State-Space Planning

Chapter 10

Outline

- ♢ State-space planning
- ♢ Progression planning (forward search)
- ♢ Heuristics
- ♢ Regression planning (backward search)
- ♢ Lifting

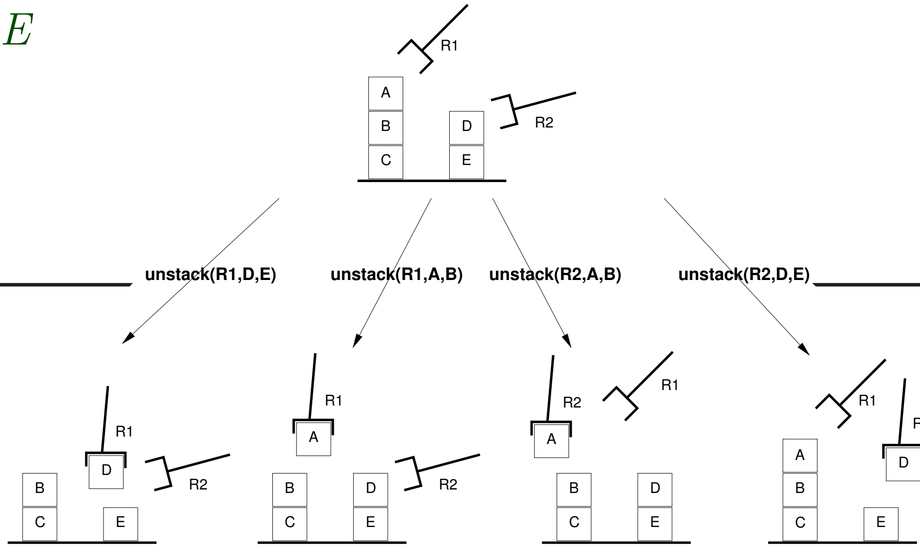
State-space planning

- Planning procedures are often search procedures
- They differ by the search space they consider
- State-space planning explores the most obvious search space:
  - Nodes labelled by states of the world
  - Actions define successor states
  - Plans are paths from the initial node to a goal node
- Search space can be explored in many ways:
  - forward, backward
  - using a variety of strategies (breadth-first, depth-first, A*, . . .)
  - using a variety of heuristics
  - The STRIPS representation enables an efficient exploration and domain-independent heuristics

Progression planning (forward search)

function Forward-Search(O, s_0, g) **returns** an action sequence, or failure

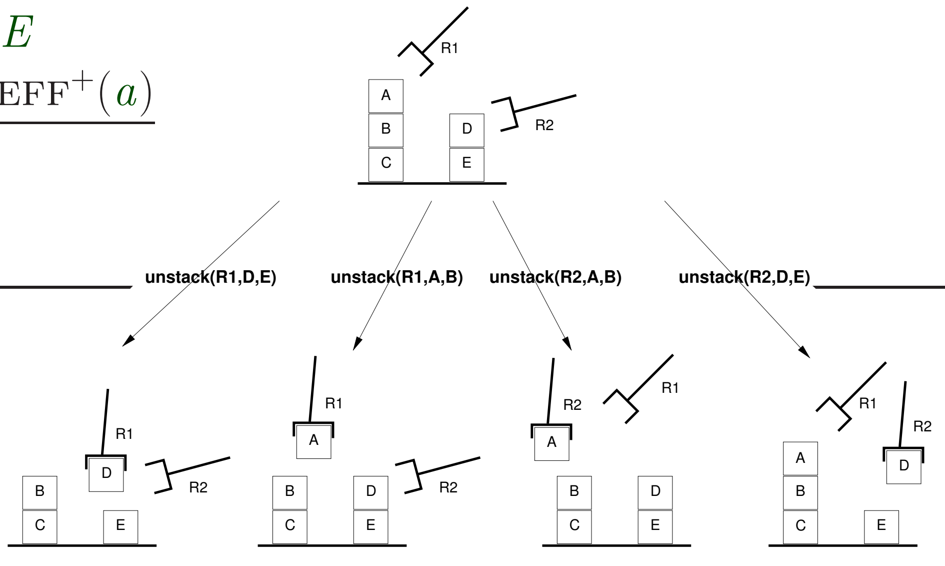
$s \leftarrow s_0$
$\pi \leftarrow \langle\rangle$
loop do
  if s satisfies g **then return** π
  $E \leftarrow \{a \mid a$ is a ground instance of an operator in O
      such that a is applicable in s $\}$
  if $E = \{\,\}$ **then return** failure
  choose an action $a \in E$
  $s \leftarrow \gamma(s, a)$
  $\pi \leftarrow \pi . a$
end



Progression planning (forward search)

function FORWARD-SEARCH(O, s_0, g) **returns** an action sequence, or failure

$s \leftarrow s_0$
$\pi \leftarrow \langle\rangle$
loop do
  if $\underline{g \subseteq s}$ **then return** π
  $E \leftarrow \{a \mid a$ is a ground instance of an operator in O such that $\underline{\mathrm{PRE}(a) \subseteq s}\}$
  if $E = \{\}$ **then return** failure
  choose an action $a \in E$
  $\underline{s \leftarrow (s \setminus \mathrm{EFF}^-(a)) \cup \mathrm{EFF}^+(a)}$
  $\pi \leftarrow \pi.a$
end



Properties of Forward-Search

Forward-Search can be used in conjunction with any search strategy to implement **choose**, breadth-first search, depth-first search, iterative-deepening, greedy search, A*, IDA*, . . .

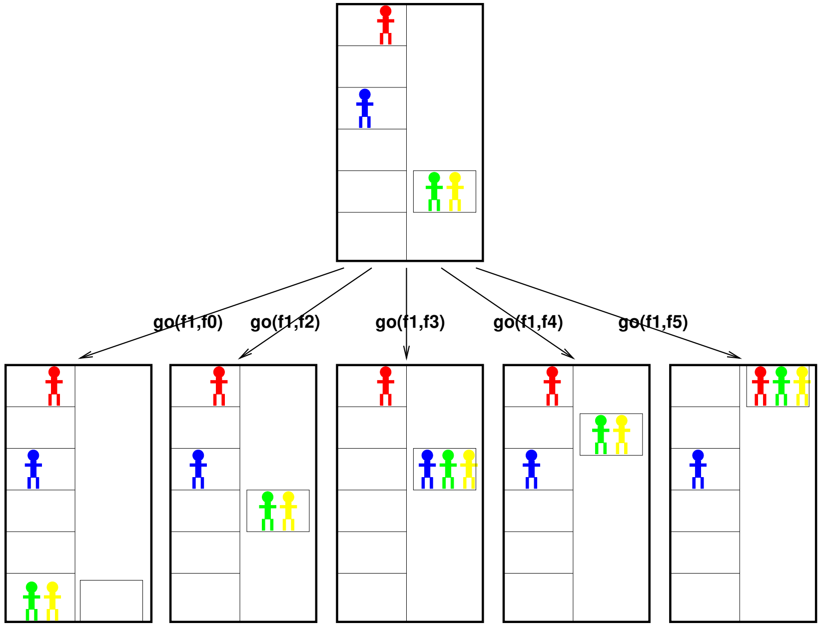
Forward-Search is sound: any plan returned is guaranteed to be a solution to the problem.

Forward-Search is complete: provided the underlying search strategy is complete, it will always return a solution to the problem if there is one.

For instance, when used with breadth-first search it will be complete, when used with depth-first search it will be complete if the state space is finite – in general, we need to detect and forbid loops.

Branching factor in Forward-Search

Forward-Search can have a large branching factor



It wastes a lot of time trying **irrelevant** actions

How do we cope with this?:

domain-specific: search control rules, heuristics
domain-independent: heuristics extracted from the STRIPS problem description
backward search: from the goal to the initial state

Domain-independent heuristics

Example: count number of unachieved goal propositions; fast, inadmissible and not very informative.

From the search lectures:

- An admissible heuristic is **optimistic**: it gives a lower bound on the true cost of a solution to the problem
- An admissible heuristic can be obtained by relaxing a problem P into a simpler problem P': the cost of any optimal solution to P' is a lower bound on the cost of the optimal solution to P

We relax STRIPS problem descriptions to obtain generic planning heuristics:

- delete relaxation heuristics
- abstraction heuristics
- landmark heuristics

Delete relaxation

Let P be a planning problem and let P^+ be the relaxed problem obtained by ignoring the negative effects (delete list) of every action

P^+ is called the delete-relaxation of P

P^+ is like P except that: $\text{EFF}^-(a) = \{\,\}$ for all a

A solution for P^+ is called a relaxed plan.

Delete relaxation

Let P be a planning problem and let P^+ be the relaxed problem obtained by ignoring the negative effects (delete list) of every action

P^+ is called the delete-relaxation of P

P^+ is like P except that: $\text{EFF}^-(a) = \{\,\}$ for all a

A solution for P^+ is called a relaxed plan.

- $s = \{\text{on}(\text{A}, \text{B}), \text{clear}(\text{A}), \text{ontable}(\text{B}), \text{holding}(\text{R1}, \text{C})\}$
- $a = \text{putdown}(\text{R1}, \text{C})$
  precondition $\{\text{holding}(\text{R1}, \text{C})\}$
  effect $\{\text{ontable}(\text{C}), \text{clear}(\text{C}), \text{handempty}(\text{R1}), \cancel{\neg\text{holding}(\text{R1}, \text{C})}\}$
- $\gamma(s, a) = \{\text{on}(\text{A}, \text{B}), \text{clear}(\text{A}), \text{ontable}(\text{B}), \text{holding}(\text{R1}, \text{C}), \text{ontable}(\text{C}), \text{clear}(\text{C}), \text{handempty}(\text{R1})\}$

Delete relaxation - intuition

Delete-relaxed planning: once a fact becomes true, it remains true forever

Real World (before)

Delete relaxation - intuition

Delete-relaxed planning: once a fact becomes true, it remains true forever

Real World (after)

Delete relaxation - intuition

Delete-relaxed planning: once a fact becomes true, it remains true forever

Relaxed World (before)

Delete relaxation - intuition

Delete-relaxed planning: once a fact becomes true, it remains true forever

Relaxed World (after)

Delete relaxation - intuition

Delete-relaxed planning: once a fact becomes true, it remains true forever

Real World (before)

Delete relaxation - intuition

Delete-relaxed planning: once a fact becomes true, it remains true forever

Real World (after)

Delete relaxation - intuition

Delete-relaxed planning: once a fact becomes true, it remains true forever

Relaxed World (before)

Delete relaxation - intuition

Delete-relaxed planning: once a fact becomes true, it remains true forever

Relaxed World (after)

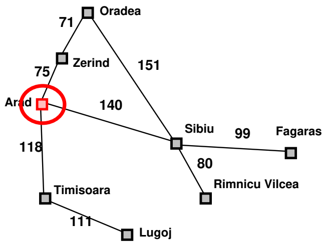
Delete relaxation - intuition

Delete-relaxed planning: once a fact becomes true, it remains true forever

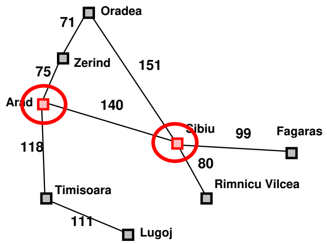
fill $\longrightarrow$

drink $\longrightarrow$

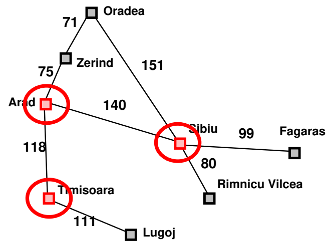
drink . . . $\longrightarrow$ . . .



go(A,S) $\longrightarrow$



go(A,T) $\longrightarrow$



go(S,F) . . . $\longrightarrow$ . . .

Once an action is applicable, it remains applicable

An action does not need to be applied more than once in a relaxed plan

Delete relaxation heuristics

The cost h^+ of an optimal solution to P^+ is a lower bound on the cost of an optimal solution of for P, hence an admissible heuristic

But, finding an optimal solution for P^+ is NP-hard
(PLANMIN is NP-complete for problems with only positive effects)

$\rightarrow$ Need to further relax the problem to get efficient admissible heuristics
gives the h^{max} heuristic [Bonet & Geffner, 1999]

Finding an arbitrary solution for P^+ (PLANSAT) is polynomial
(PLANSAT is polynomial with only positive effects)

$\rightarrow$ Relaxed plans are used to derive inadmissible heuristics
gives the h^{FF} heuristic [Hoffmann & Nebel, AIJ 2001]

$h^{\mathbf{max}}$ and $h^{\mathbf{sum}}$ heuristics

Relax the problem by ignoring the negative effects EFF^-

Further relax the problem by ignoring interactions between subgoals

Heuristics $h(s,g)$ estimate the minimum cost from s to g. When $g = \{p\}$:

- $h(s,\{p\}) = 0$ if $p \in s$
- $h(s,\{p\}) = \infty$ if $p \notin s$ and $\forall a \in A, p \notin \text{EFF}^+(a)$
- $h(s,\{p\}) = \min\limits_{\substack{a \in A \\ p \in \text{EFF}^+(a)}} [h(s, \text{PRE}(a)) + c(a)]$ otherwise

- admissible h^{max} heuristic: cost to reach a set is the max of costs. looks at the critical path: $h(s,g) = \max_{p \in g} h(s,\{p\})$
- non-admissible h^{sum} heuristic: cost to reach a set is the sum of costs. assumes subgoal independence: $h(s,g) = \sum_{p \in g} h(s,\{p\})$
- admissible: cost to reach a set is the max of costs to reach each *pair*. generalisation h^m heuristic: max of costs to reach each subset of size m

Computing $h^{\max}$

Let n be a node labelled by state s in an A* search. We need to compute $h(s, g)$ to evaluate n and put it in frontier.

It suffices to compute $h(s, \{p\})$ for each proposition p.

for each proposition p
    if $p \in s$ then $H[p] \leftarrow 0$ else $H[p] \leftarrow \infty$
repeat:
    for each action a
        $H[a] \leftarrow \max_{p \in \mathrm{PRE}(a)} H[p]$
        for each proposition $p \in \textsc{eff}^{+}(a)$
            $H[p] \leftarrow \min(H[p], H[a] + c(a))$
until a fixed point is reached
return $h(s, g) = \max_{p \in g} H[p]$

Polynomial time algorithm. For h^{sum}, replace $\max$ with $\sum$.

