From Conformant into Classical Planning: Efficient Translations That May Be Complete Too

Héctor Palacios Héctor Geffner

Department de Tecnología
Universitat Pompeu Fabra
Barcelona

ICAPS – 2007

Classical Planning

- A **classical planner** is a **solver** over the class of **models** given by:
  - a **state space** S
  - a known **initial state** $s_0 \in S$
  - a set $S_G \subseteq S$ of **goal states**
  - **actions** $A(s) \subseteq A$ applicable in each $s \in S$
  - a deterministic **transition function** $s' = f(a, s)$ for $a \in A(s)$
  - uniform **action costs** $c(a, s) = 1$
- These **models** are represented in compact form through **languages** such as Strips, ADL, PDDL, ...
- Their **solutions** (plans) are sequences of applicable actions that map s_0 into S_G

Status of Classical Planning

- The good news: **classical planning works**
  - *Large problems solved very fast (non-optimally)*
- Not so good: **limitations**
  - *No **Uncertainty** (no probabilities)*
  - *No **Incomplete Information** (no sensing)*
  - ...

Beyond Classical Planning: Two Strategies

1. **Top-down:** Develop solver for **more general class of models**; e.g., MDPs and POMDPs

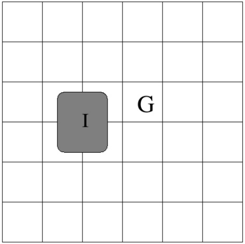
   $+$: generality
   $-$: complexity

2. **Bottom-up:** Extend the scope of **current 'classical' solvers**

   $+$: efficiency
   $-$: generality

We follow 2: we want to use **classical planning algorithms** for solving problems that involve **incomplete information** (conformant planning)

Conformant Planning: the Trouble with Incomplete Info

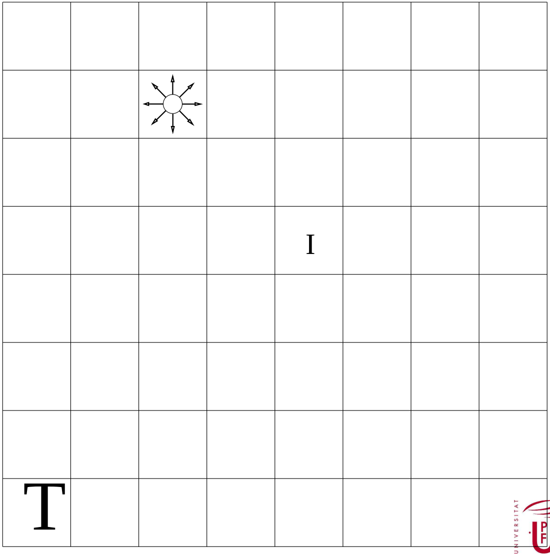


Problem: A robot must move from an **uncertain** *I* into *G* with **certainty**, one cell at a time, in a grid nxn

- Conformant and classical planning look similar except for uncertain *I* (assuming actions are deterministic).
- Yet plans can be quite different:
  best **conformant plan must move robot to a corner first!** (in order to localize)

Look-n-grab 8x8

- **Actions**: move, *look-and-grab*, putdown
- **Init**: object can be anywhere.
- **Goal**: object at Trash
- Obj **get lost** when pickup with handfull, so have to **visit Trash after each pickup**



Model for Conformant Planning

- a **set** $b_0 \subseteq S$ of possible initial states
- a set of possible goals $b_F \subseteq S$
- actions $A(s) \subseteq A$ applicable in each $s \in S$
- a **non-deterministic** state transition function F s.t. $F(a, s)$ is the **set** of next states

– *call a set of possible states, a **belief state***

– *actions then map a belief state b into a belief state b_a*

$$b_a \stackrel{\text{def}}{=} \{s' \mid s' \in F(a, s) \ \& \ s \in b\}$$

– ***task** becomes finding **action sequence** that maps b_0 into target b_F*

Who care about Conformant Planning?

- What we **really** want is observations, probabilities, time, resources, etc
- *Better* Conformant Planning leads to *better* **Planning with Observations** (contingent)
  - Contingent-FF uses Conformant-FF's heuristic
  - POND do both: conformant and contingent
- Claim: Finding **sequence of actions** between belief states is a *key point* in planning under observations, probabilities, etc.

Who care about Conformant Planning?

- What we **really** want is observations, probabilities, time, resources, etc
- *Better* Conformant Planning leads to *better* **Planning with Observations** (contingent)
  - Contingent-FF uses Conformant-FF's heuristic
  - POND do both: conformant and contingent
- Claim: Finding **sequence of actions** between belief states is a *key point* in planning under observations, probabilities, etc.

Who care about Conformant Planning?

