Logică și structuri discrete

Logica predicatelor

Marius Minea
marius@cs.upt.ro

http://www.cs.upt.ro/~marius/curs/lsd/

24 noiembrie 2015

Logică: recapitulare

Folosim logica pentru a *formaliza* raționamente
= a le exprima *riguros*

Logica ne permite să facem *demonstrații (deducții)*
din *axiome* (totdeauna adevărate)
și *ipoteze* (considerate adevărate în problema dată)
folosind *reguli de inferență* (de deducție)

$$\frac{p \qquad p \to q}{q} \qquad \textit{modus ponens}$$

Modus ponens e suficient pentru a formaliza logica propozițională
dar se pot defini și alte reguli de deducție valide.
Astfel putem simplifica demonstrațiile.

Logica propoziţională e insuficientă

Un exemplu clasic:

(1) Toţi oamenii sunt muritori.

(2) Socrate e om.

Deci, (3) Socrate e muritor.

Seamănă cu *modus ponens*

dar, premisa din (1) ("toţi oamenii")

nu e la fel cu (2) (Socrate, un anumit om)

În logica clasică: *silogisme* (anumite tipare de reguli de inferenţă)

Aristotel, stoici

nu le vom discuta aici

în logica modernă: *logica predicatelor* (logica de ordinul I)

Gottlob Frege, Charles Peirce (sec. 19)

Spre logica predicatelor

Revenim la exemplul:
(1) Toți oamenii sunt muritori.
(2) Socrate e om.
Deci, (3) Socrate e muritor.

Am putea reformula (1):
Dacă X e om, atunci X e muritor.
sau mai precis
Pentru orice X, dacă X e om, atunci X e muritor

$\Rightarrow$ avem nevoie de
variabile (X) care să ia valori într-un anumit univers
proprietăți (om, muritor) sau *relații* între variabile
funcții, pentru atributele unei variabile (ex. culoare, nota)
cuantificatori: universal (toți), existențial (unii)

Exemple: Ce vrem să exprimăm

Proprietăți mai complexe decât în logica propozițională:

$member(x, A) \rightarrow member(x, union(A, B))$ mulțimi

$leq(x, y) \rightarrow leq(f(x), f(y))$ funcție monotonă

$apel(nrX, nrY) \wedge eq(rețea(nrX), rețea(nrY)) \wedge prepay(nrX)$
$\rightarrow eq(cost(nrX, nrY), 0.11)$

$apel(nrX, nrY) \wedge fix(nrX) \wedge fix(nrY) \rightarrow eq(cost(nrX, nrY), 0.04)$

Avem:

variabile (x, y, nrX, nrY)
funcții (*union*, f, *rețea*, *cost*)
predicate (*member*, *leq*, *apel*, *prepay*, *fix*)
(egalitatea e un predicat considerată uneori separat)

Un *predicat* = o afirmație relativ la una sau mai multe variabile, care, dând valori variabilelor, poate lua valoarea adevărat sau fals.

Logica predicatelor (logica de ordinul întâi)

Precizăm întâi *simbolurile* limbajului:

- parantezele ( )
- conectorii $\neg$ și $\rightarrow$

nou cuantificatorul $\forall$ (universal)

- o mulțime de identificatori $v_0, v_1, \cdots$ pentru *variabile*
  nu neapărat propoziții, vor lua valori dintr-un *univers*

nou pentru orice $n \geq 1$ o mulțime de simboluri de *funcții* n-are

nou o mulțime (posibil vidă) de simboluri pentru *constante*
constantele pot fi privite și ca funcții de 0 argumente

nou pentru orice $n \geq 0$ o mulțime de simboluri de *predicate* n-are
propozițiile pot fi privite ca predicate de 0 argumente

Logica de ordinul întâi cu egalitate:
conține și $=$ ca simbol special pe lângă cele de mai sus.

Sintaxa: termeni și formule

Ca deobicei, noțiunile se definesc *structural recursiv*

Termenii

variabilă v sau constantă c

$f(t_1, \cdots, t_n)$ cu f funcție n-ară și $t_1, \cdots, t_n$ *termeni*

Formule (well-formed formulas, formule bine formate):

$P(t_1, \cdots, t_n)$	cu P predicat n-ar; $t_1, \cdots, t_n$ termeni
$\neg\alpha$	unde α este o formulă
$\alpha \to \beta$	unde α, β sunt formule
$\forall v\, \alpha$	cu v variabilă, α formulă: *cuantificare universală*
$t_1 = t_2$	cu t_1, t_2 termeni (în limbaje cu egalitate)

Față de logica propozițională, în loc de propoziții avem *predicate* (peste *termeni*).
Logica se numește *de ordinul I*, deoarece *cuantificatorii logici* se pot aplica doar variabilelor.
În logici *de ordin superior*, se poate cuantifica și peste predicate.

Reprezentare în ML

Termenii și formulele sunt definite structural recursiv
se pot traduce direct în tipuri recursive

```
type term = V of string
          | F of string * term list

type predform = Pr of string * term list
              | Neg of predform
              | And of predform * predform
              | Or of predform * predform
              | Forall of string * predform
```

Am ales să reprezentăm constantele ca funcții cu zero argumente.

Atât termenii cât și predicatele au argumente: listă de termeni.

