LAPORAN TUGAS BESAR KECERDASAN BUATAN

*PROLOG*



Oleh:

Muhammad Irfan Triananto Putra (09021281621046)

Muhammad Irsyad Masyhudin (09021181621030)

Jefrie Antonie Wijaya (09021281621061)

Rima Melati (09021181621011)

Najlah Afifah (09021181621035)

**Jurusan Teknik Informatika**

**FAKULTAS ILMU KOMPUTER UNIVERSITAS SRIWIJAYA**

**2018**

DAFTAR ISI

Halaman

HALAMAN JUDUL i

REASONING 1

LOGICAL PROGRAMMING DAN PROLOG 6

STUDI KASUS DAN KESIMPULAN 9

**BAB I**

**REASONING**

Bab ini membahas teknik *reasoning* (penalaran), yaitu teknik penyelesaian masalah dengan cara merepresentasikan masalah ke dalam basis pengetahuan (*knowledge base*) menggunakan logic atau bahasa formal, yaitu bahasa yang dipahami komputer (Suyanto, 2014). Disini, penulis memfokuskan pembahasan pada *first order logic*atau *predicate calculus* (kalkulus predikat)

Berbeda dengan *searching* yang merepresentasikan masalah ke dalam *state* dan ruang masalah serta menggunakan strategi pencarian untuk menemukan solusi, *reasoning* merepresentasikan masalah ke dalam basis pengetahuan dan melakukan proses penalaran untuk menemukan solusi. Pada teknik *searching*, dua masalah utama yang dihadapi adalah: kesulitan dalam menentukan apakah aturan produksi (operator) sudah lengkap atau belum? Jika masalah yang dihadapi cukup kompleks, maka akan terjadi ketidak-efisienan representasi keadaan (*state*).

*Logic* yang paling sederhana adalah *propositional logic* dimana suatu simbol menyatakan satu proposisi (fakta) yang bias bernilai benar atau salah. Simbol proposisi bias dihubungkan dengan *Boolean Connectives* sehingga membentuk suatu kalimat. Tetapi untuk masalah yang kompleks, *propositional logic* tidak bias digunakan secara praktis karena harus membangun banyak fakta untuk merepresentasikan keadaan yang sederhana.

*First-order logic* merepresentasikan fakta dan aturan di dunia nyata dengan cara yang lebih baik yaitu menggunakan objek, predikat (relasi), dan *connectives* serta *quantifier* sehingga beberapa fakta sederhana dapat direpresentasikan ke dalam suatu kalimat logika. Pada *first-order logic* semua relasi bersifat tetap pada waktu kapanpun. Untuk masalah yang kompleks dimana suatu relasi ara berubah sesuai dengan perubahan waktu, maka diperlukan logic bentuk lain yang disebut *temporal logic.* Disini penulis tidak akan membahas panjang perihal jenis *logic* lainnya seperti *proporsitional logic*, *temporal logic* dan *fuzzy logic*.

**1.1 *First-Order Logic* (*Predicate Logic* atau *Predicate Calculus*)**

Terdapat empat komponen penting pada *first-order logic*, yaitu:

* **Objects:** sesuatu dengan identitas individual (*people, houses, colors, …*)
* **Properties:** sifat yang membedakannya dari *object* yang lain (*red, circle, …*)
* **Relations:** hubungan antar *object* (*brother of, bigger than, …*)
* **Functions:** relasi yang mempunyai satu nilai (*father of, mother of, …*)

**1.1.1 Tata bahasa pada *First-Order Logic***

Tidak seperti *proporsitional logic* yang setiap ekspresi merupakan *sentence* atau kalimat (yang merepresentasikan fakta), *first-order logic* menggunakan *sentence* yang merepresentasikan fakta dan *term* yang merepresentasikan objek. Tata bahasanya secara detil sebagai berikut:

* ***Constant*** umumnya dituliskan dalam huruf besar seperti: A, X1, Budi. Pada simbol konstanta ini, setiap simbol harus menyatakan secara spesifik objek yang dimaksud dan harus dilakukan secara hati-hati agar tidak terjadi kerancuan atau ambiguitas.
* ***Variable*** biasanya dituliskan dalam huruf kecil seperti: a,x,s dan sebagainya. Variabel ini menyatakan simbol yang dapat digantikan oleh konstanta apapun.
* ***Predicate*** menyatakan relasi khusus dalam suatu model. Misal: *Berwarna* adalah suatu *predicate* yang memiliki beberapa nilai. Contoh: *Berwarna(Tas,Hitam)*.
* ***Function*** menyatakan relasi yang hanya mempunyai satu nilai. Karena setiap orang hanya punya satu ibu kandung, maka *IbuKandung* adalah suatu *function.*
* ***Terms*** merupakan ekspresi logika yang mengacu pada sebuah objek. *Terms* bias berupa *constant, variable,* atau *function*
* ***Atomic sentences***  dapat dibentuk dari Predicate(Term*, …*). Misalnya *Saudara(Ana, Ani)*
* ***Complex sentences*** merupakan sentence yang dibangun dengan menggunakan *connective*. Contoh: *Teman(Ana,Ani) -> Memberi(Ana,Ani,Permen)*.
* ***Universal quantifiers* (*∀*)** menyatakan sesuatu yang bersifat umum. Simbol ∀ dibaca ‘*For All’* (untuk semua atau untuk setiap). Misal: “untuk setiap objek x, jika x adalah anak kecil maka x suka permen”, maka dapat ditanyatakan dalam *first-order logic* sebagai: *∀x AnakKecil(x) -> Suka(x,Permen)*.
* ***Existential quantifiers* (∃)** menyatakan sesuatu yang berlaku sebagian. Simbol ∃ dibaca “Tehere Exist” (ada satu atau beberapa). Misal: “Ada objek x, jika x adalah anak kecil maka x suka permen” sebagai: *∃x AnakKecil(x) -> Suka(x,Permen)*.
* ***Nested quantifiers*** digunakan untuk menyatakan kalimat kompleks yang menggunakan *quantifiers* ganda. Misal: “Untuk semua x dan semua y, jika x adalah orangtua y maka y adalah anak dari x” sebagai: *∀x,y OrangTua(x,y) -> Anak(y,x)*.

**1.1.2 Inferensi pada *First-Order Logic***

Pada subbab ini, akan disebutkan singkat mengenai aturan inferensi:

1. **Modus Ponens**

1. **And-Elimination**

1. **And-Introduction**

1. **Or-Introduction**

1. **Double-Negation-Elimination**

1. **Unit Resolution**

1. **Resolution**

ekivalen dengan

1. **Universal Elimination**

1. **Existential Elimination**

1. **Existential Introduction**

Pada aturan inferensi di **1.1.2**, aturan *modus ponens* dapat dibaca: “Jika terdapat kalimat da nada kalimat lain yaitu , maka dapat diturunkan sebuah kalimat baru, yaitu ”.

**BAB 2**

***Logical Programming* dan PROLOG**

*Logical Programming* adalah suatu paradigma bahasa pemrograman yang memandang pernyataan logis sebagai suatu program. Bahasa pemrograman logis yang penulis akan pakai adalah **PROLOG** (*Programming in Logic)*. Didalam PROLOG, suatu program dituliskan sebagai kumpulan kalimat dalam bentuk formal atau *Horn Clause*. Perbedaan representasi antara *First-Order Logic* dan PROLOG:

**Representasi dalam *First-Order Logic*:**

Menikah(Wati,Andi)

∀x,y SaudaraKembar(x,y) -> SaudaraKandung(x,y)

∀x,y,z AnakKandung(x,y) ∧ SaudaraKandung(y,z) -> Keponakan(x,z)

**Representasi dalam PROLOG:**

Menikah(wait,andi).

SaudaraKandung(X,Y) :- SaudaraKembar(X,Y).

Keponakan(X,Z) :- AnakKandung(X,Y), SuadaraKandung(Y,Z).

Perbedaan sintaks antara representasi *First-Order Logic* (FOL) dan representasi PROLOG adalah sebagai berikut:

* Pada FOL, *quantifier* dinyatakan secara eksplisit. Sedangkan pada PROLOG, quantifier dinyatakan secara implisit oleh variable-variabel yang diinterpretasikan.Perbedaan lainnya, penulisan program di PROLOG, variable ditulis dengan diawali huruf besar, sedangkan konstanta diawali huruf kecil.
* Simbol ∧ pada FOL ditulisan sebagai ‘,’ pada PROLOG. Sedangkan simbol ∨ tidak digantikan oleh simbol apapun
* Simbol -> pada FOLD dituliskan sebagai ‘:-‘ pada PROLOG, tetapi dengan cara terbalik. Sehingga kalimat ‘p->q’ ditulisan sebagai q:-p. Sehingga PROLOG menggunakan *backward chaining* dalam proses *reasoning*

**2.1 PROLOG**

PROLOG adalah bahasa pemrograman logika yang umumnya berhubungan dengan *artificial intelligence* dan *computational linguistic*. Komponen utama dari PROLOG terdiri dari:

* Facts yaitu ekspresi predikat yang membuat *declarative statement* (pernyataan deklaratif) tentang suatu masalah. Misal: *cowok(Budi), ayah(Budi,Ani).*
* Rules yaitu ekspresi predikat yang menggunakan implikasi logis ‘:-‘ untuk menggambarkan hubungan diantara fakta-fakta. Misal: *benci(X,Y):- not(suka(X,Y)).*
* Queries yaitu suatu kalimat yang berisikan fakta dan/atau aturan yang akan di *interpret* oleh PROLOG. Jika PROLOG dapat membuktikan bahwa *query* yang diberikan benar, maka *output* yang keluar adalah ‘True’ dan menampilkan variable yang terkait dengan jawab tersebut. Sebaliknya jika tidak maka *output*-nya ‘False’. Misal:

*Dictionary / Database*:

Cowok(budi).