$h^{\mathbf{FF}}$ heuristic

Delete-relaxation heuristics so far:

- h^+ too hard to compute
- h^{max} not very informative
- h^{sum} can greatly over-estimate h^*

New heuristic h^{FF}:

- inadmissible
- compromise between h^{max} and h^{sum}
- makes sense when actions have unit costs
- relaxed reachability + relaxed plan extraction in plan graph without mutex

$h^{\textbf{FF}}$ heuristic - relaxed reachability

Level 0

ontable(A)

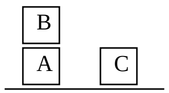
on(B,A)

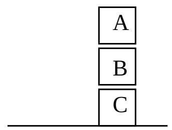
clear(B)

handempty(R)

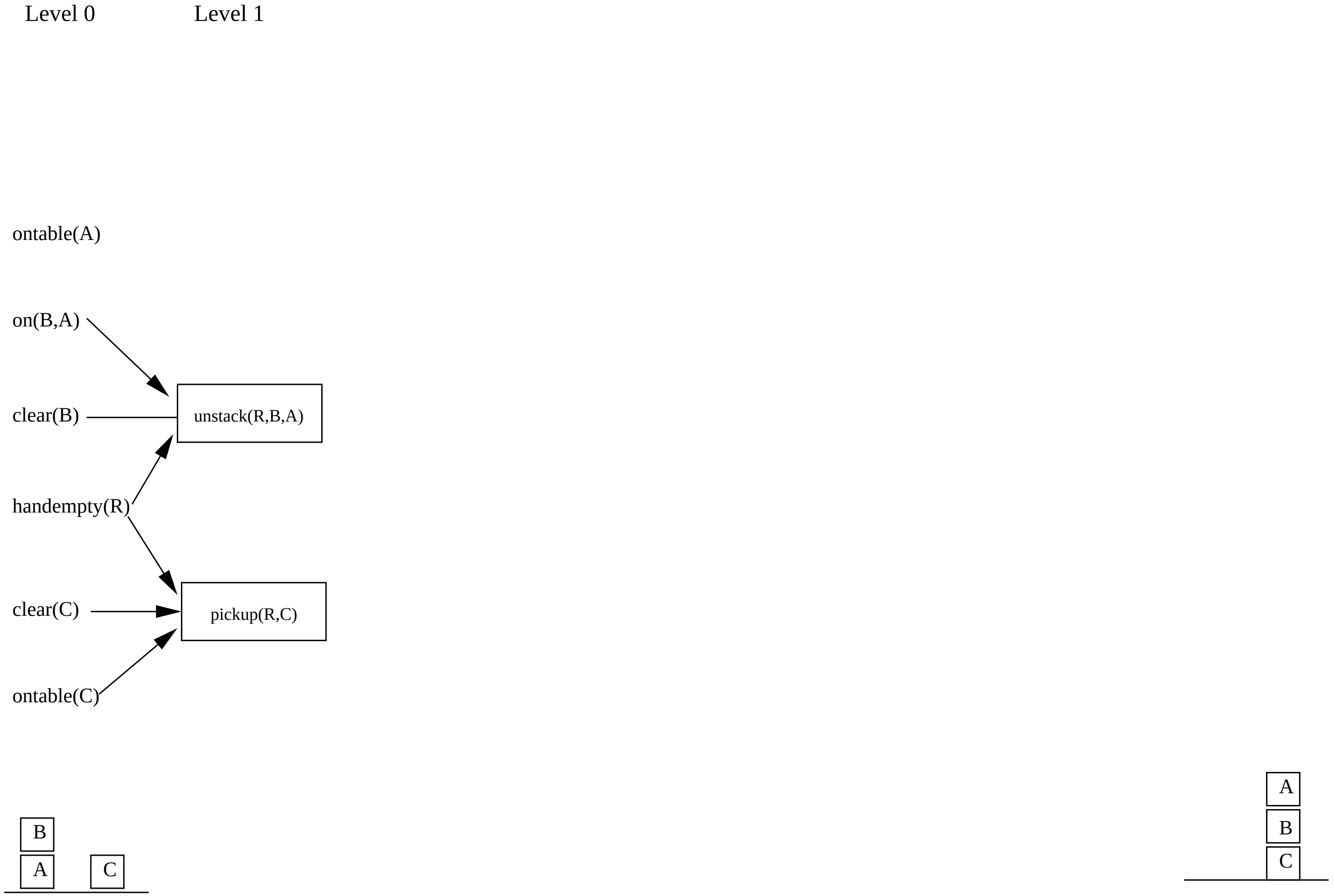
clear(C)

ontable(C)

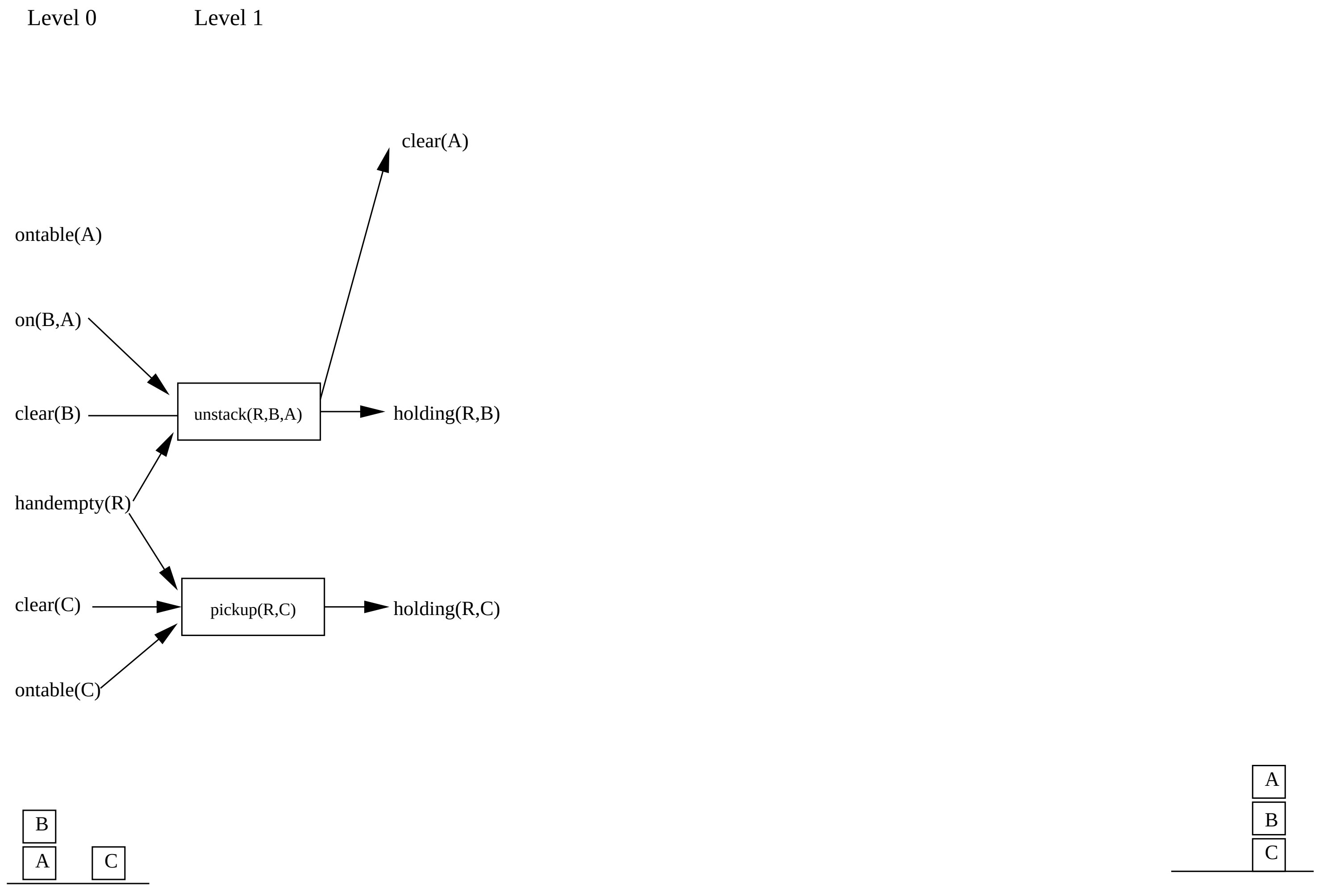




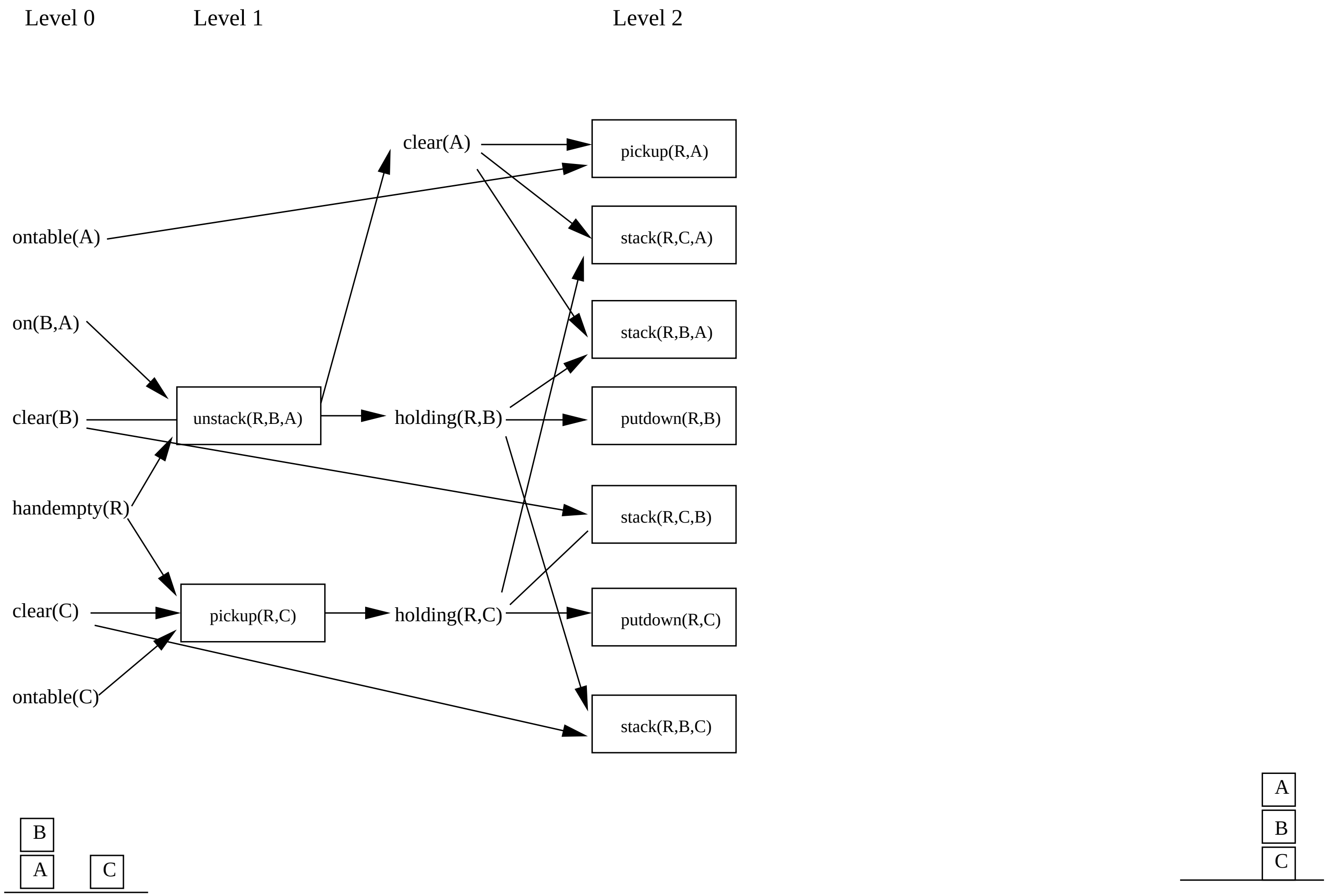
$h^{\mathbf{FF}}$ heuristic - relaxed reachability



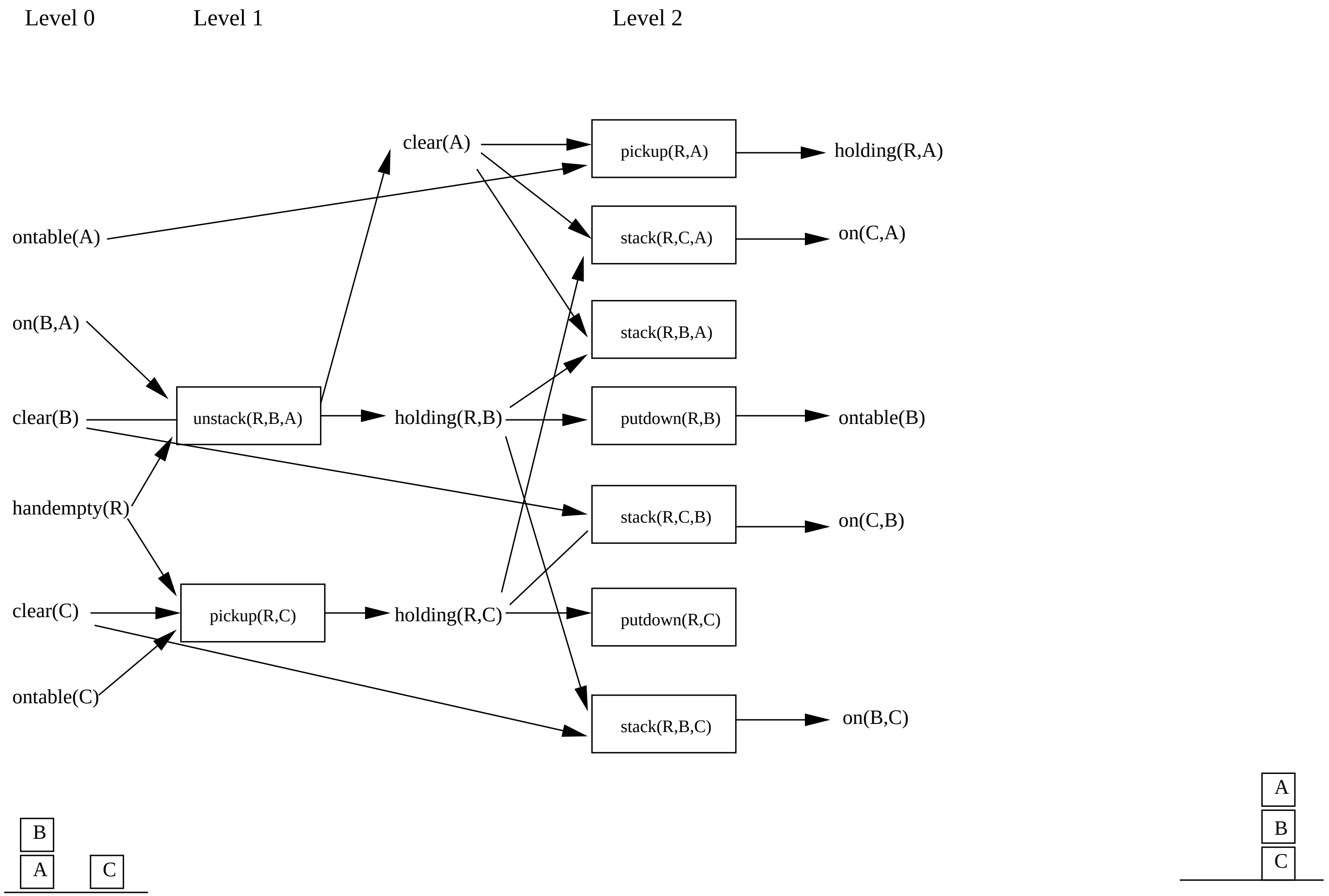
$h^{\mathbf{FF}}$ heuristic - relaxed reachability



