Chapter 14
Temporal Planning

Dana S. Nau

CMSC 722, AI Planning
University of Maryland, Spring 2008
Temporal Planning

- Motivation: want to do planning in situations where actions
  - have nonzero duration
  - may overlap in time
- Need an explicit representation of time

- In Chapter 10 we studied a “temporal” logic
  - Its notion of time is too simple: a sequence of discrete events
  - Many real-world applications require continuous time
  - How to get this?
Temporal Planning

- The book presents two equivalent approaches:
  1. Use logical atoms, and extend the usual planning operators to include temporal conditions on those atoms
     » Chapter 14 calls this the “state-oriented view”
  2. Use state variables, and specify change and persistence constraints on the state variables
     » Chapter 14 calls this the “time-oriented view”
- In each case, the chapter gives a planning algorithm that’s like a temporal-planning version of PSP
The Time-Oriented View

- We’ll concentrate on the “time-oriented view”: Sections 14.3.1–14.3.3
  - It produces a simpler representation
  - State variables seem better suited for the task
- States not defined explicitly
  - Instead, can compute a state for any time point, from the values of the state variables at that time
State Variables

● A state variable is a partially specified function telling what is true at some time $t$
  ◆ $\text{cpos}(c1) : \text{time} \rightarrow \text{containers} \cup \text{cranes} \cup \text{robots}$
    » Tells what $c1$ is on at time $t$
  ◆ $\text{rloc}(r1) : \text{time} \rightarrow \text{locations}$
    » Tells where $r1$ is at time $t$

● Might not ever specify the entire function

● $\text{cpos}(c)$ refers to a collection of state variables
  ◆ We’ll be sloppy and call it a state variable rather than a collection of state variables
Example

- DWR domain, with
  - robot \( r_1 \)
  - container \( c_1 \)
  - ship \( \text{Uranus} \)
  - locations \( \text{loc1}, \text{loc2} \)
  - cranes \( \text{crane2}, \text{crane4} \)
- \( r_1 \) is in \( \text{loc1} \) at time \( t_1 \), leaves \( \text{loc1} \) at time \( t_2 \), enters \( \text{loc2} \) at time \( t_3 \), leaves \( \text{loc2} \) at time \( t_4 \), enters \( l \) at time \( t_5 \)
- \( c_1 \) is in \( \text{pile1} \) until time \( t_6 \), held by \( \text{crane2} \) from \( t_6 \) to \( t_7 \), sits on \( r_1 \) until \( t_8 \), held by \( \text{crane4} \) until \( t_9 \), and sits on \( p \) until \( t_{10} \) or later
- \( \text{Uranus} \) stays at \( \text{dock5} \) from \( t_{11} \) to \( t_{12} \)
Temporal Assertions

- Temporal assertion:
  - **Event**: an expression of the form $x@t : (v_1,v_2)$
    - At time $t$, $x$ changes from $v_1$ to $v_2 \neq v_1$
  - **Persistence condition**: $x@[t_1,t_2) : v$
    - $x = v$ throughout the interval $[t_1,t_2)$
  - where
    - $t$, $t_1$, $t_2$ are constants or temporal variables
    - $v$, $v_1$, $v_2$ are constants or object variables

- Note that the time intervals are semi-open
  - **Why are they?**
Temporal Assertions

Temporal assertion:

- **Event**: an expression of the form $x@t : (v_1,v_2)$
  - At time $t$, $x$ changes from $v_1$ to $v_2 \neq v_1$
- **Persistence condition**: $x@[t_1,t_2) : v$
  - $x = v$ throughout the interval $[t_1,t_2)$

where
- $t, t_1, t_2$ are constants or temporal variables
- $v, v_1, v_2$ are constants or object variables

Note that the time intervals are semi-open
- Why are they?
- To prevent potential confusion about $x$’s value at the endpoints
Chronicles

- **Chronicle**: a pair $\Phi = (F,C)$
  - $F$ is a finite set of temporal assertions
  - $C$ is a finite set of constraints
    - temporal constraints and object constraints
  - $C$ must be consistent (i.e., there must exist variable assignments that satisfy it)

- **Timeline**: a chronicle for a single state variable

The book writes $F$ and $C$ in a calligraphic font
  - Sometimes I will, more often I’ll just use italics
Example

