Translation-based approaches to Conformant and Contingent Planning

Alexandre Albore and Héctor Palacios

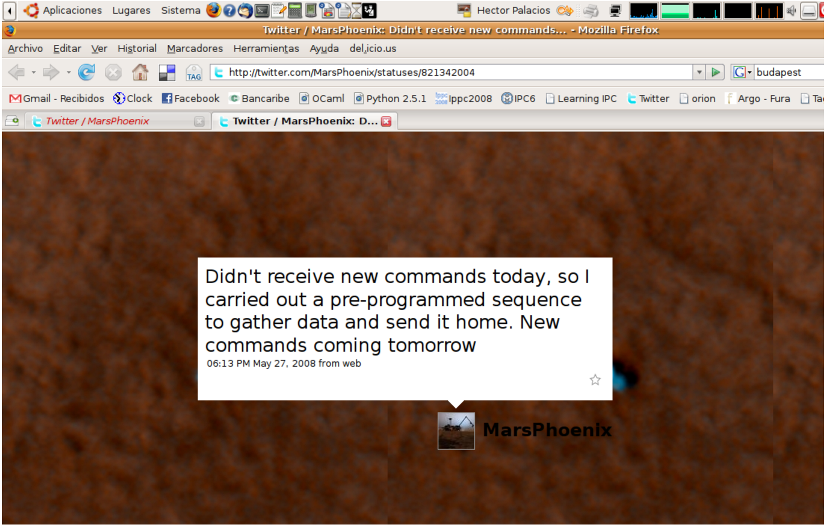
Universitat Pompeu Fabra & Universitat Carlos III de Madrid

ICAPS – June 2011

Get it real!

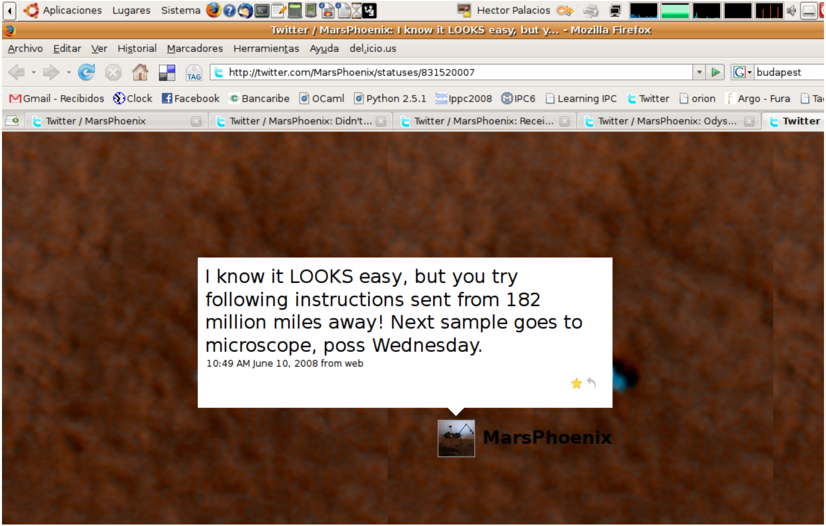
- Let's get real, but principled!

Get it real!



- Let's get real, but principled!

Get it real!



- Let's get real, but principled!

Problem addressed in this tutorial

- Planning is the

  *Problem of finding the **actions** that **achieve a goal**, starting from an **initial situation***

- Classical planning assume **complete information** on

  initial state, actions effects, . . .

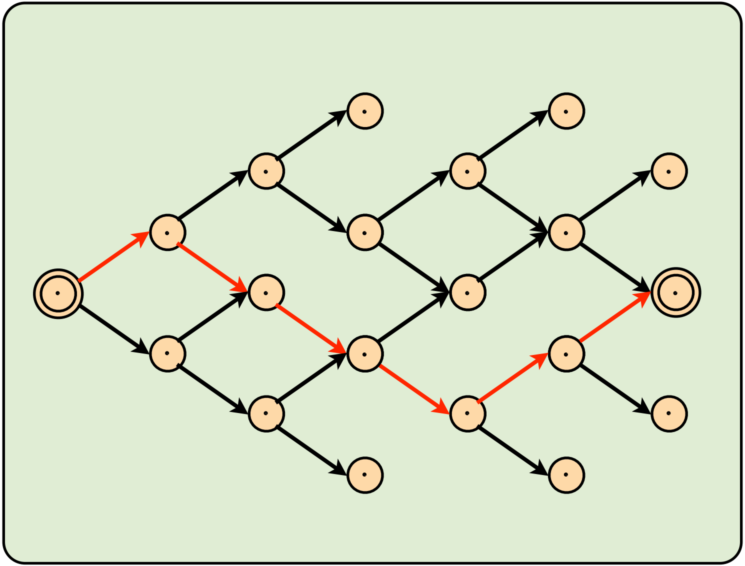
- Conformant Planning

  incomplete information on init state and effects but still one sequence of actions

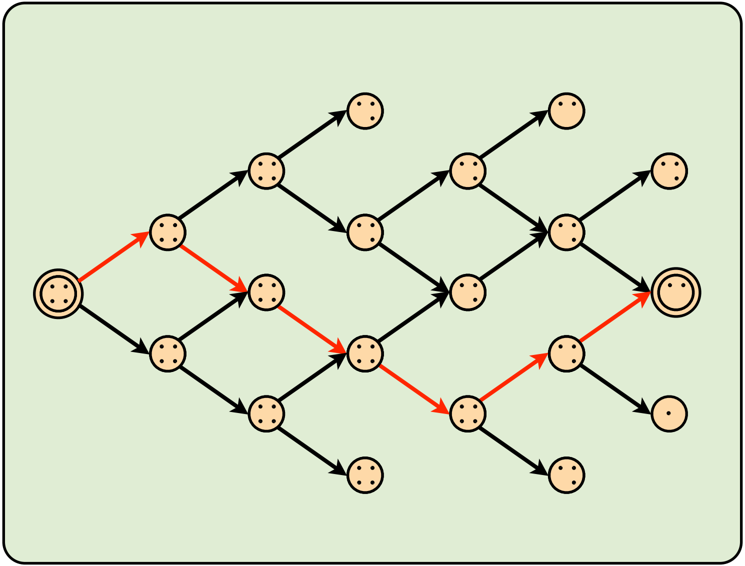
- Contingent Planning is like Conformant but

  allow observations. Plans are not sequences anymore.

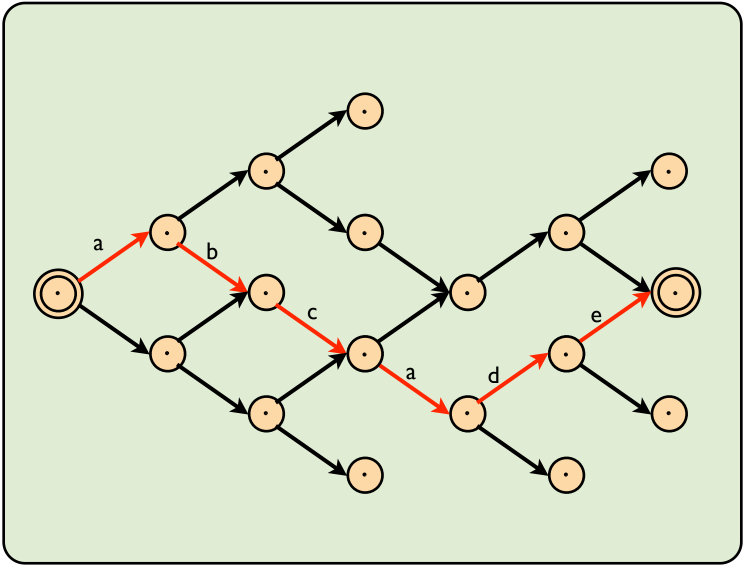
Classical Planning



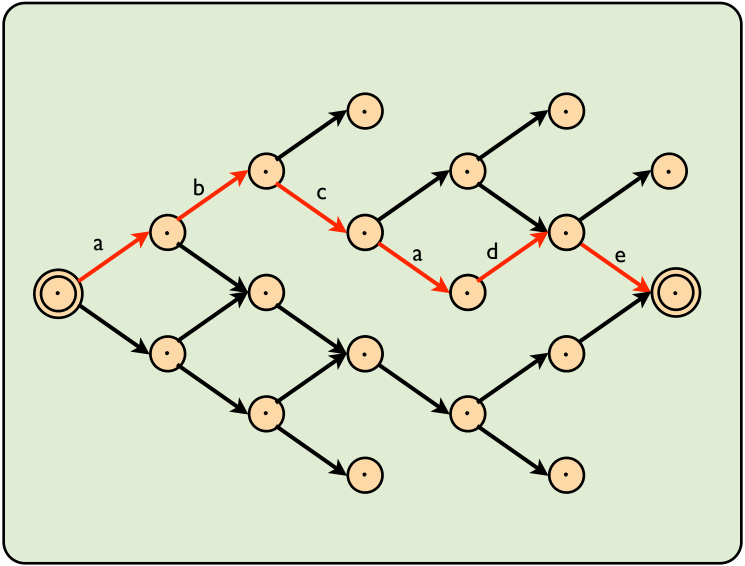
Conformant Planning



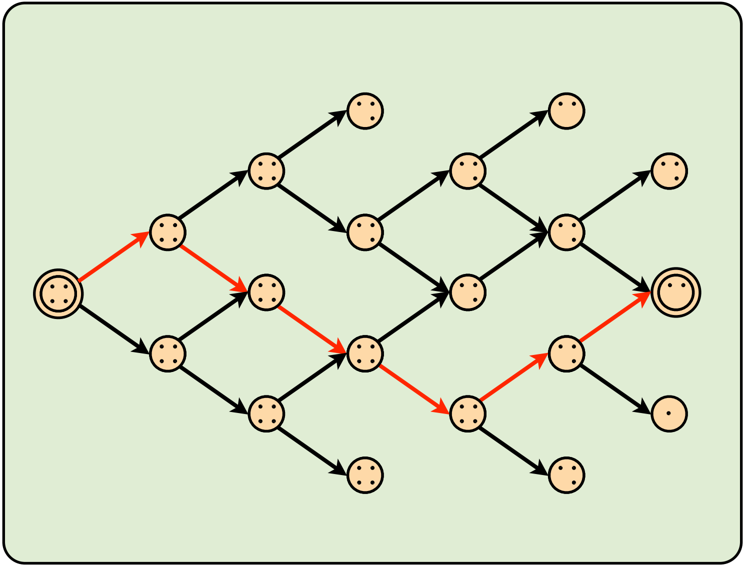
Classical problem for one state of a Conformant (I)



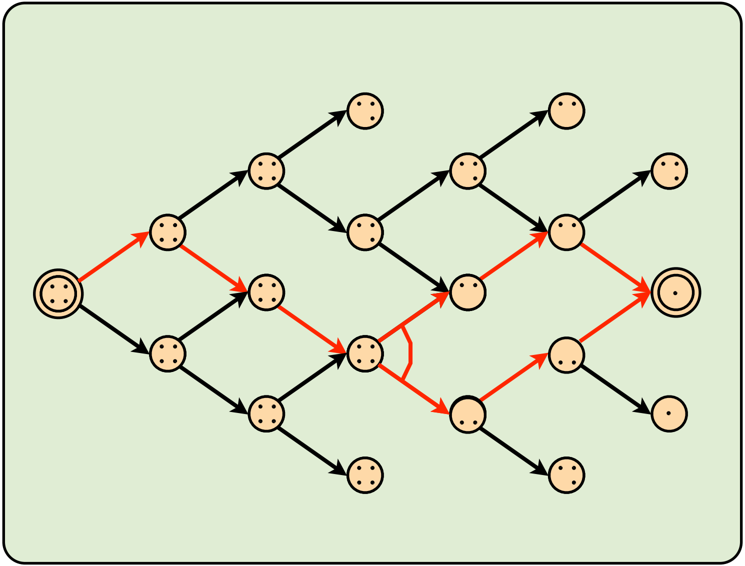
Classical problem for one state of a Conformant (II)



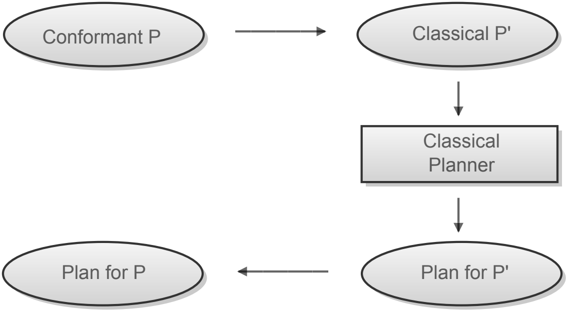
Conformant Planning (again)



Contingent Planning



Translation to Classical planning



Features

- Conformant plans are sequences like classical ones
- but Contingent are not. Something else is needed

Classical Planning

Problem of finding a **sequence** of **deterministic** actions that **achieves a goal**, starting from a **given** initial state.

- action cost = 1
- no observations

Expressed in **high-level** language

- Init: p, q
- Goal: g
- Actions:
  - a Precondition: p. Effect: r
  - b Precondition: q. Effect: $r \rightarrow g$
  - c Precondition: q. Effect: $\neg q \wedge r$
- Plan: a, b

Classical Planning

Problem of finding a **sequence** of **deterministic** actions that **achieves a goal**, starting from a **given** initial state.

- action cost = 1
- no observations

Expressed in **high-level** language

- Init: p, q
- Goal: g
- Actions:
  - `a` Precondition: p. Effect: r
  - `b` Precondition: q. Effect: $r \rightarrow g$
  - `c` Precondition: q. Effect: $\neg q \wedge r$
- Plan: `a`, `b`

Classical Planning Syntax

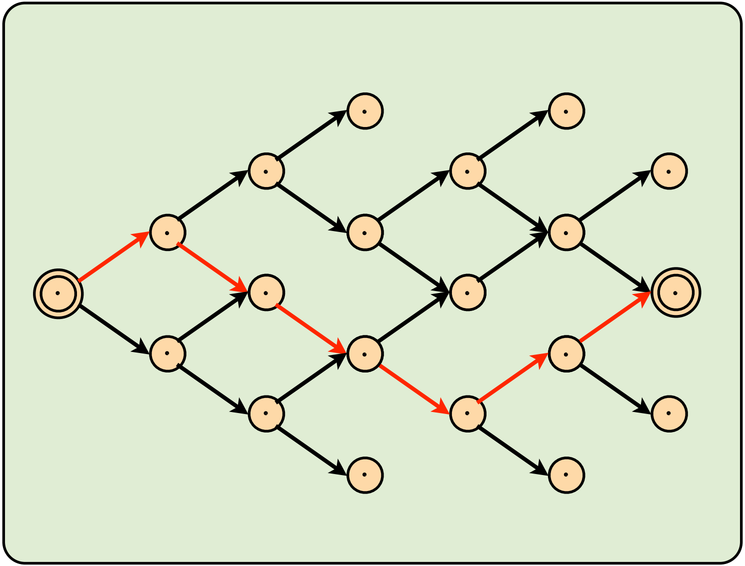
Classical planning problems P are tuples of the form $P = \langle F, I, O, G \rangle$ where

- F: **fluent** symbols in the problem
- I: set of fluents true in the **initial situation**
- O: set of operators or actions. Every **action** a has
  - a **precondition** $Pre(a)$ given by a set of fluents
  - a set of **conditional effects** $C \rightarrow L$ where C is a set of fluent literals and L is a single fluent literal.
- G: set of fluents defining the **goal**

Classical Planning Model

- **Languages** such as Strips, ADL, PDDL, ..., represent **models** in compact form
- 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** $c(a, s) = 1$
- Given a problem P, **states** of its corresponding model are **set of fluents** of P
- Their **solutions** (plans) are sequences of applicable actions that map s_0 into S_G

Classical Planning



State-of-the-art Classical Planning

Two main **approaches** currently:

- **Heuristic-search based** (McDermott, 1996; Bonet *et al.*, 1997)
- **SAT-based** (Kautz & Selman, 1992)
- The good news: **classical planning works**
  - *heuristic search-based solve large problems very fast (non-optimally)*
- Not so good: **limitations**
  - *No **Uncertainty** (no probabilities)*
  - *No **Incomplete Information** (no sensing)*

State-of-the-art Classical Planning

Two main **approaches** currently:

- **Heuristic-search based** (McDermott, 1996; Bonet *et al.*, 1997)
- **SAT-based** (Kautz & Selman, 1992)

- The good news: **classical planning works**
  - *heuristic search-based solve large problems very fast (non-optimally)*
- Not so good: **limitations**
  - *No* ***Uncertainty*** *(no probabilities)*
  - *No* ***Incomplete Information*** *(no sensing)*

Conformant Planning

- **Extend** classical planning **model** to
  - **incomplete information about initial state** and
  - non-deterministic actions
- **Conformant plan**: a **sequence** of actions that achieves the goal for **any possible** initial state and state transition
- Harder than classical planning

  verifying if sequence of actions is a conformant plan is **hard**
- For polynomial length, classical planning is *NP*-complete, but conformant planning is Σ_2^p-complete = NP^{NP}-complete

Examples

- **Cleaning robot**: there maybe debris in a grid room. A robot can collect debris in a cell. A conformant plan for cleaning the room is to collect debris in all the cells.
- **Heal a patient**: patient has a possible set of pathologies. A sequence of treatment actions that cures a patient for any of such pathologies is a conformant plan.

