
Introduction to Artificial Intelligence

Planning

Bernhard Beckert



UNIVERSITÄT KOBLENZ-LANDAU

Winter Term 2004/2005

Outline

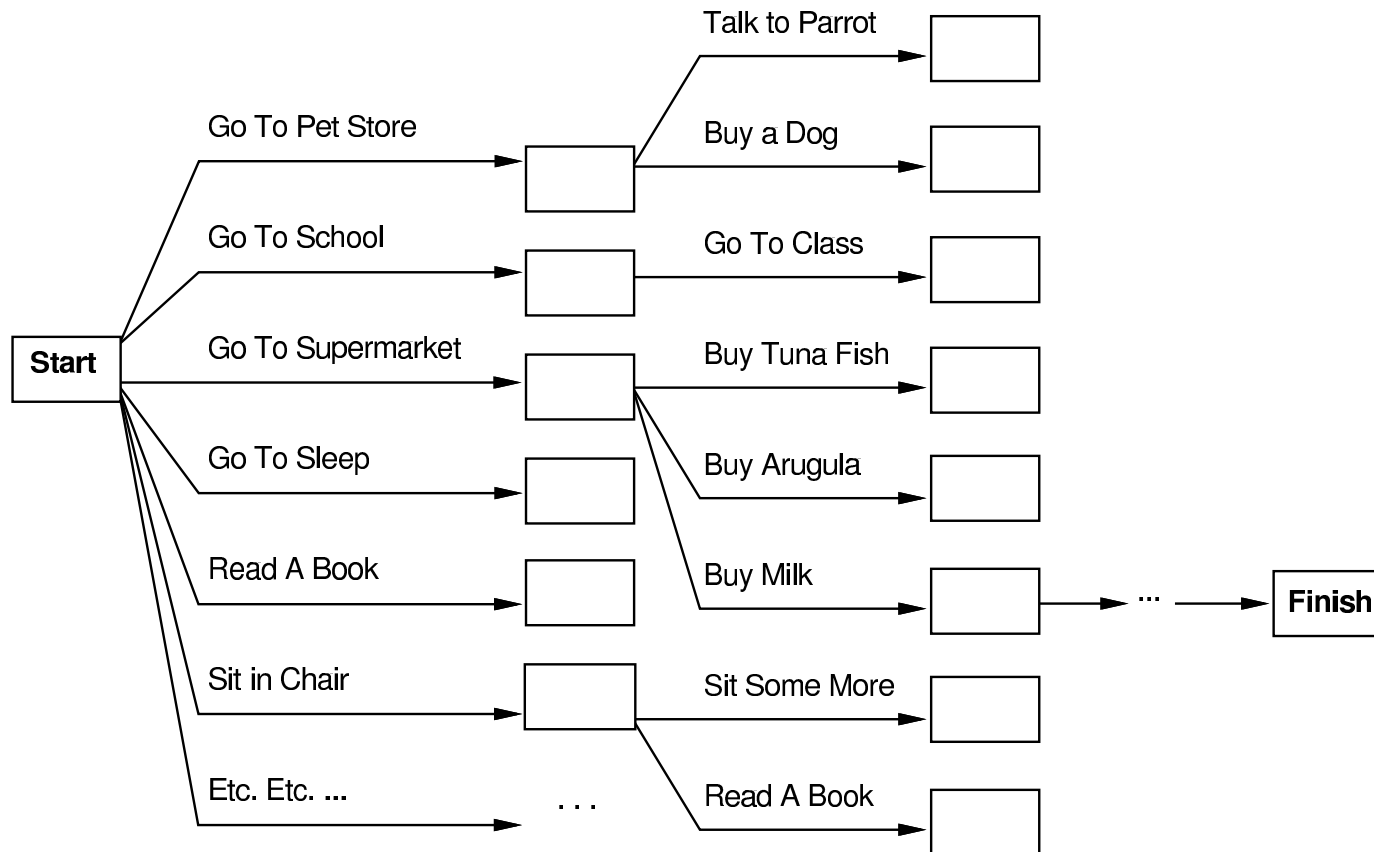
- **Search vs. planning**
- **STRIPS operators**
- **Partial-order planning**
- **The real world**
- **Conditional planning**
- **Monitoring and replanning**

Search vs. Planning

Consider the following task

Get milk, bananas, and a cordless drill

Standard search algorithms seem to fail miserably



Search vs. Planning

- **Actions have requirements & consequences that should constrain applicability in a given state**
 - ⇒ **stronger interaction between actions and states needed**

Search vs. Planning

- 🟡 **Actions have requirements & consequences that should constrain applicability in a given state**
 - ⇒ **stronger interaction between actions and states needed**
- 🟡 **Most parts of the world are independent of most other parts**
 - ⇒ **solve subgoals independently**

Search vs. Planning

- **Actions have requirements & consequences that should constrain applicability in a given state**
⇒ **stronger interaction between actions and states needed**
- **Most parts of the world are independent of most other parts**
⇒ **solve subgoals independently**
- **Human beings plan goal-directed; they construct important intermediate solutions first**
⇒ **flexible sequence for construction of solution**

Search vs. Planning

Planning systems do the following

- **Unify action and goal representation to allow selection (use logical language for both)**
- **Divide-and-conquer by subgoalings**
- **Relax requirement for sequential construction of solutions**

STRIPS

STRIPS

STandford Research Institute Problem Solver

- Tidily arranged actions descriptions
- Restricted language (function-free literals)
- Efficient algorithms

STRIPS: States

States represented by:

Conjunction of ground (function-free) atoms

Example

$At(Home), Have(Bread)$

STRIPS: States

States represented by:

Conjunction of ground (function-free) atoms

Example

$At(Home), Have(Bread)$

Closed world assumption

Atoms that are not present are assumed to be false

Example

State: $At(Home), Have(Bread)$

Implicitly: $\neg Have(Milk), \neg Have(Bananas), \neg Have(Drill)$

STRIPS: Operators

Operator description consists of:

Action name	Positive literal	<i>Buy(Milk)</i>
Precondition	Conjunction of positive literals	$At(Shop) \wedge Sells(Shop, Milk)$
Effect	Conjunction of literals	<i>Have(Milk)</i>

STRIPS: Operators

Operator description consists of:

Action name	Positive literal	$Buy(Milk)$
Precondition	Conjunction of positive literals	$At(Shop) \wedge Sells(Shop, Milk)$
Effect	Conjunction of literals	$Have(Milk)$

Operator schema

Operator containing variables

$At(p) \quad Sells(p, x)$

Buy(x)

$Have(x)$

STRIPS: Operator Application

Operator applicability

Operator o applicable in state s if:

there is substitution $Subst$ of the free variables such that

$$Subst(precond(o)) \subseteq s$$

STRIPS: Operator Application

Operator applicability

Operator o applicable in state s if:
there is substitution $Subst$ of the free variables such that

$$Subst(precond(o)) \subseteq s$$

Example

$Buy(x)$ is applicable in state

$$At(Shop) \wedge Sells(Shop, Milk) \wedge Have(Bread)$$

with substitution

$$Subst = \{ p/Shop, x/Milk \}$$

$At(p) \ Sells(p, x)$

Buy(x)

$Have(x)$

STRIPS: Operator Application

Resulting state

Computed from old state and literals in $Subst(effect)$

- Positive literals are added to the state
- Negative literals are removed from the state
- All other literals remain unchanged
(avoids the frame problem)

STRIPS: Operator Application

Resulting state

Computed from old state and literals in $Subst(effect)$

- Positive literals are added to the state
- Negative literals are removed from the state
- All other literals remain unchanged
(avoids the frame problem)

