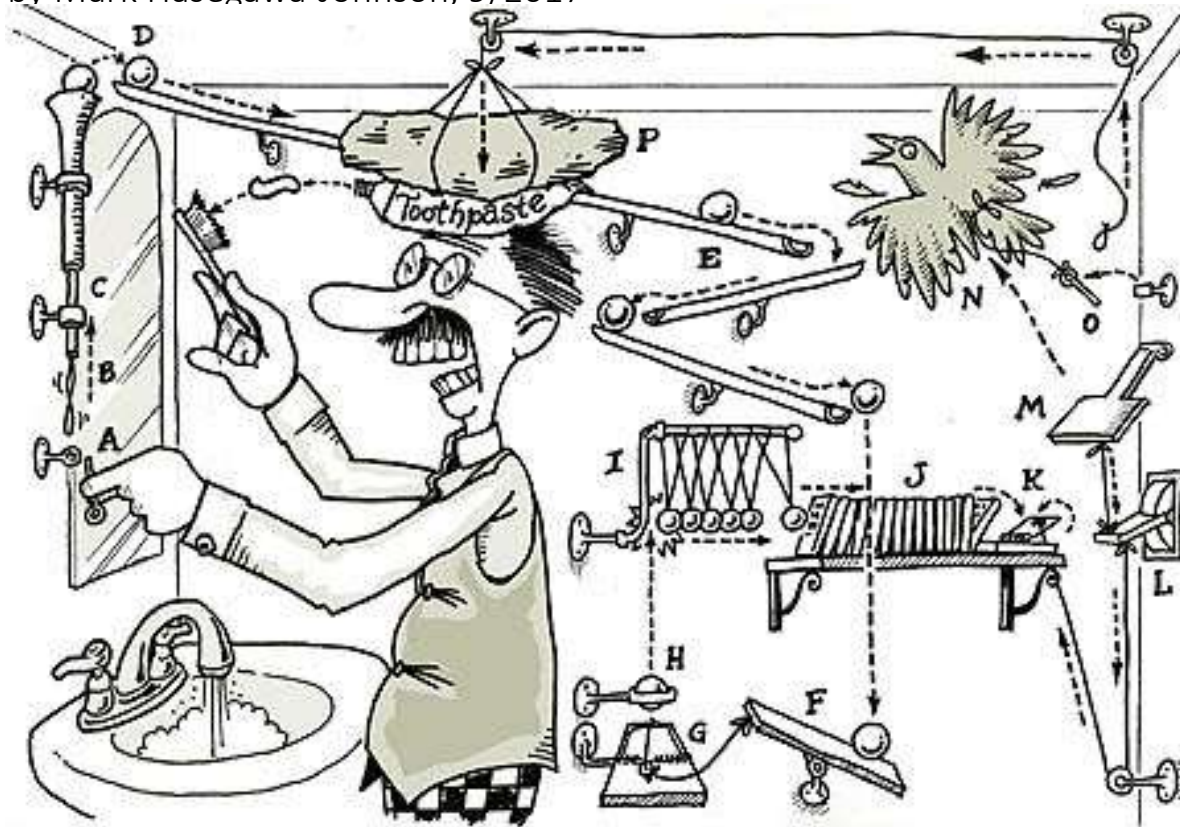


# Planning (Chapter 10)

Slides by Svetlana Lazebnik, 9/2016

with modifications by Mark Hasegawa-Johnson, 9/2017



[http://en.wikipedia.org/wiki/Rube\\_Goldberg\\_machine](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

# Automated planning and scheduling

From Wikipedia, the free encyclopedia



This article includes a [list of references](#), but **its sources remain unclear** because it has **insufficient inline citations**. Please help to [improve](#) this article by [introducing](#) more precise citations. *(January 2012)* ([Learn how and when to remove this template message](#))

**Automated planning and scheduling**, sometimes denoted as simply **AI Planning**,<sup>[1]</sup> is a branch of [artificial intelligence](#) that concerns the realization of [strategies](#) or action sequences, typically for execution by [intelligent agents](#), [autonomous robots](#) and [unmanned vehicles](#). Unlike classical [control](#) and [classification](#) problems, the solutions are complex and must be discovered and optimized in multidimensional space. Planning is also related to [decision theory](#).

