



Methods of AI

Practice Session 9
Kai-Uwe Kühnberger
Jan. 10th, 2003



Today

- As usual: Organizational issues
- Logic
 - Predicate logic
 - Some Facts about Herbrand theory
 - Non-monotonic reasoning
 - Some examples and ideas
- Some remarks concerning knowledge representation
 - KL-ONE



Organization

- Further plans

- Today

- Assignment 6 (Deadline: 17.01.2003)

- 17.01.2003

- Assignment 7 (Deadline: 24.01.2003)
 - Programming assignment 3 (Deadline: 31.01.2003)

- 24.01.2003:

- Assignment 8 (Deadline: 31.01.2003)

- Final Exam

- When: 07.02.2003, 12:00am and 2:00pm
 - Where: Same place as last time
 - Would that be ok for all of you???



Assignment 6

- Exercise 1
 - Formalize and apply logical reasoning
 - Think about the law of excluded middle
- Exercise 2
 - Straightforward
- Exercise 3 + 4
 - Write down your own opinion
 - I don't think that there are true and false answers in a strict sense



Logic

Predicate Logic



Some remarks

- Prenex-form
 - Putting quantifiers in front of a formula is governed by laws
 - $\exists x(\varphi \vee \psi) \equiv \exists x\varphi \vee \psi$ provided x is not free in ψ
 - But: $[\forall x(\exists y \varphi(x,y) \rightarrow \psi(x))] \not\equiv \forall x\exists y (\varphi(x,y) \rightarrow \psi(x))$
 - Why?
- Why is it possible to do predicate logic „without quantifiers“?
 - In the resolution algorithm no quantifiers play any role...



Herbrand Theory

- The foundation of resolution
 - Herbrand universe
 - Def.: The Herbrand universe $D(\varphi)$ of a formula φ contains
 - (i) All constants of φ (if there are no constants, a is in $D(\varphi)$)
 - (ii) For each f (n -ary) in φ and t_1, t_2, \dots, t_n in $D(\varphi)$, the term $f(t_1, t_2, \dots, t_n)$ is in $D(\varphi)$
 - Example: $\varphi = \forall x \forall y \forall z: Q(x, f(y), g(z, x))$
 - Herbrand structure
 - Def.: For a formula φ , a pair $M = \langle U, I \rangle$ is called a Herbrand structure for φ , if
 - (i) $U = D(\varphi)$
 - (ii) For each f in φ and all t_1, t_2, \dots, t_n in $D(\varphi)$ it holds:
[[$f(t_1, t_2, \dots, t_n)$]] = $f(t_1, t_2, \dots, t_n)$



Herbrand Theory

- Herbrand expansion

- Definition

- Assume $\varphi = \forall y_1 \forall y_2 \dots \forall y_n \varphi^*$ is in Skolem form. The Herbrand expansion $E(\varphi)$ is defined as follows:

$$E(\varphi) = \{\varphi^*[y_1/t_1] \dots [y_n/t_n] \mid t_1, \dots, t_n \in D(\varphi)\}$$

- Idea of the expansion: Substitute terms in $D(\varphi)$ for each variable in φ^*
 - Notice: $E(\varphi)$ contains no longer variables. Hence you can work with them as with propositional logic formulas
 - Notice further that you generate infinitely many formulas in general



Herbrand Theory

- Example for a Herbrand expansion
 - $\varphi = \forall x \forall y \forall z Q(x, f(y), g(z, x))$
 - Here are some formulas of $E(\varphi)$
 - $Q(a, f(a), g(a, a))$
 - $Q(f(a), f(a), g(a, f(a)))$
 - $Q(a, f(f(a)), g(a, a))$
 - $Q(a, f(a), g(f(a), a))$
 - Etc.

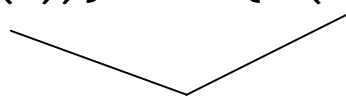


Herbrand Theory

- Theorem
 - Assume φ is in Skolem form. Then, φ is satisfiable iff φ has a Herbrand model
- Theorem (Gödel-Herbrand-Skolem)
 - Assume φ is in Skolem form. Then, φ is satisfiable iff $E(\varphi)$ is satisfiable in the sense of propositional logic
- Theorem (Herbrand)
 - A formula in Skolem form φ is not satisfiable iff there is a finite subset of $E(\varphi)$ that is not satisfiable in the sense of propositional logic



An Example of Resolution

- Assume the following formula is given
 - $\varphi = \forall x(Q(x) \wedge \neg Q(f(x)))$
 - Skolem form: $\varphi^* = Q(x) \wedge \neg Q(f(x))$
 - Clause form: $\varphi^* = \{\{Q(x)\}, \{\neg Q(f(x))\}\}$
 - $D(\varphi) = \{a, f(a), f(f(a)), \dots\}$
 - $E(F) = \{Q(a) \wedge \neg Q(f(a)), Q(f(a)) \wedge \neg Q(f(f(a))), \dots\}$
 - $\{Q(a)\} \quad \{\neg Q(f(a))\} \quad \{Q(f(a))\} \quad \{\neg Q(f(a))\}$

