Planning, Execution & Learning
1. Partial Order Planning

Reid Simmons

Partial Order Planning

- Basic Idea
  - ***Search in* plan space *and use* least commitment, *when possible***

- Plan Space Search
  - Search space is set of ***partial plans***
  - Plan is tuple <*A*, *O*, *B*>
    - *A*: Set of ***actions***, of the form $(a_i : Op_j)$
    - *O*: Set of ***orderings***, of the form $(a_i < a_j)$
    - *B*: Set of ***bindings***, of the form $(v_i = C)$, $(v_i\ {}^{1}\ C)$, $(v_i = v_j)$ or $(v_i\ {}^{1}\ v_j)$
  - Initial plan:
    - <{*start*, *finish*}, {*start* < *finish*}, {}>
    - *start* has no preconditions; Its effects are the initial state
    - *finish* has no effects; Its preconditions are the goals

Least Commitment

- Basic Idea
  - ***Make choices only that are relevant to solving the current part of the problem***

- Least Commitment Choices
  - **Orderings**: Leave actions unordered, unless they must be sequential
  - **Bindings**: Leave variables unbound, unless needed to unify with conditions being achieved
  - **Actions**: Usually not subject to "least commitment"
- Refinement
  - Only ***add*** information to the current plan
  - ***Transformational*** planning can remove choices

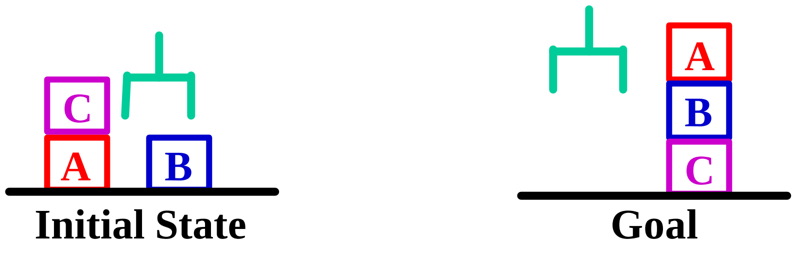
Plan Terminology

- ***Totally Ordered*** Plan
  - There exists sufficient orderings O such that all actions in A are ordered with respect to each other
- ***Fully Instantiated*** Plan
  - There exists sufficient constraints in B such that all variables are constrained to be equal to some constant
- ***Consistent*** Plan
  - There are no contradictions in O or B
- ***Complete*** Plan
  - Every precondition p of every action a_i in A is ***achieved***: There exists an effect of an action a_j that comes before a_i and unifies with p, and no action a_k that deletes p comes between a_j and a_i

NOAH [Sacerdoti, 1975]

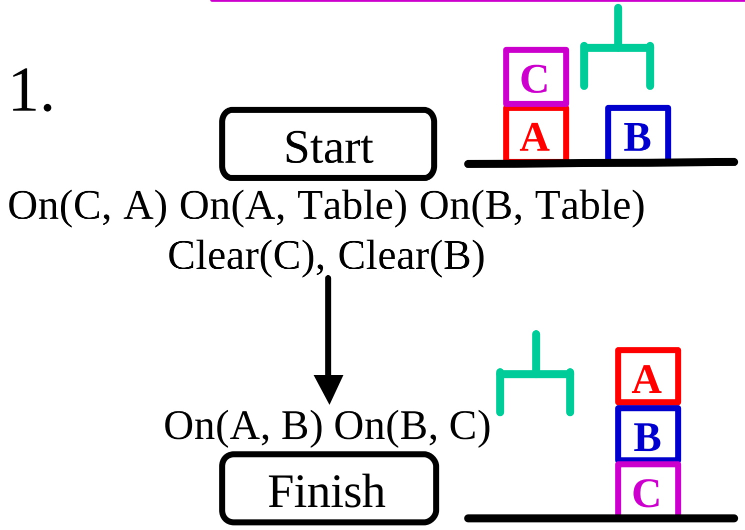
- NOAH
  - First non-linear, partial-order planner
  - Introduced notion of plan-space search
  - Used ***TOME*** (*Table of Multiple Effects*) to detect goal interactions

