

# Curs 3

### Programare logică – cazul logicii de ordinul I

#### Logica clauzelor definite/Logica Horn – un fragment al logicii de ordinul I

- Singurele formule admise sunt **clauze definite**:
  - **formule atomice**:  $P(t_1, \dots, t_n)$
  - $A_1 \wedge \dots \wedge A_n \rightarrow B$ , unde toate  $A_i, B$  sunt formule atomice.
- Majoritatea limbajelor de programare logice (de ex., Prolog) folosesc acest fragment.

#### Problema programării logice: $T \models A_1 \wedge \dots \wedge A_n$

- $T$  mulțime de clauze definite
- toate  $A_i$  sunt formule atomice

# Cuprins

---

- 1 Substituții și unificare
- 2 Sistem de deducție pentru logica Horn

## Substituții și unificare

# Substituții

## Definiție

O **substituție**  $\sigma$  este o funcție (parțială) de la variabile la termeni, adică

$$\sigma : V \rightarrow Trm_{\mathcal{L}}$$

## Exemplu

În notația uzuală,  $\sigma = \{x/a, y/g(w), z/b\}$ .

# Substituții

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

## Exemplu

- substituția  $\sigma = \{x/a, y/g(w), z/b\}$
- $\sigma(P(x, g(x), y)) = P(a, g(a), g(w))$
- substituția  $\phi = \{x/y, y/g(a)\}$
- $\phi(f(x)) = f(y)$
- $\phi(f(x)) \neq f(g(a))$

# Substituții

Două substituții  $\sigma_1$  și  $\sigma_2$  se pot compune

$$\sigma_1; \sigma_2$$

(aplicăm întâi  $\sigma_1$ , apoi  $\sigma_2$ ).

## Exemplu

$$\square t = P(u, v, x, y, z)$$

$$\square \tau = \{x/f(y), y/f(a), z/u\}$$

$$\square \mu = \{y/g(a), u/z, v/f(f(a))\}$$

$$\square (\tau; \mu)(t) = \mu(\tau(t)) = \mu(P(u, v, f(y), f(a), u)) = \\ = P(z, f(f(a)), f(g(a)), f(a), z)$$

$$\square (\mu; \tau)(t) = \tau(\mu(t)) = \tau(P(z, f(f(a)), x, g(a), z)) \\ = P(u, f(f(a)), f(y), g(a), u)$$

# Unificare

- Doi termeni  $t_1$  și  $t_2$  **se unifică** dacă există o substituție  $\theta$  astfel încât
$$\theta(t_1) = \theta(t_2).$$
- În acest caz,  $\theta$  se numește **unificatorul** termenilor  $t_1$  și  $t_2$ .
- În programarea logică, unificatorii sunt ingredientele de bază în execuția unui program.
- Un unificator  $\nu$  pentru  $t_1$  și  $t_2$  este un **cel mai general unificator** (**cgu, mgu**) dacă pentru orice alt unificator  $\nu'$  pentru  $t_1$  și  $t_2$ , există o substituție  $\mu$  astfel încât

$$\nu' = \nu; \mu.$$



# Unificator

## Exemplu

- $t = x + (y \star y) = +(x, \star(y, y))$
- $t' = x + (y \star x) = +(x, \star(y, x))$
- $\nu = \{x/y, y/y\}$ 
  - $\nu(t) = y + (y \star y)$
  - $\nu(t') = y + (y \star y)$
  - $\nu$  este **cgu**
- $\nu' = \{x/0, y/0\}$ 
  - $\nu'(t) = 0 + (0 \star 0)$
  - $\nu'(t') = 0 + (0 \star 0)$
  - $\nu' = \nu; \{y/0\}$
  - $\nu'$  este **unificator**, dar nu este **gcu**

# Algoritmul de unificare

- Pentru o mulțime finită de termeni  $\{t_1, \dots, t_n\}$ ,  $n \geq 2$ , algoritmul de unificare stabilește dacă există un cgu.
- Există algoritmi mai eficienți, dar îl alegem pe acesta pentru simplitatea sa.
- Algoritmul lucrează cu două liste:
  - Lista soluție:  $S$
  - Lista de rezolvat:  $R$
- Inițial:
  - Lista soluție:  $S = \emptyset$
  - Lista de rezolvat:  $R = \{t_1 \dot{=} t_2, \dots, t_{n-1} \dot{=} t_n\}$
- $\dot{=}$  este un simbol nou care ne ajută sa formăm perechi de termeni (ecuații).

