Fundamentele limbajelor de programare

Programare logică. Prolog. Algoritmul de Unificare

Traian Florin Șerbănuță și Andrei Sipoș

Facultatea de Matematică și Informatică, DL Info

Anul II, Semestrul II, 2024/2025

Secțiunea 1

Programare Logică

Programare logică

- Programarea logică este o paradigmă de programare bazată pe logică.
- Unul din sloganurile programării logice:

  Program = Logică + Control
- Programarea logică poate fi privită ca o deducție controlată.
- Un program scris într-un limbaj de programare logică este
  - o listă de formule într-o logică
  - ce exprimă fapte și reguli despre o problemă.

Programare logică

Exemple de limbaje de programare logică:

- Prolog
- Answer set programming (ASP)
- Datalog

Programare logică - în mod idealist

- Un “program logic” este o colecție de proprietăți presupuse (sub formă de formule logice) despre lumea programului.
- Programatorul furnizează și o proprietate (o formula logică) care poate să fie sau nu adevărată în lumea respectivă (întrebare, query).
- Sistemul determină dacă proprietatea aflată sub semnul întrebării este o consecință a proprietăților presupuse în program.
- Programatorul nu specifică metoda prin care sistemul verifică dacă întrebarea este sau nu consecință a programului.

Exemplu de program logic

$$\frac{}{\begin{array}{rcl} \texttt{oslo} & \rightarrow & \texttt{windy} \\ \texttt{oslo} & \rightarrow & \texttt{norway} \\ \texttt{norway} & \rightarrow & \texttt{cold} \\ \texttt{cold} \wedge \texttt{windy} & \rightarrow & \texttt{winterIsComing} \\ & & \texttt{oslo} \end{array}}$$

Exemplu de întrebare. Este adevărat `winterIsComing`?

Prolog

- bazat pe logica clauzelor Horn
- semantica operațională este bazată pe rezoluție
- este Turing complet

Program:

```
windy :- oslo.
norway :- oslo.
cold :- norway.
winterIsComing :- windy, cold.
oslo.
```

Intrebare:

```
?- winterIsComing.
true
```

http://swish.swi-prolog.org/

Program în Prolog = bază de cunoștințe

Exemplu. Un program în Prolog:

```
father(peter,meg).
father(peter,stewie).

mother(lois,meg).
mother(lois,stewie).

griffin(peter).
griffin(lois).

griffin(X) :- father(Y,X),  griffin(Y).
```

Un program în Prolog este o bază de cunoștințe (Knowledge Base).

Program în Prolog = mulțime de predicate

Practic, gândim un program în Prolog ca o mulțime de predicate cu ajutorul cărora descriem *lumea* (*universul*) programului respectiv.

Exemplu.

```
father(peter,meg).
father(peter,stewie).

mother(lois,meg).
mother(lois,stewie).

griffin(peter).
griffin(lois).

griffin(X) :- father(Y,X),  griffin(Y).
```

Predicate:

father/2
mother/2
griffin/1

Un program în Prolog

Program

- Un program în Prolog este format din reguli de forma

  ```
  Head :- Body.
  ```

- Head este un predicat, iar Body este o secvență de predicate separate prin virgulă.
- Regulile fără `Body` se numesc fapte.

Exemple.

- Exemplu de regulă:

  ```
  griffin(X) :- father(Y,X), griffin(Y).
  ```

- Exemplu de fapt:

  ```
  father(peter,meg).
  ```

Interpretarea din punctul de vedere al logicii

Operatorul :- este implicația logică $\leftarrow$.

Exemplu. `comedy(X) :- griffin(X).`

dacă `griffin(X)` este adevărat, atunci `comedy(X)` este adevărat.

Virgula , este conjuncția $\wedge$.

Exemplu. `griffin(X) :- father(Y,X), griffin(Y).`

dacă `father(Y,X)` și `griffin(Y)` sunt adevărate,

atunci `griffin(X)` este adevărat.

Interpretarea din punctul de vedere al logicii

Mai multe reguli cu același Head reprezintă posibilități de defini același predicat.

