# Pemrograman Logika dengan Prolog

## Kuliah Logika Matematika Semester Ganjil 2022-2023

MZI

Fakultas Informatika
Telkom University

FIF Tel-U

November 2022

# *Acknowledgements* I

*Slide* ini disusun berdasarkan materi yang terdapat pada sumber-sumber berikut:

1. *Logic Programming with Prolog*, Edisi 2, 2013, oleh Max Bramer.
2. *Discrete Mathematics and Its Applications* (Bab 1), Edisi 8, 2019, oleh K. H. Rosen.
3. *Discrete Mathematics with Applications* (Bab 3), Edisi 5, 2018, oleh S. S. Epp.
4. *Mathematical Logic for Computer Science* (Bab 5, 6), Edisi 2, 2000, oleh M. Ben-Ari.
5. *Slide* kuliah Logika Komputasional (*Computational Logic*) di Fasilkom UI oleh L. Y. Stefanus.
6. *Slide* kuliah Pemrograman Logika di ILLC, University of Amsterdam oleh U. Endriss.
7. *Slide* kuliah Pemrograman Fungsional di Fasilkom UI oleh A. Azurat.
8. *Slide* kuliah Logika Matematika di Telkom University oleh A. Rakhmatsyah, B. Purnama.

# *Acknowledgements* II

Beberapa gambar dapat diambil dari sumber-sumber di atas. *Slide* ini ditujukan untuk keperluan akademis di lingkungan FIF Telkom University. Jika Anda memiliki saran/ pendapat/ pertanyaan terkait materi dalam *slide* ini, silakan kirim email ke <pleasedontspam>@telkomuniversity.ac.id.

# Bahasan

1. Apa itu Prolog?
2. Tatacara Instalasi Prolog
3. Pemakaian Interpreter Interaktif
4. Dasar Pemrograman Prolog: Fakta-fakta Dasar dan Query
5. Konstruksi Aturan (Rule) Sederhana pada Prolog
6. Representasi Kuantor pada Prolog (Materi Suplemen)
7. Kesamaan dan Ketaksamaan Term pada Prolog (Materi Suplemen)

# Bahasan

# Epigram

*A language that doesn't affect the way you think about programming, is not worth knowing*
(*Alan Jay Perlis, first recipient of Turing Award*)

(Diambil dari `Alan Perlis Quotes`)

# Rumpun Bahasa Pemrograman

Bahasa imperatif (*imperative language*): bahasa yang menekankan **bagaimana** cara melakukan komputasi sebagai suatu aksi/ tindakan

- Bahasa struktural (*structural language*): C, Pascal, Ada, Fortran, Python, Matlab/ Octave
- Bahasa berorientasi objek (*object oriented*): C++, Java

# Rumpun Bahasa Pemrograman

Bahasa imperatif (*imperative language*): bahasa yang menekankan **bagaimana** cara melakukan komputasi sebagai suatu aksi/ tindakan

- Bahasa struktural (*structural language*): C, Pascal, Ada, Fortran, Python, Matlab/ Octave
- Bahasa berorientasi objek (*object oriented*): C++, Java

Bahasa deskriptif (*descriptive language*): bahasa yang menekankan **apa** yang dilakukan dalam komputasi

- Bahasa pemrograman logika (*logic programming*): Prolog, Flora, Logtalk
- Bahasa pemrograman fungsional (*functional programming*): Haskell, ML

# Bahasa Imperatif vs Bahasa Deskriptif

Pada bahasa imperatif:

- perintah dirangkai dengan penyambung "`;`" atau indensasi (*indentation*)
- perintah dikendalikan dengan kondisi seleksi (dengan `if-then-else` atau `case-of` *statement*) atau repetisi (dengan `for` *loop*, `while` *loop*, atau perintah `repeat-until`)
- program merupakan rangkaian perintah untuk mengubah nilai dari beberapa variabel (*state*)

# Bahasa Imperatif vs Bahasa Deskriptif

Pada bahasa imperatif:

- perintah dirangkai dengan penyambung "`;`" atau indensasi (*indentation*)
- perintah dikendalikan dengan kondisi seleksi (dengan `if-then-else` atau `case-of` *statement*) atau repetisi (dengan `for` *loop*, `while` *loop*, atau perintah `repeat-until`)
- program merupakan rangkaian perintah untuk mengubah nilai dari beberapa variabel (*state*)

Pada bahasa deskriptif – pemrograman logika:

- program adalah sekumpulan ekspresi
- program terdiri dari fakta (*facts*) dan aturan-aturan tertentu (*rules*)
- komputasi adalah suatu deduksi atau inferensi
- program tidak memakai *assignment* sebagai operator dasar seperti dalam C atau Pascal

# Mengapa Perlu Belajar Pemrograman Logika?

Pemrograman logika dengan Prolog perlu dipelajari karena:

- Prolog mudah diterapkan untuk kasus komputasi simbolik (komputasi non-numerik)
- Prolog mudah dan cocok diterapkan untuk pemecahan masalah yang menyangkut kondisi suatu objek atau relasi antara beberapa objek
- Prolog mudah dan cocok diterapkan untuk menggambarkan proses penalaran (*reasoning*) yang diperlukan dalam kecerdasan buatan (*artificial intelligence*)

Karena alasan-alasan ini, Prolog dapat diterapkan dalam bidang kecerdasan buatan, pemrosesan bahasa alami (*natural language processing*), dan basis data (*database*).

# Mengapa Perlu Belajar Pemrograman Logika?

Pemrograman logika dengan Prolog perlu dipelajari karena:

- Prolog mudah diterapkan untuk kasus komputasi simbolik (komputasi non-numerik)
- Prolog mudah dan cocok diterapkan untuk pemecahan masalah yang menyangkut kondisi suatu objek atau relasi antara beberapa objek
- Prolog mudah dan cocok diterapkan untuk menggambarkan proses penalaran (*reasoning*) yang diperlukan dalam kecerdasan buatan (*artificial intelligence*)

Karena alasan-alasan ini, Prolog dapat diterapkan dalam bidang kecerdasan buatan, pemrosesan bahasa alami (*natural language processing*), dan basis data (*database*).

Prolog kurang cocok untuk:

- komputasi yang memerlukan presisi numerik yang tinggi (contohnya komputasi grafis)
- komputasi yang memerlukan portabilitas yang tinggi (contohnya komputasi pengiriman pesan *real-time*)

# Sejarah Prolog

- Prolog merupakan akronim dari **Pro***gramming in* **Log***ic*.
- Prolog dikembangkan antara tahun 1972-1973 oleh Alain Colmerauer dan Phillipe Roussel di Aix-Marseille Université, Prancis.
- Tujuan pengembangan Prolog pada saat itu adalah untuk pemrosesan bahasa alami (bahasa manusia).
- Saat ini Prolog merupakan bahasa pemrograman logika yang paling banyak diajarkan di dunia.
- Prolog juga sudah digunakan di dunia industri, salah satunya oleh IBM dan Apache.

# *Meet Watson: The Supercomputer*

Watson adalah sebuah (super) komputer yang memiliki kemampuan untuk menjawab pertanyaan dalam bahasa alami (bahasa manusia) yang dikembangkan oleh IBM DeepQA *project*. Kecerdasan buatan yang diterapkan pada Watson dibuat dengan bahasa pemrograman Prolog.


Watson dalam kuis *Jeopardy!*. Sumber: WIKIPEDIA: Watson (Computer).

# Bahasan

# Perangkat Lunak Pendukung

*Source file* yang diperlukan untuk instalasi dapat diunduh dari beberapa tautan berikut:

- Amzi! Prolog (`http://www.amzi.com/products/prolog_products.htm`).
- Logic Programming Associates Prolog (`http://www.lpa.co.uk`).
- SWI-Prolog (`http://www.swi-prolog.org/`).
- Visual Prolog (`http://www.visual-prolog.com/`).
- W-Prolog (`http://waitaki.otago.ac.nz/~michael/wp/`).

Dalam *slide* kuliah ini, program Prolog ditulis dalam *syntax* SWI-Prolog. Semua program pada slide kuliah ini dijalankan pada SWI-Prolog 64-bit versi 7.2.2 yang telah diuji coba pada sistem operasi Windows 7 64-bit dan Linux Ubuntu 14.04 64-bit. SWI-Prolog adalah salah satu versi Prolog yang paling sederhana dan paling mudah untuk digunakan dalam mempelajari pemrograman logika dengan Prolog. SWI-Prolog dapat dijalankan pada sistem operasi Windows, Linux, maupun Macintosh.

# SWI-Prolog dan *Prolog Programming Contest*

Pada tahun 2013, SWI-Prolog adalah satu-satunya varian Prolog yang boleh digunakan dalam *Prolog Programming Contest*. Acara ini merupakan acara tahunan yang diadakan bersamaan dengan ICLP (*International Conference on Logic Programming*).

# Instalasi SWI-Prolog – Windows dan Macintosh

Untuk sistem operasi Windows dan Macintosh, *file* instalasi dapat diunduh di `http://www.swi-prolog.org/download/stable`. Untuk sistem operasi Windows XP/Vista/7/8/10 baik 32 maupun 64-bit, instalasi dapat dilakukan dengan mudah seperti instalasi program pada umumnya. Direktori instalasi dapat disesuaikan dengan kebutuhan, contohnya di `C:\Program Files\swipl`. Instalasi telah diujicoba dan berjalan cukup lancar pada sistem operasi Windows 7 64-bit dan Windows 10 64-bit.

# Instalasi SWI-Prolog – Linux (Ubuntu)

Untuk sistem operasi Linux Ubuntu, instalasi SWI-Prolog dapat dilakukan via PPA (*Personal Package Archive*). Buka terminal, kemudian ketik perintah-perintah berikut:

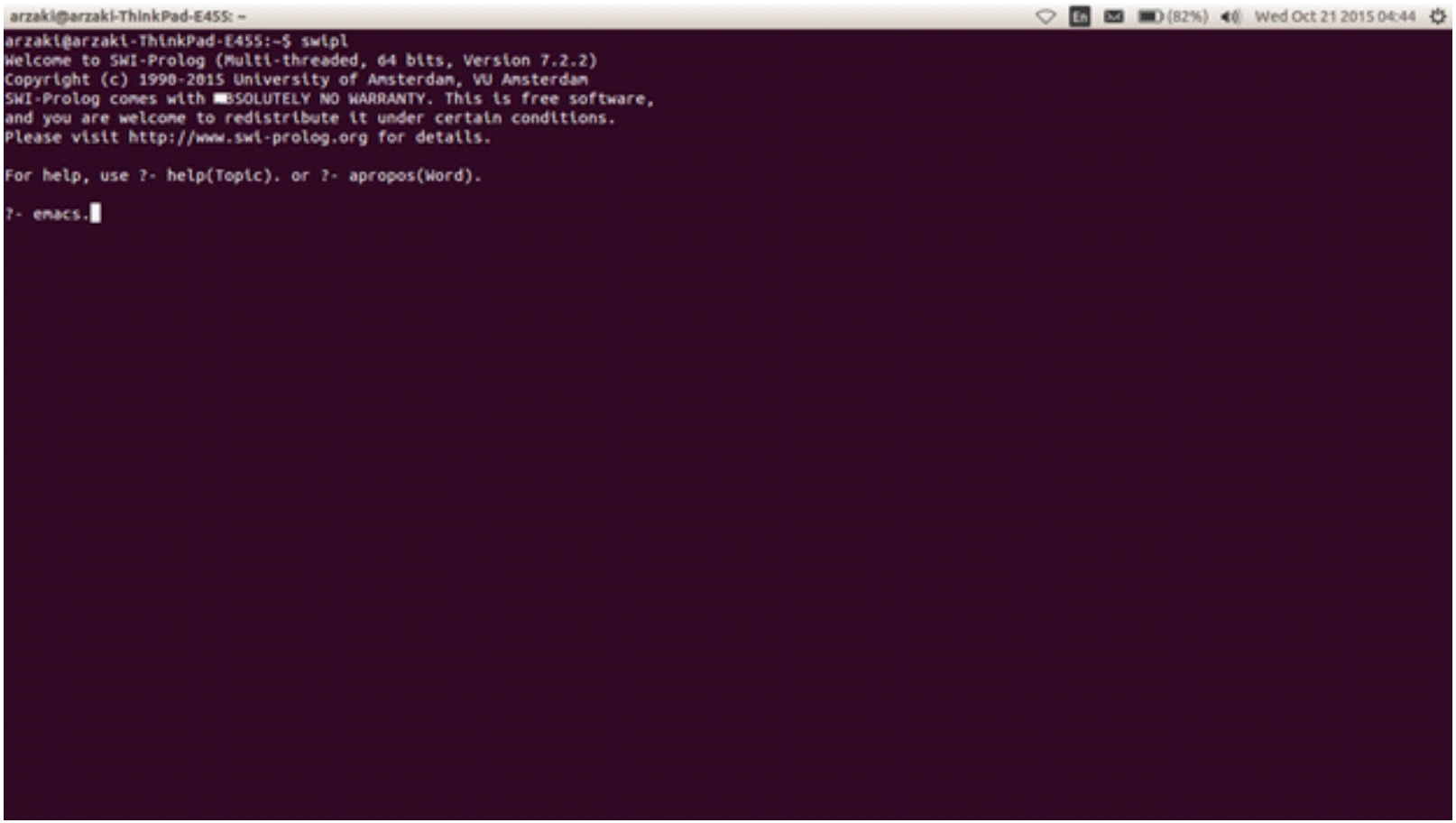
```
% sudo apt-add-repository ppa:swi-prolog/stable
% sudo apt-get update
% sudo apt-get install swi-prolog
```

Setelah instalasi, maka kita dapat menjalankan SWI-Prolog melalui terminal dengan mengetikkan perintah-perintah berikut:

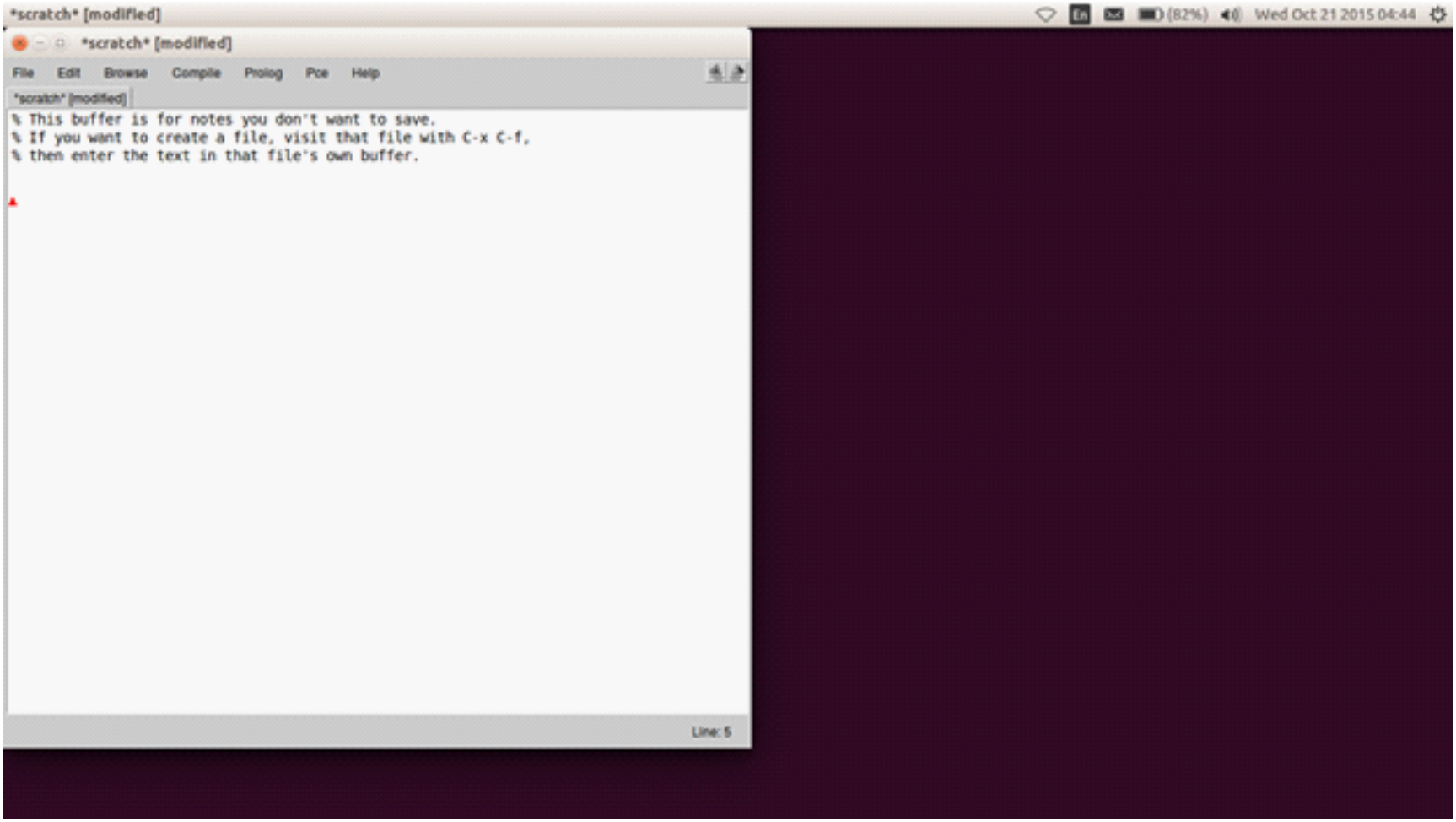
```
% swipl
?- emacs.
```

Perintah `?- emacs.` berfungsi untuk menjalankan GUI pada Linux yang serupa dengan GUI pada Windows.

Ilustrasi pemakaian GUI untuk SWI-Prolog pada Linux Ubuntu.



# Ilustrasi GUI untuk SWI-Prolog pada Linux Ubuntu.



# Instalasi SWI-Prolog – Linux (Distro Lain)

Untuk sistem operasi Linux dengan distro selain Ubuntu, instalasi dapat dilakukan dengan meninjau tautan
`http://www.swi-prolog.org/build/LinuxDistro.txt`.

# Tentang SWI-Prolog

Diambil dari `http://www.swi-prolog.org/`:

> *SWI-Prolog offers a comprehensive free Prolog environment. Since its start in 1987, SWI-Prolog development has been driven by the needs of real world applications. SWI-Prolog is widely used in research and education as well as commercial applications. Join over a million users who have downloaded SWI-Prolog.*

SWI merupakan akronim dari *Sociaal-Wetenschappelijke Informatica* ("*Social Science Informatics*"), merupakan nama lain dari grup riset *Human-Computer Studies* di *University of Amsterdam*, Belanda. SWI-Prolog ditulis oleh Jan Wielemaker.

# Bahasan

# Pemakaian Interpreter Interaktif

Ketika Anda membuka interpreter SWI-Prolog, maka Anda akan melihat *prompt*:

```
1 ?-
```

Ketikkan perintah berikut pada SWI-Prolog:
`write('Hello World'),nl,write('Welcome to Prolog'),nl.`

```
1 ?-write('Hello World'),nl,write('Welcome to Prolog'),nl.
```

SWI-Prolog akan mengeluarkan:

# Pemakaian Interpreter Interaktif

Ketika Anda membuka interpreter SWI-Prolog, maka Anda akan melihat *prompt*:

```
1 ?-
```

Ketikkan perintah berikut pada SWI-Prolog:
`write('Hello World'),nl,write('Welcome to Prolog'),nl.`

```
1 ?-write('Hello World'),nl,write('Welcome to Prolog'),nl.
```

SWI-Prolog akan mengeluarkan:

```
Hello World
Welcome to Prolog
true.
```

`write` dan `nl` adalah dua contoh *built-in-predicate* (BIP) pada SWI-Prolog, `write('x')` berfungsi untuk mengeluarkan tulisan `x` dan `nl` berfungsi untuk memindahkan keluaran berikutnya ke baris yang baru. **Perintah pada Prolog harus diakhiri oleh tanda titik (.).**

# Aritmetika pada Interpreter Interaktif

Aritmetika sederhana dapat dilakukan pada SWI-Prolog sebagai berikut.

```
1 ?- X is 1+2.
```

# Aritmetika pada Interpreter Interaktif

Aritmetika sederhana dapat dilakukan pada SWI-Prolog sebagai berikut.

```
1 ?- X is 1+2.
X = 3.
2 ?- X is 3-7.
```

# Aritmetika pada Interpreter Interaktif

Aritmetika sederhana dapat dilakukan pada SWI-Prolog sebagai berikut.

```
1 ?- X is 1+2.
X = 3.
2 ?- X is 3-7.
X = -4.
3 ?- X is 2*3.
```

# Aritmetika pada Interpreter Interaktif

Aritmetika sederhana dapat dilakukan pada SWI-Prolog sebagai berikut.

```
1 ?- X is 1+2.
X = 3.
2 ?- X is 3-7.
X = -4.
3 ?- X is 2*3.
X = 6.
4 ?- X is 7/3.
```

# Aritmetika pada Interpreter Interaktif

Aritmetika sederhana dapat dilakukan pada SWI-Prolog sebagai berikut.

```
1 ?- X is 1+2.
X = 3.
2 ?- X is 3-7.
X = -4.
3 ?- X is 2*3.
X = 6.
4 ?- X is 7/3.
X = 2.3333333333333335.
5 ?- X is 3^2.
```

[Operasi $x^y$ juga dapat ditulis `x**y`]

# Aritmetika pada Interpreter Interaktif

Aritmetika sederhana dapat dilakukan pada SWI-Prolog sebagai berikut.

```
1 ?- X is 1+2.
X = 3.
2 ?- X is 3-7.
X = -4.
3 ?- X is 2*3.
X = 6.
4 ?- X is 7/3.
X = 2.3333333333333335.
5 ?- X is 3^2.
X = 9.
6 ?- X is 3+5*2.
```

[Operasi $x^y$ juga dapat ditulis x**y]

# Aritmetika pada Interpreter Interaktif

Aritmetika sederhana dapat dilakukan pada SWI-Prolog sebagai berikut.

```
1 ?- X is 1+2.
X = 3.
2 ?- X is 3-7.
X = -4.
3 ?- X is 2*3.
X = 6.
4 ?- X is 7/3.
X = 2.3333333333333335.
5 ?- X is 3^2.
X = 9.
6 ?- X is 3+5*2.
X = 13.
7 ?- X = 3*3.
```

[Operasi $x^y$ juga dapat ditulis x**y]

# Aritmetika pada Interpreter Interaktif

Aritmetika sederhana dapat dilakukan pada SWI-Prolog sebagai berikut.

```
1 ?- X is 1+2.
X = 3.
2 ?- X is 3-7.
X = -4.
3 ?- X is 2*3.
X = 6.
4 ?- X is 7/3.
X = 2.3333333333333335.
5 ?- X is 3^2.
X = 9.
6 ?- X is 3+5*2.
X = 13.
7 ?- X = 3*3.
X = 3*3.
```

[Operasi $x^y$ juga dapat ditulis `x**y`]

Kalkulasi aritmetika pada SWI-Prolog dilakukan dengan predikat `is`. Tanda = pada Prolog digunakan untuk memeriksa kesamaan term.

# Lebih Jauh Tentang Operator Aritmetika Prolog

Pada Prolog, operator aritmetika yang digunakan adalah: $+$, $-$, $*$, $/$, $//$, dan $\mathrm{mod}$. Operator $//$ berarti pembagian bilangan bulat (operator $\mathrm{div}$).

```
1 ?- X is 9/4.
```

# Lebih Jauh Tentang Operator Aritmetika Prolog

Pada Prolog, operator aritmetika yang digunakan adalah: $+$, $-$, $*$, $/$, $//$, dan $\mathrm{mod}$. Operator $//$ berarti pembagian bilangan bulat (operator $\mathrm{div}$).

```
1 ?- X is 9/4.
X = 2.25.
2 ?- X is 9//4.
```

# Lebih Jauh Tentang Operator Aritmetika Prolog

Pada Prolog, operator aritmetika yang digunakan adalah: $+$, $-$, $*$, $/$, $//$, dan $\mathrm{mod}$. Operator $//$ berarti pembagian bilangan bulat (operator $\mathrm{div}$).

```
1 ?- X is 9/4.
X = 2.25.
2 ?- X is 9//4.
X = 2.
3 ?- X is 9 mod 4.
```

# Lebih Jauh Tentang Operator Aritmetika Prolog

Pada Prolog, operator aritmetika yang digunakan adalah: $+$, $-$, $*$, $/$, $//$, dan $\mathrm{mod}$. Operator $//$ berarti pembagian bilangan bulat (operator $\mathrm{div}$).

```
1 ?- X is 9/4.
X = 2.25.
2 ?- X is 9//4.
X = 2.
3 ?- X is 9 mod 4.
X = 1.
```

# Perbandingan Nilai Numerik pada SWI-Prolog

Perbandingan Nilai Numerik pada SWI-Prolog dapat dilakukan sebagai berikut.

```
1 ?- 1 < 2.
```

# Perbandingan Nilai Numerik pada SWI-Prolog

Perbandingan Nilai Numerik pada SWI-Prolog dapat dilakukan sebagai berikut.

```
1 ?- 1 < 2.
true.
2 ?- 3-2 > 3+2.
```

# Perbandingan Nilai Numerik pada SWI-Prolog

Perbandingan Nilai Numerik pada SWI-Prolog dapat dilakukan sebagai berikut.

```
1 ?- 1 < 2.
true.
2 ?- 3-2 > 3+2.
false.
3 ?- 3+2 =:= 6-1.
```

[tanda =:= berarti kesamaan numerik]

# Perbandingan Nilai Numerik pada SWI-Prolog

Perbandingan Nilai Numerik pada SWI-Prolog dapat dilakukan sebagai berikut.

```
1 ?- 1 < 2.
true.
2 ?- 3-2 > 3+2.
false.
3 ?- 3+2 =:= 6-1.          [tanda =:= berarti kesamaan numerik]
true.
4 ?- 3+2 =\= 3-2.          [tanda =\= berarti ketidaksamaan numerik]
```

# Perbandingan Nilai Numerik pada SWI-Prolog

Perbandingan Nilai Numerik pada SWI-Prolog dapat dilakukan sebagai berikut.

```
1 ?- 1 < 2.
true.
2 ?- 3-2 > 3+2.
false.
3 ?- 3+2 =:= 6-1.                    [tanda =:= berarti kesamaan numerik]
true.
4 ?- 3+2 =\= 3-2.                    [tanda =\= berarti ketidaksamaan numerik]
true.
5 ?- 3+2 >= 3-2.
```

# Perbandingan Nilai Numerik pada SWI-Prolog

Perbandingan Nilai Numerik pada SWI-Prolog dapat dilakukan sebagai berikut.

```
1 ?- 1 < 2.
true.
2 ?- 3-2 > 3+2.
false.
3 ?- 3+2 =:= 6-1.                    [tanda =:= berarti kesamaan numerik]
true.
4 ?- 3+2 =\= 3-2.                    [tanda =\= berarti ketidaksamaan numerik]
true.
5 ?- 3+2 >= 3-2.
true.
6 ?- 3+2 =< 3-2.
```

# Perbandingan Nilai Numerik pada SWI-Prolog

Perbandingan Nilai Numerik pada SWI-Prolog dapat dilakukan sebagai berikut.

```
1 ?- 1 < 2.
true.
2 ?- 3-2 > 3+2.
false.
3 ?- 3+2 =:= 6-1.                    [tanda =:= berarti kesamaan numerik]
true.
4 ?- 3+2 =\= 3-2.                    [tanda =\= berarti ketidaksamaan numerik]
true.
5 ?- 3+2 >= 3-2.
true.
6 ?- 3+2 =< 3-2.
false.
7 ?- 3+2 => 3-2.
```

# Perbandingan Nilai Numerik pada SWI-Prolog

Perbandingan Nilai Numerik pada SWI-Prolog dapat dilakukan sebagai berikut.

```
1 ?- 1 < 2.
true.
2 ?- 3-2 > 3+2.
false.
3 ?- 3+2 =:= 6-1.                    [tanda =:= berarti kesamaan numerik]
true.
4 ?- 3+2 =\= 3-2.                    [tanda =\= berarti ketidaksamaan numerik]
true.
5 ?- 3+2 >= 3-2.
true.
6 ?- 3+2 =< 3-2.
false.
7 ?- 3+2 => 3-2.
ERROR: Syntax error:  Operator expected
ERROR: 3+2
ERROR: ** here **
ERROR: => 3-2.
```

# Operator Perbandingan Nilai Numerik pada Prolog

Operator perbandingan nilai numerik yang dipakai pada Prolog adalah

| Simbol matematika | Simbol pada Prolog |
|---|---|
| $=$ | =:= |
| $\neq$ | =\= |
| $<$ | < |
| $>$ | > |
| $\leq$ | =< |
| $\geq$ | >= |

Simbol <= dan => **bukan operator perbandingan nilai numerik** pada Prolog.

