A Reminder about the Importance of Computing and Exploiting Invariants in Planning

Vidal Alcázar, Álvaro Torralba

PLG @ Universidad Carlos III de Madrid
http://www.plg.inf.uc3m.es/~valcazar
FAI @ Saarland University
http://fai.cs.uni-saarland.de/torralba/

ICAPS – June 9, 2015

Motivation

Invariants are known to be useful:

- FDR representation, regression, partial-order planning, SAT,...
- Several methods proposed: here h^2

Some aspects have been overlooked and/or appear scattered in the literature:

- Implementation details of h^2
- Direction of the computation of the invariants
- Huge impact in some domains!

Background

State invariants:

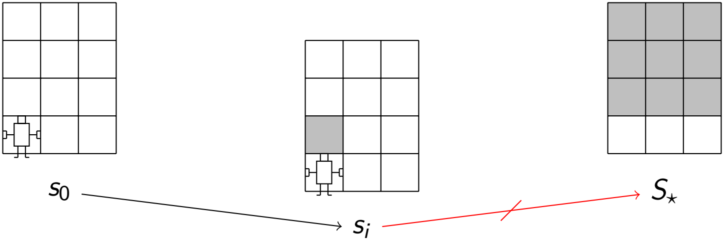
- Mutexes: $\neg$((at robot loc_1) $\wedge$ (at robot loc_2))
- "exactly-one" invariant groups:
  ((at robot loc_1) $\vee \cdots \vee$ (at robot loc_n)) + pairwise mutexes

A (slightly) more general definition of spurious state:

- State that cannot belong to a solution path
  $\Rightarrow$ instead of state unreachable from s_0
- Detectable when they are inconsistent with invariants

Spurious State

Floortile domain: robots can only paint up or down



- s_i is a forward dead end, and hence spurious
  - ... but does it violate some invariant?

How does h^2 work?

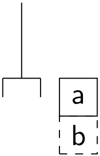
Reachability analysis in P^2: with conjunctions of two original atoms

- Unreachable h^2 atoms are mutexes
  - (at robot loc1) $\wedge$ (at robot loc2) is an unreachable h^2 atom
- Unreachable actions in P^2 are spurious!
  - Spurious actions are never applicable in progression, but can be (wrongly) used in regression, abstractions, heuristics...
  - Kind of obvious, but not highlighted/evaluated yet

Negated atoms in h^2

h^2 was originally described in STRIPS, atoms are propositions

- Negated propositions matter, though. See *Matching-Blocksworld:*



Mutex $\{(on\ a\ b), \neg(solid\ b)\}$ not found by h^2!

- Negated atoms must be explicitly represented, unless they belong to an "exactly-one" invariant group

Encoding extra information in actions

Disambiguate implicit preconditions and effects

→ find the value of some variables

→ Use mutexes in h^2 propagation

It may allow finding more mutexes and spurious actions!

Example: Throw-paint pre {}, eff {(painted loc4), (low-battery) }
If you know that (at-robot loc1) and (low-battery) are mutex then

1. ¬(at-robot loc1) is a precondition of throw-paint
2. and (painted loc4), (at-robot loc1) may be a mutex now

h^2 in regression

h^2 is a reachability analysis on P^2

- It can be done on a reversed version of P^2 too!!
    1. Disambiguate $S_\star$, assume unknown atoms are true
    2. Perform h^2 with reversed and disambiguated actions
- Already implemented by Petterson(2005) and Haslum(2008)

h^2 in regression

h^2 is a reachability analysis on P^2

- It can be done on a reversed version of P^2 too!!
  1. Disambiguate $S_\star$, assume unknown atoms are true
  2. Perform h^2 with reversed and disambiguated actions
- Already implemented by Petterson(2005) and Haslum(2008)

Reason for a more general definition of spurious state

- Doesn't always depend on s_0
- Other invariants are used to enrich h^2

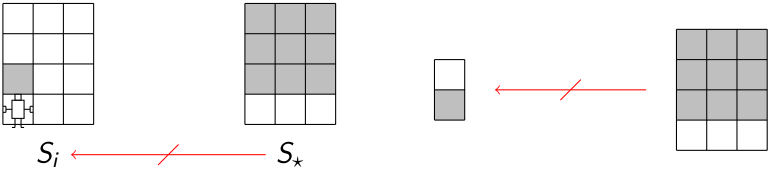
h^2 in regression: trucks with fuel

- $S_\star$ is *(at-truck goal)*
- The pairs *(at-truck goal)* $\wedge$ *(fuel n)* are assumed to be true

$$\text{(at-truck goal)} \wedge \text{(fuel n)} \xrightarrow{\textit{regression}} \text{(at-truck locx)} \wedge \text{(fuel n+1)}$$