- Init: $illness_1 \lor illness_2, alive$
- Goal: $healthy, alive$
- Actions:

$treat_1$	Precondition: true. Effect: $illness_1 \rightarrow healthy$
$treat_2$	Precondition: true. Effect: $illness_2 \rightarrow healthy$
$treat_3$	Precondition: true. Effect: $illness_2 \rightarrow healthy$, $\neg illness_2 \rightarrow \neg alive$

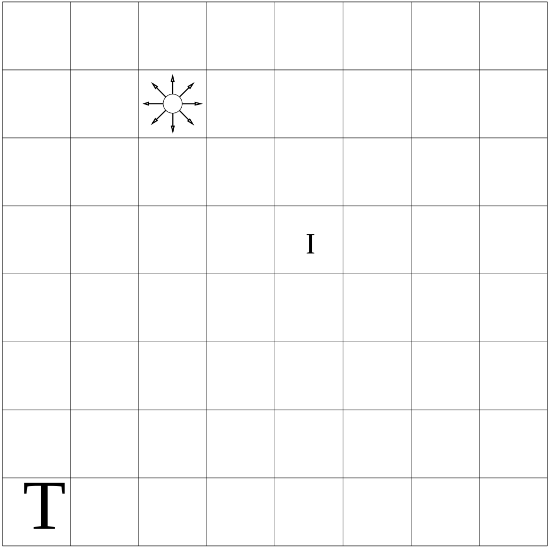
Omit precondition if true

- Init: $illness_1 \vee illness_2, alive$
- Goal: $healthy, alive$
- Actions:

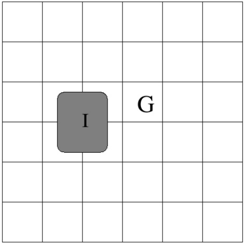
$$\begin{array}{l} \text{treat}_1: illness_1 \rightarrow healthy \\ \text{treat}_2: illness_2 \rightarrow healthy \\ \text{treat}_3: illness_2 \rightarrow healthy, \\ \qquad\quad \neg illness_2 \rightarrow \neg alive \end{array}$$

Look-n-grab 8x8

- **Actions**: `move`, `look-and-grab`, `putdown`
- **Init**: object can be anywhere.
- **Goal**: object at **Trash**
- Robot should **visit Trash after each look-and-grab**



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 *n*x*n*

- 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)

Why it's important?

- What we **really** want is observations, probabilities, time, resources, etc, yet

- *Better* Conformant Planning leads to *better* **Planning with Observations** (contingent)
  - **Contingent-FF** uses Conformant-FF's heuristic
  - **POND** does both: conformant and contingent
  - **CLG** for **planning with observations** presented in this tutorial
- Conformant planning is **relevant** to any planning setting where **actions** are applied to a **set of possible configurations.**
- **Classical planning is symbolic reachability, and conformant is reachability between set of configurations.**

Why it's important?

- What we **really** want is observations, probabilities, time, resources, etc, yet
- *Better* Conformant Planning leads to *better* **Planning with Observations** (contingent)
  - **Contingent-FF** uses Conformant-FF's heuristic
  - **POND** does both: conformant and contingent
  - **CLG** for **planning with observations** presented in this tutorial
- Conformant planning is **relevant** to any planning setting where **actions** are applied to a **set of possible configurations.**
- **Classical planning is symbolic reachability, and conformant is reachability between set of configurations.**

Why it's important?

- What we **really** want is observations, probabilities, time, resources, etc, yet
- *Better* Conformant Planning leads to *better* **Planning with Observations** (contingent)
  - **Contingent-FF** uses Conformant-FF's heuristic
  - **POND** does both: conformant and contingent
  - **CLG** for **planning with observations** presented in this tutorial
- Conformant planning is **relevant** to any planning setting where **actions** are applied to a **set of possible configurations.**
- **Classical planning is symbolic reachability, and conformant is reachability between set of configurations.**

Why it's important?

- What we **really** want is observations, probabilities, time, resources, etc, yet
- *Better* Conformant Planning leads to *better* **Planning with Observations** (contingent)
  - **Contingent-FF** uses Conformant-FF's heuristic
  - **POND** does both: conformant and contingent
  - **CLG** for **planning with observations** presented in this tutorial
- Conformant planning is **relevant** to any planning setting where **actions** are applied to a **set of possible configurations.**
- **Classical planning is symbolic reachability, and conformant is reachability between set of configurations.**

Conformant Planning Syntax

Deterministic conformant planning problems P are tuples $P = \langle F, I, O, G \rangle$ where

- F: **fluent** symbols in the problem
- I: set of **clauses** over F defining the **initial situation**
- O: set of operators or actions. Every **action** a has
  - a **precondition** $Pre(a)$ given by a set of fluents
  - a set of **conditional effects** $C \rightarrow L$ where C is a set of fluent literals and L is a single fluent literal.
- G: set of literals over F defining the (conjunctive) **goal**

Conformant Planning: Semantic

- a **set** $S_0 \subseteq S$ of possible initial states
- a set of possible goals $S_G \subseteq S$ st $s_g \in S_G$ iff $G \subseteq s_g$
- actions $A(s) \subseteq A$ applicable in each $s \in S$
- a deterministic state transition function F s.t. $F(a, s) = s'$, the state resulting of applying a on s

– *a conformant plan is an **action sequence** that maps **each** initial state s_0 in S_0 into some goal state s_g*

– *It can be cast as a **path-finding problem over belief-states***

Conformant Planning: Semantic

- a **set** $S_0 \subseteq S$ of possible initial states
- a set of possible goals $S_G \subseteq S$ st $s_g \in S_G$ iff $G \subseteq s_g$
- actions $A(s) \subseteq A$ applicable in each $s \in S$
- a deterministic state transition function F s.t. $F(a, s) = s'$, the state resulting of applying a on s

– *a conformant plan is an **action sequence** that maps **each** initial state s_0 in S_0 into some goal state s_g*

– *It can be cast as a **path-finding problem over belief-states***

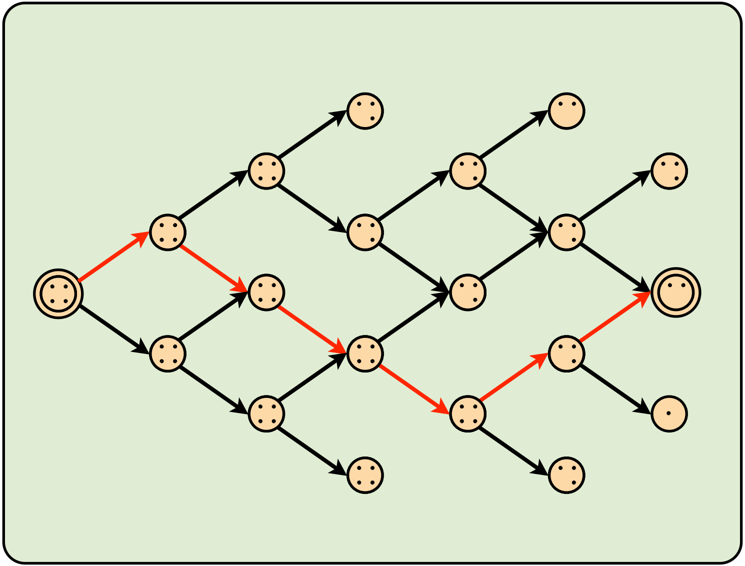
Conformant Planning: Semantic

- a **set** $S_0 \subseteq S$ of possible initial states
- a set of possible goals $S_G \subseteq S$ st $s_g \in S_G$ iff $G \subseteq s_g$
- actions $A(s) \subseteq A$ applicable in each $s \in S$
- a deterministic state transition function F s.t. $F(a, s) = s'$, the state resulting of applying a on s

– *a conformant plan is an **action sequence** that maps **each** initial state s_0 in S_0 into some goal state s_g*

– *It can be cast as a **path-finding problem over belief-states***

Conformant Planning



Belief space search

- Almost all previous approaches to conformant planning use **search** on graph whose nodes are **set of possible states** (belief states)

  Key issues:
  - **Representation**: compact and efficient
  - **Heuristic**: for guiding the search

Roadmap of First Part

- Basic Translation Scheme $K_0(P)$
- General Translation Scheme $K_{T,M}(P)$
- **Complete** Instances of $K_{T,M}(P)$
- **Conformant Width** of P bounds complexity of translation
- **Poly** translation K_i that is complete if **width** $\leq i$
- **Width** of some benchmarks
- Creating a **planner** using $K_{T,M}(P)$
- **Other** translation-based algorithms

Spoilers!

- Conformant problems mapped into classical ones
- **Plans** obtained using an **off-the-shelf classical planner**
- Translation exponential in worst case

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		**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: $C \rightarrow L$	$\Rightarrow$	$\begin{cases} a: & KC \rightarrow KL \\ a: & \neg K\neg C \rightarrow \neg K\neg L \end{cases}$

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 $C \to L$
- 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: $C \to L$	$\Rightarrow$	$\left\{ \begin{array}{llll} a: & KC & \to & KL \\ a: & \neg K\neg C & \to & \neg K\neg L \end{array} \right.$

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: $C \rightarrow L$	$\Rightarrow$	$\left\{ \begin{array}{llll} a: & KC & \rightarrow & KL \\ a: & \neg K\neg C & \rightarrow & \neg K\neg L \end{array} \right.$

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 $C \to L$
- 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: $C \to L$	$\Rightarrow$	$\left\{ \begin{array}{lrcl} a: & KC & \to & KL \\ a: & K\neg C & \to & \emptyset \\ a: & \neg K\neg C & \to & \neg K\neg L \end{array} \right.$

K_0 example

Problem P with

- Init: $p \vee q, r$, Goal: g
- Actions:

$$a : p \rightarrow q$$
$$b : q \rightarrow g$$
$$c : r \rightarrow q$$

- $< a, b >$ and $< c, b >$ are conformant plans.

$K_0(P)$ is:

- Init: Kr, Goal: Kg
- Actions:

$$a : Kp \rightarrow Kq,\ a : \neg K\neg p \rightarrow \neg K\neg q$$
$$b : Kq \rightarrow Kg,\ b : \neg K\neg q \rightarrow \neg K\neg g$$
$$c : Kr \rightarrow Kq,\ c : \neg K\neg r \rightarrow \neg K\neg q$$

- $< c, b >$ is a classical plan, but $< a, b >$ is not.

K_0 example

Problem P with

- Init: $p \vee q, r$, Goal: g
- Actions:

$$a : p \rightarrow q$$
$$b : q \rightarrow g$$
$$c : r \rightarrow q$$

- $< a, b >$ and $< c, b >$ are conformant plans.

$K_0(P)$ is:

- Init: Kr, Goal: Kg
- Actions:

$$a : Kp \rightarrow Kq,\ a : \neg K \neg p \rightarrow \neg K \neg q$$
$$b : Kq \rightarrow Kg,\ b : \neg K \neg q \rightarrow \neg K \neg g$$
$$c : Kr \rightarrow Kq,\ c : \neg K \neg r \rightarrow \neg K \neg q$$

- $< c, b >$ is a classical plan, but $< a, b >$ is not.

K_0 example. Cancellation rules

Problem P with

- Init: $p \vee q, r, s$, Goal: t, g
- Actions:

$$a : p \rightarrow \neg r,\ a : s \rightarrow t$$
$$b : r \rightarrow g$$

- $< b, a >$ is a conformant plan but $< a, b >$ is not.

$K_0(P)$ but without cancellation rules is:

- Init: $Kr, Ks, \neg Kp, \neg Kq, \neg K\neg p, \neg K\neg q$, Goal: Kt, Kg
- Actions:

$$a : Kp \rightarrow K\neg r,\ a : Ks \rightarrow Kt,$$

$$b : Kr \rightarrow Kg,$$

- $< a, b >$ and $< b, a >$ are both classical plans. ERROR
- $< b, a >$ is a classical plan but $< a, b >$ is not.

K_0 example. Cancellation rules

Problem P with

- Init: $p \vee q, r, s$, Goal: t, g
- Actions:

$$a : p \rightarrow \neg r,\ a : s \rightarrow t$$
$$b : r \rightarrow g$$

- $< b, a >$ is a conformant plan but $< a, b >$ is not.

$K_0(P)$ but without cancellation rules is:

- Init: $Kr, Ks, \neg Kp, \neg Kq, \neg K\neg p, \neg K\neg q$, Goal: Kt, Kg
- Actions:

$$a : Kp \rightarrow K\neg r,\ a : Ks \rightarrow Kt,$$

$$b : Kr \rightarrow Kg,$$

- $< a, b >$ and $< b, a >$ are both classical plans. ERROR
- $< b, a >$ is a classical plan but $< a, b >$ is not.

K_0 example. Cancellation rules

Problem P with

- Init: $p \vee q, r, s$, Goal: t, g
- Actions:

  $a : p \rightarrow \neg r$, $a : s \rightarrow t$
  $b : r \rightarrow g$

- $< b, a >$ is a conformant plan but $< a, b >$ is not.

$K_0(P)$ but with cancellation rules is:

- Init: $Kr, Ks, \neg Kp, \neg Kq, \neg K\neg p, \neg K\neg q$, Goal: Kt, Kg
- Actions:

  $a : Kp \rightarrow K\neg r$, $a : Ks \rightarrow Kt$,
  $a : \neg K\neg p \rightarrow \neg K\neg r$, $a : \neg K\neg s \rightarrow \neg K\neg t$
  $b : Kr \rightarrow Kg$, $b : \neg K\neg r \rightarrow \neg K\neg g$

- $< a, b >$ and $< b, a >$ are both classical plans. ERROR
- $< b, a >$ is a classical plan but $< a, b >$ is not.

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 **too incomplete**
  - often $K_0(P)$ will have no solution while P does
  - works only when **uncertainty is irrelevant**

- 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 **too incomplete**
  - often $K_0(P)$ will have no solution while P does
  - works only when **uncertainty is irrelevant**

- Extension $K_{T,M}(P)$ we present now **can** be both **complete and polynomial**

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**

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

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

Intuition of merge actions

- Init: Candy **in hall** (h) $\vee$ Candy **in room** (r)
- Goal: **Hold the candy** (c)
- Apply `pick-from-hall`, get Kc/h
- Apply `pick-from-room`, get Kc/r
- Then, **for sure**, holding the candy (Kc) from merge $Kc/h \wedge Kc/r \rightarrow Kc$

Intuition of merge actions

- Init: Candy **in hall** (h) $\vee$ Candy **in room** (r)
- Goal: **Hold the candy** (c)
- Apply `pick-from-hall`, get Kc/h
- Apply `pick-from-room`, get Kc/r
- Then, **for sure**, holding the candy (Kc) from merge $Kc/h \wedge Kc/r \rightarrow Kc$

Intuition of merge actions

- Init: Candy **in hall** (h) $\vee$ Candy **in room** (r)
- Goal: **Hold the candy** (c)
- Apply `pick-from-hall`, get Kc/h
- Apply `pick-from-room`, get Kc/r
- Then, **for sure**, holding the candy (Kc) from merge $Kc/h \wedge Kc/r \rightarrow Kc$

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

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

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

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

Conformant P	$\Rightarrow$	**Classical** $K_{T,M}(P)$
$\langle F, I, O, G \rangle$	$\Rightarrow$	$\langle F', I', O', G' \rangle$
Fluent L	$\Rightarrow$	$KL/t, K\neg L/t$ *(for all tags t)*
Init: known lit L	$\Rightarrow$	$KL \wedge \neg K\neg L$
if $I \models (t \supset L)$	$\Rightarrow$	$KL/t \wedge \neg K\neg L/t$
Goal L	$\Rightarrow$	KL
Operator a has prec L	$\Rightarrow$	a has prec KL
Operator a: $C \rightarrow L$	$\Rightarrow$	$\left\{ \begin{array}{llcl} & \textit{for all tags } t & & \\ a: & KC/t & \rightarrow & KL/t \\ a: & \neg K\neg C/t & \rightarrow & \neg K\neg L/t \end{array} \right.$

For each lit L and merge $m \in M$ with $m = \{t_1, \ldots, t_n\}$, add to O':

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

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

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

Conformant P		**Classical** $K_{T,M}(P)$
$\langle F, I, O, G \rangle$	$\Rightarrow$	$\langle F', I', O', G' \rangle$
Fluent L	$\Rightarrow$	$KL/t, K\neg L/t$ *(for all tags t)*
Init: known lit L	$\Rightarrow$	$KL \wedge \neg K \neg L$
if $I \models (t \supset L)$	$\Rightarrow$	$KL/t \wedge \neg K \neg L/t$
Goal L	$\Rightarrow$	KL
Operator a has prec L	$\Rightarrow$	a has prec KL
Operator a: $C \rightarrow L$	$\Rightarrow$	$\left\{ \begin{array}{llll} & \textit{for all tags } t & & \\ a: & KC/t & \rightarrow & KL/t \\ a: & \neg K \neg C/t & \rightarrow & \neg K \neg L/t \end{array} \right.$

For each lit L and merge $m \in M$ with $m = \{t_1, \ldots, t_n\}$, add to O':

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

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

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

Conformant P	$\Rightarrow$	**Classical** $K_{T,M}(P)$
$\langle F, I, O, G \rangle$	$\Rightarrow$	$\langle F', I', O', G' \rangle$
Fluent L	$\Rightarrow$	$KL/t, K\neg L/t$ *(for all tags t)*
Init: known lit L	$\Rightarrow$	$KL \wedge \neg K \neg L$
if $I \models (t \supset L)$	$\Rightarrow$	$KL/t \wedge \neg K \neg L/t$
Goal L	$\Rightarrow$	KL
Operator a has prec L	$\Rightarrow$	a has prec KL
Operator a: $C \rightarrow L$	$\Rightarrow$	$\left\{ \begin{array}{llcl} & \textit{for all tags } t & & \\ a: & KC/t & \rightarrow & KL/t \\ a: & \neg K\neg C/t & \rightarrow & \neg K \neg L/t \end{array} \right.$

