Planning with Time and Resources

Yuanlin Zhang

Outline

1. Temporal representation and reasoning
2. Planning with temporal operators
3. Integrating planning and scheduling
4. Specific systems embedding time

- Temporal references and relations
  - Discrete time steps
  - Time points (Point Algebra (PA))
    Relations: $t_1 < t_2$, $t_1 = t_2$
  - Time intervals (Interval Algebra (IA))
    Relations: t_1 `meets (before, equal, overlaps, during, starts, finishes)` t_2
- Temporal constraint networks
  - Simple temporal problems (STP)
  - Disjunctive temporal problems (DTP)
  - Temporal problems with preferences
  - Temporal problems with contingent variables
  - Temporal problem with resource constraints

- Reasoning with temporal relations: the satisfiability of the temporal relations (and entailment?)
  - IA – NP complete
  - PA – efficient algorithms
  - STP – efficient algorithms (shortest path)
  - STP with preferences – efficient algorithms
  - STP with contingent variables – efficient algorithms
  - DTP – NP complete, effective algorithms developed
- Embed representation and reasoning to general representation and reasoning system
  - First order logic [Allen83]
  - Temporal planning
  - ...

Planning with temporal operators

- Temporal planning problems and plans
  - Temporal expressions and Temporal databases
  - Temporal planning operators
  - Domain axioms
- Concurrent actions with interfering effects
- A temporal planning procedure

Atuomated planning by *Ghallab, Nau and Traverso*
Temporal data base management, *T. Dean* and *D. mcDermott*, AIJ 1987

Outline

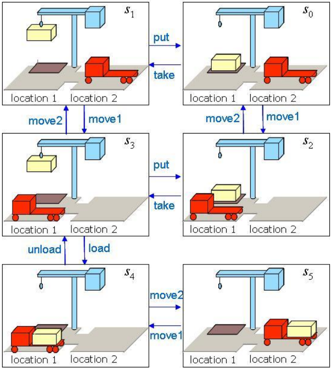
Temporal expressions and Temporal databases

- Symbols: *constant symbols, variable symbols (object variable, temporal variable)*, *object variables* range over constant symbols while *temporal variables* range over the reals
- Relation symbols: *rigid relation symbols, flexible relation symbols (fluents)*
- Constraints: *temporal constraints*: PA (difference constraints) over temporal variables. *binding constraints* on object variables: $x = y$, $x \neq y$, and $x \in D$ where D is a set of constant symbols.

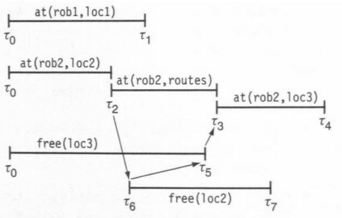
- A *temporally qualified expression (tqe)* is an expression $p(\zeta_i, \ldots, \zeta_k)@[t_s, t_e)$ where p is a flexible relation symbol, $\zeta_i, \ldots, \zeta_k$ are constants or object symbols, and t_s, t_e are temporal variables such that $t_s < t_e$.
  $\forall t$ such that $t \in [t_s, t_e)$, $p(\zeta_i, \ldots, \zeta_k)$ holds at the time t.
- A *temporal database* is a pair $\Phi = (\mathcal{F}, \mathcal{C})$ where $\mathcal{F}$ is a finite set of *tqes* and $\mathcal{C}$ is a finite set of temporal and object constraints, and is satisfiable.

A domain example: Dock-worker robots

A set of locations { `loc1, loc2`, ...}, a set of robots { `rob1, rob2`, ... }, a set of cranes { `k1, k2`, ...}, a set of piles { `p1, p2`, ... }, a set of containers { `c1, c2`, ... }, a symbol `pallet` denotes the pallet that sits at the bottom of a pile.



A example of temporal database



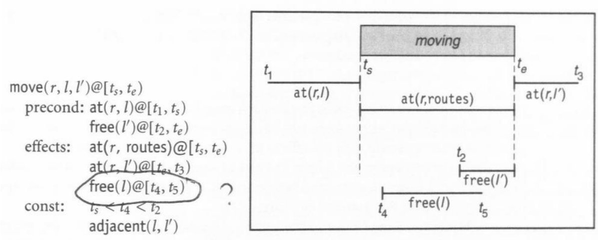
- Φ = ({ at(rob1,loc1)@[τ_0, τ_1), at(rob2,loc2@[τ_0, τ_2) ... }, { adjacent(loc1,loc2), ..., $\tau_2 < \tau_6 < \tau_5 < \tau_3$})

Remarks about temporal database

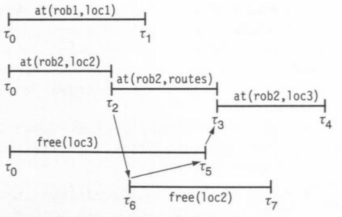
- A temporal database represents about how the world changes over time.
- The representation does not have logical connectives. Particularly, no negated atoms. CWA: a flexible relation holds in a temporal database only during the periods of time explicitly stated by *tqes* in the database; a rigid relation holds iff it is in the database.