- What we **really** want is observations, probabilities, time, resources, etc
- *Better* Conformant Planning leads to
  better **Planning with Observations** (contingent)
  - Contingent-FF uses Conformant-FF's heuristic
  - POND do both: conformant and contingent
- Claim: Finding **sequence of actions** between belief states is a *key point* in planning under observations, probabilities, etc.

Search in belief space

- GPT, MBP, POND do conformant planning by **heuristic search** in belief space. Issues:
  - **which heuristic?**
  - **explicit** representation of belief states
- Alternatives
  - Conformant-FF use different **representation**
  - Use **propositional logic**, SATPLAN-like
    1. **Model counting** for search over possible plans
    2. Construct a CNF with all possible plans, and call *once* a **SAT solver**

Search in belief space

- GPT, MBP, POND do conformant planning by **heuristic search** in belief space. Issues:
  - **which heuristic?**
  - **explicit** representation of belief states
- Alternatives
  - Conformant-FF use different **representation**
  - Use **propositional logic**, SATPLAN-like
    1. **Model counting** for search over possible plans
    2. Construct a CNF with all possible plans, and call *once* a **SAT solver**

Complexity: Classical vs. Conformant Planning

- **Complexity:** conformant planning harder than classical planning
    - *because **verification** of a conformant plan **intractable** in worst case*
- **Idea:** focus on computation of conformant plans that are **easy to verify** (e.g., in linear time in the plan length)
    - *computation of such plans **no more complex than classical planning***

Translation-based approach to Conformant Planning

- Exploiting translation-idea, effective but incomplete **translation scheme** proposed in AAAI-06
  - Plans for Conformant P obtained from plans for Classical $K(P)$
- Conformant Planner **KP** = $K(P)+$ FF did **very well** in IPC-2006
- Another Planner **T0** $= K_1(P)+$ FF even better (**1st place**)
- Translation $K_1(P)$ and more general $K_{T,M}(P)$ presented in this paper

Outline

- Basic Translation Scheme $K_0(P)$
- General Translation Scheme $K_{T,M}(P)$
- **Complete** Instances
- **Conformant Width** of P
- **Poly** translation K_i that is complete if width $\leq i$
- Experiments: Width Analysis, Performance of **T0** $= K_1(P)+$ FF

Translation from P into $K_0(P)$

For a **conformant problem** $P = \langle F, O, I, G \rangle$

- F stands for the fluents in P
- O for the operators with effects $C \rightarrow L$
- I for the initial situation (clauses over F-literals)
- G for the goal situation (set of F-literals)

Translation from P into $K_0(P)$

For a **conformant problem** $P = \langle F, O, I, G \rangle$

- F stands for the fluents in P
- O for the operators with effects $C \rightarrow L$
- I for the initial situation (clauses over F-literals)
- G for the goal situation (set of F-literals)

Conformant P	$\Rightarrow$	**Classical** $K_0(P)$
$\langle F, I, O, G \rangle$	$\Rightarrow$	$\langle F', I', O', G' \rangle$
Fluent L	$\Rightarrow$	KL, $K\neg L$ *(two fluents)*
Init: known lit L	$\Rightarrow$	$KL \wedge \neg K\neg L$
unknown lit L	$\Rightarrow$	$\neg KL \wedge \neg K\neg L$ *(both false)*
Goal L	$\Rightarrow$	KL
Operator a has prec L	$\Rightarrow$	a has prec KL
Operator a: $C \rightarrow L$	$\Rightarrow$	$\left\{ \begin{array}{lll} a: & KC & \rightarrow KL \\ a: & \neg K\neg C & \rightarrow \neg K\neg L \end{array} \right.$

Translation from P into $K_0(P)$

For a **conformant problem** $P = \langle F, O, I, G \rangle$

- F stands for the fluents in P
- O for the operators with effects $C \rightarrow L$
- I for the initial situation (clauses over F-literals)
- G for the goal situation (set of F-literals)

Conformant P	$\Rightarrow$	**Classical** $K_0(P)$
$\langle F, I, O, G \rangle$	$\Rightarrow$	$\langle F', I', O', G' \rangle$
Fluent L	$\Rightarrow$	KL, $K\neg L$ *(two fluents)*
Init: known lit L	$\Rightarrow$	$KL \wedge \neg K \neg L$
unknown lit L	$\Rightarrow$	$\neg KL \wedge \neg K \neg L$ *(both false)*
Goal L	$\Rightarrow$	KL
Operator a has prec L	$\Rightarrow$	a has prec KL
Operator a: $C \rightarrow L$	$\Rightarrow$	$\begin{cases} a: KC \rightarrow KL \\ a: \neg K \neg C \rightarrow \neg K \neg L \end{cases}$

Translation from P into $K_0(P)$

For a **conformant problem** $P = \langle F, O, I, G \rangle$

- F stands for the fluents in P
- O for the operators with effects $C \rightarrow L$
- I for the initial situation (clauses over F-literals)
- G for the goal situation (set of F-literals)

