Temporal Planning

Planning with Temporal and Concurrent Actions

Literature

- Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated Planning – Theory and Practice*, chapter 13-14. Elsevier/Morgan Kaufmann, 2004.

Why Explicit Time?

- assumption A6: implicit time
  - actions and events have no duration
  - state transitions are instantaneous
- in reality:
  - actions and events do occur over a time span
  - preconditions not only at beginning
  - effects during or even after the action
  - actions may need to maintain partial states
  - events expected to occur in future time periods
  - goals must be achieved within time bound

Overview

- Actions and Time Points
- Interval Algebra and Quantitative Time
- Planning with Temporal Operators

Time

- mathematical structure:
  - set with transitive, asymmetric ordering operation
  - discrete, dense, or continuous
  - bounded or unbounded
  - totally ordered or branching
- temporal references:
  - time points (represented by real numbers)
  - time intervals (pair of real numbers)
- temporal relations:
  - examples: before, during

Causal vs. Temporal Analysis of Actions

- example: load(crane2, cont5, robot1, interval6)

- causal analysis (what propositions hold?):
  - what propositions will change (effects)
  - what propositions are required (preconditions)
- temporal analysis (when propositions hold?):
  - when other, related assertions can/cannot be true
  - reason over:
    - time periods during which propositions must hold
    - time points at which values of state variables change

Temporal Databases

- maintain temporal references for every domain proposition
  - when does it hold
  - when does it change value
- functionality:
  - assert new temporal relations
  - querying whether temporal relation holds
  - check for consistency
- planner attempts to assert relations among temporal references

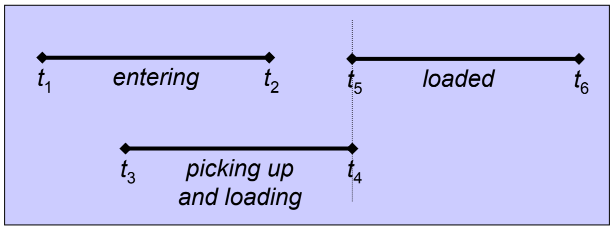
Temporal References Example: Container Loading

- load container c onto robot r at location l

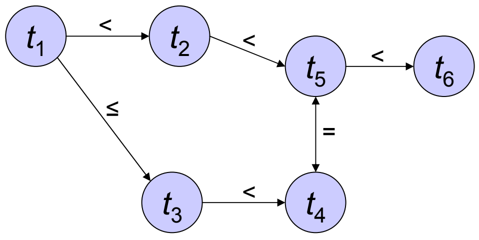
- t_1: instant at which robot r enters location l
- t_2: instant at which robot r stops at location l
  - $i_1=[t_1,t_2]$: interval corresponding to r *entering* l
- t_3: instant at which the crane starts picking up c
- t_4: instant at which crane finishes putting c on r
  - $i_2=[t_3,t_4]$: interval corresponding to *picking up and loading* c
- t_5: instant at which c begins to be loaded onto r
- t_6: instant at which c is no longer loaded onto r
  - $i_3=[t_5,t_6]$: interval corresponding to c *being loaded* onto r

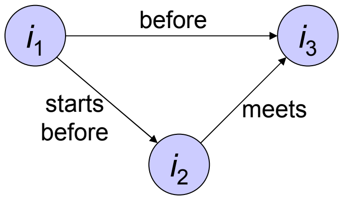
Temporal Relations Example: Container Loading

- assumption: crane is allowed to pick up container as soon as robot has entered location
- possible temporal sequences:
  - $t_1 < t_3 < t_2 < t_4 = t_5 < t_6$ (see figure) or
  - $t_1 = t_3$ or $t_2 = t_3$ or $t_2 < t_3$



Example: Temporal Relations as Constraint Networks

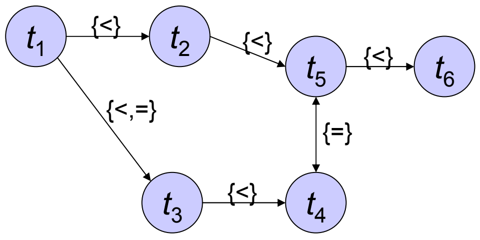

$$\begin{tikzcd}
t_1 \arrow[r, "<"] \arrow[dr, "\leq"'] & t_2 \arrow[r, "<"] & t_5 \arrow[r, "<"] \arrow[d, leftrightarrow, "="] & t_6 \\
& t_3 \arrow[r, "<"] & t_4 &
\end{tikzcd}$$


$$\begin{tikzcd}
i_1 \arrow[rr, "\text{before}"] \arrow[dr, "\text{starts before}"'] & & i_3 \\
& i_2 \arrow[ur, "\text{meets}"'] &
\end{tikzcd}$$

