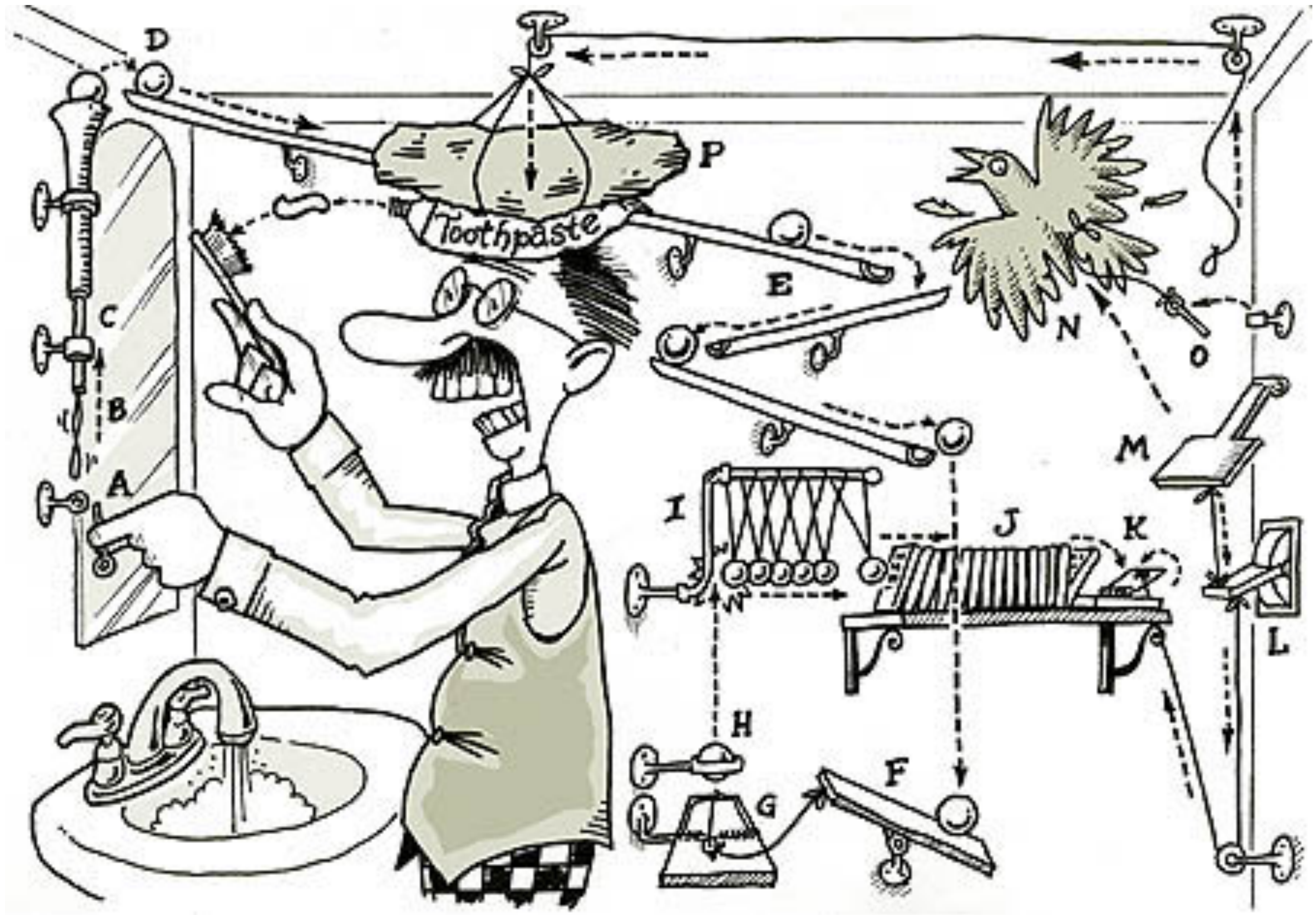
Planning (Chapter 10)



http://en.wikipedia.org/wiki/Rube_Goldberg_machine

Planning

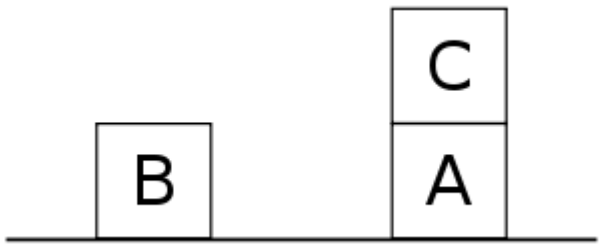
- Problem: I'm at home and I need milk, bananas, and a drill.
- How is planning different from regular search?
  - States and action sequences typically have complex internal structure
  - State space and branching factor are huge
  - Multiple subgoals at multiple levels of resolution
- Examples of planning applications
  - Scheduling of tasks in space missions
  - Logistics planning for the army
  - Assembly lines, industrial processes
  - Robotics
  - Games, storytelling