Formally

$$s' = (s \cup \{P \mid P \text{ a positive atom, } P \in Subst(effect(o))\}) \\ \setminus \{P \mid P \text{ a positive atom, } \neg P \in Subst(effect(o))\}$$

STRIPS: Operator Application

Example

Application of

Drive(a, b)

precond: $At(a), Road(a, b)$

effect: $At(b), \neg At(a)$

STRIPS: Operator Application

Example

Application of

Drive(a, b)

precond: $At(a), Road(a, b)$

effect: $At(b), \neg At(a)$

to state

$At(Koblenz), Road(Koblenz, Landau)$

STRIPS: Operator Application

Example

Application of

Drive(a, b)

precond: $At(a), Road(a, b)$

effect: $At(b), \neg At(a)$

to state

$At(Koblenz), Road(Koblenz, Landau)$

results in

$At(Landau), Road(Koblenz, Landau)$

State Space vs. Plan Space

Planning problem

Find a sequence of actions that make instance of the goal true

State Space vs. Plan Space

Planning problem

Find a sequence of actions that make instance of the goal true

Nodes in search space

Standard search: node = concrete world state

Planning search: node = partial plan

State Space vs. Plan Space

Planning problem

Find a sequence of actions that make instance of the goal true

Nodes in search space

Standard search: node = concrete world state

Planning search: node = partial plan

(Partial) Plan consists of

- Set of operator applications S_i
- Partial (temporal) order constraints $S_i \prec S_j$
- Causal links $S_i \xrightarrow{c} S_j$

Meaning: “ S_i achieves $c \in \text{precond}(S_j)$ ” (record purpose of steps)

State Space vs. Plan Space

Operators on partial plans

- add an action and a causal link to achieve an open condition
- add a causal link from an existing action to an open condition
- add an order constraint to order one step w.r.t. another

Open condition

A precondition of an action not yet causally linked

State Space vs. Plan Space

Operators on partial plans

- add an action and a causal link to achieve an open condition
- add a causal link from an existing action to an open condition
- add an order constraint to order one step w.r.t. another

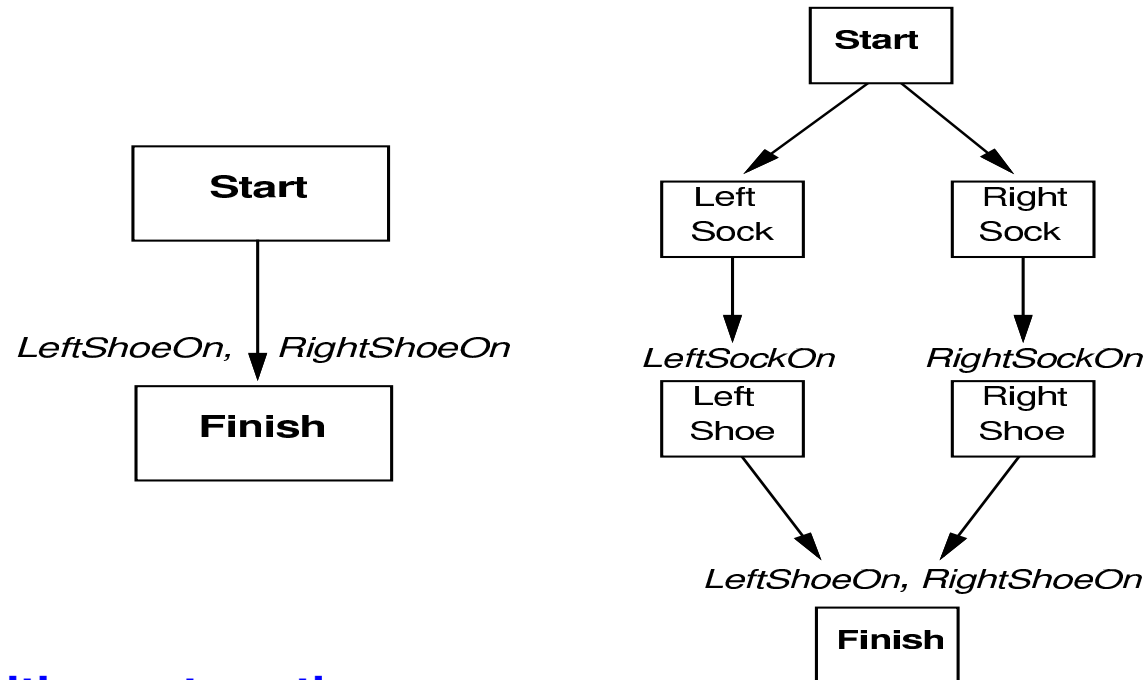
Open condition

A precondition of an action not yet causally linked

Note

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

Partially Ordered Plans

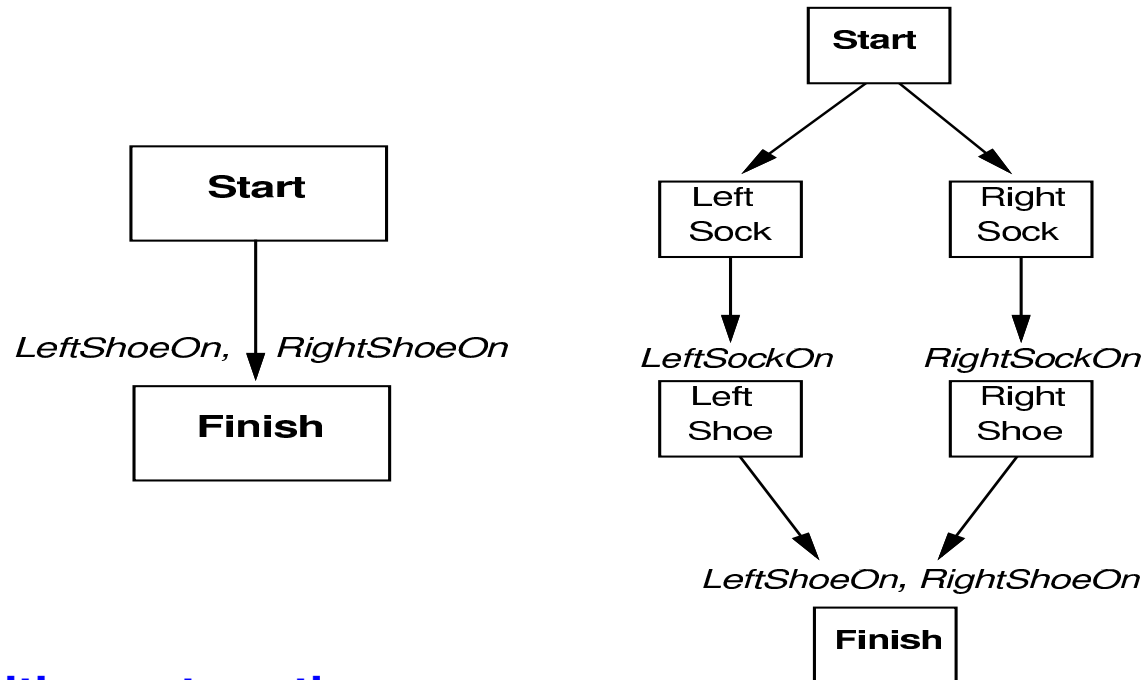


Special steps with empty action

Start no precondition, initial assumptions as effect)

Finish goal as precondition, no effect

Partially Ordered Plans



Special steps with empty action

Start no precondition, initial assumptions as effect)

Finish goal as precondition, no effect

Note

Different paths in partial plan are *not* alternative plans,
but alternative sequences of actions