# Algoritmul de unificare

Algoritmul constă în aplicarea regulilor de mai jos:

## □ SCOATE

- orice ecuație de forma  $t \doteq t$  din  $R$  este eliminată.

## □ DESCOMPUNE

- orice ecuație de forma  $f(t_1, \dots, t_n) \doteq f(t'_1, \dots, t'_n)$  din  $R$  este înlocuită cu ecuațiile  $t_1 \doteq t'_1, \dots, t_n \doteq t'_n$ .

## □ REZOLVĂ

- orice ecuație de forma  $x \doteq t$  sau  $t \doteq x$  din  $R$ , unde variabila  $x$  nu apare în termenul  $t$ , este mutată sub forma  $x \doteq t$  în  $S$ . În toate celelalte ecuații (din  $R$  și  $S$ ),  $x$  este înlocuit cu  $t$ .

# Algoritmul de unificare

Algoritmul se termină normal dacă  $R = \emptyset$ . În acest caz,  $S$  dă cgu.

Algoritmul este oprit cu concluzia inexistenței unui cgu dacă:

- 1 În  $R$  există o ecuație de forma

$$f(t_1, \dots, t_n) \doteq g(t'_1, \dots, t'_k) \text{ cu } f \neq g.$$

- 2 În  $R$  există o ecuație de forma  $x \doteq t$  sau  $t \doteq x$  și variabila  $x$  apare în termenul  $t$ .

# Algoritmul de unificare - schemă

	Lista soluție S	Lista de rezolvat R
Inițial	$\emptyset$	$t_1 \doteq t'_1, \dots, t_n \doteq t'_n$
SCOATE	S	$R', t \doteq t$
	S	$R'$
DESCOMPUNE	S	$R', f(t_1, \dots, t_n) \doteq f(t'_1, \dots, t'_n)$
	S	$R', t_1 \doteq t'_1, \dots, t_n \doteq t'_n$
REZOLVĂ	S	$R', x \doteq t$ sau $t \doteq x$ , x nu apare în t
	$x \doteq t, S[x/t]$	$R'[x/t]$
Final	S	$\emptyset$

$S[x/t]$ : în toate ecuațiile din S, x este înlocuit cu t

# Exemplu

## Exemplu

□ Ecuațiile  $\{g(y) \doteq x, f(x, h(x), y) \doteq f(g(z), w, z)\}$  au gcu?

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

□  $\nu = \{y/z, x/g(z), w/h(g(z))\}$  este cgu.

# Exemplu

## Exemplu

- Ecuațiile  $\{g(y) \doteq x, f(x, h(y), y) \doteq f(g(z), b, z)\}$  au gcu?

$S$	$R$	
$\emptyset$	$g(y) \doteq x, f(x, h(y), y) \doteq f(g(z), b, z)$	REZOLVĂ
$x \doteq g(y)$	$f(g(y), h(y), y) \doteq f(g(z), b, z)$	DESCOMPUNE
$x \doteq g(y)$	$g(y) \doteq g(z), h(y) \doteq b, y \doteq z$	- EȘEC -

- $h$  și  $b$  sunt simboluri de operații diferite!
- Nu există unificator pentru ecuațiile din  $U$ .

# Exemplu

## Exemplu

- Ecuațiile  $\{g(y) \doteq x, f(x, h(x), y) \doteq f(y, w, z)\}$  au gcu?

