PENGGUNAAN DOMAIN DAN QUERY DALAM PROLOG

Pada bagian ini akan dibahas hal-hal berikut :

1. Penggunaan Domain
2. Pengertian Program Deklaratif Murni dan Tidak Murni
3. Penggunaan Query

1. PENGGUNAAN DOMAIN

Dalam Kalkulus Predikat, domain digunakan untuk membatasi nilai parameter dari predikat. Karena, validitas/keabsahan nilai kalimat predikat hanya dapat ditentukan untuk domain yang terbatas.

Pada program Prolog juga serupa. Dengan adanya domain, maka kita dapat:

1. mendeklarasikan tipe data yang digunakan dalam program, untuk menghindari enumerasi fakta - semisal integer (kasus pertama), atau
2. membatasi nilai parameter simbolik - analogi dengan subtype (kasus kedua). Dilakukan dengan menggunakan bantuan *functor* tanpa parameter.

Contoh penggunaan domain untuk **kasus pertama**.
Program LIGHT MEAL (versi-1) adalah contoh program tanpa domain.

LIGHT MEAL (versi-2)

FAKTA

{ LittleSucc(X,Y) benar, artinya Y adalah little successor dari Y, dengan Y <10 , dan $1 \leq X \leq 8$}

```
littleSucc(1,2).
littleSucc(2,3).
littleSucc(3,4).
littleSucc(4,5).
littleSucc(5,6).
littleSucc(6,7).
littleSucc(7,8).
littleSucc(8,9).
```

ATURAN

{LittleSum(X,Y,Z) benar, artinya Z adalah "little sum", yaitu jumlah dari X+Y, dan Z ebih kecil dari 10}

```
littleSum(1,X,Y)    ←   littleSucc(X,Y).
littleSum(X',Y,Z') ←   littlleSucc(X,X'),
                       littleSum(X,Y,Z),
                       littleSucc(Z,Z').
```

Program LIGHT MEAL (versi-2) adalah contoh program dengan domain.

LIGHT MEAL (versi-2)

DOMAIN

```
i : integer
```

DEKLARASI PREDIKAT

```
littleSum(i,i,i)
```

FAKTA

{ tidak ada }

ATURAN

{littleSum(X,Y,Z) benar, artinya Z adalah "little sum", yaitu jumlah dari X+Y, dan Z ebih kecil dari 10}

```
littleSum(1,X,Y)    ←   Y = X + 1,
                        Y < 10.
littleSum(X',Y,Z') ←    X = X' - 1,
                        littleSum(X,Y,Z),
                        Z' = Z + 1,
                        Z' < 10.
```

Catatan Penting:
Dari contoh kasus tersebut diatas, terlihat bahwa penggunaan domain untuk menghindari enumerasi fakta berupa bilangan integer menyebabkan operasi aritmatika yang dilakukan sesuai dengan yang didefinisikan oleh PROLOG. Akibatnya, ada keterbatasan dalam hal proses kalkulasinya, yaitu: **urutan proses (sekuensial) jadi sangat penting.**

Contoh penggunaan domain untuk **kasus kedua**.
Cuplikan Program KELUARGA KECIL (versi-1) adalah contoh program tanpa domain.

KELUARGA KECIL (versi-1)

FAKTA

{ golDarah (X, G) benar jika golongan darah dari X adalah G }

```
golDarah (adi, b).
golDarah (tatik, a).
golDarah (didi, b).
golDarah (tita, ab).
```

Cuplikan Program KELUARGA KECIL (versi-4) adalah contoh program dengan domain.

KELUARGA KECIL (versi-4)

DOMAIN

```
d : a(), b(), ab(), o()
p : symbol
```

DEKLARASI PREDIKAT

```
golDarah (p,d)
```

FAKTA

{ golDarah (X, G) benar jika golongan darah dari X adalah G }

```
golDarah (adi, b).
golDarah (tatik, a).
golDarah (didi, b).
golDarah (tita, ab).
```

Catatan:
Penggunaan lebih lanjut dari functor akan dijelaskan pada materi kuliah selanjutnya.

