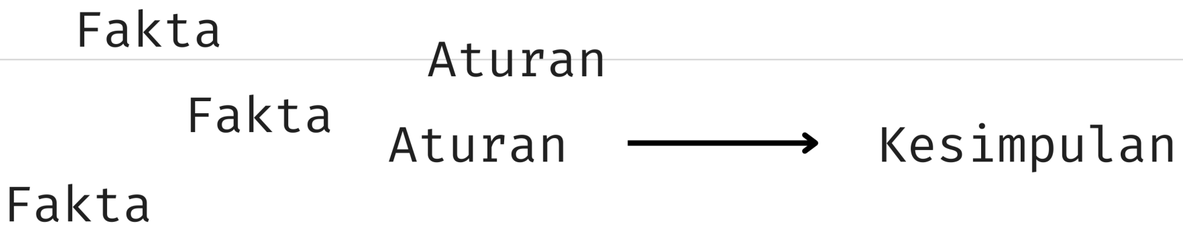
UNIVERSITAS ATMA JAYA YOGYAKARTA

21

VISUAL PROLOG
EZ to UNDERSTAND

Semoga...

Asisten Dosen Pengantar Kecerdasan Buatan
Semester Gasal 2022/2023



PL

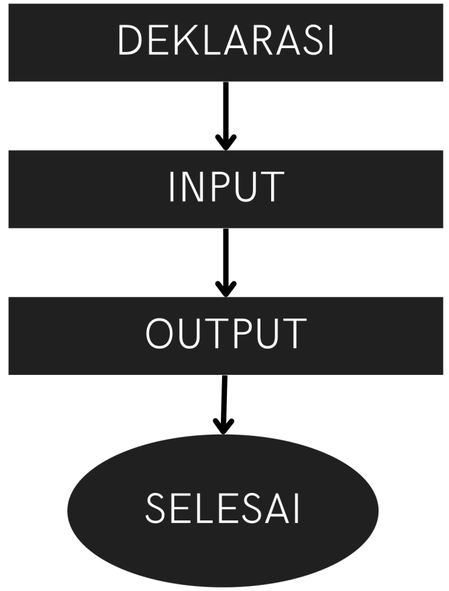
PROgramming in LOGic

Penarikan kesimpulan dari sekumpulan fakta atau aturan

Sekumpulan Aksi berurutan

Bahasa pemprograman umumnya dijelaskan sebagai sekumpulan aksi yang dijalankan secara sekuensial.

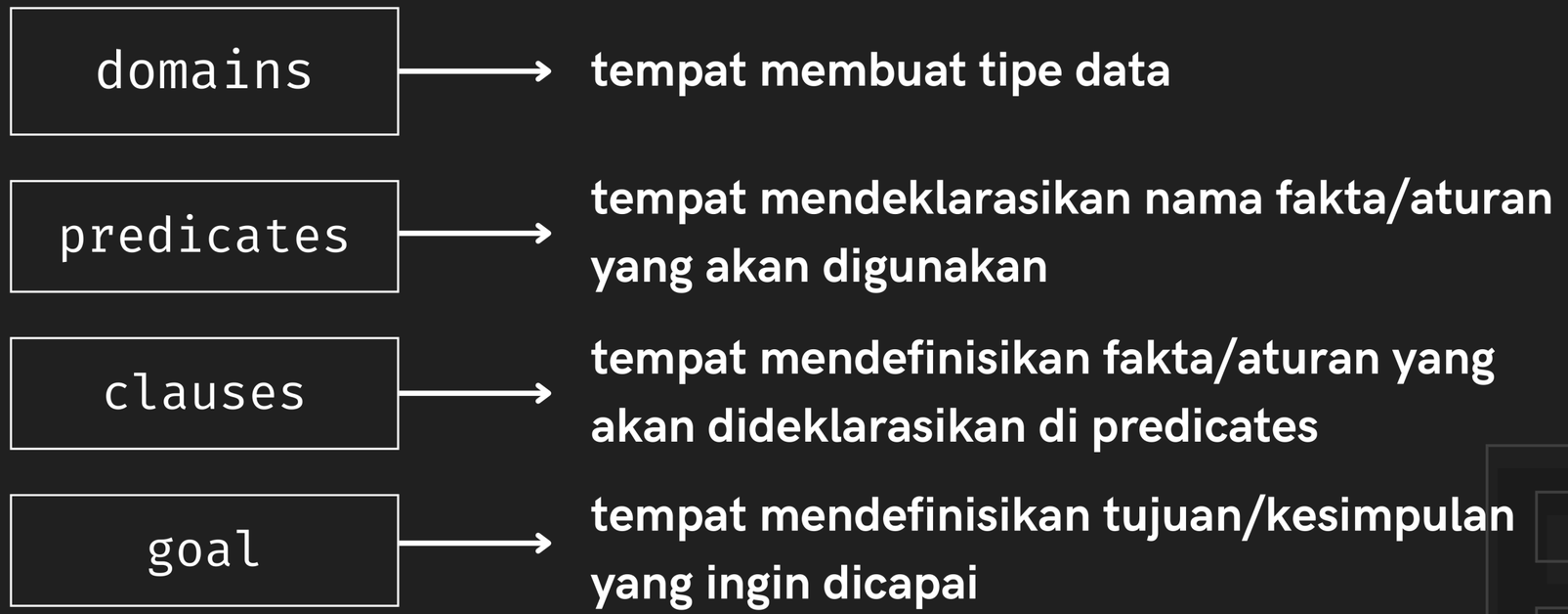
```
program INPUT_CETAK

deklarasi
var masukan: string

algoritma
INPUT masukan
OUTPUT masukan
```



PROLOG memiliki cara kerja yang hampir sama, tetapi dengan sedikit perbedaan

LOG

PROLOG secara umum dibagi menjadi 4 segment



PRO

domains

Dalam menyimpan data, kita perlu mengategorisasikan jenis datanya untuk mempermudah pengelolaan dan kinerja komputer.

Tipe data dasar

ini adalah beberapa tipe dasar yang digunakan

`string`	**menyimpan karakter huruf/angka dsbnya, ditandai dengan ""**
`symbol`	**mirip dengan string, tetapi memiliki kinerja penyimpanan dan pencarian berbeda**
`integer`	**menyimpan bilangan bulat**

01

*Detail symbol tidak perlu dipelajari lebih dalam karena kurang dipergunakan

predicates

Format fakta/aturan

```
nondeterm namaFakta(arg1, arg2,...,argN)
```

Nama dari fakta/aturan ingin dibuat.

argument-argument yang dibutuhkan, berupa tipe datanya

Contoh

```
nondeterm is_my(string, string)
```

*Pengertian nondeterm tidak perlu dipelajari lebih lanjut

*arg -> Argument atau parameter

kembali lagi

domains

```
predicates
    nondeterm is_my(string, string)
```

Agak sulit dibaca untuk memahami predicates yang kita buat. Kita dapat memberikan alias atau memberi nama baru ke tipe data dasar untuk mempermudah memahami code.

```
domains
    person = string
    relation = string
predicates
    nondeterm is_my(person, relation)
```

Lebih mudah dibaca kan?

02

Tipe data person tidak sama dengan relation,
meskipun sama sama alias dari string

Fakta

clauses

Setelah mendeklarasikan di predicates, kita bisa memberikan definisi kepada aturan atau mendeklarasikan fakta-fakta yang ada.

Pendeklarasian Fakta

```
clauses
    is_my("Budi", "Ayah").
    is_my("Dudu", "Kakek").
```

Disini kita memberikan beberapa fakta:

- Budi is my Ayah
- Dudu is my Kakek

*perhatikan tanda titik diakhir

Aturan

clauses

Pendefinisian Aturan

```
domains
    person = string
    relation = string
predicates
    nondeterm is_my(person, relation)
    nondeterm is_myFather(person)
clauses
    is_my("Budi", "Ayah").
    is_my("Dudu", "Kakek").
    is_myFather(Nama):-
         is_my(Nama, Relation),
         Relation = "Ayah".
```

Disini kita mendefinisikan aturan **Nama** is_myFather jika

- Berdasarkan fakta is_my yang ada, dicari yang mengandung arg person = Nama dan arg relation disimpan di variabel Relation, dan
- Relation adalah "Ayah".

*Semua yang diawali dengan huruf Kapital adalah Variabel

*IF PROLOG sedikit berbeda dengan bahasa pemprograman pada umumnya

goal

Kesimpulan apa yang ingin kita capai?

