Exercises: Artificial Intelligence

The farmer, fox, goose and grain

Representation

- States of the form $[\mathcal{L}|\mathcal{R}]$, where:
  - $\mathcal{L}$: *Items on left bank*
  - $\mathcal{R}$: *Items on right bank*
- $\mathcal{L}$ and $\mathcal{R}$ contain:
  - Fa: *Farmer*
  - Fo: *Fox*
  - Go: *Goose*
  - Gr: *Grain*

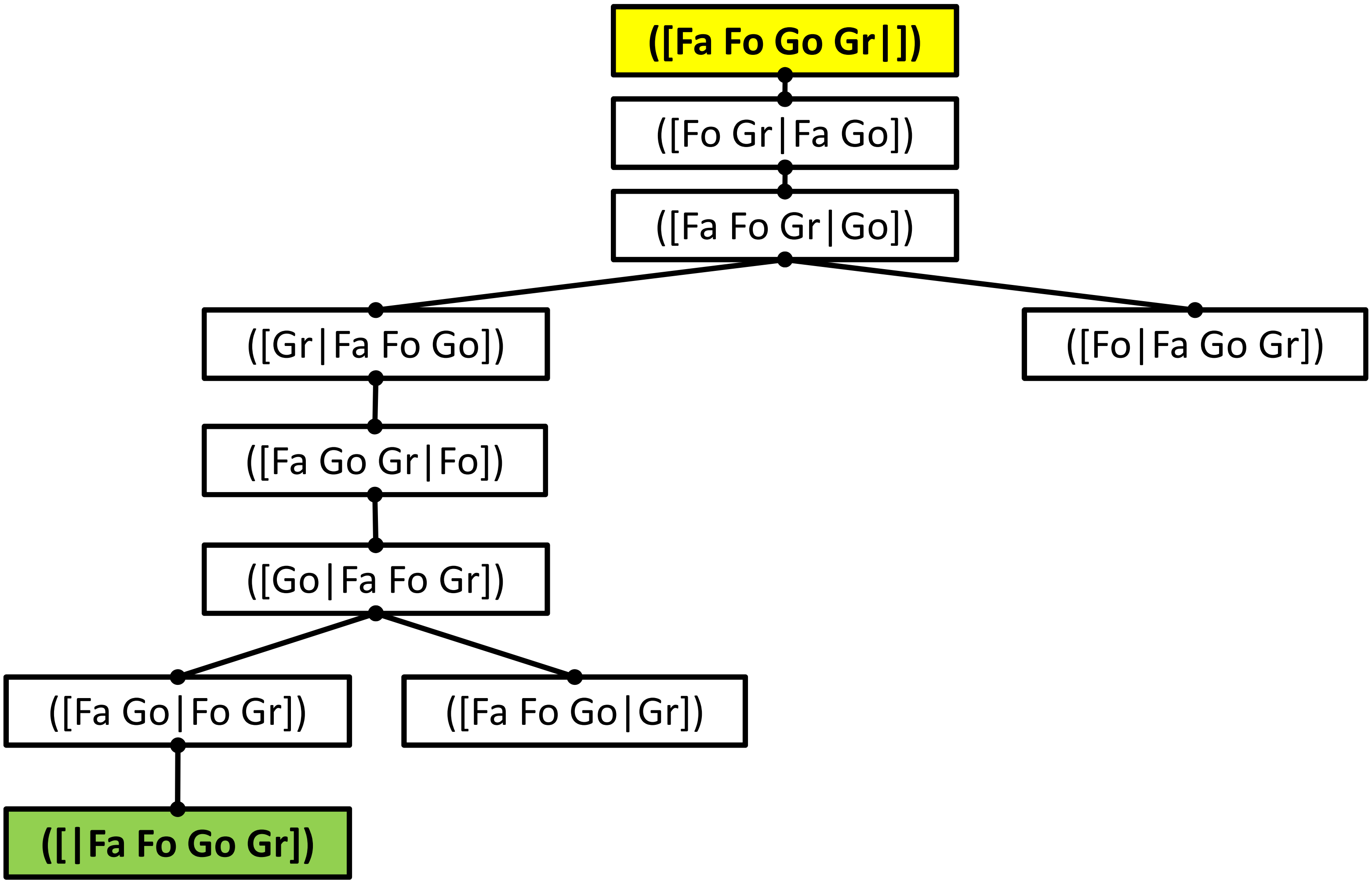
Representation

- Start: [Fa Fo Go Gr||]
- Goal: [||Fa Fo Go Gr]
- Rules:
  - R_1: [Fa $\mathcal{X}$|$\mathcal{Y}$] $\longrightarrow$ [$\mathcal{X}$|Fa $\mathcal{Y}$]
  - R_2: [$\mathcal{X}$|Fa $\mathcal{Y}$] $\longrightarrow$ [Fa $\mathcal{X}$|$\mathcal{Y}$]
  - R_3: [Fa z $\mathcal{X}$|$\mathcal{Y}$] $\longrightarrow$ [$\mathcal{X}$|Fa z $\mathcal{Y}$]
  - R_4: [$\mathcal{X}$|Fa z $\mathcal{Y}$] $\longrightarrow$ [Fa z $\mathcal{X}$|$\mathcal{Y}$]
  - No combination (Fo,Go) or (Go,Gr) on either bank, without the farmer.

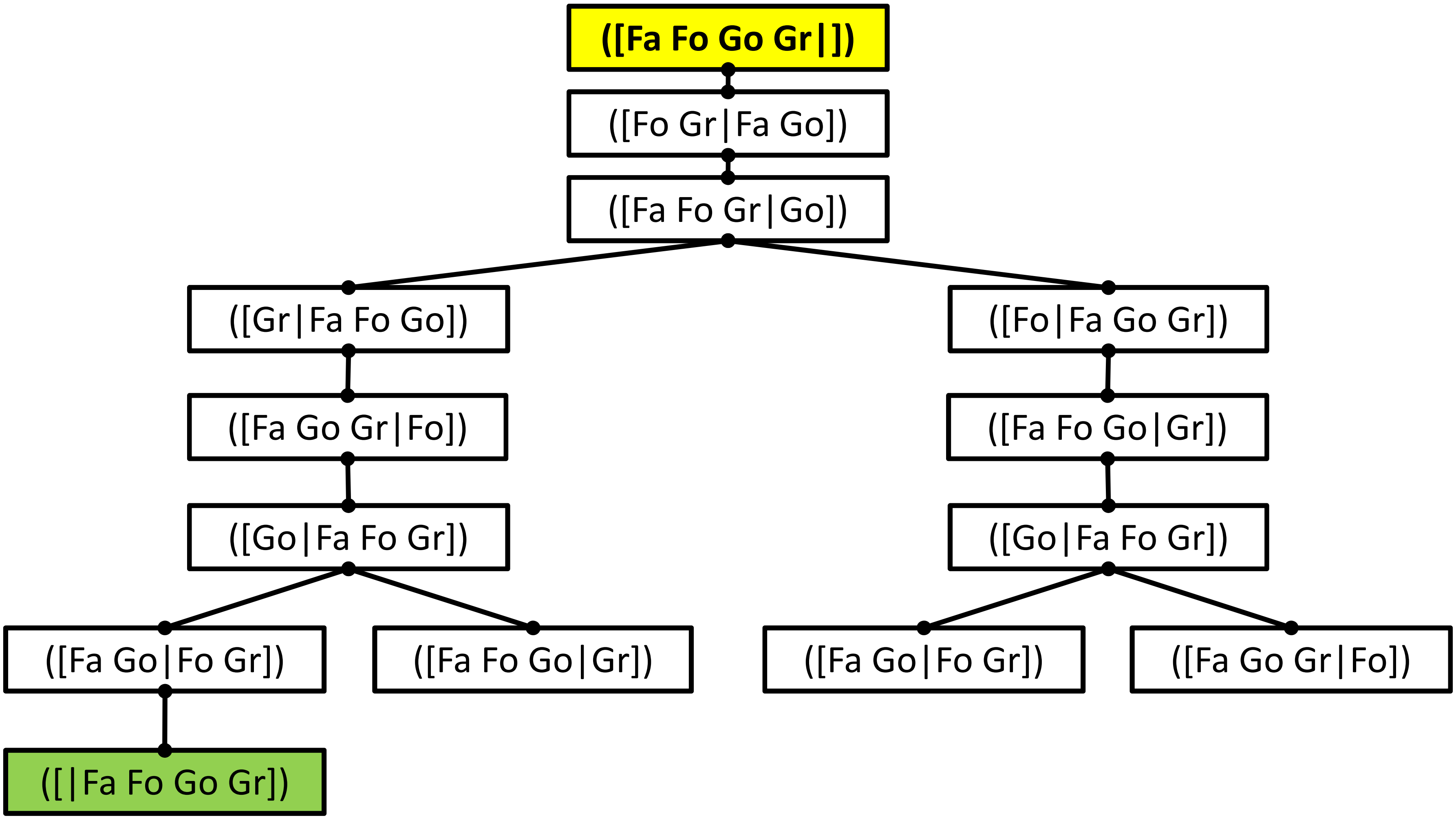
Depth-first search (queues)

- S = (<[Fa Fo Go Gr|]>)
- Q_1 = (<[Fa Fo Go Gr|][Fo Gr|Fa Go]>)
- Q_2 = (<[Fa Fo Go Gr|][Fo Gr|Fa Go][Fa Fo Gr|Go]>)
- Q_3 = (<[Fa Fo Go Gr|][Fo Gr|Fa Go][Fa Fo Gr|Go][Gr|Fa Fo Go]>,<[Fa Fo Go Gr|][Fo Gr|Fa Go][Fa Fo Gr|Go][Fo|Fa Go Gr]>)
- Q_4 = (<[Fa Fo Go Gr|][Fo Gr|Fa Go][Fa Fo Gr|Go][Gr|Fa Fo Go][Fa Go Gr|Fo]>,<[Fa Fo Go Gr|][Fo Gr|Fa Go][Fa Fo Gr|Go][Fo|Fa Go Gr]>)
- Q_5 = (<[Fa Fo Go Gr|][Fo Gr|Fa Go][Fa Fo Gr|Go][Gr|Fa Fo Go][Fa Go Gr|Fo][Go|Fa Fo Gr]>,<[Fa Fo Go Gr|][Fo Gr|Fa Go][Fa Fo Gr|Go][Fo|Fa Go Gr]>)
- Q_6 = (<[Fa Fo Go Gr|][Fo Gr|Fa Go][Fa Fo Gr|Go][Gr|Fa Fo Go][Fa Go Gr|Fo][Go|Fa Fo Gr][Fa Go|Fo Gr]>,<[Fa Fo Go Gr|][Fo Gr|Fa Go][Fa Fo Gr|Go][Gr|Fa Fo Go][Fa Go Gr|Fo][Go|Fa Fo Gr][Fa Fo Go|Gr]>,<[Fa Fo Go Gr|][Fo Gr|Fa Go][Fa Fo Gr|Go][Fo|Fa Go Gr]>)
- G = (<[Fa Fo Go Gr|][Fo Gr|Fa Go][Fa Fo Gr|Go][Gr|Fa Fo Go][Fa Go Gr|Fo][Go|Fa Fo Gr][Fa Go|Fo Gr][|Fa Fo Go Gr]>,<[Fa Fo Go Gr|][Fo Gr|Fa Go][Fa Fo Gr|Go][Gr|Fa Fo Go][Fa Go Gr|Fo][Go|Fa Fo Gr][Fa Fo Go|Gr]>,<[Fa Fo Go Gr|][Fo Gr|Fa Go][Fa Fo Gr|Go][Fo|Fa Go Gr]>)

Depth-first search (search tree)



Breadth-first search (search tree)



Exercises: Artificial Intelligence

Bidiretional Search

Bidirectional Search

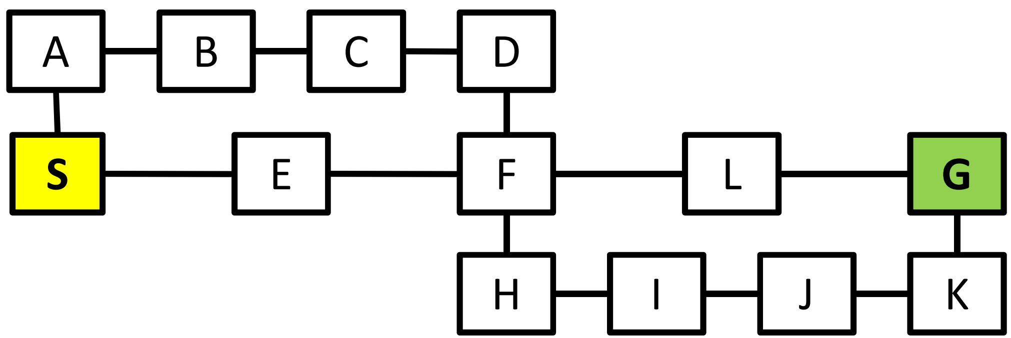
PROBLEM 1: BREADTH-FIRST?

Other methods than 2 x breadth-first

- Bidirectional search is complete for each combination with at least one complete search-strategy.
  - 2 x Breadth-first
  - 2 x Depth-first
  - Breadth-first and Depth-first
- Not each combination benefits from searching at both ends.

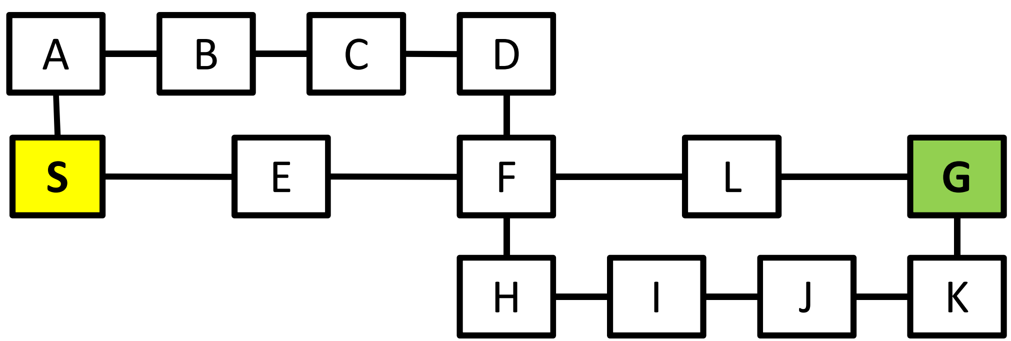
2 x Depth-first

- Forward:
  - (<S>) →(<SA>,<SE>) →(<SAB>,<SE>) →(<SABC>,<SE>) →(<SABCD>,<SE>) →**(<SABCDF>,<SE>)**
- Backward:
  - (<G>) →(<GK>,<GL>) →(<GKJ>,<GL>) →(<GKJI>,<GL>) →(<GKJIH>,<GL>) →**(<GKJIHF>,<GL>)**



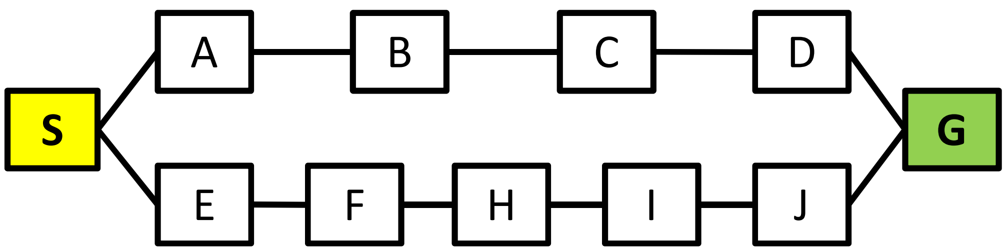
2 x Breadth-first

- Forward:
  - (<S>)→(<SA>,<SE>)→(<SE>,<SAB>)→**(<SAB>,<SEF>)**
- Backward:
  - (<G>)→(<GK>,<GL>)→(<GL>,<GKJ>)→**(<GKJ>,<GLF>)**



Breadth-first and Depth-first

- Forward (Breadth-first):
  - (<S>)→(<SA>,<SE>)→(<SE>,<SAB>)→(<SAB>,<SEF>) →**(<SEF>,<SABC>)**
- Backward (Depth-first):
  - (<G>)→(<GJ>,<GD>)→(<GJI>,<GD>)→(<GJIH>,<GD>) →**(<GJIHF>,<GD>)**

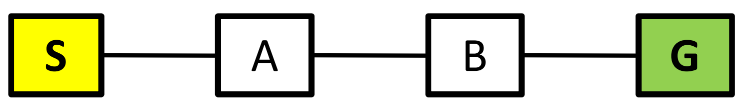


Bidirectional Search

PROBLEM 2: SHARED-STATE CHECK?

Replace shared-state check

- When only checking identical end-states, paths can cross each other unnoticed.
- Forward:
  - (<S>)→(<SA>)→(<SAB>)→(<SABG>)
- Backward:
  - (<G>)→(<GB>)→(<GBA>)→(<GBAS>)



Exercises: Artificial Intelligence

Beam Search

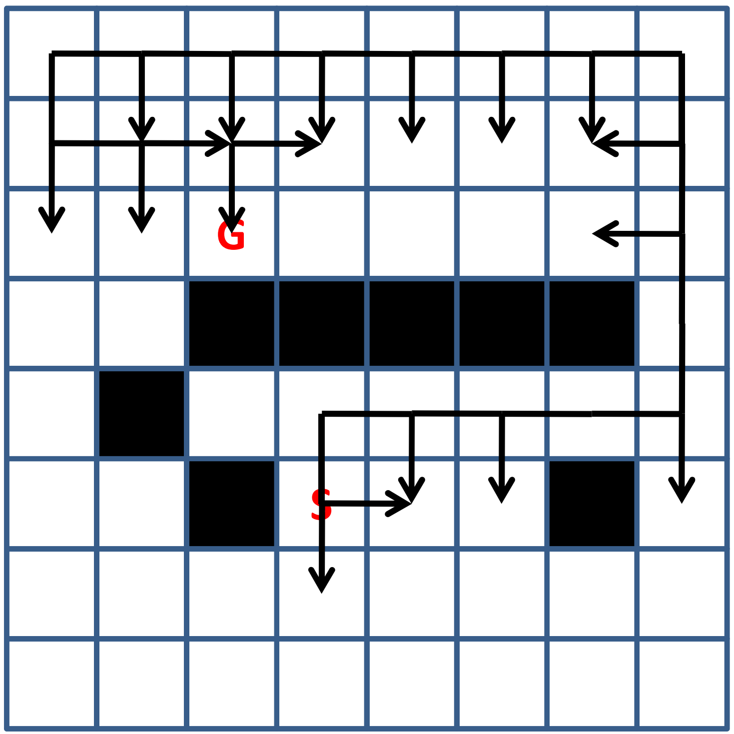
Beam Search

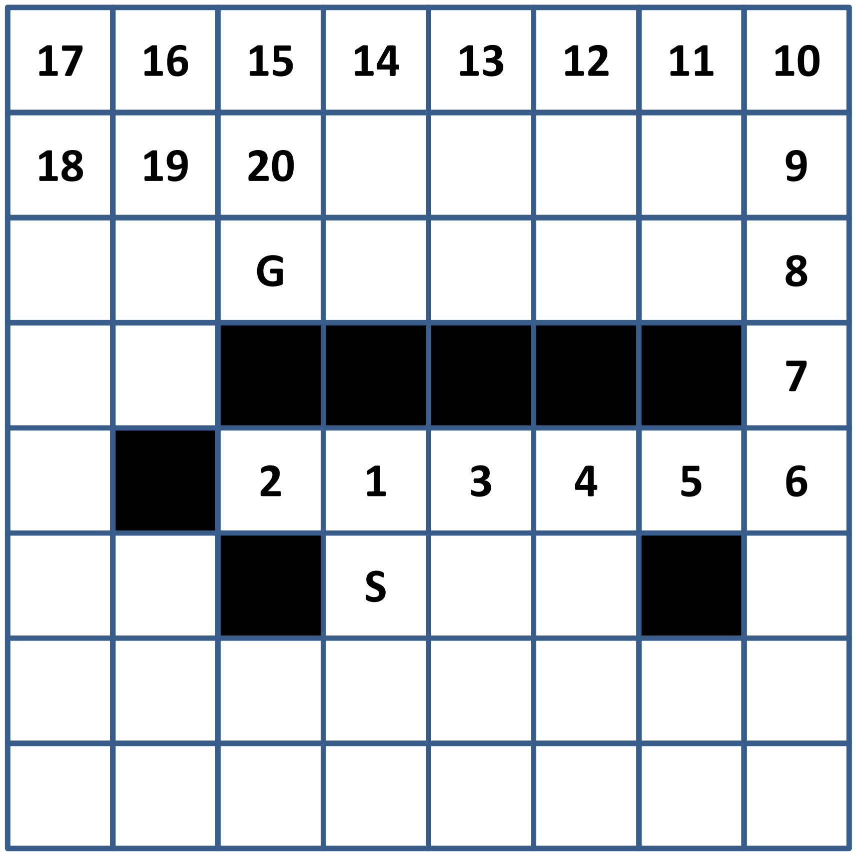
- ***Input:***
  - **QUEUE**: Path only containing root
  - **WIDTH**: Number
- ***Algorithm:***
  - **WHILE** (<u>QUEUE</u> not empty && goal not reached) **DO**
    - Remove <u>***all paths***</u> from <u>QUEUE</u>
    - Create paths to all children (of all paths)
    - Reject paths with loops
    - <u>***Sort new paths (according to heuristic)***</u>
    - <u>***(Optimization: Remove paths without successor)***</u>
    - Add <u>WIDTH</u> <u>***best paths***</u> to <u>QUEUE</u>
  - **IF** goal reached
    - **THEN** success
    - **ELSE** failure

Exercises: Artificial Intelligence

Path Search

Depth-first Search

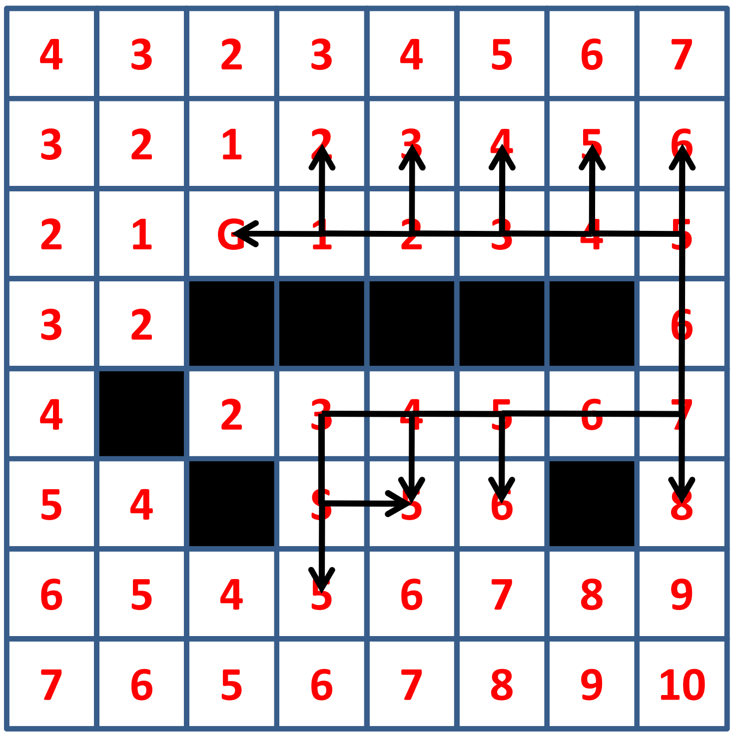


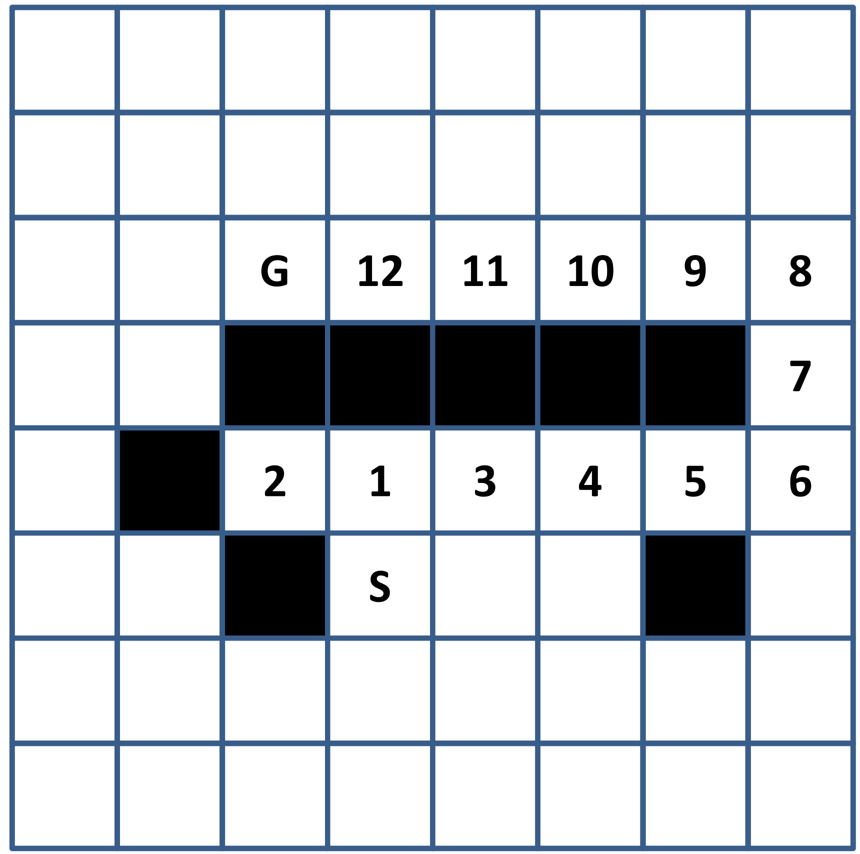


Heuristic: Manhattan Distance

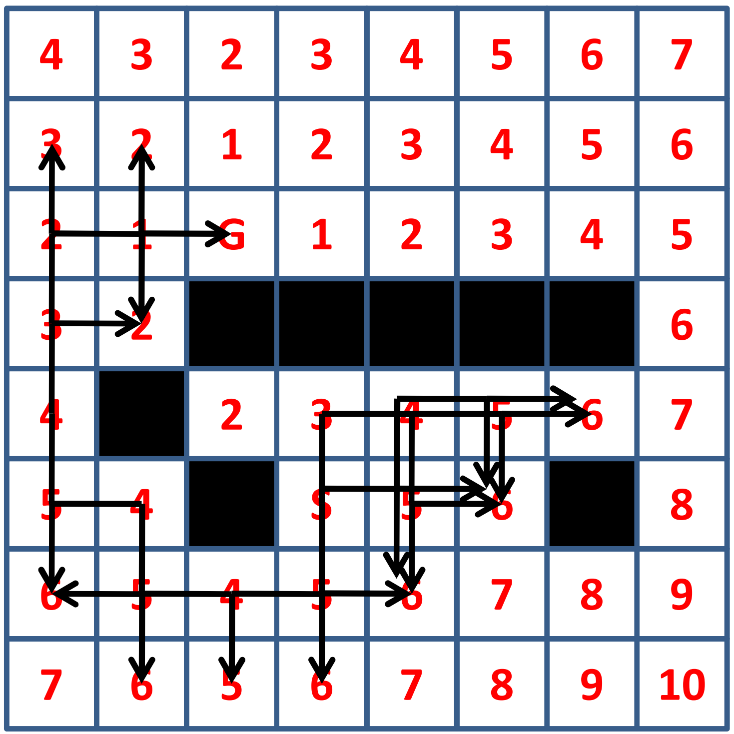
4	3	2	3	4	5	6	7
3	2	1	2	3	4	5	6
2	1	0	1	2	3	4	5
3	2						6
4		2	3	4	5	6	7
5	4		4	5	6		8
6	5	4	5	6	7	8	9
7	6	5	6	7	8	9	10

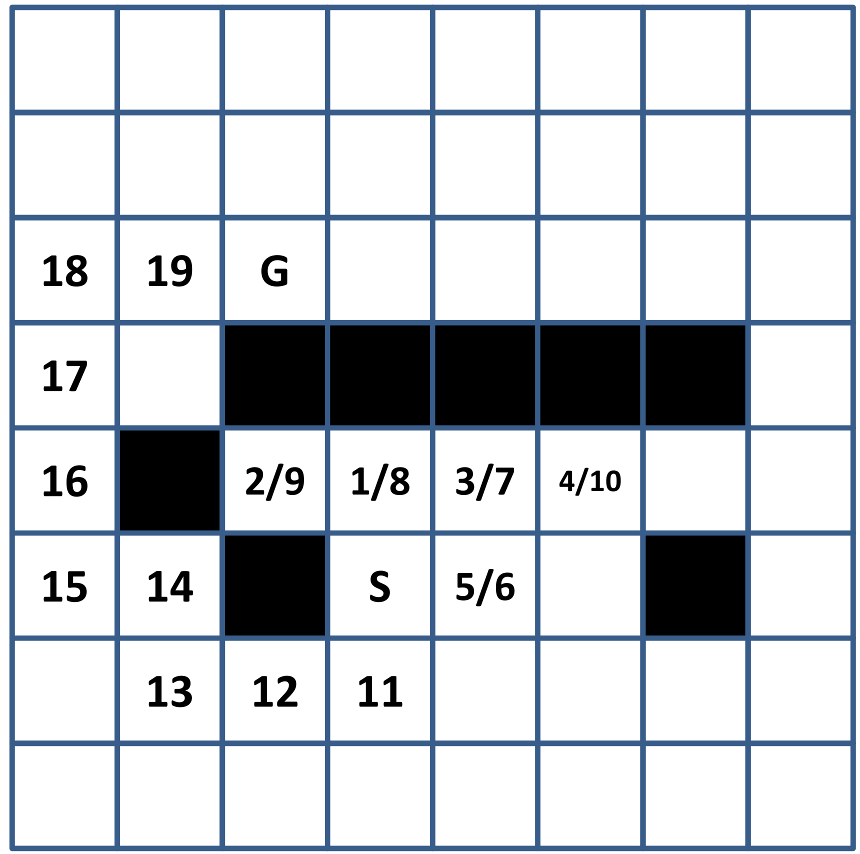
Hill-climbing I Search





Greedy Search





Exercises: Artificial Intelligence

Water Jugs

Representation

- States of the form [x,y], where:
  - x: *contents of 4 liter jug*
  - y: *contents of 3 liter jug*
- Start: [0,0]
- Goal: [2,0]

Representation

- Rules:
  - Fill x: $[x,y] \wedge x < 4 \longrightarrow [4,y]$
  - Fill y: $[x,y] \wedge y < 3 \longrightarrow [x,3]$
  - Empty x: $[x,y] \wedge x > 0 \longrightarrow [0,y]$
  - Empty y: $[x,y] \wedge y > 0 \longrightarrow [x,0]$
  - Fill x with y: $[x,y] \wedge x+y > 4 \wedge y > 0 \longrightarrow [4,(x+y-4)]$
  - Fill x with y: $[x,y] \wedge x+y \leq 4 \wedge y > 0 \longrightarrow [(x+y),0]$
  - Fill y with x: $[x,y] \wedge x+y > 3 \wedge x > 0 \longrightarrow [(x+y-3),3]$
  - Fill y with x: $[x,y] \wedge x+y \leq 3 \wedge x > 0 \longrightarrow [0,(x+y)]$

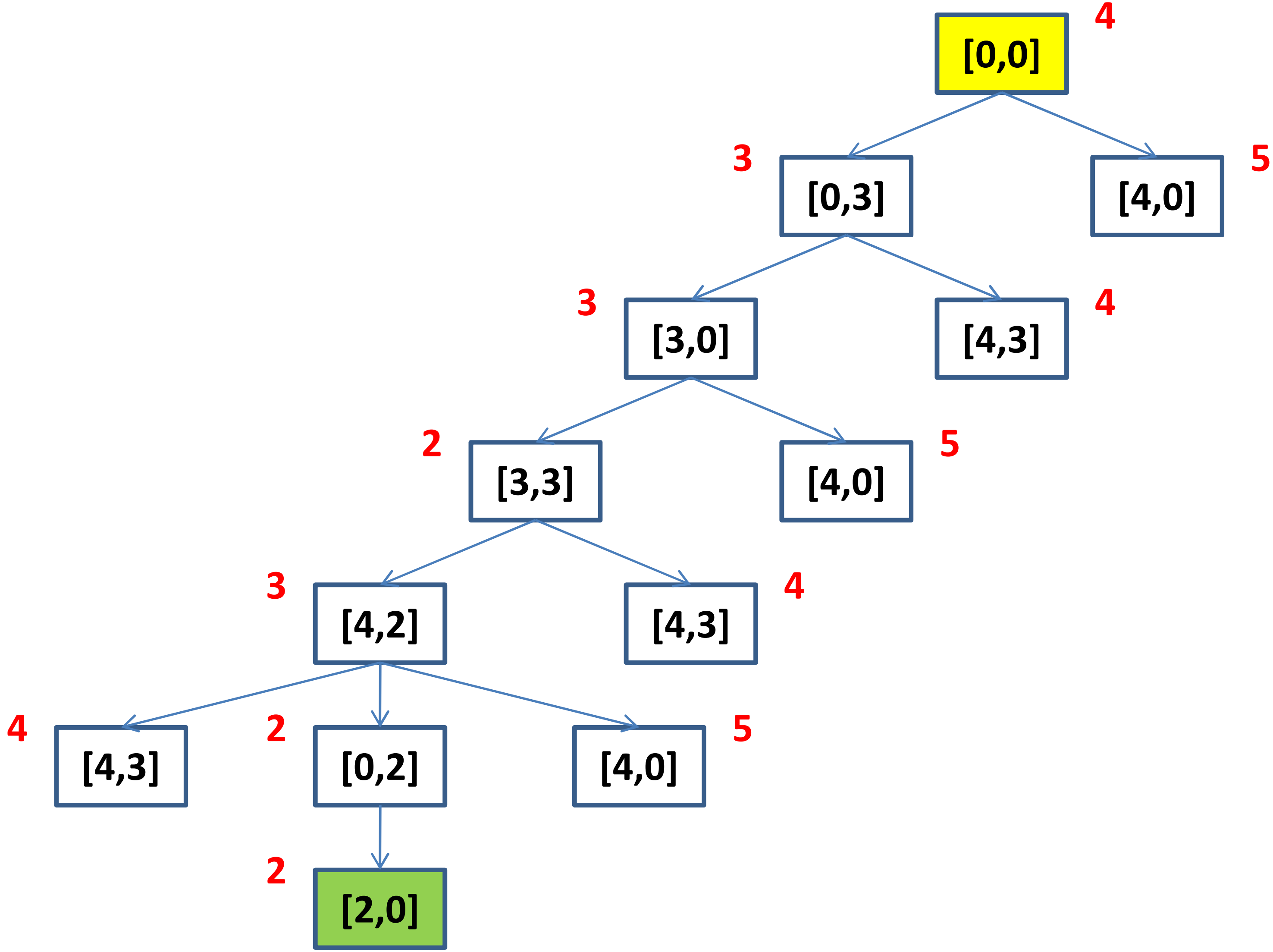
Heuristic

- $H([x,y]) = f(x) + f(y)$
- $f(x)$ is defined as follows:

x	0	1	2	3	4
f(x)	2	1	0	1	3

  – We need a jug filled with 2 liter.
  – To obtain a jug filled with 2 liter we need a jug filled with either 1 or 3 liter.
  – We consider an empty jug better than a jug filled with 4 liter.

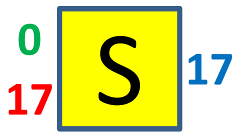
Hill-climbing II Search

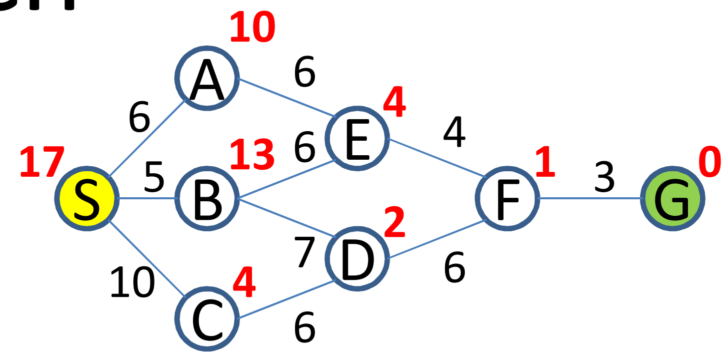


Exercises: Artificial Intelligence

A*

A* Search

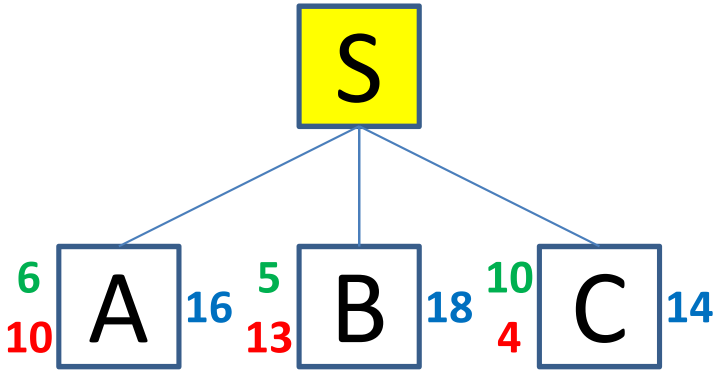


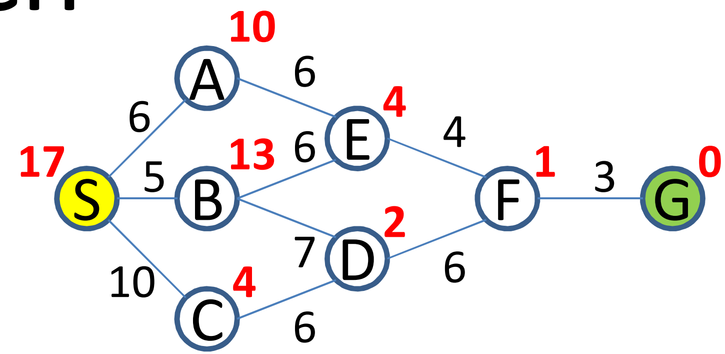


QUEUE:

S

A* Search





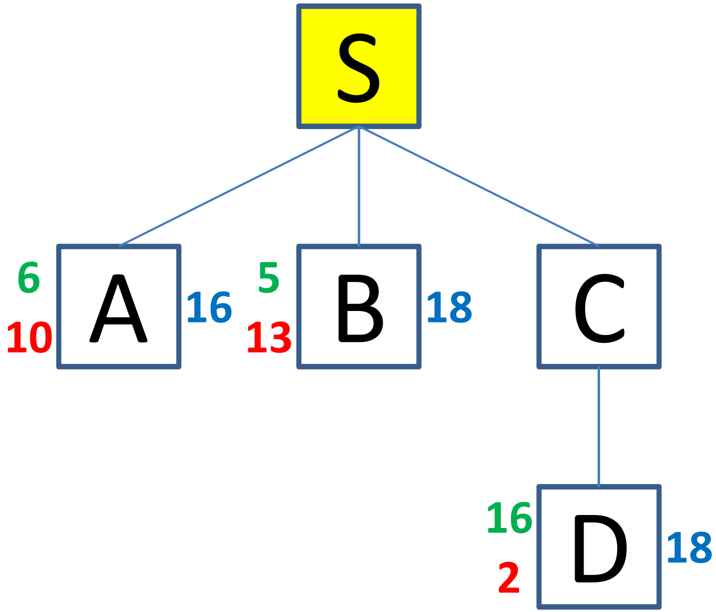
QUEUE:

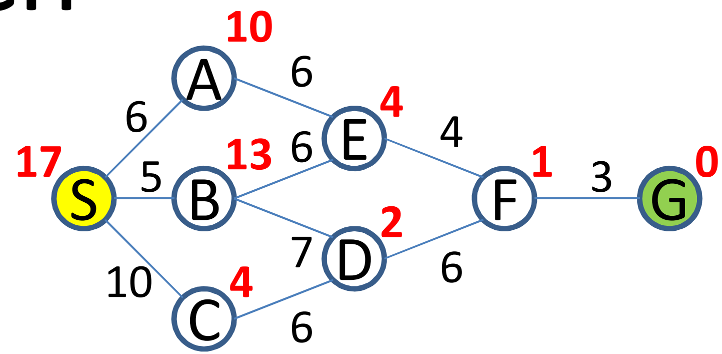
SC

SA

SB

A* Search





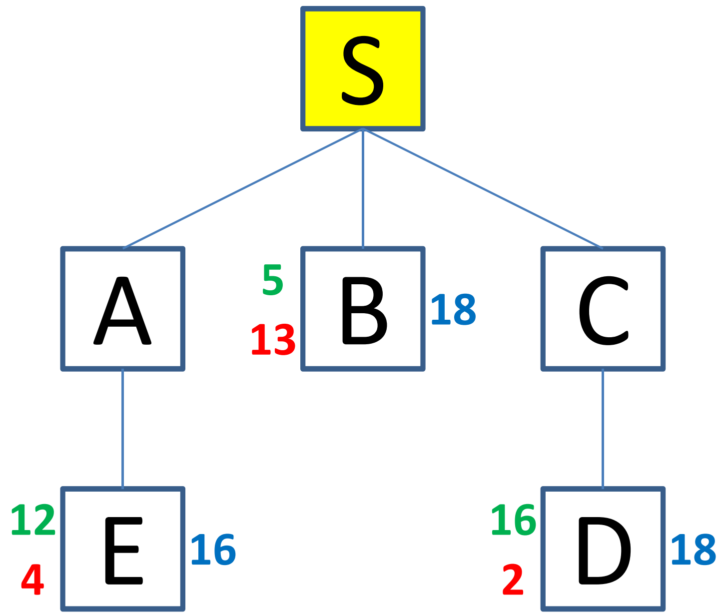
QUEUE:

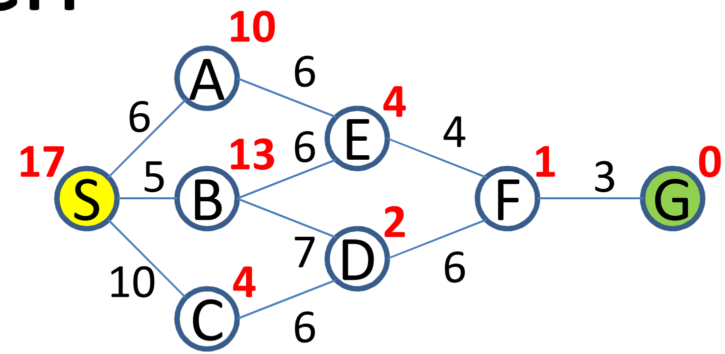
SA

SCD

SB

A* Search





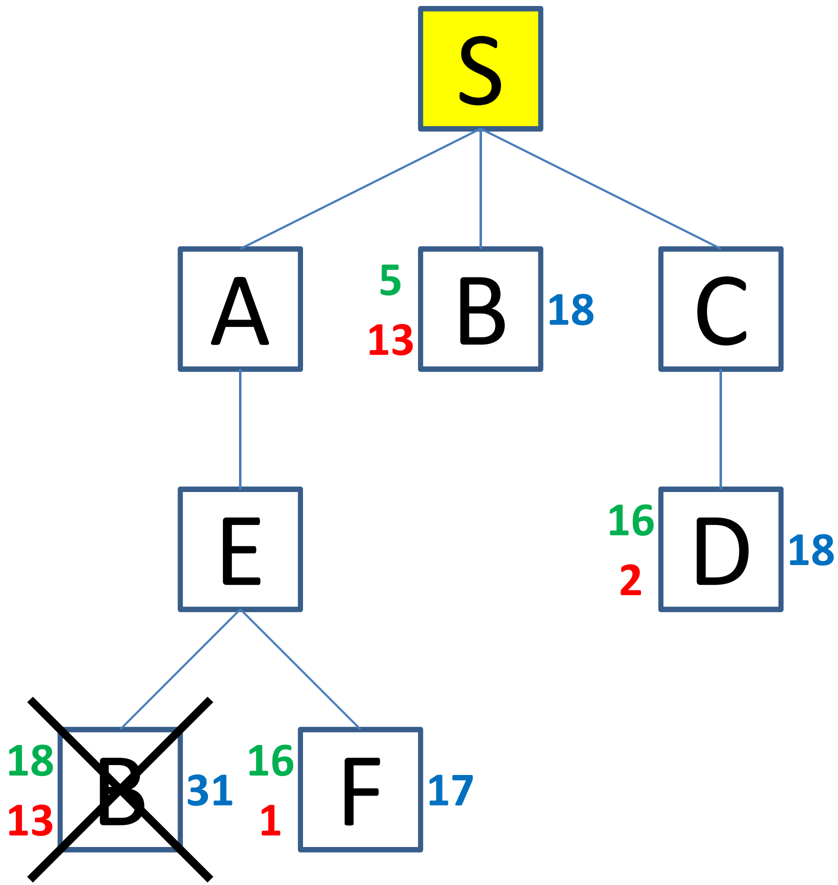
QUEUE:

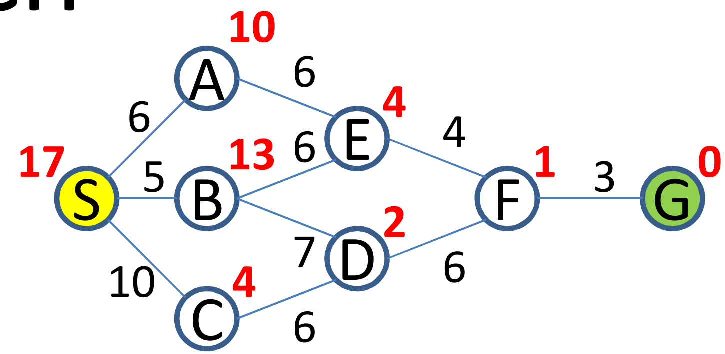
SAE

SCD

SB

A* Search





<u>QUEUE:</u>

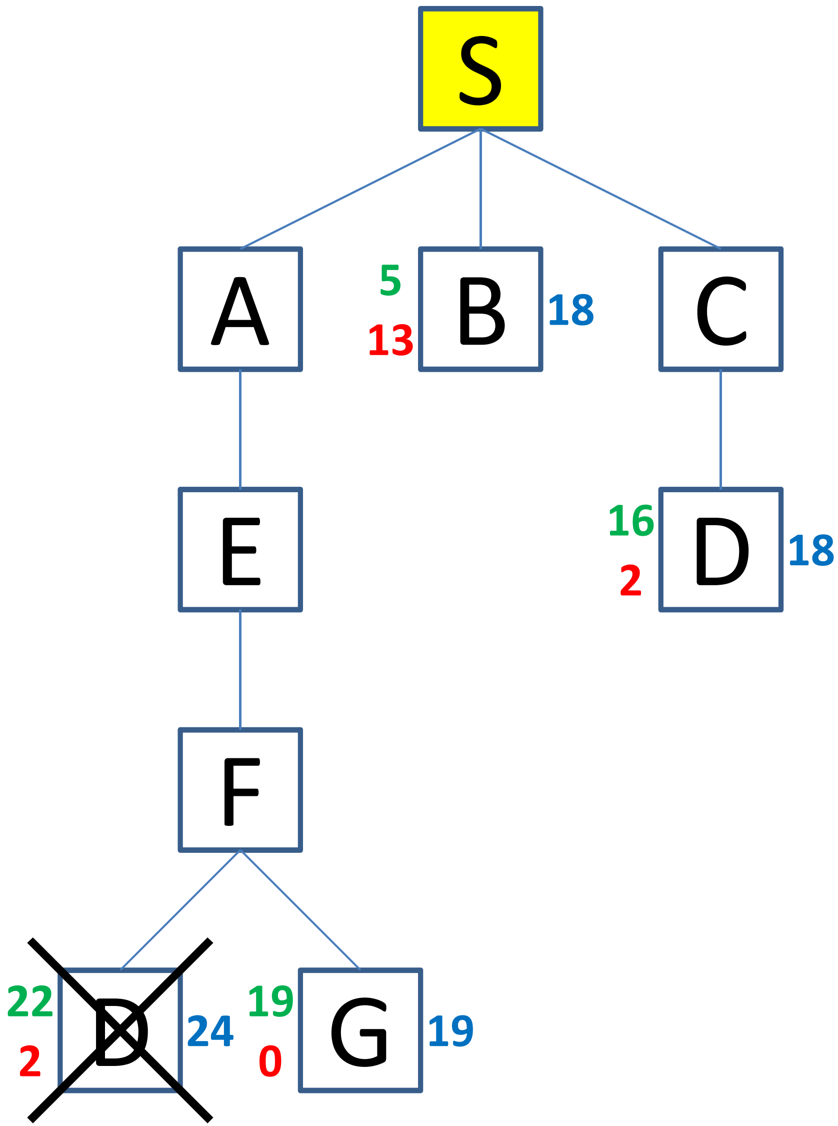
SAEF

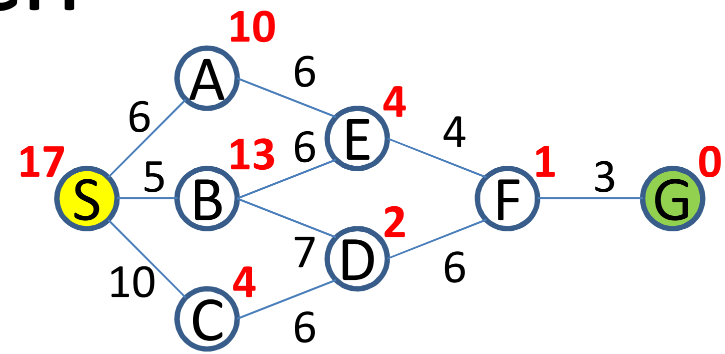
SCD

SB

<u>**SAEB**</u>

A* Search





QUEUE:

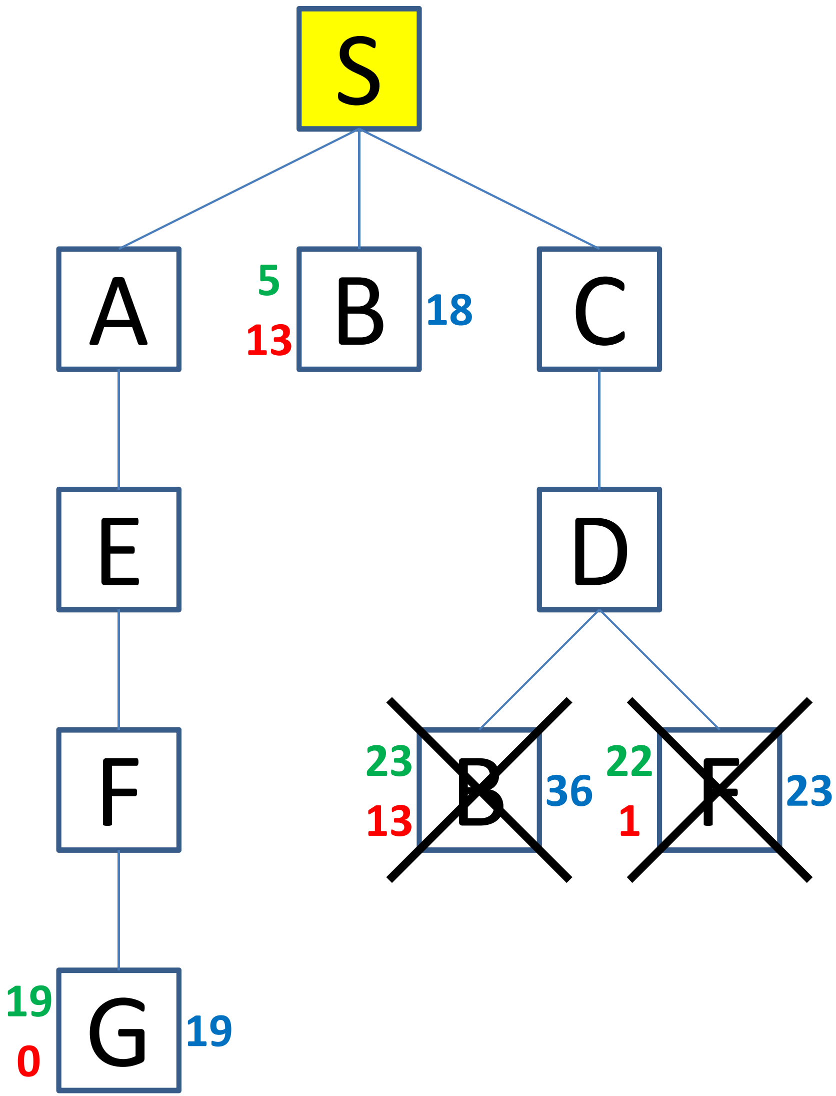
SCD

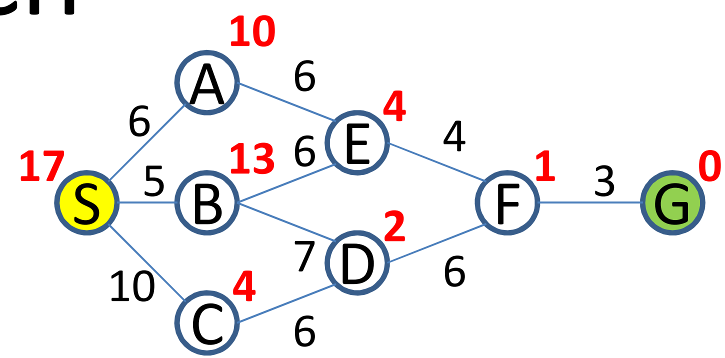
SB

SAEFG

SAEFD

A* Search





QUEUE:

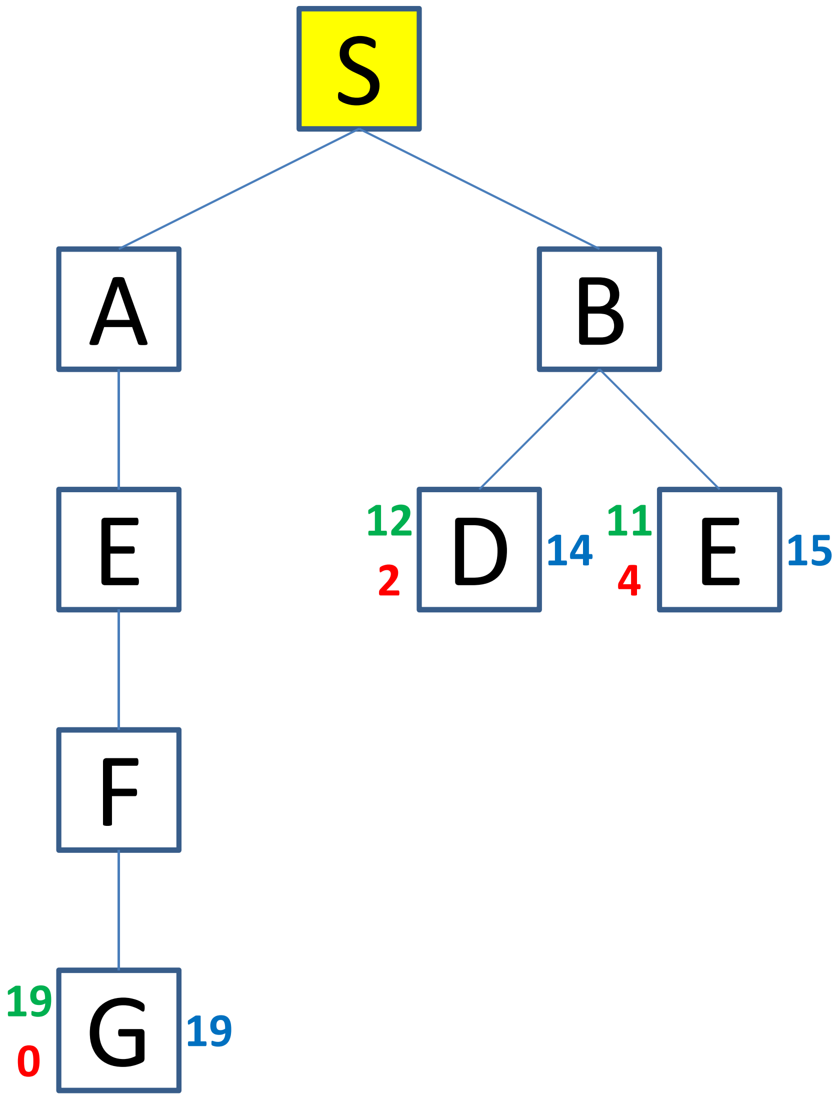
SB

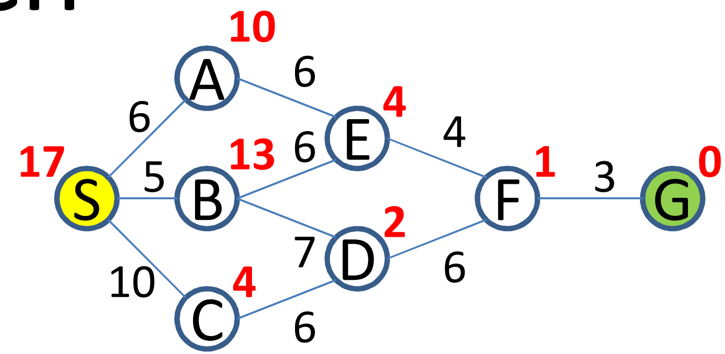
SAEFG

SCDF

SCDB

A* Search





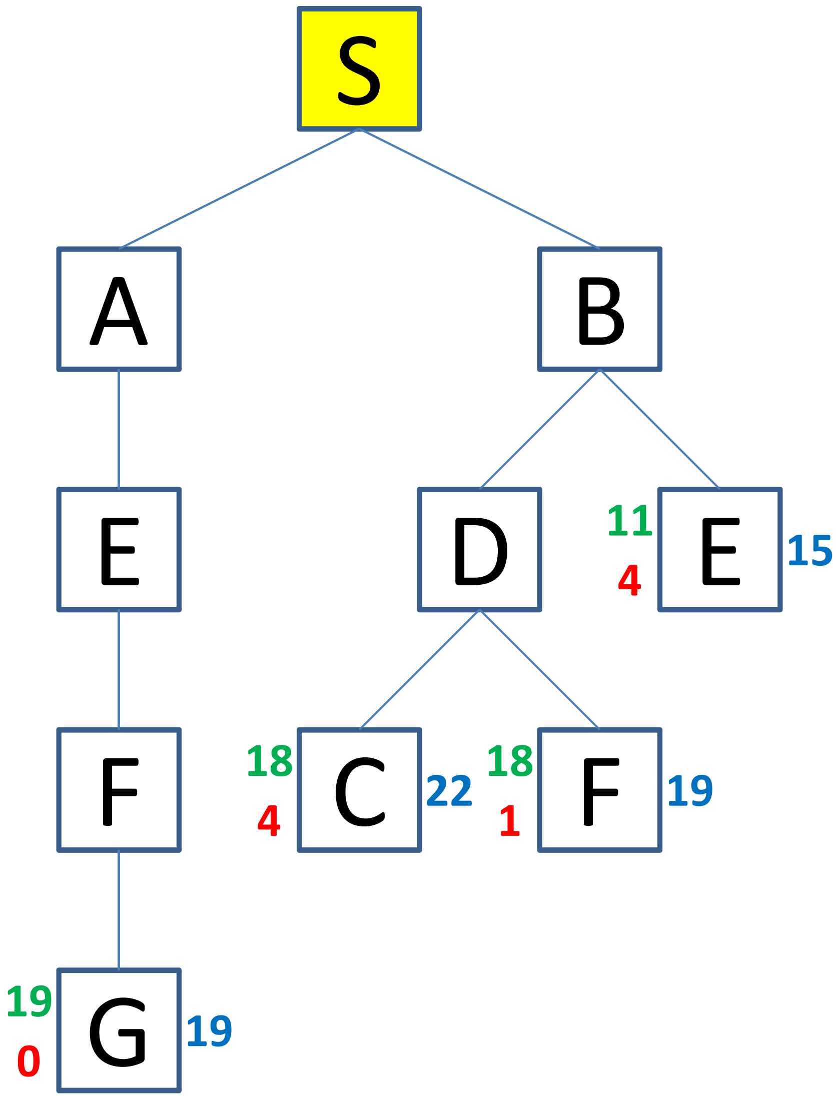
QUEUE:

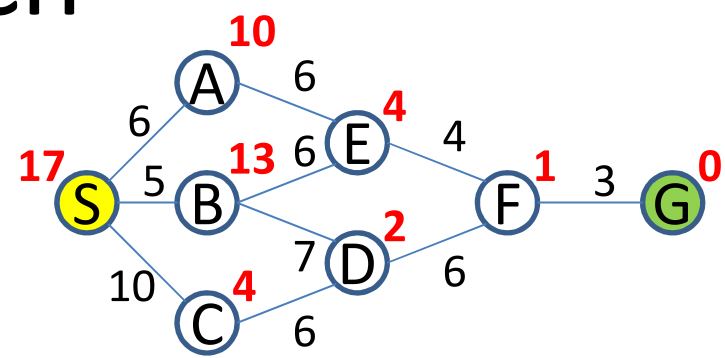
SBD

SBE

SAEFG

A* Search





QUEUE:

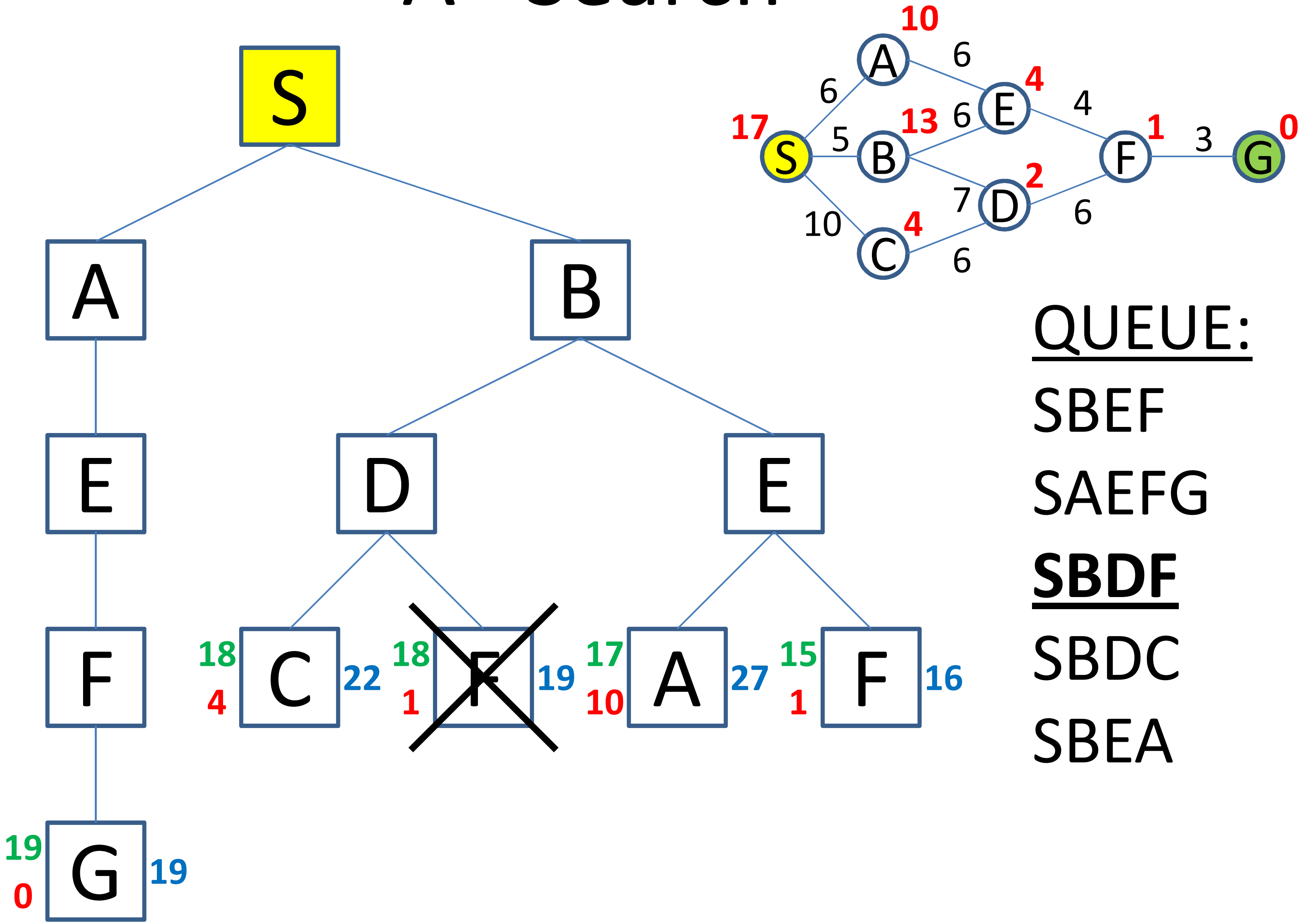
SBE

SBDF

SAEFG

SBDC

A* Search



QUEUE:

SBEF

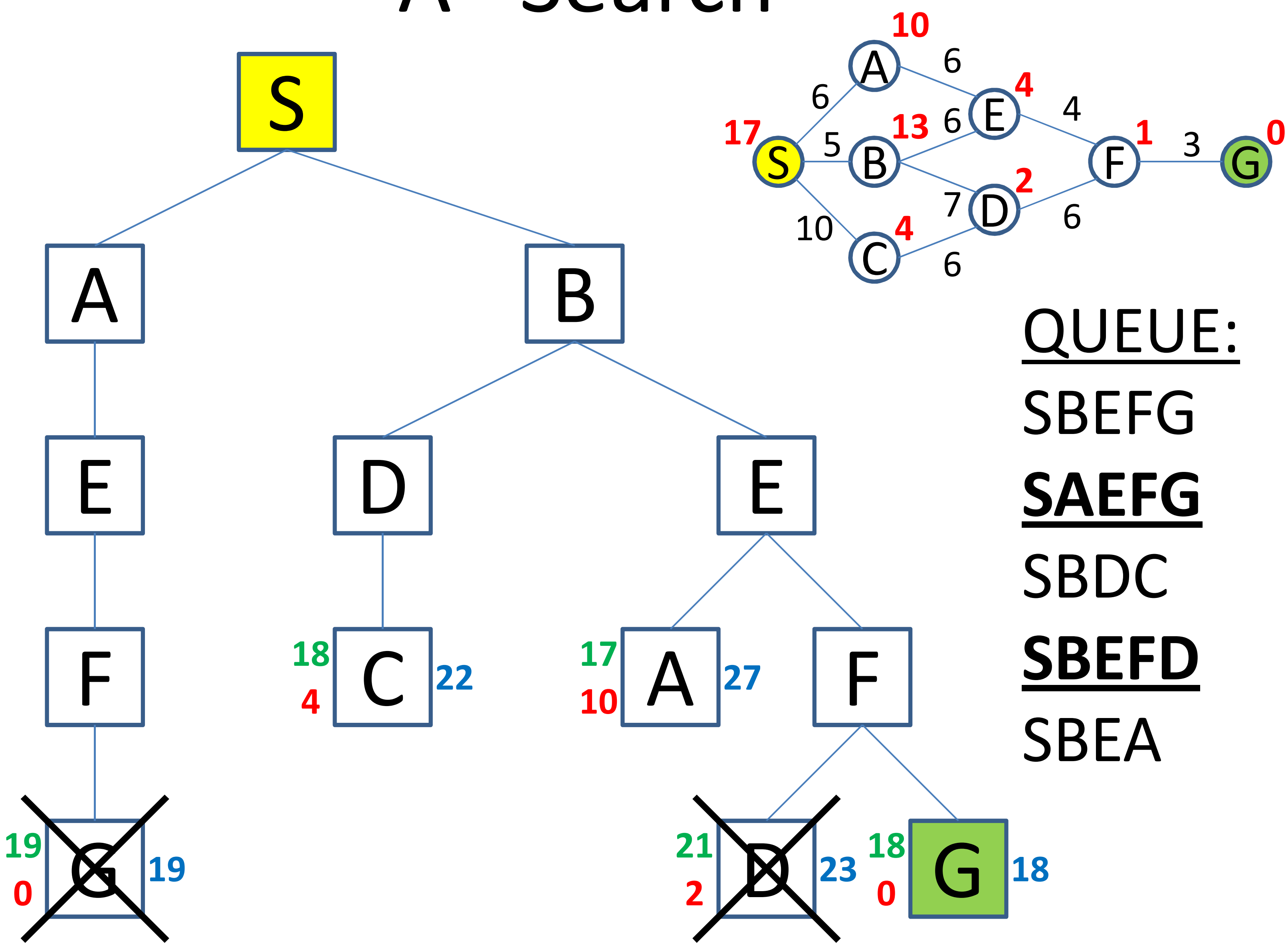
SAEFG

SBDF

SBDC

SBEA

A* Search



<u>QUEUE:</u>

SBEFG

<u>**SAEFG**</u>

SBDC

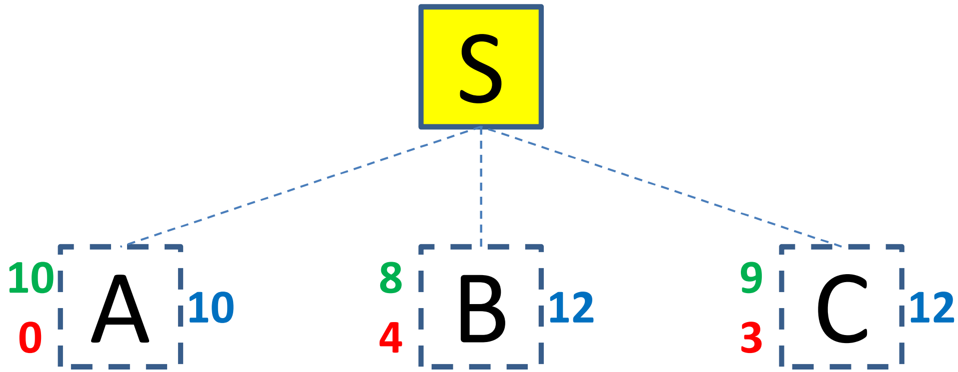
<u>**SBEFD**</u>

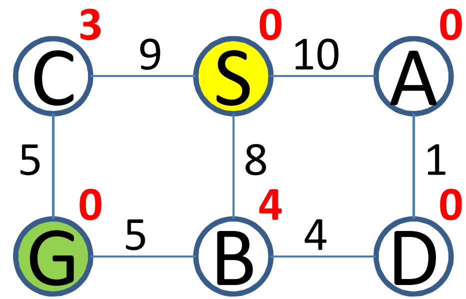
SBEA

Exercises: Artificial Intelligence

Iterated Deepening A*

IDA* Search



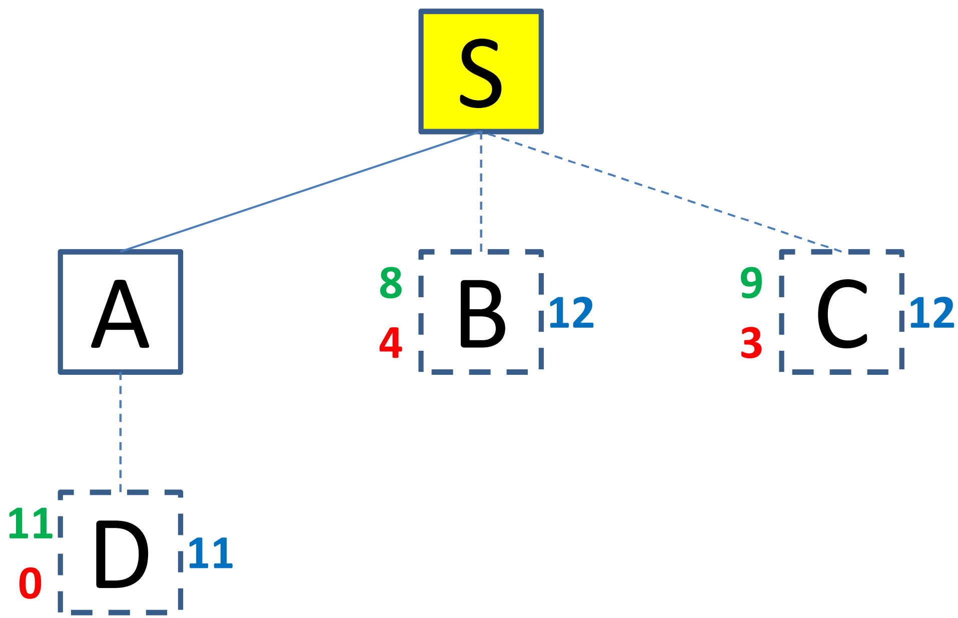


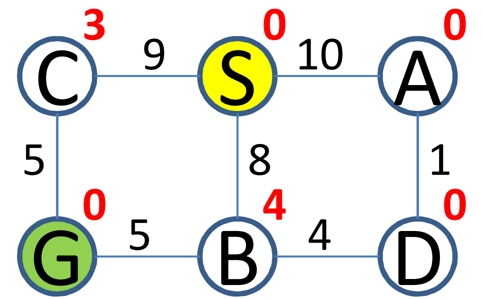
f-bound = 0

f-new = 10

Children are explored <u>depth-first</u>!

IDA* Search

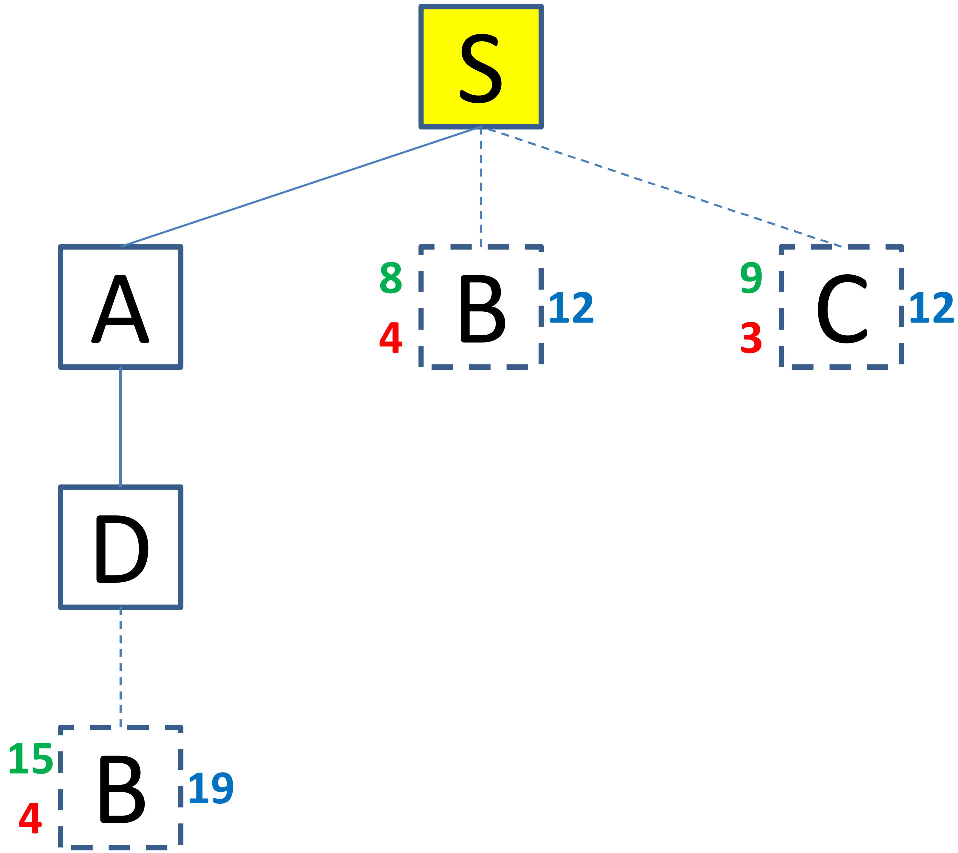


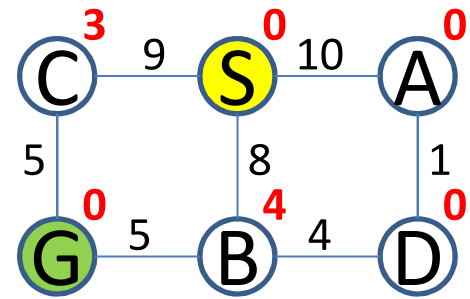


f-bound = 10

f-new = 11

IDA* Search

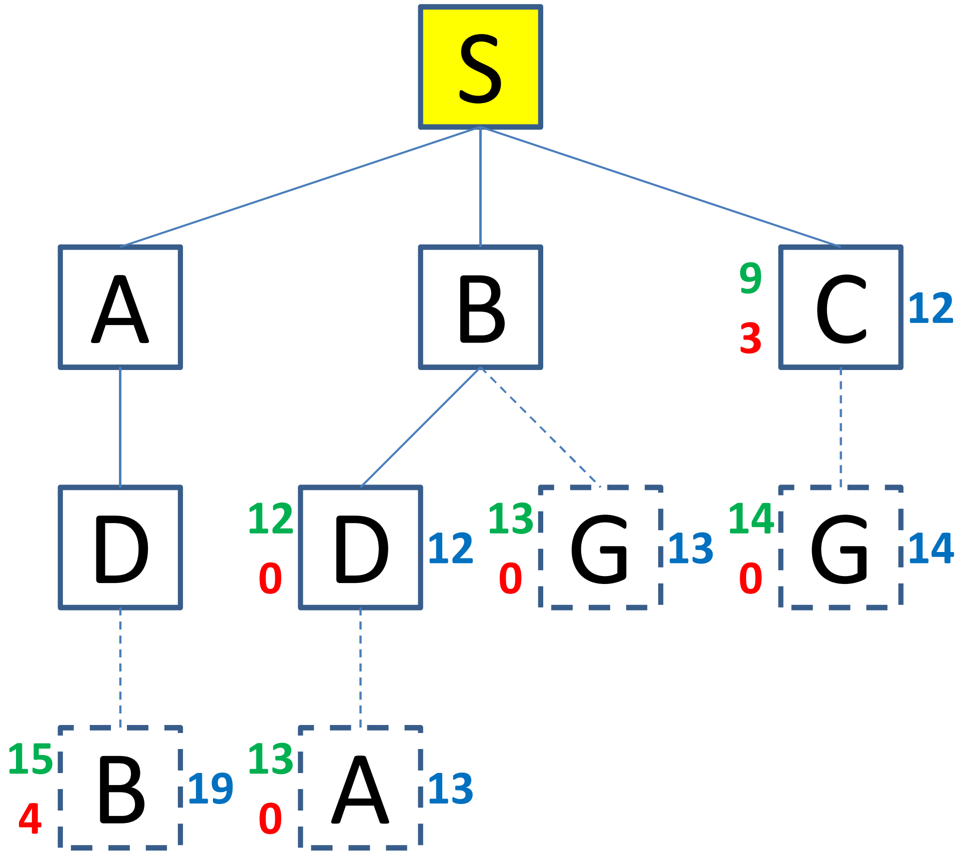


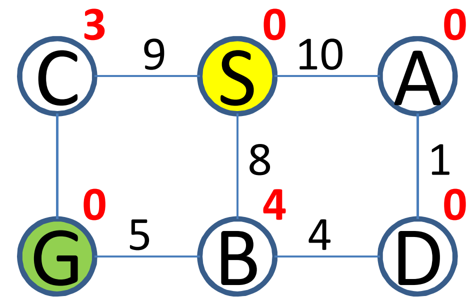


f-bound = 11

f-new = 12

IDA* Search

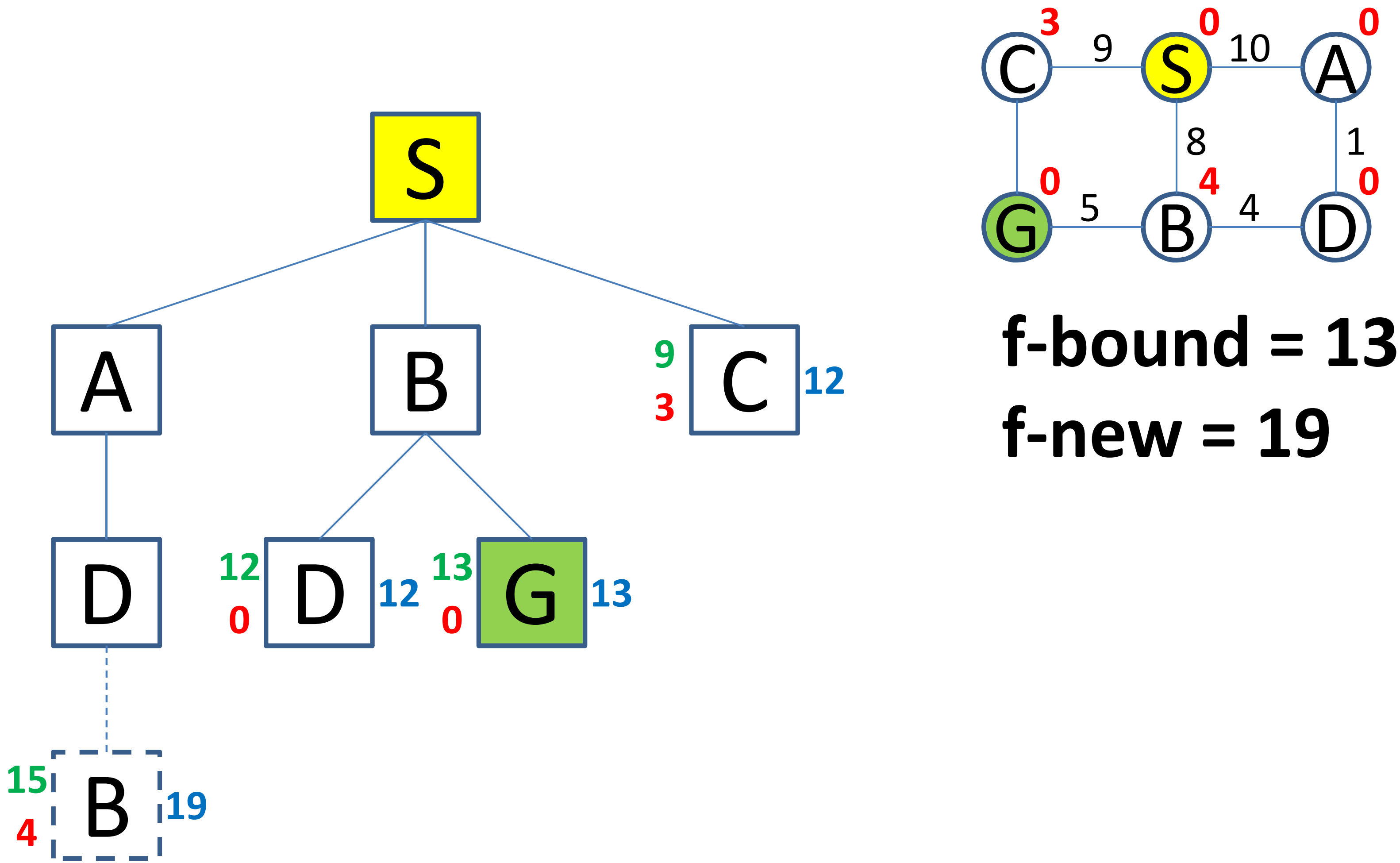




f-bound = 12

f-new = 13

IDA* Search



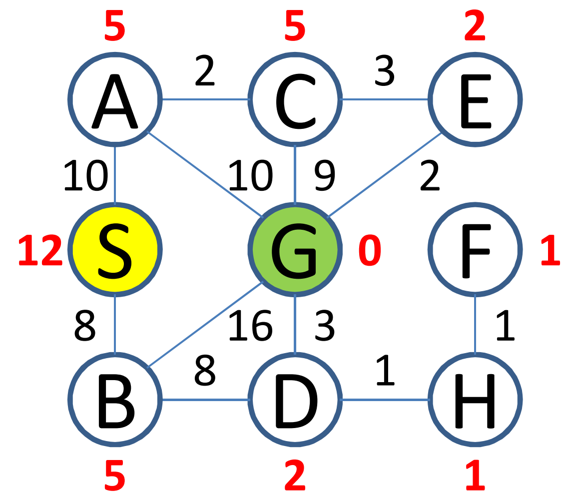
f-bound = 13

f-new = 19

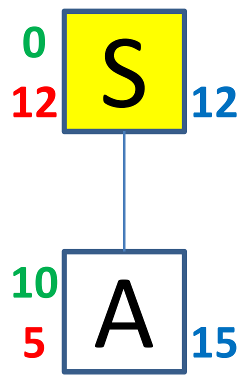
Exercises: Artificial Intelligence

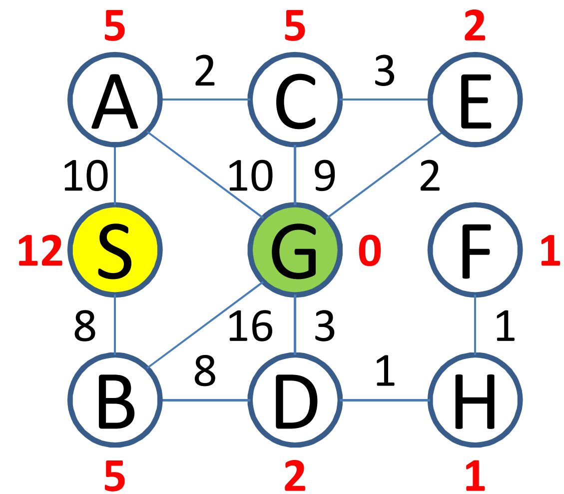
Simplified Memory-bounded A*

Problem

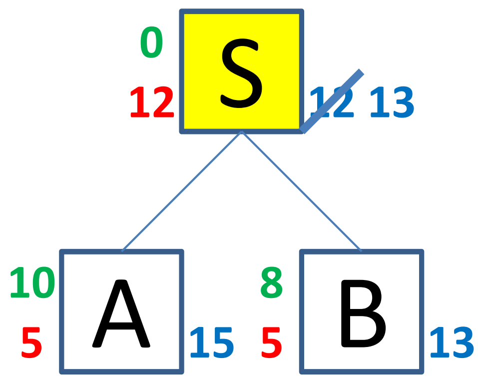


Problem

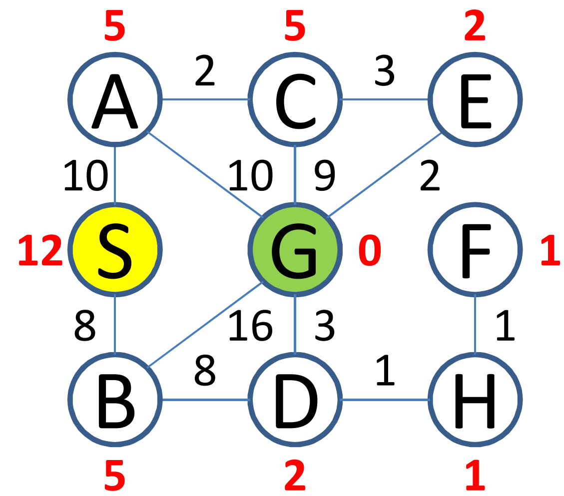




Problem

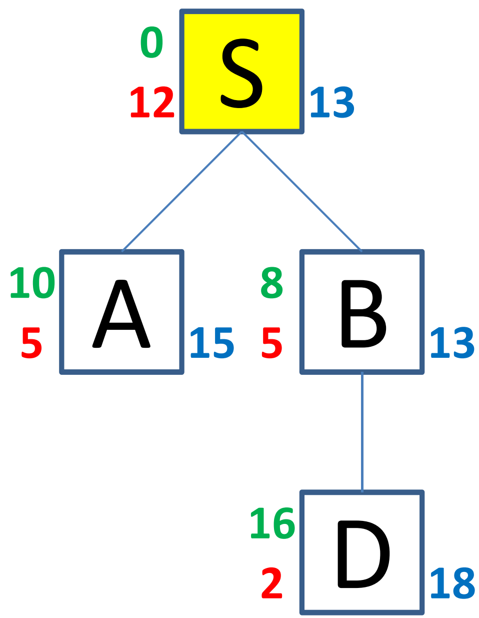


S
0
12
~~12~~ 13
A
10
5
15
B
8
5
13

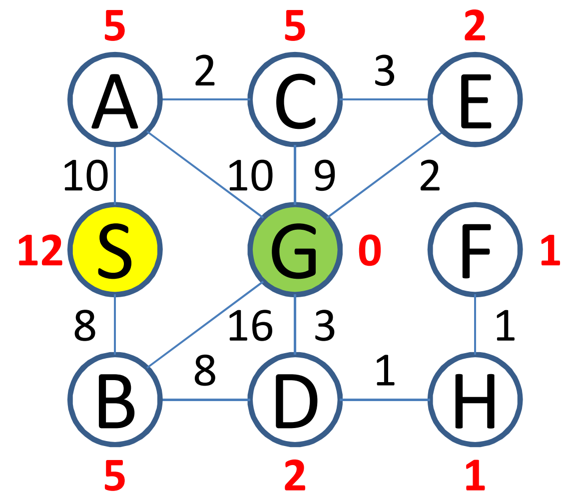


A 5
C 5
E 2
S 12
G 0
F 1
B 5
D 2
H 1
2
3
10
10
9
2
8
16
3
1
8
1

Problem

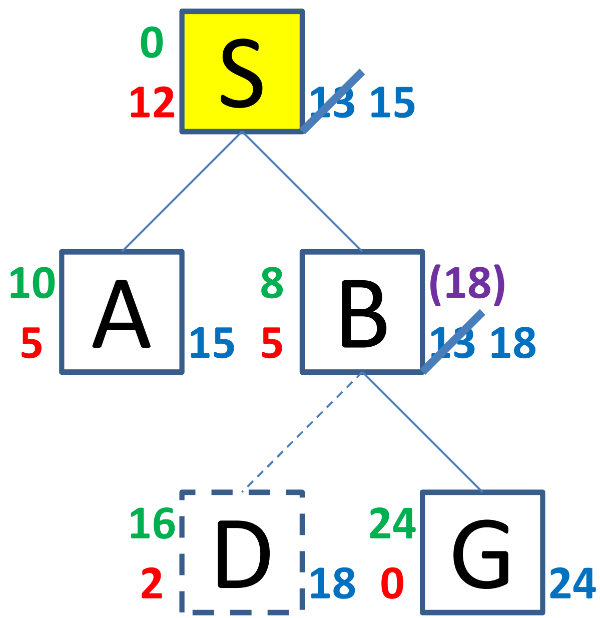


0
12
S
13
10
5
A
15
8
5
B
13
16
2
D
18

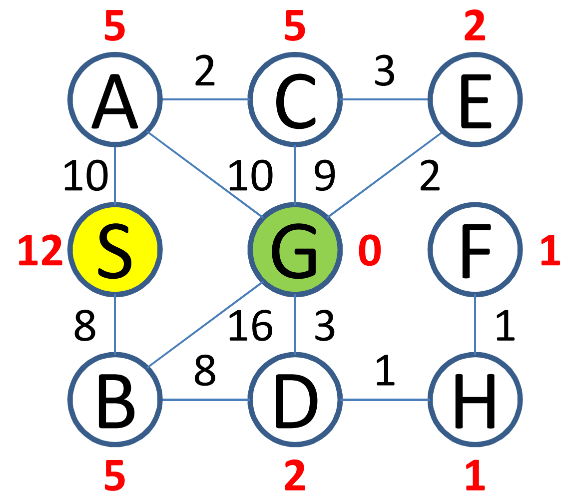


5
5
2
A
2
C
3
E
10
10
9
2
12
S
G
0
F
1
8
16
3
1
B
8
D
1
H
5
2
1

Problem

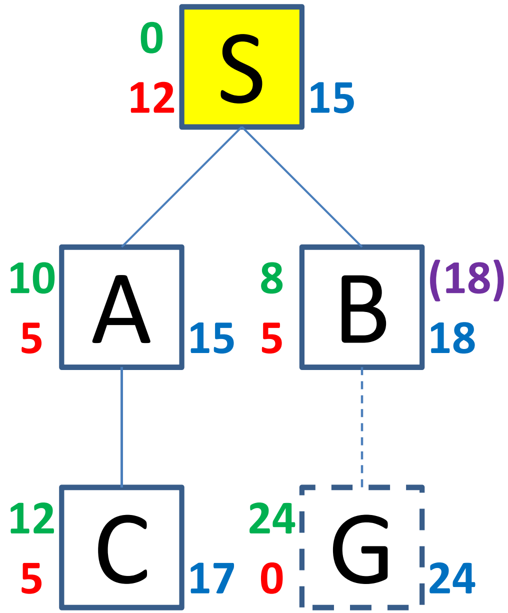


S
0
12
~~13~~ 15
A
10
5
15
B
8
5
(18)
~~13~~ 18
D
16
2
18
G
24
0
24

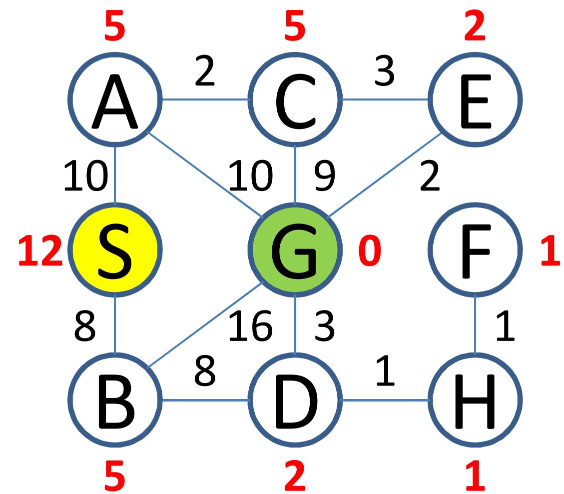


A
C
E
S
G
F
B
D
H
5
5
2
12
0
1
5
2
1
2
3
10
10
9
2
8
16
3
1
8
1

Problem

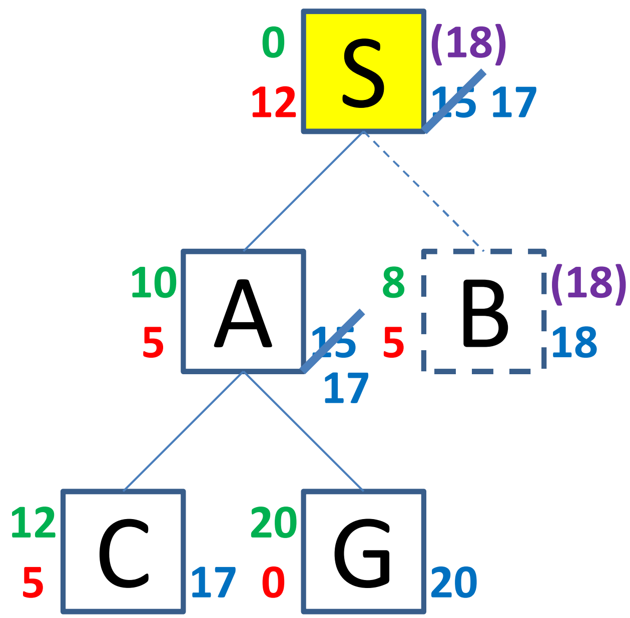


0
12
S
15
10
5
A
15
8
5
B
(18)
18
12
5
C
17
24
0
G
24

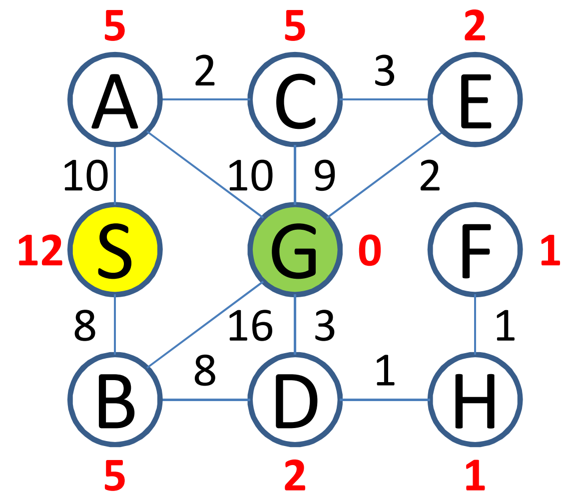


5
5
2
A
2
C
3
E
10
10
9
2
12
S
G
0
F
1
8
16
3
1
8
1
B
D
H
5
2
1

Problem

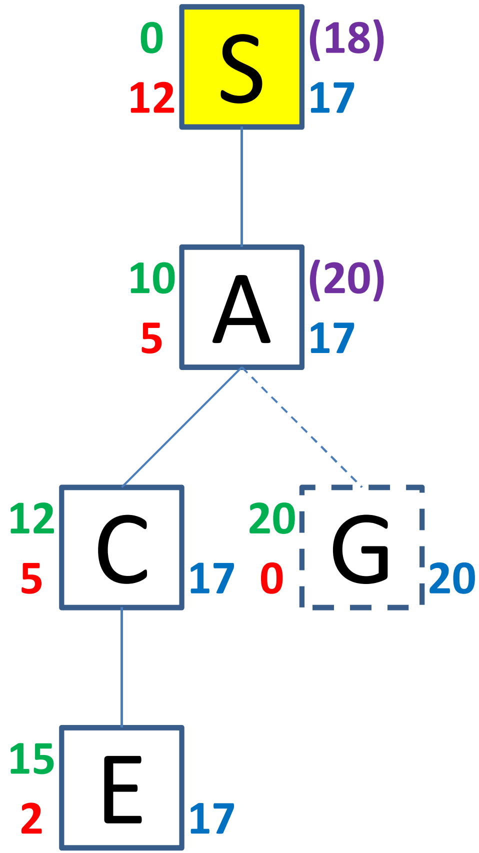


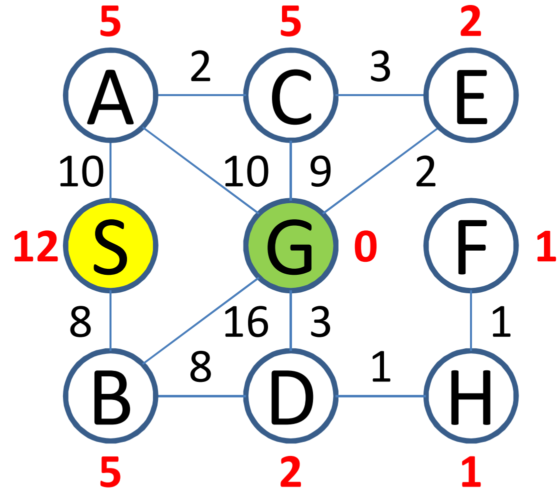
0
12
S
(18)
~~15~~ 17
10
5
A
~~15~~
17
8
5
B
(18)
18
12
5
C
17
20
0
G
20



