COMP3411-9814- Artificial Intelligence

Prolog
Finding answers, Operators

2020 – Summer Term

Tatjana Zrimec

Outline

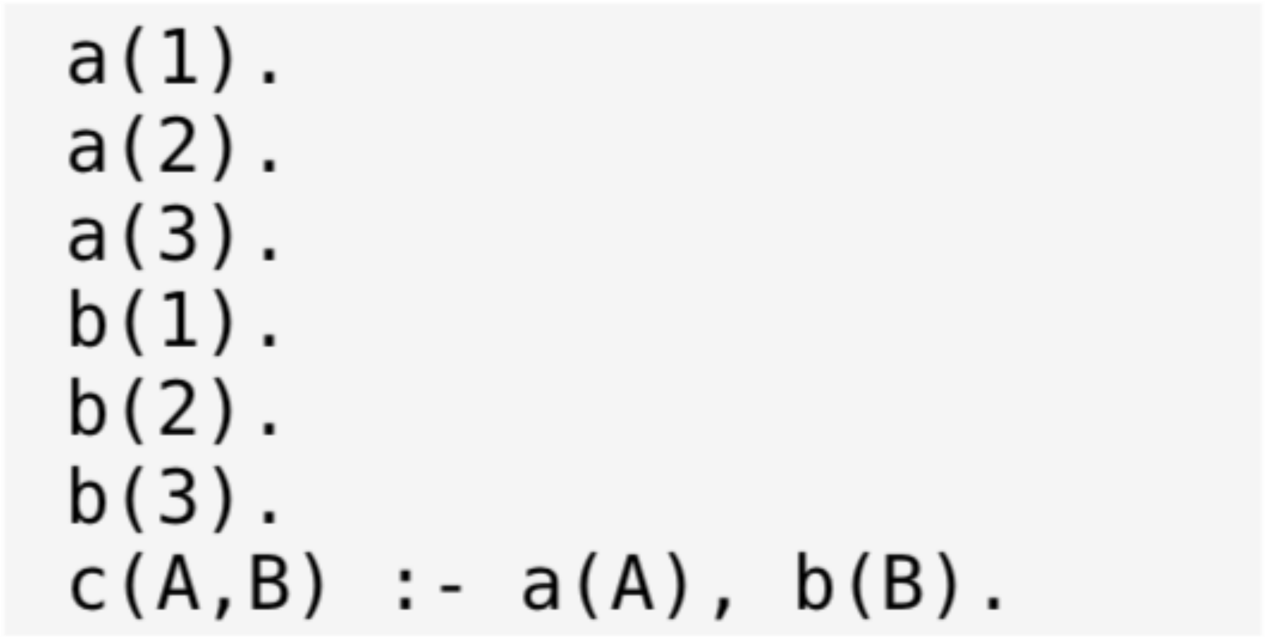
- Prolog finding answers
- Declarative and Procedural Meaning of programs
- Operators in Prolog

Example

- Prolog – finding answers

Prolog – finding answers

- Prolog uses depth first search to find answers !



```
a(1).
a(2).
a(3).
b(1).
b(2).
b(3).
c(A,B) :- a(A), b(B).
```