Misal kita ingin tau siapa ayahku, maka kita bisa buat

```
goal
    is_myFather(Ayah).
```

Misal kita ingin tau apakah Budi adalah ayahku, maka kita bisa buat

```
goal
    is_myFather("Budi").
```

*Silahkan coba di vip dan perhatikan perbedaan keluarannya

goal

```
goal
    is_myFather(Ayah).
```


output

```
Ayah=Budi
1 Solution
```

goal

```
goal
    is_myFather("Budi").
```

output

```
yes
```

goal

```
goal
    is_myFather(Ayah).
```

output

```
Ayah=Budi
1 Solution
```

Ketika kita memberikan sebuah variabel di goal, maka PROLOG akan mentrace ke clauses untuk mencari segala kemungkinan nilai Variabel tersebut

goal

```
goal
    is_myFather("Budi").
```

output

yes

Ketika kita tidak ada membuat variabel di goal, maka PROLOG akan mencoba menjalankan statement yang kita berikan merupakan BENAR atau SALAH

Mari baca alur Programnya

RUN

```
goal
 is_myFather("Budi").
```

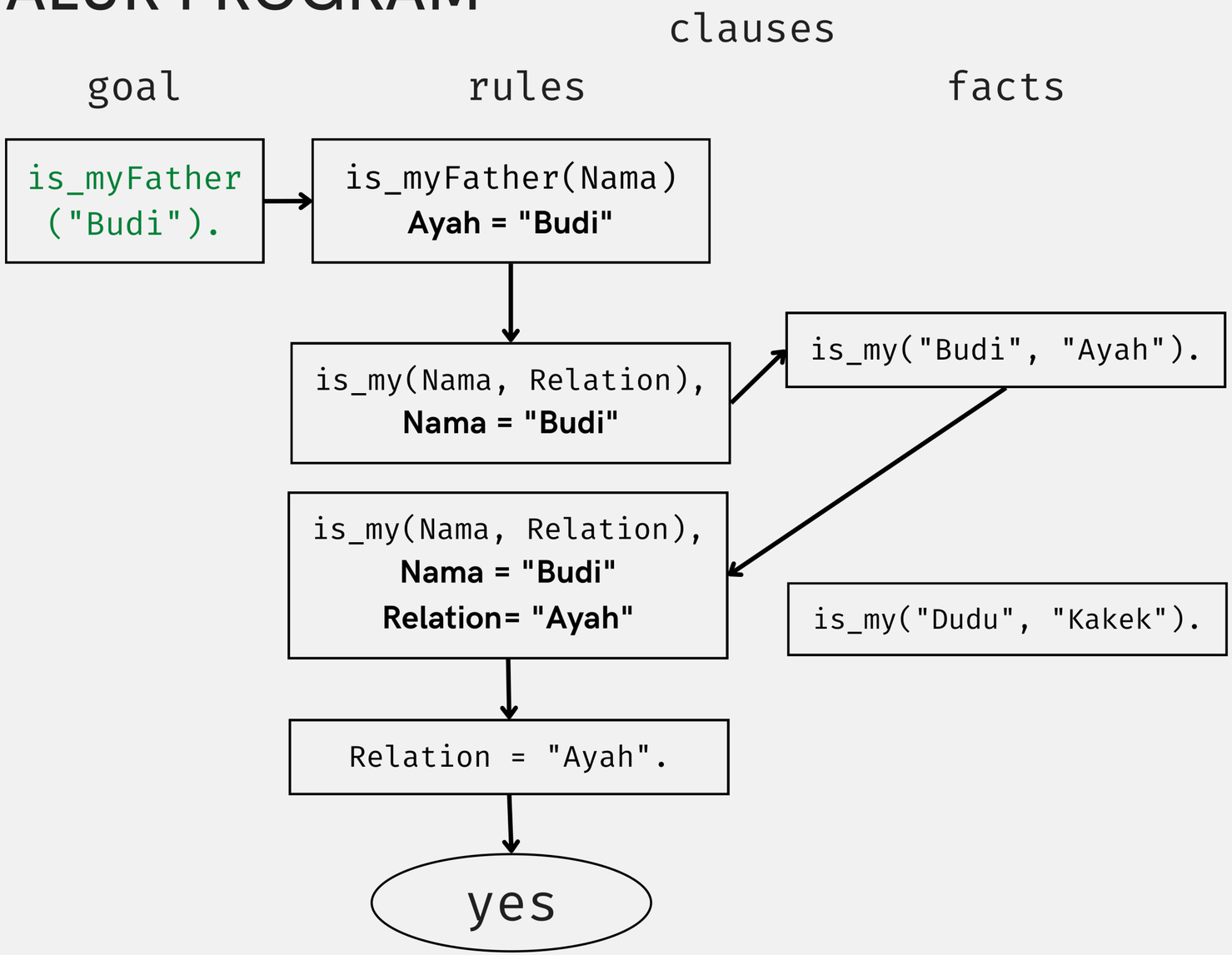
→

PROLOG

```
domains
    person = string
    relation = string
predicates
    nondeterm is_my(person, relation)
    nondeterm is_myFather(person)
clauses
    is_my("Budi", "Ayah").
    is_my("Dudu", "Kakek").
    is_myFather(Nama):-
         is_my(Nama, Relation),
         Relation = "Ayah".
goal
    is_myFather("Budi").
```

RUN

ALUR PROGRAM

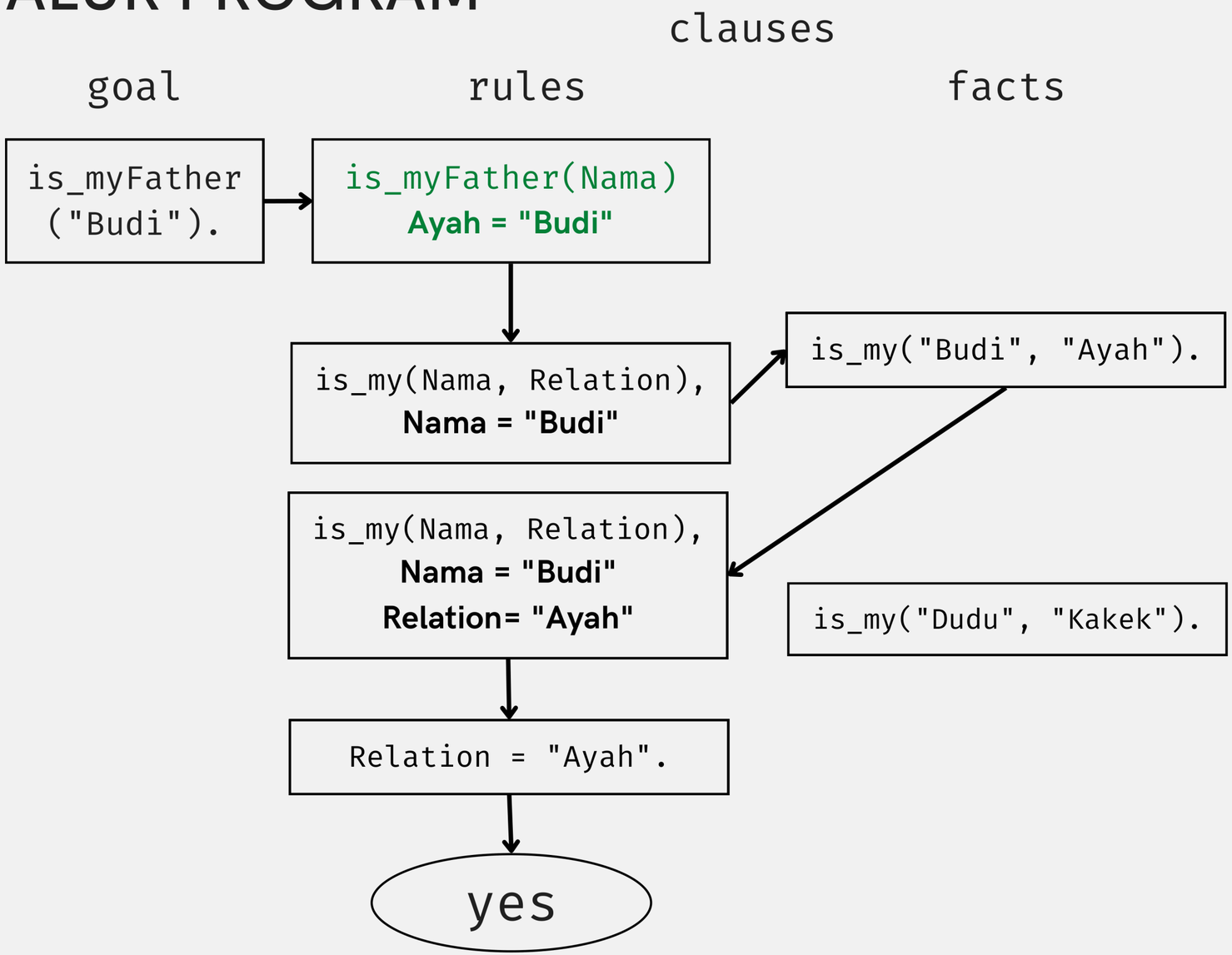


PROLOG

```
domains
    person = string
    relation = string
predicates
    nondeterm is_my(person, relation)
    nondeterm is_myFather(person)
clauses
    is_my("Budi", "Ayah").
    is_my("Dudu", "Kakek").
    is_myFather(Nama):-
          is_my(Nama, Relation),
          Relation = "Ayah".
goal
    is_myFather("Budi").
```

