Notes on Planning

Linear Planning as Search

- To define a search problem, we need to define:

Linear Planning as Search

- To define a search problem, we need to define:
  - States.
  - State successors (legal “moves” for each state).
  - An initial state.
  - A goal.
  - A test for the goal.
  - (Optionally) a cost for each moves.

Defining States in Planning

- A state is:
  - A knowledge base, in some language.
    - Can be propositional, first-order, STRIPS, or anything else.
  - The knowledge base describes what is known in that particular state.
- Example: block world.
  - A state is a knowledge base.
  - Choices of language: STRIPS, first-order.

STRIPS

- Stands for STanford Research Institute Problem Solver.
- A state of the world in STRIPS is expressed using a knowledge base.
- Statements in the knowledge base must be ground, function-free positive literals.
  - A positive literal is a predicate applied to some terms.
  - *Ground* means that the terms cannot include variables.
  - *Function-free* means that the terms cannot use functions.

Legal and Illegal Statements in STRIPS states

- Is **At(x, y)** legal as part of a STRIPS knowledge base?
- Is **At(Father(George), NewYork)** legal as part of a STRIPS knowledge base?
- Is **not(At(George, NewYork))** legal as part of a STRIPS knowledge base?

Legal and Illegal Statements in STRIPS states

- Is **At(x, y)** legal as part of a STRIPS knowledge base?
  - No, it uses variables.
- Is **At(Father(George), NewYork)** legal as part of a STRIPS knowledge base?
  - No, it uses a function.
- Is **not(At(George, NewYork))** legal as part of a STRIPS knowledge base?
  - No, it is the negation of a predicate.
  - How can we include that information in our knowledge base?

Legal and Illegal Statements in STRIPS states

- Is **At(x, y)** legal as part of a STRIPS knowledge base?
  - No, it uses variables.
- Is **At(Father(George), NewYork)** legal as part of a STRIPS knowledge base?
  - No, it uses a function.
- Is **not(At(George, NewYork))** legal as part of a STRIPS knowledge base?
  - No, it is the negation of a predicate.
  - How can we include that information in our knowledge base? **Closed world assumption.**

Preconditions of Actions in STRIPS

- Preconditions have to be positive and function free.
- Difference from statements describing a state:

Preconditions of Actions in STRIPS

- Preconditions have to be positive and function free.
- Difference from statements describing a state:
  - Can include variables.
  - on(x, y) is legal in a precondition, is not legal in a statement describing a state of the world.

Effects of Actions in STRIPS

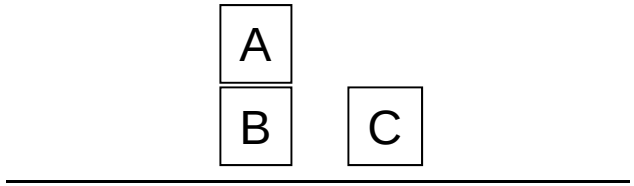
- Effects have to be function free.
- Difference from statements describing a precondition:

Effects of Actions in STRIPS

- Effects have to be function free.
- Difference from statements describing a precondition:
  - Can be negative.
  - not(clear(x))
    - legal as an effect of an action.
    - Illegal as precondition or knowledge-base statement.

Blocks World - STRIPS

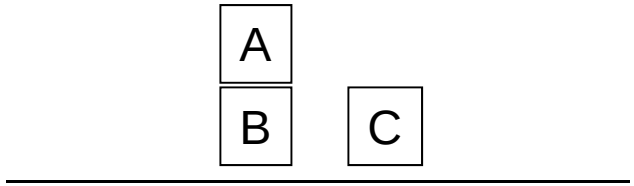
- Suppose we have:



- What do we need to include in the KB to represent that?

Blocks World - STRIPS

- Suppose we have:



- What do we need to include in the KB to represent that?

