Temporally-Expressive Planning as Constraint Satisfaction Problems

Yuxiao Hu

Department of Computer Science
University of Toronto

yuxiao (a) cs toronto edu

September 25, 2007

Motivation

As shown by Cushing *et al.* (2007), there are "temporally-expressive" planning problems that

- can be represented by PDDL 2.x
- cannot be solved by many *state-of-the-art* planners

This is due to their strong assumptions on

1. temporal annotation
   (Over-all preconditions, at end effects)
2. decision epochs
   (An action can happen only when another event is happening)

Motivation

As shown by Cushing *et al.* (2007), there are "temporally-expressive" planning problems that

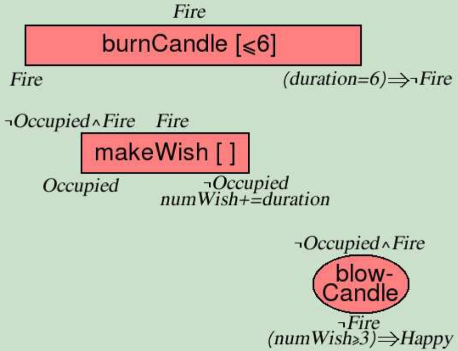
- can be represented by PDDL 2.x
- cannot be solved by many *state-of-the-art* planners

This is due to their strong assumptions on

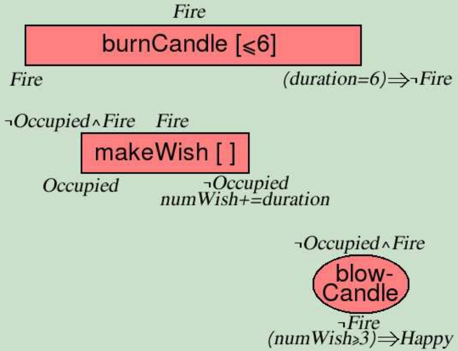
1. temporal annotation
   (Over-all preconditions, at end effects)
2. decision epochs
   (An action can happen only when another event is happening)

Goal: Solve general PDDL planning problems with a unified approach

A Running Example



A Running Example



- Required concurrency ("*makeWish*" be contained within "*burnCandle*")
- Duration inequality and duration-related effects

Approach

Our approach consists of two steps:

1. PDDL $\Longrightarrow$ BAT (Basic Action Theory)
   Based on a concurrent extension to the situation-calculus semantics of PDDL (Claßen *et al.* 2007)
2. BAT $\Longrightarrow$ CSP
   Encode the BAT into a CSP problem, and solve the CSP to obtain the plan.

Approach

Our approach consists of two steps:

1. PDDL $\Longrightarrow$ BAT (Basic Action Theory)

   Based on a concurrent extension to the situation-calculus semantics of PDDL (Claßen *et al.* 2007)

2. BAT $\Longrightarrow$ CSP

   Encode the BAT into a CSP problem, and solve the CSP to obtain the plan.

The intuition behind it is to

- model durative actions with simple (instantaneous) actions
- treat time as a numerical property, and advance it with constraints.

Logical Foundations

Concurrent temporal situation calculus (Reiter 2001) with the following syntax

- $A(\vec{x}, t)$ denotes the happening of simple action $A(\vec{x})$ at time t
- A durative action $A(\vec{x})$ is represented by
  - Action $start(A(\vec{x}), t)$: the start event of $A(\vec{x})$ at time t
  - Action $end(A(\vec{x}), t)$: the end event of $A(\vec{x})$ at time t
  - Predicate $Performing(A(\vec{x}))$: whether $A(\vec{x})$ is in progress
  - Function $since(A(\vec{x}))$: the last starting time of $A(\vec{x})$
- $\{a_1, \cdots, a_n\}$ means the concurrent happening of a_i
- $[c]\alpha$ means α holds after a list c of concurrent actions
- $\Box\alpha$ means α holds in any situation
- *now* is a special functional fluent representing the current time