RUN

ALUR PROGRAM

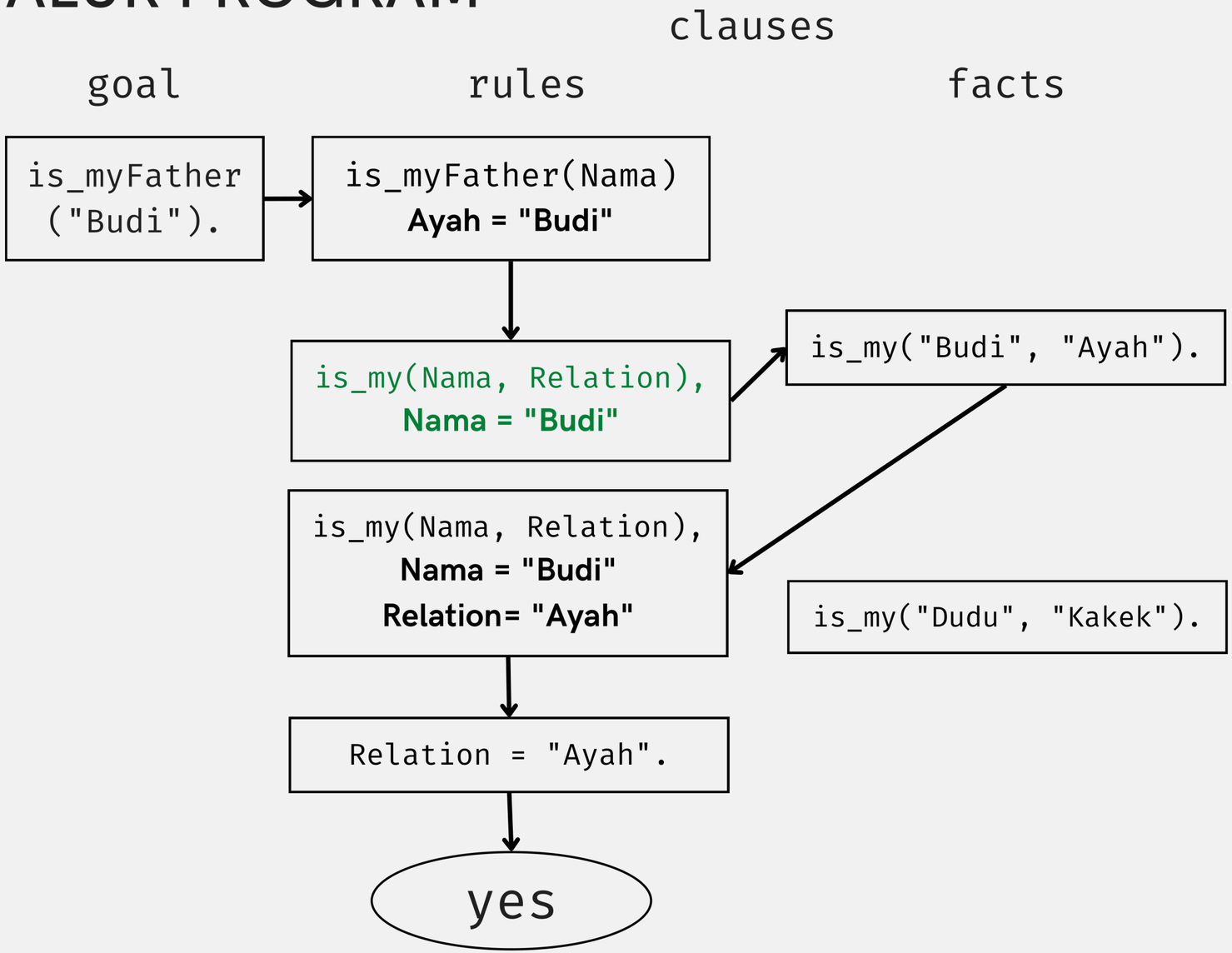


PROLOG

```
domains
    person = string
    relation = string
predicates
    nondeterm is_my(person, relation)
    nondeterm is_myFather(person)
clauses
    is_my("Budi", "Ayah").
    is_my("Dudu", "Kakek").
    is_myFather(Nama):-
          is_my(Nama, Relation),
          Relation = "Ayah".
goal
    is_myFather("Budi").
```

RUN

ALUR PROGRAM

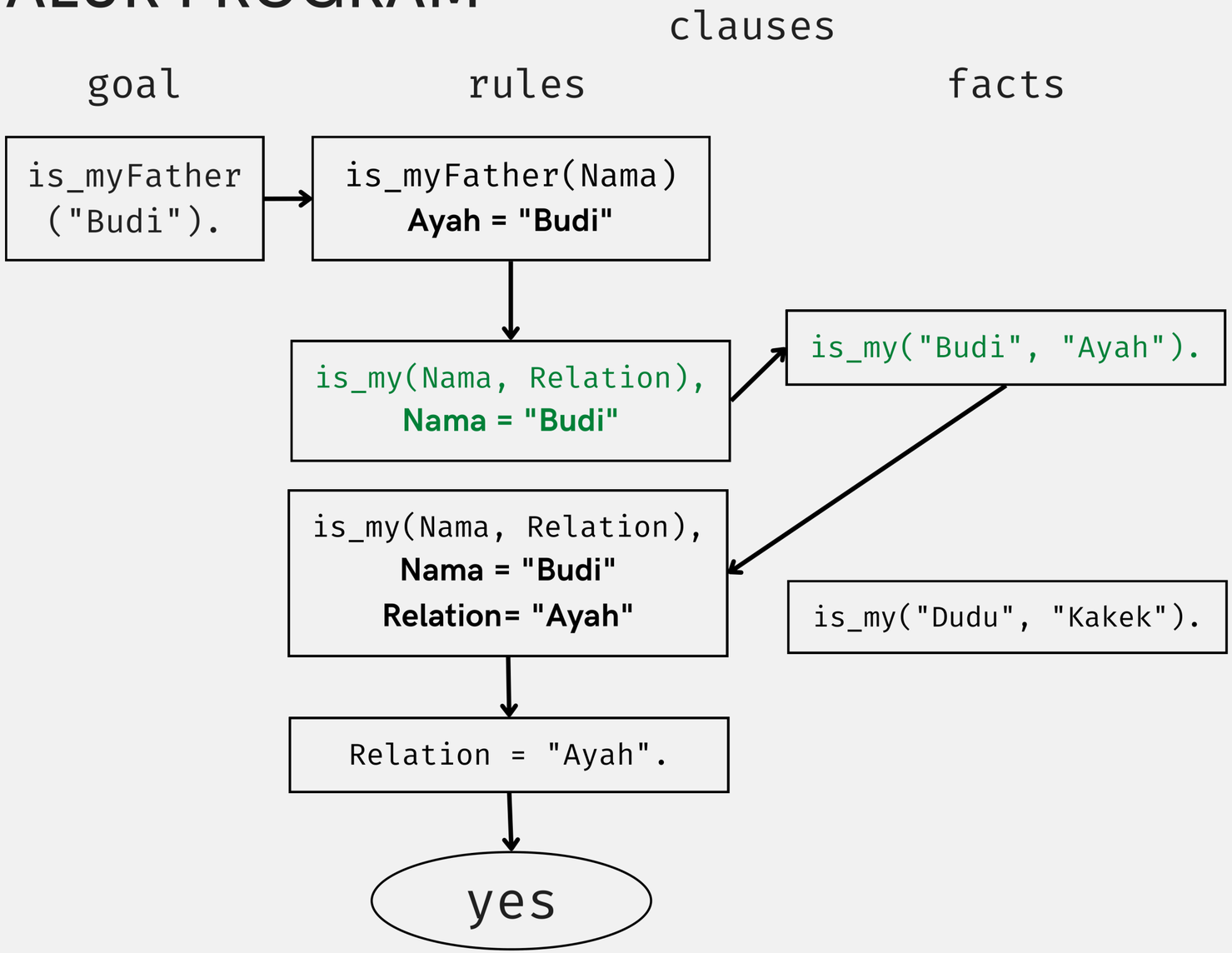


PROLOG

```
domains
    person = string
    relation = string
predicates
    nondeterm is_my(person, relation)
    nondeterm is_myFather(person)
clauses
    is_my("Budi", "Ayah").
    is_my("Dudu", "Kakek").
    is_myFather(Nama):-
          is_my(Nama, Relation),
          Relation = "Ayah".
goal
    is_myFather("Budi").
```

