Improving Efficiency of QBF Planning with Mixed Linear - Compact Tree Encodings

Améliorer l'efficacité de la planification QBF
avec des nouveaux codages d'arbres compacts

Frédéric Maris (IRIT, Université de Toulouse, France)

12èmes Journées MAFTEC, Grenoble, 26–28 mars 2025

STRIPS Planning Problems

A STRIPS **planning problem** is a tuple $\langle \mathcal{F}, I, \mathcal{A}, G \rangle$ where

- $\mathcal{F}$ is a finite set of **fluents** (atomic propositions),
- $I \subseteq \mathcal{F}$ is the set of initial fluents (initial state),
- $G \subseteq \mathcal{F}$ is the set of goal fluents,
- $\mathcal{A}$ is the set of actions.

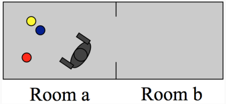
An **action** $a \in \mathcal{A}$ is a tuple $\langle Cond(a), Add(a), Del(a) \rangle$ where

- $Cond(a) \subseteq \mathcal{F}$ is the set of fluents required in order to execute a,
- $Add(a) \subseteq \mathcal{F}$ is the set of fluents added by the action a,
- $Del(a) \subseteq \mathcal{F}$ is the set of fluents removed by the action a.

Example : Gripper-3 classical planning problem

$I = \{at(ball_R, room_a), at(ball_B, room_a), at(ball_Y, room_a), at\text{-}robby(room_a),$
$free(left), free(right)\}$

$G = \{at(ball_R, room_b), at(ball_B, room_b), at(ball_Y, room_b)\}$



Action a : $PICK(ball_R, room_a, left)$

- $Cond(a) = \{at(ball_R, room_a), at\text{-}robby(room_a), free(left)\}$
- $Add(a) = \{carry(ball_R, left)\}$
- $Del(a) = \{at(ball_R, room_a), free(left)\}$

SAT Encodings vs QBF Encodings

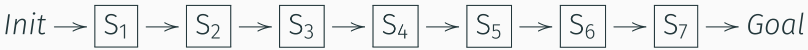


Figure 1 : Transitions of an 8-steps plan in SAT encoding

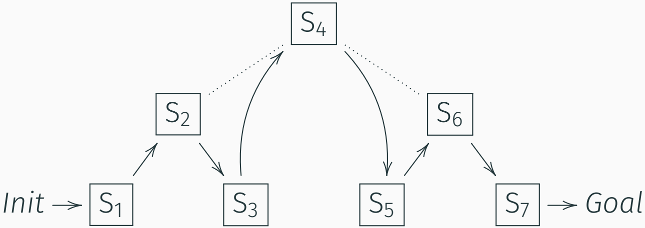


Figure 2 : Transitions of an 8-steps plan in QBF compact tree encoding (CTE)

QBF Compact Tree Encoding with Node Width

Parameters of the CTE :

- d is the fixed **depth** of the tree,
- w is the fixed **node width**.

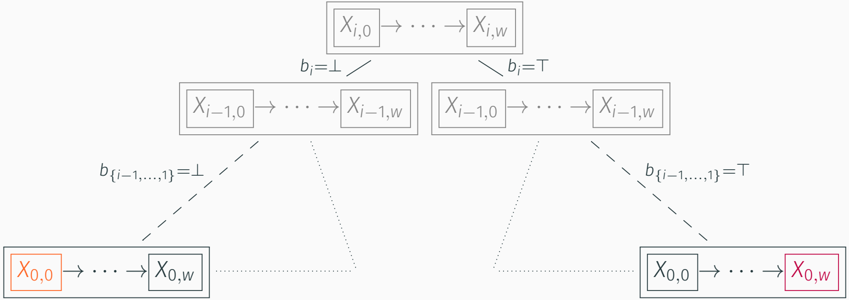
For each $i \in \{1...d\}$ and $l \in \{1...w\}$, we define

- a propositional variable b_i,
- a set of propositional variables $X_{i,l} = \mathcal{A}_{i,l} \cup \mathcal{F}_{i,l}$
  - $\mathcal{A}_{i,l} = \{a_{i,l} \,:\, a \in \mathcal{A}\}$,
  - $\mathcal{F}_{i,l} = \{f_{i,l} \,:\, f \in \mathcal{F}\}$.