$S$	$R$	
$\emptyset$	$g(y) \doteq x, f(x, h(x), y) \doteq f(y, w, z)$	REZOLVĂ
$x \doteq g(y)$	$f(g(y), h(g(y)), y) \doteq f(y, w, z)$	DESCOMPUNE
$x \doteq g(y)$	$g(y) \doteq y, h(g(y)) \doteq w, y \doteq z$	- EȘEC -

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



# Terminarea algoritmului

## Propoziție

Algoritmul de unificare se termină.

## Demonstrație

- Notăm cu
  - $N_1$ : numărul variabilelor care apar în  $R$
  - $N_2$ : numărul aparițiilor simbolurilor care apar în  $R$
- Este suficient să arătăm că perechea  $(N_1, N_2)$  descrește strict în ordine lexicografică la execuția unui pas al algoritmului:

dacă la execuția unui pas  $(N_1, N_2)$  se schimbă în  $(N'_1, N'_2)$ , atunci

$$(N_1, N_2) \geq_{lex} (N'_1, N'_2)$$

## Demonstrație (cont.)

Fiecare regulă a algoritmului modifică  $N_1$  și  $N_2$  astfel:

	$N_1$	$N_2$
SCOATE	$\geq$	$>$
DESCOMPUNE	$=$	$>$
REZOLVĂ	$>$	

- $N_1$ : numărul variabilelor care apar în  $R$
- $N_2$ : numărul aparițiilor simbolurilor care apar în  $R$



# Corectitudinea algoritmului

## Lema 1

Mulțimea unificatorilor pentru reuniunea ecuațiilor din  $R$  și  $S$  nu se modifică prin aplicarea celor trei reguli ale algoritmului de unificare.

## Demonstrație

Analizăm fiecare regulă:

- **SCOATE**: evident
- **DESCOMPUNE**: Trebuie să arătăm că

$$\begin{array}{ccc} \nu \text{ unificator pt.} & \Leftrightarrow & \nu \text{ unificator pt.} \\ f(t_1, \dots, t_n) \doteq f(t'_1, \dots, t'_n) & & t_i \doteq t'_i, \text{ or. } i = 1, \dots, n. \end{array}$$

$$\begin{aligned} & \nu \text{ unif. pt. } f(t_1, \dots, t_n) \doteq f(t'_1, \dots, t'_n) \\ \Leftrightarrow & \nu(f(t_1, \dots, t_n)) = \nu(f(t'_1, \dots, t'_n)) \\ \Leftrightarrow & f(\nu(t_1), \dots, \nu(t_n)) = f(\nu(t'_1), \dots, \nu(t'_n)) \\ \Leftrightarrow & \nu(t_i) = \nu(t'_i), \text{ or. } i = 1, \dots, n \\ \Leftrightarrow & \nu \text{ unificator pt. } t_i \doteq t'_i, \text{ or. } i = 1, \dots, n \end{aligned}$$

## Demonstrație (cont.)

### □ REZOLVĂ:

- Se observă că or. unificator  $\nu$  pt. reuniunea ecuațiile din  $R$  și  $S$ , atât înainte cât și după aplicarea regulii REZOLVĂ, trebuie să satisfacă:

$$\nu(x) = \nu(t).$$

- Pt. or. unificator  $\mu$  pt.  $x \doteq t$  observăm că:

$$(x \leftarrow t); \mu = \mu$$

- $((x \leftarrow t); \mu)(x) = \mu(t) = \mu(x)$
- $((x \leftarrow t); \mu)(y) = \mu(y)$ , or.  $y \neq x$

- Deci,

$\mu$  este un unificator pt. ec. din  $R$  și  $S$  înainte de REZOLVĂ

$\Leftrightarrow$

$\mu$  este un unificator pt. ec. din  $R$  și  $S$  după REZOLVĂ



# Corectitudinea algoritmului

## Lema 2

Variabilele care apar în partea stângă a ecuațiilor din  $S$  sunt distincte două câte două și nu mai apar în altă parte în  $S$  și  $R$ .

