8. Planning

8.1 What is planning?

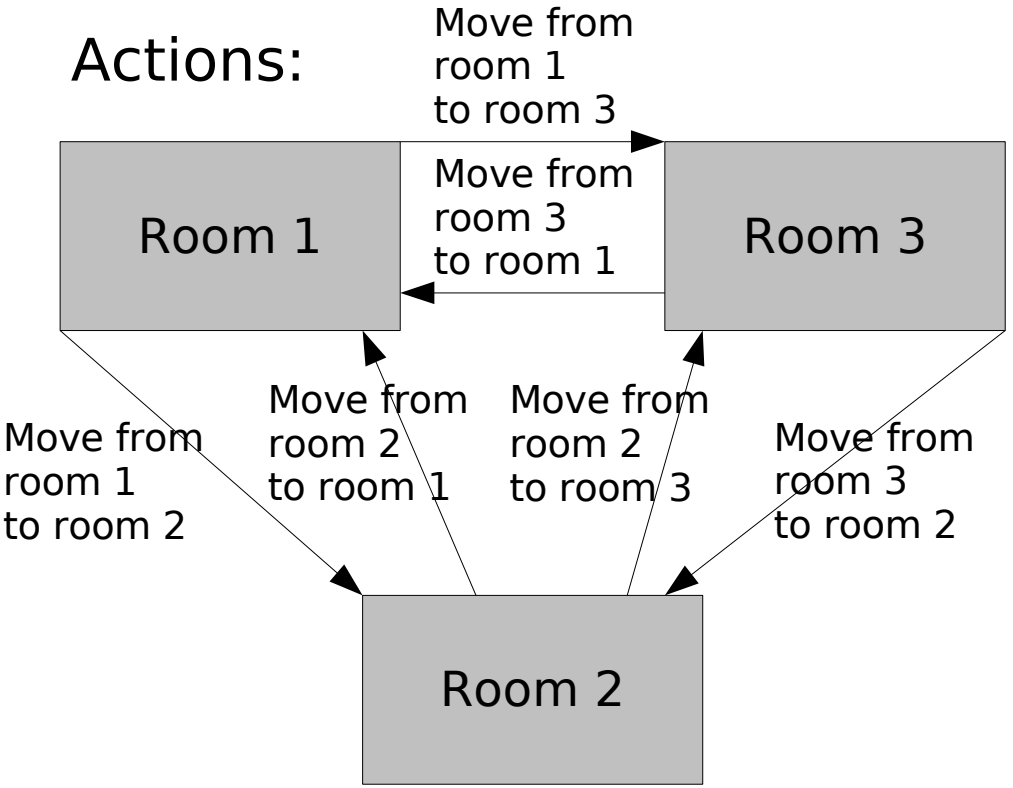
- Finding a sequence of actions that allows a problem solver accomplish a specific task.
- Is deliberate instead of reactive.
- Constructing a plan is distinct from executing the plan:
  - No connection between the order of planning and order of execution.
  - May be done by different agents.
- Applications:
  - Navigation, autonomous vehicle control, space missions
  - Manufacturing: process control
  - Language understanding; humans communicate their plans, goals and intentions with language.
- Very active and growing area of research.

8.2 Planning

- Uses much more structured knowledge than heuristic, search-based problem solver.
- Can work with any part of the solution at any given time instead from initial state towards the goal.
- Uses divide and conquer strategy $\rightarrow$ generates relevant subgoals only.
- Instead describing new states produced by actions, describe preconditions and effects of actions $\rightarrow$ complete state description not needed.
- Scales better to real-world problems than search algorithms.

8.3 Planning as search

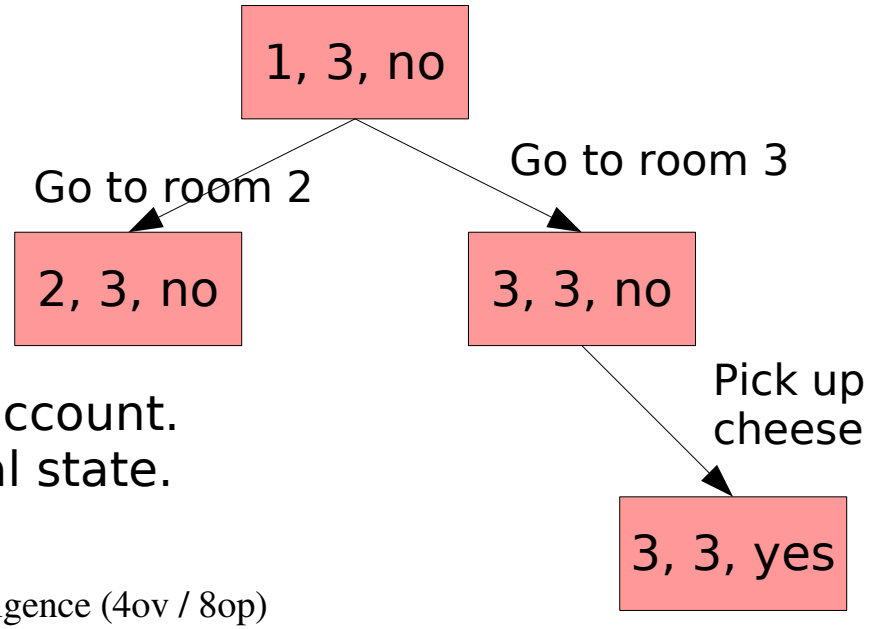
Actions:



States are three-valued vectors:

- room the robot is in
- room the cheese is in
- is robot holding the cheese?
- Example: (1,2, no), (3, 3, yes)

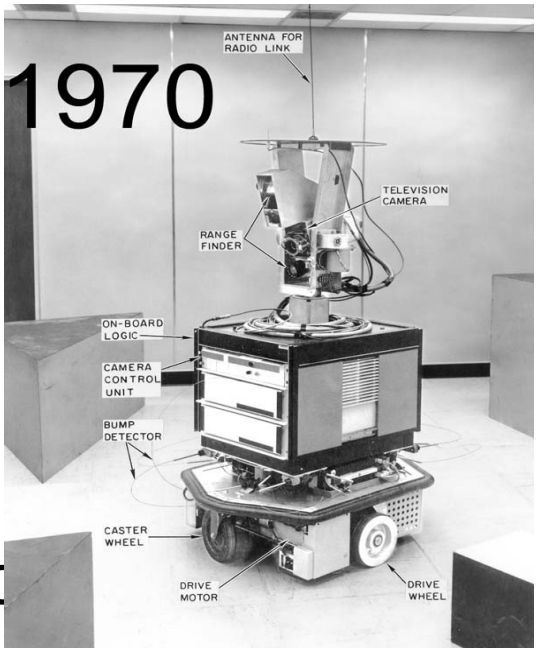
Search tree:



Problems with search:

- Does not take the effects of actions into account.
- Not always feasible to start from the initial state.