The Basic Action Theory

The basic action theory Σ consists of

- The initial database Σ_0 *e.g.*:
  $\neg Fire,\ numWish = 0,\ \neg Performing(burnCandle),\ now = 0;$
- The precondition axiom Σ_{pre}, obtained from, *e.g.*:
  $\Box Poss(end(burnCandle, t)) \supset (t - since(burnCandle)) \leq 6;$
- The successor state axioms Σ_{post}, *e.g.*:
  $\Box[c]Fire \equiv \exists t.start(burnCandle, t) \in c \vee$
  $Fire \wedge \neg(\ \exists t.end(burnCandle, t) \in c \wedge (t - since(burnCandle) = 6) \vee$
  $\exists t.blowCandle(t) \in c);$
- The unique names axioms Σ_{una};
- The foundational axioms $\mathcal{FA}$, *e.g.*:
  $\Box Poss(c) \supset now < time(c)$

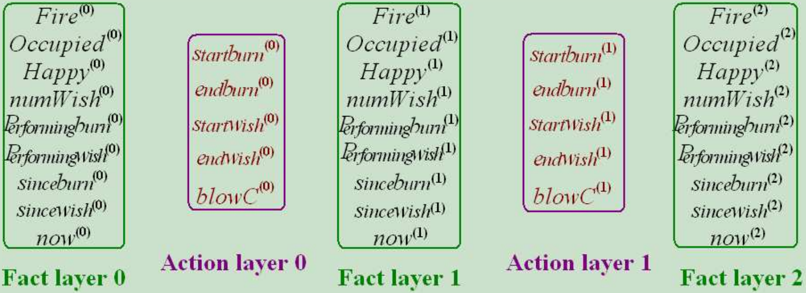
The Variable Structure

Planning by encoding the basic action theory into a CSP

- Search for increasing plan length $n = 1, 2, 3, \cdots$, where "length" means the number of concurrent happenings.
- When searching for a plan of length n, create
  - n boolean variables for each ground action term $A(\vec{o})$: $A_{\vec{o}}^{(0)}, \cdots, A_{\vec{o}}^{(n-1)}$
  - $n+1$ boolean variables for each ground predicate $P(\vec{o})$: $P_{\vec{o}}^{(0)}, \cdots, P_{\vec{o}}^{(n)}$
  - $n+1$ numerical variables for each ground function $f(\vec{o})$: $f_{\vec{o}}^{(0)}, \cdots, f_{\vec{o}}^{(n)}$

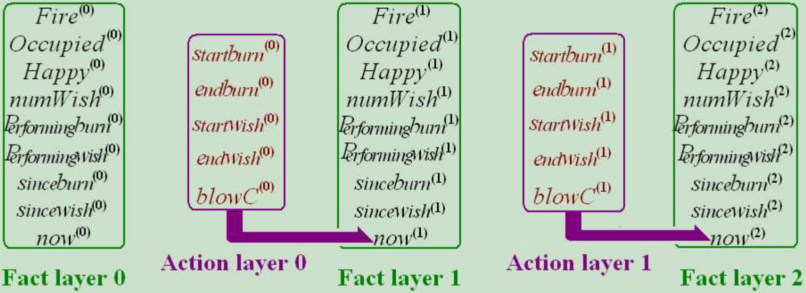
The Variable Structure

An example of searching for a plan of length 2:



The Variable Structure

An example of searching for a plan of length 2:



Constraints: Initial and Goal States

The 0th fact layer encodes the initial state

- Problem-specific fluents set according to the initial description
- Auxiliary fluents set to 0 (or FALSE)

The last fact layer encodes the goal condition

- Problem-specific goals
- All "Performing" variables must be false

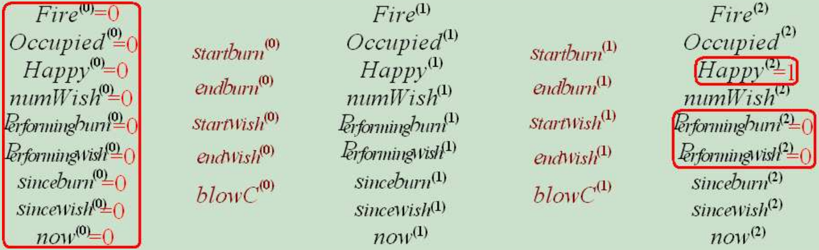
Constraints: Initial and Goal States

The 0th fact layer encodes the initial state

- Problem-specific fluents set according to the initial description
- Auxiliary fluents set to 0 (or FALSE)