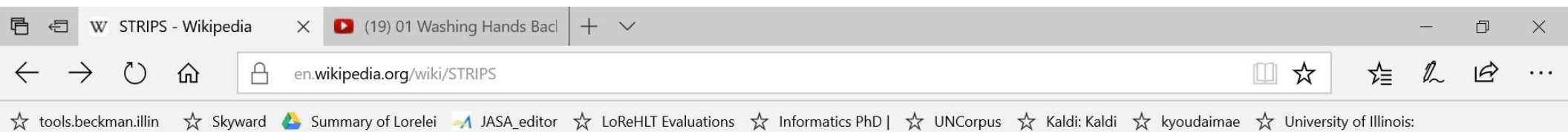
In known environments with available models, planning can be done offline. Solutions can be found and evaluated prior to execution. In dynamically unknown environments, the [strategy](#) often needs to be revised online. Models and policies must be adapted. Solutions usually resort to iterative [trial and error](#) processes commonly seen in [artificial intelligence](#). These include [dynamic programming](#), [reinforcement learning](#) and [combinatorial optimization](#). Languages used to describe planning and scheduling are often called [action languages](#).

## Contents [hide]

- 1 Overview
- 2 Domain Independent Planning
- 3 Planning Domain Modelling Languages
- 4 Algorithms for planning
  - 4.1 Classical planning
  - 4.2 Reduction to other problems

# 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:  $\text{At}(\text{home}) \wedge \text{Sells}(\text{SM}, \text{Milk}) \wedge \text{Sells}(\text{SM}, \text{Bananas}) \wedge \text{Sells}(\text{HW}, \text{drill})$
  - Goal state:  $\text{At}(\text{home}) \wedge \text{Have}(\text{Milk}) \wedge \text{Have}(\text{Banana}) \wedge \text{Have}(\text{drill})$
- **Actions** are described in terms of preconditions and effects:
  - $\text{Go}(x, y)$ 
    - **Precond:**  $\text{At}(x)$
    - **Effect:**  $\neg \text{At}(x) \wedge \text{At}(y)$
  - $\text{Buy}(x, \text{store})$ 
    - **Precond:**  $\text{At}(\text{store}) \wedge \text{Sells}(\text{store}, x)$
    - **Effect:**  $\text{Have}(x)$
- Planning is “just” a search problem



A STRIPS instance is composed of:

- An initial state;
- The specification of the goal states – situations which the planner is trying to reach;
- A set of actions. For each action, the following are included:
  - preconditions (what must be established before the action is performed);
  - postconditions (what is established after the action is performed).

Mathematically, a STRIPS instance is a quadruple  $\langle P, O, I, G \rangle$ , in which each component has the following meaning:

1.  $P$  is a set of *conditions* (i.e., **propositional variables**);
2.  $O$  is a set of *operators* (i.e., actions); each operator is itself a quadruple  $\langle \alpha, \beta, \gamma, \delta \rangle$ , each element being a set of conditions. These four sets specify, in order, which conditions must be true for the action to be executable, which ones must be false, which ones are made true by the action and which ones are made false;
3.  $I$  is the initial state, given as the set of conditions that are initially true (all others are assumed false);
4.  $G$  is the specification of the goal state; this is given as a pair  $\langle N, M \rangle$ , which specify which conditions are true and false, respectively, in order for a state to be considered a goal state.

A plan for such a planning instance is a sequence of operators that can be executed from the initial state and that leads to a goal state.