# Bahasan

# Menulis Program pada SWI-Prolog

Cara penulisan program pada SWI-Prolog dapat dilakukan menggunakan editor teks apapun (contohnya notepad atau notepad++), tetapi SWI-Prolog juga telah dilengkapi dengan editor yang dapat memeriksa *syntax* program Prolog yang kita buat.

## Cara Membuat Skrip pada Windows

Buka SWI-Prolog, kemudian pilih `File` → `New` dan beri nama *file* yang sesuai dengan kebutuhan (pastikan jenis *file* adalah *Prolog Source* atau program berekstensi `.pl`). Jika nama *file* sudah ditentukan maka kita dapat menulis pada teks editor yang disediakan.

## Cara Membuat Skrip pada Linux Ubuntu

Buka terminal, kemudian ketik `swipl`, setelah masuk ke SWI-Prolog ketikkan `emacs.` (dengan tanda `.`). Langkah berikutnya adalah pilih `File` → `New` kemudian beri nama file yang sesuai dengan kebutuhan (pastikan program berekstensi `.pl`). Jika nama *file* sudah ditentukan maka kita dapat menulis pada teks editor yang disediakan.

# Translasi Formula Logika Predikat Sederhana ke Prolog

Misalkan $D$ adalah domain yang berupa himpunan 12 orang manusia dengan $D = \{Alice, Bob, Charlie, David, Emma, Fiona, Grace, Hans, Irene, Jim, Kelly, Lily\}$.

Misalkan $\mathrm{Male}\,(x)$ adalah predikat yang menyatakan bahwa "$x$ adalah laki-laki" dan $\mathrm{Female}\,(x)$ adalah predikat yang menyatakan bahwa "$x$ adalah perempuan".

Misalkan kita memiliki fakta-fakta berikut:

- Bob, Charlie, David, Hans, dan Jim adalah laki-laki.
- Alice, Emma, Fiona, Grace, Irene, Kelly, dan Lily adalah perempuan.

Fakta-fakta tersebut dapat ditranslasikan ke dalam formula logika predikat sebagai:

# Translasi Formula Logika Predikat Sederhana ke Prolog

Misalkan $D$ adalah domain yang berupa himpunan 12 orang manusia dengan $D = \{Alice, Bob, Charlie, David, Emma, Fiona, Grace, Hans, Irene, Jim, Kelly, Lily\}$.

Misalkan $\mathrm{Male}\,(x)$ adalah predikat yang menyatakan bahwa "$x$ adalah laki-laki" dan $\mathrm{Female}\,(x)$ adalah predikat yang menyatakan bahwa "$x$ adalah perempuan".

Misalkan kita memiliki fakta-fakta berikut:

- Bob, Charlie, David, Hans, dan Jim adalah laki-laki.
- Alice, Emma, Fiona, Grace, Irene, Kelly, dan Lily adalah perempuan.

Fakta-fakta tersebut dapat ditranslasikan ke dalam formula logika predikat sebagai:

- $\mathrm{Male}\,(Bob)$, $\mathrm{Male}\,(Charlie)$, $\mathrm{Male}\,(David)$, $\mathrm{Male}\,(Hans)$, $\mathrm{Male}\,(Jim)$

# Translasi Formula Logika Predikat Sederhana ke Prolog

Misalkan $D$ adalah domain yang berupa himpunan 12 orang manusia dengan $D = \{Alice, Bob, Charlie, David, Emma, Fiona, Grace, Hans, Irene, Jim, Kelly, Lily\}$.

Misalkan $\text{Male}(x)$ adalah predikat yang menyatakan bahwa "$x$ adalah laki-laki" dan $\text{Female}(x)$ adalah predikat yang menyatakan bahwa "$x$ adalah perempuan".

Misalkan kita memiliki fakta-fakta berikut:

- Bob, Charlie, David, Hans, dan Jim adalah laki-laki.
- Alice, Emma, Fiona, Grace, Irene, Kelly, dan Lily adalah perempuan.

Fakta-fakta tersebut dapat ditranslasikan ke dalam formula logika predikat sebagai:

- $\text{Male}(Bob)$, $\text{Male}(Charlie)$, $\text{Male}(David)$, $\text{Male}(Hans)$, $\text{Male}(Jim)$
- $\text{Female}(Alice)$, $\text{Female}(Emma)$, $\text{Female}(Fiona)$, $\text{Female}(Grace)$, $\text{Female}(Irene)$, $\text{Female}(Kelly)$, $\text{Female}(Lily)$

# Skrip Prolog

Fakta-fakta tersebut dapat dituliskan dalam skrip Prolog sebagai berikut.

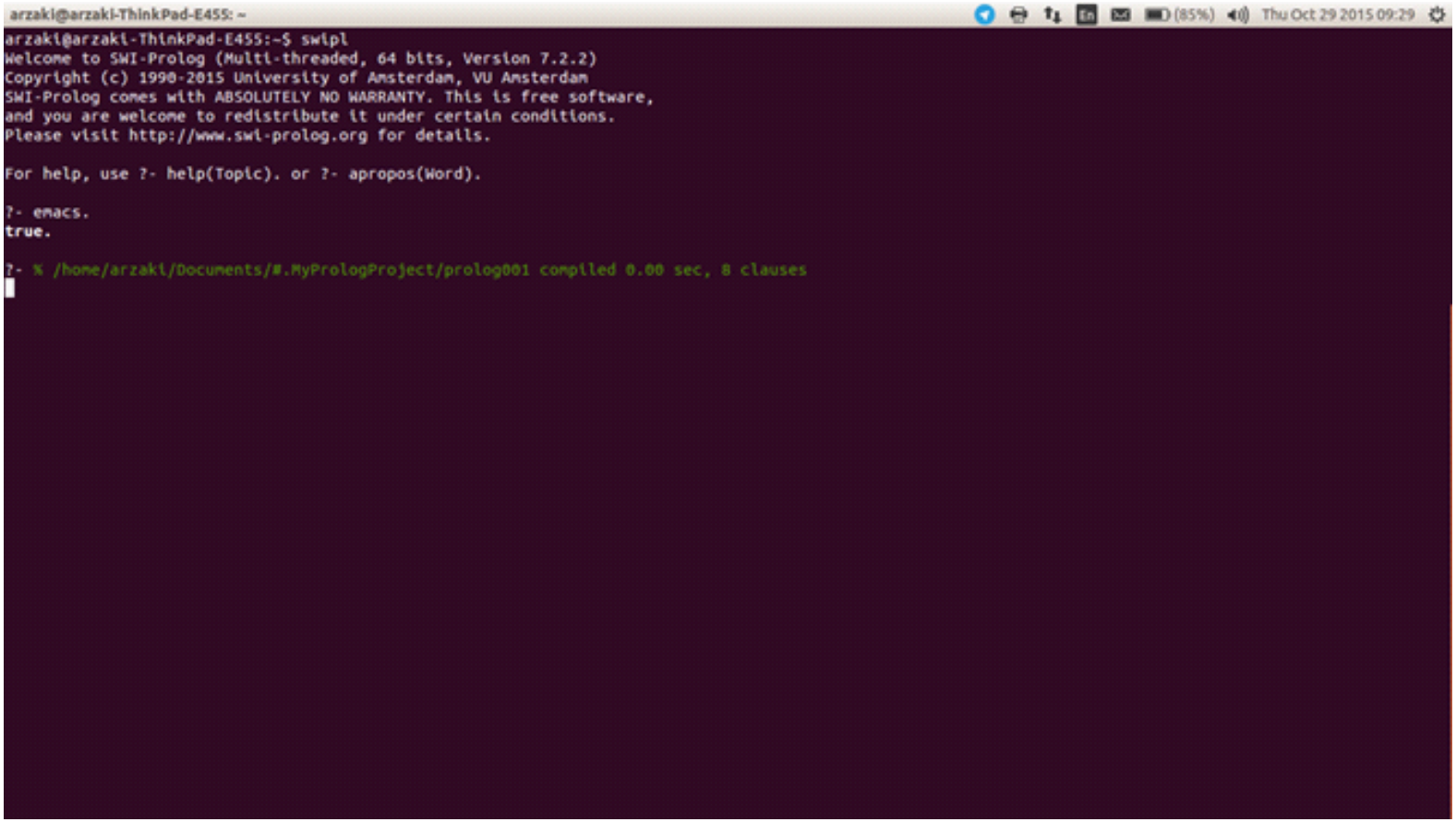
```
% male(x) menyatakan bahwa x adalah laki-laki.
/* Bob, Charlie, David, Hans, & Jim adalah laki-laki */
male(bob).
male(charlie).
male(david).
male(hans).
male(jim).
% female(x) menyatakan bahwa x adalah perempuan.
/* Alice, Emma, Fiona, Grace, Irene, Kelly, & Lily adalah
perempuan */
female(alice).
female(emma).
female(fiona).
female(grace).
female(irene).
female(kelly).
female(lily).
```

- Predikat diawali dengan huruf kecil, begitu pula dengan term konstanta. **Term konstanta tidak boleh diawali dengan huruf kapital**.
- Term konstanta boleh diapit dengan tanda kutip tunggal, contohnya `male('Bob')`.
- Komentar pada skrip Prolog diawali dengan tanda % atau diapit dengan tanda /* dan */.

## Cara Menjalankan Program: Windows dan Linux

Pada editor teks, pilih `Compile` → `Make` dan setelah itu `Compile` → `Compile Buffer`. Interpreter (atau terminal pada Linux) akan mengelurkan peringatan bahwa proses kompilasi telah dilakukan dengan sukses.

Contoh hasil proses kompilasi pada Linux Ubuntu 14.04.



# Input-Output Query pada Prolog

Setelah program berhasil dikompilasi, kita dapat memasukkan beberapa query pada interpreter interaktif (atau terminal pada Linux). Perpindahan ke baris berikutnya dapat dilakukan dengan enter atau tabulasi.

```
?- male(bob).
```
[apakah Bob adalah laki-laki?]

# Input-Output Query pada Prolog

Setelah program berhasil dikompilasi, kita dapat memasukkan beberapa query pada interpreter interaktif (atau terminal pada Linux). Perpindahan ke baris berikutnya dapat dilakukan dengan enter atau tabulasi.

```
?- male(bob).                    [apakah Bob adalah laki-laki?]
true.
?- not(male(charlie)).           [apakah Charlie bukan laki-laki?]
```

# Input-Output Query pada Prolog

Setelah program berhasil dikompilasi, kita dapat memasukkan beberapa query pada interpreter interaktif (atau terminal pada Linux). Perpindahan ke baris berikutnya dapat dilakukan dengan enter atau tabulasi.

```
?- male(bob).                          [apakah Bob adalah laki-laki?]
true.
?- not(male(charlie)).                 [apakah Charlie bukan laki-laki?]
false.
?- male(bob),female(alice).            [, adalah operator ∧]
```

# Input-Output Query pada Prolog

Setelah program berhasil dikompilasi, kita dapat memasukkan beberapa query pada interpreter interaktif (atau terminal pada Linux). Perpindahan ke baris berikutnya dapat dilakukan dengan enter atau tabulasi.

```
?- male(bob).                                  [apakah Bob adalah laki-laki?]
true.
?- not(male(charlie)).                         [apakah Charlie bukan laki-laki?]
false.
?- male(bob),female(alice).                    [, adalah operator ∧]
true.
?- male(david),male(emma).
```

# Input-Output Query pada Prolog

Setelah program berhasil dikompilasi, kita dapat memasukkan beberapa query pada interpreter interaktif (atau terminal pada Linux). Perpindahan ke baris berikutnya dapat dilakukan dengan enter atau tabulasi.

```
?- male(bob).                          [apakah Bob adalah laki-laki?]
true.
?- not(male(charlie)).                 [apakah Charlie bukan laki-laki?]
false.
?- male(bob),female(alice).                    [, adalah operator ∧]
true.
?- male(david),male(emma).
false.
?- male(david);male(fiona).                    [; adalah operator ∨]
```

# Input-Output Query pada Prolog

Setelah program berhasil dikompilasi, kita dapat memasukkan beberapa query pada interpreter interaktif (atau terminal pada Linux). Perpindahan ke baris berikutnya dapat dilakukan dengan enter atau tabulasi.

```
?- male(bob).                                  [apakah Bob adalah laki-laki?]
true.
?- not(male(charlie)).                         [apakah Charlie bukan laki-laki?]
false.
?- male(bob),female(alice).                    [, adalah operator ∧]
true.
?- male(david),male(emma).
false.
?- male(david);male(fiona).                    [; adalah operator ∨]
true.
?- female(charlie);male(grace).
```

# Input-Output Query pada Prolog

Setelah program berhasil dikompilasi, kita dapat memasukkan beberapa query pada interpreter interaktif (atau terminal pada Linux). Perpindahan ke baris berikutnya dapat dilakukan dengan enter atau tabulasi.

```
?- male(bob).                                  [apakah Bob adalah laki-laki?]
true.
?- not(male(charlie)).                         [apakah Charlie bukan laki-laki?]
false.
?- male(bob),female(alice).                    [, adalah operator ∧]
true.
?- male(david),male(emma).
false.
?- male(david);male(fiona).                    [; adalah operator ∨]
true.
?- female(charlie);male(grace).
false.
```

# Asumsi Dunia Tertutup (*Closed World Assumption*)

Pada program sebelumnya, kita dapat melakukan query berikut.

```
?- male(anakin).
```

# Asumsi Dunia Tertutup (*Closed World Assumption*)

Pada program sebelumnya, kita dapat melakukan query berikut.

```
?- male(anakin).
false.
?- female(anakin).
```

# Asumsi Dunia Tertutup (*Closed World Assumption*)

Pada program sebelumnya, kita dapat melakukan query berikut.

```
?- male(anakin).
false.
?- female(anakin).
false.
?- not(male(anakin)).
```

# Asumsi Dunia Tertutup (*Closed World Assumption*)

Pada program sebelumnya, kita dapat melakukan query berikut.

```
?- male(anakin).
false.
?- female(anakin).
false.
?- not(male(anakin)).
true.
?- not(female(anakin)).
```

# Asumsi Dunia Tertutup (*Closed World Assumption*)

Pada program sebelumnya, kita dapat melakukan query berikut.

```
?- male(anakin).
false.
?- female(anakin).
false.
?- not(male(anakin)).
true.
?- not(female(anakin)).
true.
```

## Catatan

Ketika Prolog memproses query `not(...)`, Prolog melakukannya dengan memeriksa apakah fakta `(...)` benar. Jika fakta `(...)` tidak benar, maka Prolog akan mengembalikan nilai `false`. Perlu diperhatikan bahwa penalaran (*reasoning*) yang dilakukan Prolog didasarkan pada asumsi dunia tertutup (*closed world assumption*). Berdasarkan asumsi ini, semua hal yang benar adalah hal yang tertulis pada program atau dapat diturunkan dari fakta-fakta yang ada pada program.

