Foundations of Automated Planning

Roman Barták

Charles University, Czech Republic

Lecture 04: Planning with time and resources

The conceptual model of planning assumes **implicit time**:

- actions and events are instantaneous (no duration)
- goals are verified at the end of the plan

This restricted view of planning is appropriate for studying the “logic” behind planning (situation calculus) and for formal complexity studies.

In practice the situation is slightly **different**:

- actions take some time (**duration**) to be executed
- action **preconditions** may be required also during action duration (not just at the beginning)
- action **effects** may happen before the end of the action, they may be true during action duration, or even they may become true sometime later
- **effects** of more actions may be **combined**
- **goals** may be required **during execution** of plans

What is time?

The core mathematical structure for describing time is a **set with transitive and asymmetric ordering** relation.

The set can be continuous (real numbers) or discrete (integer numbers).

The planning system will use a **database of temporal references** with a procedure for **verifying consistency** and an **inference mechanism** (to deduce new information).

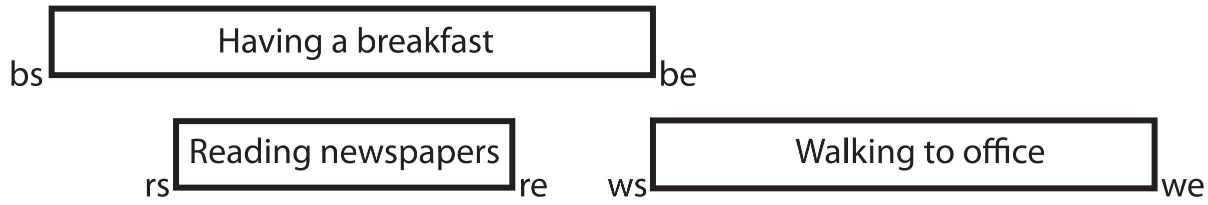
We can model time in two ways:

- **qualitative**
  relative relations (A finished before B)
- **quantitative**
  metric (numerical) relations (A started 23 minutes after B)

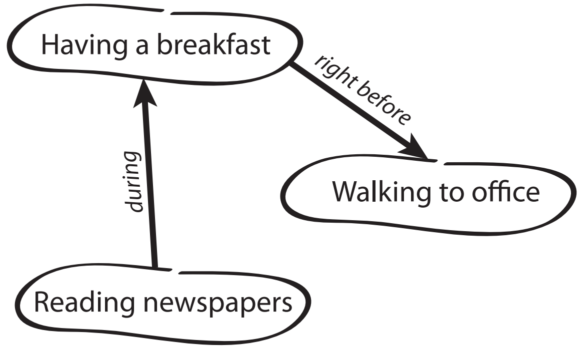
Qualitative approach

Based on **relative temporal relations** between temporal references.

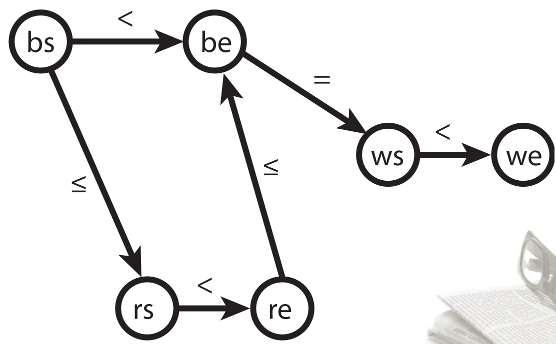
“I read newspapers during breakfast and after breakfast I walked to my office”



Temporal intervals (activities)



Time points (important events)



When **modeling time** we are interested in:

- **temporal references**
  (when something happened or hold)
    - **time points** (instants) when a state is changed
      instant is a variable over the real numbers
    - **time periods** (intervals) when some proposition is true
      interval is a pair of variables (x,y) over the real numbers, such that x<y
- **temporal relations** between temporal references
    - **ordering** of temporal references

Typical problems solved:

- verifying **consistency** of the temporal database
- asking **queries** ("*Did I read newspapers when entering the office?*")
- finding **minimal networks** to deduce inevitable relations

Point algebra - foundations

Symbolic calculus modelling qualitative relations between instants.

- There are three possible **primitive relations** between instants t_1 and t_2:
  - $[t_1 < t_2]$
  - $[t_1 > t_2]$
  - $[t_1 = t_2]$

  Relations P = {<,=,>} are called **primitive relations**.
- Partially known relation between two instants can be modelled using a set (disjunction) of primitive relations:
  - {}, {<}, {=}, {>}, {<,=}, {>,=}, {<,>}, {<,=,>}
- **Relation** r between temporal instants t and t‘ is denoted **[t r t‘]**
- Point algebra allows us to **work with relative relations** without placing the instants to particular (numeric) times.

Let R be a set of all possible relations between two instants

– {{}, {<}, {=}, {>}, {<,=}, {>,=}, {<,>}, {<,=,>}}

Symbolic operations over R:

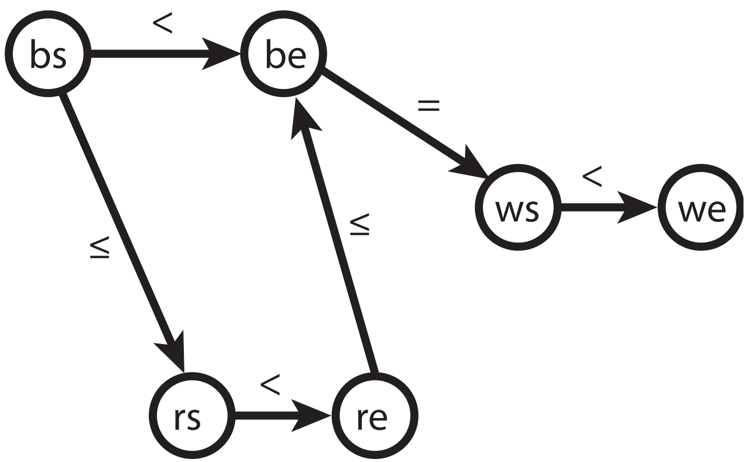
– **set operations** ∩**,** ∪
  - they express conjunction and disjunction of relations
– **composition operation** ∘
  - transitive relation for a pair of connected relations
  - $[t_1\ r\ t_2]$ and $[t_2\ q\ t_3]$ gives $[t_1\ r \circ q\ t_3]$ using the table