## Demonstrație

Exercițiu!

# Corectitudinea algoritmului

- Pres. că algoritmul de unificare se termină cu  $R = \emptyset$ .
- Fie  $x_i \doteq t_i$ ,  $i = 1, \dots, k$ , ecuațiile din  $S$ .
- Definim substituția:

$$\nu(x_i) = t_i, \text{ or. } i = 1, \dots, k.$$

- $\nu$  este corect definită (vezi Lema 2).
- Cum variabilele  $x_i$  nu apar în termenii  $t_i$ , deducem că  $\nu(t_i) = t_i = \nu(x_i)$ , or.  $i = 1, \dots, k$ .
- Deci  $\nu$  este unificator pentru  $U$  (vezi Lema 1).

# Corectitudinea algoritmului

## Lema 3

$\nu$  definit mai sus cf. algoritmului de unificare este cgu pentru  $U$ .

## Demonstrație

Exercițiu!



## Sistem de deducție pentru logica Horn



# Sistem de deducție *backchain*

## Sistem de deducție pentru clauze Horn

Pentru o mulțime  $T$  de clauze Horn, avem:

- **Axiome:** orice clauză din  $T$
- **Regula de deducție:** regula *backchain*

$$\frac{\theta(p_1) \quad \theta(p_2) \quad \dots \quad \theta(p_n) \quad (p_1 \wedge p_2 \wedge \dots \wedge p_n \rightarrow q)}{\theta(q')}$$

unde  $p_1 \wedge p_2 \wedge \dots \wedge p_n \rightarrow q \in T$ , iar  $\theta$  este cgu pentru  $q'$  și  $q$ .

# Sistem de deducție

Pentru o țintă  $q$ , trebuie să verificăm dacă  $q$  se poate unifica cu partea dreapta a unei clauze

$$p_1 \wedge \dots \wedge p_n \rightarrow q,$$

printr-un unificator  $\theta$ . Dacă da, vom verifica  $\theta(p_1), \dots, \theta(p_n)$ .

## Exemplu

Pentru ținta

$$\text{ancestor}(Q, \text{ken}),$$

putem folosi o clauză

$$\text{daughter}(X, Y) \rightarrow \text{ancestor}(Y, X)$$

cu unificatorul

$$\{Y/Q, X/\text{ken}\}$$

pentru a obține o nouă țintă

$$\text{daughter}(\text{ken}, Q).$$

# Puncte de decizie în programarea logica

Având doar această regulă, care sunt punctele de decizie în căutare?

- Ce clauză să alegem.

- Pot fi mai multe clauze a căror parte dreaptă se potrivește cu o țintă.
- Aceasta este o alegere de tip **SAU**: este suficient ca oricare din variante să reușească.

- Ordinea în care rezolvăm noile ținte.

- Aceasta este o alegere de tip **ȘI**: trebuie arătate toate țintele noi.
- Ordinea în care le rezolvăm poate afecta găsirea unei derivări, depinzând de strategia de cautare folosită.

# Strategia de căutare din Prolog

Strategia de căutare din Prolog este de tip *depth-first*,

- de sus în jos

- pentru alegerile de tip **SAU**
- alege clauzele în ordinea în care apar în program

- de la stânga la dreapta

- pentru alegerile de tip **ȘI**
- alege noile ținte în ordinea în care apar în clauza aleasă

# Proprietăți ale sistemului de inferență

## Vești bune!

- Regula *backchain* conduce la un sistem de deducție complet:

Pentru o mulțime de clauze  $T$  și o țintă  $Q$ ,

dacă  $T \models Q$ ,

atunci există o derivare a lui  $Q$  folosind regula *backchain*.

## Vești proaste!

- Strategia de căutare din Prolog este incompletă:

Poate eșua în a găsi o derivare, chiar și când există o derivare!