Exemplu.

```
comedy(X) :- family_guy(X).
comedy(X) :- south_park(X).
comedy(X) :- disenchantment(X).
```

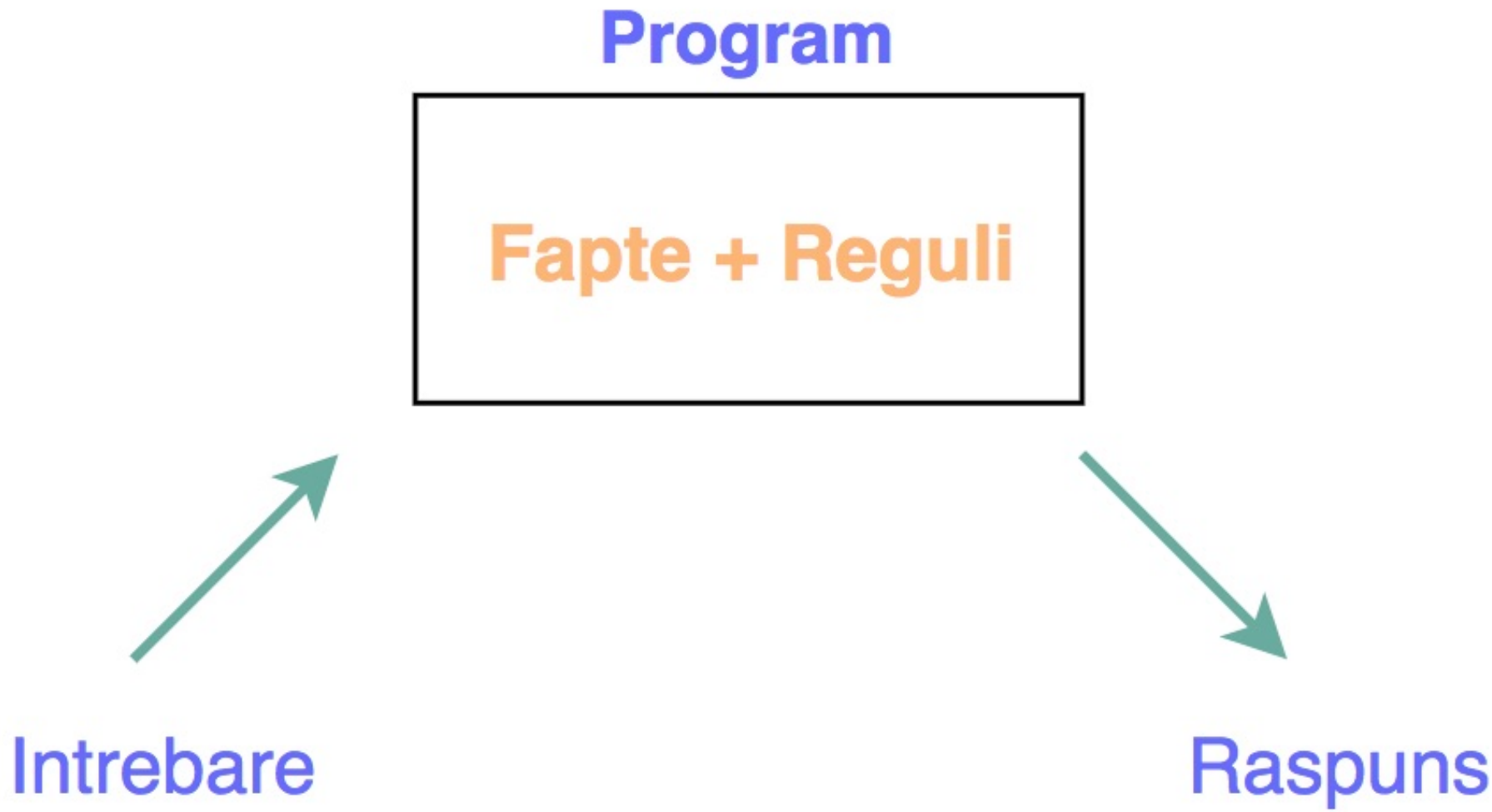
dacă `family_guy(X)` este adevărat sau `south_park(X)` este adevărat sau `disenchantment(X)` este adevărat,

atunci `comedy(X)` este adevărat.

Un program în Prolog

Cum folosim un program în Prolog?