  (on A B)

  (on-table B)

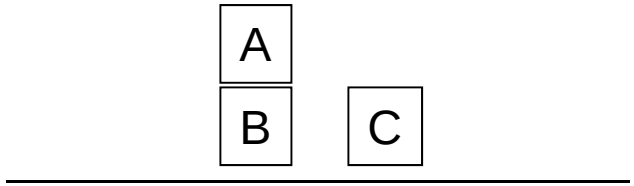
  (on-table C)

  (clear A)

  (clear C)

Blocks World - STRIPS

- Suppose we have:



- What do we need to include in the KB to represent that?

  (on A B)

  (on-table B)

  (on-table C)

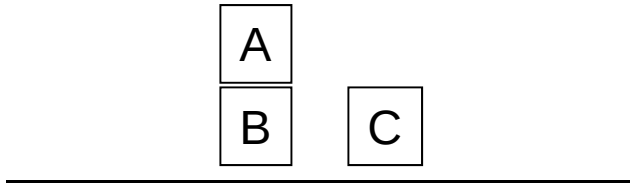
  (clear A)

  (clear C)

- How can we prove that B is not clear?

Blocks World - STRIPS

- Suppose we have:



- What do we need to include in the KB to represent that?

  (on A B)

  (on-table B)

  (on-table C)

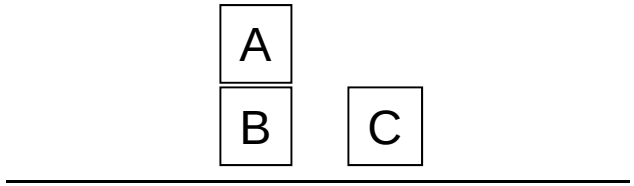
  (clear A)

  (clear C)

- How can we prove that B is not clear?
  - Using closed-world assumption.
    - The KB does not say that B is clear, => B is not clear.

Blocks World – First Order

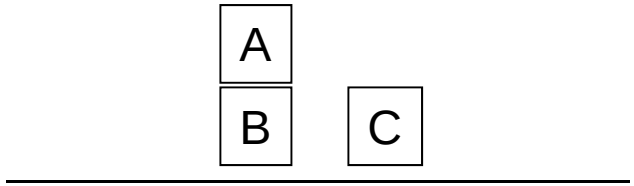
- Suppose we have:



- What do we need to include in the KB to represent that?

Blocks World – First Order

- Suppose we have:



- What do we need to include in the KB to represent that? This is what we included in STRIPS:

  on(A, B)

  on-table(B)

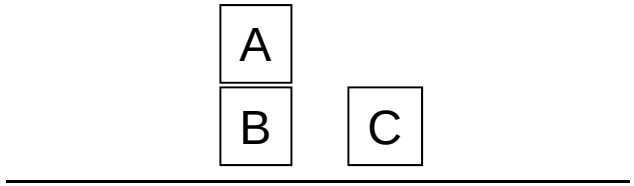
  on-table(C)

  clear(A)

  clear(C)

Blocks World – First Order

- Suppose we have:



- What do we need to include in the KB to represent that?

  on(A, B)

  on-table(B)

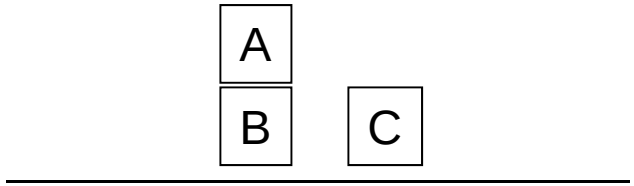
  on-table(C)

  clear(A)

  clear(C)

- How can we prove that B is not clear?

Blocks World – First Order



- Suppose we have:
- What do we need to include in the KB to represent that?

  on(A, B)

  on-table(B)

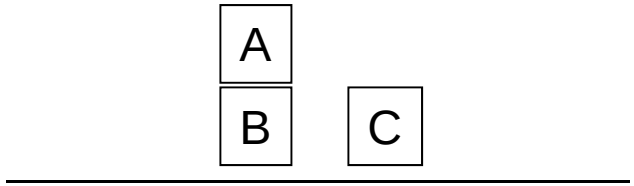
  on-table(C)

  clear(A)

  clear(C)

- How can we prove that B is not clear?
- We can’t. We need to include additional info in the knowledge base.