8.4 STRIPS



- STanford Research Institutes Problem Solver (Fikes and Nilsson, 1971)
- Most work involves simulation.
- Controlled a robot called Shakey that c
  - move from place to place
  - push objects
  - climb on and off of rigid objects
  - turn light switches on and off

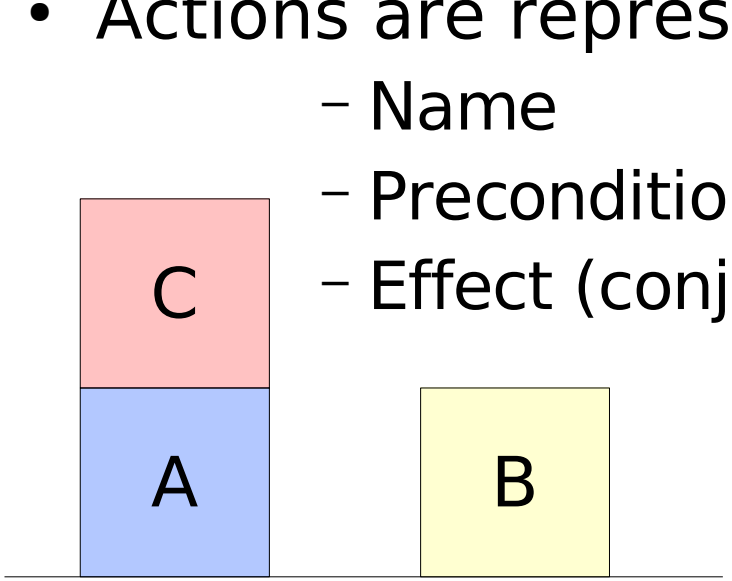
Never did these, but was able to construct plans including these actions

- Describes world with wffs in first-order predicate calculus. For instance:
  - $\forall o \forall x \forall y ((at(o,x) \wedge (x \neq y)) \rightarrow \neg at(o,y)$
- Alternately applies
  - *resolution* and *unification* to compare states
  - *means-ends* analysis to select actions:

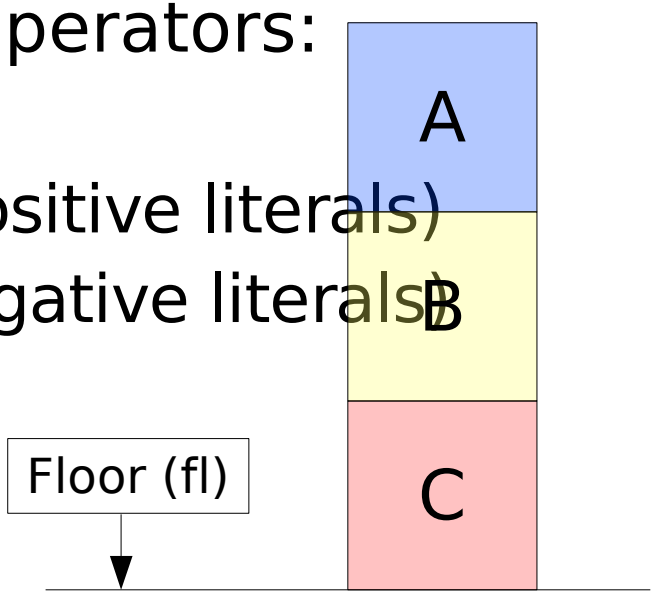
8.5 STRIPS representation

Closed-world assumption!

- *The world model* is conjunction of ground literals (no variables): $\text{on(A,fl)} \wedge \text{on(C,A)} \wedge \text{clear(C)} \wedge \text{on(B,fl)} \wedge \text{clear(B)}$.
- *Goal state* is also a conjunction of literals (may contain ∃): $\text{on(C,fl)} \wedge \text{on(B,C)} \wedge \text{on(A,B)} \wedge \text{clear(A)}$.
- Actions are represented as STRIPS operators:
  - Name
  - Precondition (conjunction of positive literals)
  - Effect (conj. of positive and negative literals)



Initial state



Goal state

8.6 STRIPS operators

- Effect defines a new state.
  - ADD list: literals to be added to the state description.
  - DELETE list: literals to be removed.
- For instance in block world:

move (X, Y, Z) {

Precondition : $on(X,Y) \wedge clear(X) \wedge clear(Z)$

Add : $on(X,Z) \wedge clear(Y)$

Delete : $clear(Z) \wedge on(X,Y)$

}

- Only a single block can be moved at the time.

