AxSAT – Bringing Axioms to SAT Planning

Gregor Behnke[1] **David Speck**[2] **Daniel Gnad**[3,4]

[1]ILLC, University of Amsterdam, The Netherlands
[2]University of Basel, Switzerland
[3]Heidelberg University, Germany
[4]Linköping University, Sweden

Setting

Classical Planning – Sequential Decision Making

Setting

Classical Planning – Sequential Decision Making

Planning as Propositional Satisfiability

Setting

Classical Planning – Sequential Decision Making

Planning as Propositional Satisfiability

How can we adapt + parallelize SAT encodings for Planning with Axioms?

Classical Planning

Definition. *A planning task is a tuple $\Pi = \langle V, A, I, G \rangle$ where:*

- *V is a set of state variables, each $v \in V$ with a finite domain D_v.*
- *A is a set of actions; each $a \in A$ is a pair $\langle pre(a), eff(a) \rangle$, of a's precondition and effect (partial assignments).*
- *Initial state I (complete assignment), goal G (partial assignment).*

$\rightarrow$ Solution ("Plan"): Action sequence mapping I into s s.t. $s \models G$.

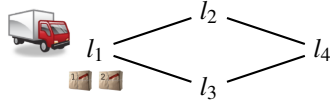
Classical Planning

Definition. *A planning task is a tuple $\Pi = \langle V, A, I, G \rangle$ where:*

- *V is a set of state variables, each $v \in V$ with a finite domain D_v.*
- *A is a set of actions; each $a \in A$ is a pair $\langle pre(a), eff(a) \rangle$, of a's precondition and effect (partial assignments).*
- *Initial state I (complete assignment), goal G (partial assignment).*

$\rightarrow$ Solution ("Plan"): Action sequence mapping I into s s.t. $s \models G$.

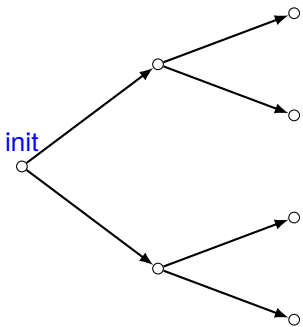
Running Example:



- $V = \{t, p_1, p_2\}$
  with $D_t = \{l_1, l_2, l_3, l_4\}$ and $D_{p_i} = \{t, l_1, l_2, l_3, l_4\}$.
- $A = \{load(p_i, x), unload(p_i, x), drive(x, x')\}$
- $I = \{(t, l_1), (p_1, l_1), (p_2, l_1)\}$. $G = \{(p_1, l_3), (p_2, l_3)\}$.

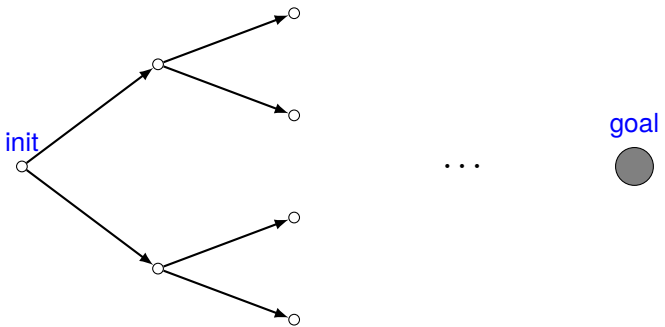
State Space of a Planning Task

State Space of a Planning Task

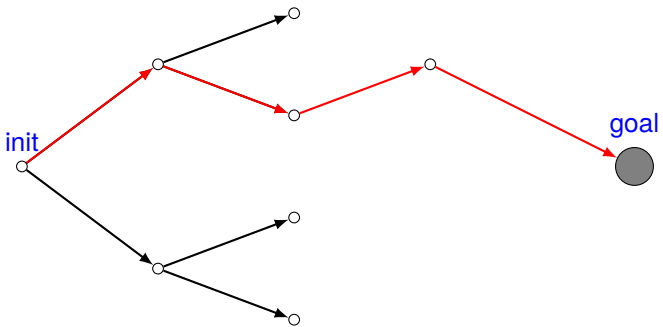


goal

State Space of a Planning Task



State Space of a Planning Task



Planning as SAT

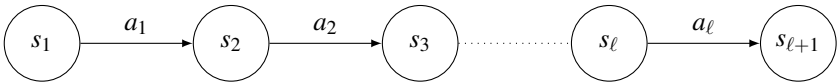
General encoding for classical planning without axioms

Input: planning task with actions $a_i \in A$, state atoms $(v,d) : v \in \mathcal{V}, d \in D_v$.
Bound ℓ on plan length.

