UCS1524 – Logic Programming

Evaluation Strategies

Session Meta Data

Author	**Dr. D. Thenmozhi**
Reviewer	
Version Number	1.2
Release Date	27 August 2022

Session Objectives

- Understanding evaluation strategies of logic programming
- Learn the concept of swapping lemma, DFS and BFS as evaluation strategies

Session Outcomes

- At the end of this session, participants will be able to
  - explain the concept of swapping lemma and apply the BFS and DFS strategies in logic program.

Agenda

- Evaluation Strategies
- Swapping Lemma
- BFS
- DFS

Evaluation strategies

- Logic programs are *non-deterministic, i.e. after each computation step there* can be more than one possibility for continuing the computation.
- *Evaluation strategy determines in which order the nondeterministic* computation steps have to be performed
- The non-determinism in logic programs occurs in two different forms:
  - type 1 nondeterminism
  - type 2 nondeterminism

Type 1 nondeterminism

- Suppose, we have already selected a particular literal (i.e. a procedure call) in the goal clause which is to be unified with some procedure head of some program clause.
- If there are several such program clauses which can be used to produce resolvents, we call this *type 1 nondeterminism.*
- *E.g*

Goal clause ?- A,B,C

Program clauses
B :-D.
B.
B :-E,F

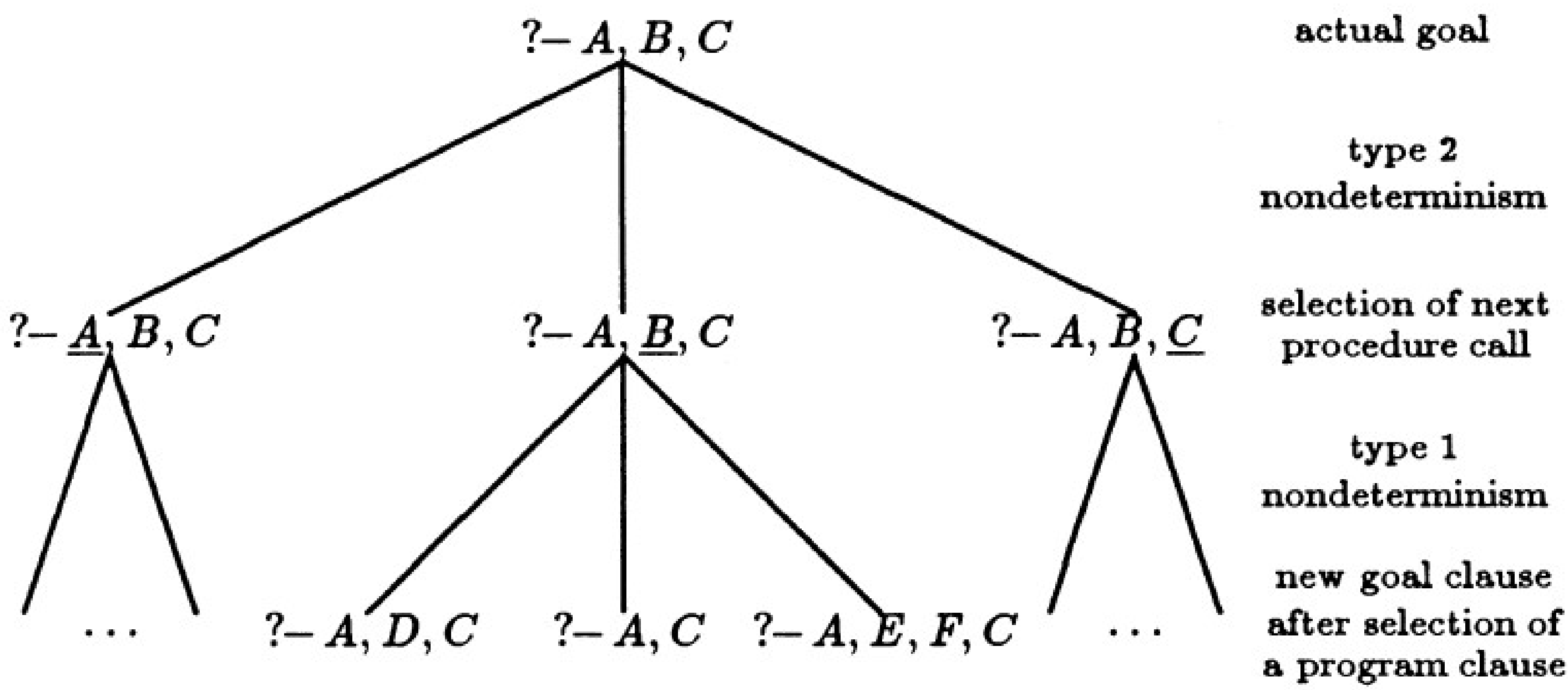
3 SLD resolvents

?- A,D,C.
?- A,C.
?- A,E,F,C.