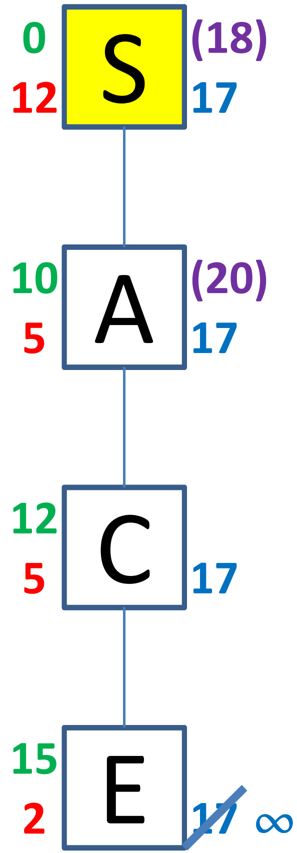
5
5
2
A
2
C
3
E
10
10
9
2
12
S
G
0
F
1
8
16
3
1
B
8
D
1
H
5
2
1

Problem

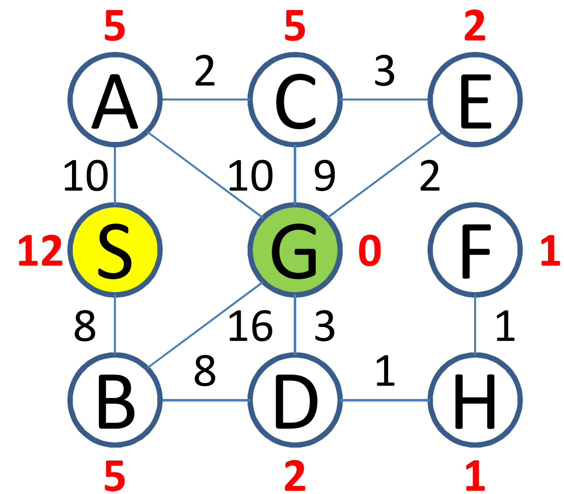

0
S
(18)
12
17
10
A
(20)
5
17
12
C
5
17
20
G
0
20
15
E
2
17


5
5
2
A
2
C
3
E
10
10
9
2
12
S
G
0
F
1
8
16
3
1
8
1
B
D
H
5
2
1

Problem

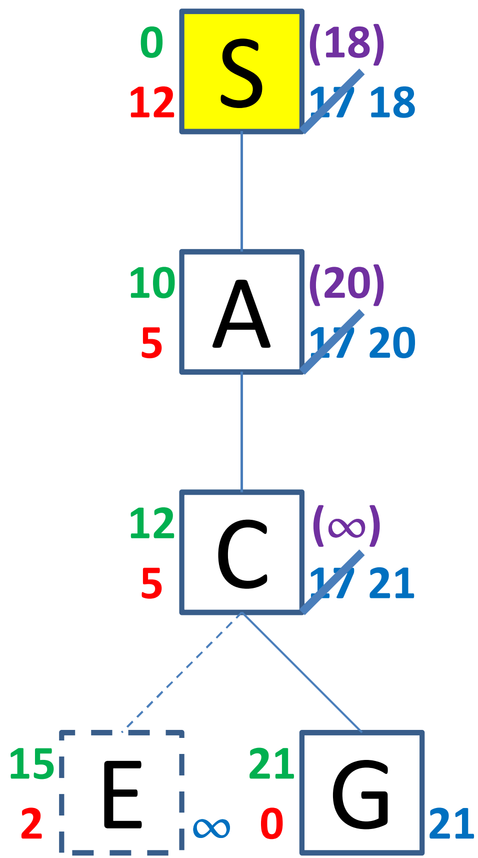


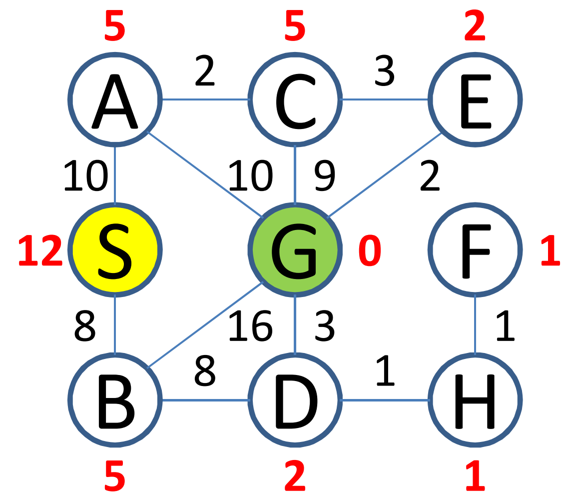
0
S
(18)
12
17
10
A
(20)
5
17
12
C
5
17
15
E
2
~~17~~ ∞



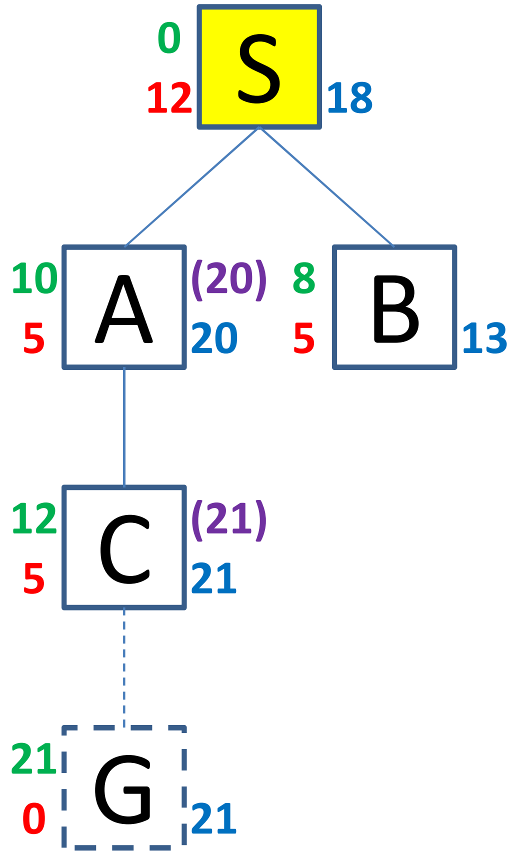
5 5 2
A 2 C 3 E
10 10 9 2
12 S G 0 F 1
8 16 3 1
B 8 D 1 H
5 2 1

Problem

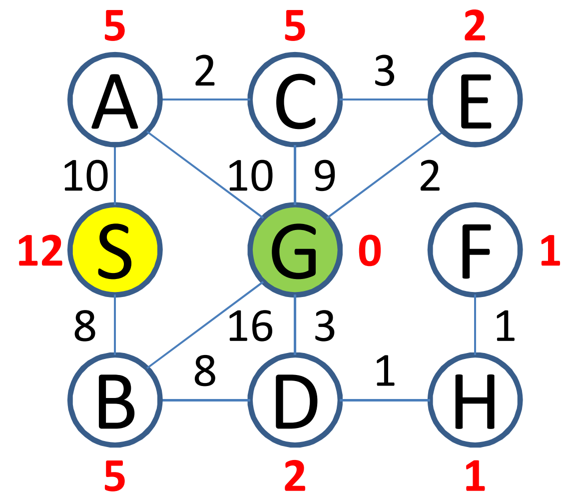

0
12
S
(18)
17 18
10
5
A
(20)
17 20
12
5
C
(∞)
17 21
15
2
E
∞
21
0
G
21


5
5
2
A
2
C
3
E
10
10
9
2
12
S
G
0
F
1
8
16
3
1
B
8
D
1
H
5
2
1

Problem

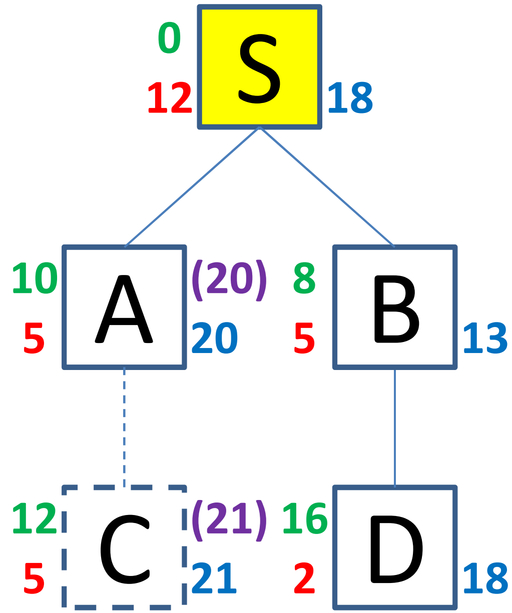


0
12
S
18
10
5
A
(20)
20
8
5
B
13
12
5
C
(21)
21
21
0
G
21

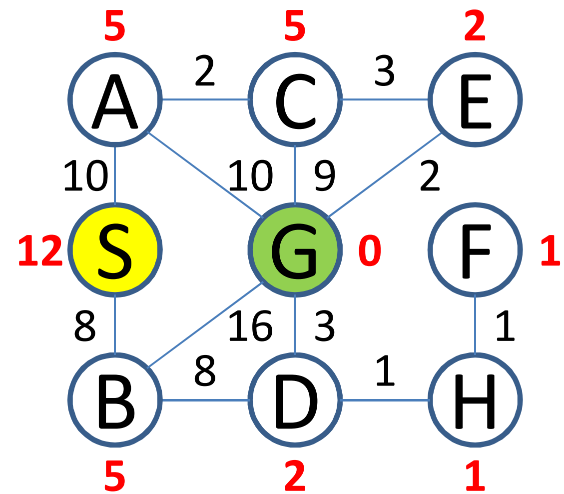


5
5
2
A
2
C
3
E
10
10
9
2
12
S
G
0
F
1
8
16
3
1
B
8
D
1
H
5
2
1

Problem

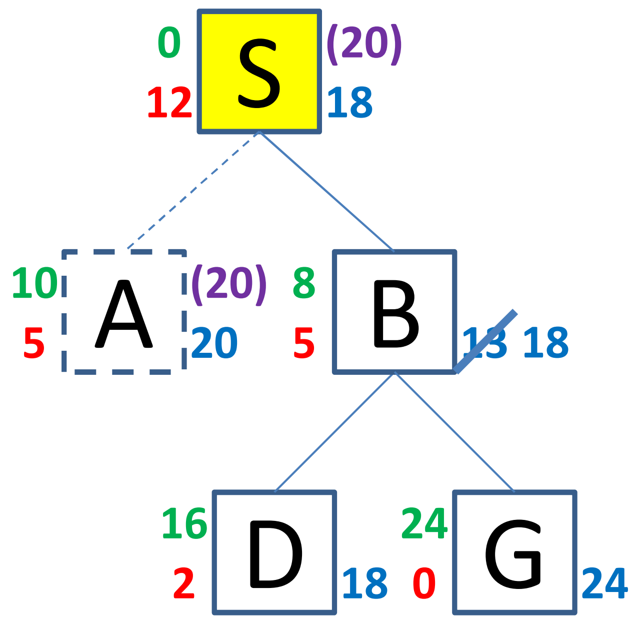


0
12
S
18
10
5
A
(20)
20
8
5
B
13
12
5
C
(21)
21
16
2
D
18

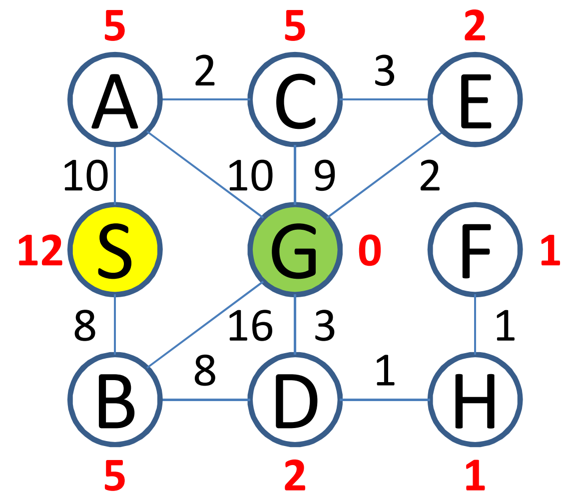


5
A
2
C
3
E
5
2
10
10
9
2
12
S
G
0
F
1
8
16
3
1
B
8
D
1
H
5
2
1

Problem

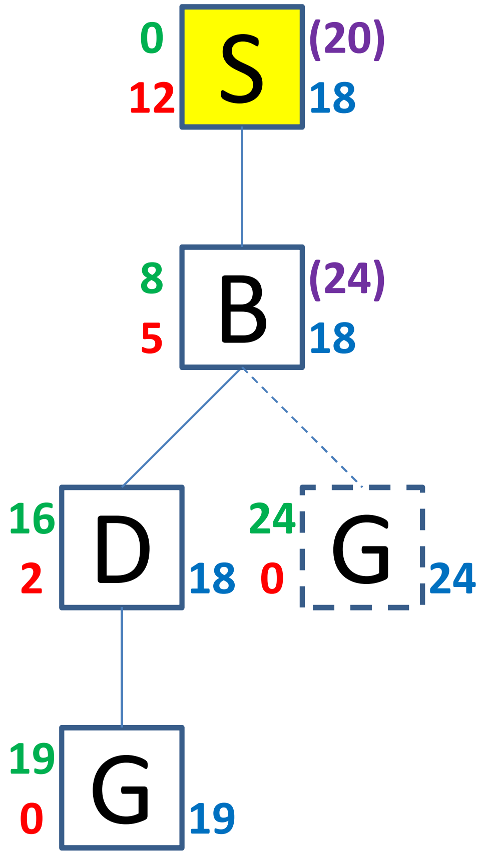


0
S
(20)
12
18
10
A
(20)
5
20
8
B
5
~~13~~ 18
16
D
2
18
24
G
0
24

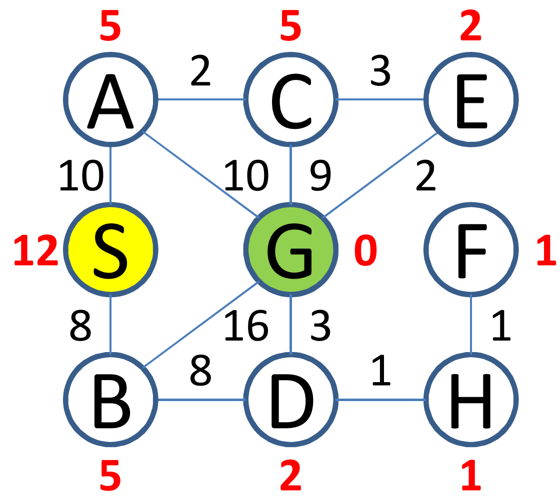


A 5
C 5
E 2
S 12
G 0
F 1
B 5
D 2
H 1
2
3
10
10
9
2
8
16
3
1
8
1

Problem

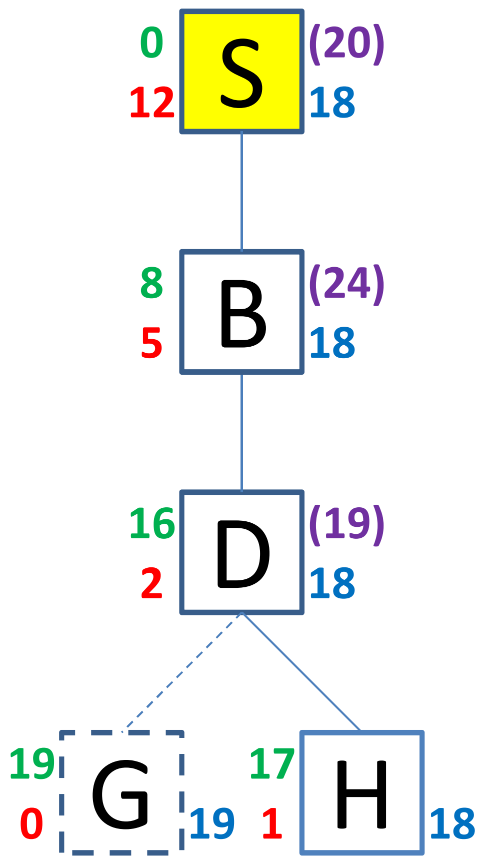


0
S
(20)
12
18
8
B
(24)
5
18
16
D
2
18
24
G
0
24
19
G
0
19

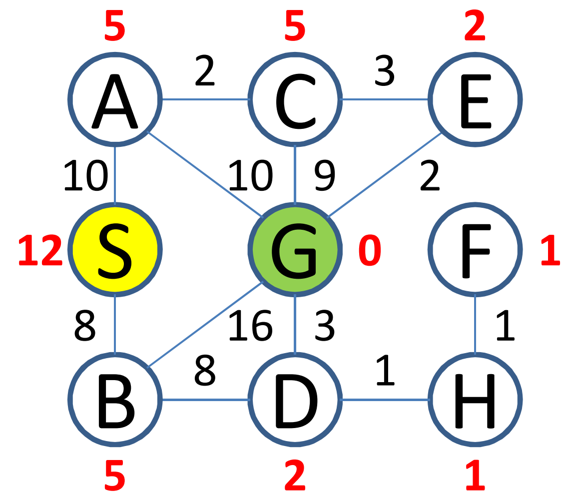


5
5
2
A
2
C
3
E
10
10
9
2
12
S
G
0
F
1
8
16
3
1
B
8
D
1
H
5
2
1

Problem

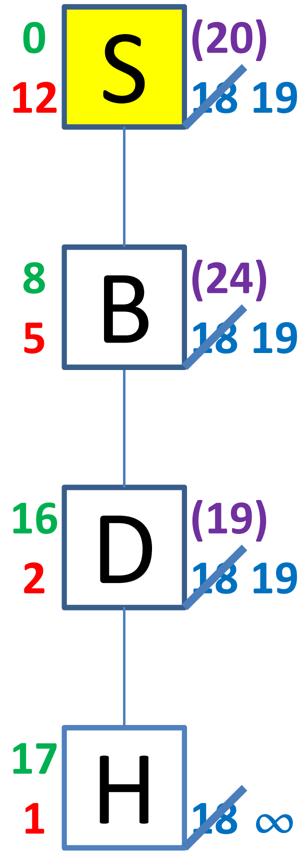


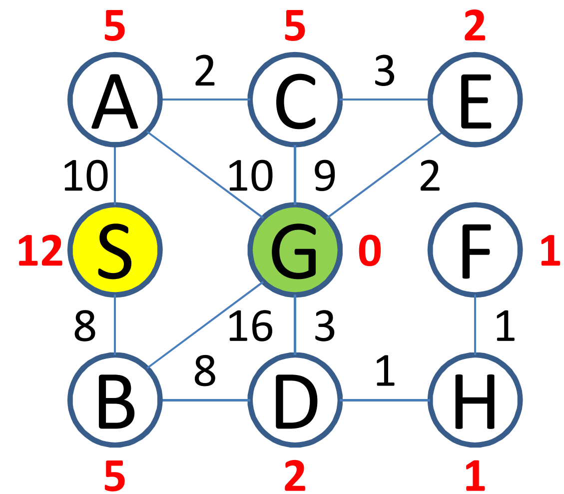
0
12
S
(20)
18
8
5
B
(24)
18
16
2
D
(19)
18
19
0
G
19
17
1
H
18



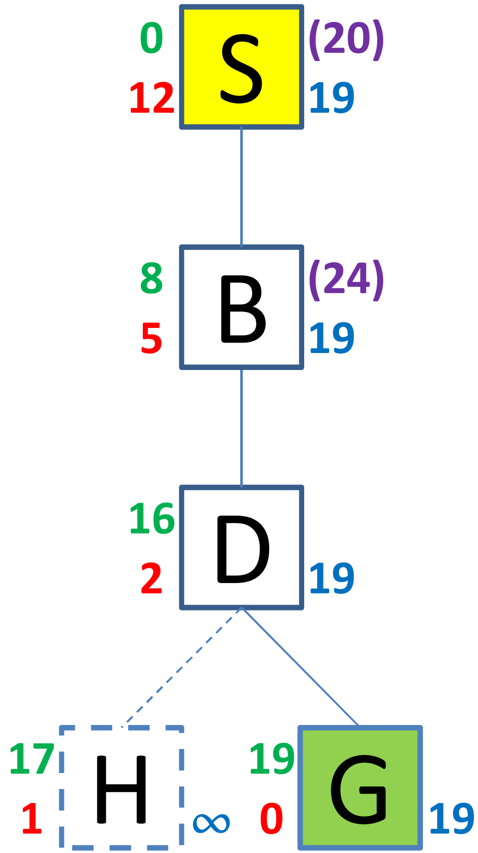
5 5 2
A 2 C 3 E
10 10 9 2
12 S G 0 F 1
8 16 3 1
B 8 D 1 H
5 2 1

Problem

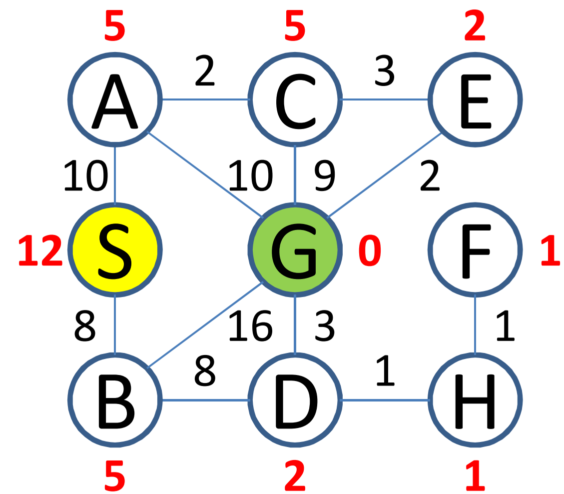

0
S
(20)
12
~~18~~ 19
8
B
(24)
5
~~18~~ 19
16
D
(19)
2
~~18~~ 19
17
H
1
~~18~~ ∞


5 5 2
A 2 C 3 E
10 10 9 2
12 S G 0 F 1
8 16 3 1
B 8 D 1 H
5 2 1

Problem



0
S
(20)
12
19
8
B
(24)
5
19
16
D
2
19
17
H
1
∞
19
G
0
19



5
A
5
C
2
E
2
3
10
10
9
2
12
S
G
0
F
1
8
16
3
1
8
1
B
D
H
5
2
1

Exercises: Artificial Intelligence

Monotonicity 1

Problem

- Prove that:
  - **IF** a heuristic function *h* satisfies the *monotonicity restriction*
    - $h(x) \leq cost(x \ldots y) + h(y)$
  - **THEN** *f* is *monotonously non-decreasing*
    - $f(s \ldots x) \leq f(s \ldots x \ldots y)$

Monotonicity 1

- *Given:*
  - ***<u>h</u>*** satisfies the ***<u>monotonicity restriction</u>***
- *Proof:*

f(S…A) = cost(S…A) + **h(A)**

≤ cost(S…A) + **cost(A…B) + h(B)**

≤ cost(S…A…B) + h(B)

≤ f(S…A…B)

Exercises: Artificial Intelligence

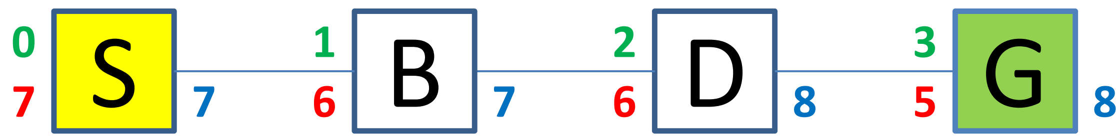
Monotonicity 2

Problem

- Prove or refute:
  - **IF** *f* is *monotonously non-decreasing*
    - f(s…x) ≤ f(s…xy)
  - **THEN** *h* is an *admissable heuristic*
    - *h* is an underestimate of the remaining path to the goal with the smallest cost
- Can an extra constraint on *h* change this?

Monotonicity 2

- *Given:*
  - *f* is *mononously non-decreasing*
- *Proof (Counter-example):*



f is monotonously non-decreasing, yet h is not an admissable heuristic.

Monotonicity 2

- *Given:*
  - ***f*** is ***mononously non-decreasing***
  - Extra constraint: ***h(G) = 0***
- *Proof:*

f(S…A) ≤ f(S…AB) ≤ … ≤ f(S…AB…G) ⟺

f(S…A) ≤ f(S…G) ⟺

cost(S…A) + h(A) ≤ cost(S…G) + h(G) ⟺

cost(S…A) + h(A) ≤ **cost(S…A)** + cost(A…G) + h(G) ⟺

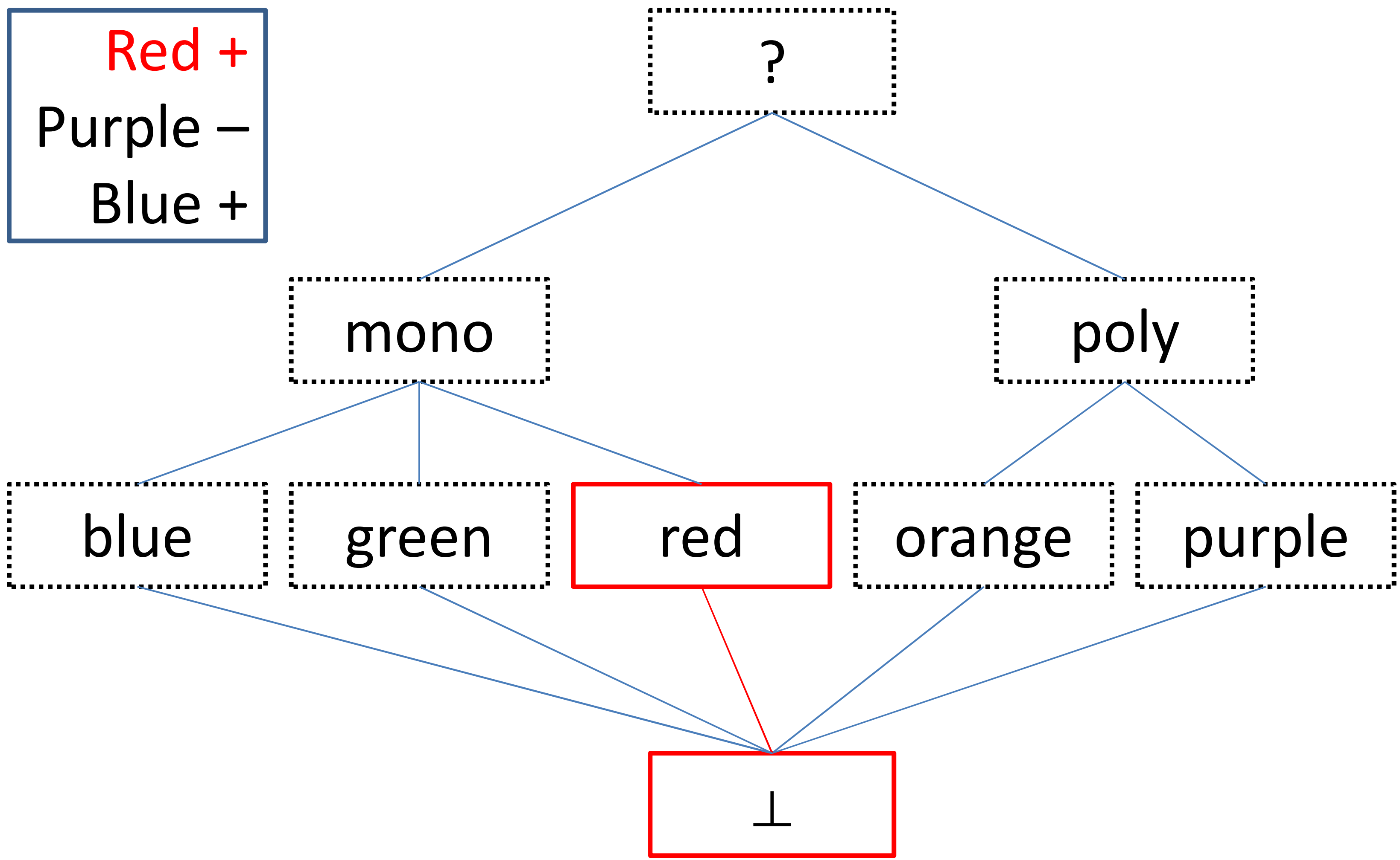
h(A) ≤ cost(A…G) + **h(G)** ⟺

h(A) ≤ cost(A…G)

