About Partial Order Reduction in Planning and Computer Aided Verification

Martin Wehrle and Malte Helmert

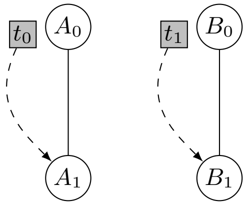
June 27, 2012

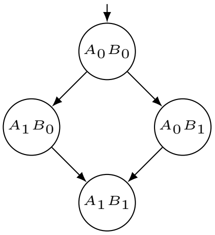
Partial order reduction

Partial order reduction (POR)

- Originally proposed for computer aided verification (CAV)
- Pruning technique to tackle state explosion problem
- Avoids redundant application orders of independent operators

Example





Partial order reduction

Currently

- POR techniques recently (re-)considered in planning
- Various existing POR techniques in CAV and planning
- Formal relationships of these techniques mostly unclear

In this paper

- Theoretical analysis of relationships of POR-based techniques
- Comparison of techniques from CAV and planning
- Investigation of *transition* and *state* reduction techniques

In this talk

- Focus on transition reduction techniques
- Outline about state reduction techniques at the end

Preliminaries

Terminology

- $\mathcal{V}$ finite set of multi-valued variables v with domain $\mathrm{dom}(v)$
- (Partial) state = (partial) function $s : \mathcal{V} \to \mathrm{dom}(\mathcal{V})$
- Operators of the form $o = \langle pre, eff \rangle$
- Operator o applicable in state s iff $s \models pre$
- Successor state obtained by setting effect variables

Planning instance

A SAS^+ planning instance is a 4-tuple $(\mathcal{V}, \mathcal{O}, s_0, s_\star)$, where

- $\mathcal{V}$ is a finite set of multi-values variables,
- $\mathcal{O}$ is a finite set of operators,
- s_0 is the initial state,
- $s_\star$ the partial goal state.

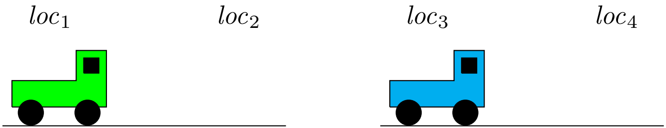
Preliminaries

Commutative operators

Operators o and o' are commutative if

- $prevars(o) \cap effvars(o') = \emptyset$, and
- $effvars(o) \cap prevars(o') = \emptyset$, and
- all $v \in effvars(o) \cap effvars(o')$ are set to the same value.

Example



$drive(loc_1, loc_2)$ and $drive(loc_3, loc_4)$ are commutative

Transition reduction techniques

Transition reduction techniques

Transition reduction techniques

- Reduce the number of applied transitions
- Guaranteed to preserve permutation of pruned paths
- Pruning decisions are path-dependent
- Not directly applicable to graph search algorithms like A^*
- Useful for tree search algorithms like IDA^*

Notation

- Path $=$ sequence of operators σ that starts in s_0
- We apply state terminology to paths
- Example: "o applicable in σ" if o applicable after applying σ

Transition reduction techniques

Sleep sets (Godefroid, 1996)

- Every path σ has a corresponding sleep set (possibly empty)
- Sleep set = set of operators which are pruned in σ

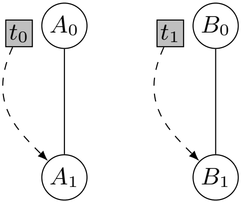
Computation

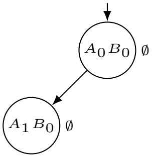
1. Search begins with empty sleep set: $\mathit{sleep}(\varepsilon) := \emptyset$
2. Sleep sets for successor paths of path σ: Consider operators $o_1, \ldots, o_n$ that are applied in σ in this order.

   $$\mathit{sleep}(\sigma o_i) := (\mathit{sleep}(\sigma) \cup \{o_1, \ldots, o_{i-1}\}) \setminus \{o \mid o, o_i \text{ not commutative}\}$$

Transition reduction techniques

Example

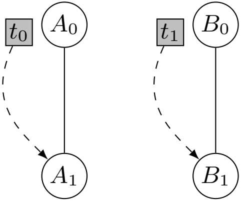


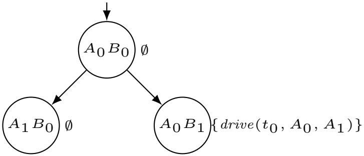


- $sleep(\varepsilon; drive(t_0, A_0, A_1)) = \emptyset \cup \emptyset \setminus \emptyset = \emptyset$