Point Algebra (PA): Relations and Constraints

- possible primitive relations P between instants t_1 and t_2: $P = \{<,=,>\}$
  - t_1 before t_2: $[t_1<t_2]$
  - t_1 equal to t_2: $[t_1=t_2]$
  - t_1 after t_2: $[t_1>t_2]$
- possible qualitative constraints R between instants:
  - sets of the above relations (interpret as disjunction)
  - $R = 2^P = \{\varnothing, \{<\}, \{=\}, \{>\}, \{<,=\}, \{<,>\}, \{=,>\}, P\}$

Container Loading Example: PA Constraints



- $[t_1 \{<\} t_2]$
- $[t_1 \{<,=\} t_3]$
- $[t_2 \{<\} t_5]$
- $[t_3 \{<\} t_4]$
- $[t_4 \{=\} t_5]$
- $[t_5 \{=\} t_4]$
- $[t_5 \{<\} t_6]$

PA: Combining Constraints

- usual set operations:
  - ∩, ∪ etc.
- composition (noted •):
  - let r, $q \in R$
  - if $[t_1\ r\ t_2]$ and $[t_2\ q\ t_3]$
  - then $[t_1\ r{\bullet}q\ t_3]$
  - $r{\bullet}q$ as defined in composition table

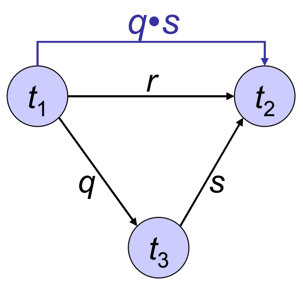
•	<	=	>
<	<	<	*P*
=	<	=	>
>	*P*	>	>

PA composition table

PA: Properties of Combined Constraints

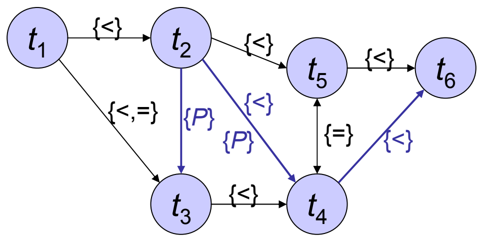
- distributive
  - $(r \cup q) \bullet s = (r \bullet s) \cup (q \bullet s)$
  - $s \bullet (r \cup q) = (s \bullet r) \cup (s \bullet q)$
- symmetrical constraint r' of r:
  - $[t_1\ r\ t_2]$ iff $[t_2\ r'\ t_1]$
  - obtained by replacing in r: < with > and vice versa
  - $(r \bullet q)' = q' \bullet r'$
- $(R,\cup,\bullet)$ is an algebra:
  - R is closed under ∪ and •
  - ∪ is an associative and commutative operation
  - identity element for ∪ is ∅
  - is an associative operation
  - identity element for • is {=}

PA: Constraint Propagation



- given constraints:
  - $[t_1\ r\ t_2]$
  - $[t_1\ q\ t_3]$
  - $[t_3\ s\ t_2]$
- implied constraint:
  - $[t_1\ r \cap q \bullet s\ t_2]$
- inconsistency:
  - if $r \cap q \bullet s = \varnothing$

Container Loading Example: Constraint Propagation



- path: t_4-t_5-t_6: $[t_4{=}t_5] \bullet [t_5{<}t_6]$ implies $[t_4{<}t_6]$
- path: t_2-t_1-t_3: $[t_2{>}t_1] \bullet [t_1{\le}t_6]$ implies $[t_2 P t_3]$
- path: t_2-t_5-t_4: $[t_2{<}t_5] \bullet [t_5{=}t_4]$ implies $[t_2{<}t_4]$
- path: t_2-t_3-t_4: $[t_2 P t_3] \bullet [t_3{<}t_4]$ implies $[t_2 P t_4]$

$\}$ $[t_2{<}t_4]$

PA Constraint Networks

- A <u>binary PA constraint network</u> is a directed graph (X,C), where:
  - $X = \{t_1,\ldots,t_n\}$ is a set of instant variables (nodes), and
  - $C \subseteq X \times X$ (the edges), c_{ij} is labelled by a constraint $r_{ij} \in R$ iff $[t_i\ r_{ij}\ t_j]$ holds.
- A tuple $\langle v_1,\ldots,v_k \rangle$ of real numbers is a <u>solution for (X,C)</u> iff $t_i = v_i$ satisfy all the constraints in C.
- (X,C) is <u>consistent</u> iff there exists at least one solution.