Exemplu: $\forall x \neg \forall y\, P(x, f(y))$

```
Forall("x", Neg(Forall("y", Pr("P",[V "x"; F("f", [V "y"])])))))
```

Despre cuantificatori. Cuantificatorul existențial $\exists$

Notăm: $\exists x\varphi \stackrel{def}{=} \neg\forall x(\neg\varphi)$
Există x pentru care ϕ e adevărată $\leftrightarrow$ nu pentru orice x ϕ e falsă.

Cei doi cuantificatori sunt *duali*. Putem scrie și $\forall x\varphi = \neg\exists x(\neg\varphi)$

Cuantificatorii au precedență mai mare decât conectorii $\neg$, $\wedge$, $\rightarrow$ ...

Un *punct* . indică aplicarea cuantificării la tot restul formulei,
până la sfârșit sau paranteză închisă (evită parantezele inutile).
$(\exists x.P(x) \rightarrow Q(x)) \wedge R(x)$ înseamnă $(\exists x(P(x) \rightarrow Q(x))) \wedge R(x)$

În formula $\forall v\varphi$ (sau $\exists v\varphi$) variabila v se numește *legată*
Variabilele care nu sunt legate se numesc *libere*

O variabilă poate fi liberă și legată în aceeași formulă.
Mai sus, x e legată în primul conjunct $P(x) \rightarrow Q(x)$
și e liberă în $R(x)$ (e în afara cuantificatorului)

Variabile libere și legate

Înțelesul unei formule *nu depinde* de variabilele legate
- înțelesul lor e "legat" de cuantificator ("pentru orice", "există")
- pot fi redenumite, fără a schimba înțelesul formulei

O formulă fără variabile libere are înțeles de sine stătător.

Rol similar: parametrii formali la funcții în limbaje de programare
- putem să îi redenumim fără a schimba efectul funcției
- `fun x -> x + 3` și `fun y -> y + 3` sunt aceeași funcție

Interpretarea unei formule *depinde* de variabilele sale libere
- (ce valoare primesc; discutăm la semantica formulelor)

La fel și `fun x -> x + y`
- nu are înțeles de sine stătător (`y` se presupune declarat anterior)
- înțelesul/efectul depinde de definiția lui `y`

Formalizarea limbajului natural

1. Fiecare investitor a cumpărat acțiuni sau obligațiuni.
2. Dacă indicele Dow Jones scade, toate acțiunile mai puțin aurul scad.
3. Dacă trezoreria crește dobânda, toate obligațiunile scad.
4. Orice investitor care a cumpărat ceva care scade nu e bucuros.
5. Dacă indicele Dow Jones scade și trezoreria crește dobânda,
toți investitorii bucuroși au cumpărat ceva acțiuni de aur.

Exemplu: `http://www.cs.utexas.edu/users/novak/reso.html`

Verbele devin *predicate* (ca în limbajul natural):
cumpără, scade, crește, ...

Subiectul și *complementele* (in)directe: *argumentele* predicatului
investitor, ceea ce cumpără (acțiuni, obligațiuni)

Atributele (proprietăți) sunt *predicate* despre entități (*argumente*)
bucuros (investitor), de aur (acțiune)

Categoriile devin predicate, cu argument entitatea din categorie
e acțiune, e obligațiune (ce se cumpără)

iO *frază* e un predicat (0 argumente) dacă verbul apare doar în ea
trezoreria crește dobânda ("crește" apare doar aici)

Exemplu de formalizare

1. Fiecare investitor a cumpărat acțiuni sau obligațiuni.

Două entități: investitorul, ce cumpără (cu două categorii)
Introducem un predicat $inv(X)$ (X e investitor)

$$inv(X) \rightarrow cumpără(X, C) \wedge (acțiune(C) \vee oblig(C))$$

Vrem formule *fără variabile libere* (independente de context)
X e cuantificat universal (*fiecare* investitor)
C e cuantificat existențial (investitorul cumpără *ceva*)

$$\forall X.inv(X) \rightarrow \exists\, C.cumpără(X, C) \wedge (acțiune(C) \vee oblig(C))$$

2. Dacă indicele Dow Jones scade, toate acțiunile mai puțin aurul scad.

$$scade(dj) \rightarrow \forall\, X.acțiune(X) \wedge \neg aur(X) \rightarrow scade(X)$$

Indicele Dow Jones e o noțiune unică $\Rightarrow$ folosim o *constantă dj*
Putem și folosi o propoziție *scadedj* (celelalte lucruri care scad sunt acțiuni, diferite de indicele general)

Exemplu de formalizare (cont.)

3. Dacă trezoreria crește dobânda, toate obligațiunile scad.

$$\mathit{creștedob} \to \forall X.\mathit{oblig}(X) \to \mathit{scade}(X)$$

Dobânda e unicul lucru care crește $\Rightarrow$ predicat fără parametri

Alternativ: o constantă *dobânda*

4. Orice investitor care a cumpărat ceva care scade nu e bucuros.

$$\forall X.\mathit{inv}(X) \to (\exists C.\mathit{cumpără}(X, C) \wedge \mathit{scade}(C)) \to \neg \mathit{bucuros}(X)$$

$\to$ asociază la dreapta, $p \to q \to r = p \to (q \to r) = p \wedge q \to r$, echivalent:

$$\forall X.\mathit{inv}(X) \wedge (\exists C.\mathit{cumpără}(X, C) \wedge \mathit{scade}(C)) \to \neg \mathit{bucuros}(X)$$