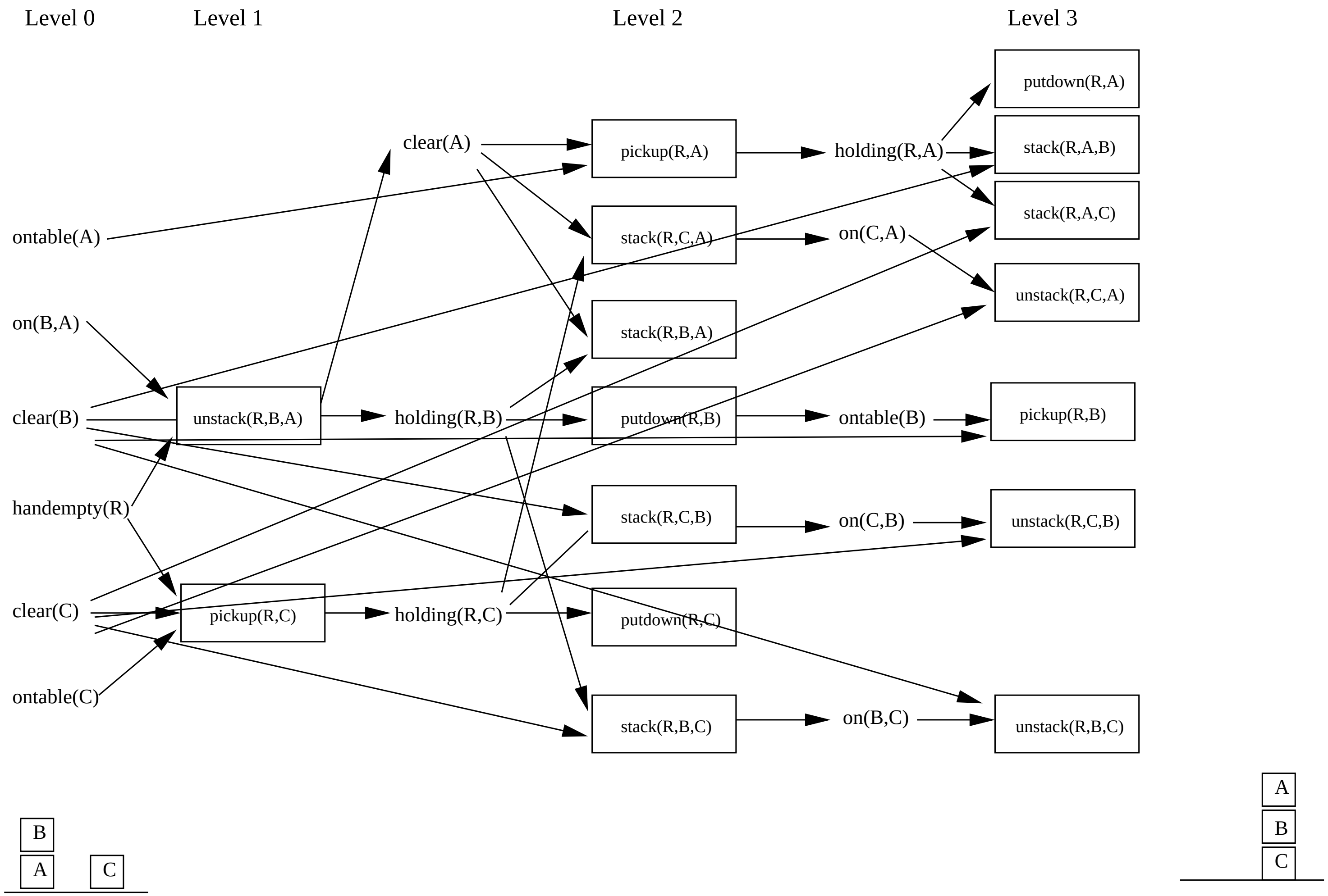
$h^{\mathbf{FF}}$ heuristic - relaxed reachability



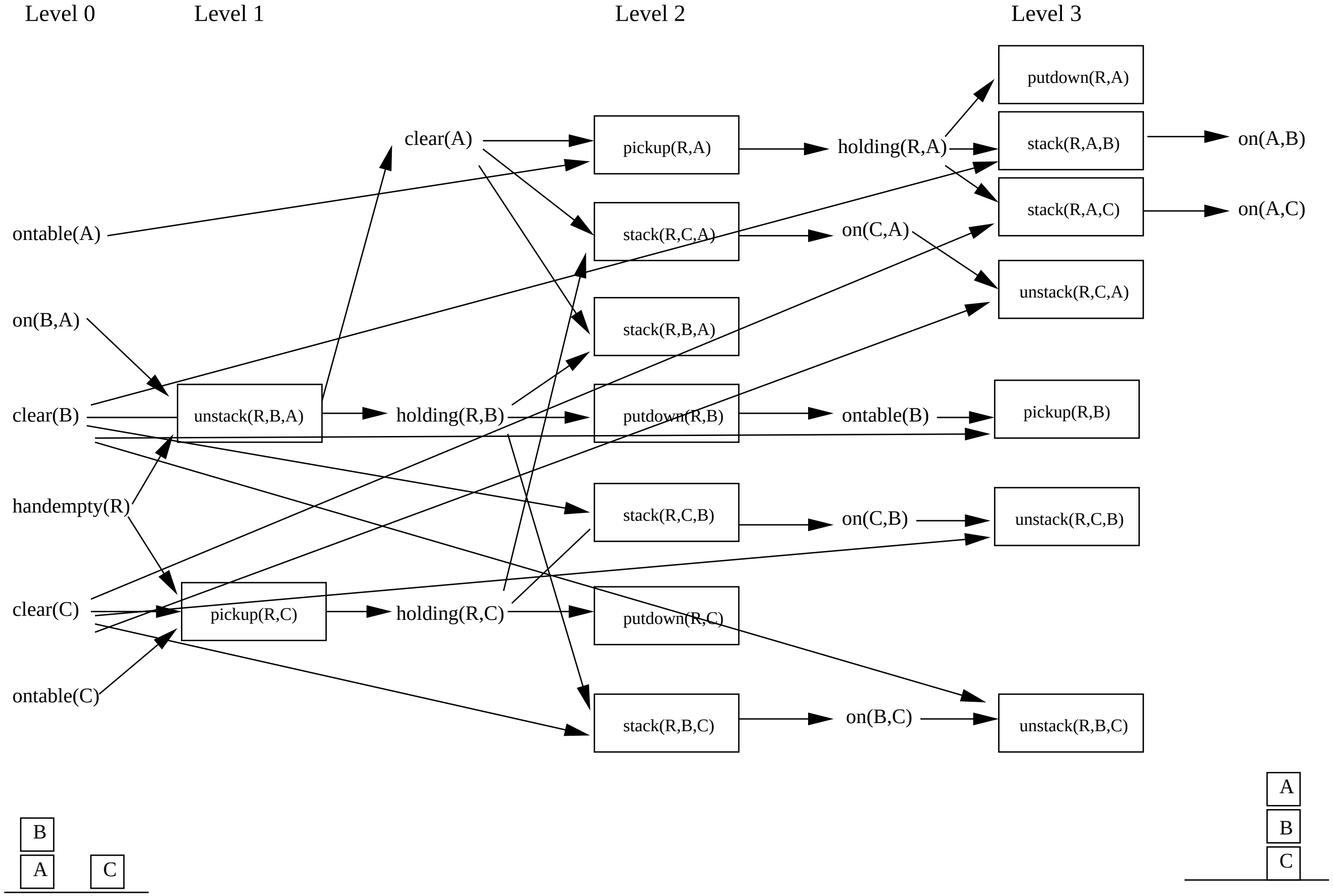
$h^{\mathbf{FF}}$ heuristic - relaxed reachability



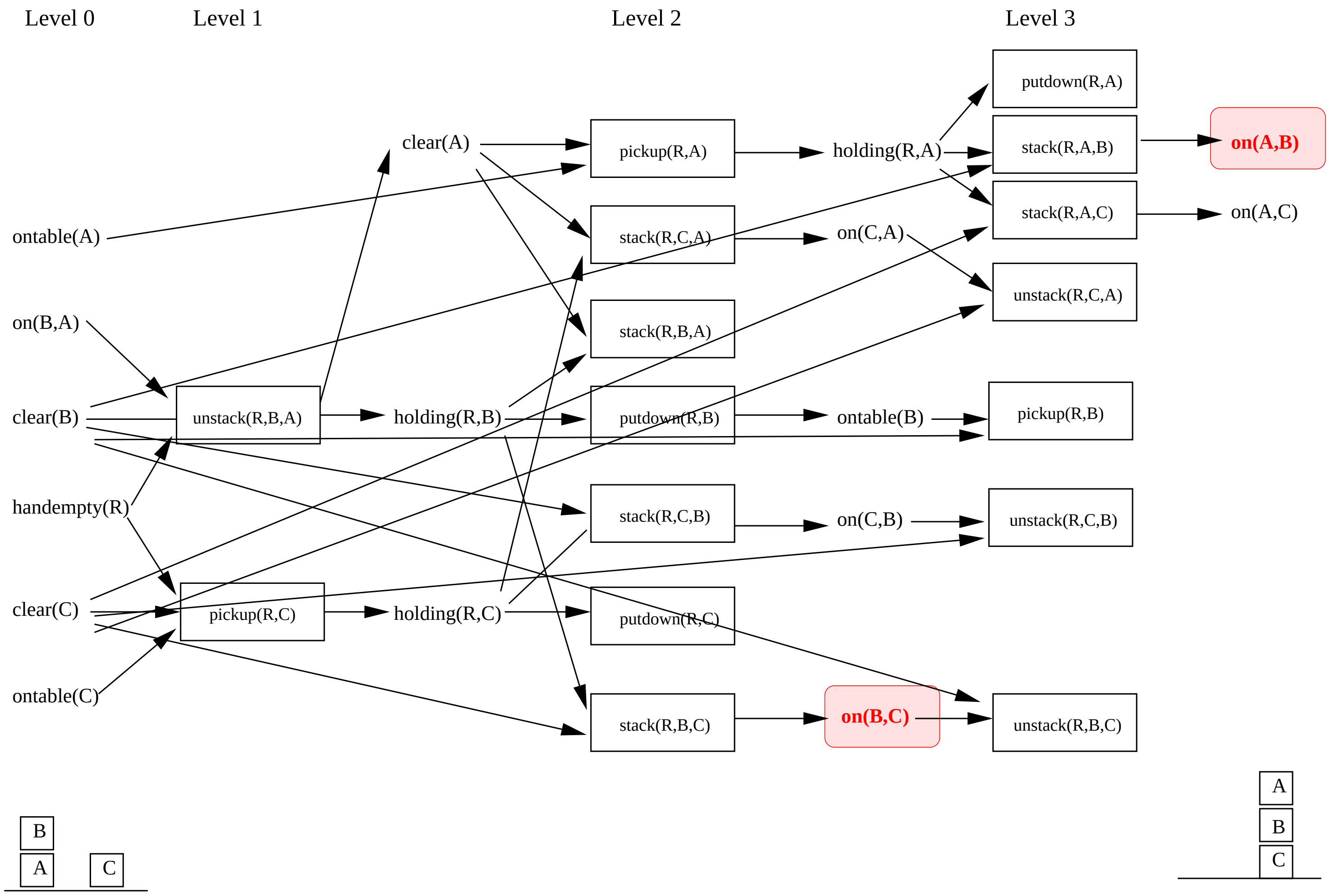
$h^{\mathbf{FF}}$ heuristic - relaxed reachability



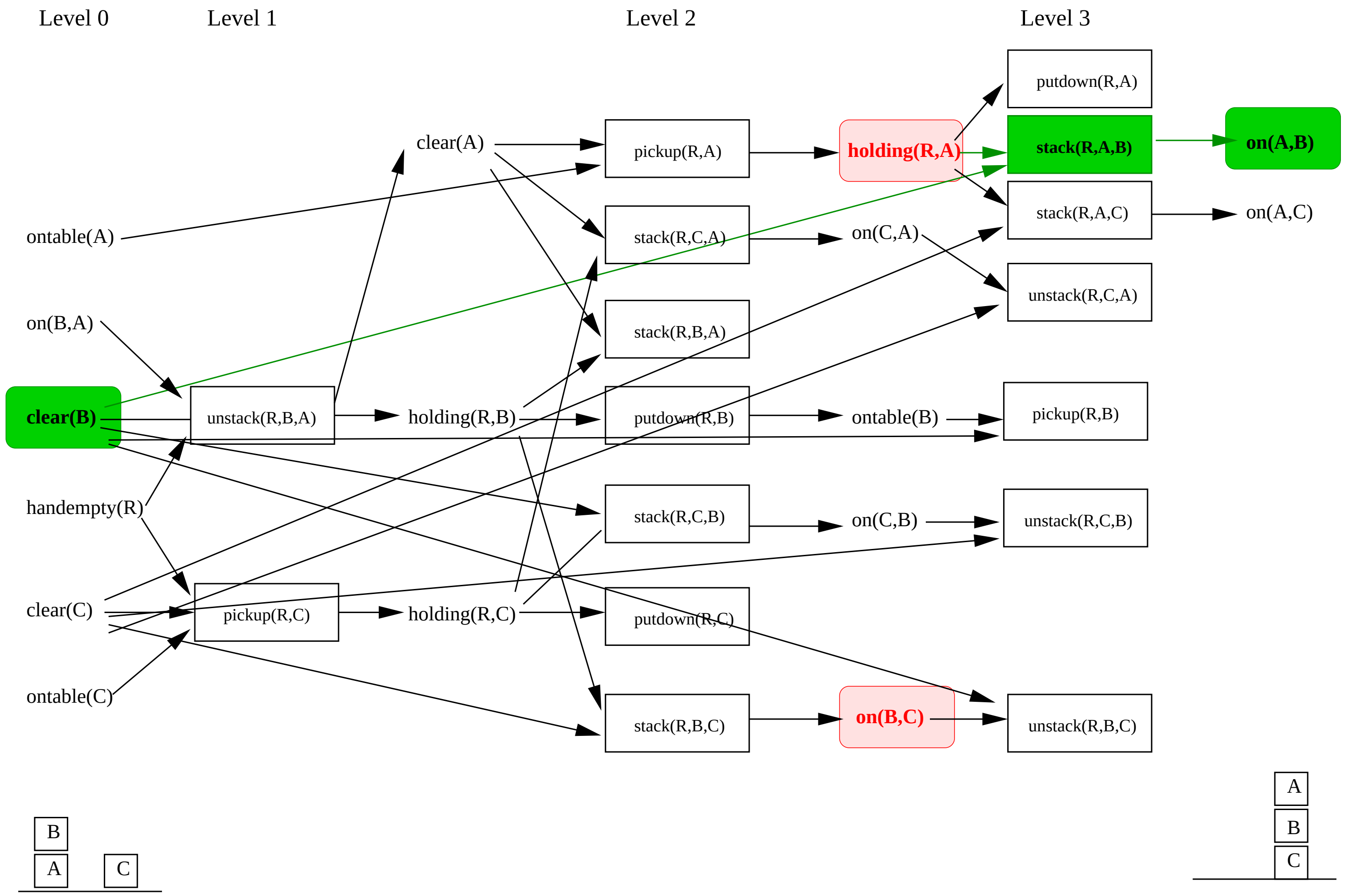
$h^{\mathbf{FF}}$ heuristic - relaxed reachability



