

# DD1361 Programmeringsparadigm HT15

## LOGIKPROGRAMMERING 1

Dilian Gurov, TCS

# Delkursinnehåll

- ▶ Satser: fakta och regler
- ▶ Kontrollflöde
  - ▶ Unifiering
  - ▶ Backtracking
  - ▶ Negation
  - ▶ Snitt
- ▶ Induktiva datatyper och rekursion
  - ▶ Inbyggda datatyper: listor
  - ▶ Datatyper med PROLOG-termer: träd
- ▶ Programmeringstekniker med PROLOG
  - ▶ Generera-och-testa

## Ämnen

- ▶ Deklarativ- och logik-programmering
- ▶ Satser: fakta och regler
- ▶ Logisk versus procedurell läsning
- ▶ Kontrollflöde, Lådmodellen
- ▶ Rekursion

## Läsmaterial

- ▶ Boken: Brna, kap. 1-5
- ▶ PROLOG-fil: intro.pl (se kurswebbsida)
- ▶ Handouts: Föreläsningsanteckningar (se kurswebbsida)

## Deklarativ programmering

- ▶ Programmet är en beskrivning av problemdomänen
- ▶ Exekveringar resulterar från frågor (queries)

## Logikprogrammering

- ▶ En konkret realisering av deklarativ programmering med hjälp av **predikat**
- ▶ Domänen beskrivs som en sammansättning av predikatlogiska formler i så-kallad **Horn-klausul** form
- ▶ Frågor besvaras med en sök-algoritm som kallas för **resolution**

# Från funktioner till predikat

Om  $f$  är en funktion så har vi:

$$\begin{array}{ll} f(a) = b & \text{om } (a, b) \in f \quad \text{när } f \text{ är en relation} \\ \text{eller} & \text{om } f(a, b) \quad \text{när } f \text{ är ett predikat} \end{array}$$

Relationer är dock mer generella: inte varje relation är en funktion.

Exempel: "Fadern till  $a$  är  $b$ " kan deklareraras:

- ▶ som funktion:  $far(a) = b$
- ▶ som predikat:  $far(a, b)$
- ▶

Däremot är "a är granne till  $b$ " en relation men ingen funktion!

## Satser: fakta

Relationer kan deklaras explicit, varje tupel för sig själv. Sådana definitioner kallas för **fakta**:

```
far(agnes, petter).  
far(per, petter).  
far(anton, per).
```

```
mor(agnes, annika).  
mor(per, annika).  
mor(kia, agnes).  
mor(anton, agnes).
```

Nu när vi har skapat en relationell databas, vill vi kunna ställa frågor (**queries**) till den.

```
?- far(agnes, per).
```

```
?- far(anton, per).
```

```
?- far(monika, per).    % Closed world assumption!
```

Notera att PROLOG inte vet vad vi menar med alla symboler: den resonerar helt symboliskt, **utan interpretation!** Så vet PROLOG inte om `far(a, b)` betyder "fadern till a är b", eller "a är fader till b", eller tom att "himlen har färgen a och det är b stycken moln på den".

Vi kan också ställa frågor som involverar variabler:

```
?- far(agnes, X). % Vem är far till agnes?  
                % Eller, för vilka X är det sant  
                % att fadern till agnes är X?
```

```
?- far(X, petter). % Vems fader är petter?  
                % Flera svar möjliga!  
                % Fås fram med 'n' eller ";"
```

```
?- far(X, Y).
```



# Satser: regler

Relationer kan också definieras med **regler**:

$$\text{farmor}(X, Y) \text{ :- } \text{far}(X, Z), \text{mor}(Z, Y).$$

Läs: farmodern till X är Y  
    **om** fadern till X är (någon) Z  
    **och** modern till Z är Y.

Vi kan observera att:

- ▶ “:-” läses som “om”, dvs implikation “ $\leftarrow$ ” i omvänd riktning
- ▶ “,” läses som “och”, dvs konjunktion “ $\wedge$ ”
- ▶ variabler läses universellt
- ▶ Formler (satser) i ett sådant form kallas för **Horn-klausuler**

# Satser: regler

Disjunktiva regler kan ges som separata regler:

foralder(X, Y) :- far(X, Y).

foralder(X, Y) :- mor(X, Y).