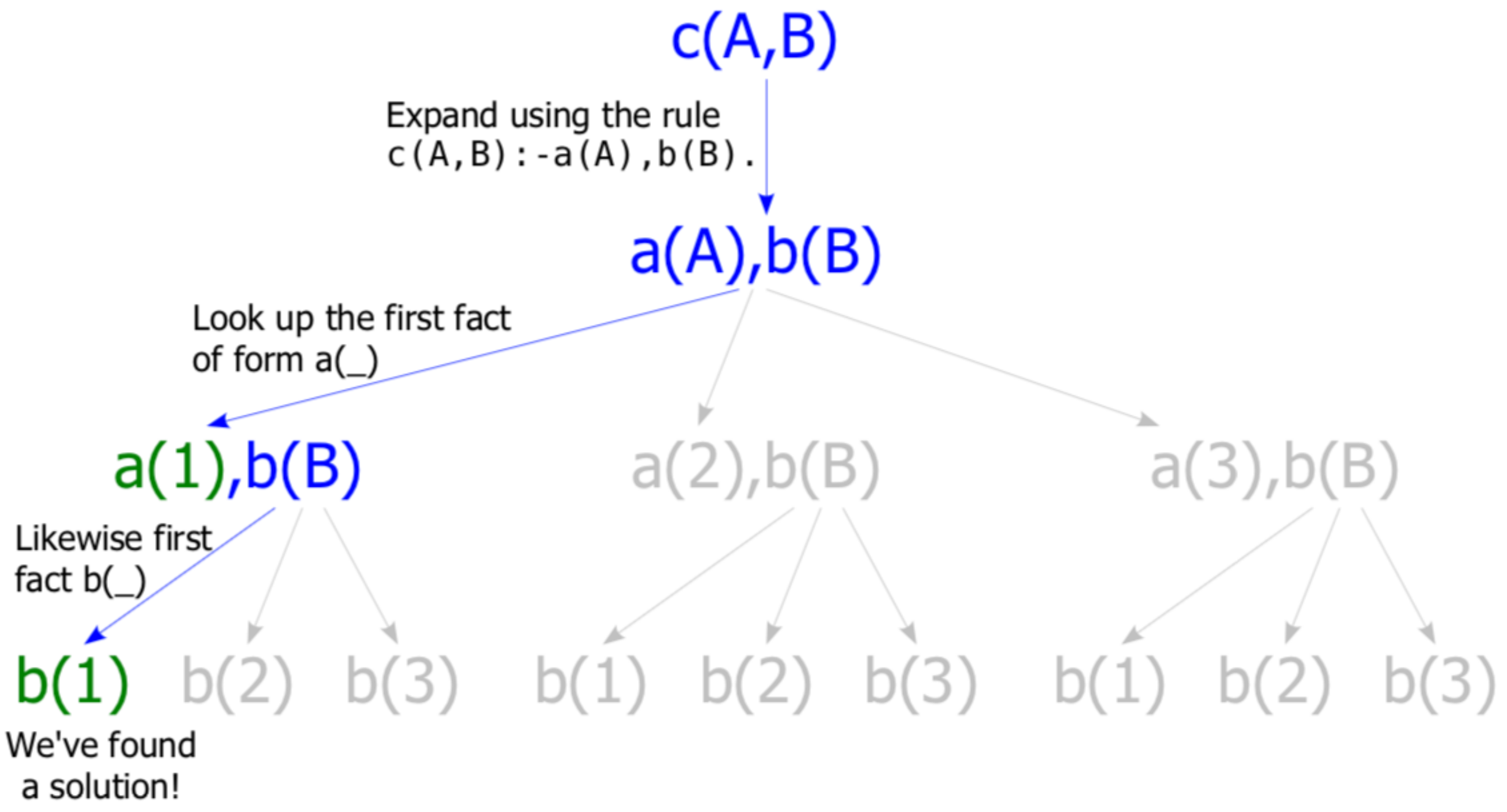
Prolog – finding answers

- Prolog uses depth first search to find answers !

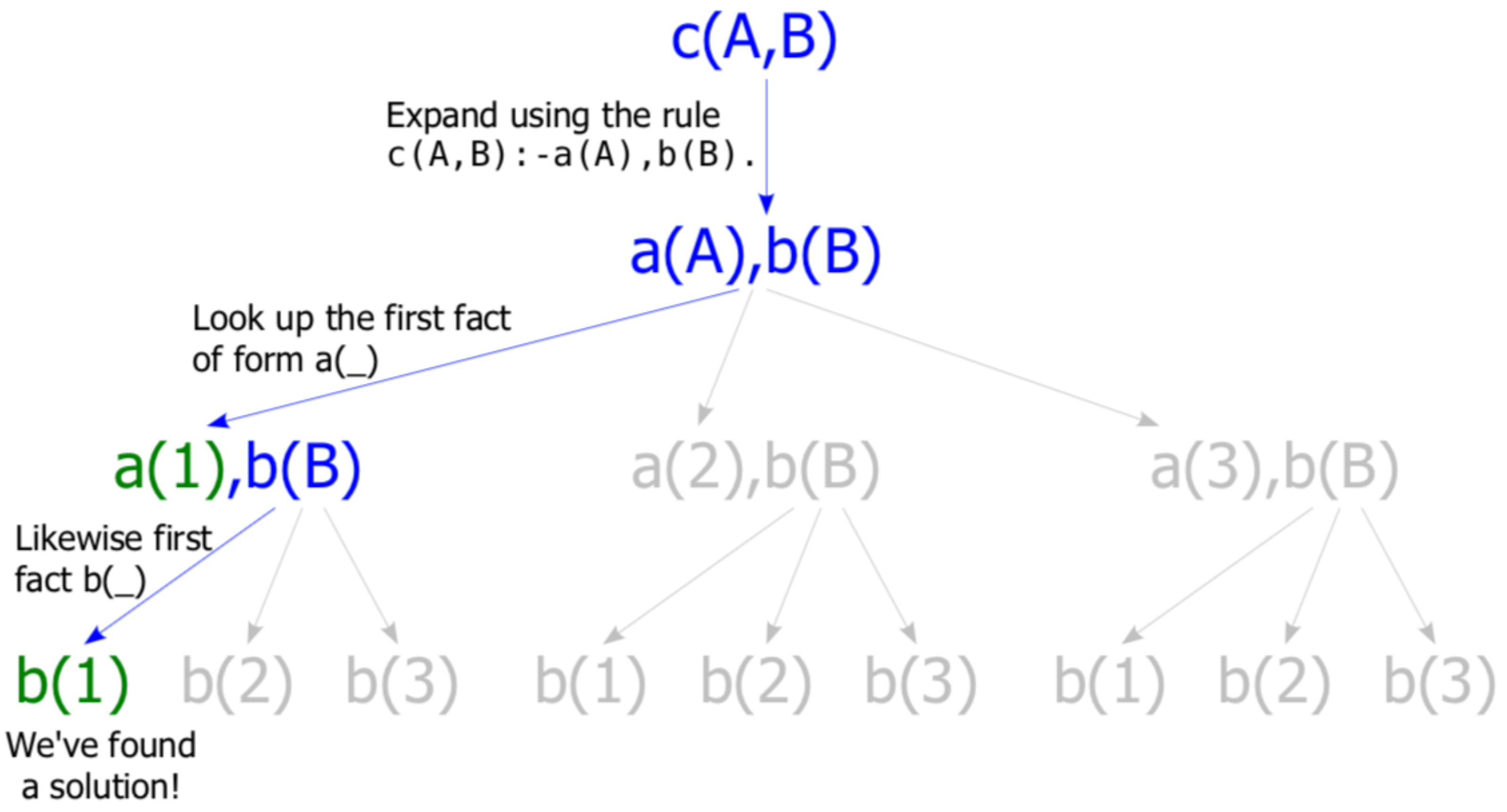
```
a(1).
a(2).
a(3).
b(1).
b(2).
b(3).
c(A,B) :- a(A), b(B).
```

