Chapter 14
Temporal Planning

Dana S. Nau
University of Maryland

3:11 PM  April 18, 2012
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 $t$
  - $\text{cpos}(c1) : \text{time} \rightarrow \text{containers U cranes U robots}$
    - 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
  - But we’ll be sloppy and just call it a state variable
DWR Example

- **robot r1**
  - in loc1 at time $t_1$
  - leaves loc1 at time $t_2$
  - enters loc2 at time $t_3$
  - leaves loc2 at time $t_4$
  - enters $l$ at time $t_5$

- **container c1**
  - in pile1 until time $t_6$
  - held by crane2 until $t_7$
  - sits on r1 until $t_8$
  - held by crane4 until $t_9$
  - sits on $p$ until $t_{10}$ (or later)

- **ship Uranus**
  - stays at 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?
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?
  - 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 $rloc(r1)$:

\[
\{ \begin{array}{l}
  rloc(r1)@t_1 : (l_1, loc1), \\
  rloc(r1)@[t_1, t_2) : loc1, \\
  rloc(r1)@t_2 : (loc1, l_2), \\
  rloc(r1)@t_3 : (l_3, loc2), \\
  rloc(r1)@[t_3, t_4) : loc2, \\
  rloc(r1)@t_4 : (loc2, l_4), \\
  rloc(r1)@t_5 : (l_5, loc3) \\
\end{array} \}
\]

\[
\{ \begin{array}{l}
  adjacent(l_1, loc1), adjacent(loc1, l_2), \\
  adjacent(l_3, loc2), adjacent(loc2, l_4), adjacent(l_5, loc3), \\
  t_1 < t_2 < t_3 < t_4 < t_5 \\
\end{array} \}
\]

Inconsistency in the book between Figure 14.5 and Example 14.9
C-consistency

A timeline \((F,C)\) is c-consistent (chronicle-consistent) if

- \(C\) is consistent, and
- 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 c-consistent iff every timeline in \((F,C)\) is c-consistent
- The book calls this consistency, not c-consistency
  - But it’s a stronger requirement than ordinary mathematical consistency
- Mathematical consistency: \(C\) doesn’t contradict the separation constraints
- c-consistency: \(C\) must actually entail the separation constraints
  - It’s sort of like saying that \((F,C)\) contains no threats
Example

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

\((F, C)\) is c-consistent
Support and Enablers

- Let $\alpha$ be either $x@t: (v,v')$ or $x@[t,t'):v$
  - Note that $\alpha$ requires $x = v$ either at $t$ or just before $t$
- Intuitively, a chronicle $\Phi = (F,C)$ supports $\alpha$ if
  - $F$ contains an assertion $\beta$ that we can use to establish $x = v$ at some time $s < t$,
    - $\beta$ is called the support for $\alpha$
  - and if it’s consistent with $\Phi$ for $v$ to persist over $[s,t)$ and for $\alpha$ to be true
- Formally, $\Phi = (F,C)$ supports $\alpha$ if
  - $F$ contains an assertion $\beta$ of the form $\beta = x@s: (w',w)$ or $\beta = x@[s',s):w$, and
  - $\exists$ separation constraints $C'$ such that the following chronicle is c-consistent:
    - $(F \cup \{ x@[s,t):v, \alpha \}, C \cup C' \cup \{ w=v, s < t \})$
  - $C'$ can either be absent from $\Phi$ or already in $\Phi$
- The chronicle $\delta = (\{ x@[s,t):v, \alpha \}, C' \cup \{ w=v, s < t \})$ is an enabler for $\alpha$
  - Analogous to a causal link in PSP
- Just as there could be more than one possible causal link in PSP, there can be more than one possible enabler
Example

\[ \beta_1 = \text{rloc}(r1) @ t_2 : (\text{loc1}, \text{routes}) \]

\[ \beta_2 = \text{rloc}(r1) @ t_4 : (\text{loc2}, \text{routes}) \]

