38050 Povo, Trento, Italy
IRST, Istituto per la Ricerca Scientifica e Tecnologica,
Paolo Traverso
Plannining as Model Checking
5. Conclusions

(c) Some experimental results

(b) Planning as OBDD-based Model Checking

(a) Planning as Symbolic Model Checking

4. Planning as Model Checking: Implementation

(c) Planning Algorithms.

(b) Planning Domains and Problems.

(a) The Idea

3. Planning as Model Checking: Theory

2. Planning: Background and Motivations.

1. Model Checking: Overview.

Index
The problem of determining whether a formula is true in a model is the Model Checking problem. 1. A domain of interest is described by a semantic model. 2. A desired property of the domain is described by a logical formula. 3. The fact that a domain satisfies a desired property is determined by checking whether the formula is true in the model.
propositions true in that state.

propositions \( T \) assigns to each state the set of atomic

where \( \mathcal{P} \) is a set of atomic

4. \( T : \mathcal{Z} \rightarrow \mathcal{P} \) is a labeling function, where \( \mathcal{P} \) is a set of atomic

such that \( \exists (m,n) \in \mathcal{E} \) for each state \( m \in \mathcal{M} \), there exists a state \( n \in \mathcal{M} \).

which gives the possible transitions between states. We require \( \mathcal{E} \)

which gives the possible transitions between states. We require \( \mathcal{E} \)

3. \( \mathcal{M} \times \mathcal{M} \models \mathcal{E} \), the transition relation,

2. \( \mathcal{M} \models \mathcal{E} \), the transition relation,

1. \( \mathcal{M} \) is a finite set of states.

A Kripke structure \( \mathcal{K} \) is a 4-tuple \( \langle \mathcal{K}, \mathcal{M}, \Sigma, \mathcal{E} \rangle \), where

\( \mathcal{M} \) is a finite set of states.

\( \mathcal{E} \) is a binary relation on \( \mathcal{M} \), the transition relation.

\( \mathcal{E} \) is a binary relation on \( \mathcal{M} \), the transition relation.

\( \mathcal{M} \times \mathcal{M} \) is a set of initial states.

A Kripke structure \( \mathcal{K} \) is a 4-tuple \( \langle \mathcal{K}, \mathcal{M}, \Sigma, \mathcal{E} \rangle \), where

\( \mathcal{M} \) is a finite set of states.

\( \mathcal{E} \) is a binary relation on \( \mathcal{M} \), the transition relation.

\( \mathcal{M} \times \mathcal{M} \) is a set of initial states.

A Kripke structure \( \mathcal{K} \) is a 4-tuple \( \langle \mathcal{K}, \mathcal{M}, \Sigma, \mathcal{E} \rangle \), where

\( \mathcal{M} \) is a finite set of states.

\( \mathcal{E} \) is a binary relation on \( \mathcal{M} \), the transition relation.

\( \mathcal{M} \times \mathcal{M} \) is a set of initial states.

A Kripke structure \( \mathcal{K} \) is a 4-tuple \( \langle \mathcal{K}, \mathcal{M}, \Sigma, \mathcal{E} \rangle \), where

\( \mathcal{M} \) is a finite set of states.

\( \mathcal{E} \) is a binary relation on \( \mathcal{M} \), the transition relation.

\( \mathcal{M} \times \mathcal{M} \) is a set of initial states.

A Kripke structure \( \mathcal{K} \) is a 4-tuple \( \langle \mathcal{K}, \mathcal{M}, \Sigma, \mathcal{E} \rangle \), where

\( \mathcal{M} \) is a finite set of states.

\( \mathcal{E} \) is a binary relation on \( \mathcal{M} \), the transition relation.

\( \mathcal{M} \times \mathcal{M} \) is a set of initial states.

A Kripke structure \( \mathcal{K} \) is a 4-tuple \( \langle \mathcal{K}, \mathcal{M}, \Sigma, \mathcal{E} \rangle \), where

\( \mathcal{M} \) is a finite set of states.

\( \mathcal{E} \) is a binary relation on \( \mathcal{M} \), the transition relation.

\( \mathcal{M} \times \mathcal{M} \) is a set of initial states.