Timeline for rloc(r1):

```
{  rloc(r1)@t_1 : (l_1, loc1),
   rloc(r1)@[t_1, t_2) : loc1,
   rloc(r1)@t_2 : (loc1, l_2),
   rloc(r1)@t_3 : (l_3, loc2),
   rloc(r1)@[t_3, t_4) : loc2,
   rloc(r1)@t_4 : (loc2, l_4),
   rloc(r1)@t_5 : (l_5, loc3)  },
{  adjacent(l_1, loc1), adjacent(loc1, l_2),
   adjacent(l_3, loc2), adjacent(loc2, l_4), adjacent(l_5, loc3),
   t_1 < t_2 < t_3 < t_4 < t_5  }).
```
Consistency

- A timeline \((F,C)\) is consistent if
  - \(C\) is consistent
  - Every pair of assertions in \(F\) are either disjoint or they refer to the same value and/or time points:
    - If \(F\) contains both \(x@[t_1,t_2]:v_1\) and \(x@[t_3,t_4]:v_2\), then \(C\) must entail \(\{t_2 \leq t_3\}, \{t_4 \leq t_1\}\), or \(\{v_1 = v_2\}\)
    - If \(F\) contains both \(x@t: (v_1,v_2)\) and \(x@[t_1,t_2]:v\) then \(C\) must entail \(\{t < t_1\}, \{t_2 < t\}, \{v = v_2, t_1 = t\}\), or \(\{t_2 = t, v = v_1\}\)
    - If \(F\) contains both \(x@t: (v_1,v_2)\) and \(x@t': (v'_1,v'_2)\) then \(C\) must entail \(\{t \neq t'\}\) or \(\{v_1 = v'_1, v_2 = v'_2\}\)

- \((F,C)\) is consistent iff the timelines for all of its state variables are consistent

- This is stronger than the usual notion of a consistent set of constraints
  - Here, the separation constraints must actually be entailed by \(C\)
  - It’s sort of like saying that \((F,C)\) contains no threats
Let \((F, C)\) include the timelines given earlier, plus some additional constraints:

- \(t_1 \leq t_6, \ t_7 < t_2, \ t_3 \leq t_8, \ t_9 < t_4, \ \text{attached}(p, \text{loc2})\)

Above, I’ve drawn the entire set of time constraints

All pairs of temporal assertions are either disjoint or refer to the same value at the same point, so \((F, C)\) is consistent
Support and Enablers

- Let $\alpha$ be the assertion $x@t:(v,v')$ or the assertion $x@[t,t'):v$

- Intuitively, a chronicle $\Phi = (F,C)$ supports $\alpha$ when
  - $\Phi$ contains an assertion $\beta$ that we can use to establish $x = v$ at some time $s < t$,
    - $\beta$ is called the support for $\alpha$
  - it is consistent to have $v$ to persist over $[s,t)$,
  - it is consistent to have $\alpha$ be true

- Formally, $\Phi = (F,C)$ supports $\alpha$ if there is an assertion $\beta$ in $F$ of the form
  $\beta = x@s:(w',w)$ or $\beta = x@[s',s)@w$
  and a set of separation constraints $C'$ that makes the following chronicle consistent:
    - $(F \cup \{x@[s,t):v\}, C \cup C' \cup \{w=v, s < t\})$

- The pair $\delta = (\{x@[s,t):v\}, C' \cup \{w=v, s < t\})$ is an enabler for $\alpha$
  - Analogous to a causal link in PSP
  - Can either be absent from $\Phi$ or already in $\Phi$

- Just like there could be more than one possible causal link in PSP, there can be more than one possible enabler
Example

- We can support $\alpha = \text{rloc}(r1)@t: (\text{routes, loc3})$ in two ways:
  - $\beta_1 = \text{rloc}(r1)@t_2: (\text{loc1, routes})$
  - $\beta_2 = \text{rloc}(r1)@t_4: (\text{loc2, routes})$

- An enabler for $\beta_2$ is
  - $\delta = (\{\text{rloc}(r1)@[t_4, t]: \text{routes, rloc}(r1)@t: (\text{routes, loc3})\}, \{t_4 < t < t_5\}$
● $\Phi = (F,C)$ supports a set of assertions $E = \{\alpha_1, \ldots, \alpha_k\}$ if there is a set of enablers $\phi = \{\delta_1, \ldots, \delta_k\}$ having the following properties:
  ◆ the chronicle is $(F,C) \cup \delta_1 \cup \ldots \cup \delta_k$ is consistent
  ◆ each $\alpha_i$ is supported by everything other than $\alpha_i$
    » i.e., $\alpha_i$ is supported by $(F \cup E - \{\alpha_i\}, C)$

● Note that some of the assertions in $E$ may support each other!