5. Dacă indicele Dow Jones scade și trezoreria crește dobânda, toți investitorii bucuroși au cumpărat ceva acțiuni de aur.

$$\mathit{scade}(\mathit{dj}) \wedge \mathit{creștedob} \to$$
$$\forall X.\mathit{inv}(X) \wedge \mathit{bucuros}(X) \to \exists C.\mathit{cumpără}(X, C) \wedge \mathit{acțiune}(C) \wedge \mathit{aur}(C)$$

Formalizarea cuantificatorilor

cuantificatorul universal ("toți") cuantifică o implicație:

Toți studenții sunt tineri
Studenți $\subseteq$ *Tineri*

$$\forall x(\mathit{student}(x) \rightarrow \mathit{tânăr}(x))$$

cuantificatorul existențial ("unii", "există") cuantifică o conjuncție

Unii tineri sunt studenți
Studenți $\cap$ *Tineri* $\neq \emptyset$

$$\exists x(\mathit{tânăr}(x) \wedge \mathit{student}(x))$$

Cuantificatorul *universal* e *distributiv față de conjuncție*:

$$\forall x(P(x) \wedge Q(x)) \leftrightarrow \forall x\, P(x) \wedge \forall x\, Q(x)$$

dar cuantificatorul *existențial* NU ~~e distributiv~~ față de conjuncție:

$$\exists x(P(x) \wedge Q(x) \not\leftrightarrow (\exists x\, P(x) \wedge \exists x\, Q(x))$$

(avem implicație, dar nu și invers, poate să nu fie același x !)

Dual, $\exists$ e distributiv față de disjuncție, $\forall$ nu e. Avem doar:

$$\exists x\, P(x) \vee \exists x\, Q(x) \rightarrow \exists x.P(x) \vee Q(x)$$

Consistență și completitudine

Ca și în logica propozițională:

demonstrația se face pur sintactic

determinarea *adevărului*: semantic, considerând *interpretări*

Dar: avem o infinitate de interpretări $\Rightarrow$ nu putem verifica toate

$\Rightarrow$ e important să putem *demonstra*

Calculul predicatelor de ordinul întâi este *consistent* și *complet*
(la fel ca și logica propozițională):

$$H \vdash \varphi \text{ dacă și numai dacă } H \models \varphi$$

H e o mulțime de ipoteze

Orice teoremă e validă (adevărată în toate interpretările/atribuirile).
Orice formulă validă (tautologie) poate fi demonstrată (e teoremă).

Dar: relația de implicație logică e doar *semidecidabilă*

dacă o formulă e o tautologie, ea poate fi demonstrată

dar dacă nu e validă, încercarea de a o demonstra (sau o refuta)
poate să continue la nesfârșit

Rezoluția: de la propoziții la predicate

În logica propozițională, rezoluția e o *regulă de inferență*: produce o nouă clauză din două clauze cu literale complementare (p și $\neg p$):

$$\frac{p \vee \alpha \qquad \neg p \vee \beta}{\alpha \vee \beta} \quad \textit{rezoluție}$$

Rezoluția e o regulă de inferență *validă*:

dacă premisele sunt adevărate, și concluzia e adevărată

dacă concluzia e contradicție, premisele formează o contradicție

Folosim rezoluția pentru a arăta că o formulă e o *contradicție*.

e o metodă de *refutație* (respingere a unei afirmații)

Demonstrăm o teoremă (tautologie) prin *reducere la absurd* arătând prin *rezoluție* că *negația ei e o contradicție* (nerealizabilă).

Pentru a demonstra că ipotezele A_k implică împreună concluzia C:

$$A_1 \wedge A_2 \ldots \wedge A_n \rightarrow C \quad \text{e tautologie}$$

arătăm că *negația* ei $\neg(A_1 \wedge A_2 \ldots \wedge A_n \rightarrow C)$ e o *contradicție*

$$A_1 \wedge A_2 \ldots \wedge A_n \wedge \neg C \quad \text{e contradicție}$$

(reducere la absurd: ipoteze adevărate + concluzia falsă)

De ce substituția și unificarea de termeni ?

În logica predicatelor, un *literal* nu e o propoziție, ci un *predicat*
nu mai avem p și $\neg p$, ci $P(\ldots)$ și $\neg P(\ldots)$
$\Rightarrow$ trebuie să analizăm și argumentele predicatului

Avem două formule în care un predicat apare pozitiv și negativ:
$\forall x.\forall y.P(x, g(y))$ și $\forall z.\neg P(z, a)$.
sau
$\forall x.\forall y.P(x, g(y))$ și $\forall z.\neg P(a, z)$
Se contrazic ?

Putem *substitui* o variabilă cuantificată universal cu *orice* termen
$\Rightarrow$ în al doilea caz, putem substitui $x \mapsto a$, $z \mapsto g(y)$
$\Rightarrow$ obținem $P(a, g(y))$ și $\neg P(a, g(y))$, *contradicție*

În primul caz, nu putem substitui y și obține a din $g(y)$
interpretare: nu putem presupune că funcția arbitrară g
ia obligatoriu și valoarea constantă a.