A *temporal planning operator* is a tuple $o = (name, precond, effects, const)$, where

- *name* is of the form $o(x_1, \ldots, x_k, t_s, t_e)$ such that o is an operator symbol, $x_1, \ldots, x_k$: object variables and temporal variables in const.
- *precond* and *effects*: *tqes*,
- *const*: a conjunction of temporal constraints and object constraints (either rigid relations or binding constraints).



Supported *tqes*



- free (l) @$[t, t')$ is supported by the two intervals in the database under the constraints: $\{l$ = loc3, $\tau_0 \leq t, t' \leq \tau_5\}$ or $\{l$ = loc2, $\tau_6 \leq t, t' \leq \tau_7\}$

- A set $\mathcal{F}$ of *tqes supports* a *tqe* $e = p(\zeta_i, \ldots, \zeta_k)@[t_1, t_2)$ iff there is in $\mathcal{F}$ a *tqe* $p(\zeta'_i, \ldots, \zeta'_k)@[\tau_1, \tau_2)$ and a subsititution σ such that $\sigma(p(\zeta_i, \ldots, \zeta_k)) = \sigma(p(\zeta'_i, \ldots, \zeta'_k))$. An enabling condition for e in $\mathcal{F}$ is the conjunction of the two temporal constraints $\tau_1 \leq t_1$ and $t_2 \leq \tau_2$, together with the binding constraint σ.
  The set of enabling condition for e is denoted by $\theta(e/\mathcal{F})$.
- $\mathcal{F}$ supports a set of *tqes* ε iff there is a substitution σ that unifies every element of ε with an element of $\mathcal{F}$. An *enabling condition* for ε is the conjunction of enabling conditions for the elements of ε. All possible enabling conditions for ε in $\mathcal{F}$ is denoted by $\theta(\varepsilon/\mathcal{F})$.

Supported temporal database

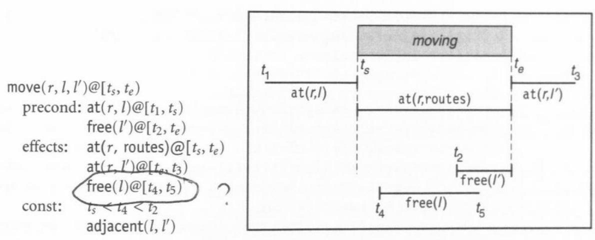
A temporal database $\Phi = (\mathcal{F}, \mathcal{C})$ *supports* a set of *tqes* ε when $\mathcal{F}$ supports ε and there is an enabling condition $c \in \theta(\varepsilon/\mathcal{F})$ that is consistent with $\mathcal{C}$. $\Phi = (\mathcal{F}, \mathcal{C})$ *supports* another temporal database $(\mathcal{F}', \mathcal{C}')$ when $\mathcal{F}$ 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 $\mathcal{C}$. $\Phi = (\mathcal{F}, \mathcal{C})$ *entails* another temporal database $(\mathcal{F}', \mathcal{C}')$ iff $\mathcal{F}$ supports $\mathcal{F}'$ and there is an enabling condition $c \in \theta(\mathcal{F}'/\mathcal{F})$ such that $\mathcal{C} \models \mathcal{C}' \cup c$.

Outline

1. Temporal representation and reasoning
2. Planning with temporal operators
   - Temporal expressions and Temporal databases
   - Temporal planning operators
   - Domain axioms
   - Temporal planning problems and plans
   - Concurrent actions with interfering effects
   - A temporal planning procedure
3. Integrating planning and scheduling

A *temporal planning operator* is a tuple $o = (name, precond, effects, const)$, where

- *name* is of the form $o(x_1, \ldots, x_k, t_s, t_e)$ such that o is an operator symbol, $x_1, \ldots, x_k$: object variables and temporal variables in const.
- *precond* and *effects*: *tqes*,
- *const*: a conjunction of temporal constraints and object constraints (either rigid relations or binding constraints).



Applicability of actions

An *action* is a partially instantiated planning operator $a = \sigma(o)$ for some substitution.

If the precondition and the constraints of an action hold with respect to some database, then the action is applicable. It will run from t_s to t_e. The new *tqes* resulting from its execution are described by its effects.