Întrebări în Prolog



Întrebări și ținte în Prolog

- Prolog poate răspunde la întrebări legate de consecințele relațiilor descrise într-un program în Prolog.
- Întrebările sunt de forma:

  `?-` $\texttt{predicat}_1\texttt{(...)},\ldots,\texttt{predicat}_n\texttt{(...)}.$

- Prolog verifică dacă întrebarea este o consecință a relațiilor definite în program.
- Dacă este cazul, Prolog caută valori pentru variabilele care apar în întrebare astfel încât întrebarea să fie o consecință a relațiilor din program.
- Un predicat care este analizat pentru a răspunde la o întrebare se numește țintă (goal).

Întrebări în Prolog

Prolog poate da 2 tipuri de răspunsuri:

- `false` – dacă întrebarea nu este o consecință a programului.
- `true` sau valori pentru variabilele din întrebare dacă întrebarea este o consecință a programului.

Exemplu

```
?- griffin(meg)
true

?- griffin(glenn)
false
```

```
?- griffin(X)
X = petr ;
X = lois ;
X = meg ;
X = stewie ;
false
```

Cum găsește Prolog răspunsul

Pentru a găsi un răspuns,
Prolog încearcă regulile în ordinea apariției lor.

Exemplu. Să presupunem că avem programul:

```
foo(a).  foo(b).  foo(c).
```

și că punem întrebarea:

```
?- foo(X).
X = a.
```

Pentru a răspunde la întrebare se caută o potrivire (unificator) între scopul `foo(X)` și baza de cunoștințe. Răspunsul este substituția care realizează unificarea, în cazul nostru `X = a`.

Cum găsește Prolog răspunsul

Exemplu. Să presupunem că avem programul:

```
foo(a).  foo(b).  foo(c).
```

și că punem întrebările:

```
?- foo(X).
X = a.

?- foo(d).
false
```

Dacă nu se poate face unificarea, răspunsul este `false`.

Cum găsește Prolog răspunsul

Exemplu. Să presupunem că avem programul:

```
foo(a).  foo(b).  foo(c).
```

și că punem întrebarea:

```
?- foo(X).
X = a.
```

Dacă dorim mai multe răspunsuri, tastăm `;`

```
?- foo(X).
X = a ;
X = b ;
X = c.
```

Cum găsește Prolog răspunsul

Exemplu.

Să presupunem că avem programul:

```
foo(a).
foo(b).
foo(c).
```

și că punem întrebarea:

```
?- foo(X).
```

```
?- trace.
true.

[trace]  ?- foo(X).
   Call: (8) foo(_4556) ? creep
   Exit: (8) foo(a) ? creep
X = a ;
   Redo: (8) foo(_4556) ? creep
   Exit: (8) foo(b) ? creep
X = b ;
   Redo: (8) foo(_4556) ? creep
   Exit: (8) foo(c) ? creep
X = c.
```

Cum găsește Prolog răspunsul

Pentru a găsi un raspuns, Prolog redenumește variabilele.

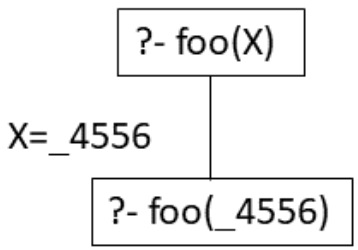
Exemplu.

Să presupunem că avem programul:

```
foo(a).
foo(b).
foo(c).
```

și că punem întrebarea:

```
?- foo(X).
```



Cum găsește Prolog răspunsul

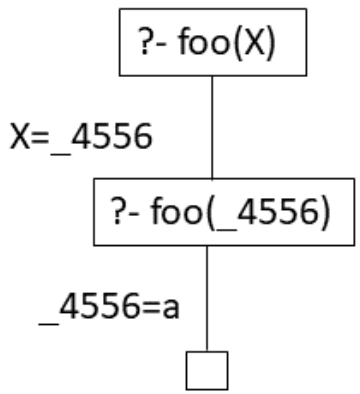
Exemplu.

Să presupunem că avem programul:

```
foo(a).
foo(b).
foo(c).
```

și că punem întrebarea:

```
?- foo(X).
```



În acest moment, a fost găsită prima soluție: `X=_4556=a`.