# Variabel dan Query Variabel pada Prolog

Variabel pada Prolog **selalu diawali dengan huruf kapital**. Biasanya variabel ditulis dengan satu huruf kapital saja (contohnya `X`, `Y`, atau `Z`). Variabel dapat digunakan pada interpreter untuk menampilkan semua objek yang memberikan nilai kebenaran untuk predikat tertentu.

```
?- male(X).
```

# Variabel dan Query Variabel pada Prolog

Variabel pada Prolog **selalu diawali dengan huruf kapital**. Biasanya variabel ditulis dengan satu huruf kapital saja (contohnya `X`, `Y`, atau `Z`). Variabel dapat digunakan pada interpreter untuk menampilkan semua objek yang memberikan nilai kebenaran untuk predikat tertentu.

```
?- male(X).
X = bob;                    [tekan tabulasi untuk melihat hasil berikutnya]
```

# Variabel dan Query Variabel pada Prolog

Variabel pada Prolog **selalu diawali dengan huruf kapital**. Biasanya variabel ditulis dengan satu huruf kapital saja (contohnya `X`, `Y`, atau `Z`). Variabel dapat digunakan pada interpreter untuk menampilkan semua objek yang memberikan nilai kebenaran untuk predikat tertentu.

```
?- male(X).
X = bob;                    [tekan tabulasi untuk melihat hasil berikutnya]
X = charlie;
```

# Variabel dan Query Variabel pada Prolog

Variabel pada Prolog **selalu diawali dengan huruf kapital**. Biasanya variabel ditulis dengan satu huruf kapital saja (contohnya `X`, `Y`, atau `Z`). Variabel dapat digunakan pada interpreter untuk menampilkan semua objek yang memberikan nilai kebenaran untuk predikat tertentu.

```
?- male(X).
X = bob;                    [tekan tabulasi untuk melihat hasil berikutnya]
X = charlie;
X = david;
```

# Variabel dan Query Variabel pada Prolog

Variabel pada Prolog **selalu diawali dengan huruf kapital**. Biasanya variabel ditulis dengan satu huruf kapital saja (contohnya `X`, `Y`, atau `Z`). Variabel dapat digunakan pada interpreter untuk menampilkan semua objek yang memberikan nilai kebenaran untuk predikat tertentu.

```
?- male(X).
X = bob;                    [tekan tabulasi untuk melihat hasil berikutnya]
X = charlie;
X = david;
X = hans;
```

# Variabel dan Query Variabel pada Prolog

Variabel pada Prolog **selalu diawali dengan huruf kapital**. Biasanya variabel ditulis dengan satu huruf kapital saja (contohnya `X`, `Y`, atau `Z`). Variabel dapat digunakan pada interpreter untuk menampilkan semua objek yang memberikan nilai kebenaran untuk predikat tertentu.

```
?- male(X).
X = bob;                    [tekan tabulasi untuk melihat hasil berikutnya]
X = charlie;
X = david;
X = hans;
X = jim.
```

```
?- female(Person).
```

```
?- female(Person).
Person = alice;
```

```
?- female(Person).
Person = alice;
Person = emma;
```

```
?- female(Person).
Person = alice;
Person = emma;
Person = fiona;
```

```
?- female(Person).
Person = alice;
Person = emma;
Person = fiona;
Person = grace;
```

```
?- female(Person).
Person = alice;
Person = emma;
Person = fiona;
Person = grace;
Person = irene;
```

```
?- female(Person).
Person = alice;
Person = emma;
Person = fiona;
Person = grace;
Person = irene;
Person = kelly;
```

```
?- female(Person).
Person = alice;
Person = emma;
Person = fiona;
Person = grace;
Person = irene;
Person = kelly;
Person = lily.
```

# Predikat Biner pada Prolog

Misalkan $D$ adalah domain yang telah dijelaskan sebelumnya dan $\text{Parent}\,(x, y)$ adalah predikat biner yang menyatakan bahwa "$x$ adalah orang tua dari $y$". Misalkan kita memiliki fakta-fakta berikut dalama formula logika predikat.

- $\text{Parent}\,(Alice, Charlie)$, $\text{Parent}\,(Bob, Charlie)$, $\text{Parent}\,(Bob, Emma)$.
- $\text{Parent}\,(Charlie, Fiona)$, $\text{Parent}\,(Charlie, Grace)$, $\text{Parent}\,(Emma, Irene)$.
- $\text{Parent}\,(Fiona, David)$, $\text{Parent}\,(Fiona, Lily)$, $\text{Parent}\,(Grace, Jim)$, $\text{Parent}\,(Grace, Kelly)$, $\text{Parent}\,(Hans, Jim)$, $\text{Parent}\,(Hans, Kelly)$.

Fakta-fakta tersebut dapat ditranslasikan ke dalam skrip Prolog menjadi:

```
% parent(x,y) menyatakan bahwa x adalah orangtua dari y
parent(alice,charlie).
parent(bob,charlie).
parent(bob,emma).
parent(charlie,fiona).
parent(charlie,grace).
parent(emma,irene).
parent(fiona,david).
parent(fiona,lily).
parent(grace,jim).
parent(grace,kelly).
parent(hans,jim).
parent(hans,kelly).
```

Fakta-fakta tersebut dapat ditambahkan ke dalam skrip Prolog yang telah dibuat sebelumnya.

## Catatan

Proses mengedit *file* Prolog yang telah dibuat sebelumnya dapat dilakukan dengan cara:
Pada Windows:

- Pada interpreter Prolog, pilih `File` → `Edit` dan kemudian pilih berkas yang akan diedit, atau
- Pada editor teks, pilih `File` → `Open` dan kemudian pilih berkas yang akan diedit.

Pada Linux Ubuntu: pada emacs pilih `File` → `Open` dan kemudian pilih berkas yang akan diedit.

Setelah program dikompilasi, kita dapat menjalankan query berikut.

```
?- parent(OrangTua,Anak).
```

Setelah program dikompilasi, kita dapat menjalankan query berikut.

```
?- parent(OrangTua,Anak).
OrangTua = alice,
Anak = charlie;
```

Setelah program dikompilasi, kita dapat menjalankan query berikut.

```
?- parent(OrangTua,Anak).
OrangTua = alice,
Anak = charlie;
OrangTua = bob,
Anak = charlie;

.
. [ada cukup banyak keluaran]
```

Setelah program dikompilasi, kita dapat menjalankan query berikut.

```
?- parent(OrangTua,Anak).
OrangTua = alice,
Anak = charlie;
OrangTua = bob,
Anak = charlie;

.
. [ada cukup banyak keluaran]

OrangTua = hans,
Anak = jim;
```

Setelah program dikompilasi, kita dapat menjalankan query berikut.

```
?- parent(OrangTua,Anak).
OrangTua = alice,
Anak = charlie;
OrangTua = bob,
Anak = charlie;

.
. [ada cukup banyak keluaran]

OrangTua = hans,
Anak = jim;
OrangTua = hans,
Anak = kelly.
```

Untuk mengetahui semua anak dari Fiona, kita dapat melakukan query `parent(fiona,X)`. Kemudian untuk mengetahui semua orang tua dari Jim, kita dapat melakukan query `parent(X,jim)`.

```
?- parent(fiona,X).
```

Untuk mengetahui semua anak dari Fiona, kita dapat melakukan query `parent(fiona,X)`. Kemudian untuk mengetahui semua orang tua dari Jim, kita dapat melakukan query `parent(X,jim)`.

```
?- parent(fiona,X).
X = david;
X = lily.

?- parent(X,jim).
```

Untuk mengetahui semua anak dari Fiona, kita dapat melakukan query `parent(fiona,X)`. Kemudian untuk mengetahui semua orang tua dari Jim, kita dapat melakukan query `parent(X,jim)`.

```
?- parent(fiona,X).
X = david;
X = lily.

?- parent(X,jim).
X = grace;
X = hans.
```

# Bahasan

Misalkan skrip yang telah dibuat sebelumnya ditambahkan dengan fakta-fakta berikut:

```
% adult(x) menyatakan bahwa x adalah orang dewasa.
/* Alice, Bob, Charlie, Emma, Fiona, Grace, & Hans adalah orang
dewasa.  */
adult(alice).
adult(bob).
adult(charlie).
adult(emma).
adult(fiona).
adult(grace).
adult(hans).
% teen(x) menyatakan bahwa x adalah remaja.
% Irene, David, & Lily adalah remaja.
teen(irene).
teen(david).
teen(lily).
```

```
% kid(x) menyatakan bahwa x adalah anak kecil.
% Jim & Kelly adalah anak kecil.
kid(jim).
kid(kelly).
```

# Translasi Formula Logika Predikat ke Prolog

Misalkan kita memiliki predikat-predikat turunan berikut:

1. $\mathrm{Gentleman}\,(x)$ : "$x$ adalah laki-laki dewasa". Predikat $\mathrm{Gentleman}\,(x)$ didefinisikan sebagai $\mathrm{Gentleman}\,(x) :=$

# Translasi Formula Logika Predikat ke Prolog

Misalkan kita memiliki predikat-predikat turunan berikut:

1. $\mathrm{Gentleman}\,(x)$ : "$x$ adalah laki-laki dewasa". Predikat $\mathrm{Gentleman}\,(x)$ didefinisikan sebagai $\mathrm{Gentleman}\,(x) := \mathrm{Male}\,(x) \wedge \mathrm{Adult}\,(x)$.
2. $\mathrm{Lady}\,(x)$ : "$x$ adalah perempuan dewasa". Predikat $\mathrm{Lady}\,(x)$ didefinisikan sebagai $\mathrm{Lady}\,(x) :=$

# Translasi Formula Logika Predikat ke Prolog

Misalkan kita memiliki predikat-predikat turunan berikut:

1. $\mathrm{Gentleman}\,(x)$ : "$x$ adalah laki-laki dewasa". Predikat $\mathrm{Gentleman}\,(x)$ didefinisikan sebagai $\mathrm{Gentleman}\,(x) := \mathrm{Male}\,(x) \wedge \mathrm{Adult}\,(x)$.
2. $\mathrm{Lady}\,(x)$ : "$x$ adalah perempuan dewasa". Predikat $\mathrm{Lady}\,(x)$ didefinisikan sebagai $\mathrm{Lady}\,(x) := \mathrm{Female}\,(x) \wedge \mathrm{Adult}\,(x)$.
3. $\mathrm{TeenBoy}\,(x)$ : "$x$ adalah laki-laki remaja". Predikat $\mathrm{TeenBoy}\,(x)$ didefinisikan sebagai $\mathrm{TeenBoy}\,(x) :=$

# Translasi Formula Logika Predikat ke Prolog

Misalkan kita memiliki predikat-predikat turunan berikut:

1. $\text{Gentleman}(x)$ : "$x$ adalah laki-laki dewasa". Predikat $\text{Gentleman}(x)$ didefinisikan sebagai $\text{Gentleman}(x) := \text{Male}(x) \wedge \text{Adult}(x)$.
2. $\text{Lady}(x)$ : "$x$ adalah perempuan dewasa". Predikat $\text{Lady}(x)$ didefinisikan sebagai $\text{Lady}(x) := \text{Female}(x) \wedge \text{Adult}(x)$.
3. $\text{TeenBoy}(x)$ : "$x$ adalah laki-laki remaja". Predikat $\text{TeenBoy}(x)$ didefinisikan sebagai $\text{TeenBoy}(x) := \text{Male}(x) \wedge \text{Teen}(x)$.
4. $\text{TeenGirl}(x)$ : "$x$ adalah perempuan remaja". Predikat $\text{TeenGirl}(x)$ didefinisikan sebagai $\text{TeenGirl}(x) :=$

# Translasi Formula Logika Predikat ke Prolog

Misalkan kita memiliki predikat-predikat turunan berikut:

1. $\mathrm{Gentleman}\,(x)$ : "$x$ adalah laki-laki dewasa". Predikat $\mathrm{Gentleman}\,(x)$ didefinisikan sebagai $\mathrm{Gentleman}\,(x) := \mathrm{Male}\,(x) \wedge \mathrm{Adult}\,(x)$.
2. $\mathrm{Lady}\,(x)$ : "$x$ adalah perempuan dewasa". Predikat $\mathrm{Lady}\,(x)$ didefinisikan sebagai $\mathrm{Lady}\,(x) := \mathrm{Female}\,(x) \wedge \mathrm{Adult}\,(x)$.
3. $\mathrm{TeenBoy}\,(x)$ : "$x$ adalah laki-laki remaja". Predikat $\mathrm{TeenBoy}\,(x)$ didefinisikan sebagai $\mathrm{TeenBoy}\,(x) := \mathrm{Male}\,(x) \wedge \mathrm{Teen}\,(x)$.
4. $\mathrm{TeenGirl}\,(x)$ : "$x$ adalah perempuan remaja". Predikat $\mathrm{TeenGirl}\,(x)$ didefinisikan sebagai $\mathrm{TeenGirl}\,(x) := \mathrm{Female}\,(x) \wedge \mathrm{Teen}\,(x)$.
5. $\mathrm{LittleBoy}\,(x)$ : "$x$ adalah anak kecil laki-laki". Predikat $\mathrm{LittleBoy}\,(x)$ didefinisikan sebagai $\mathrm{LittleBoy}\,(x) :=$

# Translasi Formula Logika Predikat ke Prolog

Misalkan kita memiliki predikat-predikat turunan berikut:

