1. Introduction

Freitag, 10. April 2015  08:30

Logic Programming

Learning several prog. languages:
- express ideas during SW development
- needed to decide which language to use in project
- eases learning of new languages
- needed to design new languages

Imper. + Fct. Languages:
- programs compute functions

Logic Languages:
- programs describe relations
- execution: ask queries, program tries to prove queries
- main application area: AI, expert systems, deductive databases,

Prolog - Implementation: SWI Prolog
(see web page)

Example: Family Tree
- : married
∪ : children

Facts and Queries

Knowledge has to be translated to Prolog - Syntax.

Prolog = Programming in Logic

Prolog program consists of (special) logic for -
unary, so-called **clauses**:
- **facts**
- **rules** (allow to deduce new knowledge from existing knowledge)

**Syntax of facts:**

```
Boolean
predicate (obj1, ..., objn) :- statements
```

1. Strings starting with lowercase letter
2. Relations are not symmetric.

**Syntax for comments:**

```
% ... end of line
/

; ;
```

**Execution:** ask queries

```
?- statement.
```

**Closed World Assumption:**

*everything that can't be deduced from program clauses is false*

### Variables in Programs

**Variables:** strings starting with capital letter or with

**Variables in programs are universally quantified**

**Ex:** all X are human

(i.e. everything is human)

*Same variables in one clause have to be instantiated in the same way:*
Likes (X, X). - everybody likes himself

Likes (X, Y). - everybody likes everybody

Variables in Queries

Variables in queries are existentially quantified - can be used to let the program compute solutions.

Ex: Who is the mother of Susanne?
(Is there an X such that ... ?)

Prolog returns a suitable answer substitution.

If there are several solutions: ‘;’ makes Prolog continue searching for answers.
Prolog searches through its prog. clauses from top to bottom.

Some program can be used to compute mothers or children =>

Prolog programs have no fixed input/output, but input/output depends on query.

?- motheroft (X, Y).
X = monika , Y = karin .

?- human (Z).  Prolog returns the most general inst. that make the query true.
true

Combined Queries
### Combined Queries

\[ \land \quad \lor \]

**Ex:** Is gerd the father of susanne?

**Combined queries are executed from left to right.**

- First solve query \( \text{married}(\text{gerd}, W) \) \( \Rightarrow \) finds an instantiation of \( W \)
- Then solve second query \( \text{motherOf}(W, \text{susanne}) \) for this instantiation of \( W \)
- If second query fails, then backtrack to the first query and try the next solution.
- Rete computes a proof tree (so-called SLD tree)

\[
\begin{align*}
mO(6, M), & \quad mO(M, a) \\
G = \text{monika} & \quad G = \text{monika} \\
M = \text{karin} & \quad M = \text{karin} \\
G = \text{rene} & \quad M = \text{susanne} \\
\text{mo(} & \text{karin, alive)} \quad \text{mo(} & \text{karin, alive)} \\
\text{y} & \quad \text{y} \\
\text{mO(} & \text{klaus, alive)} & \text{mO(} & \text{klaus, alive)} \\
& \quad \text{mO(} & \text{sus, al}) & \\
\text{empty clause} & \Rightarrow & \square & \\
\quad & \text{means that this path of the proof tree was successful}
\end{align*}
\]

### Rules

Rules allow to deduce new knowledge from existing knowledge.

**Ex:** \( F \) is the father of \( C \) if \((?:-)

there exists a \( W \) such that
F is married to W and
W is the mother of C.

Rules:
\[
\text{head} : = \underbrace{\text{statement}, \ldots, \text{statement}}_{\text{body of the rule}}.
\]

means: in order to prove head,
one can instead prove the statements
in the body.

Ex:
\[
fO(\text{gend}, Y) \\
| \text{F = gend, C = Y} \\
\quad \text{married (gend, W), motherOf (W, Y)} \\
| \text{W = renate} \\
\quad \text{motherOf (renate, Y)} \\
\quad \vdash Y = \text{susanne} \quad \vdash Y = \text{peter}
\]

Several rules for the same predicate

alternative:
\[
\text{parent (X,Y)} : = \text{motherOf (X,Y) ; fatherOf (X,Y).}
\]

( ; is defined by 2 clauses)

?- parent (X, susanne).

Mother will be found first due to the order
of prog. clauses.

Recursive Rules

Ex: ancestor predicate
2nd rule is recursive
Characteristics of Logic Programming:

- no control structures, just facts + rules
- prog. execution is automated theorem proving
- particularly suitable for AI

Plan for the lecture:
Ch. 1: Introduction to LP
Ch 2: Predicate Logic
Ch 3: Resolution (Proof Technique used in LP)
Ch 4: Syntax Semantics of LP
Ch 5: Prog. Language Prolog

Organisation

- english
- german course notes (web)
- english notes from the lecture + slides (web)
- lecture: 8:30 - 10:00 Mon.
- exercise: 10:15 - 11:45 Fri.
- video recording from 2013
• V3+U2 lecture, 2 variants (for Bachelor & Master Students)
  called V3B + V3M (Math students: V3B)
• Web site: http://verify.rwth-aachen.de/lp15
• Exercises:
  • weekly exercise sheet
  • groups of 2
  • registering for exercises: via our web site
    (until Friday next week)
• 50% of exercise points needed to participate in the exam
• exam: August 19 + September 14
• Vorgetogene Masterprüfung: register via EPA
  (June 8 - 18)