- We have immediately found a non-satisfiable set of clauses



More facts about predicate logic

- Predicate logic is complete
 - There is the well-known one-to-one relation between semantic model and syntactic calculus
- Predicate logic is semi-decidable
 - There is no algorithm that decides whether a given formula is a theorem
 - There is an algorithm that decides whether a given formula is not a theorem



Non-monotonic reasoning

Ideas of some Frameworks for
Non-monotonic Reasoning



Types of solutions

- CWA (closed world assumption)
 - Idea: When there is no information about a certain fact, the negation of this fact is true
 - $T_{CWA}(W) = T(W \cup \overline{W})$ where $\overline{W} = \{\neg G \mid G \text{ ground atom and } W \not\models G\}$
 - Problem: Under certain circumstances CWA results in inconsistencies
- Default logic
 - Idea: Two types of rules
 - First type: Classical logical rules
 - Second type: Default rules that can be overridden



Types of Solutions

- Many-valued logics
 - Idea: there are statements that are neither true nor false but something else
 - Notice: sometimes these logics have no tautologies
- Relevance logic
 - Idea: For a valid inference, the premise must be relevant for the conclusion
 - Difference to classical logic:
 - $\neq (p \wedge \neg p) \rightarrow q$
 - $\neq p \rightarrow (q \vee \neg q)$



Types of Solutions

- Circumscription
 - Idea: the extension of a certain predicate needs to be minimized
 - Usually results in a second order theory
- Modal logic
 - Idea: A modal operator can be used to express „A statement φ is consistent“
 - $\text{Child}(x) \wedge M(\text{likes_ice_cream}(x)) \rightarrow \text{likes_ice_cream}(x)$
 - If x is a child and it is consistent that x likes ice cream, then x likes ice cream



Types of Solutions

- A good introduction to different frameworks of non-monotonic reasoning can be found in:
 - W. Bibel: Wissensrepräsentation und Inferenz. Eine grundlegende Einführung, Braunschweig, Wiesbaden 1993.
 - Especially in Chapter 3 a whole bunch of different accounts is presented



Remarks concerning Knowledge Representation

KL-ONE



Representations

- The following list summarizes some important representation formalisms
 - Predicate logic
 - Other logics
 - Semantic nets
 - Frames
 - Terminological systems
 - KL-ONE language family
 - PROLOG
 - Etc.



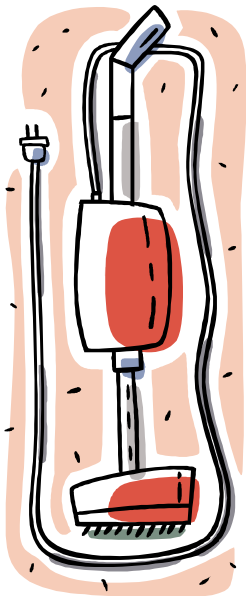
A representation formalism

- Language

C	concept symbol
$C \sqcap C'$	conjunction of concepts
$C \sqcup C'$	disjunction of concepts
$\neg C$	negation of concept
$\forall R:C$	Value restriction
$\exists R:C$	Existential value restriction