A Kripke structure \( \mathcal{K} \) is a 4-tuple \( \langle \mathcal{K}, \mathcal{M}, \Sigma, \mathcal{E} \rangle \), where

\( \mathcal{M} \) is a finite set of states.

\( \mathcal{E} \) is a binary relation on \( \mathcal{M} \), the transition relation.

\( \mathcal{M} \times \mathcal{M} \) is a set of initial states.

A Kripke structure \( \mathcal{K} \) is a 4-tuple \( \langle \mathcal{K}, \mathcal{M}, \Sigma, \mathcal{E} \rangle \), where

\( \mathcal{M} \) is a finite set of states.

\( \mathcal{E} \) is a binary relation on \( \mathcal{M} \), the transition relation.

\( \mathcal{M} \times \mathcal{M} \) is a set of initial states.

A Kripke structure \( \mathcal{K} \) is a 4-tuple \( \langle \mathcal{K}, \mathcal{M}, \Sigma, \mathcal{E} \rangle \), where

\( \mathcal{M} \) is a finite set of states.

\( \mathcal{E} \) is a binary relation on \( \mathcal{M} \), the transition relation.

\( \mathcal{M} \times \mathcal{M} \) is a set of initial states.

A Kripke structure \( \mathcal{K} \) is a 4-tuple \( \langle \mathcal{K}, \mathcal{M}, \Sigma, \mathcal{E} \rangle \), where

\( \mathcal{M} \) is a finite set of states.

\( \mathcal{E} \) is a binary relation on \( \mathcal{M} \), the transition relation.

\( \mathcal{M} \times \mathcal{M} \) is a set of initial states.

A Kripke structure \( \mathcal{K} \) is a 4-tuple \( \langle \mathcal{K}, \mathcal{M}, \Sigma, \mathcal{E} \rangle \), where

\( \mathcal{M} \) is a finite set of states.

\( \mathcal{E} \) is a binary relation on \( \mathcal{M} \), the transition relation.

\( \mathcal{M} \times \mathcal{M} \) is a set of initial states.

A Kripke structure \( \mathcal{K} \) is a 4-tuple \( \langle \mathcal{K}, \mathcal{M}, \Sigma, \mathcal{E} \rangle \), where

\( \mathcal{M} \) is a finite set of states.

\( \mathcal{E} \) is a binary relation on \( \mathcal{M} \), the transition relation.

\( \mathcal{M} \times \mathcal{M} \) is a set of initial states.

A Kripke structure \( \mathcal{K} \) is a 4-tuple \( \langle \mathcal{K}, \mathcal{M}, \Sigma, \mathcal{E} \rangle \), where

\( \mathcal{M} \) is a finite set of states.

\( \mathcal{E} \) is a binary relation on \( \mathcal{M} \), the transition relation.

\( \mathcal{M} \times \mathcal{M} \) is a set of initial states.

A Kripke structure \( \mathcal{K} \) is a 4-tuple \( \langle \mathcal{K}, \mathcal{M}, \Sigma, \mathcal{E} \rangle \), where

\( \mathcal{M} \) is a finite set of states.
Domains as Kripke Structures: An Example

1. \( W = \{1, 2, 3, 4\} \)

2. \( W_0 = \{2\} \)

3. \( T = \{(1, 2), (2, 1), (2, 2), (2, 3), (3, 2), (3, 4), (4, 3)\} \)

4. \( L(1) = \{\text{Locked}\}, \ L(2) = \emptyset, \ L(3) = \{\text{Loaded}\}, \ L(4) = \{\text{Loaded, Locked}\}. \)

Note: *infinite* evolutions of the domain encoded as
paths \( w_0 w_1 w_2 \ldots \) such that, for each \( i, \ (w_i, w_{i+1}) \in T. \)
Properties as CTL formulas: An Example

\[\text{EF} \text{Loaded} \text{ holds in state 2}\]

\[\text{AF} \text{Loaded} \text{ does not hold in state 2}\]

\[\text{EG} \text{Loaded} \text{ holds in state 2}\]

\[\text{AG} \text{Loaded} \text{ does not hold in state 2}\]
The Model Checking Problem for a CTL formula $\varphi$ and a Kripke Structure $\mathcal{K}$

Truth: $d \models \varphi$ (for each $d \models \mathcal{K}$)

- $d \models \Box \varphi$ if there exists a path $\alpha$ with $m^0_m = m$ such that $\forall m', \varphi$, $\exists \alpha' \geq \alpha$, for all $b \models \varphi$, $d \models \varphi 

- $d \models \exists \alpha \varphi$ if for all paths $\alpha$, there exists $m^0_m = m$ with $\exists \alpha \varphi$.