What does Prolog do when given this query ? c(A,B)

Depth-first solution of query c(A,B)

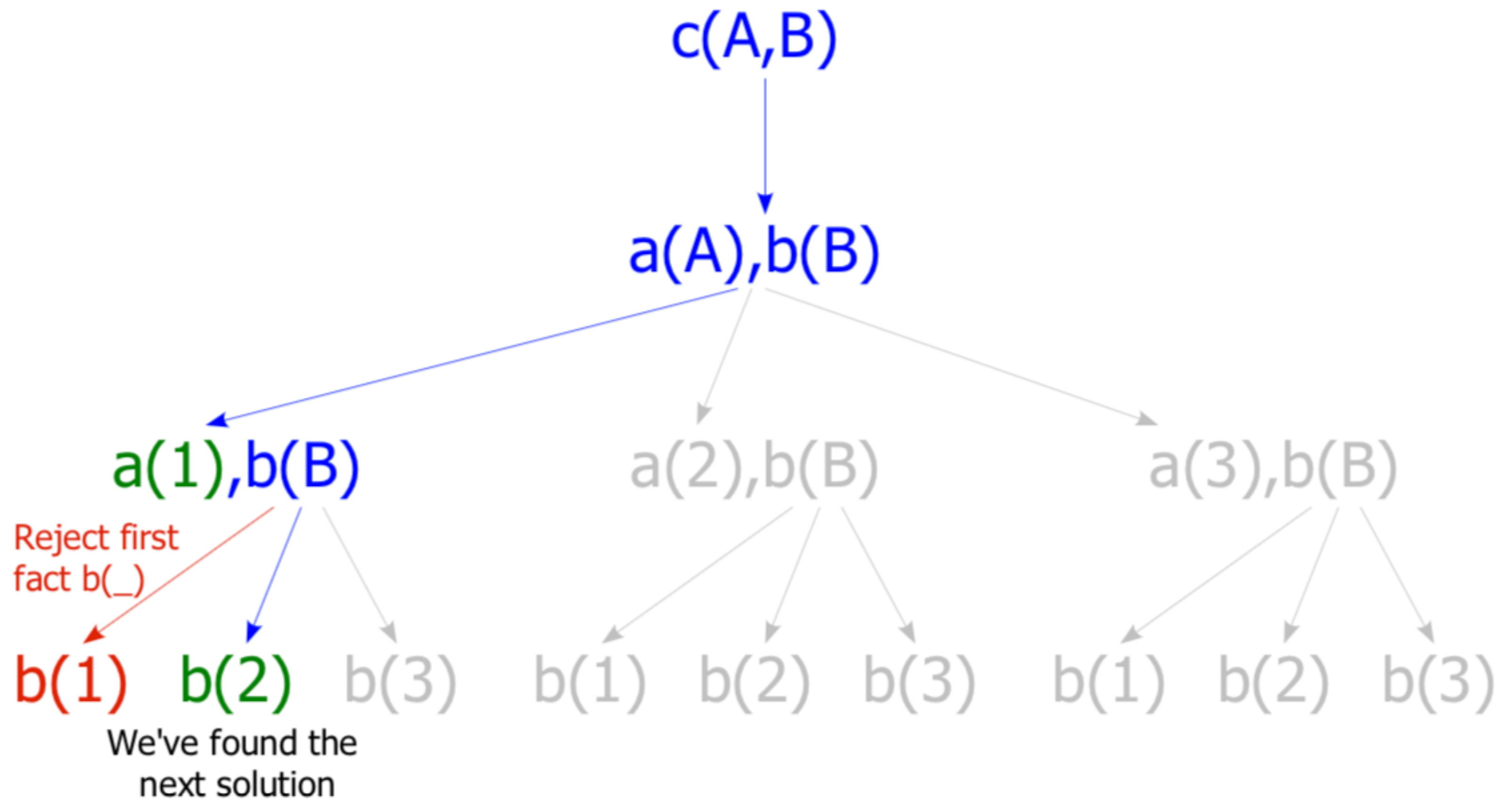


Depth-first solution of query c(A,B)