Formally, an action a is *applicable* to $\Phi = (\mathcal{F}, \mathcal{C})$ iff precond(a) is supported by $\mathcal{F}$ and there is an enabling condition $c \in \theta(precond(a)/\mathcal{F})$ such that $\mathcal{C} \cup const(a) \cup c$ is satisfiable.

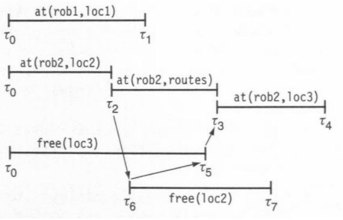
Results of applying an action

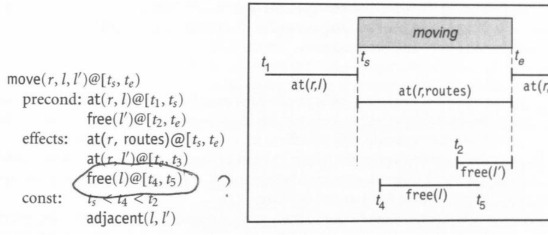
The *results* of applying an action a to a database $\Phi = (\mathcal{F}, \mathcal{C})$ is a set of databases.

$$\gamma_0(\Phi, a) = \{(\mathcal{F} \cup \text{effects}(a), \mathcal{C} \cup \text{const}(a) \cup c) \mid c \in \theta(\text{precond}(a)/\mathcal{F})\}$$

Example 14.3: `move(rob1,loc1,loc2)`$@[t_s, t_e)$ and Φ in the picture. An enabling condition of this action is

$$c = \{r = \texttt{rob1}, l = \texttt{loc1}, l' = \texttt{loc2}, \tau_0 \leq t_1, t_s \leq \tau_1, \tau_6 \leq t_2, t_e \leq \tau_7\}$$





Outline

1. Temporal representation and reasoning
2. Planning with temporal operators
   - Temporal expressions and Temporal databases
   - Temporal planning operators
   - Domain axioms
   - Temporal planning problems and plans
   - Concurrent actions with interfering effects
   - A temporal planning procedure
3. Integrating planning and scheduling

A domain axiom is a conditional expression of the form:

$$\rho : cond(\rho) \rightarrow disj(\rho)$$

where 1) cond(ρ) is a set of *tqes*; 2) disj(ρ) is a disjunction of temporal and object constraints.
Example: an object cannot be in two distinct places at the same time.

$$\{at(r, l)@[t_s, t_e), at(r', l')@[t'_s, t'_e)\} \rightarrow$$
$$(r \neq r') \vee (l = l') \vee (t_e \leq t'_s) \vee (t'_e \leq t_s))$$

A database Φ is *consistent* with an axiom ρ iff for each enabling condition $c_1 \in \theta(cond(\rho)/\mathcal{F})$, there is at least one disjunct $c_2 \in disj(\rho)$ such that $\mathcal{C} \cup c_1 \cup c_2$ is satisfiable.
A Database Φ is *consistent* with a set of axioms if it is consistent with every axiom in X.
A database Φ *satisfies* an atom ρ iff for each enabling condition $c_1 \in \theta(cond(\rho)/\mathcal{F})$, there is at least one disjunct $c_2 \in disj(\rho)$ such that $\mathcal{C} \cup c_1 \models c_2$.
A Database Φ *satisfies* a set of axioms if it satisfies every axiom in X.

After apply an action to a database, we need to augment the new database such that the augmented database will satisfy the axioms. All augmented databases after the application of a are denoted by $\gamma(\Phi, a)$.

Outline

1. Temporal representation and reasoning
2. Planning with temporal operators
   - Temporal expressions and Temporal databases
   - Temporal planning operators
   - Domain axioms
   - Temporal planning problems and plans
   - Concurrent actions with interfering effects
   - A temporal planning procedure
3. Integrating planning and scheduling

Planning domain and problems

A *temporal planning domain* is a triple $\mathcal{D} = (\Lambda, \mathcal{O}, X)$ where

- Λ is the set of all temporal databases
- $\mathcal{O}$ is a set of temporal operators
- X is a set of domain axioms

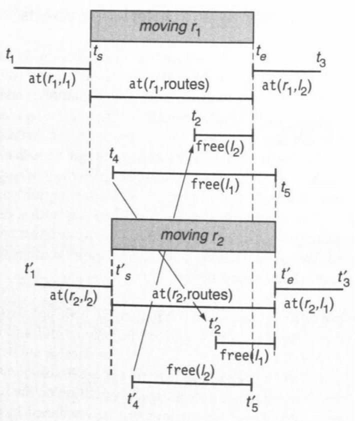
A *temporal problem* in $\mathcal{D}$ is a tuple $\mathcal{P} = (\mathcal{D}, \Phi_0, \Phi_g)$ where

- $\Phi_0 \in \Lambda$
- $\Phi_g = (\mathcal{G}, \mathcal{C}_g) \in \Lambda$ (goals of the problem as a set of $\mathcal{G}$ of *tqes* together with a set $\mathcal{C}_g$ of objects and temporal constraints)

A plan is a set $\pi = \{a_1, a_2, \ldots, a_k\}$ of actions.
Let $\gamma(\Phi, \pi)$ denote the result after applying the actions of π. π is a *solution* for a problem $\mathcal{P}$ iff there is a database in $\gamma(\Phi, \pi)$ that entails Φ_g.