Quantifiers (CTE Branching Structure)

$$\exists_{\substack{a\in\mathcal{A}\\ l\in\{0...w\}}} a_{d,l}. \exists_{\substack{f\in\mathcal{F}\\ l\in\{0...w\}}} f_{d,l}. \forall b_d. \exists_{\substack{a\in\mathcal{A}\\ l\in\{0...w\}}} a_{d-1,l}. \exists_{\substack{f\in\mathcal{F}\\ l\in\{0...w\}}} f_{d-1,l}. \forall b_{d-1}. \dots \exists_{\substack{a\in\mathcal{A}\\ l\in\{0...w\}}} a_{1,l}. \exists_{\substack{f\in\mathcal{F}\\ l\in\{0...w\}}} f_{1,l}. \forall b_1. \exists_{\substack{a\in\mathcal{A}\\ l\in\{0...w\}}} a_{0,l}. \exists_{\substack{f\in\mathcal{F}\\ l\in\{0...w\}}} f_{0,l}.$$

Encoding Initial State and Goal



$$\left[\left(\bigwedge_{i=1}^{d} \neg b_i\right) \rightarrow \left(\bigwedge_{\substack{a\in\mathcal{A}\\ Cond(a)\not\subseteq I}} \neg a_{0,0}\right)\right] \wedge \left[\left(\bigwedge_{i=1}^{d} b_i\right) \rightarrow \left(\bigwedge_{f\in G} f_{0,w}\right)\right]$$

Encoding Transitions (Conditions and Effects)

If an action a is executed in a transition of the plan, then each effect of a occurs in resulting state and each condition of a is required in previous state.

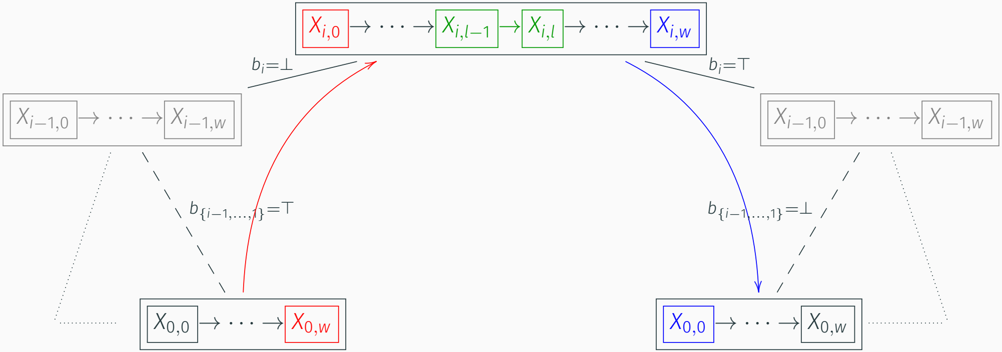
$$\bigwedge_{i=0}^{d} \bigwedge_{l=0}^{w} \bigwedge_{a \in \mathcal{A}} \left(a_{i,l} \rightarrow \left(\bigwedge_{f \in Add(a)} f_{i,l} \right) \wedge \left(\bigwedge_{f \in Del(a)} \neg f_{i,l} \right) \right)$$

$$\bigwedge_{i=1}^{d} \left[\left(\neg b_i \wedge \bigwedge_{j=1}^{i-1} b_j \right) \rightarrow \bigwedge_{a \in \mathcal{A}} \left(a_{i,0} \rightarrow \bigwedge_{f \in Cond(\mathcal{A})} f_{0,w} \right) \right]$$ (leaf to node)

$$\bigwedge_{i=0}^{d} \bigwedge_{l=1}^{w} \bigwedge_{a \in \mathcal{A}} \left(a_{i,l} \rightarrow \bigwedge_{f \in Cond(\mathcal{A})} f_{i,l-1} \right)$$ (within node)

$$\bigwedge_{i=1}^{d} \left[\left(b_i \wedge \bigwedge_{j=1}^{i-1} \neg b_j \right) \rightarrow \bigwedge_{a \in \mathcal{A}} \left(a_{0,0} \rightarrow \bigwedge_{f \in Cond(\mathcal{A})} f_{i,w} \right) \right]$$ (node to leaf)