Definim precis acest lucru: noțiunile de *substituție* și *unificare*

Substituții de termeni

O *substituție* e o *funcție* care asociază unor *variabile* niște *termeni*:

$$\{x_1 \mapsto t_1, \ \ldots, x_n \mapsto t_n\}$$

De exemplu $\{x \mapsto g(y), y \mapsto f(b), t \mapsto u\}$

Obs: se întâlnesc și notațiile x_i/t_i, sau invers, t_i/x_i

Substituția σ pe termenul T se notează uzual postfix: $T\sigma$

De exemplu $f(x, g(y, z), a, t)\{x \mapsto g(y), y \mapsto f(b), t \mapsto u\}$

$= f(g(y), g(f(b), z), a, u)$

Compunerea a două substituții e o substituție.

Unificare de termeni

Doi termeni t_1 și t_2 se pot *unifica* dacă există o substituție σ care îi face egali: $t_1\sigma = t_2\sigma$.
O astfel de substituție se numește *unificator*.

Exemplu: $f(x, g(y))\{x \mapsto a\} = f(a, g(y)) = f(a, z)\{z \mapsto g(y)\}$
deci substituția $\{x \mapsto a, z \mapsto g(y)\}$ e un *unificator* .

Mai general: avem o mulțime de ecuații (perechi de termeni)
$\{l_1 = r_1, l_2 = r_2, \ldots, l_n = r_n\}$ (numită și *problemă de unificare*).
Un unificator (o soluție a problemei) e o substituție σ astfel încât $l_i\sigma = r_i\sigma$ pentru orice i.

Cel mai general unificator (most general unifier) este acela din care orice alt unificator poate fi obținut aplicând încă o substituție.

În *rezoluție*: având clauzele $P(l_1, l_2, \ldots l_n)$ și $\neg P(r_1, r_2, \ldots r_n)$
dacă găsim un *unificator*, avem o *contradicție*.
(substituind, obținem același predicat și adevărat și fals).

Reguli de unificare

O *variabilă* x poate fi unificată cu orice *termen* t (substituție)
dacă x *nu apare* în t dar nu: x cu $f(g(y), h(x, z))$
pentru că altfel, substituția ar duce la un termen infinit

Două *constante* pot fi unificate doar dacă sunt *identice*

Doi *termeni* $f(...)$ pot fi unificați doar dacă au funcții identice, și *argumentele* (termeni) pot fi unificate unul câte unul

$\Rightarrow$ cu aceste reguli, se poate scrie un algoritm recursiv de unificare care determină *cel mai general unificator*
(orice alt unificator se poate obține din el printr-o altă substituție)

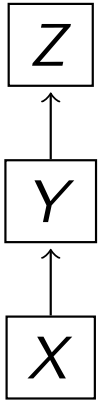
Union-Find

Structură de date+algoritm pentru lucru cu clase de echivalență.

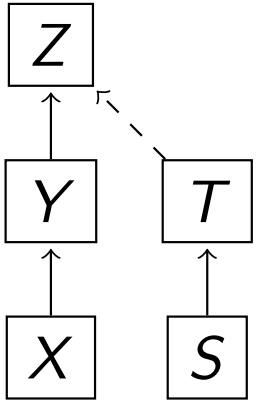
Operații:
find(element): găsește reprezentantul clasei de echivalență
union(elem1, elem2): declară elementele ca fiind echivalente
(și rămân așa mai departe)

Implementare: pădure de *arbori* cu legături de la fiu la părinte
find: returnează rădăcina (chiar nodul, dacă e singur)
union: leagă rădăcina unuia de rădăcina celuilalt

Exemplu pentru Union-Find



$find(X) = find(Y)$
$= find(Z) = Z$



$union(Y, S)$
leagă $find(Y)$ și $find(S)$

Rezoluția în calculul predicatelor

Fie două clauze A și B, și un predicat P.

Redenumim variabilele din B ca să nu fie comune cu A
(fiecare clauză are spațiul ei de variabile)

Alegem literale $P_1, ..., P_k \in A$ și $\neg P_{k+1}, ...\neg P_{k+l} \in B$ cu pred. P

Unificăm termenii $\{P_1, \ldots, P_k, P_{k+1}, \ldots P_{k+l}\}$
Fie σ unificatorul cel mai general rezultat.

Formăm clauza $(A \cup B \setminus \{P_1, \ldots P_k, P_{k+1}, \ldots P_{k+l}\})$, îi aplicăm substituția σ, și o adăugăm la mulțimea clauzelor.

Dacă repetând obținem clauza vidă, formula inițială nu e realizabilă.
Dacă nu mai putem crea rezolvenți noi, formula inițială e realizabilă.

Metoda rezoluției e *completă* relativ la refutație
pentru orice formulă nerealizabilă, va ajunge la clauza vidă
dar nu poate *determina* realizabilitatea *oricărei formule*
(există formule pentru care rulează la infinit)

Exemplu de aplicare a rezoluției

Reluăm exercițiul formalizat anterior.
Folosim () și nu . pentru a nu greși la ce se aplică cuantificatorii.