	Conformant P	$\Rightarrow$	**Classical** $K_0(P)$
	$\langle F, I, O, G \rangle$	$\Rightarrow$	$\langle F', I', O', G' \rangle$
	Fluent L	$\Rightarrow$	$KL, K\neg L$ *(two fluents)*
Init:	known lit L	$\Rightarrow$	$KL \wedge \neg K\neg L$
	unknown lit L	$\Rightarrow$	$\neg KL \wedge \neg K\neg L$ *(both false)*
	Goal L	$\Rightarrow$	KL
	Operator a has prec L	$\Rightarrow$	a has prec KL
	Operator a: $C \rightarrow L$	$\Rightarrow$	$\left\{ \begin{array}{lll} a: & KC & \rightarrow KL \\ a: & \neg K\neg C & \rightarrow \neg K\neg L \end{array} \right.$

Translation from P into $K_0(P)$

For a **conformant problem** $P = \langle F, O, I, G \rangle$

- F stands for the fluents in P
- O for the operators with effects $C \rightarrow L$
- I for the initial situation (clauses over F-literals)
- G for the goal situation (set of F-literals)

	Conformant P		Classical $K_0(P)$
	$\langle F, I, O, G \rangle$	$\Rightarrow$	$\langle F', I', O', G' \rangle$
	Fluent L	$\Rightarrow$	$KL, K\neg L$ *(two fluents)*
Init:	known lit L	$\Rightarrow$	$KL \wedge \neg K\neg L$
	unknown lit L	$\Rightarrow$	$\neg KL \wedge \neg K\neg L$ *(both false)*
	Goal L	$\Rightarrow$	KL
	Operator a has prec L	$\Rightarrow$	a has prec KL
	Operator a: $C \rightarrow L$	$\Rightarrow$	$\begin{cases} a: & KC \rightarrow KL \\ a: & \neg K\neg C \rightarrow \neg K\neg L \end{cases}$

Translation from P into $K_0(P)$

For a **conformant problem** $P = \langle F, O, I, G \rangle$

- F stands for the fluents in P
- O for the operators with effects $C \rightarrow L$
- I for the initial situation (clauses over F-literals)
- G for the goal situation (set of F-literals)

Conformant P	$\Rightarrow$	**Classical** $K_0(P)$
$\langle F, I, O, G \rangle$	$\Rightarrow$	$\langle F', I', O', G' \rangle$
Fluent L	$\Rightarrow$	KL, $K\neg L$ *(two fluents)*
Init: known lit L	$\Rightarrow$	$KL \wedge \neg K\neg L$
unknown lit L	$\Rightarrow$	$\neg KL \wedge \neg K\neg L$ *(both false)*
Goal L	$\Rightarrow$	KL
Operator a has prec L	$\Rightarrow$	a has prec KL
Operator a: $C \rightarrow L$	$\Rightarrow$	$\left\{ \begin{array}{llll} a: & KC & \rightarrow & KL \\ a: & \neg K\neg C & \rightarrow & \neg K\neg L \end{array} \right.$

Translation from P into $K_0(P)$

For a **conformant problem** $P = \langle F, O, I, G \rangle$

- F stands for the fluents in P
- O for the operators with effects $C \rightarrow L$
- I for the initial situation (clauses over F-literals)
- G for the goal situation (set of F-literals)

Conformant P	$\Rightarrow$	**Classical** $K_0(P)$
$\langle F, I, O, G \rangle$	$\Rightarrow$	$\langle F', I', O', G' \rangle$
Fluent L	$\Rightarrow$	$KL, K\neg L$ *(two fluents)*
Init: known lit L	$\Rightarrow$	$KL \wedge \neg K \neg L$
unknown lit L	$\Rightarrow$	$\neg KL \wedge \neg K \neg L$ *(both false)*
Goal L	$\Rightarrow$	KL
Operator a has prec L	$\Rightarrow$	a has prec KL
Operator a: $C \rightarrow L$	$\Rightarrow$	$\left\{ \begin{array}{llll} a: & KC & \rightarrow & KL \\ a: & \neg K \neg C & \rightarrow & \neg K \neg L \end{array} \right.$

Translation from P into $K_0(P)$

For a **conformant problem** $P = \langle F, O, I, G \rangle$

- F stands for the fluents in P
- O for the operators with effects $C \rightarrow L$
- I for the initial situation (clauses over F-literals)
- G for the goal situation (set of F-literals)