Läs: en förälder till X är Y (OBS: relation fast inte funktion!)

**om** fadern till X är Y

**eller** modern till X är Y.

Följer logiska ekvivalensen:

$$p \vee q \rightarrow r \Leftrightarrow (p \rightarrow r) \wedge (q \rightarrow r)$$

# Logisk versus procedurell läsning

## Logisk läsning

- ▶ som en predikatlogisk formel, deklarativt

## Procedurell läsning

- ▶ proceduren som PROLOG följer för att hitta ett bevis
- ▶ PROLOG läser fakta och regler uppifrån och ned, och klausuler från vänster till höger
- ▶ PROLOG försöker instansiera variablerna med termer så att målet blir sant: **unifiering**
- ▶ om detta misslyckas med ett faktum eller en regel, går PROLOG till nästa: **backtracking**

Predikatlogikens syntax har både predikatsymboler såsom funtionssymboler, tagna från olika syntaktiska mängder. I PROLOG däremot finns det tyvärr ingen syntaktisk skillnad mellan dem, och båda representeras som atomer. Det är kontexten som bestämmer hur symbolerna tolkas. Tex i satsen:

$$a(b, c(X)).$$

är  $a$  en 1-ställig predikatsymbol,  $c$  en 1-ställig funktionsymbol,  $b$  en konstant (dvs en 0-ställig funktionsymbol), och  $X$  en variabel.

# Unifiering

Två termer kan **unifieras** om det finns en substitution av variabler med termer för vilken båda termer blir syntaktiskt identiska.

## Exempel:

- ▶  $X$  och  $agnes$  unifierar med substitution  $X = agnes$
- ▶  $far(X, petter)$  och  $far(agnes, petter)$  unifierar med substitution  $X = agnes$
- ▶  $far(agnes, Y)$  och  $far(agnes, petter)$  unifierar med substitution  $Y = petter$
- ▶  $far(X, petter)$  och  $far(agnes, Y)$  unifierar med substitution  $X = agnes, Y = petter$

# Kontrollflödet i PROLOG: Exempel 1

Query:

```
?- farmor(anton, X).
```

Kontrollflöde:

- ▶ Mål: `farmor(anton, X)`.
- ▶ Skapar regelinstans:  
`farmor(X1, Y1) :- far(X1, Z1), mor(Z1, Y1)`.
- ▶ Unifierar `X1=anton, Y1=X`
- ▶ Nästa delmål: `far(anton, Z1)`.
- ▶ Vid faktum 3 unifierar `Z1=per`
- ▶ Nästa delmål: `mor(per, X)`.
- ▶ Vid faktum 5 unifierar `X=annika`
- ▶ Inga flera mål: succé!
- ▶ Svar: `X=annika`

## Kontrollflödet i PROLOG: Exempel 2

Query:

```
?- foralder(kia, X).
```

Kontrollflöde:

- ▶ Mål: `foralder(kia, X).`
- ▶ Skapar regelinstans: `foralder(X1, Y1) :- far(X1, Y1).`
- ▶ Unifierar `X1=kia, Y1=X`
- ▶ Nästa delmål: `far(kia, X).`
- ▶ Lyckas inte: backtrackar!
- ▶ Skapar regelinstans: `foralder(X2, Y2) :- mor(X2, Y2).`
- ▶ Unifierar `X2=kia, Y2=X`
- ▶ Nästa delmål: `mor(kia, X).`
- ▶ Vid faktum 6 unifierar `X=agnes`
- ▶ Inga flera mål: succé!
- ▶ Svar: `X=agnes`

Regler kan vara **rekursiva**, dvs referera till sig själv:

```
forfader(X, Y) :- foralder(X, Y).
```

```
forfader(X, Y) :- foralder(X, Z), forfader(Z, Y).
```

Betrakta frågan:

```
?- forfader(kia, X).
```

Fundera kring kontrollflödet: vad skulle hända om vi vänder på ordningen på de två reglerna, eller på de två konjunkterna?



# Kontrollflödet i PROLOG: Lådmodellen

För att få ett bättre förståelse av kontrollflödet i Prolog, läs också om **lådmodellen**: Brna kap. 5, samt extra handouts.