Formally, a state is a set of conditions: a state is represented by the set of conditions that are true in it. Transitions between states are modeled by a transition function, which is a function mapping states into new states that result from the execution of actions. Since states are represented by sets of conditions, the transition function relative to the STRIPS instance  $\langle P, O, I, G \rangle$  is a function

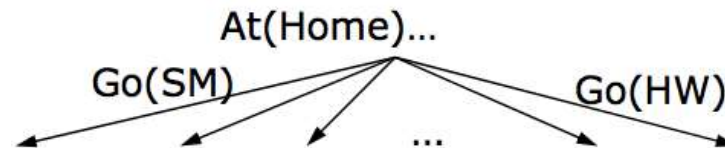
$$\text{succ} : 2^P \times O \rightarrow 2^P,$$

where  $2^P$  is the set of all subsets of  $P$ , and is therefore the set of all possible states.



# 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



<https://www.youtube.com/watch?v=QLNSkFnBYuM>

# 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?

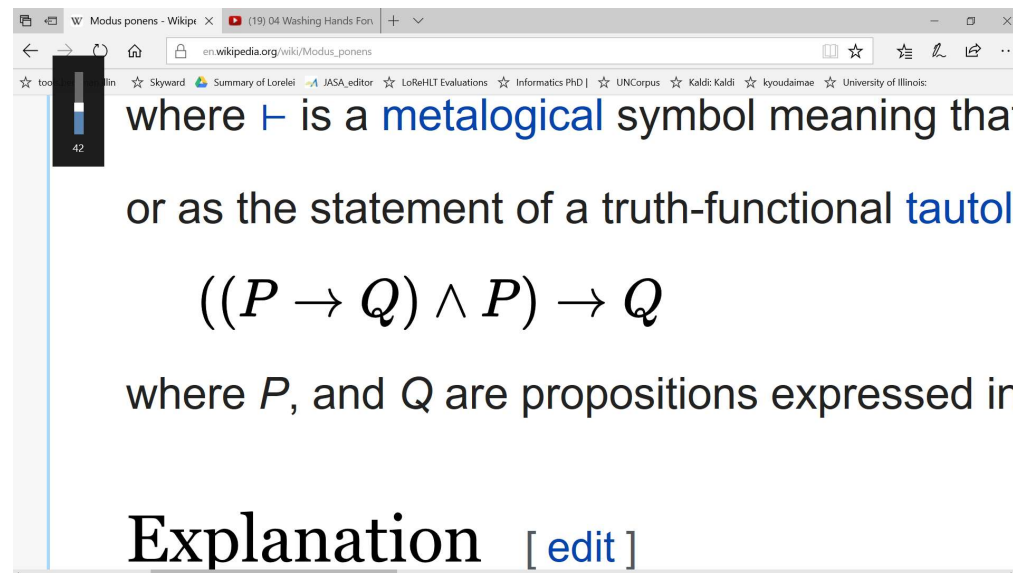
Forward chaining = start with what you already have, and then learn the next step





# Forward-chaining: each step in the search is one application of *modus ponens*

- In forward-chaining, we start from a state,  $P$ , that the search process has already achieved
- We then check to see whether it is possible to apply *modus ponens*: given the current state  $P$ , is there any operation  $P \rightarrow Q$  available?