Type 2 nondeterminism

- If the goal clause consists of n literals (i.e. procedure calls), then each of these n literals can be used for unification in the next resolution step.
- This gives n! many ways of evaluating such a goal clause.
- This freedom in the choice of the literal in the goal clause constitutes the type 2 nondeterminism.

Nondeterminism in logic programming



- Every evaluation strategy concerning type 2 nondeterminism leads to the same computation results
- Solution: follow the leftmost branch

Swapping Lemma

- The evaluation order of procedure calls is not relevant, and can be swapped without changing the computation result.

$$?\!-\ A_1, \ldots, A_i, \ldots, A_j, \ldots, A_n$$

$$B \ :\!-\ C_1, \ldots, C_k$$

$$?\!-\ (A_1, \ldots, C, \ldots, A_j, \ldots, A_n)sub_1$$

$$D \ :\!-\ E_1, \ldots, E_l$$

$$?\!-\ (A_1, \ldots, C, \ldots, E, \ldots, A_n)sub_1 sub_2$$

$$?\!-\ A_1, \ldots, A_i, \ldots, A_j, \ldots, A_n$$

$$D \ :\!-\ E_1, \ldots, E_l$$

$$?\!-\ (A_1, \ldots, A_i, \ldots, E, \ldots, A_n)sub'_1$$

$$B \ :\!-\ C_1, \ldots, C_k$$

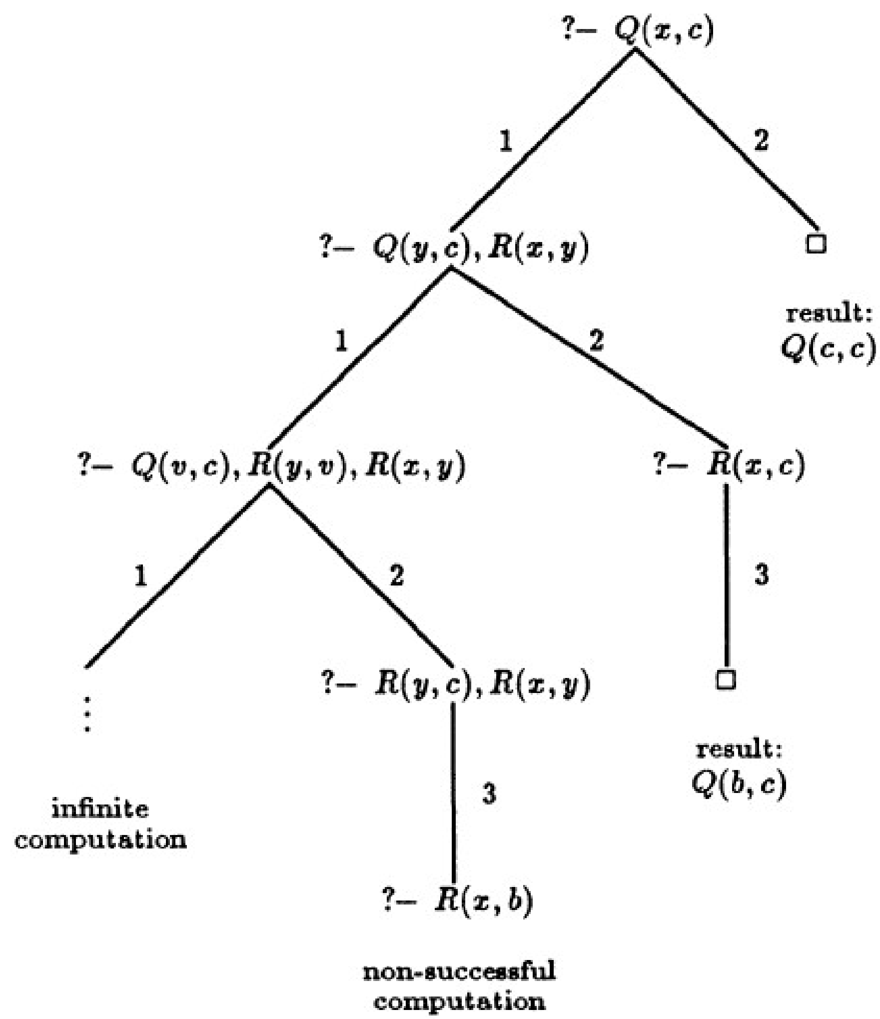
$$?\!-\ (A_1, \ldots, C, \ldots, E, \ldots, A_n)sub'_1 sub'_2$$