For each lit L and merge $m \in M$ with $m = \{t_1, \ldots, t_n\}$, add to O':

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

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

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

Conformant P	$\Rightarrow$	**Classical** $K_{T,M}(P)$
$\langle F, I, O, G \rangle$	$\Rightarrow$	$\langle F', I', O', G' \rangle$
Fluent L	$\Rightarrow$	$KL/t, K\neg L/t$ *(for all tags t)*
Init: known lit L	$\Rightarrow$	$KL \wedge \neg K\neg L$
if $I \models (t \supset L)$	$\Rightarrow$	$KL/t \wedge \neg K\neg L/t$
Goal L	$\Rightarrow$	KL
Operator a has prec L	$\Rightarrow$	a has prec KL
Operator a: $C \rightarrow L$	$\Rightarrow$	$\left\{ \begin{array}{llcl} & \textit{for all tags } t & & \\ a: & KC/t & \rightarrow & KL/t \\ a: & \neg K\neg C/t & \rightarrow & \neg K\neg L/t \end{array} \right.$

For each lit L and merge $m \in M$ with $m = \{t_1, \ldots, t_n\}$, add to O':

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

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

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

Conformant P	$\Rightarrow$	**Classical** $K_{T,M}(P)$
$\langle F, I, O, G \rangle$	$\Rightarrow$	$\langle F', I', O', G' \rangle$
Fluent L	$\Rightarrow$	$KL/t, K\neg L/t$ *(for all tags t)*
Init: known lit L	$\Rightarrow$	$KL \wedge \neg K \neg L$
if $I \models (t \supset L)$	$\Rightarrow$	$KL/t \wedge \neg K \neg L/t$
Goal L	$\Rightarrow$	KL
Operator a has prec L	$\Rightarrow$	a has prec KL
Operator a: $C \rightarrow L$	$\Rightarrow$	*for all tags t* $\left\{ \begin{array}{lrcl} a: & KC/t & \rightarrow & KL/t \\ a: & \neg K \neg C/t & \rightarrow & \neg K \neg L/t \end{array} \right.$

For each lit L and merge $m \in M$ with $m = \{t_1, \ldots, t_n\}$, add to O':

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

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

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

Conformant P	$\Rightarrow$	**Classical** $K_{T,M}(P)$
$\langle F, I, O, G\rangle$	$\Rightarrow$	$\langle F', I', O', G'\rangle$
Fluent L	$\Rightarrow$	$KL/t, K\neg L/t$ *(for all tags t)*
Init: known lit L	$\Rightarrow$	$KL \wedge \neg K\neg L$
if $I \models (t \supset L)$	$\Rightarrow$	$KL/t \wedge \neg K\neg L/t$
Goal L	$\Rightarrow$	KL
Operator a has prec L	$\Rightarrow$	a has prec KL
Operator a: $C \rightarrow L$	$\Rightarrow$	$\left\{ \begin{array}{llcl} & \multicolumn{3}{c}{\textit{for all tags } t} \\ a: & KC/t & \rightarrow & KL/t \\ a: & \neg K\neg C/t & \rightarrow & \neg K\neg L/t \end{array} \right.$

For each lit L and merge $m \in M$ with $m = \{t_1, \ldots, t_n\}$, add to O':

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

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

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

Conformant P	$\Rightarrow$	**Classical** $K_{T,M}(P)$
$\langle F, I, O, G \rangle$	$\Rightarrow$	$\langle F', I', O', G' \rangle$
Fluent L	$\Rightarrow$	$KL/t, K\neg L/t$ *(for all tags t)*
Init: known lit L	$\Rightarrow$	$KL \wedge \neg K \neg L$
if $I \models (t \supset L)$	$\Rightarrow$	$KL/t \wedge \neg K \neg L/t$
Goal L	$\Rightarrow$	KL
Operator a has prec L	$\Rightarrow$	a has prec KL
Operator a: $C \rightarrow L$	$\Rightarrow$	$\left\{ \begin{array}{llcl} & \multicolumn{3}{c}{\textit{for all tags } t} \\ a: & KC/t & \rightarrow & KL/t \\ a: & \neg K \neg C/t & \rightarrow & \neg K \neg L/t \end{array} \right.$

For each lit L and merge $m \in M$ with $m = \{t_1, \ldots, t_n\}$, add to O':

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

Idea of $K_{T,M}(P)$

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

- 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 of $K_{T,M}(P)$

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

- 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 of $K_{T,M}(P)$

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

- 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 of $K_{T,M}(P)$

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

- 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 of $K_{T,M}(P)$

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

- 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

Example of T, M

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

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

Interesting properties of the translation $K_{T,M}$?

- **Soundness**: are correct the plans we are obtaining?
  - If not, are they useful?
- **Completeness**: is there a classical plan if there is a conformant one?
  - Is there a one-to-one relationship between conformant and classical plans?
- **Performance**: what are the limitations of a planner based on this translation?
  - What is the **size** of the resulting problem?
  - How do **current** classical planners perform on the translation?

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

Soundness

- If sequence of actions π makes KL/t **true** in $K_{T,M}(P)$, π makes L true in P starting from all the initial states satisfying t
- At least one of the tags t is true
- Then, merging KL is sound

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.*

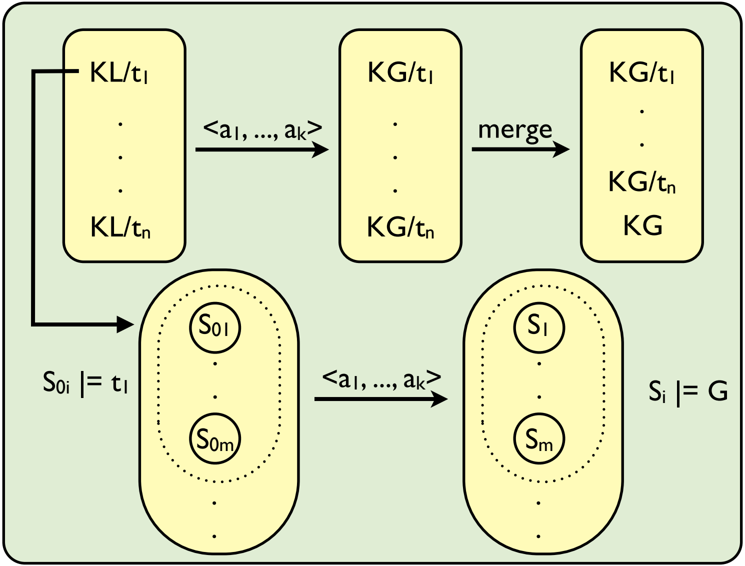
Soundness

- If sequence of actions π makes KL/t **true** in $K_{T,M}(P)$, π makes L true in P starting from all the initial states satisfying t
- At least one of the tags t is true
- Then, merging KL is sound

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.*

Soundness



KL/t1
.
.
.
KL/tn
<a1, ..., ak>
KG/t1
.
.
.
KG/tn
merge
KG/t1
.
.
KG/tn
KG
S01
.
.
S0m
.
.
S0i |= t1
<a1, ..., ak>
S1
.
.
Sm
.
.
Si |= G

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

K_{s0} is a **complete** instance of $K_{T,M}(P)$, by setting

- T to $\{\ \{\}, s_0^1, \ldots, s_0^n\ \}$, and
- M to $\{\ \{s_0^1, \ldots, s_0^n\}\ \}$

where $s_0^1, \ldots, s_0^n$ are the **possible initial states** of P.

- Only **one merge** for the disjunction of possible initial states
- **Intuition**
  - Applying actions in K_{s0} **keeps track** of each fluent L **for each** possible initial **state** s_0: KL/s_0
  - Merge goals using $KG/s_0^1 \wedge \ldots \wedge KG/s_0^n \rightarrow KG$
- This instance is **complete**, but **exponential** in the number of fluents
  - . . . although not a bad conformant planner

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

K_{s0} is a **complete** instance of $K_{T,M}(P)$, by setting

- T to $\{ \{\}, s_0^1, \ldots, s_0^n \}$, and
- M to $\{ \{s_0^1, \ldots, s_0^n\} \}$

where $s_0^1, \ldots, s_0^n$ are the **possible initial states** of P.

- Only **one merge** for the disjunction of possible initial states
- **Intuition**
  - Applying actions in K_{s0} **keeps track** of each fluent L **for each** possible initial **state** s_0: KL/s_0
  - Merge goals using $KG/s_0^1 \wedge \ldots \wedge KG/s_0^n \rightarrow KG$
- This instance is **complete**, but **exponential** in the number of fluents
  - …although not a bad conformant planner

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

K_{s0} is a **complete** instance of $K_{T,M}(P)$, by setting

- T to $\{\ \{\}, s_0^1, \ldots, s_0^n\ \}$, and
- M to $\{\ \{s_0^1, \ldots, s_0^n\}\ \}$

where $s_0^1, \ldots, s_0^n$ are the **possible initial states** of P.

- Only **one merge** for the disjunction of possible initial states
- **Intuition**
  - Applying actions in K_{s0} **keeps track** of each fluent L **for each** possible initial **state** s_0: KL/s_0
  - Merge goals using $KG/s_0^1 \wedge \ldots \wedge KG/s_0^n \rightarrow KG$
- This instance is **complete**, but **exponential** in the number of fluents
  - . . . although not a bad conformant planner

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

K_{s0} is a **complete** instance of $K_{T,M}(P)$, by setting

- T to $\{\ \{\}, s_0^1, \ldots, s_0^n\ \}$, and
- M to $\{\ \{s_0^1, \ldots, s_0^n\}\ \}$

where $s_0^1, \ldots, s_0^n$ are the **possible initial states** of P.

- Only **one merge** for the disjunction of possible initial states
- **Intuition**
  - Applying actions in K_{s0} **keeps track** of each fluent L **for each** possible initial **state** s_0: KL/s_0
  - Merge goals using $KG/s_0^1 \land \ldots \land KG/s_0^n \rightarrow KG$
- This instance is **complete**, but **exponential** in the number of fluents
  - ...although not a bad conformant planner

Example: complete but compact instance of $K_{T,M}$

- Consider the problem P_n
  - **Init**: $x_1 \vee \cdots \vee x_n$
  - **Goal**: g
  - **Actions**: $a_i : x_i \rightarrow g$
- $K_{s0}(P_n)$ size is **exponential on** n
  - $2^n - 1$ initial states
- But having a merge $\{x_1, \ldots, x_n\}$ (and according tags) generates $K_{T,M}(P_n)$ complete
  - Enough with merge $Kg/x_1 \wedge \ldots \wedge Kg/x_n \rightarrow Kg$
  - **Linear on** n
- How can we generate compact instances of $K_{T,M}$?

Example: complete but compact instance of $K_{T,M}$

- Consider the problem P_n
  - **Init**: $x_1 \vee \cdots \vee x_n$
  - **Goal**: g
  - **Actions**: $a_i : x_i \to g$
- $K_{s0}(P_n)$ size is **exponential on** n
  - $2^n - 1$ initial states
- But having a merge $\{x_1, \ldots, x_n\}$ (and according tags) generates $K_{T,M}(P_n)$ complete
  - Enough with merge $Kg/x_1 \wedge \ldots \wedge Kg/x_n \to Kg$
  - **Linear on** n
- How can we generate compact instances of $K_{T,M}$?

Example: complete but compact instance of $K_{T,M}$

- Consider the problem P_n
  - **Init**: $x_1 \vee \cdots \vee x_n$
  - **Goal**: g
  - **Actions**: $a_i : x_i \rightarrow g$
- $K_{s0}(P_n)$ size is **exponential on** n
  - $2^n - 1$ initial states
- But having a merge $\{x_1, \ldots, x_n\}$ (and according tags) generates $K_{T,M}(P_n)$ complete
  - Enough with merge $Kg/x_1 \wedge \ldots \wedge Kg/x_n \rightarrow Kg$
  - **Linear on** n
- How can we generate compact instances of $K_{T,M}$?

Example: complete but compact instance of $K_{T,M}$

- Consider the problem P_n
  - **Init**: $x_1 \vee \cdots \vee x_n$
  - **Goal**: g
  - **Actions**: $a_i : x_i \rightarrow g$
- $K_{s0}(P_n)$ size is **exponential on** n
  - $2^n - 1$ initial states
- But having a merge $\{x_1, \ldots, x_n\}$ (and according tags) generates $K_{T,M}(P_n)$ complete
  - Enough with merge $Kg/x_1 \wedge \ldots \wedge Kg/x_n \rightarrow Kg$
  - **Linear on** n
- How can we generate compact instances of $K_{T,M}$?

Covering Translation

Definition (Covering Translation)

A covering translation is a valid translation $K_{T,M}(P)$ that **includes one merge** $m = t_1, \ldots, t_n$ **that covers** L, for each precondition and goal literal L in P.

Theorem (Completeness)

Covering translations $K_{T,M}(P)$ are complete; i.e., ***if*** π ***is a conformant plan for*** P*, then there is a* ***classical plan*** π' ***for*** $K_{T,M}(P)$ *such that* π *is* π' *with the merge actions removed.*

Covering

Key notions:

- **Relevant** clauses of a literal L: $C_I(L)$
- A **tag** t **satisfies** a clause C
- A **set of tags** m **satisfies** a clause C,
  a.k.a. m **covers** C

Relevance

Definition

Informally, L **is relevant to** L' basically when $a : C \rightarrow L'$ in P with $L \in C$, plus transitivity, etc

Remark: *preconditions do not contribute to relevance*.

Given actions with rules $a : A, B \rightarrow C$, $b : C \rightarrow D$, $b : B \rightarrow \neg C$.

- A is relevant to A, C, D.
- B is relevant to $B, C, D, \neg C$.
- $\neg A$ is relevant to $\neg A, \neg C, \neg D$.
- ...

Relevant Clauses

Suppose problem P with I =

$$p \vee \neg p$$
$$bailoutbanks \vee \neg bailoutbanks$$
$$zapatero \vee merkel \vee berlusconi \vee chavez$$
$$cucumber \vee \neg cucumber$$
$$\ldots$$

- Suppose both p and $\neg p$ are relevant to goal G.
- Also suppose *bailoutbanks* is relevant to goal G, but $\neg$*bailoutbanks* is not. All other literals are not relevant.
- Will not get a solution by reasoning on $bailoutbanks \vee \neg bailoutbanks$
- Enough to reason on $p \vee \neg p$, the only *relevant* clause.

Relevant Clauses

Suppose problem P with I =

$$p \vee \neg p$$
$$bailoutbanks \vee \neg bailoutbanks$$
$$zapatero \vee merkel \vee berlusconi \vee chavez$$
$$cucumber \vee \neg cucumber$$
$$\ldots$$

- Suppose both p and $\neg p$ are relevant to goal G.
- Also suppose *bailoutbanks* is relevant to goal G, but $\neg$*bailoutbanks* is not. All other literals are not relevant.
- Will not get a solution by reasoning on $bailoutbanks \vee \neg bailoutbanks$
- Enough to reason on $p \vee \neg p$, the only *relevant* clause.

Relevant Clauses

Suppose problem P with I =

$$p \vee \neg p$$
$$bailoutbanks \vee \neg bailoutbanks$$
$$zapatero \vee merkel \vee berlusconi \vee chavez$$
$$cucumber \vee \neg cucumber$$
$$\ldots$$

- Suppose both p and $\neg p$ are relevant to goal G.
- Also suppose *bailoutbanks* is relevant to goal G, but $\neg$*bailoutbanks* is not. All other literals are not relevant.
- Will not get a solution by reasoning on $bailoutbanks \vee \neg bailoutbanks$
- Enough to reason on $p \vee \neg p$, the only *relevant* clause.

Relevant Clauses (2)

Definition

Relevant Clause A clause c in I is relevant to a literal L in P if all literals $L' \in C$ are relevant to L.
The set of clauses in I relevant to L is denoted as $C_I(L)$.

Next step: tag t **satisfy** a clause C.

Relevant Clauses (2)

Definition

Relevant Clause A clause c in I is relevant to a literal L in P if all literals $L' \in C$ are relevant to L.
The set of clauses in I relevant to L is denoted as $C_I(L)$.

Next step: tag t **satisfy** a clause C.

Satisfy

- **Warning**: cannot afford expensive inference while building translation $K(P)$.
  - But we need to check $I \models (t \supset L)$ for adding KL/t to the initial state.
  - No general inference on clauses. Use **unit-resolution** – enough when clauses in Prime Implicate form.
- Given tag t, consistent set of literals.

  t **satisfies** $C = L_1 \vee \cdots \vee L_n$ if some L_i is **in** the **consequences** of t given I, *i.e.* $I \models (t \supset L)$
- Let m a valid disjunctions of tags

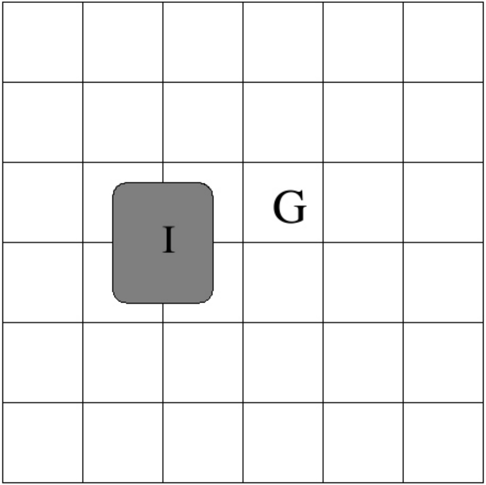
  m **satisfies a clause** C if each tag t satisfies C

Example Satisfy

Suppose $I = \{oneof(x_1, \ldots, x_n), oneof(y_1, \ldots, y_n)\}$, and x_i is relevant to any $x_j, \neg x_j$, y_i is relevant to any $y_j, \neg y_j$.
Notice than $oneof(x_1, \ldots, x_n)$ means $x_1 \vee \ldots \vee x_n$ and $\neg x_i \vee \neg x_j$, for any $i \neq j$.

