Course "Automated Planning: Theory and Practice"
Chapter 17: Temporal Planning

Teacher: Marco Roveri - marco.roveri@unitn.it

M.S. Course: Artificial Intelligence Systems (LM)
A.A.: 2025-2026
Where: DISI, University of Trento
URL: https://shorturl.at/A81hf

Last updated: Sunday 16th November, 2025

Terms of Use and Copyright

USE

This material (including video recording) is intended solely for students of the University of Trento registered to the relevant course for the Academic Year 2025-2026.

SELF-STORAGE

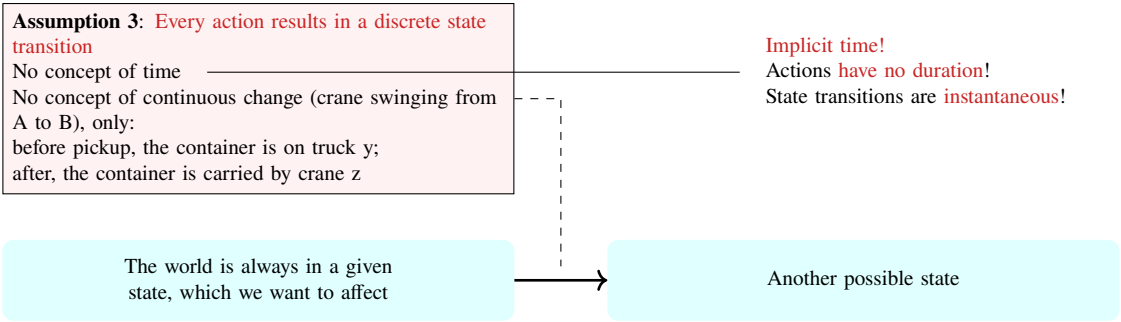
Self-storage is permitted only for the students involved in the relevant courses of the University of Trento and only as long as they are registered students. Upon the completion of the studies or their abandonment, the material has to be deleted from all storage systems of the student.

COPYRIGHT

The copyright of all the material is held by the authors. Copying, editing, translation, storage, processing or forwarding of content in databases or other electronic media and systems without written consent of the copyright holders is forbidden. The selling of (parts) of this material is forbidden. Presentation of the material to students not involved in the course is forbidden. The unauthorised reproduction or distribution of individual content or the entire material is not permitted and is punishable by law.

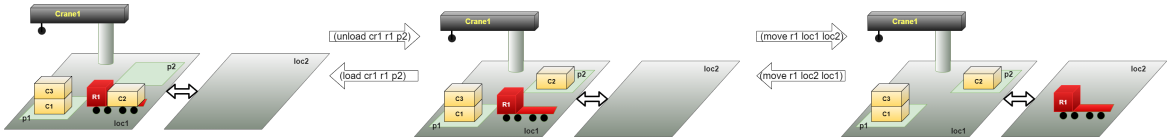
The material (text, figures) in these slides is authored by Marco Roveri with some contribution by A. Micheli and A. Bit-Monnot.

Classical planning - recap lecture 1



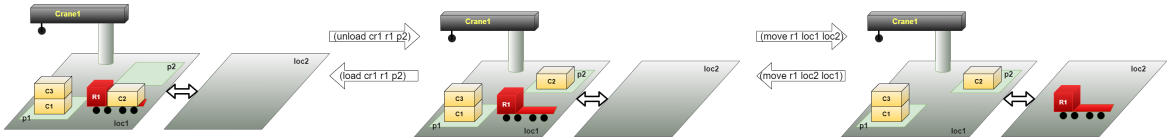
- We must know **when an action is executable** and **what it achieves**
  - $\gamma : S \times A \to 2^S$ $\gamma(s, a)$ is the set of states you may end-up in if you execute a in s;
  - $\gamma(s, a) = \emptyset$ means a cannot be executed in s;
  - $\gamma(s, a) = \{s'\}$ means executing a in s leads to s';

Why Explicit Time?

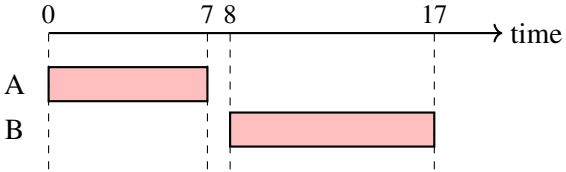


- In reality:
  - actions and events do occur over a time span
  - preconditions not only at beginning
  - effects during or even after the action
  - actions may need to maintain partial states
  - actions may overlap
    - 2nd starts before 1st ends
    - 2nd ends before 1st ends
  - events expected to occur in future time periods
  - goals must be achieved within time bound

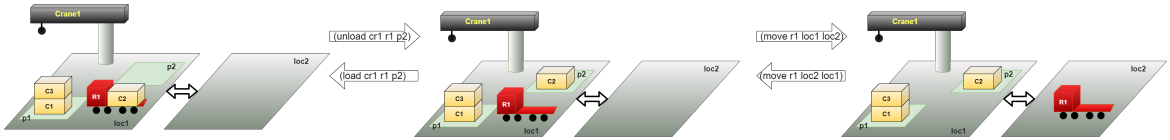
Temporal Planning: Intuition



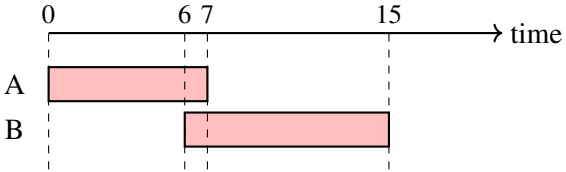
- Time and temporal constraints must be considered
  - A: `(move r1 loc2 loc1)` at 0 with duration 7
  - B: `(load cr1 r1 p2)` at 8 with duration 9



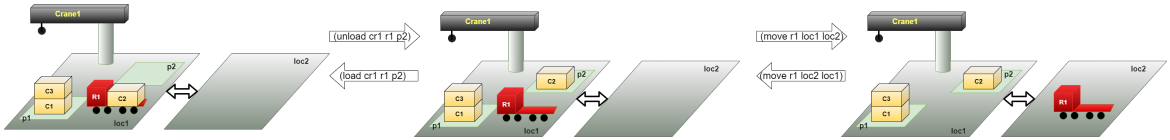
Temporal Planning: Intuition (cont.)



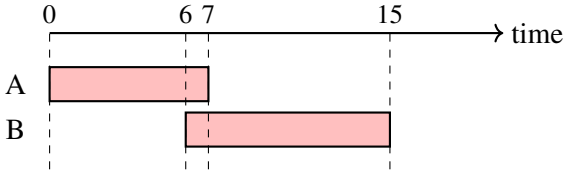
- Actions can overlap and interfere
  - A: `(move r1 loc2 loc1)` at 0 with duration 7
  - B: `(load cr1 r1 p2)` at 6 with duration 9



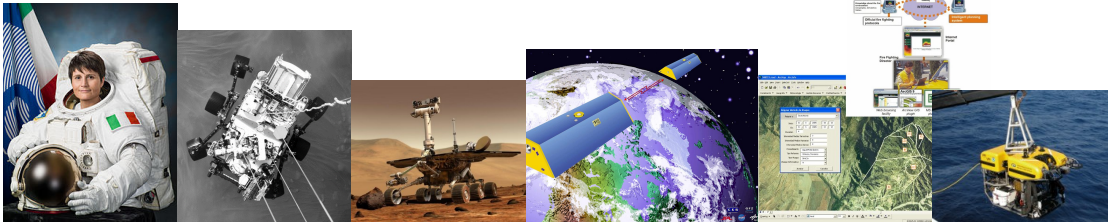
Temporal Planning: Intuition (cont.)



