LUND UNIVERSITY

Planning

Jacek Malec
AI@CS

jacek.malec@cs.lth.se

2.04.2009

AI@CS

Department of Computing Science

LUND UNIVERSITY

Plan for today

- Search vs. planning
- STRIPS operators, ADL, PDDL
- Partial-order planning
- Graphplan

AI@CS

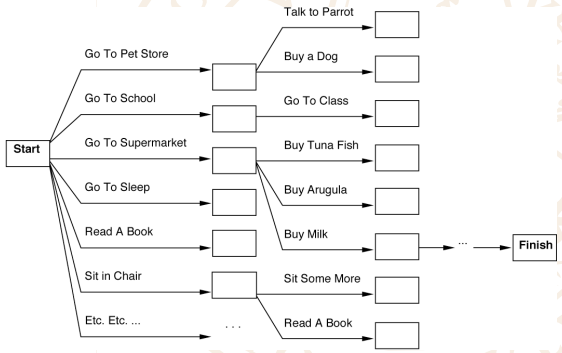
Department of Computing Science

LUND UNIVERSITY

Search vs. planning

Consider the task *get milk, bananas, and a cordless drill*

Standard search algorithms seem to fail miserably:



After-the-fact heuristic/goal test inadequate

AI@CS

Department of Computing Science

LUND UNIVERSITY

Search vs. planning contd.

Planning systems do the following:

1. open up action and goal representation to allow selection
2. divide-and-conquer by subgoaling
3. relax requirement for sequential construction of solutions

	Search	Planning
States	Python data structures	Logical sentences
Actions	Python code	Preconditions/outcomes
Goal	Python code	Logical sentence (conjunction)
Plan	Sequence from S_0	Constraints on actions

AI@CS

Department of Computing Science

Planning in situation calculus

LUND UNIVERSITY

$PlanResult(p, s)$ is the situation resulting from executing p in s

$PlanResult([], s) = s$

$PlanResult([a|p], s) = PlanResult(p, Result(a, s))$

Initial state $At(Home, S_0) \wedge \neg Have(Milk, S_0) \wedge \ldots$

Actions as Successor State axioms

$Have(Milk, Result(a, s)) \Leftrightarrow$

$[(a = Buy(Milk) \wedge At(Supermarket, s)) \vee (Have(Milk, s) \wedge a \neq \ldots)]$

Query:

$s = PlanResult(p, S_0) \wedge At(Home, s) \wedge Have(Milk, s) \wedge \ldots$

Solution:

$p =$

$[Go(Supermarket), Buy(Milk), Buy(Bananas), Go(HWS), \ldots]$

Principal difficulty: unconstrained branching, hard to apply heuristics

AI@CS

Department of Computing Science

STRIPS operators

LUND UNIVERSITY

Properly arranged actions descriptions, restricted language

ACTION: $Buy(x)$

PRECONDITION: $At(p), Sells(p, x)$

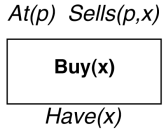
EFFECT: $Have(x)$

[Note: this abstracts away many important details!]

Restricted language ⇒ efficient algorithm

Precondition: conjunction of positive literals

Effect: conjunction of literals



AI@CS

Department of Computing Science

STRIPS vs ADL

LUND UNIVERSITY

STRIPS - Stanford Reserch Institute Problem Solver

ADL - Action Description Language

PDDL - Planning Domain Description Language

	STRIPS	ADL
State	Positive literals	Positive and negative literals
CWA	Present	Absent
Preconditions	positive literals	Logical sentences
Effects	$P \wedge \neg Q$ add, delete	$P \wedge \neg Q$ add all this
Goals	ground literals	quantified formulae
Goals	conjunction	arbitrary
Effects	conjunction	conditionals allowed

AI@CS

Department of Computing Science

Partially ordered plans

LUND UNIVERSITY

Partially ordered collection of steps with

- *Start step* has the initial state description as its effect
- *Finish step* has the goal description as its precondition
- *causal links* from outcome of one step to precondition of another
- *temporal ordering* between pairs of steps