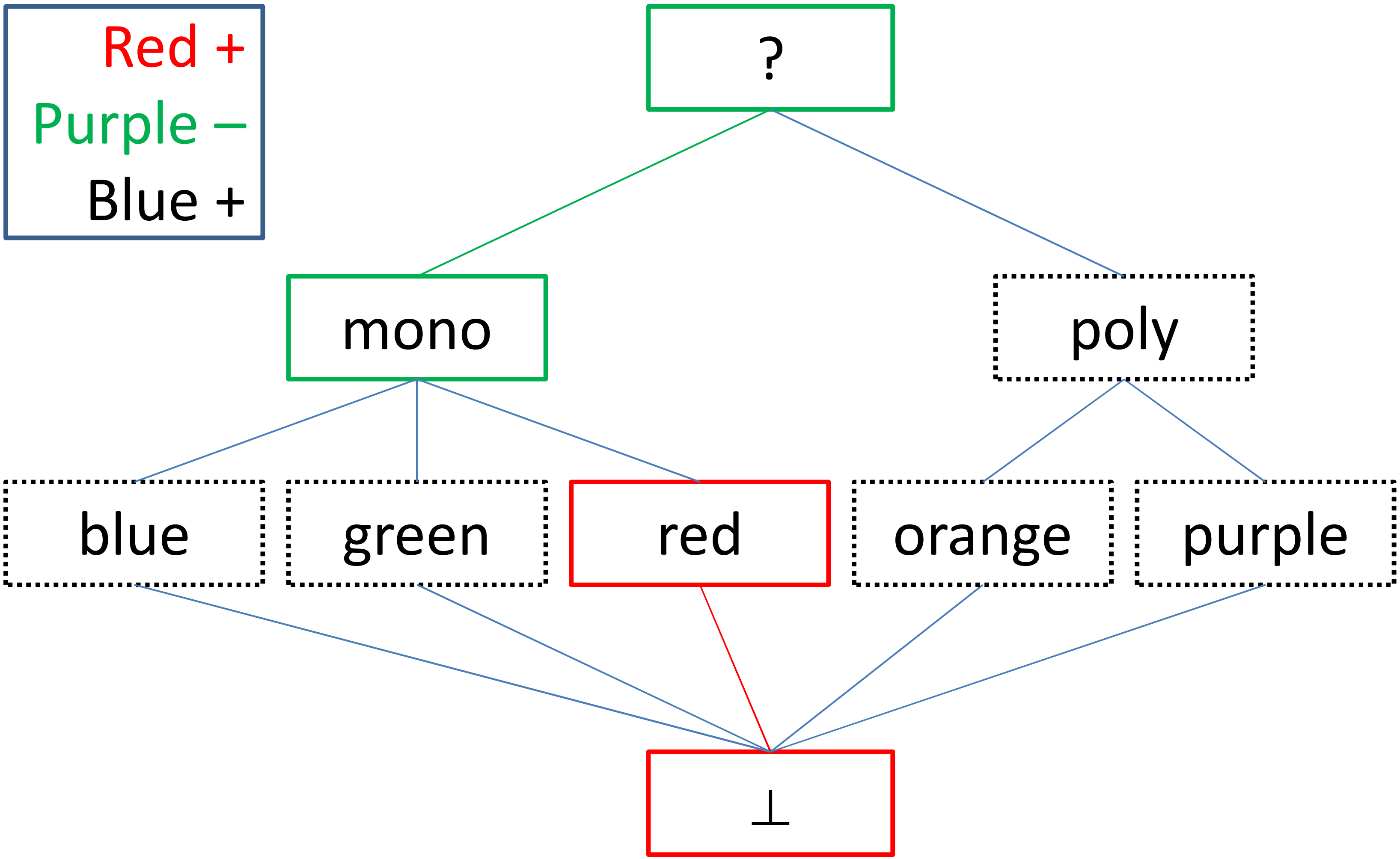
Exercises: Artificial Intelligence

Version Spaces: Colors

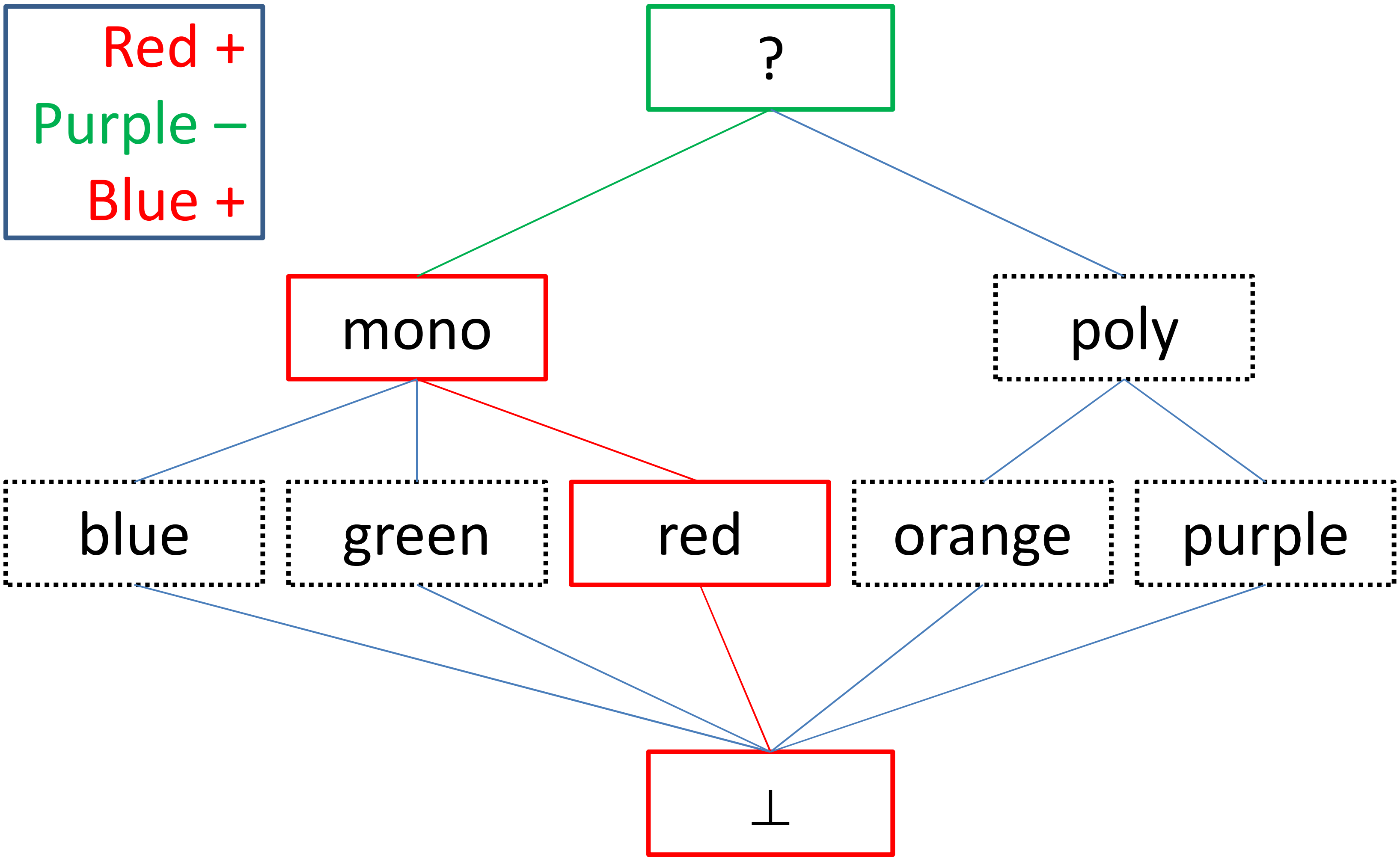
Version-Spaces Algorithm



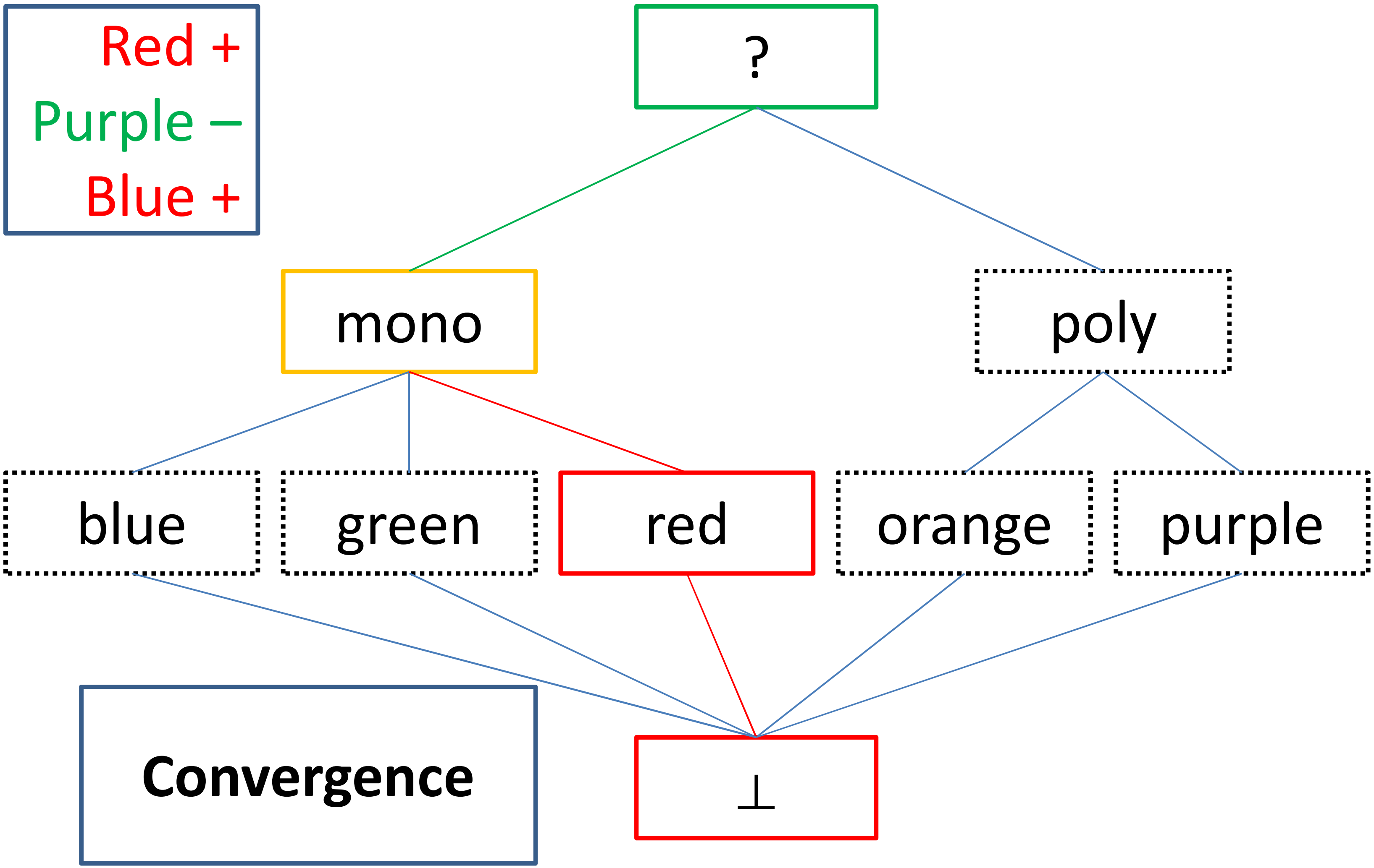
Version-Spaces Algorithm



Version-Spaces Algorithm



Version-Spaces Algorithm



Exercises: Artificial Intelligence

Version Spaces: Playing Cards

Version-Spaces Algorithm

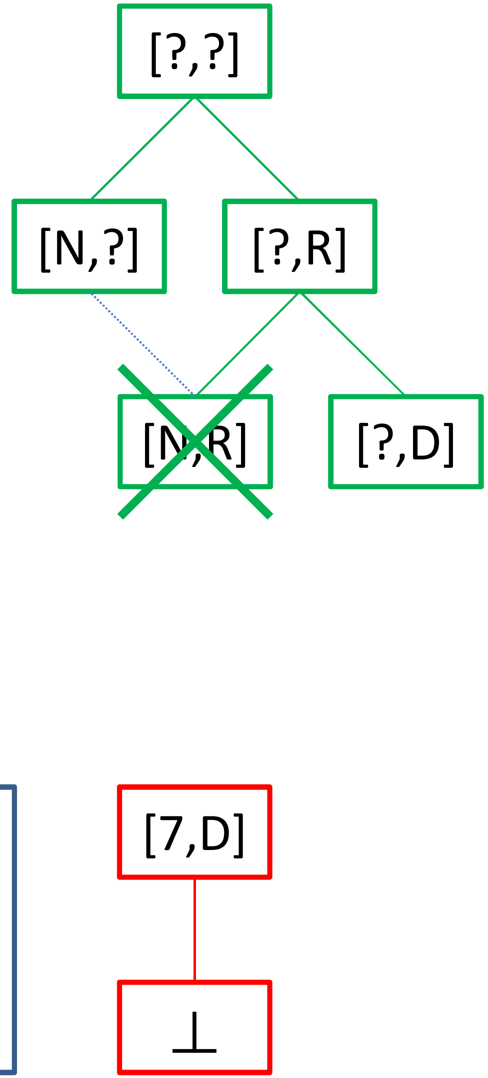
[7,D] +
[A,C] –
[Q,H] –
[9,H] +
[8,C] –

Version-Spaces Algorithm

[7,D] +
[A,C] –
[Q,H] –
[9,H] +
[8,C] –

Version-Spaces Algorithm

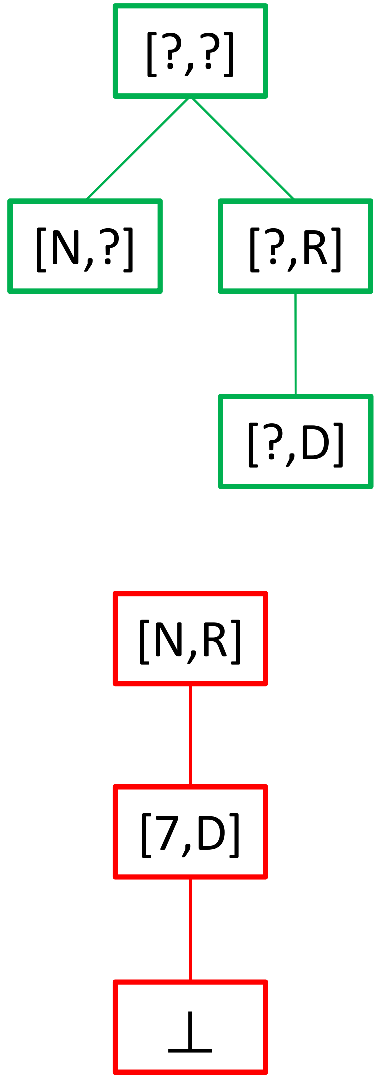
[7,D] +
[A,C] –
[Q,H] –
[9,H] +
[8,C] –



Redundant Hypotheses

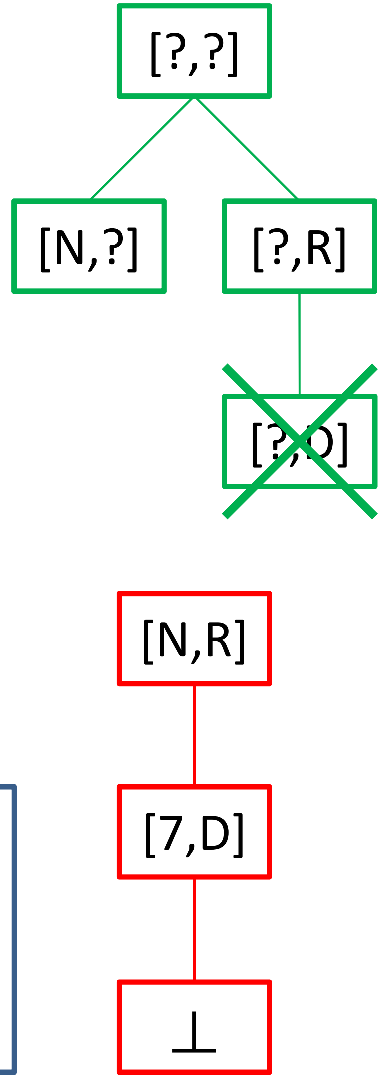
Version-Spaces Algorithm

[7,D] +
[A,C] –
[Q,H] –
[9,H] +
[8,C] –



Version-Spaces Algorithm

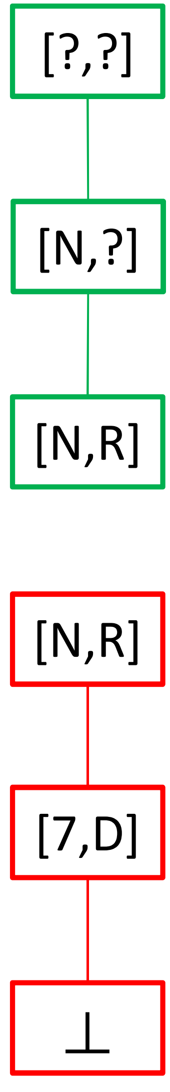
[7,D] +
[A,C] –
[Q,H] –
[9,H] +
[8,C] –



does not cover last positive example

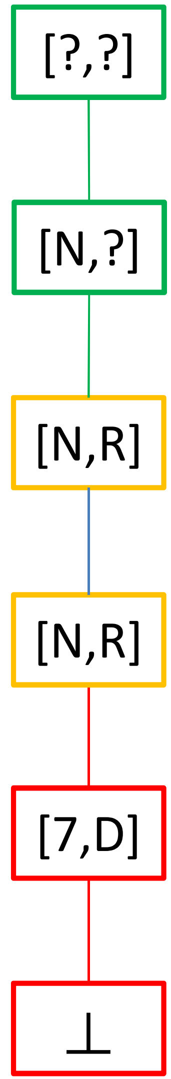
Version-Spaces Algorithm

[7,D] +
[A,C] –
[Q,H] –
[9,H] +
[8,C] –



Version-Spaces Algorithm

[7,D] +
[A,C] –
[Q,H] –
[9,H] +
[8,C] –



Convergence

Exercises: Artificial Intelligence

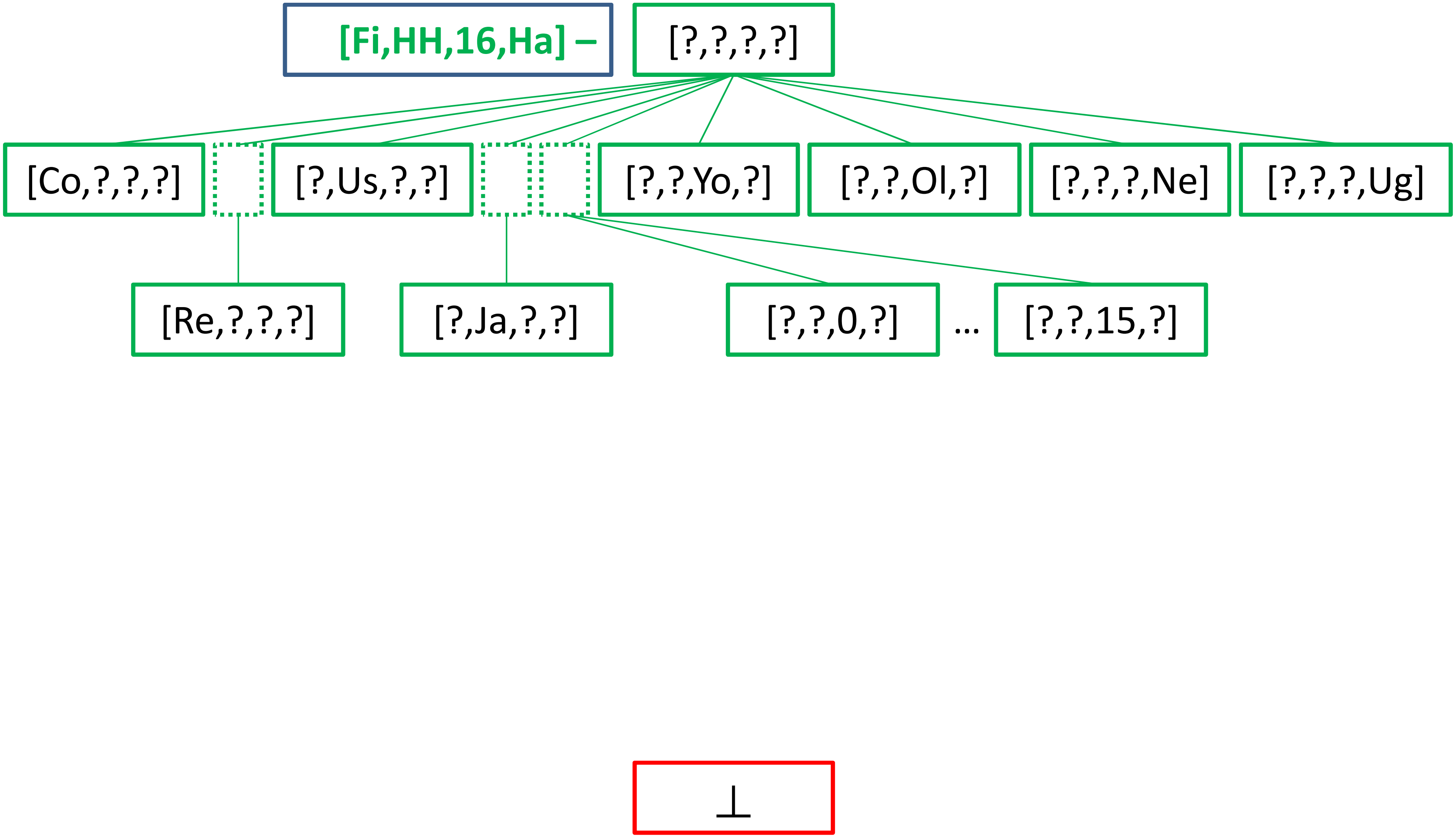
Version Spaces: Ex-exam

Version-Spaces Algorithm

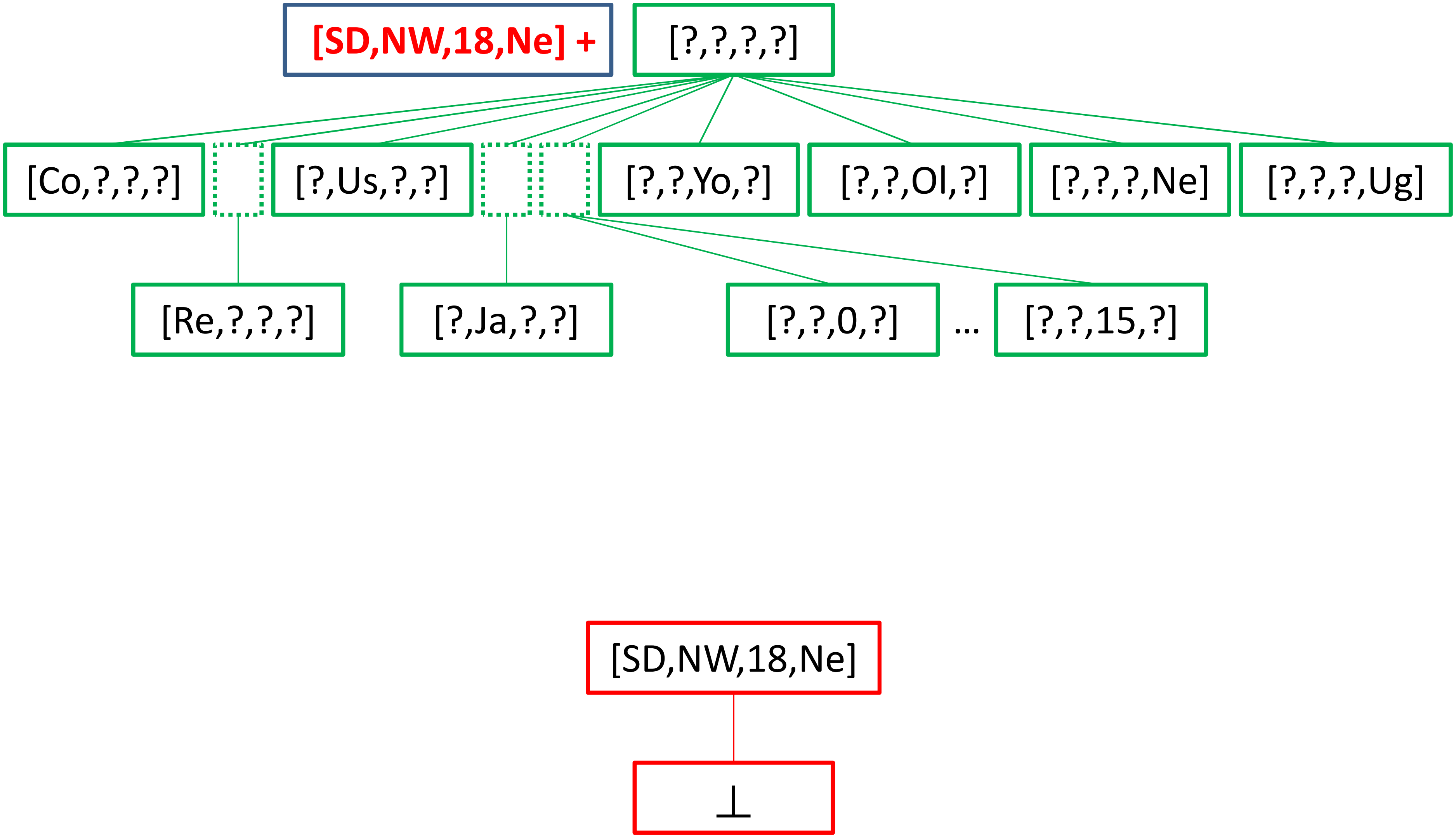
[?,?,?,?]

$\perp$

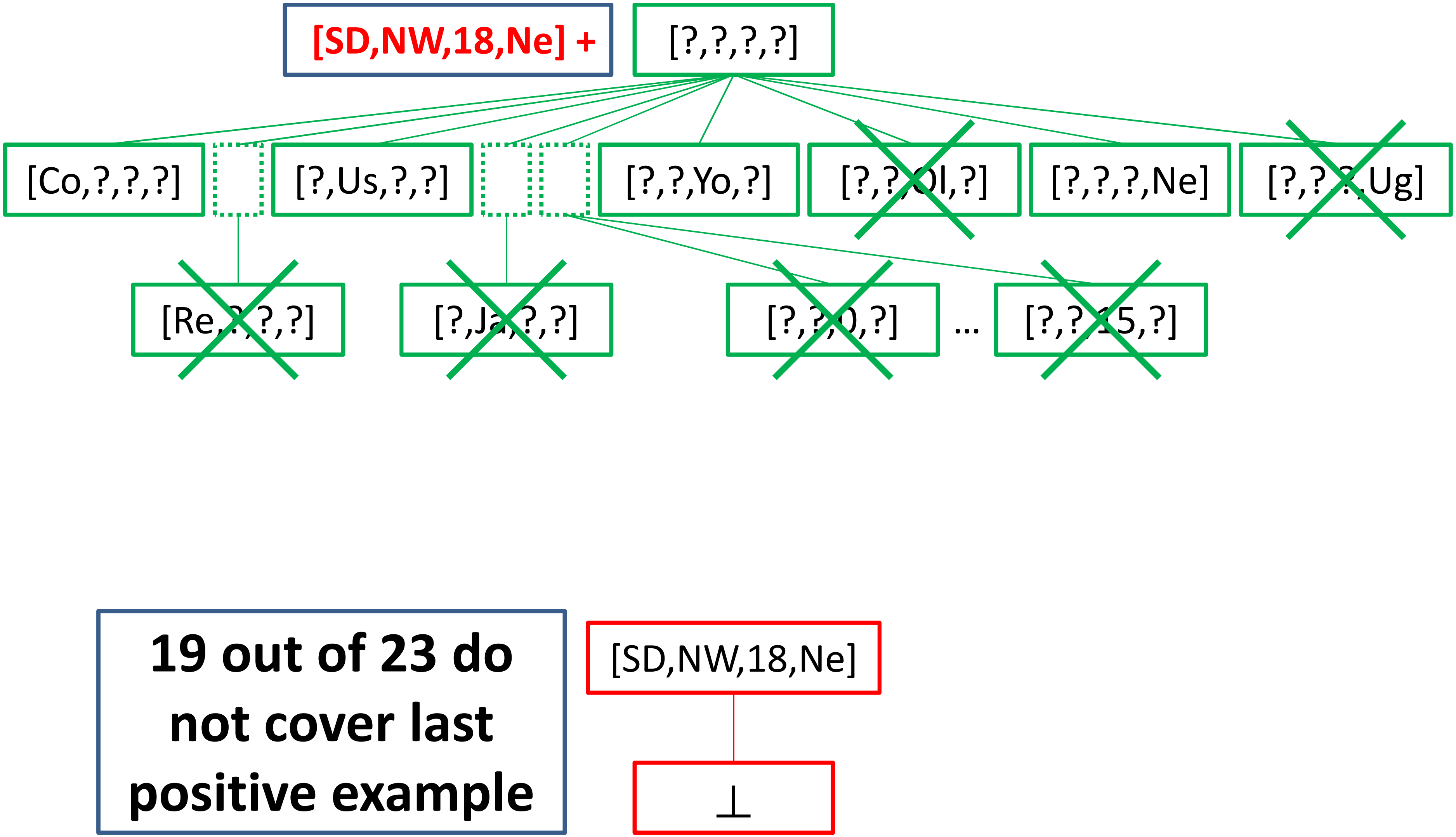
Version-Spaces Algorithm



Version-Spaces Algorithm

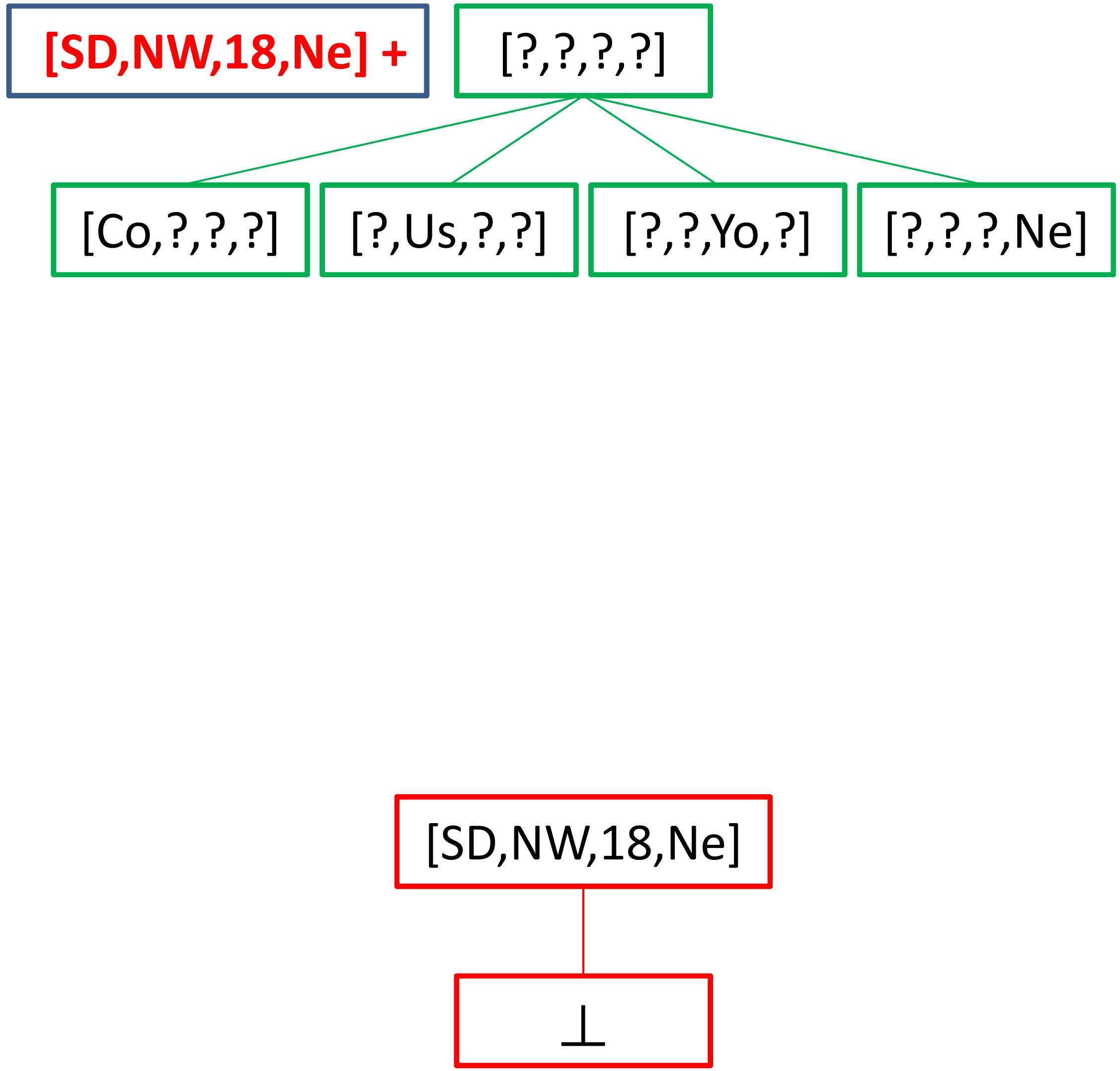


Version-Spaces Algorithm

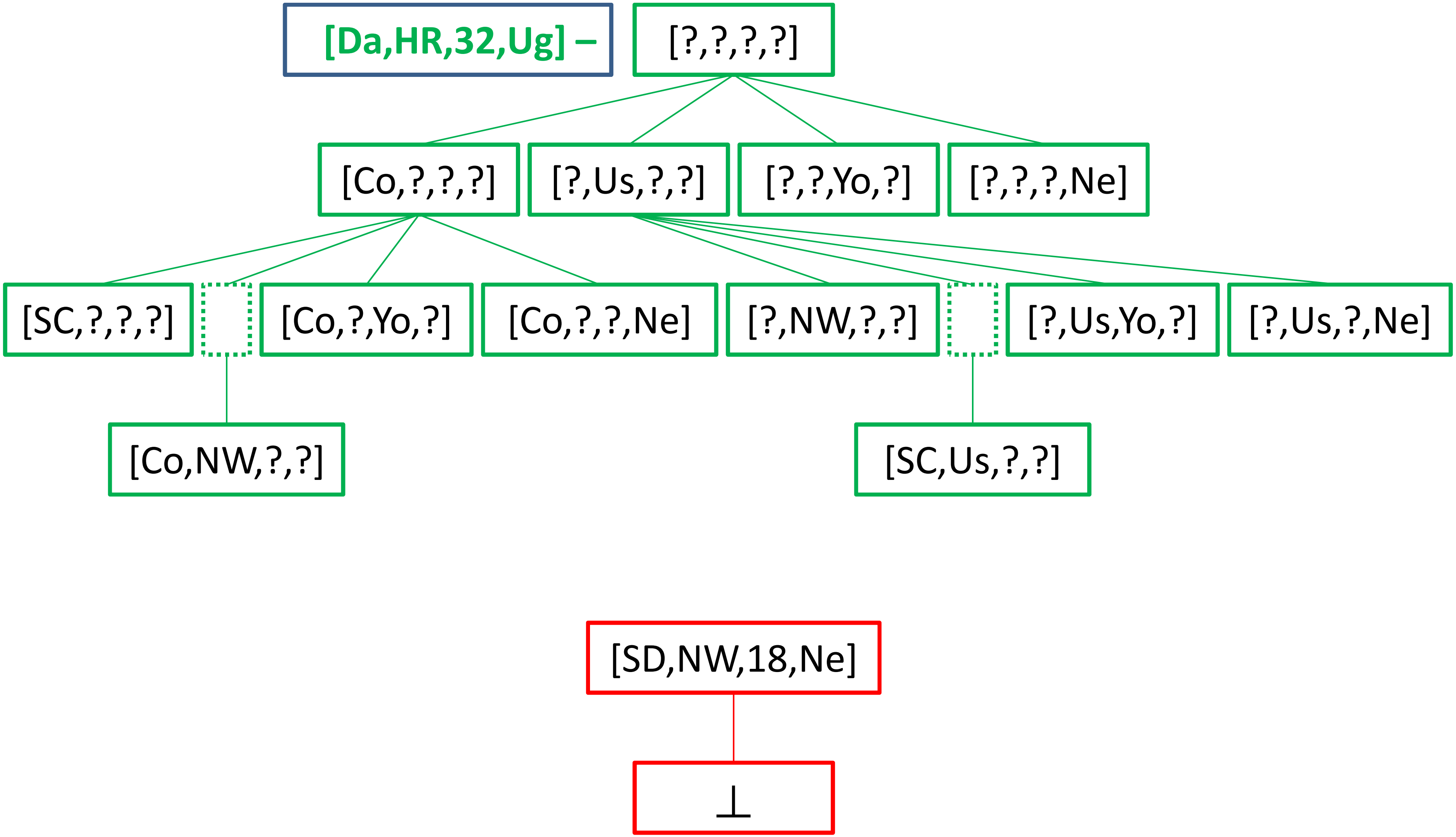


19 out of 23 do not cover last positive example

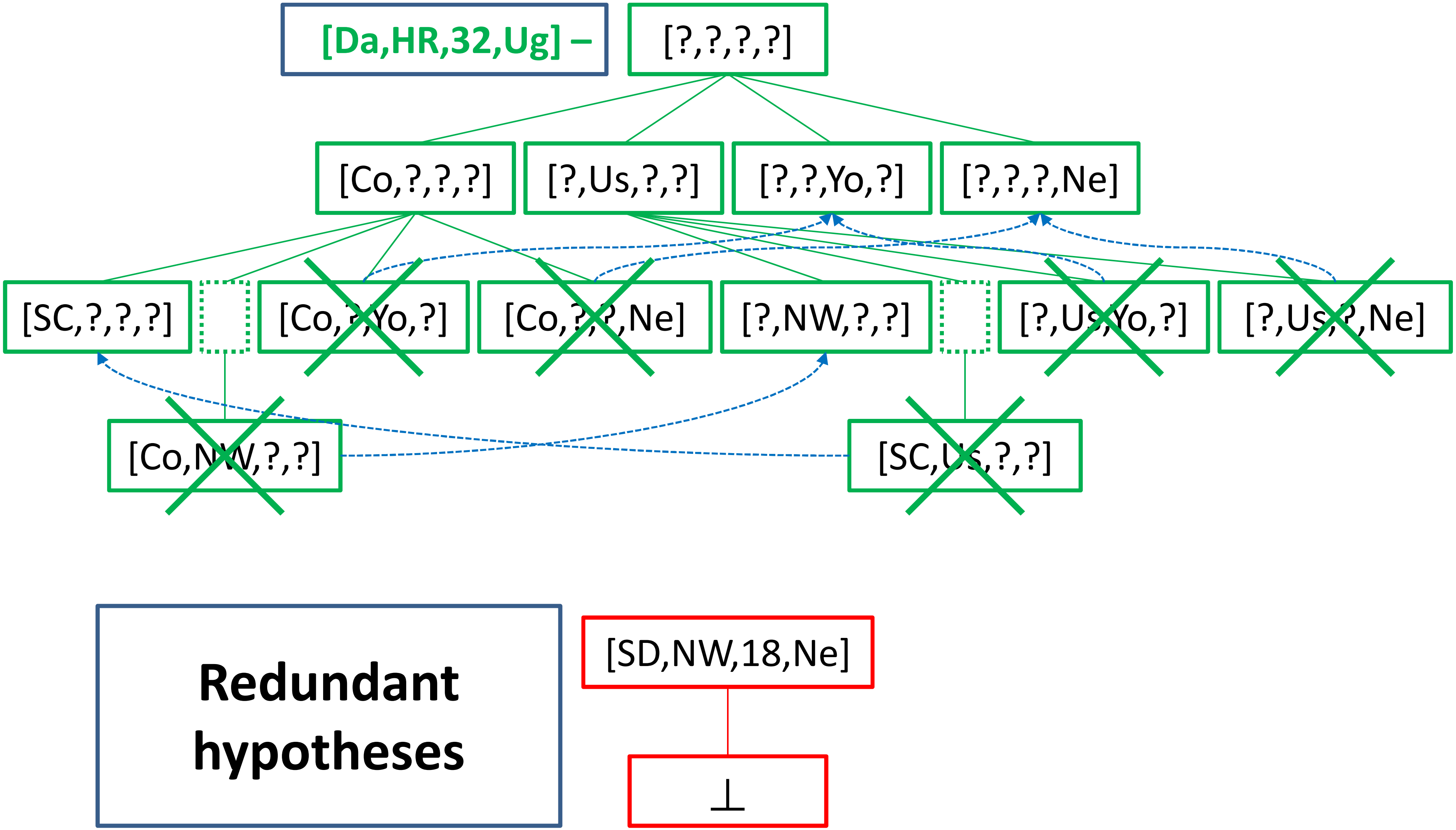
Version-Spaces Algorithm



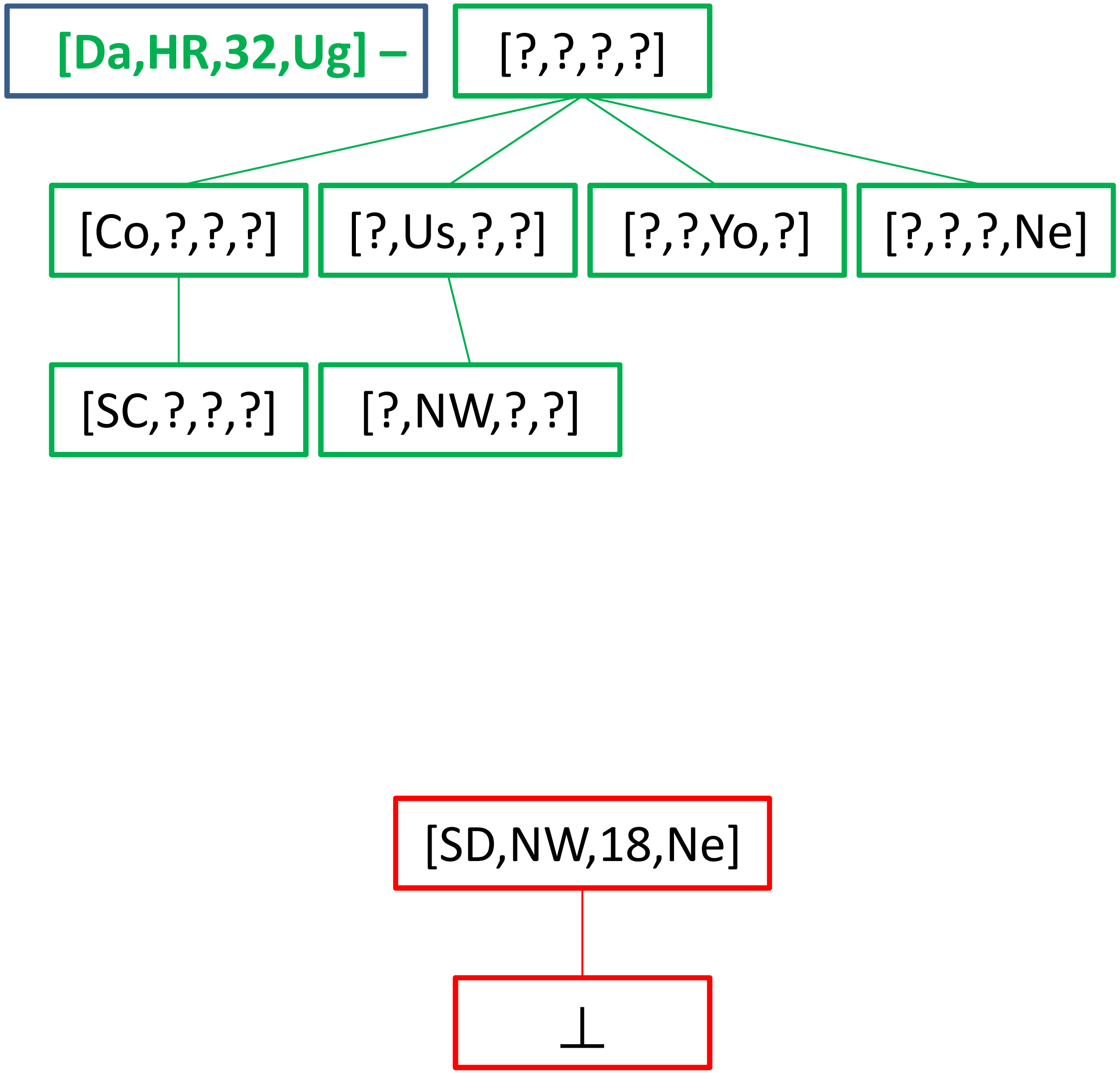
Version-Spaces Algorithm



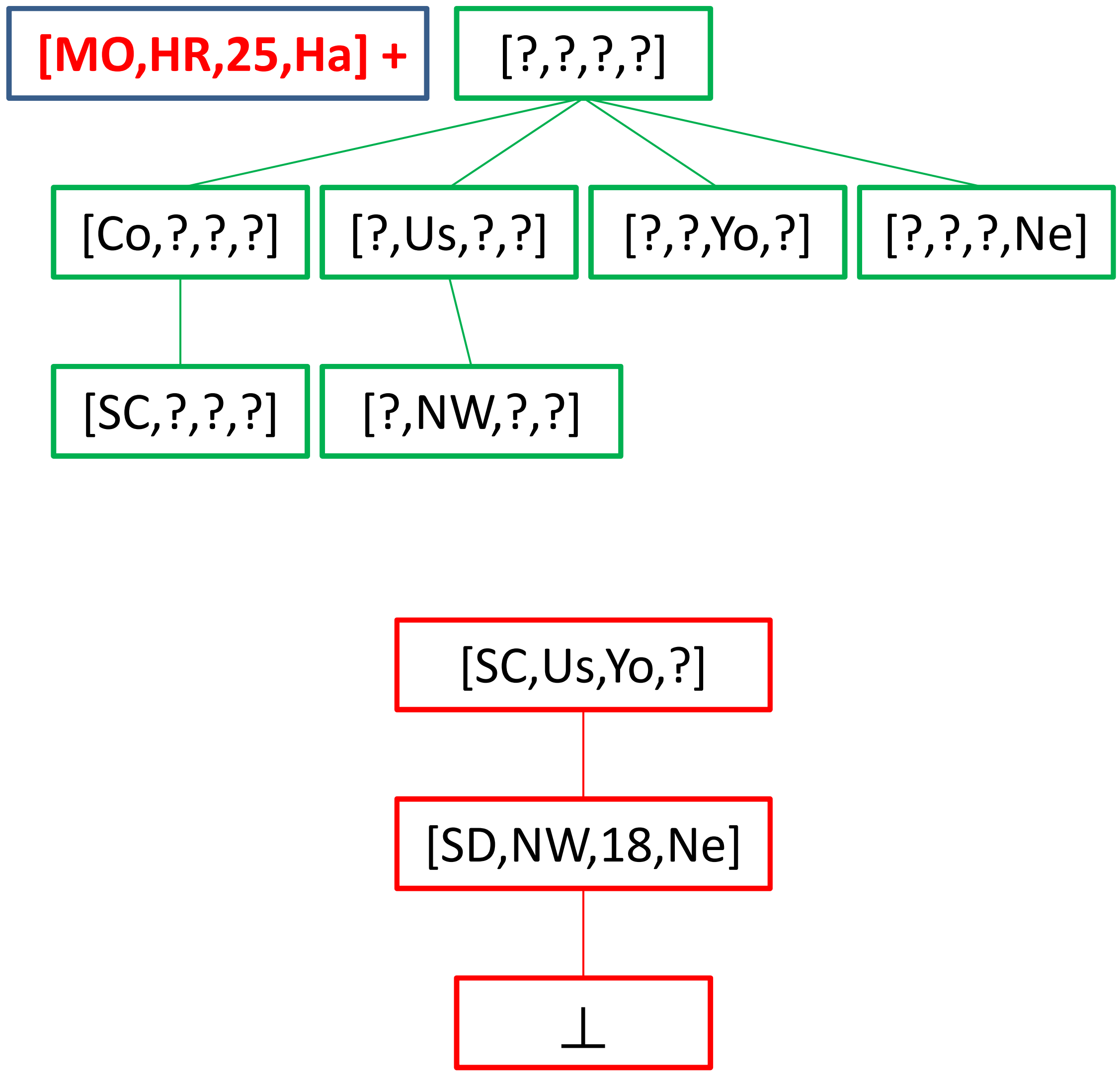
Version-Spaces Algorithm



Version-Spaces Algorithm

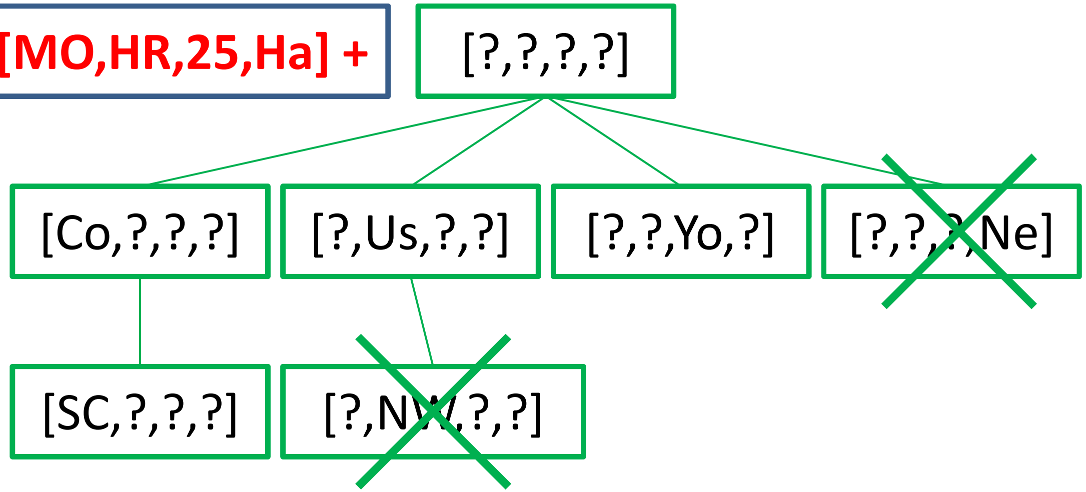


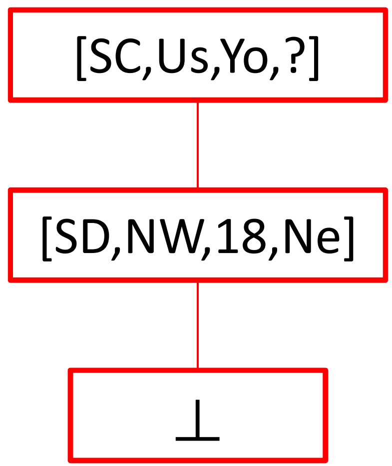
Version-Spaces Algorithm



Version-Spaces Algorithm

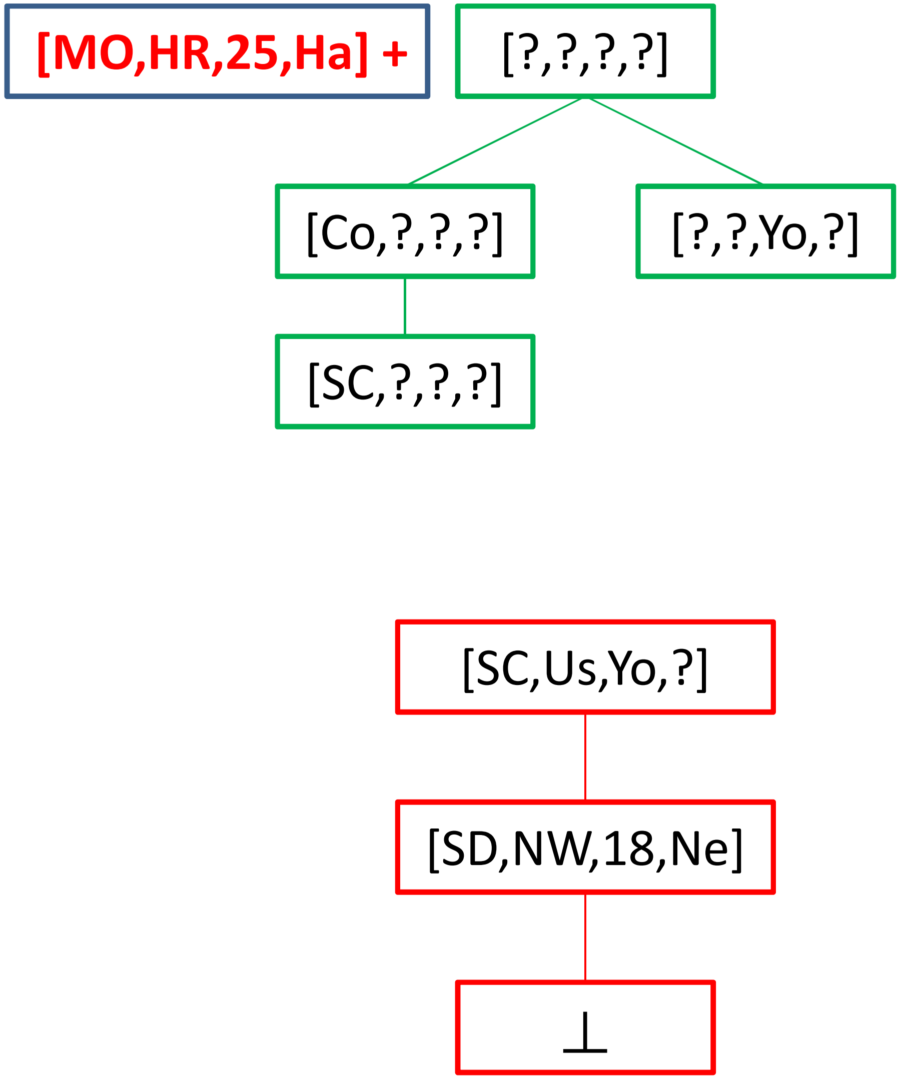
[MO,HR,25,Ha] +



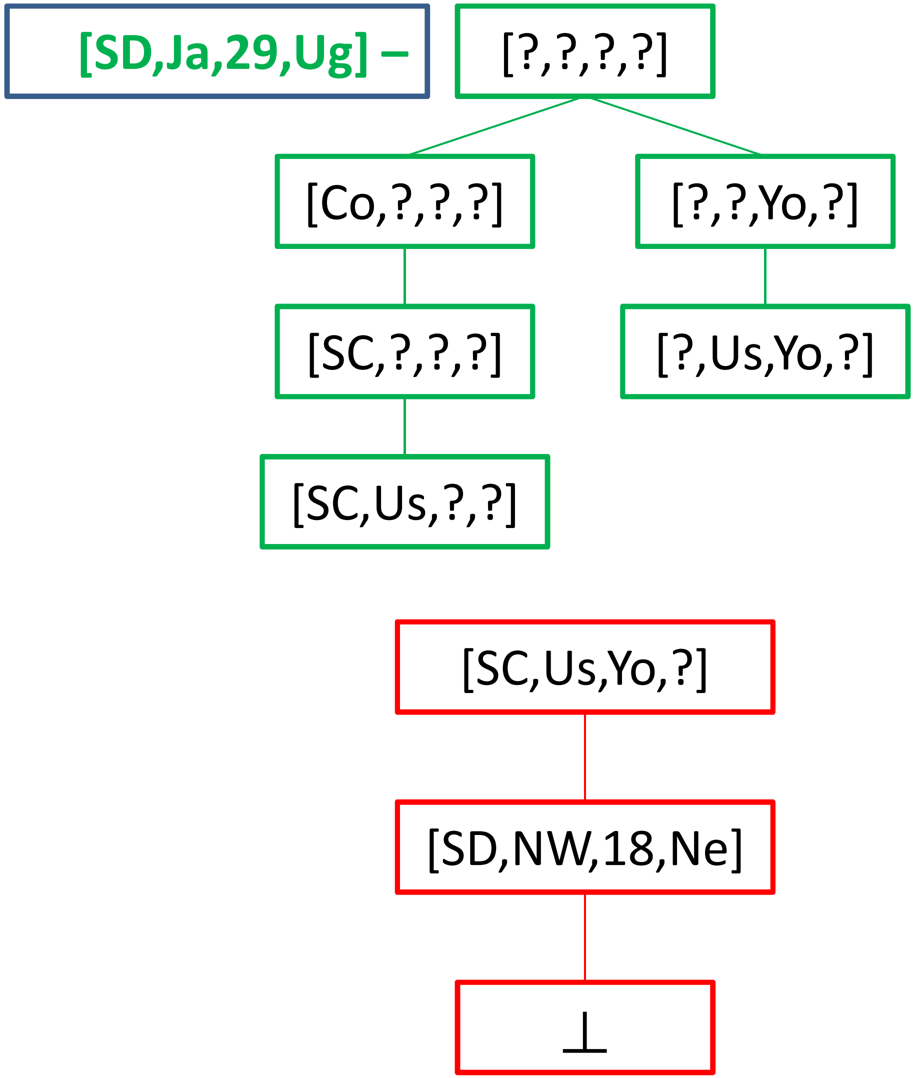


2 out of 4 do not cover last positive example

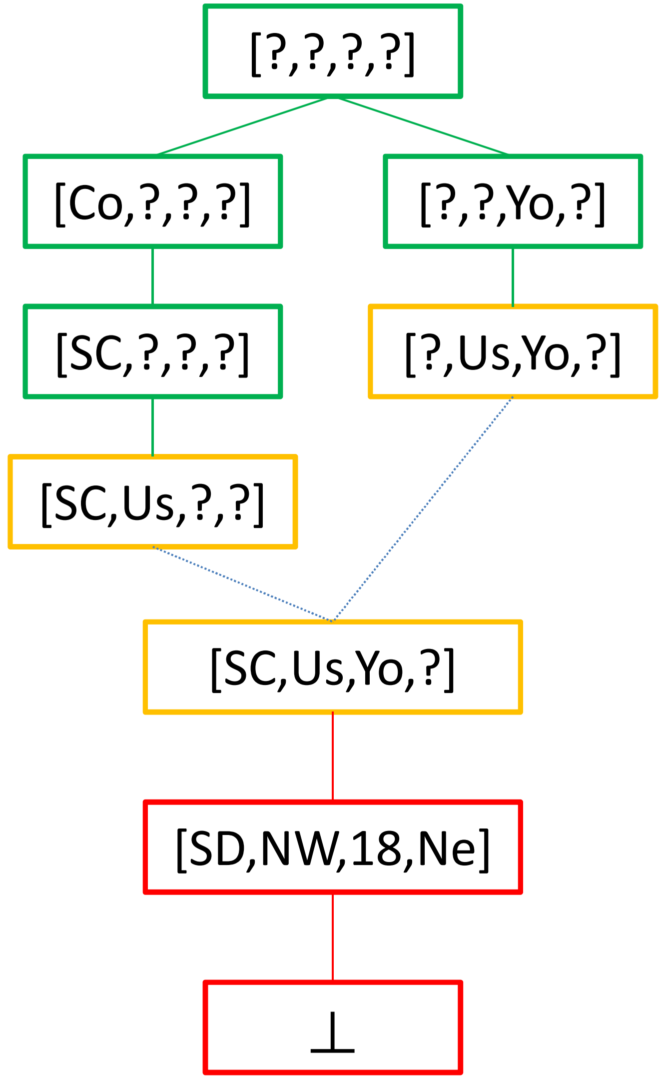
Version-Spaces Algorithm



Version-Spaces Algorithm

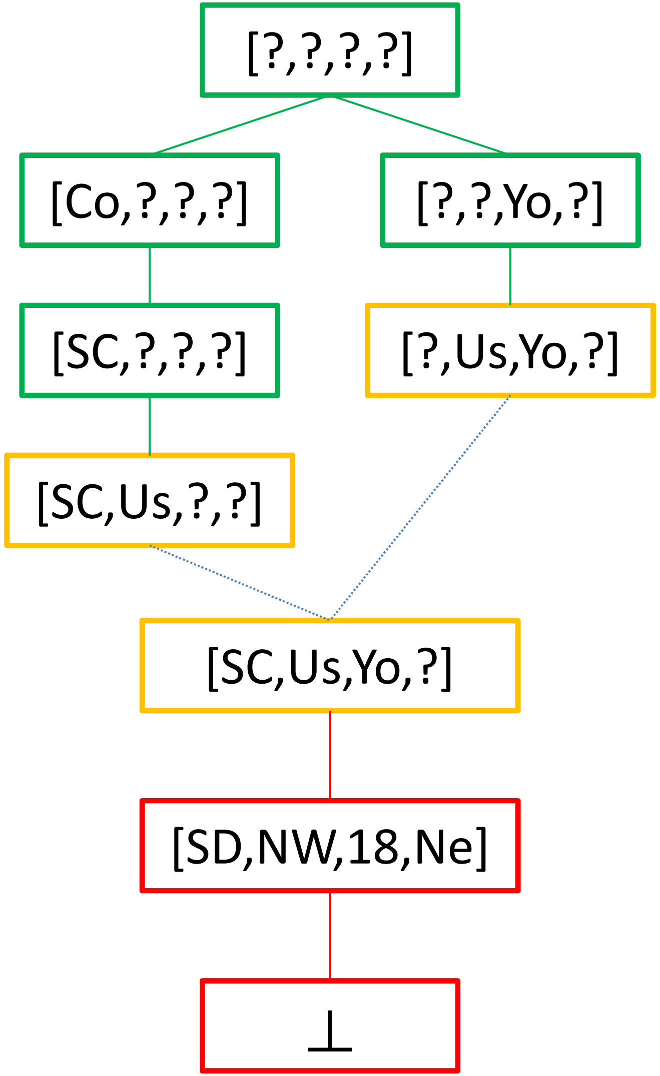


Version-Spaces Algorithm



Using the result

- [MO,HR,32,Ha]: **Maybe**
  - More Specific than [SC,Us,?,?]
  - Not more Specific than [SC,Us,Yo,?]
- [SD,HH,18,Ne]: **NO**
  - Not More Specific than [SC,Us,?,?]
  - Not More Specific than [?,Us,Yo,?]
- [Da,NW,22,Ug]: **Maybe**
  - More Specific than [?,Us,Yo,?]
  - Not more Specific than [SC,Us,Yo,?]