- Temporal Planning $\Longrightarrow$ we want to decide what to do and when to do it!
  - A: `(move r1 loc2 loc1)` at 0 with duration 7
  - B: `(load cr1 r1 p2)` at 6 with duration 9



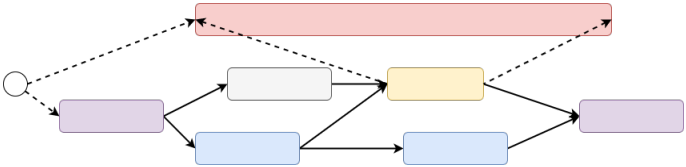
Why Temporal Planning?



- Space Mission Planning – Create daily activity plans for Spirit, Opportunity, Perseverance and for the International Space Station
- Optimal control – logistics, greenhouse logistics, modular printing control, factories
- Safety critical applications – activity plans for satellite hearth observations, plans for firefighting considering limited resources, plans for autonomy in hostile environments

Time and Temporal Reasoning

- Time is a first-class citizen - mathematical structure:
  - set with transitive, asymmetric ordering operations
  - discrete, dense, or continuous
  - bounded or unbounded
  - totally ordered or branching
- Temporal references:
  - time points
  - time intervals
- Temporal relations:
  - examples: before, during



- Answering questions about a plan
  - Consistency (meet deadlines, ...), Occurrence times (latest start time, ...), Ordering (Action A before B, Action A within B, ...), Dispatchability...
  - How many (non-overlapping) actions with duration > 0 can you fit in 10 time-units?

Temporal Reference - time interpretation

Discrete Time

- Events happen at discrete steps (e.g. $\mathbb{Z}$)
- What happens between two steps is not modeled!
- A temporal planner has to find a way to schedule events in the discrete steps

Dense Time

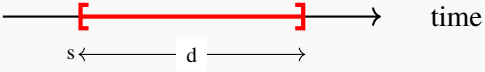
- Events can happen in a dense continuum (e.g. $\mathbb{Q}$ or $\mathbb{R}$)

- Remark: Discrete time seems "simpler" ...
  - ... but integer linear programming is NP-complete...
  - while linear programming over the rationals is polynomial!

Temporal references: Intervals and Timepoints

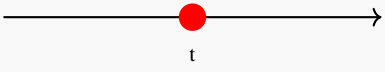
Interval

A contiguous subset of time, characterized by a start time s and a duration d.



Time Point

A single point in time, typically associated to a particular instantaneous event.



Intervals as Time Points

Temporal References: Example

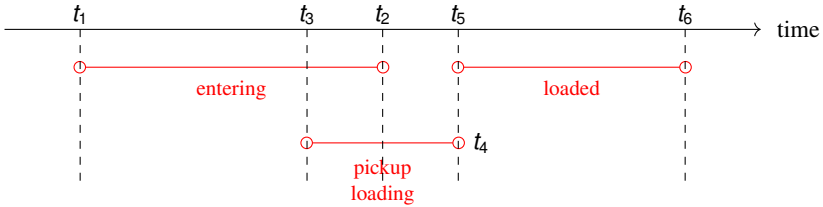
- Load a container `c` onto robot `r` at location `l` with crane `k`: `load(k,c,r,l,I4)`
  - t_1: instant at which robot `r` enters location `l`
  - t_2: instant at which robot `r` stops at location `l`
  - $i_1 = [t_1, t_2]$: interval corresponding to `r` entering `l`
  - t_3: instant at which the crane `k` starts picking up `c`
  - t_4: instant at which crane `k` 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`
  - $i_4 = [t_1, t_6]$: interval corresponding to `load(k,c,r,l,I4)`

Temporal Relations

- Specify (temporal) constraints among temporal references
  - $<, >, \geq, \leq, =, \neq$
  - before, after, meets, contains,
  - starts, ends, overlap, contains,
  - ...

Temporal Relations: Example

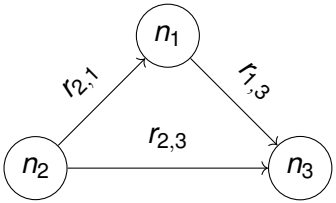
- Load a container `c` onto robot `r` at location `l` with crane `k`: `load(k,c,r,l,I4)`
  - Assumption: crane is allowed to pick up container as soon as robot has entered location
  - Possible temporal sequences:
    - $t_1 < t_3 < t_2 < t_4 = t_5 < t_6$ (see figure) or $t_1 = t_3$ or $t_2 = t_3$ or $t_2 < t_3$



  - No absolute information about durations or time positions, only binary constraints between instants or intervals

Temporal Networks

- A tuple $\langle V, E \rangle$ where
  - V is a finite set of temporal references (i.e. timepoints, intervals)
  - E is a finite set of temporal relations among temporal references
    - $E = \{r_{i,j} | r_{i,j} = n_i \otimes n_j\}$ where $n_i, n_j \in V$ and $\otimes \in \{<, \leq, >, \geq, ..., \text{before}, \text{after}, ...\}$

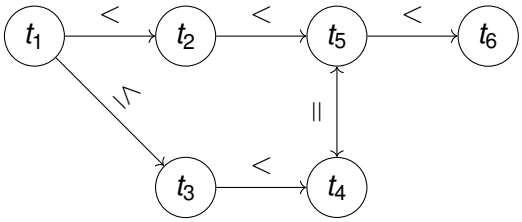


- Temporal Network Consistency: there is an instantiation (concrete value) of all temporal references (nodes) that satisfies all the temporal relations.

Temporal Network: Example

- Load a container `c` onto robot `r` at location `l` with crane `k`: `load(k,c,r,l,I4)`

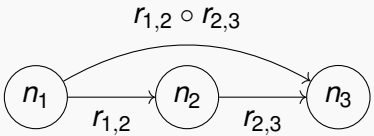
- $V = \{t_1, t_2, t_3, t_4, t_5, t_6\}$
- $E = \left\{ \begin{array}{l} \langle t_1 < t_2 \rangle, \langle t_1 \leq t_3 \rangle, \langle t_2 < t_5 \rangle, \\ \langle t_3 < t_4 \rangle, \langle t_4 = t_5 \rangle, \langle t_5 = t_4 \rangle, \\ \langle t_5 < t_6 \rangle \end{array} \right\}$



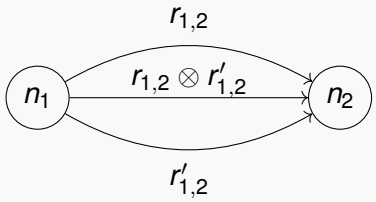
- Consistent: $t_1 = 2.0, t_2 = 3.0, t_3 = 3.0, t_4 = 4.0, t_5 = 4.0, t_6 = 5.0$