Partially Ordered Plans

Complete plan

A plan is complete iff every precondition is achieved

Partially Ordered Plans

Complete plan

A plan is complete iff every precondition is achieved

A precondition c of a step S_j is achieved (by S_i) if

- $S_i \prec S_j$
- $c \in effect(S_i)$
- there is no S_k with $S_i \prec S_k \prec S_j$ and $\neg c \in effect(S_k)$
(otherwise S_k is called a **clobberer** or **threat**)

Partially Ordered Plans

Complete plan

A plan is complete iff every precondition is achieved

A precondition c of a step S_j is achieved (by S_i) if

- $S_i \prec S_j$
- $c \in effect(S_i)$
- there is no S_k with $S_i \prec S_k \prec S_j$ and $\neg c \in effect(S_k)$
(otherwise S_k is called a **clobberer** or **threat**)

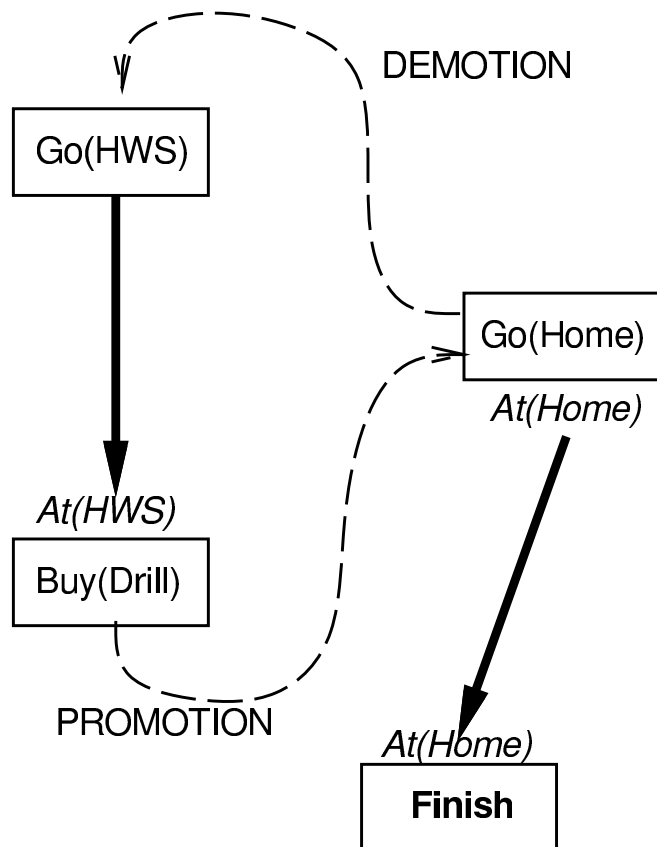
Clobberer / threat

A potentially intervening step that destroys the condition achieved by a causal link

Clobbering and Promotion/Demotion

Example

$Go(Home)$ **clobbers** $At(HWS)$



Demotion

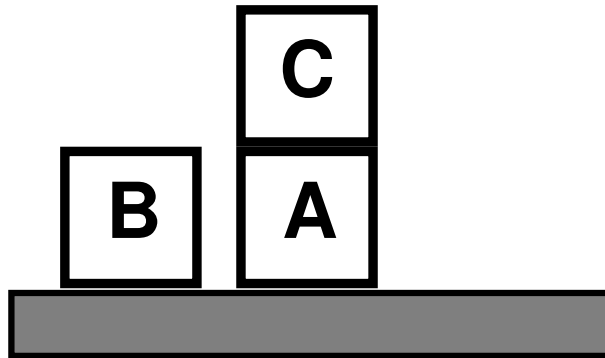
Put before $Go(HWS)$

Promotion

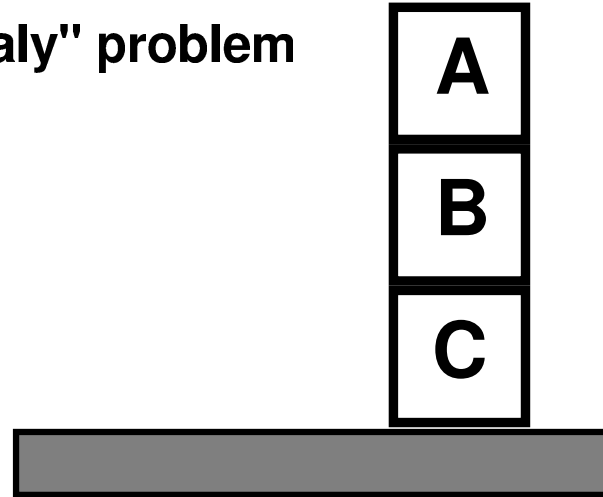
Put after $Buy(Drill)$

Example: Blocks world

"Sussman anomaly" problem



Start State



Goal State

$Clear(x) \ On(x,z) \ Clear(y)$

PutOn(x,y)

$\sim On(x,z) \ \sim Clear(y)$
 $Clear(z) \ On(x,y)$

$Clear(x) \ On(x,z)$

PutOnTable(x)

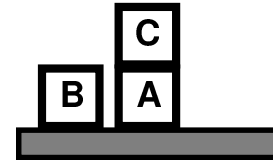
$\sim On(x,z) \ Clear(z) \ On(x, Table)$

+ several inequality constraints

Example: Blocks World

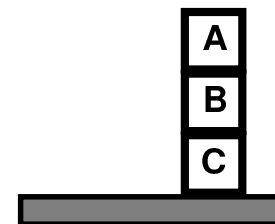
START

On(C,A) On(A,Table) Cl(B) On(B,Table) Cl(C)