Planning as SAT

General encoding for classical planning without axioms

Input: planning task with actions $a_i \in A$, state atoms $(v,d) : v \in \mathcal{V}, d \in D_v$.
Bound ℓ on plan length.

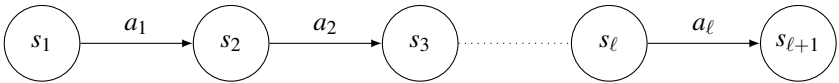


- "Time-stamped" plan skeleton with states s_i (variable assignments) and actions a_i.
- $\mathcal{I} = s_1$ and $\mathcal{G} \subseteq s_{\ell+1}$.
- Clauses that enforce correct semantics of action application: preconditions and effects.

Planning as SAT

General encoding for classical planning without axioms

Input: planning task with actions $a_i \in A$, state atoms $(v,d) : v \in \mathcal{V}, d \in D_v$.
Bound ℓ on plan length.



- "Time-stamped" plan skeleton with states s_i (variable assignments) and actions a_i.
- $\mathcal{I} = s_1$ and $\mathcal{G} \subseteq s_{\ell+1}$.
- Clauses that enforce correct semantics of action application: preconditions and effects.
- Solutions to the SAT formula are in 1-to-1 correspondence with solutions (of length $\leq \ell$) for the planning task.

Parallel Action Encodings

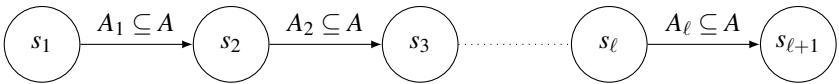
The $\exists$-step semantics

Input: planning task with actions $a_i \in A$, state atoms $(v, d) : v \in \mathcal{V}, d \in D_v$.
Bound ℓ on plan length.

Parallel Action Encodings

The $\exists$-step semantics

Input: planning task with actions $a_i \in A$, state atoms $(v,d) : v \in \mathcal{V}, d \in D_v$.
Bound ℓ on plan length.



- Allow sets of actions per time step. Impose total order over A.
- Pairs of actions $a_1 \prec a_2$ s.t. a_1 does not disable a_2.
- $\Longrightarrow$ There exists an ordering $\langle a_1, \ldots, a_n \rangle$ of actions in A_i applicable in s_i.

Classical Planning with Axioms

Definition. *A planning task is a tuple* $\Pi = \langle V, \mathcal{D}, A, Ax, I, G \rangle$ *where:*

- $\ldots$
- $\mathcal{D}$ *is a set of derived variables* $d \in \mathcal{D}$ *with binary domain (default "false").*
- Ax *is a set of axioms of the form:* $d \leftarrow \bigwedge_i d_i \wedge \bigwedge_j \neg d_j \wedge \bigwedge_k (v_k = o_k)$

Classical Planning with Axioms

Definition. *A planning task is a tuple* $\Pi = \langle V, \mathcal{D}, A, Ax, I, G \rangle$ *where:*

- *. . . (derived variables can be part of action preconditions and goal)*
- $\mathcal{D}$ *is a set of derived variables* $d \in \mathcal{D}$ *with binary domain (default "false").*
- Ax *is a set of axioms of the form:* $d \leftarrow \bigwedge_i d_i \wedge \bigwedge_j \neg d_j \wedge \bigwedge_k (v_k = o_k)$

Classical Planning with Axioms

Definition. *A planning task is a tuple* $\Pi = \langle V, \mathcal{D}, A, Ax, I, G \rangle$ *where:*

- *. . . (derived variables can be part of action preconditions and goal)*
- $\mathcal{D}$ *is a set of derived variables* $d \in \mathcal{D}$ *with binary domain (default "false").*
- Ax *is a set of axioms of the form:* $d \leftarrow \bigwedge_i d_i \wedge \bigwedge_j \neg d_j \wedge \bigwedge_k (v_k = o_k)$

Fixpoint over axioms computed for every state, only stratified axiom programs with unique fixpoint.