∘	<	=	>
<	<	<	P
=	<	=	>
>	P	>	>

The most widely used operations are ∩ and ∘, that allow combining existing and inferred relations:

– $[t_1\ r\ t_2]$ and $[t_1\ q\ t_3]$ and $[t_3\ s\ t_2]$ gives $[t_1\ r \cap (q \circ s)\ t_2]$

"I read newspapers during breakfast and after breakfast I walked to my office"



Query: *"Did I read newspapers when entering the office?"*

$[rs < we] \wedge [we < re]$

$(r_{re,be} \circ r_{be,ws} \circ r_{ws,we}) \cap (r_{re,we})$

$= (\{=,<\} \circ \{=\} \circ \{<\}) \cap \{>\}$

$= \{<\} \cap \{>\} = \{\}$

∘	<	=	>
<	<	<	P
=	<	=	>
>	P	>	>

A set of instants X together with the set of (binary) temporal relations $r_{i,j} \in R$ over these instants C forms a **PA network** (X,C).

- If some relation is not explicitly assumed in C, then we assume universal relation P.

The **PA network** consisting of instants and relations between them is **consistent** if it is possible to assign a real number to each instant in such a way that all the relations between instants are satisfied.

Claim:

The PA network (X,C) is consistent if and only if there exists a set of primitive relations $p_{i,j} \in r_{i,j}$ such that for any triple of such relations $p_{i,j} \in p_{i,k} \circ p_{k,j}$ holds.

Efficient consistency checking:

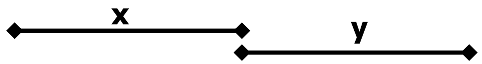
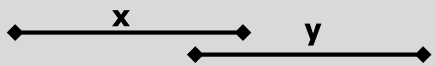
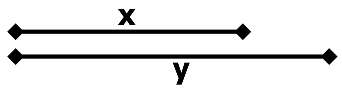
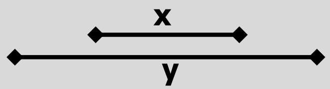
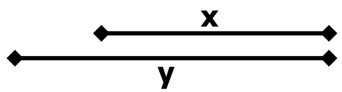
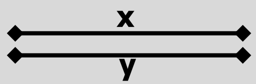
To make the PA network consistent it is enough to make its transitive closure, for example using techniques of **path consistency**.

- for each k: for each i,j: do $r_{i,j} \leftarrow r_{i,j} \cap (r_{i,k} \circ r_{k,j})$
- obtaining {} means that the network is inconsistent

Symbolic calculus modelling relations between intervals

(interval is defined by a pair of instants i^- and i^+, $[i^- < i^+]$)

There are thirteen primitive relations:

x **b**efore y	$x^+ < y^-$	
x **m**eets y	$x^+ = y^-$	
x **o**verlaps y	$x^- < y^- < x^+ \land x^+ < y^+$	
x **s**tarts y	$x^- = y^- \land x^+ < y^+$	
x **d**uring y	$y^- < x^- \land x^+ < y^+$	
x **f**inishes y	$y^- < x^- \land x^+ = y^+$	
x **e**quals y	$x^- = y^- \land x^+ = y^+$	
bi,mi,oi,si,di,fi	symmetrical relations	

Primitive relations can be again combined in sets (2^{13} relations).

Sometimes we select only a subset of possible relations that are useful for a particular application.

- for example {b,m,bi,mi} means no-overlaps and it is useful to model unary resources

set operations ∩, ∪ and the composition operation ∘

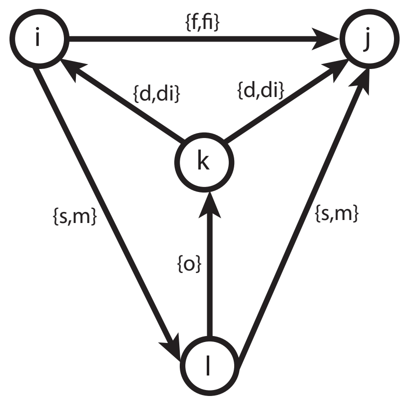
The **IA network** is **consistent** when it is possible to assign real numbers to x_i^-, x_i^+ of each interval x_i in such a way that all the relations between intervals are satisfied.

Claim:

The IA network (X,C) is consistent if and only if there exists a set of primitive relations $p_{i,j} \in r_{i,j}$ such that for any triple of such relations $p_{i,j} \in p_{i,k} \circ p_{k,j}$ holds.

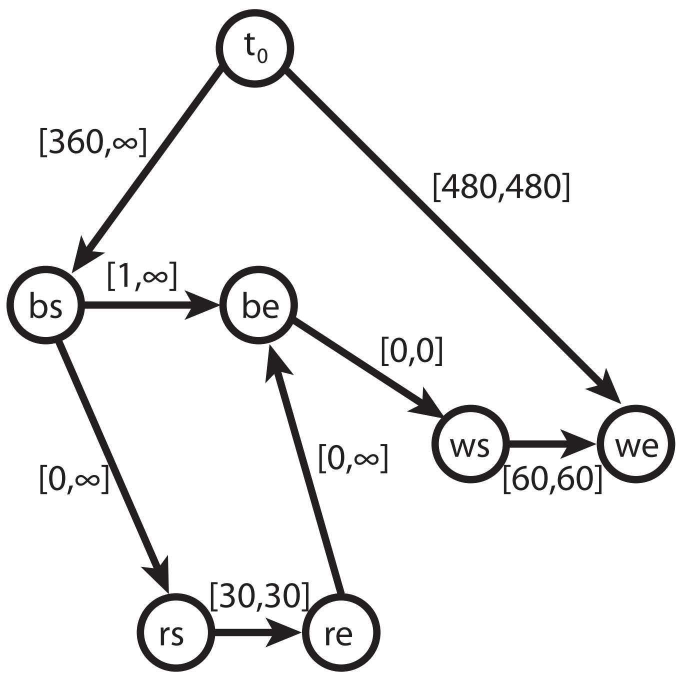
Notes:

- Path consistency is not a complete consistency technique for interval algebra.
- Consistency-checking problem for IA networks is an NP-complete problem.
- Intervals can be converted to instants but some interval relations will not be binary relations among the instants.