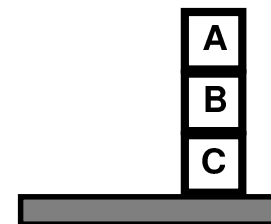
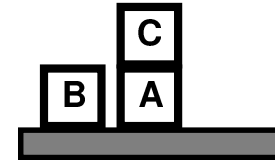
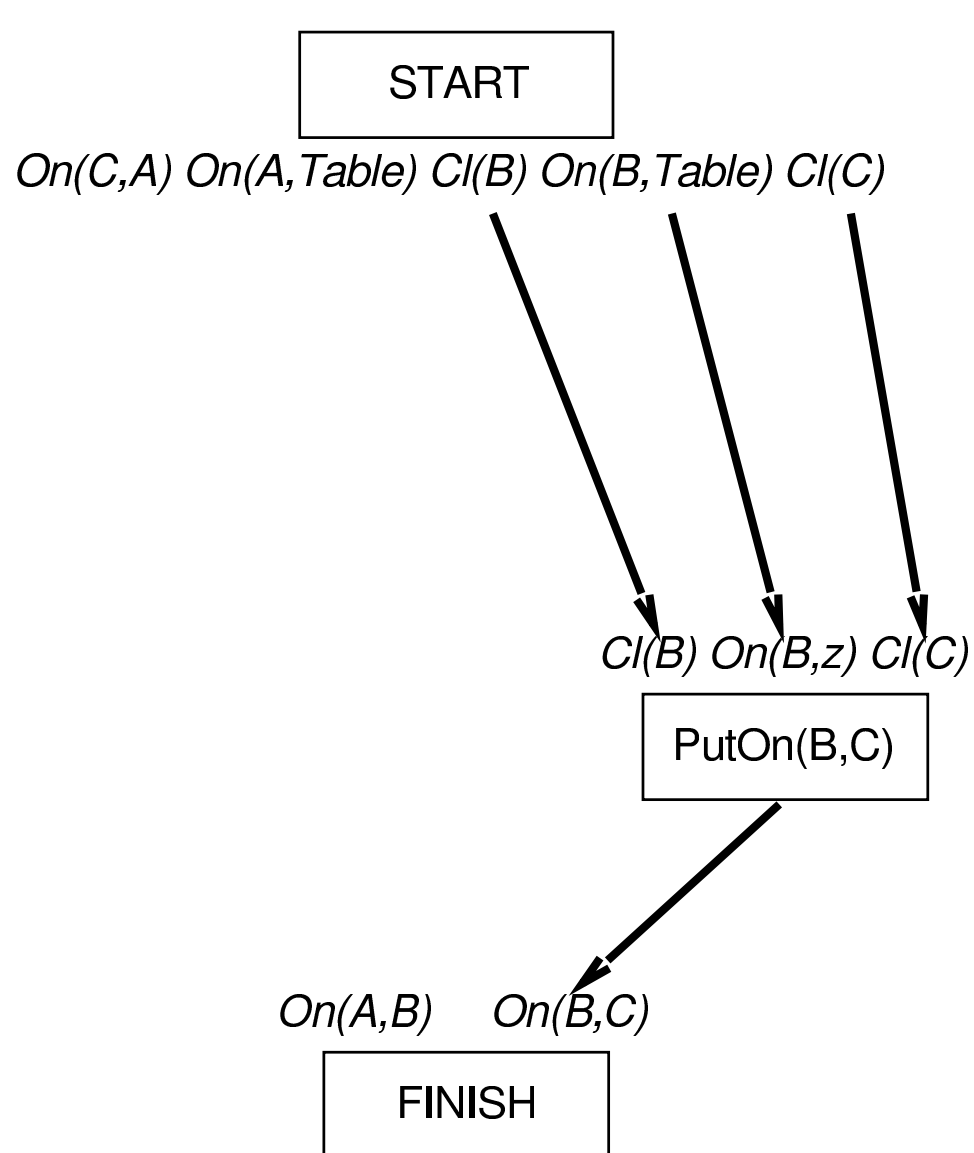


On(A,B) On(B,C)

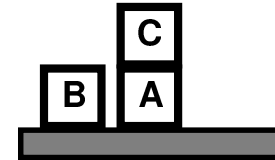
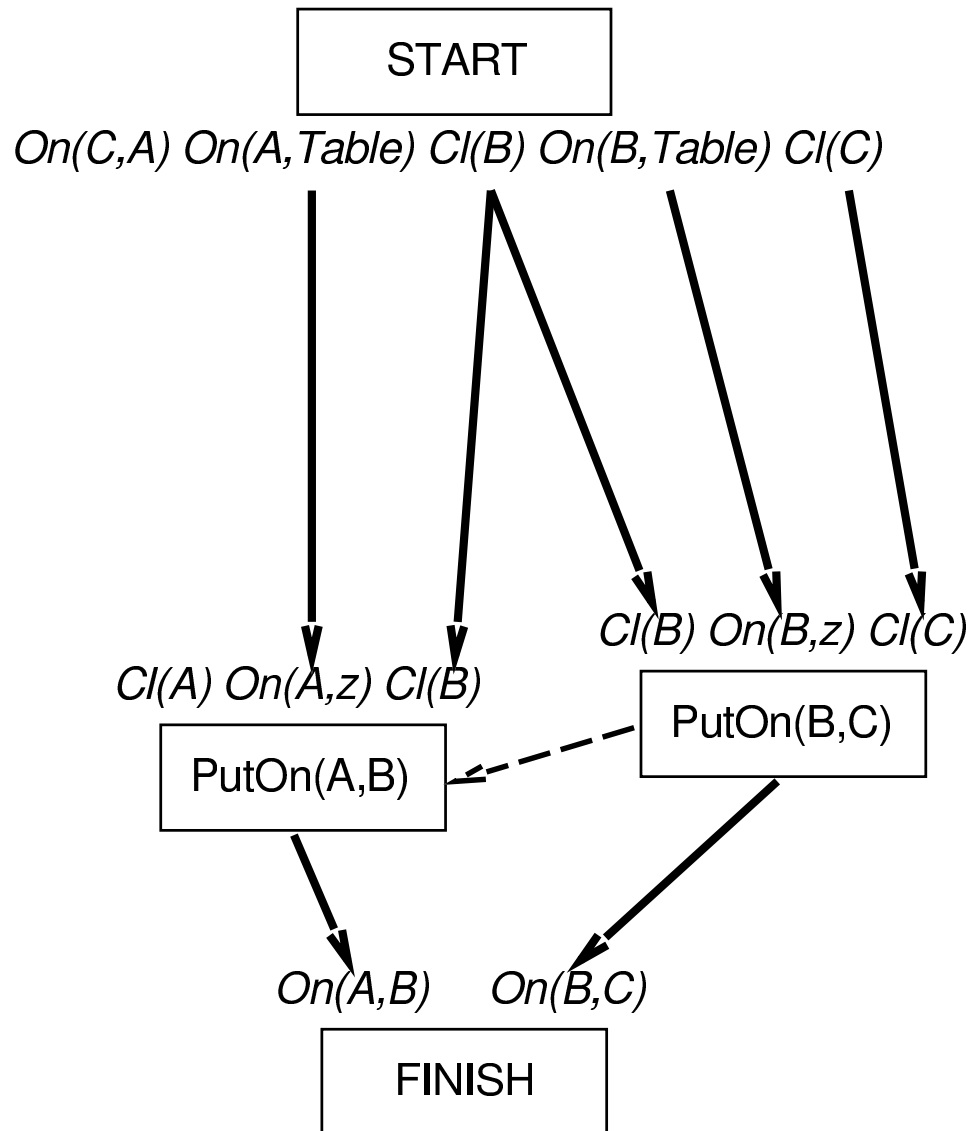
FINISH