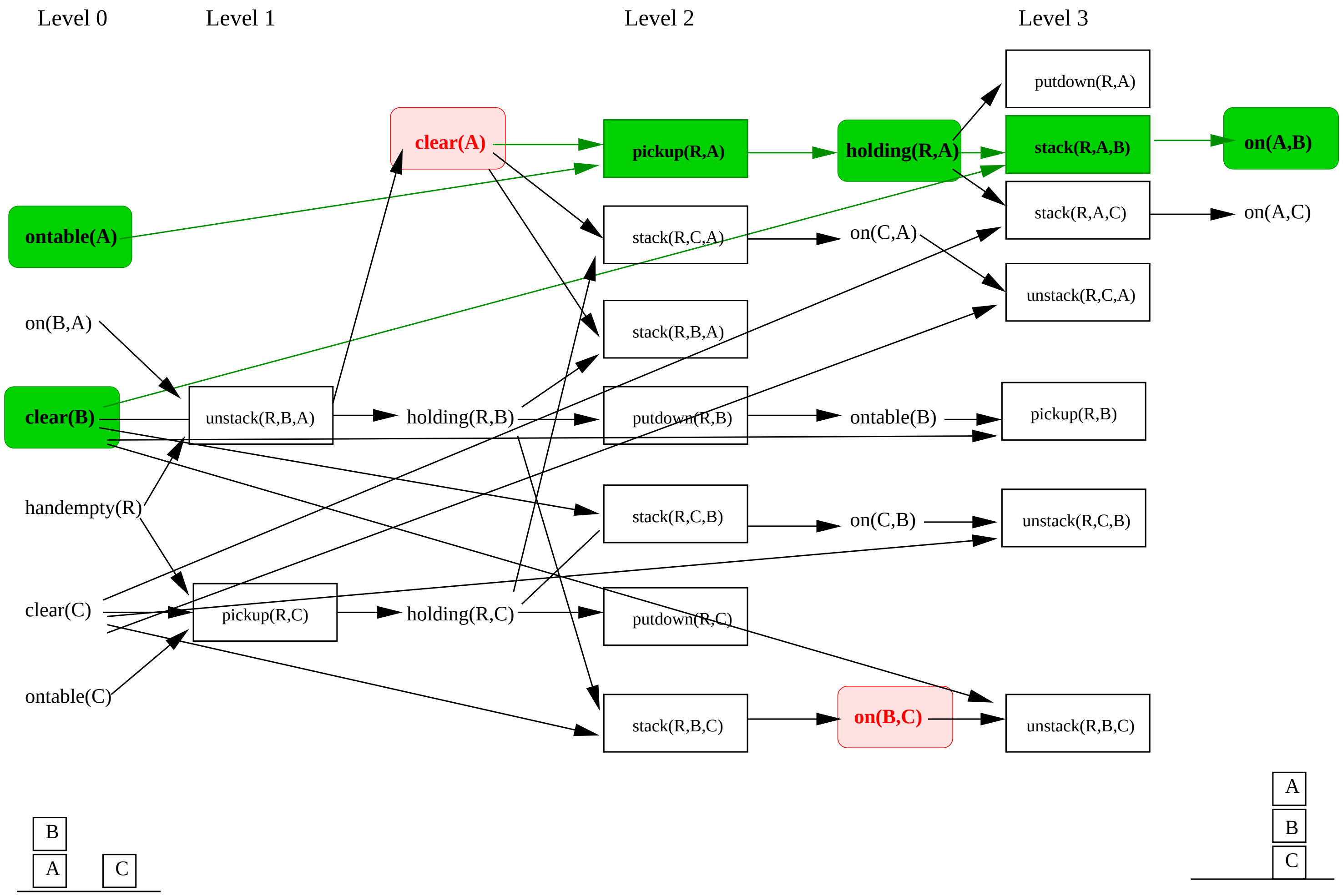
$h^{\mathbf{FF}}$ heuristic - plan extraction



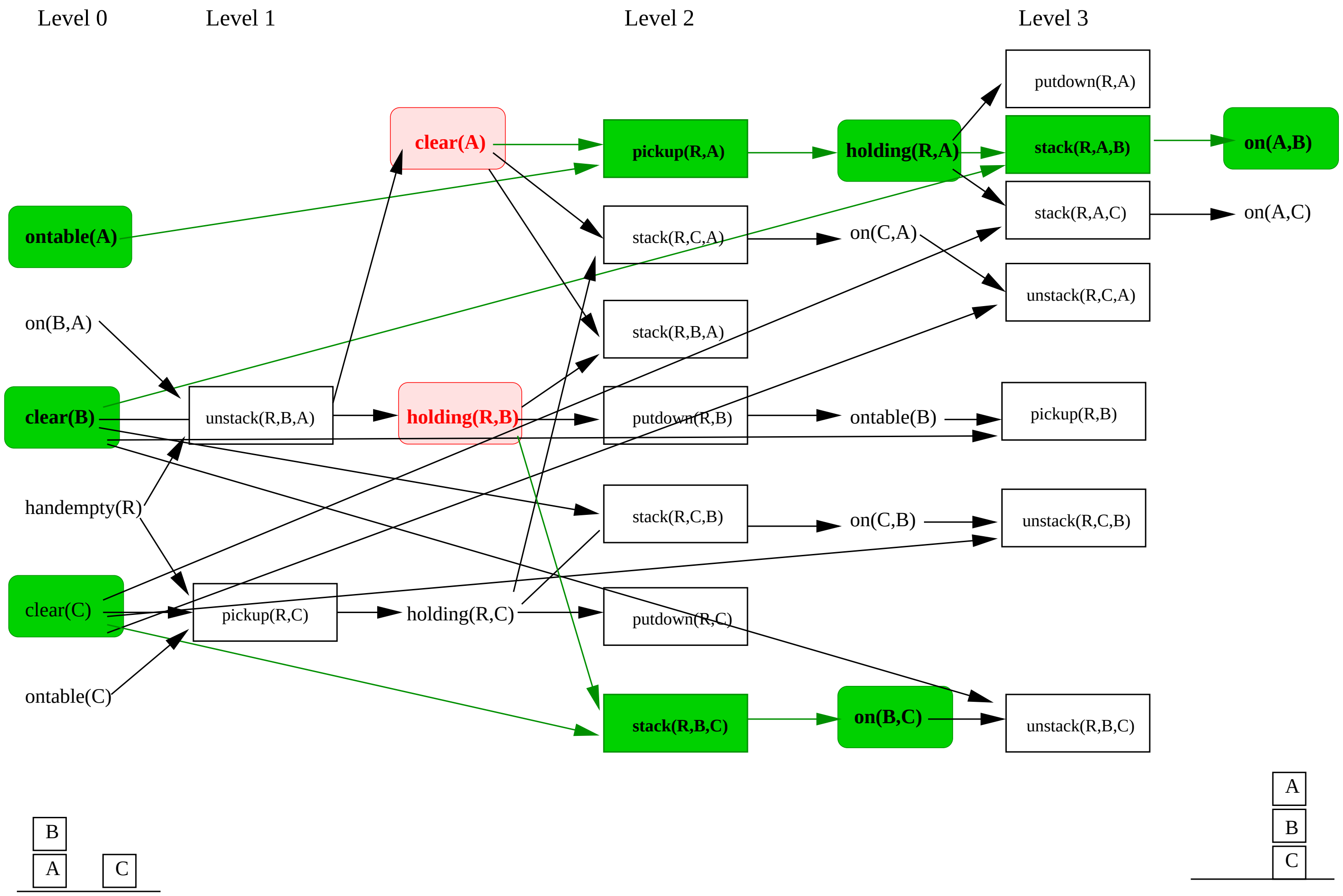
$h^{\mathbf{FF}}$ heuristic - plan extraction



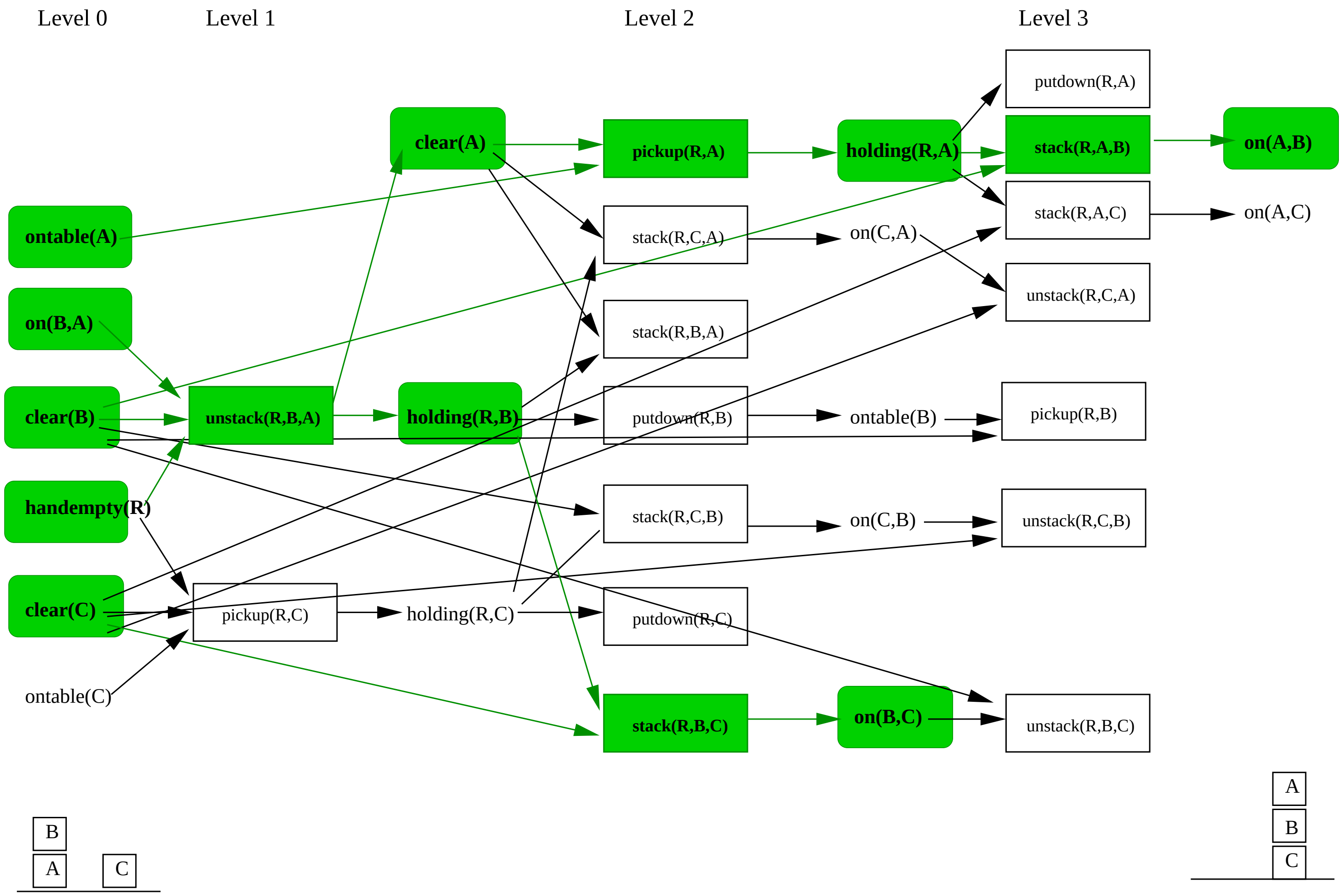
$h^{\mathbf{FF}}$ heuristic - plan extraction



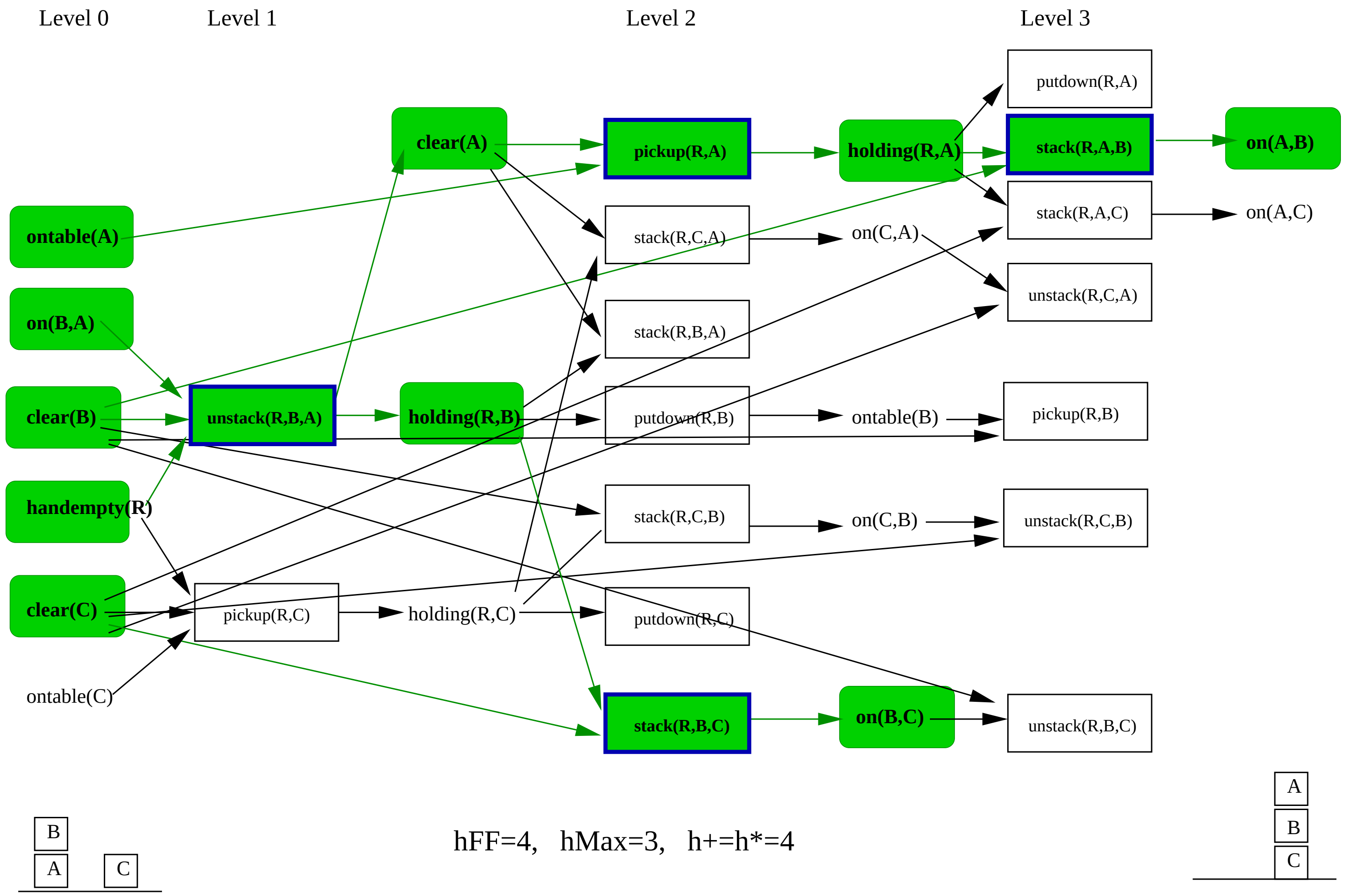
$h^{\mathbf{FF}}$ heuristic - plan extraction



h^{FF} heuristic - plan extraction



$h^{\mathbf{FF}}$ heuristic - plan extraction



hFF=4, hMax=3, h+=h*=4

$h^{\mathbf{FF}}$ heuristic

Delete-relaxation heuristics so far:

- h^{+} too hard to compute
- h^{max} not very informative
- h^{sum} can greatly over-estimate h^{*}

New heuristic h^{FF}:

- inadmissible
- compromise between h^{max} and h^{sum}
- makes sense when actions have unit costs
- relaxed reachability + relaxed plan extraction in plan graph without mutex

$\Rightarrow$ h^{max}: first level in a planning graph in which the goal appears

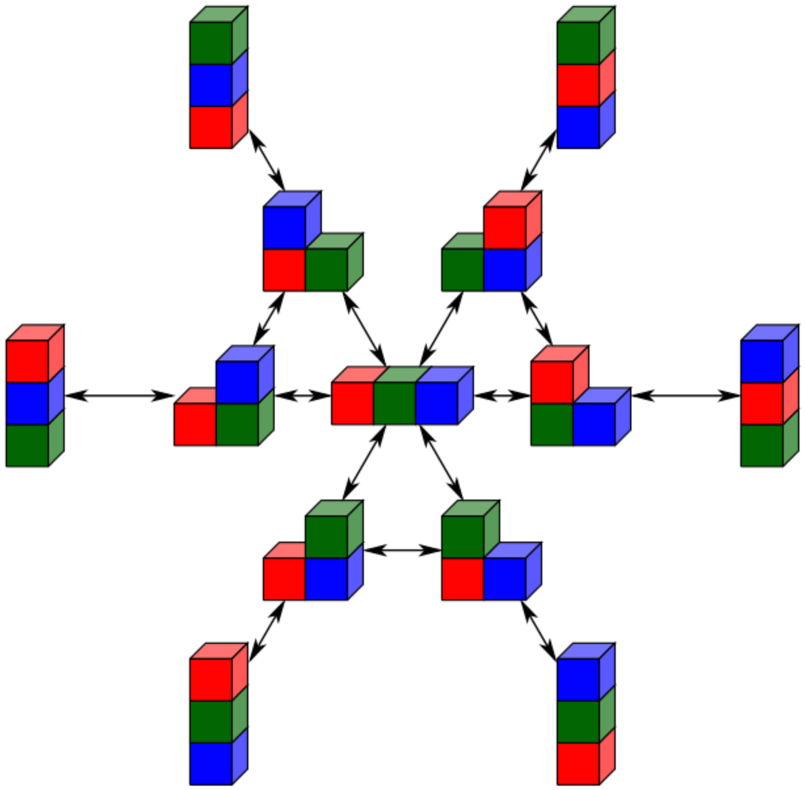
$\Rightarrow$ h^{FF}:is the number of actions in the relaxed plan

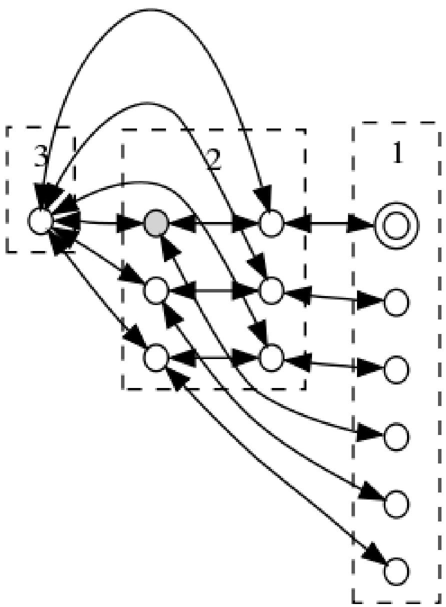
Abstraction heuristics

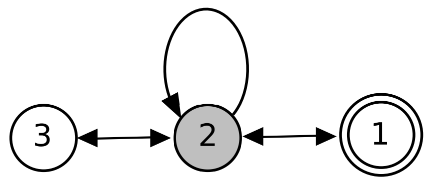
Simplify the problem by ignoring **parts** of it.

- Drop preconditions from actions
- Consider only a subset of predicates/propositions
- Count objects with a given property, ignoring the identity of objects
  e.g. count clear blocks
- Ignore so much that the abstract problem is small enough to be solved by uninformed search
- Use memory to avoid repeated searches (pattern databases)

Example: counting clear blocks







Formal definition

Problem $P' = (S', A', \gamma', s'_0, S'_G, c')$ is an abstraction of $P = (S, A, \gamma, s_0, S_G, c)$ if there exists an abstraction mapping $\phi : S \mapsto S'$, such that:

- **ϕ preserves the initial state**:
  $\phi(s_0) = s'_0$
- **ϕ preserves goal states**:
  if $s \in S_G$ then $\phi(s) \in S'_G$
- **ϕ preserves transitions**:
  if $\gamma(s, a) = t$ then $\exists a' \in A'\ \gamma'(\phi(s), a') = \phi(t)$ with $c'(a') \leq c(a)$

The abstraction heuristic $h^\phi(s, g)$ induced by ϕ is given by the the cost of the optimal path from $\phi(s)$ to $\phi(g)$ in P'

Theorem: h^ϕ is admissible (and consistent).

With the STRIPS representation, pattern database heuristics are defined by projecting the states on a subset of propositions (the pattern).

Landmark heuristics

Proposition l is a landmark for problem P iff all plans for P make l true.
e.g. clear(B) is a landmark if any block below B is misplaced.

Landmark heuristic: counts the number of yet unachieved landmarks.
generalisation of the number of unachieved goals heuristic
used in the LAMA planner [Richter, AAAI 2008]

Inadmissible (even if action costs are 1) as it assumes landmark independence. Admissible versions exist.

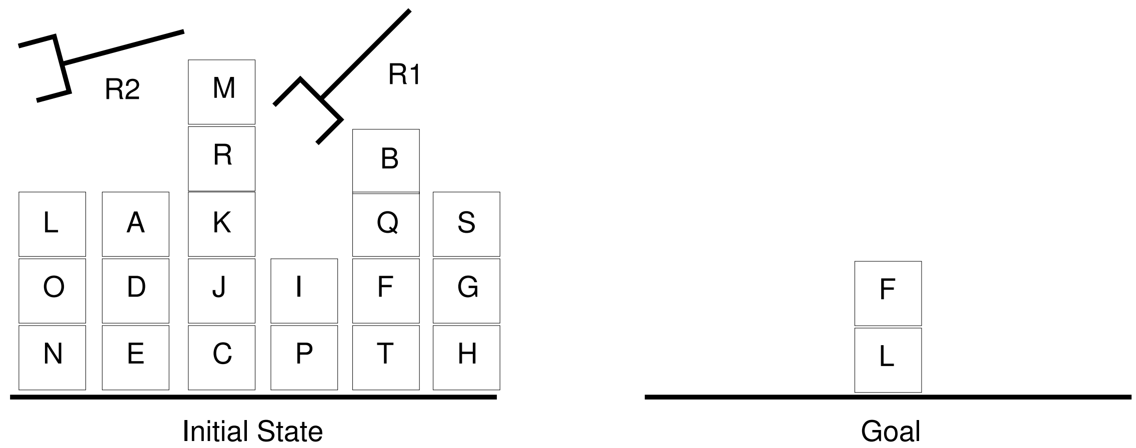
Sufficient condition for proposition l to be a landmark for problem P: the delete relaxation P^+ is not solvable when l is removed from the add-list of all actions.

A complete landmark set can be computed in polynomial time for the delete relaxation, once as pre-processing. Gives a set of landmarks for P.

The current best heuristics are landmark heuristics variants

Regression planning (backward search)

For some problems, goal directed search pays



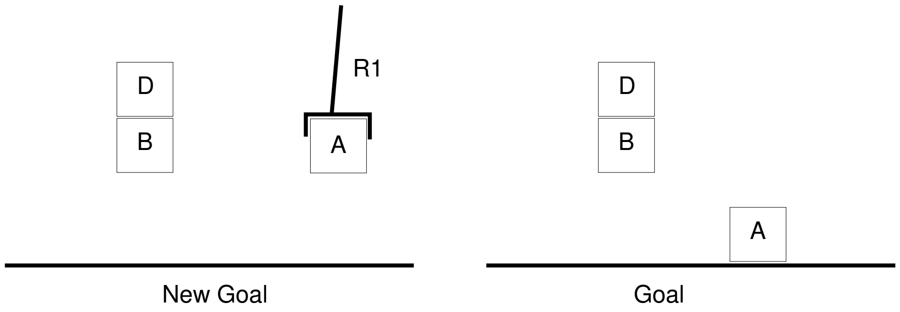
For forward search, we started at the initial state and computed state transitions, leading to a new state $s' = \gamma(s, a)$

For backward search, we start at the goal and compute inverse state transitions a.k.a regression, leading to a new **goal** $g' = \gamma^{-1}(g, a)$

What do we really mean by $\gamma^{-1}(g, a)$?? First we need to define relevance

Regression - relevance

- An action a is relevant for goal g if:
  - it makes at least one of g's propositions true: $g \cap \text{EFF}^{+}(a) \neq \{\,\}$
  - it does not make any of g's proposition false: $g \cap \text{EFF}^{-}(a) = \{\,\}$
- If a is relevant for g then: $\gamma^{-1}(g,a) = (g \setminus \text{EFF}^{+}(a)) \cup \text{PRE}(a)$



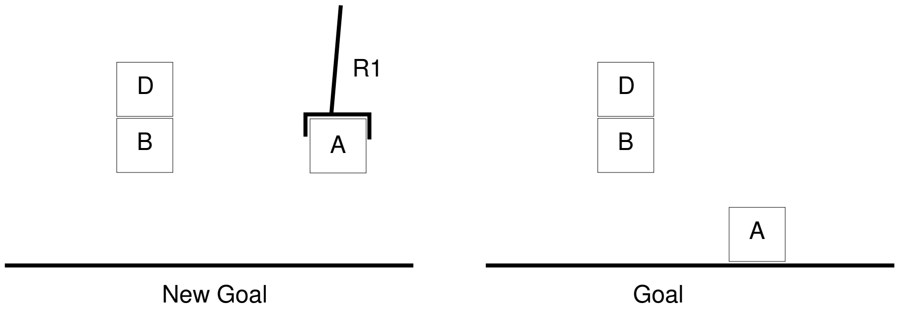
- Example:
  - $g = \{\text{on}(\text{D},\text{B}), \text{clear}(\text{D}), \text{ontable}(\text{A}), \text{clear}(\text{A})\}$
  - $a = \text{putdown}(\text{R1},\text{A})$

operator	$\text{putdown}(r,x)$
precondition	$\{\text{holding}(r,x)\}$
effect	$\{\text{ontable}(x), \text{clear}(x), \text{handempty}(r), \neg\text{holding}(r,x)\}$

  - $\gamma^{-1}(g,a) = \{\text{on}(\text{D},\text{B}), \text{clear}(\text{D}), \text{holding}(\text{R1},\text{A})\}$

Regression - relevance

- An action a is relevant for goal g if:
  - it makes at least one of g's propositions true: $g \cap \text{EFF}^+(a) \neq \{\,\}$
  - it does not make any of g's proposition false: $g \cap \text{EFF}^-(a) = \{\,\}$
- If a is relevant for g then: $\gamma^{-1}(g, a) = (g \setminus \text{EFF}^+(a)) \cup \text{PRE}(a)$



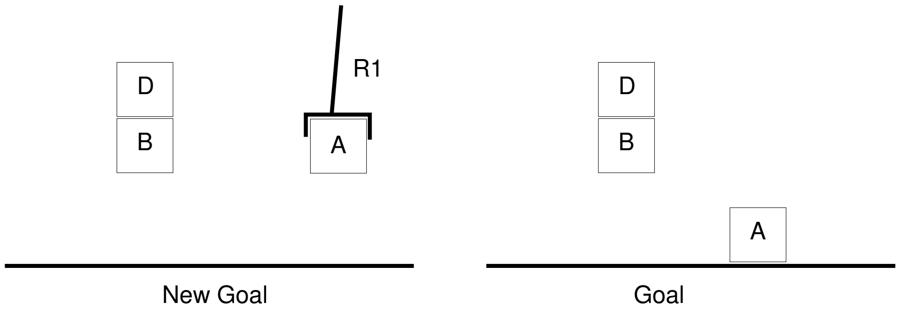
- Example:
  - $g = \{\text{on}(\text{D}, \text{B}), \text{clear}(\text{D}), \text{ontable}(\text{A}), \text{clear}(\text{A})\}$
  - $a = \text{putdown}(\text{R1}, \text{A})$

operator	$\text{putdown}(r, x)$
precondition	$\{\text{holding}(\text{R1}, \text{A})\}$
effect	$\{\text{ontable}(\text{A}), \text{clear}(\text{A}), \text{handempty}(\text{R1}), \neg\text{holding}(\text{R1}, \text{A})\}$

  - $\gamma^{-1}(g, a) = \{\text{on}(\text{D}, \text{B}), \text{clear}(\text{D}), \text{holding}(\text{R1}, \text{A})\}$

Regression - relevance

- An action a is relevant for goal g if:
  - it makes at least one of g's propositions true: $g \cap \text{EFF}^{+}(a) \neq \{\,\}$
  - it does not make any of g's proposition false: $g \cap \text{EFF}^{-}(a) = \{\,\}$
- If a is relevant for g then: $\gamma^{-1}(g, a) = (g \setminus \text{EFF}^{+}(a)) \cup \text{PRE}(a)$



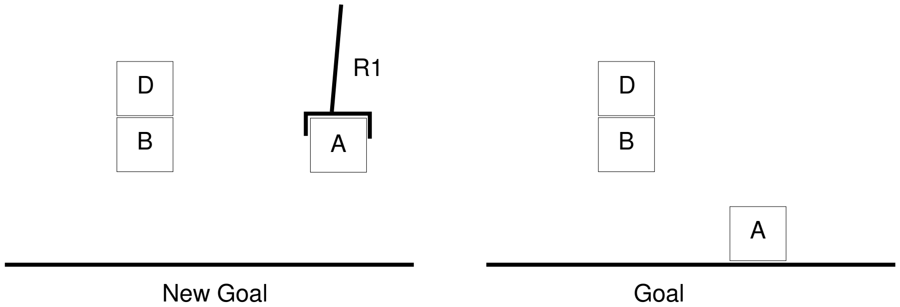
- Example:
  - $g = \{\text{on}(\text{D},\text{B}), \text{clear}(\text{D}), \underline{\text{ontable}(\text{A}), \text{clear}(\text{A})}\}$
  - $a = \text{putdown}(\text{R1},\text{A})$

operator	$\text{putdown}(r, x)$
precondition	$\{\text{holding}(\text{R1},\text{A})\}$
effect	$\{\text{ontable}(\text{A}), \text{clear}(\text{A}), \text{handempty}(\text{R1}), \neg\text{holding}(\text{R1},\text{A})\}$

  - $\gamma^{-1}(g, a) = \{\text{on}(\text{D},\text{B}), \text{clear}(\text{D}), \text{holding}(\text{R1},\text{A})\}$

Regression - relevance

- An action a is relevant for goal g if:
  - it makes at least one of g's propositions true: $g \cap \text{EFF}^{+}(a) \neq \{\,\}$
  - it does not make any of g's proposition false: $g \cap \text{EFF}^{-}(a) = \{\,\}$
- If a is relevant for g then: $\gamma^{-1}(g, a) = (g \setminus \text{EFF}^{+}(a)) \cup \text{PRE}(a)$



