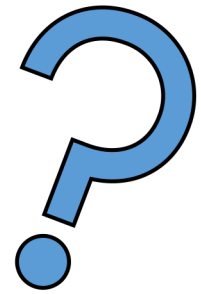
COMP9414 Tutorial

Week 3

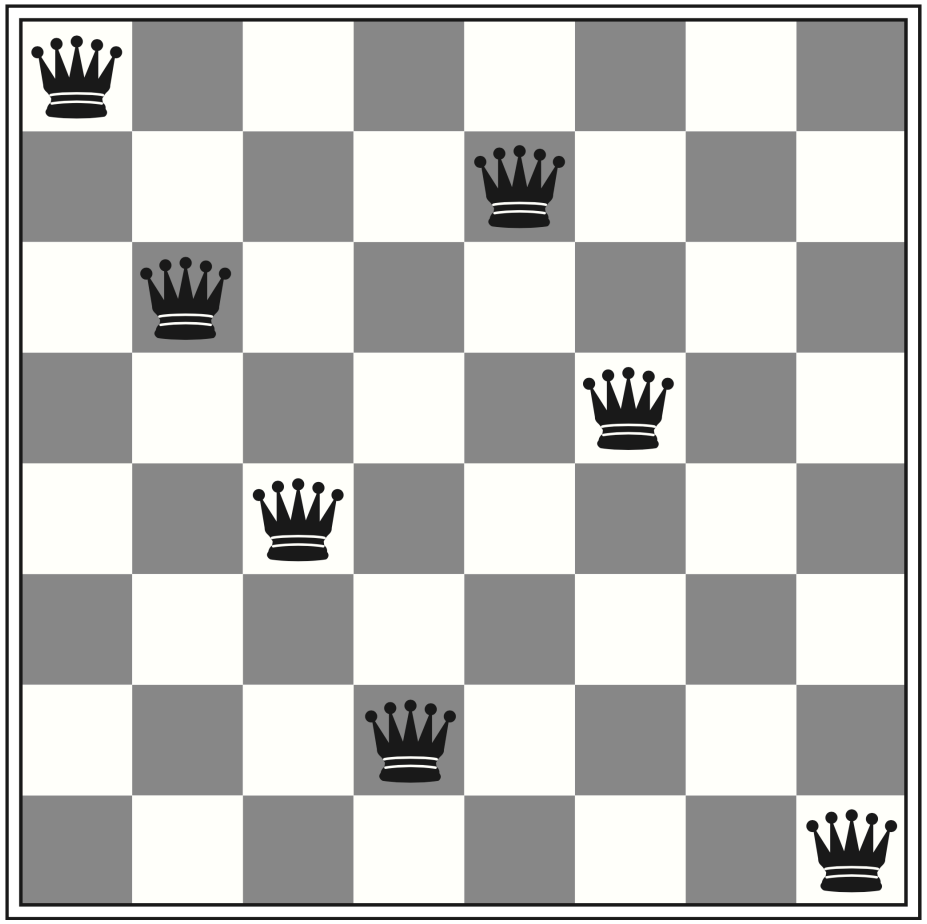
News

- Assignment 1 has been released
  - Consultation session was recorded
  - Attend the other consultations if you have further questions

- Can submit the assignment unlimited times
  - Only the final submission will be considered
  - Submit early and often
  - Become familiar with the submission procedure

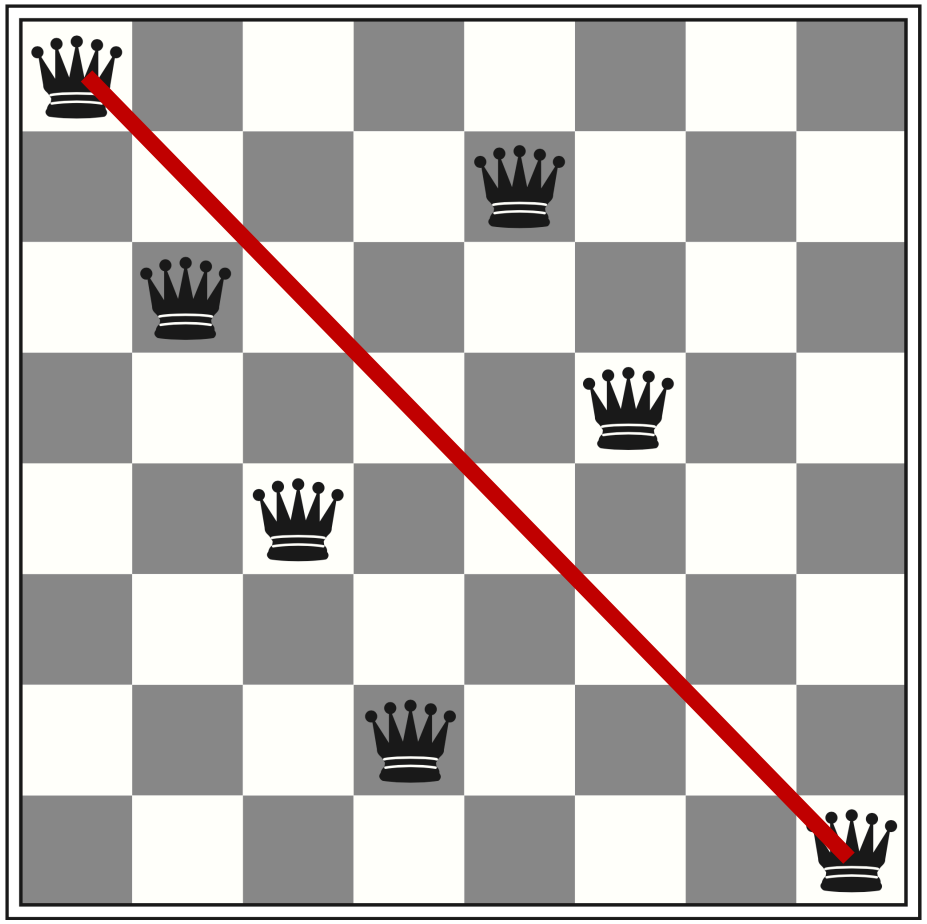


Question 1 – 8-Queens Problem



- No queen can see another queen
- One queen per row, column and diagonal

Question 1 – 8-Queens Problem



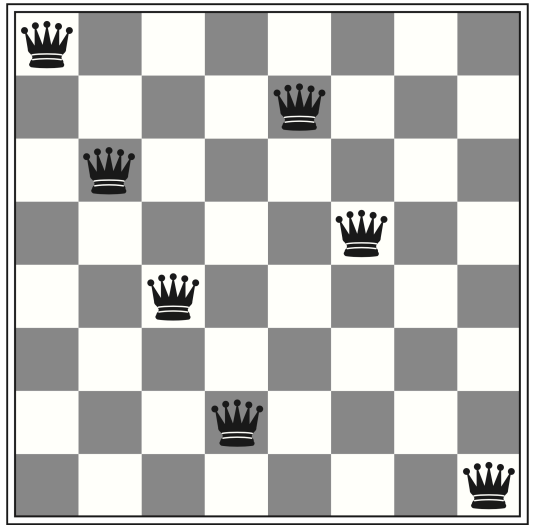
- No queen can see another queen
- One queen per row, column and diagonal

- Invalid solution

Question 1 – 8-Queens Problem

- 8 variables
  - One for each queen
  - Queen number is their row

- Each variable has the domain {1, …, 8}
  - Value is their assigned column



Queen	1	2	3	4	5	6	7	8
Q_1								
Q_2								
…								
Q_8								

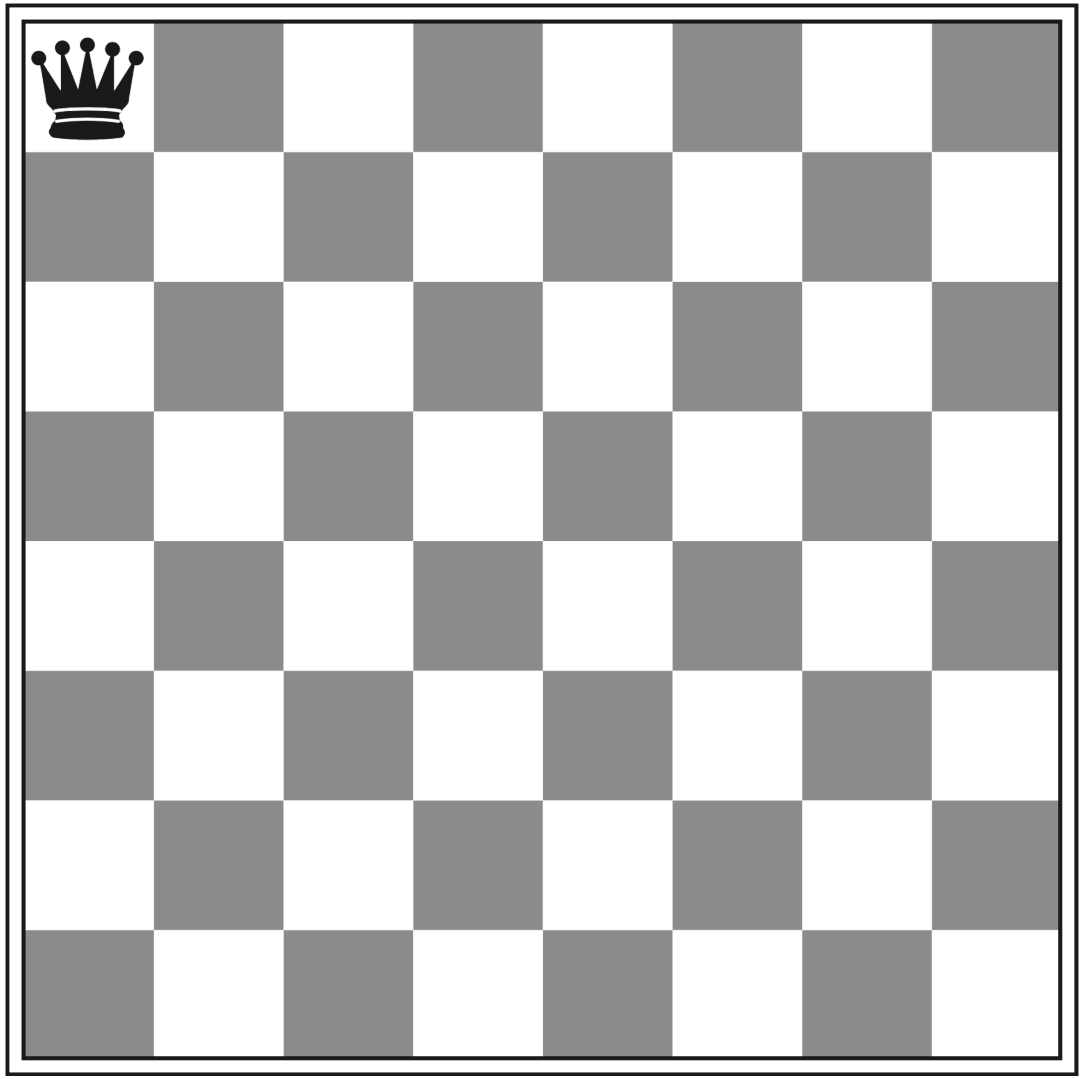
Question 1 – Forward-Checking

- Keep track of legal values for all variables
- Assigns a value to a variable
  - Remove all illegal values in remaining variables
- Ensure that at every step, only legal values remain
  - Terminate when a variable has no legal values

Question 1 – Forward-Checking

Queen	1	2	3	4	5	6	7	8
Q_1								
Q_2								
Q_3								
Q_4								
Q_5								
Q_6								
Q_7								
Q_8								