- NOAH can easily (and optimally) solve the “Sussman Anomaly” problem

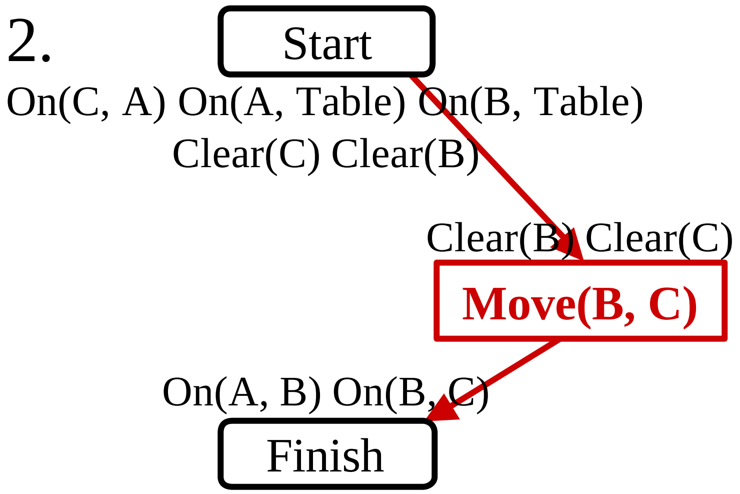


NOAH and Sussman's Anomaly

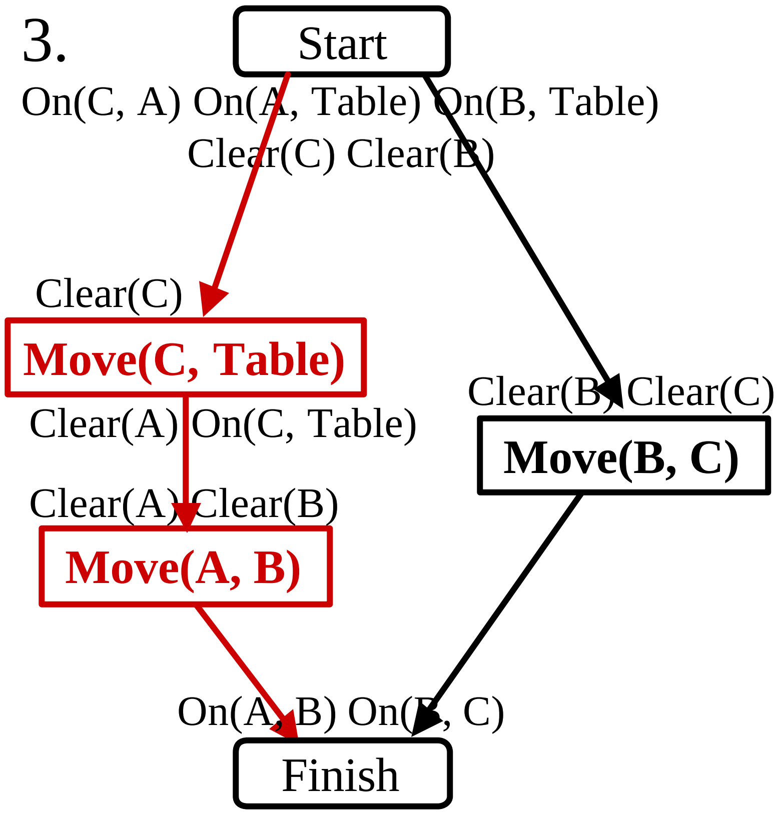
1.



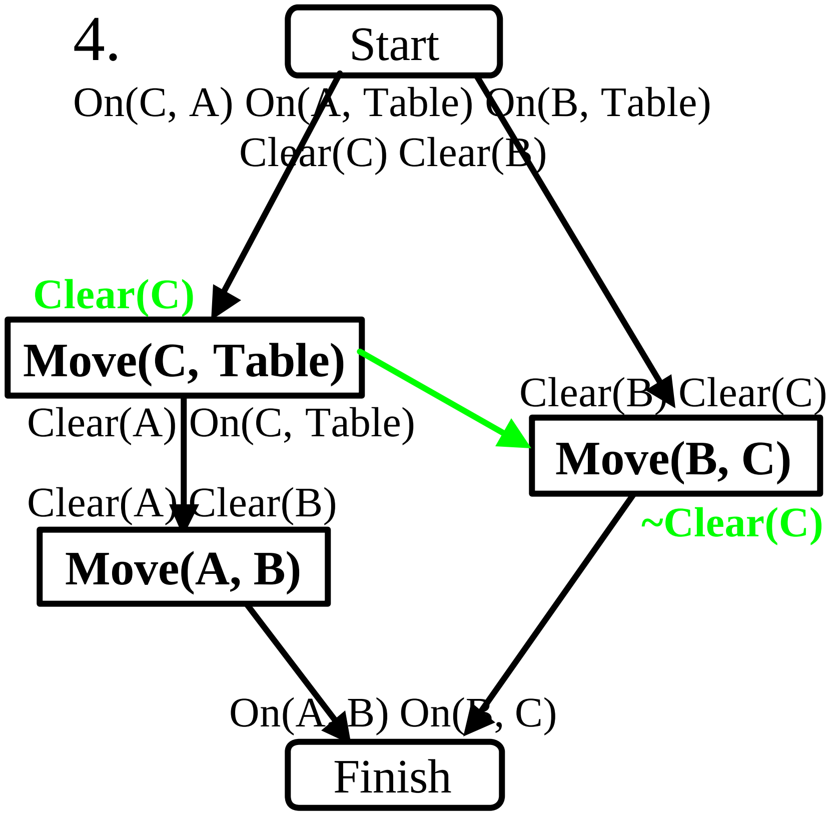
2.