Primitives in Consistent Networks

- **Proposition**: A PA network (X,C) is consistent iff
  - there is a set of primitives $p_{ij} \in r_{ij}$ for every $c_{ij} \in C$ such that
    - for every k: $p_{ij} \in (p_{ik} \bullet p_{kj})$
- note: not interested in solution, just consistency (qualitative solution)

Redundant Networks

- A primitive $p_{ij} \in r_{ij}$ is <u>redundant</u> if there is no solution in which $[t_i \; p_{ij} \; t_j]$ holds.

- idea: filter out redundant primitives until
  - either: no more redundant primitives can be found
  - or: we find a constraint that is reduced to $\varnothing$ (inconsistency)

Path Consistency: Pseudo Code

```
pathConsistency($C$)
  **while** $\neg C$.isStable() **do**
    **for** each $k$ : $1 \leq k \leq n$ **do**
      **for** each pair $i,j$: $1 \leq i < j \leq n, i \neq k, j \neq k$ **do**
        $c_{ij} \leftarrow c_{ij} \cap [c_{ik} \bullet c_{kj}]$
        **if** $c_{ij} = \varnothing$ **then return** inconsistent
```

Path Consistency: Properties

- algorithm pathConsistency(*C*) is:
  - incomplete for general CSPs
  - complete for PA networks
- network (*X*,*C*) is minimal if it has no redundant primitives in a constraint
- algorithm pathConsistency(*C*) does not guarantee a minimal network

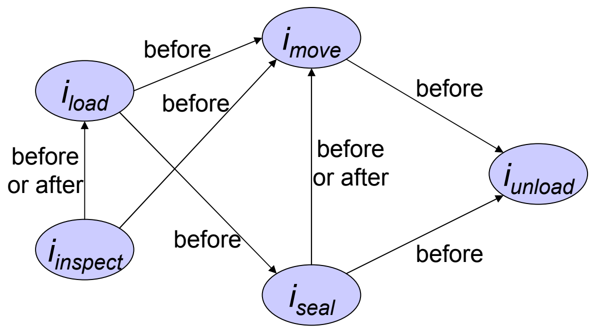
Overview

- Actions and Time Points
- Interval Algebra and Quantitative Time
- Planning with Temporal Operators

Extended Example: Inspect and Seal

- every container must be inspected and sealed:
- inspection:
  - carried out by the crane
  - must be performed *before or after* loading
- sealing:
  - carried out by robot
  - *before or after* unloading, not while moving
- corresponding intervals:
  - i_{load}, i_{move}, i_{unload}, $i_{inspect}$, i_{seal}

Inspect and Seal Example: Interval Constraint Network



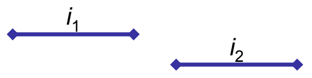
Inspect and Seal Example: Qualitative Instant Constraints

- Let i be an interval.
  - $i.b$ and $i.e$ denote two end time points
  - [$i.b \leq i.e$] constraint: beginning before end
- [i_{load} before i_{move}]:
  - [$i_{load}.b \leq i_{load}.e$] and [$i_{move}.b \leq i_{move}.e$] and
  - [$i_{load}.e < i_{move}.b$]
- [i_{move} before-or-after i_{seal}]:
  - [$i_{move}.e < i_{seal}.b$] or
  - [$i_{seal}.e < i_{move}.b$]

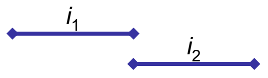
disjunction cannot be translated into binary PA constraint

Interval Algebra (IA): Relations

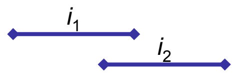
- i_1 before i_2: [i_1 b i_2]



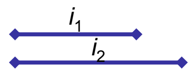
- i_1 meets i_2: [i_1 m i_2]



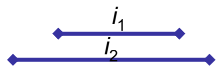
- i_1 overlaps i_2: [i_1 o i_2]



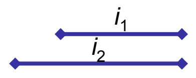
- i_1 starts i_2: [i_1 s i_2]



- i_1 during i_2: [i_1 d i_2]

