Advanced Results on Distance Estimation in Planning

2. Delete Relaxation Heuristics

How to Understand Them, and How To Improve Them

Jörg Hoffmann and Michael Katz

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Summer 2013

Agenda

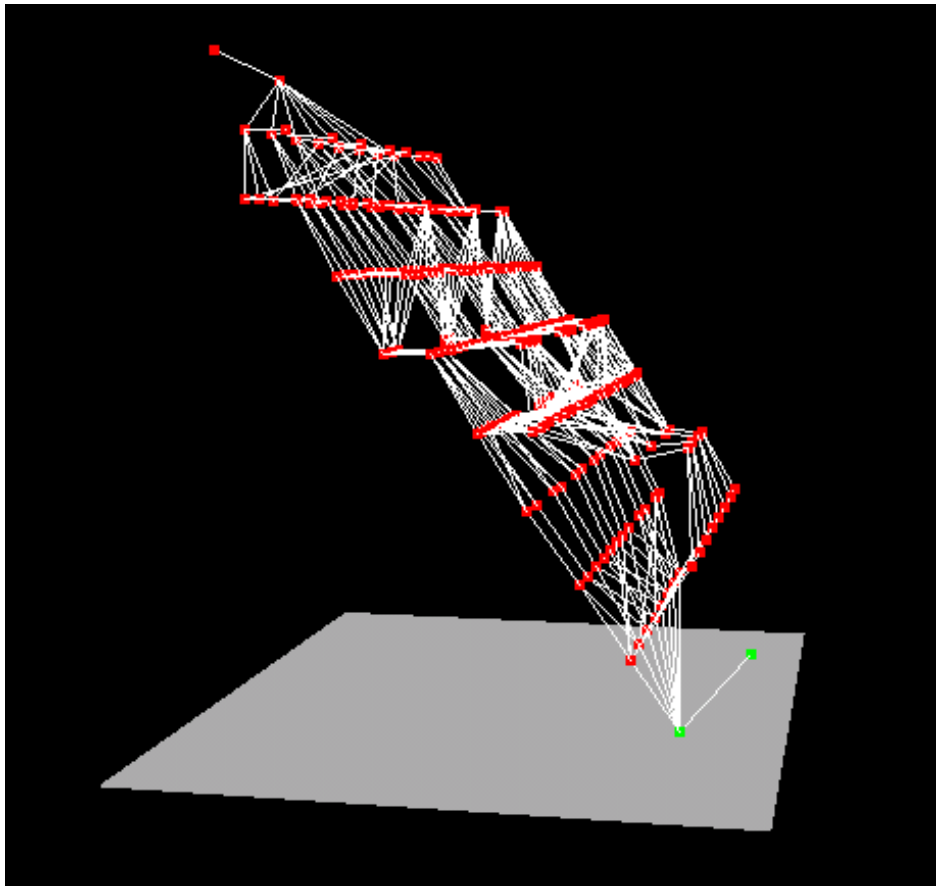
Lecture Overview

Disclaimer: I assume you're familiar with h^+ and its approximation using relaxed plan heuristics as in, e.g., FF. In case you aren't, sorry but this is probably not a suitable lecture for you.

So what will I talk about?

- Analyzing the search space surface under h^+ to explain and predict the performance of heuristic search planners [Hoffmann (2011b,a)].
- Improving h^+ (and its approximations) by marriage with the critical-path heuristics h^m [Keyder *et al.* (2012)].
- Improving h^+ (and its approximations) by relaxing only a subset of the state variables [Katz *et al.* (2013b,a)].

h^+ in Gripper



$\rightarrow$ Observations, anyone? The goal states lie at the bottom of an inclined plane.

$\rightarrow$ Intuition: Heuristic search = "dropping a marble onto this surface".

Exits, Local Minima, and Benches

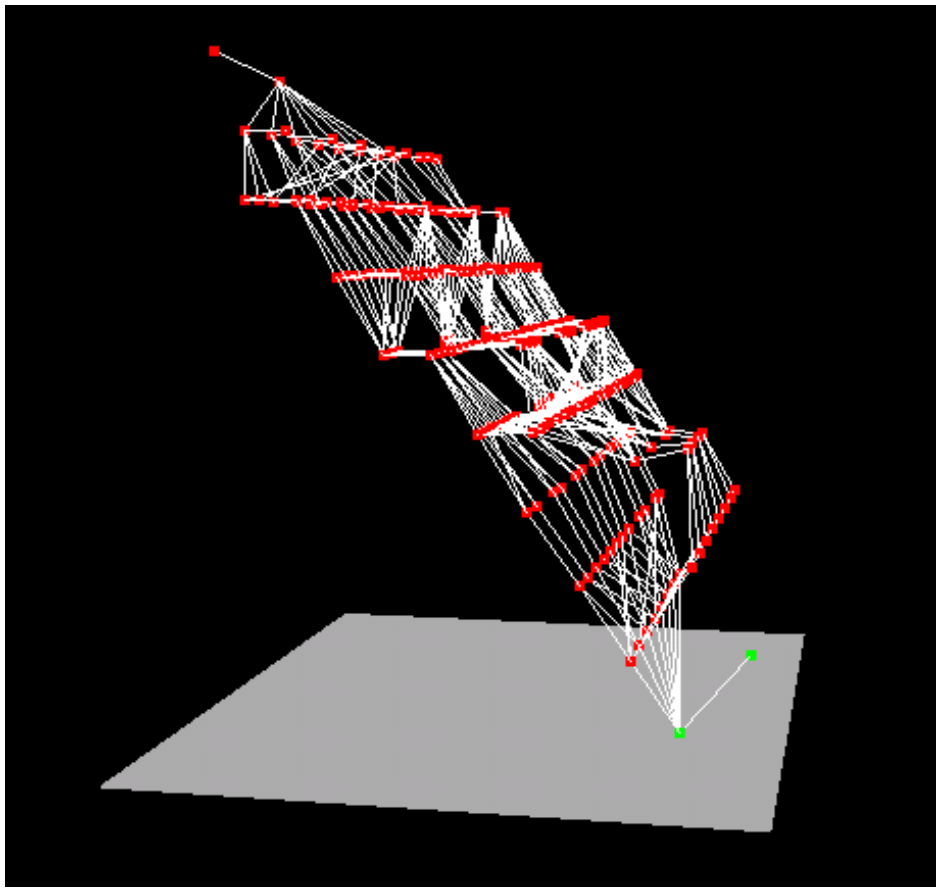
Definition (Exit, Exit Distance). *Let $\Pi = (V, I, G, A)$ be a solvable FDR planning task with state space Θ_Π, let h be a heuristic for Π, and let s be a state with $0 < h(s) < \infty$. An exit for s is a state s' reachable from s so that $h(s') = h(s)$ and Θ_Π has a transition $s' \to s''$ where $h(s'') < h(s)$. The exit distance $exd(s)$ of s under h is the length of a shortest path to an exit, or $exd(s) = \infty$ if no exit exists.*

$\to$ An exit is a state reaching which enables us to improve the heuristic value, relative to our current state s.

Definition (Local Minima, Benches). *Let $\Pi = (V, I, G, A)$ be a solvable FDR planning task with state space Θ_Π, and let h be a heuristic for Π. A path in Θ_Π is monotone if it contains no transition $s_1 \to s_2$ so that $h(s_1) < h(s_2)$. A state s with $0 < h(s) < \infty$ is a local minimum under h if there exists no monotone path to an exit; else, s is a bench under h.*

$\to$ From a local minimum, we cannot improve the heuristic value without temporarily making it worse; from a bench, we can.

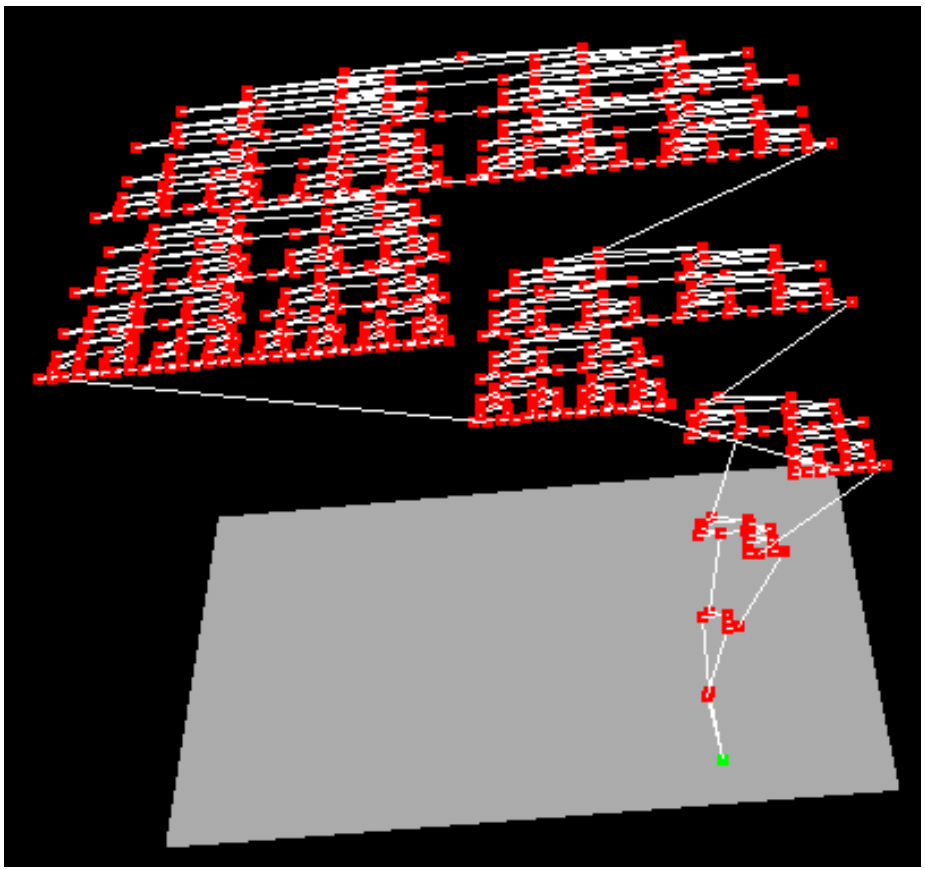
Local Minima and Benches in Gripper Benchmark Domain



→ Any local minima here? No.

→ What is the largest exit distance? 1 (e.g., the flat row in the middle).

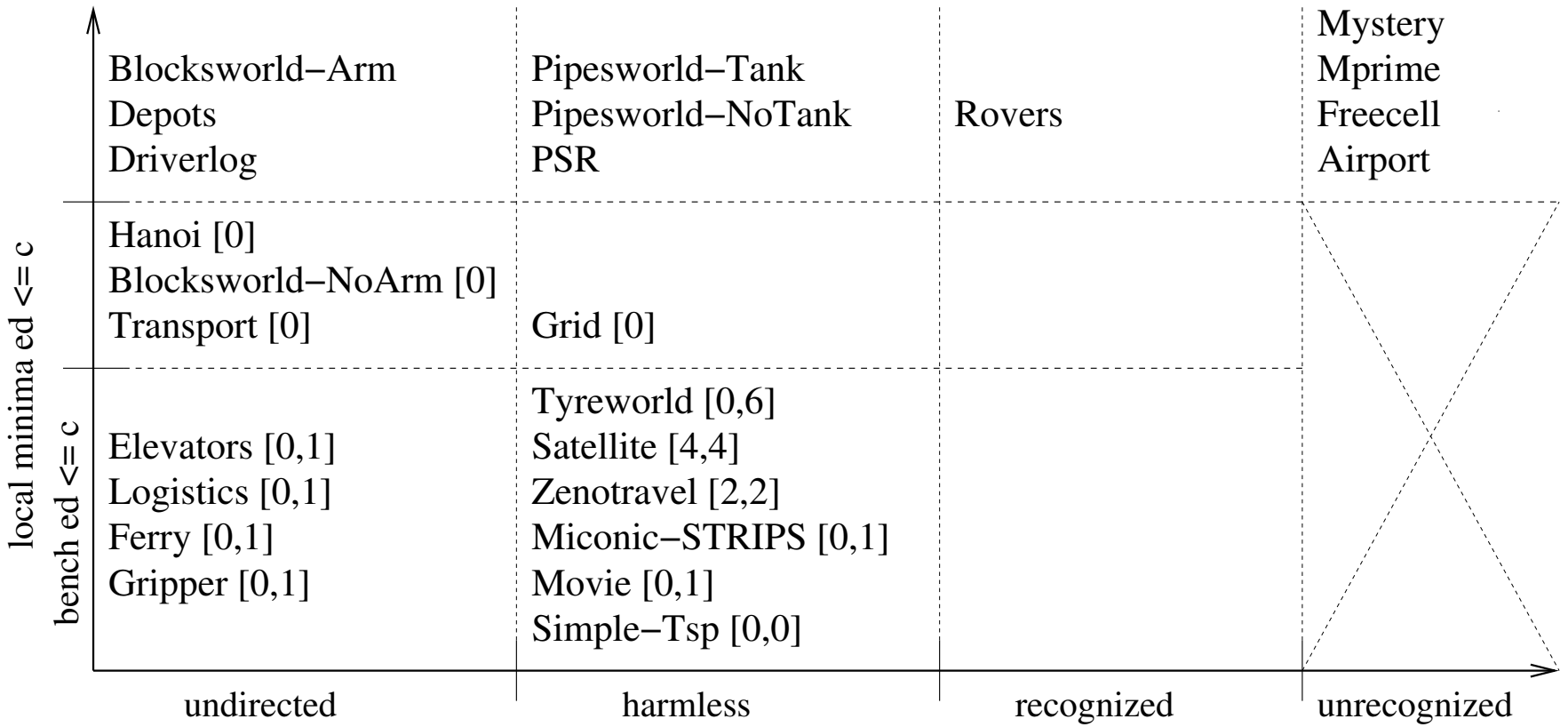
Local Minima and Benches in Towers of Hanoi



→ Any local minima here? No.