Blocks World – First Order



- Suppose we have:
- What do we need to include in the KB to represent that?

  on(A, B)

  on-table(B)

  on-table(C)

  clear(A)

  clear(C)

  on(x, y) => not(clear(y))

- In a first-order KB, inference is much more complicated than in a STRIPS KB.

Representing State Successors

- A state is a KB in some language.
- What are the successors of the state?

Representing State Successors

- A state is a KB in some language.
- What are the successors of the state?
  - The results of all possible actions that are legal on that state.

Goals, and Goal Tests

- A goal is:

Goals, and Goal Tests

- A goal is: a logical expression.
- How do we test if a state is a goal?

Goals, and Goal Tests

- A goal is: a logical expression.
- How do we test if a state is a goal?
  - We check if the state, which is a KB, entails the goal.
- How is that done in first-order logic?
- How is that done in STRIPS?

POP Planner

- It is still a search algorithm.
- However, there is an important difference from a linear planner: the meaning of a search state:
- Linear planner:
  - A search state is a possible state of the world.
  - The initial search state is the initial state of the world.
  - The goal state is a state that satisfies goal conditions.
- POP planner:
  - A search state is a partial plan.
  - The initial state is the empty plan, with specified initial conditions and goal conditions.
  - The goal state is a complete plan, with no open preconditions.

Successors in POP Planning

- In POP, a search state is a partial plan.
- A successor of a search state can be obtained by doing all of the above:
  - Adding an action to satisfy an open precondition.
  - Adding appropriate ordering constraints (links) between the action, its preconditions, and the open precondition(s) that it satisfies.
    - If multiple open preconditions are satisfied, there exist multiple ways for adding appropriate ordering constraints.
    - In that case, multiple successor nodes must be created.
  - Adding appropriate ordering constraints to handle clobbering.

Example: Ordering 10 Books

- We want to order 10 books from Amazon:
  - book3, book7, book13, book17, book20, book25, book30, book35, book40, book50.
- Facts in the knowledge base:

  has(Amazon, book1)

  has(Amaxon, book2)

  …

  has(Amazon, book1000000)
- Action buy(book, store)
  - preconds: has(store, book)

Example: Ordering 10 Books

- Viewed as search for a sequential plan:
  - branching factor: 1,000,000
  - depth of solution: 10
  - BFS or IDS would require visiting about $1{,}000{,}000^{10}$ nodes to find a solution.
    - Unnecessarily hard for a problem that humans find very easy to solve.

Example: Ordering 10 Books

- Sequential plan solution (one of many):

  buy(book3, Amazon)

  buy(book7, Amazon)

  buy(book13, Amazon)

  buy(book17, Amazon)

  buy(book20, Amazon)

  buy(book25, Amazon)

  buy(book30, Amazon)

  buy(book35, Amazon)

  buy(book40, Amazon)

  buy(book50, Amazon)

Example: Ordering 10 Books

- Viewed as POP search (DFS on partial plans):