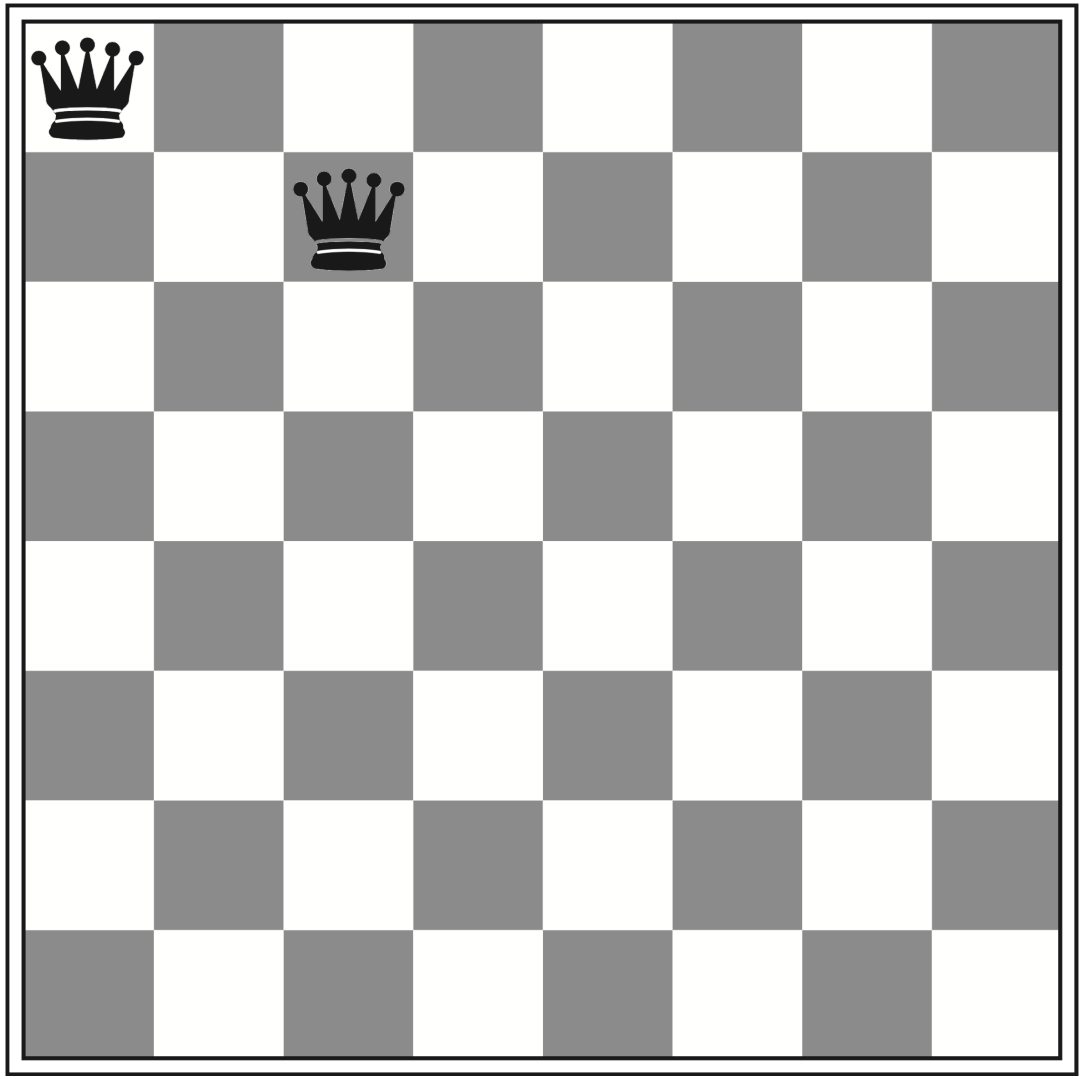
Question 1 – Forward-Checking



Question 1 – Forward-Checking

Queen	1	2	3	4	5	6	7	8
Q_1								
Q_2								
Q_3								
Q_4								
Q_5								
Q_6								
Q_7								
Q_8								

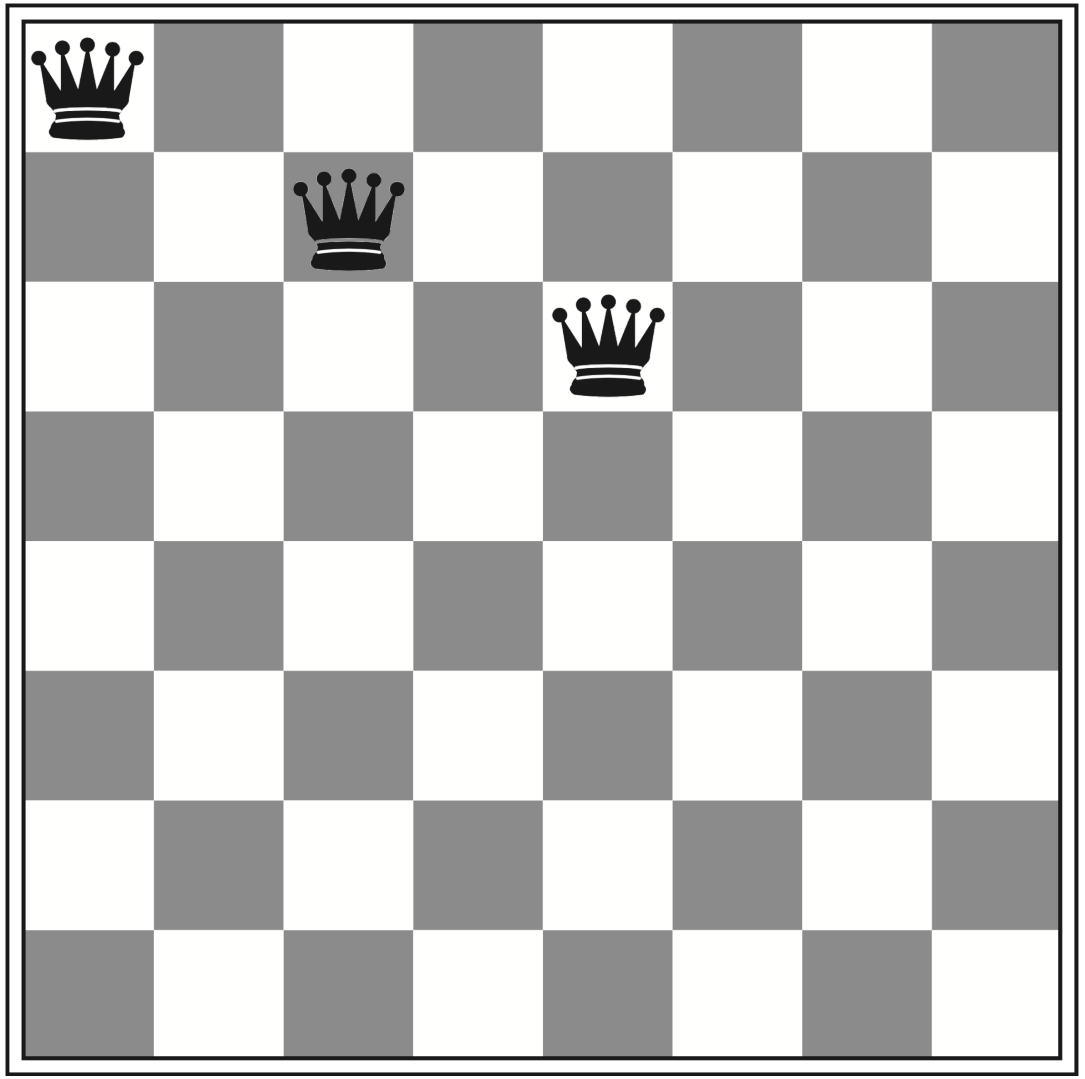
Question 1 – Forward-Checking



Question 1 – Forward-Checking

Queen	1	2	3	4	5	6	7	8
Q_1								
Q_2								
Q_3								
Q_4								
Q_5								
Q_6								
Q_7								
Q_8								

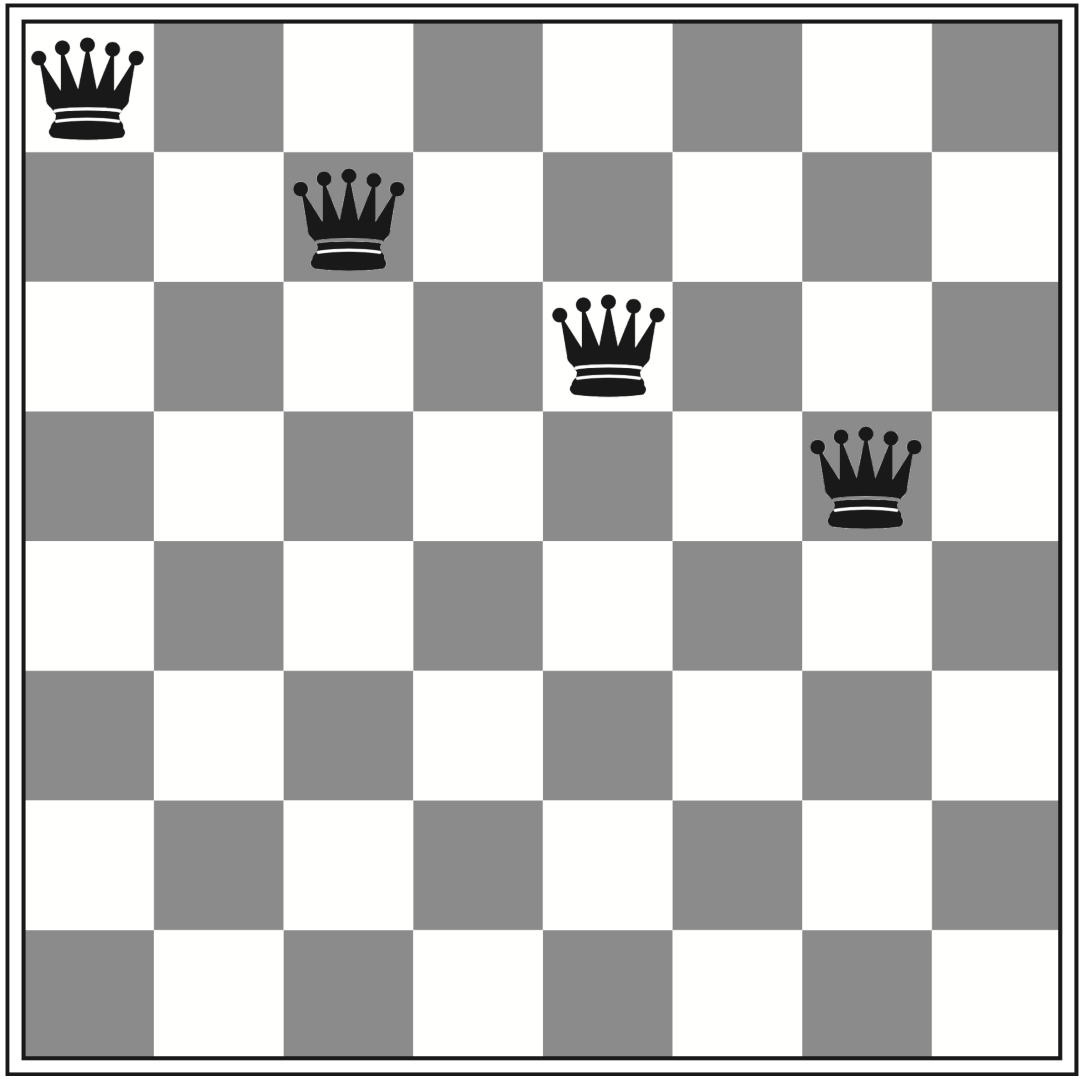
Question 1 – Forward-Checking



Question 1 – Forward-Checking

Queen	1	2	3	4	5	6	7	8
Q_1								
Q_2								
Q_3								
Q_4								
Q_5								
Q_6								
Q_7								
Q_8								