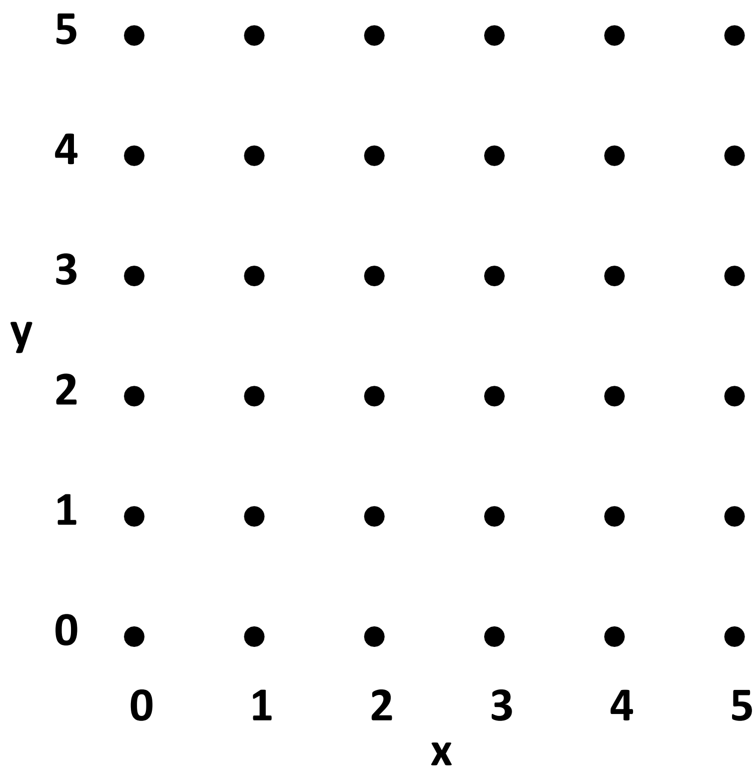
Exercises: Artificial Intelligence

Version Spaces: Computer Screen

Version-Spaces Algorithm

G = {[((0,0),5),white]}

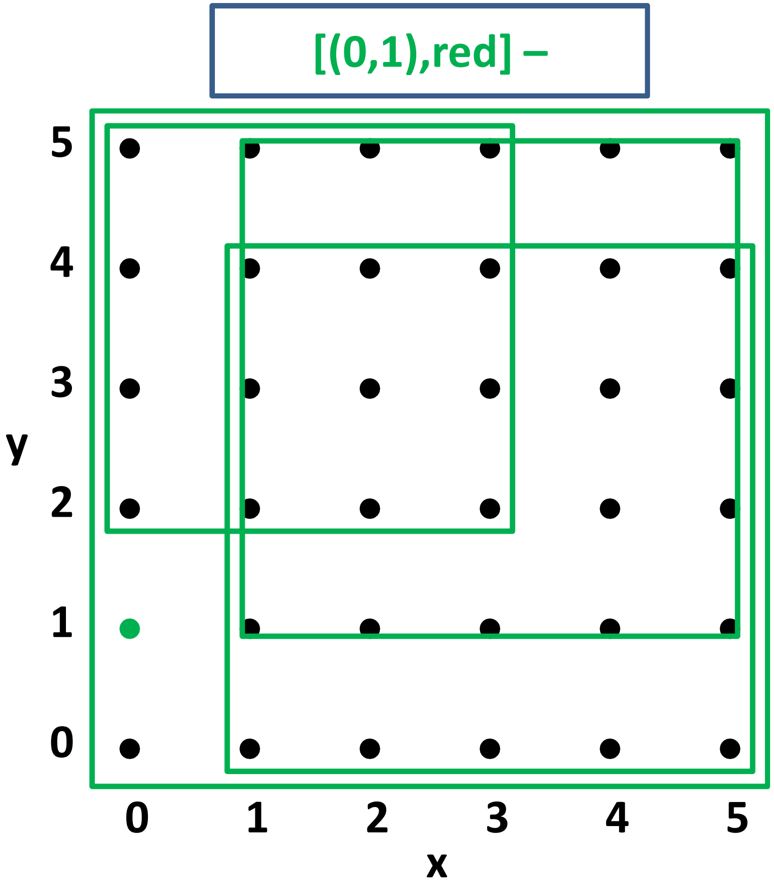
S = {⊥}



Version-Spaces Algorithm

G = {[((0,0),5),white]}

S = {⊥}



G = {

[((0,2),3),white],
[((1,0),4),white],
[((1,1),4),white],
[((0,0),5),cyan]

}

Redundant:

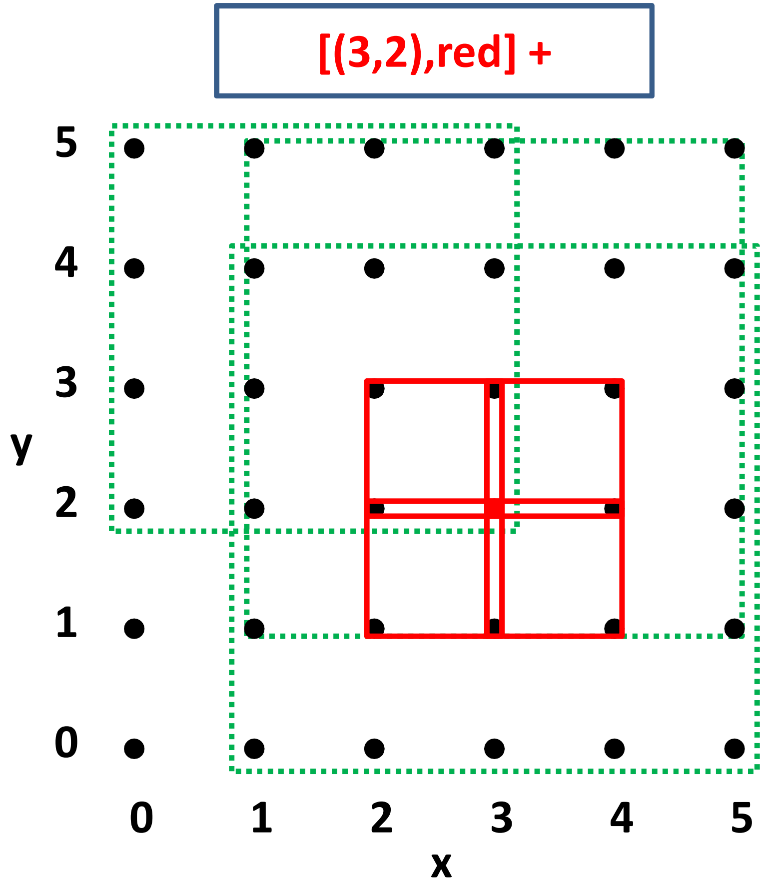
[((0,0),5),green]
[((0,0),5),blue]

S = {⊥}

Version-Spaces Algorithm

G = {[((0,2),3),white],[((1,0),4),white],[((1,1),4),white],[((0,0),5),cyan]}

S = {⊥}



G = {

[((0,2),3),white],
[((1,0),4),white],
[((1,1),4),white]

}

Removed:

[((0,0),5),cyan]

S = {

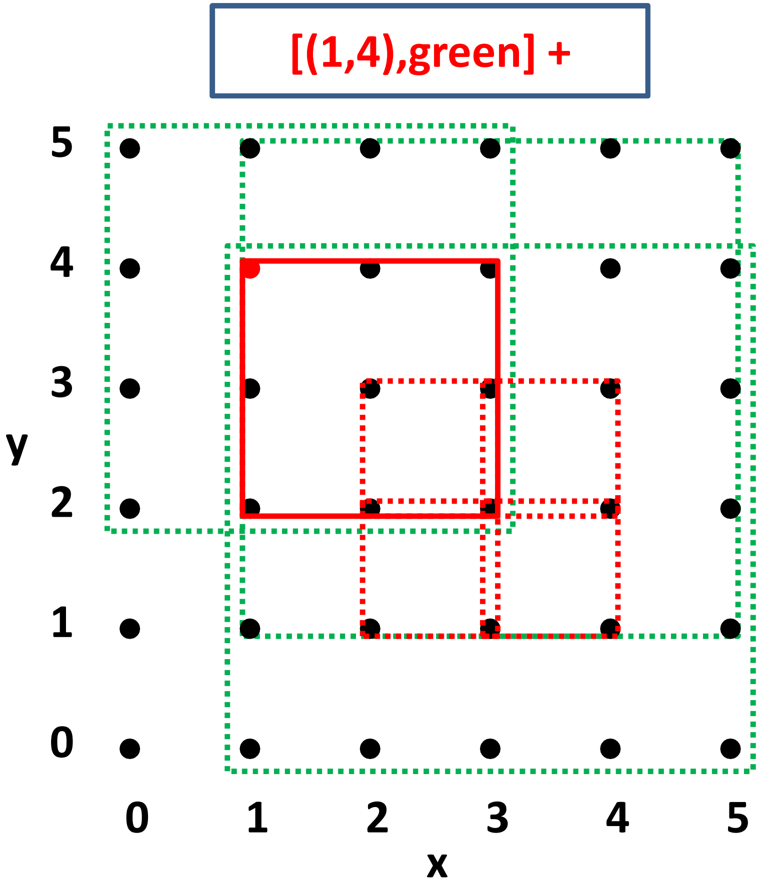
[((2,1),1),red],
[((2,2),1),red],
[((3,1),1),red],
[((3,2),1),red]

}

Version-Spaces Algorithm

G = {[((0,2),3),white],[((1,0),4),white],[((1,1),4),white]}

S = {[((2,1),1),red],[((2,2),1),red],[((3,1),1),red],[((3,2),1),red]}

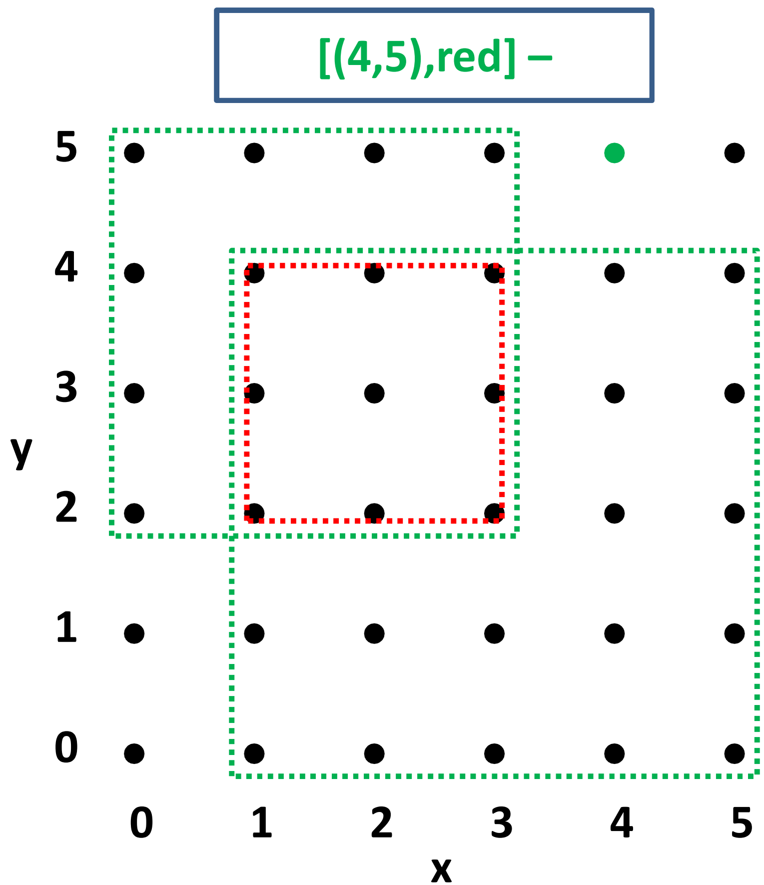


G = {
[((0,2),3),white],
[((1,0),4),white],
[((1,1),4),white]
}
S = {
[((1,2),2),yellow]
}
Redundant:
[((0,2),3),yellow]
[((1,2),3),yellow]
[((1,1),3),yellow]
[((1,1),4),yellow]
[((1,0),4),yellow]

Version-Spaces Algorithm

G = {[((0,2),3),white],[((1,0),4),white],[((1,1),4),white]}

S = {[((1,2),2),yellow]}



G = {
[((0,2),3),white],
[((1,0),4),white]
}

Redundant:
[((1,1),3),white]

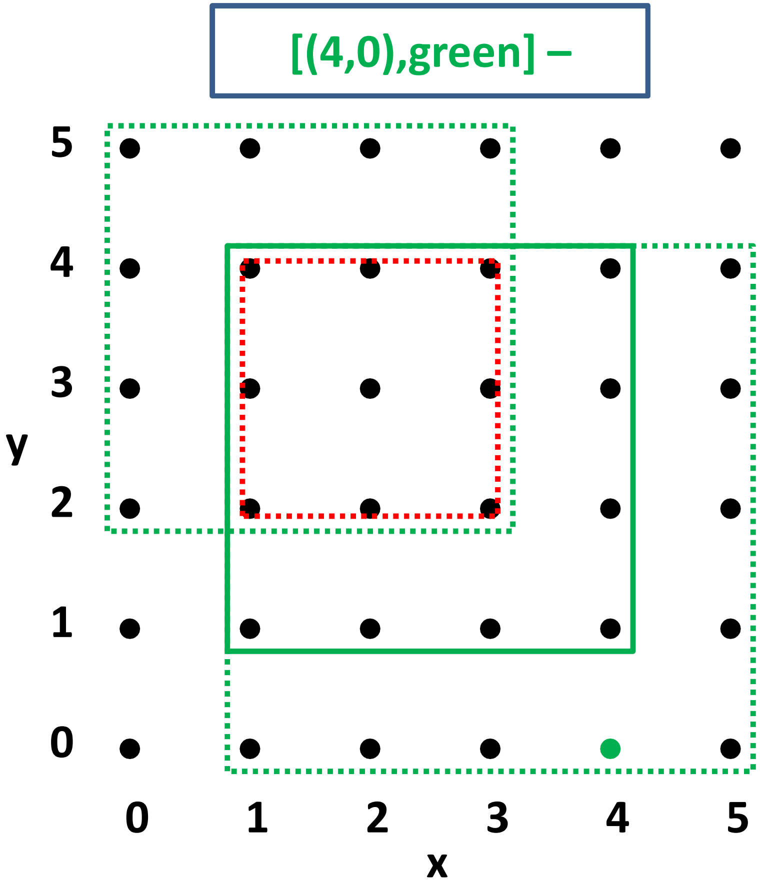
Others don't generalize S

S = {
[((1,2),2),yellow]
}

Version-Spaces Algorithm

G = {[((0,2),3),white],[((1,0),4),white]}

S = {[((1,2),2),yellow]}



G = {

[((0,2),3),white],

[((1,1),3),white]

}

Others don't generalize S

S = {

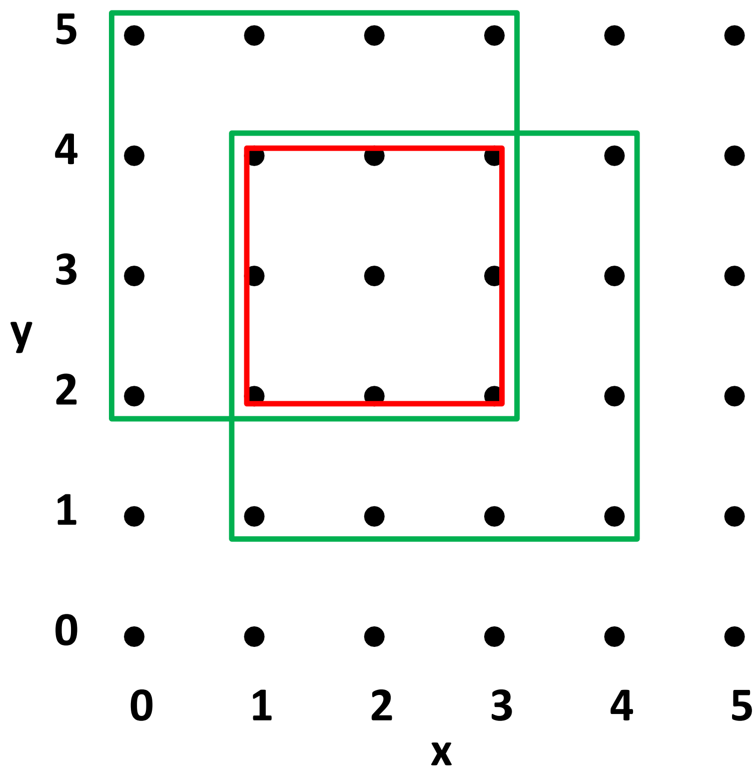
[((1,2),2),yellow]

}

Version-Spaces Algorithm

G = {[((0,2),3),white],[((1,1),3),white]}

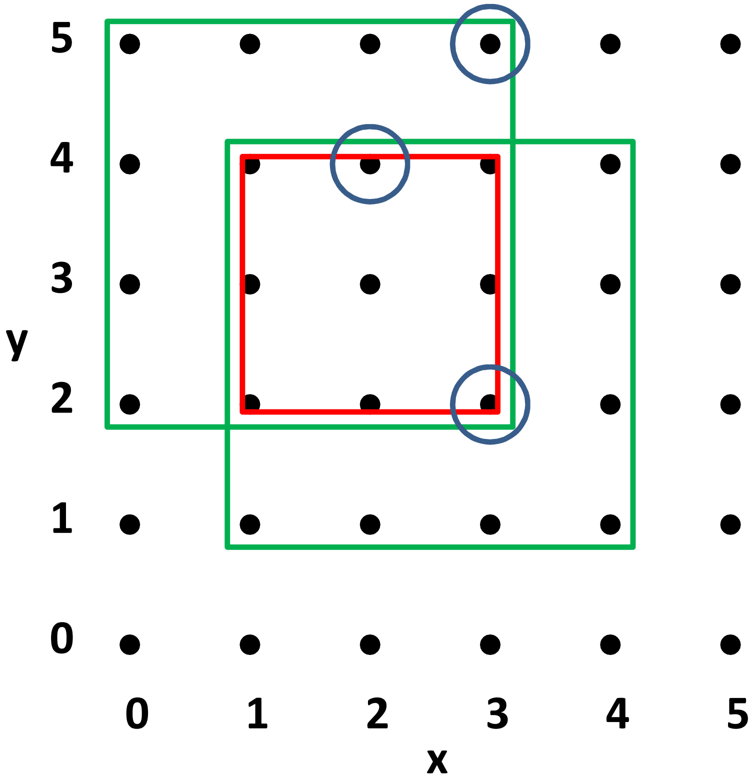
S = {[((1,2),2),yellow]}



Using the Result

G = {[((0,2),3),white],[((1,1),3),white]}

S = {[((1,2),2),yellow]}



[(3,2),green]

Yes

[(2,4),red]

Yes

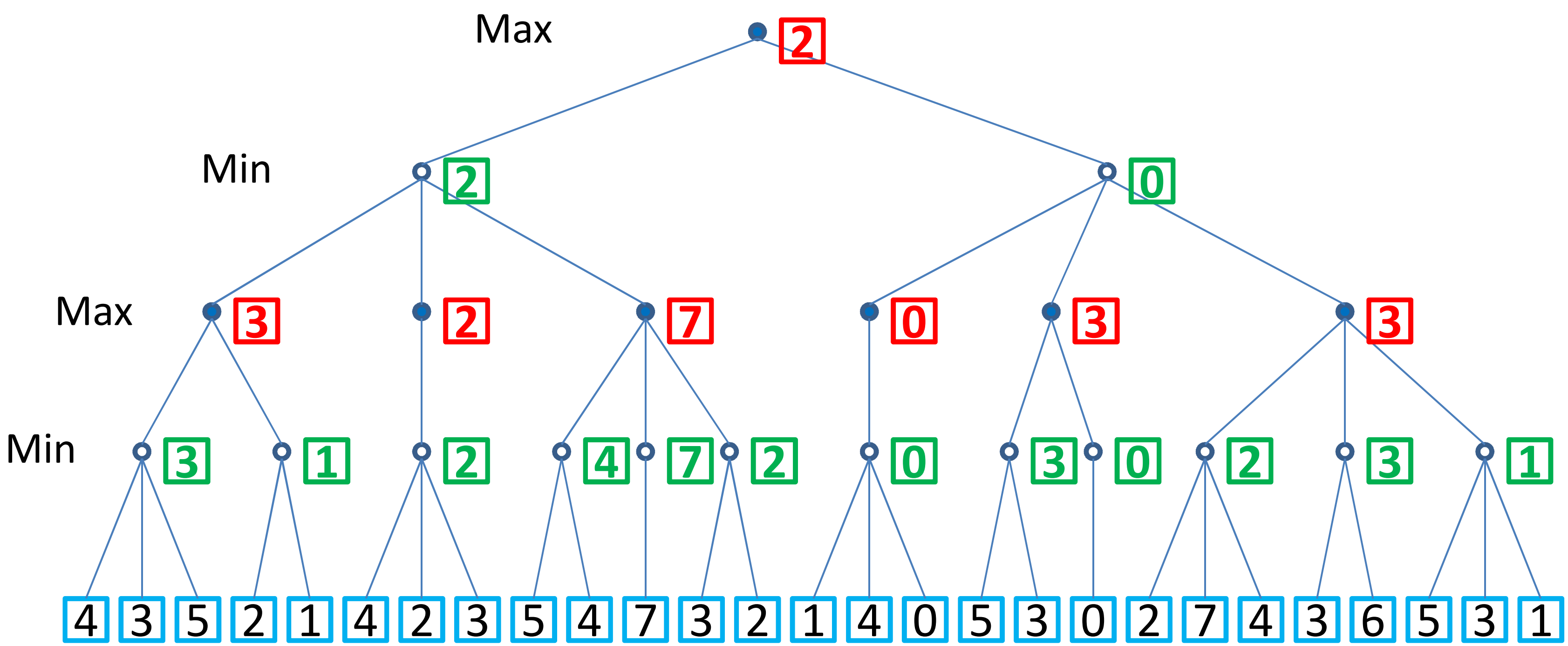
[(3,5),blue]

Maybe

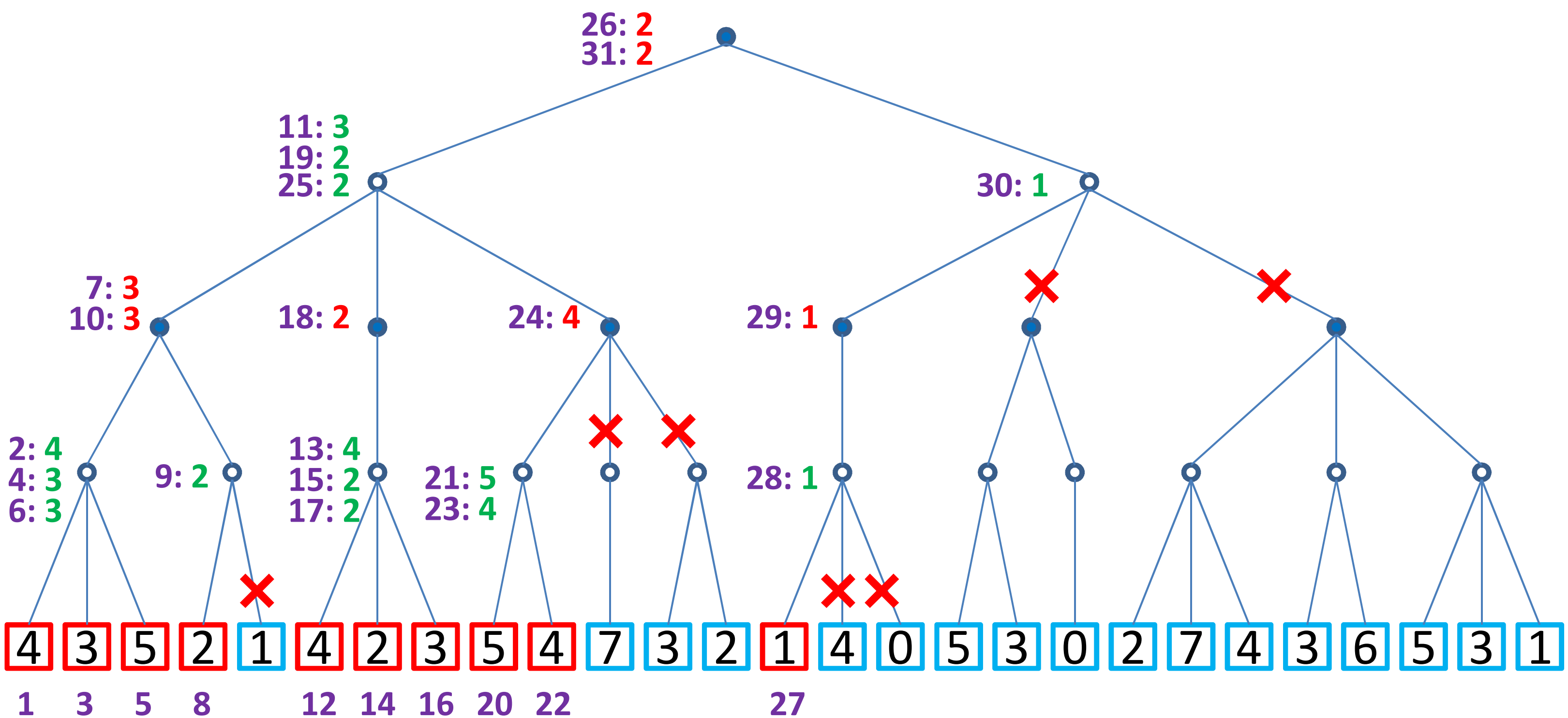
Exercises: Artificial Intelligence

MiniMax & Constraint Processing: MiniMax Algorithm

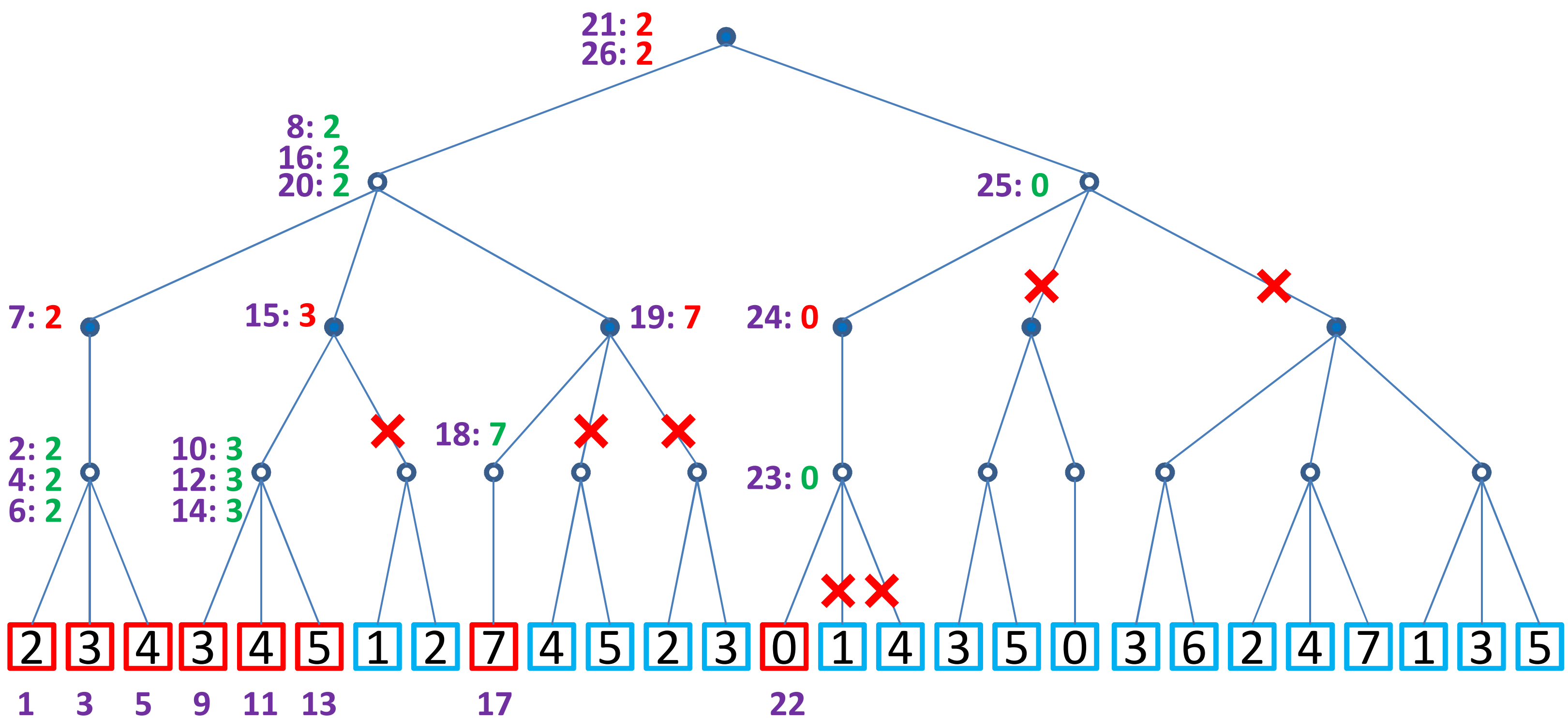
MiniMax without $\alpha\beta$-pruning



MiniMax with $\alpha\beta$-pruning



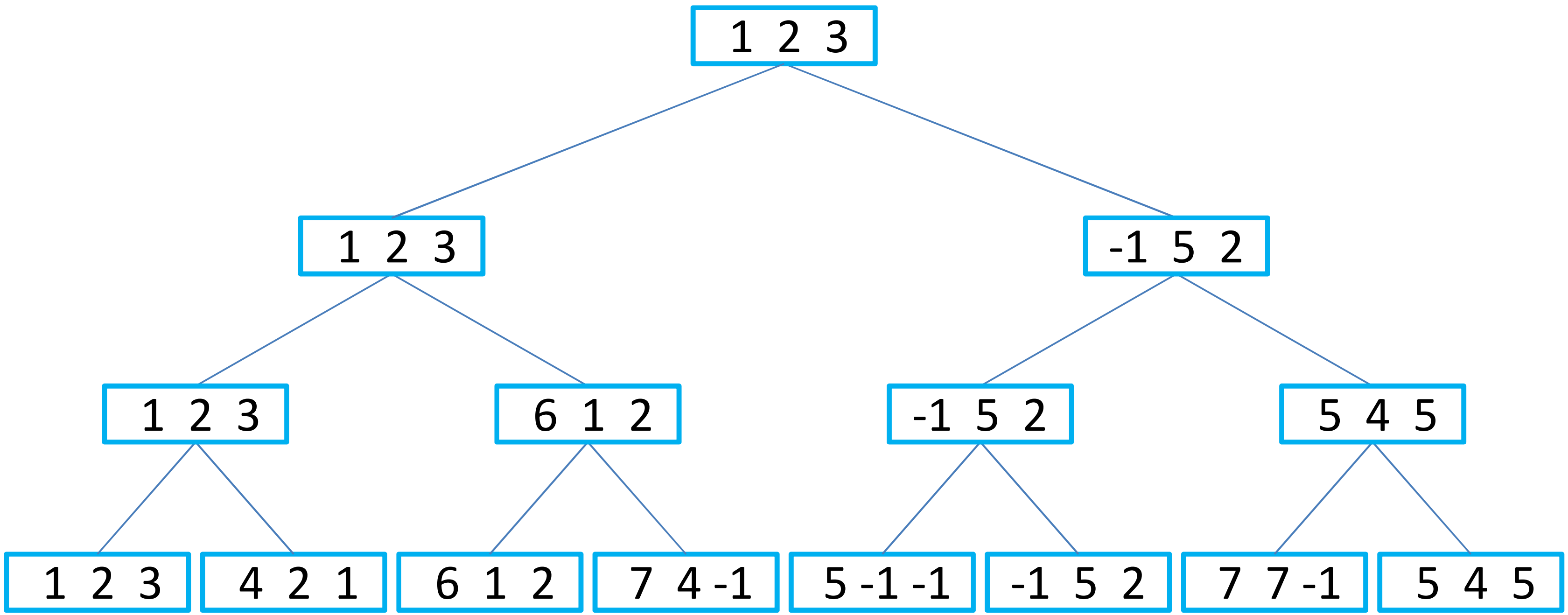
Reordering, MiniMax with $\alpha\beta$-Pruning



Exercises: Artificial Intelligence

MiniMax & Constraint Processing:
MiniMax Algorithm for 3 Players

MiniMax For 3 Players

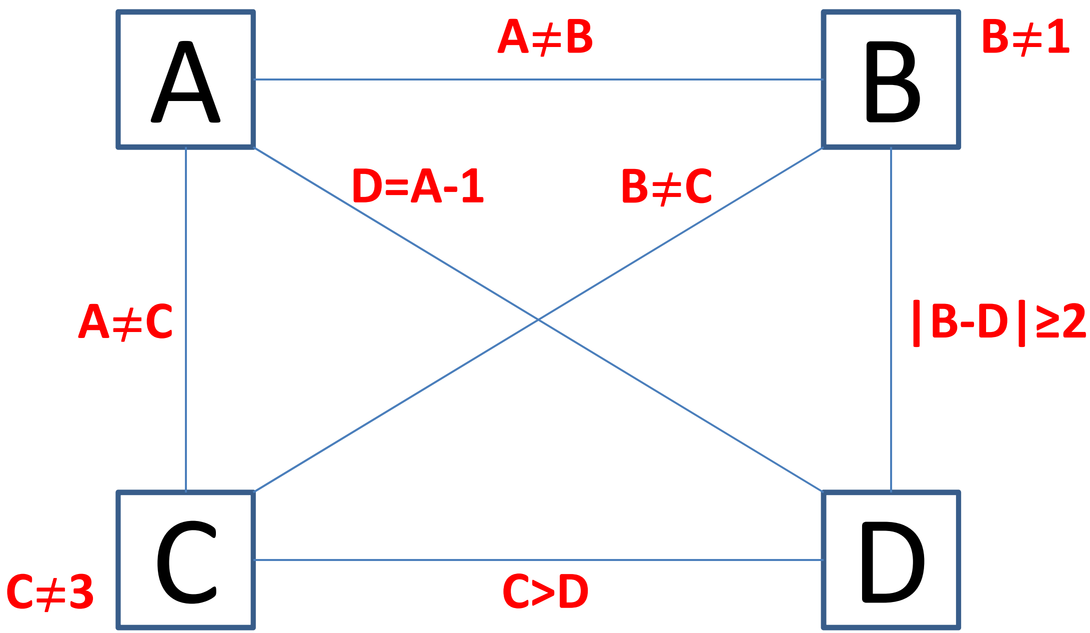


Exercises: Artificial Intelligence

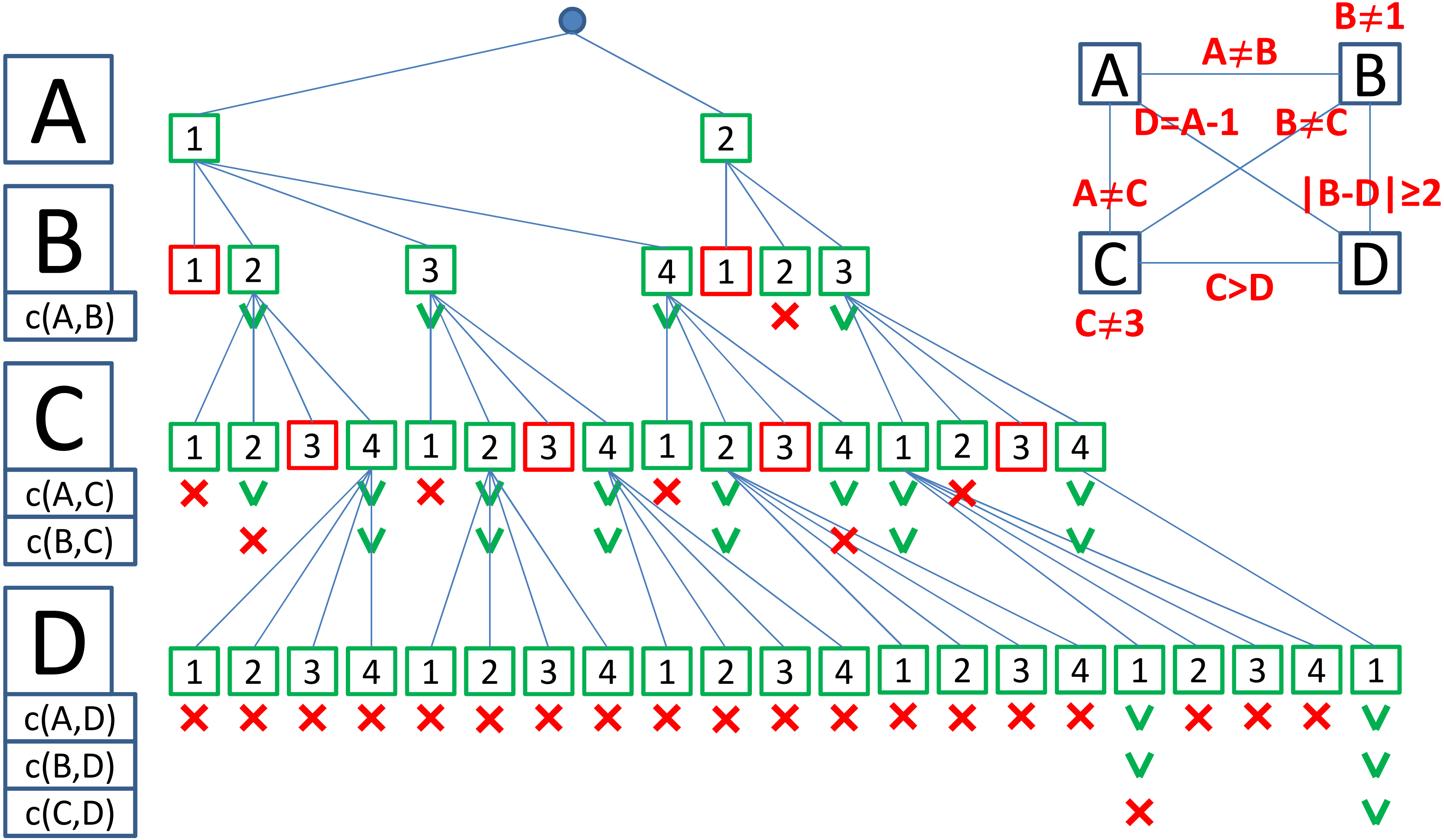
MiniMax & Constraint Processing: The 4 Houses problem

Constraint Processing

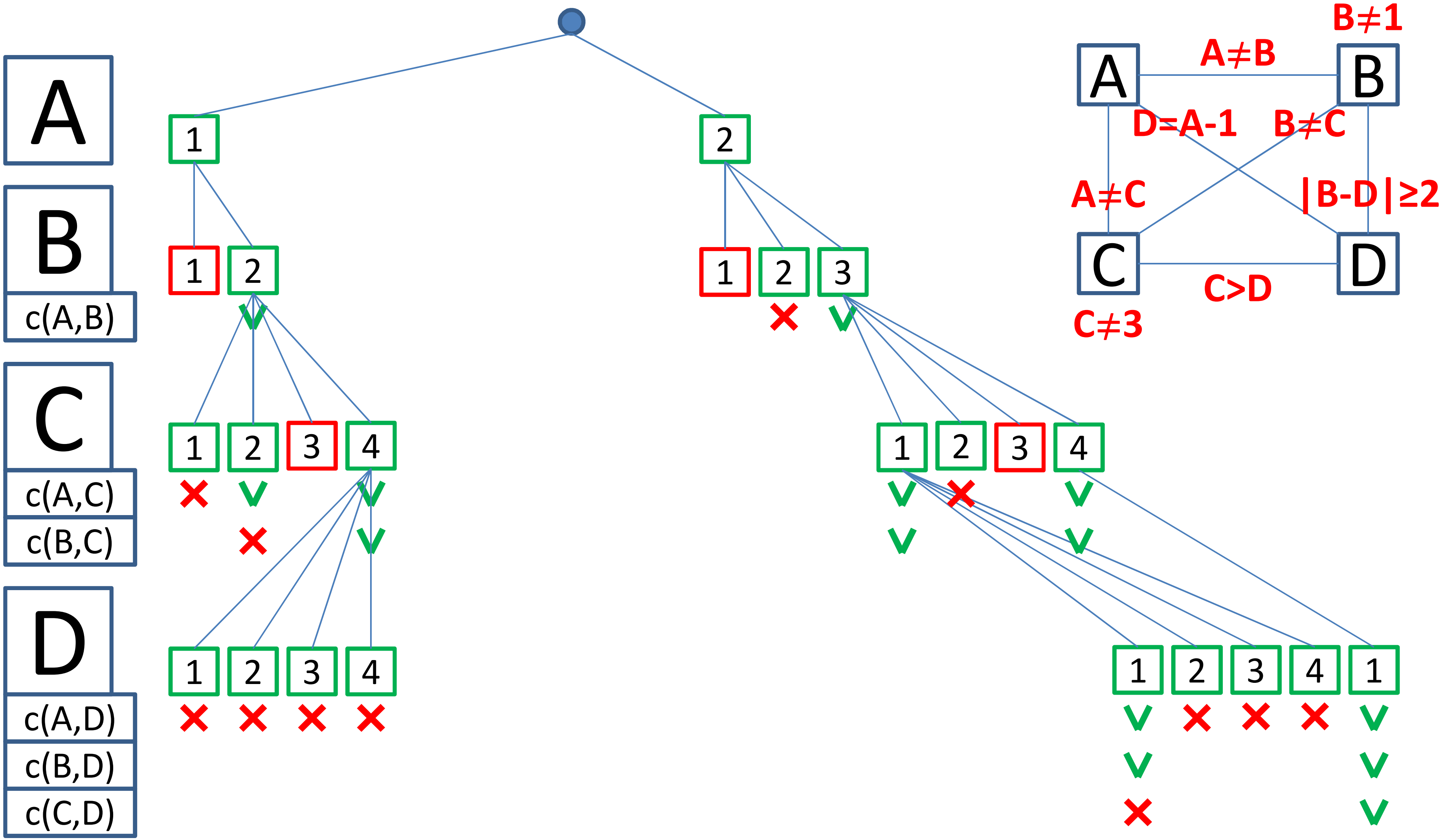
- Problem representation:



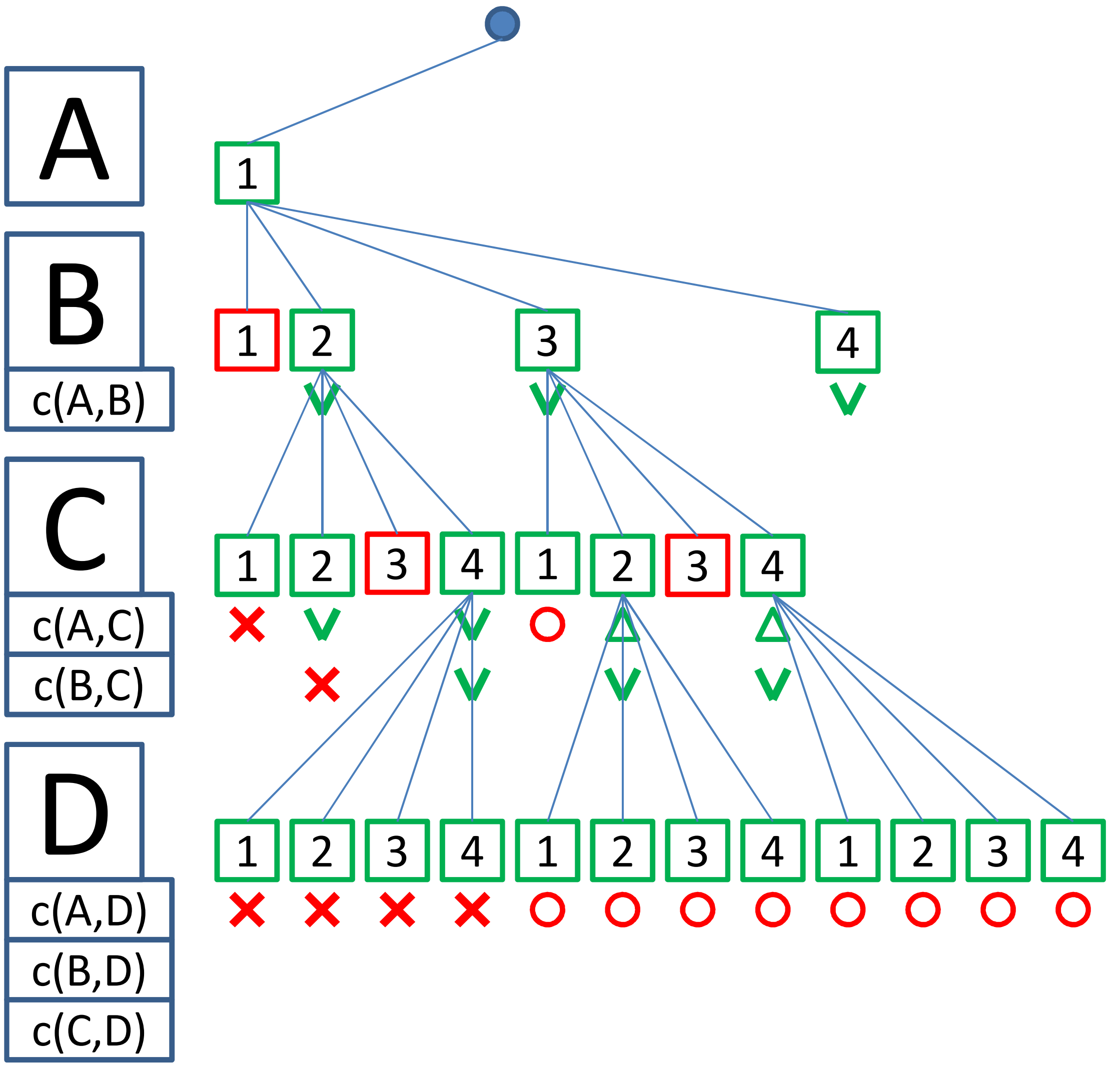
Constraint Processing: Backtracking



Constraint Processing: Backjumping

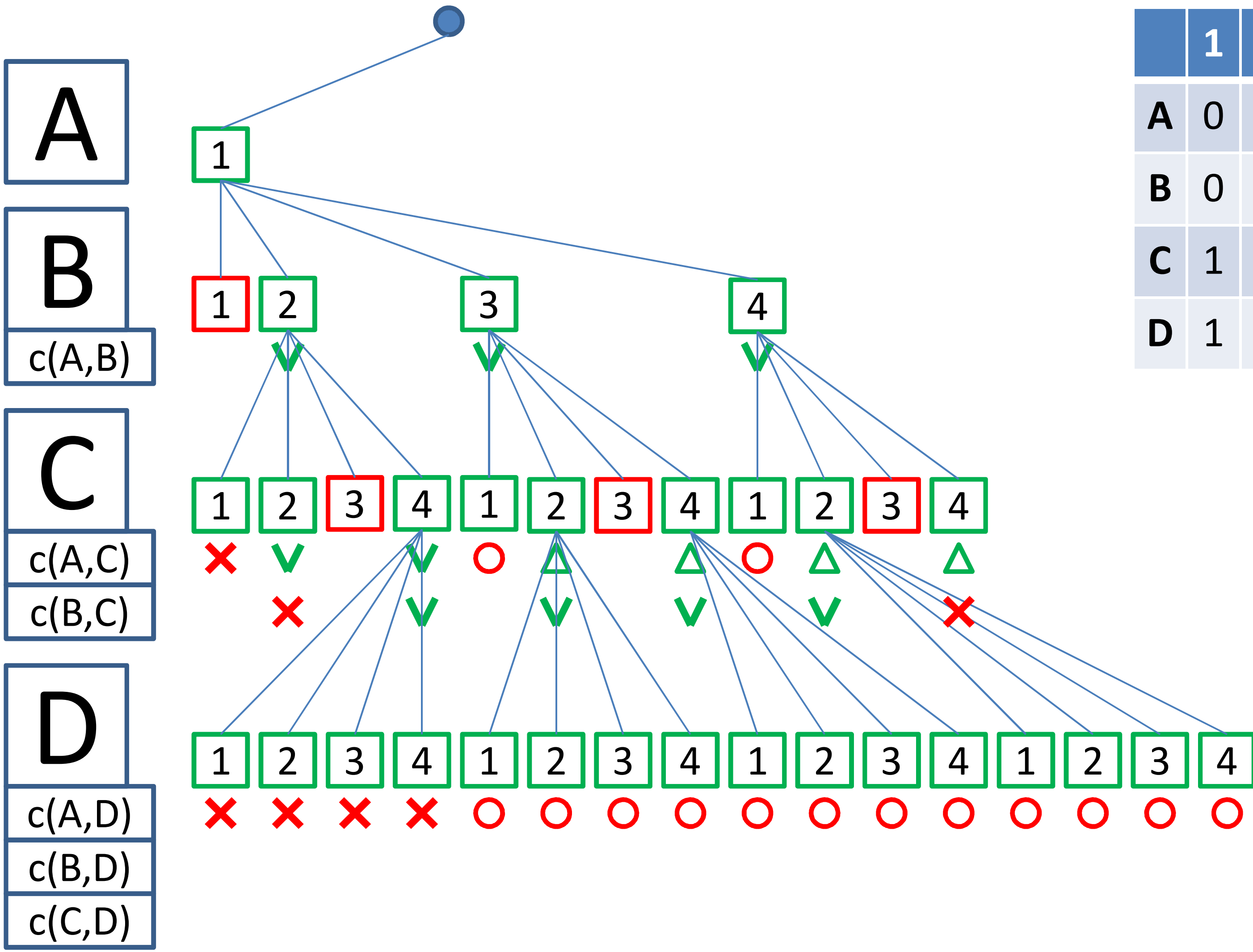


Constraint Processing: Backmarking



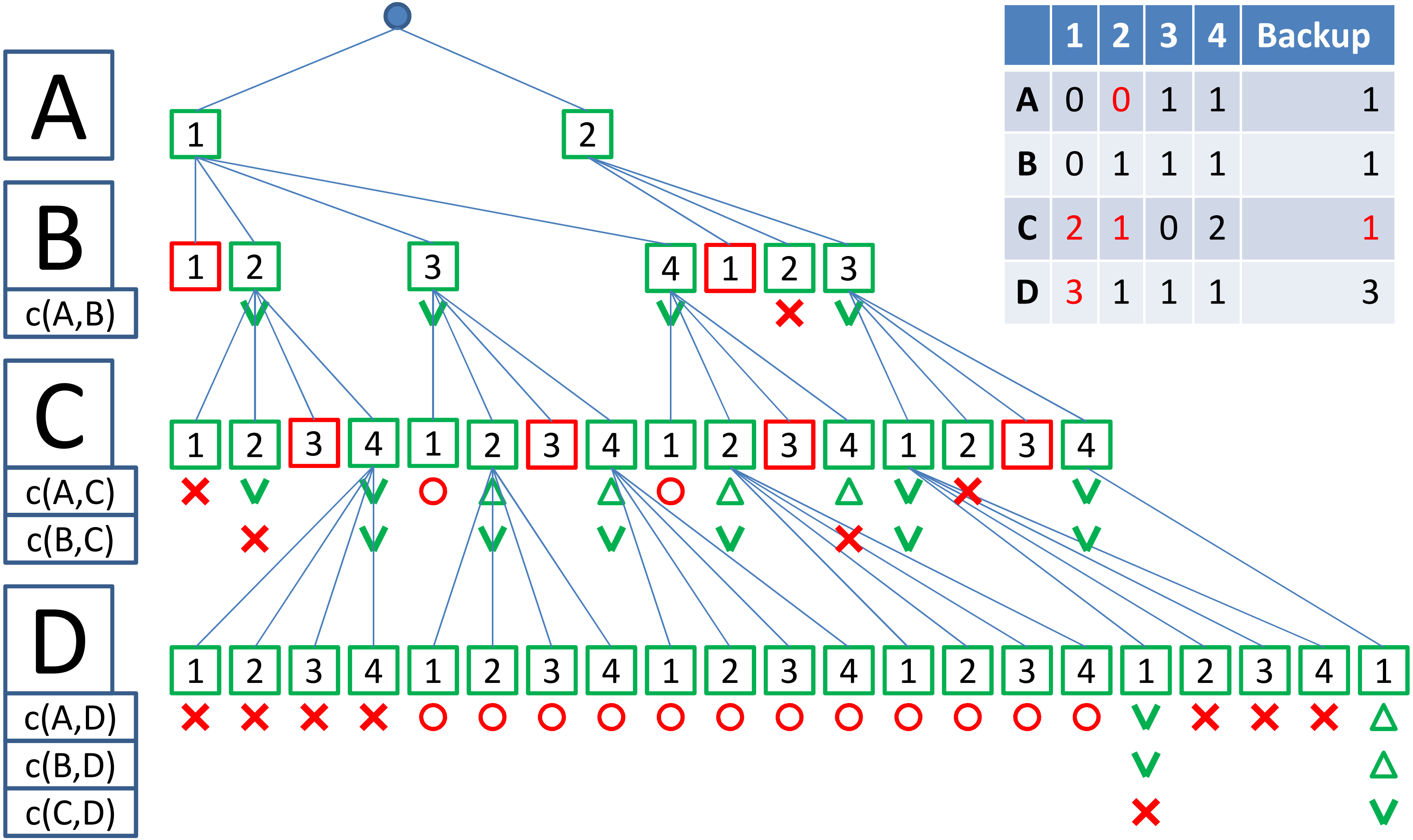
	1	2	3	4	Backup
A	0	1	1	1	1
B	0	1	1	1	1
C	1	2	0	2	2
D	1	1	1	1	2