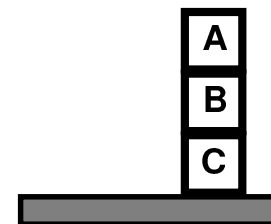
Example: Blocks World



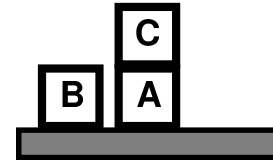
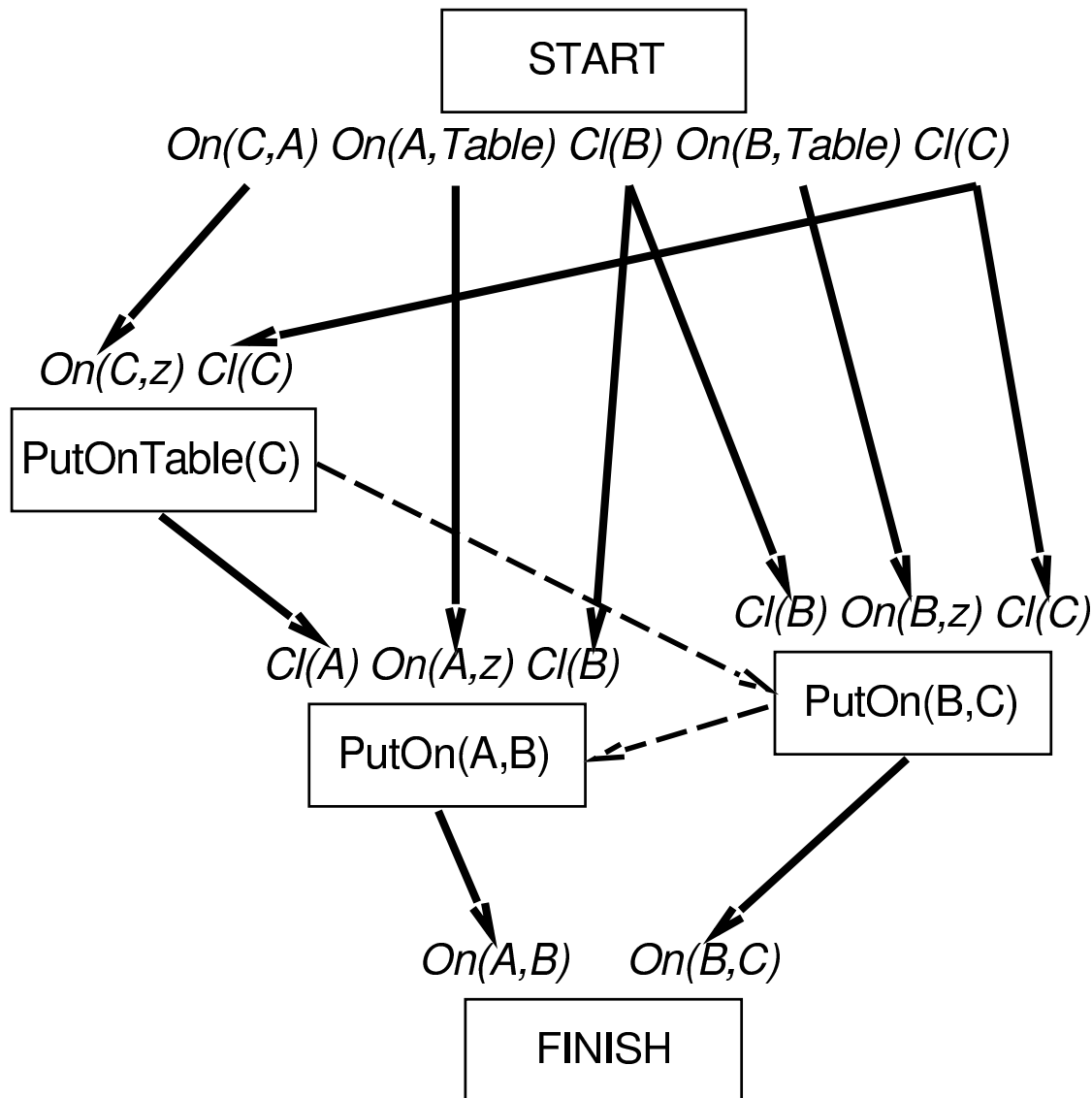
Example: Blocks World



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

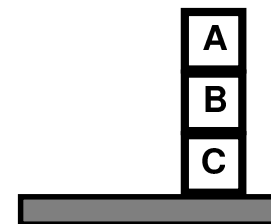


Example: Blocks World



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

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



POP (Partial Order Planner) Algorithm Sketch

function POP(*initial*, *goal*, *operators*) **returns** *plan*

plan \leftarrow MAKE-MINIMAL-PLAN(*initial*, *goal*)

loop do

if SOLUTION?(*plan*) **then return** *plan* % complete and consistent

$S_{need}, c \leftarrow$ 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 (Cont'd)

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 $S_{add} \xrightarrow{c} S_{need}$ to $LINKS(plan)$

add the ordering constraint $S_{add} \prec S_{need}$ to $ORDERINGS(plan)$

if S_{add} is a newly added step from $operators$ **then**

add S_{add} to $STEPS(plan)$

add $Start \prec S_{add} \prec Finish$ to $ORDERINGS(plan)$

POP Algorithm (Cont'd)

procedure RESOLVE-THREATS($plan$)

for each S_{threat} that threatens a link $S_i \xrightarrow{c} S_j$ in LINKS($plan$) **do**

choose either

Demotion: Add $S_{threat} \prec S_i$ to ORDERINGS($plan$)

Promotion: Add $S_j \prec S_{threat}$ to ORDERINGS($plan$)

if not CONSISTENT($plan$) **then fail**

end

Properties of POP

- **Non-deterministic search for plan,
backtracks over choicepoints on failure:**
 - **choice of S_{add} to achieve S_{need}**
 - **choice of promotion or demotion for clobberer**

Properties of POP

- **Non-deterministic search for plan,
backtracks over choicepoints on failure:**
 - **choice of S_{add} to achieve S_{need}**
 - **choice of promotion or demotion for clobberer**
- **Sound and complete**

Properties of POP

- **Non-deterministic search for plan,
backtracks over choicepoints on failure:**
 - **choice of S_{add} to achieve S_{need}**
 - **choice of promotion or demotion for clobberer**
- **Sound and complete**
- **There are extensions for:
disjunction, universal quantification, negation, conditionals**

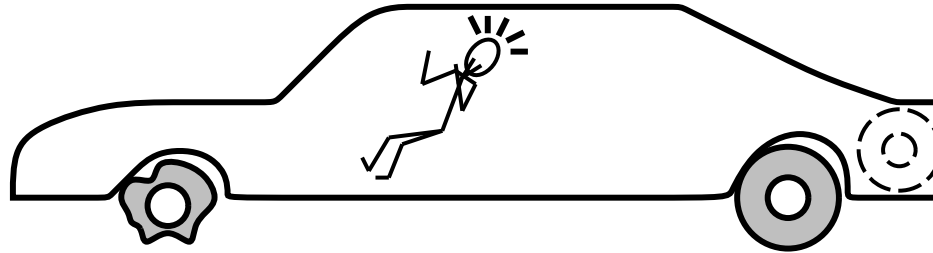
Properties of POP

- **Non-deterministic search for plan,
backtracks over choicepoints on failure:**
 - **choice of S_{add} to achieve S_{need}**
 - **choice of promotion or demotion for clobberer**
- **Sound and complete**
- **There are extensions for:
disjunction, universal quantification, negation, conditionals**
- **Efficient with good heuristics from problem description
But: very sensitive to subgoal ordering**

Properties of POP

- **Non-deterministic search for plan,
backtracks over choicepoints on failure:**
 - choice of S_{add} to achieve S_{need}
 - choice of promotion or demotion for clobberer
- **Sound and complete**
- **There are extensions for:
disjunction, universal quantification, negation, conditionals**
- **Efficient with good heuristics from problem description
But: very sensitive to subgoal ordering**
- **Good for problems with loosely related subgoals**

The Real World



START

*~Flat(Spare) Intact(Spare) Off(Spare)
On(Tire1) Flat(Tire1)*

On(x) ~Flat(x)

FINISH

On(x)

Remove(x)

Off(x) ClearHub

Off(x) ClearHub

Puton(x)

On(x) ~ClearHub

Intact(x) Flat(x)

Inflate(x)

~Flat(x)

Things Go Wrong

Incomplete information

● Unknown preconditions **Example:** $\text{Intact}(\text{Spare})?$

● Disjunctive effects

Example: $\text{Inflate}(x)$ causes
 $\text{Inflated}(x) \vee \text{SlowHiss}(x) \vee \text{Burst}(x) \vee \text{BrokenPump} \vee \dots$

Things Go Wrong

Incomplete information

● Unknown preconditions **Example:** $Intact(Spare)?$

● Disjunctive effects

Example: $Inflate(x)$ causes
 $Inflated(x) \vee SlowHiss(x) \vee Burst(x) \vee BrokenPump \vee \dots$