Conformant P	$\Rightarrow$	**Classical** $K_0(P)$
$\langle F, I, O, G \rangle$	$\Rightarrow$	$\langle F', I', O', G' \rangle$
Fluent L	$\Rightarrow$	$KL, K\neg L$ *(two fluents)*
Init: known lit L	$\Rightarrow$	$KL \wedge \neg K\neg L$
unknown lit L	$\Rightarrow$	$\neg KL \wedge \neg K\neg L$ *(both false)*
Goal L	$\Rightarrow$	KL
Operator a has prec L	$\Rightarrow$	a has prec KL
Operator a: $C \rightarrow L$	$\Rightarrow$	$\left\{ \begin{array}{lrcl} a: & KC & \rightarrow & KL \\ a: & K\neg C & \rightarrow & \emptyset \\ a: & \neg K\neg C & \rightarrow & \neg K\neg L \end{array} \right.$

Basic Properties and Extensions

- Translation $K_0(P)$ is **sound**:
  - If π is a **classical plan** that solves $K_0(P)$, then π is a **conformant plan** for P.
- But way **too incomplete**
  - often $K_0(P)$ will have no solution while P does
  - works when **uncertainty is irrelevant**
- Extension $K(P)$ in AAAI-06 is more powerful (more problems solvable) but still **basically incomplete**
- Extension $K_{T,M}(P)$ we present now **can** be both **complete and polynomial**

Basic Properties and Extensions

- Translation $K_0(P)$ is **sound**:
  - If π is a **classical plan** that solves $K_0(P)$, then π is a **conformant plan** for P.
- But way **too incomplete**
  - often $K_0(P)$ will have no solution while P does
  - works when **uncertainty is irrelevant**

- Extension $K(P)$ in AAAI-06 is more powerful (more problems solvable) but still **basically incomplete**
- Extension $K_{T,M}(P)$ we present now **can** be both **complete and polynomial**

Idea

- Given literal L and tag t, atom KL/t means
  - $K(t_0 \supset L)$: *KL true if t is true* **initially**

Example

- Conformant Problem P:
  - Init: $x_1 \vee x_2, \neg g$
  - Goal: g
  - Actions: $a_1 : x_1 \rightarrow g$, $a_2 : x_2 \rightarrow g$
- Classical Problem $K_{T,M}(P)$:
  - Init: $Kx_1/x_1, Kx_2/x_2, K\neg g, \neg Kg, \neg Kx_1, \neg K\neg x_1, \ldots$
  - After a_1: Kg/x_1, $Kx_1/x_1, Kx_2/x_2, \neg K\neg g, \neg Kg, \ldots$
  - After a_2: Kg/x_2, $Kg/x_1, Kx_1/x_2, Kx_2/x_2, \neg K\neg g, \neg Kg, \ldots$
    - New action $merge_g$: $Kg/x_1 \wedge Kg/x_2 \rightarrow Kg$
  - After $merge_g$: Kg, $Kg/x_2, Kg/x_1, Kx_1/x_2, Kx_2/x_2, \neg K\neg g, \ldots$
  - Goal satisfied: Kg

Idea

- Given literal L and tag t, atom KL/t means
  - $K(t_0 \supset L)$: *KL true if t is true* **initially**

Example

- Conformant Problem P:
  - Init: $x_1 \vee x_2, \neg g$
  - Goal: g
  - Actions: $a_1 : x_1 \rightarrow g$, $a_2 : x_2 \rightarrow g$
- Classical Problem $K_{T,M}(P)$:
  - Init: $Kx_1/x_1, Kx_2/x_2, K\neg g, \neg Kg, \neg Kx_1, \neg K\neg x_1, \ldots$
  - After a_1: Kg/x_1, $Kx_1/x_1, Kx_2/x_2, \neg K\neg g, \neg Kg, \ldots$
  - After a_2: Kg/x_2, $Kg/x_1, Kx_1/x_2, Kx_2/x_2, \neg K\neg g, \neg Kg, \ldots$
    - New action $merge_g$: $Kg/x_1 \wedge Kg/x_2 \rightarrow Kg$
  - After $merge_g$: Kg, $Kg/x_2, Kg/x_1, Kx_1/x_2, Kx_2/x_2, \neg K\neg g, \ldots$
  - Goal satisfied: Kg

Idea

- Given literal L and tag t, atom KL/t means
  - $K(t_0 \supset L)$: *KL true if t is true* **initially**

Example

- Conformant Problem P:
  - Init: $x_1 \vee x_2, \neg g$
  - Goal: g
  - Actions: $a_1 : x_1 \rightarrow g$, $a_2 : x_2 \rightarrow g$
- Classical Problem $K_{T,M}(P)$:
  - Init: $Kx_1/x_1, Kx_2/x_2, K\neg g, \neg Kg, \neg Kx_1, \neg K\neg x_1, \ldots$
  - After a_1: Kg/x_1, $Kx_1/x_1, Kx_2/x_2$, $\neg K\neg g$, $\neg Kg, \ldots$
  - After a_2: Kg/x_2, $Kg/x_1, Kx_1/x_2, Kx_2/x_2, \neg K\neg g, \neg Kg, \ldots$
    - New action $merge_g$: $Kg/x_1 \wedge Kg/x_2 \rightarrow Kg$
  - After $merge_g$: Kg, $Kg/x_2, Kg/x_1, Kx_1/x_2, Kx_2/x_2, \neg K\neg g, \ldots$
  - Goal satisfied: Kg