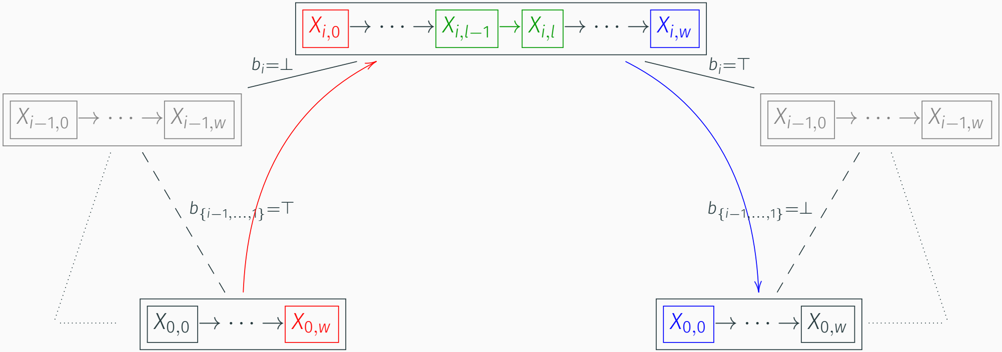
Encoding Transitions (example : preconditions of actions)



$$\bigwedge_{i=1}^{d} \left[\left(\neg b_i \wedge \bigwedge_{j=1}^{i-1} b_j \right) \rightarrow \bigwedge_{a \in \mathcal{A}} \left(a_{i,0} \rightarrow \bigwedge_{f \in Cond(\mathcal{A})} f_{0,w} \right) \right]$$

(leaf to node)

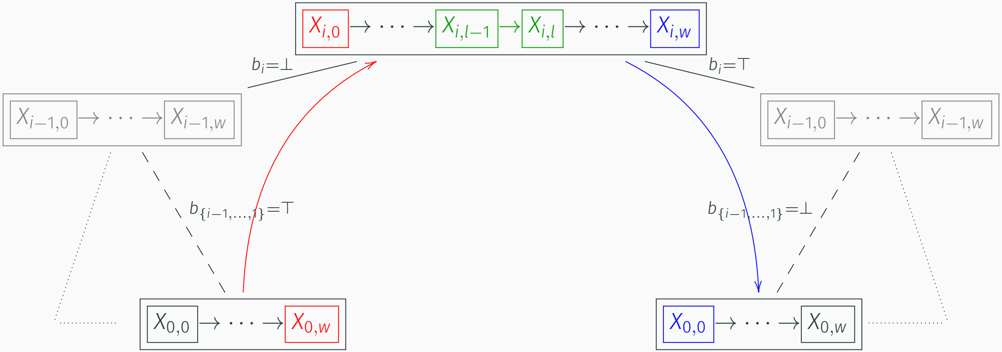
Encoding Transitions (example : preconditions of actions)



$$\bigwedge_{i=0}^{d} \bigwedge_{l=1}^{w} \bigwedge_{a \in \mathcal{A}} \left(a_{i,l} \rightarrow \bigwedge_{f \in Cond(\mathcal{A})} f_{i,l-1} \right)$$

(within node)

Encoding Transitions (example : preconditions of actions)



$$\bigwedge_{i=1}^{d} \left[\left(b_i \wedge \bigwedge_{j=1}^{i-1} \neg b_j \right) \to \bigwedge_{a \in \mathcal{A}} \left(a_{0,0} \to \bigwedge_{f \in Cond(\mathcal{A})} f_{i,w} \right) \right]$$

(node to leaf)

Encoding Transitions : Explanatory Frame-Axioms

If the value of a propositional variable corresponding to a fluent changes between two consecutive states from false to true, then an action which produces this change is executed in the plan transition between these states.