Definition: *open condition* is a precondition of a step not yet causally linked

AI@CS

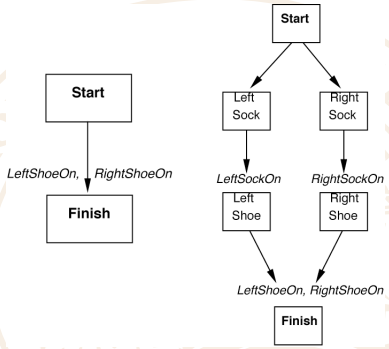
Department of Computing Science

LUND UNIVERSITY

AI@CS

Department of Computing Science

Partially ordered plans



A plan is *complete* iff every precondition is achieved

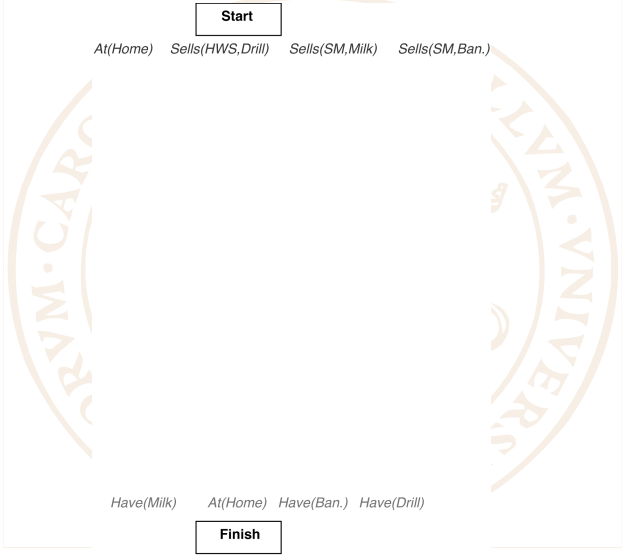
A precondition is *achieved* iff it is the effect of an earlier step
and no *possibly intervening* step undoes it

LUND UNIVERSITY

AI@CS

Department of Computing Science

Example

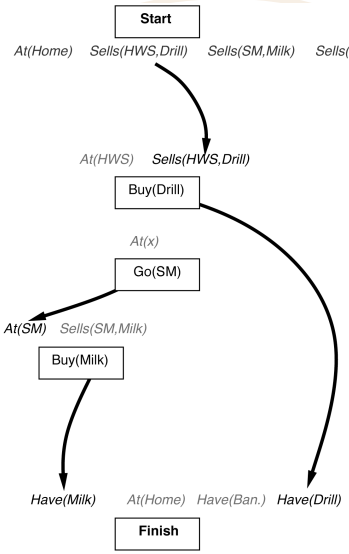


LUND UNIVERSITY

AI@CS

Department of Computing Science

Example

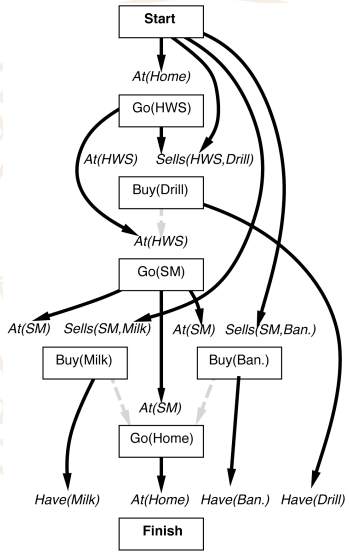


LUND UNIVERSITY

AI@CS

Department of Computing Science

Example



Planning process

Operators on partial plans:
add a link from an existing action to an open condition
add a step to fulfill an open condition
order one step wrt another

Gradually move from incomplete/vague plans to complete, correct plans

Backtrack if an open condition is unachievable or if a conflict is unresolvable

POP algorithm sketch

```
function POP(initial, goal, operators) returns plan
    plan ← MAKE-MINIMAL-PLAN(initial, goal)
    loop do
        if SOLUTION?(plan) then return plan
        S_need, c ← SELECT-SUBGOAL(plan)
        CHOOSE-OPERATOR(plan, operators, S_need, c)
        RESOLVE-THREATS(plan)
    end

function SELECT-SUBGOAL(plan) returns S_need, c
    pick a plan step S_need from STEPS(plan)
        with a precondition c that has not been achieved
    return S_need, c
```

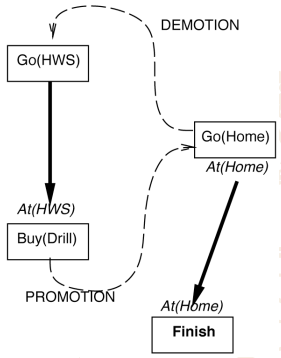

POP algorithm contd.

```
procedure CHOOSE-OPERATOR(plan, operators, S_need, c)
    choose a step S_add from operators or STEPS(plan) that has c as an effect
    if there is no such step then fail
    add the causal link S_add --c--> S_need to LINKS(plan)
    add the ordering constraint S_add ≺ S_need to ORDERINGS(plan)
    if S_add is a newly added step from operators then
        add S_add to STEPS(plan)
        add Start ≺ S_add ≺ Finish to ORDERINGS(plan)

procedure RESOLVE-THREATS(plan)
    for each S_threat that threatens a link S_i --c--> S_j in LINKS(plan) do
        choose either
            Demotion: Add S_threat ≺ S_i to ORDERINGS(plan)
            Promotion: Add S_j ≺ S_threat to ORDERINGS(plan)
        if not CONSISTENT(plan) then fail
    end
```


Clobbering and promotion/demotion

A *clobberer* is a potentially intervening step that destroys the condition achieved by a causal link. E.g., $Go(Home)$ clobbers $At(HWS)$:



Demotion: put before $Go(HWS)$

Promotion: put after $Buy(Drill)$

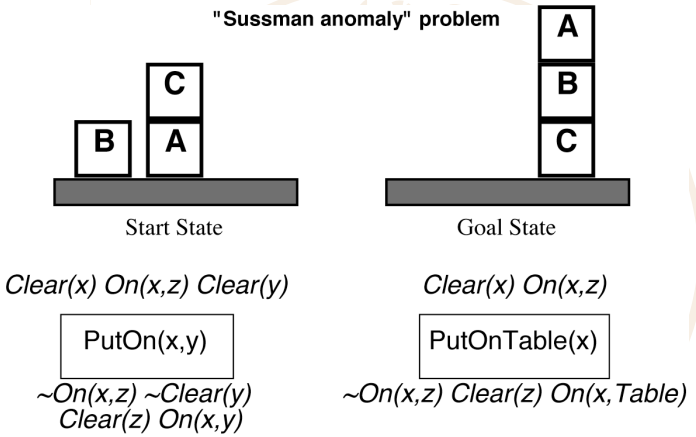
Properties of POP

Nondeterministic algorithm: backtracks at *choice* points on failure:

- choice of S_{add} to achieve S_{need}
- choice of demotion or promotion for clobberer
- selection of S_{need} is irrevocable

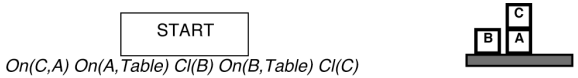
POP is sound, complete, and *systematic* (no repetition)
Extensions for disjunction, universals, negation, conditionals
Can be made efficient with good heuristics derived from problem description
Particularly good for problems with many loosely related subgoals

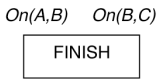
Example: Blocks world

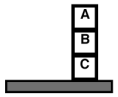


+ several inequality constraints

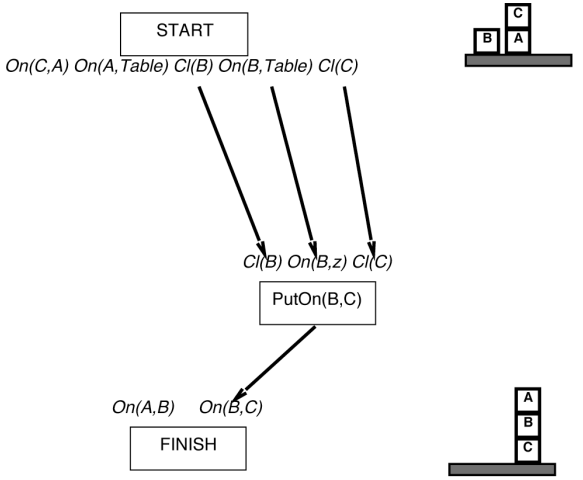
Example contd.







Example contd.

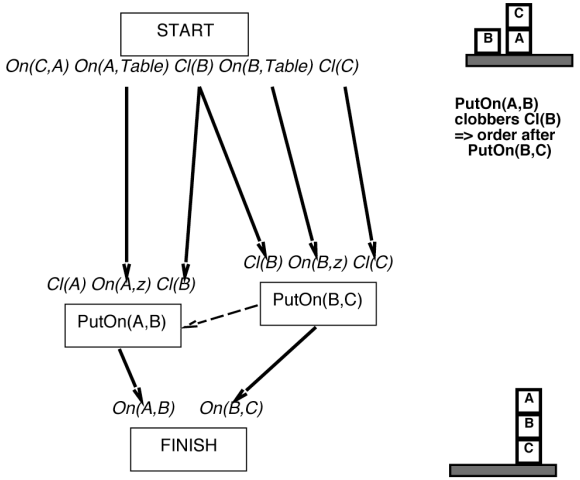


LUND UNIVERSITY

AI@CS

Department of Computing Science

Example contd.



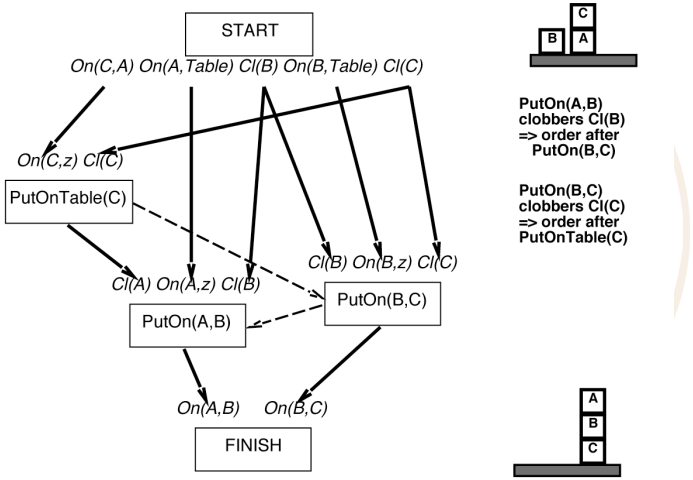
PutOn(A,B) clobbers Cl(B) => order after PutOn(B,C)

LUND UNIVERSITY

AI@CS

Department of Computing Science

Example contd.



PutOn(A,B) clobbers Cl(B) => order after PutOn(B,C)

PutOn(B,C) clobbers Cl(C) => order after PutOnTable(C)