Classical Planning with Axioms

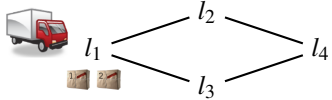
Definition. *A planning task is a tuple* $\Pi = \langle V, \mathcal{D}, A, Ax, I, G \rangle$ *where:*

- *. . . (derived variables can be part of action preconditions and goal)*
- $\mathcal{D}$ *is a set of derived variables* $d \in \mathcal{D}$ *with binary domain (default "false").*
- Ax *is a set of axioms of the form:* $d \leftarrow \bigwedge_i d_i \wedge \bigwedge_j \neg d_j \wedge \bigwedge_k (v_k = o_k)$

Fixpoint over axioms computed for every state, only stratified axiom programs with unique fixpoint.

Running Example:

- Encode truck drives using axioms: $d_{t=l_1} \leftarrow (t = l_1)$



Classical Planning with Axioms

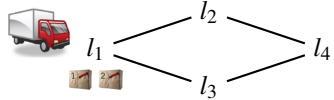
Definition. *A planning task is a tuple* $\Pi = \langle V, \mathcal{D}, A, Ax, I, G \rangle$ *where:*

- *...(derived variables can be part of action preconditions and goal)*
- $\mathcal{D}$ *is a set of derived variables* $d \in \mathcal{D}$ *with binary domain (default "false").*
- Ax *is a set of axioms of the form:* $d \leftarrow \bigwedge_i d_i \wedge \bigwedge_j \neg d_j \wedge \bigwedge_k (v_k = o_k)$

Fixpoint over axioms computed for every state, only stratified axiom programs with unique fixpoint.

Running Example:

- Encode truck drives using axioms: $d_{t=l_1} \leftarrow (t = l_1)$
  $d_{t=l_2} \leftarrow d_{t=l_1}$; $d_{t=l_3} \leftarrow d_{t=l_1}$; ...



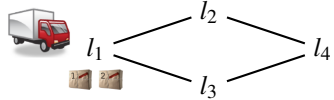
Classical Planning with Axioms

Definition. *A planning task is a tuple* $\Pi = \langle V, \mathcal{D}, A, Ax, I, G \rangle$ *where:*

- *...(derived variables can be part of action preconditions and goal)*
- $\mathcal{D}$ *is a set of derived variables* $d \in \mathcal{D}$ *with binary domain (default "false").*
- Ax *is a set of axioms of the form:* $d \leftarrow \bigwedge_i d_i \wedge \bigwedge_j \neg d_j \wedge \bigwedge_k (v_k = o_k)$

Fixpoint over axioms computed for every state, only stratified axiom programs with unique fixpoint.

Running Example:



- Encode truck drives using axioms: $d_{t=l_1} \leftarrow (t = l_1)$
  $d_{t=l_2} \leftarrow d_{t=l_1}$; $d_{t=l_3} \leftarrow d_{t=l_1}$; ...
- Truck empty as axiom: $d_{\text{t-empty}} \leftarrow \neg d_{p_1=t} \wedge \neg d_{p_2=t}$

SAT Planning with Axioms

How to include axioms in the SAT formula?

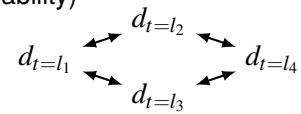
The Truth Dependency Graph (TDG): nodes $\mathcal{D}$, edge $d_1 \rightarrow d_2$ iff $\exists d_2 \leftarrow d_1 / \neg d_1$

SAT Planning with Axioms

How to include axioms in the SAT formula?

The Truth Dependency Graph (TDG): nodes $\mathcal{D}$, edge $d_1 \to d_2$ iff $\exists d_2 \leftarrow d_1 / \neg d_1$

Running example: (truck reachability)

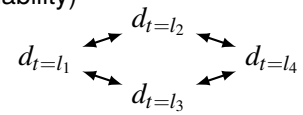


SAT Planning with Axioms

How to include axioms in the SAT formula?