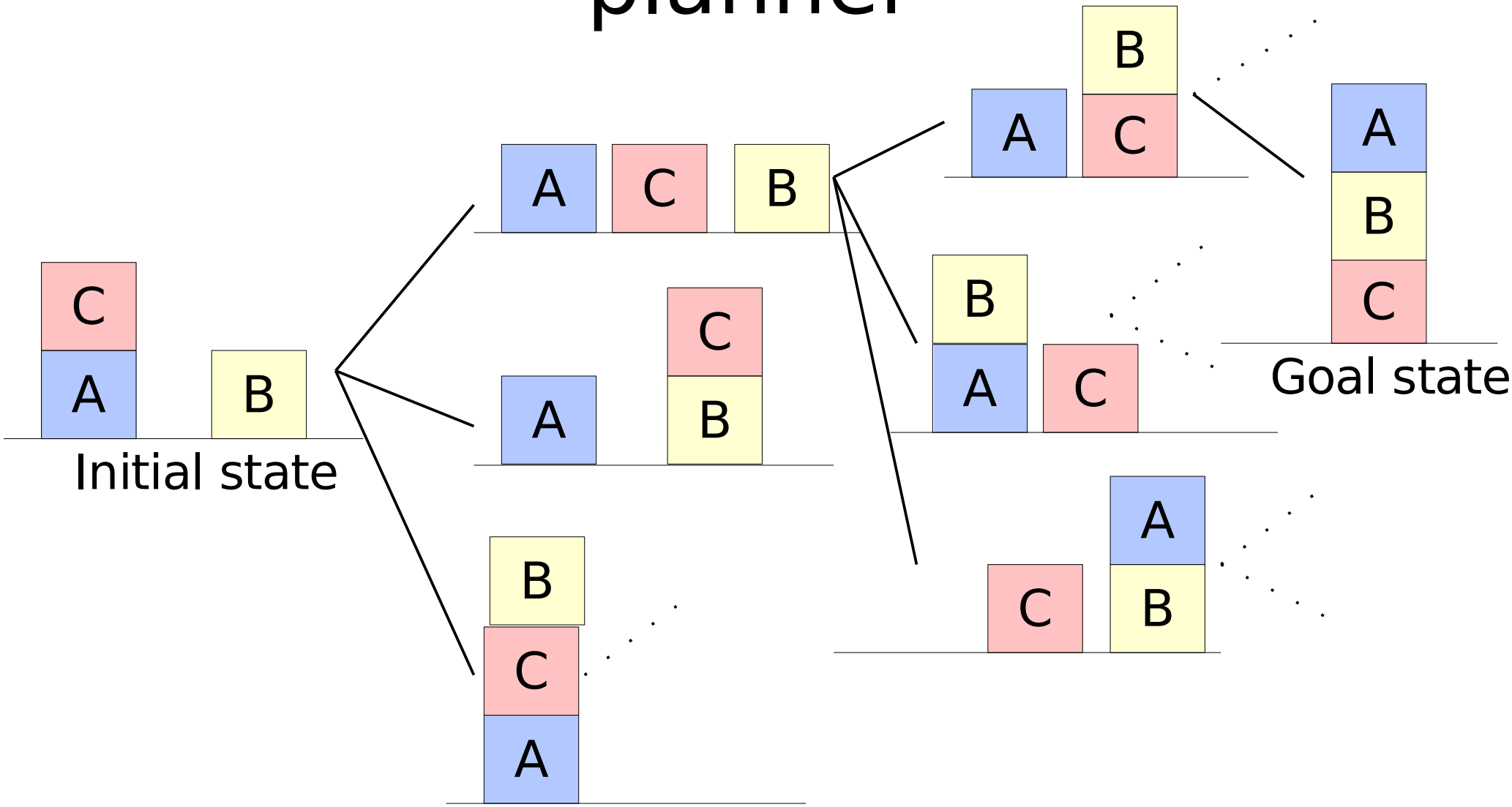
8.7 STRIPS algorithm

- First: compare the current world model to the goal state.
  - Negate the goal state
  - Show that this is inconsistent with the current model
  - If it is, i.e., the goal can be satisfied with the current model, plan is complete, else develop plan further.
- Next, select a suitable operator to move from state to (goal) state using *means-ends analysis*:
  - Compare the current state to the goal state
  - Identify the differences between the states.
  - Select the action that reduces the difference $\rightarrow$ move to next state
  - Can choose an action that is not possible in the current state, form a subgoal to move to state in which it is possible.

8.8 Situation space planners

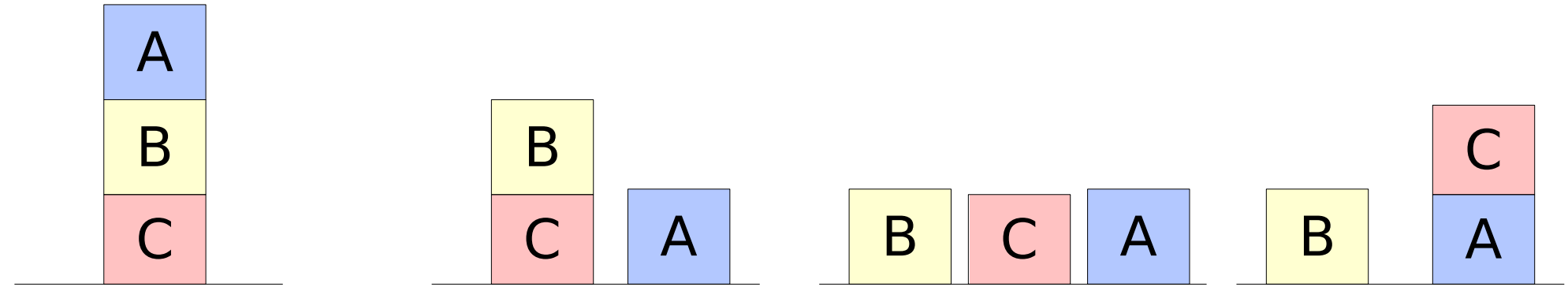
- Use STRIPS language to represent goals, conditions and effects of actions.
- Planning is search in the state space.
- Progression planner
  - Generates the plan from initial state to the goal state.
  - High branching factor
- Regression planner
  - Works backwards by generating preconditions to (goal) states and sets them as new subgoals.
  - Goal state has few conjuncts, which can be satisfied by few actions, whereas the initial state has potentially many.

8.9 Example: A progression planner



8.10 Example: A regression planner

Initial goal list: $\{on(C, fl) \wedge on(B, C) \wedge on(A, B) \wedge clear(A)\}$



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

Action
move(A,fl,B)

Add subgoal
on(A, fl)

Remove on(A,B)

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

Action
move(B,fl,C)

Add subgoal
on(B,fl)

Remove
on(B,C)

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

Action
move(C,A,fl)

Add subgoal
on(C,A)

clear(C)

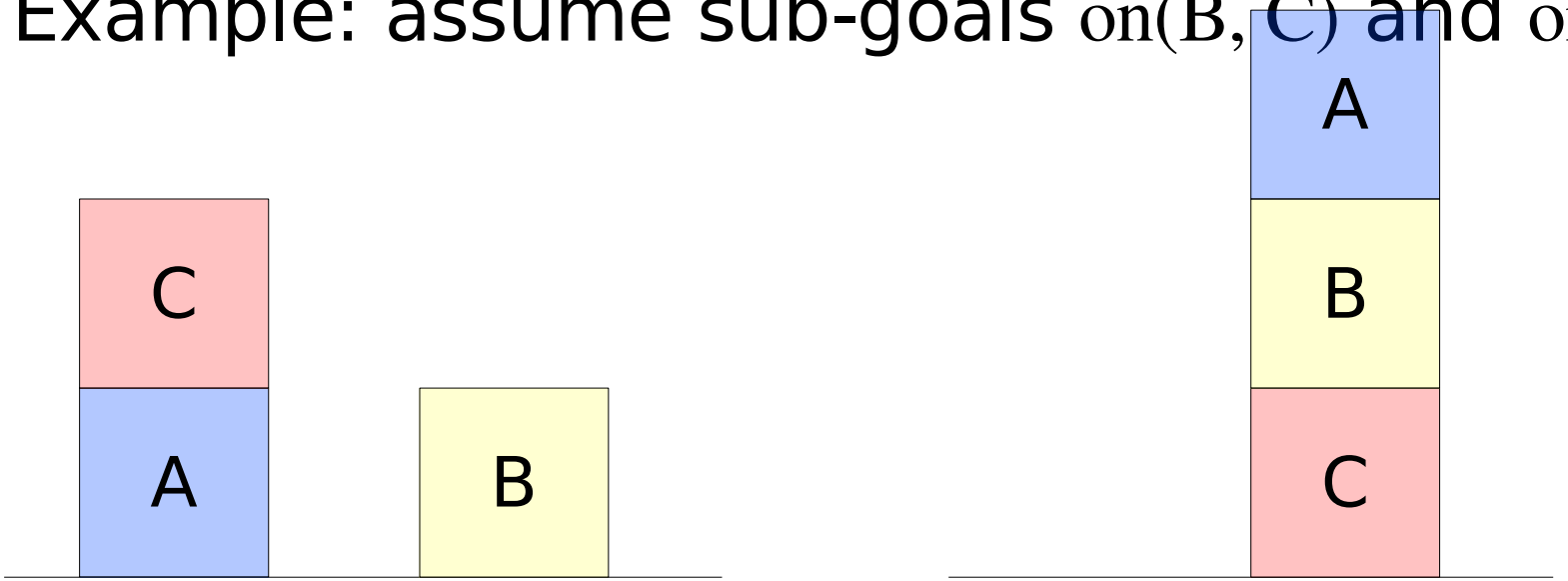
on(C,A)
on(A,fl)
on(B,fl)
clear(B)
clear(C)