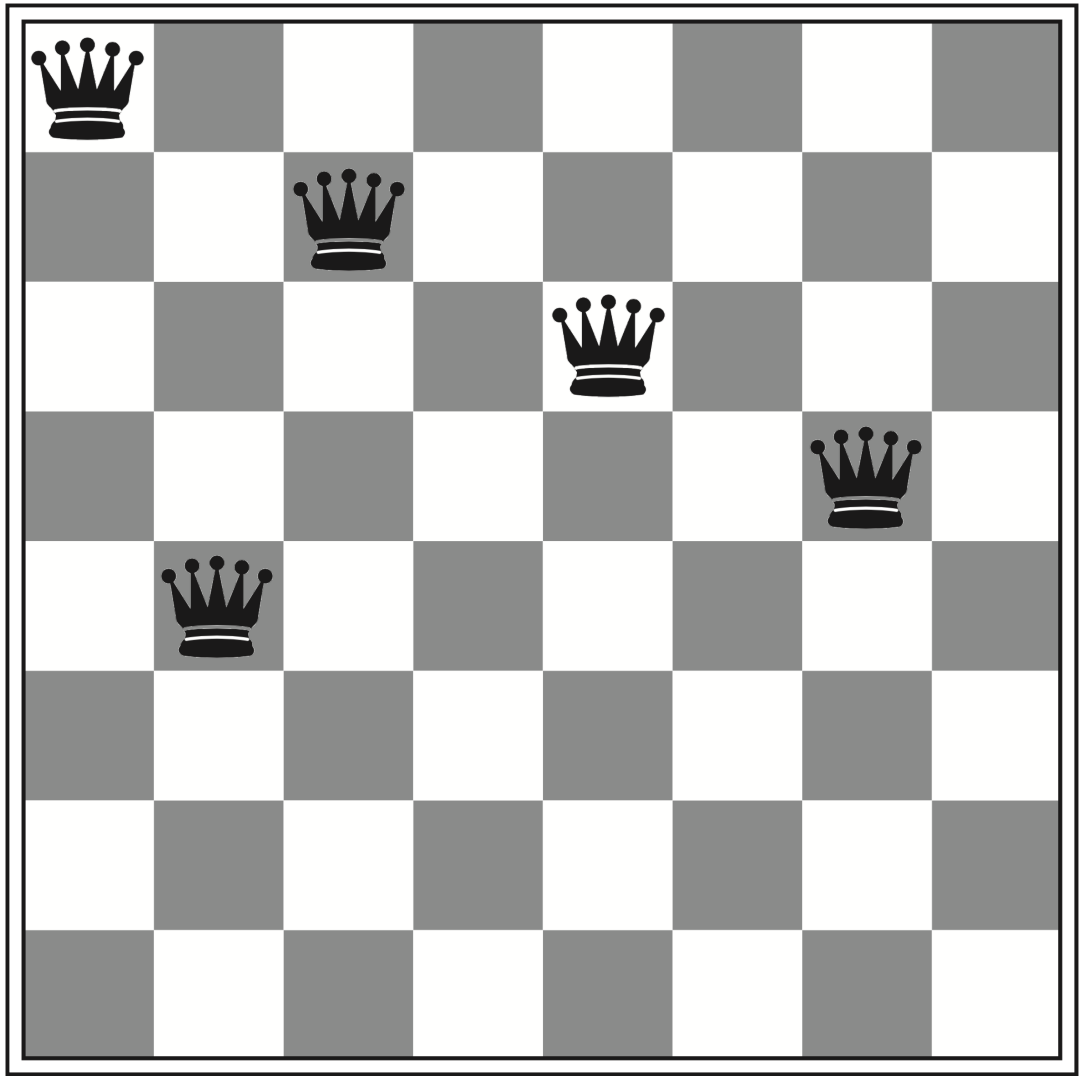
Question 1 – Forward-Checking



Question 1 – Forward-Checking

Queen	1	2	3	4	5	6	7	8
Q_1								
Q_2								
Q_3								
Q_4								
Q_5								
Q_6								
Q_7								
Q_8								

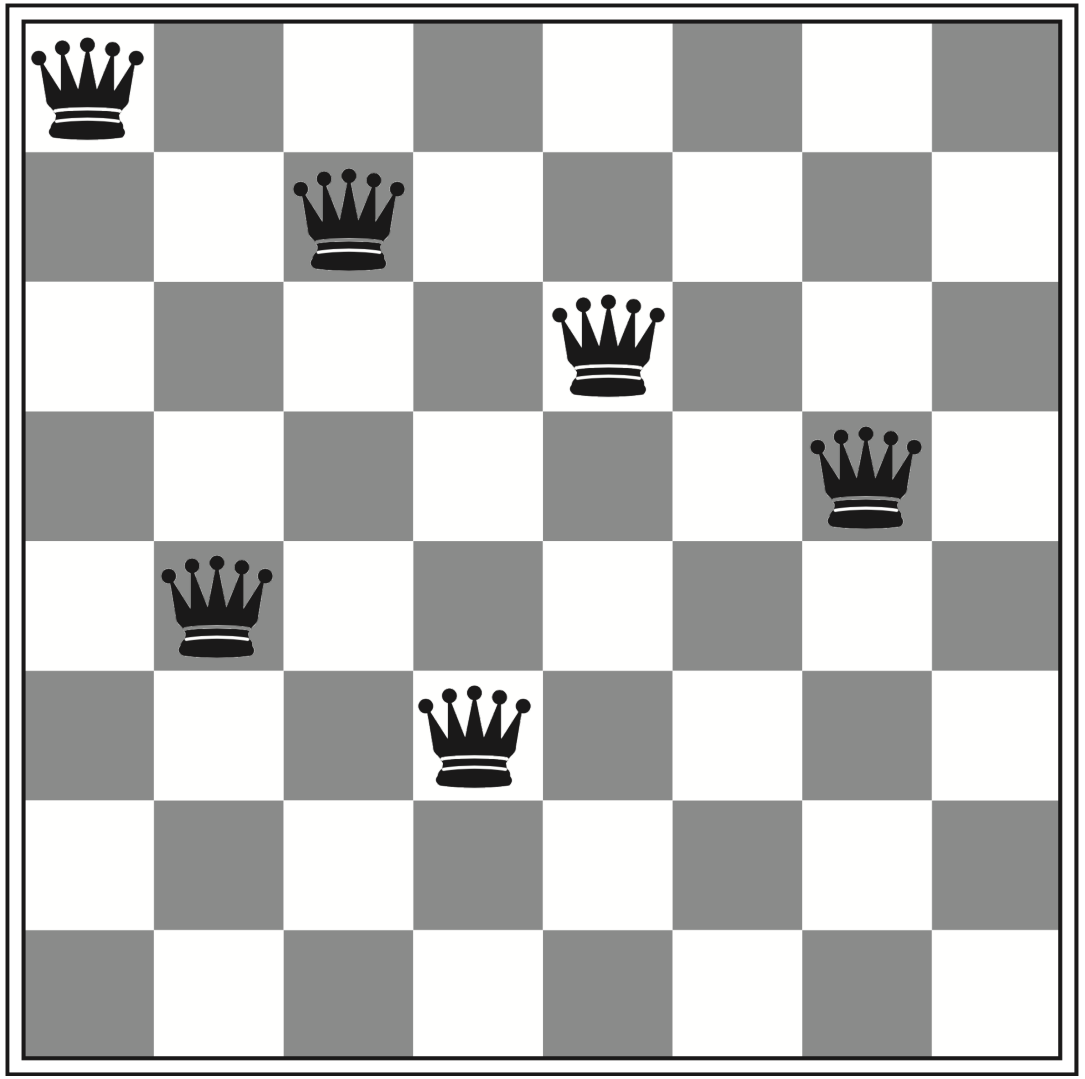
Question 1 – Forward-Checking



Question 1 – Forward-Checking

Queen	1	2	3	4	5	6	7	8
Q_1								
Q_2								
Q_3								
Q_4								
Q_5								
Q_6								
Q_7								
Q_8								

Question 1 – Forward-Checking



Question 1 – Forward-Checking

Queen	1	2	3	4	5	6	7	8
Q_1								
Q_2								
Q_3								
Q_4								
Q_5								
Q_6								
Q_7								
Q_8								

- Failure: Constraints could not be satisfied for Queen 8

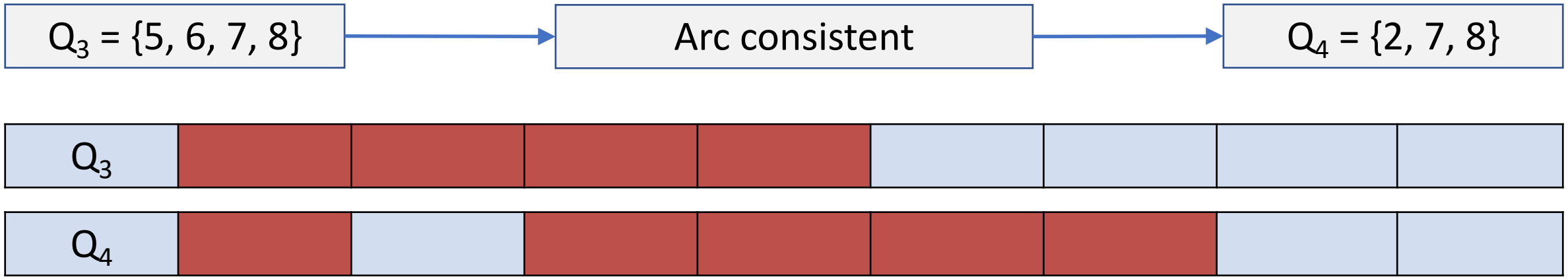
Question 1 – Arc Consistency

- Compare the domains of two variables
  - Each value in one domain should have a valid value in the other

- Check each variable to ensure consistency with all others prior to assignment

Queen	1	2	3	4	5	6	7	8
Q_1								
Q_2								
Q_3								
Q_4								
…								

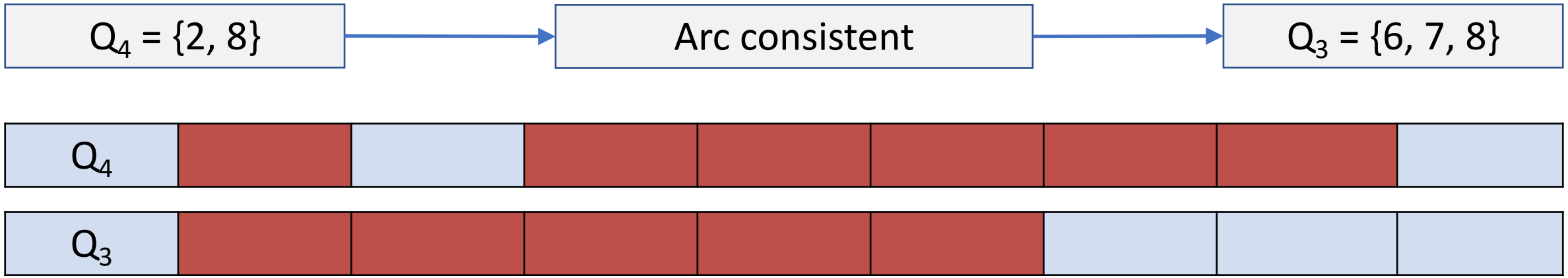
Question 1 – Arc Consistency



$Q_3 = 5$	$Q_4 = \{2, 7, 8\}$
$Q_3 = 6$	$Q_4 = \{2, 8\}$
$Q_3 = 7$	$Q_4 = 2$
$Q_3 = 8$	$Q_4 = 2$

- Every value in Q_3 has a valid value in Q_4
- Q_3 is arc-consistent towards Q_4

Question 1 – Arc Consistency



$Q_4 = 2$	$Q_3 = \{5, 6, 7, 8\}$

$Q_4 = 8$	$Q_3 = \{5, 6\}$

- Every value in Q_4 has a valid value in Q_3
- Q_4 is arc-consistent towards Q_3

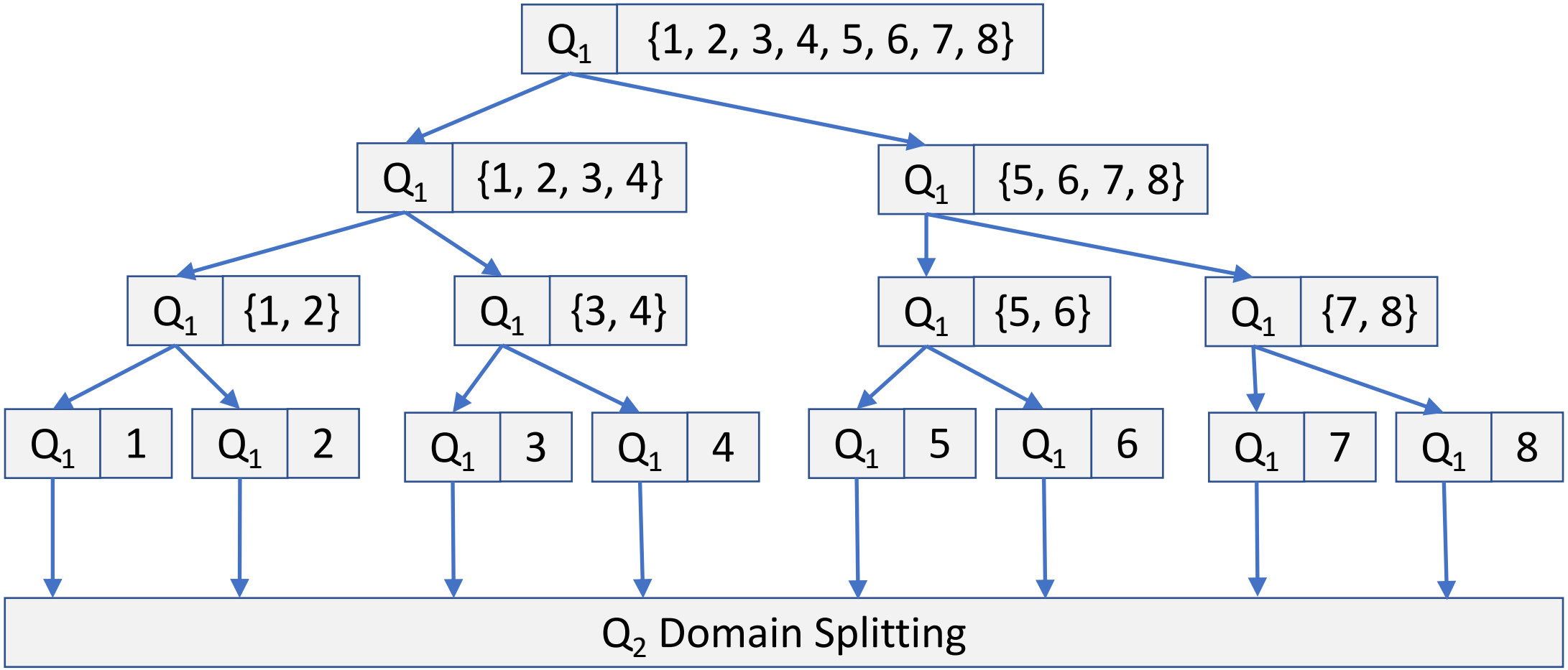
Question 1 – Domain Splitting

- Split the domain of a variable into sub-domains
  - Continue to split until each sub-domain is small enough