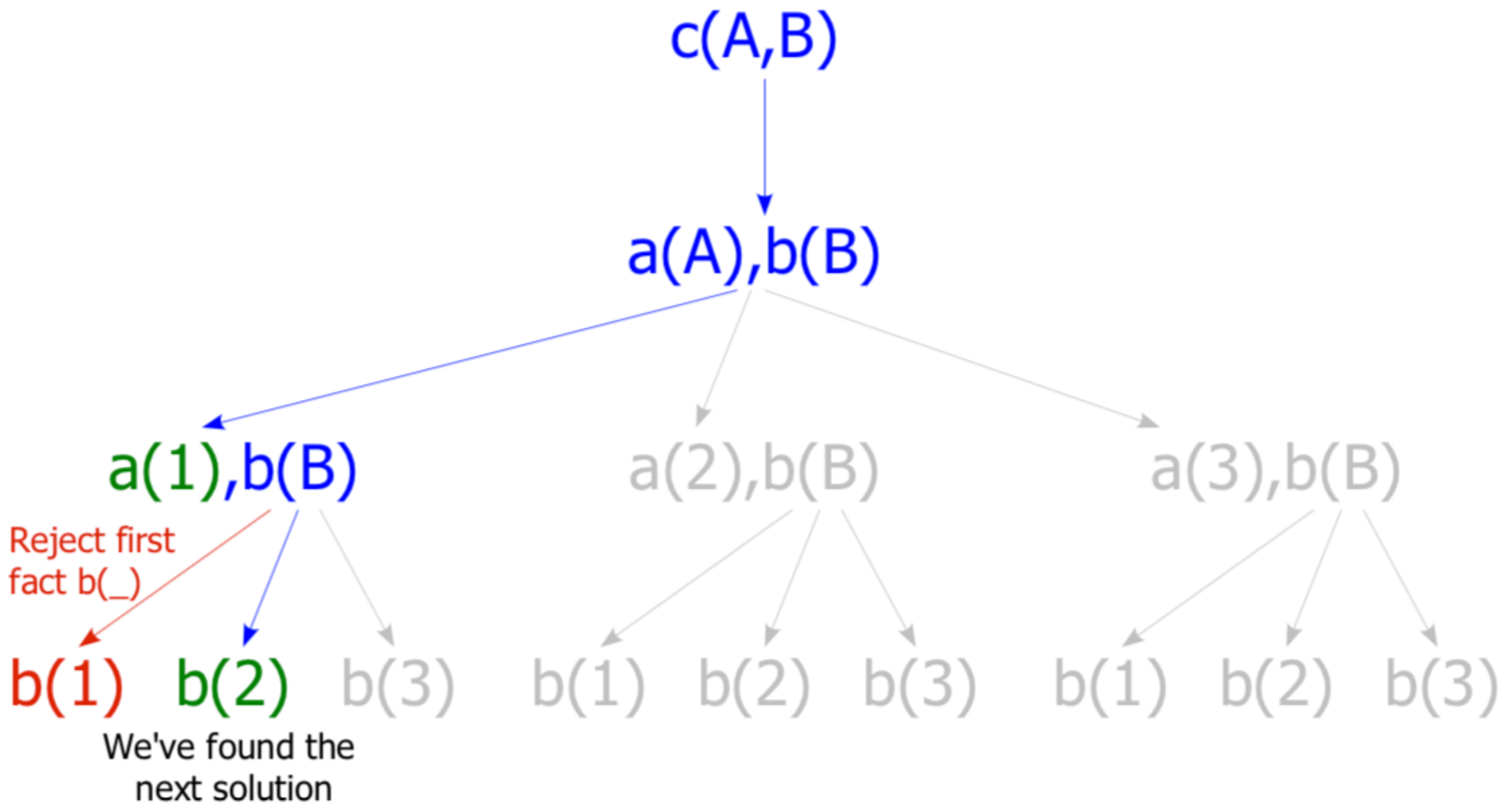


Variable bindings : A= 1, B=1

Backtrack to find another solution

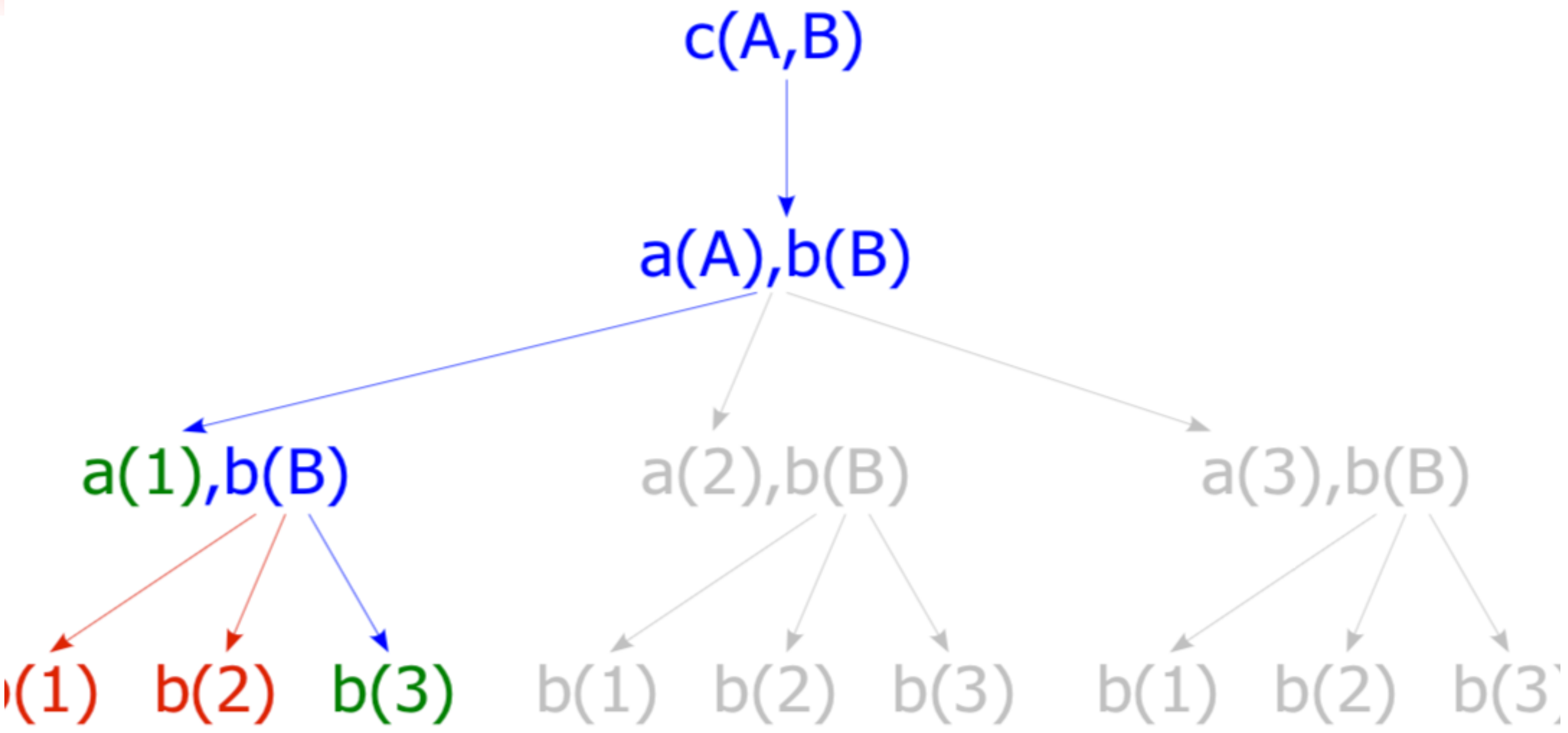


Backtrack to find another solution



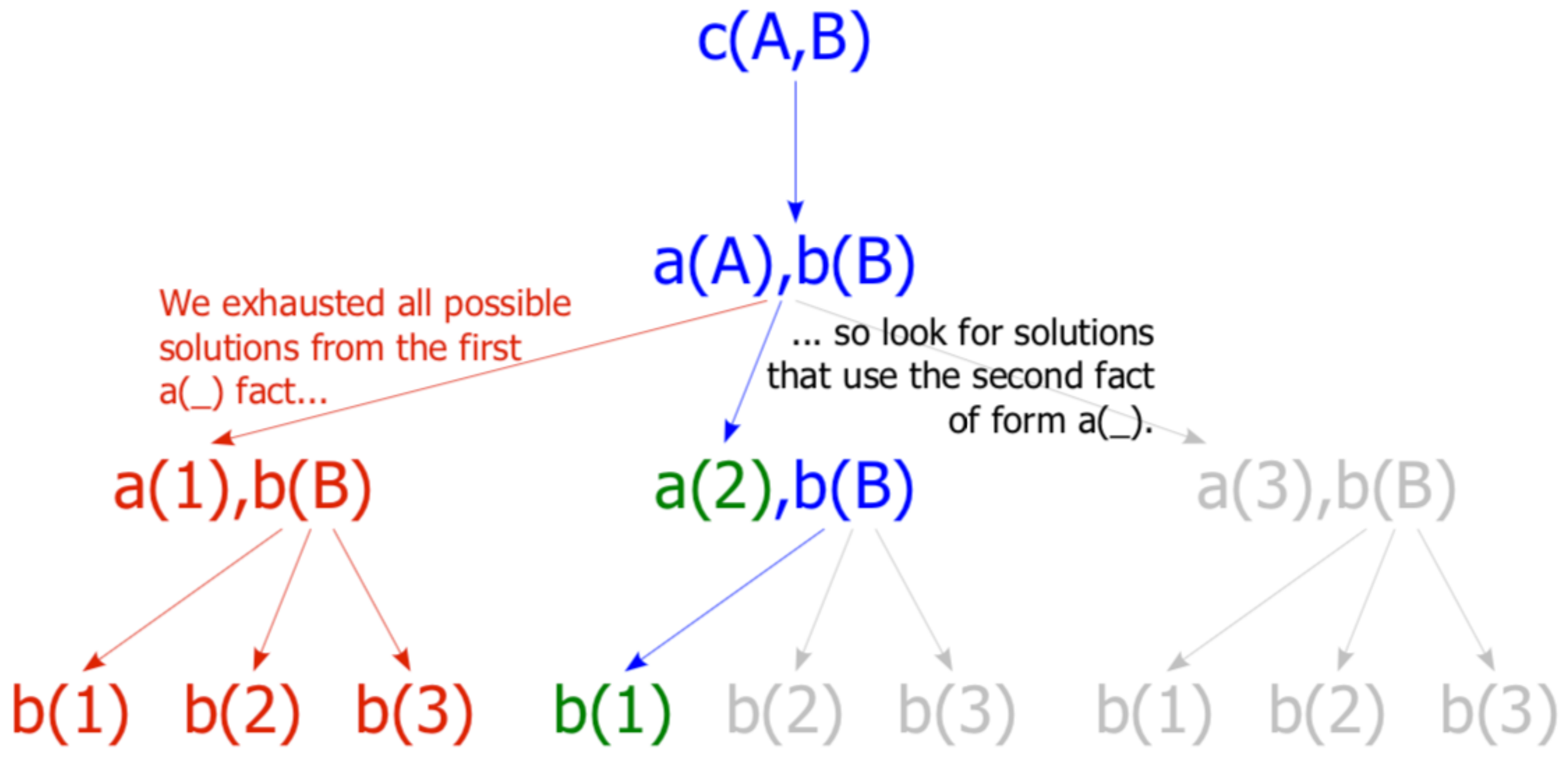
Variable bindings : A= 1, B=2

Backtrack to find another solution



Variable bindings : A= 1, B=3

Backtrack to find another solution


c(A,B)
a(A),b(B)
We exhausted all possible solutions from the first a(_) fact...
... so look for solutions that use the second fact of form a(_).
a(1),b(B)
a(2),b(B)
a(3),b(B)
b(1)
b(2)
b(3)
b(1)
b(2)
b(3)
b(1)
b(2)
b(3)

Variable bindings : A= 2, B=1

Declarative and Procedural Meaning of programs

- Let look at the clause:

  P :- Q, R.

- Declarative reading of the clause:
  - P is true if Q and R are true.
  - From Q and R follows P.

- Procedural reading:
  - To solve the problem P, solve Q and then R.
  - To prove P, first prove Q and then R.

Declarative Meaning

- Let P be program and goal G
- A goal G is true (this is a logical follow from P), if and only if:
  - (1) There is a clause C in P that is valid
  - (2) There is clause instance I of C such that
    - ✓ (a) the head of I is identical to G, and
    - ✓ (b) all the goals in the body of I are true
- In general,
  - a question to Prolog is a *list* of goals separated by comas.
  - A list of goals is true if all the goals in the list are true for some instantiation of variables.
  - The values of the variables result from the most general instantiation.