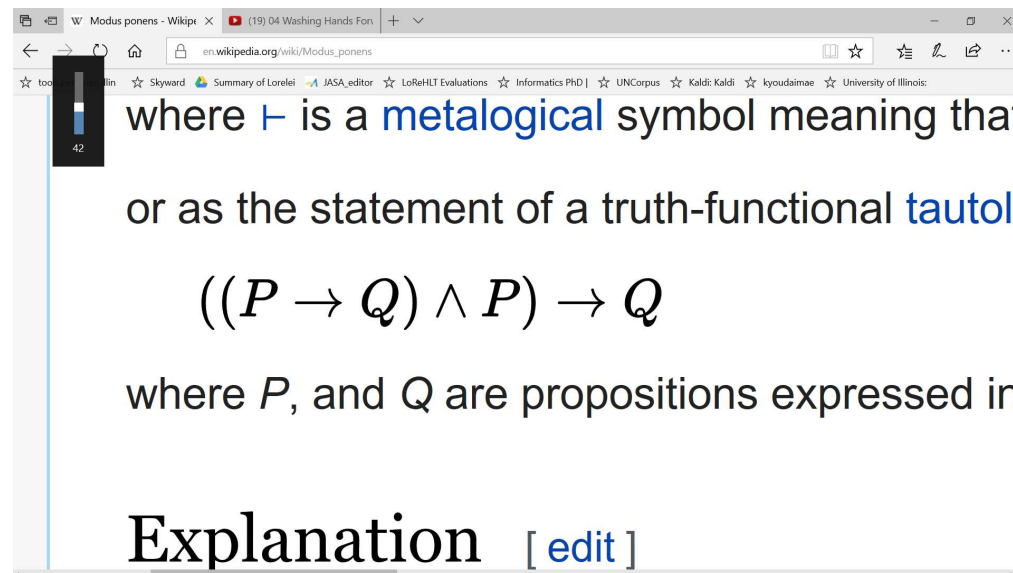


Backward chaining = start with the goal state, and search backward



Forward-chaining: each step in the search is one application of *modus ponens*

- In backward-chaining, we start from a state,  $Q$ , that we want to achieve
- We then check to see whether it is possible to apply *modus ponens* in order to generate  $Q$  from any other state,  $P$ :



A screenshot of a web browser displaying the Wikipedia page for "Modus ponens". The browser's address bar shows the URL "en.wikipedia.org/wiki/Modus\_ponens". The page content includes the text "where  $\vdash$  is a **metalogical** symbol meaning that" and "or as the statement of a truth-functional **tautology**". Below this, the logical formula 
$$((P \rightarrow Q) \wedge P) \rightarrow Q$$
 is shown. The text continues with "where  $P$ , and  $Q$  are propositions expressed in". At the bottom of the visible section, the word "Explanation" is followed by a blue "[ edit ]" link. The browser's tab bar at the top shows several open tabs, including "Modus ponens - Wikip...", "(19) 04 Washing Hands For...", and others.

where  $\vdash$  is a **metalogical** symbol meaning that

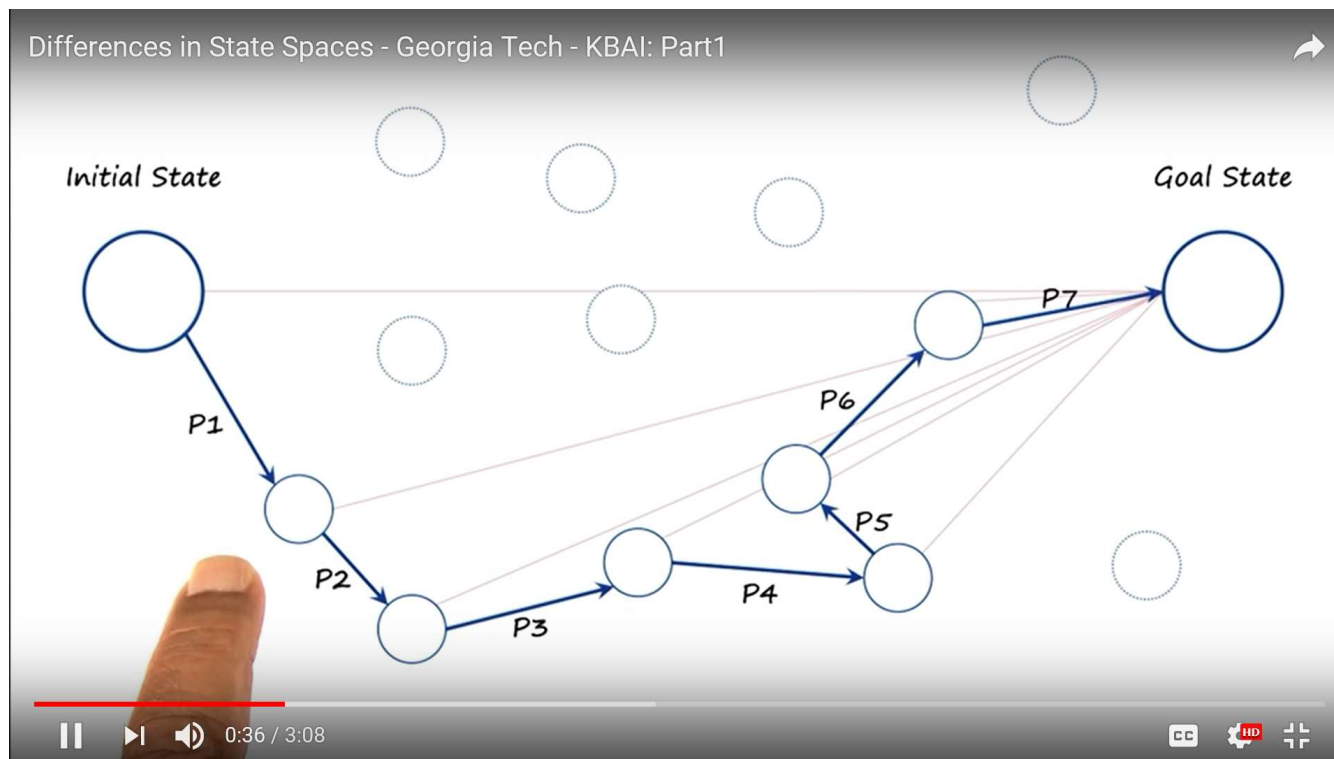
or as the statement of a truth-functional **tautology**