Constraint Processing: Backmarking



	1	2	3	4	Backup
A	0	1	1	1	1
B	0	1	1	1	1
C	1	2	0	2	2
D	1	1	1	1	3

Constraint Processing: Backmarking

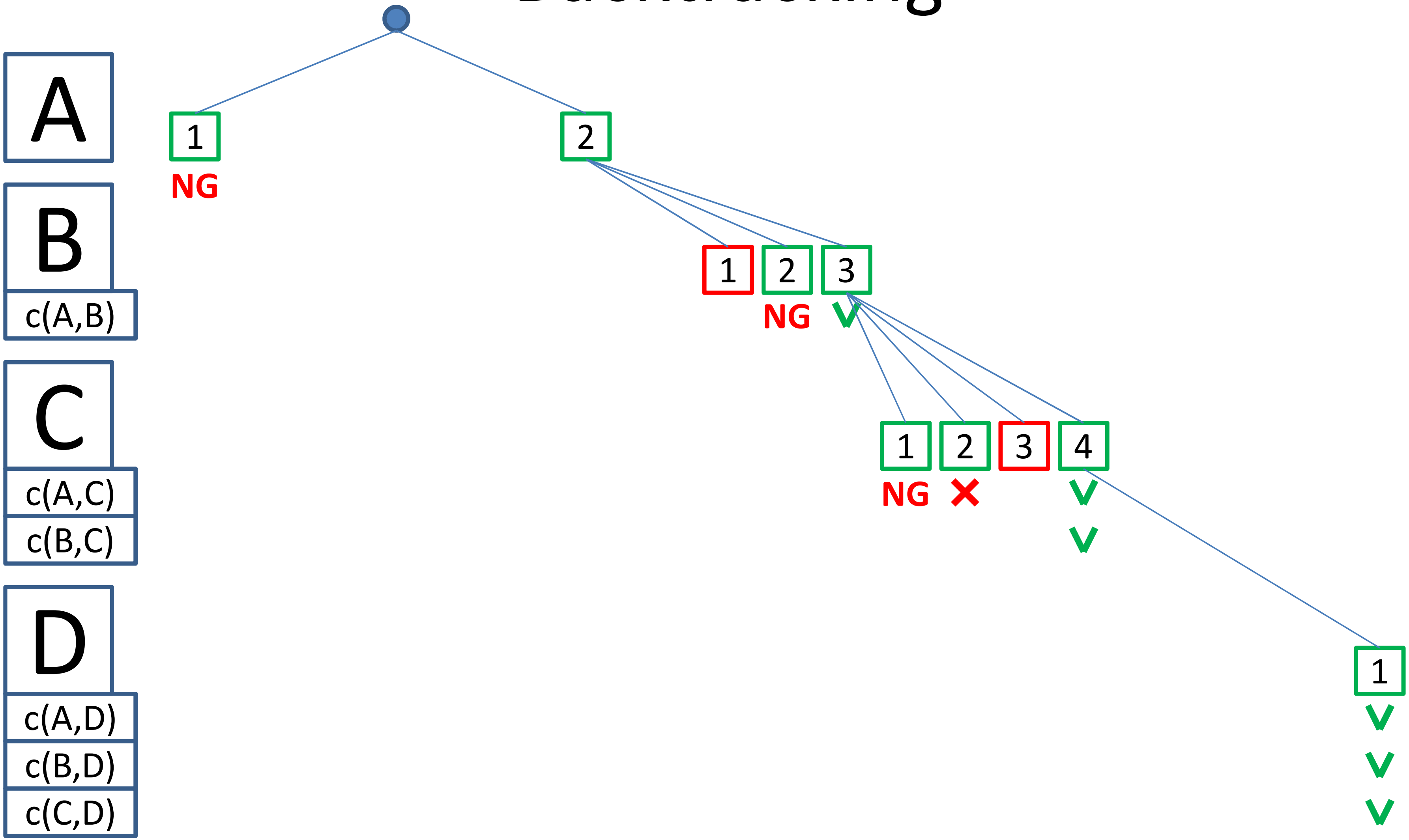


	1	2	3	4	Backup
A	0	0	1	1	1
B	0	1	1	1	1
C	2	1	0	2	1
D	3	1	1	1	3

Constraint Processing: No-goods

- {A=1}: No-good
  - No value for D such that A = D + 1
- {A=2,B=2}: No-good
  - A and B should have different houses
- {A=2,B=3}: Not a no-good: {A=2,B=3,C=4,D=1}
- {A=2,B=3,C=1}: No-good
  - A = D + 1, thus D = 1, but C = 1
- {A=2,B=4}: No-good
  - A = D + 1, thus D = 1, thus C = 3, but C cannot be 3

Constraint Processing: Intelligent Backtracking



Efficiency

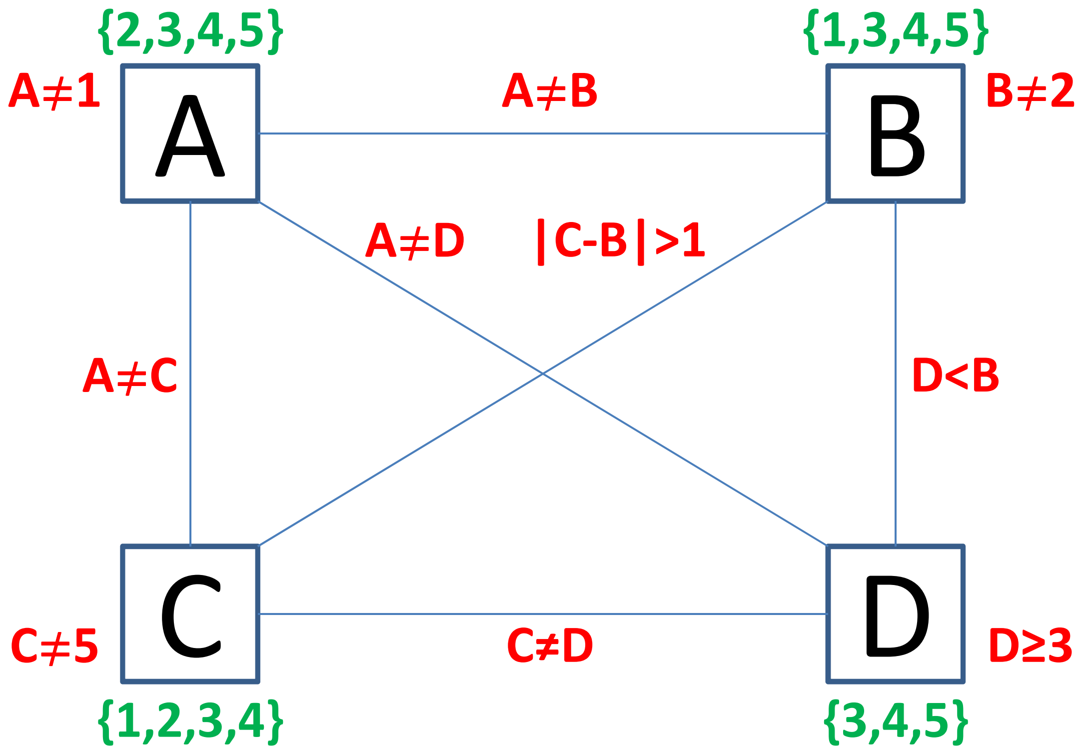
All (One solution)	**Opened Nodes**	**Checks**
Standard Backtracking	28 (13)	142 (56)
Backjumping	21 (8)	93 (30)
Backmarking	28 (13)	79 (34)
Intelligent Backtracking	6 (4)	16 (9) + NG

Exercises: Artificial Intelligence

Constraint Processing II & Waltz: The 4 Teachers problem

Problem Optimization

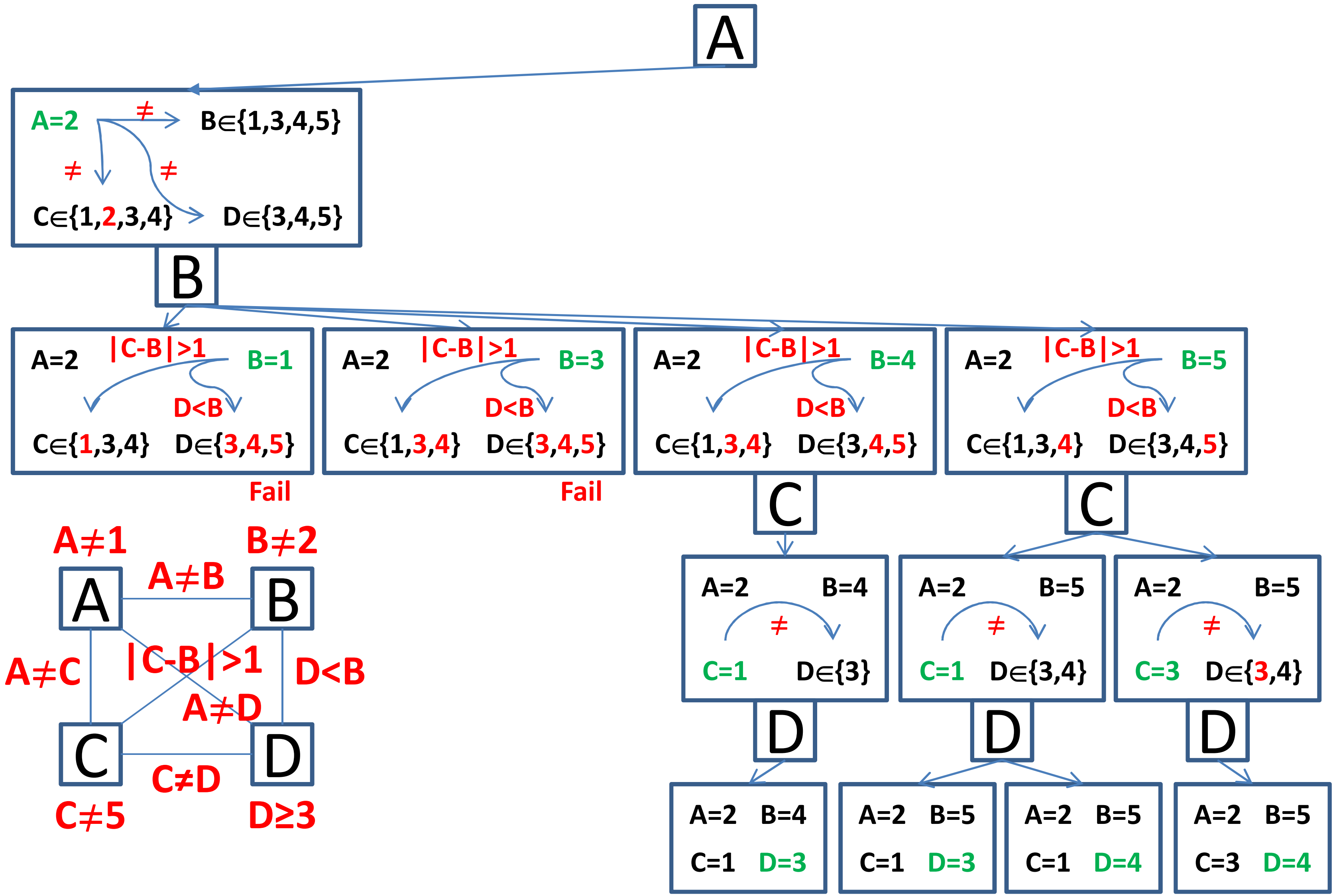
- Problem optimization:



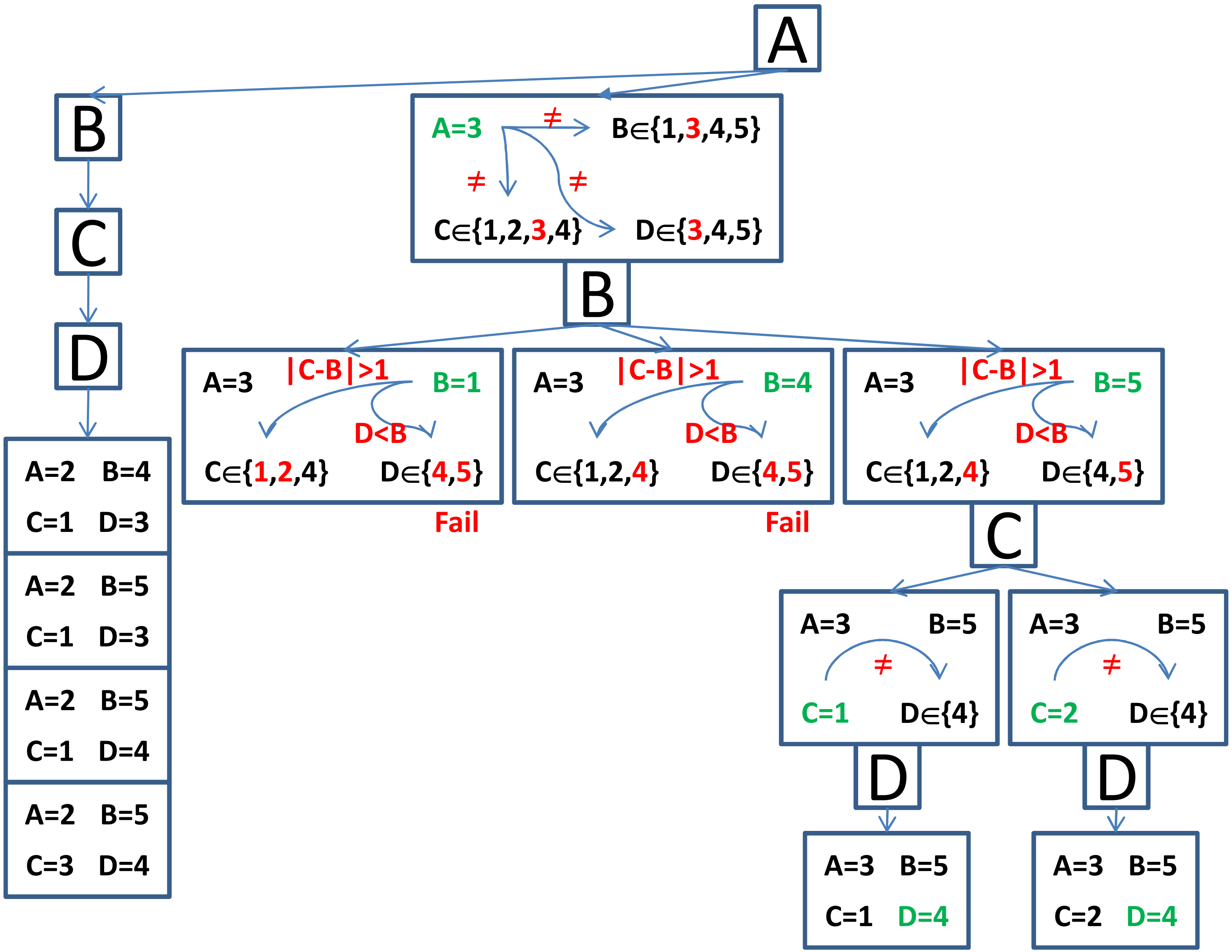
MiniMax & Constraint Processing: The 4 Houses problem

CONSTRAINT PROCESSING: FORWARD CHECKING

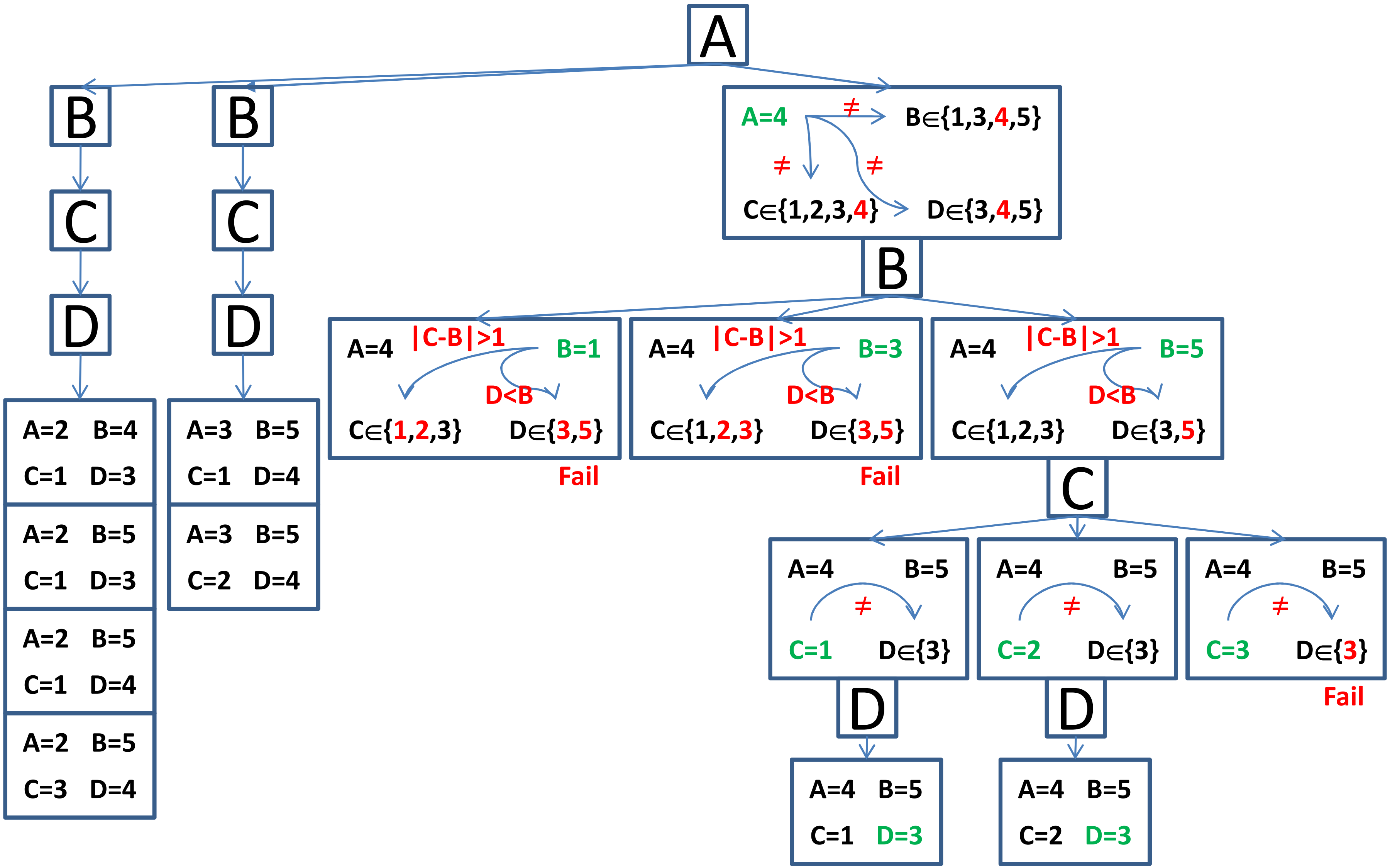
Forward Checking



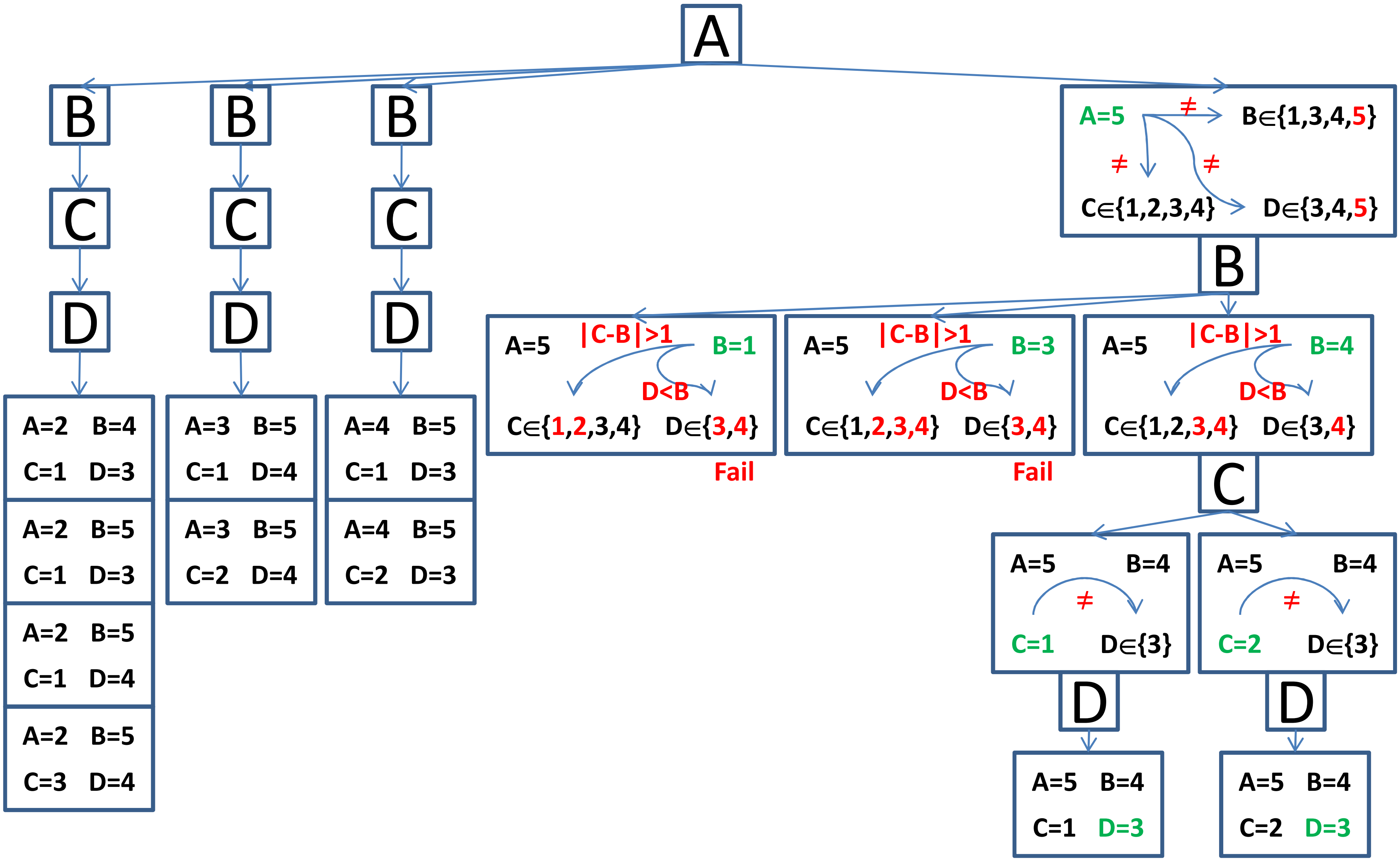
Forward Checking



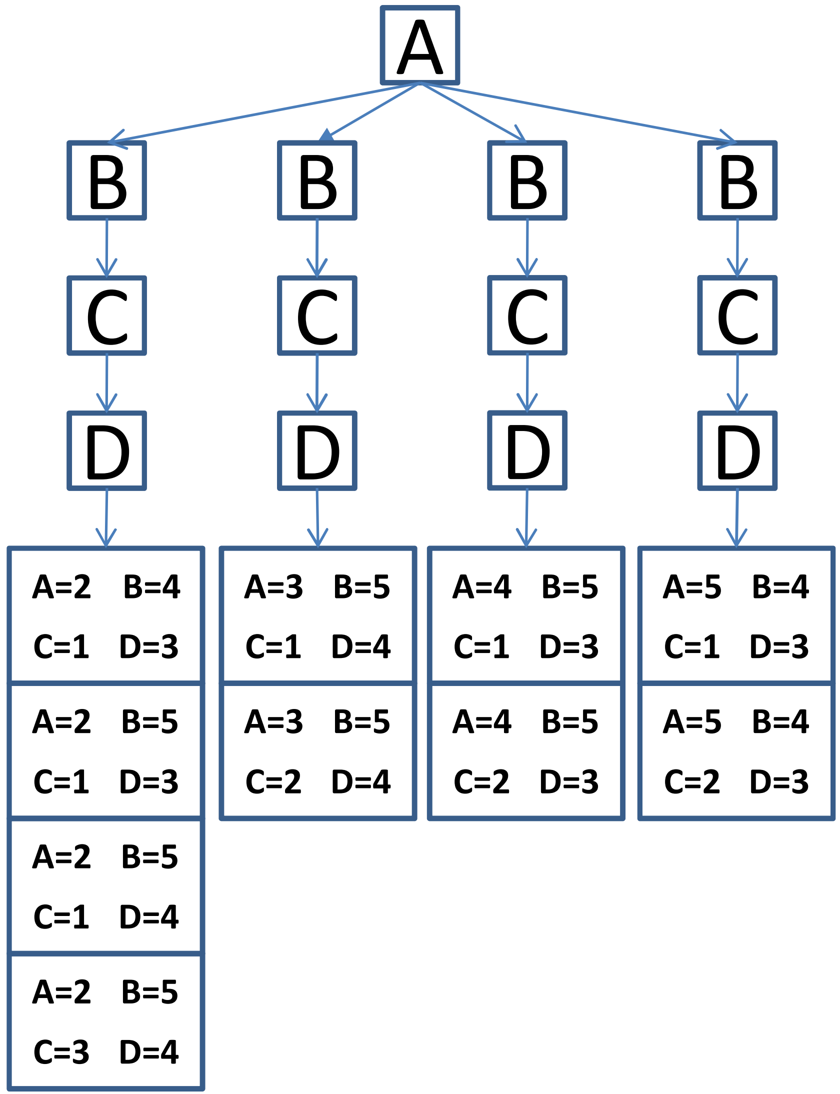
Forward Checking



Forward Checking



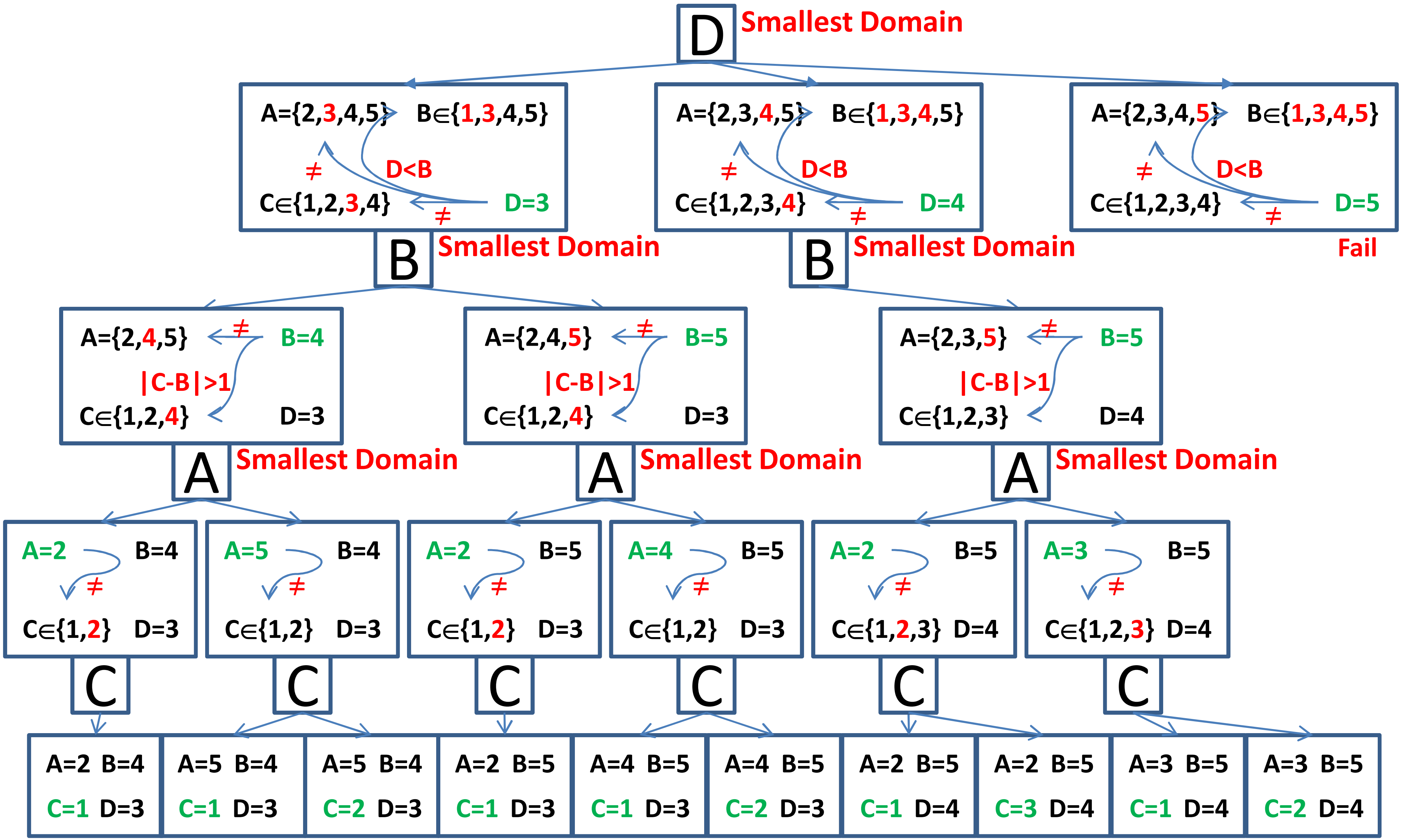
Forward Checking



MiniMax & Constraint Processing: The 4 Houses problem

CONSTRAINT PROCESSING: DYNAMIC SEARCH REARRANGEMENT FC

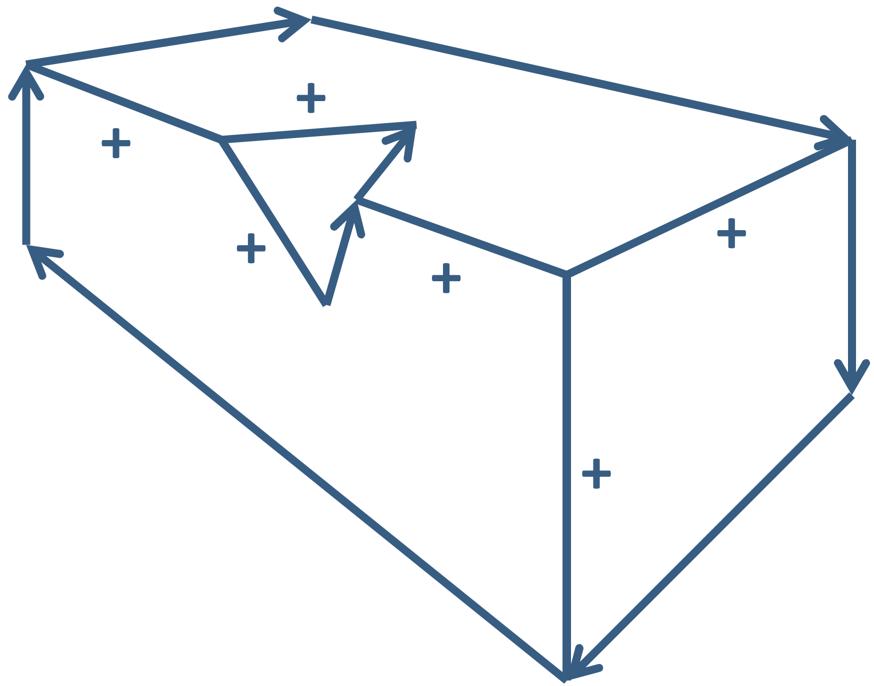
Dynamic Search Rearrangement FC



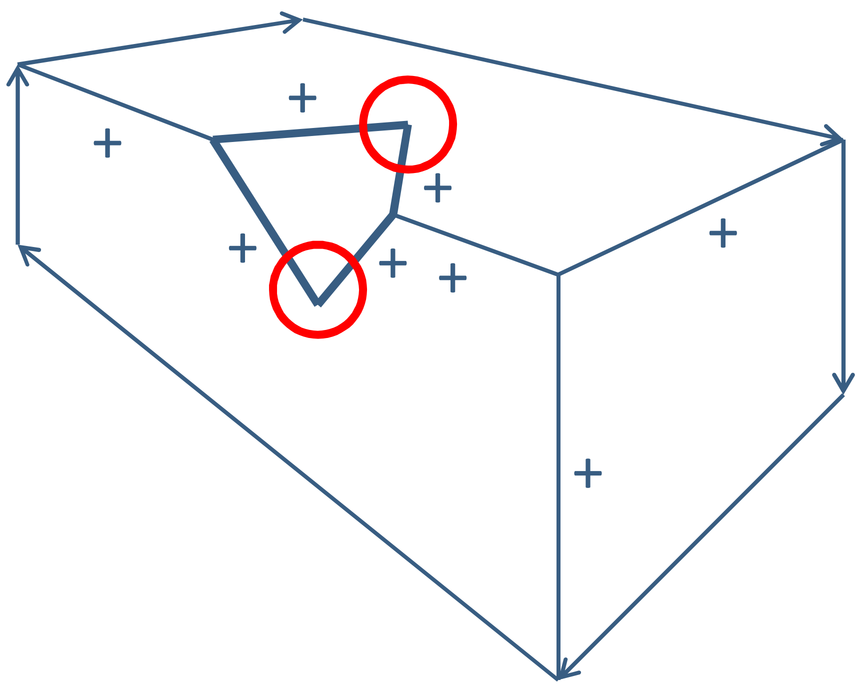
Exercises: Artificial Intelligence

Constraint Processing II & Waltz:
Waltz I

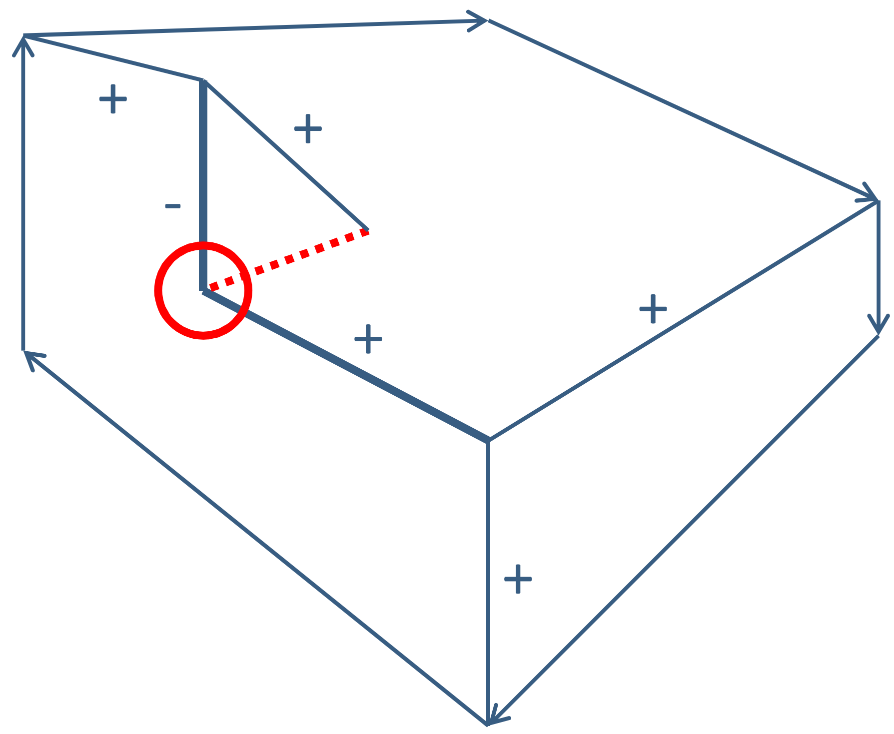
Solution



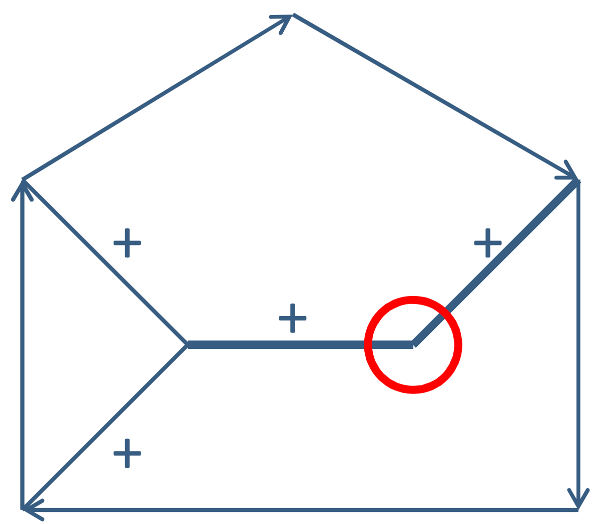
Solution



Solution

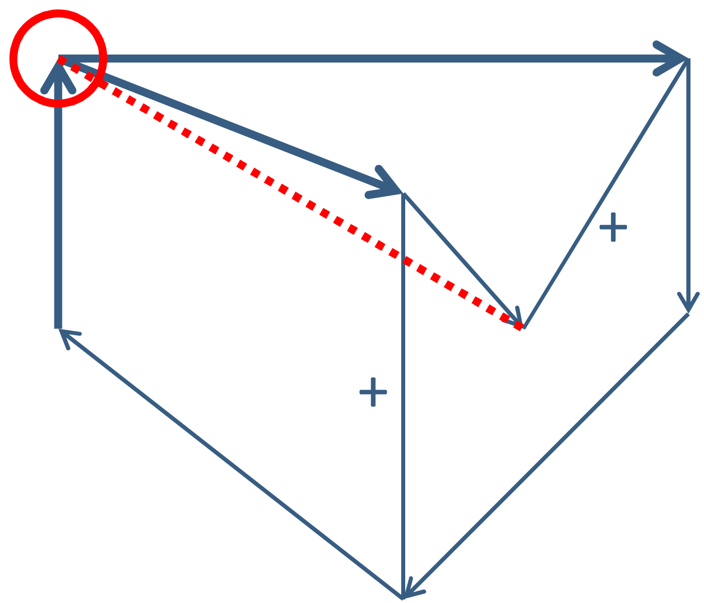


Solution



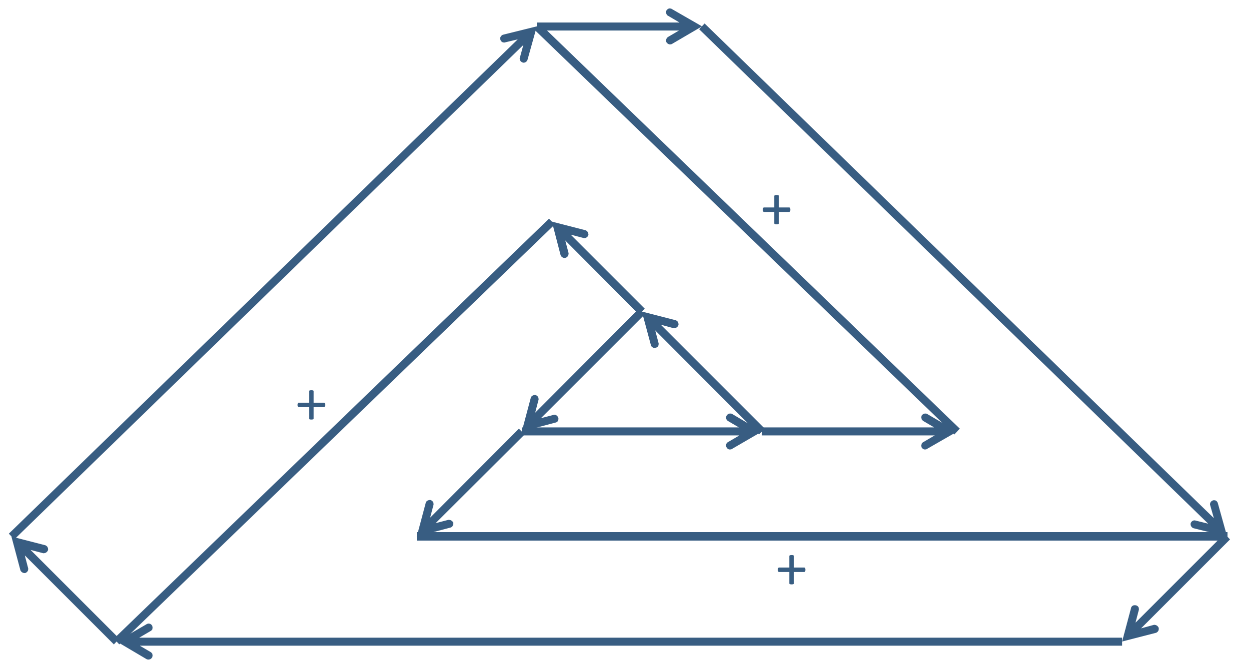
Solution

Line Drawing NOT allowed: 3-faced vertices!!



Solution

Drawing is locally correct, but is globally impossible.
Waltz procedure is local, thus, cannot detect this!