- Example:
  - $g = \{\text{on}(D, B), \text{clear}(D), \cancel{\text{ontable}(A), \text{clear}(A)}\}$
  - $a = \text{putdown}(R1, A)$

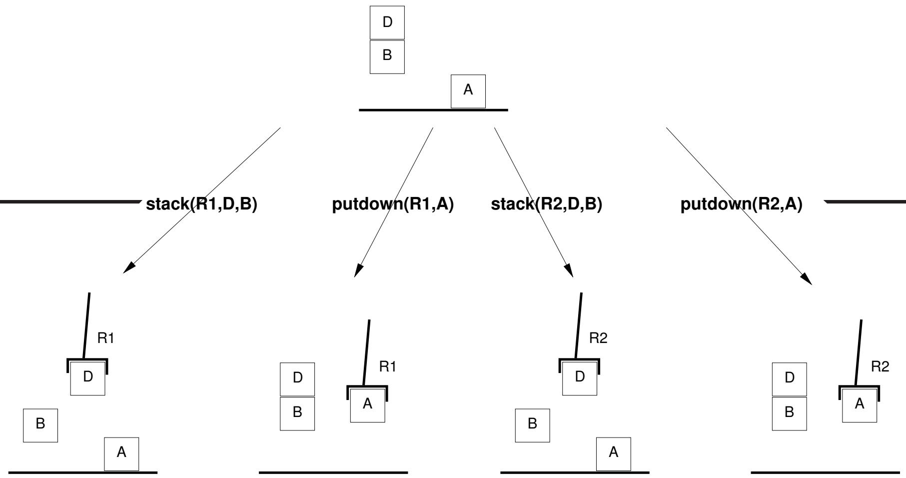
operator	$\text{putdown}(r, x)$
precondition	$\{\text{holding}(R1, A)\}$
effect	$\{\text{ontable}(A), \text{clear}(A), \text{handempty}(R1), \neg\text{holding}(R1, A)\}$

  - $\gamma^{-1}(g, a) = \{\text{on}(D, B), \text{clear}(D), \underline{\text{holding}(R1, A)}\}$

Regression planning (backward search)

function BACKWARD-SEARCH(O, s_0, g) **returns** an action sequence, or failure

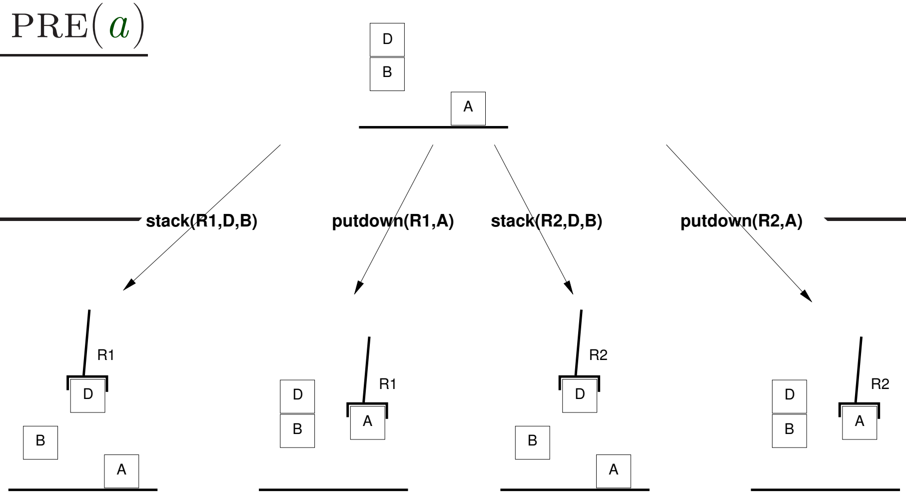
$\pi \leftarrow \langle\rangle$
loop do
  if s_0 satisfies g **then return** π
  $E \leftarrow \{a \mid a$ is a ground instance of an operator in O
    such that a is relevant for $g\}$
  if $E = \{\}$ **then return** failure
  choose an action $a \in E$
  $g \leftarrow \gamma^{-1}(g, a)$
  $\pi \leftarrow a.\pi$
end



Regression planning (backward search)

function BACKWARD-SEARCH(O, s_0, g) **returns** an action sequence, or failure

$\pi \leftarrow \langle\rangle$
loop do
  if $\underline{g \subseteq s_0}$ **then return** π
  $E \leftarrow \{a \mid a$ is a ground instance of an operator in O
    $\underline{\text{such that } g \cap \text{EFF}^+(a) \neq \{\} \text{ and } g \cap \text{EFF}^-(a) = \{\}\}}$
  if $E = \{\}$ **then return** failure
  choose an action $a \in E$
  $\underline{g \leftarrow (g \setminus \text{EFF}^+(a)) \cup \text{PRE}(a)}$
  $\pi \leftarrow a.\pi$
end



Properties of Backward-Search

Backward-Search can be used in conjunction with any search strategy to implement **choose**, breadth-first search, depth-first search, iterative-deepening, greedy search, A*, IDA*, ...

Backward-Search is sound: any plan returned is guaranteed to be a solution to the problem.

Backward-Search is complete: provided the underlying search strategy is complete, it will always return a solution to the problem if there is one.

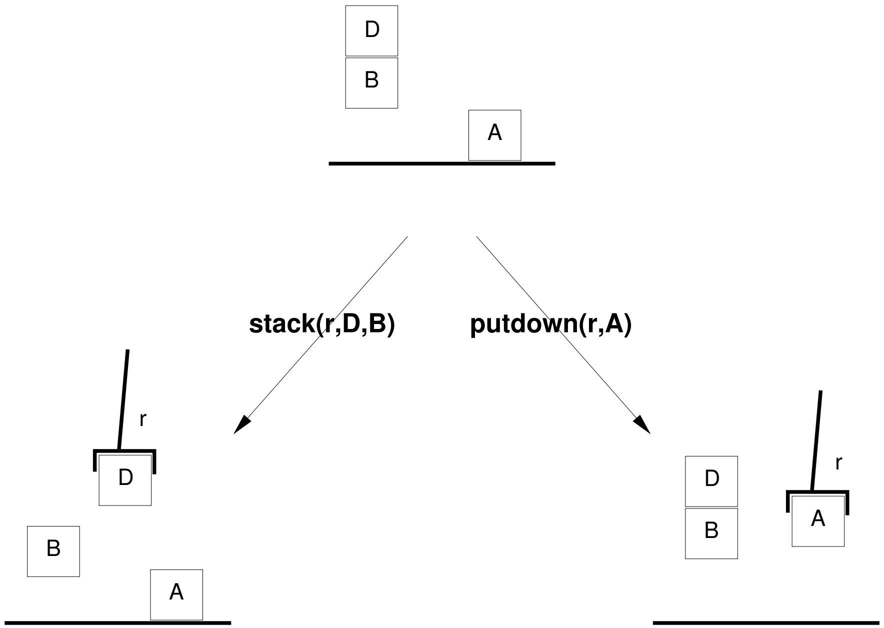
For instance, when used with breadth-first search it will be complete, when used with depth-first search it will be complete if the state space is finite – in general, need to detect and forbid loops, by checking that **no previous goal is a subset of the current one**.

Heuristics: many of the state-space heuristics are symmetric.
Forward: $h(s, g)$, backwards: $h(s_0, c)$ where c is the current goal.

Lifting

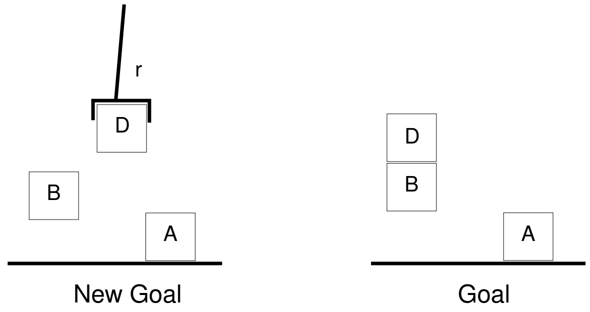
We can substancially reduce the branching factor if we only **partially instanciate** the operators.

For instance, in the Blocks World, we may not need to distinguish between using robot hand R1 and robot hand R2. Just any hand will do:



Lifted regression example

relevance: $g \cap \text{EFF}^+(a) \neq \{\,\}$, $g \cap \text{EFF}^-(a) = \{\,\}$
inverse transition: $\gamma^{-1}(g, a) = (g \setminus \text{EFF}^+(a)) \cup \text{PRE}(a)$



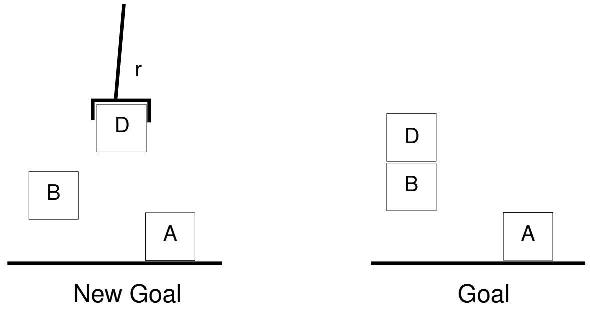
$$g \leftarrow \{\mathsf{on}(\mathsf{D}, \mathsf{B}), \mathsf{clear}(\mathsf{D}), \mathsf{ontable}(\mathsf{A}), \mathsf{clear}(\mathsf{A})\}$$

$o \leftarrow \mathsf{stack}(r, x, y)$

$$\begin{aligned} \text{PRE} &: \{\mathsf{holding}(r, x), \mathsf{clear}(y)\} \\ \text{EFF} &: \{\mathsf{on}(x, y), \mathsf{handempty}(r), \mathsf{clear}(x), \\ &\quad \neg\mathsf{holding}(r, x), \neg\mathsf{clear}(y)\} \end{aligned}$$

Lifted regression example

relevance: $g \cap \mathrm{EFF}^+(a) \neq \{\,\}$, $g \cap \mathrm{EFF}^-(a) = \{\,\}$
inverse transition: $\gamma^{-1}(g,a) = (g \setminus \mathrm{EFF}^+(a)) \cup \mathrm{PRE}(a)$



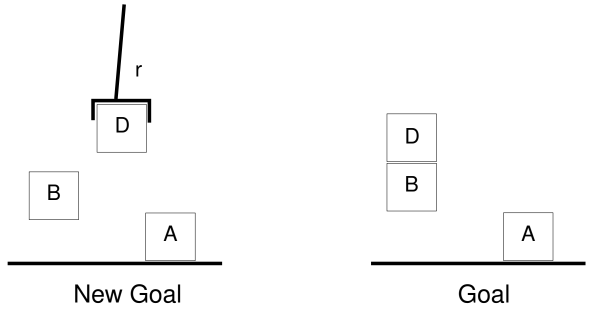
$g \leftarrow \{\underline{\mathsf{on}(\mathsf{D},\mathsf{B})}, \mathsf{clear}(\mathsf{D}), \mathsf{ontable}(\mathsf{A}), \mathsf{clear}(\mathsf{A})\}$

$o \leftarrow \mathsf{stack}(r,x,y)$
$\sigma \leftarrow \{x \leftarrow \mathsf{D}, y \leftarrow \mathsf{B}\}$

$\mathrm{PRE}: \{\mathsf{holding}(r,\mathsf{D}), \mathsf{clear}(\mathsf{B})\}$
$\mathrm{EFF}: \{\mathsf{on}(\mathsf{D},\mathsf{B}), \mathsf{handempty}(r), \mathsf{clear}(\mathsf{D}), \neg\mathsf{holding}(r,\mathsf{D}), \neg\mathsf{clear}(\mathsf{B})\}$

Lifted regression example

relevance: $g \cap \text{EFF}^{+}(a) \neq \{\,\}$, $g \cap \text{EFF}^{-}(a) = \{\,\}$
inverse transition: $\gamma^{-1}(g, a) = (g \setminus \text{EFF}^{+}(a)) \cup \text{PRE}(a)$



$$g \leftarrow \{\cancel{\mathsf{on}(\mathsf{D},\mathsf{B})}, \cancel{\mathsf{clear}(\mathsf{D})}, \mathsf{ontable}(\mathsf{A}), \mathsf{clear}(\mathsf{A})\}$$

$o \leftarrow \mathsf{stack}(r, x, y)$
$\sigma \leftarrow \{x \leftarrow \mathsf{D}, y \leftarrow \mathsf{B}\}$

$\text{PRE}:\ \{\mathsf{holding}(r, \mathsf{D}), \mathsf{clear}(\mathsf{B})\}$
$\text{EFF}:\ \{\mathsf{on}(\mathsf{D}, \mathsf{B}), \mathsf{handempty}(r), \mathsf{clear}(\mathsf{D}), \neg\mathsf{holding}(r, \mathsf{D}), \neg\mathsf{clear}(\mathsf{B})\}$

$$g \leftarrow \{\underline{\mathsf{holding}(r, \mathsf{D})}, \underline{\mathsf{clear}(\mathsf{B})}, \mathsf{ontable}(\mathsf{A}), \mathsf{clear}(\mathsf{A})\}$$

Lifted backward search

More complicated because we have to keep track of the substitutions performed.

function LIFTED-BACKWARD-SEARCH(O, s_0, g) **returns** an action sequence, or failure

$\pi \leftarrow \langle\rangle$
loop do
  if s_0 satisfies g **then return** π
  $E \leftarrow \{(o, \sigma) \mid$ o is an operator[a] in O relevant for g
    σ is a substitution that unifies[b] an atom of g
    and an atom of $\text{EFF}^+(o)$ to cause the relevance$\}$
  if $E = \{\,\}$ **then return** failure
  choose a pair $(o, \sigma) \in E$
  $g \leftarrow \gamma^{-1}(\sigma(g), \sigma(o))$
  $\pi \leftarrow \sigma(o).\sigma(\pi)$
end

[a] May need to be standardised by replacing variable symbols with new symbols that do not occur elsewhere.
[b] We take the most general unifier.

Properties of LIFTED-BACKWARD-SEARCH

LIFTED-BACKWARD-SEARCH can be used in conjunction with any search strategy to implement **choose**, breadth-first search, depth-first search, iterative-deepening, greedy search, A*, IDA*, . . .

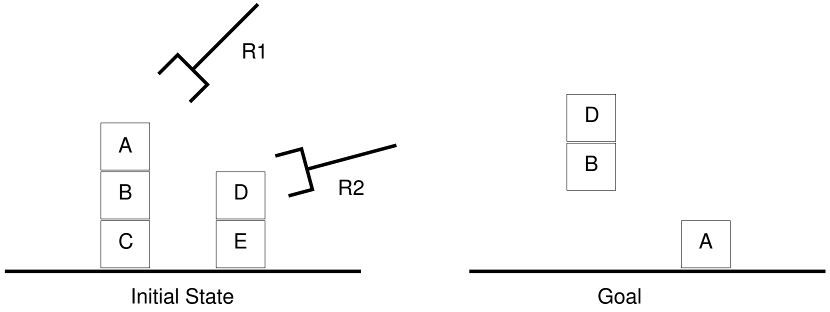
LIFTED-BACKWARD-SEARCH is sound: any plan returned is guaranteed to be a solution to the problem.

LIFTED-BACKWARD-SEARCH is complete: provided the underlying search strategy is complete, it will always return a solution to the problem if there is one.

For instance, when used with breadth-first search it will be complete, when used with depth-first search it will be complete if the state space is finite – in general, we need to detect and forbid loops, by checking that **no previous goal unifies with a subset of the current one**.