1. $\text{Gentleman}(x)$ : "$x$ adalah laki-laki dewasa". Predikat $\text{Gentleman}(x)$ didefinisikan sebagai $\text{Gentleman}(x) := \text{Male}(x) \wedge \text{Adult}(x)$.
2. $\text{Lady}(x)$ : "$x$ adalah perempuan dewasa". Predikat $\text{Lady}(x)$ didefinisikan sebagai $\text{Lady}(x) := \text{Female}(x) \wedge \text{Adult}(x)$.
3. $\text{TeenBoy}(x)$ : "$x$ adalah laki-laki remaja". Predikat $\text{TeenBoy}(x)$ didefinisikan sebagai $\text{TeenBoy}(x) := \text{Male}(x) \wedge \text{Teen}(x)$.
4. $\text{TeenGirl}(x)$ : "$x$ adalah perempuan remaja". Predikat $\text{TeenGirl}(x)$ didefinisikan sebagai $\text{TeenGirl}(x) := \text{Female}(x) \wedge \text{Teen}(x)$.
5. $\text{LittleBoy}(x)$ : "$x$ adalah anak kecil laki-laki". Predikat $\text{LittleBoy}(x)$ didefinisikan sebagai $\text{LittleBoy}(x) := \text{Male}(x) \wedge \text{Kid}(x)$.
6. $\text{LittleGirl}(x)$ : "$x$ adalah anak kecil perempuan". Predikat $\text{LittleGirl}(x)$ didefinisikan sebagai $\text{LittleGirl}(x) :=$

# Translasi Formula Logika Predikat ke Prolog

Misalkan kita memiliki predikat-predikat turunan berikut:

1. $\text{Gentleman}(x)$ : "$x$ adalah laki-laki dewasa". Predikat $\text{Gentleman}(x)$ didefinisikan sebagai $\text{Gentleman}(x) := \text{Male}(x) \wedge \text{Adult}(x)$.
2. $\text{Lady}(x)$ : "$x$ adalah perempuan dewasa". Predikat $\text{Lady}(x)$ didefinisikan sebagai $\text{Lady}(x) := \text{Female}(x) \wedge \text{Adult}(x)$.
3. $\text{TeenBoy}(x)$ : "$x$ adalah laki-laki remaja". Predikat $\text{TeenBoy}(x)$ didefinisikan sebagai $\text{TeenBoy}(x) := \text{Male}(x) \wedge \text{Teen}(x)$.
4. $\text{TeenGirl}(x)$ : "$x$ adalah perempuan remaja". Predikat $\text{TeenGirl}(x)$ didefinisikan sebagai $\text{TeenGirl}(x) := \text{Female}(x) \wedge \text{Teen}(x)$.
5. $\text{LittleBoy}(x)$ : "$x$ adalah anak kecil laki-laki". Predikat $\text{LittleBoy}(x)$ didefinisikan sebagai $\text{LittleBoy}(x) := \text{Male}(x) \wedge \text{Kid}(x)$.
6. $\text{LittleGirl}(x)$ : "$x$ adalah anak kecil perempuan". Predikat $\text{LittleGirl}(x)$ didefinisikan sebagai $\text{LittleGirl}(x) := \text{Female}(x) \wedge \text{Kid}(x)$.

Misalkan pada Prolog, keenam predikat tersebut ditranslasikan sebagai `gentleman`, `lady`, `teen_boy`, `teen_girl`, `little_boy`, dan `little_girl`. Keenam predikat ini dapat didefinisikan sebagai aturan (*rule*) yang diturunkan dari predikat-predikat yang telah ada (yaitu `male`, `female`, `adult`, `teen`, dan `kid`). Pendefinisian dilakukan sebagai berikut:

Misalkan pada Prolog, keenam predikat tersebut ditranslasikan sebagai `gentleman`, `lady`, `teen_boy`, `teen_girl`, `little_boy`, dan `little_girl`. Keenam predikat ini dapat didefinisikan sebagai aturan (*rule*) yang diturunkan dari predikat-predikat yang telah ada (yaitu `male`, `female`, `adult`, `teen`, dan `kid`). Pendefinisian dilakukan sebagai berikut:

```
gentleman(X):- male(X),adult(X).
lady(X):- female(X),adult(X).

teen_boy(X):- male(X),teen(X).
teen_girl(X):- female(X),teen(X).

little_boy(X):- male(X),kid(X).
little_girl(X):- female(X),kid(X).
```

Ekspresi `gentleman(X):- male(X),adult(X)` merupakan suatu klausa (*clause*) pada Prolog.

# Klausa (*Clause*), Fakta (*Fact*), dan Aturan (*Rule*)

Sebuah skrip Prolog merupakan kumpulan beberapa klausa (*clause*). Sebuah klausa dapat berupa fakta (*fact*) – yaitu hal yang didefinisikan benar secara langsung (contohnya `male(bob)`) atau aturan (*rule*) – yang merupakan formula yang diturunkan dari beberapa formula fakta. Setiap klausa harus diakhiri dengan tanda titik (`.`).

Suatu aturan (*rule*) dapat berbentuk
$\langle \texttt{head} \rangle$`:-` $\langle \texttt{t}_1 \rangle, \langle \texttt{t}_2 \rangle, \ldots, \langle \texttt{t}_\texttt{n} \rangle$ dengan $\texttt{n} \geq 1$. Tanda "`,`" berarti konjungsi ($\wedge$). Tanda "`,`" dapat diganti dengan "`;`" yang berarti disjungsi ($\vee$).

Pada suatu klausa:

# Klausa (*Clause*), Fakta (*Fact*), dan Aturan (*Rule*)

Sebuah skrip Prolog merupakan kumpulan beberapa klausa (*clause*). Sebuah klausa dapat berupa fakta (*fact*) – yaitu hal yang didefinisikan benar secara langsung (contohnya `male(bob)`) atau aturan (*rule*) – yang merupakan formula yang diturunkan dari beberapa formula fakta. Setiap klausa harus diakhiri dengan tanda titik (`.`).

Suatu aturan (*rule*) dapat berbentuk
$\langle \texttt{head} \rangle$`:-` $\langle \texttt{t}_1 \rangle, \langle \texttt{t}_2 \rangle, \ldots, \langle \texttt{t}_\texttt{n} \rangle$ dengan $\texttt{n} \geq 1$. Tanda "`,`" berarti konjungsi ($\wedge$). Tanda "`,`" dapat diganti dengan "`;`" yang berarti disjungsi ($\vee$).

Pada suatu klausa:

- $\langle \texttt{head} \rangle$ disebut kepala (*head*) dari klausa, $\langle \texttt{head} \rangle$ biasanya mendefinisikan suatu predikat baru yang dapat diturunkan dari predikat pada fakta

# Klausa (*Clause*), Fakta (*Fact*), dan Aturan (*Rule*)

Sebuah skrip Prolog merupakan kumpulan beberapa klausa (*clause*). Sebuah klausa dapat berupa fakta (*fact*) – yaitu hal yang didefinisikan benar secara langsung (contohnya `male(bob)`) atau aturan (*rule*) – yang merupakan formula yang diturunkan dari beberapa formula fakta. Setiap klausa harus diakhiri dengan tanda titik (`.`).

Suatu aturan (*rule*) dapat berbentuk
$\langle \texttt{head} \rangle$`:-` $\langle \texttt{t}_1 \rangle, \langle \texttt{t}_2 \rangle, \ldots, \langle \texttt{t}_\texttt{n} \rangle$ dengan $\texttt{n} \geq 1$. Tanda "`,`" berarti konjungsi ($\wedge$). Tanda "`,`" dapat diganti dengan "`;`" yang berarti disjungsi ($\vee$).

Pada suatu klausa:

- $\langle \texttt{head} \rangle$ disebut kepala (*head*) dari klausa, $\langle \texttt{head} \rangle$ biasanya mendefinisikan suatu predikat baru yang dapat diturunkan dari predikat pada fakta
- $\langle \texttt{t}_1 \rangle, \langle \texttt{t}_2 \rangle, \ldots, \langle \texttt{t}_\texttt{n} \rangle$ disebut badan (*body*) dari klausa, badan dari suatu klausa merepresentasikan kondisi yang direpresentasikan oleh kepala klausa

# Klausa (*Clause*), Fakta (*Fact*), dan Aturan (*Rule*)

Sebuah skrip Prolog merupakan kumpulan beberapa klausa (*clause*). Sebuah klausa dapat berupa fakta (*fact*) – yaitu hal yang didefinisikan benar secara langsung (contohnya `male(bob)`) atau aturan (*rule*) – yang merupakan formula yang diturunkan dari beberapa formula fakta. Setiap klausa harus diakhiri dengan tanda titik (`.`).

Suatu aturan (*rule*) dapat berbentuk
$\langle \texttt{head} \rangle$ `:-` $\langle \texttt{t}_1 \rangle, \langle \texttt{t}_2 \rangle, \ldots, \langle \texttt{t}_\texttt{n} \rangle$ dengan $\texttt{n} \geq 1$. Tanda "`,`" berarti konjungsi ($\wedge$). Tanda "`,`" dapat diganti dengan "`;`" yang berarti disjungsi ($\vee$).

Pada suatu klausa:

- $\langle \texttt{head} \rangle$ disebut kepala (*head*) dari klausa, $\langle \texttt{head} \rangle$ biasanya mendefinisikan suatu predikat baru yang dapat diturunkan dari predikat pada fakta
- $\langle \texttt{t}_1 \rangle, \langle \texttt{t}_2 \rangle, \ldots, \langle \texttt{t}_\texttt{n} \rangle$ disebut badan (*body*) dari klausa, badan dari suatu klausa merepresentasikan kondisi yang direpresentasikan oleh kepala klausa
- `:-` disebut leher (*neck*) dari klausa, simbol ini serupa dengan simbol *assignment* pada program imperatif, secara semantik, leher klausa dapat dibaca sebagai kata "jika (*if*)".

Pada setiap klausa biasanya: seluruh variabel diawali dengan huruf kapital dan term konstanta (objek) diawali dengan huruf kecil.

# Makna (Semantik) Deklaratif pada Prolog

Misalkan terdapat klausa:
⟨P⟩ `:-` ⟨Q⟩`,`⟨R⟩`.`

Klausa tersebut memiliki makna deklaratif:
"⟨P⟩ benar bila ⟨Q⟩ **dan** ⟨R⟩ benar", atau
"jika ⟨Q⟩ **dan** ⟨R⟩ dipenuhi, maka ⟨P⟩ juga dipenuhi".

Serupa dengan hal di atas, klausa ⟨P⟩ `:-` ⟨Q⟩`;`⟨R⟩`.` berarti "jika ⟨Q⟩ **atau** ⟨R⟩ dipenuhi, maka ⟨P⟩ dipenuhi".

## Contoh

# Makna (Semantik) Deklaratif pada Prolog

Misalkan terdapat klausa:
⟨P⟩ `:-` ⟨Q⟩`,`⟨R⟩`.`

Klausa tersebut memiliki makna deklaratif:
"⟨P⟩ benar bila ⟨Q⟩ **dan** ⟨R⟩ benar", atau
"jika ⟨Q⟩ **dan** ⟨R⟩ dipenuhi, maka ⟨P⟩ juga dipenuhi".

Serupa dengan hal di atas, klausa ⟨P⟩ `:-` ⟨Q⟩`;`⟨R⟩`.` berarti "jika ⟨Q⟩ **atau** ⟨R⟩ dipenuhi, maka ⟨P⟩ dipenuhi".

## Contoh

Kita memiliki `gentlemen(X):- male(X),adult(X)`, ini setara dengan kondisi formula logika predikat yang menyatakan $\text{Male}(x) \wedge \text{Adult}(x) \rightarrow \text{Gentleman}(x)$. Dengan perkataan lain, jika $x$ adalah laki-laki dan $x$ dewasa, maka $x$ adalah *gentleman*.

# Contoh Translasi Sederhana dengan Disjungsi

Dari skrip Prolog yang telah kita buat sebelumnya, misalkan kita ingin menyatakan dua hal berikut:

1. semua remaja maupun anak kecil laki-laki menyukai permainan FIFA21
2. semua remaja maupun anak kecil perempuan menyukai permainan *Candy Crush*

Kita dapat mendefinisikan formula logika predikat:

1. $\text{LovesFIFA21}\,(x) :=$

# Contoh Translasi Sederhana dengan Disjungsi

Dari skrip Prolog yang telah kita buat sebelumnya, misalkan kita ingin menyatakan dua hal berikut:

1. semua remaja maupun anak kecil laki-laki menyukai permainan FIFA21
2. semua remaja maupun anak kecil perempuan menyukai permainan *Candy Crush*

Kita dapat mendefinisikan formula logika predikat:

1. $\text{LovesFIFA21}\,(x) := \text{TeenBoy}\,(x) \vee \text{LittleBoy}\,(x)$
2. $\text{LovesCandyCrush}\,(x) :=$

# Contoh Translasi Sederhana dengan Disjungsi

Dari skrip Prolog yang telah kita buat sebelumnya, misalkan kita ingin menyatakan dua hal berikut:

1. semua remaja maupun anak kecil laki-laki menyukai permainan FIFA21
2. semua remaja maupun anak kecil perempuan menyukai permainan *Candy Crush*

Kita dapat mendefinisikan formula logika predikat:

1. $\text{LovesFIFA21}\,(x) := \text{TeenBoy}\,(x) \vee \text{LittleBoy}\,(x)$
2. $\text{LovesCandyCrush}\,(x) := \text{TeenGirl}\,(x) \vee \text{LittleGirl}\,(x)$

Kedua formula tersebut dapat ditranslasikan ke dalam Prolog menjadi:

# Contoh Translasi Sederhana dengan Disjungsi

Dari skrip Prolog yang telah kita buat sebelumnya, misalkan kita ingin menyatakan dua hal berikut:

1. semua remaja maupun anak kecil laki-laki menyukai permainan FIFA21
2. semua remaja maupun anak kecil perempuan menyukai permainan *Candy Crush*

Kita dapat mendefinisikan formula logika predikat:

1. $\text{LovesFIFA21}\,(x) := \text{TeenBoy}\,(x) \vee \text{LittleBoy}\,(x)$
2. $\text{LovesCandyCrush}\,(x) := \text{TeenGirl}\,(x) \vee \text{LittleGirl}\,(x)$

Kedua formula tersebut dapat ditranslasikan ke dalam Prolog menjadi:

```
loves_FIFA21(X):- teen_boy(X); little_boy(X).
loves_CandyCrush(X):- teen_girl(X); little_girl(X).
```

# Pendefinisian Predikat Baru dengan Penukaran Objek

Predikat $\mathrm{Parent}\,(x, y)$ menyatakan bahwa "$x$ adalah orang tua $y$". Kita dapat membuat predikat $\mathrm{Child}\,(x, y)$ yang menyatakan bahwa "$x$ adalah anak dari $y$". Predikat $\mathrm{Child}\,(x, y)$ dapat didefinisikan memakai predikat $\mathrm{Parent}\,(x, y)$ yang sudah ada sebelumnya. Perhatikan bahwa