RUN

ALUR PROGRAM

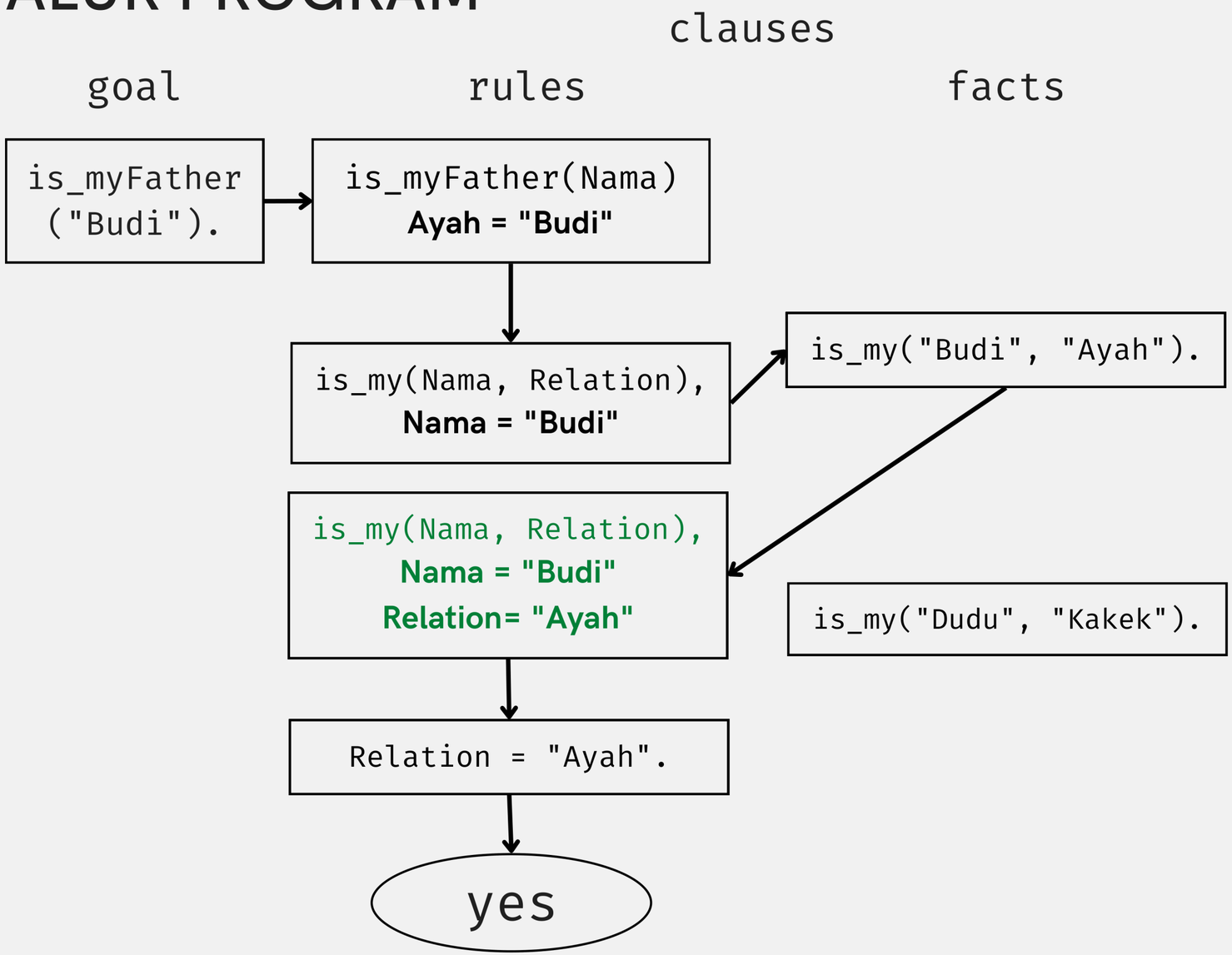


PROLOG

```
domains
    person = string
    relation = string
predicates
    nondeterm is_my(person, relation)
    nondeterm is_myFather(person)
clauses
    is_my("Budi", "Ayah").
    is_my("Dudu", "Kakek").
    is_myFather(Nama):-
          is_my(Nama, Relation),
          Relation = "Ayah".
goal
    is_myFather("Budi").
```

RUN

ALUR PROGRAM

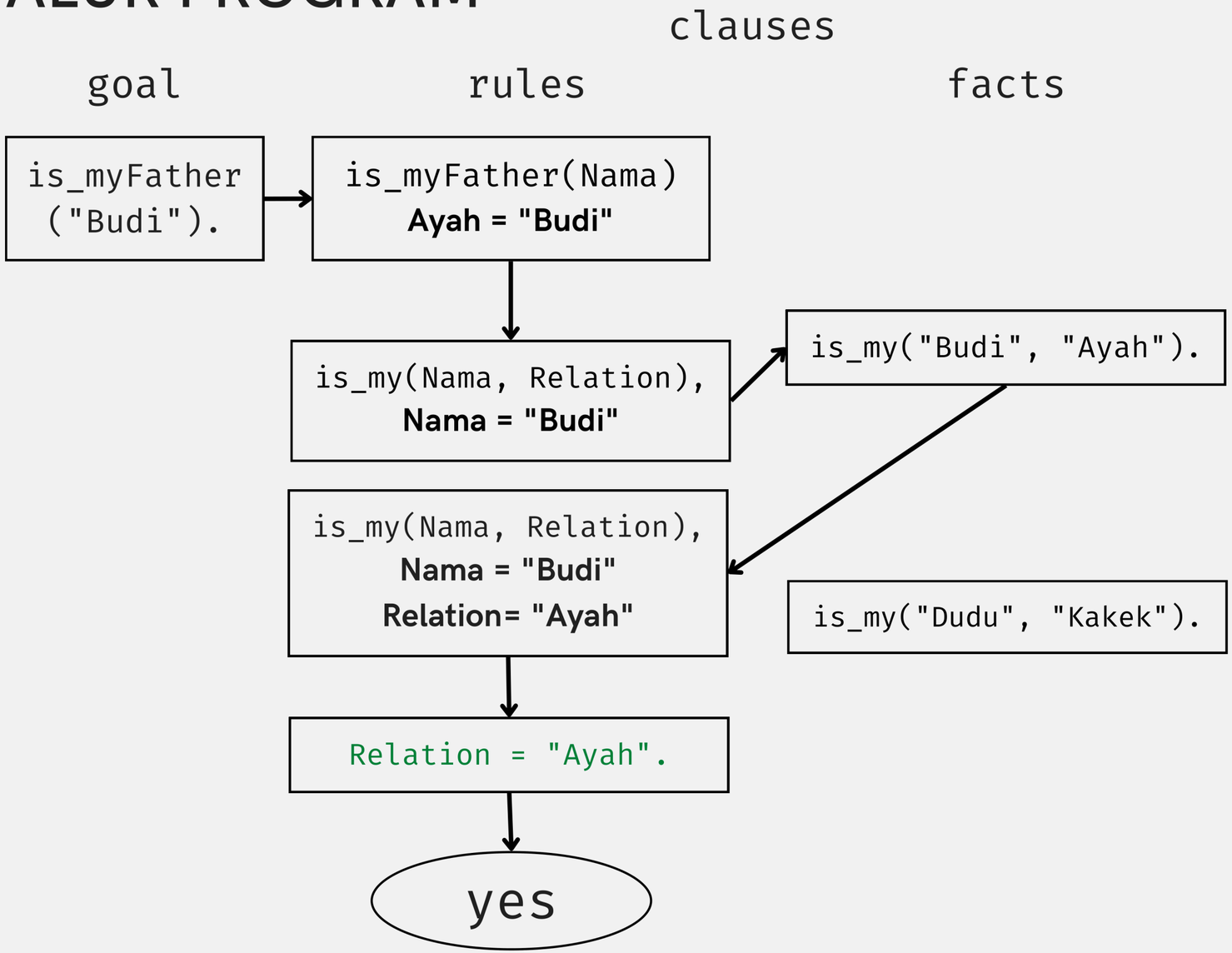


PROLOG

```
domains
    person = string
    relation = string
predicates
    nondeterm is_my(person, relation)
    nondeterm is_myFather(person)
clauses
    is_my("Budi", "Ayah").
    is_my("Dudu", "Kakek").
    is_myFather(Nama):-
         is_my(Nama, Relation),
         Relation = "Ayah".
goal
    is_myFather("Budi").
```

RUN

ALUR PROGRAM

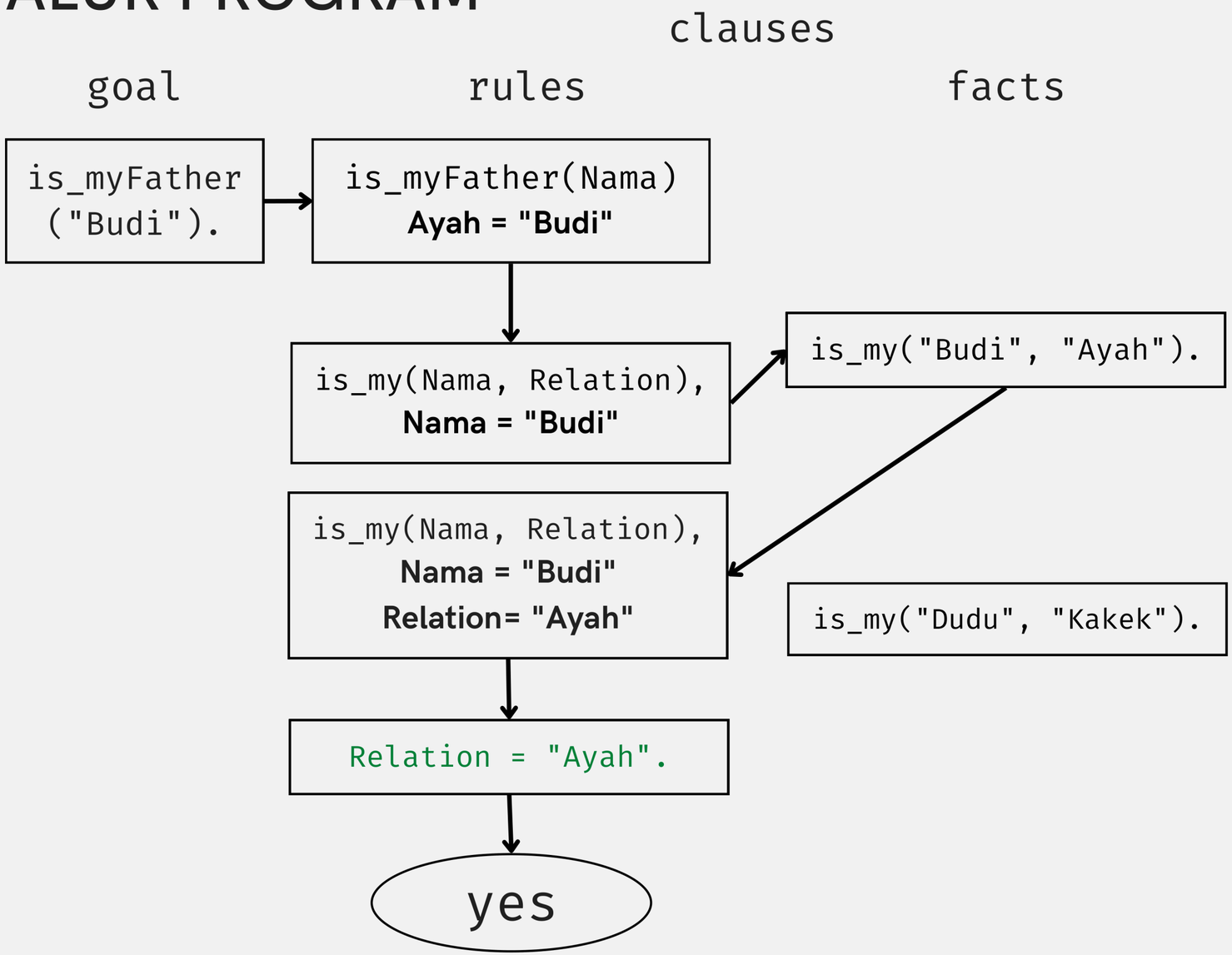


PROLOG

```
domains
    person = string
    relation = string
predicates
    nondeterm is_my(person, relation)
    nondeterm is_myFather(person)
clauses
    is_my("Budi", "Ayah").
    is_my("Dudu", "Kakek").
    is_myFather(Nama):-
         is_my(Nama, Relation),
         Relation = "Ayah".
goal
    is_myFather("Budi").
```