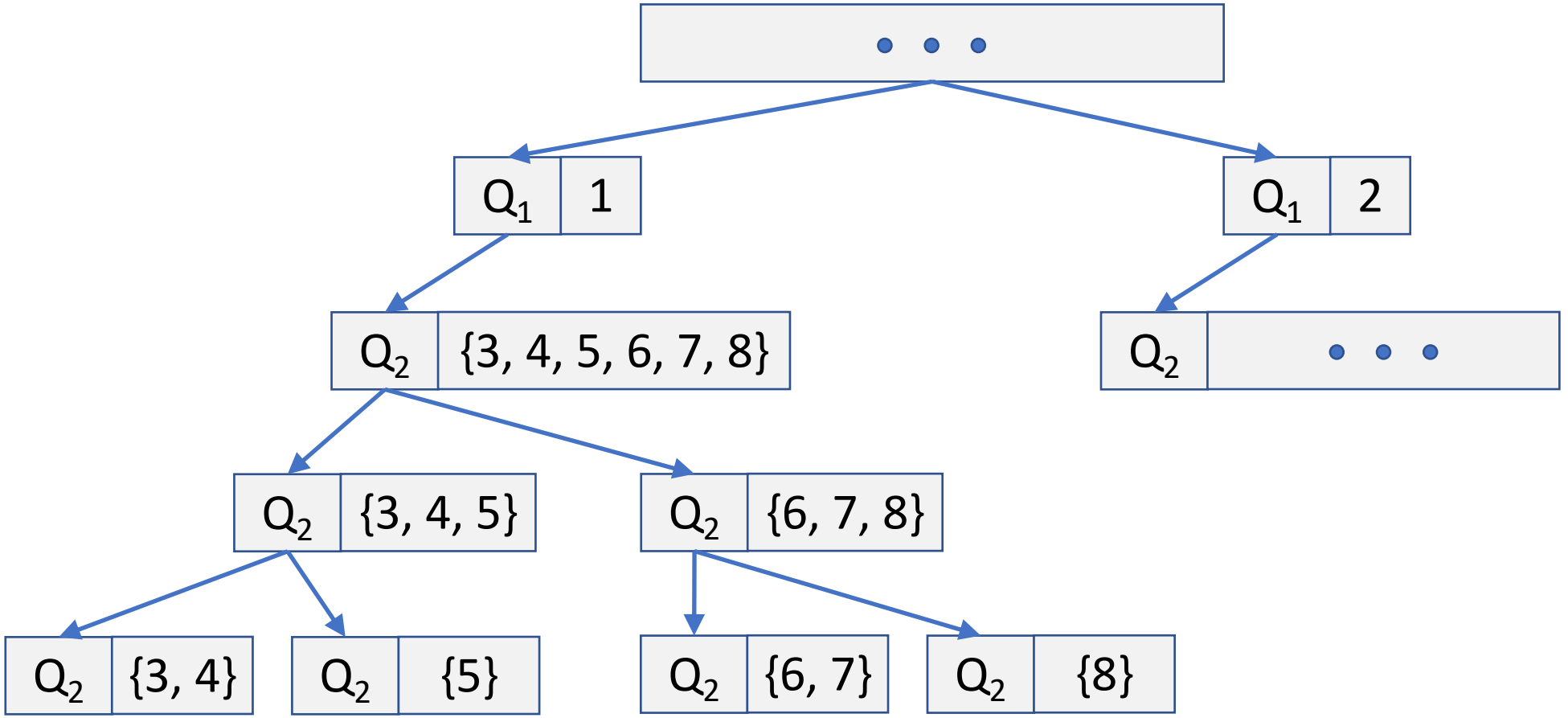
- For each minimal sub-domain, do the same for another variable
  - Maintain arc-consistency and domain consistency
  - Stop splitting when inconsistency is reached

- Can traverse the resultant tree with a graph search algorithm
  - Such as depth-first search