● $\phi = \{\delta_1, \ldots, \delta_k\}$ is an enabler for $E$
Four ways to support the pair of assertions
\[ \alpha_1 = rloc(r1)@t:(routes, loc3) \]
\[ \alpha_2 = rloc(r1)@[t',t'']:loc3 \]

To support \( \alpha_1 \), use either \( \beta_1 = rloc(r1)@t_2:(loc1, routes) \)
\[ \text{or } \beta_2 = rloc(r1)@t_4:(loc2, routes) \]

To support \( \alpha_2 \), use either \( \alpha_1 \), or \( \beta = rloc(r1)@t_5:(routes,l) \) with the binding \( l=loc3 \)
Enabling and Supporting Another Chronicle

Let $\Phi' = (\mathcal{F}', \mathcal{C}')$ be a chronicle such that $\Phi$ supports $\mathcal{F}'$, and let $\theta(\Phi'/\Phi)$ be the set of all possible enablers of $\mathcal{F}'$ in $\Phi$ augmented with the constraints $\mathcal{C}'$ of $\Phi'$, i.e.,

$$\theta(\Phi'/\Phi) = \{\phi \cup (\emptyset, \mathcal{C}') \mid \phi \text{ is an enabler of } \mathcal{F}'\}.$$ 

$\theta(\Phi'/\Phi)$ is empty when $\Phi$ does not support $\mathcal{F}'$. Note that every element of $\theta(\Phi'/\Phi)$ is a chronicle that contains $\Phi'$; it is also called an *enabler* for $\Phi'$.

**Definition 14.11** A consistent chronicle $\Phi = (\mathcal{F}, \mathcal{C})$ supports a chronicle $\Phi' = (\mathcal{F}', \mathcal{C}')$ iff $\Phi$ supports $\mathcal{F}'$ and there is an enabler $\phi \in \theta(\Phi'/\Phi)$ such that $\Phi \cup \phi$ is a consistent chronicle. Chronicle $\Phi$ *entails* $\Phi'$ iff it supports $\Phi'$ and there is an enabler $\phi \in \theta(\Phi'/\Phi)$ such that $\phi \subseteq \Phi$. 

Chronicles as Planning Operators

● Chronicle planning operator: a pair $o = (\text{name}(o), (F(o), C(o)))$ such that
  ◆ name($o$) is an expression of the form $o(t_s, t_e, \ldots, v_1, v_2, \ldots)$
    » $o$ is an operator symbol
    » $t_s, t_e, \ldots, v_1, v_2, \ldots$ are all the temporal and object variables in $o$
  ◆ $(F(o), C(o))$ is a chronicle

● Action: a (partially) instantiated operator $a$

● If $\Phi$ supports $(F(a), C(a))$, then $a$ is applicable to $\Phi$
  ◆ $a$ may be applicable in several ways, so the result is a set of chronicles
  ◆ $\gamma(\Phi, a) = \{\Phi \cup \phi \mid \phi \in \theta(a/\Phi)\}$
Example: moving a robot

\[
\text{move}(t_s, t_e, t_1, t_2, r, l, l') = \\
\{ rloc(r)@t_s : (l, \text{routes}), \\
rloc(r)@[t_s, t_e) : \text{routes}, \\
rloc(r)@t_e : (\text{routes}, l'), \\
contains(l)@t_1 : (r, \text{empty}), \\
contains(l')@t_2 : (\text{empty}, r), \\
t_s < t_1 < t_2 < t_e, \\
adjacent(l, l') \}
\]
Applying a Set of Actions

- If we want to execute a set of actions, they may support each other!
  - Couldn’t happen in classical planning, because the actions that support $a_i$ must finish before $a_i$ starts
  - In temporal planning, the actions can overlap

- Let $\pi = \{a_1, \ldots, a_k\}$ be the set of actions
- Let $\Phi_\pi = \bigcup_i (F(a_i), C(a_i))$
- If $\Phi$ supports $\Phi_\pi$
  - then $\pi$ is applicable to $\Phi$
- Result is a set of chronicles
  $\gamma(\Phi, \pi) = \{\Phi \cup \phi \mid \phi \in \theta(\Phi_\pi / \Phi)\}$

Dana Nau: Lecture slides for Automated Planning
Licensed under the Creative Commons Attribution-NonCommercial-ShareAlike License: http://creativecommons.org/licenses/by-nc-sa/2.0/
Domains and Problems