START

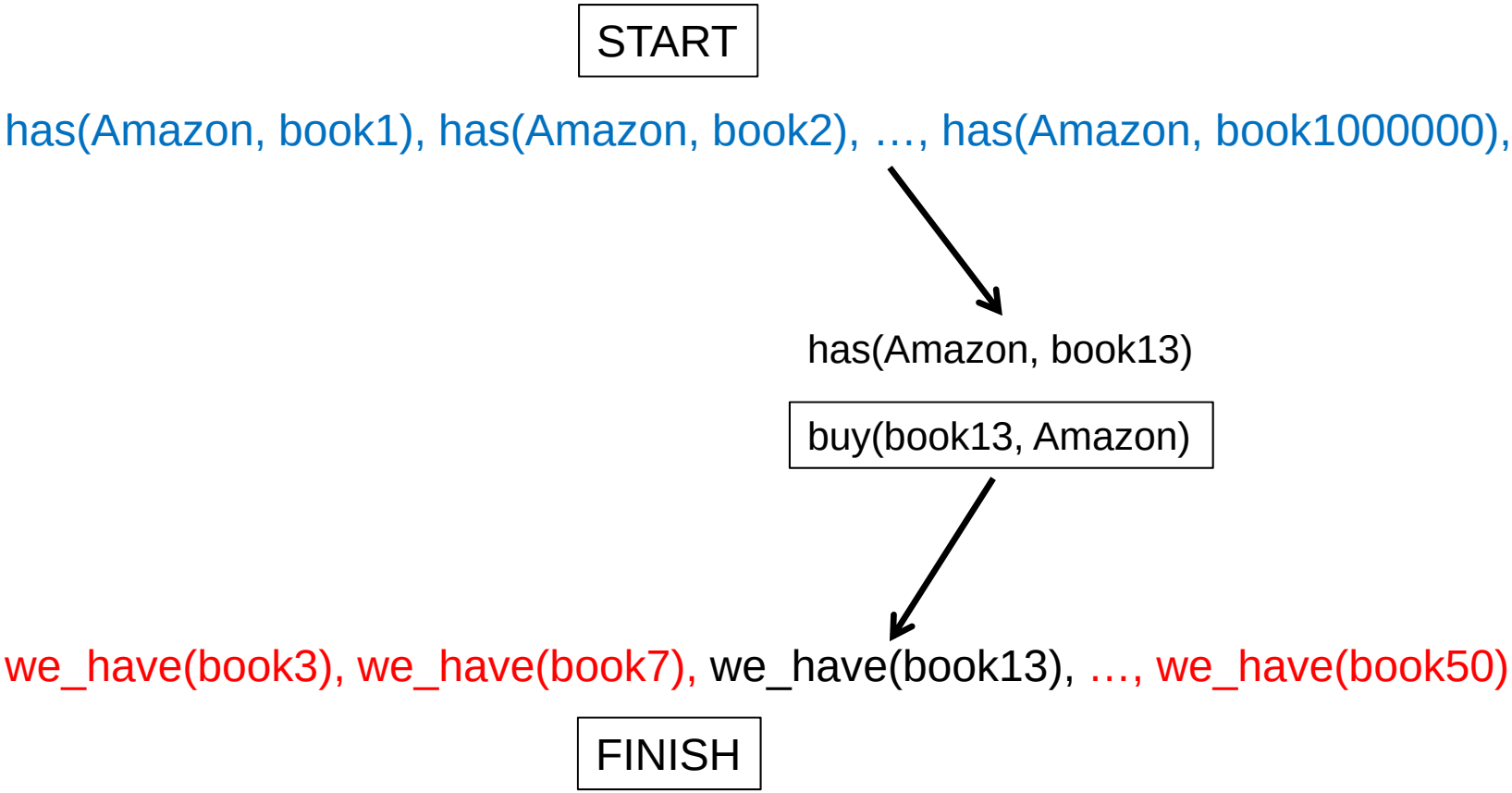
has(Amazon, book1), has(Amazon, book2), …, has(Amazon, book1000000),

we_have(book3), we_have(book7), we_have(book13), …, we_have(book50)

