A Formal Verification of Strong Stubborn Set Based Pruning

Travis Rivera Petit <travis.riverapetit@stud.unibas.ch>

Philosophisch-Naturwissenschaftlichen Fakultät, University of Basel

18.05.2020

Roadmap

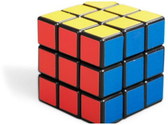
1. Classical Planning
2. Strong Stubborn Set based pruning
3. Isabelle/HOL Implementation
4. Contributions & Future work

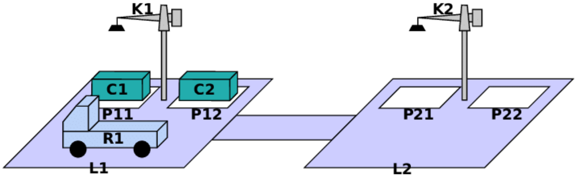
Roadmap

1. Classical Planning
2. Strong Stubborn Set based pruning
3. Isabelle/HOL Implementation
4. Contributions & Future work

Classical Planning

Branch of AI that studies single agent, static, deterministic, fully observable, discrete search problems.





Definition

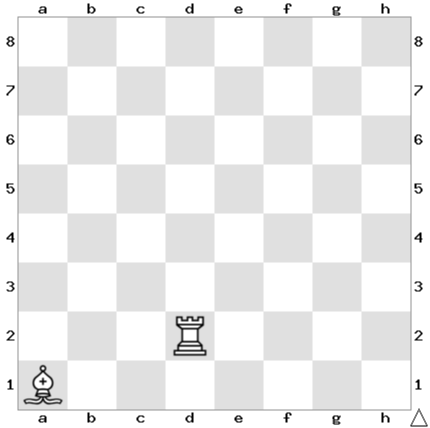
A transition system is a 6-tuple

$$\mathcal{T} = \langle S, T, A, cost, s_0, G\rangle$$

1. S is a set of states.
2. $T \subseteq S \times A \times S$ is a set of transitions $t = \langle src\, t, act\, t, dst\, t\rangle$.
3. A is a set of action.
4. $cost$ is a function $A \rightarrow \mathbb{N}_0$.
5. s_0 is the initial state.
6. $G \subseteq S$ is the set of goals.

op is an operator in $\mathcal{T}$ if $op \subseteq T \land \forall t, t' \in op : act\, t = act\, t'$.

An example



- $S = \{\text{positions}\}$
- $T = \{\langle \text{position, move, effect} \rangle\}$
- $A = \{\text{possible moves}\}$
- $cost \equiv 1$
- $op_i = \{\langle s, act_i, s' \rangle \in T\}$
- solution $=$ sequence of operators.

State spaces tend to be too vast!

One solution: Pruning

State spaces tend to be too vast!

One solution: Pruning

However ... pruning procedures are tricky to prove.

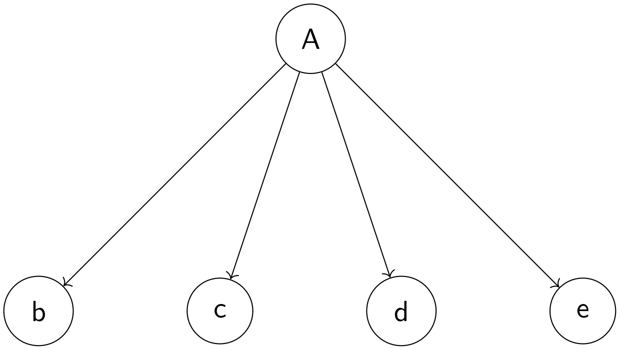
This thesis: Validate correctness of Strong Stubbron Set based pruning for transition systems in Isabelle/HOL.