$\mathrm{Child}\,(x, y)$ berarti

# Pendefinisian Predikat Baru dengan Penukaran Objek

Predikat $\mathrm{Parent}\,(x, y)$ menyatakan bahwa "$x$ adalah orang tua $y$". Kita dapat membuat predikat $\mathrm{Child}\,(x, y)$ yang menyatakan bahwa "$x$ adalah anak dari $y$". Predikat $\mathrm{Child}\,(x, y)$ dapat didefinisikan memakai predikat $\mathrm{Parent}\,(x, y)$ yang sudah ada sebelumnya. Perhatikan bahwa

$\mathrm{Child}\,(x, y)$ berarti "$x$ adalah anak dari $y$"
berarti

# Pendefinisian Predikat Baru dengan Penukaran Objek

Predikat $\mathrm{Parent}\,(x, y)$ menyatakan bahwa "$x$ adalah orang tua $y$". Kita dapat membuat predikat $\mathrm{Child}\,(x, y)$ yang menyatakan bahwa "$x$ adalah anak dari $y$". Predikat $\mathrm{Child}\,(x, y)$ dapat didefinisikan memakai predikat $\mathrm{Parent}\,(x, y)$ yang sudah ada sebelumnya. Perhatikan bahwa

| | | |
|---|---|---|
| $\mathrm{Child}\,(x, y)$ | berarti | "$x$ adalah anak dari $y$" |
| | berarti | "$y$ adalah orang tua dari $x$" |
| | berarti | |

# Pendefinisian Predikat Baru dengan Penukaran Objek

Predikat $\mathrm{Parent}\,(x, y)$ menyatakan bahwa "$x$ adalah orang tua $y$". Kita dapat membuat predikat $\mathrm{Child}\,(x, y)$ yang menyatakan bahwa "$x$ adalah anak dari $y$". Predikat $\mathrm{Child}\,(x, y)$ dapat didefinisikan memakai predikat $\mathrm{Parent}\,(x, y)$ yang sudah ada sebelumnya. Perhatikan bahwa

$$\begin{array}{lll} \mathrm{Child}\,(x, y) & \text{berarti} & \text{"}x \text{ adalah anak dari } y\text{"} \\ & \text{berarti} & \text{"}y \text{ adalah orang tua dari } x\text{"} \\ & \text{berarti} & \mathrm{Parent}\,(y, x)\,. \end{array}$$

Jadi kita dapat mendefinisikan $\mathrm{Child}\,(x, y) :=$

# Pendefinisian Predikat Baru dengan Penukaran Objek

Predikat $\mathrm{Parent}\,(x, y)$ menyatakan bahwa "$x$ adalah orang tua $y$". Kita dapat membuat predikat $\mathrm{Child}\,(x, y)$ yang menyatakan bahwa "$x$ adalah anak dari $y$". Predikat $\mathrm{Child}\,(x, y)$ dapat didefinisikan memakai predikat $\mathrm{Parent}\,(x, y)$ yang sudah ada sebelumnya. Perhatikan bahwa

$$\begin{array}{lll} \mathrm{Child}\,(x, y) & \text{berarti} & \text{"}x \text{ adalah anak dari } y\text{"} \\ & \text{berarti} & \text{"}y \text{ adalah orang tua dari } x\text{"} \\ & \text{berarti} & \mathrm{Parent}\,(y, x)\,. \end{array}$$

Jadi kita dapat mendefinisikan $\mathrm{Child}\,(x, y) := \mathrm{Parent}\,(y, x)$. Pada Prolog hal ini dapat dilakukan dengan mudah sebagai berikut.

# Pendefinisian Predikat Baru dengan Penukaran Objek

Predikat $\mathrm{Parent}\,(x, y)$ menyatakan bahwa "$x$ adalah orang tua $y$". Kita dapat membuat predikat $\mathrm{Child}\,(x, y)$ yang menyatakan bahwa "$x$ adalah anak dari $y$". Predikat $\mathrm{Child}\,(x, y)$ dapat didefinisikan memakai predikat $\mathrm{Parent}\,(x, y)$ yang sudah ada sebelumnya. Perhatikan bahwa

$$\begin{array}{lll} \mathrm{Child}\,(x, y) & \text{berarti} & \text{"}x \text{ adalah anak dari } y\text{"} \\ & \text{berarti} & \text{"}y \text{ adalah orang tua dari } x\text{"} \\ & \text{berarti} & \mathrm{Parent}\,(y, x)\,. \end{array}$$

Jadi kita dapat mendefinisikan $\mathrm{Child}\,(x, y) := \mathrm{Parent}\,(y, x)$. Pada Prolog hal ini dapat dilakukan dengan mudah sebagai berikut.

```
child(X,Y):- parent(Y,X).
```

# Latihan 1

## Latihan

1. Diberikan domain $D$ dan predikat $\mathrm{Parent}\,(x, y)$ : "$x$ adalah orang tua dari $y$", $\mathrm{Male}\,(x)$ : "$x$ adalah laki-laki", dan $\mathrm{Female}\,(x)$ : "$x$ adalah perempuan". Hanya dengan memakai predikat-predikat ini, berikan definisi untuk predikat $\mathrm{Father}\,(x, y)$ dan $\mathrm{Mother}\,(x, y)$, predikat $\mathrm{Father}\,(x, y)$ berarti "$x$ adalah ayah dari $y$" dan predikat $\mathrm{Mother}\,(x, y)$ berarti "$x$ adalah ibu dari $y$".
2. Gunakan hasil pada nomor 1 untuk mendefinisikan `father(X,Y)` dan `mother(X,Y)` pada Prolog.

Solusi:

# Latihan 1

## Latihan

1. Diberikan domain $D$ dan predikat $\mathrm{Parent}\,(x, y)$ : "$x$ adalah orang tua dari $y$", $\mathrm{Male}\,(x)$ : "$x$ adalah laki-laki", dan $\mathrm{Female}\,(x)$ : "$x$ adalah perempuan". Hanya dengan memakai predikat-predikat ini, berikan definisi untuk predikat $\mathrm{Father}\,(x, y)$ dan $\mathrm{Mother}\,(x, y)$, predikat $\mathrm{Father}\,(x, y)$ berarti "$x$ adalah ayah dari $y$" dan predikat $\mathrm{Mother}\,(x, y)$ berarti "$x$ adalah ibu dari $y$".
2. Gunakan hasil pada nomor 1 untuk mendefinisikan `father(X,Y)` dan `mother(X,Y)` pada Prolog.

Solusi:

1. $\mathrm{Father}\,(x, y) := \mathrm{Parent}\,(x, y) \wedge \mathrm{Male}\,(x)$ dan $\mathrm{Mother}\,(x, y) := \mathrm{Parent}\,(x, y) \wedge \mathrm{Female}\,(x)$. Hal ini terjadi karena ayah adalah orang tua laki-laki dan ibu adalah orang tua perempuan.

# Latihan 1

## Latihan

1. Diberikan domain $D$ dan predikat $\text{Parent}(x, y)$ : "$x$ adalah orang tua dari $y$", $\text{Male}(x)$ : "$x$ adalah laki-laki", dan $\text{Female}(x)$ : "$x$ adalah perempuan". Hanya dengan memakai predikat-predikat ini, berikan definisi untuk predikat $\text{Father}(x, y)$ dan $\text{Mother}(x, y)$, predikat $\text{Father}(x, y)$ berarti "$x$ adalah ayah dari $y$" dan predikat $\text{Mother}(x, y)$ berarti "$x$ adalah ibu dari $y$".
2. Gunakan hasil pada nomor 1 untuk mendefinisikan `father(X,Y)` dan `mother(X,Y)` pada Prolog.

Solusi:

1. $\text{Father}(x, y) := \text{Parent}(x, y) \wedge \text{Male}(x)$ dan $\text{Mother}(x, y) := \text{Parent}(x, y) \wedge \text{Female}(x)$. Hal ini terjadi karena ayah adalah orang tua laki-laki dan ibu adalah orang tua perempuan.
2. Kita memiliki `father(X,Y):- parent(X,Y),male(X)` dan `mother(X,Y):- parent(X,Y),female(X).`

# Bahasan

# Represetasi Kuantor Universal pada Prolog

Misalkan terdapat suatu klausa yang berbentuk:
$\langle \texttt{head} \rangle$ :- $\langle \texttt{t}_1 \rangle, \langle \texttt{t}_2 \rangle, \ldots, \langle \texttt{t}_\texttt{n} \rangle$ dengan $\texttt{n} \geq 1$ (operator , yang menyatakan konjungsi dapat diganti dengan ; yang menyatakan disjungsi).
Semantik deklaratif dari klausa di atas adalah:
Bila $\langle \texttt{t}_1 \rangle$, $\langle \texttt{t}_2 \rangle$, ..., $\langle \texttt{t}_\texttt{n} \rangle$ benar, maka $\langle \texttt{head} \rangle$ juga benar, atau
$\langle \texttt{t}_1 \rangle \wedge \langle \texttt{t}_2 \rangle \wedge \cdots \wedge \langle \texttt{t}_\texttt{n} \rangle \rightarrow \langle \texttt{head} \rangle$ bernilai T.

## Representasi Kuantor Universal pada Prolog

Jika sebuah variabel muncul pada kepala/ $\langle \texttt{head} \rangle$ dari suatu klausa, maka variabel tersebut diikat oleh kuantor universal ($\forall$).

## Contoh

# Represetasi Kuantor Universal pada Prolog

Misalkan terdapat suatu klausa yang berbentuk:
$\langle \texttt{head} \rangle$ `:-` $\langle \texttt{t}_1 \rangle, \langle \texttt{t}_2 \rangle, \ldots, \langle \texttt{t}_\texttt{n} \rangle$ dengan $\texttt{n} \geq 1$ (operator `,` yang menyatakan konjungsi dapat diganti dengan `;` yang menyatakan disjungsi).
Semantik deklaratif dari klausa di atas adalah:
Bila $\langle \texttt{t}_1 \rangle$, $\langle \texttt{t}_2 \rangle$, $\ldots$, $\langle \texttt{t}_\texttt{n} \rangle$ benar, maka $\langle \texttt{head} \rangle$ juga benar, atau
$\langle \texttt{t}_1 \rangle \wedge \langle \texttt{t}_2 \rangle \wedge \cdots \wedge \langle \texttt{t}_\texttt{n} \rangle \rightarrow \langle \texttt{head} \rangle$ bernilai $\mathrm{T}$.

## Representasi Kuantor Universal pada Prolog

Jika sebuah variabel muncul pada kepala/ $\langle \texttt{head} \rangle$ dari suatu klausa, maka variabel tersebut diikat oleh kuantor universal ($\forall$).

## Contoh

Kita telah melihat aturan `gentleman(X):- male(X),adult(X)`. Dalam logika predikat, aturan ini dapat ditranslasikan sebagai
$\mathrm{Male}(x) \wedge \mathrm{Adult}(x) \rightarrow \mathrm{Gentleman}(x)$. Karena `X` merupakan variabel yang muncul pada kepala/ $\langle \texttt{head} \rangle$ suatu klausa, maka `X` diikat dengan kuantor universal. Ini berarti aturan `gentleman(X):- male(X),adult(X)` dapat dinyatakan sebagai formula $\forall x \left(\mathrm{Male}(x) \wedge \mathrm{Adult}(x) \rightarrow \mathrm{Gentleman}(x)\right)$.

Dari aturan-aturan yang telah didefinisikan sebelumnya, kita memiliki:

- `lady(X):- female(X),adult(X)` dapat dinyatakan sebagai

Dari aturan-aturan yang telah didefinisikan sebelumnya, kita memiliki:

- `lady(X):- female(X),adult(X)` dapat dinyatakan sebagai $\forall x \left(\text{Female}\left(x\right) \wedge \text{Adult}\left(x\right) \rightarrow \text{Lady}\left(x\right)\right)$
- `teen_boy(X):- male(X),teen(X)` dapat dinyatakan sebagai

Dari aturan-aturan yang telah didefinisikan sebelumnya, kita memiliki:

- `lady(X):- female(X),adult(X)` dapat dinyatakan sebagai $\forall x \left(\text{Female}\left(x\right) \wedge \text{Adult}\left(x\right) \rightarrow \text{Lady}\left(x\right)\right)$
- `teen_boy(X):- male(X),teen(X)` dapat dinyatakan sebagai $\forall x \left(\text{Male}\left(x\right) \wedge \text{Teen}\left(x\right) \rightarrow \text{TeenBoy}\left(x\right)\right)$
- `teen_girl(X):- female(X),teen(X)` dapat dinyatakan sebagai

Dari aturan-aturan yang telah didefinisikan sebelumnya, kita memiliki:

- `lady(X):- female(X),adult(X)` dapat dinyatakan sebagai $\forall x\,(\text{Female}\,(x) \wedge \text{Adult}\,(x) \rightarrow \text{Lady}\,(x))$
- `teen_boy(X):- male(X),teen(X)` dapat dinyatakan sebagai $\forall x\,(\text{Male}\,(x) \wedge \text{Teen}\,(x) \rightarrow \text{TeenBoy}\,(x))$
- `teen_girl(X):- female(X),teen(X)` dapat dinyatakan sebagai $\forall x\,(\text{Female}\,(x) \wedge \text{Teen}\,(x) \rightarrow \text{TeenGirl}\,(x))$
- `little_boy(X):- male(X),kid(X)` dapat dinyatakan sebagai

Dari aturan-aturan yang telah didefinisikan sebelumnya, kita memiliki:

- `lady(X):- female(X),adult(X)` dapat dinyatakan sebagai $\forall x\,(\text{Female}\,(x) \wedge \text{Adult}\,(x) \rightarrow \text{Lady}\,(x))$
- `teen_boy(X):- male(X),teen(X)` dapat dinyatakan sebagai $\forall x\,(\text{Male}\,(x) \wedge \text{Teen}\,(x) \rightarrow \text{TeenBoy}\,(x))$
- `teen_girl(X):- female(X),teen(X)` dapat dinyatakan sebagai $\forall x\,(\text{Female}\,(x) \wedge \text{Teen}\,(x) \rightarrow \text{TeenGirl}\,(x))$
- `little_boy(X):- male(X),kid(X)` dapat dinyatakan sebagai $\forall x\,(\text{Male}\,(x) \wedge \text{Kid}\,(x) \rightarrow \text{LittleBoy}\,(x))$
- `little_girl(X):- female(X),kid(X)` dapat dinyatakan sebagai

Dari aturan-aturan yang telah didefinisikan sebelumnya, kita memiliki:

- `lady(X):- female(X),adult(X)` dapat dinyatakan sebagai $\forall x\,(\text{Female}\,(x) \wedge \text{Adult}\,(x) \rightarrow \text{Lady}\,(x))$
- `teen_boy(X):- male(X),teen(X)` dapat dinyatakan sebagai $\forall x\,(\text{Male}\,(x) \wedge \text{Teen}\,(x) \rightarrow \text{TeenBoy}\,(x))$
- `teen_girl(X):- female(X),teen(X)` dapat dinyatakan sebagai $\forall x\,(\text{Female}\,(x) \wedge \text{Teen}\,(x) \rightarrow \text{TeenGirl}\,(x))$
- `little_boy(X):- male(X),kid(X)` dapat dinyatakan sebagai $\forall x\,(\text{Male}\,(x) \wedge \text{Kid}\,(x) \rightarrow \text{LittleBoy}\,(x))$
- `little_girl(X):- female(X),kid(X)` dapat dinyatakan sebagai $\forall x\,(\text{Female}\,(x) \wedge \text{Kid}\,(x) \rightarrow \text{LittleGirl}\,(x))$
- `loves_FIFA21(X):- teen_boy(X);little_boy(X)` dapat dinyatakan sebagai

Dari aturan-aturan yang telah didefinisikan sebelumnya, kita memiliki:

- `lady(X):- female(X),adult(X)` dapat dinyatakan sebagai $\forall x\,(\text{Female}\,(x) \wedge \text{Adult}\,(x) \rightarrow \text{Lady}\,(x))$
- `teen_boy(X):- male(X),teen(X)` dapat dinyatakan sebagai $\forall x\,(\text{Male}\,(x) \wedge \text{Teen}\,(x) \rightarrow \text{TeenBoy}\,(x))$
- `teen_girl(X):- female(X),teen(X)` dapat dinyatakan sebagai $\forall x\,(\text{Female}\,(x) \wedge \text{Teen}\,(x) \rightarrow \text{TeenGirl}\,(x))$
- `little_boy(X):- male(X),kid(X)` dapat dinyatakan sebagai $\forall x\,(\text{Male}\,(x) \wedge \text{Kid}\,(x) \rightarrow \text{LittleBoy}\,(x))$
- `little_girl(X):- female(X),kid(X)` dapat dinyatakan sebagai $\forall x\,(\text{Female}\,(x) \wedge \text{Kid}\,(x) \rightarrow \text{LittleGirl}\,(x))$
- `loves_FIFA21(X):- teen_boy(X);little_boy(X)` dapat dinyatakan sebagai $\forall x\,(\text{TeenBoy}\,(x) \vee \text{LittleBoy}\,(x) \rightarrow \text{LovesFIFA21}\,(x))$
- `loves_CandyCrush(X):- teen_girl(X);little_girl(X)` dapat dinyatakan sebagai

Dari aturan-aturan yang telah didefinisikan sebelumnya, kita memiliki:

- `lady(X):- female(X),adult(X)` dapat dinyatakan sebagai $\forall x\,(\text{Female}\,(x) \wedge \text{Adult}\,(x) \rightarrow \text{Lady}\,(x))$
- `teen_boy(X):- male(X),teen(X)` dapat dinyatakan sebagai $\forall x\,(\text{Male}\,(x) \wedge \text{Teen}\,(x) \rightarrow \text{TeenBoy}\,(x))$
- `teen_girl(X):- female(X),teen(X)` dapat dinyatakan sebagai $\forall x\,(\text{Female}\,(x) \wedge \text{Teen}\,(x) \rightarrow \text{TeenGirl}\,(x))$
- `little_boy(X):- male(X),kid(X)` dapat dinyatakan sebagai $\forall x\,(\text{Male}\,(x) \wedge \text{Kid}\,(x) \rightarrow \text{LittleBoy}\,(x))$
- `little_girl(X):- female(X),kid(X)` dapat dinyatakan sebagai $\forall x\,(\text{Female}\,(x) \wedge \text{Kid}\,(x) \rightarrow \text{LittleGirl}\,(x))$
- `loves_FIFA21(X):- teen_boy(X);little_boy(X)` dapat dinyatakan sebagai $\forall x\,(\text{TeenBoy}\,(x) \vee \text{LittleBoy}\,(x) \rightarrow \text{LovesFIFA21}\,(x))$
- `loves_CandyCrush(X):- teen_girl(X);little_girl(X)` dapat dinyatakan sebagai $\forall x\,(\text{TeenGirl}\,(x) \vee \text{LittleGirl}\,(x) \rightarrow \text{LovesCandyCrush}\,(x))$

Kemudian

- `child(X,Y):- parent(Y,X)` dapat dinyatakan sebagai

Kemudian

- `child(X,Y):- parent(Y,X)` dapat dinyatakan sebagai
  $\forall x \forall y \left(\text{Parent}\left(y, x\right) \rightarrow \text{Child}\left(x, y\right)\right)$
- `father(X,Y):- parent(X,Y),male(X)` dapat dinyatakan sebagai

Kemudian

- `child(X,Y):- parent(Y,X)` dapat dinyatakan sebagai
  $\forall x \forall y \left(\text{Parent}\left(y, x\right) \rightarrow \text{Child}\left(x, y\right)\right)$
- `father(X,Y):- parent(X,Y),male(X)` dapat dinyatakan sebagai
  $\forall x \forall y \left(\text{Parent}\left(x, y\right) \wedge \text{Male}\left(x\right) \rightarrow \text{Father}\left(x, y\right)\right)$
- `mother(X,Y):- parent(X,Y),female(X)` dapat dinyatakan sebagai

Kemudian

- `child(X,Y):- parent(Y,X)` dapat dinyatakan sebagai
  $\forall x \forall y \left(\text{Parent}\left(y, x\right) \rightarrow \text{Child}\left(x, y\right)\right)$
- `father(X,Y):- parent(X,Y),male(X)` dapat dinyatakan sebagai
  $\forall x \forall y \left(\text{Parent}\left(x, y\right) \wedge \text{Male}\left(x\right) \rightarrow \text{Father}\left(x, y\right)\right)$
- `mother(X,Y):- parent(X,Y),female(X)` dapat dinyatakan sebagai
  $\forall x \forall y \left(\text{Parent}\left(x, y\right) \wedge \text{Female}\left(x\right) \rightarrow \text{Mother}\left(x, y\right)\right)$

# Representasi Kuantor Eksistensial pada Prolog

Misalkan kita mendefinisikan predikat $\mathrm{Grandparent}(x, y)$ : "$x$ adalah *grandparent* (kakek atau nenek, tanpa melihat jenis kelaminnya) dari $y$". Kita dapat mendefinsikan predikat $\mathrm{Grandparent}(x, y)$ dari predikat $\mathrm{Parent}(x, y)$. Perhatikan bahwa

$\mathrm{Grandparent}(x, y)$ berarti

# Representasi Kuantor Eksistensial pada Prolog

Misalkan kita mendefinisikan predikat $\mathrm{Grandparent}(x, y)$ : "$x$ adalah *grandparent* (kakek atau nenek, tanpa melihat jenis kelaminnya) dari $y$". Kita dapat mendefinsikan predikat $\mathrm{Grandparent}(x, y)$ dari predikat $\mathrm{Parent}(x, y)$. Perhatikan bahwa

| | | |
|---|---|---|
| $\mathrm{Grandparent}(x, y)$ | berarti | "$x$ adalah *grandparent* dari $y$" |
| | berarti | |

# Representasi Kuantor Eksistensial pada Prolog

Misalkan kita mendefinisikan predikat $\mathrm{Grandparent}(x, y)$ : "$x$ adalah *grandparent* (kakek atau nenek, tanpa melihat jenis kelaminnya) dari $y$". Kita dapat mendefinsikan predikat $\mathrm{Grandparent}(x, y)$ dari predikat $\mathrm{Parent}(x, y)$. Perhatikan bahwa

| | | |
|---|---|---|
| $\mathrm{Grandparent}(x, y)$ | berarti | "$x$ adalah *grandparent* dari $y$" |
| | berarti | "**terdapat** $z$ sehingga $x$ adalah orang tua $z$ **dan** $z$ adalah orang tua $y$" |
| | berarti | |

# Representasi Kuantor Eksistensial pada Prolog

Misalkan kita mendefinisikan predikat $\mathrm{Grandparent}(x, y)$ : "$x$ adalah *grandparent* (kakek atau nenek, tanpa melihat jenis kelaminnya) dari $y$". Kita dapat mendefinsikan predikat $\mathrm{Grandparent}(x, y)$ dari predikat $\mathrm{Parent}(x, y)$. Perhatikan bahwa

| | | |
|---|---|---|
| $\mathrm{Grandparent}(x, y)$ | berarti | "$x$ adalah *grandparent* dari $y$" |
| | berarti | "**terdapat** $z$ sehingga $x$ adalah orang tua $z$ **dan** $z$ adalah orang tua $y$" |
| | berarti | terdapat $z$ sehingga $\mathrm{Parent}\,(x, z) \wedge \mathrm{Parent}\,(z, y)$ |
| | berarti | |

# Representasi Kuantor Eksistensial pada Prolog

Misalkan kita mendefinisikan predikat $\mathrm{Grandparent}(x, y)$ : "$x$ adalah *grandparent* (kakek atau nenek, tanpa melihat jenis kelaminnya) dari $y$". Kita dapat mendefinsikan predikat $\mathrm{Grandparent}(x, y)$ dari predikat $\mathrm{Parent}(x, y)$. Perhatikan bahwa

| | | |
|---|---|---|
| $\mathrm{Grandparent}(x, y)$ | berarti | "$x$ adalah *grandparent* dari $y$" |
| | berarti | "**terdapat** $z$ sehingga $x$ adalah orang tua $z$ **dan** $z$ adalah orang tua $y$" |
| | berarti | terdapat $z$ sehingga $\mathrm{Parent}\,(x, z) \wedge \mathrm{Parent}\,(z, y)$ |
| | berarti | $\exists z\,(\mathrm{Parent}\,(x, z) \wedge \mathrm{Parent}\,(z, y))$. |

Pada Prolog, hal ini dapat ditranslasikan sebagai

```
grandparent(X,Y):-
```

# Representasi Kuantor Eksistensial pada Prolog

Misalkan kita mendefinisikan predikat $\mathrm{Grandparent}(x, y)$ : "$x$ adalah *grandparent* (kakek atau nenek, tanpa melihat jenis kelaminnya) dari $y$". Kita dapat mendefinsikan predikat $\mathrm{Grandparent}(x, y)$ dari predikat $\mathrm{Parent}(x, y)$. Perhatikan bahwa

| | | |
|---|---|---|
| $\mathrm{Grandparent}(x, y)$ | berarti | "$x$ adalah *grandparent* dari $y$" |
| | berarti | "**terdapat** $z$ sehingga $x$ adalah orang tua $z$ **dan** $z$ adalah orang tua $y$" |
| | berarti | terdapat $z$ sehingga $\mathrm{Parent}\,(x, z) \wedge \mathrm{Parent}\,(z, y)$ |
| | berarti | $\exists z\,(\mathrm{Parent}\,(x, z) \wedge \mathrm{Parent}\,(z, y))$. |

Pada Prolog, hal ini dapat ditranslasikan sebagai

```
grandparent(X,Y):- parent(X,Z),parent(Z,Y).
```

## Representasi Kuantor Eksistensial pada Prolog

Jika sebuah variabel muncul pada badan (*body*) dari suatu klausa tetapi tidak muncul pada kepala/ ⟨head⟩ dari klausa tersebut, maka variabel tersebut diikat oleh kuantor eksistensial ($\exists$).

Akibatnya representasi formula logika predikat dari

```
grandparent(X,Y):- parent(X,Z),parent(Z,Y).
```

adalah

## Representasi Kuantor Eksistensial pada Prolog

Jika sebuah variabel muncul pada badan (*body*) dari suatu klausa tetapi tidak muncul pada kepala/ ⟨head⟩ dari klausa tersebut, maka variabel tersebut diikat oleh kuantor eksistensial ($\exists$).

Akibatnya representasi formula logika predikat dari

```
grandparent(X,Y):- parent(X,Z),parent(Z,Y).
```

adalah

$$\forall x \forall y \left(\exists z \left(\text{Parent}\left(x, z\right) \wedge \text{Parent}\left(z, y\right)\right) \rightarrow \text{Grandparent}\left(x, y\right)\right) \text{ atau}$$
$$\forall x \forall y \exists z \left(\text{Parent}\left(x, z\right) \wedge \text{Parent}\left(z, y\right) \rightarrow \text{Grandparent}\left(x, y\right)\right).$$

Dengan fakta-fakta program sebelumnya, hasil query untuk `grandparent(Grandparent,Grandkid)` adalah:

```
?- grandparent(Grandparent,Grandkid).
```

Dengan fakta-fakta program sebelumnya, hasil query untuk `grandparent(Grandparent,Grandkid)` adalah:

```
?- grandparent(Grandparent,Grandkid).
Grandparent = alice,
Grandkid = fiona ;
```

Dengan fakta-fakta program sebelumnya, hasil query untuk `grandparent(Grandparent,Grandkid)` adalah:

```
?- grandparent(Grandparent,Grandkid).
Grandparent = alice,
Grandkid = fiona ;
Grandparent = alice,
Grandkid = grace ;


: [ada cukup banyak keluaran]
```

Dengan fakta-fakta program sebelumnya, hasil query untuk `grandparent(Grandparent,Grandkid)` adalah:

```
?- grandparent(Grandparent,Grandkid).
Grandparent = alice,
Grandkid = fiona ;
Grandparent = alice,
Grandkid = grace ;


: [ada cukup banyak keluaran]

Grandparent = charlie,
Grandkid = kelly;
```