Trace example

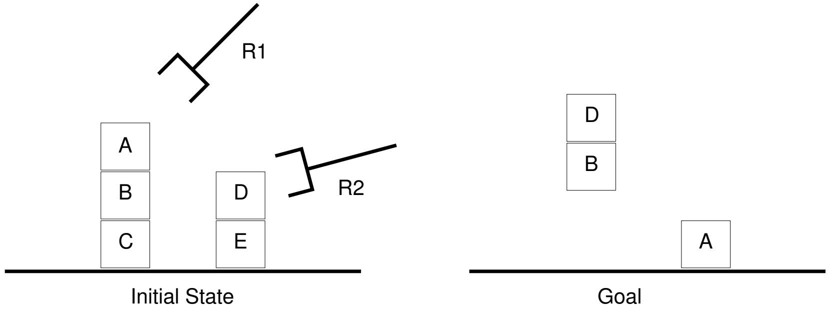
relevance: $g \cap \text{EFF}^+(a) \neq \{\,\}$, $g \cap \text{EFF}^-(a) = \{\,\}$
inverse transition: $\gamma^{-1}(g, a) = (g \setminus \text{EFF}^+(a)) \cup \text{PRE}(a)$



1. $g \leftarrow \{\mathsf{on}(\mathsf{D}, \mathsf{B}), \mathsf{clear}(\mathsf{D}), \mathsf{ontable}(\mathsf{A}), \mathsf{clear}(\mathsf{A})\}$
   $\pi \leftarrow \langle\rangle$

Trace example

relevance: $g \cap \text{EFF}^+(a) \neq \{\,\}$, $g \cap \text{EFF}^-(a) = \{\,\}$
inverse transition: $\gamma^{-1}(g, a) = (g \setminus \text{EFF}^+(a)) \cup \text{PRE}(a)$

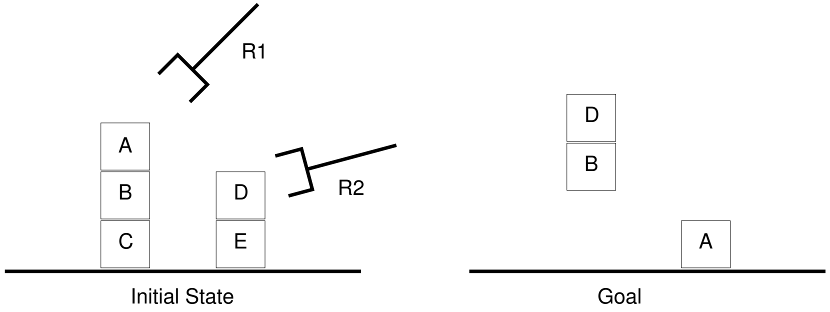


1. $g \leftarrow \{\underline{\mathsf{on}(\mathsf{D}, \mathsf{B})}, \mathsf{clear}(\mathsf{D}), \mathsf{ontable}(\mathsf{A}), \mathsf{clear}(\mathsf{A})\}$
   $\pi \leftarrow \langle\rangle$

   $o \leftarrow \mathsf{stack}(r, x, y)$, $\quad$ PRE : $\{\mathsf{holding}(r, \mathsf{D}), \mathsf{clear}(\mathsf{B})\}$
   $\sigma \leftarrow \{x \leftarrow \mathsf{D}, y \leftarrow \mathsf{B}\}$ $\quad$ EFF : $\{\mathsf{on}(\mathsf{D}, \mathsf{B}), \mathsf{handempty}(r), \mathsf{clear}(\mathsf{D}), \neg\mathsf{holding}(r, \mathsf{D}), \neg\mathsf{clear}(\mathsf{B})\}$

Trace example

relevance: $g \cap \text{EFF}^+(a) \neq \{\,\}$, $g \cap \text{EFF}^-(a) = \{\,\}$
inverse transition: $\gamma^{-1}(g,a) = (g \setminus \text{EFF}^+(a)) \cup \text{PRE}(a)$



1. $g \leftarrow \{\cancel{\mathsf{on}(\mathsf{D},\mathsf{B})}, \cancel{\mathsf{clear}(\mathsf{D})}, \mathsf{ontable}(\mathsf{A}), \mathsf{clear}(\mathsf{A})\}$

   $\pi \leftarrow \langle\rangle$

   $o \leftarrow \mathsf{stack}(r,x,y)$, $\quad$ PRE : $\{\mathsf{holding}(r,\mathsf{D}), \mathsf{clear}(\mathsf{B})\}$

   $\sigma \leftarrow \{x \leftarrow \mathsf{D}, y \leftarrow \mathsf{B}\}$ $\quad$ EFF : $\{\mathsf{on}(\mathsf{D},\mathsf{B}), \mathsf{handempty}(r), \mathsf{clear}(\mathsf{D}), \neg\mathsf{holding}(r,\mathsf{D}), \neg\mathsf{clear}(\mathsf{B})\}$

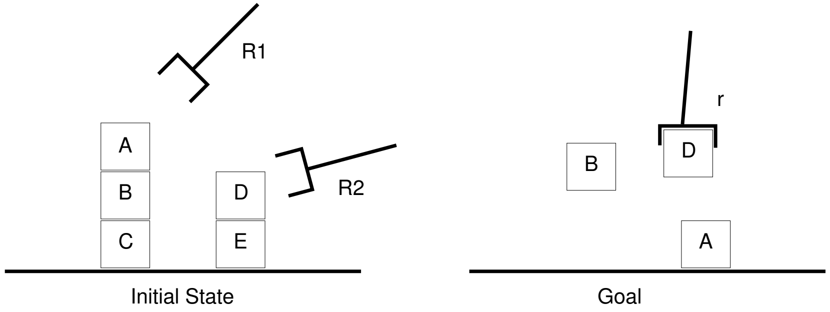
2. $g \leftarrow \{\underline{\mathsf{holding}(r,\mathsf{D})}, \underline{\mathsf{clear}(\mathsf{B})}, \mathsf{ontable}(\mathsf{A}), \mathsf{clear}(\mathsf{A})\}$

   $\pi \leftarrow \langle \mathsf{stack}(r,\mathsf{D},\mathsf{B})\rangle$

Trace example

relevance: $g \cap \text{EFF}^{+}(a) \neq \{\,\}$, $g \cap \text{EFF}^{-}(a) = \{\,\}$

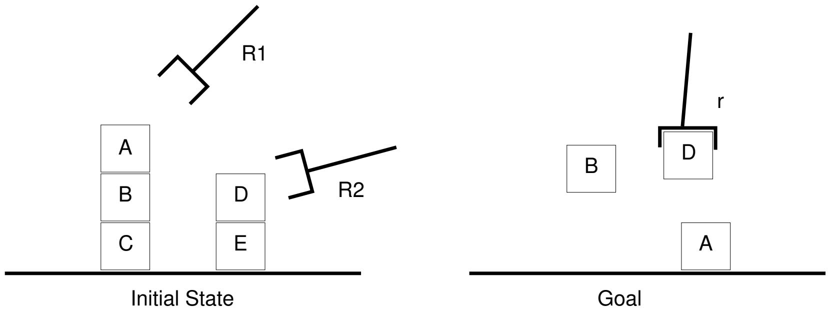
inverse transition: $\gamma^{-1}(g, a) = (g \setminus \text{EFF}^{+}(a)) \cup \text{PRE}(a)$



2. $g \leftarrow \{\mathsf{holding}(r, \mathsf{D}), \mathsf{clear}(\mathsf{B}), \mathsf{ontable}(\mathsf{A}), \mathsf{clear}(\mathsf{A})\}$
   $\pi \leftarrow \langle \mathsf{stack}(r, \mathsf{D}, \mathsf{B}) \rangle$

Trace example

relevance: $g \cap \text{EFF}^+(a) \neq \{\,\}$, $g \cap \text{EFF}^-(a) = \{\,\}$
inverse transition: $\gamma^{-1}(g, a) = (g \setminus \text{EFF}^+(a)) \cup \text{PRE}(a)$



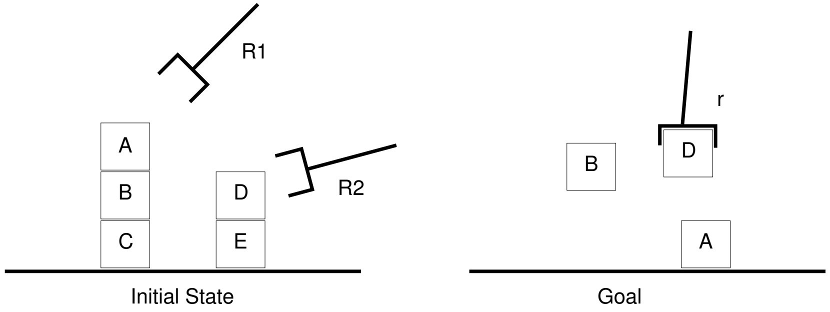
2. $g \leftarrow \{\mathsf{holding}(r, \mathsf{D}), \mathsf{clear}(\mathsf{B}), \underline{\mathsf{ontable}(\mathsf{A})}, \mathsf{clear}(\mathsf{A})\}$
   $\pi \leftarrow \langle \mathsf{stack}(r, \mathsf{D}, \mathsf{B}) \rangle$

$$o \leftarrow \mathsf{putdown}(r', x), \qquad \text{PRE}: \{\mathsf{holding}(r', \mathsf{A})\}$$
$$\sigma \leftarrow \{x \leftarrow \mathsf{A}\} \qquad \text{EFF}: \{\mathsf{ontable}(\mathsf{A}), \mathsf{handempty}(r'), \mathsf{clear}(\mathsf{A}), \neg\mathsf{holding}(r', \mathsf{A})\}$$

Trace example

relevance: $g \cap \mathrm{EFF}^{+}(a) \neq \{\,\}$, $g \cap \mathrm{EFF}^{-}(a) = \{\,\}$

inverse transition: $\gamma^{-1}(g, a) = (g \setminus \mathrm{EFF}^{+}(a)) \cup \mathrm{PRE}(a)$



2. $g \leftarrow \{\mathsf{holding}(r, \mathsf{D}), \mathsf{clear}(\mathsf{B}), \cancel{\mathsf{ontable}(\mathsf{A})}, \cancel{\mathsf{clear}(\mathsf{A})}\}$

   $\pi \leftarrow \langle \mathsf{stack}(r, \mathsf{D}, \mathsf{B}) \rangle$

   $o \leftarrow \mathsf{putdown}(r', x)$, $\quad$ PRE : $\{\mathsf{holding}(r', \mathsf{A})\}$

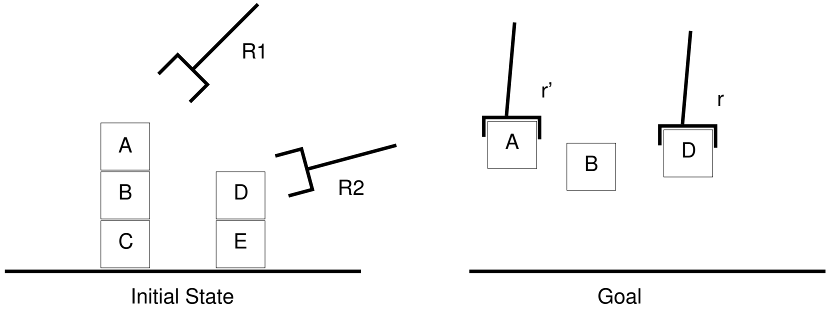
   $\sigma \leftarrow \{x \leftarrow \mathsf{A}\}$ $\quad$ EFF : $\{\mathsf{ontable}(\mathsf{A}), \mathsf{handempty}(r'), \mathsf{clear}(\mathsf{A}), \neg\mathsf{holding}(r', \mathsf{A})\}$

3. $g \leftarrow \{\mathsf{holding}(r, \mathsf{D}), \mathsf{clear}(\mathsf{B}), \underline{\mathsf{holding}(r', \mathsf{A})}\}$

   $\pi \leftarrow \langle \mathsf{putdown}(r', \mathsf{A}), \mathsf{stack}(r, \mathsf{D}, \mathsf{B}) \rangle$

Trace example

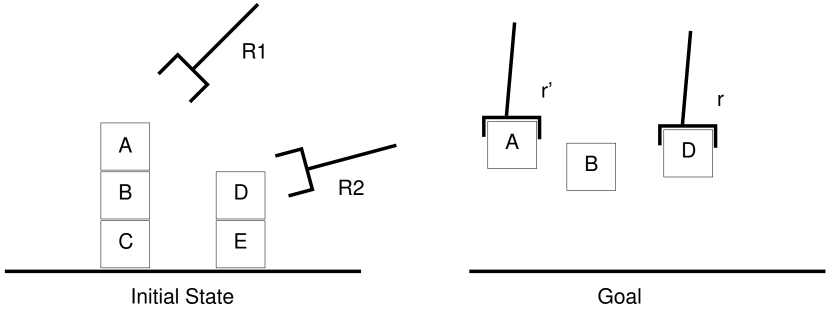
relevance: $g \cap \text{EFF}^+(a) \neq \{\,\}$, $g \cap \text{EFF}^-(a) = \{\,\}$
inverse transition: $\gamma^{-1}(g, a) = (g \setminus \text{EFF}^+(a)) \cup \text{PRE}(a)$



3. $g \leftarrow \{\text{holding}(r, \text{D}), \text{clear}(\text{B}), \text{holding}(r', \text{A})\}$
   $\pi \leftarrow \langle \text{putdown}(r', \text{A}), \text{stack}(r, \text{D}, \text{B})\rangle$

Trace example

relevance: $g \cap \text{EFF}^+(a) \neq \{\,\}$, $g \cap \text{EFF}^-(a) = \{\,\}$
inverse transition: $\gamma^{-1}(g, a) = (g \setminus \text{EFF}^+(a)) \cup \text{PRE}(a)$



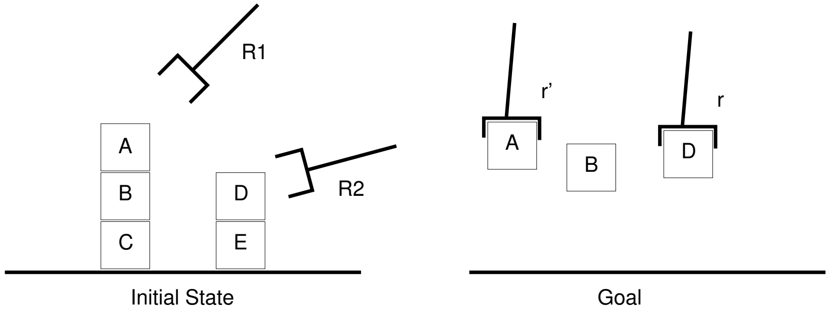
3. $g \leftarrow \{\underline{\mathsf{holding}(r, \mathsf{D})}, \mathsf{clear}(\mathsf{B}), \mathsf{holding}(r', \mathsf{A})\}$
   $\pi \leftarrow \langle \mathsf{putdown}(r', \mathsf{A}), \mathsf{stack}(r, \mathsf{D}, \mathsf{B})\rangle$

$o \leftarrow \mathsf{unstack}(r'', x, y)$,
$\sigma \leftarrow \{r'' \leftarrow r, x \leftarrow \mathsf{D}, y \neq \mathsf{B}\}$