$$((P \rightarrow Q) \wedge P) \rightarrow Q$$

where  $P$ , and  $Q$  are propositions expressed in

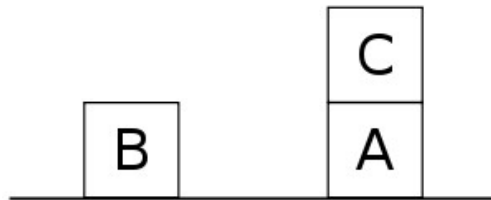
**Explanation** [ [edit](#) ]

# Heuristics for planning: means-ends analysis

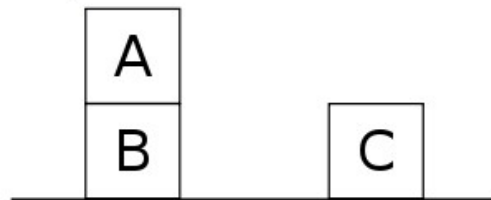


# Challenges of planning: “Sussman anomaly”

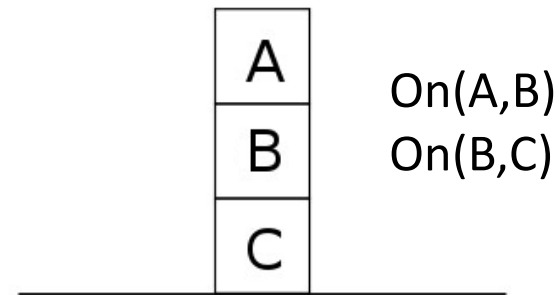
Start state:



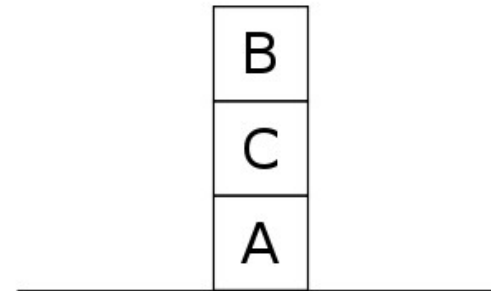
Let's try to achieve  
 $\text{On}(A,B)$ :



Goal state:



Let's try to achieve  
 $\text{On}(B,C)$ :



[http://en.wikipedia.org/wiki/Sussman\\_Anomaly](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](http://en.wikipedia.org/wiki/Sussman_Anomaly)