# Variabel Singleton

Misalkan kita mendefinisikan predikat $\text{Has-a-Child}(x)$ : "$x$ memiliki anak".
Predikat $\text{Has-a-Child}(x)$ bernilai $\text{T}$ bila terdapat $y$ sehingga $\text{Parent}\,(x, y)$ bernilai $\text{T}$. Ini berarti $x$ memiliki anaka bila terdapat $y$ sehingga $x$ orang tua dari $y$.
Akibatnya $\text{Has-a-Child}(x)$ dapat didefinisikan sebagai

$$\text{Has-a-Child}\,(x) :=$$

# Variabel Singleton

Misalkan kita mendefinisikan predikat $\text{Has-a-Child}(x)$ : "$x$ memiliki anak". Predikat $\text{Has-a-Child}(x)$ bernilai $\text{T}$ bila terdapat $y$ sehingga $\text{Parent}\,(x, y)$ bernilai $\text{T}$. Ini berarti $x$ memiliki anaka bila terdapat $y$ sehingga $x$ orang tua dari $y$. Akibatnya $\text{Has-a-Child}(x)$ dapat didefinisikan sebagai

$$\text{Has-a-Child}\,(x) := \exists y \text{Parent}\,(x, y)\,.$$

Atau dalam bentuk implikasi

# Variabel Singleton

Misalkan kita mendefinisikan predikat $\text{Has-a-Child}(x)$ : "$x$ memiliki anak". Predikat $\text{Has-a-Child}(x)$ bernilai $\text{T}$ bila terdapat $y$ sehingga $\text{Parent}\,(x, y)$ bernilai $\text{T}$. Ini berarti $x$ memiliki anaka bila terdapat $y$ sehingga $x$ orang tua dari $y$. Akibatnya $\text{Has-a-Child}(x)$ dapat didefinisikan sebagai

$$\text{Has-a-Child}\,(x) := \exists y \text{Parent}\,(x, y)\,.$$

Atau dalam bentuk implikasi $\forall x\,(\exists y\,(\text{Parent}\,(x, y) \rightarrow \text{Has-a-Child}\,(x)))$. Jika hal ini ditranslasikan ke Prolog secara langsung, maka kita memiliki skrip

# Variabel Singleton

Misalkan kita mendefinisikan predikat $\text{Has-a-Child}(x)$ : "$x$ memiliki anak". Predikat $\text{Has-a-Child}(x)$ bernilai $\text{T}$ bila terdapat $y$ sehingga $\text{Parent}\,(x, y)$ bernilai $\text{T}$. Ini berarti $x$ memiliki anaka bila terdapat $y$ sehingga $x$ orang tua dari $y$. Akibatnya $\text{Has-a-Child}(x)$ dapat didefinisikan sebagai

$$\text{Has-a-Child}\,(x) := \exists y \text{Parent}\,(x, y)\,.$$

Atau dalam bentuk implikasi $\forall x\,(\exists y\,(\text{Parent}\,(x, y) \rightarrow \text{Has-a-Child}\,(x)))$. Jika hal ini ditranslasikan ke Prolog secara langsung, maka kita memiliki skrip

```
has_a_child(X):- parent(X,Y).
```

Skrip di atas tidak dapat dikompilasi oleh SWI-Prolog, SWI-Prolog akan mengeluarkan peringatan: `Singleton variables: [Y]`.

## Tentang Variabel Singleton

Variabel singleton dapat berupa variabel yang muncul pada predikat biner (atau predikat $n$-ari, $n \geq 2$) di bagian badan (*body*) suatu klausa, namun variabel tersebut tidak muncul pada bagian kepala/ $\langle$`head`$\rangle$ klausa.

Untuk memperbaiki masalah variabel singleton ini kita dapat memperbaikinya dengan dua cara.

1. Tambahkan "_" (garis bawah/ *underscore*) di depan variabel singleton. Sehingga skrip untuk `has_a_child(X)` menjadi:

## Tentang Variabel Singleton

Variabel singleton dapat berupa variabel yang muncul pada predikat biner (atau predikat $n$-ari, $n \geq 2$) di bagian badan (*body*) suatu klausa, namun variabel tersebut tidak muncul pada bagian kepala/ ⟨`head`⟩ klausa.

Untuk memperbaiki masalah variabel singleton ini kita dapat memperbaikinya dengan dua cara.

1. Tambahkan "_" (garis bawah/ *underscore*) di depan variabel singleton. Sehingga skrip untuk `has_a_child(X)` menjadi:

```
has_a_child(X):- parent(X,_Y).
```

2. Berikan penjelasan tambahkan terkait variabel singleton yang tidak mengubah makna deklaratif dari klausa yang dibentuk. Karena domain untuk `Y` adalah manusia dan `Y` dapat pasti memenuhi salah satu di antara `male(Y)` atau `female(Y)`, maka skrip untuk `has_a_child(X)` menjadi:

## Tentang Variabel Singleton

Variabel singleton dapat berupa variabel yang muncul pada predikat biner (atau predikat $n$-ari, $n \geq 2$) di bagian badan (*body*) suatu klausa, namun variabel tersebut tidak muncul pada bagian kepala/ ⟨`head`⟩ klausa.

Untuk memperbaiki masalah variabel singleton ini kita dapat memperbaikinya dengan dua cara.

1. Tambahkan "_" (garis bawah/ *underscore*) di depan variabel singleton. Sehingga skrip untuk `has_a_child(X)` menjadi:

```
has_a_child(X):- parent(X,_Y).
```

2. Berikan penjelasan tambahkan terkait variabel singleton yang tidak mengubah makna deklaratif dari klausa yang dibentuk. Karena domain untuk `Y` adalah manusia dan `Y` dapat pasti memenuhi salah satu di antara `male(Y)` atau `female(Y)`, maka skrip untuk `has_a_child(X)` menjadi:

```
has_a_child(X):- parent(X,Y),(male(Y);female(Y)).
```

Skrip ini memiliki representasi formula logika predikat berikut:

## Tentang Variabel Singleton

Variabel singleton dapat berupa variabel yang muncul pada predikat biner (atau predikat $n$-ari, $n \geq 2$) di bagian badan (*body*) suatu klausa, namun variabel tersebut tidak muncul pada bagian kepala/ ⟨`head`⟩ klausa.

Untuk memperbaiki masalah variabel singleton ini kita dapat memperbaikinya dengan dua cara.

1. Tambahkan "_" (garis bawah/ *underscore*) di depan variabel singleton. Sehingga skrip untuk `has_a_child(X)` menjadi:

```
has_a_child(X):- parent(X,_Y).
```

2. Berikan penjelasan tambahkan terkait variabel singleton yang tidak mengubah makna deklaratif dari klausa yang dibentuk. Karena domain untuk `Y` adalah manusia dan `Y` dapat pasti memenuhi salah satu di antara `male(Y)` atau `female(Y)`, maka skrip untuk `has_a_child(X)` menjadi:

```
has_a_child(X):- parent(X,Y),(male(Y);female(Y)).
```

Skrip ini memiliki representasi formula logika predikat berikut:
$\forall x \left(\exists y \text{Parent}\left(x, y\right) \wedge \left(\text{Male}\left(y\right) \vee \text{Female}\left(y\right)\right) \rightarrow \text{Has-a-Child}\left(x\right)\right)$.

# Latihan 2

## Latihan

1. Diberikan domain $D$, predikat $\mathrm{Father}\,(x, y)$ : "$x$ adalah ayah dari $y$", dan predikat $\mathrm{Mother}\,(x, y)$ : "$x$ adalah ibu dari $y$". Hanya dengan memakai predikat-predikat ini, berikan definisi untuk predikat $\text{Is-a-Daddy}(x)$ dan $\text{Is-a-Mommy}(x)$, predikat $\text{Is-a-Daddy}(x)$ berarti "$x$ adalah seorang ayah" dan predikat $\text{Is-a-Mommy}(x)$ berarti "$x$ adalah seroang ibu".
2. Gunakan hasil pada nomor 1 untuk mendefinisikan `is_a_daddy(X)` dan `is_a_mommy(X)` pada Prolog.

Solusi:

Solusi:

1. $\text{Is-a-Daddy}(x) := \exists y \text{Father}\,(x, y)$ dan $\text{Is-a-Mommy}(x) := \exists y \text{Mother}\,(x, y)$.

Solusi:

1. $\text{Is-a-Daddy}(x) := \exists y \text{Father}\,(x, y)$ dan $\text{Is-a-Mommy}(x) := \exists y \text{Mother}\,(x, y)$.
2. Kita memiliki
   `is_a_daddy(X):- father(X,_Y)` dan
   `is_a_mommy(X):- mother(X,_Y).`
   Pendefinisian dapat pula dilakukan sebagai:

Solusi:

1. $\text{Is-a-Daddy}(x) := \exists y \text{Father}\,(x, y)$ dan $\text{Is-a-Mommy}(x) := \exists y \text{Mother}\,(x, y)$.
2. Kita memiliki
   `is_a_daddy(X):- father(X,_Y)` dan
   `is_a_mommy(X):- mother(X,_Y).`
   Pendefinisian dapat pula dilakukan sebagai:
   `is_a_daddy(X):- father(X,Y),(male(Y);female(Y))` dan
   `is_a_mommy(X):- mother(X,Y),(male(Y);female(Y)).`

# Bahasan

# Kesamaan dan Ketaksamaan Term pada Prolog

Misalkan $\texttt{t}_1$ dan $\texttt{t}_2$ adalah dua term pada Prolog, kita dapat memeriksa kesamaan maupun ketaksamaan untuk $\texttt{t}_1$ dan $\texttt{t}_2$ dengan operator == atau = (untuk kesamaan) dan \== atau \= (untuk ketaksamaan).

Misalkan predikat $\mathrm{Sibling}\,(x, y)$ berarti "$x$ adalah saudara dekat (setidaknya memiliki ayah atau ibu yang sama) dari $y$". Kita dapat mendefinisikan predikat ini sebagai berikut:

$\mathrm{Sibling}(x, y)$ berarti

# Kesamaan dan Ketaksamaan Term pada Prolog

Misalkan $\texttt{t}_1$ dan $\texttt{t}_2$ adalah dua term pada Prolog, kita dapat memeriksa kesamaan maupun ketaksamaan untuk $\texttt{t}_1$ dan $\texttt{t}_2$ dengan operator == atau = (untuk kesamaan) dan \== atau \= (untuk ketaksamaan).

Misalkan predikat $\text{Sibling}\,(x, y)$ berarti "$x$ adalah saudara dekat (setidaknya memiliki ayah atau ibu yang sama) dari $y$". Kita dapat mendefinisikan predikat ini sebagai berikut:

$\text{Sibling}(x, y)$ berarti "$x$ adalah saudara dekat dari $y$"
berarti

# Kesamaan dan Ketaksamaan Term pada Prolog

Misalkan $\texttt{t}_1$ dan $\texttt{t}_2$ adalah dua term pada Prolog, kita dapat memeriksa kesamaan maupun ketaksamaan untuk $\texttt{t}_1$ dan $\texttt{t}_2$ dengan operator == atau = (untuk kesamaan) dan \== atau \= (untuk ketaksamaan).

Misalkan predikat $\mathrm{Sibling}\,(x, y)$ berarti "$x$ adalah saudara dekat (setidaknya memiliki ayah atau ibu yang sama) dari $y$". Kita dapat mendefinisikan predikat ini sebagai berikut:

| | | |
|---|---|---|
| $\mathrm{Sibling}(x, y)$ | berarti | "$x$ adalah saudara dekat dari $y$" |
| | berarti | "$x$ dan $y$ adalah orang berbeda yang memiliki orang tua (ayah atau ibu) yang sama" |
| | berarti | |

# Kesamaan dan Ketaksamaan Term pada Prolog

Misalkan $\texttt{t}_1$ dan $\texttt{t}_2$ adalah dua term pada Prolog, kita dapat memeriksa kesamaan maupun ketaksamaan untuk $\texttt{t}_1$ dan $\texttt{t}_2$ dengan operator == atau = (untuk kesamaan) dan \== atau \= (untuk ketaksamaan).

Misalkan predikat $\text{Sibling}\,(x, y)$ berarti "$x$ adalah saudara dekat (setidaknya memiliki ayah atau ibu yang sama) dari $y$". Kita dapat mendefinisikan predikat ini sebagai berikut:

| | | |
|---|---|---|
| $\text{Sibling}(x, y)$ | berarti | "$x$ adalah saudara dekat dari $y$" |
| | berarti | "$x$ dan $y$ adalah orang berbeda yang memiliki orang tua (ayah atau ibu) yang sama" |
| | berarti | **terdapat** $z$ sehingga $\text{Parent}(z, x)$ **dan** $\text{Parent}(z, y)$ **dan** $x \neq y$ |
| | berarti | |

# Kesamaan dan Ketaksamaan Term pada Prolog

Misalkan $\texttt{t}_1$ dan $\texttt{t}_2$ adalah dua term pada Prolog, kita dapat memeriksa kesamaan maupun ketaksamaan untuk $\texttt{t}_1$ dan $\texttt{t}_2$ dengan operator == atau = (untuk kesamaan) dan \== atau \= (untuk ketaksamaan).

Misalkan predikat $\text{Sibling}\,(x, y)$ berarti "$x$ adalah saudara dekat (setidaknya memiliki ayah atau ibu yang sama) dari $y$". Kita dapat mendefinisikan predikat ini sebagai berikut:

| | | |
|---|---|---|
| $\text{Sibling}(x, y)$ | berarti | "$x$ adalah saudara dekat dari $y$" |
| | berarti | "$x$ dan $y$ adalah orang berbeda yang memiliki orang tua (ayah atau ibu) yang sama" |
| | berarti | **terdapat** $z$ sehingga $\text{Parent}(z, x)$ **dan** $\text{Parent}(z, y)$ **dan** $x \neq y$ |
| | berarti | $\exists z\,(\text{Parent}\,(x, z) \wedge \text{Parent}\,(z, y)) \wedge (x \neq y)$. |

Dalam formula logika predikat, kita memiliki $\text{Sibling}(x, y) := \exists z\,(\text{Parent}\,(z, x) \wedge \text{Parent}\,(z, y)) \wedge (x \neq y)$, sehingga kita memiliki $\forall x \forall y\,(\exists z\,(\text{Parent}\,(z, x) \wedge \text{Parent}\,(z, y)) \wedge (x \neq y) \rightarrow \text{Sibling}\,(x, y))$.

Dalam Prolog, formula logika predikat $\forall x \forall y \left(\exists z \left(\text{Parent}\left(z, x\right) \wedge \text{Parent}\left(z, y\right)\right) \wedge \left(x \neq y\right) \rightarrow \text{Sibling}\left(x, y\right)\right)$ dapat direpresentasikan sebagai aturan berikut:

Dalam Prolog, formula logika predikat
$\forall x \forall y \left(\exists z \left(\text{Parent}\left(z, x\right) \wedge \text{Parent}\left(z, y\right)\right) \wedge \left(x \neq y\right) \rightarrow \text{Sibling}\left(x, y\right)\right)$ dapat direpresentasikan sebagai aturan berikut:

```
sibling(X,Y):-
            parent(Z,X), % Z adalah orang tua dari X
            parent(Z,Y), % Z adalah orang tua dari Y
            X \== Y. % X dan Y adalah orang yang berbeda
```

Penulisan di atas merupakan cara penulisan lain dari

```
sibling(X,Y):- parent(Z,X),parent(Z,Y),X \== Y.
```