

Computational Logic

Bagian Kedua:

Logic Programming

- ◆ Pengajar : L. Yohanes Stefanus
email: yohanes@cs.ui.ac.id
kantor: Fasilkom Gedung A Ruang 1220
- ◆ Jadwal : Selasa & Kamis, 14:00 - 16:00
Diselingi istirahat 15 menit.
- ◆ World Wide Web: <http://telaga.cs.ui.ac.id/WebKuliah/logic/>
- ◆ Tujuan : Mengajarkan konsep dan teknik *logic programming* dengan bahasa Prolog (versi ISO).



◆ Textbook:

Ivan Bratko.
Prolog Programming for Artificial Intelligence, 3rd Edition.
Addison-Wesley, 2001.

K. R. Apt
From Logic Programming to Prolog.
Prentice Hall, 1997.



◆ Software :

Implementasi Prolog yang dipakai adalah SWI-Prolog versi 5.4.7.
Bisa di-download dari <http://www.swi-prolog.org>.
SWI-Prolog dibangun oleh Jan Wielemaker dari University of Amsterdam. Ada fasilitas GUI.



Prerequisites

- ◆ Matematika Diskret
- ◆ Struktur Data dan Algoritma
- ◆ *Computational Logic*, Bagian Pertama



Paradigma-Paradigma Pemrograman

- ◆ Bahasa pemrograman (programming language): notasi yang dipakai untuk menentukan, mengorganisasi, dan melakukan penalaran tentang komputasi.
- ◆ Banyak bahasa pemrograman telah diciptakan. Sebagian besar tidak sempat menjadi populer. Contoh-contoh yang (masih) banyak dipakai: LISP, Prolog, C++, FORTRAN, ML, Haskell, Ada, Java, ...



- ◆ Perancang suatu bahasa pemrograman harus menyeimbangkan antara:
 - ◆ membuat komputer mudah (convenient) untuk dipakai
 - ◆ membuat komputer dapat dipakai secara efisien.

machine language
 ↓
 assembly language
 ↓
 high-level language



- ◆ Keuntungan-keuntungan dari bahasa pemrograman high-level:
 - ◆ lebih mudah dibaca oleh manusia
 - ◆ tidak bergantung pada jenis mesin
 - ◆ tersedia program-library
 - ◆ ada pengecekan yang dapat membantu deteksi error
- ◆ Dijkstra:

"Program testing can be used to show the presence of bugs, but never to show their absence!"



- ◆ Paradigma pemrograman: cara/pola berpikir tentang pemrograman.
- ◆ Ada 4 paradigma utama:
 - 1) imperative programming
 - ◆ komputasi dipandang sebagai suatu barisan aksi/tindakan (a sequence of actions).
 - ◆ menekankan "how".
 - ◆ Contoh bahasa pemrograman yang mendukung: FORTRAN, C, ...
 - 2) functional programming
 - ◆ komputasi berdasarkan fungsi/kategori, dimana fungsi mempunyai status yang sama dengan nilai-nilai lainnya. (Functions are first-class values).
 - ◆ Contoh bahasa pemrograman yang mendukung: LISP, ML, Haskell, ...



3) Logic Programming

- Program terdiri dari facts (fakta-fakta) dan rules (aturan-aturan).
- Komputasi adalah deduksi.
- Menekankan "what".
- Contoh bahasa pemrograman yang mendukung: Prolog.

4) Object-Oriented Programming

- Program adalah simulasi.
- Objek bereaksi terhadap message.
- Class mendeskripsikan himpunan objek.
- Contoh bahasa pemrograman yang mendukung: C++, Smalltalk, Java, ...



Logic programming:

Algorithm = Logic + Control

-Kowalski



- Logic programming is the use of the clausal form of first order logic as a practical computer programming language.
 - program = set of axioms
 - computation = proof of a statement from the axioms
- Prolog = Programming in Logic
- Prolog lahir sekitar tahun 1972, dipelopori oleh Colmerauer dan Roussel, di Prancis.
Tokoh-tokoh penting lainnya: Kowalski, Apt, dan van Emden.