"I got up at 6 o'clock. I read newspapers for 30 minutes during the breakfast. After the breakfast I walked to my office which took me one hour. I entered the office at 8:00AM".

When did I start my breakfast?



- 360 =< bs, *"I got up at 6 o'clock"*
- bs =< rs, re =< be, *"I read newspapers during breakfast"*
- re-rs = 30, "I read newspapers for 30 minutes"
- be = ws, *"after breakfast I walked to my office"*
- we-ws = 60, *"[walking] took me one hour"*
- we = 480, *"I entered the office at 8:00AM"*

bs =< rs = re-30 =< be-30 = ws-30 = (we-60)-30 = 390

I started my breakfast between 6:00AM and 6:30AM.

The basic temporal primitives are again **time points**, but now the relations are numerical.

Simple **temporal constraints** for instants t_i and t_j:

- unary: $a_i \leq t_i \leq b_i$
- binary: $a_{ij} \leq t_i - t_j \leq b_{ij}$,

where a_i, b_i, a_{ij}, b_{ij} are (real) constants

Notes:

- Unary relation can be converted to a binary one, if we use some fix origin reference point t_0.
- $[a_{ij},b_{ij}]$ denotes a constraint between instants t_i a t_j.
- It is possible to use disjunction of simple temporal constraints.

Simple Temporal Network (STN)

- only simple temporal constraints $r_{ij}= [a_{ij},b_{ij}]$ are used
- **operations**:
  - composition: $r_{ij} \circ r_{jk} = [a_{ij}+a_{jk}, b_{ij}+b_{jk}]$
  - intersection: $r_{ij} \cap r'_{ij} = [\max\{a_{ij},a'_{ij}\}, \min\{b_{ij},b'_{ij}\}]$
- **STN** is **consistent** if there is an assignment of values to instants satisfying all the temporal constraints.
- **Path consistency** is a complete technique making STN consistent (all inconsistent values are filtered out, one iteration is enough). Another option is using all-pairs minimal distance **Floyd-Warshall algorithm**.

Temporal planning involves reasoning on time.

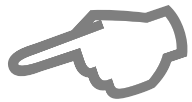
Actions do not describe state transitions only but they specify how the **state variables evolve in time** and what are the **prevailing conditions**:

- actions have **duration**
  - going from A to B takes some time
- **preconditions** must hold at specific time of action execution
  - place B must be free right before arrival
- similarly action **effects** happen at specific times of the action
  - place A is made empty right after leaving it
- actions can **interfere** to achieve a **joint effect**
  - to open doors we need to press the handle and push (or pull) the doors
- **goals** and **known intermediate states** can be spread in time
  - a dock is closed for a given time interval due to maintenance so vessels cannot use it
  - customer A will be served before the customer B

Planning with temporal operators

– Action specification contains information when the preconditions must hold, when the effects become active and there are temporal relations between the time points and intervals.

Planning with chronicles



– Actions describe partially defined functions how the state variables are being changed in time.

Planning graph and time

– Actions are split into three parts – start, middle, and end – and state layers have duration.

Multi-valued state variables describe some properties depending on world states.

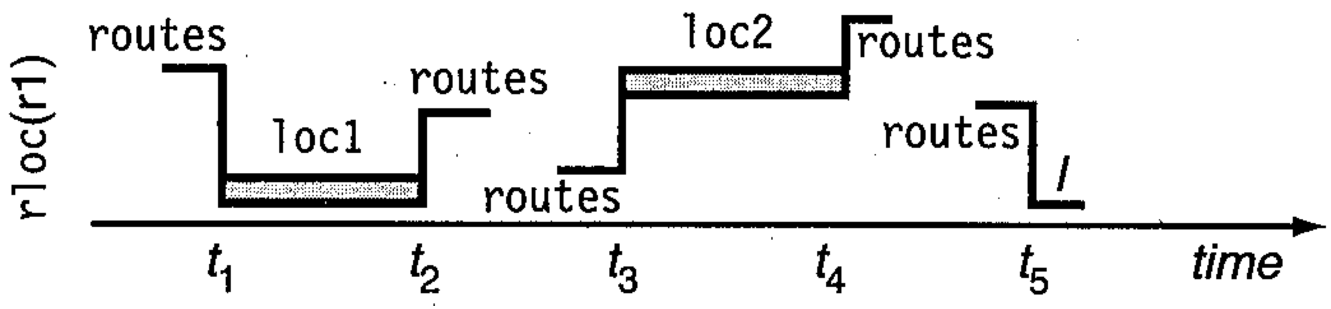
- rloc: robots x S → locations

Now **state variables** will depend on exact **time**:

- rloc: robots x time → locations

Example:

- At time t_1 robot r1 entered place loc1, where it stayed till time t_2 and then left.
- At time t_3, $t_2<t_3$, robot r1 arrived to place loc2, where it stayed till time t_4 and then left.
- At time t_5, $t_4<t_5$, robot r1 arrived to some not-yet specified place l.



The evolution of a state variable can be specified partially with ”holes” where the value is unknown.

- During planning, this evolution will be concretised.