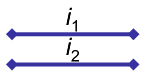


- i_1 finishes i_2: [i_1 f i_2]



IA: Relations and Constraints

- possible <u>primitive relations *P*</u> between intervals i_1 and i_2:
  - just described: $[i_1\ b\ i_2]$, $[i_1\ m\ i_2]$, $[i_1\ o\ i_2]$, $[i_1\ s\ i_2]$, $[i_1\ d\ i_2]$, $[i_1\ f\ i_2]$
  - symmetrical: $[i_1\ b'\ i_2]$, $[i_1\ m'\ i_2]$, $[i_1\ o'\ i_2]$, $[i_1\ s'\ i_2]$, $[i_1\ d'\ i_2]$, $[i_1\ f'\ i_2]$
  - i_1 equals i_2: $[i_1\ e\ i_2]$



- possible <u>qualitative constraints *R*</u> between instants:
  - sets of the above relations (interpret as disjunction)
  - $R = 2^P = \{\varnothing, \{b\}, \{m\}, \{o\},\ldots, \{b,m\}, \{b,o\},\ldots, \{b,m,o\},\ldots, P\}$
  - examples: while = $\{s,d,f\}$; disjoint = $\{b,b'\}$

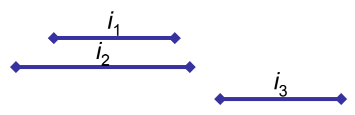
Operations on Relations

- set operations: ∩, ∪ etc.
- composition: •

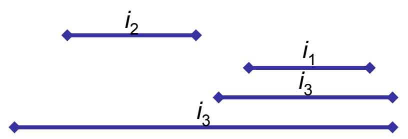
•	*b*	*m*	*o*	*s*	*d*	*f*	*b'*	*d'*	*f'*
b	*b*	*b*	*b*	*b*	*u*∪*v*	*u*∪*v*	*P*	*b*	*b*
m	*b*	*b*	*b*	*m*	*v*	*v*	*u'*∪*v'*	*b*	*b*
o	*b*	*b*	*u*	*o*	*v*	*v*	*u'*∪*v'*	*u'*∪*w'*	*u*
s	*b*	*b*	*u*	*s*	*d*	*d*	*b'*	*u'*∪*w'*	*u*
d	*b*	*b*	*u*∪*v*	*d*	*d*	*d*	*b'*	*P*	*u*∪*v*

Properties of Composition

- transitive
  - if $[i_1\ r\ i_2]$ and $[i_2\ q\ i_3]$ then $[i_1\ (r \bullet q)\ i_3]$
- distributive
  - $(r \cup q) \bullet s = (r \bullet s) \cup (q \bullet s)$
  - $s \bullet (r \cup q) = (s \bullet r) \cup (s \bullet q)$
- not commutative
  - $[i_1\ (r \bullet q)\ i_2]$ does not imply $[i_1\ (q \bullet r)\ i_2]$
  - example: $d \bullet b = \{b\}$; $b \bullet d = \{b,m,o,s,d\}$

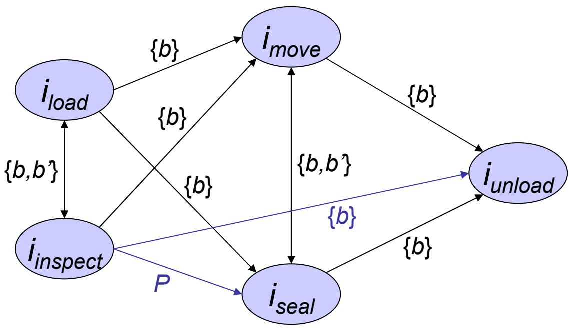


$[i_1\ (d \bullet b)\ i_3]$



$[i_1\ (b \bullet d)\ i_3]$

Inspect and Seal Example: Interval Constraint Propagation



- $i_{inspect}$-i_{move}-i_{unload}: $[i_{inspect}\ \{b\} \bullet \{b\}\ i_{unload}] = [i_{inspect}\ \{b\}\ i_{unload}]$
- $i_{inspect}$-i_{load}-i_{seal}: $[i_{inspect}\ \{b,b'\} \bullet \{b\}\ i_{seal}] = [i_{inspect}\ P\ i_{seal}]$

IA Constraint Networks

- A binary IA constraint network is a directed graph (X,C), where:
  - $X = \{i_1,\ldots,i_n\}$ is a set of interval variables $i_j=(i_j.b, i_j.e)$, where $i_j.b \leq i_j.e$, and
  - $C \subseteq X \times X$ (the edges), c_{ij} is labelled by a constraint $r_{ij} \in R$ iff $[i_i\ r_{ij}\ i_j]$ holds.
- A tuple $\langle v_1,\ldots,v_k \rangle$ of pairs of real numbers $(v_i.b, v_i.e)$ is a solution for (X,C) iff $v_i.b \leq v_i.e$ $i_i=v_i$ satisfy all the constraints in C.
- (X,C) is consistent iff there exists at least one solution.