Outline

- In classical planning, one can introduce a new action for the combination of two interfering actions so that their joint preconditions and effects can be expressed.
- With *tqes*, one can avoid the introduction of new actions.
- Example. `r1` and `r2` initially at `loc1, loc2` respectively. Consider `move(r1,loc1,loc2)`@$[t_s, t_e)$ and `move(r2,loc2,loc1)`@$[t'_s, t'_e)$.



Applicability of actionS

A pair of actions $\{a_1, a_2\}$ is *applicable* to $\Phi = (\mathcal{F}, \mathcal{C})$ when:

- $\mathcal{F} \cup effects(a_2)$ supports precond(a_1),
- $\mathcal{F} \cup effects(a_1)$ supports precond(a_2),
- $c_1 \in \theta(a_1/(\mathcal{F} \cup effects(a_2))$, and $c_2 \in \theta(a_2/(\mathcal{F} \cup effects(a_1))$, such that $\mathcal{C} \cup const(a_1) \cup c_1 \cup const(a_2) \cup c_2$ is satisfiable.

The application of $\{a_1, a_2\}$ is

$$\gamma(\Phi, \{a_1, a_2\}) = \cup_i \{\psi(\Phi_i, X) \mid \Phi_i \in \gamma_0(\Phi, \{a_1, a_2\})\}$$

where

$$\begin{aligned}\gamma_0(\Phi, \{a_1, a_2\}) = \{(&\mathcal{F} \cup \mathit{effects}(a_1) \cup \mathit{effects}(a_2), \\ &\mathcal{C} \cup \mathit{const}(a_1) \cup c_1 \cup \mathit{const}(a_2) \cup c_2) \\ &\mid c_1, c_2 \text{same as above}\}\end{aligned}$$

Applicability of a set of actions

The applicability of an action in the set is defined with respect to $\mathcal{F}$ and the effects of all the other actions in the set.

Outline

1. Temporal representation and reasoning
2. Planning with temporal operators
   - Temporal expressions and Temporal databases
   - Temporal planning operators
   - Domain axioms
   - Temporal planning problems and plans
   - Concurrent actions with interfering effects
   - A temporal planning procedure
3. Integrating planning and scheduling

Ideas

A planning problem $(\mathcal{O}, X, \Phi_0, \Phi_g)$. To solve the problem we maintain a data structure $\Omega = (\Phi, G, \mathcal{K}, \pi)$. Initially, $\Phi = (\mathcal{F}_0, \mathcal{C}_0 \cup \mathcal{C}_g)$, $G = \mathcal{G}$, $\pi = \emptyset$, and $\mathcal{K} = \emptyset$.
G: open goals. $\mathcal{K}$: set of pending enabling constraints and consistency conditions.

Open goals

Open goals. For $e \in G$, a *resolver* is either

- A *tqe* in $\mathcal{F}$ supports *e*. Ω is refined as

$$\mathcal{K} \leftarrow \mathcal{K} \cup \{\theta(e/\mathcal{F})\}$$
$$G \leftarrow G - \{e\}$$

- An action *a* whose effects(*a*) supports *e* and const(a) is consistent with $\mathcal{C}$. In this case, Ω is refined as

$$\pi \leftarrow \pi \cup \{a\}$$
$$\mathcal{F} \leftarrow \mathcal{F} \cup effects(a)$$
$$\mathcal{C} \leftarrow \mathcal{C} \cup const(a)$$
$$G \leftarrow (G - \{e\}) \cup precond(a)$$
$$\mathcal{K} \leftarrow \mathcal{K} \cup \{\theta(a/\mathcal{F})\}$$

Unsatisfied Axioms

(some axiom is not satisfied), to resolve this,

$$\mathcal{K} \leftarrow \mathcal{K} \cup \{\theta(X/\Phi)\}.$$

Threats

$C_i \in \mathcal{K}$ needs to be entailed by Φ. To resolve this, update Ω as

$$\mathcal{C} \leftarrow \mathcal{C} \cup c, c \in \mathcal{C}_i \text{ and consistent with } \mathcal{C}$$

$$\mathcal{K} \leftarrow \mathcal{K} - \{C_i\}$$

The procedure will resolve the open goal and threats until there is no open goal and threats.