- The tag $\{x_1, y_1\}$ satisfies all clauses.
  because the consequence of $\{x_1, y_1\}$ is $\{x_1, y_1, \neg x_2, \neg y_2, \ldots, \neg x_n, \neg x_n\}$.
- The merge $m = \{x_1, \ldots, x_n\}$ satisfies $C_I(x_n)$, and m is valid.
- The merge $m' = \{\{x_1, y_1\}, \ldots, \{x_n, y_n\}, \}$ satisfies $C_I(x_n)$, but m' is not valid.

Grid problem



Example Satisfy (2)

Suppose $I = \{oneof(x_1, \ldots, x_n), oneof(y_1, \ldots, y_n)\}$, and x_i is relevant to any $x_j, \neg x_j$, y_i is relevant to any $y_j, \neg y_j$. Also suppose x_i is relevant to any $y_j, \neg y_j$, and y_i is relevant to $x_j, \neg x_j$. *Everything is relevant to everything*.

- the tag $\{x_1, y_1\}$ satisfies both clauses.
- The merge $m = \{x_1, \ldots, x_n\}$ does not satisfy $C_I(x_n)$, even though m is valid.
- The merge $m' = \{\{x_1, y_1\}, \ldots, \{x_n, y_n\}, \}$ does satisfy $C_I(x_n)$, but m' is not valid.
- The merge $m'' = \{x_1, \ldots, x_n\} \times \{y_1, \ldots, y_n\}$ does satisfy $C_I(x_n)$, and m'' is valid.

Covering Translation

Definition (Covering Merges)

A valid merge m in a translation $K_{T,M}(P)$ **covers** a literal L if m **satisfies** $C_I(L)$, the set of clauses in I **relevant** to L

Definition (Covering Translation)

A covering translation is a valid translation $K_{T,M}(P)$ that **includes one merge** $m = t_1, \ldots, t_n$ **that covers** L, for each precondition and goal literal L in P.

Theorem (Completeness)

Covering translations $K_{T,M}(P)$ are complete; i.e., ***if*** π ***is a conformant plan for*** *P, then there is a* ***classical plan*** π' ***for*** *$K_{T,M}(P)$ such that π is π' with the merge actions removed.*

Example of Covering Translation

Example: K_{s0}

The merge $\{s_0^1, \ldots, s_0^n\}$ is covering because (1) is valid (2) each initial state s_0^i satisfies each clause

Example: oneof

If $C_I(L) = \{L_1 \vee \cdots \vee L_n, \neg L_i \vee \neg L_j \text{ for all } i \neq j\}$, then the merge $\{L_1, \ldots, L_n\}$ is covering because (1) disjunction in I are valid and (2) each L_i implies $\neg L_j$ (for $j \neq i$) and then L_i satisfies each clause in $C_I(L)$

Example of Covering Translation

Example: K_{s0}

The merge $\{s_0^1, \ldots, s_0^n\}$ is covering because (1) is valid (2) each initial state s_0^i satisfies each clause

Example: `oneof`

If $C_I(L) = \{L_1 \vee \cdots \vee L_n, \neg L_i \vee \neg L_j \text{ for all } i \neq j\}$, then the merge $\{L_1, \ldots, L_n\}$ is covering because (1) disjunction in I are valid and (2) each L_i implies $\neg L_j$ (for $j \neq i$) and then L_i satisfies each clause in $C_I(L)$

Cover it!

- Covering translation guarantee **completeness.**
- How do we **get** a covering translation?

  In principle we want small T, M
- **Naive**: just **combinations** of clauses is **unbounded** on size

  ... but sometimes is a good idea.

Width

Definition (Width of Literal)

The conformant width of a literal L, written $w(L)$, is the **size** of the *smallest set of clauses* $\mathcal{C}$ in $C_I^*(L)$ such that cover $c(\mathcal{C})$ *satisfies* $C_I(L)$.

- Roughly, cover $c(\mathcal{C})$ is combination of literals of clauses $\mathcal{C}$
- $C_I^*(L) =$ relevant clauses $C_I(L)$ $\cup$ tautologies for unknown literals $p \vee \neg p$
- Idea: smallest $\mathcal{C}$ can be made of
  - Clauses in $C_I(L)$
  - Last resort: combination of tautologies $p \vee \neg p$
- Then, $w(L)$ is at most n, the number of unknown fluents
- If $C_I(L)$ is empty, $w(L) = 0$

Width

Definition (Width of Literal)

The conformant width of a literal L, written $w(L)$, is the **size** of the *smallest set of clauses* $\mathcal{C}$ in $C_l^*(L)$ such that cover $c(\mathcal{C})$ *satisfies* $C_I(L)$.

- Roughly, cover $c(\mathcal{C})$ is combination of literals of clauses $\mathcal{C}$
- $C_I^*(L)$ = relevant clauses $C_I(L)$ $\cup$ tautologies for unknown literals $p \vee \neg p$
- Idea: smallest $\mathcal{C}$ can be made of
  - Clauses in $C_I(L)$
  - Last resort: combination of tautologies $p \vee \neg p$
- Then, $w(L)$ is at most n, the number of unknown fluents
- If $C_I(L)$ is empty, $w(L) = 0$

Width

Definition (Width of Literal)

The conformant width of a literal L, written $w(L)$, is the **size** of the *smallest set of clauses* $\mathcal{C}$ in $C_I^*(L)$ such that cover $c(\mathcal{C})$ *satisfies* $C_I(L)$.

- Roughly, cover $c(\mathcal{C})$ is combination of literals of clauses $\mathcal{C}$
- $C_I^*(L) =$ relevant clauses $C_I(L) \cup$ tautologies for unknown literals $p \vee \neg p$
- Idea: smallest $\mathcal{C}$ can be made of
  - Clauses in $C_I(L)$
  - Last resort: combination of tautologies $p \vee \neg p$
- Then, $w(L)$ is at most n, the number of unknown fluents
- If $C_I(L)$ is empty, $w(L) = 0$

Width

Definition (Width of Literal)

The conformant width of a literal L, written $w(L)$, is the **size** of the *smallest set of clauses* $\mathcal{C}$ in $C_I^*(L)$ such that cover $c(\mathcal{C})$ *satisfies* $C_I(L)$.

- Roughly, cover $c(\mathcal{C})$ is combination of literals of clauses $\mathcal{C}$
- $C_I^*(L) =$ relevant clauses $C_I(L) \cup$ tautologies for unknown literals $p \vee \neg p$
- Idea: smallest $\mathcal{C}$ can be made of
  - Clauses in $C_I(L)$
  - Last resort: combination of tautologies $p \vee \neg p$
- Then, $w(L)$ is at most n, the number of unknown fluents
- If $C_I(L)$ is empty, $w(L) = 0$

Width

Definition (Width of Literal)

The conformant width of a literal L, written $w(L)$, is the **size** of the *smallest set of clauses* $\mathcal{C}$ in $C_I^*(L)$ such that cover $c(\mathcal{C})$ *satisfies* $C_I(L)$.

- Roughly, cover $c(\mathcal{C})$ is combination of literals of clauses $\mathcal{C}$
- $C_I^*(L) =$ relevant clauses $C_I(L)\ \cup$ tautologies for unknown literals $p \vee \neg p$
- Idea: smallest $\mathcal{C}$ can be made of
  - Clauses in $C_I(L)$
  - Last resort: combination of tautologies $p \vee \neg p$
- Then, $w(L)$ is at most n, the number of unknown fluents
- If $C_I(L)$ is empty, $w(L) = 0$

Width

Definition (Width of Problem)

The conformant width of a problem P, written as $w(P)$, is $w(P) = \max_L w(L)$, where L ranges over the precondition and goal literals in P.

- **Calculate** $w(L)$ requires find a subset of clauses of $C_I^*(L)$ whose cover satisfies $C_I(L)$
  - $\rightarrow$ **exponential** on size of $C_I^*(L)$
- But **verify** whether $w(L) \leq i$ is **polynomial** for fixed i
  - $\rightarrow$ For each subset of i clauses, try to get a cover

Width (examples)

- If $C_I(L)$ is $oneof(x_1, \ldots, x_m)$, then $w(L) = 1$ because $\mathcal{C} = \{x_1 \vee \cdots \vee x_m\}$ generates the cover $c(\mathcal{C}) = \{\{x_1\}, \ldots, \{x_m\}\}$ that satisfies $C_I(L)$.
- If $C_I(L)$ is $(p \vee \neg p)$ and $(q \vee \neg q)$, then $w(L) = 2$ as the smallest $\mathcal{C}$ in $C_I^*(L)$ whose cover satisfies $C_I(L)$ is $C_I(L)$ itself.
- Sqr-center. Init = oneof$(x_1, \ldots, x_n)$, oneof$(y_1, \ldots, y_n)$. Goal = x_{center}, y_{center}. Actions: `up`, `down`, `left`, `right`. Rules like `up`: $y_i \rightarrow y_{i+1} \wedge \neg y_i$
  - Has **width 1** because x_i not relevant to y_j

Width (examples)

- If $C_I(L)$ is $oneof(x_1, \ldots, x_m)$, then $w(L) = 1$ because $\mathcal{C} = \{x_1 \vee \cdots \vee x_m\}$ generates the cover $c(\mathcal{C}) = \{\{x_1\}, \ldots, \{x_m\}\}$ that satisfies $C_I(L)$.
- If $C_I(L)$ is $(p \vee \neg p)$ and $(q \vee \neg q)$, then $w(L) = 2$ as the smallest $\mathcal{C}$ in $C_I^*(L)$ whose cover satisfies $C_I(L)$ is $C_I(L)$ itself.
- Sqr-center. Init = oneof($x_1, \ldots, x_n$), oneof($y_1, \ldots, y_n$). Goal = x_{center}, y_{center}. Actions: `up`, `down`, `left`, `right`. Rules like `up`: $y_i \rightarrow y_{i+1} \wedge \neg y_i$
  - Has **width 1** because x_i not relevant to y_j

Width (examples)

- If $C_I(L)$ is $oneof(x_1, \ldots, x_m)$, then $w(L) = 1$ because $\mathcal{C} = \{x_1 \vee \cdots \vee x_m\}$ generates the cover $c(\mathcal{C}) = \{\{x_1\}, \ldots, \{x_m\}\}$ that satisfies $C_I(L)$.
- If $C_I(L)$ is $(p \vee \neg p)$ and $(q \vee \neg q)$, then $w(L) = 2$ as the smallest $\mathcal{C}$ in $C_I^*(L)$ whose cover satisfies $C_I(L)$ is $C_I(L)$ itself.
- Sqr-center. Init = oneof$(x_1, \ldots, x_n)$, oneof$(y_1, \ldots, y_n)$. Goal = x_{center}, y_{center}. Actions: `up`, `down`, `left`, `right`. Rules like `up`: $y_i \rightarrow y_{i+1} \wedge \neg y_i$
  - Has **width 1** because x_i not relevant to y_j

Translation $K_i(P)$

Definition (Translation K_i)

The translation $K_i(P)$ is obtained from $K_{T,M}(P)$ where

- If $w(P) \leq i$, then one merge $m = c(\mathcal{C})$ for the selected clauses $\mathcal{C}$ of each precond and goal literal L in P.
- Otherwise, one merge $m = c(\mathcal{C})$ for L **for each set** $\mathcal{C}$ **of** i **clauses** in $C_i^*(L)$.
- T is the collection of tags appearing in those merges and the empty tag.

Theorem (Properties K_i)

For a fixed i, the translation $K_i(P)$ is sound, polynomial, and if $w(P) \leq i$, covering and complete.

Translation $K_i(P)$

Definition (Translation K_i)

The translation $K_i(P)$ is obtained from $K_{T,M}(P)$ where

- If $w(P) \leq i$, then one merge $m = c(\mathcal{C})$ for the selected clauses $\mathcal{C}$ of each precond and goal literal L in P.
- Otherwise, one merge $m = c(\mathcal{C})$ for L **for each set** $\mathcal{C}$ **of** i **clauses** in $C_I^*(L)$.
- T is the collection of tags appearing in those merges and the empty tag.

Theorem (Properties K_i)

For a fixed i, the translation $K_i(P)$ is sound, polynomial, and if $w(P) \leq i$, covering and complete.

Translation $K_i(P)$

Definition (Translation K_i)

The translation $K_i(P)$ is obtained from $K_{T,M}(P)$ where

- If $w(P) \leq i$, then one merge $m = c(\mathcal{C})$ for the selected clauses $\mathcal{C}$ of each precond and goal literal L in P.
- Otherwise, one merge $m = c(\mathcal{C})$ for L **for each set $\mathcal{C}$ of i clauses** in $C_I^*(L)$.
- T is the collection of tags appearing in those merges and the empty tag.

Theorem (Properties K_i)

For a fixed i, the translation $K_i(P)$ is sound, polynomial, and if $w(P) \leq i$, covering and complete.

Width of Conformant Benchmarks

	Domain-Parameter	# Unknown Fluents	Width
1	Safe-n combinations	n	1
2	UTS-n locs	n	1
3	Ring-n rooms	$4n$	1
4	Bomb-in-the-toilet-n bombs	n	1
5	Comm-n signals	n	1
6	Square-Center-$n \times n$ grid	$2n$	1
7	Cube-Center-$n \times n \times n$ cube	$3n$	1
8	Grid-n shapes of n keys	$n \times m$	1
9	Logistics n pack m locs	$n \times m$	1
10	Coins-n coins m locs	$n \times m$	1
11	Block-Tower-n Blocks	$n \times (n-1) + 3n + 1$	same
12	Sortnet-n bits	n	n
13	Adder n pairs of bits	$2n$	$2n$
14	Look-and-Grab m objs from $n \times n$ locs	$n \times n \times m$	m
15	1-dispose m objs from $n \times n$ locs	$n \times n \times m$	m

Width of some problems

- Blocks have **maximal width**.
- But blocks, with a **magic action** to achieve the goal
  - Trivial (solved by K_0)
- **Look-n-grab** for m objs has width m, but does not depend on size of the grid.
  - Why? Every clause relevant to `handempty`, that is relevant to all goals

Conformant Width: intuitions

- It is **not** necessary to deal with all relevant clauses $C_I(L)$ to achieve KL, for L goal or precond
  - **some** of them are **enough** for deciding the others
  - How many? $w(L)$
- Let P_N a problem of size N, having $w(P_N) = i$ for any N. It maybe that for $K_i(P_N)$:
  - the number of tags grows linear on N, but ...
  - the number of initial states of P_N grows exponentially on N
  - **How can be K_i complete?**

    *A **tag** t **summarize information** about all the initial states consistent with t*

Conformant Width: intuitions

- It is **not** necessary to deal with all relevant clauses $C_I(L)$ to achieve KL, for L goal or precond
  - **some** of them are **enough** for deciding the others
  - How many? $w(L)$
- Let P_N a problem of size N, having $w(P_N) = i$ for any N. It maybe that for $K_i(P_N)$:
  - the number of tags grows linear on N, but ...
  - the number of initial states of P_N grows exponentially on N
  - **How can be K_i complete?**

    *A **tag** t **summarize information** about all the initial states consistent with t*

Basis

- Given P a conformant problem and $S \subseteq S_0$ a **subset of the possible initial states** of P.
- Let $P[S]$ the conformant problem that is like P but with the **set of initial states restricted to** S.

Definition

S is a **basis** for P iff any conformant plan for $P[S]$ is a conformant plan for P.

Theorem

Conformant problems P with $width(P) \leq i$ have basis of size $|S|$ exponential in i. (Even if $|S_0|$ is exponential on number of fluents)

You can plan just for a basis (if you are able to find one)! **Why?**

Basis

- Given P a conformant problem and $S \subseteq S_0$ a **subset of the possible initial states** of P.
- Let $P[S]$ the conformant problem that is like P but with the **set of initial states restricted to** S.

Definition

S is a **basis** for P iff any conformant plan for $P[S]$ is a conformant plan for P.