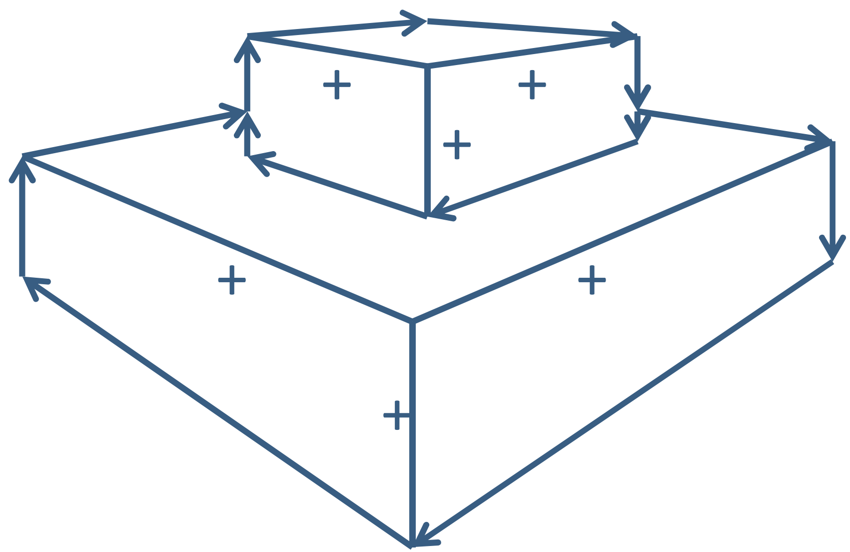


Exercises: Artificial Intelligence

Constraint Processing II & Waltz: Waltz II

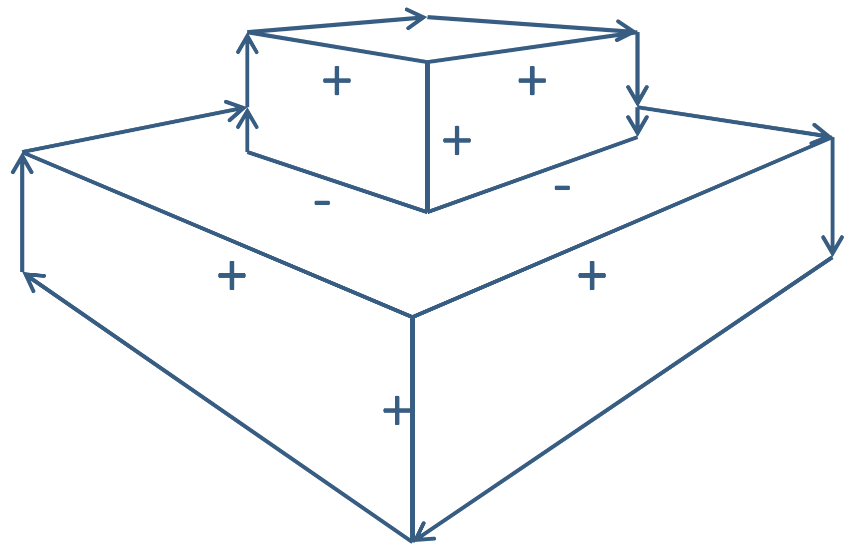
Solution

- Solution 1: Floating cube



Solution

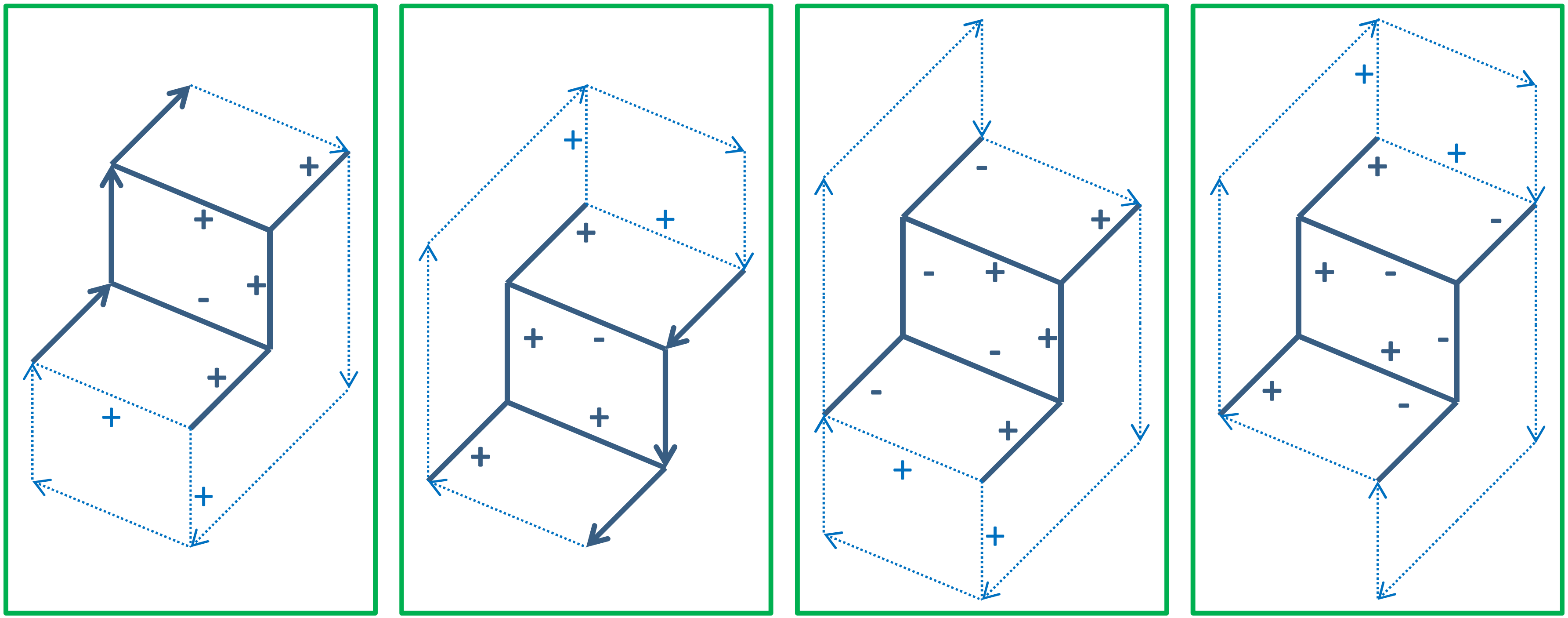
- Solution 2: Sitting cube



Exercises: Artificial Intelligence

Constraint Processing II & Waltz: Waltz III

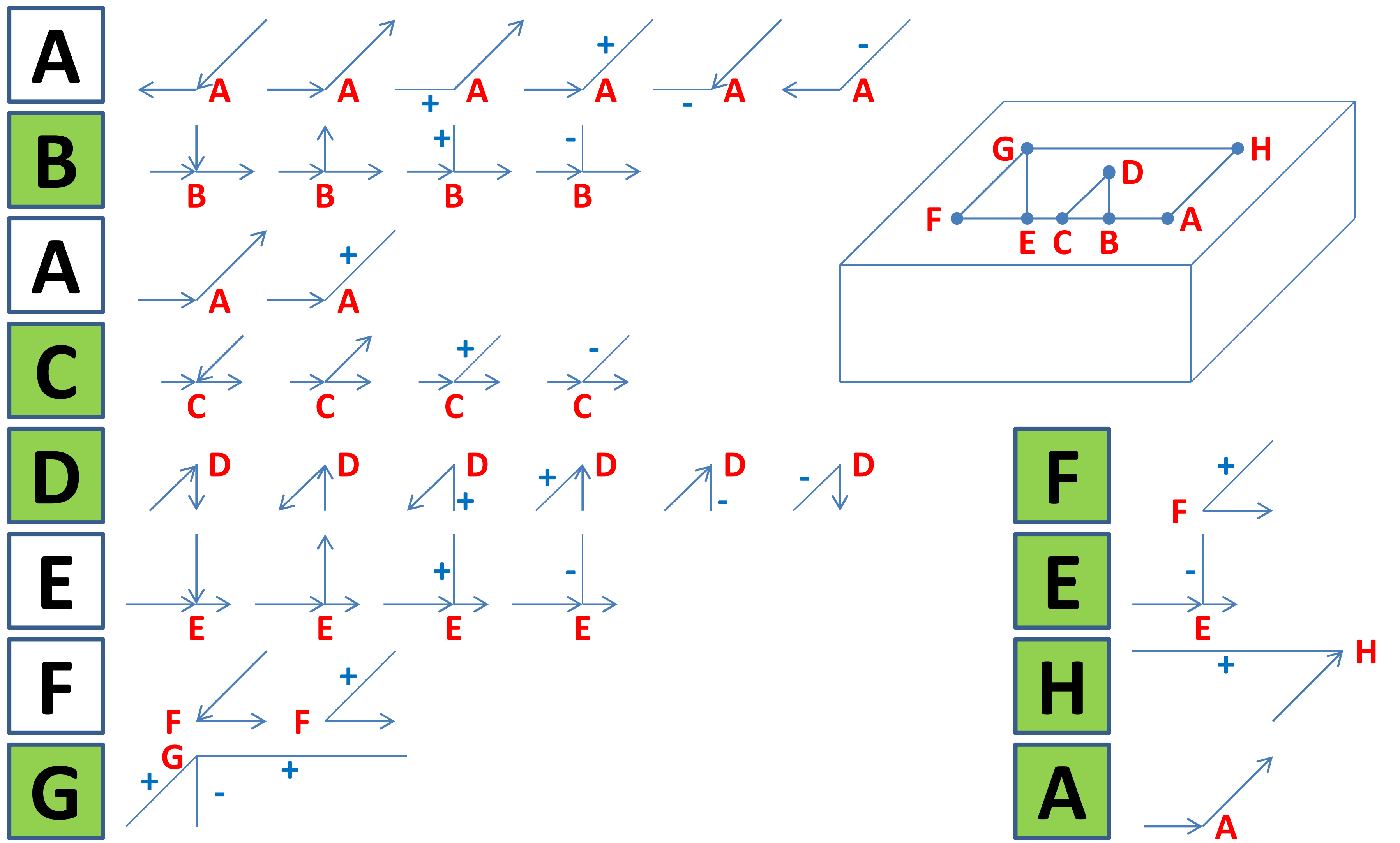
Solution



Exercises: Artificial Intelligence

Constraint Processing II & Waltz:
Waltz IV

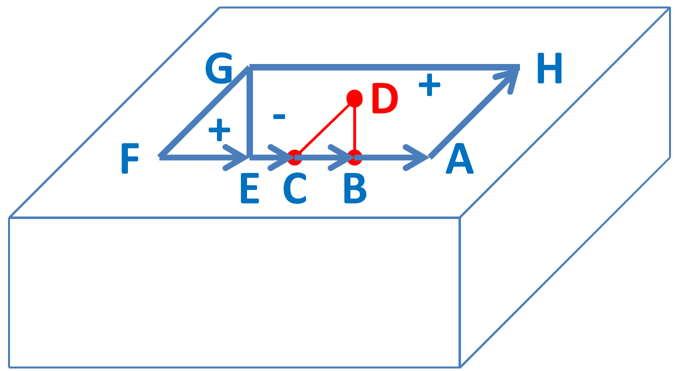
Solution



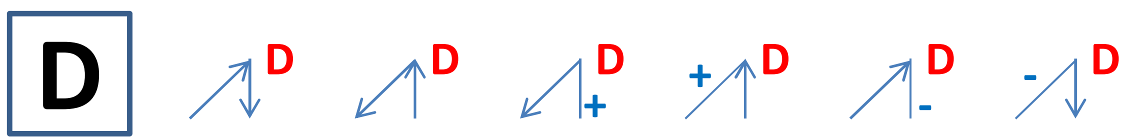
A
B
A
C
D
E
F
G
A A A + A + A - A - A
B B + B - B
A A + A
C C + C - C
D D D + D + D D - D - D
E E + E - E
F F + F
G + - +
G H
D
F A
E C B
F
+
F
E
-
E
H
H
+
A
A

Solution

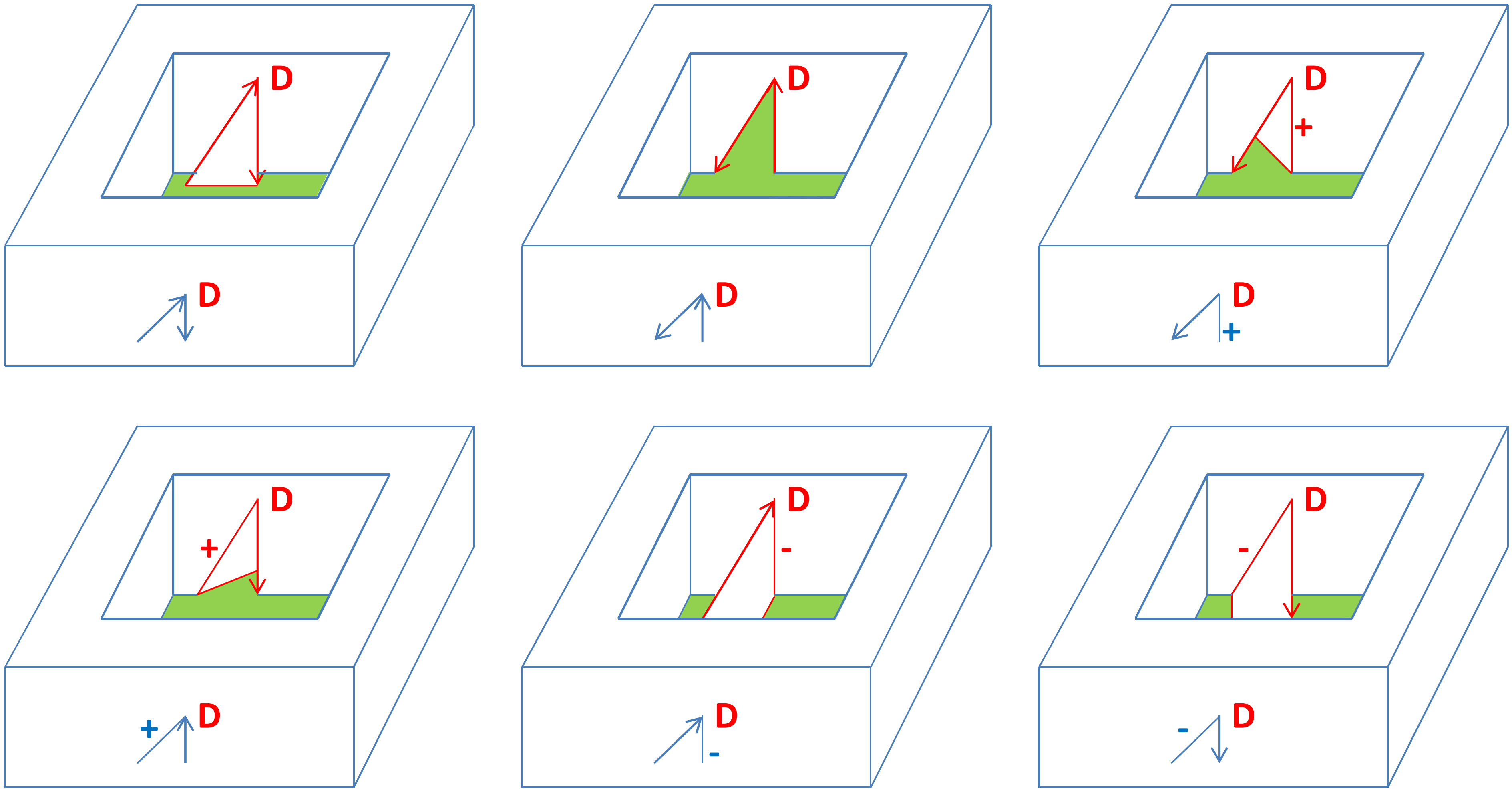
- We can determine **all nodes except** for **D**:



- **D** can still take **6 interpretations**:



Solution



Exercises: Artificial Intelligence

Constraint Processing II & Waltz:
Waltz V

Termination Waltz

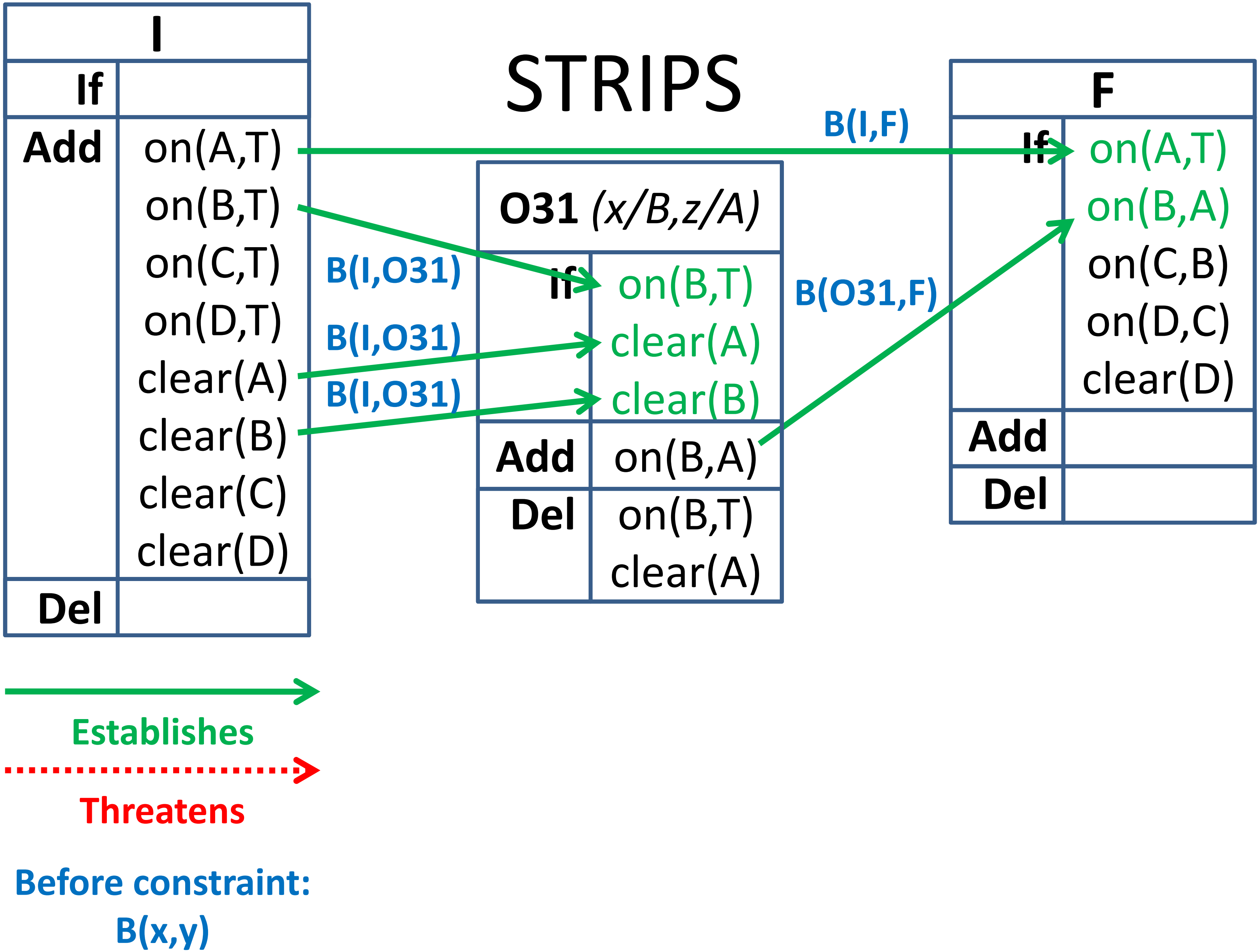
- ***Waltz****'s procedure* ***terminates*** *if*
  - <u>No possibilities</u> for some vertex **OR**
  - <u>No reduction</u> of junction piles
- ***Waltz****'s procedure* ***does not terminate*** *if*
  - <u>Only non-empty piles</u> **AND**
  - <u>Reduction of piles possible</u>

- **BUT**
  - Piles are finite $\Longrightarrow$ Number of iterations finite

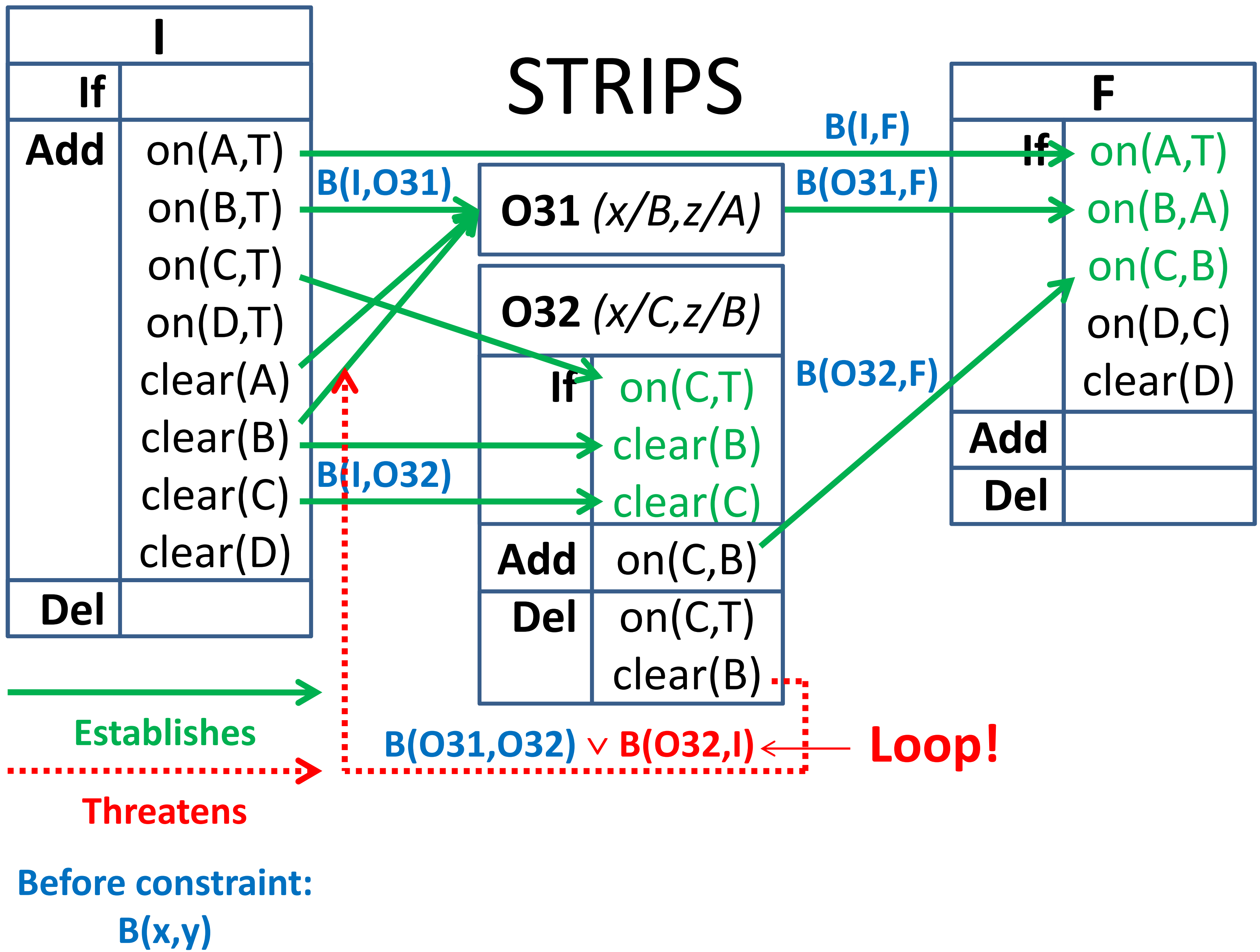
  $\Longrightarrow$ **Waltz**'s procedure **terminates**

Exercises: Artificial Intelligence

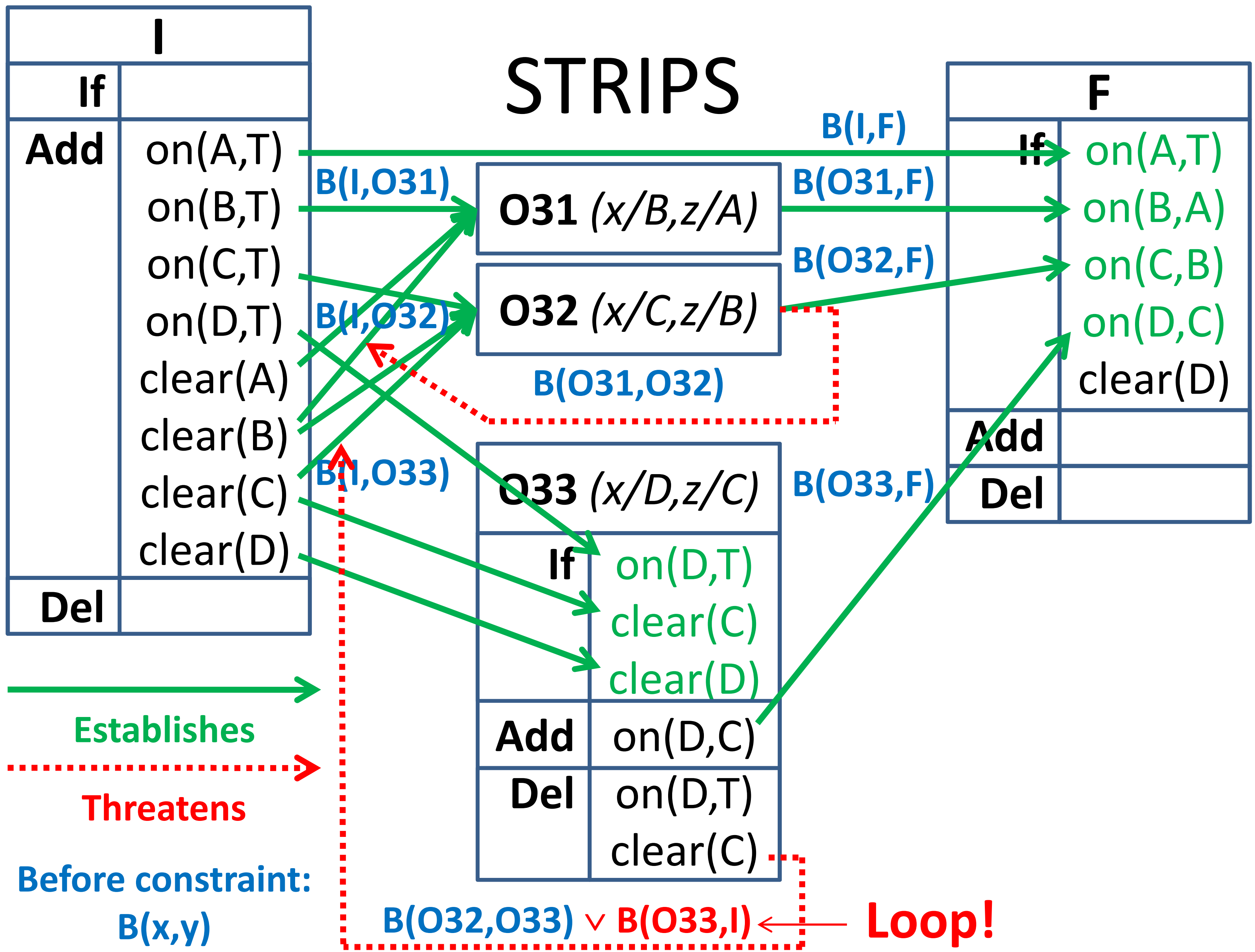
Planning & Logic: Blocks world



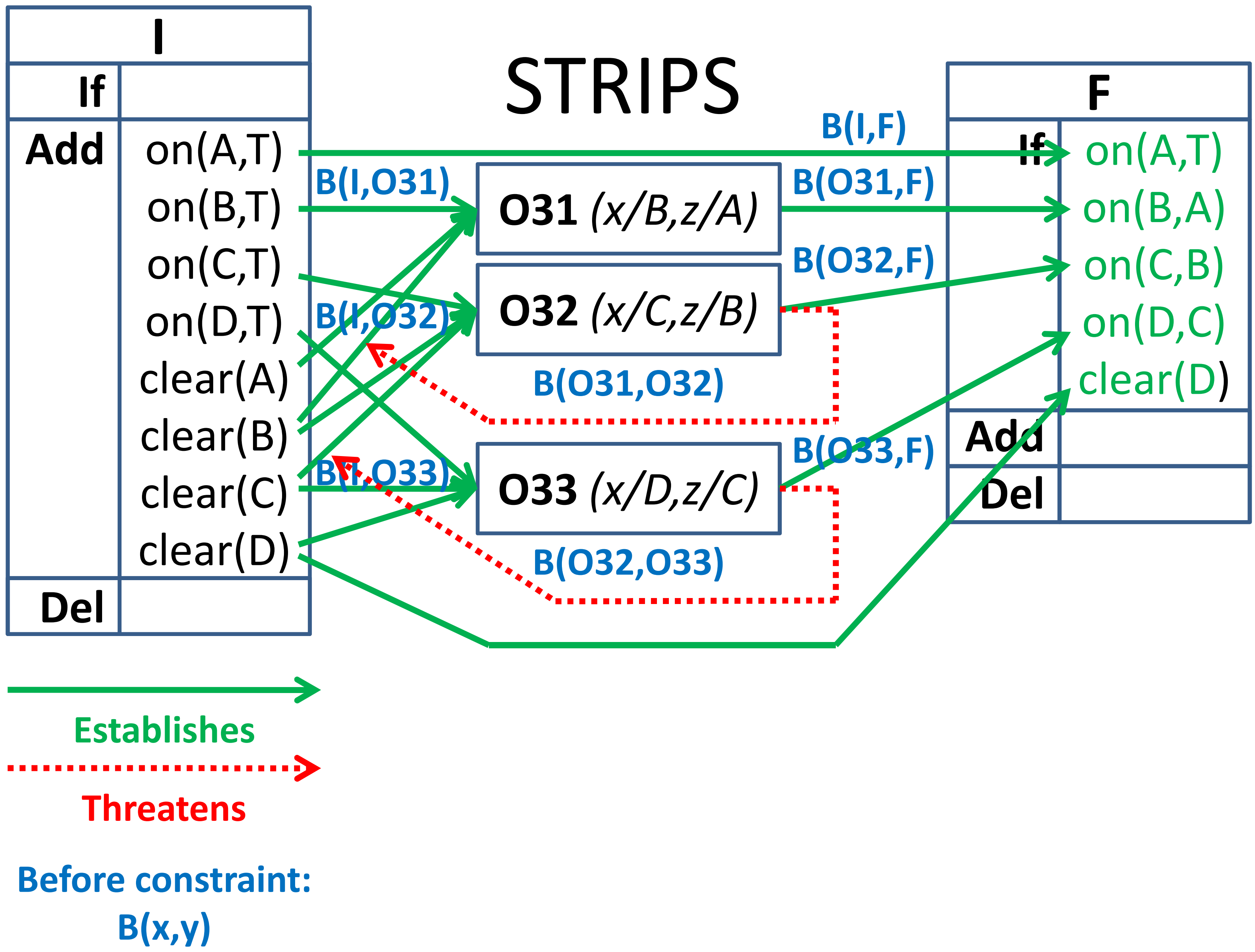
I
If
Add
on(A,T)
on(B,T)
on(C,T)
on(D,T)
clear(A)
clear(B)
clear(C)
clear(D)
Del
STRIPS
O31 (x/B,z/A)
If
on(B,T)
clear(A)
clear(B)
Add
on(B,A)
Del
on(B,T)
clear(A)
F
If
on(A,T)
on(B,A)
on(C,B)
on(D,C)
clear(D)
Add
Del
B(I,F)
B(I,O31)
B(I,O31)
B(I,O31)
B(O31,F)
Establishes
Threatens
Before constraint:
B(x,y)



STRIPS
I
If
Add
on(A,T)
on(B,T)
on(C,T)
on(D,T)
clear(A)
clear(B)
clear(C)
clear(D)
Del
B(I,F)
B(I,O31)
B(O31,F)
O31 (x/B,z/A)
O32 (x/C,z/B)
If
on(C,T)
clear(B)
clear(C)
Add
on(C,B)
Del
on(C,T)
clear(B)
B(O32,F)
B(I,O32)
F
If
on(A,T)
on(B,A)
on(C,B)
on(D,C)
clear(D)
Add
Del
Establishes
Threatens
Before constraint:
B(x,y)
B(O31,O32) ∨ B(O32,I)
Loop!

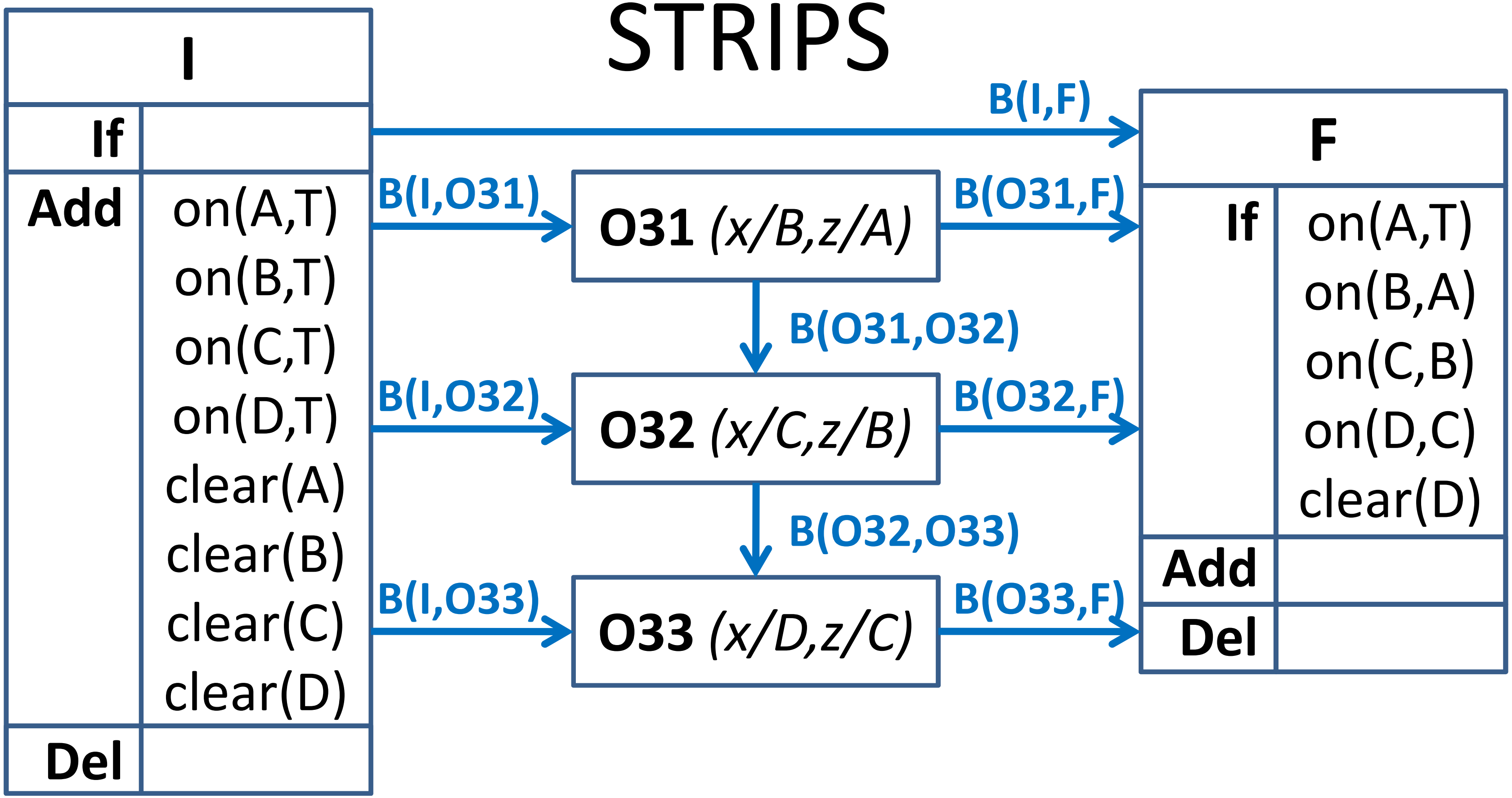


STRIPS
I
If
Add
on(A,T)
on(B,T)
on(C,T)
on(D,T)
clear(A)
clear(B)
clear(C)
clear(D)
Del
B(I,F)
B(I,O31)
B(O31,F)
B(O32,F)
B(I,O32)
B(O31,O32)
B(I,O33)
B(O33,F)
O31 (x/B,z/A)
O32 (x/C,z/B)
O33 (x/D,z/C)
If
on(D,T)
clear(C)
clear(D)
Add
on(D,C)
Del
on(D,T)
clear(C)
F
If
on(A,T)
on(B,A)
on(C,B)
on(D,C)
clear(D)
Add
Del
Establishes
Threatens
Before constraint:
B(x,y)
B(O32,O33) ∨ B(O33,I)
Loop!



I
STRIPS
If
Add
on(A,T)
on(B,T)
on(C,T)
on(D,T)
clear(A)
clear(B)
clear(C)
clear(D)
Del
B(I,F)
B(I,O31)
B(O31,F)
O31 (x/B,z/A)
B(O32,F)
O32 (x/C,z/B)
B(I,O32)
B(O31,O32)
O33 (x/D,z/C)
B(I,O33)
B(O33,F)
B(O32,O33)
F
If
on(A,T)
on(B,A)
on(C,B)
on(D,C)
clear(D)
Add
Del
Establishes
Threatens
Before constraint:
B(x,y)

STRIPS



Are the before constraints satisfiable?

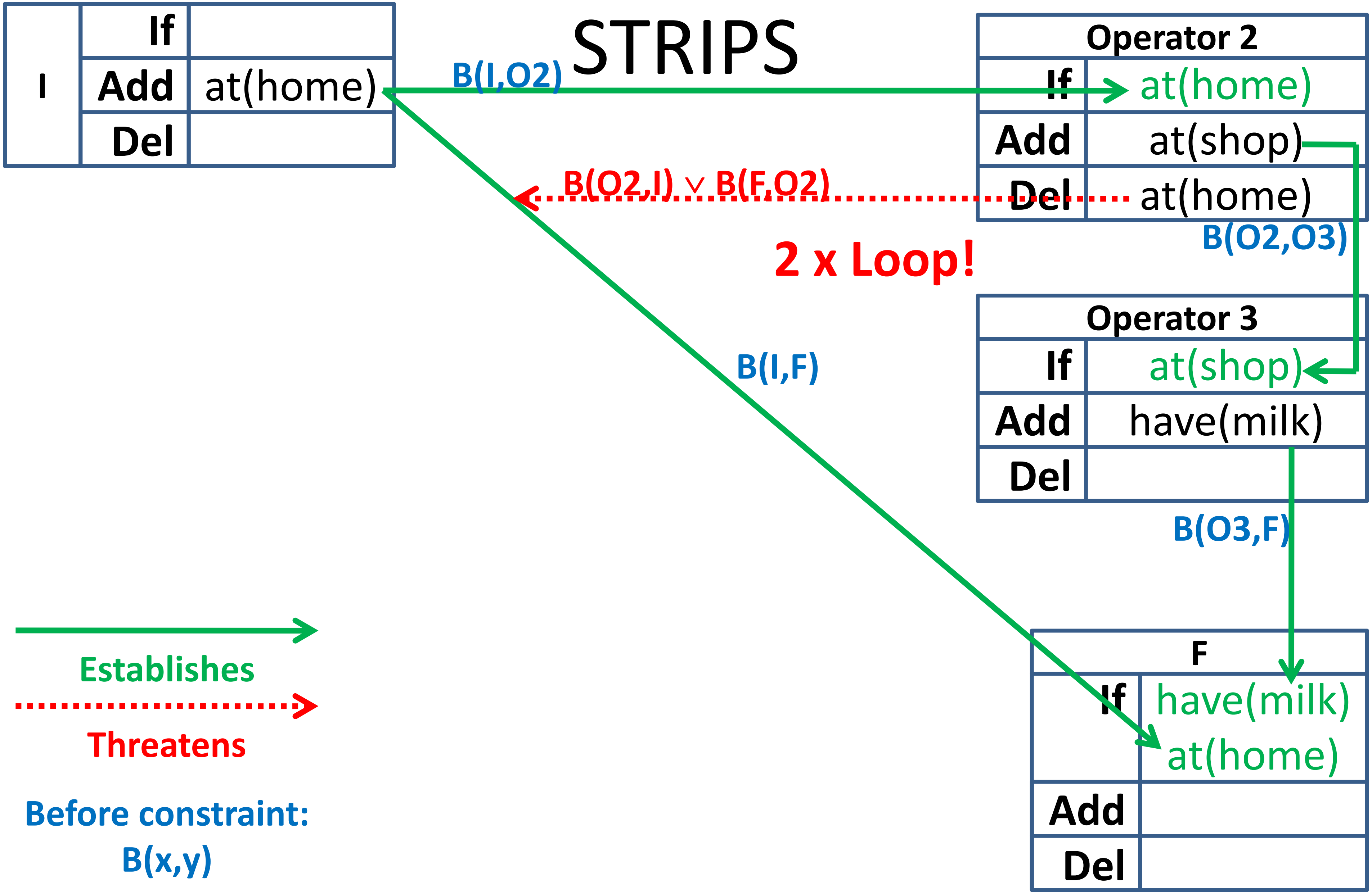
YES:

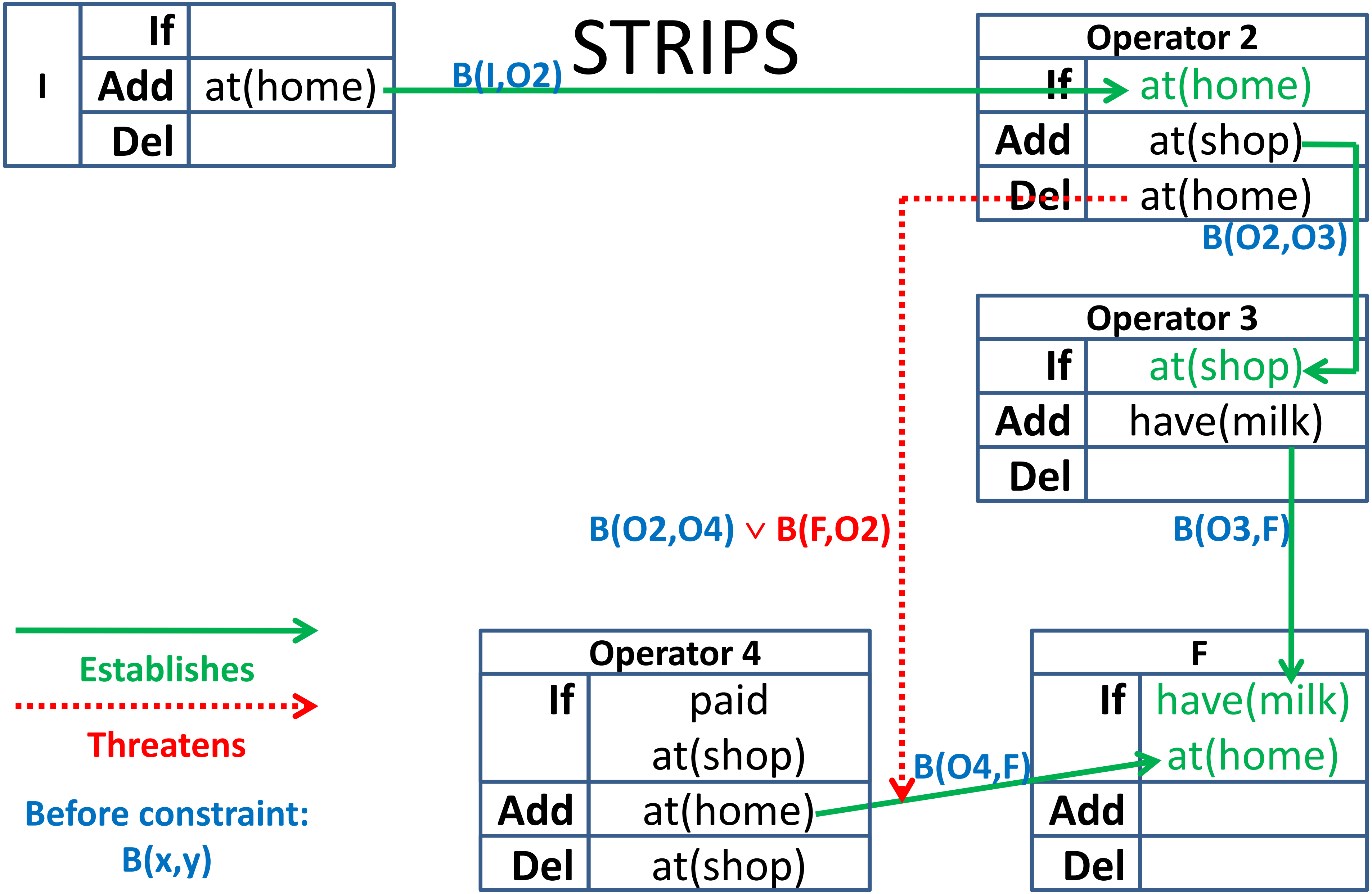
$\longrightarrow$ **O31** $\longrightarrow$ **O32** $\longrightarrow$ **O33** $\longrightarrow$

Exercises: Artificial Intelligence

Planning & Logic: Buying milk

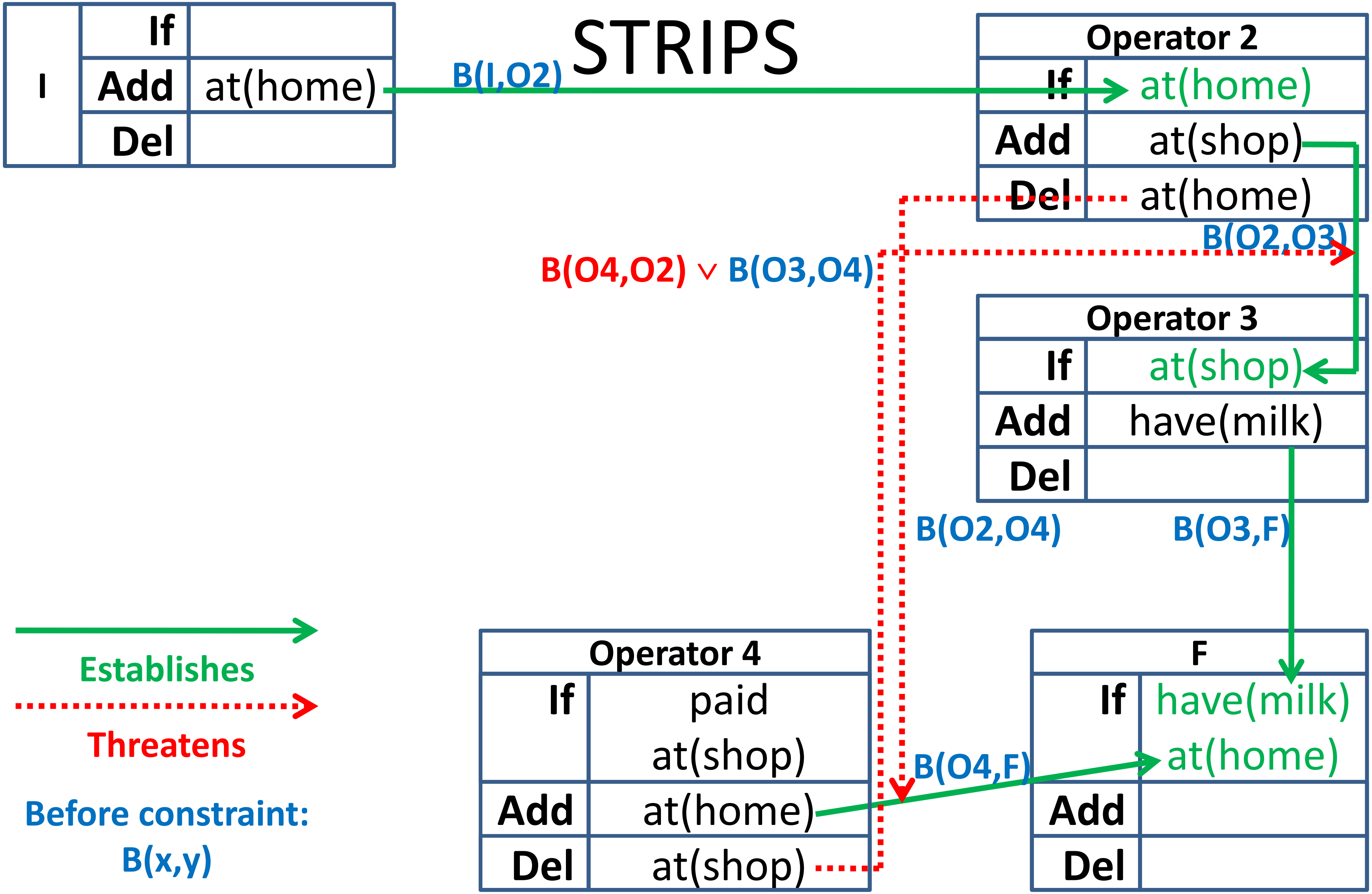
STRIPS


I
If
Add
at(home)
Del
B(I,O2)
Operator 2
If
at(home)
Add
at(shop)
Del
at(home)
B(O2,I) ∨ B(F,O2)
B(O2,O3)
2 x Loop!
Operator 3
If
at(shop)
Add
have(milk)
Del
B(I,F)
B(O3,F)
F
If
have(milk)
at(home)
Add
Del
Establishes
Threatens
Before constraint:
B(x,y)