Transition reduction techniques

Example

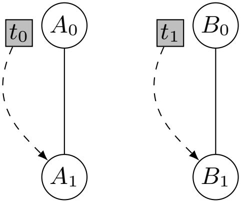


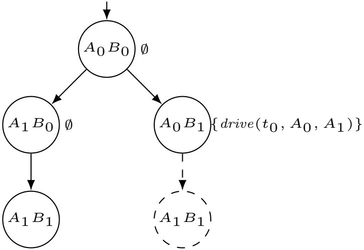


- $sleep(\varepsilon; drive(t_0, A_0, A_1)) = \emptyset$
- $sleep(\varepsilon; drive(t_1, B_0, B_1)) = \{drive(t_0, A_0, A_1)\}$

Transition reduction techniques

Example





- $sleep(\varepsilon; drive(t_0, A_0, A_1)) = \emptyset$
- $sleep(\varepsilon; drive(t_1, B_0, B_1)) = \{drive(t_0, A_0, A_1)\}$
- Path $\varepsilon; drive(t_1, B_0, B_1); drive(t_0, A_0, A_1)$ is not generated

Transition reduction techniques

Commutativity pruning (Haslum & Geffner, 2000)

- Impose (arbitrary) total order $<$ on operators
- Successor path $\sigma o o'$ of σo is not generated (pruned) if
  1. o and o' are commutative, and
  2. $o' < o$
- "Prune paths with commutative operators in wrong order"

Transition reduction techniques

Proposition

Under the same total order $<$ on the operators, every path pruned by commutativity pruning is also pruned by sleep sets.

Proof

Consider path $\sigma o o'$ pruned by commutativity pruning (CP).

1. $\sigma o o'$ pruned by CP, therefore $o' < o$ and o, o' commutative
2. o and o' commutative, therefore o' applicable in σ
3. Therefore, $o' \in \{\hat{o} \mid \hat{o} < o$ and $\hat{o}$ applicable in $\sigma\} =: A$
4. Moreover, $o' \notin \{\hat{o} \mid \hat{o}$ and o not commutative$\} =: NC$
5. Same order $<$ for both: $sleep(\sigma o) = (sleep(\sigma) \cup A) \setminus NC$
6. Hence, $o' \in sleep(\sigma o)$

Transition reduction techniques

Proposition

There exist paths pruned by sleep sets and not pruned by CP.

Why?

- Intuitively: sleep sets store more information than CP
- Concrete example given in the paper

Transition reduction techniques

Stratified planning (Chen et al., 2009)

- SCCs $C_1 < \cdots < C_n$ of causal graph in topological ordering
- Ordering $<$ such that edges from C_i to C_j only if $i \leq j$
- $level(v) := i$ iff $v \in C_i$
- $level(o) := i$ iff ex. effect variable v in o with $level(v) = i$

Pruning algorithm

Prune path $\sigma o o'$ if

1. $level(o') > level(o)$, and
2. o' does not read a variable that is written by o, and
3. o' and o do not write a common variable.

Transition reduction techniques

Let $<_c$ be an ordering such that $o <_c o'$ if $level(o) > level(o')$.

Proposition

Every path pruned by stratified planning is also pruned by commutativity pruning with $<_c$.

Proof sketch

Consider the path $\sigma o o'$ pruned by stratified planning (SP).

- In this case, o and o' are commutative
- By definition: $level(o') > level(o)$ implies $o' <_c o$.
- Therefore, $\sigma o o'$ is pruned by commutativity pruning.

Transition reduction techniques

Proposition

There exist paths pruned by commutativity pruning and not by SP.

Example

- Variables build single SCC in causal graph
- No pruning by SP because all operators have equal level
- Commutative operators can still be pruned by CP
- Concrete example given in the paper

Transition reduction techniques

Transition reduction techniques: Results summary

1. Sleep sets strictly dominate commutativity pruning
2. Commutativity pruning strictly dominates stratified planning

What about state reduction techniques?

Further results: State reduction

Further results

State reduction techniques

- Reduce the size of explored state space
- State-dependent (not path-dependent)
- Applicable to graph search algorithms like A^*

Results summary

- Corrected expansion core method (Chen & Yao, 2009) is special case of strong stubborn sets (Valmari, 1991)
- Therefore: Pruning power of expansion core is theoretically at most as high as pruning power of strong stubborn sets
- What about the practice?
  $\rightsquigarrow$ A Stubborn Set Algorithm for Optimal Planning
  (Alkhazraji, Wehrle, Mattmüller, Helmert; ECAI 2012)

Conclusions

Summary

- POR techniques from CAV and planning are strongly related
- CAV techniques generalize investigated planning techniques

Ongoing and future work

- Impact of design choices to compute strong stubborn sets?
- Investigation of weak stubborn sets (Valmari, 1991)