Idea

- Given literal L and tag t, atom KL/t means
  - $K(t_0 \supset L)$: *KL true if t is true* **initially**

Example

- Conformant Problem P:
  - Init: $x_1 \vee x_2, \neg g$
  - Goal: g
  - Actions: $a_1 : x_1 \rightarrow g$, $a_2 : x_2 \rightarrow g$
- Classical Problem $K_{T,M}(P)$:
  - Init: $Kx_1/x_1, Kx_2/x_2, K\neg g, \neg Kg, \neg Kx_1, \neg K\neg x_1, \ldots$
  - After a_1: Kg/x_1, $Kx_1/x_1, Kx_2/x_2$, $\neg K\neg g$, $\neg Kg, \ldots$
  - After a_2: Kg/x_2, $Kg/x_1, Kx_1/x_2, Kx_2/x_2, \neg K\neg g, \neg Kg, \ldots$
    - New action *merge*$_g$: $Kg/x_1 \wedge Kg/x_2 \rightarrow Kg$
  - After *merge*$_g$: Kg, $Kg/x_2, Kg/x_1, Kx_1/x_2, Kx_2/x_2, \neg K\neg g, \ldots$
  - Goal satisfied: Kg

Idea

- Given literal L and tag t, atom KL/t means
  - $K(t_0 \supset L)$: *KL true if t is true* **initially**

Example

- Conformant Problem P:
  - Init: $x_1 \vee x_2, \neg g$
  - Goal: g
  - Actions: $a_1 : x_1 \rightarrow g$, $a_2 : x_2 \rightarrow g$
- Classical Problem $K_{T,M}(P)$:
  - Init: $Kx_1/x_1, Kx_2/x_2, K\neg g, \neg Kg, \neg Kx_1, \neg K\neg x_1, \ldots$
  - After a_1: Kg/x_1, $Kx_1/x_1, Kx_2/x_2$, $\neg K\neg g$, $\neg Kg, \ldots$
  - After a_2: Kg/x_2, $Kg/x_1, Kx_1/x_2, Kx_2/x_2, \neg K\neg g, \neg Kg, \ldots$
    - New action $merge_g$: $Kg/x_1 \wedge Kg/x_2 \rightarrow Kg$
  - After $merge_g$: Kg, $Kg/x_2, Kg/x_1, Kx_1/x_2, Kx_2/x_2, \neg K\neg g, \ldots$
  - Goal satisfied: Kg

Key elements in Translation $K_{T,M}(P)$

- a set T of **tags** t: consistent set of **assumptions** (literals) about the **initial situation** I

$$I \not\models \neg t$$

- a set M of **merges** m: **valid subsets of tags**

$$I \models \bigvee_{L \in m} L$$

- Literals KL/t meaning that L **is true given that initially** t; *i.e.* $K(t_0 \supset L)$

Key elements in Translation $K_{T,M}(P)$

- a set T of **tags** t: consistent set of **assumptions** (literals) about the **initial situation** I

$$I \not\models \neg t$$

- a set M of **merges** m: **valid subsets of tags**

$$I \models \bigvee_{L \in m} L$$

- Literals KL/t meaning that L **is true given that initially** t; *i.e.* $K(t_0 \supset L)$

Example of T, M

Example

Given $I = \{p \vee q, v \vee \neg w\}$, T and M can be:

$$
\begin{aligned}
T &= \{\{\}, p, q, v, \neg w\} & T' &= \{\{\}, \{p, v\}, \{q, v\}, \ldots\} \\
M &= \{\{p, q\}, \{v, \neg w\}\} & M' &= \ldots
\end{aligned}
$$

Translation $K_{T,M}(P)$

For Conformant $P = \langle F, I, O, G \rangle$, $K_{T,M}(P)$ is $\langle F', I', O', G' \rangle$

- **F′**: KL/t for every lit L in F and t in T
- **I′**: KL/t if $I \models (t \supset L)$
- **G′**: KL for $L \in G$
- For every effect t in T and $a : L_1 \wedge \cdots \wedge L_n \rightarrow L$ in O, add to O'
  - $a : KL_1/t \wedge \cdots \wedge KL_n/t \rightarrow KL/t$
  - $a : \neg K\neg L_1/t \wedge \cdots \wedge \neg K\neg L_n/t \rightarrow \neg K\neg L/t$
- **prec** $L \Rightarrow$ **prec** KL
- **Merge** actions in O': for each lit L and merge $m \in M$ with $m = \{t_1, \ldots, t_n\}$

$$merge_{L,m} : KL/t_1 \wedge \ldots \wedge KL/t_n \rightarrow KL$$

Properties of Translation $K_{T,M}$

- If T contains only the empty tag, $K_{T,M}(P)$ reduces to $K_0(P)$
- $K_{T,M}(P)$ is always **sound**

We will see that...