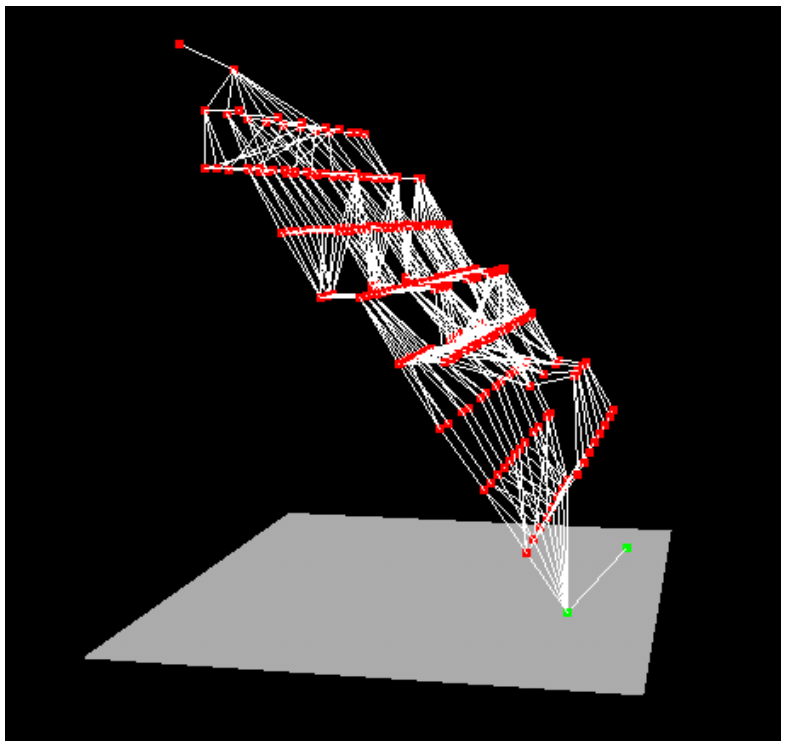
→ What is the largest exit distance? $2^5 = 32$ from initial state to exit off topmost plateau: Solve 5-discs to be able to move the bottom disc into place.

Proved Surface Properties under h^+ [Hoffmann (2002, 2005)]



Legend: x-axis: 4 classes regarding dead ends, each domain in "highest" class of any of its instances. y-axis: Exists constant bound on exit distance from bench states and/or local minimum states in the domain? [lm,bench] where both exist, [lm] where only the latter exists; lm=0 means no local minima at all. Bottom right crossed out since unrecognized dead ends imply infinite exit distance.

Surface Properties under h^+ in Gripper Benchmark



- **Empirical result** (see picture): In this particular instance of the Gripper benchmark, under h^+ there aren't any local minima and the maximal exit distance any state has is 1.
- **Theoretical result** (see previous slide): The same is true for *every* instance of the Gripper benchmark.

How to prove this??

. . . sit down, consider each of the 25 domains in turn, have fun.

Identify patterns: (that apply across domains)

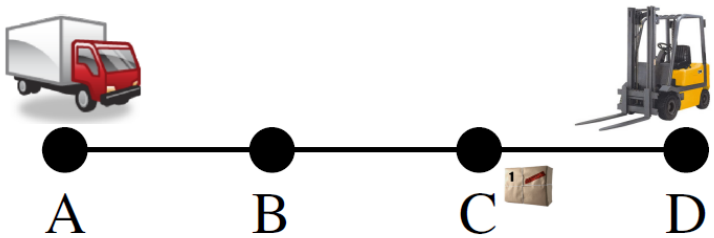
(a) Absence of dead ends: In many domains, all actions are invertible and thus the state spaces are undirected. (In others, it's a little, but not much, more complicated than that.)

(b) Absence of local minima: Given a state s, show how to construct a monotone path $\vec{a}_s$ to an exit.
$\rightarrow$ Typically, this works by considering an optimal relaxed plan $\vec{a}_s^+$ for s, and constructing $\vec{a}_s$ by using only the actions in $\vec{a}_s^+$. This path is monotone because, for any state s_i it traverses, *a relaxed plan $\vec{a}_{s_i}^+$ for s_i can be obtained from $\vec{a}_s^+$ by replacing some actions with their inverses.*

(c) Exit distance: How long does $\vec{a}_s$ need to be until we can obtain $\vec{a}_{s_i}^+$ by *removing* an action from $\vec{a}_s^+$, without replacement?

$\rightarrow$ **We now illustrate (b) with an example.**

Example Local Minimum Proof: Logistics with Forklift



- Initial state: $t = A, f = D, p = C$.
- Goal: $t = A, f = D, p = D$.
- Actions: $drt(X, Y), drf(X, Y), lo(X), ul(X)$.

$\rightarrow$ We identify a monotone exit path $\vec{a}_s$ for the inital state $s := I$, and thereby prove that this state is not a local minimum.

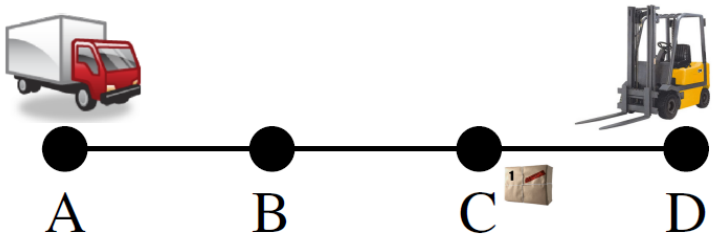
$\rightarrow$ Wanna guess what the path $\vec{a}_s$ will be?

$$I = s \xrightarrow{drt(A,B)} s_1 \xrightarrow{drt(B,C)} s_2 \xrightarrow{drf(D,C)} s_3 \xrightarrow{lo(C)} s'$$

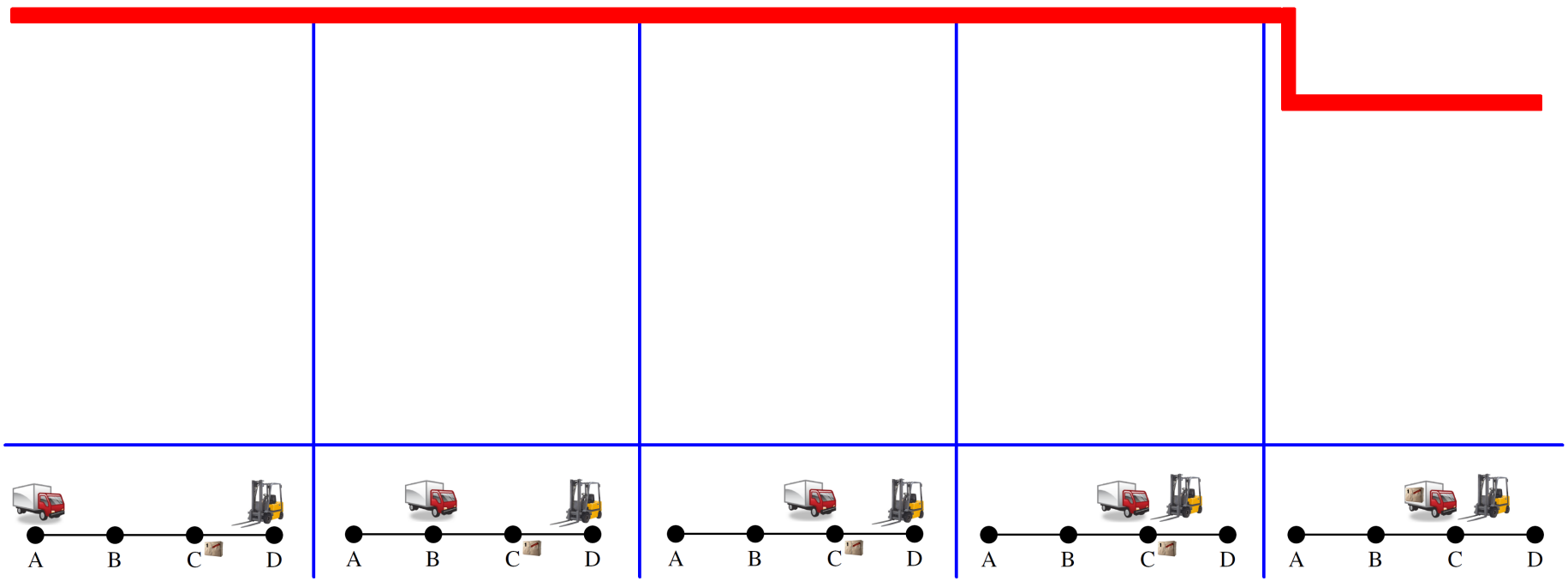
s_3 will be the exit state: $h^+(s') < h^+(s_3) = h^+(s)$.

$\rightarrow$ **Argument summary:** There is a variable v_0 (here: p) that moves egoistically; let the first action affecting v_0 in the optimal relaxed plan $\vec{a}_s^+$ be a_0 (here: $lo(C)$). We can construct a path $\vec{a}_s$ using only actions from $\vec{a}_s^+$, bringing all supporting variables (here: t and f) into the values required by the precondition of a_0. h^+ is monotone on this path because of invertibility, and decreases strictly after executing a_0.