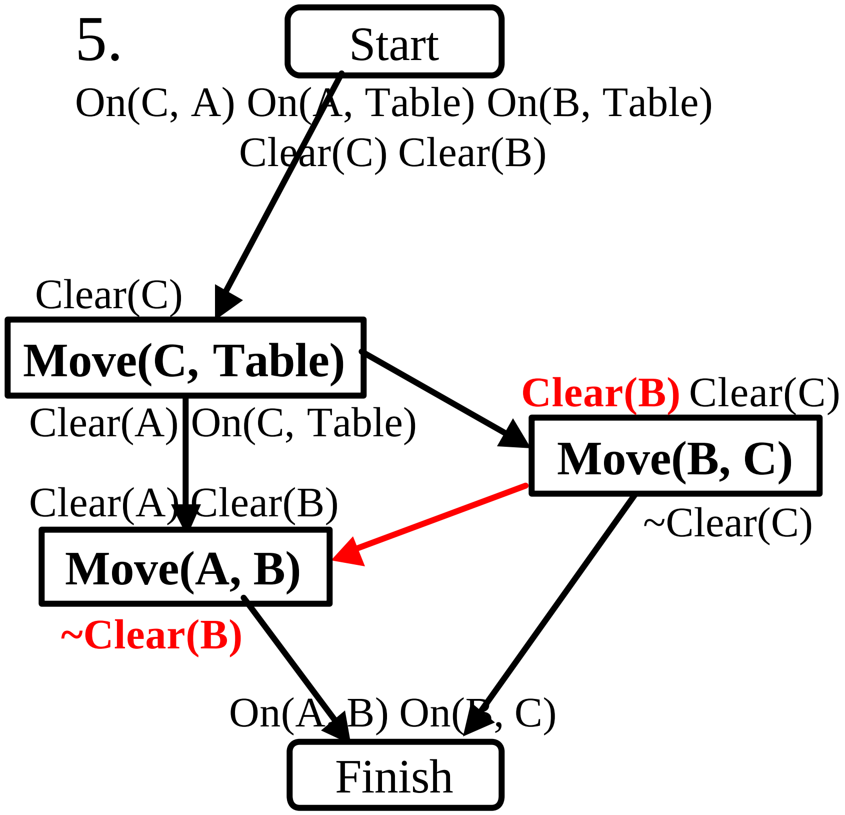


3.



NOAH and Sussman's Anomaly





Modal Truth Criterion [Chapman, 1987]

- Modal Truth Criterion (MTC)
  - Formalized criterion for determining whether a (partial) plan achieves a given precondition *p* at a given step *s*
    - *p* is true in *s* if:

      $*t* ((’ *t* < *s*) Ù ’ asserts(*t*, *p*)) Ù
      " C ((’ *s* < *C*) Ú
      " *q* ((à *q* » *p*) Þ ’ ~denies(*C*, *q*) Ú
      $*W* ((’ *C* < *W*) Ù (’ *W* < *s*) Ù
      $*r* (asserts(*W*, *r*) Ù (*p* » *q*) Þ (*p* » *r*)))))
- Can be used to generate planning algorithm (TWEAK)
  - **step addition / establishment**
  - **promotion/demotion**
  - **separation**
  - **white knight**

SNLP [McAllester & Rosenblitt, 1991]

- Systematic Non-Linear Planner (SNLP)
  - Efficient way to determine which preconditions are achieved
  - Explore each node in search space at most once
    - Not clear whether this is an advantage…
- Causal Links
  - The "purpose" of an action (which condition it supports)
  - $a_i \rightarrow^c a_j$, where a_i, a_j are actions and c is an effect of a_i
  - Plan = $<A, O, B, L>$
- Threats
  - Action a_k with an effect c' that might "clobber" a causal link
  - ***Promotion***: Order a_k after a_j
  - ***Demotion*** : Order a_k before a_i
  - ***Separation*** : Constrain c' so that it does not unify with c (non-codesignation constraint)