We will restrict to **piecewise constant functions** that can be described using two types of **temporal assertions**:

- **event $x@t:(v_1,v_2)$** specifies the instantaneous change of the value of x from v_1 to v_2 ($v_1 \neq v_2$) at time t
  - $x@t:(v_1,v_2) \equiv (\exists t_0\ \forall t'\ (t_0 < t' < t)\ x(t') = v_1) \land\ x(t) = v_2$
- **persistence condition $x@[t_1,t_2):u$** specifies that the value of x persists as being equal to u over the interval $[t_1,t_2)$
  - $x@[t_1,t_2):u \equiv \forall t\ (t_1 \leq t < t_2)\ x(t) = u$

There is the following relation between events and persistence conditions:

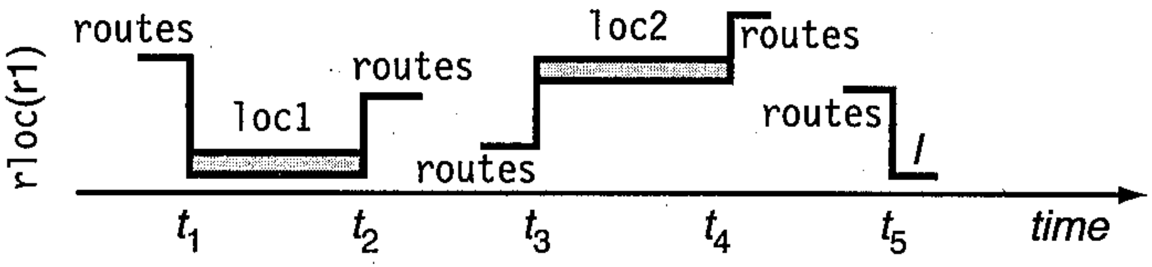
$$x@t:(v_1,v_2) \equiv\ v_1 \neq v_2 \land \exists t_1, t_2\ (t_1 < t < t_2)\ x@[t_1,t):v_1 \land x@[t,t_2):v_2$$

Chronicle

A **chronicle** for a set of state variables is a pair Φ=(F,C), where:

- F is a set of **temporal assertions** over the state variables (i.e. events and persistence conditions)
- C is a set of constraints of two types:
  - **object constraints**, i.e., constraints connecting object variables in the form of x∈D, x=y, x≠y and rigid relations
  - **temporal constraints**, i.e., constraints over the temporal variables using the point algebra (<,=,>)

Timeline is a chronicle for a single state variable.



({ 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,l) }
{ adjacent(l_1,loc1),
adjacent(loc1,l_2),
adjacent(l_3,loc2),
adjacent(loc2,l_4),
adjacent(l_5,l),
t_1<t_2<t_3<t_4<t_5 })

To ensure that the **timeline can specify a valid evolution** of a state variable, there must **not be any two conflicting temporal assertions** – temporal assertions that allow different values of the state variable at the same time.

Temporal conflicts can be avoided by requiring a timeline to contain, either explicitly or implicitly, separation constraints that make each pair of assertions non-conflicting.

The **separation constraint** for a pair of assertions is defined as follows:

- for $x@[t_1,t_2):v_1$ a $x@[t_3,t_4):v_2$ there are three possible separation constraints:
  - $t_2 \leq t_3$, $t_4 \leq t_1$, $v_1 = v_2$
- for $x@t:(v_1,v_2)$ a $x@[t_1,t_2):v$ there are four possible separation constraints:
  - $t<t_1$, $t_2<t$, $(t_1=t \land v=v_2)$, $(t_2=t \land v=v_1)$
- for $x@t:(v_1,v_2)$ a $x@t':(v_1',v_2')$ there are two possible separation constraints:
  - $t \neq t'$, $(v_1=v_1' \land v_2=v_2')$

Note:

- Assertions can also be separated by constraints on difference of the object variables in the assertions (or example assertions for state variables rloc(r) and rloc(r') can be separated by a constraint $r \neq r'$).

Timeline Φ=(F,C) for the state variable x is **consistent** iff C is consistent (there is a solution) and for each pair of temporal assertions from F there is a separation constraint entailed by C.

- the separation constraint can be a part of C
- or it can be entailed by C (to be true in any solution of C)

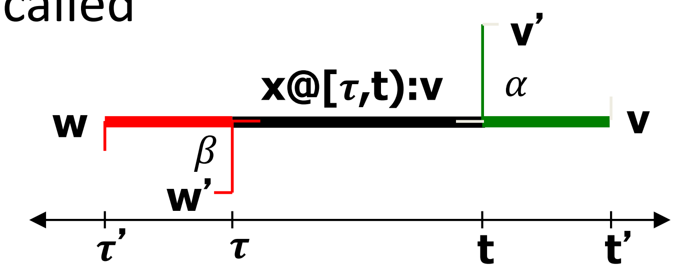
A **chronicle** is **consistent** iff all its timelines are consistent.

Note:

- Consistency requires the separation constraints to be entailed by C; it is not enough if the separation constraints can be added to C without a conflict.

A consistent **chronicle** Φ =(F,C) **supports an assertion** α (α being either **x@t:(v,v')** or **x@[t,t'):v**) iff there is in F an assertion β that asserts a value w for α (β is either **x@τ:(w',w)** or **x@[τ', τ):w**) and there exists a set of separation constraints c such that
Φ ∪ ({α, x@[τ,t):v}, {w=v, τ <t} ∪ c) is a consistent chronicle.

- $\Phi \cup \Phi' = (F \cup F', C \cup C')$, $\Phi \subseteq \Phi' \equiv (F \subseteq F' \wedge C \subseteq C')$,
- β is called a **support** for α in α
- the pair δ = **({α, x@[τ,t):v}, {w=v, τ <t}** ∪ **c)** is called an **enabler** for α in Φ



Notes:

- The chronicle must be consistent before enabling α.
- The enabler is a chronicle.
- The support for α is looked only for value v, that is before the time t. This is because the support will be used as a causal explanation for α.
- There can be several ways to enable an assertion α in Φ.