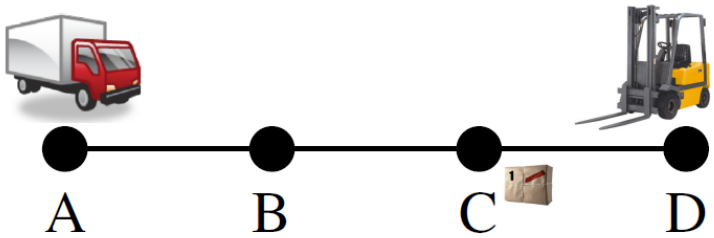
Example Local Minimum Proof: Logistics with Forklift



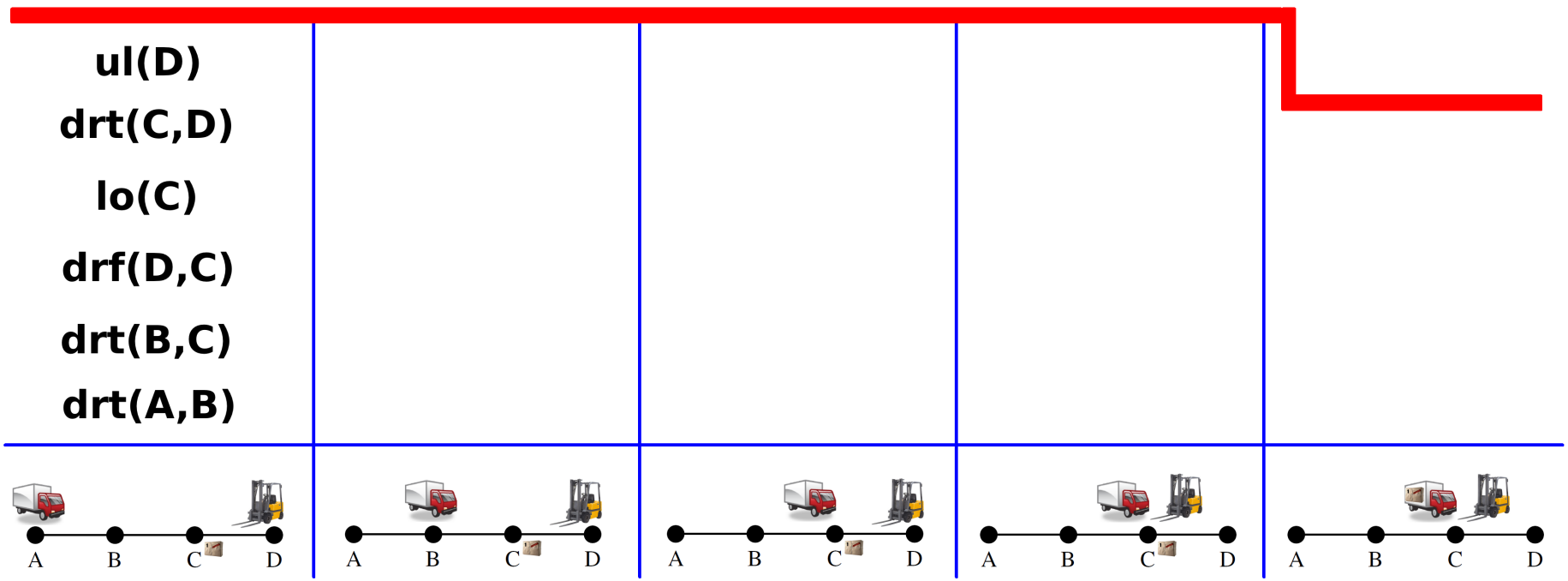
- Initial state: $t = A, f = D, p = C$.
- Goal: $t = A, f = D, p = D$.
- Actions: $drt(X, Y), drf(X, Y), lo(X), ul(X)$.



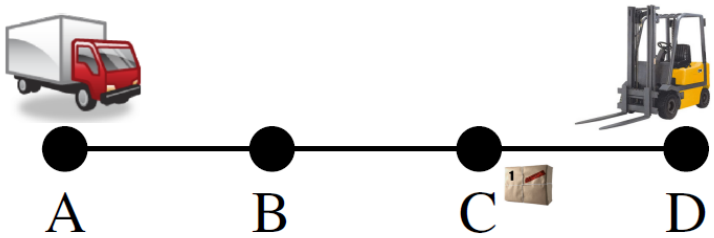
Example Local Minimum Proof: Logistics with Forklift



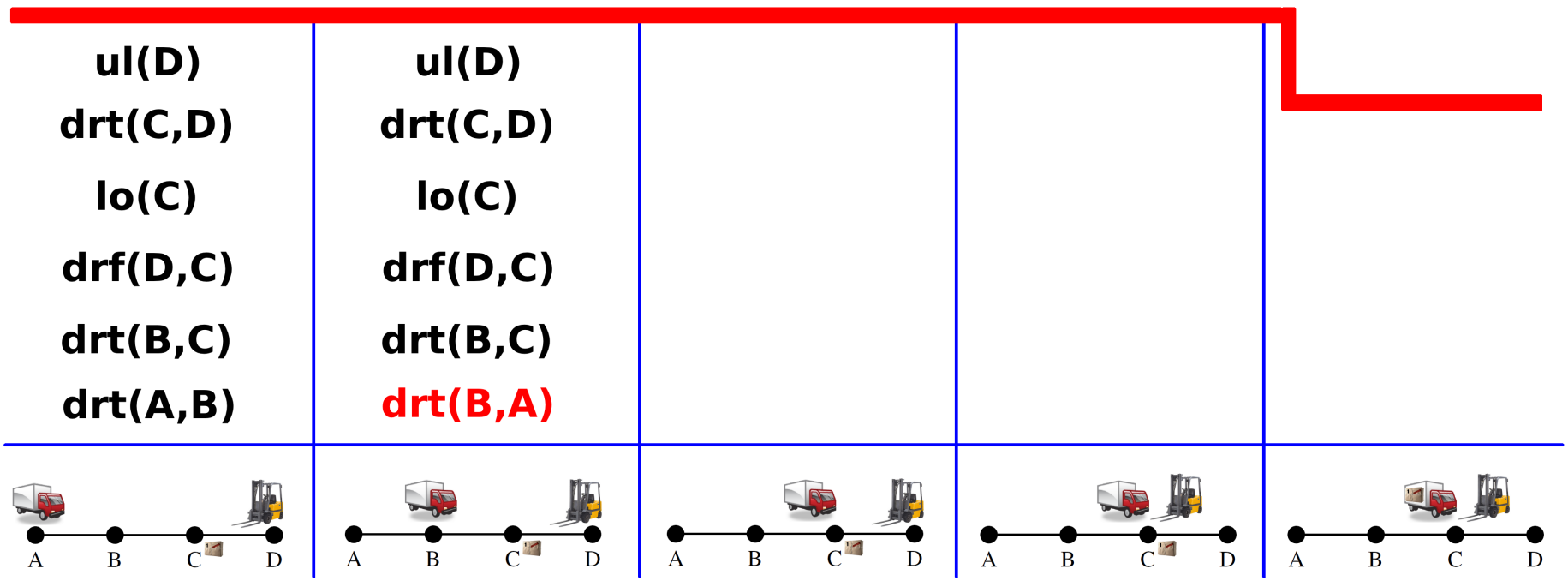
- Initial state: $t = A, f = D, p = C$.
- Goal: $t = A, f = D, p = D$.
- Actions: $drt(X, Y), drf(X, Y), lo(X), ul(X)$.



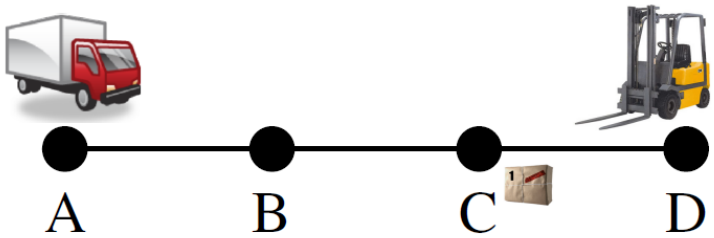
Example Local Minimum Proof: Logistics with Forklift



- Initial state: $t = A, f = D, p = C$.
- Goal: $t = A, f = D, p = D$.
- Actions: $drt(X, Y), drf(X, Y), lo(X), ul(X)$.

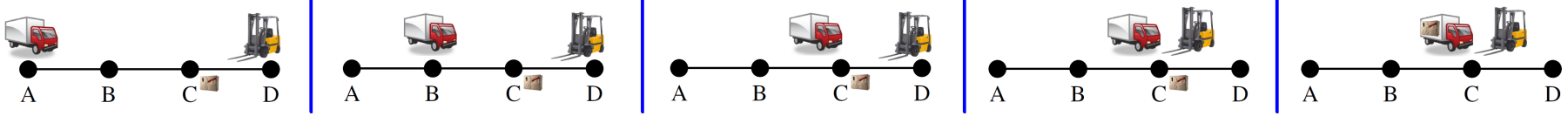


Example Local Minimum Proof: Logistics with Forklift

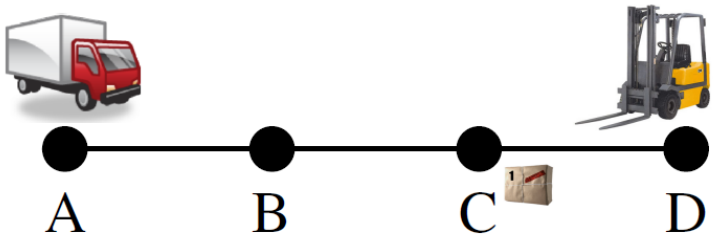


- Initial state: $t = A, f = D, p = C$.
- Goal: $t = A, f = D, p = D$.
- Actions: $drt(X, Y), drf(X, Y), lo(X), ul(X)$.

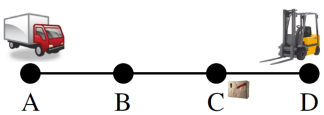
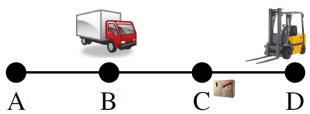
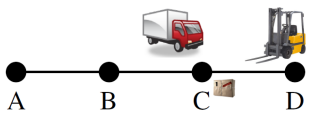
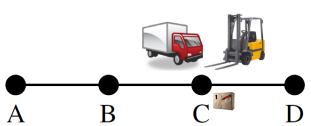
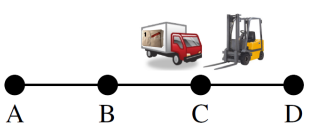
ul(D)	ul(D)	ul(D)		
drt(C,D)	drt(C,D)	drt(C,D)		
lo(C)	lo(C)	lo(C)		
drf(D,C)	drf(D,C)	drf(D,C)		
drt(B,C)	drt(B,C)	drt(C,B)		
drt(A,B)	drt(B,A)	drt(B,A)		



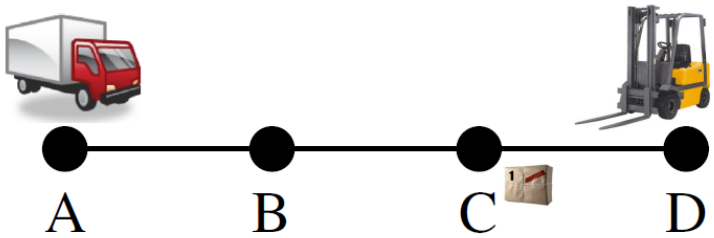
Example Local Minimum Proof: Logistics with Forklift



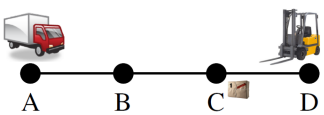
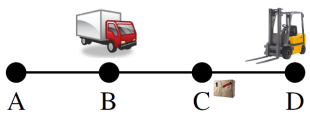
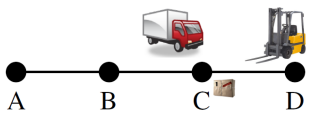
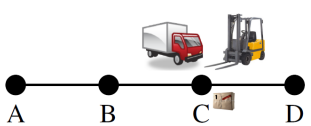
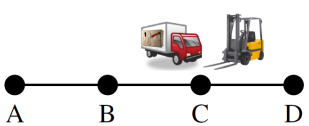
- Initial state: $t = A, f = D, p = C$.
- Goal: $t = A, f = D, p = D$.
- Actions: $drt(X, Y), drf(X, Y), lo(X), ul(X)$.

ul(D)	**ul(D)**	**ul(D)**	**ul(D)**	
drt(C,D)	**drt(C,D)**	**drt(C,D)**	**drt(C,D)**	
lo(C)	**lo(C)**	**lo(C)**	**lo(C)**	
drf(D,C)	**drf(D,C)**	**drf(D,C)**	**drf(C,D)**	
drt(B,C)	**drt(B,C)**	**drt(C,B)**	**drt(C,B)**	
drt(A,B)	**drt(B,A)**	**drt(B,A)**	**drt(B,A)**	
				

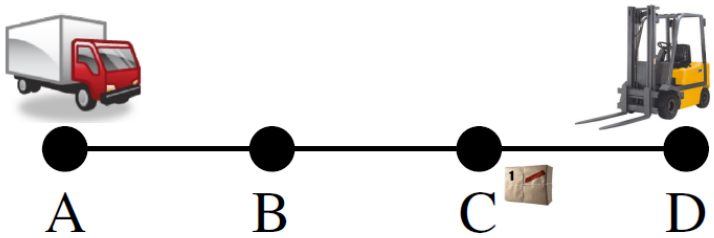
Example Local Minimum Proof: Logistics with Forklift



- Initial state: $t = A, f = D, p = C$.
- Goal: $t = A, f = D, p = D$.
- Actions: $drt(X, Y), drf(X, Y), lo(X), ul(X)$.

ul(D)	ul(D)	ul(D)	ul(D)	
drt(C,D)	drt(C,D)	drt(C,D)	drt(C,D)	ul(D)
lo(C)	lo(C)	lo(C)	lo(C)	drt(C,D)
drf(D,C)	drf(D,C)	drf(D,C)	drf(C,D)	drf(C,D)
drt(B,C)	drt(B,C)	drt(C,B)	drt(C,B)	drt(C,B)
drt(A,B)	drt(B,A)	drt(B,A)	drt(B,A)	drt(B,A)
				

Proof by Example?