- $d \models \varphi \lor \psi$ if $d \models \varphi$ or $d \models \psi$.

- $d \models \varphi \land \psi$ if $d \models \varphi$ and $d \models \psi$.

- $d \not\models \varphi$ if $d \not\models \varphi$.

- $d \models \varphi \implies \psi$ if $d \models \varphi \implies \psi$.

- $d \models \varphi$ if $d \models \varphi$.

- $d \models \varphi \iff d \models \psi$ if $d \models \varphi \iff d \models \psi$.

- $d \models (\varphi \wedge \psi)$ if $d \models (\varphi \wedge \psi)$.

- $d \models (\varphi \vee \psi)$ if $d \models (\varphi \vee \psi)$.

- $d \models ((\varphi \wedge \psi) \vee (\varphi \wedge \psi))$ if $d \models ((\varphi \wedge \psi) \vee (\varphi \wedge \psi))$.

- $\mathcal{K}$ semantics (Kripke Structure $\mathcal{K}$, CTL formula $\varphi$):
\[ \land (b \mathbf{XX} \lor \mathbf{dX} \lor d) \land (b \mathbf{X} \lor d) \land b = b \cap d \]

that

that \( (b \cap d) \mathbf{X} \lor (b \cap d) \mathbf{XX} \lor (b \cap d) \mathbf{dX} \lor (b \cap d) \lor \mathbf{XX} \lor \mathbf{dX} \lor d \)

all states (in some state \( s \), such that \( \mathbf{d} \in s \), \( \mathbf{X} \in s \), \( \lor \in s \), \( \lor \land \in s \))

An atomic formula \( t \) is model checked by verifying that

for all \( s \in M^0 \)

An atomic formula \( t \) is model checked by verifying that

For instance:

Algorithms for model checking exploit the structure of CTL formulas.
An example: Model Checking $\text{EF}^p$

1. function $\text{MCHECK}_{\text{EF}}(p,K)$
2. $\text{CurrentStates} := \emptyset$
3. $\text{NextStates} := \text{STATES}(p,K)$
4. while $\text{NextStates} \neq \text{CurrentStates}$ do
5.   if ($W_0 \subseteq \text{NextStates}$) then return True;
6.   $\text{CurrentStates} := \text{NextStates}$;
7.   $\text{NextStates} := \text{NextStates} \cup \text{ONESTEPMCHECK}(\text{NextStates},K)$;
8. endwhile
9. return False;

$\text{ONESTEPMCHECK}(\text{States}, K) = \{s \in W : \exists s'. (s' \in \text{States} \land T(s, s'))\}$
Planning as Model Checking
Classical (Strips-like) Planning: a simple example

action: load
  prec: ¬ Locked, ¬ Loaded
  post:
    add-list: Loaded
    del-list:

action: unload
  prec: ¬ Locked, Loaded
  post:
    add-list:
    del-list: Loaded

action: lock
  prec: ¬ Locked
  post:
    add-list: Locked
    del-list:
Classical Planning Assumptions:

- The initial situation is completely specified
- Actions have deterministic effects
- Plans are sequences of actions

```
plan = load ; lock
```
Planning in non-deterministic domains: Motivations

In several practical domains:

- The initial situation is partially specified
- Actions have non-deterministic effects
- Plans as sequences of actions are bound to failure

plan = load ; lock
Planning in non-deterministic domains: Motivations

Several practical domains require:

- Conditional Plans
- **Strong Plans**, i.e., plans that are guaranteed to achieve the goal in spite of non determinism.

plan = load ; if Misplaced then fix ; lock
Planning in non-deterministic domains: Motivations

Several practical domains require:

- Iterative Plans, encoding trial-and-error strategies
- **Strong Cyclic Plans**, i.e., plans whose executions
  - always have a possibility of terminating and, ...
  - when they do, they are guaranteed to achieve the goal

\[
\text{plan} = \text{repeat load until Loaded ; lock}
\]
Automatically and efficiently generate plans in non-deterministic domains is still an open problem.