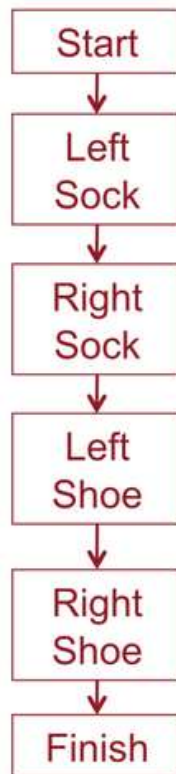
# Situation space planning vs. plan space planning

- **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

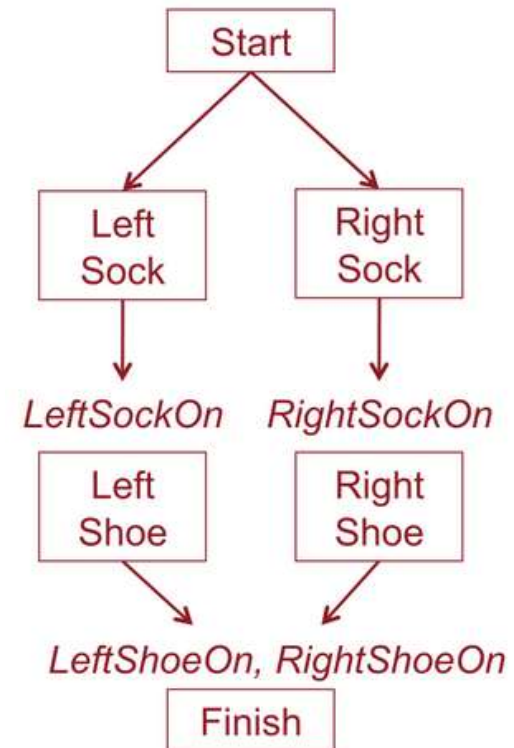
# Partial order planning

- Task: put on socks and shoes

Total order (linear) plans



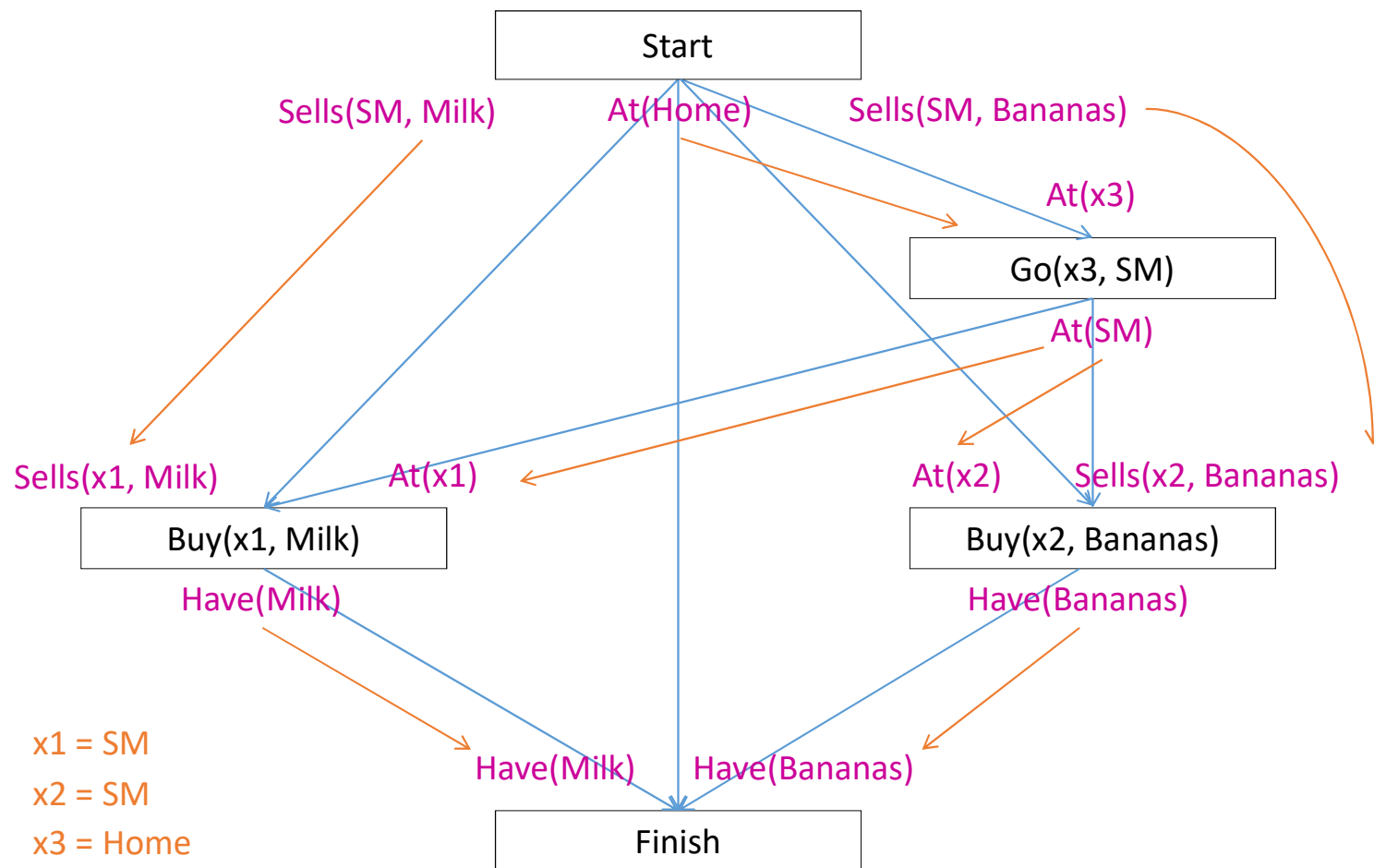
Partial order plan



# Partial Order Planning Example



# 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](mailto:riedl@cc.gatech.edu)

Twitter: [@mark\\_riedl](https://twitter.com/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>



# Application of planning: the Gale-Church alignment algorithm

Table 2  
Output from alignment program.

English	French
According to our survey, 1988 sales of mineral water and soft drinks were much higher than in 1987, reflecting the growing popularity of these products. Cola drink manufacturers in particular achieved above-average growth rates.	Quant aux eaux minérales et aux limonades, elles rencontrent toujours plus d'adeptes. En effet, notre sondage fait ressortir des ventes nettement supérieures à celles de 1987, pour les boissons à base de cola notamment.
The higher turnover was largely due to an increase in the sales volume.	La progression des chiffres d'affaires résulte en grande partie de l'accroissement du volume des ventes.