- Initial state: $t = A, f = D, p = C$.
- Goal: $t = A, f = D, p = D$.
- Actions: $drt(X, Y), drf(X, Y), lo(X), ul(X)$.

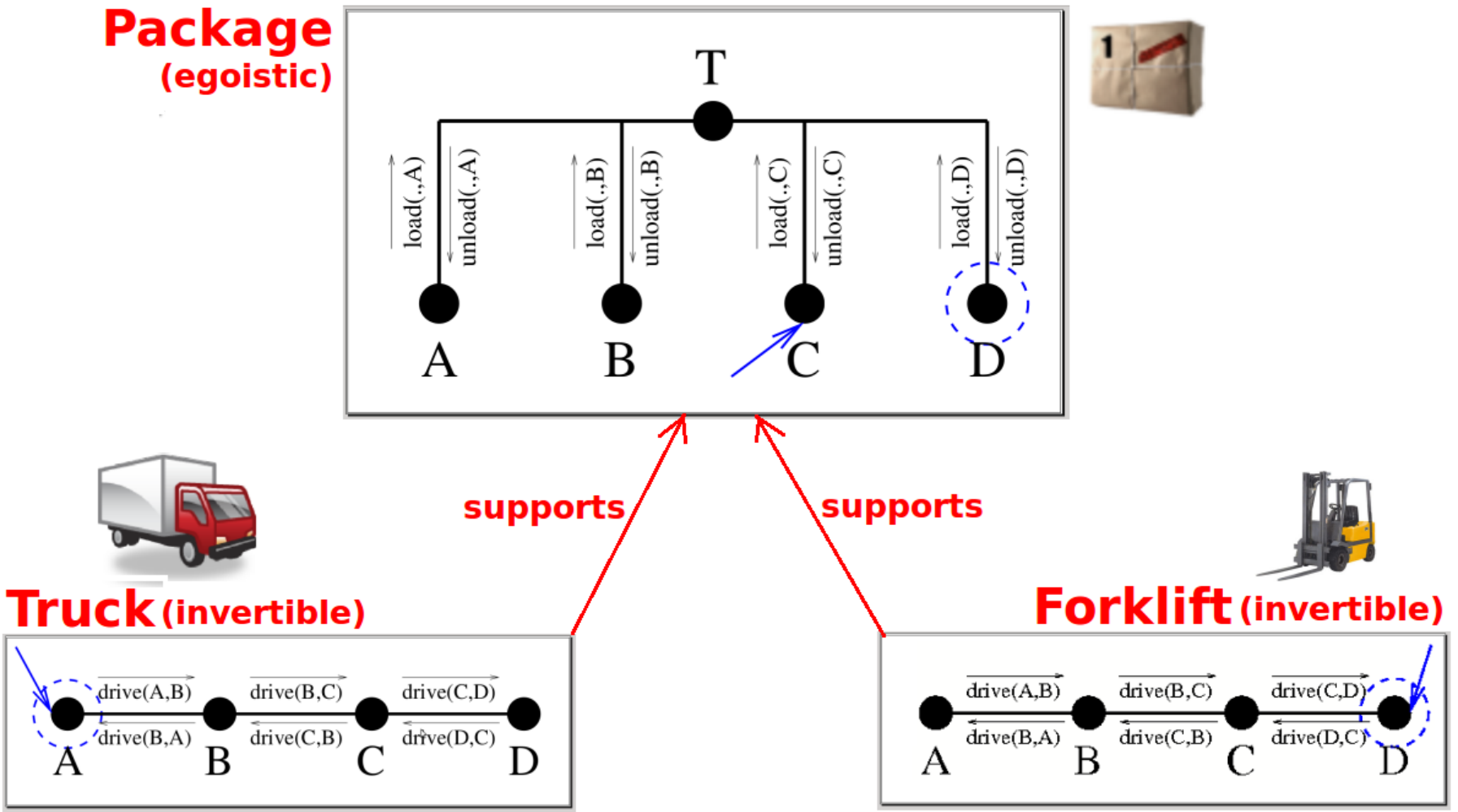
$\rightarrow$ This works for *all* states s:

$\rightarrow$ Argument summary: There is a variable v_0 that moves egoistically; let the first action affecting v_0 in the optimal relaxed plan $\vec{a}_s^+$ be a_0. We can construct a path $\vec{a}_s$ using only actions from $\vec{a}_s^+$, bringing all supporting variables into the values required by the precondition of a_0. h^+ is monotone on this path because of invertibility, and decreases strictly after executing a_0.

- Package at $x \neq D$: $v_0 = p$, $a_0 = lo(x)$, support: truck and forklift.
- Package in truck: $v_0 = p$, $a_0 = ul(D)$, support: truck and forklift.
- Package at D: $v_0 = t$ or f, $a_0 = drt(X, Y)$ or $drf(X, Y)$, support: none.

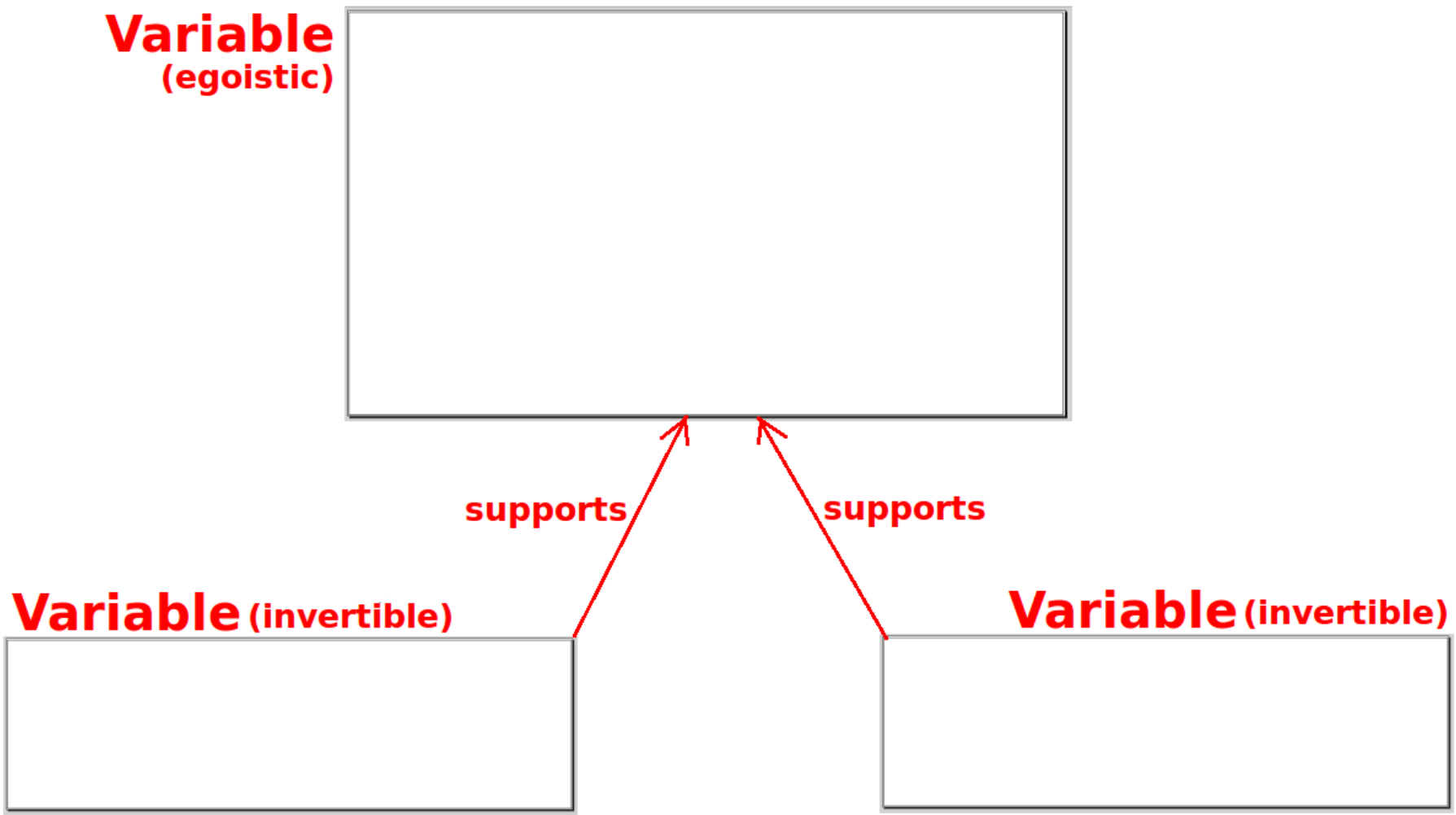
$\rightarrow$ Similar proof arguments work in many domains. Indeed, we can abstract from the domain and apply the same proof considering just the causal graph!

Here's what we looked at so far ...



... and moving the package/truck/forklift does not affect anything else.

. . . and here's what we get by removing irrelevant detail



. . . and every action affects only one variable.

Causal Graphs, and Invertibility

Definition (Causal Graph). *Let $\Pi = (V, I, G, A)$ be an FDR planning task. The causal graph is a digraph with vertices V, and has an arc (x, y) iff $x \neq y$ and there exists an action $a \in A$ such that either (i) x appears in $pre(a)$ and y appears in $eff(a)$, or (ii) x and y both appear in $eff(a)$.*

$\rightarrow$ The "support" arrows on the previous slide are type (i) causal graph arcs.

Definition (Invertibility). *Let $\Pi = (V, I, G, A)$ be an FDR planning task and let $x \in V$.*

- *The domain transition graph (DTG) of x is a labeled digraph with vertices D_x, and an arc (d, d') induced by action a iff $eff(a)[x] = d'$, and either $pre(a)[x] = d$ or x is not mentioned in $pre(a)$. The arc is labeled with its outside condition $pre(a)[V \setminus \{x\}]$.*
- *An arc (d, d') is invertible if there exists an arc (d', d) with outside condition $\phi' \subseteq \phi$ where ϕ is the outside condition of (d, d'). Variable x is invertible if all arcs in its DTG are invertible.*

$\rightarrow$ In our Logistics example, all variables are invertible.

The Connection Between Causal Graphs and h^+

Theorem. *Let $\Pi = (V, I, G, A)$ be a solvable FDR planning task such that $CG(\Pi)$ is acyclic and, for all $x \in V$ that are not leaves in $CG(\Pi)$, x is invertible. Then Π does not contain any local minima under h^+.*

Proof Comments:

- The leaf variables in the causal graph (no outgoing arcs) are egoistic because no other variable depends on them. Invertibility is required only for the support variables.
- "Every action affects only one variable" (cf. slide 17) is ensured because otherwise the (ii) arcs in the causal graph would yield a cycle.
- The (i) arcs in the causal graph are the "supports" arrows on slide 17.

  $\rightarrow$ Rather than just two independent support variables, we can have arbitrary acyclic support. Basically, we might need a driver to first get into the truck so the truck can move, etc. This does not break our construction. The root variables in the causal graph (no incoming arcs) will move freely as do the truck and forklift in our example.

The TorchLight Tool

Proof Structure: (generalizes theorem on previous slide)

(A) Given optimal relaxed plan for s, sufficient criterion for "s is no local minimum".

(B) Sufficient criterion for "(A) will apply to all s".

Global Analysis:

- Test criterion (B).
- Only sufficient, not necessary.
- And what about domains *with* local minima?

Approximate Local Analysis:

- Randomly sample states s.
- Generate a (not necessarily optimal) relaxed plan; test criterion (A).
- Success rate = percentage of s where criterion applies.