Temporal Reasoning Rules

Composition



Set Operations ($\otimes \in \{\cup, \cap, \ldots\}$)



Point Algebra: Relations and Constraints

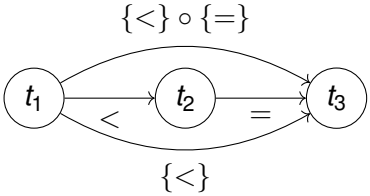
- Point Algebra is a symbolic calculus that enables us to relate in time a set of instants with qualitative constraints without necessarily ordering them.
- Possible primitive relations P between 2 timepoints t_1 and t_2: $P = \{<, =, >\}$
  - Exactly one of the following will be true:
    - $t_1 < t_2$: t_1 before t_2
    - $t_1 = t_2$: t_1 equal t_2 (i.e. same time instant)
    - $t_1 > t_2$: t_1 after t_2
- Possible qualitative constraints R between timepoints: $t_1 \xrightarrow{r \in R} t_2$
  - Sets of the above relations (interpreted as disjunctions)
  - $R = 2^P = \{\emptyset, \{<\}, \{=\}, \{>\}, \{<, =\}, \{<, >\}, \{=, >\}, P\}$

R	Meaning	R	Meaning
$\{<\}$	before	$\{<, =\}$	before or equal ($\leq$)
$\{>\}$	after	$\{>, =\}$	after or equal ($\geq$)
$\{=\}$	equal	$\{>, <\}$	different ($\neq$)
$\{\}$	contradiction	$\{>, =, <\}$	tautology

Point Algebra: Composition

- Given $t_1 \xrightarrow{r_{1,2}} t_2 \xrightarrow{r_{2,3}} t_3$
  - infer $t_1 \xrightarrow{r_{1,3}} t_3$ where
  - $r_{1,3} = r_{1,2} \circ r_{2,3}$

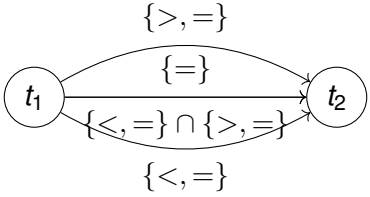
$\circ$	$<$	$=$	$>$
$<$	$<$	$<$	$\{<, =, >\}$
$=$	$<$	$=$	$>$
$>$	$\{<, =, >\}$	$>$	$>$



Point Algebra: Set operations ($\cap, \cup, \ldots$)

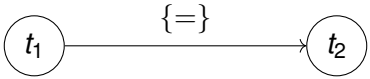
- Given $t_1 \xrightarrow{r_{1,2}} t_2$ and $t_1 \xrightarrow{r'_{1,2}} t_2$
  - infer $t_1 \xrightarrow{r} t_2$ where
  - $r = r_{1,2} \cap r'_{1,2}$

- Remark: $t_1 \xrightarrow{r_{1,2}} t_2$ and $t_1 \xrightarrow{r'_{1,2}} t_2$
  - are weaker than $t_1 \xrightarrow{r_{1,2} \cap r'_{1,2}} t_2$
  - $\Longrightarrow$ they are redundant, and can be removed!



Point Algebra: Set operations ($\cap, \cup, \ldots$)

- Given $t_1 \xrightarrow{r_{1,2}} t_2$ and $t_1 \xrightarrow{r'_{1,2}} t_2$
  - infer $t_1 \xrightarrow{r} t_2$ where
  - $r = r_{1,2} \cap r'_{1,2}$



- Remark: $t_1 \xrightarrow{r_{1,2}} t_2$ and $t_1 \xrightarrow{r'_{1,2}} t_2$
  - are weaker than $t_1 \xrightarrow{r_{1,2} \cap r'_{1,2}} t_2$
  - $\Longrightarrow$ they are redundant, and can be removed!

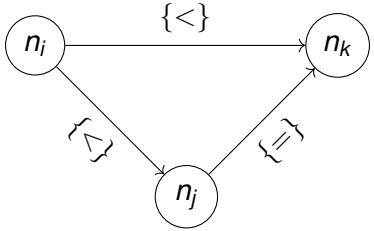
Point Algebra: Properties of Combined Constraints

- Distributive:
  - $(r_1 \cup r_2) \circ r_3 = (r_1 \circ r_3) \cup (r_2 \circ r_3)$
  - $r_1 \circ (r_2 \cup r_3) = (r_1 \circ r_2) \cup (r_1 \circ r_3)$
- Symmetrical constraint r' of r
  - obtained replacing $<$ with $>$ and vice-versa
  - $(r_1 \circ r_2)' = r_2' \circ r_1'$
  - $t_1\ r\ t_2 \leftrightarrow t_2\ r'\ t_1$
- $(R, \cup, \circ)$ is an algebra:
  - R is closed under $\cup$ and
  - $\cup$ is an associative and commutative operation
  - identity element for $\cup$ is $\emptyset$
  - $\circ$ is an associative operation
  - identity element for $\circ$ is $\{=\}$

Point Algebra: Consistency

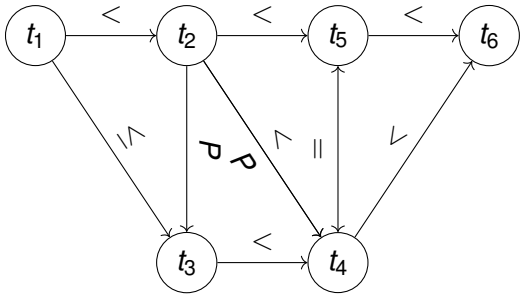
- For each node n_i, n_j, n_k:
  - $r_{n_i,n_k} = r_{n_i,n_k} \cap (r_{n_i,n_j} \circ r_{n_j,n_k})$
  - repeat to fix point

- Intuition
  - Infers all pair-wise orderings
  - Network is consistent if all pairs of edges have a valid ordering
    - $\Longrightarrow$ there are no $n_i \xrightarrow{\{\}} n_j$



- Properties
  - This algorithm is:
    - incomplete for general CSPs
    - complete for PA networks

Point Algebra: Consistency Example

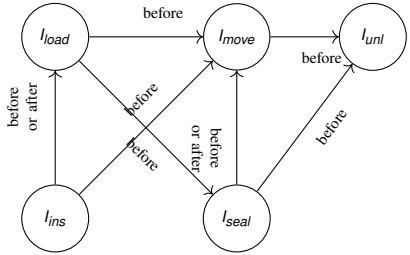


$P = \{<, =, >\}$

- path: $t_4 - t_5 - t_6$: $[t_4 = t_5] \circ [t_5 < t_6]$ implies $[t_4 < t_6]$
- path: $t_2 - t_1 - t_3$: $[t_2 > t_1] \circ [t_1 \leq t_3]$ implies $[t_2\{<, =, >\}t_3]$
- path: $t_2 - t_5 - t_4$: $[t_2 < t_5] \circ [t_5 = t_4]$ implies $[t_2 < t_4]$
- path: $t_2 - t_3 - t_4$: $[t_2\{<, =, >\}t_3] \circ [t_3 < t_4]$ implies $[t_2\{<, =, >\}t_4]$
- last two give $[t_2 < t_4]$

Allen's Interval Algebra

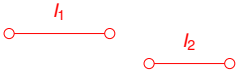
- Qualitative orderings between intervals
  - Let I be an interval:
    - $I.b$ and $I.e$ denote the begin and end time points of the interval
    - $I.b \leq I.e$: beginning before end!
- Example:
  - Every container must be inspected and sealed:
    - inspection: carried out by the crane, must be performed before or after loading
    - sealing: carried out by robot before or after unloading, not while moving
  - Intervals: I_{load}, I_{move}, I_{unl}, I_{ins}, I_{seal}
    - [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 [$I_{seal}.e < I_{move}.b$]



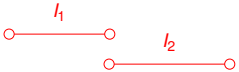
Remark: Disjunction cannot be translated into a binary PA constraint!