- Unreachable pairs in regression are mutex: {*(at-truck distance2toGoal)*, *(fuel level1)*}
- If encountered forward, the state is a dead end
- move (locx locDistance2toGoal fuel2 fuel1) is spurious

h^2 in regression: Floortile



1. Disambiguate goal: robot in bottom row
2. Run bw-h2:
   - All the *paint-down* actions are discarded by bw-h^2 in Floortile!
   - S_i contains binary mutexes (painted tile1-2) $\wedge$ (not-painted tile1-3)

Our algorithm

1. Fw-h2 $\rightarrow$ find mutexes and spurious actions
2. Disambiguate actions and goal
3. Bw-h2 $\rightarrow$ find mutexes and spurious actions
4. If bw-h2 found something new: disambiguate and repeat fw-h2
5. If fw-h2 found something new: disambiguate and repeat bw-h2

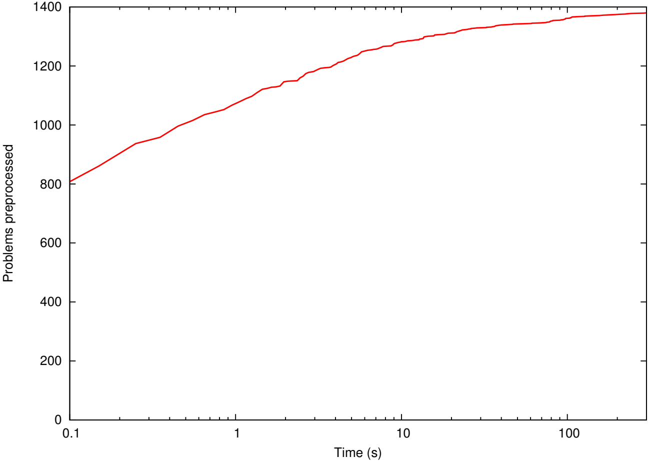
Return set of valid operators, fw-mutexes and bw-mutexes

State invariants in benchmark domains

- **Low overhead**: 300*s* threshold enough except in 3 domains
- **h2 fw-mutexes**: 33 out of 44 domains
- **h2 bw-mutexes**: 16 out of 44 domains
- **Multiple iterations** in 11 out of 44 domains

Domain	% Facts	% Ops	Domain	% Facts	% Ops
Tidybot	31	85	Scan-08	0	43
Airport	38	73	Pegsol-08	14	30
Parc-11	28	68	Floortile	18	38
Woodw-11	4	52	Nomystery	6	38
Trucks	5	46	Sokoban-11	22	24
TPP	12	45	Mystery	6	23

Time: (optimal benchmarks)



Coverage: Highlighted Domains

		Optimal					Satisficing			
		Blind			LM-cut		FD		LAMA	
Domain	#	-	h^2	h^2+mut	-	h^2	-	h^2	-	h^2
Airport	50	22	+5		28	+1	37	+2	35	+3
Floortile-11	20	2	+6	(+12)	7	+7	7	+13	6	+14
Parcprinter-11	20	6	+10		13	+4	3	+15	14	+6
Pipes-notank	50	17	0		17	0	44	-2	43	+1
Sokoban-08	30	22	+5	(+6)	30	0	28	0	29	0
Tidybot-11	20	12	0		14	+3	15	+2	16	+3
Woodwork-11	20	3	+1		12	+3	20	0	20	0
$\sum$	1396	533	+41	(+49)	747	+30	1138	+35	1296	+30

Conclusions

- Computing h^2 invariants is very helpful!
  - Both forward and backward
  - Simply remove operators inconsistent with invariants
  - Increases coverage for different optimal and satisficing planners
- Important implementation details
  - Disambiguation
  - Negated propositions
  - Spurious actions

Thanks for your attention

Questions?