Incorrect information

● Current state incorrect **Example:** spare NOT intact

● Missing/incorrect postconditions in operators

Things Go Wrong

Incomplete information

- Unknown preconditions **Example:** $Intact(Spare)?$

- Disjunctive effects

Example: $Inflate(x)$ causes
 $Inflated(x) \vee SlowHiss(x) \vee Burst(x) \vee BrokenPump \vee \dots$

Incorrect information

- Current state incorrect **Example:** spare NOT intact
- Missing/incorrect postconditions in operators

Qualification problem

- Can never finish listing all the required preconditions and possible conditional outcomes of actions

Solutions

Conditional planning

- Plan to obtain information (**observation actions**)
- Subplan for each contingency

Example: $[Check(Tire1), \text{If}(Intact(Tire1), [Inflate(Tire1)], [CallHelp])]$

Disadvantage: Expensive because it plans for many unlikely cases

Solutions

Conditional planning

- Plan to obtain information (**observation actions**)
- Subplan for each contingency

Example: $[Check(Tire1), If(Intact(Tire1), [Inflate(Tire1)], [CallHelp])]$

Disadvantage: Expensive because it plans for many unlikely cases

Monitoring/Replanning

- Assume normal states / outcomes
- Check progress **during execution**, replan if necessary

Disadvantage: Unanticipated outcomes may lead to failure

Conditional Planning

Execution of conditional plan

$[\dots, \mathbf{If}(p, [thenPlan], [elsePlan]), \dots]$

Check p against current knowledge base, execute $thenPlan$ or $elsePlan$

Conditional Planning

Execution of conditional plan

$[\dots, \text{If}(p, [\text{thenPlan}], [\text{elsePlan}]), \dots]$

Check p against current knowledge base, execute *thenPlan* or *elsePlan*

Conditional planning

Just like POP except:

If an open condition can be established by observation action

- add the action to the plan
- complete plan for each possible observation outcome
- insert conditional step with these subplans

CheckTire(x)

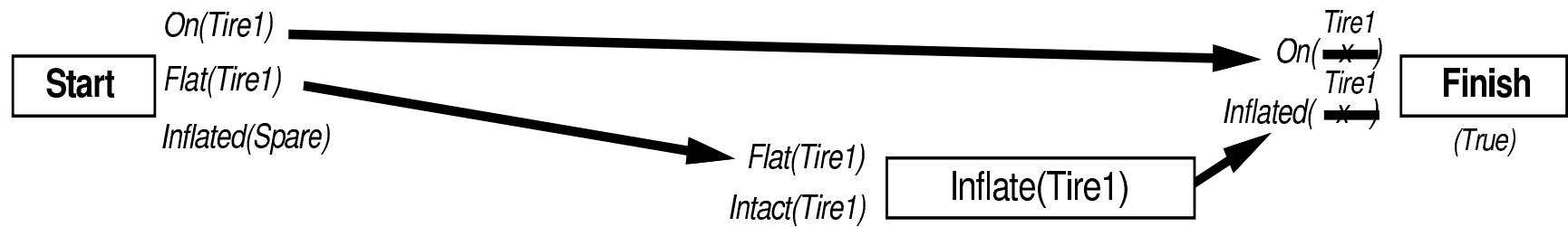
KnowsIf(Intact(x))

Conditional Planning Example

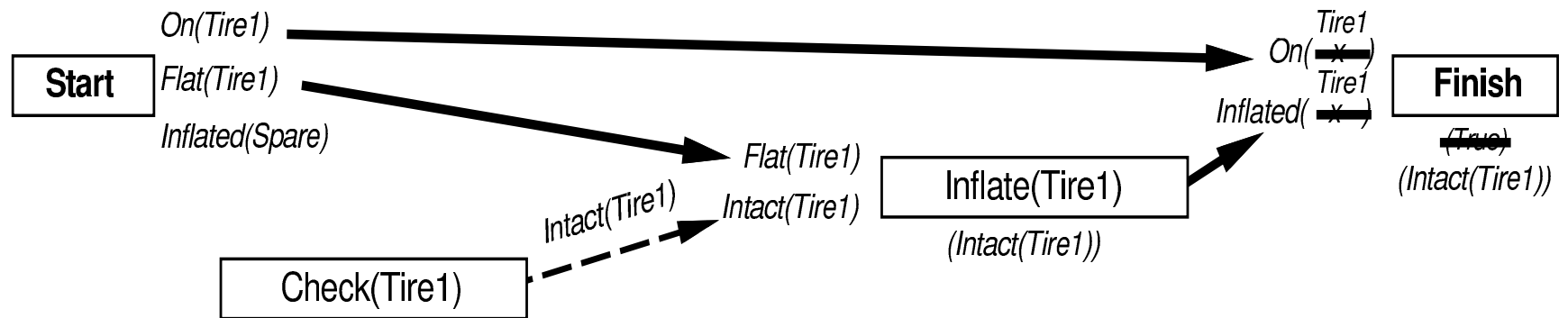
Start *On(Tire1)*
Flat(Tire1)
Inflated(Spare)

On(x)
Inflated(x) **Finish**
(True)