UCPOP [Penberthy & Weld, 1992]

- Universal, Conditional Partial-Order Planner (UCPOP)
  - Extension of SNLP to handle more expressive operators
    - Conditionals
    - Disjunction in preconditions
    - Universal and existential quantification

- Uses ***unification*** to find necessary bindings
  - Most General Unifier: MGU(p, q, B) = {(v_i, x_i), … }

- Uses ***constraint satisfaction*** to prove consistency of plans
  - Consistent orderings
  - Consistent variable bindings (co-designation)

UCPOP Language Extensions

- Conditionals
  - ***(when (?b ^{1} table) (clear ?b))***
  - Add a new threat resolution mechanism: ***confrontation***
    - Add the ***negation*** of conditional effect antecedent to the set of goals that must be achieved
- Disjunction in Preconditions
  - Add a new choice point to the algorithm that non-deterministically chooses to achieve one of the disjuncts
- Quantification
  - Typed formula: ***(forall (<type> <var>) <expression>)***
  - ***Universal***: Expand into equivalent conjunct (assumes finite, known universe of objects)
  - ***Existential***: Replace quantification with Skolem function ***((<type> <var$_i$>) & <expression>\{(<var>, <var$_i$>)})***

UCPOP & MTC

- The Modal Truth Criterion was used to prove that, for expressive operator representations, determining whether a plan achieves its conditions is NP-hard!
- UCPOP can handle expressive operators, yet it can trivially determine whether it has found a plan that achieves all the conditions
- How to reconcile this apparent contradiction?
  - MTC ***proves whether***: Need to find necessary and sufficient conditions
  - UCPOP ***ensures achievement***: Only need sufficient conditions
  - UCPOP pushes complexity from per-node cost to search space size
  - This is a **win** if search is (usually) well focused

UCPOP Algorithm

- UCPOP(*initial-state*, *goals*)
  - *plan* = <*A*={*Start*, *Finish*}, *O*={*Start* < *Finish*}, *B*={}, *L*={}>
  - *agenda* = {(*goals*, *Finish*)}
  - Repeat until *agenda* is empty
    - **Select** (and remove) an ***open condition*** (q, a_c) from *agenda*
    - If q is quantified, then expand and add it to *agenda*
    - If q is a conjunction, then add each conjunct to *agenda*
    - If q is a disjunction, then **choose** one disjunct and add to *agenda*
    - If q is a literal and $a_p \rightarrow^{\sim q} a_c$ exists in L, then **Fail**
    - Else **choose** a_p (either a new action or an existing action from A) that has an effect r that unifies with q
      - Add {$a_p \rightarrow^{q} a_c$ } to L
      - Add MGU(q, r, B) to B
      - Add {($a_p < a_c$), ($a_p < Finish$), ($Start < a_p$)} to O
      - If a_p is new, add preconditions to *agenda* and any variable constraints to B
    - For each causal link $a_i \rightarrow^{p} a_j$ and each a_t action which threatens the link, **choose** a resolution mechanism
      - ***Promotion***: Add ($a_j < a_t$) to O
      - ***Demotion*** : Add ($a_t < a_i$) to O
      - ***Confrontation*** : If threatening effect is conditional, with antecedent S and effect R, add {($\sim S$\MGU(p, r, B), a_t)} to *agenda*
    - **Fail** if *plan* is inconsistent

UCPOP and the Briefcase World

- **Move**(b, src, dest)
  Pre: briefcase(b), at(b, src), src ^{1} dest
  Effect: at(b, dest), ~at(b, src),
  (forall (object x) (when in(x, b) (at(x, dest) & ~at(x, src))))