- Stochastic Planning (focus on probabilistic distributions)
- Conformant Planning (solutions are sequences of actions)
- Contingency Planning (lack of efficient automatic planners)
- Reactive Systems (focus on execution, no plan generation)

Planning with non-determinism: State of the Art
Practical (automatically efficient planning for large size problems).

- General (e.g., non-deterministic planning).

Some Features:

- Planning is true.
- Planning is done by constructing sets of states where a temporal formula is true.
- Planning is done by constructing sets of states where a temporal formula is true.
- Planning is done by constructing sets of states where a temporal formula is true.
- Planning is done by constructing sets of states where a temporal formula is true.
- Planning is done by constructing sets of states where a temporal formula is true.

A model-based approach to planning:

Planning as Model Checking: The Idea
A planning domain $D$ is a 4-tuple $\langle F, S, A, R \rangle$ where

1. $F$ is a finite set of fluents,
2. $S \subseteq 2^F$ is a finite set of states,
3. $A$ is a finite set of actions,
4. $R \subseteq S \times A \times S$ is a transition relation.

$a \in A$ is executable in $s \in S$ if there exists $s' \in S$ such that $R(s, a, s')$.
Planning Problems

A planning problem $P$ for a planning domain $D = \langle F, S, A, R \rangle$ is a 3-tuple $\langle D, I, G \rangle$, where

1. $I \subseteq S$ is the set of initial states,
2. $G \subseteq S$ is the set of goal states.
A *plan* for a planning domain $D = \langle F, S, A, R \rangle$ is a set of pairs
$$\{ \langle s, a \rangle : s \in S, a \in A, a \text{ executable in } s \}$$

Set of states of a plan $\pi$: $States(\pi) = \{ s : \exists a \in A. \langle s, a \rangle \in \pi \}$. 

plan = \{ <2,load> , <3,lock> , <5,fix> \}
Plan Executions

An execution of a plan $\pi$ from the state $s_0 \in S$ is an infinite sequence $s_0, s_1, \ldots$ of states in $S$ such that, for all $i \geq 0$,

1. if $s_i \in States(\pi)$ then, for some $a \in A$, $(s_i, a) \in \pi$ and $s_{i+1} \in R(s_i, a)$.
2. if $s_i \notin States(\pi)$ then $s_i = s_{i+1}$,

$$\text{plan} = \{ <2, \text{load}> , <3, \text{lock}> , <5, \text{fix}> \}$$

$$\text{executions} = \begin{array}{c}
1, 1, 1, 1, \ldots \\
2, 2, 2, 2, \ldots \\
2, 2, 3, 4, \ldots \\
2, 3, 4, 4, \ldots \\
2, 5, 3, 4, \ldots
\end{array}$$
propositions true in that state.

propositions \( T \) assigns to each state the set of atomic

where \( p \) is a set of atomic

4. \( T : 2^p \leftrightarrow M : 2^n \) \(\forall m \in M \) there exists a state \( M \in M \) which gives the possible transitions between states. We require \( T \)

3. \( M \times M \supseteq \mathcal{L} \)

the transition relation.

2. \( M \supseteq M^0 \)

1. \( M \) is a finite set of states.

A Kripke Structure \( K \) is a 4-tuple \( \langle T, M^0, \mathcal{L}, \rangle \), where

Background: Kripke Structures
Induced Kripke Structures

Let $\pi$ be a plan for the planning domain $D$. The Kripke structure $K = \langle W, T, L \rangle$ induced by $\pi$ is defined as

1. $W = S$;
2. $T(s, s')$ iff $\langle s, a \rangle \in \pi$ and $s' \in R(s, a)$, or $s \not\in States(\pi)$ and $s = s'$;
3. $L(s) = s$.