8.11 Apparent problems

- What if an action has an unexpected outcome?
  - Continue regardless
  - Revise plan
- What if an action undoes effects of previous actions?
- Ordering of operators
- Frame problem
  - Anything assumed true before applying an operator, is assumed true after applying unless included in the operator's DELETE list.
  - Relation of ADD and DELETE lists is not explicit but implicit.
- Uncertainty

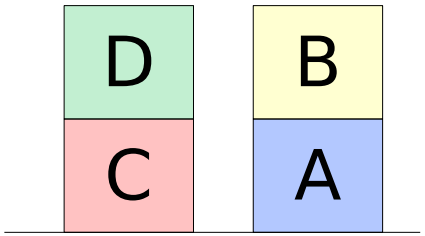
8.12 Sussman anomaly

- Problem occurs (at least in the early linear planning systems) if subgoals interact with each other.
- Solution: partial-order planning
- Example: assume sub-goals on(B, C) and on(A,B)

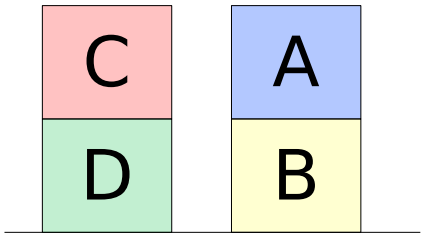


8.13 Partial order planning

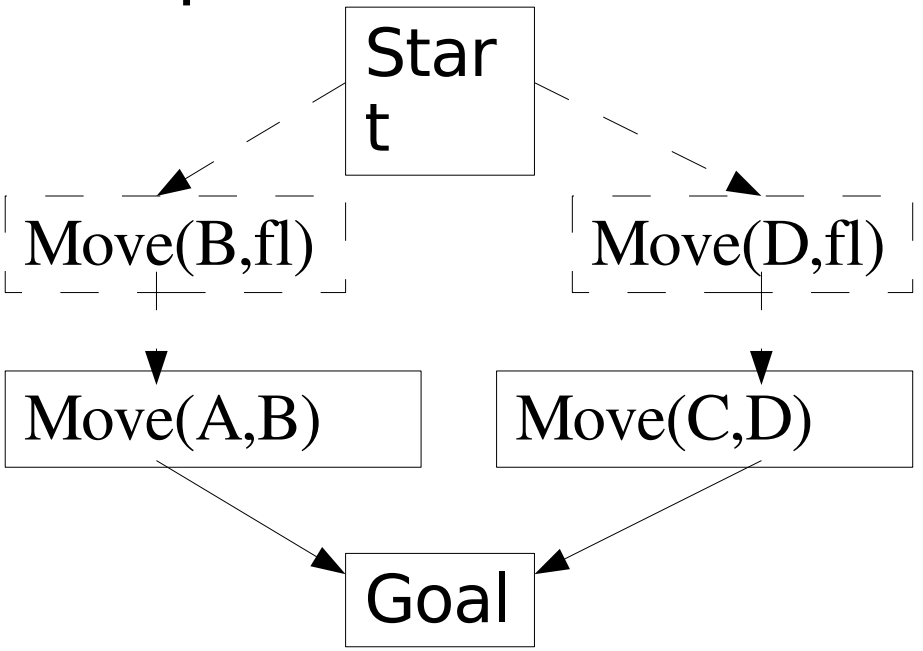
- The two previous methods search in the space of situations; they find total order plans.
- We can also use partial order planning (POP) that searches in the space of plans.
- For instance,



Initial state

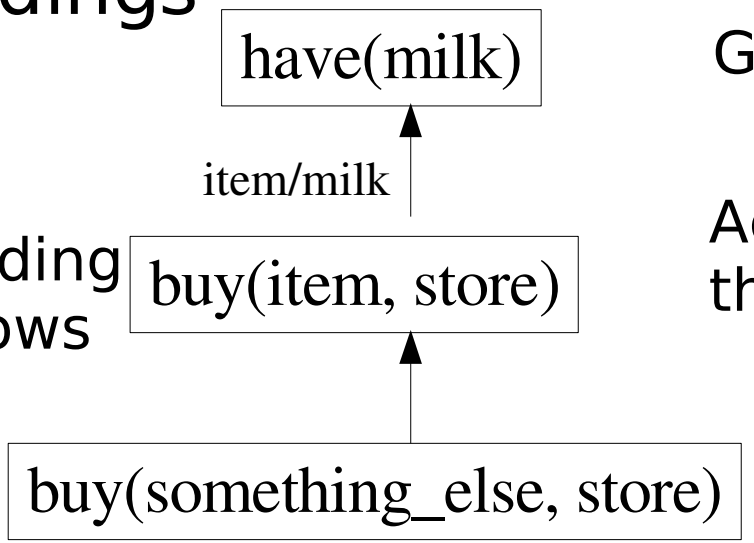


Goal state



8.14 Principle of least commitment

- Order of actions
  - Ordering of two steps left unspecified.
  - Only make choices about things that are currently cared about; otherwise likely the wrong choice is made.
- Variable bindings



Goal state

Action that achieves the goal.

Postponing the binding of variable store allows a good choice later.

8.15 Mutual exclusion

- *Mutual exclusion* (mutex) between effects of actions means they cannot be true at the same time.
- Conditions for actions to be mutex:
  - Two actions have inconsistent effects, for instance $\text{move}(A,B) \Rightarrow \text{on}(A,B)$ and $\text{move}(A,\text{fl}) \Rightarrow \text{on}(A,\text{fl})$.
  - The effect of an action interferes with the precondition of another.
  - Actions have complementary preconditions, say P and $\neg P$.
  - A proposition is inconsistent with or a negation of another proposition, for instance $\text{on}(A, B)$ is mutex with $\text{on}(A, \text{fl})$ and $\neg\text{on}(A,B)$.