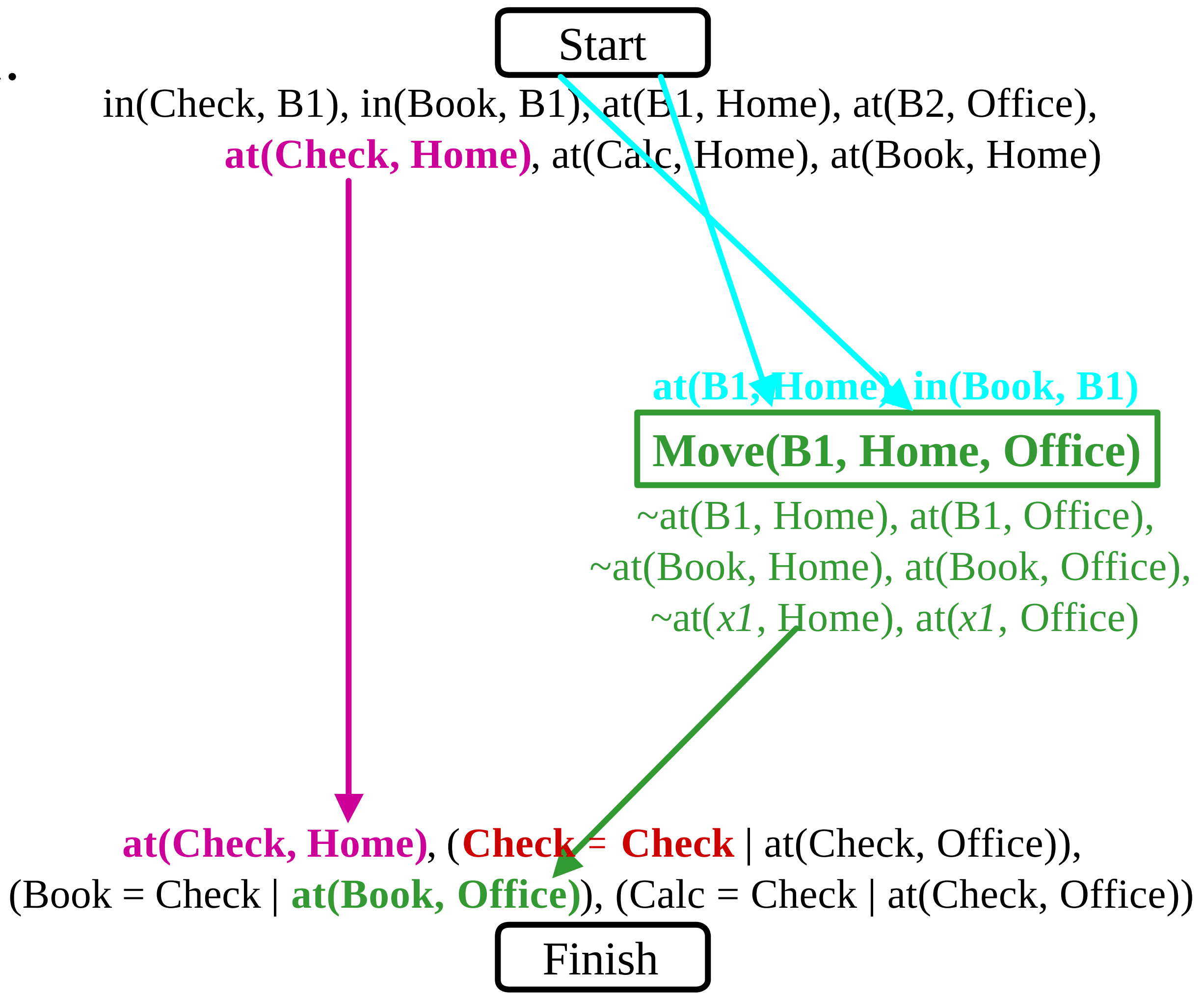
- **Take-Out**(x, b)
  Pre: in(x, b)
  Effect: ~in(x, b)

  Put-In(x, b, loc)
  Pre: briefcase(b), at(x, loc), at(b, loc), x ^{1} b
  Effect: in(x, b)

- **Initial**: in(Check, B1), in(Book, B1), at(B1, Home), at(B2, Office),
  at(Check, Home), at(Calc, Home), at(Book, Home),
  object(Check), object(Book), object(Calc),
  briefcase(B1), briefcase(B2)
- **Goal**: at(Check, Home), (forall (object x) (x = Check | at(x, Office)))