A consistent **chronicle** Φ =(F,C) **supports a set of assertions** ε iff each assertion $\alpha_i \in \varepsilon$ is supported by (F ∪ ε -{α_i}, C) with an enabler δ_i such that $\Phi \cup \phi$ is a consistent chronicle, where $\phi = \cup_i \delta_i$.

Notes:

- The definition allows an assertion $\alpha_i \in \varepsilon$ to support another assertion $\alpha_j \in \varepsilon$ with respect to Φ as long as the union of the enablers is consistent with Φ. This allows synchronisation of several actions with **interfering effects**.
- ϕ is called an **enabler** for ε (again, the enabler is not unique)

Let Φ'=(F',C') be a chronicle such that Φ supports F' and let **$\theta(\Phi'/\Phi)$ = {ϕ ∪ (∅,C') | ϕ is enabler for F'}** be a set of all possible enablers. Then a consistent **chronicle** Φ =(F,C) **supports chronicle** Φ'=(F',C'), iff Φ supports F' and there is an enabler $\phi \in \theta(\Phi'/\Phi)$ such that $\Phi \cup \phi$ is consistent chronicle.

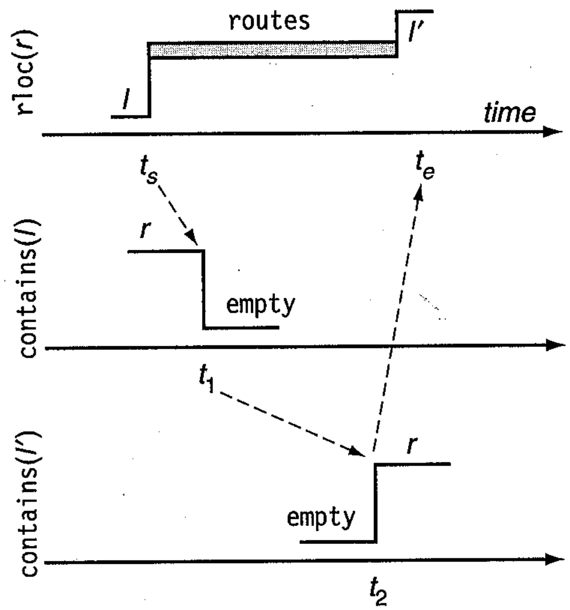
Φ **entails** Φ' iff Φ supports Φ' and there is an enabler $\phi \in \theta(\Phi'/\Phi)$ such that $\phi \subseteq \Phi$.

A chronicle planning operator is a pair o = (name(o), (F(o),C(o))):

- name(o) is a syntactic expression of the form $o(t_s,t_e,t_1,\ldots,v_1,v_2,\ldots)$ containing all temporal and object variables in the operator (o is an operator symbol)
- (F(o),C(o)) is a chronicle

Example *(simplified):*

move(t_s,t_e,t_1,t_2,r,l,l') =
{rloc(r)@t_s : (l,routes),
rloc(r)@[t_s,t_e) : routes,
rloc(r)@t_e : (routes,l'),
contains(l)@t_1 : (r,empty),
contains(l')@t_2 : (empty,r),
$t_s < t_1 < t_2 < t_e$,
adjacent(l,l') }



The differences from classical planning operators are

- **no distinction** between **preconditions and effects**
- **an operator is applied** not to a state but **to a chronicle**
- the result of **applying** an instance of operator to a chronicle is **not unique**

An action is a partially instantiated operator.

Action a=(F(a),C(a)) is **applicable** to a chronicle Φ iff Φ supports the chronicle (F(a),C(a)).

The result of applying a to Φ is not unique but a set of chronicles $\gamma(\Phi,a) = \{\Phi \cup \phi \mid \phi \in \theta(a/\Phi)\}$.

A set of actions $\pi=\{a_1,\ldots,a_n\}$ is **applicable** to Φ iff Φ supports $\Phi_\pi = \cup_i (F(a_i),C(a_i))$.

The result of applying π to Φ is the set of chronicles $\gamma(\Phi,\pi) = \{\Phi \cup \phi \mid \phi \in \theta(\Phi_\pi/\Phi)\}$.

A **temporal planning problem** is a triple $P=(O, \Phi_0, \Phi_g)$, where

- O is a set of chronicle planning operators
- Φ_0 is a consistent chronicle that represents an initial scenario describing the rigid relations, the initial state, and the expected evolution that will take place independently of the actions to be planned
- Φ_g is a consistent chronicle that represents the goals

A **solution plan** for a problem P is a set of actions $\pi=\{a_1,\ldots,a_n\}$, each being an instance of operator in O, such that that there is a chronicle in $\gamma(\Phi_0, \pi)$ that entails Φ_g.

Planning with chronicles

The planning procedure is derived from **plan-space planning**.

For a planning problem P=(O, Φ_0, Φ_g) we start with the chronicle Φ=(F_0,$C_0 \cup C_g$), a set of open goals G=F_g, an empty plan π=∅, and an empty set of threats K= ∅.

```
CP(Φ, G, K, π)
    if G = K = ∅ then return(π)
    perform the two following steps in any order
        if G ≠ ∅ then do
            select any α ∈ G
            if θ(α/Φ) ≠ ∅ then return(CP(Φ, G − {α}, K ∪ {θ(α/Φ)}, π))
            else do
                relevant ← {a | a contains a support for α}
                if relevant = ∅ then return(failure)
                nondeterministically choose a ∈ relevant
                return(CP(Φ ∪ (F(a), C(a)), G ∪ F(a), K ∪ {θ(a/Φ)}, π ∪ {a}))
        if K ≠ ∅ then do
            select any C ∈ K
            threat-resolvers ← {φ ∈ C | φ consistent with Φ}
            if threat-resolvers = ∅ then return(failure)
            nondeterministically choose φ ∈ threat-resolvers
            return(CP(Φ ∪ φ, G, K − C, π))
end
```

Open goal

- is either supported by Φ, then its enablers are added to K
- otherwise, a resolver is an action that supports the goal and this action is added to the system

Threats is a pending set of enablers.

- From each set of enablers we need to select one that is consistent with Φ and its added to Φ.