Atuomated planning by *Ghallab, Nau and Traverso*

Temporal databases (chronicles) + resources

- Planning domain and problem are represented by state variables. There is also a set of resource variables.
- A *temporal assertion* is like: $z@t : -q$, $z@t : +q$, $z@[t, t') : q$.
- In addition to the temporal constraints, we have the resource constraints: the use of a resource z should never exceed its capacity.
- Planner needs to detect and resolve the resource conflict.

Outline

1. Temporal representation and reasoning
2. Planning with temporal operators
3. Integrating planning and scheduling
4. Specific systems embedding time
   - A general theory about action and time
   - Constraint-based attribute and interval planning
   - Temporal planning with continuous changes
   - Adding time and intervals to Golog+HTN
   - PDDL

Towards a general theory of action and time, *James Allen*, AI journal 1984

A temporal logic

- Time interval (e.g., in contrast to time points) is argued to be proper to represent time.
- HOLDS(p, t) denotes property p holds during t.
- Seven relations among time intervals: DURING, STARTS, FINISHES, BEFORE, OVERLAP, MEETS, EQUAL(t_1, t_2).
- Axioms are given on the mutual exclusiveness, transitivity on the intervals. Inference algorithms are also given (in a different paper)

Application of temporal logic to actions

Axioms about `HOLDS`

HOLDS(p, T) *iff* ($\forall t$. IN(t, T) $\Rightarrow$ HOLDS(p, t))

HOLDS(not(P), T) $\iff$ ($\forall t$.IN(t, T) $\Rightarrow \neg$HOLDS(p, t))

OCCUR and OCCURRING axioms

OCCUR(e, t) $\wedge$ IN(t', t) $\Rightarrow \neg$OCCUR(e, t').

change position can be expressed by OCCUR, HOLDS and temporal logic

OCCURRING(p, t) $\Rightarrow \exists t'$. IN(t', t) $\wedge$ OCCURRING(p, t').

FALLING(object) is defined by OCCURRING.

Actions, intensional actions are discussed. Examples: John is running (to school, juggling three balls). "You hid that book from me!"

Outline

Constraint-based attribute and interval planning, *J. Frank* and *A. Jonsson*, Constraints 2002.

- Definition of the planning problems
- Representation of the planning problems
- Build a plan

Background

- Planning is used to sequence the operations for spacecraft both on the ground and on-board.
- Spacecraft operations
    - Involves time constraints: start time, end time, duration
    - Involves resource constraints: memory and power
    - Are mutually constrained in a variety of ways

Properties of the expected system

- Time, resources, mutual exclusion, concurrency (grounding leads to large program)
- Express and meet maintenance goals

Constraint based representation and reasoning system

Interval and attribute

- Given a set of attribute symbol $\mathcal{A}$, a set of atoms $\mathcal{P}$, an *interval* is `holds`(A, n, m, P) where $A \in \mathcal{A}$ and $P \in \mathcal{P}$ and n, m are numbers and $n \leq m$.
- Intended meaning: an attribute of A: a mapping from time to $\mathcal{P}$; `holds`(A, n, m, P): attribute A takes the value of P from time n to m.
- Example: $\mathcal{A}$ = { `Location, Arm-state`}, $\mathcal{P}$ = { `Going(rock, lander), Going(lander,hill), Collect-Sample(hill), Idle(), Off()`}. Intervals: `holds(Location, 10, 20 , Going(rock,lander))`, `holds(Arm-state, 10, 20, Off())`.

Domain constraints

A *domain constraint* describes the necessary conditions under which an interval holds in the domain (dynamic system).

Example: Since the robot arm is fragile, Interval `holds(Location, 10, 20 , Going(rock,lander))` requires the arm to be off during the move: `holds(Arm-State, s, e, Off())` such that $[10, 20] \subseteq [s, e]$.

A domain constraint is *configuration rule* of the form $I \Rightarrow O$ where I is an interval and O is a disjunction of conjunctive intervals. Each disjunct of O is called a *configuration*.

Planning domain and plan

A *planning domain* $\mathcal{D}$ is a tuple (I, A, R) where A is a set of attributes, I intervals, and R a set of configuration rules. A *candidate plan* for a domain $\mathcal{D}$ is a set of intervals. A candidate plan P is an *extension* of plan P_C if $P \subseteq P_C$. A *valid plan* is a candidate plan such that for each interval I, and for each configuration rule R whose head matches I, there is a conjunct of the body of R whose intervals appear also in the plan.

Planning problem

A *planning problem instance* is a tuple $(\mathcal{D}, P_C)$ where $\mathcal{D}$ is the domain and P_C a candidate plan. A *solution* to the instance is a valid plan P that is an extension of P_C.