Theorem

Conformant problems P with width$(P) \leq i$ have basis of size $|S|$ exponential in i. (Even if $|S_0|$ is exponential on number of fluents)

You can plan just for a basis (if you are able to find one)! **Why?**

Basis examples

Oneof

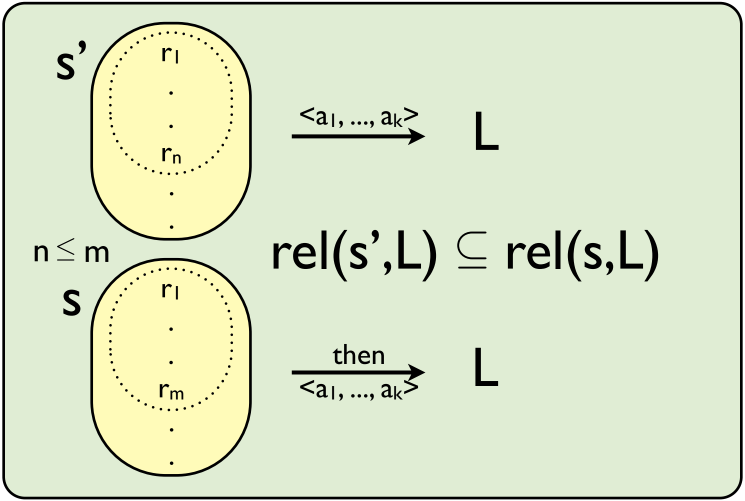
- Consider a problem P with $I = \{x_1 \vee \cdots \vee x_n, \neg x_i \vee \neg x_j \text{ for all } i \neq j\}$.
  A basis maybe:

$$\{x_1, \neg x_2, \ldots, \neg x_n\}$$
$$\{\neg x_1, x_2, \ldots, \neg x_n\}$$
$$\ldots$$
$$\{\neg x_1, \neg x_2, \ldots, x_n\}$$

- Consider a problem P with $I = \{x_1 \vee \cdots \vee x_n\}$.
  A basis is the same previous set of states.

- Why is this a basis for both problems?

Monotonicity



There exist a Basis!

- Giving literal L and a covering merge $m = \{t_1, \ldots, t_n$, for any state s there exist i s.t. $rel(t_i^*, L) \subseteq rel(s, L)$.
- Pick s_i s.t. $rel(t_i^*, L) \subseteq rel(s_i, L)$ and there is no s_j s.t. $rel(s_j, L) \subset rel(s_i, L)$.

  Hint: like picking the set of 'smaller' s_i'
- The set $\{s_1, \ldots, s_n\}$ is a basis!

Hint:

- You don't need to use $K_{T,M}$. If you are able to identify a basis S, do free-style conformant plan with initial states S.
- If you use a subset of initial states S that is not a basis, you will not get sound solutions.
  - Will be useful for relaxations/heuristics.

There exist a Basis!

- Giving literal L and a covering merge $m = \{t_1, \ldots, t_n$, for any state s there exist i s.t. $rel(t_i^*, L) \subseteq rel(s, L)$.
- Pick s_i s.t. $rel(t_i^*, L) \subseteq rel(s_i, L)$ and there is no s_j s.t. $rel(s_j, L) \subset rel(s_i, L)$.

  Hint: like picking the set of 'smaller' s_i'
- The set $\{s_1, \ldots, s_n\}$ is a basis!

Hint:

- You don't need to use $K_{T,M}$. If you are able to identify a basis S, do free-style conformant plan with initial states S.
- If you use a subset of initial states S that is not a basis, you will not get sound solutions.
  - Will be useful for relaxations/heuristics.

Other instances of $K_{T,M}$?

- Remember you just need:
  - Valid set of **tags** T
  - **Merges**: valid disjunctions of tags in M.
- Grab clauses in I and do whatever you method you have to do so.

  Hint: get your favorite SAT-solver/model-enumeration technique and salt as you need.
- Clear **semantics** of $K_{T,M}$ tell you the consequences of using invalid or uncovering merges.

  Use with responsibility. Thinks may get easier or more complicated.
- In any state where you get $\neg KL/t \wedge \neg K\neg L/t$, you know you **lost track** of L for any initial state satisfying t.

  Monitor execution!

Other instances of $K_{T,M}$?

- Remember you just need:
  - Valid set of **tags** T
  - **Merges**: valid disjunctions of tags in M.
- Grab clauses in I and do whatever you method you have to do so.

  Hint: get your favorite SAT-solver/model-enumeration technique and salt as you need.
- Clear **semantics** of $K_{T,M}$ tell you the consequences of using invalid or uncovering merges.

  Use with responsibility. Thinks may get easier or more complicated.
- In any state where you get $\neg KL/t \wedge \neg K\neg L/t$, you know you **lost track** of L for any initial state satisfying t.

  Monitor execution!

Other instances of $K_{T,M}$?

- Remember you just need:
  - Valid set of **tags** T
  - **Merges**: valid disjunctions of tags in M.
- Grab clauses in I and do whatever you method you have to do so.

  Hint: get your favorite SAT-solver/model-enumeration technique and salt as you need.
- Clear **semantics** of $K_{T,M}$ tell you the consequences of using invalid or uncovering merges.

  Use with responsibility. Thinks may get easier or more complicated.
- In any state where you get $\neg KL/t \wedge \neg K\neg L/t$, you know you **lost track** of L for any initial state satisfying t.

  Monitor execution!

Other instances of $K_{T,M}$?

- Remember you just need:
  - Valid set of **tags** T
  - **Merges**: valid disjunctions of tags in M.
- Grab clauses in I and do whatever you method you have to do so.

  Hint: get your favorite SAT-solver/model-enumeration technique and salt as you need.
- Clear **semantics** of $K_{T,M}$ tell you the consequences of using invalid or uncovering merges.

  Use with responsibility. Thinks may get easier or more complicated.
- In any state where you get $\neg KL/t \wedge \neg K\neg L/t$, you know you **lost track** of L for any initial state satisfying t.

  Monitor execution!

Translation *Kmodels*(P)

Definition

The translation *Kmodels*(P) from the general $K_{T,M}(P)$

- Merge m for each precond and goal L:
  models* of $C_I(L)$ that are consistent with I

Theorem

The translation Kmodels(P) is sound and complete.

Key points:

- *Kmodels is equivalent to K_{S0} when **all** the clauses in I are **relevant to all** the precondition and goal literals L.*
- *But Kmodels **exponential** on number of **vars in** $C_I(L)$, while K_{S0} exponential in the number of unknown **vars in** I.*

Translation $Kmodels(P)$

Definition

The translation $Kmodels(P)$ from the general $K_{T,M}(P)$

- Merge m for each precond and goal L:
  models* of $C_I(L)$ that are consistent with I

Theorem

The translation $Kmodels(P)$ is sound and complete.

Key points:

– *Kmodels is equivalent to K_{S0} when **all** the clauses in I are **relevant to all** the precondition and goal literals L.*

– *But Kmodels **exponential** on number of **vars in** $C_I(L)$, while K_{S0} exponential in the number of unknown **vars in** I.*

The planner T_0

- Conformant Planner T_0, **winner** at IPC-2006, was based on K_1 + FF, an effective classical planner.
  - Using SAT-based conformant planner when FF did not find solution in K_1
- version for IPC-2008 K_1 + *Kmodels*
  - `CpA(H)` was the winner.

T_0 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
- If using K_{s0}, *Kmodels* or K_i for width $\leq i$ **cancellation** can be **tracked by support** rules
  - Given rule $C \to L$, instead of both $KC \to KL$ and $\neg K \neg C \to \neg K \neg L$
  - keep only $KC \to KL \wedge \neg K \neg L$
- For **invariant** oneof$(x_1, \ldots, x_n)$: keep Kx_i updated. Example:

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

- Sometimes for width > 1, can be solved if allowing merge **not only** for precs and goal

Translating P into $K_1(P)$: size

	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

- After some simplifications made for T_0 to the PDDL
- Translation is not the bottleneck

Performance on current classical planners?

- Size of grounded instances
- Support for conditional effects
- Sensibility of heuristics

Thanks FF for

- accepting big grounded PDDLs
- dealing with lots of conditional effects

We still got issues with LAMA.

Digression: on conditional effects

- **Conditional effects are very expressive!**

  one of the few ADL extensions than cannot be compiled away with some blow-up
- If classical **planning is symbolical reachability** where differences from an state to another are
  - verified easily (STRIPS preconditions)
  - represented compactly (STRIPS add and delete)
- Conditional effects are
  - essentially **different** because simultaneous changes by the same action
  - also a **compact** representation of change
- Button line: ***good support of conditional effects is needed from classical planners***. Challenge accepted!
  - Current planners are tested with hand-made problems with a few cond-effects.
  - Even simple cases are not well treated.

Sampling

(Albore *et al*, ICAPS-2011). ***IIIb, Wednesday 10:30h.***

- **Sampling**: pick a set of initial states and plan for them.
  - A complete sample will be a basis!
- Recall $P[S]$ is the conformant problem P but restricted to the set of initial states S.
- Let $KS(P) = K_{s0}(P[S])$. **Complete if S is a basis!**
- Define **new instance** $K_S^i(P)$ that is
  - **Exponential** on i, the size of tags.
  - Always **complete**.
  - **Not always sound.**
  - Sound if conformant width $w(P) \leq i$.

Sampling

(Albore *et al*, ICAPS-2011). ***IIIb, Wednesday 10:30h.***

- **Sampling**: pick a set of initial states and plan for them.
  - A complete sample will be a basis!
- Recall $P[S]$ is the conformant problem P but restricted to the set of initial states S.
- Let $KS(P) = K_{s0}(P[S])$. **Complete if S is a basis!**
- Define **new instance** $K_S^i(P)$ that is
  - **Exponential** on i, the size of tags.
  - Always **complete**.
  - **Not always sound.**
  - Sound if conformant width $w(P) \leq i$.

Sampling

(Albore *et al*, ICAPS-2011). ***IIIb, Wednesday 10:30h.***

- **Sampling**: pick a set of initial states and plan for them.
  - A complete sample will be a basis!
- Recall $P[S]$ is the conformant problem P but restricted to the set of initial states S.
- Let $KS(P) = K_{s0}(P[S])$. **Complete if S is a basis!**
- Define **new instance** $K^i_S(P)$ that is
  - **Exponential** on i, the size of tags.
  - Always **complete**.
  - **Not always sound**.
  - Sound if conformant width $w(P) \leq i$.

Sampling (2)

- $K_S^i(P)$ is $KS(P)$ with a base of size exponential on i.
  - **Why** may $K_S^i(P)$ be **unsound**?
  - Relaxation of $K_{T,M}$ allows to always get a solution!
- Almost classic belief state planner using $K_S^i(P)$ for heuristic.
  - Tricky part was choosing a good approximated basis.
    Spoiler: minimal cardinality on propositional logic!
- See (Shani & Brafman, 2011), that is based on $K_{T,M}$ for using sampling in contingent planning.

Sampling (2)

- $K_S^i(P)$ is $KS(P)$ with a base of size exponential on i.
  - **Why** may $K_S^i(P)$ be **unsound**?
  - Relaxation of $K_{T,M}$ allows to always get a solution!
- Almost classic belief state planner using $K_S^i(P)$ for heuristic.
  - Tricky part was choosing a good approximated basis.
    Spoiler: minimal cardinality on propositional logic!
- See (Shani & Brafman, 2011), that is based on $K_{T,M}$ for using sampling in contingent planning.

Sampling (2)

- $K_S^i(P)$ is $KS(P)$ with a base of size exponential on i.
  - **Why** may $K_S^i(P)$ be **unsound**?
  - Relaxation of $K_{T,M}$ allows to always get a solution!
- Almost classic belief state planner using $K_S^i(P)$ for heuristic.
  - Tricky part was choosing a good approximated basis.
    Spoiler: minimal cardinality on propositional logic!
- See (Shani & Brafman, 2011), that is based on $K_{T,M}$ for using sampling in contingent planning.

Related Work

- **Belief state search** (Bonet & Geffner, 2000)
  - Translation to classical planning allows to use
    - in many cases a very **compact** representation
    - classical planning **heuristics**
- **0-approximation** (Baral & Son, 1997)
  - Incomplete semantic used for conformant planning
  - Extended to be **complete** with exponential saving respect to standard semantic (Son & Tu, 2006)
  - Some problems are **exponential** for complete 0-approximation, but **have width 1**
    - `CpA` (Tran *et al*, 2009) has optimization for not being exponential in some of these problems.

Related Work

- **Belief state search** (Bonet & Geffner, 2000)
  - Translation to classical planning allows to use
    - in many cases a very **compact** representation
    - classical planning **heuristics**
- **0-approximation** (Baral & Son, 1997)
  - Incomplete semantic used for conformant planning
  - Extended to be **complete** with exponential saving respect to standard semantic (Son & Tu, 2006)
  - Some problems are **exponential** for complete 0-approximation, but **have width 1**
    - `CpA` (Tran *et al*, 2009) has optimization for not being exponential in some of these problems.

T_0 vs CpA

- $K_{T,M}$ based: **local context** for each literals. Complete: context is enough for achieving the problem
- 0-approximation extended to be complete: minimal **global context** for achieving the problem
- $K_{T,M}$ maybe be **exponential** better than the 0-approx.
- **Merging one-of** helps CpA
- We get **classical problem.** CpA: search algorithm, heuristics.
- But classical problem can be quite big. CpA may have advantage.
- More **recent planners** *CNF*, *DNF* explore different representations and transitions functions.

T_0 vs CpA

- $K_{T,M}$ based: **local context** for each literals. Complete: context is enough for achieving the problem
- 0-approximation extended to be complete: minimal **global context** for achieving the problem
- $K_{T,M}$ maybe be **exponential** better than the 0-approx.
- **Merging one-of** helps CpA
- We get **classical problem**. CpA: search algorithm, heuristics.
- But classical problem can be quite big. CpA may have advantage.
- More **recent planners** *CNF*, *DNF* explore different representations and transitions functions.

T_0 vs CpA

- $K_{T,M}$ based: **local context** for each literals. Complete: context is enough for achieving the problem
- 0-approximation extended to be complete: minimal **global context** for achieving the problem
- $K_{T,M}$ maybe be **exponential** better than the 0-approx.
- **Merging one-of** helps CpA
- We get **classical problem**. CpA: search algorithm, heuristics.
- But classical problem can be quite big. CpA may have advantage.
- More **recent planners** *CNF*, *DNF* explore different representations and transitions functions.

Summary of first part

- 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
- Characterization of the complexity of the complete $K_{T,M}$ in term of the **conformant width**
- Translation scheme K_i that is **always polynomial** and complete if conformant width $\leq i$
- A conformant planner T_0 based on instances of $K_{T,M}$

References

- [Baral & Son, ILPS-1997]. Baral, C., & Son, T. C. *Approximate reasoning about actions in presence of sensing and incomplete information*. ILPS-1997.
- [Bonet & Geffner, AIPS-2000]. Bonet, B., & Geffner, H. *Planning with incomplete information as heuristic search in belief space*. AIPS-2000.
- [Son & Tu, KR-2006]. Son, T. C., & Tu, P. H. *On the completeness of approximation based reasoning and planning in action theories with incomplete information.* KR-2006.
- [Tran *et al*, PADL-2009]. Tran, D., Nguyen, H., Pontelli, E., & Son, T. C. *Improving performance of conformant planners: Static analysis of declarative planning domain specifications*. PADL-2009.
- [Palacios & Geffner, JAIR-2009]. *Compiling Uncertainty Away in Conformant Planning Problems with Bounded Width*. Palacios, H., & Geffner, H.. JAIR 2009.
- [Albore *et al*, ICAPS-2011]*Effective Heuristics and Belief Tracking for Planning with Incomplete Information.* Albore, A., Ramirez, M., & Geffner, H. ICAPS-2011.
- [Shani & Brafman, IJCAI-2011]. *Replanning in Domains with Partial Information and Sensing Actions*. Shani, G., & Brafman, Ronen. IJCAI-2011.

More references on the second part!

Translation-based Approaches to Conformant and Contingent Planning

Part II

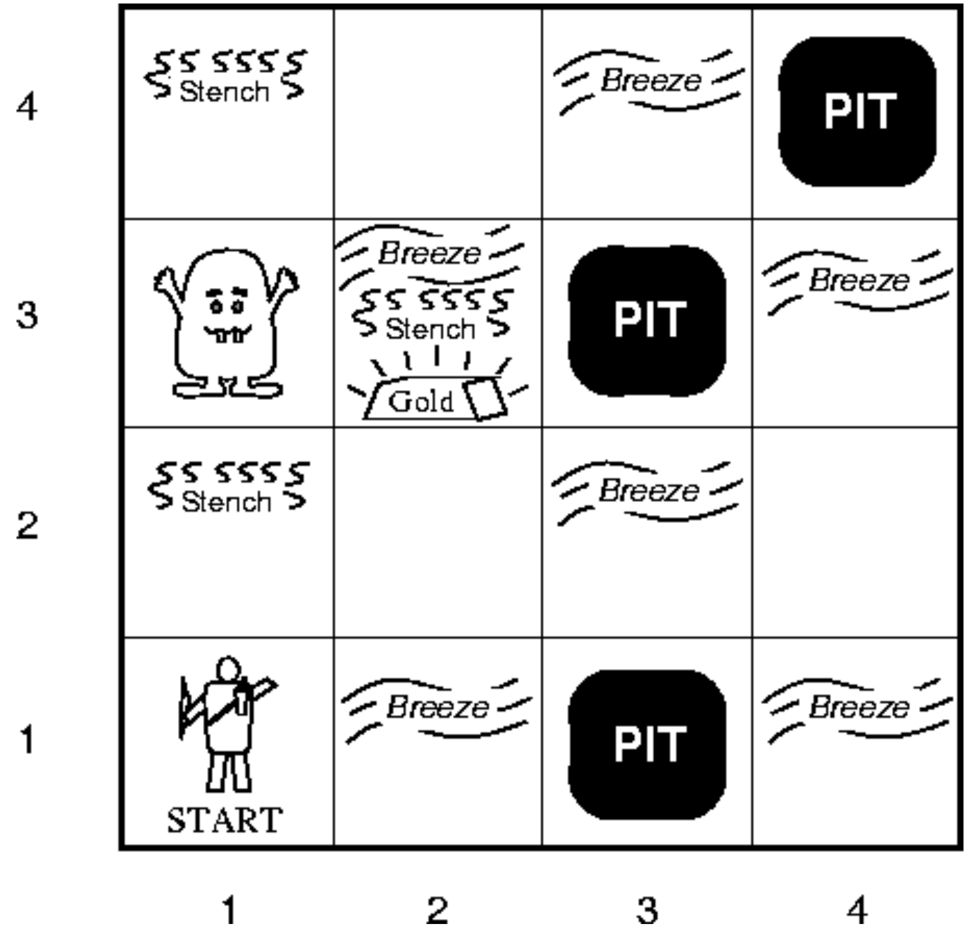
Contingent Planning

- Conformant problem =
  classical problem + **incomplete information**

- Contingent problem =
  conformant problem + **sensing actions**

- STRIPS Problem P= <F, I, A, G> with three extensions:
  - I is a well-formed formula over F, encoding uncertainty
  - Actions $a \in A$ may have **conditional effects**
  - **Sensing actions**

Action selection in Wumpus



from Russell & Norvig

What should the agent do next?