A_1: $\forall X(inv(X) \rightarrow \exists\, C(cump(X, C) \wedge (act(C) \vee oblig(C))))$

A_2: $scadedj \rightarrow \forall\, X(act(X) \wedge \neg aur(X) \rightarrow scade(X))$

A_3: $creștedob \rightarrow \forall\, X(oblig(X) \rightarrow scade(X))$

A_4: $\forall X(inv(X) \rightarrow (\exists\, C(cump(X, C) \wedge scade(C)) \rightarrow \neg bucuros(X)))$

C: $scadedj \wedge creștedob \rightarrow$
$\forall\, X(inv(X) \wedge bucuros(X) \rightarrow \exists\, C(cump(X, C) \wedge act(C) \wedge aur(C)))$

Pentru a demonstra prin *reducere la absurd*, aplicând rezoluția, că $A_1 \wedge A_2 \wedge A_3 \wedge A_4 \rightarrow C$, *negăm concluzia*:

$\neg C$: $scadedj \wedge creștedob \wedge$
$\neg\forall\, X(inv(X) \wedge bucuros(X) \rightarrow \exists\, C(cump(X, C) \wedge act(C) \wedge aur(C)))$

Negăm concluzia *la început*, înainte de a transforma cuantificatorii!

Transformăm implicația, ducem negația până la predicate

Transformăm implicația: $A \to B = \neg A \vee B$, $\neg(A \to B) = A \wedge \neg B$
Ducem $\neg$ *înăuntru*: $\neg\forall x P(x) = \exists x \neg P(x) \quad \neg\exists x P(x) = \forall x \neg P(x)$
ATENȚIE! În $\forall x(...)$, când transformăm $\to$ etc. *înăuntrul* $(...)$
NU se schimbă cuantificatorul care e *în afară* (nici la $\exists x$)

$\forall X(\neg inv(X) \vee \exists C(cump(X, C) \wedge (act(C) \vee oblig(C))))$

$\neg scadedj \vee \forall X(\neg act(X) \vee aur(X) \vee scade(X))$

$\neg creștedob \vee \forall X(\neg oblig(X) \vee scade(X))$

$\forall X(\neg inv(X) \vee \neg\exists C(cump(X, C) \wedge scade(C)) \vee \neg bucuros(X))$
devine
$\forall X(\neg inv(X) \vee \forall C(\neg cump(X, C) \vee \neg scade(C)) \vee \neg bucuros(X))$

$scadedj \wedge creștedob \wedge$
$\exists X(inv(X) \wedge bucuros(X) \wedge \neg\exists C(cump(X, C) \wedge act(C) \wedge aur(C)))$
devine $scadedj \wedge creștedob \wedge$
$\exists X(inv(X) \wedge bucuros(X) \wedge \forall C(\neg cump(X, C) \vee \neg act(C) \vee \neg aur(C)))$

Skolemizare: eliminăm cuantificatorii existențiali

Redenumim variabilele cuantificate cu *nume unic* în fiecare formulă, pentru a putea elimina ulterior cuantificatorii. De ex.
$\forall x\, P(x) \vee \exists x\, \forall y\, Q(x, y)$ devine $\forall x\, P(x) \vee \exists z\, \forall y\, Q(z, y)$

Eliminăm cuantificatorii existențiali (*skolemizare*):
Pentru $\exists y$ în *interiorul* lui $\forall x_1 \ldots \forall x_n$, introducem o *funcție Skolem* $y = g(x_1, \ldots, x_n)$: valoarea lui y depinde de $x_1, \ldots x_n$

$\forall X(\neg inv(X) \vee \exists\, C(cump(X, C) \wedge (act(C) \vee oblig(C))))$ devine
$\forall X(\neg inv(X) \vee (cump(X, f(X)) \wedge (act(f(X)) \vee oblig(f(X)))))$
acel C care există depinde de $X \Rightarrow$ alegem o nouă funcție $f(X)$

Atenție! în fiecare formulă se aleg *noi funcții* Skolem, nu aceleași!

Pentru $\exists y$ în *exterior*, se alege o nouă *constantă Skolem*

$scadedj \wedge crește dob \wedge \exists\, X(inv(X) \wedge bucuros(X)$
$\wedge \forall\, C(\neg cump(X, C) \vee \neg act(C) \vee \neg aur(C)))$
în loc de $\exists X$ alegem pentru X o nouă constantă b:
$scadedj \wedge crește dob \wedge inv(b) \wedge bucuros(b)$
$\wedge \forall C(\neg cump(b, C) \vee \neg act(C) \vee \neg aur(C))$

Eliminăm cuantificatorii universali

În acest moment, toți cuantificatorii universali pot fi duși în față (*forma normală prenex*), și apoi eliminați (implicit, variabilă = orice)

$\neg inv(X) \vee (cump(X, f(X)) \wedge (act(f(X)) \vee oblig(f(X))))$

$\neg scadedj \vee \neg act(X) \vee aur(X) \vee scade(X)$

$\neg creștedob \vee \neg oblig(X) \vee scade(X)$

$\neg inv(X) \vee \neg cump(X, C) \vee \neg scade(C) \vee \neg bucuros(X)$

$scadedj \wedge creștedob \wedge inv(b) \wedge bucuros(b)$
$\wedge(\neg cump(b, C) \vee \neg act(C) \vee \neg aur(C))$

Scriem forma clauzală

Ducem conjuncția în exterior (prin distributivitate) și scriem fiecare clauză separat (*formă clauzală*)

(1) $\neg inv(X) \vee cump(X, f(X))$

(2) $\neg inv(X) \vee act(f(X)) \vee oblig(f(X)))$

(3) $\neg scadedj \vee \neg act(X) \vee aur(X) \vee scade(X)$

(4) $\neg creștedob \vee \neg oblig(X) \vee scade(X)$

(5) $\neg inv(X) \vee \neg cump(X, C) \vee \neg scade(C) \vee \neg bucuros(X)$

(6) $scadedj$

(7) $creștedob$

(8) $inv(b)$

(9) $bucuros(b)$

(10) $\neg cump(b, C) \vee \neg act(C) \vee \neg aur(C)$

Generăm rezolvenți până la clauza vidă

Căutăm predicate $P(...)$ și $\neg P(...)$ și unificăm, obținând rezolvenții:

(11) $\neg act(X) \vee aur(X) \vee scade(X)$ (3, 6)

(12) $\neg cump(b, C) \vee \neg act(C) \vee scade(C)$ (10, 11, $X = C$)

(13) $\neg oblig(X) \vee scade(X)$ (4, 7)

Când unificăm, redenumim clauzele să nu aibe variabile comune:

(13) $\neg oblig(Y) \vee scade(Y)$ unificăm cu (2), are X

(14) $\neg inv(X) \vee act(f(X)) \vee scade(f(X))$ (2, 13)

(15) $\neg cump(b, f(X)) \vee \neg inv(X) \vee scade(f(X))$ (12, 14)

(16) $\neg cump(b, C) \vee \neg scade(C) \vee \neg bucuros(b)$ (5, 8)

(17) $\neg cump(b, C) \vee \neg scade(C)$ (9, 16)

(18) $\neg cump(b, f(X)) \vee \neg inv(X)$ (15, 17, $C = f(X)$)

(19) $\neg inv(b)$ (1, 18, $X = b$)

(20) $\emptyset$ (contradicție = succes în reducerea la absurd) (8, 19)

Programare logică: Prolog

```
fiu(ion, petre).
fiu(george, ion).
fiu(radu, ion).
fiu(petre, vasile).
desc(X, Y) :- fiu(X, Y).
desc(X, Z) :- fiu(X, Y), desc(Y, Z).
```

fapte (predicate adevărate)

reguli: `:-` e implicație $\leftarrow$
scrisă: stânga *dacă* dreapta
virgula e conjuncție $\wedge$

X e descendentul lui Y dacă X e fiul lui Y.

Variabilele din stânga sunt cuantificate *universal* (în ambele părți)
Variabilele care apar doar în dreapta sunt cuantificate *existențial*.

X e descendent din Z dacă există Y astfel încât X e fiul lui Y
și Y e descendent din Z

Regula 2: $\forall X \forall Z . desc(X, Z) \leftarrow \exists Y . fiu(X, Y) \wedge desc(Y, Z)$

$\forall X \forall Z . desc(X, Z) \vee \neg \exists Y . fiu(X, Y) \wedge desc(Y, Z)$

$\forall X \forall Z . desc(X, Z) \vee \forall Y \neg (fiu(X, Y) \wedge desc(Y, Z))$

Cuantificatorii universali se pot elimina, rezultă:

$$desc(X, Z) \vee \neg fiu(X, Y) \vee \neg desc(Y, Z)$$

Rezoluția în Prolog

Fie ca întrebare/scop/țintă (engl. *goal*) `desc(X, vasile).`
O *soluție* = o valoare pentru X care face predicatul adevărat.
O formulă e *realizabilă* dacă *negația* e o *contradicție*.

Rezoluția în Prolog

Fie ca întrebare/scop/țintă (engl. *goal*) `desc(X, vasile).`
O *soluție* = o valoare pentru X care face predicatul adevărat.
O formulă e *realizabilă* dacă *negația* e o *contradicție*.

Scriem negația întrebării: $\neg$ `desc(X, vasile).`
adică: `desc(X, vasile)` e *fals* pentru orice `X`.
Încercăm să derivăm o contradicție cu ipotezele folosind *rezoluția*.

Rezoluția în Prolog

Fie ca întrebare/scop/țintă (engl. *goal*) `desc(X, vasile).`
O *soluție* = o valoare pentru X care face predicatul adevărat.
O formulă e *realizabilă* dacă *negația* e o *contradicție*.

Scriem negația întrebării: $\neg$ `desc(X, vasile).`
adică: `desc(X, vasile)` e *fals* pentru orice `X`.
Încercăm să derivăm o contradicție cu ipotezele folosind *rezoluția*.

Alegem pentru unificare prima regulă (redenumind cu variabile noi):
`desc(X1, Y1)` $\vee \neg$ `fiu(X1, Y1).`
Ca rezolvent, obținem $\neg$ `fiu(X, vasile).` `X1=X, Y1=vasile`

Rezoluția în Prolog

Fie ca întrebare/scop/țintă (engl. *goal*) `desc(X, vasile).`
O *soluție* = o valoare pentru X care face predicatul adevărat.
O formulă e *realizabilă* dacă *negația* e o *contradicție*.

Scriem negația întrebării: $\neg$ `desc(X, vasile).`
adică: `desc(X, vasile)` e *fals* pentru orice `X`.
Încercăm să derivăm o contradicție cu ipotezele folosind *rezoluția*.

Alegem pentru unificare prima regulă (redenumind cu variabile noi):
`desc(X1, Y1)` $\vee \neg$ `fiu(X1, Y1).`
Ca rezolvent, obținem $\neg$ `fiu(X, vasile).` `X1=X, Y1=vasile`

Alegem pentru unificare faptul `fiu(petre, vasile)` (nr. 4).
Obținem ca rezolvent clauza vidă (contradicție) `X=petre`

Rezoluția în Prolog

Fie ca întrebare/scop/țintă (engl. *goal*) `desc(X, vasile).`
O *soluție* = o valoare pentru X care face predicatul adevărat.
O formulă e *realizabilă* dacă *negația* e o *contradicție*.