Hoffmann vs. TorchLight

Zenotravel	Hanoi [0]
Satellite	Airport [0]
Rovers	Blocksworld–Arm [30]
PSR	Mystery [39]
Pipesworld–Tank	Pipesworld–Tank [40]
Pipesworld–NoTank	Mprime [49]
Mystery	PSR [50]
Mprime	Freecell [56]
Freecell	Blocksworld–NoArm [57]
Driverlog	Pipesworld–NoTank [76]
Depots	Grid [80]
Blocksworld–Arm	Depots [81]
Airport	Zenotravel [95]
Tyreworld	Tyreworld [100]
Transport	Transport [100]
Simple–Tsp	Simple–Tsp [100]
Movie	Satellite [100]
Miconic–STRIPS	Rovers [100]
Logistics	Movie [100]
Hanoi	Miconic–STRIPS [100]
Gripper	Logistics [100]
Grid	Gripper [100]
Ferry	Ferry [100]
Elevators	Elevators [100]
Blocksworld–NoArm	Driverlog [100]

- Success rate: average per-domain from single sample state per-instance.

Hoffmann vs. TorchLight

Zenotravel	Hanoi [0]
Satellite	Airport [0]
Rovers	Blocksworld–Arm [30]
PSR	Mystery [39]
Pipesworld–Tank	Pipesworld–Tank [40]
Pipesworld–NoTank	Mprime [49]
Mystery	PSR [50]
Mprime	Freecell [56]
Freecell	Blocksworld–NoArm [57]
Driverlog	Pipesworld–NoTank [76]
Depots	Grid [80]
Blocksworld–Arm	Depots [81]
Airport	Zenotravel [95]
Tyreworld	Tyreworld [100]
Transport	Transport [100]
Simple–Tsp	Simple–Tsp [100]
Movie	Satellite [100]
Miconic–STRIPS	Rovers [100]
Logistics	Movie [100]
Hanoi	Miconic–STRIPS [100]
Gripper	Logistics [100]
Grid	Gripper [100]
Ferry	Ferry [100]
Elevators	Elevators [100]
Blocksworld–NoArm	Driverlog [100]

- Not all domains are "fully recognized" . . .
- . . . mostly because Hoffmann is too optimistic.

Hoffmann vs. TorchLight

Zenotravel	Hanoi [0]
Satellite	Airport [0]
Rovers	Blocksworld–Arm [30]
PSR	Mystery [39]
Pipesworld–Tank	Pipesworld–Tank [40]
Pipesworld–NoTank	Mprime [49]
Mystery	PSR [50]
Mprime	Freecell [56]
Freecell	Blocksworld–NoArm [57]
Driverlog	Pipesworld–NoTank [76]
Depots	Grid [80]
Blocksworld–Arm	Depots [81]
Airport	Zenotravel [95]
Tyreworld	Tyreworld [100]
Transport	Transport [100]
Simple–Tsp	Simple–Tsp [100]
Movie	Satellite [100]
Miconic–STRIPS	Rovers [100]
Logistics	Movie [100]
Hanoi	Miconic–STRIPS [100]
Gripper	Logistics [100]
Grid	Gripper [100]
Ferry	Ferry [100]
Elevators	Elevators [100]
Blocksworld–NoArm	Driverlog [100]

- Some new domains are "fully recognized" ...
- ... mostly because Hoffmann is too pessimistic.

Hoffmann vs. TorchLight

Zenotravel	Hanoi [0]
Satellite	Airport [0]
Rovers	Blocksworld–Arm [30]
PSR	Mystery [39]
Pipesworld–Tank	Pipesworld–Tank [40]
Pipesworld–NoTank	Mprime [49]
Mystery	PSR [50]
Mprime	Freecell [56]
Freecell	Blocksworld–NoArm [57]
Driverlog	Pipesworld–NoTank [76]
Depots	Grid [80]
Blocksworld–Arm	Depots [81]
Airport	Zenotravel [95]
Tyreworld	Tyreworld [100]
Transport	Transport [100]
Simple–Tsp	Simple–Tsp [100]
Movie	Satellite [100]
Miconic–STRIPS	Rovers [100]
Logistics	Movie [100]
Hanoi	Miconic–STRIPS [100]
Gripper	Logistics [100]
Grid	Gripper [100]
Ferry	Ferry [100]
Elevators	Elevators [100]
Blocksworld–NoArm	Driverlog [100]

- Success rates are more than a "yes/no" answer!

TorchLight Performance Summary

- Very fast. (Almost always, Fast Downward's STRIPS-2-FDR translator, which TorchLight uses to translate PDDL input to FDR, takes more runtime.)
- Global analysis succeeds (answer "I proved that there are no local minima at all in this planning task") in 4 domains of the table on slide 10.
- Approximate local analysis yields success rates that nicely correspond to how challenging the domain is for planners using delete relaxation heuristics (cf. previous slide).

  $\rightarrow$ Concretely, comparing runtime distributions for benchmark sets A vs. B whose success rate is below (A) vs. above (B) a threshold T, the average runtime of FF is statistically significantly smaller in A than in B. The same goes for LAMA.

Questionnaire

- Variables: $at : \{Sy, Ad, Br, Pe, Ad\}$;
  $v(x) : \{T, F\}$ for $x \in \{Sy, Ad, Br, Pe, Ad\}$.
- Actions: $drive(x, y)$ where x, y have a road, with
  $pre = (at = x)$, $eff = (at = y)$; and
  $visit(x)$ with $pre = (at = x)$, $eff = (v(x) = T)$.
- Initial state: $at = Sy, v(Sy) = T, v(x) = F$ for $x \neq Sy$.
- Goal: $at = Sy, v(x) = T$ for all x.

Question!

What are the causal graph $CG(\Pi)$ arcs in this example?

$\rightarrow (at, v(x))$ for $x \in \{Sy, Ad, Br, Pe, Ad\}$.

Question!

Does this example contain local minima under h^+?

$\rightarrow$ No. The $v(x)$ variables are egoistic; at is the support variable. To construct an exit path, drive to the nearest yet non-visited location x, then apply $visit(x)$.

The (Happy) Marriage of h^+ and h^m

→ Idea: Instead of h^+, consider $h^+(\Pi^C_{ce})$ where Π^C_{ce} is a compiled task capturing some of the reasoning underlying h^m.

- h^m reasons about all m-conjunctions of facts.
- Haslum [2009] devised Π^m where $h^m = h^1(\Pi^m)$. So Π^m captures reasoning about all m-conjunctions, even though Π^m *has empty delete lists*.
- But: $h^+(\Pi^m)$ is not admissible. Fixed by [Haslum (2012)] in Π^C, which furthermore allows to select an arbitrary conjunction subset C to reason about.
- But: Size of Π^C is exponential in $|C|$. Fixed by [Keyder *et al.* (2012)] in Π^C_{ce}.

→ We now introduce Π^m, Π^C, and Π^C_{ce}. We prove that $h^+(\Pi^C_{ce})$ converges i.e., $h^+(\Pi^C_{ce}) = h^*$ for sufficiently large C.

The Partner Agency: Where h^+ and h^m Meet

We refer to sets c of propositional facts, $|c| > 1$, as conjunctions.

Definition (π-Fluent). *Given a conjunction $c = \{p_1, \ldots, p_n\}$, the π-fluent for c is denoted π_c; it represents the truth value of the conjunction $p_1 \wedge \cdots \wedge p_n$. Given a fact set F and a set C of conjunctions, $F^C := F \cup \{\pi_c \mid c \in C, c \subseteq F\}$.*

$\rightarrow$ We need to arrange the compiled task in a way so that π_c "captures" the truth of c. (The formal meaning of "capture" will be: represent closely enough for our heuristics to converge to h^*.)

- The set of facts is given by F^C for some C.
- The goal is given by G^C.
- The initial state is given by I^C.
- When does an action a potentially make π_c true? If $add(a) \cap c \neq \emptyset$ and $del(a) \cap c = \emptyset$. (We assume that $add(a) \cap del(a) = \emptyset$.)

$\rightarrow$ The compilations Π^m, Π^C, Π^C_{ce} differ from each other in (A) how C can be chosen, and in (B) the construction of the actions.

The Π^m Compilation [Haslum (2009)]

Set C of Conjunctions: Fixed; $C = \{c \subseteq F \mid 1 < |c| \leq m\}$.

Compiled actions: One compiled action a^c for each conjunction $c \in C$ potentially made true by original action a:

- $pre(a^c) = [pre(a) \cup (c \setminus add(a))]^C$.
- $add(a^c) = add(a) \cup \{\pi_c\}$.
- $del(a^c) = \emptyset$.

Advantages:

- $h^m = h^1(\Pi^m)$ and thus convergence: Singleton-goal critical paths over π-fluents correspond to size-m-goal critical paths.

Disadvantages:

- No choice of C.
- $h^+(\Pi^m)$ is not admissible: Each π^c requires a different action representative (see also next slide).

Π^m Example

Example (Π^m example)

Consider the STRIPS planning task $\Pi = (P, I, G, A)$ with: $P = \{p, q, r\}$; $I = \{p, q\}$; $G = \{p, q, r\}$; $A = \{a\}$ where $pre_a = \emptyset$, $add_a = \{r\}$, $del_a = \emptyset$. In Π^2, we have:

- $P^C = \{p, q, r, \pi_{\{p,q\}}, \pi_{\{p,r\}}, \pi_{\{q,r\}}\}$.
- $I^C = \{p, q, \pi_{\{p,q\}}\}$.
- $G^C = \{p, q, r, \pi_{\{p,q\}}, \pi_{\{p,r\}}, \pi_{\{q,r\}}\}$.
- $A^C = \{a, a^{\{p,r\}}, a^{\{q,r\}}\}$, where
  - $pre(a^{\{p,r\}}) = \{p\}$, $add(a^{\{p,r\}}) = \{r, \pi_{\{p,r\}}\}$; and
  - $pre(a^{\{q,r\}}) = \{q\}$, $add(a^{\{q,r\}}) = \{r, \pi_{\{q,r\}}\}$.

$\rightarrow h^*(\Pi) = 1$, yet $h^+(\Pi^m) = 2$, as both compiled actions $a^{\{p,r\}}$ and $a^{\{q,r\}}$ are required to achieve $\pi_{\{p,r\}}$ and $\pi_{\{q,r\}}$.

The Π^C Compilation [Haslum (2012)]

Set C of Conjunctions: Arbitrary.

Compiled actions: One compiled action $a^{C'}$ for each set $C' \subseteq C$ of conjunctions potentially made true by original action a:

- $pre(a^{C'}) = [pre(a) \cup \bigcup_{c' \in C'}(c' \setminus add(a))]^C$.
- $add(a^{C'}) = add(a) \cup \{\pi_c \mid c \in C'\}$.
- $del(a^{C'}) = \emptyset$.

Advantages:

- $h^+(\Pi^C)$ is admissible: A plan for Π can be transformed into a (relaxed) plan for Π^C by always selecting the maximal C' made true by an action.
- $h^+(\Pi^C)$ converges: See later.

Disadvantages:

- Enumerates subsets of $C \implies$ size of Π^C is worst-case exponential in $|C|$.

Π^C Example

Example (Π^C example)

Consider the STRIPS planning task $\Pi = (P, I, G, A)$ with: $P = \{p, q, r\}$; $I = \{p, q\}$; $G = \{p, q, r\}$; $A = \{a\}$ where $pre_a = \emptyset$, $add_a = \{r\}$, $del_a = \emptyset$. In Π^C with $C = \{c \subseteq P \mid 1 < |c| \leq 2\}$, we have:

- P^C, I^C, G^C: As before.
- $A^C = \{a^{\emptyset}, a^{\{\{p,r\}\}}, a^{\{\{q,r\}\}}, a^{\{\{p,r\},\{q,r\}\}}\}$, where
  - $pre(a^{\emptyset}) = \emptyset$, $add(a^{\emptyset}) = \{r\}$;
  - $pre(a^{\{\{p,r\}\}}) = \{p\}$, $add(a^{\{\{p,r\}\}}) = \{r, \pi_{\{p,r\}}\}$;
  - $pre(a^{\{\{q,r\}\}}) = \{q\}$, $add(a^{\{\{q,r\}\}}) = \{r, \pi_{\{q,r\}}\}$;
  - $pre(a^{\{\{p,r\},\{q,r\}\}}) = \{p, q, \pi_{\{p,q\}}\}$, $add(a^{\{\{p,r\},\{q,r\}\}}) = \{r, \pi_{\{p,r\}}, \pi_{\{q,r\}}\}$.