FINISH

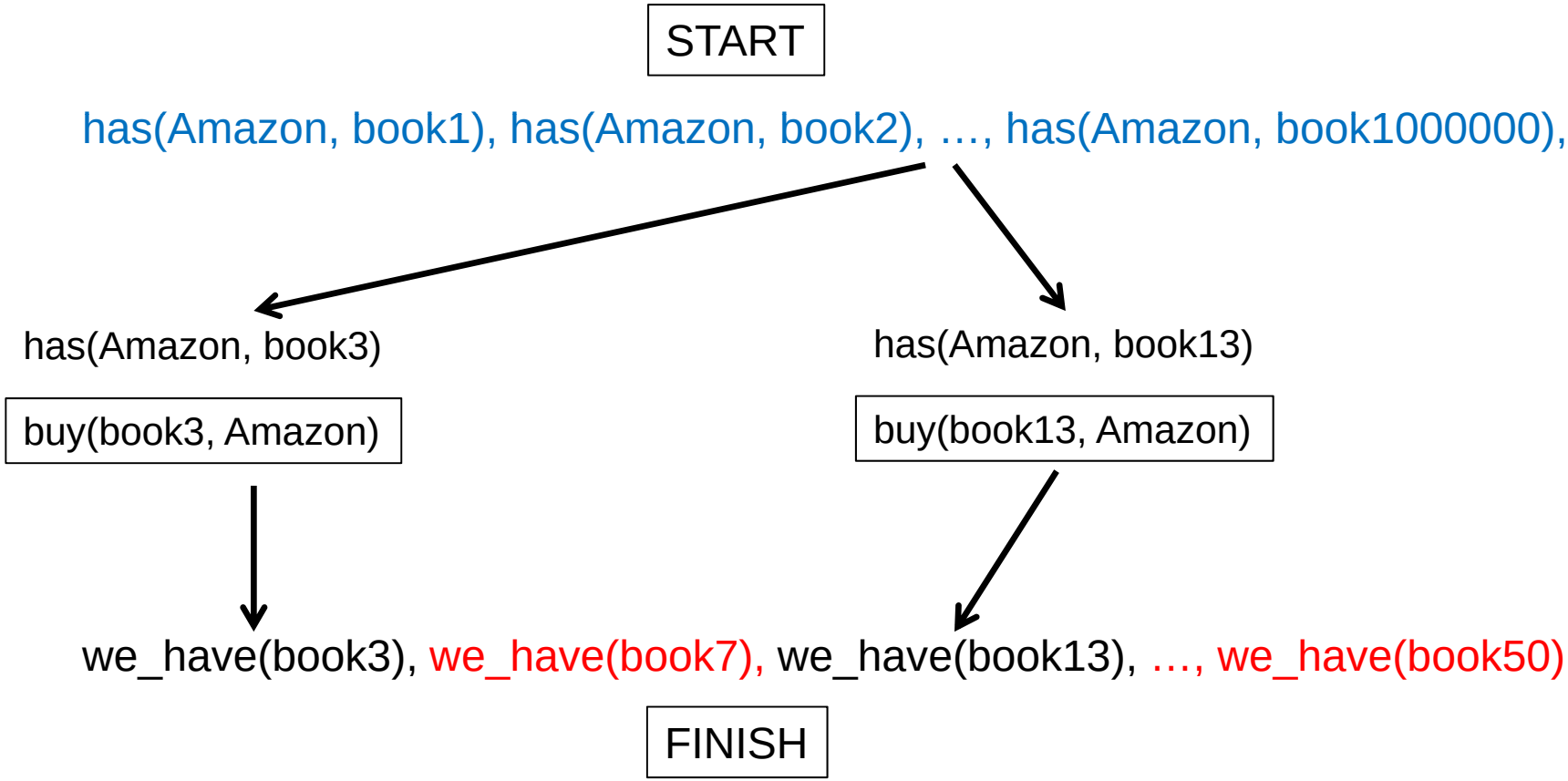
Example: Ordering 10 Books

- Viewed as POP search (DFS on partial plans):
  - Pick an open precondition to satisfy