2. PENGERTIAN PROGRAM DEKLARATIF MURNI DAN TIDAK MURNI

Berikut ini akan dijelaskan secara lebih rinci pengertian dari program deklaratif murni dan tidak murni.

Program Deklaratif Murni:

- Terdiri dari fakta dan aturan dalam bentuk kalkulus predikat orde satu (quantifier hanya menjadi batas dari lingkup variabel terikat, dan bukannya predikat)
- Predikat adalah benar-benar menunjukkan relasi antar objek yang menjadi parameter. Semua parameter adalah **parameter input**
- Pada body dari aturan (ruas kanan tanda $\rightarrow$), karakter koma berarti operator AND secara logika, sehingga urut-urutan evaluasi tidak mempengaruhi hasil query
- Komputasi program adalah murni pencocokan
- Program hanya melakukan manipulasi/pencocokan terhadap **fakta simbolik**
- Cut dan fail tidak ada gunanya (akan dibahas pada bagian ybs.)

Catatan penting:

- sebuah program rekursif mungkin saja menjadi sebuah program deklaratif murni!

Program Deklaratif menjadi tidak "murni" akibat:

- adanya kalkulasi
- transformasi bentuk yang tidak mungkin dinyatakan dalam kalkulus predikat orde satu (seperti pada Plus) akan menjadi ***sequence***. Dalam hal ini tanda koma pada body aturan (di ruas kanan tanda $\rightarrow$) berarti urutan pencocokan aturan dan bukan AND secara logika. Jika urutan penulisan dibalik, hasil program akan "salah"
- adanya parameter yang interpretasinya adalah **parameter output** (walaupun merupakan parameter dari predikat)
- aspek prosedural (yang akan dibahas pada bagian ybs)
- Cut dan fail sangat penting karena mempengaruhi hasil query (akan dibahas pada bagian ybs.)

Interpretasi Hasil Query adalah sbb:

- apakah hasil query terhadap fakta sesuai dengan dunia nyata (misalnya pada contoh keluarga kecil). Kebenaran ini hanya dapat dicek melalui observasi terhadap dunia nyata.
- apakah hasil query sesuai dengan deduksi fakta dan/atau aturan (misalnya hasil diagnosa penyakit suatu sistem pakar). Kebenaran ini harus dicek berdasarkan *knowledge*/pengetahuan yang dimiliki yang dinyatakan dalam bentuk fakta dan/atau aturan.
- apakah hasil query "lengkap". Kelengkapan hanya dapat dicek melalui dunia nyata, domain, atau melalui *constraint*/kendala yang dituliskan dalam spesifikasi.
- apakah hasil query menangkap hal yang tidak lengkap (misalnya pada contoh terhadap warnaRambut, yang seharusnya muncul untuk setiap individu dalam domain); ini dapat dicek melalui constraint.

Jadi, sebaiknya constraint kelengkapan dan relasi antar fakta/aturan juga dituliskan untuk sebuah program deklaratif murni. Contoh berikut adalah untuk contoh keluarga kecil.

KELUARGA KECIL (versi-1a)

FAKTA

{orang(X) benar jika X "orang", yaitu individu dalam domain simbolik X dan Y pada program sbb:}
{ Constraint : Hanya orang-orang yang menjadi fokus basis data yang mewakili dunia nyata yang muncul dalam fakta }

```
orang (adi).
orang (tatik).
orang (didi).
orang (tita).
```

{ayah(X,Y) benar jika X adalah ayah dari Y }
{ Constraint :

- *tidak semua kombinasi antar dua individu orang mempunyai relasi ini .*
- *Hanya subset yang mewakili dunia nyata yang muncul dalam fakta }*

```
ayah (adi, didi).
ayah (adi, tita).
```