Plan = \{ <2,load>, <3,lock>, <5,fix> \}

Induced Kripke Structure:
Given a finite set \( \mathcal{P} \) of atomic propositions, CTL formulas are defined as follows:

\[
\begin{align*}
\neg \Phi & \quad \text{(for every path)} \\
\Phi_1 \land \Phi_2 & \quad \text{(for every path)} \\
\Phi \land \psi & \quad \text{(for every path)} \\
\exists x \Phi & \quad \text{(for every path)} \\
\forall \exists \Phi & \quad \text{(for every path)} \\
\end{align*}
\]

for some path \( (b \land \Phi) \), \( \forall \exists \Phi \)

or

\[
\begin{align*}
\forall \exists \Phi & \quad \text{(for every path)} \\
\exists x \Phi & \quad \text{(for every path)} \\
\end{align*}
\]

The following are CTL formulas:

1. Every atomic proposition \( \phi \in \mathcal{P} \) is a CTL formula;
2. If \( b \) and \( \Phi \) are CTL formulas, then so are

\[
\begin{align*}
\neg \Phi & \quad \text{(for every path)} \\
\Phi_1 \land \Phi_2 & \quad \text{(for every path)} \\
\Phi \land \psi & \quad \text{(for every path)} \\
\exists x \Phi & \quad \text{(for every path)} \\
\forall \exists \Phi & \quad \text{(for every path)} \\
\end{align*}
\]

\[\text{Computation Tree Logic (CTL)}\]
Computation Tree Logic (CTL): Examples

EF $\text{Loaded}$ holds in state 2

AF $\text{Loaded}$ does not hold in state 2

EG $\text{Loaded}$ holds in state 2

AG $\text{Loaded}$ does not hold in state 2
The Model Checking Problem for a CTL formula $\phi$ and a Kripke Structure $K$.

Truth: $d \models \phi$ if, for each $m \in \mathcal{W}$, $d \models \phi$.

- $d \models \phi$ if $\exists m \in \mathcal{W}$ such that $d \models \phi$.
- $d \models \phi$ if $\forall m \in \mathcal{W}$ such that $d \models \phi$.
- $d \models \phi$ if $\exists m \in \mathcal{W}$ such that $d \models \phi$.
- $d \models \phi$ if $\forall m \in \mathcal{W}$ such that $d \models \phi$.
- $d \models \phi$ if $\exists m \in \mathcal{W}$ such that $d \models \phi$.
- $d \models \phi$ if $\forall m \in \mathcal{W}$ such that $d \models \phi$.
- $d \models \phi$ if $\exists m \in \mathcal{W}$ such that $d \models \phi$.
- $d \models \phi$ if $\forall m \in \mathcal{W}$ such that $d \models \phi$.
- $d \models \phi$ if $\exists m \in \mathcal{W}$ such that $d \models \phi$.
- $d \models \phi$ if $\forall m \in \mathcal{W}$ such that $d \models \phi$.
- $d \models \phi$ if $\exists m \in \mathcal{W}$ such that $d \models \phi$.
- $d \models \phi$ if $\forall m \in \mathcal{W}$ such that $d \models \phi$.
- $d \models \phi$ if $\exists m \in \mathcal{W}$ such that $d \models \phi$.
- $d \models \phi$ if $\forall m \in \mathcal{W}$ such that $d \models \phi$.
- $d \models \phi$ if $\exists m \in \mathcal{W}$ such that $d \models \phi$.
- $d \models \phi$ if $\forall m \in \mathcal{W}$ such that $d \models \phi$.
- $d \models \phi$ if $\exists m \in \mathcal{W}$ such that $d \models \phi$.
- $d \models \phi$ if $\forall m \in \mathcal{W}$ such that $d \models \phi$.
- $d \models \phi$ if $\exists m \in \mathcal{W}$ such that $d \models \phi$.
- $d \models \phi$ if $\forall m \in \mathcal{W}$ such that $d \models \phi$.
- $d \models \phi$ if $\exists m \in \mathcal{W}$ such that $d \models \phi$.
- $d \models \phi$ if $\forall m \in \mathcal{W}$ such that $d \models \phi$.
- $d \models \phi$ if $\exists m \in \mathcal{W}$ such that $d \models \phi$.
- $d \models \phi$ if $\forall m \in \mathcal{W}$ such that $d \models \phi$.
- $d \models \phi$ if $\exists m \in \mathcal{W}$ such that $d \models \phi$.
- $d \models \phi$ if $\forall m \in \mathcal{W}$ such that $d \models \phi$.
- $d \models \phi$ if $\exists m \in \mathcal{W}$ such that $d \models \phi$.
- $d \models \phi$ if $\forall m \in \mathcal{W}$ such that $d \models \phi$.
- $d \models \phi$ if $\exists m \in \mathcal{W}$ such that $d \models \phi$.
- $d \models \phi$ if $\forall m \in \mathcal{W}$ such that $d \models \phi$.
- $d \models \phi$ if $\exists m \in \mathcal{W}$ such that $d \models \phi$.
- $d \models \phi$ if $\forall m \in \mathcal{W}$ such that $d \models \phi$.
- $d \models \phi$ if $\exists m \in \mathcal{W}$ such that $d \models \phi$.
- $d \models \phi$ if $\forall m \in \mathcal{W}$ such that $d \models \phi$.
- $d \models \phi$ if $\exists m \in \mathcal{W}$ such that $d \models \phi$.
- $d \models \phi$ if $\forall m \in \mathcal{W}$ such that $d \models \phi$.
- $d \models \phi$ if $\exists m \in \mathcal{W}$ such that $d \models \phi$.
Weak Solutions