- For suitable choices of T,M translation is **complete**
- . . . and sometimes **polynomial** as well

Intuition of soundness

- Idea:
  - if sequence of actions π makes KL/t true in $K_{T,M}(P)$
  - π makes L true in P over all **trajectories** starting at initial states satisfying t

Theorem (Soundness $K_{T,M}(P)$)

*If π is a **plan that solves the classical** planning problem $K_{T,M}(P)$, then the action sequence π' that results from π by dropping the merge actions is a **plan that solves the conformant** planning problem P.*

Intuition of soundness

- Idea:
  - if sequence of actions π makes KL/t true in $K_{T,M}(P)$
  - π makes L true in P over all **trajectories** starting at initial states satisfying t

Theorem (Soundness $K_{T,M}(P)$)

*If π is a **plan that solves the classical** planning problem $K_{T,M}(P)$, then the action sequence π' that results from π by dropping the merge actions is a **plan that solves the conformant** planning problem P.*

A complete but exponential instance of $K_{T,M}(P)$: K_{s0}

If possible initial states are $s_0^1, \ldots, s_0^n$, scheme K_{s0} is the instance of $K_{T,M}(P)$ with

- $T = \{ \{\}, s_0^1, \ldots, s_0^n \}$
- $M = \{ \{s_0^1, \ldots, s_0^n\} \}$
  i.e., only **one merge** for the disjunction of possible initial states
- **Intuition**: applying actions in K_{s0} keeps track of each fluent for each possible initial states
- This instance is **complete**, but exponential is the number of fluents
  - ...although not a bad conformant planner

A complete but exponential instance of $K_{T,M}(P)$: K_{s0}

If possible initial states are $s_0^1, \ldots, s_0^n$, scheme K_{s0} is the instance of $K_{T,M}(P)$ with

- $T = \{ \{\}, s_0^1, \ldots, s_0^n \}$
- $M = \{ \{s_0^1, \ldots, s_0^n\} \}$
  i.e., only **one merge** for the disjunction of possible initial states

- **Intuition**: applying actions in K_{s0} keeps track of each fluent for each possible initial states

- This instance is **complete**, but exponential is the number of fluents
  - …although not a bad conformant planner

A complete but exponential instance of $K_{T,M}(P)$: K_{s0}

If possible initial states are $s_0^1, \ldots, s_0^n$, scheme K_{s0} is the instance of $K_{T,M}(P)$ with

- $T = \{ \{\}, s_0^1, \ldots, s_0^n \}$
- $M = \{ \{s_0^1, \ldots, s_0^n\} \}$
  i.e., only **one merge** for the disjunction of possible initial states

- **Intuition**: applying actions in K_{s0} keeps track of each fluent for each possible initial states

- This instance is **complete**, but exponential is the number of fluents
  - ...although not a bad conformant planner

Performance of K_{s0} + FF

		Planners exec time (s)			
Problem	$\#S_0$	K_{s0}	KP	POND	CFF
Bomb-10-1	1k	648,9	0	1	0
Bomb-10-5	1k	2795,4	0,1	3	0
Bomb-10-10	1k	5568,4	0,1	8	0
Bomb-20-1	1M	$> 1.8G$	0,1	4139	0
Sqr-4-16	4	0,3	fail	1131	13,1
Sqr-4-24	4	1,6	fail	$> 2h$	321
Sqr-4-48	4	57,5	fail	$> 2h$	$> 2h$
Sortnet-6	64	2,2	fail	2,1	fail
Sortnet-7	128	27,9	fail	17,98	fail
Sortnet-8	256	$> 1.8G$	fail	907,1	fail

Translation time included in all tables.

Road to Complete but Compact Translations

Theorem

Scheme $K_{T,M}$ is ***complete*** *if for every precondition and goal literal L in P, there is a merge $m = t_1, \ldots, t_n$ that* ***covers*** *L*

A merge m covers L if for all t_i in m, t_i **hits*** $C_I(L)$, the set of clauses in I **relevant** to L

Observation: When such merges can be generated in poly-time, then can have a poly-size instance of $K_{T,M}$ that is complete

Hitting & Hitting*

- t **hits** a set of clauses S if for each clause c in S, there is a literal $L' \in c$ such that
$$L' \in t$$
- t **hits*** a set of clauses S if for each clause c in S, there is a literal $L' \in c$ such that
$$I \models t \supset L'$$