RUN

ALUR PROGRAM

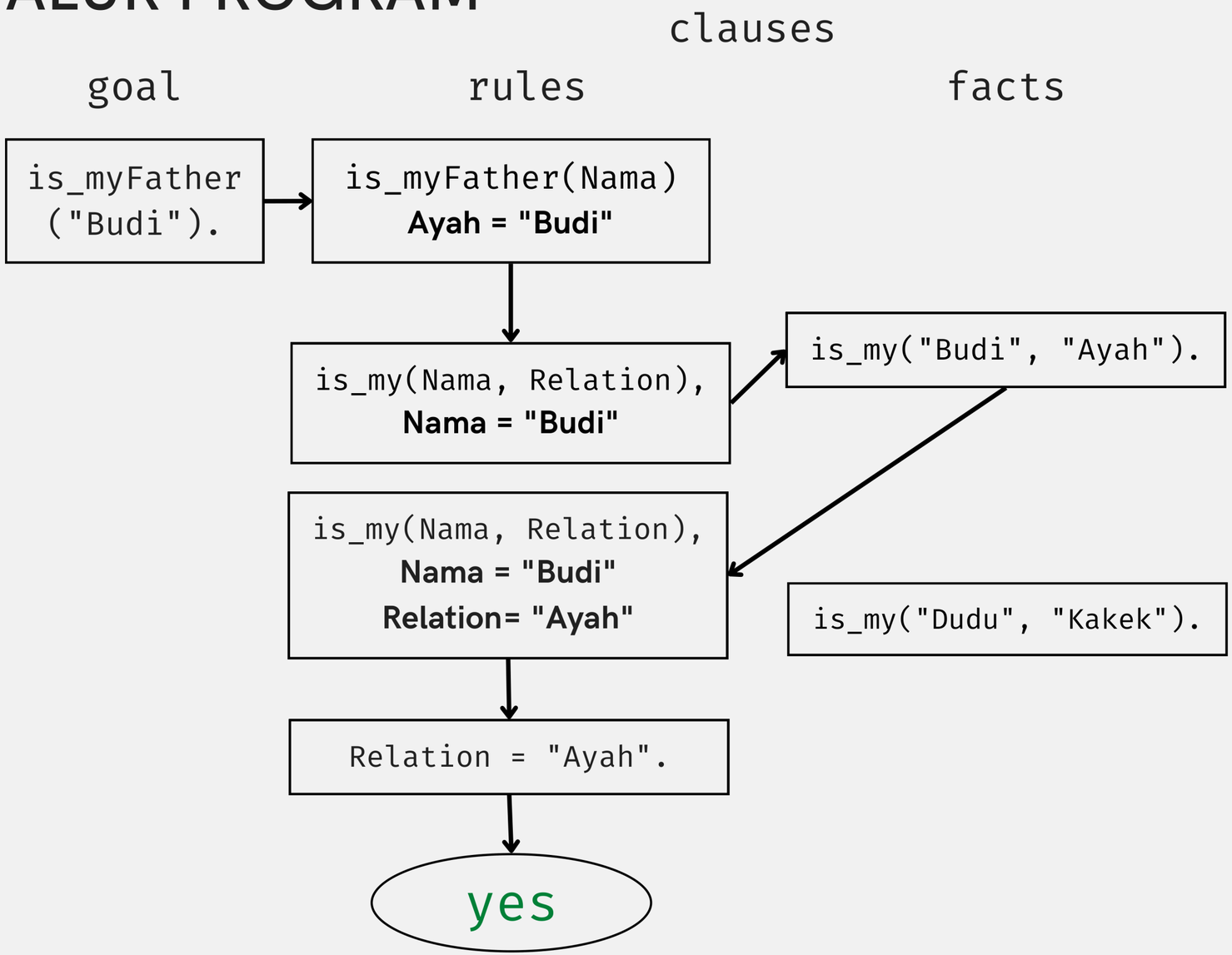


PROLOG

```
domains
    person = string
    relation = string
predicates
    nondeterm is_my(person, relation)
    nondeterm is_myFather(person)
clauses
    is_my("Budi", "Ayah").
    is_my("Dudu", "Kakek").
    is_myFather(Nama):-
         is_my(Nama, Relation),
         Relation = "Ayah".
goal
    is_myFather("Budi").
```

RUN

ALUR PROGRAM

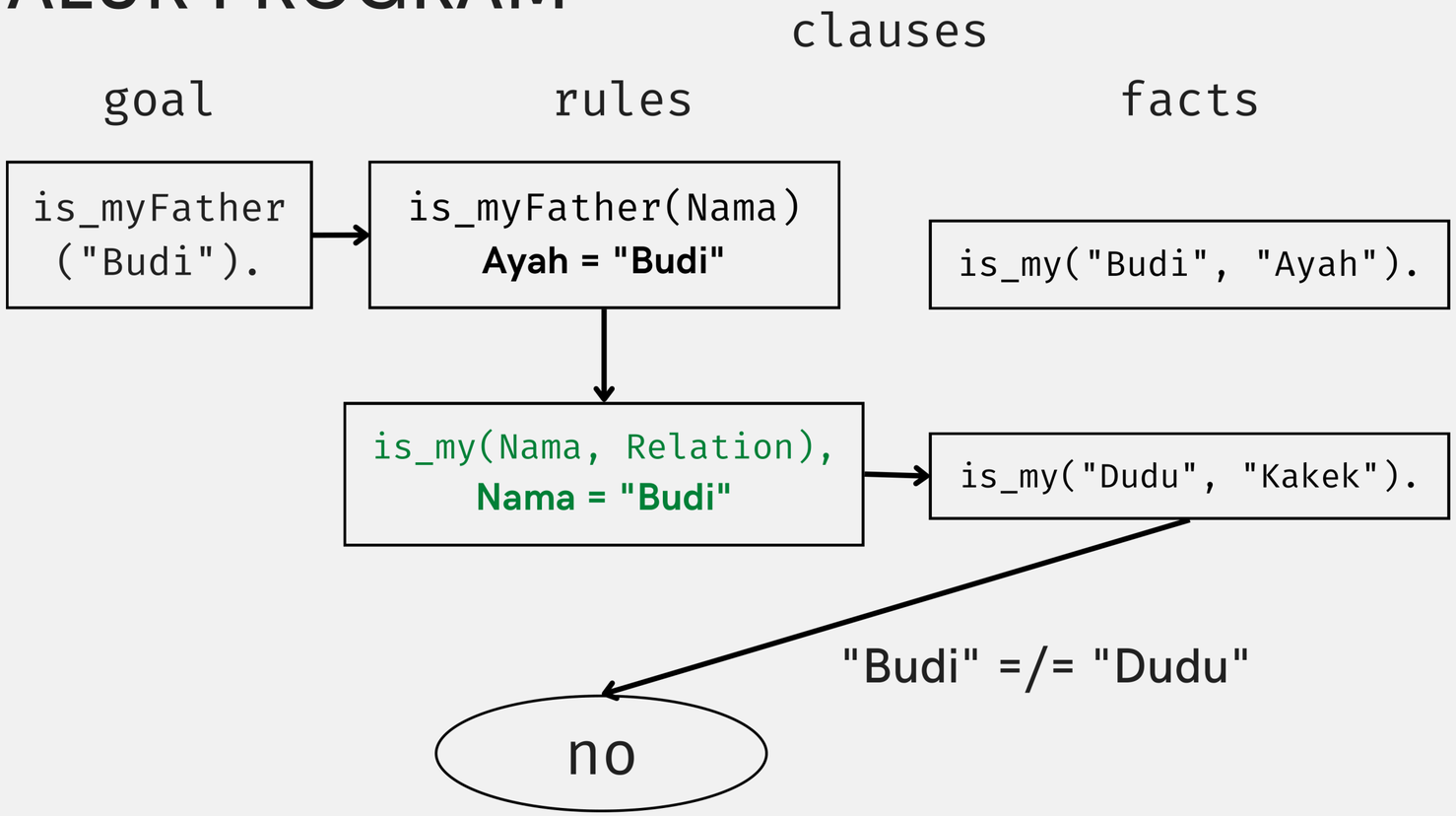


PROLOG

```
domains
    person = string
    relation = string
predicates
    nondeterm is_my(person, relation)
    nondeterm is_myFather(person)
clauses
    is_my("Budi", "Ayah").
    is_my("Dudu", "Kakek").
    is_myFather(Nama):-
        is_my(Nama, Relation),
        Relation = "Ayah".
goal
    is_myFather("Budi").
```