Representation of planning domain

An interval is $\texttt{holds}(a, s, e, \texttt{p}(x_1, \ldots, x_k)$ where $a, s, e, x_1, \ldots, x_n$ are variables. Each variable has a domain.
A *plan* is a set of intervals with the constraints on the variables of the intervals.
Example: $\texttt{holds(Arm-State,}\ s_o\texttt{,}\ e_o\texttt{,}\ \texttt{Off()),}$ $\texttt{holds(Location,}\ s_g\texttt{,}\ e_g\texttt{,}\ \texttt{Going(}l_g\texttt{,}\ i_g\texttt{))}$ with constraints $s_o \leq s_g, e_g \leq e_o$ and $\texttt{travel(}l_g, i_g, s_g, e_g\texttt{)}$.

Compatibility (configuration rule)

A compatibility (a set of configuration rules[1])

Head: holds($a, s_l, t_l, l(x_1, \ldots, x_k)$)

  Guards: $\{G_i\}$ G_i: the domain of v_i is D_i for $v_i \in \{a, s_l, t_l, x_1, \ldots, x_k\}$

Parameter constraints: $\{C_j(Y_j)\}$

Disjunction of configurations: $\{O_k\}$

  Each configuration O_k is a conjunct of

    an interval J_{kl}, and

    a constraint C_{kl} over the variables of l and J_{kl}.

[1] A configuration rule is ground

Informally, the semantics of a compatibility is that for any plan, $(\texttt{holds}(I) \wedge G_1, \ldots, G_i) \Rightarrow (C_1, \ldots, C_j \wedge (O_1 \vee \ldots \vee O_k))$ where $O_s \equiv (holds(J_{k1}) \wedge C_{j1}) \wedge \cdots \wedge (holds(J_k l) \wedge C_{jl})$. Multiple compatibilities may be applicable to a given interval, and all such compatibilities hold simultaneously.

Example

Before the attribute `Loc` can take `Going(`x, y`)`, it has to be `At` or `Turing`.

Head: `holds(Loc,` s_g, e_g, `Going(`x, y`))`
Parameter constraints: travelTime(x, y, s_g, e_g)
Disjunction of configurations:

Configuration: O_1

Configuration interval: `holds(Arm-State,` s_0, e_0`, Off())`
Configuration constraint: $s_0 \leq s_g, e_g \leq e_0$
Configuration interval: `holds(Loc,` s_a, e_a`, At(`x`))`
Configuration constraint: $s_g = e_a$

Configuration: O_2

- Configuration interval: `holds(Arm-State,` s_0, e_0, `Off())`
- Configuration constraint: $s_0 \leq s_g, e_g \leq e_0$
- Configuration interval: `holds(Loc,` s_a, e_a, `Turning(`x`))`
- Configuration constraint: $s_g = e_a$

Configuration: O_3

- ...
- Configuration interval: `holds(Loc,` s_a, e_a, `At(`y`))`
- Configuration constraint: $e_g = s_a$

Configuration: O_4

- ...
- Configuration interval: `holds(Loc,` s_a, e_a, `Turning(`y`))`
- Configuration constraint: $e_g = s_a$

Example on resources

Head: holds(Resources, s_g, e_g, Change(i, d, f))
Parameter constraints: $f = i - d$
Disjunction of configurations:
  Configuration: O_1
    Configuration interval: holds(Resources, s_0, e_0, Has(x))
    Configuration constraint: $s_g = e_0, i = x$
    Configuration interval: holds(Resources, s_1, e_2, Has(y))
    Configuration constraint: $e_g = s_1, f = y$

Sufficient extension of a plan A plan P is a *sufficient extension* of a plan P_C if 1) for every interval $I \in P_C$, there is $J \in P$ such that I and J matches and the domain of any variable of J is a subset of that of the corresponding variable in I; and 2) P satisfies all the constraints given in P_C.

A procedure to build a plan

Given a planning problem $(\mathcal{D}, P_C)$, find a solution. The following procedure is based on (partial order causal link) POCL planner. Let $P = P_C$.

- For violated compatibility:
  - Constraints are added to force an *existing* interval in P to satisfy the compatibility
  - If not matching intervals in P, add new intervals to satisfy the compatibility
- (? Processing of unsequenced intervals)
- Unassigned variable: nondeterministically select a value from its domain

The procedure is correct and complete.

Constraint representation and processing

In EUROPA (a CAIP implementation), simple temporal network (difference constraints), and procedural constraint. However, no language is provided for for procedural constraint. (A functional language?) A procedural constraint, e.g., travelTime(x, y, s_g, e_g), could be just a procedure to enforce arc consistency or bound consistency.