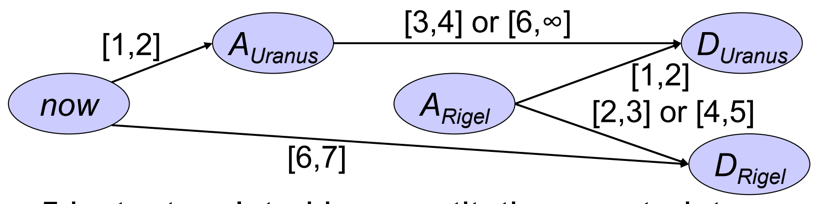
Primitives in Consistent Networks

- **Proposition**: A IA network (X,C) is consistent iff
  - there is a set of primitives $p_{ij} \in r_{ij}$ for every $c_{ij} \in C$ such that
    - for every k: $p_{ij} \in (p_{ik} \bullet p_{kj})$
- idea: filter out redundant primitives using path consistency algorithm until
  - either: no more redundant primitives can be found
  - or: we find a constraint that is reduced to $\varnothing$ (inconsistency)
- note: path consistency not complete for IA networks

Example: Quantitative Temporal Relations

- ship: Uranus
  - arrives within 1 or 2 days
  - will leave either with
    - light cargo (stay docked 3 to 4 days) or
    - full load (stay docked at least six days)
- ship: Rigel
  - to be serviced on
    - express dock (stay docked 2 to 3 days)
    - normal dock (stay docked 4 to 5 days)
  - must depart 6 to 7 days from now
- Uranus must depart 1 to 2 days after Rigel arrives

Example: Quantitative Temporal Constraint Network



- 5 instants related by quantitative constraints
  - e.g. $(2 \leq D_{Rigel}\text{-}A_{Rigel} \leq 3) \vee (4 \leq D_{Rigel}\text{-}A_{Rigel} \leq 5)$
- possible questions:
  - When should the Rigel arrive?
  - Can it be serviced on a normal dock?
  - Can the Uranus take a full load?

Overview

- Actions and Time Points
- Interval Algebra and Quantitative Time
- Planning with Temporal Operators

Temporally Qualified Expressions (*tqe*)

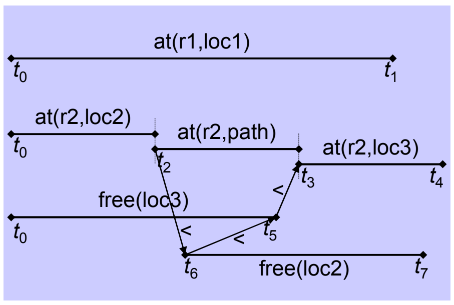
- *tqe*: expression of the form:
  $p(o_1,\dots,o_k)@[t_b,t_e[$
  where:
  - p is a flexible relation in the planning domain,
  - $o_1,\dots,o_k$ are object constants or variables, and
  - t_b,t_e are temporal variables such that $t_b<t_e$.
- *tqe* $p(o_1,\dots,o_k)@[t_b,t_e[$ asserts that:
  - for every time point t: $t_b \le t < t_e$ implies that $p(o_1,\dots,o_k)$ holds
  - $[t_b,t_e[$ is semi-open to avoid inconsistencies

Temporal Database

- A <u>temporal database</u> is a pair $\Phi=(\mathcal{F},\mathcal{C})$ where:
  - $\mathcal{F}$ is a finite set of *tqe*s,
  - $\mathcal{C}$ is a finite set of temporal and object constraints, and
  - $\mathcal{C}$ has to be consistent, i.e. there exist possible values for the variables that meet all the constraints.