Contingent Planning:
Sensing and Incomplete Information

- Finding a solution in presence of partial or incomplete information.
  - The belief states space size which is combinatorially large.
  - Difficult to obtain **informed heuristics** in belief space.

- The solution strongly depends on the observation outcome.
  - The size of the solution grows **exponentially** with the number of possible observations.

    Thus verification and/or generation of a plan takes **exponential** time.

A Translation-based approach to Contingent Planning

- Contingent problems cannot be translated into classical ones, as they have **different solution forms** (trees vs. sequences).

- **Offline** planning: provide solution tree for all possible contingencies

- **Online** planning: action sequence generated on-the-fly (interleaving planning and execution).

- As for conformant planning, translation compiles beliefs away: states represent “belief states” over P.

Compiling into classical planning: the CLG approach

- Contingent problem P translated into **fully observable but non-deterministic problem** $X_{T,M}(P)$.

  - Sensing is modeled as actions with **non-deterministic effects**
  - $X_{T,M}(P)$ has complete information!
    Solutions to $\mathbf{X_{T,M}(P)}$ yield solutions for P!

  ...but how to deal with sensing?
  Search has to make explicit effort to obtain information.

- Later on, we will **guide** the search using relaxation $\mathbf{X^{+}(P)}$, that is **also** a classical planning problem.

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

- Contingent problem P = Conformant problem P' + Sensing actions.

- $X_{T,M}(P) = K_{T,M}(P') +$ Deductive Actions + Sensing Actions

  - **Deductive actions**:

    tag refutation: $KL/t \wedge K\neg L \;\rightarrow\; K\neg t$

    contingent merge: $\bigwedge_{t \in m, m \in M_L} (KL/t \vee K\neg t) \;\rightarrow\; KL$

  - **Sensing actions** obs(L) from P encoded in $X_{T,M}(P)$ as non-deterministic actions:

$$obs(L) : \neg KL \wedge \neg K\neg L \rightarrow KL \,|\, K\neg L$$

Complete Translation X_{S0} *(P)*

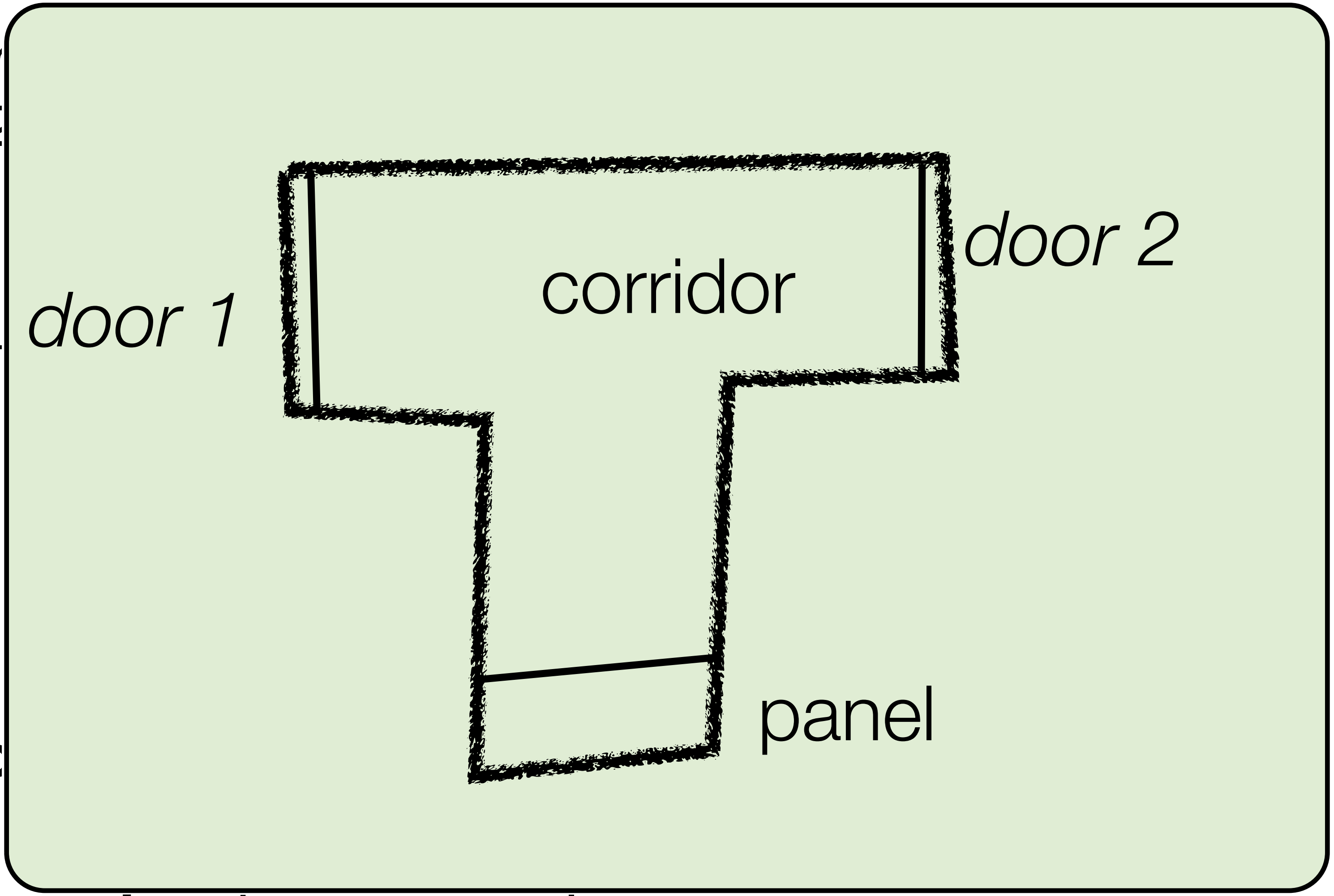
- Translation X_{S0} (P) is special case of $X_{T,M}$(P) with:
  - T equal to the set of **all** possible initial states of P
  - M containing a merge m=T for each precondition and goal literal L of P.

Theorem: X_{S0} (P) is **sound** and **complete**.

This translation is suitable when number of initial states is low;
in worst case exponential in number of uncertain fluents.

Example: Problem P

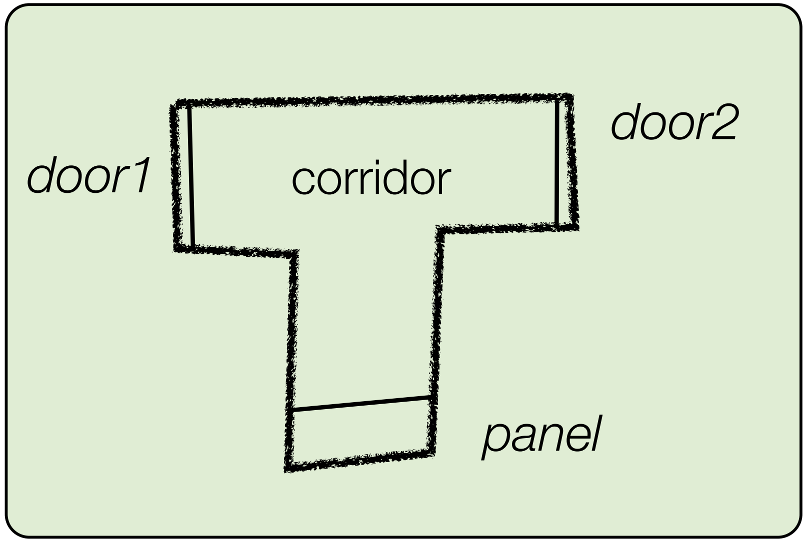
- **Fluer**
  door2
- **Init**:
  oneo
  ∧¬go
- **Goal**
- **Actic**



- **Observation**: inspect-panel

Example: Problem P

- **Fluents**: opened-door1, opened-door2, corridor, door1, door2, panel, gold-found
- **Init**: oneof(opened-door1, opened-door2) ∧ at(corridor) ∧¬gold-found
- **Goal**: gold-found
- **Actions**: goto(?pos, ?dest), open(?door)
- **Observation**: inspect-panel



Example
Problem P - Actions

goto(?pos, ?dest):

pre: *at(?pos)*

effect: *at(?dest)* ∧ ¬ *at(?pos)*

open(?door):

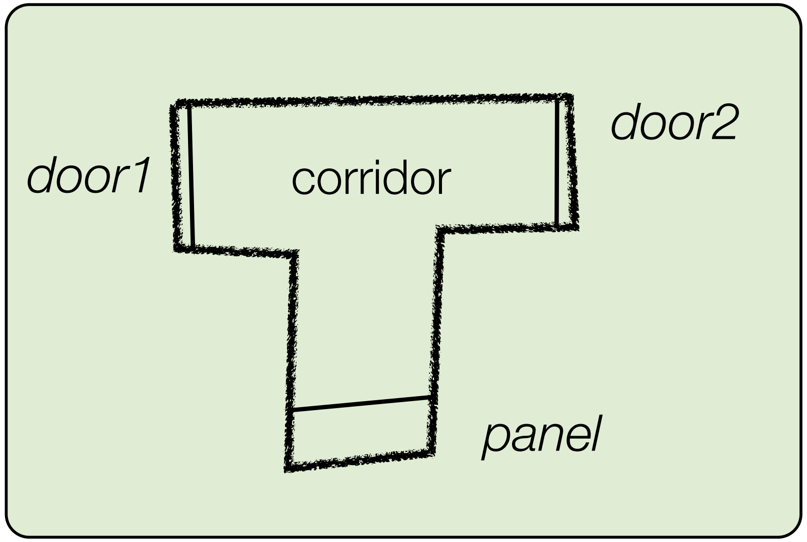
pre: *at(?door)* ∧ *opened(?door)*

effect: *gold-found*

inspect-panel:

pre: *at(panel)*

observation:
opened-door1
| ¬ *opened-door1*



Example
X_{S0} (P) translation

- Tags (2 possible states):

  $s1 \models$ opened-door1 $\wedge \neg$ opened-door2

  $s2 \models$ opened-door2 $\wedge \neg$ opened-door1

- Merge: {s1, s2}

- Init:

  K opened-door1/s1 $\wedge$ K $\neg$ opened-door2/s1

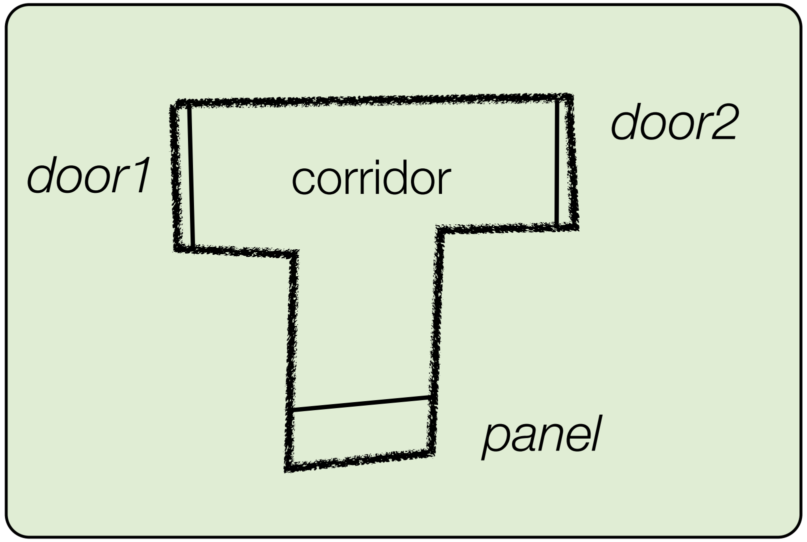
  K opened-door2/s2 $\wedge$ K $\neg$ opened-door1/s2

  K at(corridor)/* $\wedge$ K$\neg$gold-found/* $\wedge$ $\neg$K...

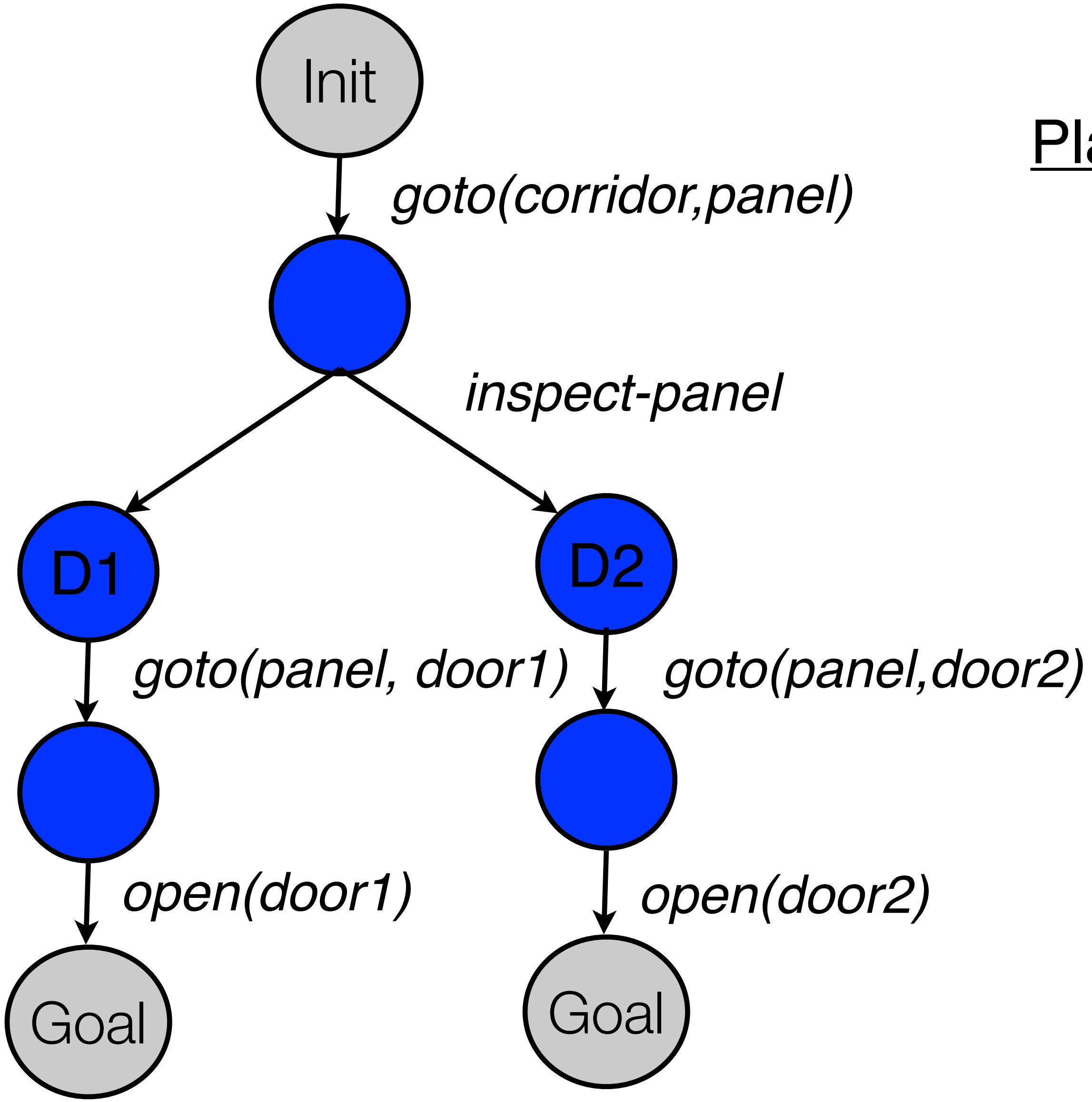
Example with X_{S0} (P) translation
A possible Plan

Plan:

goto(panel),
inspect-panel,
goto(observed-open-door),
open(observed-open-door).