Outline

1. Temporal representation and reasoning
2. Planning with temporal operators
3. Integrating planning and scheduling
4. Specific systems embedding time
   - A general theory about action and time
   - Constraint-based attribute and interval planning
   - Temporal planning with continuous changes
   - Adding time and intervals to Golog+HTN
   - PDDL

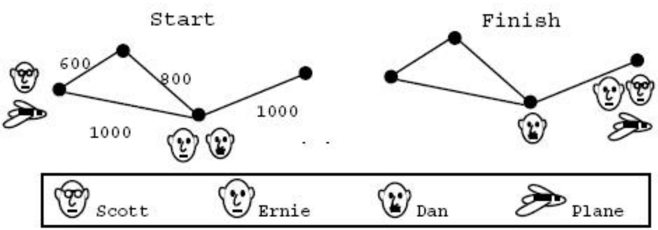
Temporal planning with continuous change, *J. Penberthy* and D. Weld, AAAI-94
ZENO is a planner that allows

- Actions occurring over extended intervals of time
- Deadline goals
- Metric preconditions and effects and continuous change
- Simultaneous actions without interfering effects

Actions and goals

ZENO uses a typed, first-order language with equality to describe goals and the effects of actions. A point-based model of time is adopted; temporal functions and relations use a time point as their first argument. All types except time: finite.

An example problem



Action fast-fly

Schema Fast-Fly (m, l)

at-time: $[t_s, t_e]$

precondition:

$\forall_{\mathtt{time}} t \;\; t \in [t_s, t_e] \supset \; fuel(\mathrm{t}, plane) > 0 \;\; \wedge$

at$(t_s$,plane,$m) \;\; \wedge$

dist$(m,l)=\nu_2 \;\wedge$ mpg(plane)$=\nu_3$

constraints:

$\nu_4 = -600/\nu_3\,, \;\; t_e = t_s + \nu_2/600$

effect:

at$(t_e$,plane,$l) \;\; \wedge$

$\forall_{\mathtt{time}} t \;\; t \in (t_s, t_e] \supset \neg \mathtt{at}(t, \mathtt{plane}, m) \;\; \wedge$

$[\forall_{\mathtt{human}} o \;\; \forall_{\mathtt{time}} t$

$(t \in (t_s, t_e] \wedge \; \mathtt{in}(t, o)) \supset \; \neg \mathtt{at}(t, o, m) \wedge \; \mathtt{at}(t_e, o, l)] \;\; \wedge$

$\forall_{\mathtt{time}} t \;\; t \in [t_s, t_e] \supset \frac{\partial}{\partial t} \mathtt{fuel}(t, \mathtt{plane}) = \nu_4$

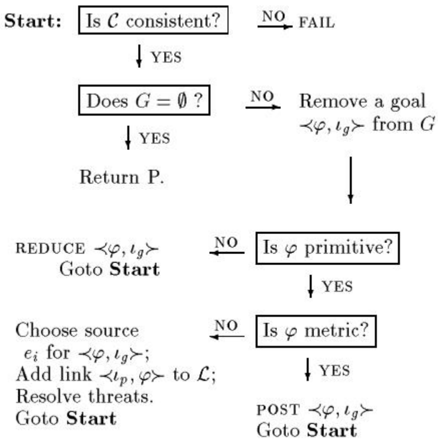
Plans

A *plan* is a tripe $< S, L, C >$ where S is a set of steps (i.e., instantiated action schemata), L is a set of causal links, and C is a set of constraints.

A *planning problem* is defined as a partial plan with a signle dummy step:

Schema Dummy
  at-time: $[t_0, t_1]$
  precondition:
    at(t_1,scott,city-d) $\wedge$ at(t_1,ernie,city-d)
  constraints:
    $t_0 < t_1 \leq t_0 + 5.5$
  effect:
    at(t_0,scott,city-a) $\wedge$ at(t_0,ernie,city-c) $\wedge$
    at(t_0,dan,city-c) $\wedge$ fuel(t_0,plane)=500

Algorithm to build a plan



The search space consists of nodes $< P, G >$ where p is a partially specified plan and G is a goal agenda.
The planner begins at a node where $P =< \mathcal{S}, \mathcal{L}, \mathcal{C} >$.