Temporal Database: Example

- robot r1 is at location loc1
- robot r2 moves from location loc2 to location loc3



Φ = ({
at(r1,loc1)@[t_0,t_1[,
at(r2,loc2)@[t_0,t_2[,
at(r2,path)@[t_2,t_3[,
at(r2,loc3)@[t_3,t_4[,
free(loc3)@[t_0,t_5[,
free(loc2)@[t_6,t_7[},
{ adjacent(loc2,loc3),
$t_2<t_6<t_5<t_3$ })

Inference over *tqes*

- A set $\mathcal{F}$ of *tqe*s supports a (single) *tqe* $e=p(v_1,\ldots,v_k)@[t_b,t_e[$ iff:
  - there is a *tqe* $p(o_1,\ldots,o_k)@[t_1,t_2[$ in $\mathcal{F}$ and
  - there is a substitution σ such that:
    - $\sigma(p(v_1,\ldots,v_k)) = p(o_1,\ldots,o_k)$.
- An enabling condition for *e* in $\mathcal{F}$ is the conjunction of the following constraints:
  - $t_1 \le t_b$, $t_e \le t_2$ and
  - the variable binding constraints in σ.

Inference over *tqes*: Example

- $\mathcal{F}$ = {at(r1,loc1)@[t_0,t_1[, at(r2,loc2)@[t_0,t_2[, at(r2,path)@[t_2,t_3[, at(r2,loc3)@[t_3,t_4[, free(loc3)@[t_0,t_5[, free(loc2)@[t_6,t_7[}
  - $\mathcal{F}$ supports free(*l*)@[*t*,*t'*[
  - with enabling conditions:
    - $t_0 \le t$, $t' \le t_5$, and l=loc3, or
    - $t_6 \le t$, $t' \le t_7$, and l=loc2.

Inference over Sets of *tqes*

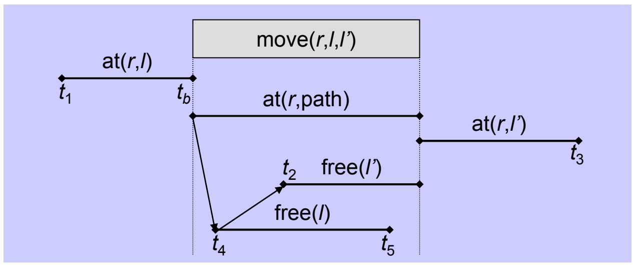
- A set $\mathcal{F}$ of *tqe*s supports a set $\mathcal{E}$ of *tqe*s iff:
  - there is a substitution σ such that:
    - $\mathcal{F}$ supports every *tqe* $e \in \mathcal{E}$ using substitution σ.
- The set of enabling conditions for a single *tqe* e in $\mathcal{F}$ is denoted $\Theta(e/\mathcal{F})$.
- The set of enabling conditions for a set of *tqe*s $\mathcal{E}$ in $\mathcal{F}$ is denoted $\Theta(\mathcal{E}/\mathcal{F})$.

Inference over Temporal Databases

- A temporal database $\Phi=(\mathcal{F},\mathcal{C})$ supports a set $\mathcal{E}$ of *tqe*s iff:
  - $\mathcal{F}$ supports $\mathcal{E}$ and
  - there is an enabling condition $c \in \Theta(\mathcal{E}/\mathcal{F})$ that is consistent with $\mathcal{C}$.
- A temporal database $\Phi=(\mathcal{F},\mathcal{C})$ supports another temporal database $\Phi'=(\mathcal{F}',\mathcal{C}')$ iff:
  - $\mathcal{F}$ supports $\mathcal{F}'$ and
  - there is an enabling condition $c \in \Theta(\mathcal{F}'/\mathcal{F})$ such that
    - $\mathcal{C}' \cup c$ is consistent with $\mathcal{C}$.
- A temporal database $\Phi=(\mathcal{F},\mathcal{C})$ entails another temporal database $\Phi'=(\mathcal{F}',\mathcal{C}')$ iff:
  - $\mathcal{F}$ supports $\mathcal{F}'$ and
  - there is an enabling condition $c \in \Theta(\mathcal{F}'/\mathcal{F})$ such that
    - $\mathcal{C}$ entails $\mathcal{C}' \cup c$.