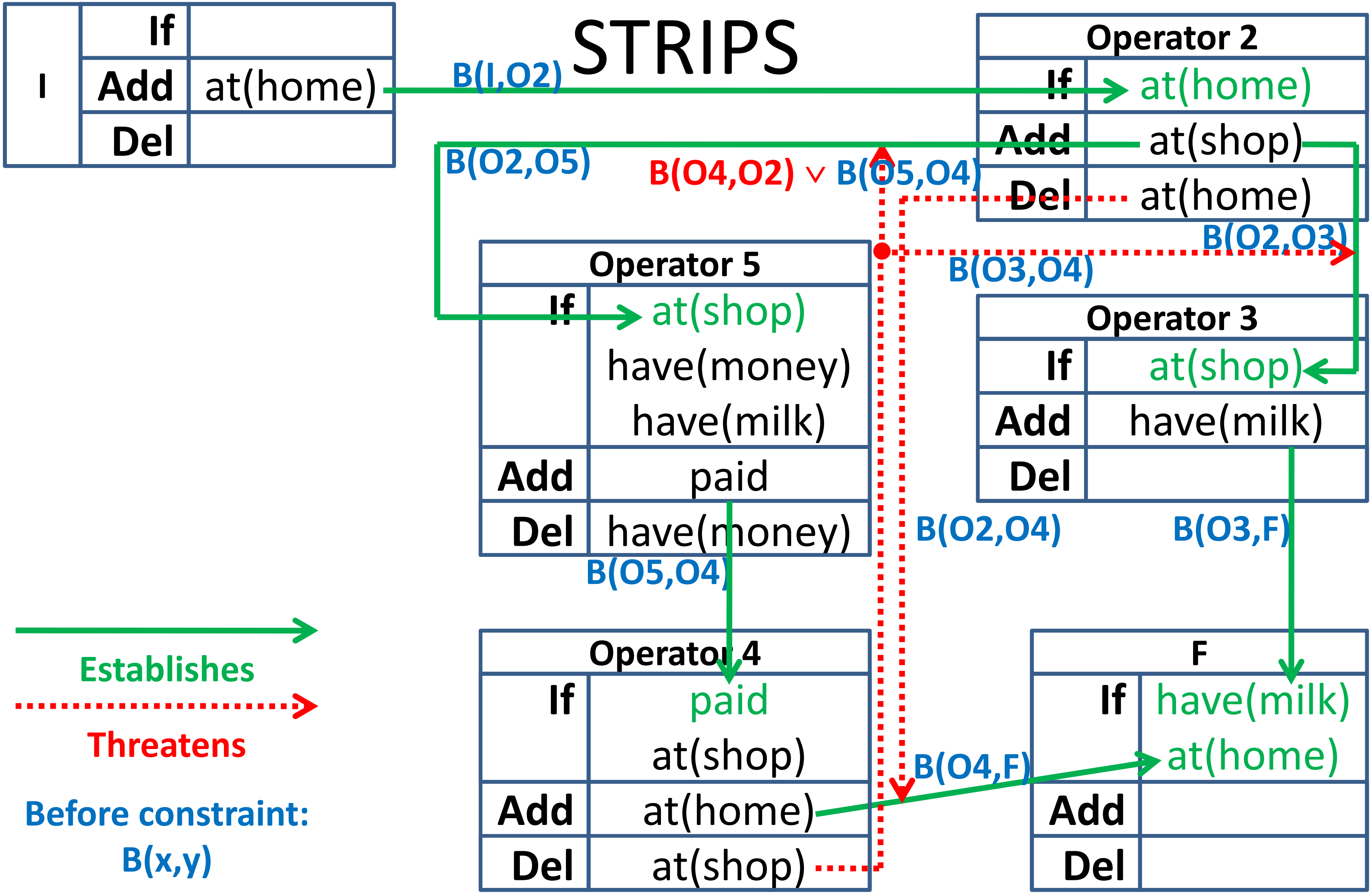
STRIPS
I
If
Add at(home)
Del
B(I,O2)
Operator 2
If at(home)
Add at(shop)
Del at(home)
B(O2,O3)
Operator 3
If at(shop)
Add have(milk)
Del
B(O3,F)
B(O2,O4) ∨ B(F,O2)
Operator 4
If paid
at(shop)
Add at(home)
Del at(shop)
B(O4,F)
F
If have(milk)
at(home)
Add
Del
Establishes
Threatens
Before constraint:
B(x,y)

STRIPS

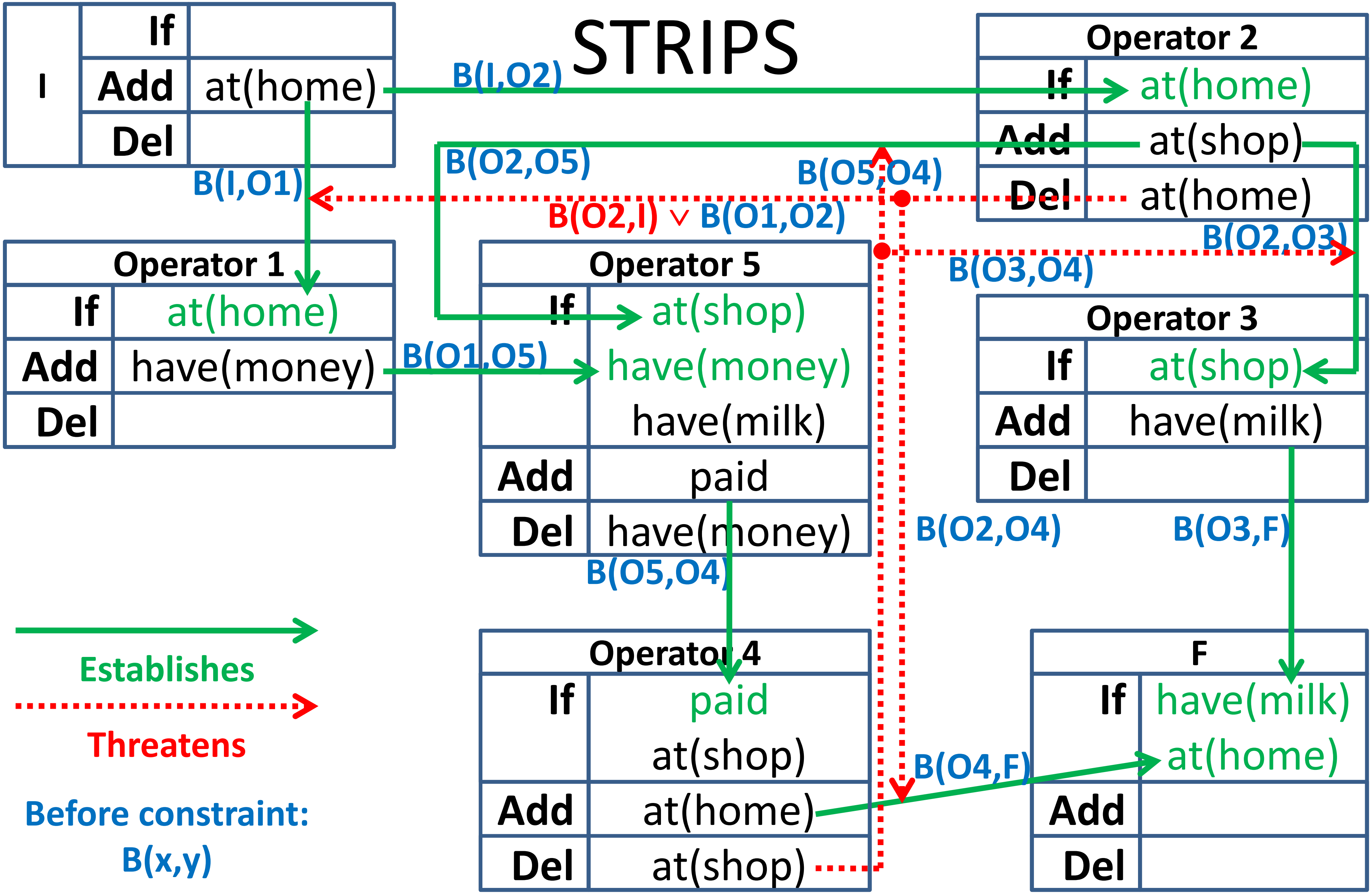


I
If
Add
at(home)
Del
B(I,O2)
Operator 2
If
at(home)
Add
at(shop)
Del
at(home)
B(O2,O3)
B(O4,O2) ∨ B(O3,O4)
Operator 3
If
at(shop)
Add
have(milk)
Del
B(O2,O4)
B(O3,F)
Establishes
Threatens
Before constraint:
B(x,y)
Operator 4
If
paid
at(shop)
Add
at(home)
Del
at(shop)
B(O4,F)
F
If
have(milk)
at(home)
Add
Del

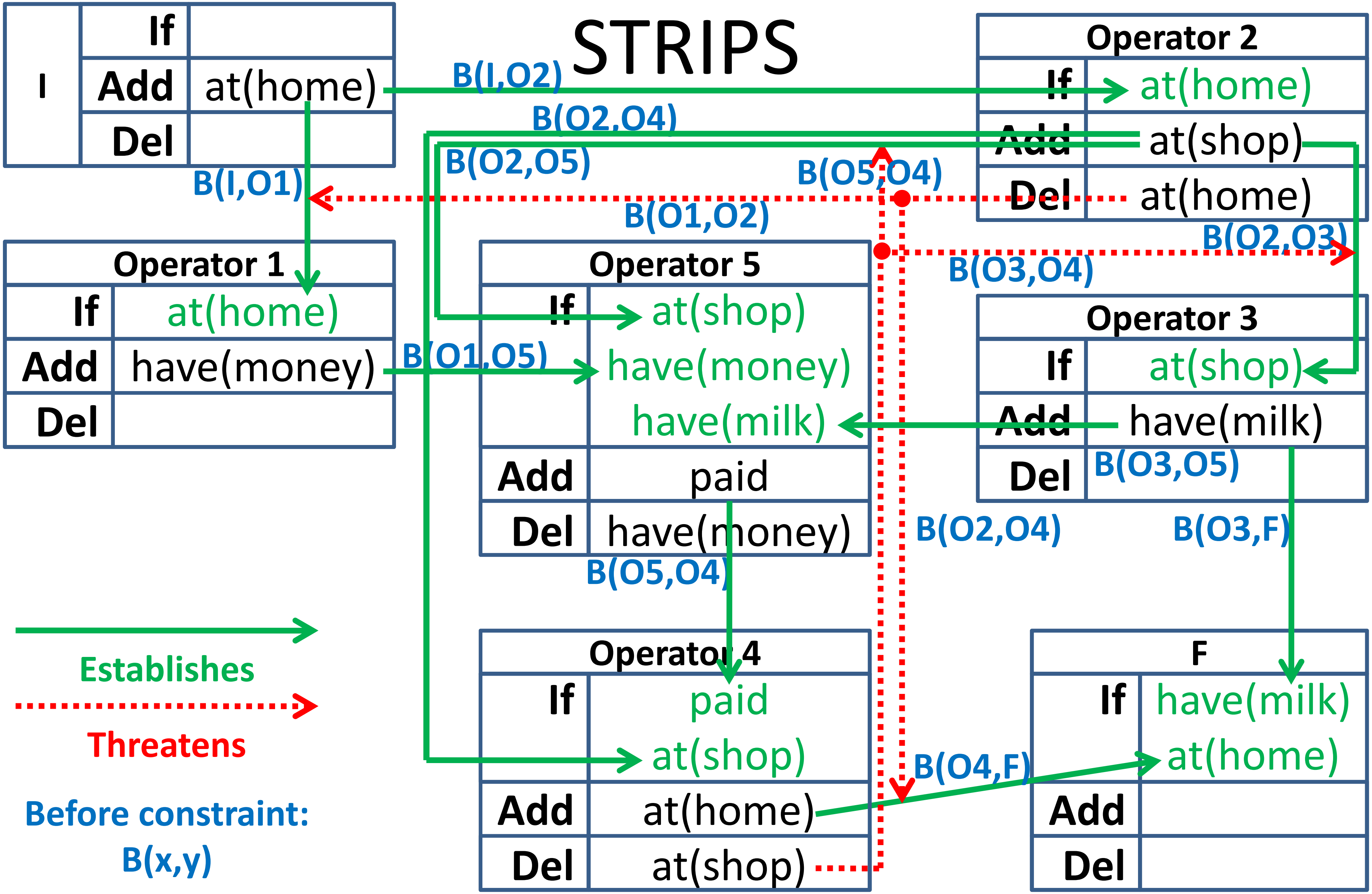
STRIPS


I
If
Add at(home)
Del
B(I,O2)
Operator 2
If at(home)
Add at(shop)
Del at(home)
B(O2,O5)
B(O4,O2) ∨ B(O5,O4)
B(O2,O3)
B(O3,O4)
Operator 5
If at(shop)
have(money)
have(milk)
Add paid
Del have(money)
Operator 3
If at(shop)
Add have(milk)
Del
B(O2,O4)
B(O3,F)
B(O5,O4)
Operator 4
If paid
at(shop)
Add at(home)
Del at(shop)
B(O4,F)
F
If have(milk)
at(home)
Add
Del
Establishes
Threatens
Before constraint:
B(x,y)

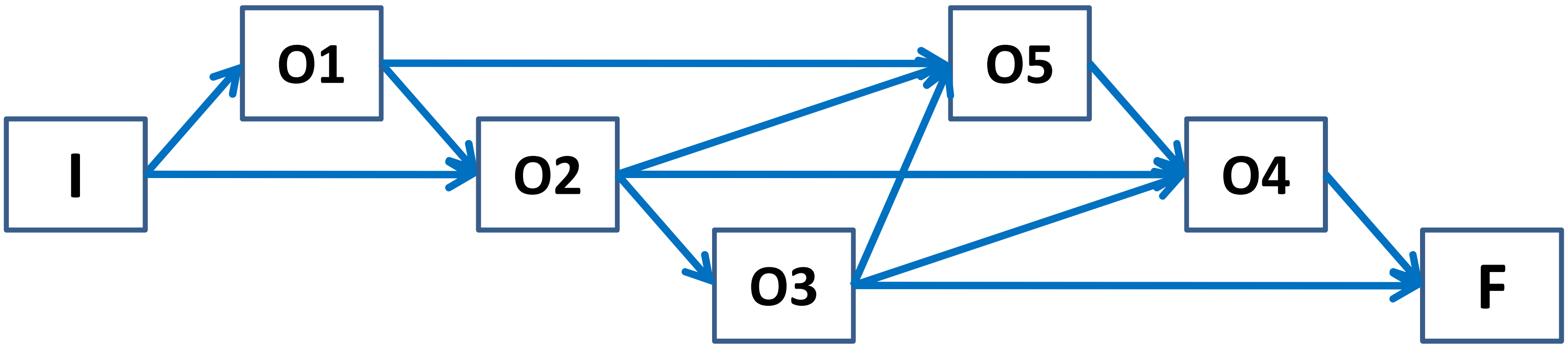
STRIPS



I
If
Add
at(home)
Del
Operator 2
If
at(home)
Add
at(shop)
Del
at(home)
Operator 1
If
at(home)
Add
have(money)
Del
Operator 5
If
at(shop)
have(money)
have(milk)
Add
paid
Del
have(money)
Operator 3
If
at(shop)
Add
have(milk)
Del
Operator 4
If
paid
at(shop)
Add
at(home)
Del
at(shop)
F
If
have(milk)
at(home)
Add
Del
B(I,O2)
B(O2,O5)
B(O5,O4)
B(I,O1)
B(O2,I) ∨ B(O1,O2)
B(O2,O3)
B(O3,O4)
B(O1,O5)
B(O2,O4)
B(O3,F)
B(O5,O4)
B(O4,F)
Establishes
Threatens
Before constraint:
B(x,y)


STRIPS
I
If
Add
at(home)
Del
Operator 1
If
at(home)
Add
have(money)
Del
Operator 2
If
at(home)
Add
at(shop)
Del
at(home)
Operator 3
If
at(shop)
Add
have(milk)
Del
Operator 5
If
at(shop)
have(money)
have(milk)
Add
paid
Del
have(money)
Operator 4
If
paid
at(shop)
Add
at(home)
Del
at(shop)
F
If
have(milk)
at(home)
Add
Del
B(I,O2)
B(O2,O4)
B(O2,O5)
B(I,O1)
B(O5,O4)
B(O1,O2)
B(O2,O3)
B(O3,O4)
B(O1,O5)
B(O3,O5)
B(O2,O4)
B(O3,F)
B(O5,O4)
B(O4,F)
Establishes
Threatens
Before constraint:
B(x,y)

STRIPS



Are the before constraints satisfiable?

YES:

$\longrightarrow$ **O1** $\longrightarrow$ **O2** $\longrightarrow$ **O3** $\longrightarrow$ **O5** $\longrightarrow$ **O4** $\longrightarrow$

Exercises: Artificial Intelligence

Planning & Logic: English to Logic

Problem & Solution

- *Not all students take both history and biology*

$$\neg\forall x\ [student(x) \Rightarrow takes(x,hist) \land takes(x,bio)]$$

$$\Leftrightarrow\ [A \Rightarrow B \Leftrightarrow \neg A \lor B]$$

$$\neg\forall x\ [\neg[student(x)] \lor [takes(x,hist) \land takes(x,bio)]]$$

$$\Leftrightarrow\ [\neg\forall x\ (F) \Leftrightarrow \exists x\ (\neg F)]$$

$$\exists x\ \neg[\neg[student(x)] \lor [takes(x,hist) \land takes(x,bio)]]$$

$$\Leftrightarrow\ [\neg(A \lor B) \Leftrightarrow \neg A \land \neg B],[\neg(A \land B) \Leftrightarrow \neg A \lor \neg B]$$

$$\exists x\ [student(x) \land [\neg takes(x,hist) \lor \neg takes(x,bio)]]$$

Problem & Solution

- *No person likes a smart vegetarian*

$$\forall x\ \forall y\ [person(x) \wedge vegetarian(y) \wedge smart(y) \Rightarrow \neg likes(x,y)]$$

$$\Leftrightarrow\ [A \Rightarrow B \Leftrightarrow \neg A \vee B]$$

$$\forall x\ \forall y\ [\neg[person(x) \wedge vegetarian(y) \wedge smart(y)] \vee \neg likes(x,y)]$$

$$\Leftrightarrow\ [\neg A \vee \neg B \Leftrightarrow \neg(A \wedge B)]$$

$$\forall x\ \forall y\ \neg[person(x) \wedge vegetarian(y) \wedge smart(y) \wedge likes(x,y)]$$

$$\Leftrightarrow\ [\forall x\ \neg(F) \Leftrightarrow \neg\exists x\ (F)]$$

$$\neg\ \exists x\ \exists y\ [person(x) \wedge vegetarian(y) \wedge smart(y) \wedge likes(x,y)]$$

Problem & Solution

- *There is a woman who likes all men who are not vegetarians.*

  $\exists x[\text{woman}(x) \wedge [\forall y\ [\text{man}(y) \wedge \neg \text{vegetarian}(y) \Rightarrow \text{likes}(x,y)]]]$

Problem & Solution

- *The best score in history was better than the best score in biology.*

  $\forall x\ \forall y\ [\text{bestscore}(\text{hist},x) \wedge \text{bestscore}(\text{bio},y) \Rightarrow \text{better}(x,y)]$

Problem & Solution

- *Every person who dislikes all vegetarians is smart.*

  $\forall x[\ person(x) \wedge [\forall y\ [vegetarian(y) \Rightarrow \neg likes(x,y)]] \Rightarrow smart(x)]$

Problem & Solution

- *There is a barber who shaves all men in town who do not shave themselves.*

$$\exists x\ [\text{barber}(x) \wedge [\forall y\ [\text{townsman}(y) \wedge \neg \text{shaves}\ (y,y) \Rightarrow \text{shaves}(x,y)]]]$$

$$\Leftrightarrow$$

$$\exists x\ [\text{barber}(x) \wedge [\forall y\ [\neg[\text{townsman}(y) \wedge \neg \text{shaves}\ (y,y)] \vee \text{shaves}(x,y)]]]$$

$$\Leftrightarrow$$

$$\exists x\ [\text{barber}(x) \wedge [\forall y\ \neg[\text{townsman}(y) \wedge \neg \text{shaves}\ (y,y) \wedge \neg \text{shaves}(x,y)]]]$$

$$\Leftrightarrow$$

$$\exists x\ [\text{barber}(x) \wedge\ [\neg\ \exists y\ [\text{townsman}(y) \wedge \neg \text{shaves}\ (y,y) \wedge \neg \text{shaves}(x,y)]]]$$

Problem & Solution

- *No person likes a professor unless the professor is smart.*

$$\forall x\ \forall y\ [person(x) \wedge professor(y) \Rightarrow [likes(x,y) \Rightarrow smart(y)]] \Leftrightarrow$$

$$\forall x\ \forall y\ [person(x) \wedge professor(y) \Rightarrow [\neg likes(x,y) \vee smart(y)]] \Leftrightarrow$$

$$\forall x\ \forall y\ [\neg[person(x) \wedge professor(y)] \vee [\neg likes(x,y) \vee smart(y)]] \Leftrightarrow$$

$$\forall x\ \forall y\ [\neg[person(x) \wedge professor(y)] \vee \neg[likes(x,y) \wedge \neg smart(y)]] \Leftrightarrow$$

$$\forall x\ \forall y\ \neg[person(x) \wedge professor(y) \wedge likes(x,y) \wedge \neg smart(y)] \Leftrightarrow$$

$$\neg\ \exists x\ \exists y\ [person(x) \wedge professor(y) \wedge likes(x,y) \wedge \neg smart(y)]$$

Problem & Solution

- *Only one person failed both history and biology.*

$$\exists! x \; student(x) \wedge failed(x,hist) \wedge failed(x,bio)$$

Note that: $\exists! x \; p(x) \Leftrightarrow \exists x \; p(x) \wedge [\forall y \; [p(y) \Rightarrow x=y]]$

Problem & Solution

- *Politicians can fool some of the people all the time, and they can fool all of the people some of the time, but they can't fool all the people all of the time.*

$$\forall x\ [\text{politician}(x) \Rightarrow [\exists y\ \text{people}(y) \wedge [\forall t\ \text{time}(t) \Rightarrow \text{fool}(x,y,t)]]]$$

$$\forall x\ [\text{politician}(x) \Rightarrow [\exists t\ \text{time}(t) \wedge [\forall y\ \text{people}(y) \Rightarrow \text{fool}(x,y,t)]]]$$

$$\forall x\ [\text{politician}(x) \Rightarrow \neg[\forall y\ \forall t\ [\text{people}(y) \wedge \text{time}(t)] \Rightarrow \text{fool}(x,y,t)]]$$

Exercises: Artificial Intelligence

Planning & Logic: And-Or-If

Problem & Solution

- *One more outburst like that and you are in contempt of court.*

$$\text{outburst} \Rightarrow \text{court}$$

$$\textbf{NOT}\text{: outburst} \wedge \text{court}$$

Problem & Solution

- *Either the Red Sox win or I'm out ten dollars.*

$$\text{redSoxWin} \Leftrightarrow \neg\text{outTenDollars}$$

NOT: $\text{redSoxWin} \vee \text{outTenDollars}$

Problem & Solution

- *Maybe I'll come to the party and maybe I won't.*

$$maybeComeToParty \lor \neg maybeComeToParty$$

NOT: $maybeComeToParty \land \neg maybeComeToParty$

Exercises: Artificial Intelligence

Planning & Logic: Weird Logic

Problem & Solution

- *I don't jump off the Empire State Building implies if I jump off the Empire State Building, then I float safely to the ground.*
  - *Translating the meaning of the sentence is not possible*

$$\neg \text{jumpESB} \Rightarrow [\text{jumpESB} \Rightarrow \text{floatTTGround}] \Leftrightarrow$$

$$\neg \text{jumpESB} \Rightarrow [\neg \text{jumpESB} \vee \text{floatTTGround}] \Leftrightarrow$$

$$\text{jumpESB} \vee \neg \text{jumpESB} \vee \text{floatTTGround}$$

Problem & Solution

- *It is not the case that if you attempt this exercise you will get an F. Therefore, you will attempt this exercise.*
  - *Translating the meaning of the sentence is not possible*

$$\neg[\text{attempt} \Rightarrow \text{getF}] \Rightarrow \text{attempt} \Leftrightarrow$$

$$\neg[\neg\text{attempt} \vee \text{getF}] \Rightarrow \text{attempt} \Leftrightarrow$$

$$\neg\text{attempt} \vee \text{getF} \vee \text{attempt}$$

Exercises: Artificial Intelligence

Automated Reasoning: Good to walk

Solution

- *We assume that it is not good to walk:*
  - **false ← walk**
- *Given:*
  - raining ∨ snowing ∨ dry (← true)
  - warm (← true)
  - false ← raining
  - false ← snowing
  - walk ← nice
  - nice ← dry ∧ warm

Solution

false ← walk

walk ← nice

false ← nice

nice ← dry ∧ warm

false ← dry ∧ warm

warm ← true

false ← dry

raining ∨ snowing ∨ dry ← true

raining ∨ snowing ← true

false ← raining

snowing ← true

false ← snowing

$\boxed{\text{false} \leftarrow \text{true}}$

Exercises: Artificial Intelligence

Automated Reasoning: MGU

Solution

MGU: $\{x/f(A), w/f(A), y/A\}$
Result: $p(f(A),f(A),g(z,A))$

- *What is the m.g.u. of: $p(f(y),w,g(z,y)) = p(x,x,g(z,A))$*
  - *Init: $p(f(y),w,g(z,y)) = p(x,x,g(z,A))$*
  - *Case 5: $f(y) = x$, $w = x$, $g(z,y) = g(z,A)$*
  - *Case 1: $x = f(y)$, $w = x$, $g(z,y) = g(z,A)$*
  - *Case 4: $x = f(y)$, $w = f(y)$, $g(z,y) = g(z,A)$*
  - *Case 5: $x = f(y)$, $w = f(y)$, $z = z$, $y = A$*
  - *Case 2: $x = f(y)$, $w = f(y)$, $y = A$*
  - ***Case 4:** $x = f(A)$, $w = f(A)$, $y = A$*

Solution

- *What is the m.g.u. of: $p(A,x,f(g(y))) = p(z,f(z),f(A))$*
  - *Init: $p(A,x,f(g(y))) = p(z,f(z),f(A))$*
  - *Case 5: $A = z$, $x = f(z)$, $f(g(y)) = f(A)$*
  - *Case 1: $z = A$, $x = f(z)$, $f(g(y)) = f(A)$*
  - *Case 4: $z = A$, $x = f(A)$, $f(g(y)) = f(A)$*
  - *Case 5: $z = A$, $x = f(A)$, $g(y) = A$*
  - ***Case 5:** stop := true*

Solution

- *What is the m.g.u. of: $q(x,x) = q(y,f(y))$*
  - *Init: $q(x,x) = q(y,f(y))$*
  - *Case 5: $x = y, x = f(y)$*
  - *Case 4: $x = y, y = f(y)$*
  - ***Case 3:** stop := true*

Solution

MGU: $\{x/g(f(a),f(a)),\ u/f(a),\ v/f(a)\}$
Result: $f(g(f(a),f(a)),g(f(a),\ f(a)))$

- *What is the m.g.u. of:* $f(x,g(f(a),u)) = f(g(u,v),x)$
  - *Init:* $f(x,g(f(a),u)) = f(g(u,v),x)$
  - *Case 5:* $x = g(u,v),\ g(f(a),u) = x$
  - *Case 4:* $x = g(u,v),\ g(f(a),u) = g(u,v)$
  - *Case 5:* $x = g(u,v),\ f(a) = u,\ u = v$
  - *Case 1:* $x = g(u,v),\ u = f(a),\ u = v$
  - *Case 4:* $x = g(f(a),v),\ u = f(a),\ f(a) = v$
  - *Case 1:* $x = g(f(a),v),\ u = f(a),\ v = f(a)$
  - ***Case 4:*** $x = g(f(a),f(a)),\ u = f(a),\ v = f(a)$

Exercises: Artificial Intelligence

Automated Reasoning: Resolution

Solution

- *Assumption: Peter has no mother-in-law*
  - false $\leftarrow$ mother-in-law(x,Peter)
- *Given:*
  - mother-in-law(x,y) $\leftarrow$ mother(x,z) $\wedge$ married(z,y)
  - mother(x,y) $\leftarrow$ female(x) $\wedge$ parent(x,y)
  - female(An) ($\leftarrow$ true)
  - parent(An,Maria) ($\leftarrow$ true)
  - married(Maria,Peter) ($\leftarrow$ true)

Solution

- false $\leftarrow$ mother-in-law(x,Peter)
  - mother-in-law(x',y') $\leftarrow$ mother(x',z') $\wedge$ married(z',y')
  - {x'/x, y'/Peter}
- false $\leftarrow$ mother(x,z') $\wedge$ married(z',Peter)

Solution

- false $\leftarrow$ mother-in-law(x,Peter)
- false $\leftarrow$ mother(x,z') $\wedge$ married(z',Peter)
  - mother(x',y') $\leftarrow$ female(x') $\wedge$ parent(x',y')
  - {x'/x, y'/z'}
- false $\leftarrow$ female(x) $\wedge$ parent(x,z') $\wedge$ married(z',Peter)

Solution

- false $\leftarrow$ mother-in-law(x,Peter)
- false $\leftarrow$ mother(x,z') $\wedge$ married(z',Peter)
- false $\leftarrow$ female(x) $\wedge$ parent(x,z') $\wedge$ married(z',Peter)
  - female(An)
  - {x/An}
- false $\leftarrow$ parent(An,z') $\wedge$ married(z',Peter)

Solution

- false $\leftarrow$ mother-in-law(x,Peter)
- false $\leftarrow$ mother(x,z') $\wedge$ married(z',Peter)
- false $\leftarrow$ female(x) $\wedge$ parent(x,z') $\wedge$ married(z',Peter)
- false $\leftarrow$ parent(An,z') $\wedge$ married(z',Peter)
  - parent(An,Maria)
  - {z'/Maria}
- false $\leftarrow$ married(Maria,Peter)

Solution

$\boxed{\{x/An\}}$

- false ← mother-in-law(x,Peter)
- false ← mother(x,z’) ∧ married(z’,Peter)
- false ← female(x) ∧ parent(x,z’) ∧ married(z’,Peter)
- false ← parent(An,z’) ∧ married(z’,Peter)
- false ← married(Maria,Peter)
  – married(Maria,Peter)
- false ← true (□)

Solution

- *Assumption: "There is no valid colouring"*
  - *false* ← nb(b,g),nb(g,n),nb(n,b)
- *Given:*
  - c(R) (← true)
  - c(G) (← true)
  - c(B) (← true)
  - nb(x,y) ← c(x), c(y), diff(x,y)
    - diff/2 succeeds when arguments cannot be unified

Solution

- false ← nb(b,g) ∧ nb(g,n) ∧ nb(n,b)
  - nb(x’,y’) ← c(x’), c(y’), diff(x’,y’)
  - {x’/b,y’/g}
- false ← c(b) ∧ c(g) ∧ diff(b,g) ∧ nb(g,n) ∧ nb(n,b)

Solution

- false $\leftarrow$ nb(b,g) $\wedge$ nb(g,n) $\wedge$ nb(n,b)
- false $\leftarrow$ c(b) $\wedge$ c(g) $\wedge$ diff(b,g) $\wedge$ nb(g,n) $\wedge$ nb(n,b)
  - nb(x',y') $\leftarrow$ c(x'), c(y'), diff(x',y')
  - {x'/g,y'/n}
- false $\leftarrow$ c(b) $\wedge$ c(g) $\wedge$ diff(b,g) $\wedge$ c(n) $\wedge$ diff(g,n) $\wedge$ nb(n,b)

Solution

- false $\leftarrow$ nb(b,g) $\wedge$ nb(g,n) $\wedge$ nb(n,b)
- false $\leftarrow$ c(b) $\wedge$ c(g) $\wedge$ diff(b,g) $\wedge$ nb(g,n) $\wedge$ nb(n,b)
- false $\leftarrow$ c(b) $\wedge$ c(g) $\wedge$ diff(b,g) $\wedge$ c(n) $\wedge$ diff(g,n) $\wedge$ nb(n,b)
  - nb(x',y') $\leftarrow$ c(x'), c(y'), diff(x',y')
  - {x'/n,y'/b}
- false $\leftarrow$ c(b) $\wedge$ c(g) $\wedge$ diff(b,g) $\wedge$ c(n) $\wedge$ diff(g,n) $\wedge$ diff(n,b)

Solution

- false $\leftarrow$ nb(b,g) $\wedge$ nb(g,n) $\wedge$ nb(n,b)
- false $\leftarrow$ c(b) $\wedge$ c(g) $\wedge$ diff(b,g) $\wedge$ nb(g,n) $\wedge$ nb(n,b)
- false $\leftarrow$ c(b) $\wedge$ c(g) $\wedge$ diff(b,g) $\wedge$ c(n) $\wedge$ diff(g,n) $\wedge$ nb(n,b)
- false $\leftarrow$ c(b) $\wedge$ c(g) $\wedge$ diff(b,g) $\wedge$ c(n) $\wedge$ diff(g,n) $\wedge$ diff(n,b)
  - c(R)
  - {b/R}
- false $\leftarrow$ c(g) $\wedge$ diff(R,g) $\wedge$ c(n) $\wedge$ diff(g,n) $\wedge$ diff(n,R)

Solution

- false $\leftarrow$ nb(b,g) $\wedge$ nb(g,n) $\wedge$ nb(n,b)
- false $\leftarrow$ c(b) $\wedge$ c(g) $\wedge$ diff(b,g) $\wedge$ nb(g,n) $\wedge$ nb(n,b)
- false $\leftarrow$ c(b) $\wedge$ c(g) $\wedge$ diff(b,g) $\wedge$ c(n) $\wedge$ diff(g,n) $\wedge$ nb(n,b)
- false $\leftarrow$ c(b) $\wedge$ c(g) $\wedge$ diff(b,g) $\wedge$ c(n) $\wedge$ diff(g,n) $\wedge$ diff(n,b)
- false $\leftarrow$ c(g) $\wedge$ diff(R,g) $\wedge$ c(n) $\wedge$ diff(g,n) $\wedge$ diff(n,R)
  - c(G)
  - {g/G}
- false $\leftarrow$ diff(R,G) $\wedge$ c(n) $\wedge$ diff(G,n) $\wedge$ diff(n,R)

Solution

- false $\leftarrow$ nb(b,g) $\wedge$ nb(g,n) $\wedge$ nb(n,b)
- false $\leftarrow$ c(b) $\wedge$ c(g) $\wedge$ diff(b,g) $\wedge$ nb(g,n) $\wedge$ nb(n,b)
- false $\leftarrow$ c(b) $\wedge$ c(g) $\wedge$ diff(b,g) $\wedge$ c(n) $\wedge$ diff(g,n) $\wedge$ nb(n,b)
- false $\leftarrow$ c(b) $\wedge$ c(g) $\wedge$ diff(b,g) $\wedge$ c(n) $\wedge$ diff(g,n) $\wedge$ diff(n,b)
- false $\leftarrow$ c(g) $\wedge$ diff(R,g) $\wedge$ c(n) $\wedge$ diff(g,n) $\wedge$ diff(n,R)
- false $\leftarrow$ diff(R,G) $\wedge$ c(n) $\wedge$ diff(G,n) $\wedge$ diff(n,R)
  - c(B)
  - {n/B}
- false $\leftarrow$ diff(R,G) $\wedge$ diff(G,B) $\wedge$ diff(B,R)

Solution

$\boxed{\{b/R, g/G, n/B\}}$

- false ← nb(b,g) ∧ nb(g,n) ∧ nb(n,b)
- false ← c(b) ∧ c(g) ∧ diff(b,g) ∧ nb(g,n) ∧ nb(n,b)
- false ← c(b) ∧ c(g) ∧ diff(b,g) ∧ c(n) ∧ diff(g,n) ∧ nb(n,b)
- false ← c(b) ∧ c(g) ∧ diff(b,g) ∧ c(n) ∧ diff(g,n) ∧ diff(n,b)
- false ← c(g) ∧ diff(R,g) ∧ c(n) ∧ diff(g,n) ∧ diff(n,R)
- false ← diff(R,G) ∧ c(n) ∧ diff(G,n) ∧ diff(n,R)
- false ← diff(R,G) ∧ diff(G,B) ∧ diff(B,R)
  - Built-in diff/2: succeeds for different arguments
- false ← true (□)

Alternative solution

$\{\underline{\mathbf{b/B}}, g/G, \underline{\mathbf{n/R}}\}$

- false ← nb(b,g) ∧ nb(g,n) ∧ nb(n,b)
- false ← c(b) ∧ c(g) ∧ diff(b,g) ∧ nb(g,n) ∧ nb(n,b)
- false ← c(b) ∧ c(g) ∧ diff(b,g) ∧ c(n) ∧ diff(g,n) ∧ nb(n,b)
- false ← c(b) ∧ c(g) ∧ diff(b,g) ∧ c(n) ∧ diff(g,n) ∧ diff(n,b)
- false ← c(g) ∧ diff(**B**,g) ∧ c(n) ∧ diff(g,n) ∧ diff(n,**B**)
- false ← diff(**B**,G) ∧ c(n) ∧ diff(G,n) ∧ diff(n,**B**)
- false ← diff(**B**,G) ∧ diff(G,**R**) ∧ diff(**R**,**B**)
  - Built-in diff/2: succeeds for different arguments
- false ← true (□)

Or consistency = Continue search

- false ← nb(b,g) ∧ nb(g,n) ∧ nb(n,b)
- false ← c(b) ∧ c(g) ∧ diff(b,g) ∧ nb(g,n) ∧ nb(n,b)
- false ← c(b) ∧ c(g) ∧ diff(b,g) ∧ c(n) ∧ diff(g,n) ∧ nb(n,b)
- false ← c(b) ∧ c(g) ∧ diff(b,g) ∧ c(n) ∧ diff(g,n) ∧ diff(n,b)
- false ← c(g) ∧ diff(R,g) ∧ c(n) ∧ diff(g,n) ∧ diff(n,R)
- false ← diff(R,**<u>R</u>**) ∧ c(n) ∧ diff(**<u>R</u>**,n) ∧ diff(n,R)
- false ← diff(R,**<u>R</u>**) ∧ diff(**<u>R</u>**,B) ∧ diff(B,R)
  - diff(R,R) is false
- false ← false

Exercises: Artificial Intelligence

Automated Reasoning: Predicate Resolution

Solution

- Formula in implicative normal form:
  - $\forall x\ p(x) \vee \neg r(f(x))$
    - $p(x) \leftarrow r(f(x))$
  - $\forall x\ \forall y\ r(f(x)) \vee r(f(f(y)))$
    - $r(f(x)) \vee r(f(f(y)))$ $(\leftarrow \text{true})$
- Assumption

$\neg\ [\forall x\ \exists y\ p(f(x)) \wedge r(y)] \Leftrightarrow \exists x\ \forall y\ \neg[p(f(x)) \wedge r(y)] \Leftrightarrow$

$\forall y\ \neg[p(f(A)) \wedge r(y)] \Leftrightarrow \text{false} \leftarrow p(f(A)) \wedge r(y)$

Solution

- false $\leftarrow$ p(f(A)) $\wedge$ r(y)
  - p(x’) $\leftarrow$ r(f(x’))
  - {x’/f(A)}
- false $\leftarrow$ r(f(f(A))) $\wedge$ r(y)

Solution

- false $\leftarrow$ p(f(A)) $\wedge$ r(y)
- false $\leftarrow$ r(f(f(A))) $\wedge$ r(y)
  - Factoring: mgu(r(f(f(A))) = r(y)) = {y/f(f(A))}
- false $\leftarrow$ r(f(f(A))) $\wedge$ r(f(f(A)))

Solution

$\{y/f(f(A))\}$

- false ← p(f(A)) ∧ r(y)
- false ← r(f(f(A))) ∧ r(y)
- false ← r(f(f(A))) ∧ r(f(f(A)))
  - r(f(x’)) ∨ r(f(f(y’))) (← true)
    - Factoring: mgu(r(f(x’)) = r(f(f(y’)))) = {x’/f(y’)}
  - r(f(f(y’))) (← true)
  - {y’/A}
- false ← true (□)

Exercises: Artificial Intelligence

Automated Reasoning:
Movable Objects

Solution: Movable Objects

- ***English to logic***
- ***Logic to implicative normal form***
  - *Model*
  - *Assumption to prove*
- ***Apply resolution***
  - *Derive inconsistency:*
  - *Model + negated assumption*

Solution: Model to logic

- If all movable objects are blue, then all non-movable objects are green.
  - **$(\forall x\ mov(x) \rightarrow blue(x)) \rightarrow (\forall y\ \neg mov\ (y) \rightarrow green(y))$**
- If there exists a non-movable object, then all movable objects are blue.
  - **$(\exists x\ \neg mov(x)) \rightarrow (\forall y\ mov\ (y) \rightarrow blue(y))$**
- D is a non-movable object.
  - **$\neg mov(D)$**

Solution: Implicative normal form

- **($\forall$x mov(x) $\rightarrow$ blue(x)) $\rightarrow$ ($\forall$y $\neg$mov (y) $\rightarrow$ green(y))**
  - **mov(A) $\vee$ mov(y) $\vee$ green(y) ($\leftarrow$ true)**
    mov(y) $\vee$ green(y) $\leftarrow$ blue(A)
- **($\exists$x $\neg$mov(x)) $\rightarrow$ ($\forall$y mov (y) $\rightarrow$ blue(y))**
  - **mov (x) $\vee$ blue(y) $\leftarrow$ mov(y)**
- **$\neg$mov(D)**
  - **false $\leftarrow$ mov(D)**
- Negated assumption: **$\neg\exists$x green(x) $\leftrightarrow$ $\forall$x $\neg$green(x)**
  - **false $\leftarrow$ green(x)**

Solution: Implicative normal form

- **Prove** using resolution:
  - Assumption: **false ← green(x)**
- **Model:**
  - mov(A) ∨ mov(y) ∨ green(y) (← true)
  - mov(y) ∨ green(y) ← blue(A)
  - mov (x) ∨ blue(y) ← mov(y)
  - false ← mov(D)

Solution: Resolution

mov (x) ∨ blue(y) ← mov(y)   false ← mov(D)

{x/D}

blue(y) ← mov(y)   mov(y1) ∨ green(y1) ← blue(A)

{y/A}

mov(y1) ∨ green(y1) ← mov(A)   mov(A) ∨ mov(y2) ∨ green(y2)

{}

mov(y1) ∨ green(y1) ∨ mov(y2) ∨ green(y2) ← true

Factoring:{y2/y1}

mov(y1) ∨ green(y1) ← true   false ← mov(D)

{y1/D}

green(D) ← true   false ← green(x)

{x/D}

false ← true

Exercises: Artificial Intelligence

Automated Reasoning: Politicians

Problem: Politicians

- ***Given:***
  - If a poor politician exists, then all politicians are male.
  - If people are friends with a politician, then this politician is poor and female.
  - Lazy people have no friends.
  - People are either male or female, but not both.
  - If Joel is not lazy, then he is a politician.
- ***Proof by resolution:***
  - There exists no person who is a friend of Joel.

Solution: English to logic

- $(\exists x\ pol(x) \wedge poor(x)) \rightarrow (\forall y\ pol(y) \rightarrow male(y))$.
- $\forall x\ (pol(x) \wedge (\exists y\ fr(y,x))) \rightarrow poor(x) \wedge fem(x)$.
- $\forall x\ lazy(x) \rightarrow (\neg(\exists y\ fr(y,x)))$.
- $\forall x\ (male(x) \vee fem(x)) \wedge (\neg(male(x) \wedge fem(x)))$.
- $\neg lazy(Joel) \rightarrow pol(Joel)$.

Solution: Implicative normal form

- $male(y) \leftarrow pol(x) \wedge poor(x) \wedge pol(y)$
- $poor(x) \leftarrow pol(x) \wedge fr(y,x)$
- $fem(x) \leftarrow pol(x) \wedge fr(y,x)$
- $false \leftarrow lazy(x) \wedge fr(y,x)$
- $male(x) \vee fem(x)$
- $false \leftarrow male(x) \wedge fem(x)$
- $lazy(Joel) \vee pol(Joel)$

Solution: Implicative normal form

- Prove:
  - There exists no person who is a friend of Joel
    - $\neg\exists x\ fr(x,Joel) \leftrightarrow \forall x\ \neg fr(x,Joel)$
- Negate assumption:
  - There exists a person who is a friend of Joel
    - $\exists x\ fr(x,Joel)$
  - Call the friend S
    - $fr(S,Joel)$

Solution: Implicative normal form

- $male(y) \leftarrow pol(x) \wedge poor(x) \wedge pol(y)$
- $poor(x) \leftarrow pol(x) \wedge fr(y,x)$
- $fem(x) \leftarrow pol(x) \wedge fr(y,x)$
- $false \leftarrow lazy(x) \wedge fr(y,x)$
- $male(x) \vee fem(x)$
- $false \leftarrow male(x) \wedge fem(x)$
- $lazy(Joel) \vee pol(Joel)$
- **fr(S,Joel)**

Solution: Apply Resolution

- **male(y1) ← pol(x1) ∧ <u>poor(x1)</u> ∧ pol(y1)**
  - <u>poor(x2)</u> ← pol(x2) ∧ fr(y2,x2)
  - RESOLUTION: **{x2/x1}**
- **male(y1) ← <u>pol(x1)</u> ∧ <u>pol(y1)</u> ∧ fr(y2,x1)**
  - FACTORING: **{y1/x1}**
- **male(x1) ← pol(x1) ∧ fr(y2,x1)**
  - *'Politicians who have friends must be male'*

Solution: Apply Resolution

- **<u>male(x1)</u>** ← **pol(x1)** ∧ **fr(y2,x1)**
  - false ← <u>male(x3)</u>∧fem(x3)
  - RESOLUTION: **{x3/x1}**
- **false** ← **pol(x1)** ∧ **fr(y2,x1)** ∧ **fem(x1)**
  - *‘Politicians who have friends cannot be female’*

Solution: Apply Resolution

- **false ← pol(x1) ∧ fr(y2,x1) ∧ <u>fem(x1)</u>**
  - <u>fem(x4)</u> ← pol(x4) ∧ fr(y4,x4)
  - RESOLUTION: **{x4/x1}**
- **false ← <u>pol(x1)</u> ∧ fr(y2,x1) ∧ <u>pol(x1)</u> ∧ fr(y4,x1)**
  - FACTORING: **{}**
- **false ← pol(x1) ∧ <u>fr(y2,x1)</u> ∧ <u>fr(y4,x1)</u>**
  - FACTORING: **{y4/y2}**
- **false ← pol(x1) ∧ fr(y2,x1)**
  - *‘Politicians do not have friends’*

Solution: Apply Resolution

- **false ← <u>pol(x1)</u> ∧ fr(y2,x1)**
  - lazy(Joel) ∨ <u>pol(Joel)</u>
  - RESOLUTION: **{x1/Joel}**
- **lazy(Joel) ← fr(y2,Joel)**
  - *'If Joel has friend, then he must be lazy'*

Solution: Apply Resolution

- **<u>lazy(Joel)</u> ← fr(y2,Joel)**
  - false ← <u>lazy(x5)</u> ∧ fr(y5,x5)
  - RESOLUTION: **{x5/Joel}**
- **false ← <u>fr(y2,Joel)</u> ∧ <u>fr(y5,Joel)</u>**
  - FACTORING: **{y5/y2}**
- **false ← fr(y2,Joel)**
  - *'Joel does not have any friends'*

Solution: Apply Resolution

- **false ← <u>fr(y2,Joel)</u>**
  - <u>fr(S,Joel)</u>
  - RESOLUTION: **{y2/S}**
- **false ← true**