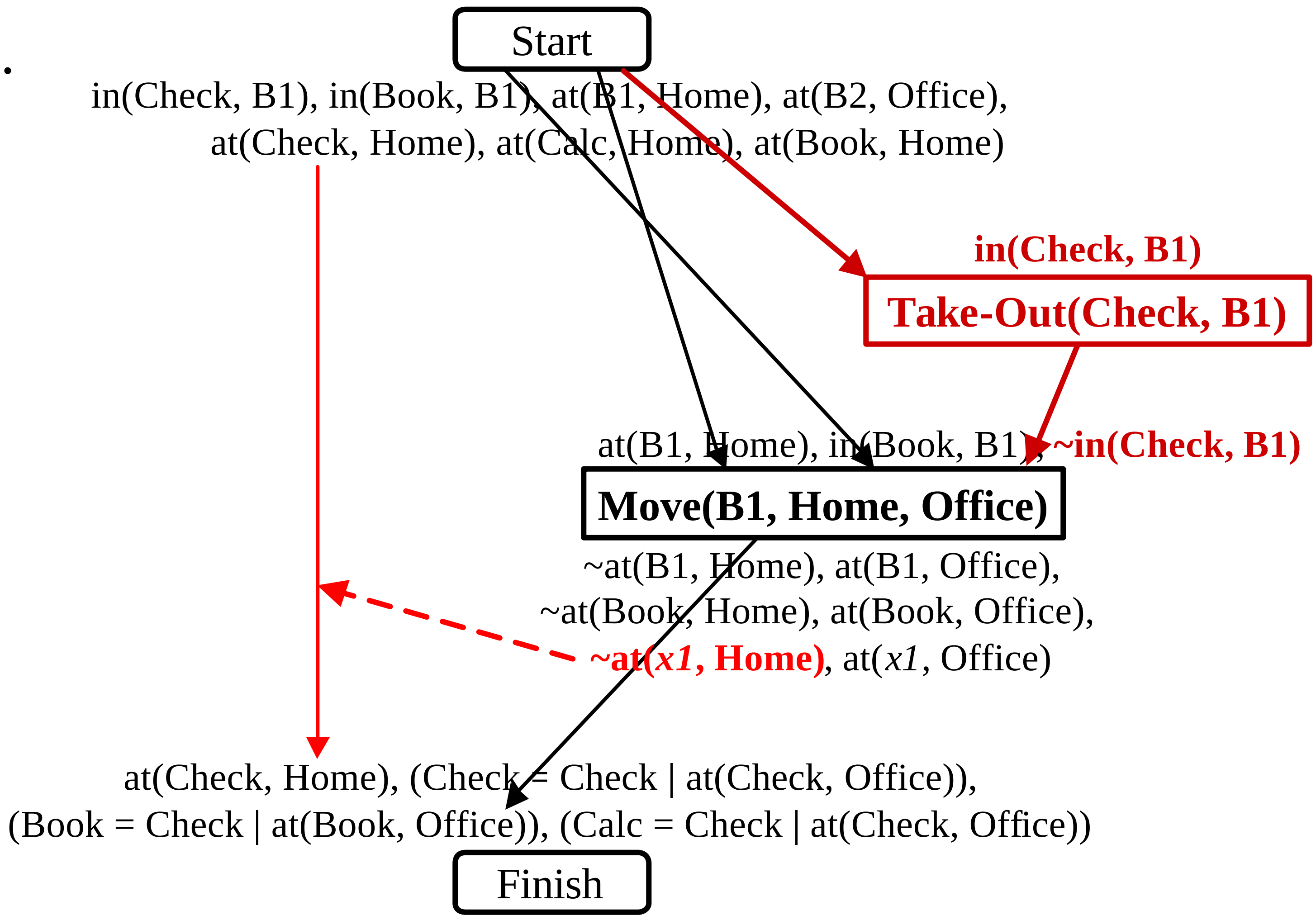
UCPOP Briefcase World Example

1.


Start
in(Check, B1), in(Book, B1), at(B1, Home), at(B2, Office),
at(Check, Home), at(Calc, Home), at(Book, Home)
at(B1, Home), in(Book, B1)
Move(B1, Home, Office)
~at(B1, Home), at(B1, Office),
~at(Book, Home), at(Book, Office),
~at(*x1*, Home), at(*x1*, Office)
at(Check, Home), (Check = Check | at(Check, Office)),
(Book = Check | at(Book, Office)), (Calc = Check | at(Check, Office))
Finish