A representation for planning

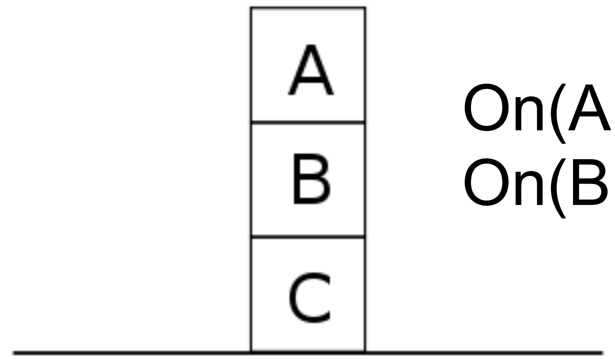
- STRIPS (Stanford Research Institute Problem Solver): classical planning framework from the 1970s
- **States** are specified as conjunctions of predicates
  - Start state: At(Home) ∧ Sells(SM, Milk) ∧ Sells(SM, Bananas) ∧ Sells(HW, Drill)
  - Goal state: At(Home) ∧ Have(Milk) ∧ Have(Banana) ∧ Have(Drill)
- **Actions** are described in terms of preconditions and effects:
  - Go(x, y)
    - **Precond:** At(x)
    - **Effect:** ¬At(x) ∧ At(y)
  - Buy(x, store)
    - **Precond:** At(store) ∧ Sells(store, x)
    - **Effect:** Have(x)
- Planning is “just” a search problem

Challenges of planning: "Sussman anomaly"

Start state:

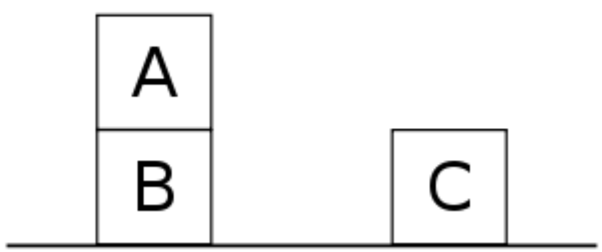


Goal state:

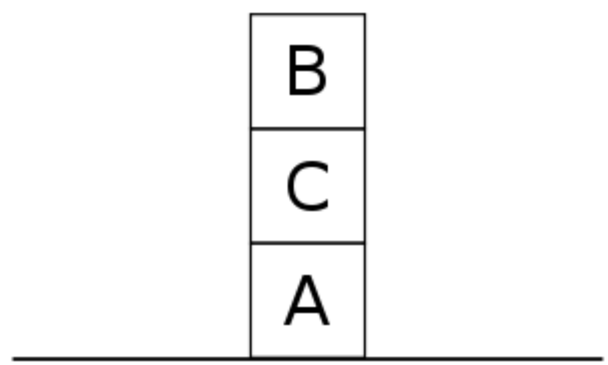


On(A,B)
On(B,C)

Let's try to achieve On(A,B):



Let's try to achieve On(B,C):



http://en.wikipedia.org/wiki/Sussman_Anomaly

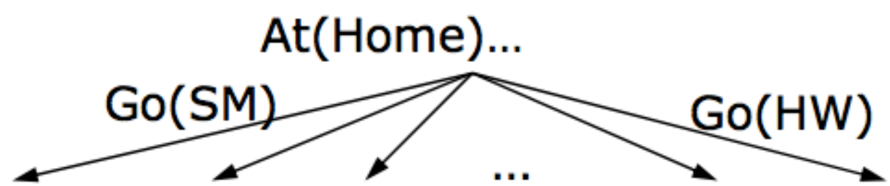
Challenges of planning: “Sussman anomaly”