- Let \( \Phi \) be as shown
- Then \( \Phi \) supports
  \[ \alpha_1 = \text{rloc}(r1) @ t : (\text{routes}, \text{loc3}) \]
  in two different ways:
    - \( \beta_1 \) establishes \( \text{rloc}(r1) = \text{routes} \) at time \( t_2 \)
      - this can support \( \alpha_1 \) if we constrain \( t_2 < t < t_3 \)
      - enabler is \( \delta_1 = \{ \text{rloc}(r1) @ [t_2,t) : \text{routes}, \alpha_1 \}, \{t_2 < t < t_3\} \)
    - \( \beta_2 \) establishes \( \text{rloc}(r1) = \text{routes} \) at time \( t_4 \)
      - this can support \( \alpha_1 \) if we constrain \( t_4 < t < t_5 \)
      - enabler is \( \delta_2 = \{ \text{rloc}(r1) @ [t_4,t) : \text{routes}, \alpha_1 \}, \{t_4 < t < t_5\} \)
Enabling Several Assertions at Once

- $\Phi = (F,C)$ supports a set of assertions $E = \{\alpha_1, \ldots, \alpha_k\}$ if both of the following are true:
  - $F \cup E$ contains a support $\beta_i$ for $\alpha_i$ other than $\alpha_i$ itself
  - There are enablers $\delta_1, \ldots, \delta_k$ for $\alpha_1, \ldots, \alpha_k$ such that the chronicle $\Phi \cup \delta_1 \cup \ldots \cup \delta_k$ is c-consistent

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

- $\phi = \{\delta_1, \ldots, \delta_k\}$ is an enabler for $E$
Example

1. Let $\Phi$ be as shown.
2. Let $\alpha_1$ be the same as before: $\alpha_1 = rloc(r1)@t:(\text{routes, loc3})$.
3. Let $\alpha_2 = rloc(r1)@[t',t'']:\text{loc3}$.

Then $\Phi$ supports $\{\alpha_1, \alpha_2\}$ in four different ways:

- As before, for $\alpha_1$ we can use either $\beta_1$ and $\delta_1$ or $\beta_2$ and $\delta_2$.
- We can support $\alpha_2$ with $\beta_3 = rloc(r1)@t_5:(\text{routes}, l)$.
  - Enabler is $\delta_3 = (\{rloc(r1)@[t_5,t'):\text{loc3}, \alpha_2\}, \{l = \text{loc3, } t_5 < t'\})$.
- Or we can support $\alpha_2$ with $\alpha_1$.
  - If we supported $\alpha_1$ with $\beta_1$ and enabled it with $\delta_1$, the enabler for $\alpha_2$ is $\delta_4 = (\{rloc(r1)@[t,t'):\text{loc3}, \alpha_2\}, \{t < t' < t_3\})$.
  - If we supported $\alpha_1$ with $\beta_1$ and enabled it with $\delta_2$, then replace $t_3$ with $t_5$ in $\delta_4$. 

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/
One Chronicle Supporting Another

- Let $\Phi' = (F', C')$ be a chronicle, and suppose $\Phi = (F, C)$ supports $F'$.
- Let $\delta_1, \ldots, \delta_k$ be all the possible enablers of $\Phi'$
  - For each $\delta_i$, let $\delta'_i = \delta_1 \cup C'$
- If there is a $\delta'_i$ such that $\Phi \cup \delta'_i$ is c-consistent,
  - Then $\Phi$ supports $\Phi'$, and $\delta'_i$ is an enabler for $\Phi'$
  - If $\delta'_i \subseteq \Phi$, then $\Phi$ entails $\Phi'$
- The set of all enablers for $\Phi'$ is $\theta(\Phi/\Phi') = \{\delta'_i : \Phi \cup \delta'_i \text{ is c-consistent}\}$
Chronicles as Planning Operators

- Chronicle planning operator: a pair \( o = (\text{name}(o), (F(o), C(o))) \), where
  - \( \text{name}(o) \) 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 a chronicle \( \Phi \) 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: Operator for Moving a Robot