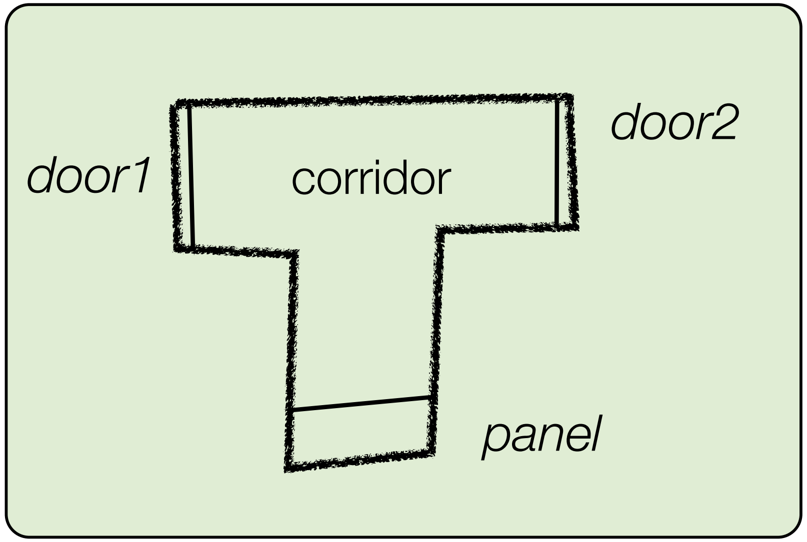


Example with X_{S0} (P) translation
A possible Plan

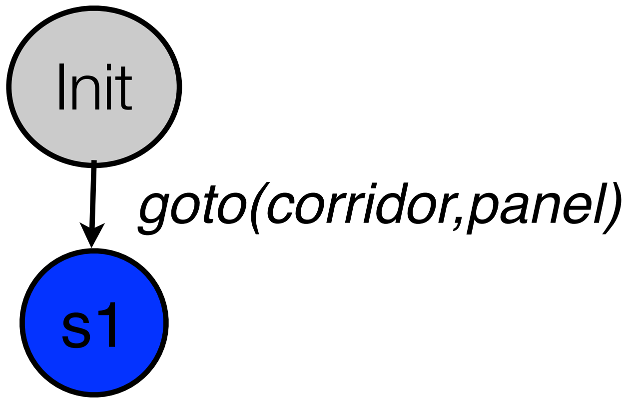


<u>Plan</u>:

goto(panel),
inspect-panel,
goto(observed-open-door),
open(observed-open-door).



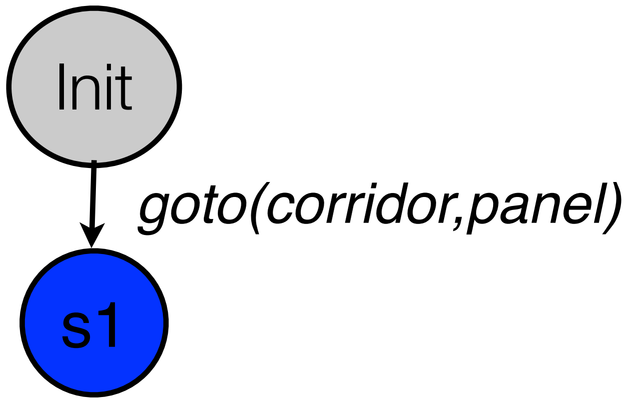
Example with X_{S0} (P) translation goto(corridor, panel)



Init:

K at(corridor)/s1 ∧
K at(corridor)/s2 ∧ *K at(corridor)*
∧ *...*

Example with X_{S0} (P) translation goto(corridor, panel)



Init:

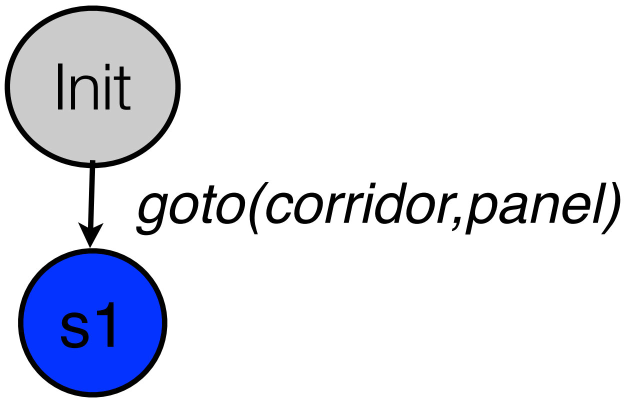
K at(corridor)/s1 ∧
K at(corridor)/s2 ∧ *K at(corridor)*
∧ ...

goto(corridor, panel):

pre: *K at(corridor)*

effect: *K at(panel)* ∧ *K¬at(corridor)* ∧ *K at(panel)/ t* ∧
K¬at(corridor)/ t ∧ ...

Example with X_{S0} (P) translation goto(corridor, panel)



s1:

K at(panel)/s1 ∧

K at(panel)/s2 ∧ *K at(panel)*

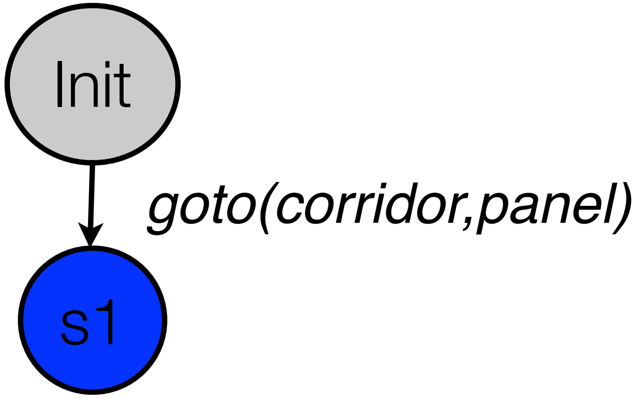
∧ ...

goto(corridor, panel):

pre: *K at(corridor)*

effect: *K at(panel)* ∧ *K¬at(corridor)* ∧ *K at(panel)/ t* ∧ *K¬at(corridor)/ t* ∧ ...

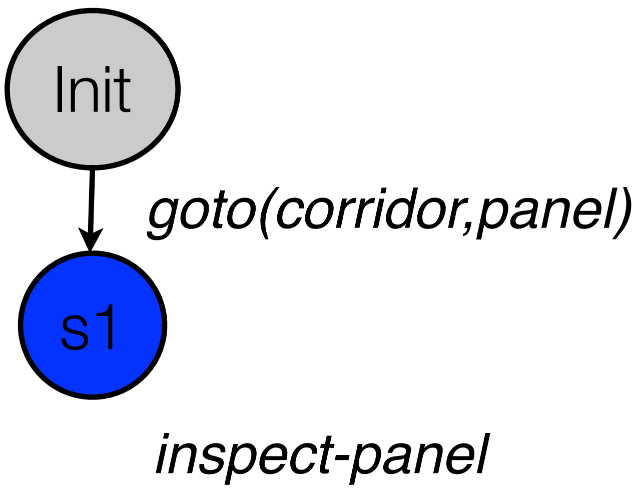
Example with X_{S0} (P) translation inspect-panel



s1:

K at(panel)/s1 ∧ *K at(panel)/s2* ∧ *K at(panel)*∧ *...*

Example with X_{S0} (P) translation inspect-panel



<u>s1</u>:

$K\ at(panel)/s1 \wedge K\ at(panel)/s2 \wedge$
$K\ at(panel) \wedge \ldots$

inspect-panel

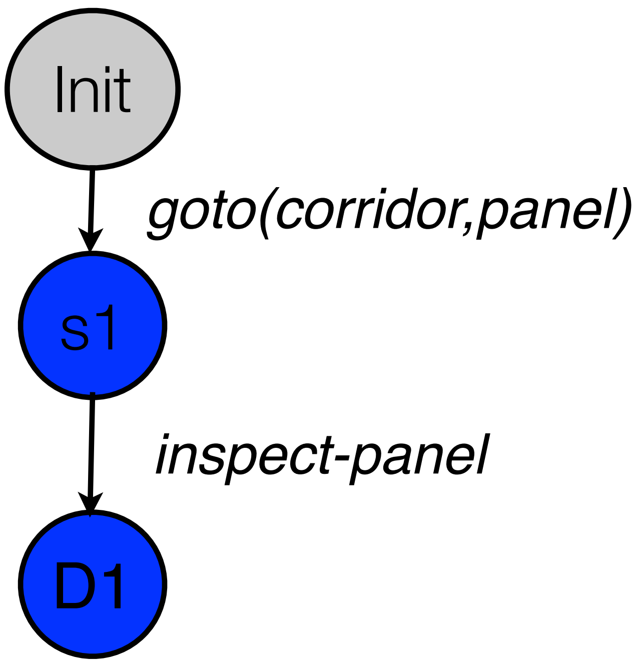
pre: $K\ at(panel)$

observation:

$\neg K\ opened\text{-}door1 \wedge \neg K \neg\ opened\text{-}door1$

$\rightarrow \quad K\ opened\text{-}door1 \mid K \neg\ opened\text{-}door1$

Example with X_{S0} (P) translation inspect-panel



D1:

K at(panel)/s1 ∧ *K at(panel)/s2* ∧
K at(panel)∧ *¬K¬ opened-door1* ∧
K opened-door1 ∧...

inspect-panel

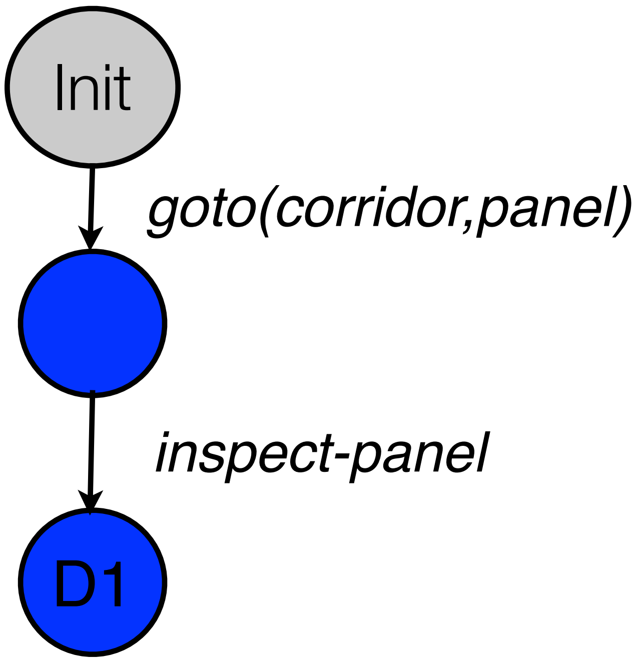
pre: *K at(panel)*

observation:

¬K opened-door1 ∧ *¬K¬ opened-door1*

→ *K opened-door1* | *K ¬ opened-door1*

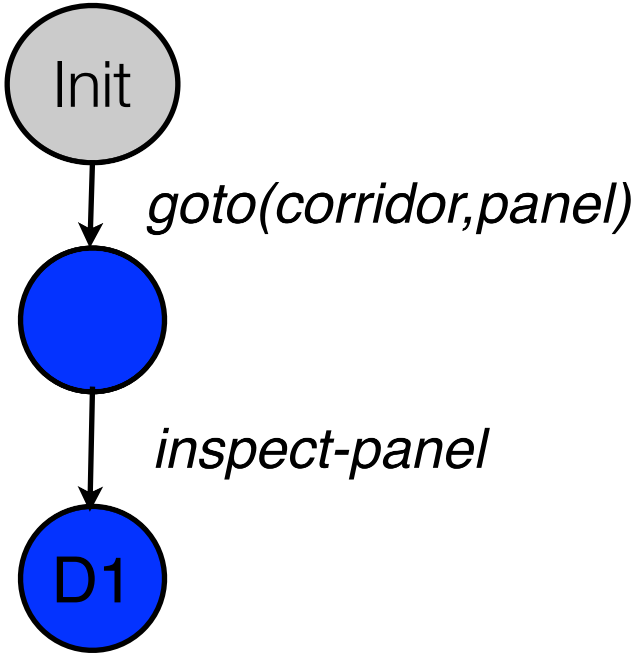
Example with X_{S0} (P) translation tag-refutation: $KL/t \wedge K \neg L \rightarrow K \neg t$



D1:

K at(panel)/s1 $\wedge$ *K at(panel)/s2* $\wedge$
K at(panel) $\wedge$ *K opened-door1* $\wedge$
K¬opened-door1/s2 $\wedge$ *...*

Example with X_{S0} (P) translation tag-refutation: $KL/t \wedge K\neg L \rightarrow K\neg t$



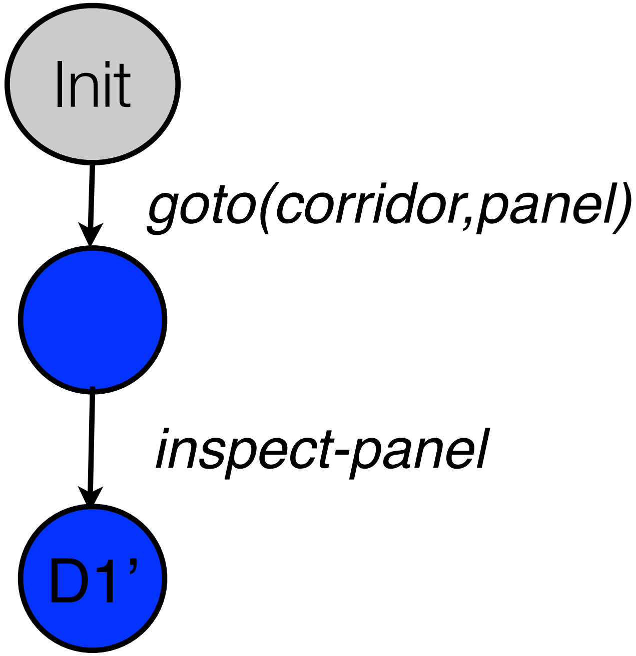
D1:
$K\ at(panel)/s1 \wedge K\ at(panel)/s2 \wedge$
$K\ at(panel) \wedge K\ opened\text{-}door1 \wedge$
$K\neg opened\text{-}door1/s2 \wedge \ldots$

tag-refutation

pre: *true*

effect:
$K\neg opened\text{-}door1/s2 \wedge K\ opened\text{-}door1 \rightarrow K\neg s2$

Example with X_{S0} (P) translation tag-refutation: $KL/t \wedge K \neg L \rightarrow K\neg t$



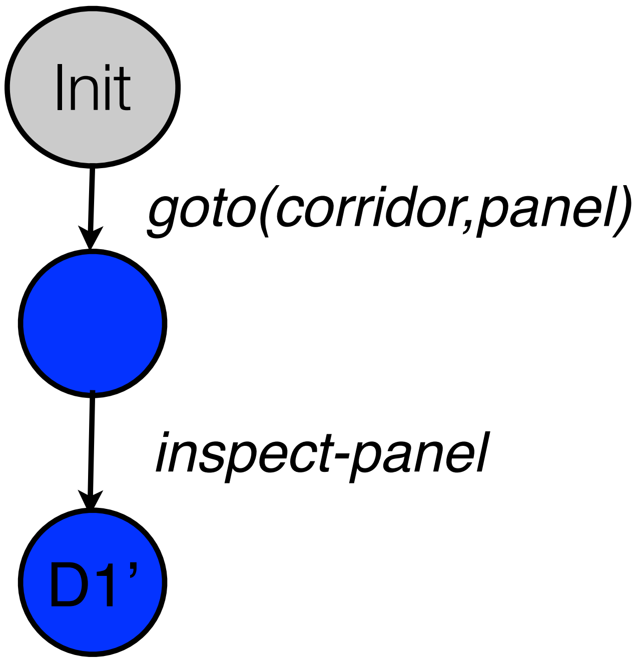
D1':
K at(panel)/s1 ∧ *K at(panel)/s2* ∧
K at(panel)∧ *K opened-door1* ∧
K¬opened-door1/s2 ∧ *K ¬s2* ∧...

tag-refutation

pre: *true*

effect:
K¬opened-door1/s2 ∧ *K opened-door1* → *K ¬ s2*

Example with X_{S0} (P) translation tag-refutation: $KL/t \wedge K\neg L \rightarrow K\neg t$



D1':
$K\ at(panel)/s1 \wedge K\ at(panel)/s2 \wedge$
$K\ at(panel) \wedge K\ opened\text{-}door1 \wedge$
$K\neg opened\text{-}door1/s2 \wedge K\neg s2 \wedge \ldots$

...and from now on, no uncertainty is left
⇒classical planning problem (solved like K_0)

General Translations that are Complete

- Let O(L) be the observables relevant to L.
- Let $C_I^O(L)$ be the clauses in $\mathcal{I}$ relevant to L or O(L).
- $\mathcal{I}$ is assumed to be in **prime implicate form**.

Definition: A valid translation $X_{T,M}(P)$ is **covering** if
for each precondition and goal literal L of P,
M contains a merge m for L that satisfies each clause in $C_I^O(L)$.

Theorem: Covering translations are **sound** and **complete**.

Width and Complexity

- Width of a problem $w(P)$ is roughly the **size of the tags** needed for completeness.
- The translation $X_i(P)$ is a special case of $X_{T,M}(P)$, with tags of size $\leq i$.
- For fixed i, translation $X_i(P)$ is **polynomial**, and **complete** if $w(P) \leq i$.
- Most contingent benchmarks turn out to have **width 1**.

where are we?

where are we?

- $X_{T,M}(P)$, fully-observable non-deterministic problem, done

where are we?