The Truth Dependency Graph (TDG): nodes $\mathcal{D}$, edge $d_1 \to d_2$ iff $\exists d_2 \leftarrow d_1 / \neg d_1$

Running example: (truck reachability)



1. Identify strongly connected components (SCCs) in TDG + sort topologically.
2. For every SCC do:
   a. Layer axioms in SCC to break self-supporting cycles + deal with negation.
   b. Introduce SAT variables for every derived variable per layer per state.
   c. Include these variables in action preconditions and goal.

SAT Planning with Axioms

Action Parallelism

Without axioms:

Action a_1 disables a_2 in state if both applicable, but $\langle a_1, a_2 \rangle$ not executable.

The Disabling Graph: nodes A, edge $a_1 \rightarrow a_2$ if applicable in same state + a_1 disables a_2

SAT Planning with Axioms

Action Parallelism

Without axioms:

Action a_1 disables a_2 in state if both applicable, but $\langle a_1, a_2 \rangle$ not executable.

The Disabling Graph: nodes A, edge $a_1 \rightarrow a_2$ if applicable in same state + a_1 disables a_2

Sort actions within SCC of disabling graph, disabled actions first.
Forbid disabled actions in same step.

SAT Planning with Axioms

Action Parallelism

Without axioms:

Action a_1 disables a_2 in state if both applicable, but $\langle a_1, a_2 \rangle$ not executable.

The Disabling Graph: nodes A, edge $a_1 \rightarrow a_2$ if applicable in same state + a_1 disables a_2

Sort actions within SCC of disabling graph, disabled actions first.
Forbid disabled actions in same step.

With axioms:

- Extend disabling relation by derived variables changing as indirect consequence of applying an action a.
- **NP**-hard, so we over-approximate set of derived variables possibly changed by a.

Optimizations

Special topologies in Truth Dependency Graph

- Implicational SCC: $d_{ex} \rightarrow d_1 \rightarrow \cdots \rightarrow d_n$:
  $\Longrightarrow$ add axioms $d_1 \leftarrow d_{ex}, \ldots, d_n \leftarrow d_{ex}$

Optimizations

Special topologies in Truth Dependency Graph

- Implicational SCC: $d_{ex} \rightarrow d_1 \rightarrow \cdots \rightarrow d_n$:
  $\Longrightarrow$ add axioms $d_1 \leftarrow d_{ex}, \ldots, d_n \leftarrow d_{ex}$
- One-variable-dependent SCC: additionally allow one variable condition
  $d_1 \leftarrow d_{ex} \wedge (v, d)$.

Optimizations

Special topologies in Truth Dependency Graph

- Implicational SCC: $d_{ex} \rightarrow d_1 \rightarrow \cdots \rightarrow d_n$:
  $\Longrightarrow$ add axioms $d_1 \leftarrow d_{ex}, \ldots, d_n \leftarrow d_{ex}$
- One-variable-dependent SCC: additionally allow one variable condition $d_1 \leftarrow d_{ex} \wedge (v, d)$.

Dealing with dense Disabling Graphs

- Identify actions $\mathcal{L} \subseteq A$ with high connectivity.
- Do not allow parallelism for actions $\mathcal{L}$ and sort them last in every step.
- Avoids constructing large disabling graphs and high memory overhead.

Experiments – Coverage

Number of solved instances within 30min and 3.5GB

		Coverage							Action Parallelism	
				Iterative		Parallel			Iterative	Parallel
Domain	#	blind	SymK	Seq	∃	Seq	∃	LAMA	∃	∃
airport-adl	50	19	19	11	21	28	**49**	40	2.24	1.98
assembly	30	0	11	2	16	17	**30**	**30**	4.7	4.34
appn-adl	33	11	23	16	**33**	**33**	**33**	**33**	8.41	8.08
cats-horndl	20	**20**	**20**	11	**20**	**20**	**20**	**20**	13.1	2.62
fridge	30	0	1	0	**30**	21	**30**	**30**	26.6	26.6
miconic-axioms	150	60	**150**	35	**150**	**150**	**150**	**150**	15.5	6.2
optical-telegraphs	48	2	3	1	**48**	6	**48**	4	32.45	25.5
philosophers	48	5	12	3	**48**	12	**48**	46	25.5	22.95
psr-large	50	14	25	22	25	25	28	**44**	5.84	3.27
psr-middle	36	26	**36**	35	**36**	**36**	**36**	**36**	6.43	3.88
queens-horndl	30	23	10	**30**	**30**	**30**	**30**	**30**	3.93	2.07
taskassign-horndl	20	**20**	12	**20**	**20**	**20**	**20**	**20**	11.95	5.51
trucks	30	8	9	5	19	13	**25**	15	2.7	2.44
Σ	1269	650	806	583	908	975	**1123**	1111	5.06	3.92