Cum găsește Prolog răspunsul

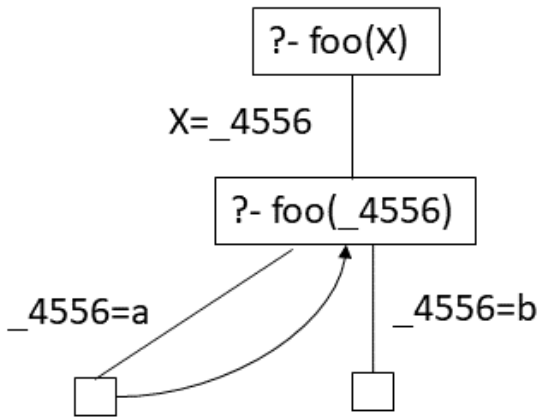
Exemplu.

Să presupunem că avem programul:

```
foo(a).
foo(b).
foo(c).
```

și că punem întrebarea:

```
?- foo(X).
```



Dacă se dorește încă un răspuns, atunci se face un pas înapoi în arborele de căutare și se încearcă satisfacerea țintei cu o nouă valoare.

Cum găsește Prolog răspunsul

Exemplu.

2 Să presupunem că avem programul:

```
foo(a).
foo(b).
foo(c).
```

și că punem întrebarea:

```
?- foo(X).
```

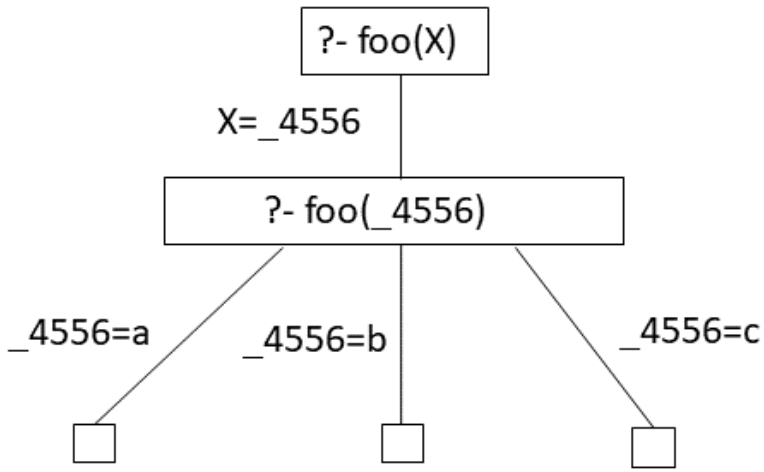


Figure 1: arborele de căutare

Cum găsește Prolog răspunsul

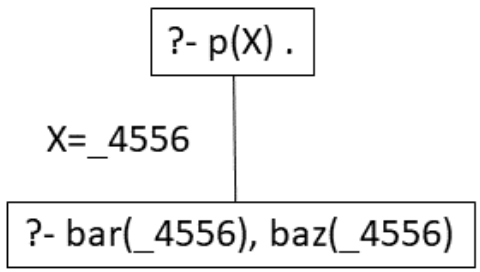
Exemplu.

Să presupunem că avem programul:

```
bar(b).
bar(c).
baz(c).
```

și că punem întrebarea:

```
?- bar(X),baz(X).
```



Cum găsește Prolog răspunsul

Prolog se întoarce la ultima alegere dacă o subțintă eșuează.

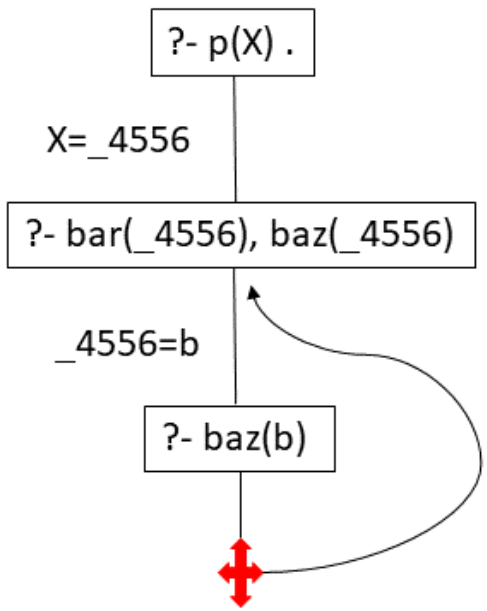
Exemplu.