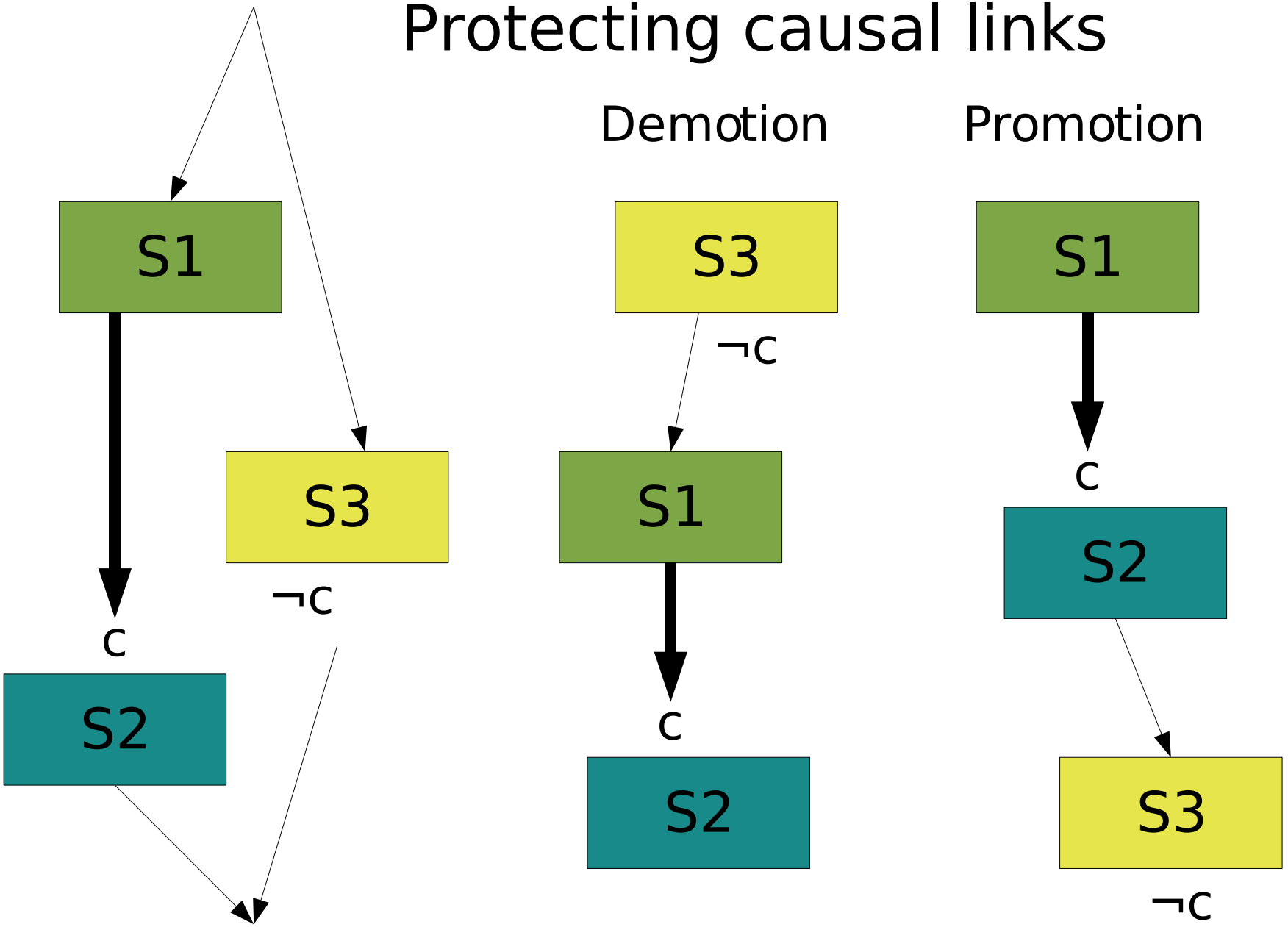
8.16 Plan components

- Two additional actions:
  - *START* with no preconditions, effect defined by the literals of the initial state.
  - *FINISH* with no effects, precondition defined by the literals of the goal state.
- *Ordering constraints* for steps $S_i \ll S_j$, meaning S_i should be accomplished before S_j.
- *Causal links* $S_i \xrightarrow{c} S_j$, meaning S_i produces c, which is S_j's precondition.
- Solution = *complete* and *consistent* plan
  - Complete = every precondition of every step is achieved.
  - Consistent = no contradictions in the ordering.

8.17 Searching for plan

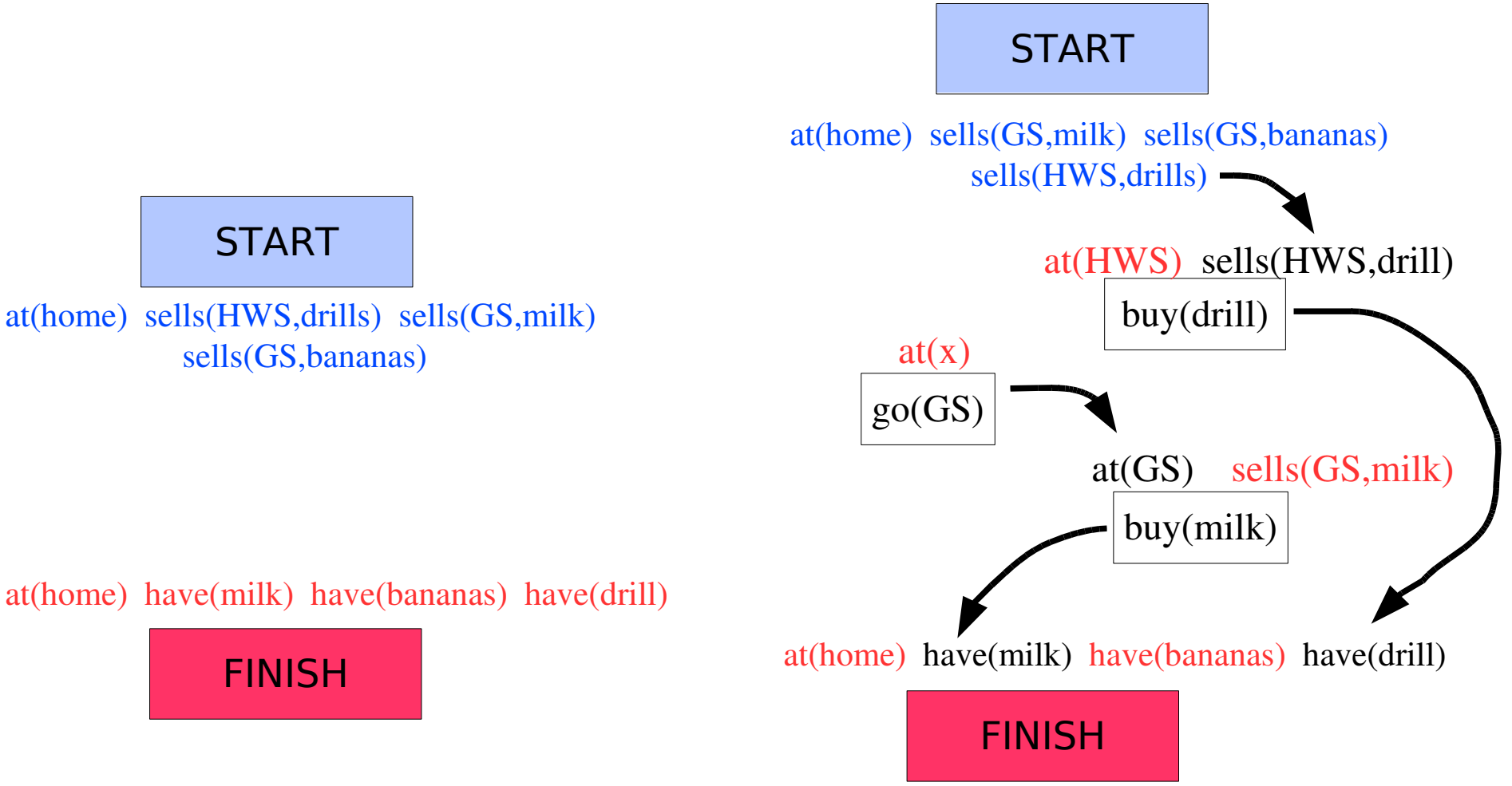
- Start with two actions START (S) and FINISH (F), ordering constraint $S \ll F$, and preconditions for F.
- LOOP
  1. Choose an open precondition p for B, and the action A that produces p.
  2. Add causal link A $\xrightarrow{p}$ B, and constraint $A \ll B$.
  3. Solve conflicts by demoting or promoting the conflicting action C, i.e., adding a constraint $C \ll A$ or $B \ll C$.
  4. If conflict cannot be solved, backtrack.
  5. If open preconditions remain, go to LOOP.