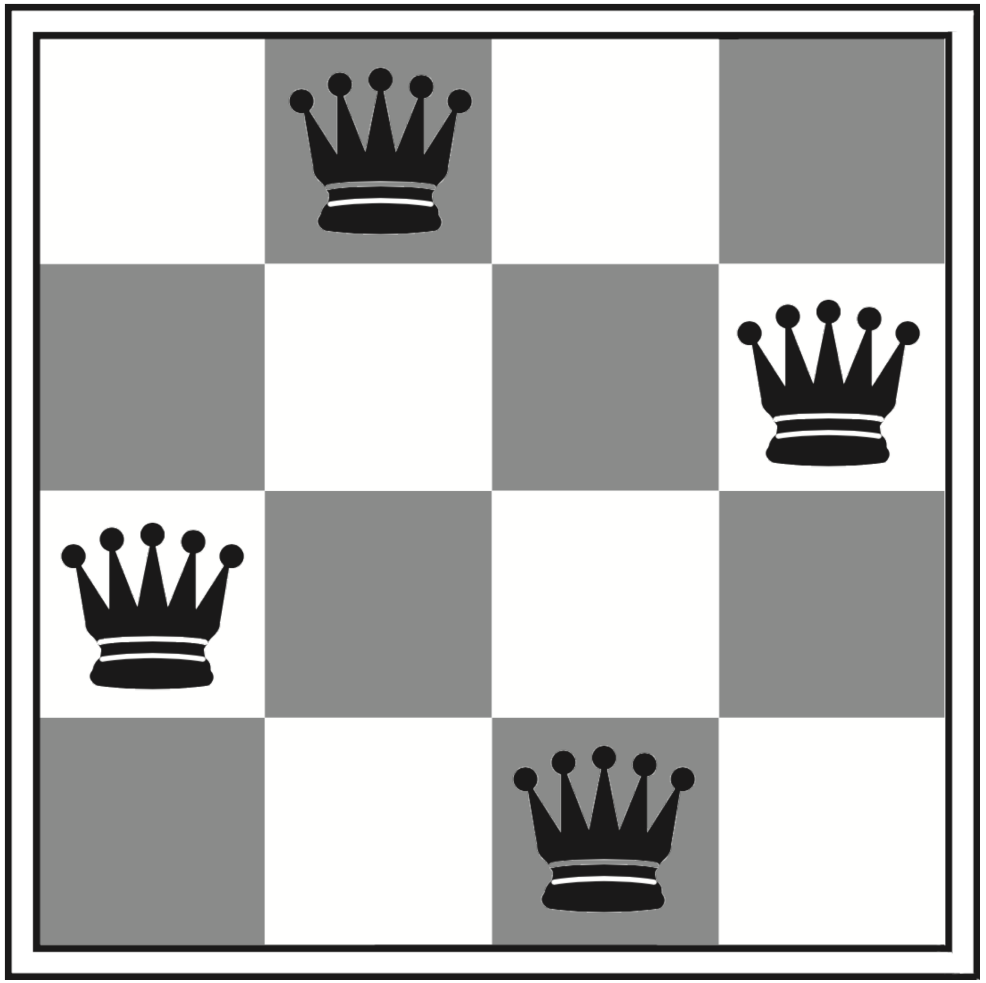
Question 1 – Domain Splitting



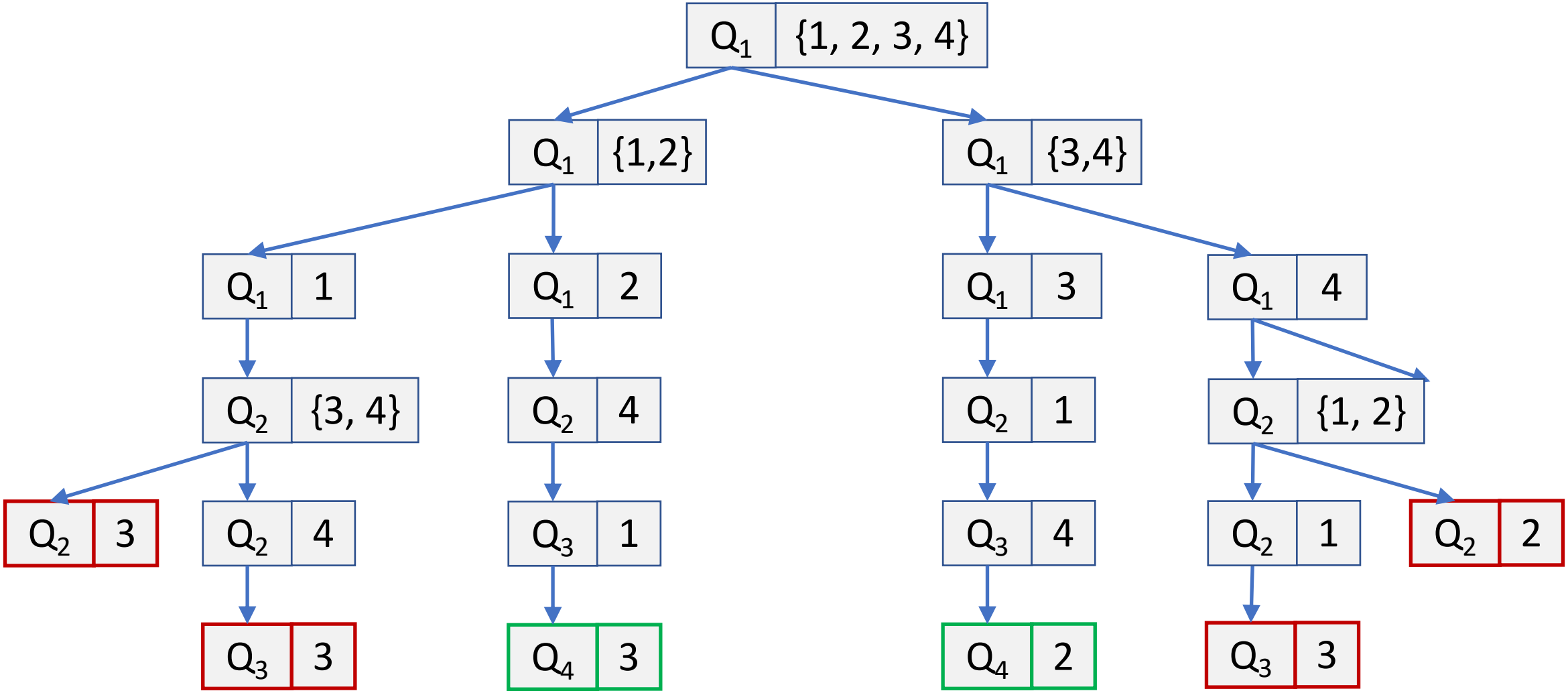
Question 1 – Domain Splitting



Question 1 – 4-Queens Problem

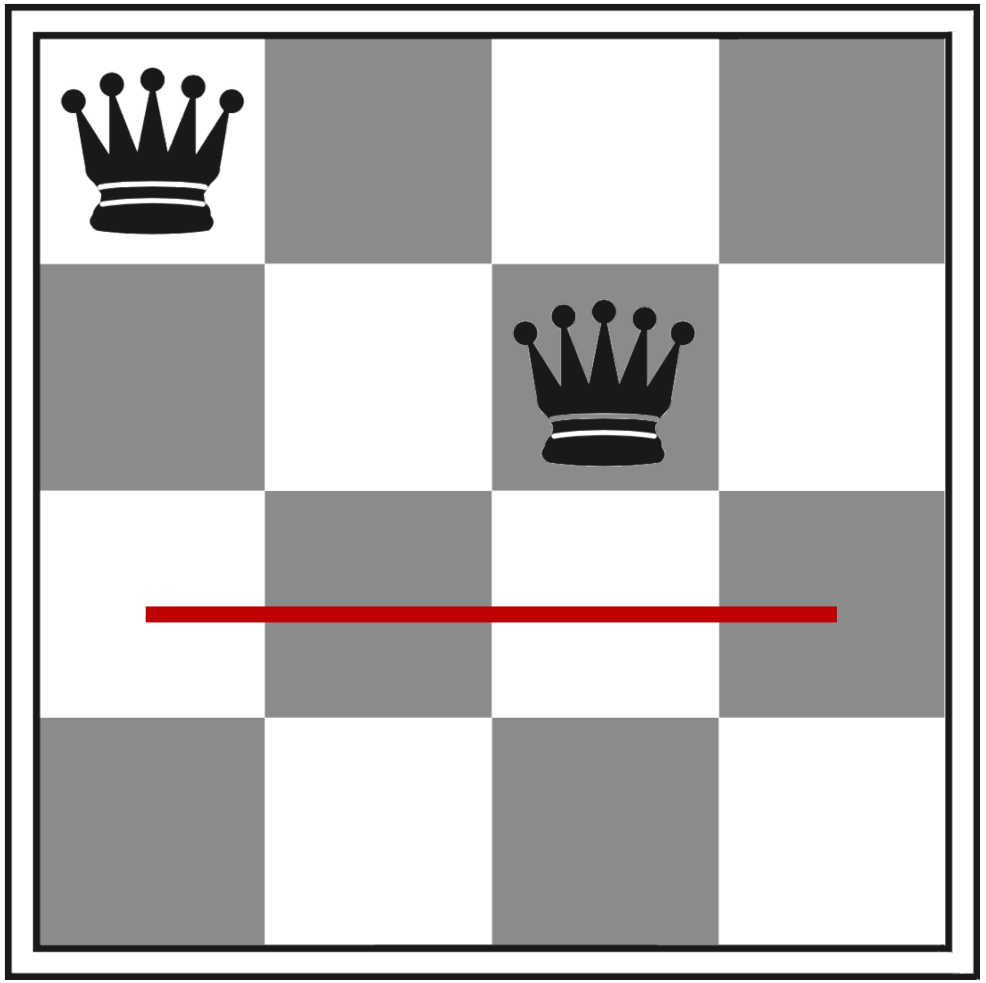


Question 1 – Domain Splitting



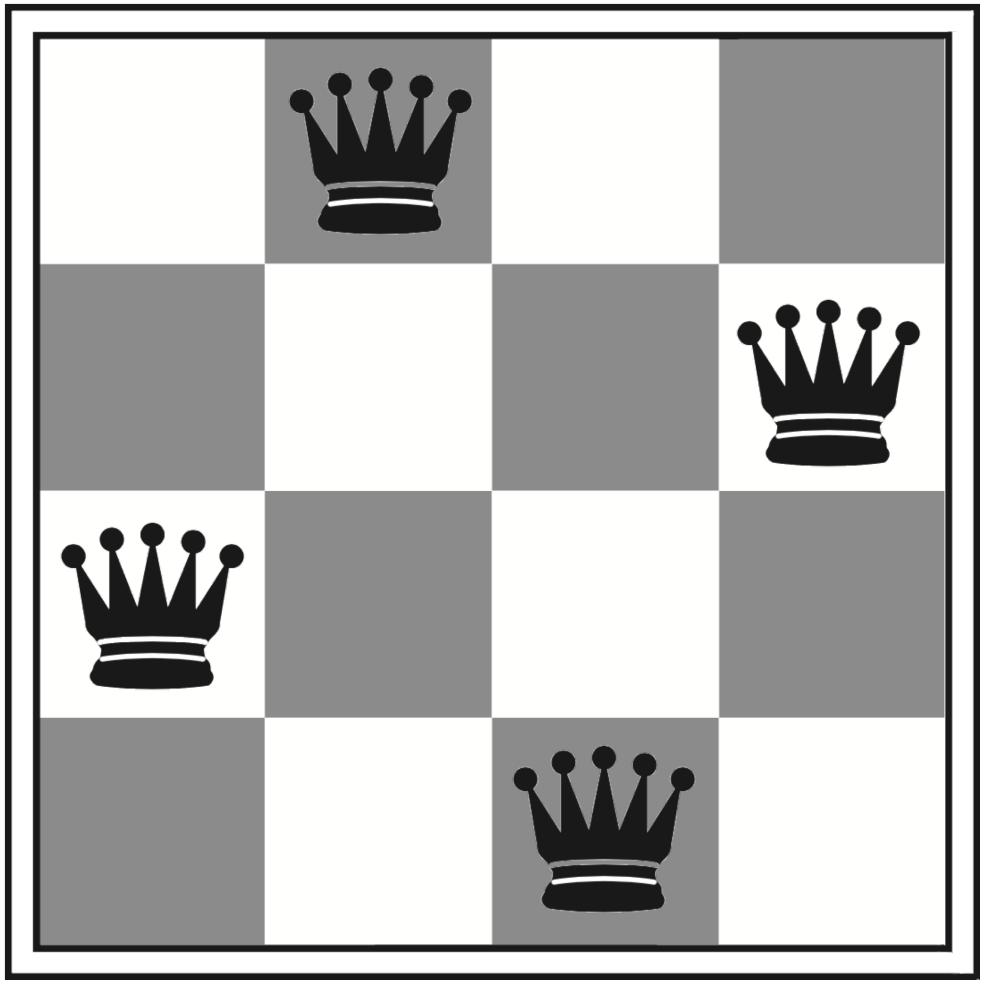
Question 1 – 4-Queens Problem

Q_1	1
Q_2	3
Q_3	



Question 1 – 4-Queens Problem

Q_1	2
Q_2	4
Q_3	1
Q_4	3



Planning

- Agent
  - Knowledge base, goals
- Environment
  - Exists in a particular state
  - Described using a series of literals
- Agent must execute actions to control the environment state

Specifying Actions (STRIPS)

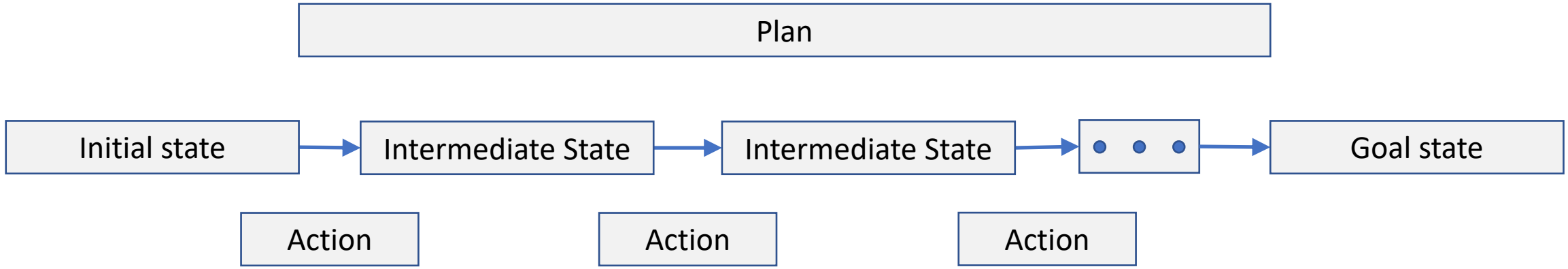
Each action has:

- Preconditions
  - Requirements to execute the action
- Add List
  - Literals made true by the action
- Delete List
  - Literals made false by the action

Specifying Actions (STRIPS)

Select a series of actions

- Initial state ⇒ Goal state

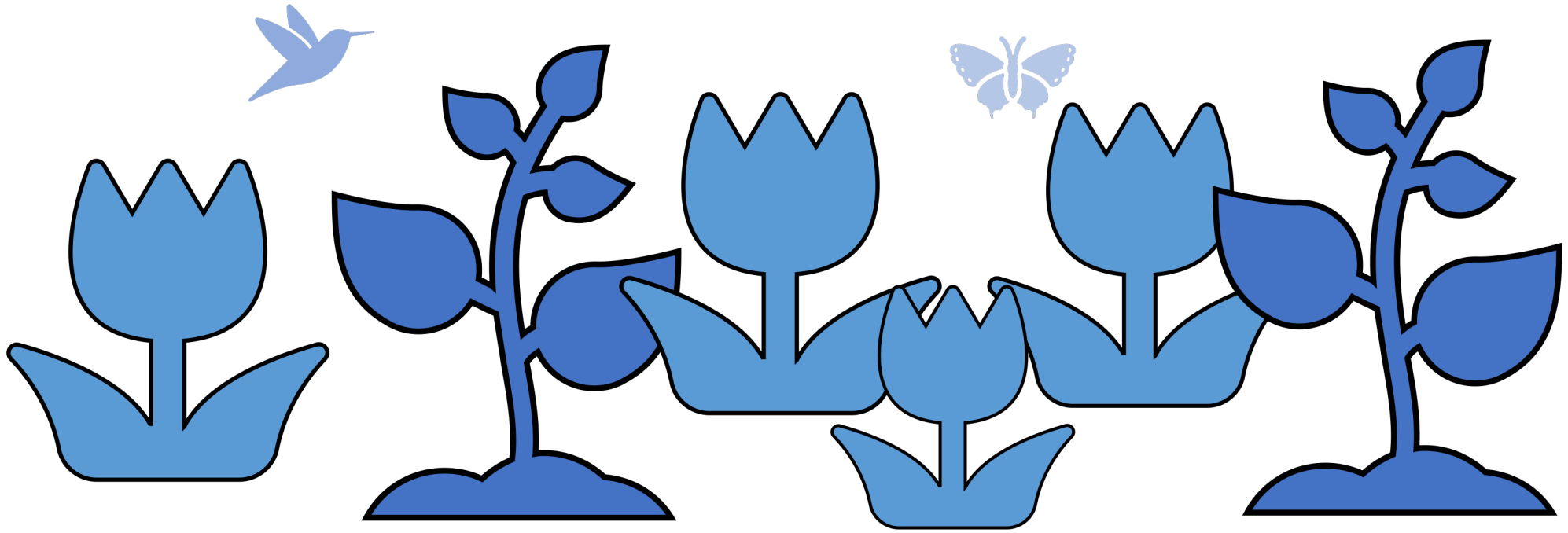


- Actions may not always execute successfully
- Backtracking may be required

Planning – Environment Example

Literals: [healthy, bugs_present, roses_present, already_watered]

Actions: [Remove_bugs, Plant_roses, Water_garden]



Specifying Actions (STRIPS)

Current state: [¬already_watered, ¬bugs_present, roses_present]

Water_garden:

- Preconditions:
  [¬already_watered, ¬bugs_present]

- Add List:
  [healthy, already_watered]

- Delete List:
  [¬already_watered]

Specifying Actions (STRIPS)

New state: [already_watered, ¬bugs_present, roses_present, healthy]

Water_garden:

- Preconditions:
  [¬already_watered, ¬bugs_present]

- Add List:
  [healthy, already_watered]

- Delete List:
  [¬already_watered]

Question 2 - Block World

Actions:

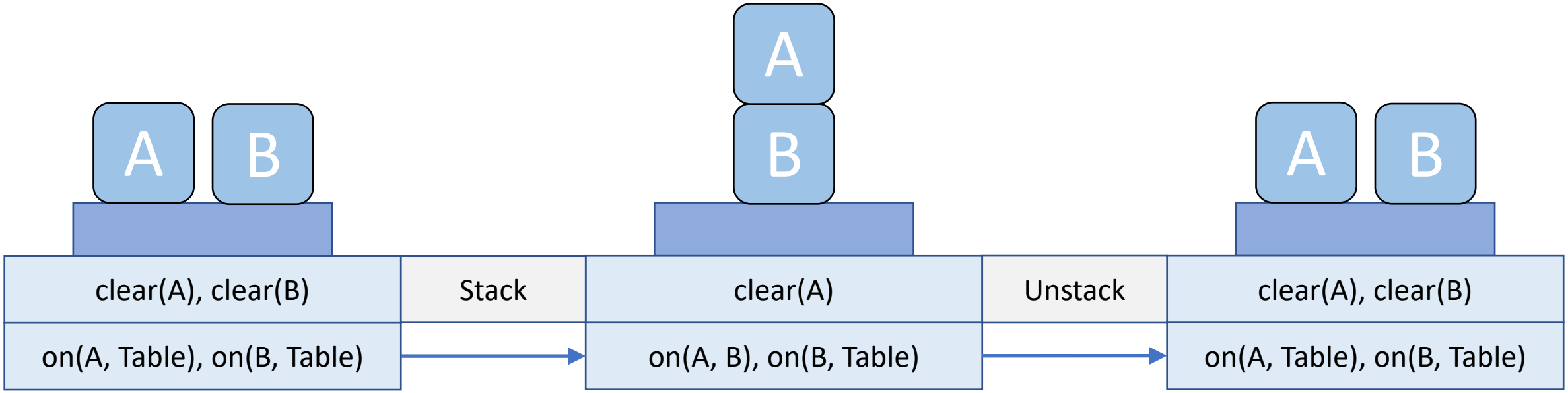
- Stack(A, B)
- Unstack(A)