The last fact layer encodes the goal condition

- Problem-specific goals
- All "Performing" variables must be false



Constraints: Action Preconditions

For each formula of the form $\Box Poss(A) \supset \pi_A$, construct an action precondition constraint, *e.g.*:

From (part of) precondition axiom in the BAT

$$\Box Poss(end(burnCandle, t)) \supset (t - since(burnCandle)) \leq 6$$

we obtain the action precondition constraint

$$e_{nd}burn^{(i)} \supset (now^{(i+1)} - s_{ince}burn^{(i)}) \leq 6$$

Constraints: Action Preconditions

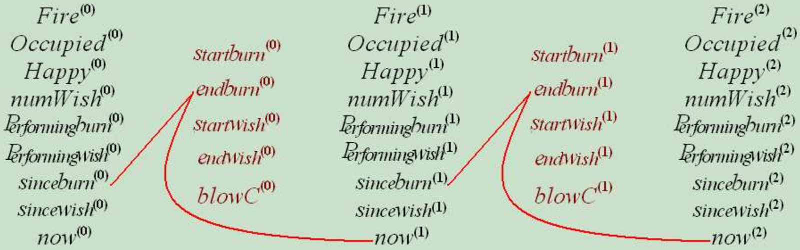
For each formula of the form $\Box Poss(A) \supset \pi_A$, construct an action precondition constraint, *e.g.*:

From (part of) precondition axiom in the BAT

$$\Box Poss(end(burnCandle, t)) \supset (t - since(burnCandle)) \leq 6$$

we obtain the action precondition constraint

$$e_{nd}burn^{(i)} \supset (now^{(i+1)} - s_{ince}burn^{(i)}) \leq 6$$



Constraints: Successor States

For each formula of the form $\Box[c]F \equiv \Phi_F$, construct a successor state constraint, *e.g.*:

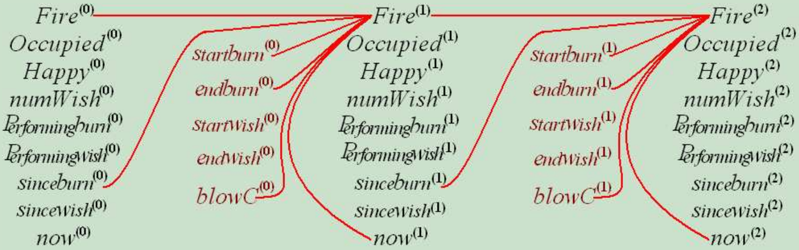
From the successor state axiom in the BAT

$$\Box[c]Fire \equiv \exists t.start(burnCandle, t) \in c \lor$$
$$Fire \land \neg(\exists t.end(burnCandle, t) \in c \land (t - since(burnCandle) = 6) \lor$$
$$\exists t.blowCandle(t) \in c)$$

we obtain the successor state constraint

$$Fire^{(i+1)} \equiv s_{tart}burn^{(i)} \lor$$
$$Fire^{(i)} \land \neg(e_{nd}burn^{(i)} \land (now^{(i+1)} - s_{ince}burn^{(i)} = 6) \lor blowC^{(i)})$$

Constraints: Successor States



$$Fire^{(i+1)} \equiv s_{tart}burn^{(i)} \vee$$

$$Fire^{(i)} \wedge \neg(e_{nd}burn^{(i)} \wedge (now^{(i+1)} - s_{ince}burn^{(i)} = 6) \vee blowC^{(i)})$$

Constraints: Action Happening Times

To ensure chronological order of happenings, the foundational axiom in BAT

$$\Box Poss(c) \supset now < time(c)$$

is encoded as the action happening time constraint

$$now^{(i)} < now^{(i+1)}$$

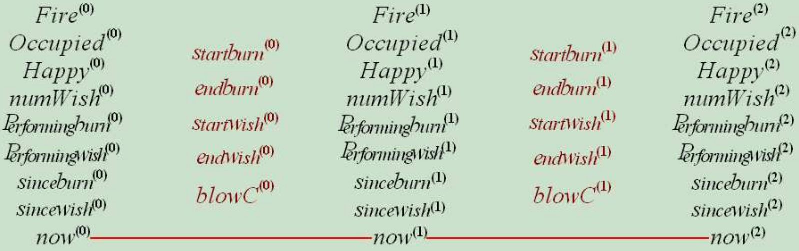
Constraints: Action Happening Times

To ensure chronological order of happenings, the foundational axiom in BAT

$$\Box Poss(c) \supset now < time(c)$$

is encoded as the action happening time constraint

$$now^{(i)} < now^{(i+1)}$$



Other Constraints

In addition, we need the following constraints (details in paper)

- Invariant constraints;
- Non-null step constraints;
- Mutex constraints;
- Timed-initial literal enforcement constraints.