RUN

ALUR PROGRAM

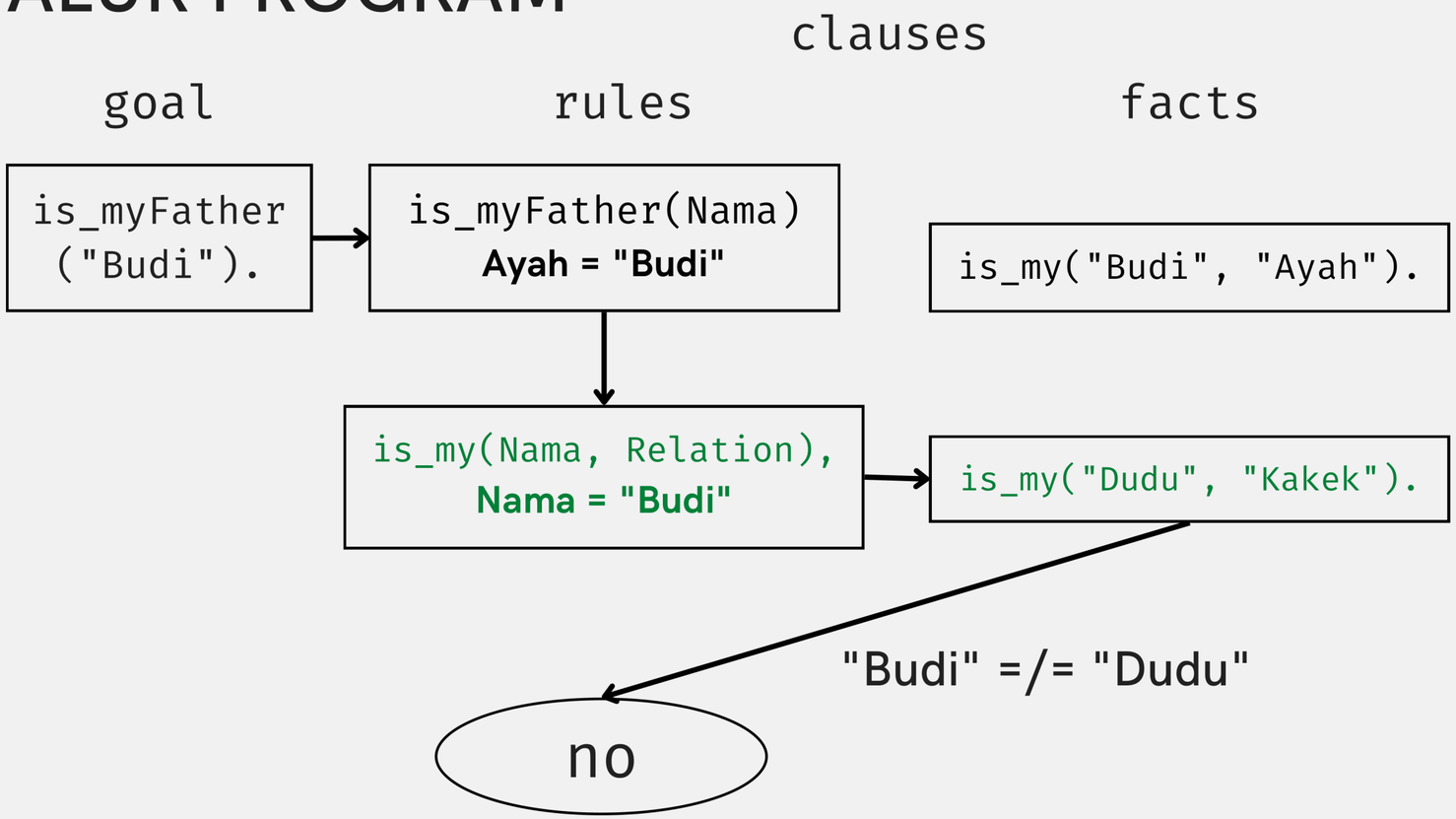


PROLOG

```
domains
    person = string
    relation = string
predicates
    nondeterm is_my(person, relation)
    nondeterm is_myFather(person)
clauses
    is_my("Budi", "Ayah").
    is_my("Dudu", "Kakek").
    is_myFather(Nama):-
        is_my(Nama, Relation),
        Relation = "Ayah".
goal
    is_myFather("Budi").
```

RUN

ALUR PROGRAM

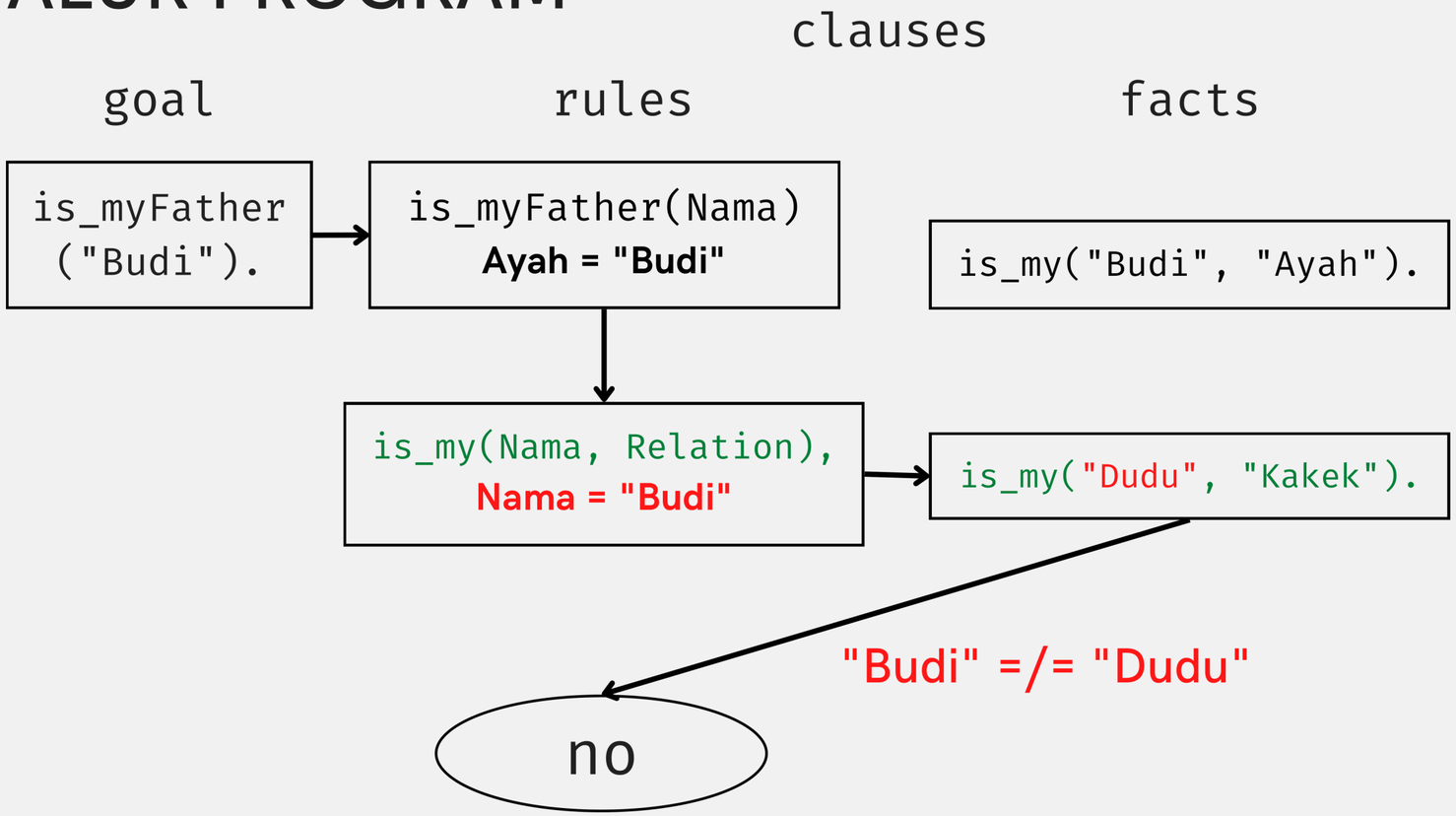


PROLOG

```
domains
    person = string
    relation = string
predicates
    nondeterm is_my(person, relation)
    nondeterm is_myFather(person)
clauses
    is_my("Budi", "Ayah").
    is_my("Dudu", "Kakek").
    is_myFather(Nama):-
        is_my(Nama, Relation),
        Relation = "Ayah".
goal
    is_myFather("Budi").
```