Allen's Interval Algebra: Relations

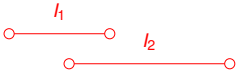
- l_1 before l_2: l_1 b l_2



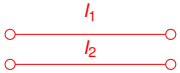
- l_1 meets l_2: l_1 m l_2



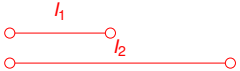
- l_1 overlaps l_2: l_1 o l_2



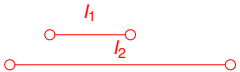
- l_1 equal l_2: l_1 e l_2



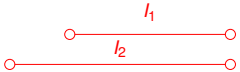
- l_1 starts l_2: l_1 s l_2



- l_1 during l_2: l_1 d l_2



- l_1 finishes l_2: l_1 f l_2



- + all the 6 inverse ($\otimes'$ s.t. $\otimes \in \{b, s, m, d, o, f\}$):
  - l_1 before $l_2 \leftrightarrow l_2$ before' l_1

Allen's Interval Algebra: Operations on Relations

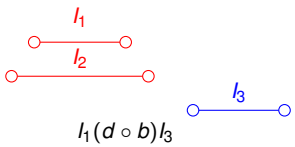
- Set operations: $\cap, \cup, \ldots$
- Composition: $\circ$

∘	e	b	m	o	s	d	f	b'	d'	f'
e	e	b	m	o	s	d	f	b'	d'	f'
b	b	b	b	b	b	u ∪ v	u ∪ v	P	b	b
m	m	b	b	b	m	v	v	u' ∪ v'	b	b
o	o	b	b	u	o	v	v	u' ∪ v'	u ∪ w'	u
s	s	b	b	u	s	d	d	b'	u ∪ w'	u
d	d	b	b	u ∪ v	d	d	d	b'	P	u ∪ v
f	f	m	m	o	d	d	f	b'	u' ∪ v'	x
b'	b'	P	w ∪ u'	o	w ∪ u'	w ∪ u'	b'	b'	b'	b'
d'	d'	u ∪ w'	v'	o	o ∪ w'	y	v'	u' ∪ v'	d'	d'
f'	f'	b	m	o	o	v	x	u' ∪ v'	d'	f'
s'	s'	u ∪ w'	o ∪ w'	o ∪ w'	{s,e,s'}	o' ∪ w	o'	b'	d'	d'

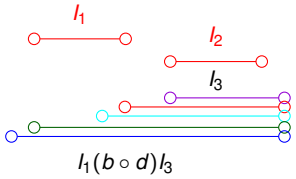
u={b,m,o}; v={o,s,d}; w={d,f}, x={e,f'}

Allen's Interval Algebra: Operations on Relations (cont.)

- Properties of composition
  - transitive
    - if $[I_1\ r\ I_2]$ and $[I_2\ q\ I_3]$ then $[I_1(r \circ q)I_3]$
  - distributive
    - $(r \cup q) \circ s = (r \circ s) \cup (q \circ s)$
    - $s \circ (r \cup q) = (s \circ r) \cup (s \circ q)$
  - not commutative
    - $[I_1(r \circ q)I_2]$ does not imply $[I_1(q \circ r)I_2]$
    - example: $(I_1\{d\}I_2) \circ (I_2\{b\}I_3) = I_1\{b\}I_3$ while $(I_1\{b\}I_2) \circ (I_2\{d\})I_3) = I_1\{b, m, o, s, d\}I_3$

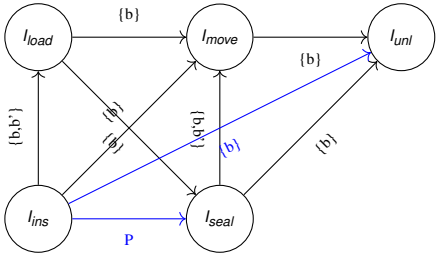


I_3 could start only after I_1



I_3 could start before, at the same time as, during, at the end of, or after I_1

Allen's Interval Algebra: Example



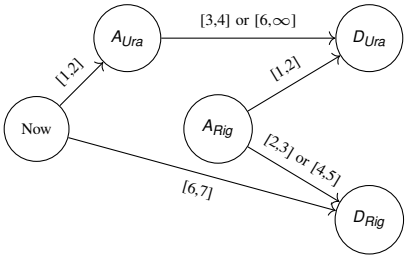
- $l_{ins} - l_{move} - l_{unl}$: $[l_{ins}\{b\} \circ \{b\} l_{unl}] = [l_{ins}\{b\} l_{unl}]$
- $l_{ins} - l_{load} - l_{seal}$: $[l_{ins}\{b, b'\} \circ \{b\} l_{seal}] = [l_{ins}\ P\ l_{seal}]$

Allen's Interval Algebra: Consistency

- For each node n_i, n_j, n_k:
  - $r_{n_i,n_k} = r_{n_i,n_k} \cap (r_{n_i,n_j} \circ r_{n_j,n_k})$
  - repeat to fix point
- Intuition
  - Filter out redundant primitives using path consistency algorithms until
    - either: no more redundant primitive relations can be found
    - or: we find a constraint that is reduced to $\emptyset$ (inconsistency)
- Properties
  - Path consistency is not complete for IA networks!
- Reduce to Point Algebra by considering relations between timepoints $l.b$ and $l.e$

Simple Temporal Networks: Example

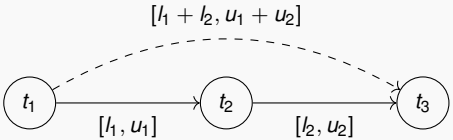
- Quantitative Temporal Relations between timepoints
  - ship: Uranus
    - arrives within 1 or 2 days
    - will leave either with
      - light cargo (stay docked 3 to 4 days) or
      - full load (stay docked at least six day)
  - ship: Rigel
    - to be serviced on
      - express dock (stay docked 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



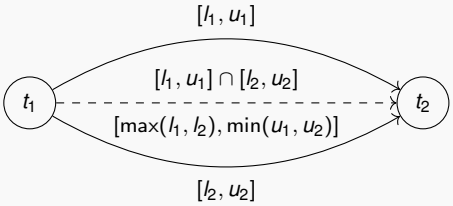
- 5 instants related by quantitative constraints
  - e.g. $(2 \leq D_{Rig} - A_{Rig} \leq 3) \vee (4 \leq D_{Rig} - A_{Rig} \leq 5)$