Implementation and Test Result

We hand-encoded several problem with constraint programming language *Choco* (choco.sourceforge.net), which supports

1. Higher-order constraints, *e.g.* implies(A, and(B,C));
2. Numerical constraints.

Can solve problems with required concurrency, duration inequalities and duration-related effects.

Example Result

$$Fire^{(0)}=0 \quad Occupied^{(0)}=0 \quad Happy^{(0)}=0 \quad numWish^{(0)}=0 \quad Performingburn^{(0)}=0 \quad PerformingWish^{(0)}=0 \quad sinceburn^{(0)}=0 \quad sinceWish^{(0)}=0 \quad now^{(0)}=0$$

$$Startburn^{(0)}=1 \quad endburn^{(0)}=0 \quad StartWish^{(0)}=0 \quad endWish^{(0)}=0 \quad blowC^{(0)}=0$$

$$Fire^{(1)}=1 \quad Occupied^{(1)}=0 \quad Happy^{(1)}=0 \quad numWish^{(1)}=0 \quad Performingburn^{(1)}=1 \quad PerformingWish^{(1)}=0 \quad sinceburn^{(1)}=1 \quad sinceWish^{(1)}=0 \quad now^{(1)}=1$$

$$Startburn^{(1)}=0 \quad endburn^{(1)}=0 \quad StartWish^{(1)}=1 \quad endWish^{(1)}=0 \quad blowC^{(1)}=0$$

$$Fire^{(2)}=1 \quad Occupied^{(2)}=1 \quad Happy^{(2)}=0 \quad numWish^{(2)}=0 \quad Performingburn^{(2)}=1 \quad PerformingWish^{(2)}=1 \quad sinceburn^{(2)}=1 \quad sinceWish^{(2)}=2 \quad now^{(2)}=2$$

$$Startburn^{(2)}=0 \quad endburn^{(2)}=0 \quad StartWish^{(2)}=0 \quad endWish^{(2)}=1 \quad blowC^{(2)}=0$$

$$Fire^{(3)}=1 \quad Occupied^{(3)}=0 \quad Happy^{(3)}=0 \quad numWish^{(3)}=3 \quad Performingburn^{(3)}=1 \quad PerformingWish^{(3)}=0 \quad sinceburn^{(3)}=1 \quad sinceWish^{(3)}=2 \quad now^{(3)}=5$$

$$Startburn^{(3)}=0 \quad endburn^{(3)}=1 \quad StartWish^{(3)}=0 \quad endWish^{(3)}=0 \quad blowC^{(3)}=1$$

$$Fire^{(4)}=0 \quad Occupied^{(4)}=0 \quad Happy^{(4)}=1 \quad numWish^{(4)}=3 \quad Performingburn^{(4)}=0 \quad PerformingWish^{(4)}=0 \quad sinceburn^{(4)}=1 \quad sinceWish^{(4)}=2 \quad now^{(4)}=6$$

Example Result

$Fire^{(0)}=0$
$Occupied^{(0)}=0$
$Happy^{(0)}=0$
$numWish^{(0)}=0$
$Performingburn^{(0)}=0$
$PerformingWish^{(0)}=0$
$sinceburn^{(0)}=0$
$sinceWish^{(0)}=0$
$now^{(0)}=0$

$Startburn^{(0)}=1$
$endburn^{(0)}=0$
$StartWish^{(0)}=0$
$endWish^{(0)}=0$
$blowC^{(0)}=0$

$Fire^{(1)}=1$
$Occupied^{(1)}=0$
$Happy^{(1)}=0$
$numWish^{(1)}=0$
$Performingburn^{(1)}=1$
$PerformingWish^{(1)}=0$
$sinceburn^{(1)}=1$
$sinceWish^{(1)}=0$
$now^{(1)}=1$