Să presupunem că avem programul:

```
bar(b).
bar(c).
baz(c).
```

și că punem întrebarea:

```
?- bar(X),baz(X).
```



Cum găsește Prolog răspunsul

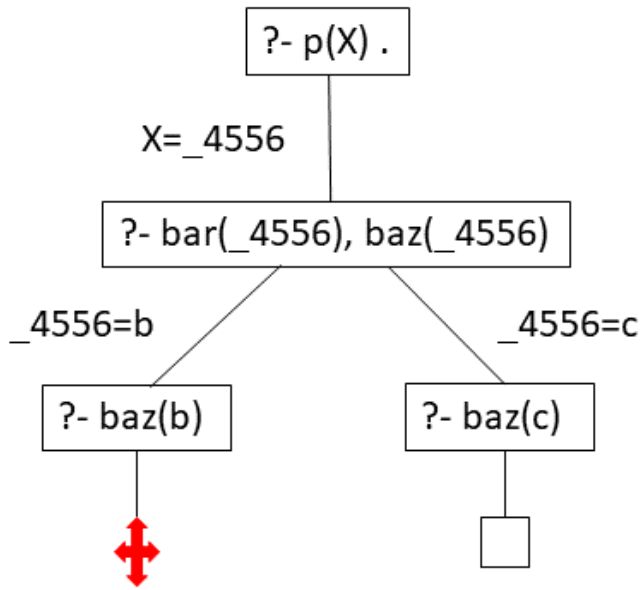
Exemplu.

Să presupunem că avem programul:

```
bar(b).
bar(c).
baz(c).
```

și că punem întrebarea:

```
?- bar(X),baz(X).
```



Soluția găsită este: `X=_4556=c`.

Problema colorării hărților

Să se coloreze o hartă dată cu o mulțime de culori dată astfel încât oricare două țări vecine să fie colorate diferit.

Cum modelăm această problemă în Prolog?

Trebuie să definim:

- culorile
- harta
- constrângerile

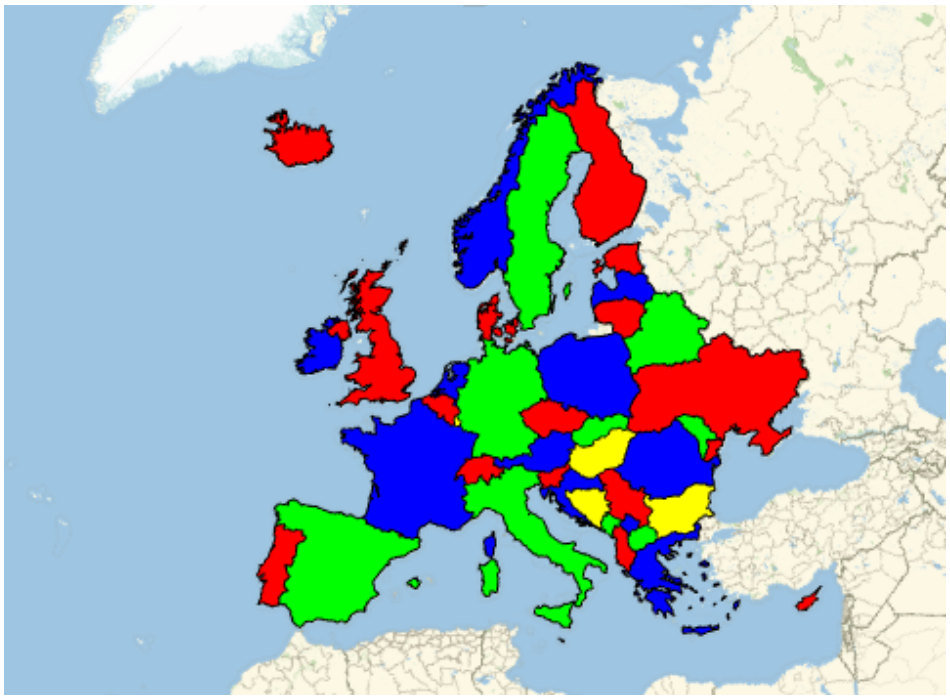


Figure 2: Hartă colorată

Problema colorării hărților

Definim culorile, harta și constrângerile.

```
culoare(albastru).
culoare(rosu).
culoare(verde).
culoare(galben).

harta(RO,SE,MD,UA,BG,HU) :- vecin(RO,SE), vecin(RO,UA),
                            vecin(RO,MD), vecin(RO,BG),
                            vecin(RO,HU), vecin(UA,MD),
                            vecin(BG,SE), vecin(SE,HU).

vecin(X,Y) :- culoare(X), culoare(Y), X \== Y.
```