Employment and investment levels also climbed.	L'emploi et les investissements ont également augmenté.

# Application of planning: the Gale-Church alignment algorithm

1. Let  $d(x_1, y_1; 0, 0)$  be the cost of substituting  $x_1$  with  $y_1$ ,
2.  $d(x_1, 0; 0, 0)$  be the cost of deleting  $x_1$ ,
3.  $d(0, y_1; 0, 0)$  be the cost of insertion of  $y_1$ ,
4.  $d(x_1, y_1; x_2, 0)$  be the cost of contracting  $x_1$  and  $x_2$  to  $y_1$ ,

83

Computational Linguistics

Volume 19, Number 1

5.  $d(x_1, y_1; 0, y_2)$  be the cost of expanding  $x_1$  to  $y_1$  and  $y_2$ , and
6.  $d(x_1, y_1; x_2, y_2)$  be the cost of merging  $x_1$  and  $x_2$  and matching with  $y_1$  and  $y_2$ .

# 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
  - Instead of “static,” the world is “semidynamic:” we can’t think forever
- Actions at different levels of granularity: hierarchical planning
  - In order to make the depth of the search smaller, we might convert the world from “fully observable” to “partially observable”
- Contingencies: actions failing
  - Instead of being “deterministic,” maybe the world is “stochastic”
- Incorporating sensing and feedback
  - Possibly necessary to address stochastic or multi-agent environments