Now we know how to use **time in planning**

- planning with chronicles

We already have some **resources** in planning

- for example a hand or a crane

A **state variable** with two values occupied/empty is not an efficient model to describe several identical resources – it does not matter which hand is used to pick up the block (the hands are symmetrical).

We can model a set of identical unary resources using a **single multi-valued state variable** describing the **number of available resources**.

- the domain for the variable is **numeric** (the number of resources)
- changes of values are **relative** (the resources are taken and returned)

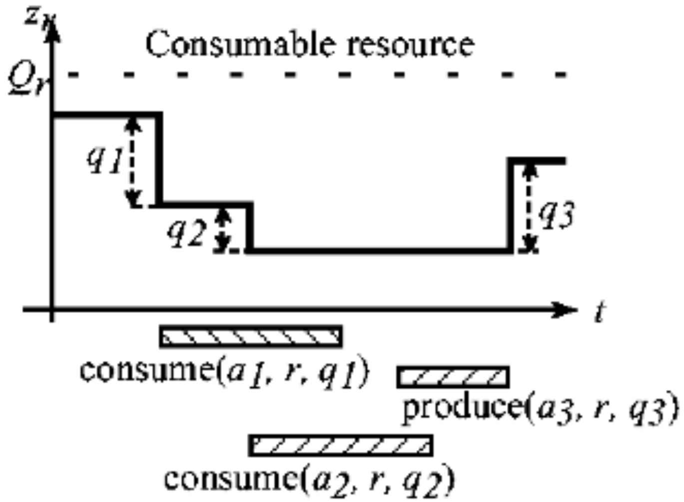
A state variable describes how some property of the object changes in time.

- the changes are **absolute** (location changed from loc1 to loc2)

Similarly we can describe the capacity profile of the resource, i.e., how the available capacity changes with time, using a **capacity variable**.

- resources x time → {0,1,…,Q}, where Q is a maximal capacity
- the domain is numeric
- the changes of values are **relative** (available capacity is increased or decreased by some amount)

Note:
we assume **instant changes**



We can describe changes of capacity variables using **temporal assertions for resources**.

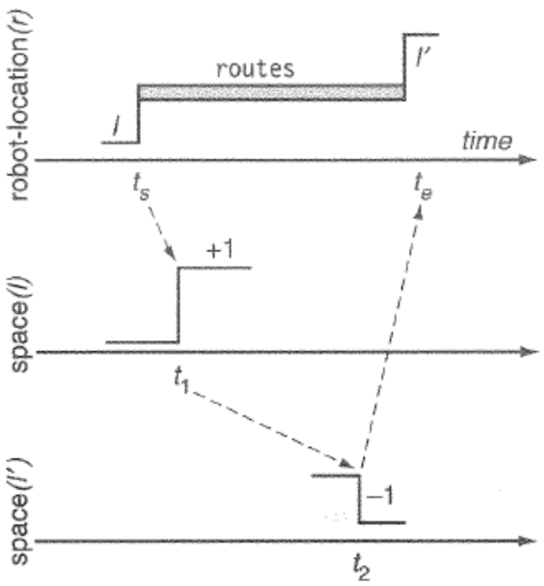
- **decrease** of capacity z@t:-q
- **increase** of capacity z@t:+q
- **borrowing** of capacity z@[t,t'):q

Notes:

- this is a description of **relative** changes
- z@[t,t'):q ≡ z@t:-q ∧ z@t':+q
- z@t:-q ≡ z@[t,∞):q
- z@t:+q ≡ z@0:+q ∧ z@[0,t):q
  - at the beginning we increase the capacity from Q to Q+q and we borrow the increased capacity till time t
- it is necessary to specify the maximal capacity for each capacity variable in the problem description

Planning operator is a chronicle with temporal assertions and constraints.

To work with resources we need to **add** to a chronicle just the **temporal assertions for resources**.



$$
\begin{aligned}
\text{move}(t_s, t_e, t_1, t_2, r, l, l') = \quad \{ \quad & \text{robot-location}(r)@t_s : (l, \text{routes}),\\
& \text{robot-location}(r)@[t_s, t_e) : \text{routes},\\
& \text{robot-location}(r)@t_e : (\text{routes}, l'),\\
& \text{space}(l)@t_1 : +1,\\
& \text{space}(l')@t_2 : -1,\\
& t_s < t_1 < t_2 < t_e,\\
& \text{adjacent}(l, l') \quad \}
\end{aligned}
$$

We will only assume actions that borrow resource capacity so the assertions have the form z@[t,t'):q.

We need to extend the notion of consistency to cover assertions for resources, i.e., to assume capacity limits.

A **set of temporal assertions** R_z for resource z is **conflicting** iff there is a subset $\{z@[t_i,t_i'):q_i \mid i \in I\} \subseteq R_z$ such that:

- assertions from this subset overlap in time, i.e., it is possible to assign times t_i such that $\cap_{i \in I} [t_i,t_i') \neq \emptyset$
- $\Sigma_{i \in I} q_i > Q$

Notes:

- Resource conflict means a possible **exceeding of resource capacity**.
- The resource conflict can only appear between the assertions for the same resource variable.

A **chronicle is consistent** iff all temporal assertions over all state variables are consistent and there is no conflicting set of assertions for capacity variables.

How to discover resource conflicts?