Relations:

- on(A, B)
- clear(A)

Objects:

- Blocks A, B, C
- Table



Question 2 – Stack Action

Current state: [clear(A), clear(B), on(A, Table), on(B, Table)]

stack(A, B):

- Preconditions:
  [clear(A), clear(B)]

- Add List:
  [on(A, B)]

- Delete List:
  [clear(B), on(A, Table)]

Relations:

- on(A, B)
- clear(A)

New state: [clear(A), on(A, B), on(B, Table)]

Question 2 – Unstack Action

Current state: [clear(A), on(A, B), on(B, Table)]

unstack(A):

- Preconditions:
  [clear(A), on(A, B)]

- Add List:
  [on(A, Table), clear(B)]

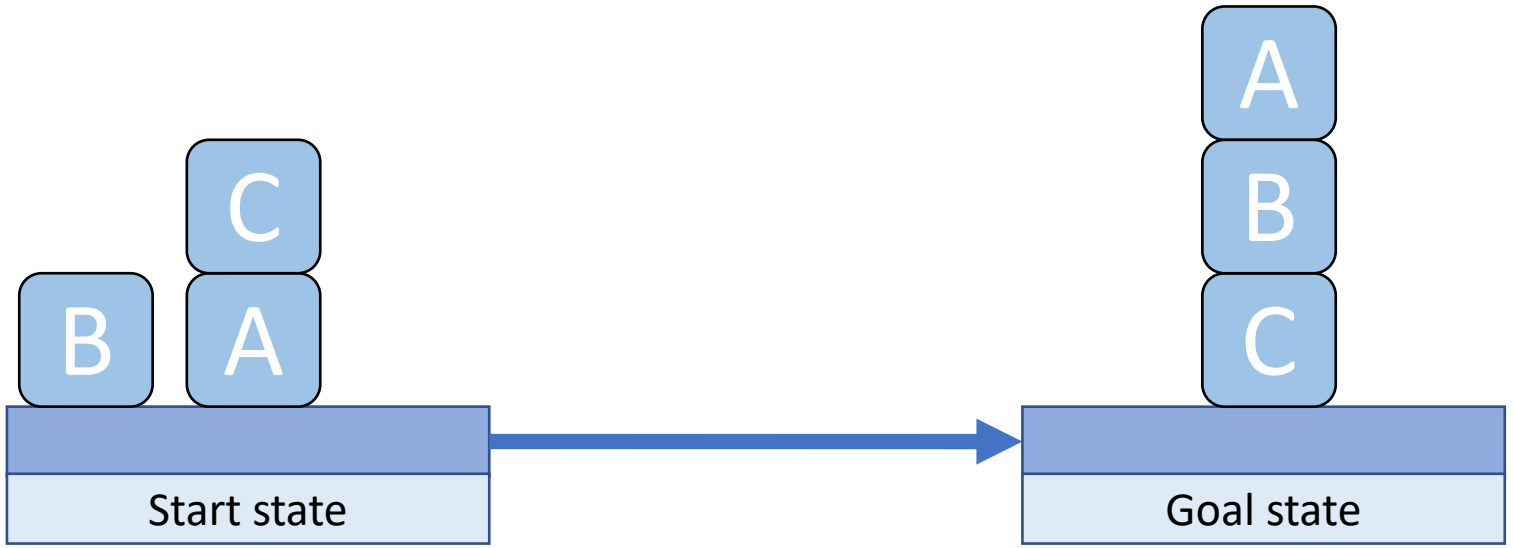
- Delete List:
  [on(A, B)]

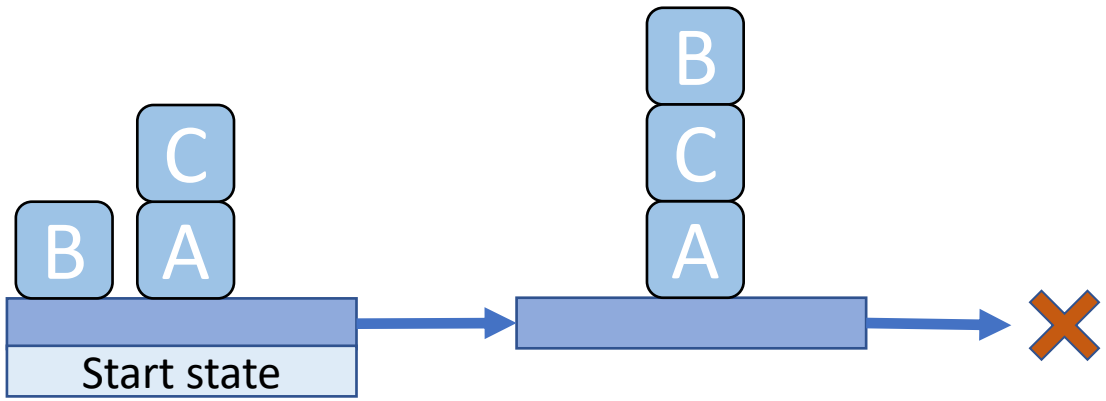
Relations:

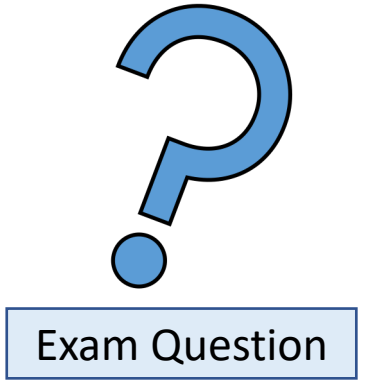
- on(A, B)
- clear(A)

New state: [clear(A), clear(B), on(A, Table), on(B, Table)]

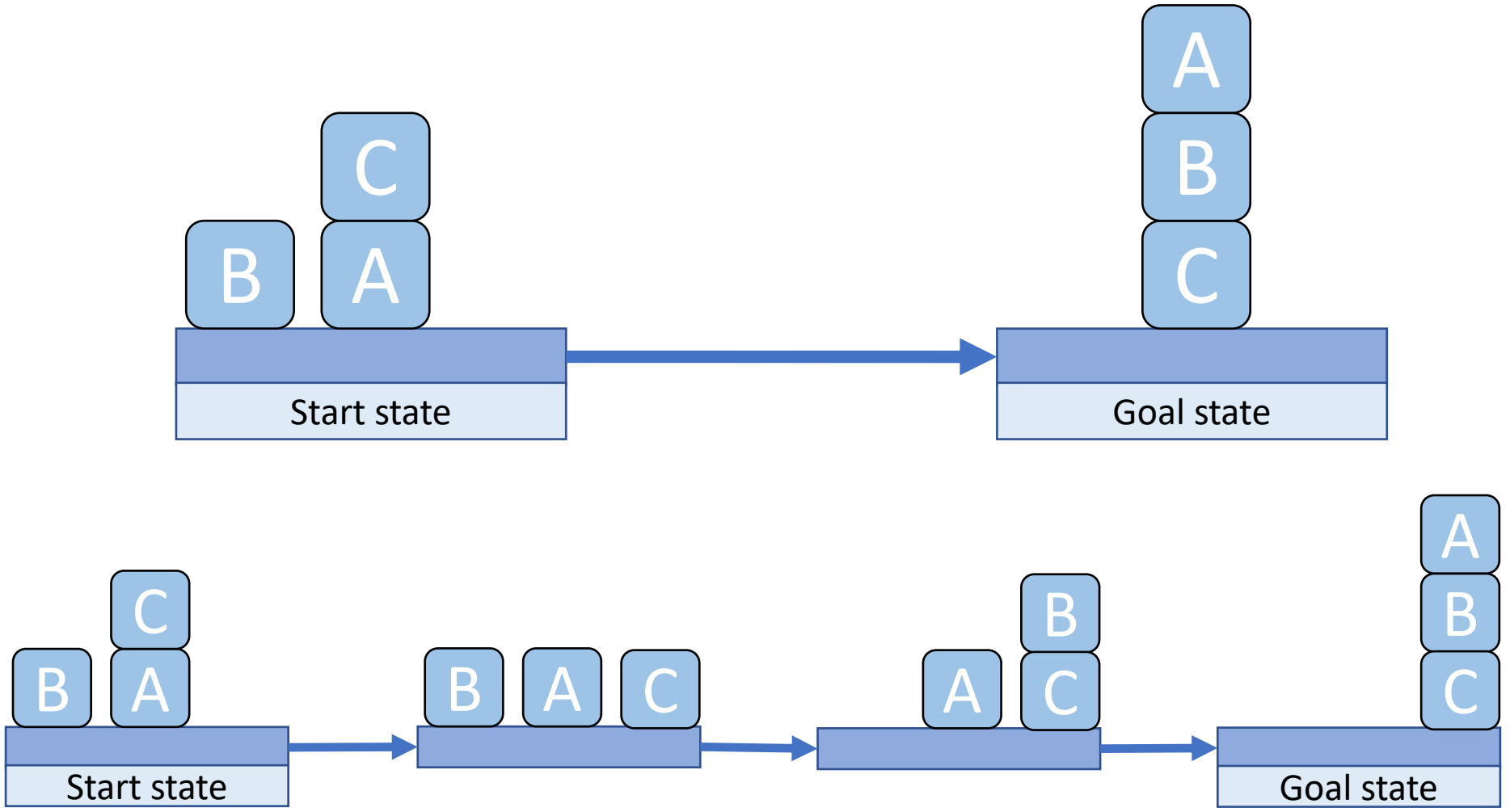
Question 3 - Sussman Anomaly



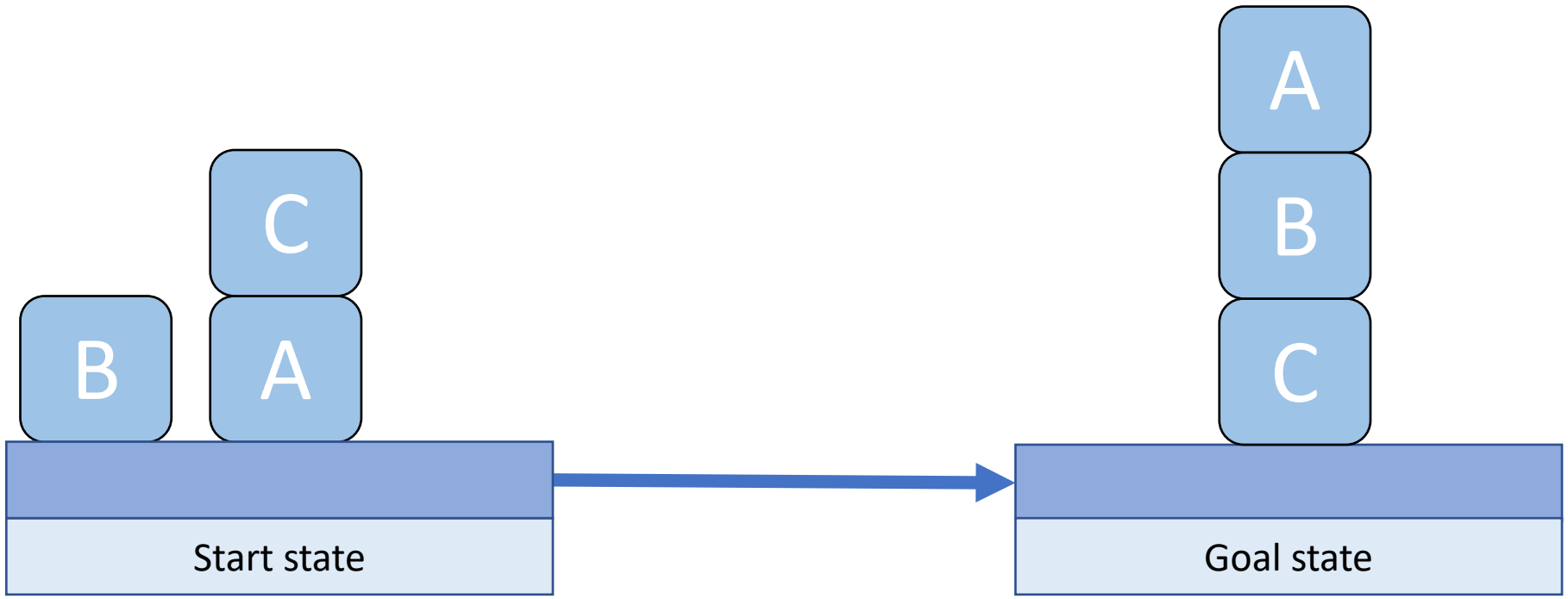




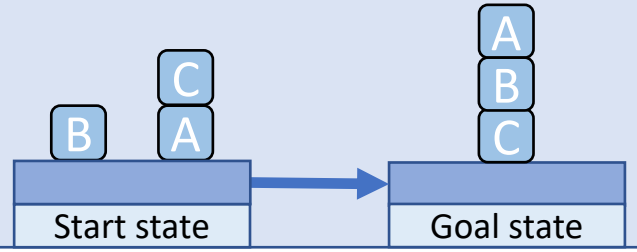
Question 3 - Sussman Anomaly



Question 3



Question 3



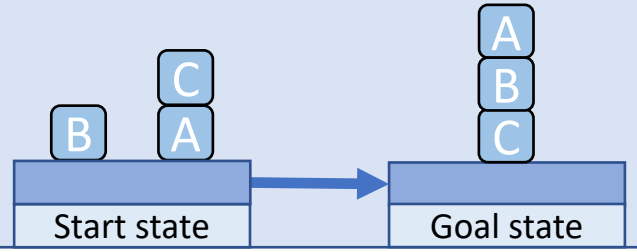
Stack(A, _):