Declarative and Procedural Meaning of programs

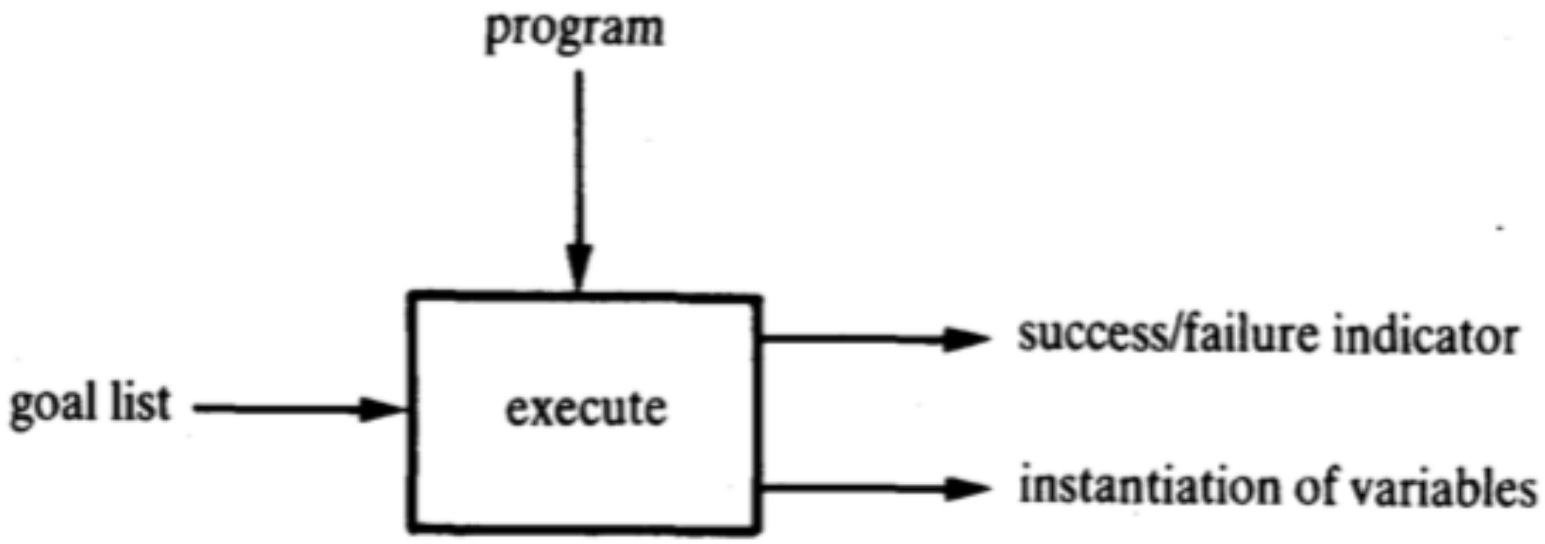
- A & B is logically equivalent to B & A

- Declarative meaning - only the relations defined by the program - *What* will be the output of a program
- The order of the goals in the clauses ***does not influence*** the declarative meaning

- The procedural meaning – *how* the relations are actually derived by the Prolog system
  - The algorithm
- The order of the goals in the clauses **influence** the procedural meaning

Procedural meaning

The procedural meaning specifies *how* Prolog answer questions.



Input/output view of the procedure that executes a list of goals.

A procedural meaning is an algorithm for executing a list of goals with respect to a given program.

Ivan Bratko book pp 41.

Programs and goals

P :- Q, R. % comma denotes the conjunction of goals
% they all have to be true.

P :- Q; R. % semicolon denotes the disjunction of goals
% P is true if Q is true or R is tyre.

P :- Q; R. Is same as P :- Q.
P :- R.

Operators in Prolog

Operator notation

For example:

    2*a + b*c   %only for ease of use

With operator notation:

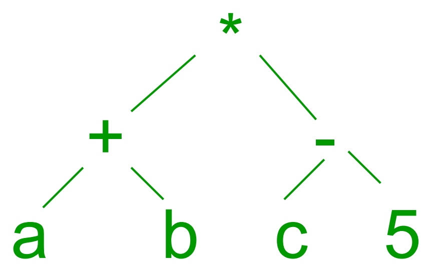
+(*(2,a), *(b,c)) = 2*a + b*c

+, * are infix operators in Prolog

The Arithmetic Expressions are also Trees

- For example: (a + b) * (c - 5)
- Written as an expression with the functors:

*(+(a, b), -(c, 5))



Operators in Prolog

- In addition to providing a user friendly operator notation for certain functors, Prolog also let's you define your own operators.

```
:- op(Precedence, Type, Name).
```

- Precedence is a number between 0 and 1200.

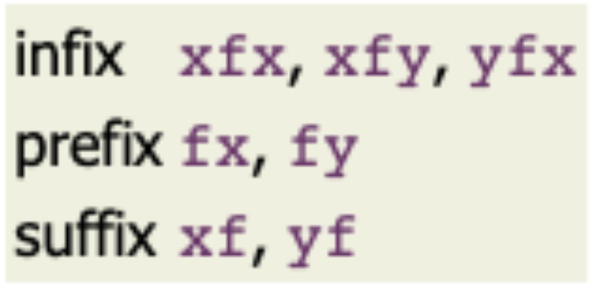
For example,

- the precedence of "=" is 700,
- the precedence of "+" is 500,
- the precedence of "* " is 400.