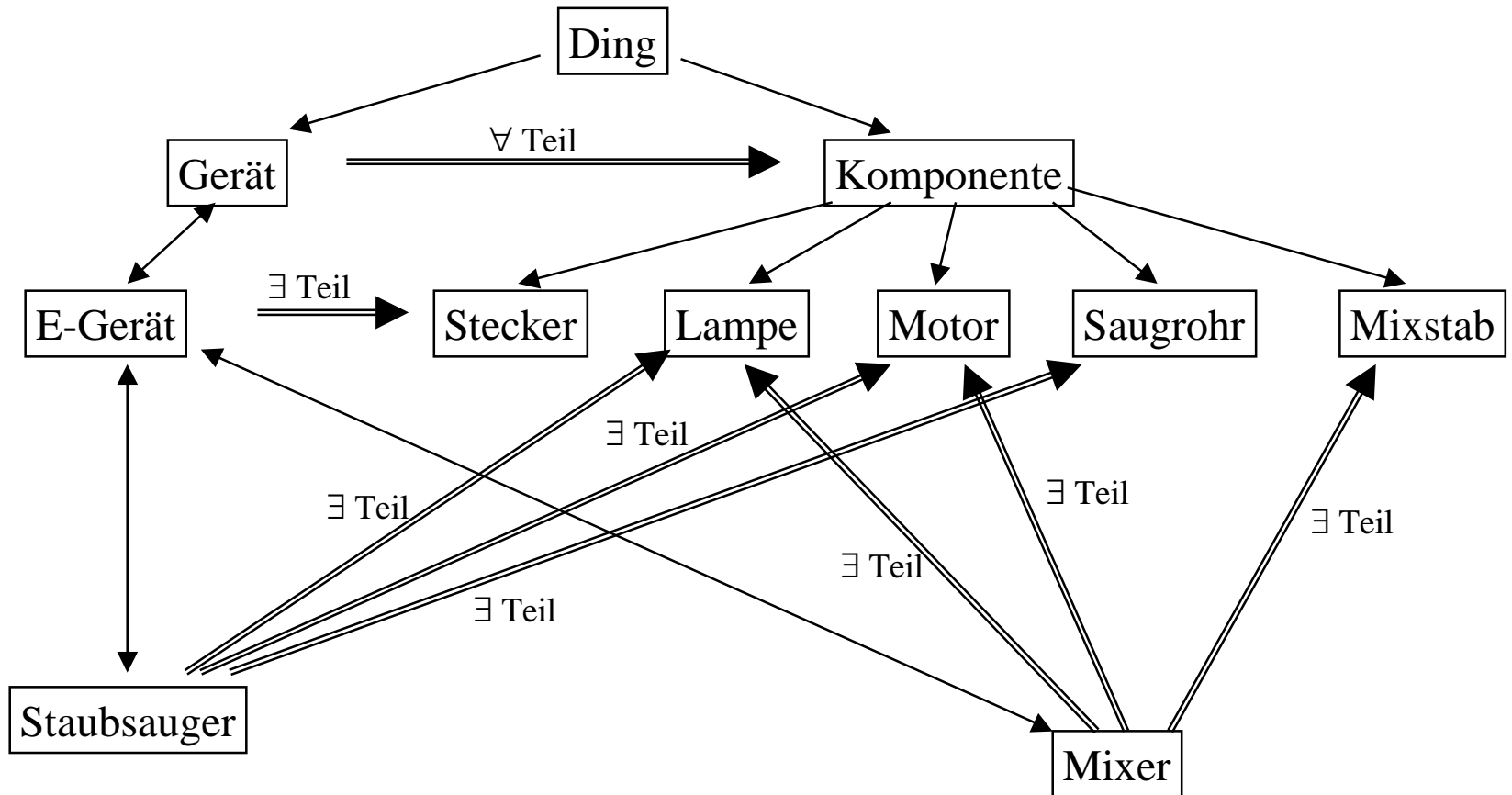
- Example: $\text{Mann} \sqcap \text{Elternteil}$ denotes Vater
- $\forall \text{Kinder}$: Mann denotes someone who has only male children

Terminology



Komponente	\sqsubseteq	Ding	\sqsubseteq	(partial definition)
Motor	\sqsubseteq	Komponente	\doteq	(total definition)
Lampe	\sqsubseteq	$\text{Komponente} \sqcap \neg\text{Motor}$		
Stecker	\sqsubseteq	$\text{Komponente} \sqcap \neg\text{Motor} \sqcap \neg\text{Lampe}$		
Saugrohr	\sqsubseteq	$\text{Komponente} \sqcap \neg\text{Motor} \sqcap \neg\text{Lampe} \sqcap \neg\text{Stecker}$		
Mixstab	\sqsubseteq	$\text{Komponente} \sqcap \neg\text{Motor} \sqcap \neg\text{Lampe} \sqcap \neg\text{Stecker} \sqcap \neg\text{Saugrohr}$		
Geraet	\sqsubseteq	$\forall\text{Teil: Komponente} \sqcap \text{Ding} \sqcap \neg\text{Komponente}$		
E-Geraet	\doteq	$\text{Geraet} \sqcap \exists\text{Teil: Stecker}$		
Staubsauger	\doteq	$\text{E-Geraet} \sqcap \forall\text{Teil: Komponente} \sqcap \exists\text{Teil: Motor} \sqcap \exists\text{Teil: Lampe} \sqcap \exists\text{Teil: Saugrohr}$		
Mixer	\doteq	$\text{E-Geraet} \sqcap \forall\text{Teil: Komponente} \sqcap \exists\text{Teil: Motor} \sqcap \exists\text{Teil: Lampe} \sqcap \exists\text{Teil: Mixstab}$		

Terminology





Semantics

- Why semantics at all?
 - We can communicate the importance of representation formalisms
 - Expressive power can be specified
 - Behavior of systems can be compared with representation language
 - Computability can be checked
 - Inference algorithms can be developed and the soundness and completeness can be proven
 - Etc.



Semantics of our System

Representation formalism	Logic
$C_1 \sqcap C_2$	$C_1(x) \wedge C_2(x)$
$C_1 \sqcup C_2$	$C_1(x) \vee C_2(x)$
$\neg C$	$\neg C(x)$
$\forall R:C$	$\forall y: R(x,y) \Rightarrow C(y)$
$\exists R:C$	$\exists y: R(x,y) \wedge C(y)$
$B \sqsubseteq C$	$\forall x: B(x) \Rightarrow C(x)$
$B \doteq C$	$\forall x: B(x) \Leftrightarrow C(x)$
$a \in C$	$C(a)$
$\langle a,b \rangle \in R$	$R(a,b)$



Inferences

- Questions:
 - How should an algorithm that is able to recognize a Staubsauger look like?
 - How should an algorithm that is able to build a Staubsauger look like?
- If we have a decision algorithm, then
 - If the algorithm gives positive answer, then $a \in C$
 - If $a \in C$, then algorithm gives positive answer
 - The algorithm terminates