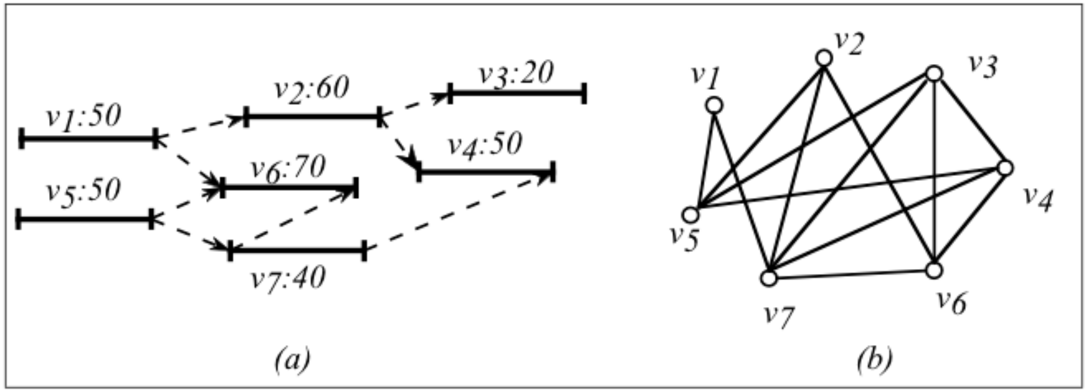
Claim:

Intervals from a set I can overlap iff any pair of intervals from I can overlap.

$(\cap_{i \in I} [t_i,t_i') \neq \emptyset \Leftrightarrow \forall i,j \in I: [t_i,t_i') \cap [t_j,t_j') \neq \emptyset)$

The set of intervals/assertions can be represented using a **graph**:

- nodes describe intervals/assertions
- edges connect nodes with overlapping intervals



We will look for a clique U in the graph such that $\Sigma_{i \in U} q_i > Q$.
More precisely, we will look for smallest (inclusion) cliques with this property – **minimal critical sets** (MCS)

How to find all minimal critical sets?

- index all nodes (in any order)
- for each node, explore in the DFS style all cliques containing this node and the nodes with smaller indexes
- all cliques exceeding the resource capacity are remembered (and not further extended)

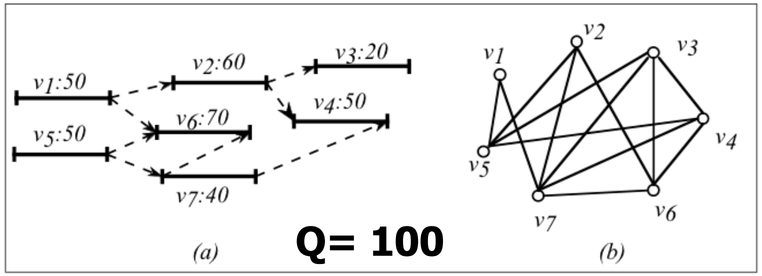
> so-far found part of clique (annotates p in MCS-expand(p))
>
> pending candidates to be included in the clique (they are connected with every node in p) (annotates "pending(p) do")
>
> not finding identical cliques (annotates "$j < i$")

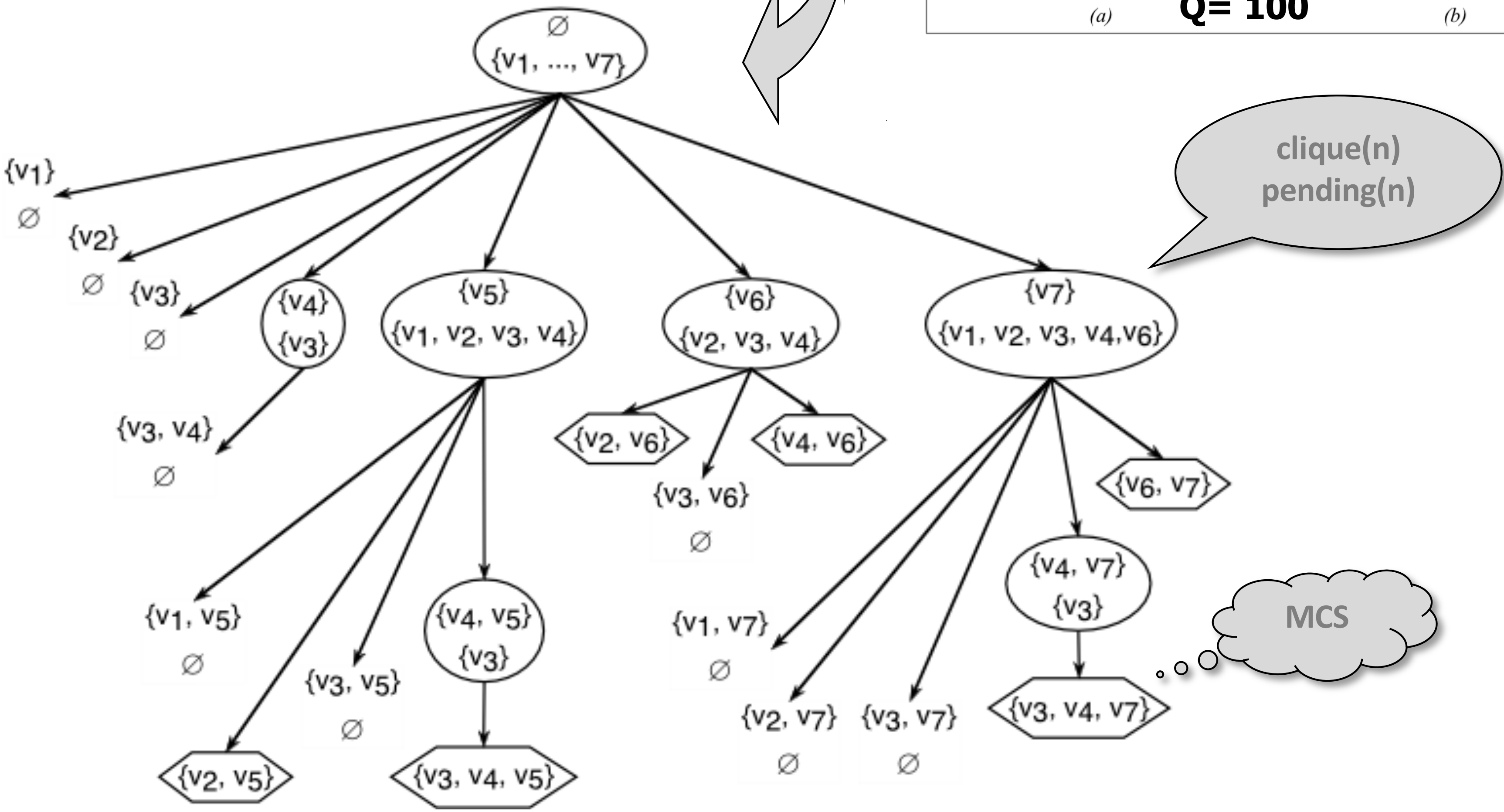
```
MCS-expand(p)
    for each v_i ∈ pending(p) do
        add a new node m_i successor of p
        pending(m_i) ← {v_j ∈ pending(p) | j < i and (v_i, v_j) ∈ E}
        clique(m_i) ← clique(p) ∪ {v_i}
        if clique(m_i) is over-consuming then MCS ← MCS ∪ clique(m_i)
        else if pending(m_i) ≠ ∅ then MCS-expand(m_i)
end
```