# Sistemul de inferență backchain

Notăm  $T \vdash_b Q$  dacă există o derivare a lui  $Q$  din  $T$  folosind sistemul de inferență *backchain*.

## Teoremă

*Sistemul de inferență backchain este corect și complet pentru formule atomice fără variabile  $Q$ .*

$$T \models Q \quad \text{dacă și numai dacă} \quad T \vdash_b Q$$

Sistemul de inferență *backchain* este corect și complet și pentru formule atomice cu variabile  $Q$ .

$$T \models \theta(Q) \quad \text{dacă și numai dacă} \quad T \vdash_b \theta(Q),$$

pentru o substituție  $\theta$ .

# Corectitudine

## Propoziție (Corectitudine)

Dacă  $T \vdash_b Q$ , atunci  $T \models Q$ .

## Demonstrație [schiță] [\*]

- Presupunem că toate clauzele din  $T$  sunt adevărate.
- Ne uităm, inductiv, la cazurile care pot să apară în derivarea lui  $Q$ .

□

# Completitudine

## Teoremă (Completitudine)

*Dacă  $T \models Q$ , atunci  $T \vdash_b Q$ .*

Trebuie să arătăm că

pentru orice structură și orice interpretare,  
dacă orice clauză din  $T$  este adevărată, atunci și  $Q$  este adevărată,



există o derivare a lui  $Q$  din  $T$ .



# Completitudine

Demonstrația este mai simplă deoarece

este suficient să ne uităm la o singură structură!

De ce? Vom da mai multe detalii peste câteva cursuri.

Vom construi o structură specială  $\mathcal{LH}$  astfel încât

$$T \vdash_b Q \quad \text{dacă} \quad \mathcal{LH} \models Q$$

# Modele Herbrand

Fie  $\mathcal{L}$  un limbaj de ordinul I.

- Presupunem că are cel puțin un simbol de constantă!
- Dacă nu are, adăugăm un simbol de constantă.

**Universul Herbrand** este mulțimea  $T_{\mathcal{L}}$  a tuturor termenilor lui  $\mathcal{L}$  fără variabile.

Un **model Herbrand** este o structură  $\mathcal{H} = (T_{\mathcal{L}}, \mathbf{F}^{\mathcal{H}}, \mathbf{P}^{\mathcal{H}}, \mathbf{C}^{\mathcal{H}})$ , unde

- pentru orice simbol de constantă  $c$ ,  $c^{\mathcal{H}} = c$
- pentru orice simbol de funcție  $f$  de aritate  $n$ ,  
 $f^{\mathcal{H}}(t_1, \dots, t_n) = f(t_1, \dots, t_n)$

# Modele Herbrand

Într-un model Herbrand, nu spunem nimic despre interpretarea simbolurilor de relații!

- pentru orice simbol de relație  $R$  de aritate  $n$ ,  $R^{\mathcal{H}}(t_1, \dots, t_n) \subseteq (T_{\mathcal{L}})^n$

## Exemplu

- Interpretăm toate simbolurile de predicate ca fiind adevărate peste tot.
- Adică, pentru orice simbol de relație  $R$  de aritate  $n$ ,  $R^{\mathcal{H}} = (T_{\mathcal{L}})^n$ .
- Această structură este model pentru orice mulțime de clauze definite.
- **Exercițiu:** De ce?

# Modele Herbrand

Definim o **ordine** între modelele Herbrand:

$\mathcal{H}_1 \leq \mathcal{H}_2$  este definită astfel:

pt. or. sb. de predicat  $R$  de aritate  $n$  și or. termeni  $t_1, \dots, t_n$   
dacă  $M_1 \models R(t_1, \dots, t_n)$ , atunci  $M_2 \models R(t_1, \dots, t_n)$

- Ne interesează **cel mai mic model Herbrand!**
- De ce există? Este unic?
- Folosim o construcție de punct fix pentru a stabili când sunt adevărate predicatele.