Given: A planning domain $D$ and problem $P$

A plan $\pi$ and the Induced Kripke Structure $K$

The plan $\pi$ is a weak solution to the planning problem $P$ if for all $s \in I$, $K, s \models EF G$.

Planning domain

$\begin{array}{c}
1 \\
\text{lock} \\
\Downarrow \\
\text{unlock} \\
2 \\
\text{load} \\
\Downarrow \\
3 \\
\text{fix} \\
\Downarrow \\
\text{lock} \\
4 \\
\text{unlock} \\
5 \\
\end{array}$

plan = \{ <2,\text{load}> , <3,\text{lock}> \}

Induced Kripke Structure:

$\begin{array}{c}
1 \\
\Downarrow \\
2 \\
\Downarrow \\
3 \\
\Downarrow \\
4 \\
\Downarrow \\
5 \\
\end{array}$
**Strong Solutions**

Given: A planning domain $D$ and problem $P$

A plan $\pi$ and th Induced Kripke Structure $K$

The plan $\pi$ is a strong solution to the planning problem $P$ if for all $s \in I$, $K, s \models AF G$.

Planning domain

\[
\begin{array}{c}
1 \quad \text{unlock} \\
2 \quad \text{load} \quad \text{lock} \\
3 \quad \text{unload} \quad \text{fix} \\
4 \quad \text{unlock} \\
5 \\
\end{array}
\]

Plan = \{ <2, load> , <3, lock> , <5, fix> \}

Induced Kripke Structure:

\[
\begin{array}{c}
1 \\
2 \\
3 \\
4 \quad 5 \\
\end{array}
\]
Strong Cyclic Solutions

Given: A planning domain $D$ and problem $P$
A plan $\pi$ and the Induced Kripke Structure $K$

The plan $\pi$ is a strong cyclic solution to the planning problem $P$ if for all $s \in I$, $K, s \models AGEF G$.

Planning domain

$\begin{align*}
1 & \xrightarrow{\text{unlock}} 2 \\
2 & \xrightarrow{\text{load}} 3 \\
3 & \xrightarrow{\text{fix}} 5 \\
3 & \xrightarrow{\text{unload}} 4 \\
4 & \xrightarrow{\text{unlock}} 4
\end{align*}$

$\text{plan} = \{ <2, \text{load}> , <3, \text{lock}>, <5, \text{fix}> \}$

Induced Kripke Structure:

$\begin{align*}
1 & \\
2 & \xrightarrow{} 3 \\
3 & \xrightarrow{} 5 \\
4 & \xrightarrow{} 1
\end{align*}$
Planning Algorithms

- Sets of states manipulated during the search
- OBDD based implementation
**Weak Planning**

- **function** `WeakPlan(P)`
  - `States := ∅; Plan := ∅; ...
  - **while** `I ∉ States` **and** not all states visited so far **do**
    - `π := π ∪ PruneStates(WeakPreImg(States, D), States)`.
    - `States := States ∪ ProjectActions(WeakPreImg(States, D))`.
  - **if** `I ∉ States` **then return** `π` **else return** `Fail`