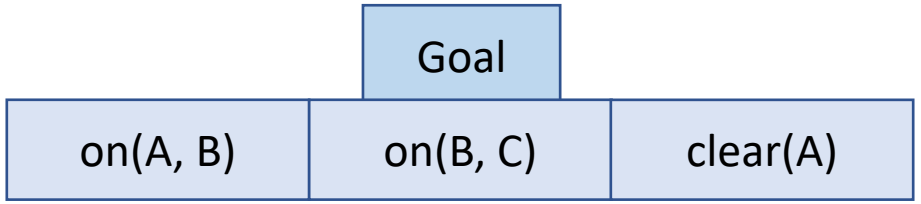
- Preconditions:
  [clear(A), clear(_)]
- Add List:
  [on(A, _)]
- Delete List:
  [clear(_), on(A, Table)]

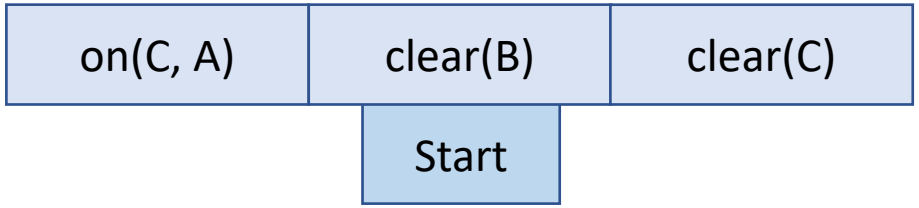
Unstack(A):

- Preconditions:
  [clear(A), on(A, _)]
- Add List:
  [on(A, Table), clear(_)]
- Delete List:
  [on(A, _)]

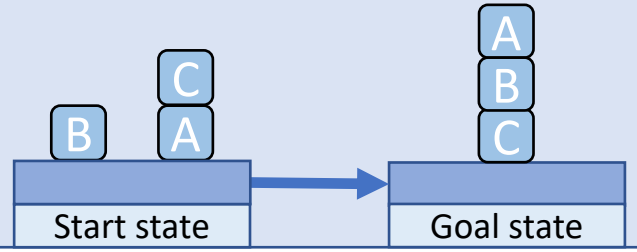
Question 3

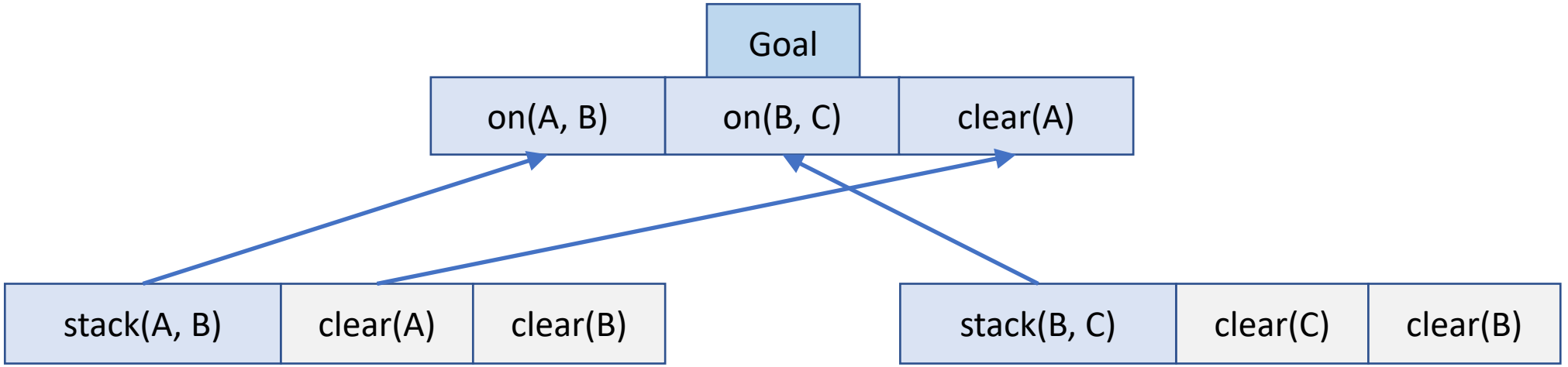

B
C
A
A
B
C
Start state
Goal state

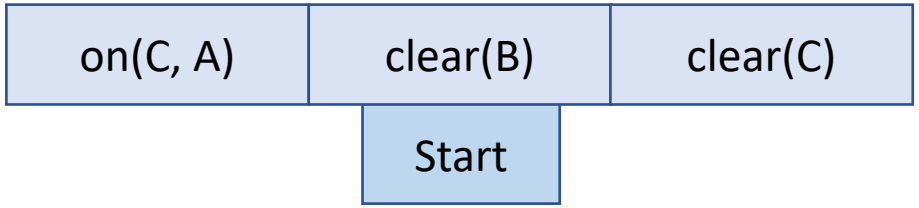

Goal
on(A, B)
on(B, C)
clear(A)


on(C, A)
clear(B)
clear(C)
Start

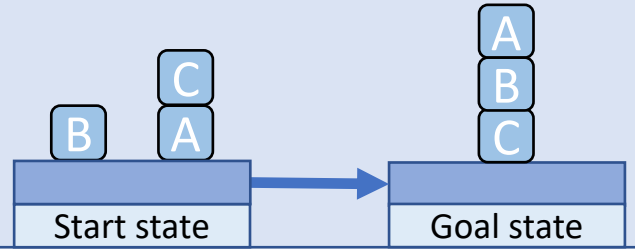
Question 3

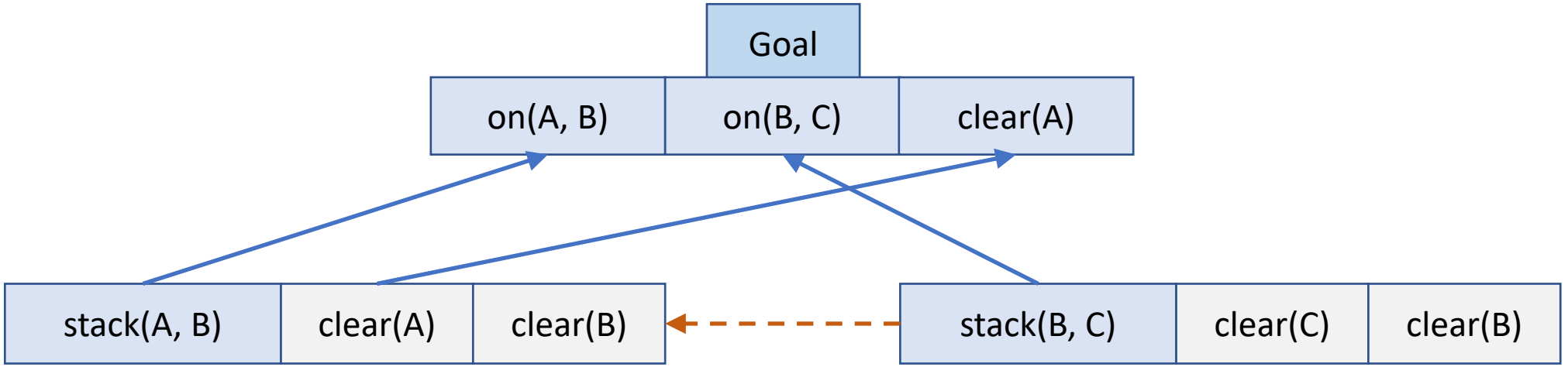

B
C
A
Start state
A
B
C
Goal state

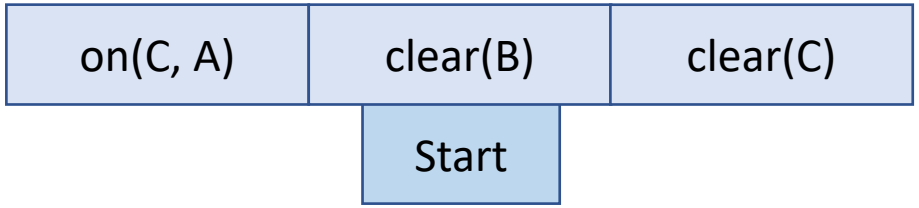

Goal
on(A, B)
on(B, C)
clear(A)
stack(A, B)
clear(A)
clear(B)
stack(B, C)
clear(C)
clear(B)


on(C, A)
clear(B)
clear(C)
Start

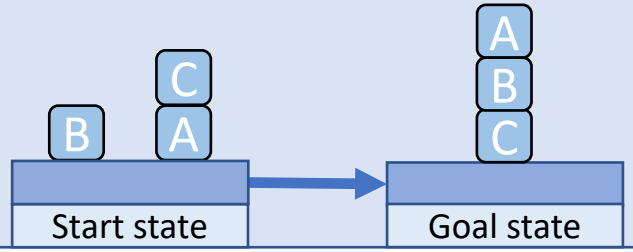
Question 3

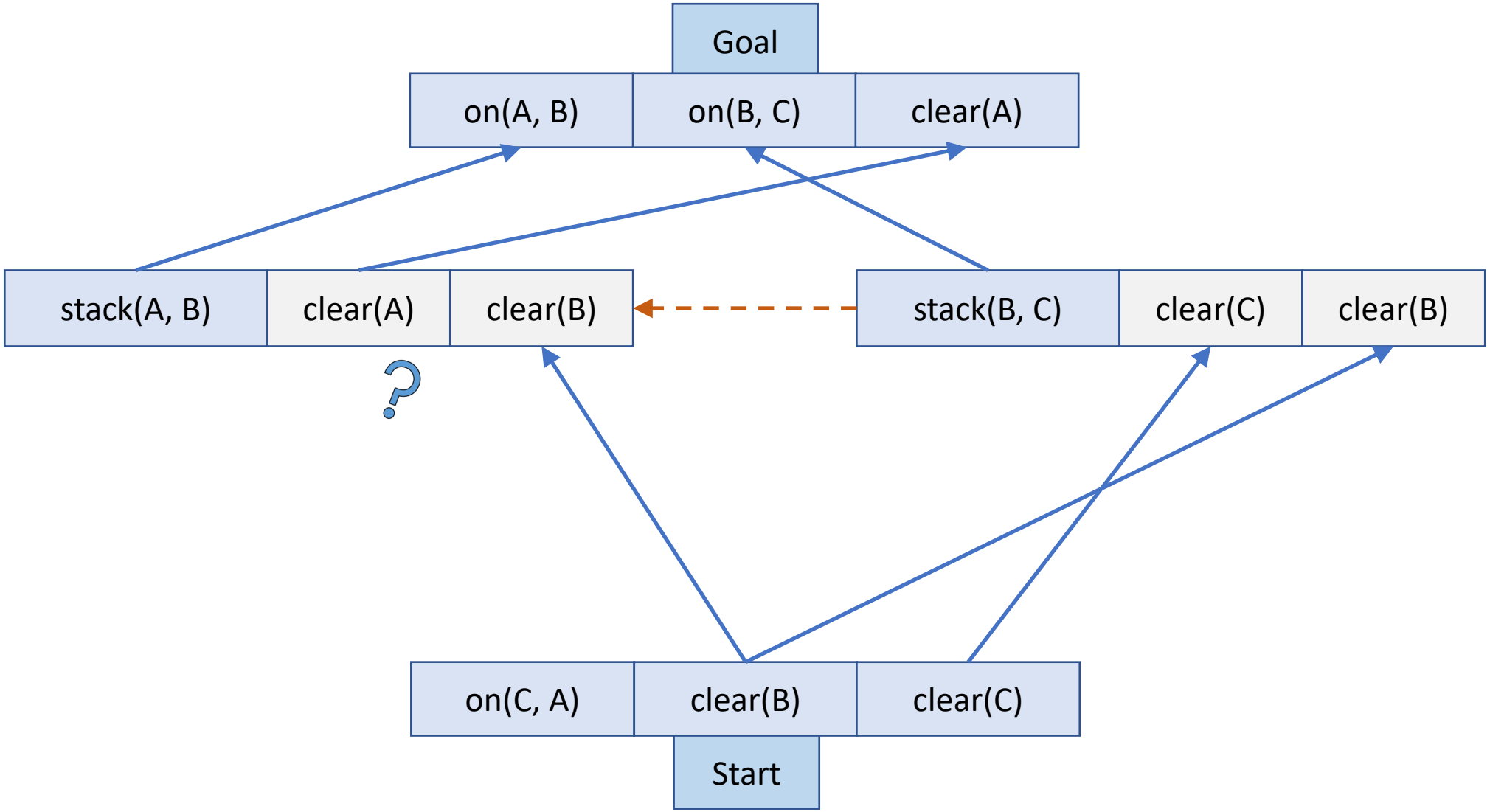

B
C
A
A
B
C
Start state
Goal state


Goal
on(A, B)
on(B, C)
clear(A)
stack(A, B)
clear(A)
clear(B)
stack(B, C)
clear(C)
clear(B)