Roadmap

1. Classical Planning
2. Strong Stubborn Set based pruning
3. Isabelle/HOL Implementation
4. Contributions & Future work

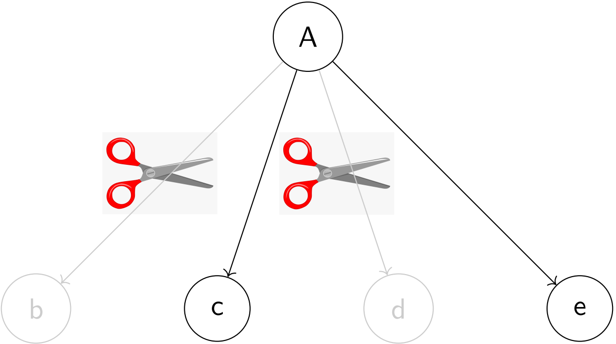
Pruning

State space pruning is a domain-independent technique that narrows down the set of applicable operators into an optimality preserving set.



Pruning

State space pruning is a domain-independent technique that narrows down the set of applicable operators into an optimality preserving set.



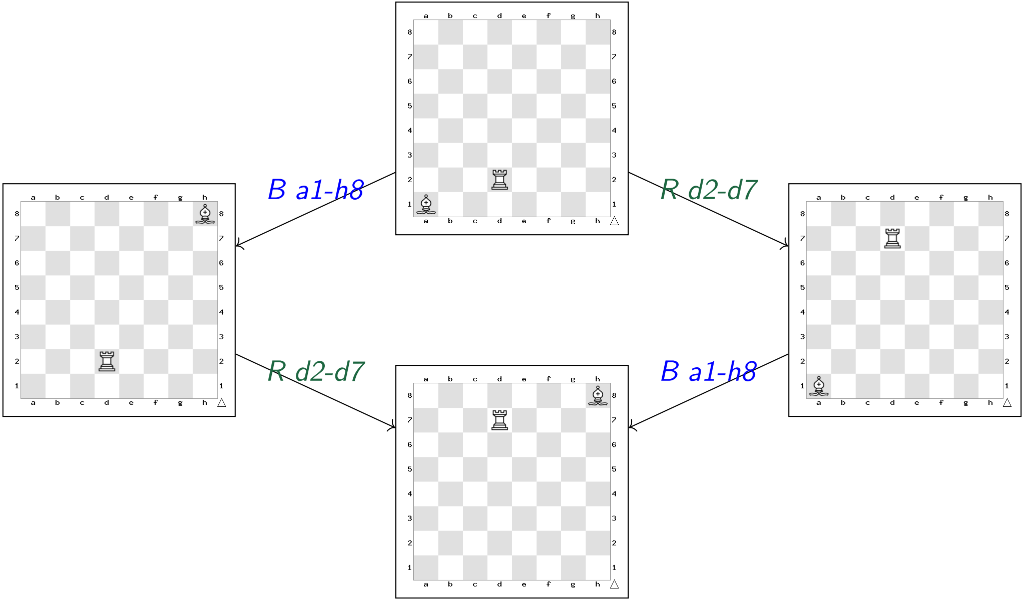
Strong Stubborn Sets

First introduced in the area of model checking.

Then adopted to classical planning in SAS^+.

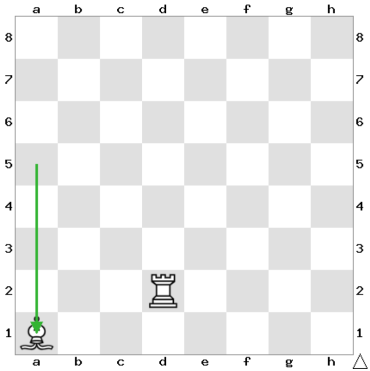
Here: to transition systems.

Idea: exploit properties about independent operators.


B a1-h8
R d2-d7
R d2-d7
B a1-h8

Necessary enabling set

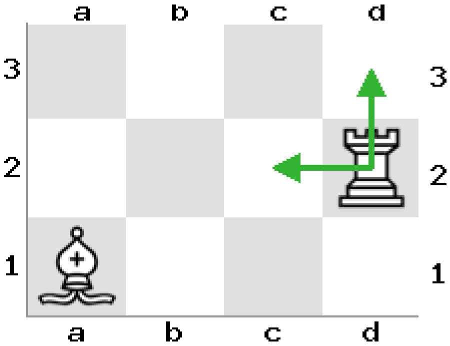
N is a necessary enabling set for op in s if
$\forall$ solution π for s that contains op: $\exists\ op' \in \texttt{set}(\pi) \cap N$ that comes before op in π.



$$N = \{B\text{-}a1\text{-}b2,\ B\text{-}a1\text{-}c3, \ldots, B\text{-}a1\text{-}h8\}$$

Disjunctive action landmark

A disjunctive action landmark L for a state $s \in S$ is a set of operators such that for every solution for s, there exists an operator in that path that is also in L.