Protecting causal links

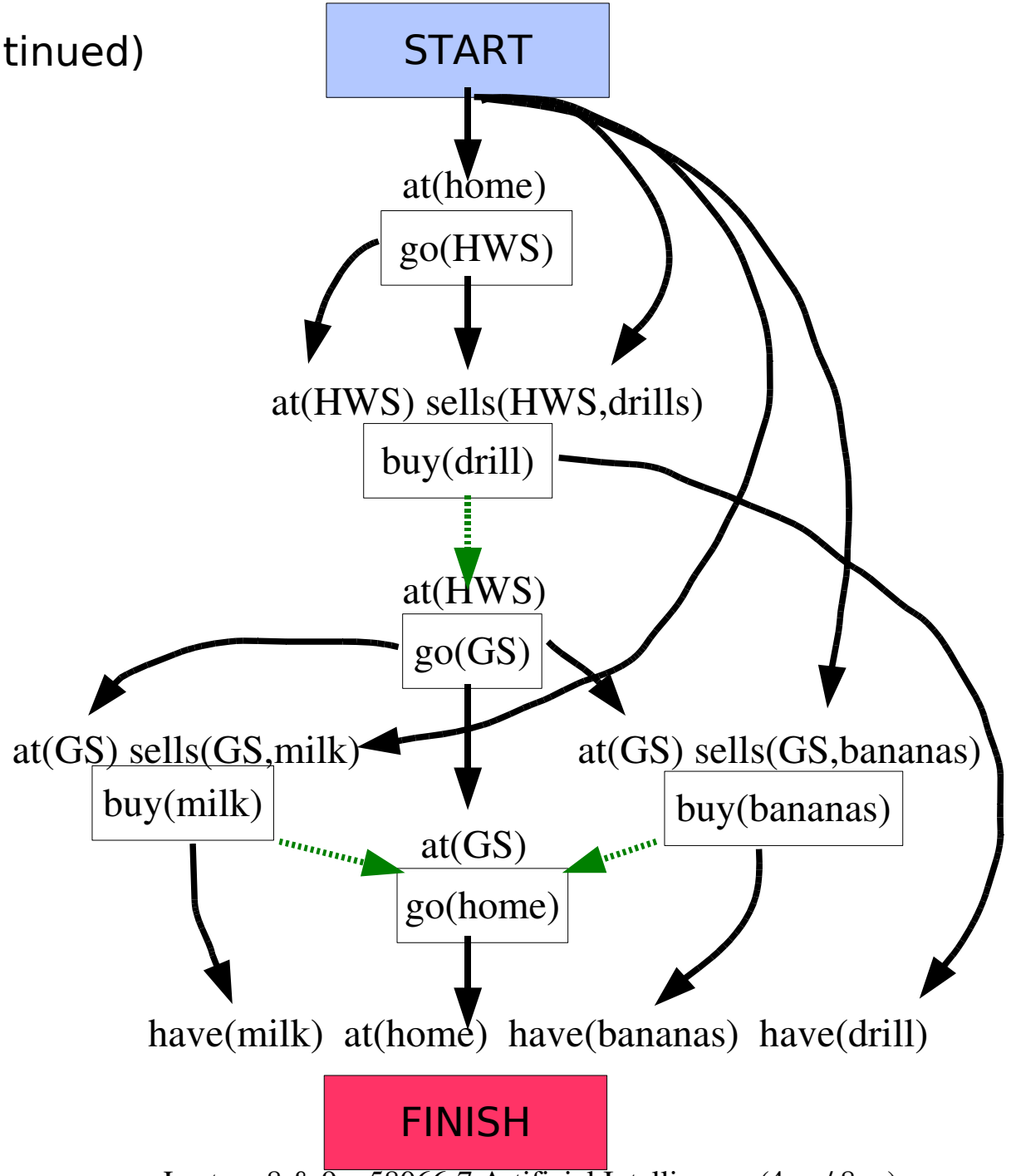


8.18 Planning example

- Goal: at(home) ∧ have(milk) ∧ have(bananas) ∧ have(drill)



(Example continued)



8.19 Traditional deductive approach

- Initial conditions together with a sequence of actions implies the goal state:

$$\exists\, x_1,y_1,z_1,x_2,y_2,z_2$$

$$\text{on}(A,B,1) \wedge \text{on}(B,fl,1) \wedge \text{clear}(a,1) \wedge$$

$$\text{move}(x_1,y_1,z_1,1) \wedge \text{move}(x_2,y_2,z_2,2)$$

$$\rightarrow \text{on}(B,A,3)$$

- Some interpretations lead to invalid plans, that allow
  - Two moves at a time.
  - changes in the world even if the precondition for action is false; not all the worlds are like block world.

8.20 Planning as satisfiability

- Set of axioms so that ANY interpretation gives a valid plan. For example the initial and goals states:
  - $\text{on}(A,B,1) \wedge \text{on}(B,fl,1) \wedge \text{clear}(A,1) \wedge \text{on}(B,A,3)$
- Similar axioms describe actions.
- Rule out the possibility of an action if its preconditions are false.
- An action implies its effect and its preconditions. For instance:
  - $\forall x,y,z,i\ \text{move}(x,y,z,i) \rightarrow (\text{clear}(x,i) \wedge \text{clear}(z,i) \wedge \text{on}(x,y,i)$

- Only one action occurs at a time:
  - $\forall x, x', y, y', z, z', i\ (x \neq x') \vee (y \neq y') \vee (z \neq z') \rightarrow \neg move(x,y,z,i) \vee \neg move(x',y',z',i)$
- Some action occurs every time:
  - $\forall i < N\ \exists x,y,z\ move(x,y,z,i)$
- Specify a “do nothing” action.