{ibu(X,Y) benar jika X adalah ibu dari Y }
{ Constraint :

- *tidak semua kombinasi antar dua individu orang mempunyai relasi ini .*
- *Hanya subset yang mewakili dunia nyata yang muncul dalam fakta }*

```
ibu (tatik, didi).
ibu (tatik, tita).
```

{lebihTua (X,Y) benar jika X lebih tua dari Y }
{ Constraint :

- *tidak semua kombinasi antar dua individu orang mempunyai relasi ini .*
- *Hanya subset yang mewakili dunia nyata yang muncul dalam fakta*

- *Jika Lebihtua (X,Y) muncul sebagai fakta, maka tidak boleh ada lebihTua(X,X) atau LebihTua(Y,X)}*

```
lebihtua (adi,tatik).
lebihtua (adi,didi).
lebihtua (adi,tita).
lebihtua (tatik,didi).
lebihtua (tatik,tita).
lebihtua (adi,tita).
```

{ tinggiBadan (X,N) benar jika tinggi badan dari X adalah N }
{ Constraint :
- *setiap individu orang dalam domain harus mempunyai relasi ini }*

```
tinggiBadan (adi, 170).
tinggiBadan (tatik, 165).
tinggiBadan (didi, 125).
tinggiBadan (tita, 107).
```

{ warnaRambut (X, W) benar jika warna rambut dari X adalah W }
{ Constraint :
- *setiap individu orang dalam domain harus mempunyai relasi ini }*

```
warnaRambut (adi, hitam).
warnaRambut (tatik, hitam).
warnaRambut (didi, hitam).
warnaRambut (tita, hitam).
```

{ golDarah (X, G) benar jika golongan darah dari X adalah G }
{ Constraint :
- *setiap individu orang dalam domain harus mempunyai relasi ini }*

```
golDarah (adi, b).
golDarah (tatik, a).
golDarah (didi, b).
golDarah (tita, ab).
```

ATURAN

{kakak (X,Y) benar jika X adalah saudara Y dan X lebih tua dari Y }

```
kakak (X,Y) ← ayah(X,Z), ayah(Y,Z), ibu(X,W), ibu(Y,W),
                  lebihTua(X,Y).
```

QUERY

{ Siapakah ibu dari tita ? }

```
⇒ ibu(X,tita)?
```

{ Siapakah ayah tita ? }

```
⇒ ayah(X,tita)?
```

{Buatlah daftar orang yang mempunyai hubungan kakak-adik }

```
⇒ kakak(X,Y)?
```

{Buatlah daftar orang yang ayahnya adalah adi }

```
⇒ ayah(adi,X)?
```

3. PENGGUNAAN QUERY

Query dapat digunakan untuk:

1. memperoleh arti dari program (*logic*)
2. mengetahui kebenaran program (*correct & complete)*
3. memperoleh informasi dari program

Selain itu, yang juga harus diperhatikan bahwa query dari sebuah program dapat berupa query yang benar ataupun salah. Dalam arti, bahwa query yang benar merepresentasikan arti dari program, sedangkan query yang salah tidak merepresentasikan arti dari program (atau dengan kata lain, merepresentasikan arti yang salah dari program).

Untuk itu, perlu diketahui bagaimana cara membentuk query yang benar. Prinsip dasarnya:

1. query dari program deklaratif murni adalah benar, jika:
   - query merupakan konsekuensi lojik dari program logika,
   - query tersebut mempunyai parameter yang dapat merepresentasikan *input*/masukan dan/atau *output*/keluaran, tergantung nilai parameternya (konstanta atau variabel); sesuai dengan spesifikasi fakta/aturannya.
   - untuk conjunctive query, urutan query tidak menentukan hasil perolehan.
2. query dari program deklaratif tidak murni adalah benar, jika
   - query merupakan konsekuensi lojik dari program logika,
   - query tersebut mempunyai parameter yang merepresentasikan *input*/masukan bernilai konstanta dan parameter yang merepresentasikan *output*/keluaran berupa variabel; sesuai dengan spesifikasi fakta/aturannya.
   - untuk conjunctive query, urutan query dapat menentukan hasil perolehan.

