

# Paradigme de Programare

Conf. dr. ing. Andrei Olaru

andrei.olaru@cs.pub.ro | cs@andreiolaru.ro  
Departamentul de Calculatoare

2020





# Procesul de demonstrare



- 1 Inițializarea **stivei de scopuri** cu scopul solicitat;
- 2 Inițializarea **substituției** (utilizate pe parcursul unificării) cu mulțimea vidă;
- 3 Extragerea scopului din **vârful** stivei și determinarea **primei** clauze din program cu a cărei concluzie **unifică**;
- 4 Îmbogățirea corespunzătoare a **substituției** și adăugarea **premiselor** clauzei în stivă, în ordinea din program;
- 5 Salt la pasul 3.



- 6 În cazul **imposibilității** satisfacerii scopului din vârful stivei, **revenirea** la scopul anterior (*backtracking*), și încercarea altei modalități de satisfacere;
- 7 **Succes** la **golirea** stivei de scopuri;
- 8 **Eșec** la imposibilitatea satisfacerii **ultimului** scop din stivă.



## Ex Exemplu

```
1 parent( andrei , bogdan ).  
2 parent( andrei , bianca ).  
3 parent( bogdan , cristi ).  
4  
5 grandparent( X , Y ) :- parent( X , Z ) , parent( Z , Y ) .
```

- $true \Rightarrow parent( andrei , bogdan )$
- $true \Rightarrow parent( andrei , bianca )$
- $true \Rightarrow parent( bogdan , cristi )$
- $\forall x . \forall y . \forall z . ( parent( x , z ) \wedge parent( z , y ) \Rightarrow grandparent( x , y ) )$

# Exemplul genealogic (1)



$S = \emptyset$

$G = \{gp(X, Y)\}$



$gp(X1, Y1) :- p(X1, Z1), p(Z1, Y1)$



$S = \{X = X1, Y = Y1\}$

$G = \{p(X1, Z1), p(Z1, Y1)\};$



$p(\text{andrei}, \text{bogdan})$



... 1



$p(\text{andrei}, \text{bianca})$



... 2



$p(\text{bogdan}, \text{cristi})$



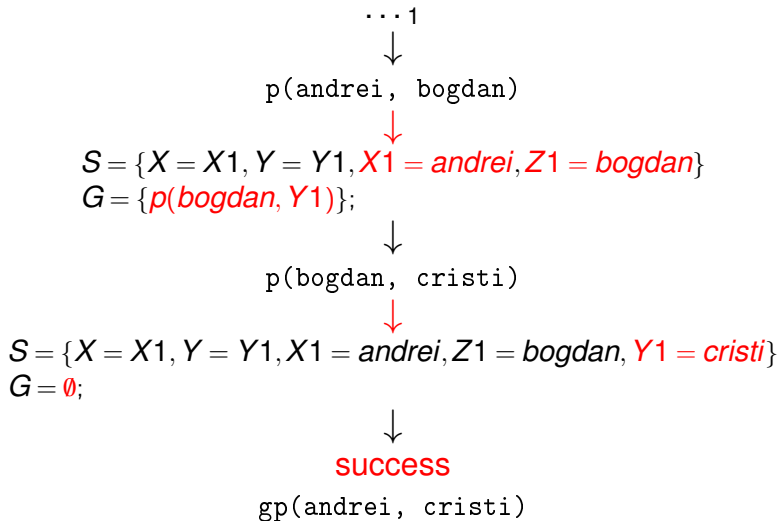
... 3



# Exemplul genealogic (2)

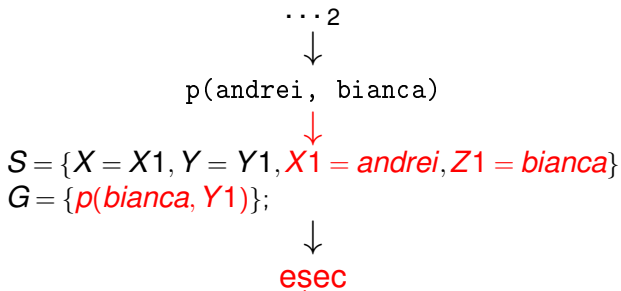


Ramura 1



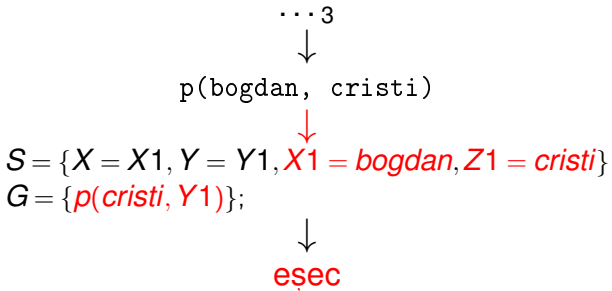
# Exemplul genealogic (3)

## Ramura 2



# Exemplul genealogic (4)

## Ramura 3





- Ordinea evaluării / încercării demonstrării scopurilor
  - Ordinea **clauzelor** în program;
  - Ordinea **premiselor** în cadrul regulilor.
  