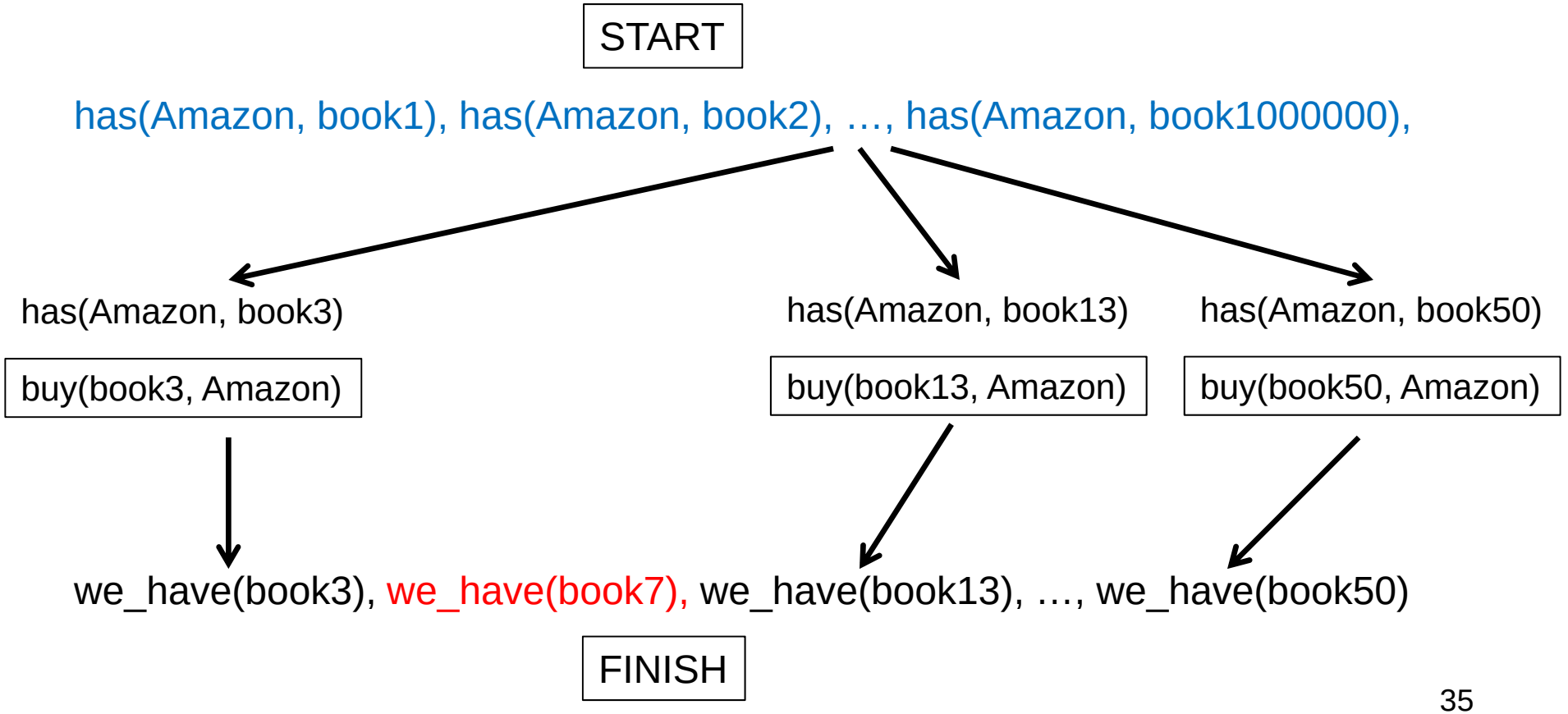
Example: Ordering 10 Books

- Viewed as POP search (DFS on partial plans):
  - Pick another open precondition to satisfy