# Cel mai mic model Herbrand

- O **instanță de bază** a unei clauze  $Q_1(X_1) \wedge \dots \wedge Q_n(X_n) \rightarrow P(Y)$  este rezultatul obținut prin înlocuirea variabilelor cu termeni fără variabile.
- Pentru o mulțime de clauze  $T$ , o formulă atomică  $P$  și o mulțime de formule atomice  $X$ ,

$oneStep_T(P, X)$  este adevărat

dacă există o instanță de bază a unei clauze

$Q_1(X_1) \wedge \dots \wedge Q_n(X_n) \rightarrow P(Y)$  din  $T$  astfel încât  $P$  este instanța lui  $P(Y)$  și instanța lui  $Q_i(X_i)$  este în  $X$ , pentru orice  $i = 1, \dots, n$ .

- **Baza Herbrand**  $B_{\mathcal{L}}$  este mulțimea formulelor atomice fără variabile.
- Pentru o mulțime de clauze  $T$ , definim

$$f_T : \mathcal{P}(B_{\mathcal{L}}) \rightarrow \mathcal{P}(B_{\mathcal{L}})$$

$$f_T(X) = \{P \in B_{\mathcal{L}} \mid oneStep_T(P, X)\}$$

# Cel mai mic model Herbrand

## Exemplu

- Fie  $\mathcal{L}$  un limbaj cu un sb. de constantă  $0$ , un sb. de funcție unară  $s$  și un sb. de relație unară  $par$
- $T_{\mathcal{L}} = \{0, s(0), s(s(0)), \dots\}$
- Fie  $T$  mulțimea clauzelor:

$$par(0)$$

$$par(X) \rightarrow par(s(s(X)))$$

- Instance de bază:

- $par(0) \rightarrow par(s(s(0)))$

- $par(s(0)) \rightarrow par(s(s(s(0))))$

- $f_T(\{\}) = \{par(0)\}$
- $f_T(\{par(0)\}) = \{par(0), par(s(s(0)))\}$
- $f_T(\{par(s(0))\}) = \{par(0)\}$
- $f_T(\{par(s(s(0))))\} = \{par(0), par(s(s(s(s(0))))\}$

# Cel mai mic model Herbrand

- $f_T$  este monotonă, deci are un cel mai mic punct fix  $FP_T$ !
- $FP_T$  este reuniunea tuturor mulțimilor

$$f_T(\{\}), f_T(f_T(\{\})), f_T(f_T(f_T(\{\}))), \dots$$

- Pentru o mulțime de clauze  $T$ , definim **cel mai mic model Herbrand**  $\mathcal{LH}$  ca fiind un model Herbrand în care

un predicat  $R(t_1, \dots, t_n)$   
este adevărat      ddacă       $R(t_1, \dots, t_n) \in FP_T$ .

- $\mathcal{LH} \models T$ .  
Exercițiu: De ce?

# Cel mai mic model Herbrand

## Propoziție

Pentru orice formulă atomică  $Q$ ,

$$T \vdash_B Q \quad \text{dacă} \quad \mathcal{LH} \models Q.$$

## Demonstrație (schiță) $[*]$

- Implicația de la stânga la dreapta rezultă ușor din corectitudinea sistemului de inferență *backchain*.
- Implicația de la dreapta la stânga este mai complicată.
  - $Q$  apare în interpretările simbolurilor de predicate din  $\mathcal{LH}$
  - Deci  $Q$  este obținut după un număr finit  $n$  de aplicări ale lui  $f_T$
  - Se arată prin inducție după  $n$  că pentru fiecare formulă care apare prin aplicări ale lui  $f_T$  există o derivare în sistemul de inferență *backchain*.



# Completitudine

## Teoremă (Completitudine)

*Dacă  $T \models Q$ , atunci  $T \vdash_B Q$ .*

## Demonstrație

- Dacă  $T \models Q$ , atunci  $\mathcal{LH} \models Q$
- Deci  $T \vdash_B Q$

□



Pe săptămâna viitoare!