Relevance

- $L \longrightarrow L'$ in P, read as 'L **is relevant to** L''
  1. $L \longrightarrow L$
  2. $L \longrightarrow L'$ if $a : C \rightarrow L'$ in P with $L \in C$
  3. $L \longrightarrow L'$ if $L \longrightarrow L''$ and $L'' \longrightarrow L'$
  4. $L \longrightarrow L'$ if $L \longrightarrow \neg L''$ and $L'' \longrightarrow \neg L'$
- $L \longrightarrow L'$: uncertainty in L affects L'

Conformant Width

- Clause C is **relevant** to L if **all** literals in C are relevant to L
- $C_I(L)$ = set of clauses in I relevant to L, with tautologies $L' \vee \neg L'$ when both relevant to L

Definition

$width(L)$ = min number of clauses in $C_I(L)$ such that any t hitting those clauses, hits* **all** $C_I(L)$

Definition

$width(P)$ = max width(L) over **all** preconds and goals L

Conformant Width: intuitions

- For each L, goal or prec, we want to achieve KL
- It is **not** necessary to deal with all relevant clauses $C_I(L)$
  - **some** of them are **enough** for deciding the others
- How many? $width(L)$

Some consequences:

- $width(P)$ remains the same if we **copy** the same problem and put all together
  - so, we can deal with a **group of simple-and-decoupled subproblems**
- Width is **worst-case**: sometimes the problem is easier

Conformant Width and Tractability

- If width(L) $\leq i$ for fixed i, a merge that **covers** L generated in poly-time
- If width(P) $\leq i$ for fixed i, a poly-size and complete translation follows from theorem above
- In paper, translation K_i formulated that is **poly** for fixed i, and **complete** if $width(P) \leq i$
- Current conformant benchmarks have conformant **width 1**, except: blocks, sortnet, adder
- Conformant Planner **T0** $= K_1(P)+$ FF best at IPC-2006

Conformant Width and Tractability

- If width(L) $\leq i$ for fixed i, a merge that **covers** L generated in poly-time
- If width(P) $\leq i$ for fixed i, a poly-size and complete translation follows from theorem above
- In paper, translation K_i formulated that is **poly** for fixed i, and **complete** if $width(P) \leq i$
- Current conformant benchmarks have conformant **width 1**, except: blocks, sortnet, adder
- Conformant Planner **T0** $= K_1(P)+$ FF best at IPC-2006

Conformant Width and Tractability

- If width(L) $\leq i$ for fixed i, a merge that **covers** L generated in poly-time
- If width(P) $\leq i$ for fixed i, a poly-size and complete translation follows from theorem above
- In paper, translation K_i formulated that is **poly** for fixed i, and **complete** if $width(P) \leq i$
- Current conformant benchmarks have conformant **width 1**, except: blocks, sortnet, adder
- Conformant Planner **T0** $= K_1(P) +$ FF best at IPC-2006

T0 optimizations

- **Non-uniform tags**: tags for L are only literals in $C_I(L)$
- Remove from PDDL KL/t and cond-effects that does not affect merge results
- For invariant oneof$(x_1, \ldots, x_n)$: keep Kx_i updated. Example:

$$K\neg x_1 \wedge \ldots \wedge K\neg x_{n-1} \rightarrow Kx_n$$

- **Thanks FF** for
  - accepting big grounded PDDLs
  - dealing with lots of conditional effects

Translating P into $K_1(P)$

	P		Translation	$K_1(P)$	
Problem	#Fluents	#Effects	time (secs)	#Fluents	#Effects
Bomb-100-100	402	40200	1,36	1304	151700
Sqr-64-ctr	130	504	2,34	16644	58980
Sqr-120-ctr	242	952	12,32	58084	204692
Logistics-4-10-10	872	7640	1,44	1904	16740
1-Dispose-8-3	486	1984	26,72	76236	339410
Look-n-Grab-8-1-1	356	2220	4,03	9160	151630

- Actually, after some simplifications made for T0 to the PDDL
- Translation is not the bottleneck

Total time of $K_1(P)$ + FF (translation + search)

	$T0$		KP		CFF	
problem	time (sec)	len	time (sec)	len	time (sec)	len
Bomb-100-60	5,6	140	4,54	140	9,38	140
Bomb-50-50	1,11	50	0,96	50	0,1	50
Sqr-8-ctr	0,07	26	0,05	0	70,63	50
Sqr-12-ctr	0,1	32	0,07	32	$> 2h$	
Sqr-64-ctr	10,68	188	1,66	188	$> 2h$	
Sqr-120-ctr	$> 1.8G$		13,23	356	$> 1.8G$	
Sqr-4-16-ctr	0,2	86	fail		13,13	140
Sqr-4-20-ctr	0,51	128	fail		73,73	214
Sqr-4-64-ctr	267,3	1118	fail		$> 2h$	
Log-3-10-10	3,42	109	2,67	109	4,67	108
Log-4-10-10	6,52	125	3,07	125	4,36	121
Comm-24	0,7	418	fail		37,52	359
Comm-25	0,84	453	fail		56,13	389