RUN

ALUR PROGRAM

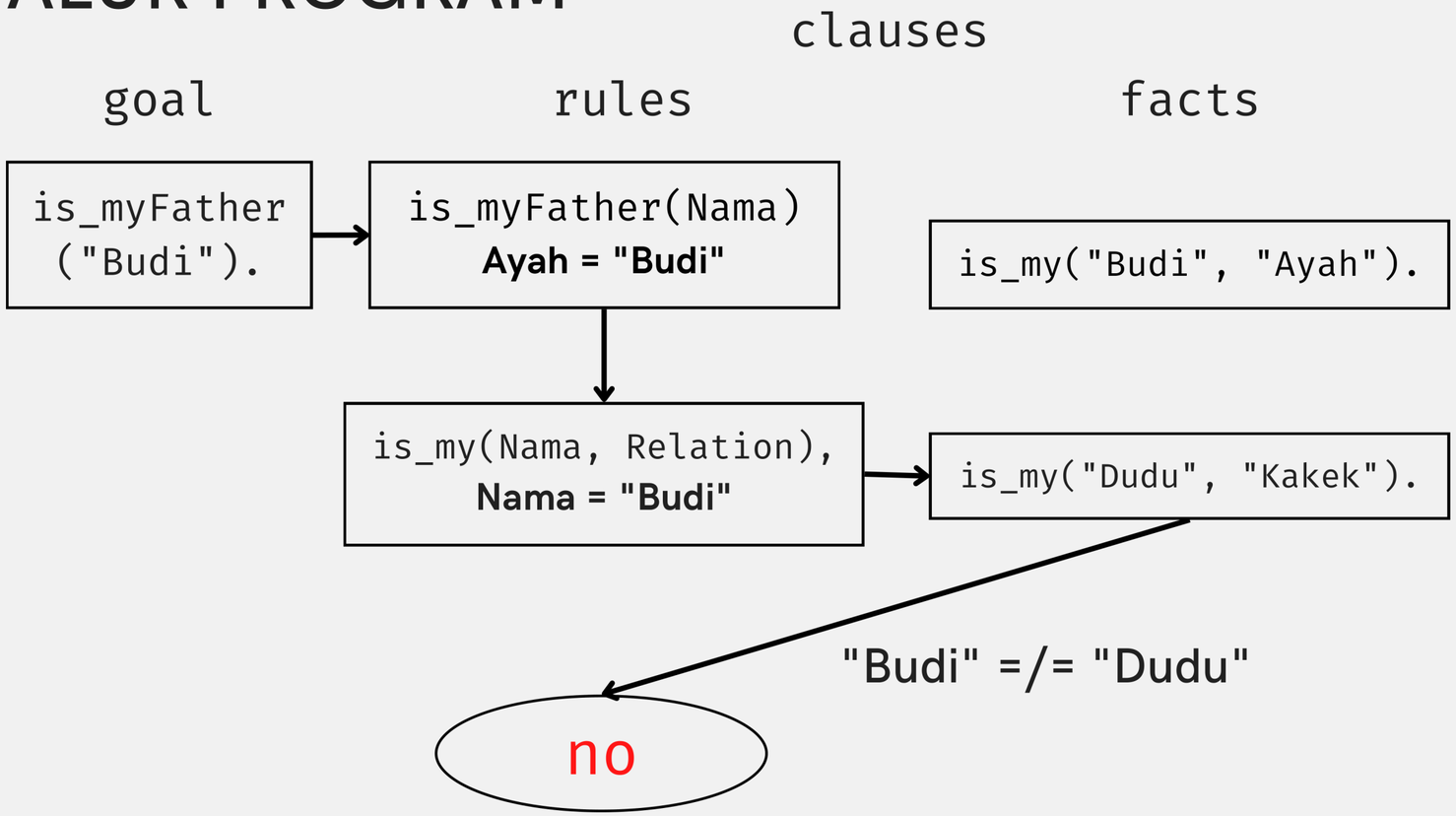


PROLOG

```
domains
    person = string
    relation = string
predicates
    nondeterm is_my(person, relation)
    nondeterm is_myFather(person)
clauses
    is_my("Budi", "Ayah").
    is_my("Dudu", "Kakek").
    is_myFather(Nama):-
         is_my(Nama, Relation),
         Relation = "Ayah".
goal
    is_myFather("Budi").
```

RUN

ALUR PROGRAM



AND CUT OR

AND ,

Everything must
be TRUE

Tanda pemisah per kalimat di PROLOG dilakukan
oleh koma (,) dan titik koma (;).

AND

Tanda AND ditandai dengan koma (,) dan memiliki karakteristik yang
sama dengan Logika AND di matematika atau pemprograman lainnya.

```
clauses
   is_myFather(Nama):-
        is_my(Nama, Relation),
        Relation = "Ayah",
        write(Nama, " adalah ayahku").
```

Baris berikutnya akan
dijalankan Jika dan hanya Jika
baris sekarang Benar

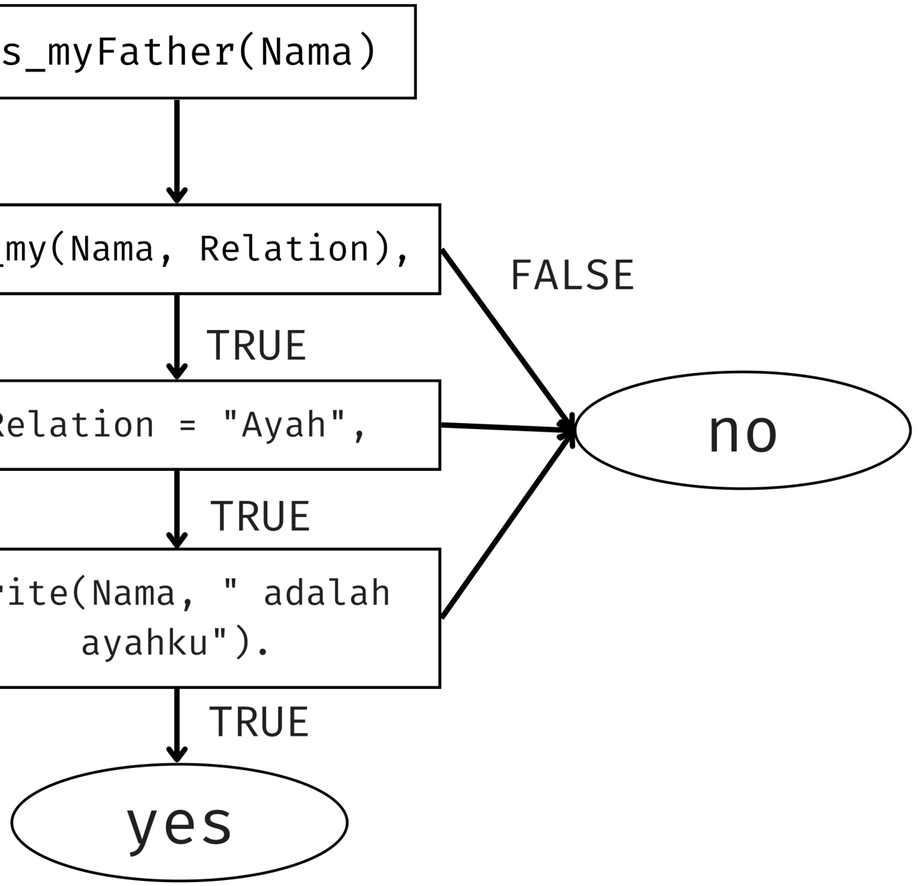
Jika kita jadikan pseudocode akan seperti ini kurang lebih alur kerjanya

```
func is_myFather(Nama: person){
   is_my(Nama, get(Relation))
   IF(is_my(Nama, get(Relation) == TRUE){
       IF(Relation == "Ayah"){
           OUTPUT Nama + " adalah ayahku"
       }
   }
}
```

AND ,

Everything must be TRUE

```
clauses
   is_myFather(Nama):-
     is_my(Nama, Relation),
     Relation = "Ayah",
     write(Nama, " adalah ayahku").
```



OR ,!;

Everbody make mistake right?

Mari tidak memusingkan arti cut (!)

OR

Tanda OR ditandai dengan titik koma (;) dan terkadang perlu ditemanin dengan tanda CUT (!) menjadi (, ! ;)

```
clauses
   is_myFather(Nama):-
        is_my(Nama, Relation),
        Relation = "Ayah",
        write(Nama, " adalah ayahku"),!;

        write(Nama, " bukan ayahku").
```

BLOK 1

BLOK 2

Tanda OR akan membuat code kita seakan-akan terbagi menjadi 2 blok terpisah.

OR ,!;

Everbody make mistake right?

Mari tidak memusingkan arti cut (!)

```
clauses
   is_myFather(Nama):-
        is_my(Nama, Relation),
        Relation = "Ayah",
        write(Nama, " adalah ayahku"),!;

        write(Nama, " bukan ayahku").
```

Jika kita jadikan pseudocode akan seperti ini kurang lebih alur kerjanya

```
    func is_myFather(Nama: person){
       is_my(Nama, get(Relation))
       IF(is_my(Nama, get(Relation) == TRUE){
           IF(Relation == "Ayah"){
               OUTPUT Nama + " adalah ayahku"
           }ELSE{
               OUTPUT Nama + " bukan ayahku"
           }
       }ELSE{
          OUTPUT Nama + " bukan ayahku"
       }
    }
```

OR ,!;

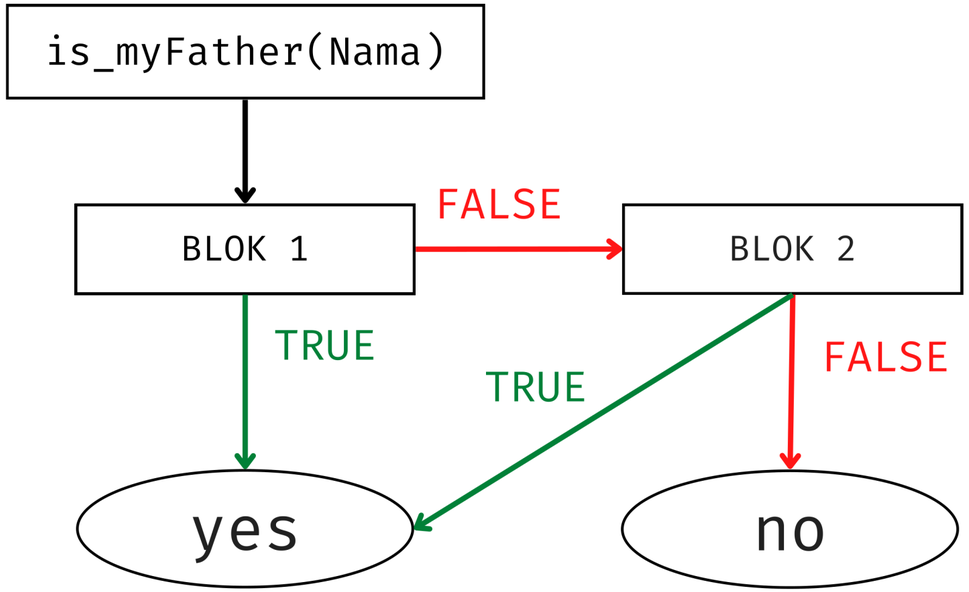
Everbody make mistake right?