- The algorithm starts with a clique found so far (at the beginning it is empty) and a set of pending candidates to be included in the clique (at the beginning contains all nodes).
- We look for possible extensions of the clique by a node v_i (and then nodes with index smaller than i).

Resource conflict detection: an example

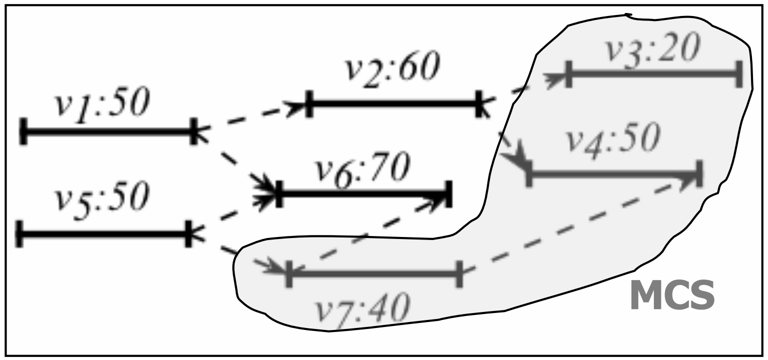
MCS-expand(p)
  for each $v_i \in$ pending(p) do
    add a new node m_i successor of p
    pending(m_i) $\leftarrow \{v_j \in$ pending(p) $\mid j < i$ and $(v_i, v_j) \in E\}$
    clique(m_i) $\leftarrow$ clique(p) $\cup \{v_i\}$
    if clique(m_i) is over-consuming than MCS $\leftarrow$ MCS $\cup$ clique(m_i)
    else if pending(m_i) $\neq \emptyset$ than MCS-expand(m_i)
end





How to remove a resource conflict?

- Let U= {z@[t_i,t_i'):q_i | i∈I} be a minimal critical set then any temporal constraint t_i'< t_j for i,j∈I removes the resource conflict.
  - this constraint removes edge (i,j) from the graph so U is no more a clique
  - any larger clique U': U⊆U' is no more a clique
  - no smaller clique U': U' ⊆ U was conflicting
- Some of suggested temporal **constraints** can be **in temporal conflict** with other constraints.
  - Example: t_4'<t_7 is in conflict with t_7'<t_4' and t_7<t_7'
  - Such resolvers are not used!
- Some suggested constraints are **too strong** (force removal of other edges from the graph).
  - Example: t_4'<t_3 is too strong as it forces t_7'<t_3 (via t_7'<t_4')
  - The planning algorithm will select one resolver to repair MCS so it is better to use only the necessary resolvers so they do not force other resolvers.



CPR($\Phi, G, \mathcal{K}, \mathcal{M}, \pi$)
  if $G = \mathcal{K} = \mathcal{M} = \emptyset$ then return(π)
  perform the three following steps in any order
    if $G \neq \emptyset$ then do
      select any $\alpha \in G$
      if $\theta(\alpha/\Phi) \neq \emptyset$ then return(CPR($\Phi, G - \{\alpha\}, \mathcal{K} \cup \theta(\alpha/\Phi), \mathcal{M}, \pi$))
      else do
        $relevant \leftarrow \{a \mid a$ applicable to Φ and has a provider for $\alpha\}$
        if $relevant = \emptyset$ then return(failure)
        nondeterministically choose $a \in relevant$
        $\mathcal{M}' \leftarrow$ the update of $\mathcal{M}$ with respect to $\Phi \cup (\mathcal{F}(a), \mathcal{C}(a))$
        return(CPR($\Phi \cup (\mathcal{F}(a), \mathcal{C}(a)), G \cup \mathcal{F}(a), \mathcal{K} \cup \{\theta(a/\Phi)\}, \mathcal{M}', \pi \cup \{a\}$))
    if $\mathcal{K} \neq \emptyset$ then do
      select any $C \in \mathcal{K}$
      $threat\text{-}resolvers \leftarrow \{\phi \in C \mid \phi$ consistent with $\Phi\}$
      if $threat\text{-}resolvers = \emptyset$ then return(failure)
      nondeterministically choose $\phi \in threat\text{-}resolvers$
      return(CPR($\Phi \cup \phi, G, \mathcal{K} - C, \mathcal{M}, \pi$))
    if $\mathcal{M} \neq \emptyset$ then do
      select $U \in \mathcal{M}$
      $resource\text{-}resolvers \leftarrow \{\phi$ resolver of $U \mid \phi$ is consistent with $\Phi\}$
      if $resource\text{-}resolvers = \emptyset$ then return(failure)
      nondeterministically choose $\phi \in resource\text{-}resolvers$
      $\mathcal{M}' \leftarrow$ the update of $\mathcal{M}$ with respect to $\Phi \cup \phi$
      return(CPR($\Phi \cup \phi, G, \mathcal{K}, \mathcal{M}', \pi$))
end

We just extent the algorithm for **planning with chronicles** to work with minimal conflict sets (in M) to resolve resource conflicts

© 2023 Roman Barták
Charles University, Prague, Czech Republic
bartak@ktiml.mff.cuni.cz