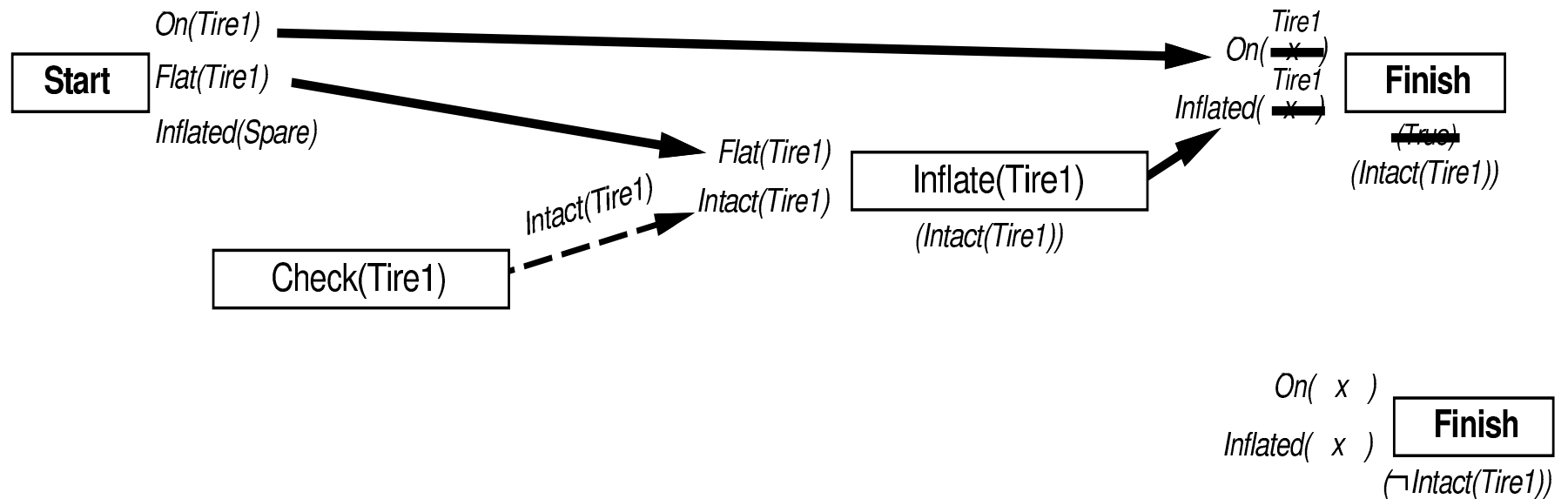
Conditional Planning Example



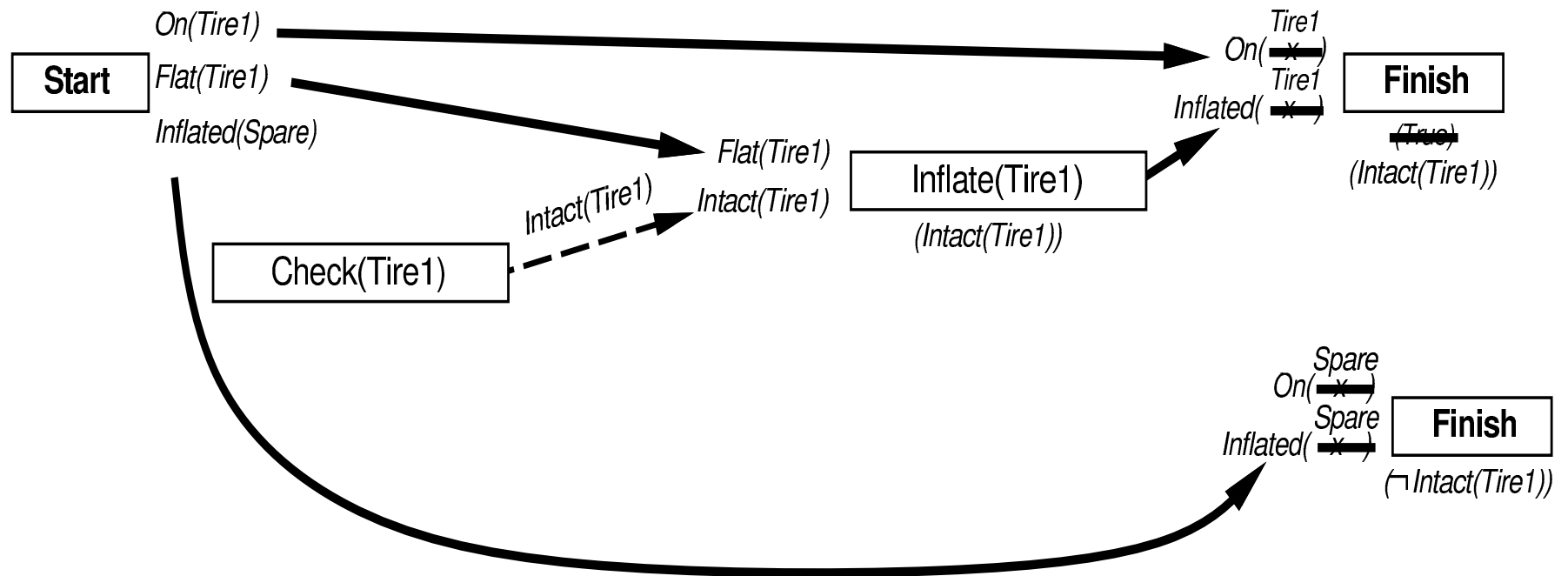
Conditional Planning Example



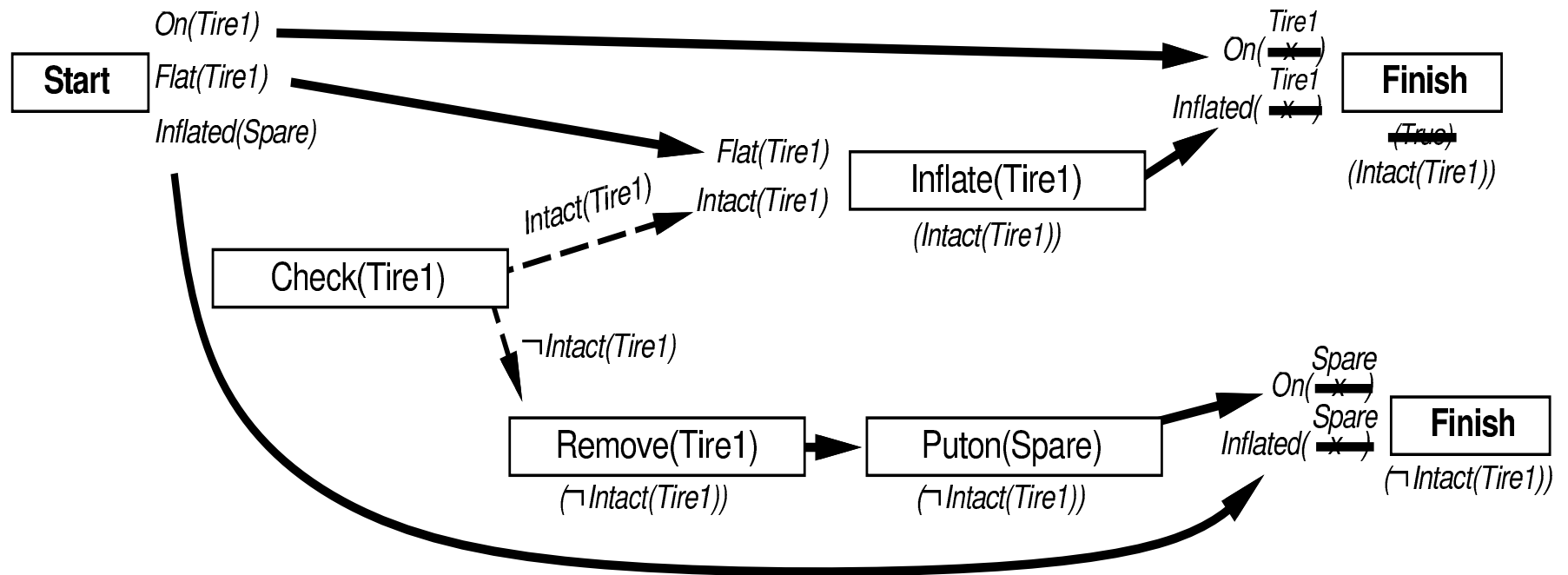
Conditional Planning Example



Conditional Planning Example



Conditional Planning Example



Monitoring

Execution monitoring

Failure: Preconditions of remaining plan not met

Monitoring

Execution monitoring

Failure: Preconditions of remaining plan not met

Action monitoring

Failure: Preconditions of next action not met
(or action itself fails, e.g., robot bump sensor)