Outline

1. Temporal representation and reasoning
2. Planning with temporal operators
3. Integrating planning and scheduling
4. Specific systems embedding time
   - A general theory about action and time
   - Constraint-based attribute and interval planning
   - Temporal planning with continuous changes
   - Adding time and intervals to Golog+HTN
   - PDDL

Outline

PSSL2.1: An extension to PDDL for expressing temporal planning domains, *Maria Fox* and *Derek Long*, JAIR 2003

- Numeric expressions, conditions and effects
- Plan metrics
- Durative actgions

Numeric expressions, conditions and effects

```
(define (domain jug-pouring)
    (:requirements : typing : fluents)
    (:types jug)
    (:functions
        (amount ?j - jug)
        (capacity ?j - jug))
    (:action pour
        :parameters (?jug1 ?jug2 - jug)
        :precondition (>= (- (capacity ?jug2) (amount ?jug2))
            (amount ?jug1))
        :effect (and (assign (amount ?jug1) 0)
            (increase (amount ?jug2) (amount ?jug1))))
)
```

Plan metrics

```
(:metric minimize (+ (* 2 (fuel-used car)) (fuel-used truck)))
```

Durative actions

- Discretised durative action: temporally annotated conditions and effects
  - (at start p) // At the start of the interval
  - (at end p) // At the end of the interval
  - (over all p) // over the interval with both ends open
- Continuous durative action

Discretised durative actions

```
(:durative-action load-truck
    :parameters (?t - truck) (?l - location)
                (?o - cargo) (?c - crane)
    :duration (= ?duration 5)
    :condition (and (at start (at ?t ?l))
                    (at start (at ?o ?l))
                    (at start (empty ?c))
                    (over all (at ?t ?l)))
    :effect ( and (at end (in ?o ?t))
                  (at start (holding ?c ?o))
                  (at start (not (at ?o ?l)))
                  (at end (not (holding ?c ?o))))
)
```

Interpretation of concurrent plans

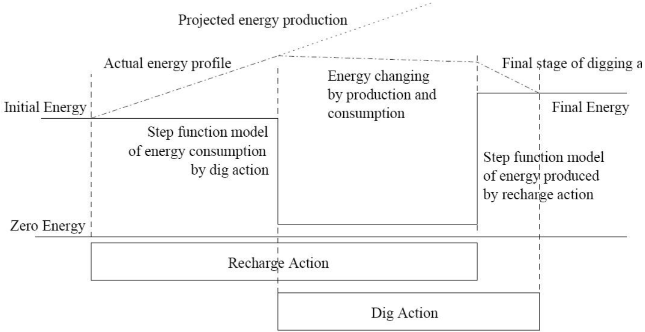
Previous PDDL does not allow concurrent actions. With the introduction of time, concurrent actions are possible in a plan.

Valid plan:

- For an action with precondition P to start at time t, there must be a half open interval immediately preceding t in which P holds
- Conservative on the validity of simultaneous update of and access to a state proposition. Example: simultaneous actions A and B. A has precondition P and effects ($notP$) and Q, while B has precondition $P \vee Q$ and effect R. The application of A and B simultaneously is considered as *ill-defined*. Rule of *no moving targets*: no two actions can simultaneously make use of a value if one of the two is accessing the value to update it – the value is a moving target for the other action to access. (like *mutex lock* in POSIX).

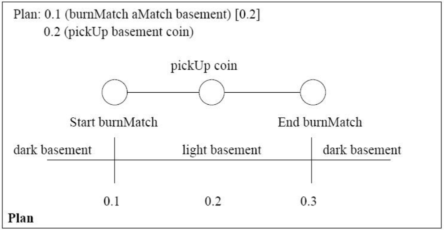
- No numeric value be accessed and updated simultaneously at the start or end point of a durative action.
- ...

Discretised durative action to model the production and consumption of a resource (conservative resource updating.



An example ...

An example of a problem with a durative action useful for its start effects. The duration of burnMatch is between 0 to 5. The action can terminate early if the planner considers it appropriate.



Durative actions with continuous effects

`#t` refers to the coninuously changing time from the start of a durative action during its execution. Continuous effect is represented:

(decrease (fuel-level ?p) (*#t (consumption-rate ?p)))

In contrast, a discretised durative effect

```
(at end (decrease (fuel-level ?p)
    (* (flight-time ?a ?b) (consumption-rate ?p))))
```

An example of fly and refuel (mid-air)

```
(:durative-action fly
     :parameters (?p - airplane ?a ?b - airport)
     :duration (= ?duration (flight-time ?a ?b))
     :condition (and (at start (at ?p ?a))
                     (over all (inflight ?p))
                     (over all (>= (fuel-level ?p) 0)))
     :effect (and (at start (not (at ?p ?a)))
                  (at start (inflight ?p))
                  (at end (not (inflight ?p)))
                  (at end (at ?p ?b))
                  (decrease (fuel-level ?p)
                            (* #t (fuel-consumption-rate ?p)))))

(:action midair-refuel
     :parameters (?p)
     :precondition (inflight ?p)
     :effect (assign (fuel-level ?p) (fuel-capacity ?p)))
```

Semantics

The semantics of the language is based on

- State transition model
- Lifschitz' semantics

Vila's: non-monotonic stuff

Barral's work