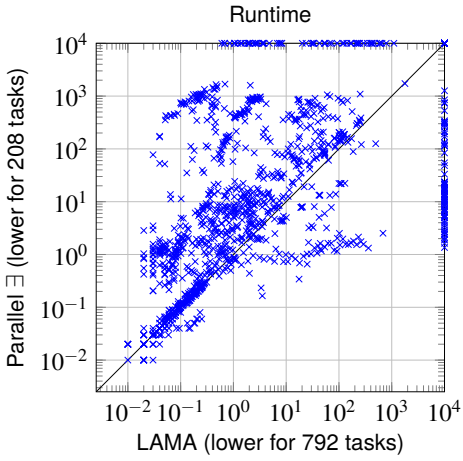
Experiments – Coverage

Number of solved instances within 30min and 3.5GB

		Coverage							Action Parallelism	
				Iterative		Parallel			Iterative	Parallel
Domain	#	blind	SymK	Seq	∃	Seq	∃	LAMA	∃	∃
openstacks	30	7	16	5	7	20	22	**30**	1.66	1.35
openstacks-sat08-adl	25	4	12	2	3	12	14	**25**	1.37	1.33
snowman-basic	41	**30**	29	13	12	16	17	22	1.06	0.97
snowman-reach	41	**27**	25	19	17	20	18	23	1.07	0.9
sokoban-axioms	30	24	25	12	12	13	13	**28**	1	0.81
tpsa-horndl	15	4	6	10	9	12	12	**15**	1.17	1
Σ	1269	650	806	583	908	975	**1123**	1111	5.06	3.92

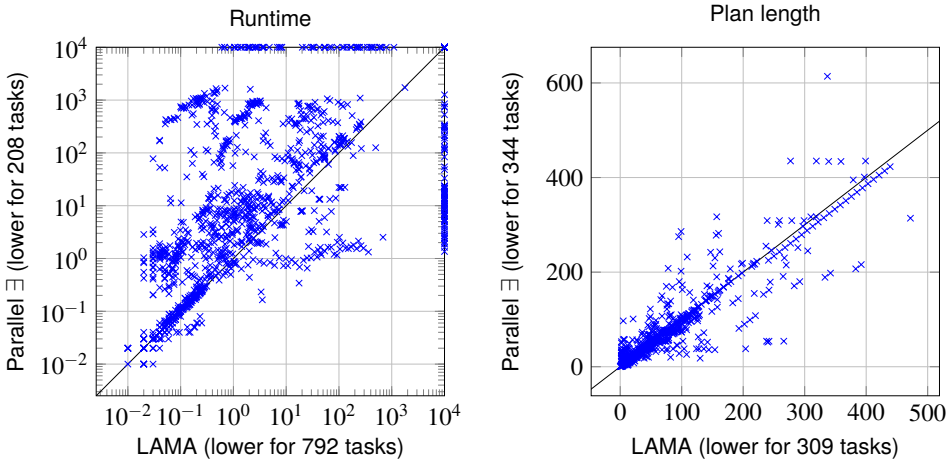
Experiments

Per-instance comparison



Experiments

Per-instance comparison



Conclusions

Summary

- Planning as SAT is a successful approach for classical planning.
- Introduced support for axioms, including action parallelism.
- State-of-the-art performance.

Conclusions

Summary

- Planning as SAT is a successful approach for classical planning.
- Introduced support for axioms, including action parallelism.
- State-of-the-art performance.

Future Work

- Targeted encoding for orthogonal planning techniques (decoupled search).
- Support for stronger forms of parallelism (relaxed $\exists$, ...).
- Evaluate derived variables in SAT, not approximate.