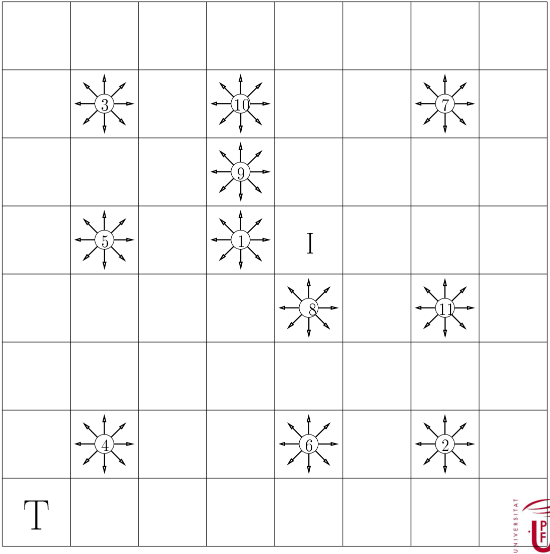
T0: new domains

	T0		*KP*	
problem	time	len	time	len
Push-to-4-1*	0,16	64	$> 1.8G$	
Push-to-4-2*	0,3	67	0,16	69
Push-to-4-3*	0,48	83	0,22	71
Push-to-8-3	1153,16	395	10,12	291
Push-to-12-1	$> 2h$		$> 1.8G$	
1-Dispose-8-1	124,5	1268	fail	
1-Dispose-8-2	699,11	1268	fail	
1-Dispose-8-3	1296,02	1268	fail	
1-Dispose-12-1	$> 2h$		fail	
Look-n-Grab-8-1-1	45,27	212	fail	
Look-n-Grab-8-1-2	84,04	88	fail	

- * = problems solved by CFF
- **Push-to**: goal is hold object. Pick-up at two of corners
- **1-Dispose**: object to trash, but hand has capacity one.

Look-n-grab 8x8

- **Actions**: move, *look-and-grab*, putdown
- **Init**: object can be anywhere.
- **Goal**: object at Trash
- Obj **get lost** when pickup with handfull, so have to **visit Trash after each pickup**
- Plan **len**: 212 acts
- **Time**: 46s



Width of some problems

- Sqr-center. Init = oneof($x_1, \ldots, x_n$), oneof($y_1, \ldots, y_n$). Goal = x_{center}, y_{center}
  - Has **width 1** because x_i not relevant to y_j
- Blocks, with a **magic action** to achieve the goal
  - Trivial (solved by K_0) but width high

Summary

- A **general** $K_{T,M}$ translation scheme for mapping from conformant P into classical P'
- A number of interesting **instances**: K_0, K_{s0}, K_i
- A notion of **conformant width** that distinguishes hard from simple conformant problems
- Translation scheme K_i that is **always polynomial** and complete if conformant width $\leq i$
- Planner **T0** $= K_1(P) +$ FF
- On going work: as a base for an **action selection mechanism** for Contingent Planning

Mapping to Propositional Logic (1)

Let T_P, a **satplan**-like propositional theory for the conformant problem P with fixed **horizon** n

- T_P **encodes all the executions** starting at **some** possible initial state
- Thus, **SAT call** would give a plan for **one** initial state: **optimistic** plan, *not what we want*

Mapping to Propositional Logic (2)

Two ideas for Conformant **Optimal** Planning

- **Search** over plans space (ICAPS-05), checking

$$\text{plan } \pi \text{ is conformant } \Leftrightarrow \#\mathit{Models}(T_P \,|\, \pi) = \#\text{init states of } P$$

- Create **new formula** T'_P encoding all possible plans and call a SAT solver **once** upon T'_P (CAEPIA-05)

$$T'_P = \bigwedge_{s_0 \in \mathit{Init}} \mathit{project}[\; T_P \,|\, s_0 \;;\; \mathit{Actions}\;]$$

How?

- **Knowledge Compilation** to *d-DNNF* allowed us to do **model counting** and **projection** feasible
- *d-DNNF* is a normal form related to **OBDD**

Mapping to Propositional Logic (2)

Two ideas for Conformant **Optimal** Planning

- **Search** over plans space (ICAPS-05), checking

$$\text{plan } \pi \text{ is conformant } \Leftrightarrow \#Models(T_P \mid \pi) = \#\text{init states of } P$$

- Create **new formula** T'_P encoding all possible plans and call a SAT solver **once** upon T'_P (CAEPIA-05)

$$T'_P = \bigwedge_{s_0 \in Init} project[\, T_P \mid s_0 \;;\; Actions \,]$$

How?

- **Knowledge Compilation** to *d-DNNF* allowed us to do **model counting** and **projection** feasible
- *d-DNNF* is a normal form related to **OBDD**