Operators in Prolog

:- op(Precedence, Type, Name).

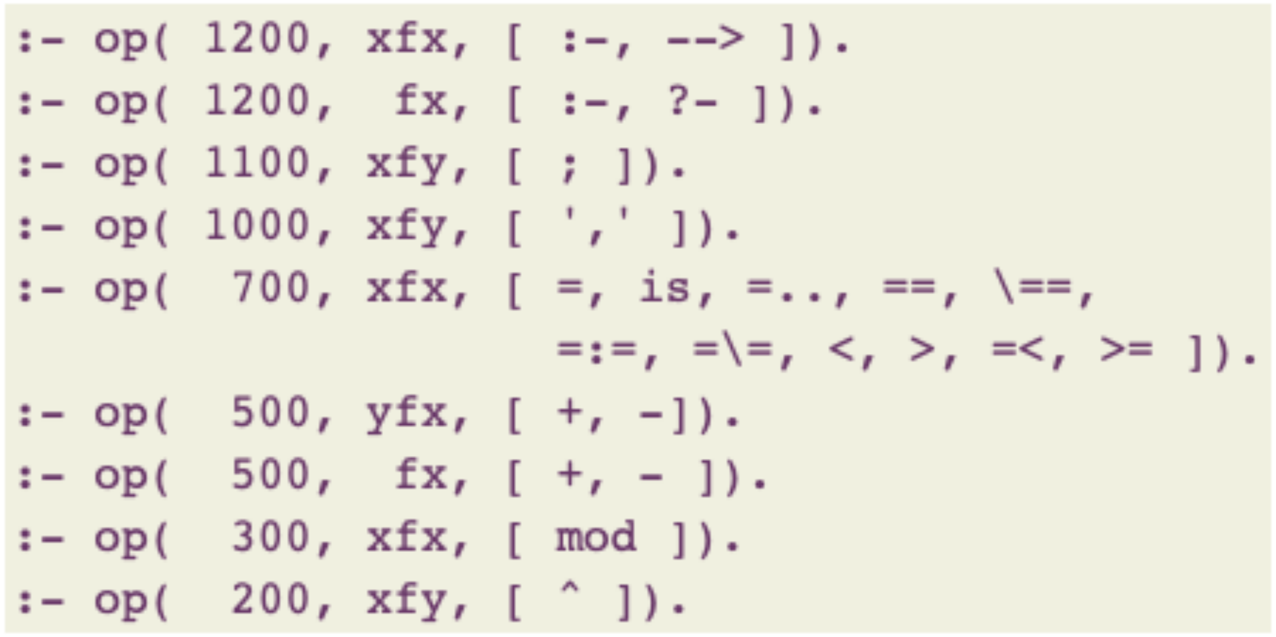
- Type is an atom specifying the type and associativity of the operator.
  - In the case of + this atom is yfx, which says that + is an infix operator f represents the operator and x and y the arguments.
  - x stands for an argument which has a precedence which is lower than the precedence of + and y stands for an argument which has a precedence which is lower or equal to the precedence of +.
  - There are the following possibilities for what Type may look like



infix	xfx, xfy, yfx
prefix	fx, fy
suffix	xf, yf

Operators in Prolog

- Here are the definitions for some of the built-in operators.
- Operators with the same properties can be specified in one statement by giving a list of their names instead of a single name as third argument of op.



```
:- op( 1200, xfx, [ :-, --> ]).
:- op( 1200,  fx, [ :-, ?- ]).
:- op( 1100, xfy, [ ; ]).
:- op( 1000, xfy, [ ',' ]).
:- op(  700, xfx, [ =, is, =.., ==, \==,
                    =:=, =\=, <, >, =<, >= ]).
:- op(  500, yfx, [ +, -]).
:- op(  500,  fx, [ +, - ]).
:- op(  300, xfx, [ mod ]).
:- op(  200, xfy, [ ^ ]).
```

One final thing to note is, that operator definitions don't specify the meaning of an operator, but only describe how it can be used syntactically. An operator definition doesn't say anything about when a query involving this operator will evaluate to true. It is only a definition extending the syntax of Prolog (Brarko, pp74).

User defined operators

An example how we can define two infix operators has and supports and then Prolog would allow to write "peter has information" and "floor supports table" as a facts in the database.

```
has( peter, information).
supports( floor, table).
```

Can be written with operators:

```
:- op( 600, xfx, has).
:- op( 600, xfx, supports).

peter has information.
floor supports table.
```