$\rightarrow h^+(\Pi^C) = 1 = h^*(\Pi)$, as the compiled action $a^{\{\{p,r\},\{q,r\}\}}$ achieves $\pi_{\{p,r\}}$ and $\pi_{\{q,r\}}$ simultaneously.

The Π^C_{ce} Compilation [Keyder *et al.* (2012)]

Set C of Conjunctions: Arbitrary.

Compiled actions: One compiled action a^C for each original action a:

- $pre(a^C) = pre(a)^C$, $del(a^C) = \emptyset$.
- $add(a^C) = add(a)$.
- For each conjunction $c \in C$ potentially made true by a, a conditional effect with condition $[pre(a) \cup (c \setminus add(a))]^C$ and add π_c.

Advantages:

- $h^+(\Pi^C_{ce})$ is admissible: Any plan for Π is a (relaxed) plan for Π^C_{ce}.
- $h^+(\Pi^C_{ce})$ converges: See later.
- Growth linear in $|C|$.

Disadvantages:

- Information loss relative to Π^C: no cross-context conditions, see next slide.

Π^C_{ce} Example

Example (Π^C_{ce} example)

Consider the STRIPS planning task $\Pi = (P, I, G, A)$ with: $P = \{p, q, r\}$; $I = \{p, q\}$; $G = \{p, q, r\}$; $A = \{a\}$ where $pre_a = \emptyset$, $add_a = \{r\}$, $del_a = \emptyset$. In Π^C_{ce} with $C = \{c \subseteq P \mid 1 < |c| \leq 2\}$, we have:

- P^C, I^C, G^C: As before.
- $A^C = \{a^C\}$, where
  - $pre(a^C) = \emptyset$; $add(a^C) = \{r\}$;
  - conditional effect 1: cond $\{p\}$, add $\{\pi_{\{p,r\}}\}$;
  - conditional effect 2: cond $\{q\}$, add $\{\pi_{\{q,r\}}\}$.

$\rightarrow h^+(\Pi^C_{ce}) = 1$, as applying a triggers both conditional effects.

$\rightarrow$ So where is the information loss?

$\rightarrow$ In Π^C, triggering both $\pi_{\{p,r\}}$ and $\pi_{\{q,r\}}$, by compiled action $a^{\{\{p,r\},\{q,r\}\}}$, has the cross-context precondition $\pi_{\{p,q\}}$. That precondition is missing in Π^C_{ce}! E.g., if p and q are mutex, then $h^+(\Pi^C_{ce})$ misses a dead end.

Convergence Proof for $h^+(\Pi^C)$

→ What do we need to prove? For every planning task Π there exists C so that $h^+(\Pi^C) = h^*(\Pi)$.

→ Anybody got an idea how to prove this? Inherit this property from h^m and thus from $h^1(\Pi^m)$.

Lemma. *Let $\Pi = (P, I, G, A)$ be a STRIPS planning task, and let $C = \{c \subseteq P \mid 1 < |c| \leq m\}$. Then $h^1(\Pi^m) = h^1(\Pi^C)$.*

Proof. Π^m and Π^C are identical except for the action sets. We first note that h^1 values are computed by considering only a single add effect of an action at a time. The inequality $h^1(\Pi^m) \leq h^1(\Pi^C)$ is then easy to see by verifying that, for every add effect π_c of an action $a^{C'}$ in Π^C, the action a^c of Π^m *dominates* this add effect of $a^{C'}$, i.e., $\pi_c \in add(a^c)$ and $pre(add(a^c)) \subseteq pre(a^{C'})$. The proof is similar for the inequality $h^1(\Pi^C) \leq h^1(\Pi^m)$, observing that for any action a^c in Π^m, the action $a^{\{c\}}$ in Π^C dominates the add effects of a^c.

→ In our example: e.g., $a^{\{p,r\}}$ in Π^m dominates the add effect $\pi_{\{p,r\}}$ of $a^{\{\{p,r\},\{q,r\}\}}$ in Π^C. Vice versa, $a^{\{\{p,r\}\}}$ in Π^C is equal to (and thus dominates every add effect of) $a^{\{p,r\}}$ in Π^m.

Convergence Proof for $h^+(\Pi^C)$, ctd.

Theorem. *Let $\Pi = (P, I, G, A)$ be a STRIPS planning task. Then there exists C such that $h^+(\Pi^C) = h^*(\Pi)$.*

Proof. Trivially, $h^*(\Pi) = h^m(\Pi)$ for sufficiently high m. Haslum [2009] showed that $h^m(\Pi) = h^1(\Pi^m)$. By the lemma on the previous slide, for $C = \{c \subseteq P \mid 1 < |c| \leq m\}$ we have $h^1(\Pi^m) = h^1(\Pi^C)$. Choosing an appropriate m and the corresponding C, we thus have that $h^*(\Pi) = h^m(\Pi) = h^1(\Pi^m) = h^1(\Pi^C)$. Together with the fact that $h^1(\Pi^C) \leq h^+(\Pi^C)$, and since $h^+(\Pi^C) \leq h^*(\Pi)$ by admissibility of $h^+(\Pi^C)$, the claim follows.

Convergence Proof for $h^+(\Pi^C_{ce})$

→ What do we need to prove? For every planning task Π there exists C so that $h^+(\Pi^C_{ce}) = h^*(\Pi)$.

→ Anybody got an idea how to prove this? Inherit this property from Π^C.

Lemma. *Let $\Pi = (P, I, G, A)$ be a STRIPS planning task. Then $h^1(\Pi^C) \leq h^+(\Pi^C_{ce})$.*

Proof. Consider a planning task $\Pi^C_{\text{no-cc}}$ identical to Π^C except that it drops cross-context π-fluents from preconditions. We show that (A) $h^1(\Pi^C) \leq h^1(\Pi^C_{\text{no-cc}})$, and (B) $h^1(\Pi^C_{\text{no-cc}}) \leq h^+(\Pi^C_{ce})$.

For (A), every add effect π_c of an action $a^{C'}$ in $\Pi^C_{\text{no-cc}}$ is dominated by the action $a^{\{c\}}$ in Π^C: Reducing the conjunction set to the singleton $\{c\}$ gets rid of any cross-context preconditions.

For (B), it suffices to show that $h^+(\Pi^C_{\text{no-cc}}) \leq h^+(\Pi^C_{ce})$. That holds because relaxed plans for Π^C_{ce} can be transformed into relaxed plans for Π^C: For action a in a relaxed plan for Π^C_{ce}, if C' is the set of conjunctions that are added by conditional effects of a when it is applied in that plan, then the action $a^{C'}$ in $\Pi^C_{\text{no-cc}}$ has the same preconditions and effects as a.

Convergence Proof for $h^+(\Pi^C_{ce})$, ctd.

Theorem. *Let $\Pi = (P, I, G, A)$ be a STRIPS planning task. Then there exists C such that $h^+(\Pi^C_{ce}) = h^*(\Pi)$.*

Proof. Choosing an appropriate m, we have $h^*(\Pi) = h^m(\Pi) = h^1(\Pi^m)$. Choosing an appropriate C, by the lemma on slide 33 we get $h^1(\Pi^m) = h^1(\Pi^C)$, and by the lemma on the previous slide we get $h^1(\Pi^C) \leq h^+(\Pi^C_{ce})$. Since $h^+(\Pi^C_{ce}) \leq h^*(\Pi)$ by admissibility of $h^+(\Pi^C_{ce})$, the claim follows.

Remarks

Some more stuff we already know:

- One can choose C in practice by a kind of abstraction refinement process on the initial state: Compute a relaxed plan, extract reasons why it doesn't work on the real task, extract conjunctions pertaining to that reason ([Haslum (2012)] does this very systematically).
- Current methods drastically improve performance of satisficing planning, but only in very few cases.
- Hardly any improvement for optimal planning with LM-cut and admissible landmarks.

Some stuff we don't know yet: (a small selection :-)

- Better methods for selecting C?
- When does it work well, when doesn't it?
- Anything to be done about better admissible heuristics?

Questionnaire

- Facts: $at(x)$ for $x \in \{Sy, Ad, Br, Pe, Ad\}$; $v(x)$ for $x \in \{Sy, Ad, Br, Pe, Ad\}$.
- Actions: $drive(x, y)$ where x, y have a road, with $pre = at(x)$, $eff = at(y)$; and $visit(x)$ with $pre = at(x)$, $eff = v(x)$.
- Initial state: $at = Sy, v(Sy) = T, v(x) = F$ for $x \neq Sy$.
- Goal: $at = Sy, v(x) = T$ for all x.

Question!

What is $h^*(\Pi)$? What is $h^+(\Pi)$?

$\rightarrow h^*(\Pi) = 8$: We need to take each road segment twice. $h^+(\Pi) = 4$: The relaxed plan does not need to drive back.

Question!

What is $h^+(\Pi^C_{ce})$ for $C = \{\{at(x), at(y)\} \mid x \neq y\}$?

$\rightarrow$ Still $h^+(\Pi^C_{ce}) = 4$: The goal remains the same and does not contain any π-fluents; the regular fluents are achieved by the same relaxed plan as before.

Questionnaire, ctd.

- Facts: $at(x)$ for $x \in \{Sy, Ad, Br, Pe, Ad\}$; $v(x)$ for $x \in \{Sy, Ad, Br, Pe, Ad\}$.
- Actions: $drive(x, y)$ where x, y have a road, with $pre = at(x)$, $eff = at(y)$; and $visit(x)$ with $pre = at(x)$, $eff = v(x)$.
- Initial state: $at = Sy, v(Sy) = T, v(x) = F$ for $x \neq Sy$.
- Goal: $at = Sy, v(x) = T$ for all x.

Question!

What if $C = \{\{at(Sy), v(y)\} \mid y \in \{Ad, Br, Pe, Ad\}\}$?

$\rightarrow$ Then $h^+(\Pi^C_{ce}) = 5$: To achieve the goals $\pi_{\{at(Sy),v(y)\}}$, it suffices to achieve each $v(x)$ first, then execute an arbitrary action driving into Sydney.

Question!

How do we need to define C to obtain $h^+(\Pi^C_{ce}) = h^*(\Pi) = 8$?

$\rightarrow$ Proof: Use $m = 10$. Can we do better? Yes: $C = \{\{at(x), v(y)\}\}$. Then $drive(x, z)$ achieves $\pi_{\{at(z),v(y)\}}$ only under condition $\pi_{\{at(x),v(y)\}}$ so achieving $\pi_{\{at(Sy),v(y)\}}$ requires the relaxed plan to go back to Sydney from y.

Who Said We Need to Relax *All* Variables?

→ Idea: Red-black planning relaxes only *some* variables! These red variables accumulate their values, while the others, the black variables, retain the true (value-switching) semantics.

- Earlier works: Identify variable x that moves freely and independently, and account for a tour of x that visits all values of x required in a relaxed plan [Fox and Long (2001); Keyder and Geffner (2008)].

  → Drawbacks: Strict requirements on x, just one variable, tour through required values not necessarily meaningful (e.g.: two values between which we actually need to switch back and forth).
- Katz et al. [2013b]: All these drawbacks disappear, in principle, when using red-black planning instead.
- Katz et al. [2013a]: A concrete red-black plan heuristic that can be computed in polynomial time.

→ We now introduce red-black planning, and the tractability result underlying the implemented heuristic function.