Simple Temporal Networks: Composition & Intersection

Composition

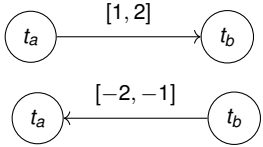


Intersection

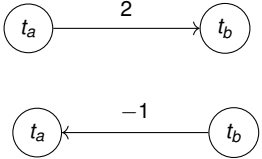


Simple Temporal Networks: Distance Graph

- Equivalent $= \begin{cases} 1 \leq t_b - t_a \leq 2 \\ -2 \leq t_a - t_b \leq -1 \end{cases}$

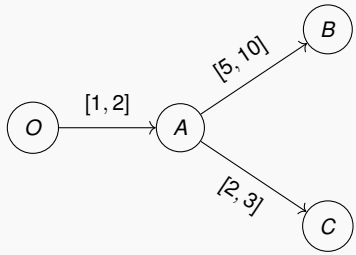


- Distance Graphs: only allow upper bounds on differences
  - $t_b - t_a \leq 2$
  - $t_a - t_b \leq -1$

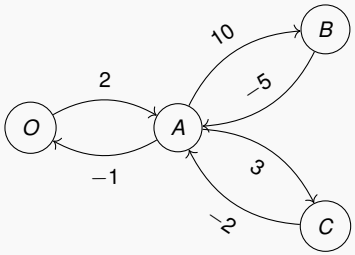


Simple Temporal Networks: Distance Graph (cont.)

Original STN



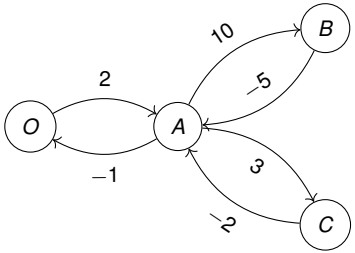
Distance Graph



- Composition: $(A \xrightarrow{x} B) \circ (B \xrightarrow{y} C) = A \xrightarrow{x+y} C$
- Intersection: $(A \xrightarrow{x} B) \cap (A \xrightarrow{y} B) = A \xrightarrow{\min(x,y)} B$

Simple Temporal Network: Consistency

- An STN is consistent if and only if there is no cycle of negative length in the distance graph!
- Length of shortest path from A to B:
  - = max delay from A to B
  - = − min delay from B to A
- Intuition
  - Path is a composition (i.e. the sum) of edges
  - Shortest path is the one with minimum length (intersection = min)



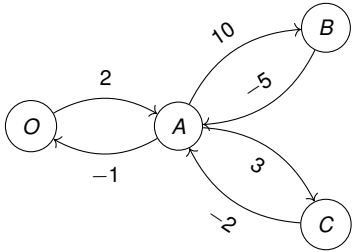
Simple Temporal Network: Consistency (cont.)

Earliest/Latest start time of timepoints

- Bellman-Ford Algorithm:
  - One-to-All shortest path
    - forward graph: max delay
    - backward graph: min delay
  - From O timepoint
  - Complexity: $O(|N| \times |E|)$
  - Incremental: Cesta and Oddi [9]

Possible delay from O:

A	[1,2]
B	[6,12]
C	[3,5]



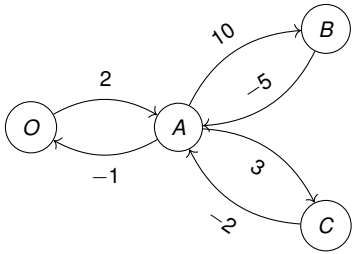
Simple Temporal Network: Consistency (cont.)

Min/Max delays between all pairs of timepoints

- Floyd-Warshall Algorithm
  - All-Pairs shortest path
  - Complexity: $O(|N|^3)$
  - Incremental: Planken et al. [31]

Max delay from-to:

	O	A	B	C
O	0	2	12	5
A	-1	0	10	3
B	-6	-5	0	-2
C	-3	-2	8	0

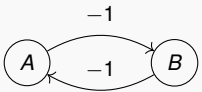


Simple Temporal Network: Consistency (cont.)

- An STN is consistent if and only if there is no cycle of negative length in the distance graph!

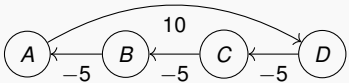
Deadlock

- $(A\{b\}B) \wedge (B\{b\}A)$



Deadline

- D at most 10 after A



- Edges of a negative cycle constitute an unsatisfiable subset of constraints.
  - To make the STN consistent, at least one of those constraints must be relaxed
  - Can be provided by classical algorithms (Bellman-Ford,...)

STN and Beyond

- STN with Uncertainty by Vidal and Fargier [41]
- Disjunctive STN by Tsamardinos and Pollack [39]
- Conditional STN by Combi et al. [15]
- Multi-Agent STN by Casanova et al. [8]
- Time-Dependent STN by Pralet and Verfaillie [32]

Temporal Planning

- Key questions in Temporal Planning
  - What actions?
  - Which parameters?

  } Tied by grounding into operators
  - When?

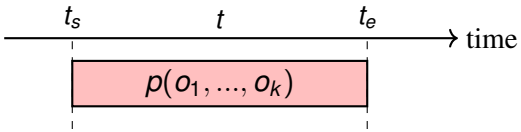
  } Fixed by progression of the search

- Two complementary approaches
  - Time-oriented approaches
  - State-oriented approaches

Temporally Qualified Expressions (TQE)

- TQEs are expressions of the form: $p(o_1, ..., o_k)@[t_b, t_e)$ where:
  - p is a flexible relation in the planning domain,
  - $o_1, ..., o_k$ are object constants or variables, and
  - t_b,t_e are temporal variables such that $t_b < t_e$.

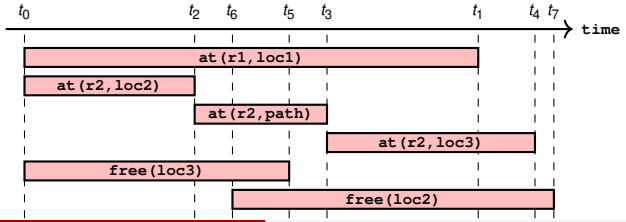
- TQE $p(o_1, .., o_k)@[t_b, t_e)$ asserts that:
  - for every time point t: $t_b \leq t < t_e$ implies that $p(o_1, .., o_k)$ holds
  - $[t_b, t_e)$ is semi-open to avoid inconsistencies



Temporal Database

- A temporal database is a pair $\Phi = (\mathcal{F}, \mathcal{C})$ where:
  - $\mathcal{F}$ is a finite set of TQEs,
  - $\mathcal{C}$ is a finite set of temporal and object constraints
    - temporal constraints: e.g. time point algebra
    - object constraints: rigid relations
  - $\mathcal{C}$ has to be consistent,
    - there exist possible values for the variables that meet all the constraints.

- Example:
  - robot `r1` is at location `loc1`, robot `r2` moves from location `loc2` to location `loc3`



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

Inference over TQEs

- A set $\mathcal{F}$ of TQEs supports a (single) TQE $e = p(v_1, ..., v_k)@[t_b, t_e)$ iff:
  - there is a TQE $p(o_1, ..., o_k)@[t_1, t_2)$ in $\mathcal{F}$ and
  - there is a substitution σ such that: $\sigma(p(v_1, ..., v_k)) = p(o_1, ..., o_k)$.