- $X_{T,M}(P)$, fully-observable non-deterministic problem, done
- Relaxation $\mathbf{X^+(P)}$ to guide the search

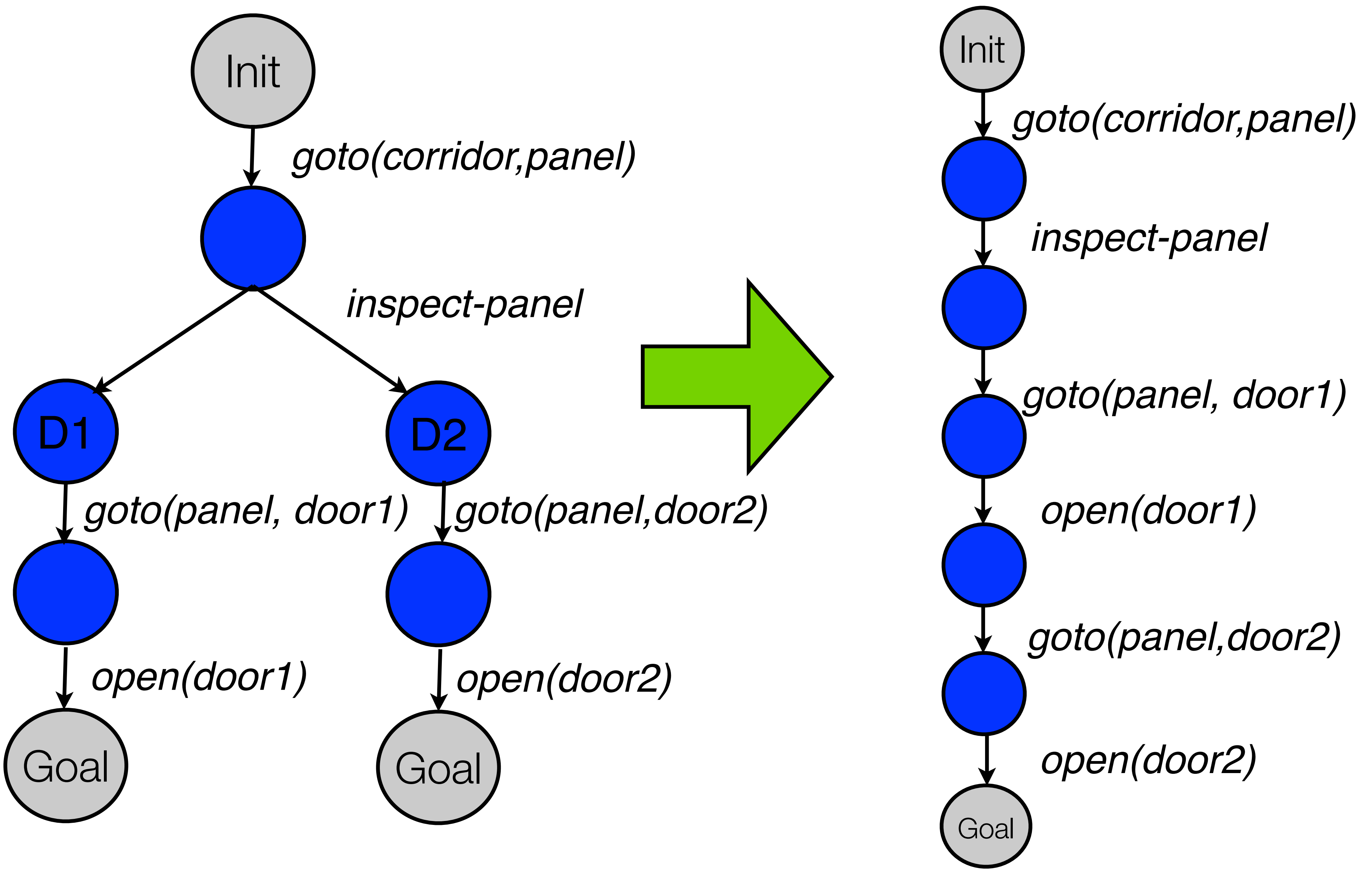
Relaxation *$X^+(P)$*

- Drop "delete" effects, (like in classical planning).
- Move preconditions in as conditions [Hoffmann & Brafman, 2005].
- Make sensing actions obs(L) **deterministic**, by adding contingent knowledge operator M:

$$obs(L) : \neg KL \wedge \neg K\neg L \rightarrow ML \wedge M\neg L \wedge o(L)$$

- Use M-literal ML as preconditions of action a in $X_{T,M}(P)$, if L is precondition of a in P.
- $X^+(P)$ is a **classical planning problem.** Solutions for $X_{T,M}(P)$ are solutions $X^+(P)$.

Relaxing on action preconditions



Example with $X^{+}(P)$

Effects of an observation in $X^{+}(P)$

<u>S</u>:
K at(panel)/opened-door1 $\wedge$
K at(panel)/opened-door2 $\wedge$
K at(panel)$\wedge$ *M at(panel)* $\wedge$...

Example with $X^+(P)$

Effects of an observation in $X^+(P)$

S:
K at(panel)/opened-door1 ∧
K at(panel)/opened-door2 ∧
K at(panel)∧ *M at(panel)* ∧...

inspect-panel:

pre: *M at(panel)*

observation:
¬*Kopened-door1* ∧ ¬*K*¬ *opened-door1*
→ *M opend-door1* ∧ *M* ¬ *opened-door1* ∧ *o(opened-door1)*

Example with $X^+(P)$

Effects of an observation in $X^+(P)$

added by M-K rule:
KL → ML

S:
K at(panel)/opened-door1 ∧
K at(panel)/opened-door2 ∧
K at(panel)∧ *M at(panel)* ∧...

inspect-panel:

pre: *M at(panel)*

observation:
¬Kopened-door1 ∧ *¬K¬ opened-door1*
→ *M opend-door1* ∧ *M ¬ opened-door1* ∧ *o(opened-door1)*

Example with $X^{+}(P)$

Effects of an observation in $X^{+}(P)$

<u>S'</u>:
K at(panel)/opened-door1 $\wedge$
K at(panel)/opened-door2 $\wedge$ *K at(panel)*$\wedge$
M at(panel) $\wedge$ *M opened-door1* $\wedge$
M ¬ *opened-door1* $\wedge$ *o(opened-door1)* $\wedge$ ...

inspect-panel:

pre: *M at(panel)*

observation:
¬*Kopened-door1* $\wedge$ ¬*K*¬ *opened-door1*
→ *M opend-door1* $\wedge$ *M* ¬ *opened-door1* $\wedge$ *o(opened-door1)*

Example with $X^+(P)$ applying derivation rules

S':
K at(panel) $\wedge$ *M at(panel)* $\wedge$ *M gold-at(door1)* $\wedge$
M $\neg$ *gold-at(door1)* $\wedge$ *o(gold-at(door1))* $\wedge$ ...

Example with $X^+(P)$ applying derivation rules

S':
K at(panel) $\wedge$ *M at(panel)* $\wedge$ *M gold-at(door1)* $\wedge$
M $\neg$ *gold-at(door1)* $\wedge$ *o(gold-at(door1))* $\wedge$ ...

M-contingent merge:

effect:
M$\neg$ *opened-door1* $\rightarrow$ *M opened-door2*

Example with $X^+(P)$ applying derivation rules

<u>S''</u>:

K at(panel) ∧ *M at(panel)* ∧ *M gold-at(door1)* ∧
M ¬ gold-at(door1) ∧ *o(gold-at(door1))* ∧
M gold-at(door2) ∧ ...

M-contingent merge:

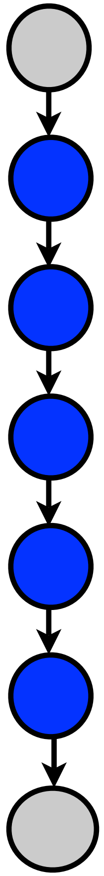
effect:

M¬ opened-door1 → *M opened-door2*

Example with $X^+(P)$
a possible plan

- In $X^+(P)$, the preconditions of the actions open(door1) and open(door2) hold in the relaxed translation.

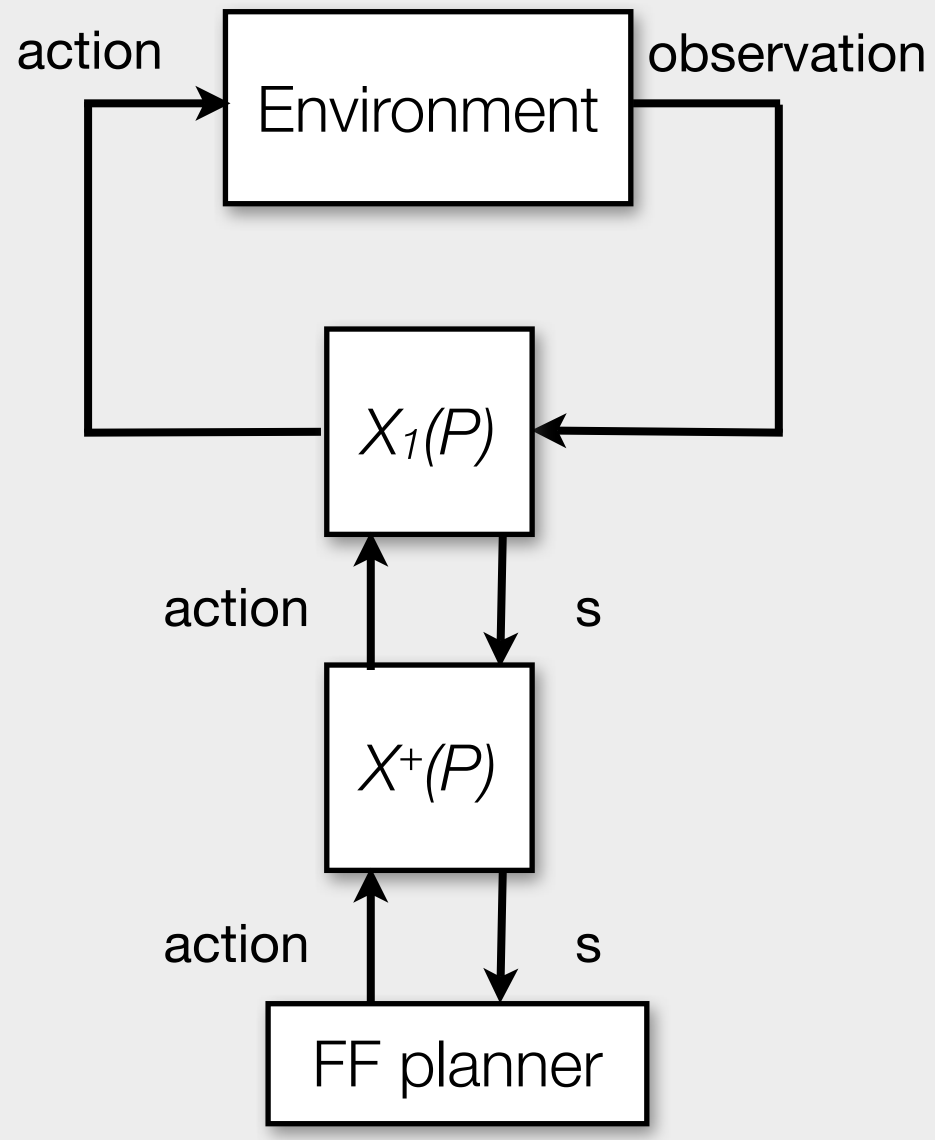
- A solution plan would be, from Init:
  1. goto(corridor, panel)
  2. inspect-panel, (observation)
  3. goto(panel, door1)
  4. open(door1) → K gold-found/opened-door1
  5. goto(door1,door2)
  6. open(door2) → K gold-found/opened-door2



- After last action, the goal would be reached because of merge rule: *K gold-found/opened-door1* ∧ *K gold-found/opened-door2* → *K gold-found*

Closed Loop Greedy Planner

- The CLG planner uses:
  - translation $X_1(P)$ to **keep track of beliefs**;
  - relaxation $X_1^+(P)$, that is a classical planning problem, to **select action to do next.**



Using assumptions on sensing outcome

- **Freespace** assumption [Koenig at al. 2003]
- **Safe Assumption-based planning**: belief monitoring and LTL assumptions [Albore & Bertoli 2006]
- **Preferences** on observation outcome [Likhachev & Stentz 2009]
- **Sampling** and replanning [Shani & Brafman 2011]
    - Based on CLG's and T0's ideas

Another approach on how to Solve contingent problems with classical planners

- Conditions under wich partially observable problems can be solved by classical planners.
- **Simple problem** [Bonet & Geffner 2011]:
  - non-unary clauses in Init are all **invariant**
  - no hidden fluent appear in the body of a conditional effect
- **Width of P = 1**
- **Connected** space.

Planning under optimism

- K'(P) fully-observable non-deterministic problem (based on K_0) **solved by a classical translation** K(P), using 2 rules:

- Assumption: if (C,L) is a sensing action, then

  pre: $KC \land \neg KL \land \neg K\neg L$ effect: KL

  pre: $KC \land \neg KL \land \neg K\neg L$ effect: $K\neg L$

- $KC \rightarrow KL$ for invariants $\neg C \lor L$ in P

- A prefix of the plan is always executable, until KC is achieved. Then the assumption can be revealed by sensing.

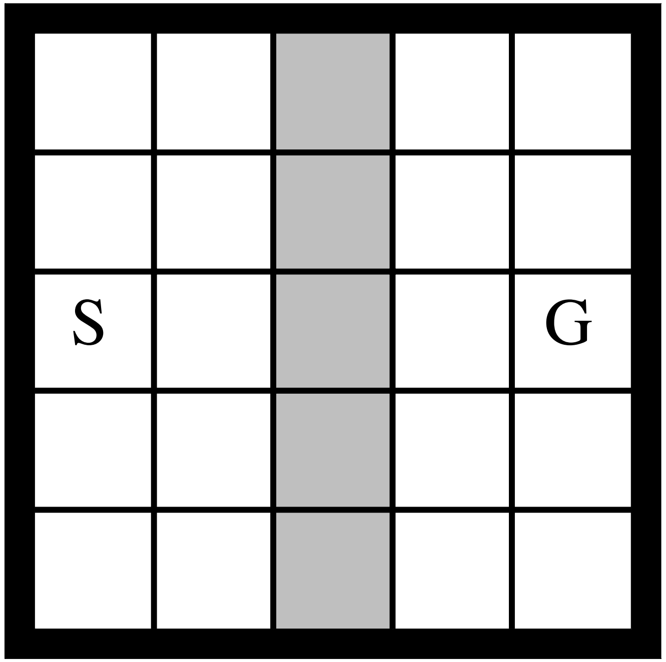
Planning under optimism

- If the assumption turns out to be false, then **replan**
- If the space is connected, replanning is always possible, and reaching the goal is guaranteed if a solution exists.

Dead-ends [Albore & Geffner 2009]

- Dead-ends are situations for which there is no strong solution:
  - Belief state is a dead-end when **at least one state** is a dead-end.
  - State is a dead-end when the goal cannot be reached **even given full observability** (e.g. minesweeper).
- Contingent and POMDPs planners will deliver **no solution** when initial belief is a dead-end.
- Yet these situations are quite common...

Example with no full solution plan



- The cells in the middle column can be blocked.
- 2^5 possible wall configurations.

- **Only 1 wall configuration** brings to a dead-end situation.
- Full contingent solution is however non-existent.

Planners for Problems with No Strong Solutions

- When there is not a strategy that works in **all** cases, we may look for a strategy working in **most** cases.

- Non-solvable contingent planning problems can be converted into solvable ones by introducing **assumptions**.

- The aim is finding a solution for the maximum number of states in the belief state.

- **CLG$^+$** = CLG$_{online}$ + **assumptive-actions** + **costs**.

Encoding Assumptions Into CLG$^+$ (pay-for-tags)

- **Assuming** ***K¬ t*** to make it possible to merge *KL*

$$\bigwedge_{t \in m, m \in M_L} (KL/t \vee K\neg t) \;\rightarrow\; KL$$

- Assumptive actions are encoded in $X_{T,M}(P)$ as deterministic actions with **high cost**: $\neg Kt \wedge \neg K\neg t \;\rightarrow\; K\neg t$

- Consequences:
  - Plans with assumption are the last option when generating relaxed plans,
  - Thus cost optimisation will result in plan strategies that are as strong as possible.

Use of Assumptions

- Assumptions are integrated in 3 steps, for action selection:
  1. Don’t use them.
  2. Use in relaxed plan but not execute it (ie. excluded from helpful actions of FF). Observations can help later.
  3. Allow to execute them as last resort.
     Like “betting”, taking a risk.

Problems with Dead-end States

- Some situations might be dead-end.
- These problems are not solvable by existing contingent or POMDPs planners (infinite heuristic)
- **Examples**: Wumpus, Navigation in Unknown Map, Learning Unknown Model when observation allow to uncover action effects.

Problems with Pure Dead-end States

- **Insoluble problems** even if no state is initially dead-end any policy will work for some states, but not for others.
- These problems are solved by "**betting**", executing an assumption, to get out from the impasse.
- In case the bet is wrong, the execution naturally fails.
- But it is a risk that has to be taken.
- **Example**: Minesweeper, certain instances of Wumpus domain

Problems with High Contingent Width

- Contingent width is a **measure of the complexity** of the problem.

- Roughly, the size of the tags needed to have a **complete translation** $X_{T,M}(P)$.

- Problems with a high contingent width are **solvable problems** with a contingent width > 1

- **Example**: Binary tree of doors.

Summary of Second Part

- Translation $X_{T,M}(P)$ for contingent problems
- Conditions for completeness and contingent width
- Heuristic relaxation $X^+(P)$
- Solving contingent problems with classical planners
- Dealing with dead-ends, CLG+ planner

Summary of the tutorial

- The presented approaches for conformant and contingent planning relies on:
  - Translate problems into classical planning.
  - Use such translations for action application and to obtain useful heuristics to guide the search.
  - In the case of conformant planning, both action applications and heuristics were done simoultaniously.

Conclusions

- Translation-based approach has a **clear semantics** including:
  - Conditions for **completness** and **soundness**;
  - Structural properties characterizing the size of complete translations (width).
- Planners based on complete and sound translations are **competitive**.
- Better performance can be obtained by
  - focusing on **special cases** (‘simple’ problems, with dead-ends)
  - obtaining **heuristics** from unsound but feasible translations