- Recomandare: premisele **mai ușor** de satisfăcut și **mai specifice** primele – exemplu: axiome.



### *Forward chaining (data-driven)*

- Derivarea **tuturor** concluziilor, pornind de la datele inițiale;
- **Opre** la obținerea scopului (scopurilor);

### *Backward chaining (goal-driven)*

- Utilizarea **exclusivă** a regulilor care pot contribui efectiv la satisfacerea scopului;
- Determinarea regulilor a căror concluzie **unifică** cu scopul;
- Încercarea de satisfacere a **premiselor** acestor reguli ș.a.m.d.

# Strategii de control

## Algoritm Backward chaining



1. **BackwardChaining**(*rules, goals, subst*)  
lista **regulilor** din program, stiva de **scopuri**, **substituția** curentă, inițial vidă.  
**returns** satisfiabilitatea scopurilor
2. **if** *goals* =  $\emptyset$  **then**
3.     **return** SUCCESS
4.     *goal*  $\leftarrow$  *head*(*goals*)
5.     *goals*  $\leftarrow$  *tail*(*goals*)
6.     **for-each** *rule*  $\in$  *rules* **do**     // în ordinea din program
7.         **if** *unify*(*goal, conclusion*(*rule*), *subst*)  $\rightarrow$  *bindings*
8.             *newGoals*  $\leftarrow$  *premises*(*rule*)  $\cup$  *goals*     // **adâncime**
9.             *newSubst*  $\leftarrow$  *subst*  $\cup$  *bindings*
10.            **if** *BackwardChaining*(*rules, newGoals, newSubst*)
11.            **then return** SUCCESS
12. **return** FAILURE





### Ex | Minimul a două numere

```
1 min(X, Y, M) :- X =< Y, M is X.
2 min(X, Y, M) :- X > Y, M is Y.
3
4 min2(X, Y, M) :- X =< Y, M = X.
5 min2(X, Y, M) :- X > Y, M = Y.
6
7 % Echivalent cu min2.
8 min3(X, Y, X) :- X =< Y.
9 min3(X, Y, Y) :- X > Y.
```



# Exemplu – Minimul a două numere



## Utilizare

---

```
1  ?- min(1+2, 3+4, M).
2  M = 3 ;
3  false.
4
5  ?- min(3+4, 1+2, M).
6  M = 3.
7
8  ?- min2(1+2, 3+4, M).
9  M = 1+2 ;
10 false.
11
12 ?- min2(3+4, 1+2, M).
13 M = 1+2.
```

# Exemplu – Minimul a două numere



## Observații

- Condiții mutual exclusive:  $X \leq Y$  și  $X > Y \rightarrow$  cum putem **elimina** redundanța?

### Ex Exemplu

```
1 min4(X, Y, X) :- X <= Y.  
2 min4(X, Y, Y).
```

```
1 ?- min4(1+2, 3+4, M).  
2 M = 1+2 ;  
3 M = 3+4.
```

- **Greșit!**



- Soluție: **oprirea** recursivității după prima satisfacere a scopului.



### Exemplu

```
1 min5(X, Y, X) :- X =< Y, !.
```

```
2 min5(X, Y, Y).
```

```
1 ?- min5(1+2, 3+4, M).
```

```
2 M = 1+2.
```



- La **prima** întâlnire → **satisfacere**;
- La **a doua** întâlnire în momentul revenirii (*backtracking*) → **eșec**, cu inhibarea **tuturor** căilor ulterioare de satisfacere a scopului care a unificat cu concluzia regulii curente;
- Utilitate în **eficientizarea** programelor.



### Ex | Exemplu

```
1 girl(mary).
2 girl(ann).
3
4 boy(john).
5 boy(bill).
6
7 pair(X, Y) :- girl(X), boy(Y).
8 pair(bella, harry).
9
10 pair2(X, Y) :- girl(X), !, boy(Y).
11 pair2(bella, harry).
```

# Operatorul *cut*

## Utilizare

---



```
1  ?- pair(X, Y).
2  X = mary,
3  Y = john ;
4  X = mary,
5  Y = bill ;
6  X = ann,
7  Y = john ;
8  X = ann,
9  Y = bill ;
10 X = bella,
11 Y = harry.
```

```
1  ?- pair2(X, Y).
2  X = mary,
3  Y = john ;
4  X = mary,
5  Y = bill.
```



## Ex Exemplu

```
1 nott(P) :- P, !, fail.  
2 nott(P).
```

- P: atom – exemplu: boy(john)
- dacă P este **satisfiabil**:
  - eșecul **primei** reguli, din cauza lui fail;
  - abandonarea celei **de-a doua** reguli, din cauza lui !;
  - rezultat: nott(P) **nesatisfiabil**.
- dacă P este **nesatisfiabil**:
  - eșecul **primei** reguli;
  - succesul celei **de-a doua** reguli;
  - rezultat: nott(P) **satisfiabil**.



- Prolog: structura unui program, funcționarea unei demonstrații
- ordinea evaluării, algoritmul de control al demonstrației
- tehnici de control al execuției.