Mari tidak memusingkan arti cut (!)

```
clauses
    is_myFather(Nama):-
          is_my(Nama, Relation),
          Relation = "Ayah",
          write(Nama, " adalah ayahku"),!;

          write(Nama, " bukan ayahku").
```

BLOK 1

BLOK 2



Mutiple Predicates

Twin, triple, quadra...

```
clauses
   is_myFather(Nama):-
        is_my(Nama, Relation),
        Relation = "Ayah",
        write(Nama, " adalah ayahku"),!;

        write(Nama, " bukan ayahku").
```

Dapat kita ubah menjadi seperti ini

```
clauses
   is_myFather(Nama):-
        is_my(Nama, Relation),
        Relation = "Ayah",
        write(Nama, " adalah ayahku").

   is_myFather(Nama):-
        is_my(Nama, Relation),
        Relation <> "Ayah",
        write(Nama, " bukan ayahku").
```

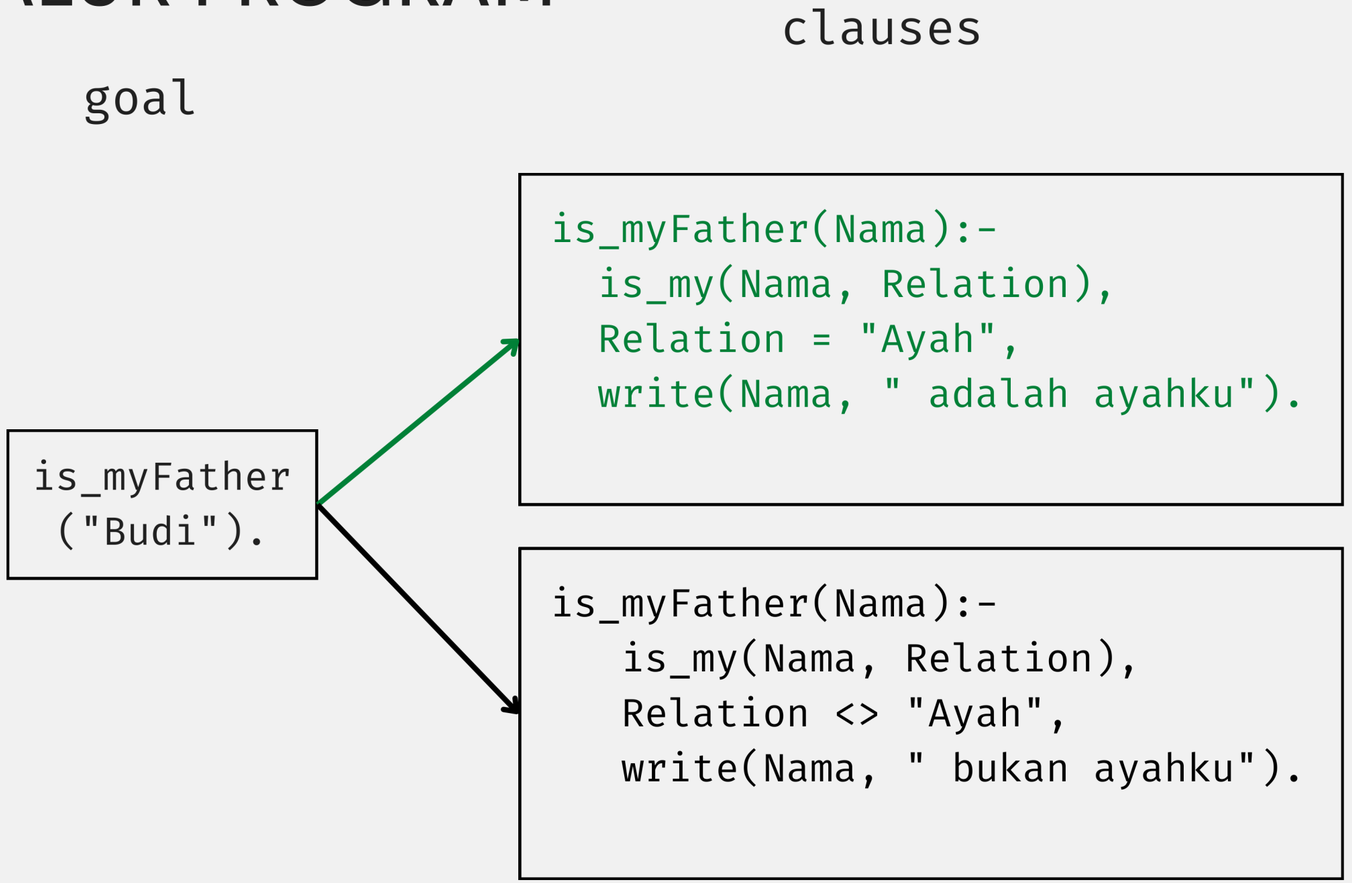
Bedanya apa?

PROLOG

```
domains
    person = string
    relation = string
predicates
    nondeterm is_my(person, relation)
    nondeterm is_myFather(person)
clauses
    is_my("Budi", "Ayah").
    is_my("Dudu", "Kakek").
    is_myFather(Nama):-
        is_my(Nama, Relation),
        Relation = "Ayah",
        write(Nama, " adalah ayahku").

    is_myFather(Nama):-
        is_my(Nama, Relation),
        Relation <> "Ayah",
        write(Nama, " bukan ayahku").
goal
    is_myFather("Budi").
```

RUN

ALUR PROGRAM



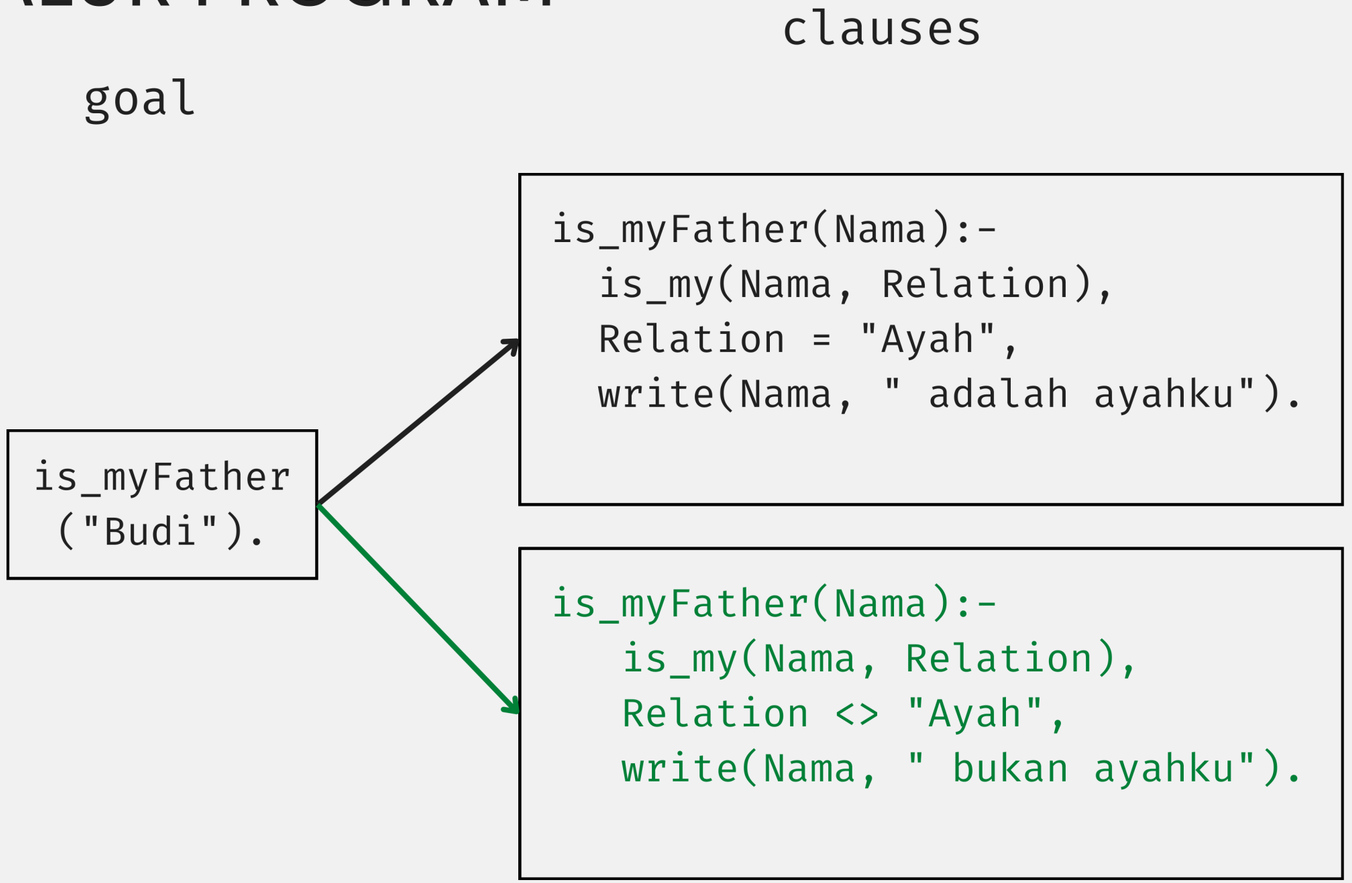
*Cara kerja isi di dlm predicates sama dengan penjelasan alur program sebelumnya sehingga tidak dibuat ulang lg

PROLOG

```
domains
    person = string
    relation = string
predicates
    nondeterm is_my(person, relation)
    nondeterm is_myFather(person)
clauses
    is_my("Budi", "Ayah").
    is_my("Dudu", "Kakek").
    is_myFather(Nama):-
        is_my(Nama, Relation),
        Relation = "Ayah",
        write(Nama, " adalah ayahku").

    is_myFather(Nama):-
        is_my(Nama, Relation),
        Relation <> "Ayah",
        write(Nama, " bukan ayahku").
goal
    is_myFather("Budi").
```

RUN

ALUR PROGRAM



*Perhatikan Predicates yang pertama kali dibaca adalah paling atas & predicates berikutnya dibaca juga meskipun predicates sebelumnya benar atau salah.

EZ?

PROLOG just another way to talking to Computer