Temporal Planning Operators: Example

- move(r,l,l')@[t_b,t_e[
  - preconditions: at(r,l)@[t_1,t_b[, free(l')@[t_2,t_e[
  - effects: at(r,path)@[t_b,t_e[, at(r,l')@[t_e,t_3[, free(l')@[t_4,t_5[
  - constraints: t_b<t_4<t_2, adjacent(l,l')



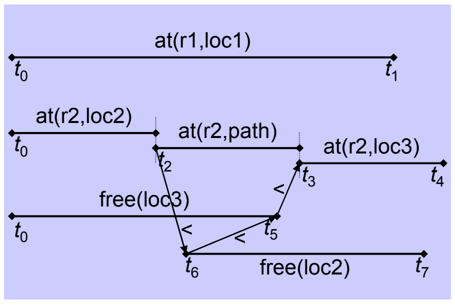
Temporal Planning Operators

- A <u>temporal planning operator *o*</u> is a tuple (name(o), precond(o), effects(o), constr(o)), where:
  - name(o) is an expression of the form $a(x_1,\ldots,x_k,t_b,t_e)$ such that:
    - a is a unique operator symbol,
    - $x_1,\ldots,x_k$ are the object variables appearing in o, and
    - t_b,t_e are temporal variables in o,
  - precond(o) and effects(o) are sets of *tqe*s, and
  - constr(o) is a conjunction of the following constraints:
    - temporal constraints on t_b,t_e and possibly further time points,
    - rigid relations between objects, and
    - binding constraints of the form x=y, x≠y, or x∈D.

Applicability of Temporal Planning Operators

- A temporal planning operator o is applicable to a temporal database $\Phi=(\mathcal{F},\mathcal{C})$ iff:
  - precond(o) is supported by $\mathcal{F}$ and
  - there is an enabling condition c in Θ(precond(o)/$\mathcal{F}$) such that:
    - $\mathcal{C} \cup$ constr(o) $\cup$ c is consistent.
- The result of applying an applicable action a to Φ is a set of possible temporal databases
  - $\gamma_0(\Phi,a)$ = { ($\mathcal{F} \cup$ effects(a), $\mathcal{C} \cup$ constr(a) $\cup$ c) | $c \in \Theta$(precond(a)/$\mathcal{F}$) }

Applicable Operator: Example



- operator: move(r,l,l')@[t_b,t_e[
  - at(r1,loc1)@[t_0,t_1[supports at(r,l)@[t'_1,t_b[
  - free(loc2)@[t_6,t_7[supports free(l')@[t'_2,t_e[
  - enabling condition: {r=rob1, l=loc1, l=loc1, $t_0 \leq t'_1$, $t_b \leq t_1$, $t_6 \leq t'_2$, $t_e \leq t_7$}
  - consistent
- move(r1,loc1,loc2) is applicable

Domain Axioms: Example

- no object can be in two places at the same time:
  $\{\text{at}(r,l)@[t_b,t_e[,\ \text{at}(r',l')@[t'_b,t'_e[\} \rightarrow$
  $(r \neq r') \vee (l = l') \vee (t_e \leq t'_b) \vee (t'_e \leq t_b)$
- every location can be occupied by one robot only:
  $\{\text{at}(r,l)@[t_1,t'_1[,\ \text{free}(l')@[t_2,t'_2[\} \rightarrow$
  $(l \neq l') \vee (t'_1 \leq t_2) \vee (t'_2 \leq t_1)$

Domain Axioms

- A domain axiom α is an expression of the form: cond(α) → disj(α) where:
  - cond(α) is a set of *tqe*s and
  - disj(α) is a disjunction of temporal and object constraints.
- A temporal database $\Phi=(\mathcal{F},\mathcal{C})$ is consistent with α iff:
  - cond(α) is supported by $\mathcal{F}$ and
  - for every enabling condition $c_1 \in \Theta(\text{cond}(\alpha)/\mathcal{F})$
    - there is at least one disjuct $c_2 \in$ disj(α) such that
      - $\mathcal{C} \cup c_1 \cup c_2$ is consistent.

Temporal Planning Domains

- A temporal planning domain is a triple $\mathcal{D} = (S_\Phi, \mathcal{O}, X)$ where:
  - S_Φ is the set of all temporal databases that can be defined with the constraints and the constant, variable, and relation symbols in our representation,
  - $\mathcal{O}$ is the set of temporal planning operators, and
  - X is a set of domain axioms.

Temporal Planning Problems

- A temporal planning problem in $\mathcal{D}$ is a triple $\mathcal{P} = (\mathcal{D}, \Phi_0, \Phi_g)$ where:
  - $\mathcal{D} = (S_\Phi, \mathcal{O}, X)$ is a temporal planning domain,
  - $\Phi_0 = (\mathcal{F}, \mathcal{C})$ is a database in S_Φ that satisfies the axioms in X.
    - represents the initial scenario including:
      - initial state of the world
      - predicted evolution independent of planned actions
  - $\Phi_g = (\mathcal{G}, \mathcal{C}_g)$ is a database in S_Φ where:
    - $\mathcal{G}$ is a set of *tqe*s representing the goals of the problem
    - $\mathcal{C}_g$ are object and temporal constraints on variables in $\mathcal{G}$.