$Startburn^{(1)}=0$
$endburn^{(1)}=0$
$StartWish^{(1)}=1$
$endWish^{(1)}=0$
$blowC^{(1)}=0$

$Fire^{(2)}=1$
$Occupied^{(2)}=1$
$Happy^{(2)}=0$
$numWish^{(2)}=0$
$Performingburn^{(2)}=1$
$PerformingWish^{(2)}=1$
$sinceburn^{(2)}=1$
$sinceWish^{(2)}=2$
$now^{(2)}=2$

$Startburn^{(2)}=0$
$endburn^{(2)}=0$
$StartWish^{(2)}=0$
$endWish^{(2)}=1$
$blowC^{(2)}=0$

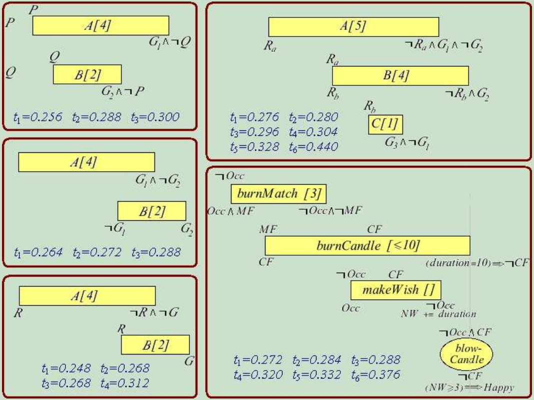
$Fire^{(3)}=1$
$Occupied^{(3)}=0$
$Happy^{(3)}=0$
$numWish^{(3)}=3$
$Performingburn^{(3)}=1$
$PerformingWish^{(3)}=0$
$sinceburn^{(3)}=1$
$sinceWish^{(3)}=2$
$now^{(3)}=5$

$Startburn^{(3)}=0$
$endburn^{(3)}=1$
$StartWish^{(3)}=0$
$endWish^{(3)}=0$
$blowC^{(3)}=1$

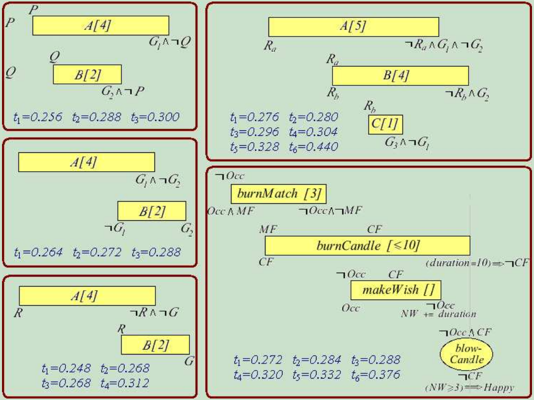
$Fire^{(4)}=0$
$Occupied^{(4)}=0$
$Happy^{(4)}=1$
$numWish^{(4)}=3$
$Performingburn^{(4)}=0$
$PerformingWish^{(4)}=0$
$sinceburn^{(4)}=1$
$sinceWish^{(4)}=2$
$now^{(4)}=6$

PDDL plan: $\{(1 : burnCandle[5]), (2 : makeWish[3]), (6 : blowCandle)\}$

Other Experiments



Other Experiments



- No comparison with *state-of-the-art* planners available yet.
- Presumably slower on existing (temporally-simple) benchmarks, since we have only focused on generality, not yet efficiency.

Conclusion and Future Work

We have

1. extended a declarative semantics of PDDL with true concurrency
2. proposed a general solution to PDDL planning problems based on a CSP encoding of the declarative semantics
   - Handles arbitrary PDDL temporal annotations
   - Determines happening times by satisfying constraints
   - Solves **temporally-expressive** PDDL problems (possibly with **duration-related constraints** and **effects**) in a **unified search**.

Conclusion and Future Work

We have

1. extended a declarative semantics of PDDL with true concurrency
2. proposed a general solution to PDDL planning problems based on a CSP encoding of the declarative semantics
   - Handles arbitrary PDDL temporal annotations
   - Determines happening times by satisfying constraints
   - Solves **temporally-expressive** PDDL problems (possibly with **duration-related constraints** and **effects**) in a **unified search**.

Future work:

1. Implementation and optimization of automatic translation
2. Constraints and preferences in PDDL 3.0