Red-Black Planning

Definition (Red-Black Planning). *A red-black planning task is a tuple $\Pi = (V^{\mathsf{B}}, V^{\mathsf{R}}, I, G, A)$ where V^{B} is a set of black variables, V^{R} is a set of red variables, and everything else is exactly as for FDR tasks. The semantics is:*

- *A state s assigns each $x \in V^{\mathsf{B}} \cup V^{\mathsf{R}}$ a subset $s[x] \subseteq D_x$, where $|s[x]| = 1$ for all $x \in V^{\mathsf{B}}$.*
- *Action a is applicable in s iff $pre(a)[x] \in s[x]$ for all x mentioned in $pre(a)$.*
- *Applying a in s changes the value of black effect variables x to $\{eff(a)[x]\}$, and changes the value of red effect variables x to $s[x] \cup \{eff(a)[x]\}$.*
- *An action sequence $\langle a_1, \dots, a_k \rangle$ is a red-black plan if $G[x] \in I[\![\langle a_1, \dots, a_k \rangle]\!][x]$ for all x mentioned in G.*

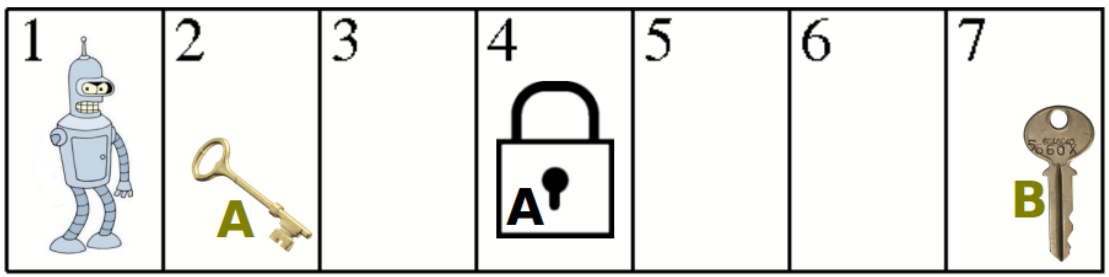
Given an FDR task $\Pi = (V, I, G, A)$ and a subset $V^{\mathsf{R}} \subseteq V$ of variables, the red-black relaxation of Π is the red-black task $\Pi^{*+} = (V \setminus V^{\mathsf{R}}, V^{\mathsf{R}}, I, G, A)$. A plan for Π^{*+} is a red-black relaxed plan for Π, and the length of a shortest possible red-black relaxed plan is denoted $h^{*+}(\Pi)$.

$\rightarrow$ If we set $V^{\mathsf{R}} := V$, then $h^{*+}(\Pi) = h^+(\Pi)$.

$\rightarrow$ If we set $V^{\mathsf{R}} := \emptyset$, then $h^{*+}(\Pi) = h^*(\Pi)$.

Red-Black Planning: "Grid" Example

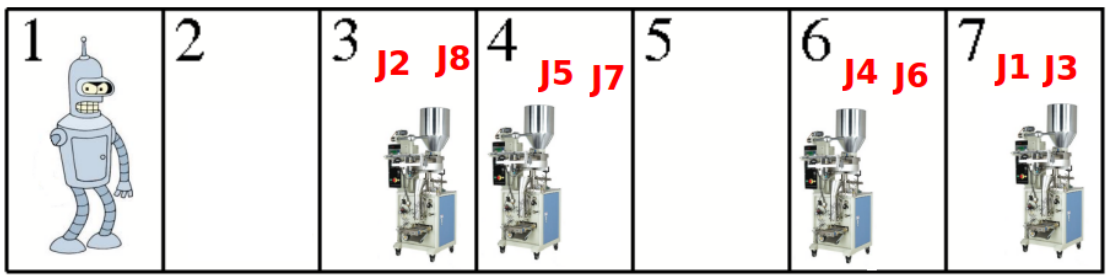
- Variables: robot position R; key A position A, key B position B; hand-free: F in $\{0, 1\}$; lock-open O in $\{0, 1\}$.
- Initial state: As shown. Goal: $B = 1$.
- Actions: *take*(x, y); *drop*(x, y); *open*(x, y); *move*(x, y) where $|x - y| = 1$, precondition $O = 1$ in case $\{x, y\} \cap \{4\} \neq \emptyset$.



- Plan for this task? Move to 2, take key A at 2, move to 3, open lock, move to 7, drop key A at 7, take key B at 7, move to 1, drop key B at 1. $h^*(\Pi) = 17$.
- Relaxed plan for this task? ..., move to 7, take key B at 7, drop key B at 1. $h^+(\Pi) = 10$.
- Red-black plan for this task if $V^R = \{R, A, B, O\}$? ..., move to 7, drop key A at 7, take key B at 7, drop key B at 1. $h^{*+}(\Pi) = 11$.
- Red-black plan for this task if $V^R = \{A, B, O\}$? ..., move to 7, drop key A at 7, take key B at 7, move to 1, drop key B at 1. $h^{*+}(\Pi) = 17$.

Red-Black Planning: "Jobs" Example

- Variables: robot position R; jobs $J_1, \ldots, J_n$ in $\{0, 1\}$.
- Initial state: $R = 1, J_1 = 0, \ldots, J_n = 0$. Goal: $J_1 = 1, \ldots, J_n = 1$.
- Actions: *move*(x, y) where $|x - y| = 1$; *do-job*(i, x) precondition $J_i = 0$, $J_{i-1} = 1$, and $R = x$ as shown.



- Plan for this task? Move to jobs in sequence as shown. $h^*(\Pi) = 30$.
- Relaxed plan for this task? Move across to 7 once, do jobs. $h^+(\Pi) = 14$.
- Accounting for a "tour" of x? Move across to 7 once, do jobs. "Refined" heuristic $h(\Pi) = 14$.
- Red-black plan for this task if $V^{\mathrm{R}} = \{J_1, \ldots, J_n\}$? Move to jobs in sequence as shown. $h^{*+}(\Pi) = 30$.

Ok, Ok. But How To *Generate* Red-Black Plans?

→ Follow the relaxed plan heuristic approach: Generate a red-black relaxed plan for every search state, take its length as the distance estimate.

For this, red-black plan generation must be tractable! Is it?

- In general: no. (Set $V^{\mathrm{R}} := \emptyset$.)
- Need to identify tractable fragments!

Results by Katz et al. [2013b]:

- Tractable if number and size of black variables constant. . . . Runtime then exponential in the product of their domain sizes :-(
- Plan *existence* tractable if black causal graph is acyclic and task is reversible. . . . Fine, but we need *plan generation* not *plan existence* :-(

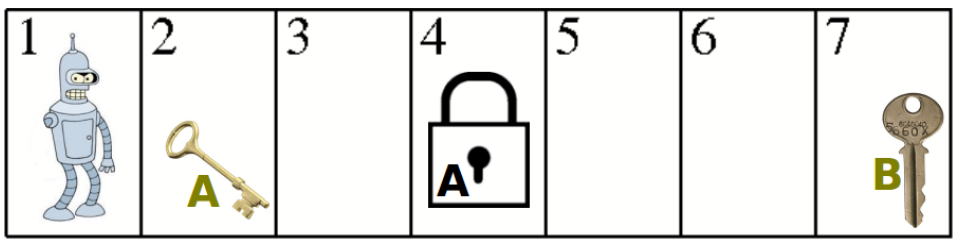
Result by Katz et al. [2013a]:

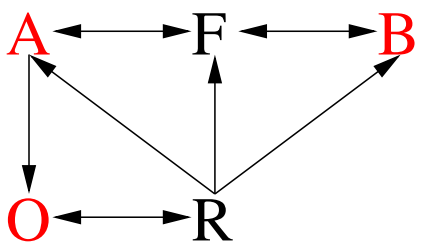
- Tractable if black causal graph is acyclic and all black variables are relaxed side effects invertible (RSE-invertible).

Black Causal Graphs

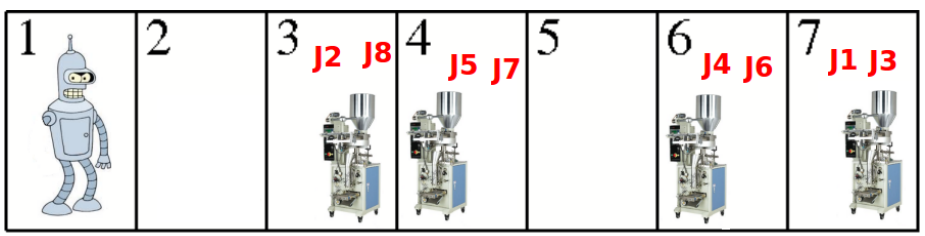
Definition (Black Causal Graph). *Let $\Pi = (V^{\mathsf{B}}, V^{\mathsf{R}}, I, G, A)$ be a red-black planning task. The causal graph is a digraph with vertices V, and has an arc (x, y) iff $x \neq y$ and there exists an action $a \in A$ such that either (i) x appears in $pre(a)$ and y appears in $eff(a)$, or (ii) x and y both appear in $eff(a)$. The black causal graph is the sub-graph of the causal graph induced by V^{B}.*

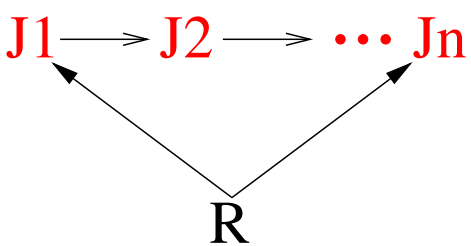
→ Causal graph in Grid: (black causal graph acyclic)





→ Causal graph in Jobs: (black causal graph acyclic)





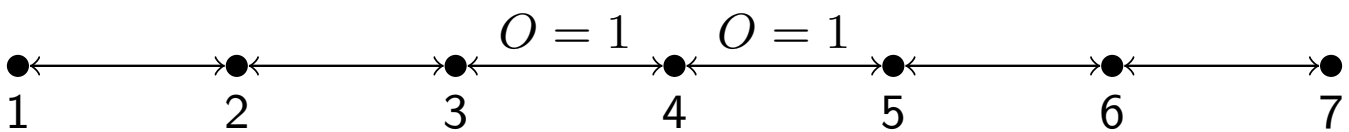
Black DAG Invertibility

Definition (RSE-Invertibility). *Let $\Pi = (V^{\mathsf{B}}, V^{\mathsf{R}}, I, G, A)$ be a red-black planning task, let $V = V^{\mathsf{B}} \cup V^{\mathsf{R}}$ and let $x \in V$.*

- *The domain transition graph (DTG) of x is a labeled digraph with vertices D_x, and an arc (d, d') induced by action a iff $\mathit{eff}(a)[x] = d'$, and either $\mathit{pre}(a)[x] = d$ or x is not mentioned in $\mathit{pre}(a)$. The arc is labeled with its outside condition $\mathit{pre}(a)[V \setminus \{x\}]$ and its outside effect $\mathit{eff}(a)[V \setminus \{x\}]$.*
- *An arc (d, d') is RSE-invertible if there exists an arc (d', d) with outside condition $\phi' \subseteq \phi \cup \psi$ where ϕ and ψ are the outside condition respectively outside effect of (d, d'). Variable x is RSE-invertible if all arcs in its DTG are RSE-invertible.*

$\rightarrow$ Intuition: To go back over transition (d, d'), (d', d) allows to use outside conditions *and* outside effects of (d, d'), which is valid provided these outside effects are red. (As is the case in acyclic black causal graphs.)

$\rightarrow$ DTG for R in Grid example: (RSE-invertible, as is that for F)



The Tractable Fragment

Theorem. *Red-black plan generation restricted to red-black planning tasks whose black causal graph is acyclic, and all of whose black variables are RSE-invertible, is polynomial-time.*

→ The theorem does not impose *any* restriction on the red variables!

- Are the theorem prerequisites satisfied in Grid with $V^{\mathsf{R}} = \{A, B, O\}$?

  → Yes! (Which is *good*, cf. slide 43.)
- Are the theorem prerequisites satisfied in Jobs with $V^{\mathsf{R}} = \{J_1, \ldots, J_n\}$?

  → Yes! (Which is *good*, cf. slide 44.)

→ We next prove that, under the given prerequisites, any relaxed plan for the task can be efficiently post-processed into a red-black plan for the task.

→ Why does this prove the theorem? If no relaxed plan exists, then trivially no red-black plan exists.

The Tractable Fragment: Proof

$\pi := \langle a_1 \rangle$ // $\Pi = \langle V^{\mathsf{B}}, V^{\mathsf{R}}, I, G, A \rangle$ *and* $\pi^+ = \langle a_1, \ldots, a_n \rangle$ *is a relaxed plan for* Π
for $i = 2$ to n **do**
  if $pre(a_i)[V^{\mathsf{B}}] \not\subseteq I[\![\pi]\!]$ **then**
    $\pi^{\mathsf{B}} :=$ Achieve$(pre(a_i)[V^{\mathsf{B}}])$
    $\pi := \pi \circ \pi^{\mathsf{B}}$
  endif
  $\pi := \pi \circ \langle a_i \rangle$
endfor
if $G[V^{\mathsf{B}}] \not\subseteq I[\![\pi]\!]$ **then**
  $\pi^{\mathsf{B}} :=$ Achieve$(G[V^{\mathsf{B}}])$
  $\pi := \pi \circ \pi^{\mathsf{B}}$
endif
return π

→ So what is the main idea here? In between any two actions from the relaxed plan (and in front of the goal), insert sub-plans π^{B} achieving any conditions required on the black variables.