- Temporal planning domain: a pair $D = (\Lambda_\Phi, O)$
  - $O = \{\text{all chronicle planning operators in the domain}\}$
  - $\Lambda_\Phi = \{\text{all chronicles allowed in the domain}\}$

- Temporal planning problem on $D$: a triple $P = (D, \Phi_0, \Phi_g)$
  - $D$ is the domain
  - $\Phi_0$ and $\Phi_g$ are initial chronicle and goal chronicle
  - $O$ is the set of chronicle planning operators

- Statement of the problem $P$: a triple $P = (O, \Phi_0, \Phi_g)$
  - $O$ is the set of chronicle planning operators
  - $\Phi_0$ and $\Phi_g$ are initial chronicle and goal chronicle

- Solution plan: a set of actions $\pi = \{a_1, \ldots, a_n\}$ such that at least one chronicle in $\gamma(\Phi_0, \pi)$ entails $\Phi_g$
As in plan-space planning, there are two kinds of flaws:

- Open goal: a tqe that isn’t yet enabled
- Threat: an enabler that hasn’t yet been incorporated into $\Phi$

$$\text{CP}(\Phi, G, \mathcal{K}, \pi)$$

if $G = \mathcal{K} = \emptyset$ then return($\pi$)
perform the two following steps in any order
if $G \neq \emptyset$ then do
  select any $\alpha \in G$
  if $\theta(\alpha/\Phi) \neq \emptyset$ then return($\text{CP}(\Phi, G - \{\alpha\}, \mathcal{K} \cup \{\theta(\alpha/\Phi)\}, \pi)$)
else do
  $relevant \leftarrow \{a \mid a \text{ contains a support for } \alpha\}$
  if $relevant = \emptyset$ then return(failure)
  nondeterministically choose $a \in relevant$
  return($\text{CP}(\Phi \cup (F(a), C(a)), G \cup F(a), \mathcal{K} \cup \{\theta(a/\Phi)\}, \pi \cup \{a\})$)
if $\mathcal{K} \neq \emptyset$ then do
  select any $C \in \mathcal{K}$
  threat-resolvers $\leftarrow \{\phi \in C \mid \phi \text{ consistent with } \Phi\}$
  if $threat\text{-resolvers} = \emptyset$ then return(failure)
  nondeterministically choose $\phi \in threat\text{-resolvers}$
  return($\text{CP}(\Phi \cup \phi, G, \mathcal{K} - C, \pi)$)
end
Resolving Open Goals

- Let $\alpha$ be an open goal

- Case 1: $\Phi$ supports $\alpha$
  - Resolver: any enabler for $\alpha$ that’s consistent with $\Phi$
  - Refinement:
    - $G \leftarrow G - \{\alpha\}$
    - $K \leftarrow K \cup \theta(\alpha/\Phi)$

- Case 2: $\Phi$ doesn’t support $\alpha$
  - Resolver: an action $a = (F(a), C(a))$ that supports $\alpha$
    - We don’t yet require $a$ to be supported by $\Phi$
  - Refinement:
    - $\pi \leftarrow \pi \cup \{a\}$
    - $\Phi \leftarrow \Phi \cup (F(a), C(a))$
    - $G \leftarrow G \cup F(a)$ put all of $a$’s tges into $G$
    - $K \leftarrow K \cup \theta(a/\Phi)$ put $a$’s set of enablers into $K$
Resolving Threats

- **Threat**: each enabler in $K$ that isn’t yet entailed by $\Phi$ is threatened
  - For each $C$ in $K$, we need only one of the enablers in $C$
    - They’re alternative ways to achieve the same thing
  - “Threat” means something different here than in PSP, because we won’t try to entail *all* of the enablers
    - Just the one we select
- Resolver: any enabler $\phi$ in $C$ that is consistent with $\Phi$
- Refinement:
  - $K \leftarrow K - C$
  - $\Phi \leftarrow \Phi \cup \phi$
Example

- $\Phi_0$ is as shown:

- $\Phi_g = \Phi_0 \cup (E, \{\})$, where
  - $E = \{\alpha_1 = rloc(r1)@t:(routes, loc3) \\
    \alpha_2 = rloc(r1)@[t',t'']:loc3)\}$

- As before, let $\beta_1 = rloc(r1)@t_2:(loc1, routes)$
  - $\beta_2 = rloc(r1)@t_4:(loc2, routes)$

- Also, let $\beta = rloc(r1)@t_5:(routes, l)$