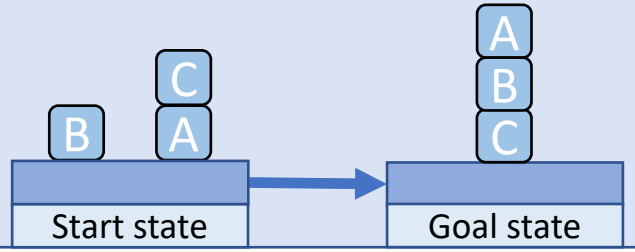

on(C, A)
clear(B)
clear(C)
Start

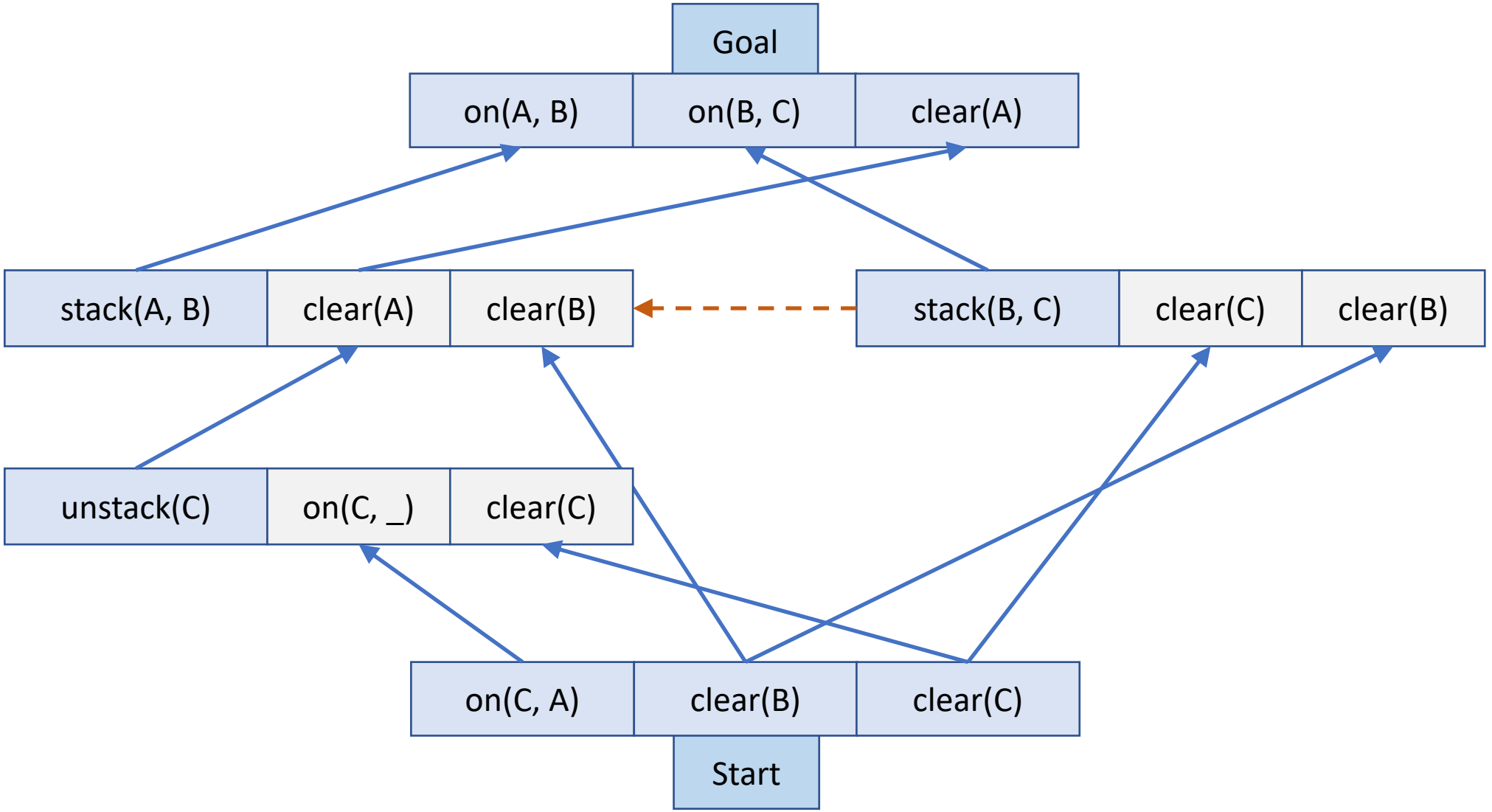
Question 3


B
C
A
A
B
C
Start state
Goal state

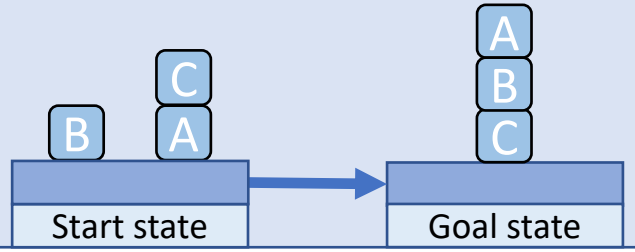

Goal
on(A, B)
on(B, C)
clear(A)
stack(A, B)
clear(A)
clear(B)
?
stack(B, C)
clear(C)
clear(B)
on(C, A)
clear(B)
clear(C)
Start

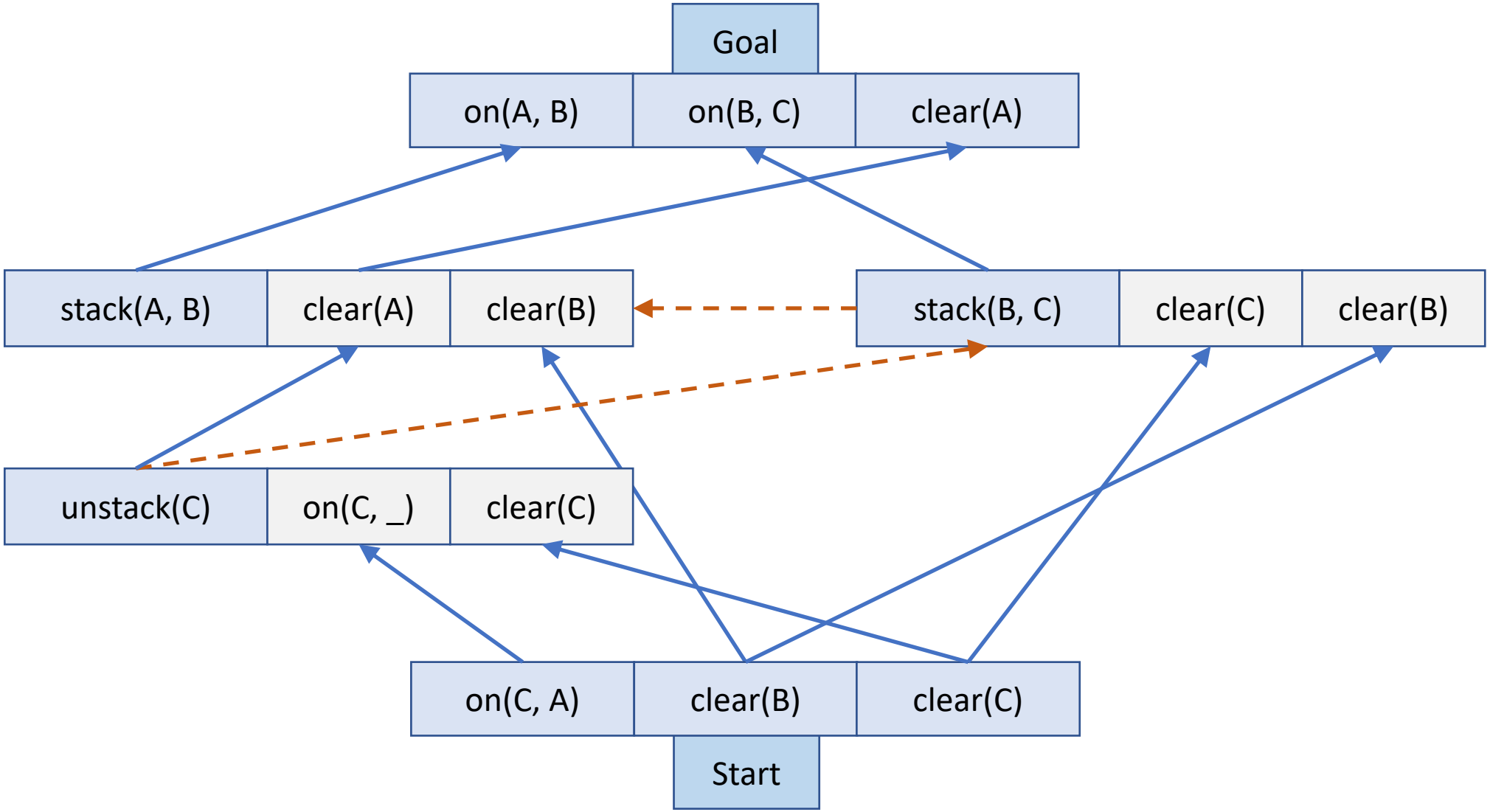
Question 3


B
C
A
A
B
C
Start state
Goal state

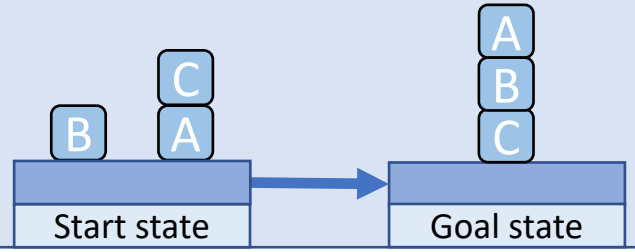

Goal
on(A, B)
on(B, C)
clear(A)
stack(A, B)
clear(A)
clear(B)
stack(B, C)
clear(C)
clear(B)
unstack(C)
on(C, _)
clear(C)
on(C, A)
clear(B)
clear(C)
Start

Question 3


B
C
A
Start state
A
B
C
Goal state


Goal
on(A, B)
on(B, C)
clear(A)
stack(A, B)
clear(A)
clear(B)
stack(B, C)
clear(C)
clear(B)
unstack(C)
on(C, _)
clear(C)
on(C, A)
clear(B)
clear(C)
Start

Question 3



Start	on(C, A)	clear(B)	clear(C)	

The clear(A) precondition of stack(A,B) does not hold in the initial state, so unstack(C) is added to the plan.

Because stack(A,B) deletes clear(B), which is a precondition of stack(B,C), stack(B,C) must be before stack(A,B).

For the same reason, unstack(C) must be before move(B, C).

The plan is therefore: unstack(C), stack(B, C), stack(A, B).

Goal	on(A, B)	on(B, C)	clear(A)	

Notes

- All python files for question 3 can be found at:
  - https://artint.info/AIPython/