$$\bigwedge_{i=0}^{d}\bigwedge_{l=1}^{w}\bigwedge_{f\in\mathcal{F}}\left(\left(\neg f_{i,l-1}\wedge f_{i,l}\right)\rightarrow\left(\bigvee_{\substack{a\in\mathcal{A}\\ f\in Add(a)}} a_{i,l}\right)\right) \qquad \text{(within node)}$$

$$\bigwedge_{i=1}^{d}\left[\left(\neg b_i\wedge\bigwedge_{j=1}^{i-1} b_j\right)\rightarrow\bigwedge_{f\in\mathcal{F}}\left(\left(\neg f_{0,w}\wedge f_{i,0}\right)\rightarrow\left(\bigvee_{\substack{a\in\mathcal{A}\\ f\in Add(a)}} a_{i,0}\right)\right)\right] \qquad \text{(leaf to node)}$$

$$\bigwedge_{i=1}^{d}\left[\left(b_i\wedge\bigwedge_{j=1}^{i-1}\neg b_j\right)\rightarrow\bigwedge_{f\in\mathcal{F}}\left(\left(\neg f_{i,w}\wedge f_{0,0}\right)\rightarrow\left(\bigvee_{\substack{a\in\mathcal{A}\\ f\in Add(a)}} a_{0,0}\right)\right)\right] \qquad \text{(node to leaf)}$$

Prevent Negative Interactions

Finally, a last encoding rule is required to prevent negative interactions. Contradictory effects are already disallowed by previous rules (positive and negative effects of actions within a same step). We further states : if an action removes a fluent which is needed by another action, then these two actions cannot be both executed in a same plan transition.

$$\bigwedge_{i=0}^{d} \bigwedge_{l=0}^{w} \bigwedge_{\substack{(a,a')\in\mathcal{A}^2 \\ a\neq a' \\ Cond(a)\cap Del(a')\neq\emptyset}} (\neg a_{i,l} \vee \neg a'_{i,l})$$

Experiments on Classical Planning Benchmarks

Planning Problem	SAT +1	SAT ×2	CTE $w = 1$	CTE $w = 2$	CTE $w = 3$	CTE $w = 4$	CTE $w = 5$	CTE $w = 6$
BlocksWorld-6	17.65	**13.07**	(2) 24.85	(1) **14.47**	(1) 24.05	(1) 38.89	(1) 49.95	(1) 64.84
Gripper-6	2.76	1.86	(3) 3.31	(2) 2.14	(2) 3.23	(1) **1.21**	(1) 1.41	(1) 1.60
Gripper-7	253	49.12	(3) 4.66	(3) 22.96	(2) 4.67	(2) 14.59	(1) 2.57	(1) **2.35**
Ferry-8	$*$	39.76	(4) 70.09	(4) $*$	(3) 14.46	(3) 508	(2) **7.40**	(2) 15.16
Hanoi-5	$*$	131	(4) 111	(4) 196	(3) 125	(3) 221	(2) **51.81**	(2) 103

Table 1 : Plan generation time (in seconds) for SAT with incremental plan length, and CTE with different node widths and incremental tree depth. For CTE, the value given in parentheses is the depth for which a plan is found ; $*$ represents timeout over 10mn.

Experiments on Classical Planning Benchmarks

Planning Problem	SAT $+1$	SAT $\times 2$	CTE $w = 1$	CTE $w = 2$	CTE $w = 3$	CTE $w = 4$	CTE $w = 5$	CTE $w = 6$
BlocksWorld-6	17.65	**13.07**	(2) 24.85	(1) **14.47**	(1) 24.05	(1) 38.89	(1) 49.95	(1) 64.84
Gripper-6	2.76	1.86	(3) 3.31	(2) 2.14	(2) 3.23	(1) **1.21**	(1) 1.41	(1) 1.60
Gripper-7	253	49.12	(3) 4.66	(3) 22.96	(2) 4.67	(2) 14.59	(1) 2.57	(1) **2.35**
Ferry-8	$*$	39.76	(4) 70.09	(4) $*$	(3) 14.46	(3) 508	(2) **7.40**	(2) 15.16
Hanoi-5	$*$	131	(4) 111	(4) 196	(3) 125	(3) 221	(2) **51.81**	(2) 103

Table 1 : Plan generation time (in seconds) for SAT with incremental plan length, and CTE with different node widths and incremental tree depth. For CTE, the value given in parentheses is the depth for which a plan is found ; $*$ represents timeout over 10mn.

Question : how to find the good value for w ?