- An enabling condition for e in $\mathcal{F}$ is the conjunction of the following constraints:
  - $t_1 \leq t_b$, $t_e \leq t_2$ and
  - the variable binding constraints in σ.

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

- Note: if $p(o_1, ..., o_k)$ holds during $[t_1, t_2)$ it must also hold over any sub-interval

Inference over Sets of TQEs

- A set $\mathcal{F}$ of TQEs supports a set $\mathcal{E}$ of TQEs 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 TQEs $\mathcal{E}$ in $\mathcal{F}$ is denoted $\Theta(\mathcal{E}/\mathcal{F})$.

Inference over Temporal Databases

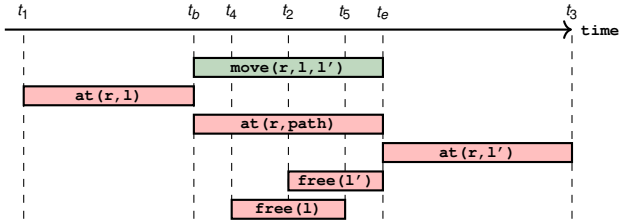
- A temporal database $\Phi = (\mathcal{F}, \mathcal{C})$ supports a set $\mathcal{E}$ of TQEs iff:
  - $\mathcal{F}$ supports $\mathcal{E}$ and
  - there is an enabling condition $c \in \Theta(\mathcal{E}/\mathcal{F})$ that is consistent with $\mathcal{C}$.
- A temporal database $\Phi = (\mathcal{F},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

- A temporal planning operator o is a tuple $(name(o), precond(o), effects(o), constr(o))$, where:
  - $name(o)$ is an expression of the form $a(x_1, ..., x_k, t_b, t_e)$ such that:
    - a is a unique operator symbol,
    - $x_1, ..., x_k$ 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 TQEs, 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 $x = y$, $x \neq y$, or $x \in D$.

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: $t_b < t_4 < t_2$, $adjacent(l, l')$



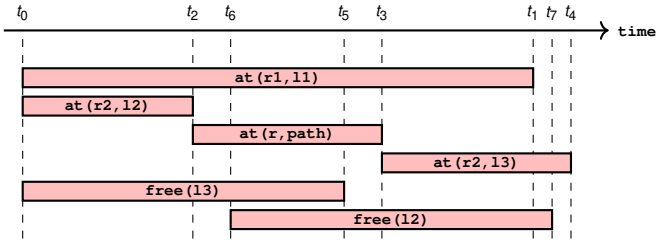
Applicability of Temporal Planning Operators

- 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 $\Theta(precond(o)/\mathcal{F})$ such that:
    - $\mathcal{C} \cup constr(o) \cup c$ is consistent.

- The result of applying an applicable action a (i.e. an instance of an operator o) to Φ is a set of possible temporal databases
  - $\gamma_0(\Phi, a) = \{(\mathcal{F} \cup effects(a), \mathcal{C} \cup constr(a) \cup c) | c \in \Theta(precond(a)/\mathcal{F})\}$

- Note 1: Result is a set not because of non-determinism but different ways in which a can be inserted
- Note 2: applying an action a does not remove anything from the temporal database

Applicability of Temporal Planning Operators: Example



- Operator: $move(r, l, l')@[t_b, t_e)$
  - $at(r1, l1)@[t_0, t_1)$ supports $at(r, l)@[t'_1, t_b)$
  - $free(l2)@[t_6, t_7)$ supports $free(l')@[t'_2, t_e)$
  - enabling condition: $\{r = r1, l = l1, l' = l2, t_0 \leq t'_1, t_b \leq t_1, t_6 \leq t'_2, t_e \leq t_7\}$ consistent
- $\Longrightarrow$ $move(r1, l1, l2)$ is applicable
- Another possibility: $move(r2, l3, l2)$

Domain Axioms

- A domain axiom α is an expression of the form: $cond(\alpha) \rightarrow disj(\alpha)$ where:
  - $cond(\alpha)$ is a set of TQEs and
  - $disj(\alpha)$ is a disjunction of temporal and object constraints.

- A temporal database $\Phi = (\mathcal{F}, \mathcal{C})$ is consistent with α iff:
  - $cond(\alpha)$ is supported by $\mathcal{F}$ and
  - for every enabling condition $c_1 \in \Theta(cond(\alpha)/\mathcal{F})$
    - there is at least one disjunct $c_2 \in disj(\alpha)$ such that $\mathcal{C} \cup c_1 \cup c_2$ is consistent.

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

Temporal Planning Domains and Problems

- A temporal planning domain is a triple $\mathcal{D} = (\mathcal{S}_\Phi, \mathcal{O}, \mathcal{X})$ where:
  - $\mathcal{S}_\Phi$ 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
  - $\mathcal{X}$ is a set of domain axioms.

- A temporal planning problem in $\mathcal{D}$ is a triple $\mathcal{P} = (\mathcal{D}, \Phi_0, \Phi_g)$ where:
  - $\mathcal{D} = (\mathcal{S}_\Phi, \mathcal{O}, \mathcal{X})$ is a temporal planning domain,
  - $\Phi_0 = (\mathcal{F}, \mathcal{C})$ is a database in $\mathcal{S}_\Phi$ that satisfies the axioms in $\mathcal{X}$.
    - 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 $\mathcal{S}_\Phi$ where:
    - $\mathcal{G}$ is a set of TQEs representing the goals of the problem
    - $\mathcal{C}_g$ are object and temporal constraints on variables in $\mathcal{G}$.

Statement of Temporal Planning Problem

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

Temporal Planning: Remark

- Concurrent actions
  - Example: swap locations of two robots
    - only one robot at each location at any time
    - path may hold multiple robots
  - Problem:
    - `move(r1,loc1,loc2)`: not applicable
    - `move(r2,loc2,loc1)`: not applicable
  - Solution: Apply both at the same time: applicable
  - Temporal planning can handle such concurrent actions!
    - It provides a higher espressivity!

Temporal Planning Search Procedure: Search Space

- $\Omega = (\Phi, \mathcal{G}, \mathcal{K}, \pi)$ where
  - $\Phi = (\mathcal{F}, \mathcal{C})$: current temporal database, initially Φ_0
  - $\mathcal{G}$: set of current open goals, initially taken from $\Phi_g = (\mathcal{G}, \mathcal{C}_g)$
  - $\mathcal{K} = \{\mathcal{C}_1, ..., \mathcal{C}_i\}$: 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.