Statement of a Planning Problem

- A statement of a planning problem is a tuple $P = (\mathcal{O}, X, \Phi_0, \Phi_g)$ where:
  - is a set of temporal planning operators,
  - is a set of domain axioms,
  - $\Phi_0=(\mathcal{F}_0,\mathcal{C}_0)$ is a database in S_Φ representing the initial scenario, and
  - $\Phi_g=(\mathcal{G},\mathcal{C}_g)$ is a database in S_Φ representing the goals of the problem.

Concurrent Actions

- problem: swap locations of two robots
  - only one robot at each location at any time
  - path may hold multiple robots
- move(r1,loc1,loc2): not applicable
- move(r2,loc2,loc1): not applicable
- apply both at the same time: applicable

- temporal planning can handle such concurrent actions

Temporal Planning Procedure

```
TPS(Ω)
    flaws ← Ω.getFlaws()
    if flaws=∅ then return Ω
    flaw ← flaws.chooseOne()
    resolvers ← flaw.getResolvers(Ω)
    if resolvers=∅ then return failure
    resolver ← resolvers.selectOne()
    Ω' ← Ω.refine(resolver)
    return TPS(Ω')
```

Structure of Ω

- $\Omega = (\Phi,\mathcal{G},\mathcal{K},\pi)$: current processing stage of the planning problem, where:
  - $\Phi = (\mathcal{F},\mathcal{C})$: current temporal database, initially Φ_0
  - $\mathcal{G}$: set of current open goals, initially taken from $\Phi_g=(\mathcal{G},\mathcal{C}_g)$
  - $\mathcal{K} = \{C_1,\ldots,C_i\}$: set of pending conditions (initially empty):
    - sets of enabling conditions of actions and
    - sets of consistency conditions of axioms,
  - π: set of actions in the current plan, initially empty.

Flaw Type: Open Goal Resolver: Existing *tqe*

- goal: unsupported *tqe e* in $\mathcal{G}$
- assumption:
  - tqe in $\mathcal{F}$ that can support *e*
- resolver:
  - $\mathcal{K} \leftarrow \mathcal{K} \cup \{\Theta(e/\mathcal{F})\}$
  - $\mathcal{G} \leftarrow \mathcal{G} - \{e\}$

Flaw Type: Open Goal Resolver: New Action

- goal: unsupported *tqe e* in $\mathcal{G}$
- assumption:
  - action *a* (instance of operator *o*)
    - has effects(*a*) that support *e* and and
    - constr(*a*) are consistent with $\mathcal{C}$
- resolver:
  - $\pi \leftarrow \pi \cup \{a\}$
  - $\mathcal{F} \leftarrow \mathcal{F} \cup$ effects(*a*)
  - $\mathcal{C} \leftarrow \mathcal{C} \cup$ constr(*a*)
  - $\mathcal{G} \leftarrow (\mathcal{G} - \{e\}) \cup$ precond(*a*)
  - $\mathcal{K} \leftarrow \mathcal{K} \cup \{\Theta(\text{precond}(a)/\Phi)\}$

Flaw Type: Unsatisfied Axiom
Resolver: Add Conditions

- axiom α: cond(α) → disj(α) and
  - cond(α) is supported by $\mathcal{F}$
  - disj(α) is not supported by $\mathcal{F}$
- assumption:
  - there are consistency conditions $\Theta(\alpha/\Phi)$ such that disj(α) is supported by $\mathcal{F}$
- resolver:
  - $\mathcal{K} \leftarrow \mathcal{K} \cup \{\Theta(\alpha/\Phi)\}$

Flaw Type: Threat
Resolver: Add Constraints

- consistency condition $C_i \in \mathcal{K}$ that is not entailed by Φ
- assumption:
  - $c \in C_i$ is consistent with $\mathcal{C}$
- resolver:
  - $\mathcal{C} \leftarrow \mathcal{C} \cup c$
  - $\mathcal{K} \leftarrow \mathcal{K} - \{C_i\}$

Overview

- Actions and Time Points
- Interval Algebra and Quantitative Time
- Planning with Temporal Operators