**WeakPreImg(States, π):**
returns all the `(s, a)` possibly leading to states of `States`

**PruneStates(π, States):**
eliminates from `π` the `(s, a)` whose states are already in `States`
(3) \[ \{ s \in (s', a) \cup \text{actions} : \text{states}(s') \neq \text{states}(s, a) \} = (\text{actions}(s')) \]

(2) \[ \{ s, a \in \text{states} \mid \text{states}(s', a) \subseteq \text{states}(s) \} = (\text{states}(s')) \]

(1) \[ \text{WeakPlanning}(\text{states}, D) \]

```
function WeakPlanning(\text{plan}) {
    \text{return Plan};
    \text{CurrentStates} \equiv \text{NextStates};
    \text{NextStates} \equiv \text{NextStates} \cap \text{PruneStates}(\text{OneStepPlan})
    \text{Plan} \equiv \text{Plan} \cap \text{PruneStates}(\text{OneStepPlan})
    \text{OneStepPlan} \equiv \text{WeakPlanning}(\text{NextStates}, D)
    \text{then return Plan}.
    \text{if } I \subseteq \text{NextStates} \text{ then return Plan.}
    \text{while } (\text{CurrentStates} \neq \text{NextStates}) \text{ do}
        \text{CurrentStates} \equiv \text{NextStates};
        \text{Plan} \equiv \text{Plan} \cup \text{WeakPlanning}(\text{Plan}, D)
    \text{end while}
}
```
Strong Planning

function STRONGPLAN(P)
. States := ∅; Plan := ∅; ...
. while I ⊈ States and not all states visited so far do
. . π := π ∪ PRUNESTATES(STRONGPREIMG(States, D), States).
. . States := States ∪ PROJECTACTIONS(STRONGPREIMG(States, D))
. . if I ⊈ States then return π else return Fail

STRONGPREIMG(States, π):
returns all the ⟨s, a⟩ necessarily leading to states of States
The strong cyclic planning algorithm: intuitions

Strong cyclic plans: plans whose executions satisfy \( \text{AGEFG} \)

\[
\text{EFp} = p \text{ or EXp or EXEXp or EXEXEXp or ...}
\]

\[
\text{AGq} = q \text{ and AXq and AXAXq and AXAXAXq and ...}
\]
Strong Cyclic Planning: intuitions (cont.)

plan = \{<1,a>,<2,b>\}

plan = \{<1,a>,<2,b>\}

plan = \{<1,a>\}

plan = \{<1,a>,<2,b>\}
removes loops with no possibility of termination

\[ \text{PRUNE\_CONNECTED}(p, \nabla) : \]

removes all states that may lead to non-goal states not in \( \nabla \)

\[ \text{PRUNE\_OUTGOING}(p, \nabla) : \]

returns all the \( \langle s, a \rangle \) possibly leading to states of \( \nabla \) or \( G \)

\[ \text{ONE\_STEP\_BACK}(p, \nabla) : \]

7. if \( I \neq \emptyset \) then return \( \emptyset \) else return False

6. \( \nabla := \text{PRUNE\_CONNECTED}(p, \nabla) \).

5. while \( \nabla \) has state-action pairs to be pruned away do

4. \( \nabla := \text{ONE\_STEP\_BACK}(p, \nabla) \).

3. while \( \emptyset \) states of \( \nabla \) and not all states visited so far do

2. \( \nabla := \emptyset \).

1. function \text{STRONG\_CYCLIC\_PLAN}(p)
The strong cyclic planning algorithm: intuitions (cont.)
Propositional formulas as OBBDDs

Quantified boolean formulas (QBF)

Sets (e.g., sets of states, plans) as propositional and

The approach:

NuSMV, a state of the art OBBDD based symbolic model checker

The results: the Model Based Planner (MBP), built on top of

Ordered Binary Decision Diagrams (OBDDs)

The ideas: exploit the work on symbolic model checking based

spaces.

The problem: realistic planning domains have often large state

Planning via Symbolic Model Checking
Planning Domains via Symbolic Model Checking

Let $D$ be a planning domain $\langle F, S, A, R \rangle$:

- Fluents as boolean variables (state variables) $x$
- A state corresponds to a complete assignment to the variables in $x$.
- A propositional formula $\phi(x)$ encodes the set of the states corresponding to the assignments that make $\phi$ true
- Actions as boolean variables (action variables) $\alpha$
- A formula $\psi(x, \alpha)$ encodes a relation between states and actions (e.g., a plan)
Planning Domains via Symbolic Model Checking (cont.)