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

- Just like several temporal assertions can support each other, several actions can also support each other
  - Let $\pi = \{a_1, \ldots, a_k\}$ be a 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
    $\gamma(\Phi,\pi) = \{\Phi \cup \phi \mid \phi \in \theta(\Phi_\pi/\Phi)\}$

- Example:
  - Suppose $\Phi$ asserts that at time $t_0$, robots $r_1$ and $r_2$ are at adjacent locations $\text{loc}_1$ and $\text{loc}_2$
  - Let $a_1$ and $a_2$ be as shown
  - Then $\Phi$ supports $\{a_1, a_2\}$ with
    $l_1 = \text{loc}_1$, $l_2 = \text{loc}_2$, $l'_1 = \text{loc}_2$, $l'_2 = \text{loc}_1$,
    $t_0 < t_s < t_1 < t'_2$, $t_0 < t'_s < t'_1 < t_2$
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 \( \Phi \)

\[
\mathcal{CP}(\Phi, G, \mathcal{K}, \pi)
\]

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(\( \mathcal{CP}(\Phi, G - \{\alpha\}, \mathcal{K} \cup \{\theta(\alpha/\Phi)\}, \pi) \))

else do

\[
\text{relevant} \leftarrow \{a \mid a \text{ contains a support for } \alpha\}
\]

if \( \text{relevant} = \emptyset \) then return(failure)

nondeterministically choose \( a \in \text{relevant} \)

return(\( \mathcal{CP}(\Phi \cup (\mathcal{F}(a), \mathcal{C}(a)), G \cup \mathcal{F}(a), \mathcal{K} \cup \{\theta(a/\Phi)\}, \pi \cup \{a\}) \))

if \( \mathcal{K} \neq \emptyset \) then do

select any \( C \in \mathcal{K} \)

\[
\text{threat-resolvers} \leftarrow \{\phi \in C \mid \phi \text{ consistent with } \Phi\}
\]

if \( \text{threat-resolvers} = \emptyset \) then return(failure)

nondeterministically choose \( \phi \in \text{threat-resolvers} \)

return(\( \mathcal{CP}(\Phi \cup \phi, G, \mathcal{K} - C, \pi) \))
Resolving Open Goals

- Let $\alpha \in G$ 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:
    - $\pi \leftarrow \pi \cup \{a\}$
    - $\Phi \leftarrow \Phi \cup (F(a), C(a))$
    - $G \leftarrow G \cup F(a)$ Don’t remove $\alpha$ yet: we haven’t chosen an enabler for it
      - We’ll choose one in a later call to CP, in Case 1 above
    - $K \leftarrow K \cup \theta(a/\Phi)$ 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:
    - $K \leftarrow K - C$
    - $\Phi \leftarrow \Phi \cup \phi$
Let $\Phi_0$ be as shown, and $\Phi_g = \Phi_0 \cup \{\alpha_1, \alpha_2\}, \{\} \}$, where $\alpha_1$ and $\alpha_2$ are the same as before:

- $\alpha_1 = rloc(r1)@t: (\text{routes, loc3})$
- $\alpha_2 = rloc(r1)@[t', t'']:\text{loc3}$

As we saw earlier, we can support $\{\alpha_1, \alpha_2\}$ from $\Phi_0$

- Thus CP won’t add any actions
- It will return a modified version of $\Phi_0$ that includes the enablers for $\{\alpha_1, \alpha_2\}$
Modified Example

Let $\Phi_0$ be as shown, and $\Phi_g = \Phi_0 \cup \{\{\alpha_1,\alpha_2\},\{\}\}$, where $\alpha_1$ and $\alpha_2$ are the same as before:

- $\alpha_1 = \text{rloc}(r1)@t:\text{(routes, loc3)}$
- $\alpha_2 = \text{rloc}(r1)@[t',t'']:\text{loc3}$

This time, CP will need to insert an action $\text{move}(t_s, t_e, t_1, t_2, r1, \text{loc4, loc3})$

with $t_5 < t_s < t_1 < t_2 < t_e$