Canonical computation of logic program

- A computation of a logic program is called *canonical if in each computation* step the *first literal (i.e. the literal at the leftmost position) in the goal* clause is used for the resolution
- Example

1. $Q(x, z) \text{ :- } Q(y, z), R(x, y)$
2. $Q(x, x).$
3. $R(b, c).$

$?\!- \; Q(x, c).$



If in every step, the first program clause is used for resolution, we may obtain an infinite computation
Type 2 can be resolved but not Type 1

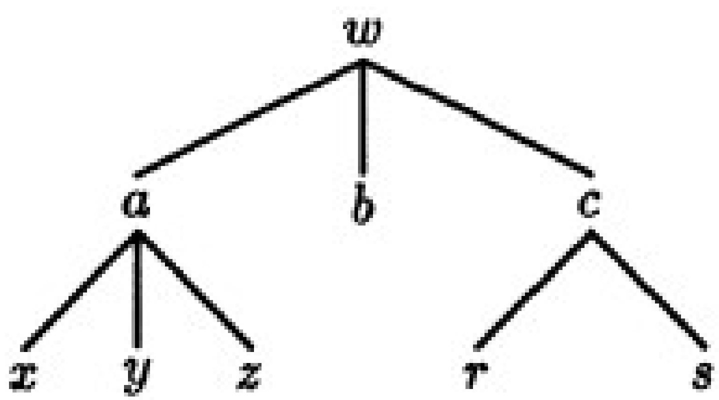
Elimination of Type 1 nondeterminism

Search strategies

- BFS
  - The search in the tree is performed so that all nodes on depth t are visited (e.g. from left to right) before any node on depth t + 1 is visited (t = 0, 1, 2,...).
  - To reach the nodes in the computation tree of depth t, the breadth-first search strategy needs to visit exponentially many nodes (in t)
  - It should be clear that every successful computation in the computation tree of a logic program can be found this way after finitely many steps. In other words, the breadth-first search evaluation strategy is *complete.*
- DFS
  - Standard interpreters for the programming language PROLOG use the depth-first search evaluation strategy

Search strategies

- DFS
  - Starting from the root of the tree, the subtrees are visited in some fixed order (from left to right) recursively.
  - In contrast to breadth-first search the search goes into the depth of the tree first.
  - Whenever a node is reached that has no sons to search left the search returns to the father node (backtracking) and continues with the next brother node
  - However, DFS is incomplete



$w, a, b, c, x, y, z, r, s$ BFS

$w, a, x, y, z, b, c, r, s.$ DFS

Prolog Evaluation Strategy

Given: Logic program $F = (C_1, C_2, \ldots, C_n)$, where
$C_i = B_i \ \text{:-}\ D_{i,1}, \ldots, D_{i,n_i}$, and goal clause $G = ?\text{-}\ A_1, \ldots, A_k$.

The main routine consists of

success := **false**;
evaluate(G, [ ]);
if success **then** write('yes') **else** write('no');

Prolog Evaluation Strategy

```
procedure evaluate(G : goalclause ; sub : substitution);
var i : integer;
begin
  if G = □ then
      begin
        H := (A_1 ∧ ··· ∧ A_k)sub;
        write('RESULT:',H);
        success := true
      end
    else {assume G has the form G = ?- E_1, ..., E_k}
      begin
        i := 0;
        while (i < n) and not success do
          begin
            i := i + 1;
            if {E_1, B_i} is unifiable using most general unifier s
              (where the variables in B_i have been renamed first)
              then
                evaluate( ?- (D_{i,1}, ..., D_{i,n_i}, E_2, ..., E_k)s, subs)
            end
          end
    end;
```

Summary

- Evaluation strategies
- Types of nondeterminism
  - Type 1 nondeterminism
  - Type 2 nondeterminism
- Swapping Lemma
- Searching strategies
  - BFS
  - DFS

Check your understanding

- Perform DFS and write the possible answers for the program
  - male(philip).
  - female(elizabeth).
  - parent(elizabeth, charles).
  - parent(elizabeth, anne).
  - parent(philip, anne).
  - father(X, Y) :- parent(X, Y), male(X)

  with a Query
  - ?- father(X, anne).

Check your understanding

- Perform DFS and write the possible answers for the program
  - P(a,a).
  - P(a,b).
  - P(x, y) :- P(y, x)

with the given goal clause

  - ?- P(a, z), P(z, a)