Temporal Planning: Search Procedure

- Intuition
  - Start from a state associated to the empty plan, containing only the information about the initial facts and goals
  - Such partial plan is flawed: it might contains violations of the consistency rules
    - unsupported condition
    - conflicting assignment
    - ...
  - TPS algorithm: refines the partial plan until no flaw remains

```
function TPS(Ω)
    flaws ← Ω.GetFlaws()
    if flaws = ∅ then
        return Ω
    end if
    flaw ← flaws.ChooseOne()
    resolvers ← flaw.GetResolvers()
    if resolvers = ∅ then
        return failure
    end if
    resolver ← resolvers.ChooseOne()
    Ω′ ← Ω.Refine(resolver)
    return TPS(Ω′)
end function
```

Temporal Planning: Flaw Types

- Open Goal - Existing TQE
  - unsupported TQE e in $\mathcal{G}$
  - assumption: exists a TQE in $\mathcal{F}$ that can support e
  - resolver: $\mathcal{K} \leftarrow \mathcal{K} \cup \{\Theta(e/\mathcal{F})\}$
    $\mathcal{G} \leftarrow \mathcal{G} \setminus \{e\}$

- Open Goal - New Action
  - unsupported TQE e in $\mathcal{G}$
  - action a (instance of operator o)
    - has $\mathit{effects}(a)$ that support e and
    - $\mathit{constr}(a)$ are consistent with $\mathcal{C}$
  - resolver: $\pi \leftarrow \pi \cup \{a\}$
    $\mathcal{F} \leftarrow \mathcal{F} \cup \mathit{effects}(a)$
    $\mathcal{C} \leftarrow \mathcal{C} \cup \mathit{constr}(a)$
    $\mathcal{G} \leftarrow (\mathcal{G} \setminus \{e\}) \cup \mathit{precond}(a)$
    $\mathcal{K} \leftarrow \mathcal{K} \cup \{\Theta(\mathit{precond}(a)/\Phi)\}$

Temporal Planning: Flaw Types (cont.)

- Unsatisfied Axiom - Add Conditions
  - axiom α: $\mathit{cond}(\alpha) \rightarrow \mathit{disj}(\alpha)$ and
    - $\mathit{cond}(\alpha)$ is supported by $\mathcal{F}$
    - $\mathit{disj}(\alpha)$ is not supported by $\mathcal{F}$
  - assumption: there are consistency conditions $\Theta(\alpha/\Phi)$ such that $\mathit{disj}(\alpha)$ is supported by $\mathcal{F}$
  - resolver: $\mathcal{K} \leftarrow \mathcal{K} \cup \{\Theta(\alpha/\Phi)\}$
- Threat - Add Constraints
  - consistency condition $\mathcal{C}_i \in \mathcal{K}$ that is not entailed by Φ
  - assumption: $\mathbf{c} \in \mathcal{C}_i$ is consistent with $\mathcal{C}$
  - resolver: $\mathcal{C} \leftarrow \mathcal{C} \cup \mathbf{c}$
    $\mathcal{K} \leftarrow \mathcal{K} \setminus \{\mathbf{C}_i\}$

Temporal Planning: Search Strategies

- TPS generates a search tree which can be explored:
  - In Depth-First, Breadth-First
  - Through heuristic search (A*, weighted A*, greedy-best-first, ...)

- Choices to make:
  - Which flaw to fix?
    - (often) with least resolvers (minimize branching factor)
  - Which partial plan (A*) or resolver (DFS) ?
    - Domain specific
    - Least-commitment
    - Distance to consistency Bit-Monnot [5] and Bit-Monnot et al. [6]

Action-Based Languages with Time

- PDDL 2.1 by Fox and Long [21]
- Action Notation Modeling Language - ANML - by Smith et al. [38]
- Natural Definition Language - NDL - by Rintanen [35]

PDDL-2.1 - Support for Durative Actions (cont.)

```
(define (domain travel)
  (:requirements :strips :typing :durative-actions)
  (:types truck location cargo crane)
  (:predicates (at ?t - truck) ...)

(:durative-action load-truck
  :parameters (?t - truck) (?l - location) (?o - cargo) (?c - crane)
  :duration (and (>= ?duration 5) (<= ?duration 10))
  :condition (and (at start (at ?t ?l))
                  (at start (at ?o ?l))
                  (at start (empty ?c))
                  (over all (at ?t ?l))
                  (at end (holding ?c ?o)))
  :effect (and (at start (holding ?c ?o))
               (at start (not (at ?o ?l)))
               (at end (not (holding ?c ?o)))
               (at end (in ?o ?t))))
```

PDDL-2.1 - Support for Durative Actions

```
(define (domain floor-tile)
 (:requirements :strips :typing :durative-actions)
 (:types robot  tile  color-object)
 (:predicates (robotAt ?r-robot ?x-tile) (up ?x-tile ?y-tile) (down ?x-tile ?y-tile)
              (right ?x-tile ?y-tile) (left ?x-tile ?y-tile) (clear ?x-tile)
              (painted ?x-tile ?c-color) (robot-has ?r-robot ?c-color)
              (availableColor ?c-color) (free-color ?r-robot ))

(:durative-action change-color
   :parameters (?r-robot ?c-color ?c2-color)
   :duration (= ?duration 5)
   :condition (and (at start (robot-has ?r ?c ))
                   (over all (availableColor ?c2)))
   :effect (and (at start (not(robot-has ?r ?c )))
                (at end (robot-has ?r ?c2))))

(:durative-action paintUp
   :parameters (?r-robot ?y-tile ?x-tile ?c-color)
   :duration (= ?duration 2)
   :condition (and (over all (robot-has ?r ?c )) (over all (up ?y ?x))
                   (at start (robotAt ?r ?x)) (at start (clear ?y)))
   :effect (and (at start (not(clear ?y)))
                (at end (painted ?y ?c))))
...)
```

ϵ-separation

- Some languages (like e.g. PDDL 2.1) require "ϵ-separation"
  - When two events interfere[1] they need to be separated by a minimum quantum of time $\epsilon > 0$
    - For instance, the event of preparing the coffee and the event of drinking the coffee, the first shall causally happen before the second!
  - If ϵ is fixed a-priori, the time interpretation can be considered discrete
  - On the other hand, if the planner needs to find an ϵ that works, then time interpretation might be dense

[1] e.g. they cannot be freely reordered

Required Concurrency Cushing et al. [16]

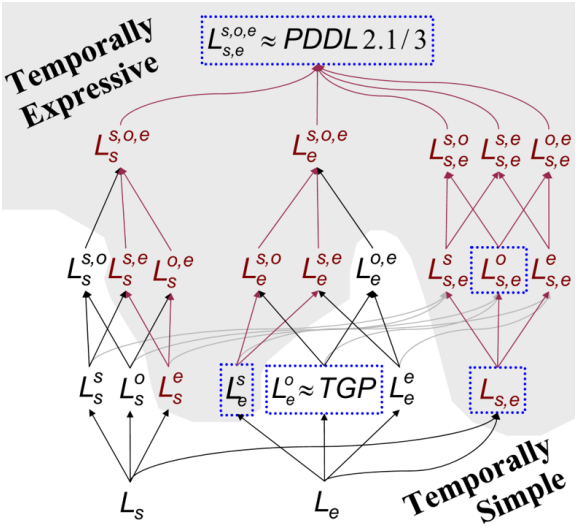
- A plan is said concurrent when there exists a time t at which more than one action is running, otherwise the plan is sequential.
- A solvable planning problem has required concurrency when all solutions are concurrent.