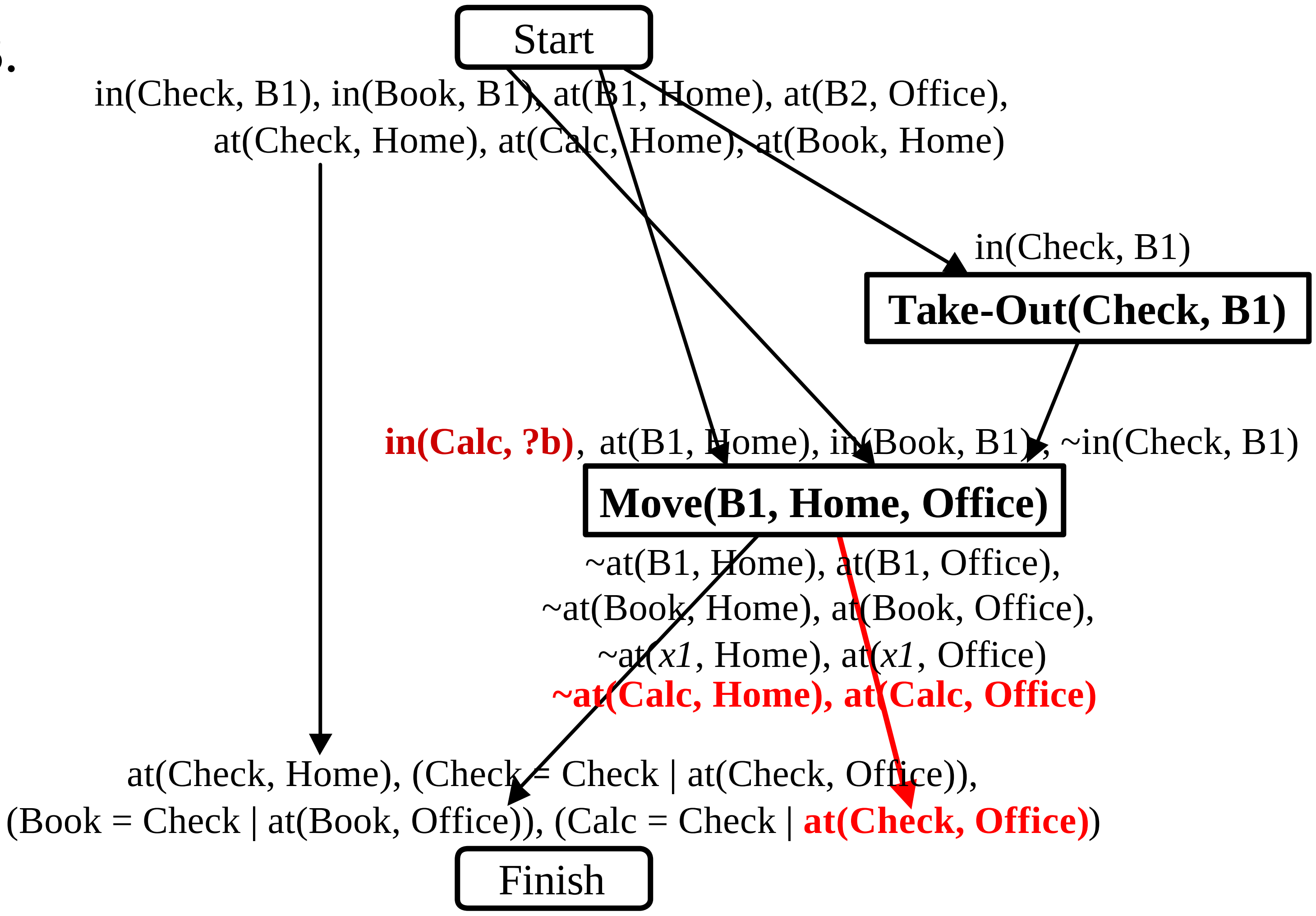
UCPOP Briefcase World Example

2.


Start
in(Check, B1), in(Book, B1), at(B1, Home), at(B2, Office),
at(Check, Home), at(Calc, Home), at(Book, Home)
in(Check, B1)
Take-Out(Check, B1)
at(B1, Home), in(Book, B1), ~in(Check, B1)
Move(B1, Home, Office)
~at(B1, Home), at(B1, Office),
~at(Book, Home), at(Book, Office),
~at(x1, Home), at(x1, Office)
at(Check, Home), (Check = Check | at(Check, Office)),
(Book = Check | at(Book, Office)), (Calc = Check | at(Check, Office))
Finish