Scriem negația întrebării: $\neg$ `desc(X, vasile).`
adică: `desc(X, vasile)` e *fals* pentru orice `X`.
Încercăm să derivăm o contradicție cu ipotezele folosind *rezoluția*.

Alegem pentru unificare prima regulă (redenumind cu variabile noi):
`desc(X1, Y1)` $\vee$ $\neg$ `fiu(X1, Y1).`
Ca rezolvent, obținem $\neg$ `fiu(X, vasile).` `X1=X, Y1=vasile`

Alegem pentru unificare faptul `fiu(petre, vasile)` (nr. 4).
Obținem ca rezolvent clauza vidă (contradicție) `X=petre`

Deci, `desc(X, vasile)` *NU e fals* pentru orice `X`.
`desc(petre, vasile)` e *adevărat*. `X=petre` e o soluție.

Rezoluția în Prolog

Fie ca întrebare/scop/țintă (engl. *goal*) `desc(X, vasile).`
O *soluție* = o valoare pentru X care face predicatul adevărat.
O formulă e *realizabilă* dacă *negația* e o *contradicție*.

Scriem negația întrebării: $\neg$ `desc(X, vasile).`
adică: `desc(X, vasile)` e *fals* pentru orice `X`.
Încercăm să derivăm o contradicție cu ipotezele folosind *rezoluția*.

Alegem pentru unificare prima regulă (redenumind cu variabile noi):
`desc(X1, Y1)` $\vee \neg$ `fiu(X1, Y1).`
Ca rezolvent, obținem $\neg$ `fiu(X, vasile).` `X1=X, Y1=vasile`

Alegem pentru unificare faptul `fiu(petre, vasile)` (nr. 4).
Obținem ca rezolvent clauza vidă (contradicție) `X=petre`

Deci, `desc(X, vasile)` *NU e fals* pentru orice `X`.
`desc(petre, vasile)` e *adevărat*. `X=petre` e o soluție.

Mai există și alte soluții... (cont.)

Exemplu Prolog (cont.)

Pornim tot cu negația întrebării: ¬`desc(X, vasile)`.

Exemplu Prolog (cont.)

Pornim tot cu negația întrebării: ¬`desc(X, vasile)`.

Unificăm cu regula 2 (redenumind din nou variabilele):

`desc(X2, Z2)` $\vee \neg$ `fiu(X2, Y2)` $\vee \neg$ `desc(Y2, Z2)`

Obținem: $\neg$ `fiu(X, Y2)` $\vee \neg$ `desc(Y2, vasile)` `X2=X, Z2=vasile`

Exemplu Prolog (cont.)

Pornim tot cu negația întrebării: ¬`desc(X, vasile)`.

Unificăm cu regula 2 (redenumind din nou variabilele):
`desc(X2, Z2)` $\vee \neg$ `fiu(X2, Y2)` $\vee \neg$ `desc(Y2, Z2)`
Obținem: $\neg$ `fiu(X, Y2)` $\vee \neg$ `desc(Y2, vasile)` `X2=X, Z2=vasile`

Unificăm cu `fiu(ion, petre)` (nr. 1) `X=ion, Y2=petre`
Obținem rezolventul $\neg$ `desc(petre, vasile)`.

Exemplu Prolog (cont.)

Pornim tot cu negația întrebării: ¬`desc(X, vasile)`.

Unificăm cu regula 2 (redenumind din nou variabilele):
`desc(X2, Z2)` $\vee \neg$ `fiu(X2, Y2)` $\vee \neg$ `desc(Y2, Z2)`
Obținem: $\neg$ `fiu(X, Y2)` $\vee \neg$ `desc(Y2, vasile)` `X2=X, Z2=vasile`

Unificăm cu `fiu(ion, petre)` (nr. 1) `X=ion, Y2=petre`
Obținem rezolventul $\neg$ `desc(petre, vasile)`.

Am obtinut înainte `desc(petre, vasile)` $\Rightarrow$ derivăm clauza vidă.
$\Rightarrow$ `X=ion` e încă o soluție pentru întrebarea inițială.

Exemplu Prolog (cont.)

Pornim tot cu negația întrebării: ¬`desc(X, vasile)`.

Unificăm cu regula 2 (redenumind din nou variabilele):
`desc(X2, Z2)` $\vee \neg$ `fiu(X2, Y2)` $\vee \neg$ `desc(Y2, Z2)`
Obținem: $\neg$ `fiu(X, Y2)` $\vee \neg$ `desc(Y2, vasile)` `X2=X, Z2=vasile`

Unificăm cu `fiu(ion, petre)` (nr. 1) `X=ion, Y2=petre`
Obținem rezolventul $\neg$ `desc(petre, vasile)`.

Am obtinut înainte `desc(petre, vasile)` $\Rightarrow$ derivăm clauza vidă.
$\Rightarrow$ `X=ion` e încă o soluție pentru întrebarea inițială.

Dacă întrebarea are variabile, interpretorul Prolog va genera toate soluțiile posibile (toate unificările/substituțiile pentru variabile).

Altfel, determină dacă predicatul dat (fără variabile) e adevărat.