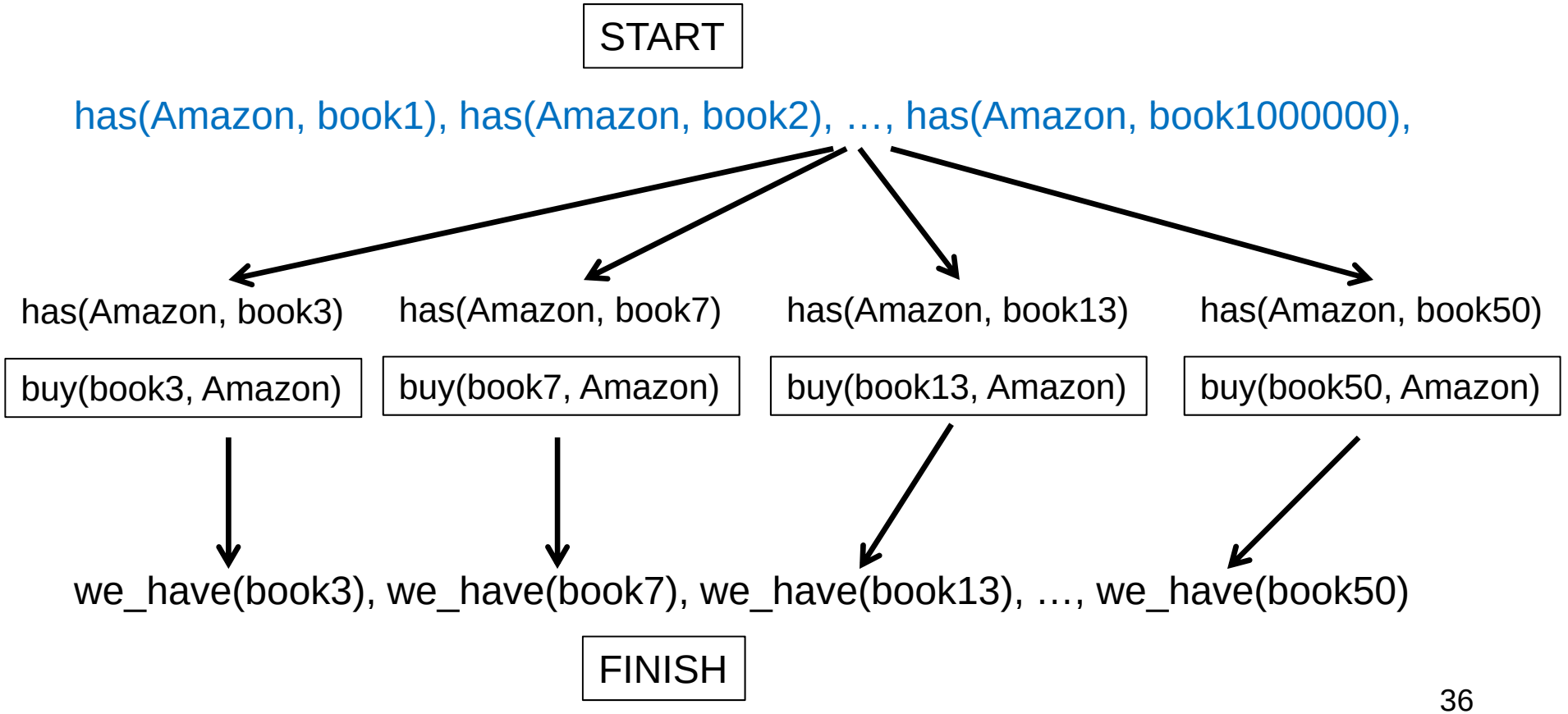
Example: Ordering 10 Books

- Viewed as POP search (DFS on partial plans):
  - Pick another open precondition to satisfy



Example: Ordering 10 Books

- Viewed as POP search (DFS on partial plans):
  - Pick another open precondition to satisfy



Comparison of Linear Search and POP

- In easy examples like this, POP does very well, and visits relatively few search nodes.
  - Much better than using searching for a sequential plan.
- POP can always get stuck and take a very long time, for examples that are more complicated.
  - Heuristics can help in that case, but we will not address that topic in detail in this class.

Heuristics in POP Planning

- Assume that the cost of the plan is the number of actions in the plan.
- Heuristic 1: number of open preconditions.
  - Is this admissible?

Heuristics in POP Planning

- Assume that the cost of the plan is the number of actions in the plan.
- Heuristic 1: number of open preconditions.
  - Is this admissible? Not if a single action can satisfy multiple open preconditions.

Heuristics in POP Planning

- Assume that the cost of the plan is the number of actions in the plan.
- Heuristic 1: number of open preconditions.
  - Is this admissible? Not if a single action can satisfy multiple open preconditions.
- Heuristic 2: smallest number of actions needed to complete the plan.
  - Is this admissible?

Heuristics in POP Planning

- Assume that the cost of the plan is the number of actions in the plan.
- Heuristic 1: number of open preconditions.
  - Is this admissible? Not if a single action can satisfy multiple open preconditions.
- Heuristic 2: smallest number of actions needed to complete the plan.
  - Is this admissible? Yes.
  - Is this doable? Yes if you can quickly establish a lower bound on this number.