→ And what remains to prove? That the sub-plans π^{B} can be constructed in polynomial time – how to "Achieve$(pre(a_i)[V^{\mathsf{B}}])$"?

The Tractable Fragment: Proof, ctd.

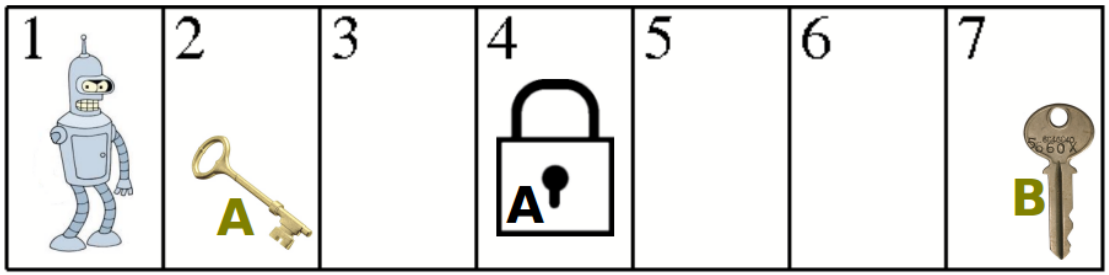
→ Constructing π^{B} = solving a (regular) planning task Π^{B} over black variables:

Procedure: Achieve(π, g)
$F := I \cup \bigcup_{a \in \pi} \mathit{eff}(a)$ *// (i) every fact touched by our prefix π so far*
for $x \in V^{\mathsf{B}}$ **do** $D^{\mathsf{B}}(x) := \{d \mid d \in D(x), (x,d) \in F\}$ **endfor**
$I^{\mathsf{B}} := I[\![\pi]\!][V^{\mathsf{B}}]$, $G^{\mathsf{B}} := g$ *// (ii) end point of prefix/black condition needed*
// actions whose (iii) preconditions (red and black) we have touched,
//and whose (iv) black effects we have touched
$A^{\mathsf{B}} := \{a^{\mathsf{B}} \mid \text{ex. } a \in A, \mathit{pre}(a) \subseteq F, \mathit{eff}(a)[V^{\mathsf{B}}] \subseteq F, a^{\mathsf{B}} = \langle \mathit{pre}(a)[V^{\mathsf{B}}], \mathit{eff}(a)[V^{\mathsf{B}}]\rangle\}$
$\Pi^{\mathsf{B}} := \langle V^{\mathsf{B}}, I^{\mathsf{B}}, G^{\mathsf{B}}, A^{\mathsf{B}}\rangle$
$\langle a'^{\mathsf{B}}_1, \ldots, a'^{\mathsf{B}}_k\rangle :=$ an FDR plan for Π^{B}
return $\pi^{\mathsf{B}} := \langle a'_1, \ldots, a'_k\rangle$

- By (ii), π^{B} has the desired start/end. By (iii) and (iv), Π^{B} is well-defined, and by (iii) any red conditions of $\langle a'_1, \ldots, a'_k\rangle$ are true in $I[\![\pi]\!]$.
- Π^{B} has an acyclic causal graph.
- By (i), all DTGs in Π^{B} are strongly connected: Every value is reached by π, and with RSE-invertibility we can invert these paths.
- So Π^{B} is polynomial-time solvable, under a succinct plan representation (macros) [Chen and Giménez (2008)].

The Tractable Fragment: "Grid" Example

Relaxed plan π^+: Move to 2, take key A at 2, move to 3, open lock, move to 7, take key B at 7, drop key B at 1. Say that $R = A, B, O$.



- When does the algorithm call "Achieve($pre(a_i)[V^{\mathsf{B}}]$)" the first time? In front of "take key B at 7": The hand is not empty, i.e., black variable $F = 0$ does not have the required value $F = 1$.
- What does Π^{B} look like at this point? Variables R, F with their full domains, $I = \{R = 7, F = 0\}$, $G = \{F = 1\}$; all move actions (projected to R, i.e.without condition on lock); pick actions for key A at 2 and key B at 7, projected to F; drop actions for key A anywhere. We get π^{B} dropping key A at 7.
- When does the algorithm call "Achieve($pre(a_i)[V^{\mathsf{B}}]$)" the next time? In front of "drop key B at 1": The robot is at the wrong position, i.e., black variable $R = 7$ does not have the required value $R = 1$.
- What does Π^{B} look like at this point? Same as above, except $I = \{R = 7, F = 0\}$, $G = \{R = 1\}$. We get π^{B} moving R to 1.

Remarks

Some more stuff we already know:

- A simple way to make this *real* fast is to reduce the black causal graph to have *no arcs at all* (rather than being acyclic): Solving Π^{B} then just means moving each black var into place individually.
- One can choose the red variables by simple greedy methods preferring, e.g., to remove a lot of arcs from the black causal graph, or to remove variables that do not have many "conflicts" in a relaxed plan for the initial state.
- The presented algorithm may heavily over-estimate due to arbitrary relaxed plan choices (just try to re-order π^+ a bit on the previous slide) ...
- With an algorithm that relies less on π^+, we currently improve the relaxed plan heuristic almost consistently, and outperform it in several domains.

Some stuff we don't know yet: (a small selection :-)

- Better methods for selecting the red variables? When does it work well, when doesn't it? Anything to be done about admissible heuristics?
- Combination with Π^C_{ce}?

Questionnaire

- Variables: $at : \{Sy, Ad, Br, Pe, Ad\}$; $v(x) : \{T, F\}$ for $x \in \{Sy, Ad, Br, Pe, Ad\}$.
- Actions: $drive(x, y)$ where x, y have a road, with $pre = (at = x)$, $eff = (at = y)$; and $visit(x)$ with $pre = (at = x)$, $eff = (v(x) = T)$.
- Initial state: $at = Sy, v(Sy) = T, v(x) = F$ for $x \neq Sy$.
- Goal: $at = Sy, v(x) = T$ for all x.

Question!

What is a maximal set of red variables so that $h^{*+}(\Pi) = h^*(\Pi) = 12$?

$\rightarrow$ Simply paint all $v(x)$ variables red: With just the at variable black, any red-black plan is a real plan.

Question!

Do you see a general rule here which variables we can safely paint red?

$\rightarrow$ Leaf variables (no outgoing arcs) in the causal graph: Such variables are "egoistic" (see up front), and any non-redundant red-black plan is a real plan since it moves each of them along a simple (acyclic) DTG path to its goal value.

Summary

- Search space surface analysis has proved very suitable to understand h^+ (though not any other heuristic function in planning, yet).
- We can identify fragments with particular topology (no local minima) using similar criteria as used for identifying tractable fragments (causal graphs, domain transition graphs).
- We can combine delete relaxation heuristics with critical path heuristics by computing the former on compiled tasks representing some of the reasoning performed in the latter.
- This interpolates between relaxed planning and real planning in that sufficiently large compilations enforce convergence to h^*.
- A much easier way to interopolate between relaxed planning and real planning is to simply relax ("paint red") only some of the variables.
- There are significant tractable fragments of red-black planning, that we have only begun to exploit.

Reading

- *Analyzing Search Topology Without Running Any Search: On the Connection Between Causal Graphs and h^+* [Hoffmann (2011a)].

  Available at:

  `http://fai.cs.uni-saarland.de/hoffmann/papers/jair11.pdf`

  Content: Detailed paper on TorchLight, proving a more general version of the theorem outlined here, and giving a comprehensive empirical evaluation. Maybe you'd rather read the short version [Hoffmann (2011b)] which summarizes these results.

Reading

- *Semi-Relaxed Plan Heuristics* [Keyder *et al.* (2012)].

  Available at:

  `http://fai.cs.uni-saarland.de/hoffmann/papers/icaps12a.pdf`

  Content: The Π^C_{ce} encoding and its practical use in satisficing planning.

- *Incremental Lower Bounds for Additive Cost Planning Problems* [Haslum (2012)].

  Available at:

  `http://www.aaai.org/ocs/index.php/ICAPS/ICAPS12/paper/view/4703/4716`

  Content: The Π^C encoding and its practical use for improved lower bounds in optimal planning. Analysis how to extract conflicts in a way so that a relaxed plan will not "make the same mistake again".

Reading

- *Who Said We Need to Relax All Variables?* [Katz *et al.* (2013b)].

  Available at:

  `http://fai.cs.uni-saarland.de/hoffmann/papers/icaps13a.pdf`

  Content: Introduces the red-black framework and identifies two tractable fragments, neither of which is of immediate practical use. (Theory paper.)

- *Red-Black Relaxed Plan Heuristics* [Katz *et al.* (2013a)].

  Available at:

  `http://fai.cs.uni-saarland.de/hoffmann/papers/aaai13.pdf`

  Content: Refines one of the two previous fragments to a more restricted fragments, namely the one presented here, that *is* of immediate practical use. Experiments with that fragment and simple ways of instantiating the other elements needed.

References I

Hubie Chen and Omer Giménez. Causal graphs and structurally restricted planning. In Jussi Rintanen, Bernhard Nebel, J. Christopher Beck, and Eric Hansen, editors, *Proceedings of the 18th International Conference on Automated Planning and Scheduling (ICAPS 2008)*, pages 36–43. AAAI Press, 2008.

Maria Fox and Derek Long. Hybrid STAN: Identifying and managing combinatorial optimisation sub-problems in planning. In B. Nebel, editor, *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI-01)*, pages 445–450, Seattle, Washington, USA, August 2001. Morgan Kaufmann.

Patrik Haslum. $h^m(P) = h^1(P^m)$: Alternative characterisations of the generalisation from $h^{\max}$ to h^m. In Alfonso Gerevini, Adele Howe, Amedeo Cesta, and Ioannis Refanidis, editors, *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS 2009)*, pages 354–357. AAAI Press, 2009.

Patrik Haslum. Incremental lower bounds for additive cost planning problems. In Blai Bonet, Lee McCluskey, José Reinaldo Silva, and Brian Williams, editors, *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS 2012)*, pages 74–82. AAAI Press, 2012.

References II

Jörg Hoffmann. Local search topology in planning benchmarks: A theoretical analysis. In M. Ghallab, J. Hertzberg, and P. Traverso, editors, *Proceedings of the 6th International Conference on Artificial Intelligence Planning and Scheduling (AIPS-02)*, pages 92–100, Toulouse, France, April 2002. Morgan Kaufmann.

Jörg Hoffmann. Where 'ignoring delete lists' works: Local search topology in planning benchmarks. *Journal of Artificial Intelligence Research*, 24:685–758, 2005.

Jörg Hoffmann. Analyzing search topology without running any search: On the connection between causal graphs and h^+. *Journal of Artificial Intelligence Research*, 41:155–229, 2011.

Jörg Hoffmann. Where ignoring delete lists works, part II: Causal graphs. In Fahiem Bacchus, Carmel Domshlak, Stefan Edelkamp, and Malte Helmert, editors, *Proceedings of the 21st International Conference on Automated Planning and Scheduling (ICAPS 2011)*, pages 98–105. AAAI Press, 2011.

Michael Katz, Jörg Hoffmann, and Carmel Domshlak. Red-black relaxed plan heuristics. In Marie desJardins and Michael Littman, editors, *Proceedings of the 27th National Conference of the American Association for Artificial Intelligence (AAAI'13)*, Bellevue, WA, USA, July 2013. AAAI Press. Forthcoming.

References III

Michael Katz, Jörg Hoffmann, and Carmel Domshlak. Who said we need to relax *All* variables? In Daniel Borrajo, Simone Fratini, Subbarao Kambhampati, and Angelo Oddi, editors, *Proceedings of the 23rd International Conference on Automated Planning and Scheduling (ICAPS 2013)*. AAAI Press, 2013. Forthcoming.

Emil Keyder and Hector Geffner. Heuristics for planning with action costs revisited. In Malik Ghallab, editor, *Proceedings of the 18th European Conference on Artificial Intelligence (ECAI-08)*, pages 588–592, Patras, Greece, July 2008. Wiley.

Emil Keyder, Jörg Hoffmann, and Patrik Haslum. Semi-relaxed plan heuristics. In Blai Bonet, Lee McCluskey, José Reinaldo Silva, and Brian Williams, editors, *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS 2012)*. AAAI Press, 2012.