Exemplu Prolog: inversarea listelor

Noțiunile recursive se scriu similar în logică și programare funcțională.
Pentru liste, folosim constanta `nil` și constructorul `cons`.
Inversarea devine *predicat* cu 3 argumente: lista, acumulator, rezultat.

```
rev3(nil, R, R).
rev3(cons(H, T), Ac, R) :- rev3(T, cons(H, Ac), R).
rev(L, R) :- rev3(L, nil, R)
```

Când lista e vidă, rezultatul e acumulatorul.
Altfel, adăugând capul listei la acumulator, obținem același rezultat.
Inversarea se obține luând acumulatorul (auxiliar) lista vidă.

Dând ca întrebare $rev(c(1, c(2, c(3, nil))), X)$ obținem
$X = c(3, c(2, c(1, nil)))$, derivarea (raționamentul) fiind:

$rev(c(1, c(2, c(3, nil))), X)$	`L1=c(1,c(2,c(3,nil))), R1=X`
$\leftarrow rev3(c(1, c(2, c(3, nil))), nil, X)$	`H1=1, T1=c(2,c(3,nil)), Ac1=nil`
$\leftarrow rev3(c(2, c(3, nil)), c(1, nil), X)$	`H2=2, T2=c(3,nil), Ac2=c(1,nil)`
$\leftarrow rev3(c(3, nil), c(2, c(1, nil)), X)$	`H3=3, T3=nil, Ac3=c(2,c(1,nil))`
$\leftarrow rev3(nil, c(3, c(2, c(1, nil))), X)$	`X=c(3,c(2,c(1,nil)))`

Rezoluție și programare logică

Prolog folosește un caz particular de rezoluție: *clauzele Horn*
= clauze cu *cel mult un literal pozitiv*

clauze definite: $p_i \leftarrow h_1 \wedge ... \wedge h_k$ (implicație)
se transformă în $p_i \vee \neg h_1 \vee ... \vee \neg h_k$
fapte: p_j (se afirmă, ipoteze)
întrebare/scop/țintă (*goal*): $false \leftarrow g_1 \wedge ... \wedge g_m$
se transformă în $\neg g_1 \vee ... \vee \neg g_m$
(arată prin contradicție că $g_1 \wedge ... \wedge g_m$ e adevărată)

Pentru astfel de clauze, există o metodă de rezoluție mai simplă:
se pornește de la țintă
se înlocuiește (cu unificare) pe rând fiecare scop parțial g_j
cu conjuncția premiselor care îl fac adevărat
privit ca rezoluție: se unifică un scop g_j cu o concluzie p_i,
rezultă înlocuirea cu $\neg h_1 \vee ... \vee \neg h_k$

Revedem: transformarea în formă clauzală

Similar cu logica de ordinul I, cu pași în plus pentru cuantificatori

Fie $\forall x[\neg P(x) \rightarrow \exists y(D(x,y) \wedge \neg(E(f(x),y) \vee E(x,y))] \wedge \neg\forall x P(x)$

(1) Eliminăm toți conectorii în afară de $\wedge$, $\vee$, $\neg$:
$\forall x[\neg\neg P(x) \vee \exists y(D(x,y) \wedge \neg(E(f(x),y) \vee E(x,y)))] \wedge \neg\forall x P(x)$

(2) Ducem negațiile înăuntru, până la predicate:
$\forall x[P(x) \vee \exists y(D(x,y) \wedge \neg E(f(x),y) \wedge \neg E(x,y))] \wedge \exists x \neg P(x)$

(3) Redenumim variabilele cuantificate, cu nume unice în formulă
$\forall x[P(x) \vee \exists y(D(x,y) \wedge \neg E(f(x),y) \wedge \neg E(x,y))] \wedge \exists z \neg P(z)$

(4) Eliminăm cuantificatorii existențiali (*skolemizare*)
Pentru $\exists y$ în *interiorul* lui $\forall x_1 \ldots \forall x_n$, introducem o *funcție* Skolem
$y = g(x_1, \ldots, x_n)$: valoarea lui y depinde de $x_1, \ldots x_n$ aici $g(x)$
Pentru $\exists y$ în *exterior*, se alege o nouă *constantă* Skolem aici a
$\forall x[P(x) \vee (D(x,g(x)) \wedge \neg E(f(x),g(x)) \wedge \neg E(x,g(x)))] \wedge \neg P(a)$

Forma clauzală (cont.)

(5) Aducem la *forma normală prenex*: cuantificatorii $\forall$ în față

$$\forall x([P(x) \vee (D(x, g(x)) \wedge \neg E(f(x), g(x)) \wedge \neg E(x, g(x)))] \wedge \neg P(a))$$

(6) Eliminăm prefixul cu cuantificatorii universali (devin impliciți)

$$[P(x) \vee (D(x, g(x)) \wedge \neg E(f(x), g(x)) \wedge \neg E(x, g(x)))] \wedge \neg P(a)$$

(7) Convertim la forma normală conjunctivă

$$(P(x) \vee D(x, g(x))) \wedge (P(x) \vee \neg E(f(x), g(x)))$$
$$\wedge (P(x) \vee \neg E(x, g(x))) \wedge \neg P(a)$$

(8) Eliminăm $\wedge$ și scriem disjuncții ca și clauze separate

$$P(x) \vee D(x, g(x))$$
$$P(x) \vee \neg E(f(x), g(x))$$
$$P(x) \vee \neg E(x, g(x))$$
$$\neg P(a)$$