Problema colorării hărților

Ce răspuns primim?

```
?- harta(RO,SE,MD,UA,BG,HU).
RO = albastru,
SE = UA, UA = rosu,
MD = BG, BG = HU, HU = verde
```

Secțiunea 2

Unificare

Termeni (Notații)

- $x, y, z, \ldots$ pentru variabile
- $f, g, h, \ldots$ pentru simboluri de funcții arbitrare
  - *Aritatea* lui f e numărul de argumente ale simbolului f
- $a, b, c, \ldots$ pentru constante (simboluri de funcții fără argumente)
- $s, t, u, \ldots$ pentru termeni, formați din:
  - variabile, constante,
  - prin aplicarea de simboluri
  - de un număr finit de ori
- $Var(t)$ mulțimea variabilelor care apar în t
- ecuații $s \doteq t$ pentru o pereche de termeni

Termeni (Exemple)

- $f(x, g(x, a), y)$ este un termen, unde
  - f are aritate 3,
  - g are aritate 2,
  - a este o constanta
- $Var(f(x, g(x, a), y)) = \{x, y\}$

Substituții

- Substituțiile sunt o modalitate de a înlocui variabile cu alți termeni.
- Substituțiile se aplică simultan pe toate variabilele.

Aplicarea unei substituții – Exemplu:

- $\Theta = \{x \mapsto f(x, y), y \mapsto g(a)\}$
- $t = f(x, g(f(x, f(y, z))))$
- $\Theta(t) = f(f(x, y), g(f(f(x, y), f(g(a), z))))$

Unificare

Doi termeni t_1 și t_2 se **unifică** dacă există valori pentru variabile, adică o substituție Θ, care egalează cei doi termeni:

$$\Theta(t_1) = \Theta(t_2)$$

În acest caz, Θ se numește **unificator** al termenilor t_1 și t_2.

Spunem ca un termen t e **mai general** decât un termen t' dacă t' se obține din t prin "specializarea" unor variabile, adică înlocuirea lor cu termeni.

- Formal, există o substituție Δ astfel încât $t' = \Delta(t)$
- De exemplu, $f(x, a, y)$ este mai general decât $f(g(y), a, h(b))$

Spunem că un unificator Θ e **mai general** decât alt unificator Θ' dacă există o substituție Δ astfel încât $\Theta'(x) = \Delta(\Theta(x))$ pentru orice variabilă x

Unificatori (Exemplu)

- $t = x + (y * y) = +(x, *(y, y))$
- $t' = x + (y * x) = +(x, *(y, x))$
- $\Theta = \{x \mapsto y\}$
  - $\Theta(t) = y + (y * y)$
  - $\Theta(t') = y + (y * y)$
- $\Theta' = \{x \mapsto 0, y \mapsto 0\}$
  - $\Theta'(t) = 0 + (0 * 0)$
  - $\Theta'(t') = 0 + (0 * 0)$
  - $\Theta' = \Theta; \{y \mapsto 0\}$
- Θ este mai general decât Θ' (e chiar cel mai general)

Secțiunea 3

Un algoritm de unificare

Algoritm de unificare – inițializare

Pentru o mulțime (finită) de ecuații $\{t_1 \doteq t'_1, \ldots, t_n \doteq t'_n\}$, algoritmul calculează (dacă există) un unificator Θ pentru toate ecuațiile.

$$\Theta(t_1) = \Theta(t'_1), \ldots, \Theta(t_n) = \Theta(t'_n)$$

Algoritmul lucrează cu două obiecte:

- Lista de rezolvat: R
- Substituția calculată: S

Inițial:

- Lista de rezolvat: $R = \{t_1 \doteq t'_1, \ldots, t_n \doteq t'_n\}$
- Substituția calculată: $S = \varnothing$ – identitatea

Algoritm de unificare – execuție

Algoritmul constă în aplicarea nedeterministă a regulilor de mai jos:

SCOATE o ecuație de forma $t \doteq t$ din R este eliminată.

DESCOMPUNE o ecuație de forma $f(t_1, \ldots, t_n) \doteq f(t'_1, \ldots, t'_n)$ din R este înlocuită cu ecuațiile $t_1 \doteq t'_1, \ldots, t_n \doteq t'_n$.