- Prolog adalah suatu bahasa declarative, bukan bahasa procedural. Prolog tidak menggunakan assignment sebagai operasi dasar seperti dalam bahasa C atau Java.
- Prolog cocok untuk komputasi simbolik (non-numerik).
- Prolog cocok untuk pemecahan masalah yang menyangkut objek dan relasi antar objek.
- Contoh:
Fakta bahwa Tom adalah parent dari Bob dapat ditulis dalam Prolog:
`parent(tom,bob).`



```

/* Figure 1.8 The family program.*/
parent( pam, bob).      % Pam is a parent of Bob
parent( tom, bob).
parent( tom, liz).
parent( bob, ann).
parent( bob, pat).
parent( pat, jim).

female( pam).          % Pam is female
female( liz).
female( ann).
female( pat).

male( tom).           % Tom is male
male( bob).
male( jim).

```



```

offspring( Y, X) :-    % Y is an offspring of X if
parent( X, Y).        % X is a parent of Y

mother( X, Y) :-      % X is the mother of Y if
parent( X, Y),        % X is a parent of Y and
female( X).           % X is female

grandparent( X, Z) :- % X is a grandparent of Z if
parent( X, Y),        % X is a parent of Y and
parent( Y, Z).        % Y is a parent of Z

```



```

sister( X, Y) :-      % X is a sister of Y if
parent( Z, X),
parent( Z, Y),        % X and Y have the same parent and
female( X),           % X is female and
X \= Y.               % X and Y are different

predecessor( X, Z) :- % Rule pr1: X is a predecessor of Z
parent( X, Z).

predecessor( X, Z) :- % Rule pr2: X is a predecessor of Z
parent( X, Y),
predecessor( Y, Z).

```



Relations

- ◆ Prolog programming consists of defining **relations** and querying about relations.
- ◆ Querying about relations, by means of **questions**, resembles querying a database. Prolog's answer to a question consists of a set of objects that satisfy the question.
- ◆ The arguments of relations can be **atoms** (such as tom), or **variables** (such as X). Variables start with a upper-case letter.
- ◆ In the course of computation, a variable can be substituted by another object. We say that a variable becomes **instantiated**.



Clauses

- ◆ A Prolog program consists of **clauses**. Each clause terminates with a **full stop**. There are three types of clauses: **facts**, **rules**, and **questions**.
- ◆ Facts declare things that are always true.
- ◆ Rules declare things that are true depending on a given condition.
- ◆ By means of questions the user can ask the program what things are true.
- ◆ A clause consists of a **head** and a **body**. The body is a list of goals separated by commas. **Commas** are understood as **conjunctions**.



- ◆ Facts are clauses that have a head and the **empty** body.
- ◆ Questions only have the body.
- ◆ Rules have the head and the **non-empty** body.
- ◆ Rules can be recursive.
- ◆ A procedure is a set of clauses about the same relation.
- ◆ A relation can be specified by facts, simply stating the n-tuples of objects that satisfy the relation, or by stating rules about the relation.



Conjunction of goals

- ◆ Questions to the system consist of one or more **goals**. A sequence of goals (separated by commas) means the conjunction of the goals.
- ◆ If several answers satisfy the question then Prolog will find as many of them as desired by the user (by typing a semicolon).
- ◆ To establish whether an object satisfies a query is often a complicated process that involves **logical inference**, exploring among alternatives and possibly **backtracking**. All this is done automatically by the Prolog system and is hidden from the user.



How does Prolog answer questions?

- ◆ Example:

```
?- predecessor( tom, pat ).
```

```
yes
```

- ◆ A question is a sequence of one or more goals.
- ◆ To answer a question, Prolog tries to satisfy all the goals.
- ◆ To satisfy a goal means to demonstrate that the goal is true, i.e., the goal logically follows from the facts and rules in the program.



- ◆ If the goals contain variables, Prolog also has to find the particular objects (in place of the variables) satisfying the goals.
- ◆ The particular instantiation of variables to these objects is displayed to the user.
- ◆ If the goals can not be satisfied, the answer is "no".
- ◆ To satisfy a goal, Prolog finds a proof sequence. Prolog starts from the goal and, using rules, substitutes the current goal with new goals, until new goals happen to be simple facts.



Tree of execution trace