Monitoring

Execution monitoring

Failure: Preconditions of remaining plan not met

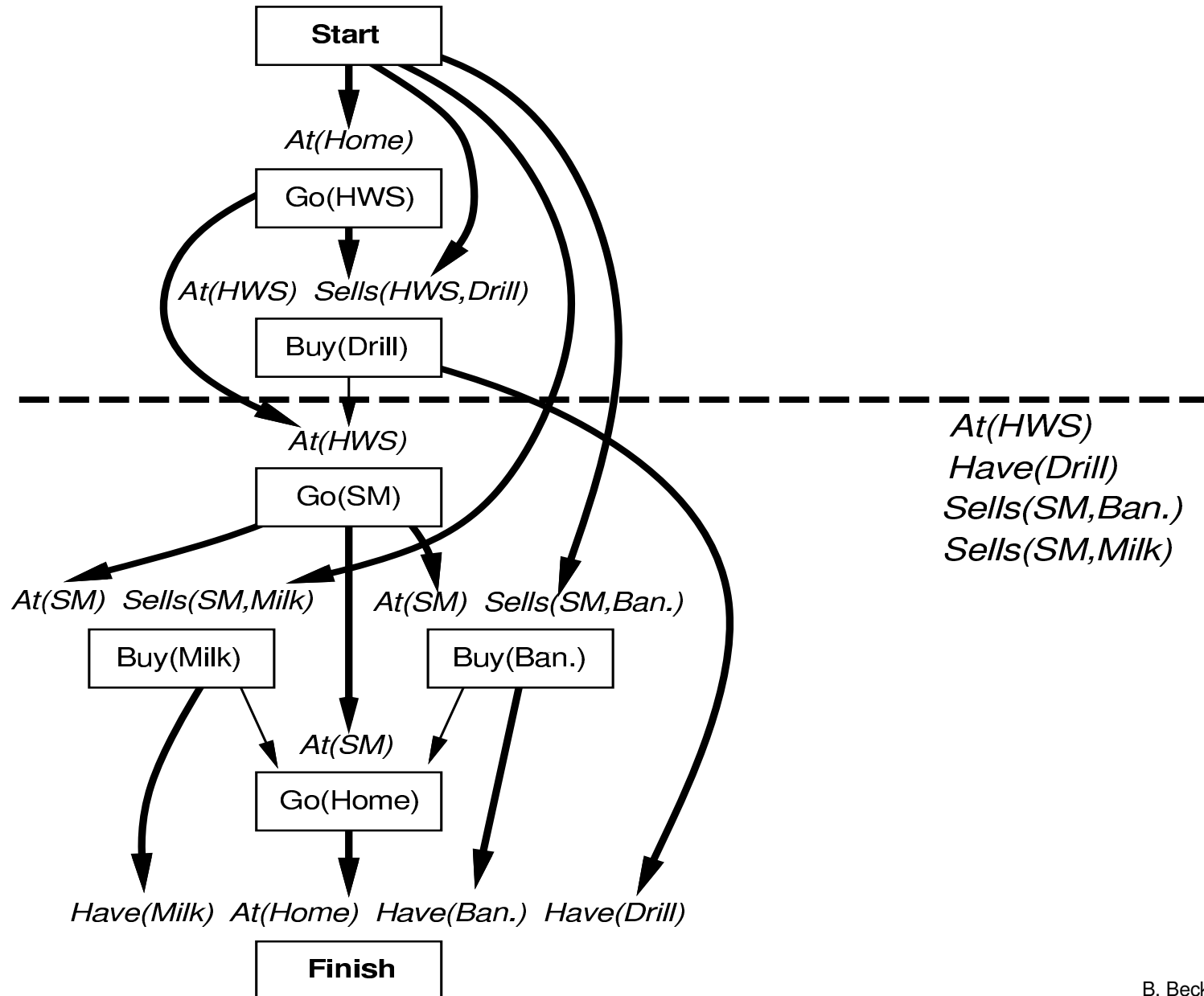
Action monitoring

Failure: Preconditions of next action not met
(or action itself fails, e.g., robot bump sensor)

Consequence of failure

Need to **replan**

Preconditions for Remaining Plan



Replanning

Simplest

On failure, replan from scratch

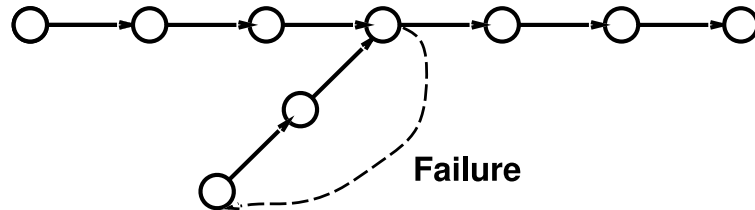
Replanning

Simplest

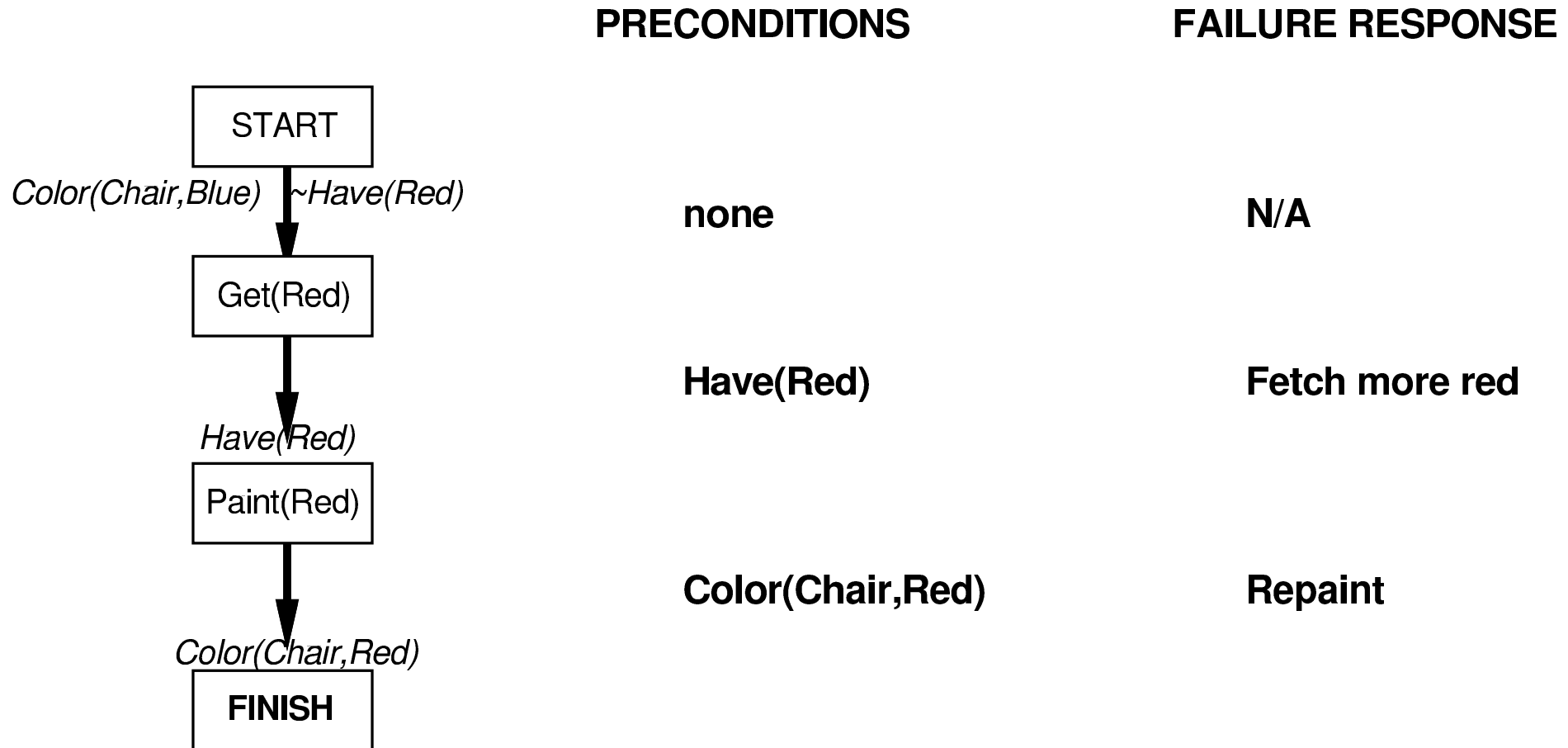
On failure, replan from scratch

Better

Plan to get back on track by reconnecting to best continuation



Replanning: Example



Summary Planning

- Differs from general problem search;
subgoals solved independently
- STRIPS: restricted format for actions, logic-based
- Nodes in search space are partial plans
- POP algorithm
- Standard planning cannot cope with incomplete/incorrect information
- Conditional planning with sensing actions to complete information;
expensive at planning stage
- Replanning based on monitoring of plan execution;
expensive at execution stage