- Shows the limitations of *non-interleaved* planners that consider subgoals in sequence and try to satisfy them one at a time
  - If you try to satisfy subgoal X and then subgoal Y, X might undo some preconditions for Y, or Y might undo some effects of X
- More powerful planning approaches must *interleave* the steps towards achieving multiple subgoals

http://en.wikipedia.org/wiki/Sussman_Anomaly

Algorithms for planning

- **Forward (progression) state-space search:** starting with the start state, find all applicable actions (actions for which preconditions are satisfied), compute the successor state based on the effects, keep searching until goals are met
  - Can work well with good heuristics



Algorithms for planning

- **Forward (progression) state-space search:** starting with the start state, find all applicable actions (actions for which preconditions are satisfied), compute the successor state based on the effects, keep searching until goals are met
  - Can work well with good heuristics
- **Backward (regression) relevant-states search:** to achieve a goal, what must have been true in the previous state?

Situation space planning vs. plan space planning

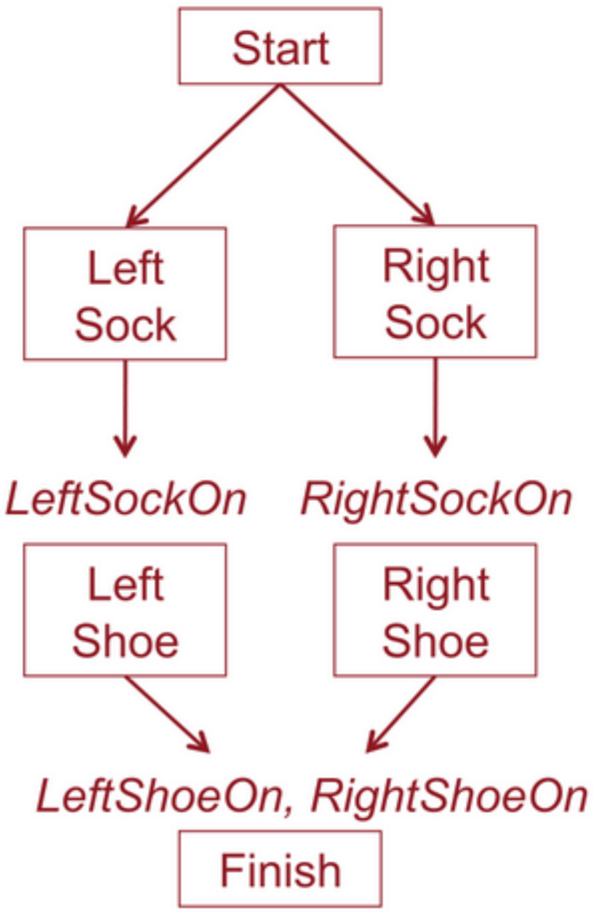
- **Situation space planners:** each node in the search space represents a world state, arcs are actions in the world
  - Plans are sequences of actions from start to finish
  - Must be *totally ordered*
- **Plan space planners:** nodes are (incomplete) plans, arcs are transformations between plans
  - Actions in the plan may be *partially ordered*
  - **Principle of least commitment:** avoid ordering plan steps unless absolutely necessary

Partial order planning

- Task: put on socks and shoes

Total order (linear) plans

Partial order plan



Partial Order Planning Example

Start

Sells(SM, Milk) At(Home) Sells(SM, Bananas)