8.21 Satplan

- Planning as solving satisfiability problems.
- *Satplan*, a domain independent planning program was introduced by Kautz and Selman (1992) but did not turn out practical.
- A few years later it proved competitive with contemporary technology.
- A short surge of interest in SAT planning died out soon when heuristic methods proved superior.
- Satplan won the 2004 International planning competition.
- Reasons for improvement:
  - Satisfiability solvers are more powerful (e.g., GSAT).
  - Problems in 2004 IPC were intrinsically hard.
  - Prior problems were easily solved by heuristic forward chaining methods. 2005 →real world planning problems

8.22 Domain independent planning

- General STRIPS planning problems hard, but some domains can be solved in polynomial time.
- Efficient domain-specific planning algorithms can be learned, evolved, or programmed.
- Problems with relaxed solution criteria occur in everyday life.
- Domain independent planning:
  - Feasible but non-optimal planning easy
  - Both feasible and optimal planning hard

8.23 Real world planning

- STRIPS world
  - Deterministic actions
  - Static world
- Real world
  - Activities need resources and take time.
  - Scheduling constraints (e.g., construction work)
  - Uncertainties in actions and states
- Also we are interested in
  - Optimal action sequences
  - Solution quality

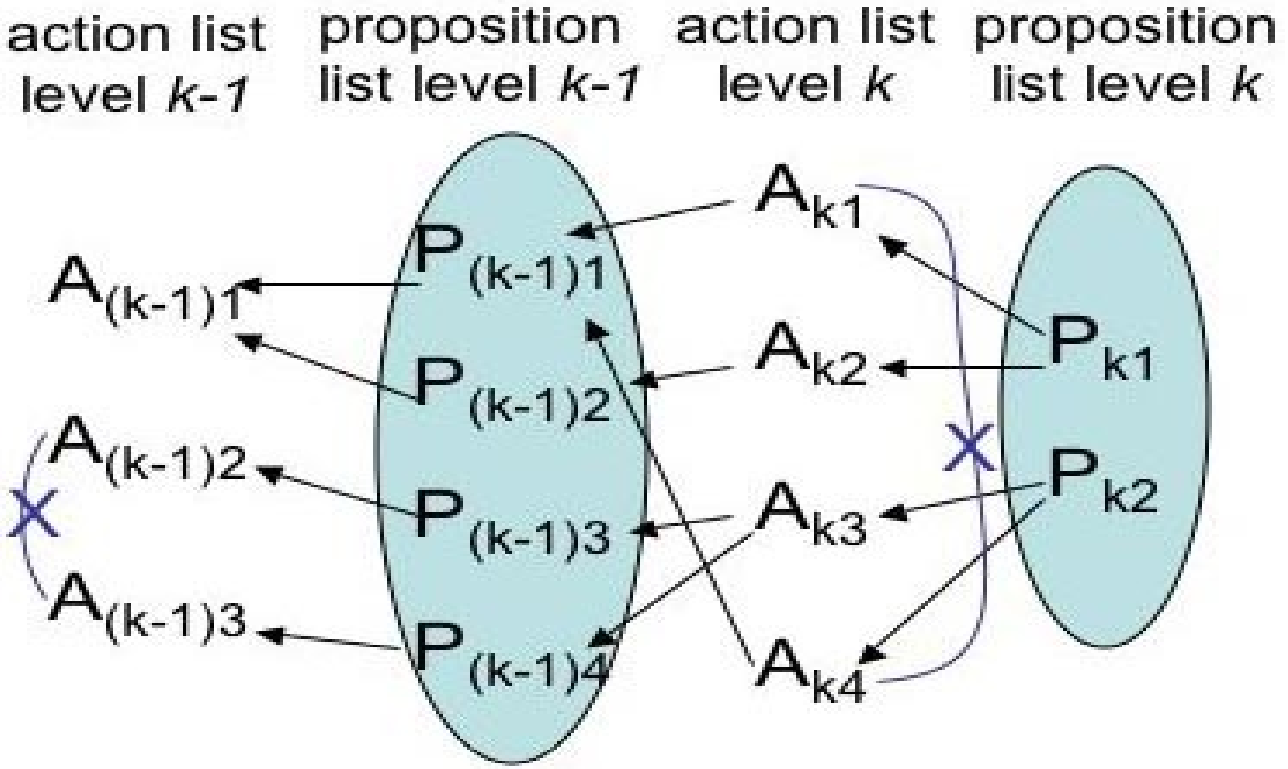
8.24 Planning graphs

- GraphPlan (Blum and Furst, 1995)
- Base of *reachability heuristics: m*eans for informative look-ahead in heuristic planners.
- Foundations in classical planning
- Flexible to adapt to more expressive scenarios with action costs, goal utility, numeric resources, time, and uncertainty.
- Relatively new development.
  - Discovered by Ghallab & Laruelle, 1994
  - Independently rediscovered by Drew McDermott (1996) and Bonet & Geffner (1999)

- Planning graph for state s is a leveled graph
  - At each level there is a proposition layer $P_i(s)$ and an action layer $A_i(s)$:
  - An action layer A_i consists of all actions that have all of their precondition proposition in P_i.
  - A proposition layer P_i consists of the set of propositions given by positive effects of action in A_{i-1}.
  - Noops (implicit actions) guarantee that propositions in P_{i-1} are retained in P_i.
- Graph is constructed until
  - it has leveled off = two consecutive levels consists of the same propositions, or
  - the goal is reachable = every goal proposition is present in a proposition layer.

8.25 Reachability heuristics

- To estimate the plan cost for a transition between two states.
- Two types: level-based and relaxed plans.



8.26 Level-based heuristics

- Cost is related to the level where proposition first appears.
- Set-level heuristic:
  - cost is associated to the index of the first proposition layer consisting of all goal propositions.
  - assumes positive interaction between subgoals: actions taken to achieve one goal help to achieve another goal (does not always hold)
- Alternatively
  - Assign cost to each individual goal proposition
  - Aggregate by maximization or summation to get the heuristic.

8.27 Relaxed plan heuristics

- Identify actions that are required to support all goals.
- Compute a plan for relaxed version of the problem
  - ignores negative effects of actions
  - captures all causal chains involved in supporting the goals.
  - by relaxing deletes, ignores some key parts of the domain structure
- Relaxed plan is a subgraph of the planning graph
  - represented by action layers, no noops and empty actions.
  - omits proposition layers

8.28 Planning models

	Unit cost	Boolean variables	Atomic time	Known state	Deterministic actions
Classical planning	yes	yes	yes	yes	yes
Cost-based planning	**no**	yes	yes	yes	yes
Planning under uncertainty	yes	yes	yes	**no** (partially)	**no** (uncertain)
Temporal planning	yes	yes	**no** (durative)	yes	yes
Planning with resources	yes	**no** (real)	yes	yes	yes