Let $D$ be a planning domain $\langle F, S, A, R \rangle$:

- Transition relations are formulas in state variables $x$, action variables $\alpha$, and next state variables $x'$: $R(x, \alpha, x')$

- Example:

$$\neg Loaded \land \neg Locked \land \neg Misplaced \land Act = load \supset$$

$$\neg Locked' \land \neg (Loaded' \land Misplaced')$$

Diagram showing state transitions:

1. Locked
   - lock
2. Loaded
   - load
   - unlock
3. Loaded
   - unload
   - fix
4. Locked
   - lock
   - unlock
5. Misplaced
   - Misplaced
   - fix
   - lock
Planning Domains via Symbolic Model Checking (cont.)

Let $D$ be a planning domain $\langle F, S, A, R \rangle$:

- Transition relations are formulas in state variables $x$, action variables $\alpha$, and next state variables $x'$: $R(x, \alpha, x')$

- Example:

$$
(\neg \text{Loaded} \land \neg \text{Locked} \land \neg \text{Misplaced} \land \text{Act} = \text{load}) \models \\
\neg \text{Locked}'
$$
\[
((\forall x' \exists x \mathcal{P}(x')) \lor \exists x \mathcal{Q}(x)) \subseteq \exists x \mathcal{Q}(x)
\]

Some examples:

Planning via Symbolic Model Checking: Algorithms
\begin{align*}
((\overline{x},a)\in States \lor (\overline{x})_{\overline{x}} \in States, D)_{\overline{x}} & \models (a, x) \phi \\
\text{StrongPrem}(States, D) & \text{ represented by the formula} \\
((\overline{x},a)\in States \lor (\overline{x})_{\overline{x}} \in States, D)_{\overline{x}} & \models (a, x) \phi \\
\text{WeakPrem}(States, D) & \text{ represented by the formula} \\
\end{align*}

Some examples:

Planning via Symbolic Model Checking: Algorithms
Planning via Symbolic Model Checking with OBDDs

- OBDD’s are a compact representation of the assignments satisfying (and falsifying) a given boolean formula.
- Operations over sets of states (e.g. union, intersection) are implemented as boolean operations (e.g. conjunction, disjunction) over OBDDs.
Planning via Symbolic Model Checking with OBDDs

Transitions are represented by OBDD’s in the variables
\( x \) (current state), \( a \) (actions), \( x' \) (next states)

\((\neg Loaded \land Locked \land lock) \Rightarrow \neg Loaded' \land \neg Locked'\)

\((\neg Loaded \land \neg Locked \land lock) \Rightarrow \neg Locked'\)
Experimental Analysis: the problems

- the Hunter and Prey problem (extensions: universal; cyclic)

- the Beam problem (strong cyclic solutions)
Future Work
Conclusions

Planning as Model Checking is:

- a step towards planning as synthesis from temporal specifications;
- practical (fast algorithms);
- well-founded (correct, complete algorithms);
- a solution to some problems in planning with non-determinism;
- a novel approach to planning.
F. Giunchiglia, P. Traverso. Planning as Model Checking, ECP99

Daniele Traverso, Verdi. Strong Cyclic Planning Revisited, ECP99

Cimatti, Roveri: Conformant Planning via model checking, ECP99

Universal Plans in Non-Deterministic Domains, AIPS98

Cimatti, Roveri, Traverso. Automatic OBDD-Based Generation of

Domains via Model Checking, AIPS98

Cimatti, Roveri, Traverso. Strong Planning in Non-Deterministic

via Model Checking: A Decision Procedure for AR, ECP97


Bibliography
Moshe Y. Vardi (Rice University)

Marco Pistore (IRST)

Marco Roveri (IRST)

Fausto Giunchiglia (University of Trento and IRST)

Enrico Giunchiglia (University of Genoa)

Marco Daniele (IRST)

Alessandro Cimatti (IRST)

Piergiorgio Bertoli (IRST)

Acknowledgements