Cowok(ares).

Cewek(ani).

*Query*:

Cowok(X).

*Output*:

X = budi;

X = ares

**BAB 3**

**STUDI KASUS DAN KESIMPULAN**

Pada bab ini, penulis mengambil 2 studi kasus, yaitu “*Hukum Pernikahan”* dan “*Saya adalah Kakek dari diri saya sendiri”*. Pengambilan studi kasus ini didasarkan pada materi yang diajarkan dan soal ujian tengah semester sebelumnya. Disini penulis langsung melakukan konversi soal yang diberikan dalam bentuk PROLOG.

**3.1 Studi Kasus Hukum Pernikahan**

**Soal:**

Hukum pernikahan menyatakan bahwa suatu pernikahan adalah tidak sah jika kedua mempelai memiliki hubungan keponakan. Wati menikah dengan Andi. Dimana Wati adalah anak kandung Budi yang merupakan saudara kembar Andi. Buktikan bahwa pernikahan Andi dan Wati adalah tidak sah.

**Konversi ke PROLOG**:

sah(menikah(X,Y)) :- keponakan(X,Y),menikah(X,Y),!,fail.

menikah(wati,andi).

anak\_kandung(wati,budi).

saudara\_kembar(budi,andi).

saudara\_kandung(X,Y) :- saudara\_kembar(X,Y).

keponakan(X,Z) :- anak\_kandung(X,Y),saudara\_kandung(Y,Z).

**Query yang diberikan:**

* ?- sah(menikah(wati,andi).

false.

* ?- saudara\_kembar(X,Y).

X = budi,

Y = andi.

**3.2 Studi Kasus *“Saya adalah Kakek dari diri saya sendiri”***

**Soal:**

I married a widow (call her W) who has a grown-up daughter (D).My father (F), who visited us quite often, fell in love with my step-daughter and married her. Hence, my father became my son-in-law and my step-daughter became my mother. Some months later, my wife gave birth to a son (S1), who becomes the brother-in-law of my father, as well as my uncle. The wife of my father, that is, my step-daughter, also had a son (S2).

**Konversi ke PROLOG:**

% menikah(suami,istri).

menikah(i,w).

menikah(f,d).

% ortu\_asli(ayah/ibu, anak).

ortu\_asli(f,i).

ortu\_asli(w,d).

ortu\_asli(w,s1).

ortu\_asli(i,s1).

ortu\_asli(d,s2).

ortu\_asli(f,s2).

ayah\_tiri(X,Y) :- menikah(X,Z),ortu\_asli(Z,Y), \+ortu\_asli(X,Y).

ibu\_tiri(X,Y) :- menikah(Z,X), ortu\_asli(Z,Y), \+ortu\_asli(X,Y).

ortu\_tiri(X,Y) :- ayah\_tiri(X,Y).

ortu\_tiri(X,Y) :- ibu\_tiri(X,Y).

ortu(X,Y) :- ortu\_tiri(X,Y).

ortu(X,Y) :- ortu\_asli(X,Y).

istri(X,Y) :- menikah(Y,X).

suami(X,Y) :- menikah(X,Y).

eyang(X,Y) :-

ortu(X,Z),

ortu(Z,Y).

saudara(X,Y) :-

ortu(Z,X),

ortu(Z,Y),

X \= Y.

menantu(f,i).

**Query yang diberikan:**

* eyang(i,i).

?- true ;

* istri(X,Y).

X = w,

Y = i ;

X = d,

Y = f.

* ortu\_asli(X,Y).

X = f,

Y = i ;

X = w,

Y = d ;

X = w,

Y = s1 ;

X = i,

Y = s1 ;

X = d,

Y = s2 ;

X = f,

Y = s2.

**3.3 Kesimpulan**

Menarik kesimpulan logika menggunakan PROLOG perlu menggunakan aturan-aturan maupun fakta tambahan pada *database*-nya, dikarenakan akan terjadi *infinite loop* jika ada kerancuan, seperti pada kasus **3.2** dimana terdapat hubungan keluarga yang rumit. PROLOG juga menggunakan notasi *Horn clause* dimana bagian kiri dari suatu kalimat adalah kesimpulan dan harus *single positive literal*. Misal pada kasus **3.1**: *“tidak sah menikah jika kedua mempelai memiliki hubungan keponakan”*. Kesimpulan yang ditarik adalah “*tidak sah”*, tetapi pada sintaks di PROLOG tidak diperbolehkan menggunakan negasi pada kesimpulan.

Sintaks yang salah:

not(sah(menikah(X,Y))) :- keponakan(X,Y),menikah(X,Y)

Sintaks yang diperbolehkan:

sah(menikah(X,Y)) :- keponakan(X,Y),menikah(X,Y),!,fail.