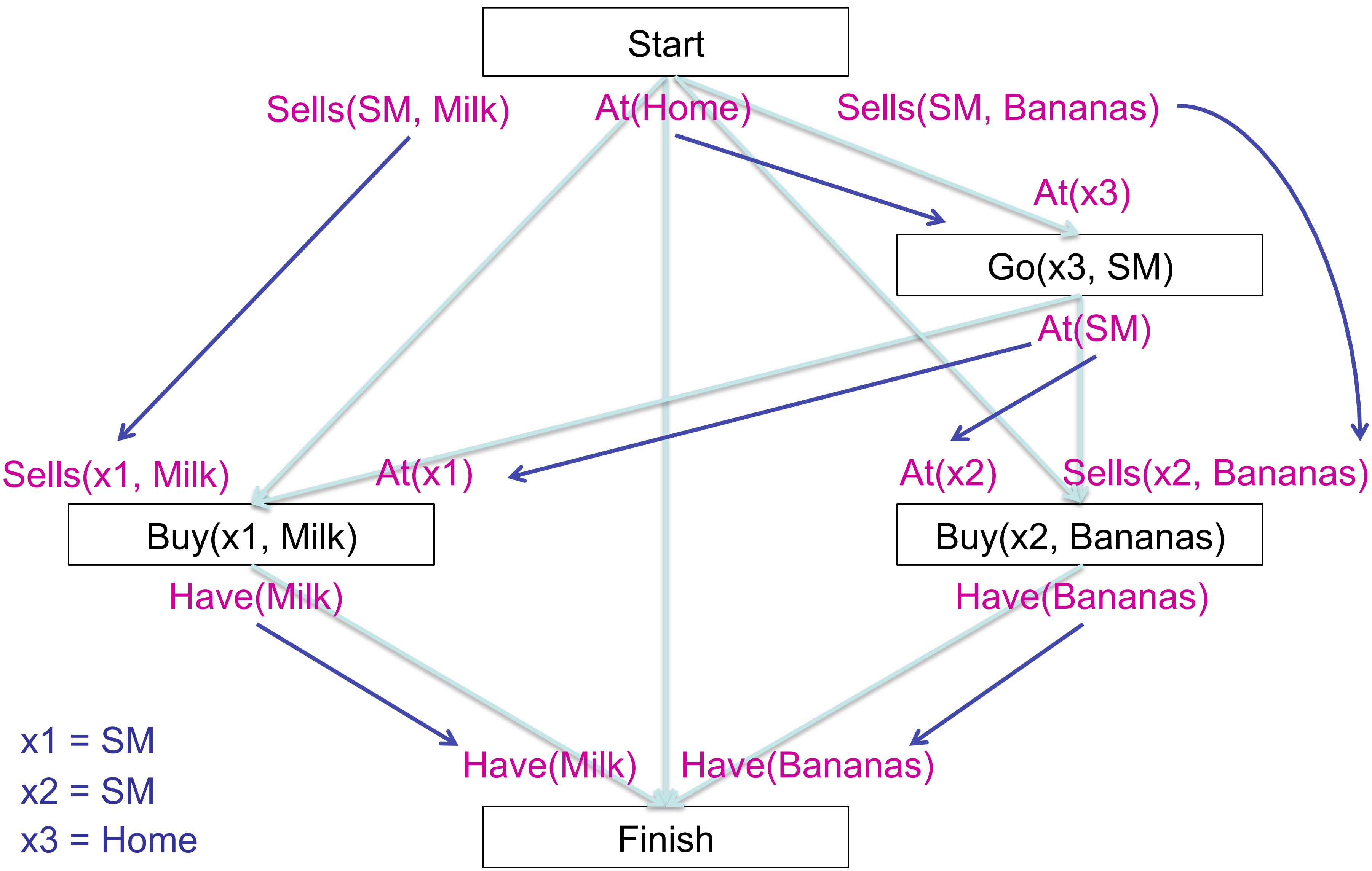
Start: empty plan

Action: find flaw in the plan and modify plan to fix the flaw

Have(Milk) Have(Bananas)

Finish

Partial Order Planning Example



Application of planning: Automated storytelling

Home

Mark Riedl

Mark Riedl, Ph.D.
Assistant Professor

School of Interactive Computing
College of Computing
Georgia Institute of Technology

GVU Center
RIM Center

Email: riedl@cc.gatech.edu

Twitter: @mark_riedl

https://research.cc.gatech.edu/inc/mark-riedl

Application of planning: Automated storytelling

- Applications
  - Personalized experience in games
  - Automatically generating training scenarios (e.g., for the army)
  - Therapy for kids with autism
  - Computational study of creativity

https://research.cc.gatech.edu/inc/mark-riedl

Complexity of planning

- Planning is *PSPACE-complete*
  - The length of a plan can be exponential in the number of "objects" in the problem!
- Example: towers of Hanoi

Complexity of planning

- Planning is *PSPACE-complete*
  - The length of a plan can be exponential in the number of “objects” in the problem!
  - So is game search
- Archetypal PSPACE-complete problem: *quantified boolean formula* (QBF)
  - Example: is this formula true?

    $\exists x_1 \forall x_2 \exists x_3 \forall x_4\ (x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_2 \vee x_3 \vee \neg x_4)$
- Compare to SAT:

  $\exists x_1 \exists x_2 \exists x_3 \exists x_4\ (x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_2 \vee x_3 \vee \neg x_4)$
- Relationship between SAT and QBF is akin to the relationship between puzzles and games

Real-world planning

- Resource constraints
- Actions at different levels of granularity: hierarchical planning
- Incorporating sensing and feedback
- Contingencies: actions failing
- Uncertainty