Contoh Program Deklaratif Murni: Program Keluarga Kecil.

KELUARGA KECIL (versi-1)

FAKTA

{ayah(X,Y) benar jika X adalah ayah dari Y }

```
ayah (adi, didi).
ayah (adi, tita).
```

{ibu(X,Y) benar jika X adalah ibu dari Y }

```
ibu (tatik, didi).
ibu (tatik, tita).
```

{lebihTua (X,Y) benar jika X lebih tua dari Y }

```
lebihtua (adi,tatik).
lebihtua (adi,didi).
lebihtua (adi,tita).
lebihtua (tatik,didi).
lebihtua (tatik,tita).
lebihtua (adi,tita).
```

{ tinggiBadan (X,N) benar jika tinggi badan dari X adalah N }

```
tinggiBadan (adi, 170).
tinggiBadan (tatik, 165).
tinggiBadan (didi, 125).
tinggiBadan (tita, 107).
```

{ warnaRambut (X, W) benar jika warna rambut dari X adalah W }

```
warnaRambut (adi, hitam).
warnaRambut (tatik, hitam).
warnaRambut (didi, hitam).
warnaRambut (tita, hitam).
```

{ golDarah (X, G) benar jika golongan darah dari X adalah G }

```
golDarah (adi, b).
golDarah (tatik, a).
golDarah (didi, b).
golDarah (tita, ab).
```

ATURAN

{Kakak (X,Y) benar, jika X adalah saudara Y dan X lebih tua dari Y }

```
Kakak (X,Y) ← Ayah(Z,X), Ayah(Z,Y), Ibu(W,X), Ibu(W,Y),
                  LebihTua(X,Y).
```

Contoh: Hasil Query yang benar dan salah dari Program Pohon Keluarga Kecil versi 1.

QUERY

1. *{ Benarkah tatik ibu dari tita ? }*
   ⇒ `ibu(tatik,tita)?`
2. *{ Siapakah ayah tita ? }*
   ⇒ `ayah(X,tita)?`
3. *{Buatlah daftar orang yang ayahnya adalah adi }*
   ⇒ `ayah(adi,X)?`
4. *{Buatlah daftar orang yang mempunyai hubungan kakak-adik }*
   ⇒ `kakak(X,Y)?`
5. *{Apakah didi lebih tua dari tatik ? }*
   ⇒ `lebihTua(didi,tita)?`
6. *{Berikanlah daftar anggota keluarga kecil (orang) yg rambutnya hitam dan golongan darahnya A }*
   ⇒ `warnaRambut(X,hitam), golDarah(X,a)?`
   `atau`
   ⇒ `golDarah(X,a), warnaRambut(X,hitam)?`
7. *{ Siapakah ibu dari tatik ? }*
   ⇒ `ibu(X,tatik)?`
8. *{ Tita ayah siapa ? }*
   ⇒ `ayah(tita,X)?`

Penjelasan dari hasil query tersebut diatas:

Nomor Query	Benar /Salah	Masukan	Keluaran	Keterangan
1	benar	tatik, tita	-	
2	benar	tita	X	
3	benar	adi	X	
4	benar	-	X, Y	
5	benar	didi, tita	-	
6	benar	hitam, a	X	
7	salah	tatik	X	Query bukan konsekuensi lojik dari program, lebih lanjut dapat disimpulkan bahwa program tidak lengkap secara logika. Tetapi dari aspek constraint: kesalahan muncul karena yang ditanyakan sudah diluar domain.
8	salah	tita	X	Query bukan konsekuensi lojik dari program, lebih lanjut dapat disimpulkan bahwa query tidak sesuai dengan spesifikasinya.