOBS UNIVERSITY

References

[Tob00] Stephan Tobies. PSpace reasoning for graded modal logics. *Journal of Logic and Computation*, 11:85–106, 2000.