PRE : $\{\mathsf{on}(\mathsf{D}, y), \mathsf{clear}(\mathsf{D}), \mathsf{handempty}(r)\}$
EFF : $\{\mathsf{holding}(r, \mathsf{D}), \mathsf{clear}(\mathsf{y}),$
$\neg\mathsf{handempty}(r), \neg\mathsf{on}(\mathsf{D}, y), \neg\mathsf{clear}(\mathsf{D})\}$

Trace example

relevance: $g \cap \text{EFF}^{+}(a) \neq \{\ \}$, $g \cap \text{EFF}^{-}(a) = \{\ \}$
inverse transition: $\gamma^{-1}(g, a) = (g \setminus \text{EFF}^{+}(a)) \cup \text{PRE}(a)$



3. $g \leftarrow \{\cancel{\mathsf{holding}(r, \mathsf{D})}, \mathsf{clear}(\mathsf{B}), \mathsf{holding}(r', \mathsf{A})\}$

   $\pi \leftarrow \langle \mathsf{putdown}(r', \mathsf{A}), \mathsf{stack}(r, \mathsf{D}, \mathsf{B})\rangle$

   $o \leftarrow \mathsf{unstack}(r'', x, y)$, PRE : $\{\mathsf{on}(\mathsf{D}, y), \mathsf{clear}(\mathsf{D}), \mathsf{handempty}(r)\}$

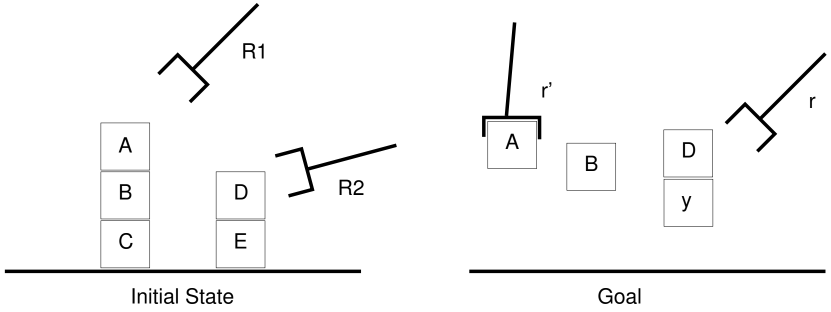
   $\sigma \leftarrow \{r'' \leftarrow r, x \leftarrow \mathsf{D}, y \neq \mathsf{B}\}$ EFF : $\{\mathsf{holding}(r, \mathsf{D}), \mathsf{clear}(\mathsf{y})$, $\neg\mathsf{handempty}(r), \neg\mathsf{on}(\mathsf{D}, y), \neg\mathsf{clear}(\mathsf{D})\}$

4. $g \leftarrow \{\underline{\mathsf{on}(\mathsf{D}, y)}, \underline{\mathsf{clear}(\mathsf{D})}, \underline{\mathsf{handempty}(r)}, \mathsf{clear}(\mathsf{B}), \mathsf{holding}(r', \mathsf{A}), y \neq \mathsf{B}\}$

   $\pi \leftarrow \langle \mathsf{unstack}(r, \mathsf{D}, y), \mathsf{putdown}(r', \mathsf{A}), \mathsf{stack}(r, \mathsf{D}, \mathsf{B})\rangle$ with $y \neq \mathsf{B}$

Trace example

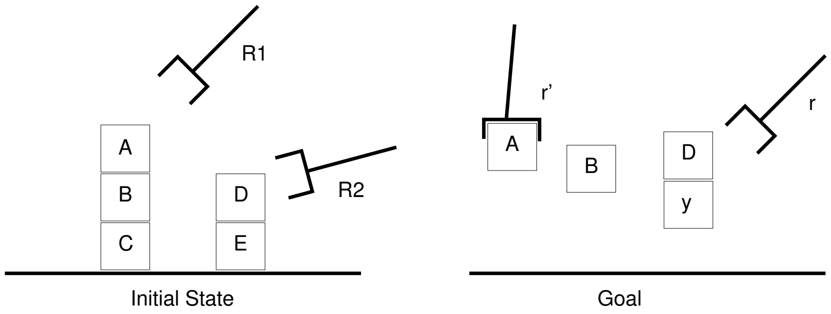
relevance: $g \cap \text{EFF}^+(a) \neq \{\,\}$, $g \cap \text{EFF}^-(a) = \{\,\}$
inverse transition: $\gamma^{-1}(g, a) = (g \setminus \text{EFF}^+(a)) \cup \text{PRE}(a)$



4. $g \leftarrow \{\mathsf{on}(\mathsf{D}, y), \mathsf{clear}(\mathsf{D}), \mathsf{handempty}(r), \mathsf{clear}(\mathsf{B}), \mathsf{holding}(r', \mathsf{A}), y \neq \mathsf{B}\}$
   $\pi \leftarrow \langle \mathsf{unstack}(r, \mathsf{D}, y), \mathsf{putdown}(r', \mathsf{A}), \mathsf{stack}(r, \mathsf{D}, \mathsf{B})\rangle$ with $y \neq \mathsf{B}$

Trace example

relevance: $g \cap \text{EFF}^{+}(a) \neq \{\,\}$, $g \cap \text{EFF}^{-}(a) = \{\,\}$
inverse transition: $\gamma^{-1}(g, a) = (g \setminus \text{EFF}^{+}(a)) \cup \text{PRE}(a)$



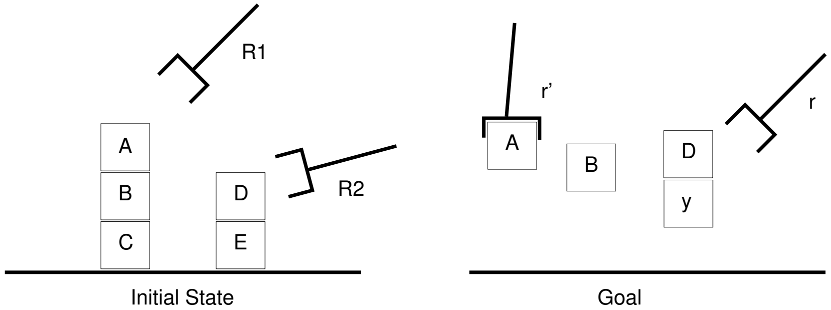
4. $g \leftarrow \{\mathsf{on}(\mathsf{D}, y), \mathsf{clear}(\mathsf{D}), \mathsf{handempty}(r), \mathsf{clear}(\mathsf{B}), \underline{\mathsf{holding}(r', \mathsf{A})}, y \neq \mathsf{B}\}$
$\pi \leftarrow \langle \mathsf{unstack}(r, \mathsf{D}, y), \mathsf{putdown}(r', \mathsf{A}), \mathsf{stack}(r, \mathsf{D}, \mathsf{B})\rangle$ with $y \neq \mathsf{B}$

$o \leftarrow \mathsf{unstack}(r'', x, y')$,
$\sigma \leftarrow \{r'' \leftarrow r', x \leftarrow \mathsf{A}, y' \leftarrow \mathsf{B}, r' \neq r\}$

PRE : $\{\mathsf{on}(\mathsf{A}, \mathsf{B}), \mathsf{clear}(\mathsf{A}), \mathsf{handempty}(r')\}$
EFF : $\{\mathsf{holding}(r', \mathsf{A}), \mathsf{clear}(\mathsf{B}),$
$\neg\mathsf{handempty}(r'), \neg\mathsf{on}(\mathsf{A}, \mathsf{B}), \neg\mathsf{clear}(\mathsf{A})\}$

Trace example

relevance: $g \cap \text{EFF}^{+}(a) \neq \{\,\}$, $g \cap \text{EFF}^{-}(a) = \{\,\}$
inverse transition: $\gamma^{-1}(g, a) = (g \setminus \text{EFF}^{+}(a)) \cup \text{PRE}(a)$



4. $g \leftarrow \{\mathsf{on}(\mathsf{D}, y), \mathsf{clear}(\mathsf{D}), \mathsf{handempty}(r), \cancel{\mathsf{clear}(\mathsf{B})}, \cancel{\mathsf{holding}(r', \mathsf{A})}, y \neq \mathsf{B}\}$

   $\pi \leftarrow \langle \mathsf{unstack}(r, \mathsf{D}, y), \mathsf{putdown}(r', \mathsf{A}), \mathsf{stack}(r, \mathsf{D}, \mathsf{B})\rangle$ with $y \neq \mathsf{B}$

   $o \leftarrow \mathsf{unstack}(r'', x, y')$,
   $\sigma \leftarrow \{r'' \leftarrow r', x \leftarrow \mathsf{A}, y' \leftarrow \mathsf{B}, r' \neq r\}$

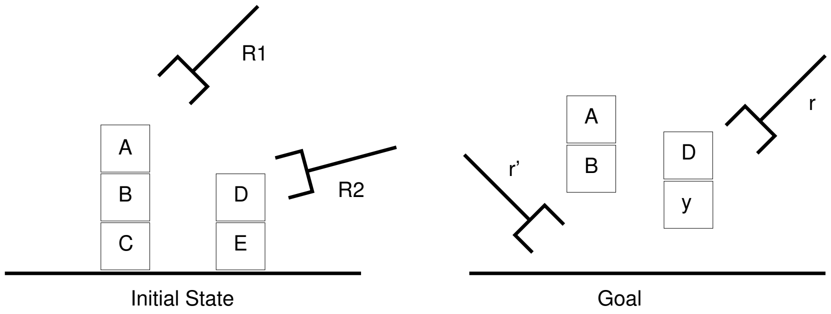
   $\text{PRE}: \{\mathsf{on}(\mathsf{A}, \mathsf{B}), \mathsf{clear}(\mathsf{A}), \mathsf{handempty}(r')\}$
   $\text{EFF}: \{\mathsf{holding}(r', \mathsf{A}), \mathsf{clear}(\mathsf{B}),$
   $\neg\mathsf{handempty}(r'), \neg\mathsf{on}(\mathsf{A}, \mathsf{B}), \neg\mathsf{clear}(\mathsf{A})\}$

5. $g \leftarrow \{\mathsf{on}(\mathsf{D}, y), \mathsf{clear}(\mathsf{D}), \mathsf{handempty}(r), \underline{\mathsf{on}(\mathsf{A}, \mathsf{B})}, \underline{\mathsf{clear}(\mathsf{A})}, \underline{\mathsf{handempty}(r')}, y \neq \mathsf{B}, r \neq r'\}$

   $\pi \leftarrow \langle \mathsf{unstack}(r', \mathsf{A}, \mathsf{B}), \mathsf{unstack}(r, \mathsf{D}, y), \mathsf{putdown}(r', \mathsf{A}), \mathsf{stack}(r, \mathsf{D}, \mathsf{B})\rangle$ with $y \neq \mathsf{B}, r \neq r'$

Trace example

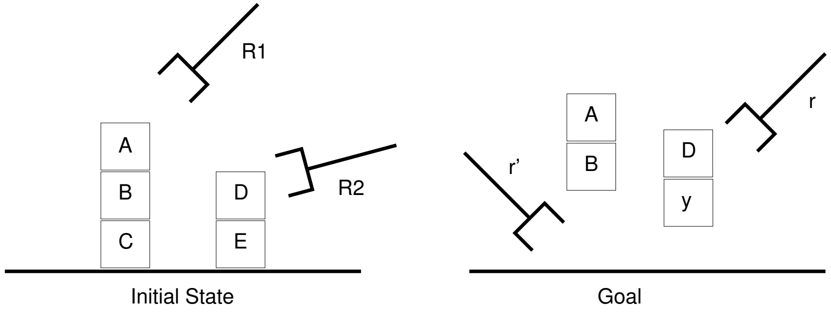
relevance: $g \cap \text{EFF}^+(a) \neq \{\,\}$, $g \cap \text{EFF}^-(a) = \{\,\}$
inverse transition: $\gamma^{-1}(g, a) = (g \setminus \text{EFF}^+(a)) \cup \text{PRE}(a)$



5. $g \leftarrow \{\text{on}(\mathsf{D}, y), \text{clear}(\mathsf{D}), \text{handempty}(r), \text{on}(\mathsf{A}, \mathsf{B}), \text{clear}(\mathsf{A}), \text{handempty}(r'), y \neq \mathsf{B}, r \neq r'\}$
   $\pi \leftarrow \langle \text{unstack}(r', \mathsf{A}, \mathsf{B}), \text{unstack}(r, \mathsf{D}, y), \text{putdown}(r', \mathsf{A}), \text{stack}(r, \mathsf{D}, \mathsf{B})\rangle$ with $y \neq \mathsf{B}, r \neq r'$

Trace example

relevance: $g \cap \text{EFF}^+(a) \neq \{\,\}$, $g \cap \text{EFF}^-(a) = \{\,\}$
inverse transition: $\gamma^{-1}(g, a) = (g \setminus \text{EFF}^+(a)) \cup \text{PRE}(a)$



5. $g \leftarrow \{\mathsf{on}(\mathsf{D}, y), \mathsf{clear}(\mathsf{D}), \mathsf{handempty}(r), \mathsf{on}(\mathsf{A}, \mathsf{B}), \mathsf{clear}(\mathsf{A}), \mathsf{handempty}(r'), y \neq \mathsf{B}, r \neq r'\}$
   $\pi \leftarrow \langle \mathsf{unstack}(r', \mathsf{A}, \mathsf{B}), \mathsf{unstack}(r, \mathsf{D}, y), \mathsf{putdown}(r', \mathsf{A}), \mathsf{stack}(r, \mathsf{D}, \mathsf{E})\rangle$ with $y \neq \mathsf{B}, r \neq r'$

   initial state: $s = \{\mathsf{on}(\mathsf{D}, \mathsf{E}), \mathsf{clear}(\mathsf{D}), \mathsf{handempty}(\mathsf{R1}), \mathsf{on}(\mathsf{A}, \mathsf{B}), \mathsf{clear}(\mathsf{A}), \mathsf{handempty}(\mathsf{R2}), \ldots\}$
   s satisfies $g : \sigma \leftarrow \{r \leftarrow \mathsf{R1}, r' \leftarrow \mathsf{R2}, y \leftarrow \mathsf{E}\}$

   result plan: $\pi \leftarrow \langle \mathsf{unstack}(\mathsf{R2}, \mathsf{A}, \mathsf{B}), \mathsf{unstack}(\mathsf{R1}, \mathsf{D}, \mathsf{E}), \mathsf{putdown}(\mathsf{R2}, \mathsf{A}), \mathsf{stack}(\mathsf{R1}, \mathsf{D}, \mathsf{B})\rangle$

Summary

State-space planning produces totally-ordered plans by a forward or backward search in the state space. This requires domain-independent heuristics or domain-specific control rules to be efficient

The Delete-relaxation of a problem ignores delete lists. Once a fact becomes true in a delete-relaxed problem, it remains true. The optimal delete-relaxed plan cost h^+ is an admissible heuristic but is NP-hard to compute.

The critical path heuristic h^{max} is admissible and computable in polynomial time but additionally ignores anything but the hardest subgoal of each goal set. h^{sum} is an inadmissible variant which sums the costs of the subgoals.

The inadmissible h^{FF} heuristic returns the number of actions in a relaxed plan, extracted from the relaxed reachability graph. Abstractions and landmarks lead to other powerful heuristics

Backward search requires the ability to regress goals and leads to a lifted algorithm which does not need to fully instanciate operators