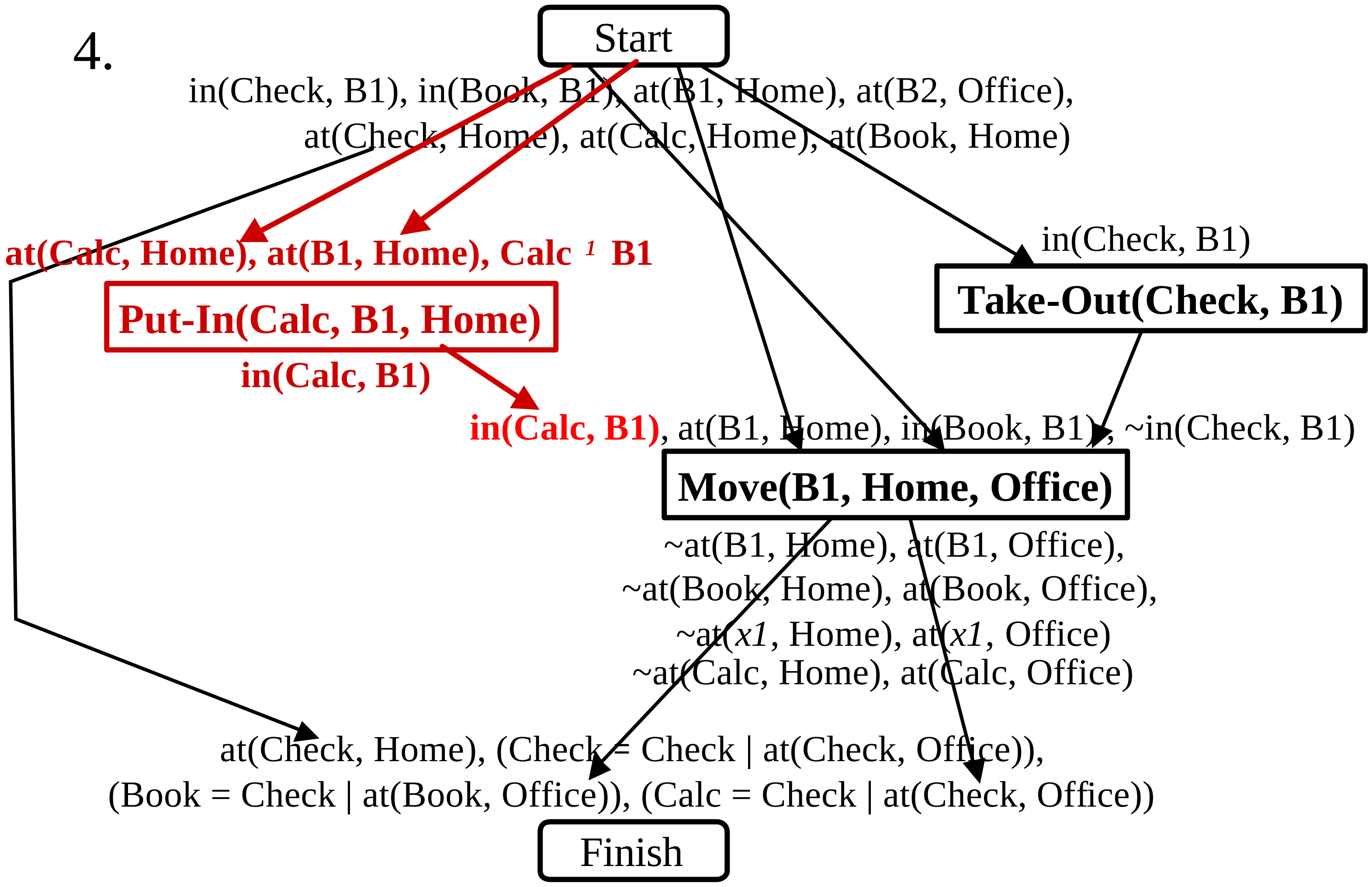
UCPOP Briefcase World Example

3.


Start
in(Check, B1), in(Book, B1), at(B1, Home), at(B2, Office),
at(Check, Home), at(Calc, Home), at(Book, Home)
in(Check, B1)
Take-Out(Check, B1)
in(Calc, ?b), at(B1, Home), in(Book, B1), ~in(Check, B1)
Move(B1, Home, Office)
~at(B1, Home), at(B1, Office),
~at(Book, Home), at(Book, Office),
~at(x1, Home), at(x1, Office)
~at(Calc, Home), at(Calc, Office)
at(Check, Home), (Check = Check | at(Check, Office)),
(Book = Check | at(Book, Office)), (Calc = Check | at(Check, Office))
Finish

UCPOP Briefcase World Example

4.


Start
in(Check, B1), in(Book, B1), at(B1, Home), at(B2, Office),
at(Check, Home), at(Calc, Home), at(Book, Home)
at(Calc, Home), at(B1, Home), Calc ^{1} B1
Put-In(Calc, B1, Home)
in(Calc, B1)
in(Check, B1)
Take-Out(Check, B1)
in(Calc, B1), at(B1, Home), in(Book, B1), ~in(Check, B1)
Move(B1, Home, Office)
~at(B1, Home), at(B1, Office),
~at(Book, Home), at(Book, Office),
~at(x1, Home), at(x1, Office)
~at(Calc, Home), at(Calc, Office)
at(Check, Home), (Check = Check | at(Check, Office)),
(Book = Check | at(Book, Office)), (Calc = Check | at(Check, Office))
Finish

Partial Order Planning: Discussion

- **Advantages**
  - Partial order planning is ***sound*** and ***complete***
  - Typically produces ***optimal*** solutions (plan length)
  - Least commitment may lead to shorter search times

- **Disadvantages**
  - Significantly more complex algorithms (higher *per-node* cost)
  - Hard to determine what is true in a state
  - Larger search space, since concurrent actions are allowed