Definition

A Strong Stubborn Set SSS for $s \in S$ if the following hold:

- SSS contains a disjunctive action landmark for s.
- if $op \in SSS$ and $\neg app(op, s)$ then SSS contains a necessary enabling set for op in s.
- if $op \in SSS$ and $app(op, s)$ then SSS contains all the operators op' for which op and op' are dependent.

Theorem

Let $s \in S$ be an active state and SSS be a Strong Stubborn Set for s. Then there exists an $op \in SSS$ that starts some optimal solution for s.

Proof sketch:

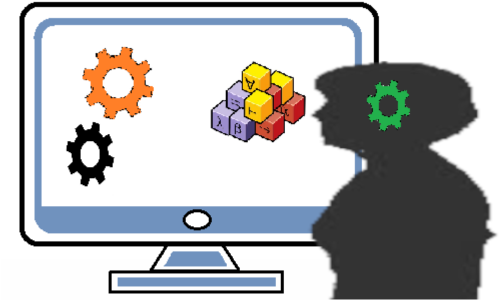
- s active so $\exists$ solution $\pi = \langle op_1, \ldots, op_n \rangle$.
- SSS contains a disjunctive landmark $\implies$ $\mathtt{set}(\pi) \cap SSS \neq \emptyset$.
- Let then op in π s.t. it has the lowest inedex in π and $op \in SSS$.
- $\neg app(op, s) \implies SSS$ contains a necessary enabling set for op in $s \implies \exists\ op'$ comes before op and $op' \in SSS \cap \mathtt{set}(\pi) \cap SSS$ $\lightning$.
- $\exists op'$ in π : op' comes before op and op and op' are dependent $\implies op' \in SSS$ $\lightning$.
- Thus moving op to the front of π is also an optimal solution.

Roadmap

1. Classical Planning
2. Strong Stubborn Set based pruning
3. Isabelle/HOL Implementation
4. Contributions & Future work

Isabelle

Isabelle is an interactive theorem prover.
Proofs are well defined $\implies$ proof search is suited for automation.
Isabelle/HOL provides a higher-order logic theorem proving environment.



Implementation

- Bottom up approach.
- 72 lemmas proven before tackling the main theorem.

```
lemma
  fixes L :: "'a operator set" and pi :: "'a path"
  shows "L ∩ set pi ≠ {} ⟹ pi ≠ [] ⟹ first_occurrence pi L ∈ L ∩ set pi"
proof (induct pi)
  case Nil
  from this show ?case by auto
next
  case (Cons x xs)

  from this have ONE: "(x ∉ L ∧ xs = []) ⟹ ?case" by simp
  from local.Cons have TWO: "(x ∉ L ∧ xs ≠ []) ⟹ ?case" by simp
  from local.Cons have THREE: "x ∈ L ⟹ ?case" by simp

  from ONE and TWO and THREE show ?case by auto
qed
```

The Isabelle/HOL proof

Roadmap

1. Classical Planning
2. Strong Stubborn Set based pruning
3. Isabelle/HOL Implementation
4. Contributions & Future work

Contributions

1. Validate an important theorem about the optimality preserving property of Strong Stubborn Set based pruning.
2. Adapt the theory of Strong Stubborn Sets to transition systems.
3. Provide an Isabelle/HOL base code for future proofs.

Future Work

1. Validate correctness of Strong Stubborn Set finding algorithms.
2. Validate the the correctness of the optimality preserving property in SAS^{+}

Contributions

1. Validate an important theorem about the optimality preserving property of Strong Stubborn Set based pruning.
2. Adapt the theory of Strong Stubborn Sets to transition systems.
3. Provide an Isabelle/HOL base code for future proofs.

"Who fails to plan, plans to fail"

proverb