8.29 Example: Rover problem

```
(define (domain rovers_classical)
    (:requirements :strips :typing)
    (:types location data)
    (:predicates
        (at ?x – location)
        (avail ?d – data ?x – location)
        (comm ?d – data)
        (have ?d – data))
    (:action drive
        :parameters (?x ?y – location)
        :precondition (at ?x)
        :effect (and (at ?y) (not(at ?x))))
    (:action comm
        :parameters (?d  – data)
        :precondition (have ?d)
        :effect (comm ?d))
    (:action sample
        :parameters (?d  – data ?x - location)
        :precondition (and (at ?x) (avail ?d ?x)
        :effect (have?d)))
```

```
(define (problem rovers_classical1)
    (:domain rovers_classical)
    (:objects
        soil image rock – data
        alpha beta gamma -location)
        (:init     (at alpha)
                   (avail soil alpha)
                   (avail rock beta)
                   (avail image gamma))
        (:goal (and (comm soil)
                    (comm image)
                    (comm rock))))
```

Plan:

(sample(soil,alpha), commun(soil), drive(alpha,beta),sample(rock,beta), commun(rock), drive(beta,gamma), sample(image,gamma), commun(image))

8.30 Alternative planning models

- Cost-based planning
  - Propositions have achievement costs.
  - Costs to support and execute and action.
  - Propagate cost functions in planning graph.
  - Different costs at different levels.
  - Rover example: cost of driving from beta to gamma is higher than from gamma to beta.
  - The costs monotonically decrease over time
    - ← the cheapest supporter for a proposition is always chosen
    - ← the number of supporters increase with time

- Planning with resources
  - State variables have integer or real values.
  - Reachable values represented as intervals, or actually upper and lower bounds: some intermediate values may not be reachable.
  - Actions can assign, increment or decrement these values.
  - Action preconditions can be inequalities.
  - Rover example:
    - battery power
    - add new preconditions and action 'recharge'.

- Temporal planning
  - actions have real duration.
  - actions may execute concurrently.
  - Actions may have their effect in future
    - state is not only what is true now, but also what the plan commits to making true in the future.
    - durative action can have effect at any time t:

      $$t_a < t < t_a + dur(a)$$

    - all preconditions of action a must be reachable by t.
  - Rover example:
    - each action gives its effect at the end of execution.
    - communicate data while driving

- Non-deterministic planning
  - More expressive representation dealing with conditional effects:

    operator move-briefcase {

      parameters: LOCATION l_1, l_2

      preconditions: at(briefcase, l_1)

      effects:

        ADD: at(briefacse, l_2)

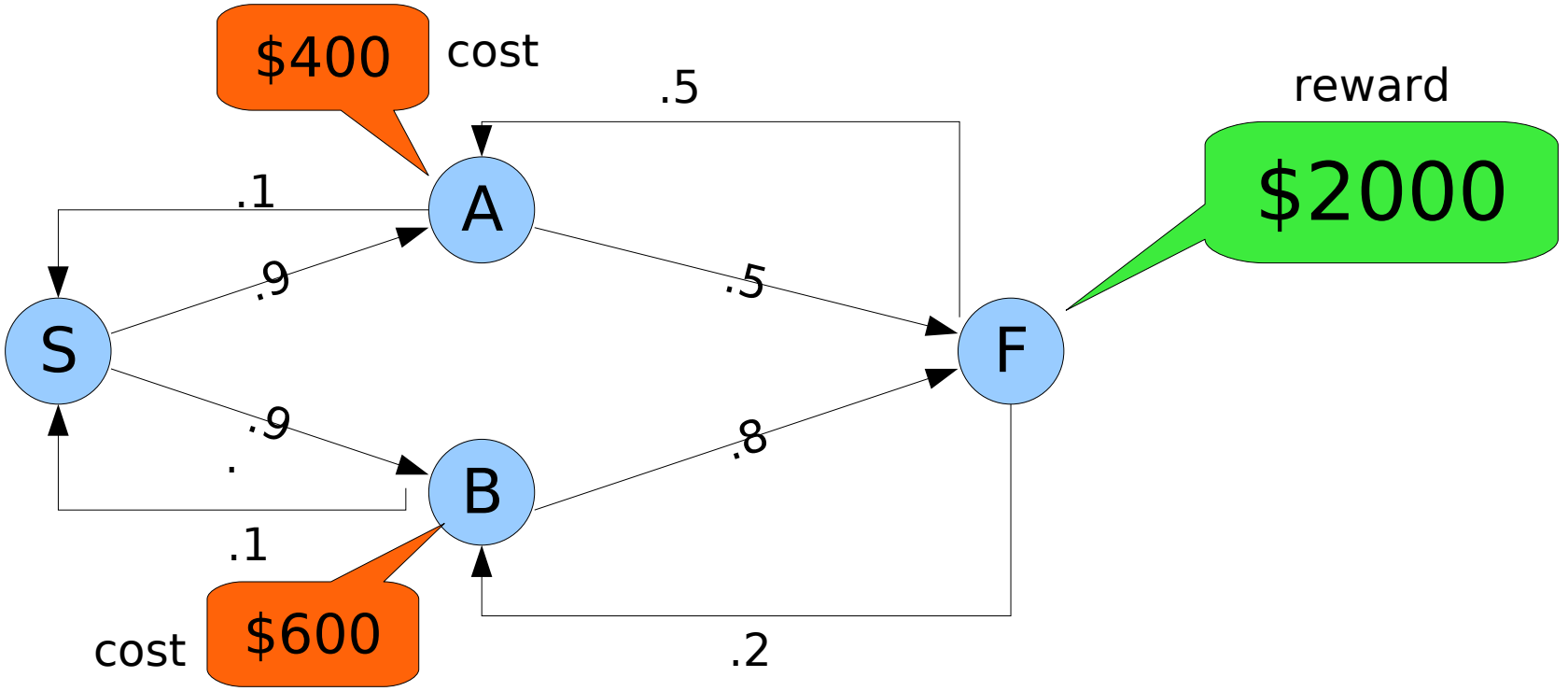
        DELETE: at(briefcase, l_1)

      conditions:

        forAll X:
        when(in(X,briefcase),ADD(at(X,l_2),DEL(at(X,l_1)))

    }

- Uncertainty in actions
  - Stochastic planning: generate a sequence that potentially leads to goal.
  - Probability distribution over states, and uncertain outcome of an action with a probability distribution.



- Uncertainty in states
  - Due to partial observations
  - Uncertain states and conditional effects of actions.
- Example:
  - Patient is either infected (I) or hydrated (H).
  - Wrong decision about medication may result in death (D).
  - One action:

```
Medicate {
    preconditions:
    effect:
        (when I → ¬I)
        (when ¬H → D)
}
```

Two possible worlds:

w1=(¬I,¬H,¬D)

w2=(I,H,¬D)

Another action:

```
Drink {
    preconditions:
    effect: H
}
```

8.31 Knowledge-based methods

- *Hierarchical task network* (HTN)
  - Activities are called *tasks*, which decompose into *primitive tasks*, that can be directly executed.
  - Plan represented as a network: nodes are tasks, edges denote ordering constraints.
  - Decomposing a task into subtasks = expanding the plan → more details revealed.
  - Conflicts are searched and resolved by *critics*.
  - A way of achieving a task = *method*.

- HTN example: Painting room