REZOLVĂ o ecuație de forma $x \doteq t$ sau $t \doteq x$ din R, unde variabila x nu apare în termenul t, este mutată sub forma $x \mapsto t$ în S.
În toți ceilalți termeni (din R și S), x este înlocuit cu t.

EȘEC (conflict) există în R o ecuație de forma $f(s_1, \ldots, s_n) \doteq g(t_1, \ldots, t_m)$ cu $f \neq g$.

EȘEC (ciclu) există în R o ecuație de forma $x \doteq t$ sau $t \doteq x$ cu $t \neq x$ și $x \in Var(t)$.

Algoritm de unificare – terminare

Algoritmul se termină normal dacă $R = \emptyset$.
În acest caz, S este *cel mai general unificator* pentru intrarea dată.

Algoritmul este oprit cu concluzia inexistenței unui unificator dacă este folosită vreuna din regulile care produce eșec

Algoritm de unificare - schemă

	Lista soluție S	Lista de rezolvat R
Inițial	$\varnothing$	$t_1 \doteq t'_1, \ldots, t_n \doteq t'_n$
SCOATE	$\dfrac{S}{S}$	$\dfrac{R', t \doteq t}{R'}$
DESCOMPUNE	$\dfrac{S}{S}$	$\dfrac{R', f(t_1, \ldots, t_n) \doteq f(t'_1, \ldots t'_n)}{R', t_1 \doteq t'_1, \ldots t_n \doteq t'_n}$
REZOLVĂ	$\dfrac{S}{x \mapsto t, S[x := t]}$	$\dfrac{R', x \doteq t}{R'[x := t]}$ dacă $x \notin Var(t)$
Final	S	$\varnothing$

- $R[x := t]$: x este înlocuit cu t în toate ecuațiile din R
- $S[x := t]$: x este înlocuit cu t în $S(y)$ pentru orice varibilă y
- în regula REZOLVĂ putem avea și $t \doteq x$ în R'

Exemplu: $\{g(y) \doteq x,\ f(x, h(x), y) \doteq f(g(z), w, z)\}$

S	R	pas
$\varnothing$	$g(y)\doteq x,\ f(x,h(x),y)\doteq f(g(z),w,z)$	REZOLVĂ
$x\mapsto g(y)$	$f(g(y),h(g(y)),y)\doteq f(g(z),w,z)$	DESCOMPUNE
$x\mapsto g(y)$	$g(y)\doteq g(z),\ h(g(y))\doteq w,\ y\doteq z$	REZOLVĂ
$w\mapsto h(g(y)),x\mapsto g(y)$	$g(y)\doteq g(z),\ y\doteq z$	REZOLVĂ
$y\mapsto z,x\mapsto g(z),w\mapsto h(g(z))$	$g(z)\doteq g(z)$	SCOATE
$y\mapsto z,x\mapsto g(z),w\mapsto h(g(z))$	$\varnothing$	

$\Theta = \{y \mapsto z,\ x \mapsto g(z),\ w \mapsto h(g(z))\}$ este cmgu.

Exemplu: $\{g(y) \doteq x,\ f(x, h(y), y) \doteq f(g(z), b, z)\}$

S	R	
$\varnothing$	$g(y) \doteq x,\ f(x, h(y), y) \doteq f(g(z), b, z)$	DESCOMPUNE
$\varnothing$	$g(y) \doteq x,\ x \doteq g(z),\ h(y) \doteq b,\ y \doteq z$	EȘEC (conflict)

- h și b sunt simboluri de funcții diferite!
- Nu există unificator pentru acești termeni.

Exemplu: $\{g(y) \doteq x,\ f(x, h(x), y) \doteq f(y, w, z)\}$

S	R	
$\varnothing$	$g(y) \doteq x,\ f(x, h(x), y) \doteq f(y, w, z)$	DESCOMPUNE
$\varnothing$	$g(y) \doteq x,\ x \doteq y,\ h(x) \doteq w,\ y \doteq z$	REZOLVĂ
$x \mapsto y$	$g(y) \doteq y,\ h(y) \doteq w,\ y \doteq z$	EȘEC (ciclu)

- În ecuația $g(y) \doteq y$, variabila y apare în termenul $g(y)$.
- Nu există unificator pentru aceste ecuații.