  - Many different modeling languages have been proposed for planning with durative actions (see previous slides)
  - Cushing et al. [16] studied various restrictions of PDDL 2.1.3, characterized by the times at which preconditions and effects may be 'placed' within an action.

  - $L_{\text{effects}}^{\text{preconditions}}$:
    - s "at-start"
    - e "at-end"
    - o "over-all"

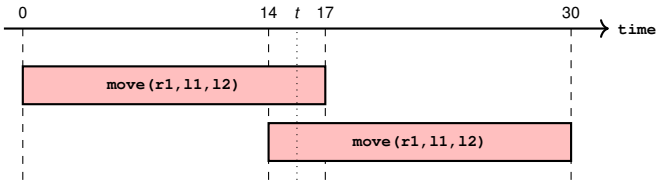
    preconditions $\in \{s, e, o\}$, effects $\in \{s, e\}$
  - Example:
    - $L_{s,e}^{o}$ is a language where every action precondition must hold over all of its execution, and effects may occur at start or at end.

Required Concurrency Cushing et al. [16] (cont.)



Plan with Action Self-Overlapping

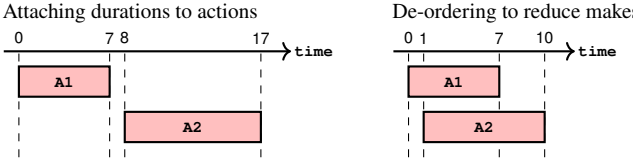
- A plan contains action self-overlapping if there exists a time t at which two or more instances of the same ground action A are executing at the same time



- Remarks
  - Action self-overlapping is rarely useful in practice as shown by Fox and Long [22]
  - Action-based temporal planning without action self-overlapping is PSPACE-Complete as shown by Rintanen [33]

Temporal Planning: Temporally Simple Problems

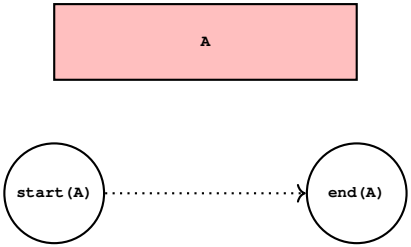
- For temporally-simple problems, one can ignore time and temporal constraints
  - Find a plan π for the classical planning problem having one instantaneous action for each durative action with the same (pre)conditions and effects
  - Post-process the plan, attaching durations to actions and de-ordering the events to reduce the makespan.
- This is what performed in the YAHSP2 planner by Vidal [42]
- Example:
  - Time Abstracted Plan: A1;A2
  - start(A1) supports start(A2)



- **Remark**: Incomplete for problems with required concurrency; no guarantee to get makespan optimal solutions!

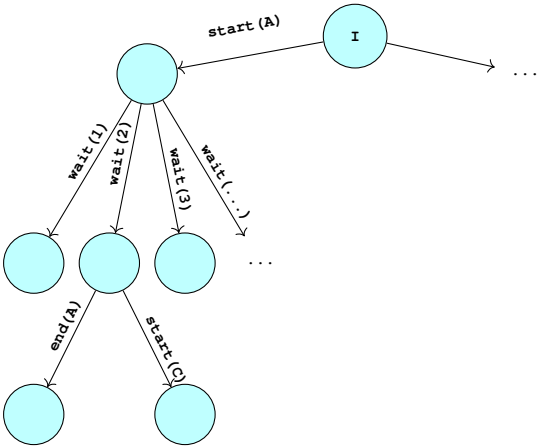
Snap Actions: Intuition

- Split each durative action `A` into two instantaneous actions: `start(A)` and `end(A)`

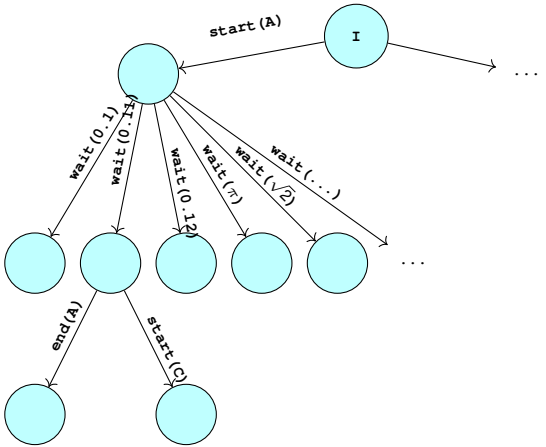


- The search method needs to ensure that for each `start(A)` there exists exactly one `end(A)` and that durative conditions are respected.

Search: Splitting over dense/discrete time



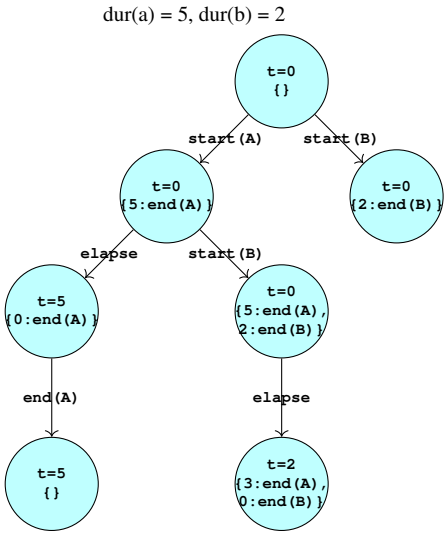
Split over discrete time



Split over dense time

Temporal Planning with Decision Epochs

- Augment the planning state with
  - a timestamp t
  - an agenda of open actions
- At each expansion step, we can either
  - start a new action instance
  - advance time
- Time can advance only to the first (with a closer deadline) event in the agenda:
  - the agenda is then updated to reflect this time pass
- A goal state must satisfy the goal conditions and have an empty agenda



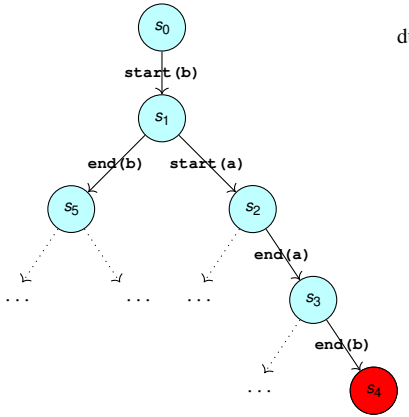
Temporal Planning with Decision Epochs (cont.)

- Many planners adopt decision epochs:
  - TLPlan by Bacchus and Ady [2]
  - SAPA by Do and Kambhampati [19]
  - Temporal Fast Downward (TFD) by Eyerich et al. [20]
  - ...
- Remark: Decision Epoch is incomplete as shown by Cushing et al. [16]
  - Decision Epoch planners cannot start actions at arbitrary times: *only at the start or end of another action in the agenda*

Temporal Planning with symbolic STN

- Augment the planning state with
  - an STN of past and future time-points
  - an agenda of future events
- Time advance is implicit:
  - a path in the search tree represents a total (or a partial) order of the events
- A goal state
  - must satisfy the goal conditions,
  - have an empty agenda
  - have a consistent STN

Temporal Planning with symbolic STN: Example



dur(a) = 5, dur(b) = 2

STN_{s_1} :
- end(b) − start(b) = 2

STN_{s_2} :
- end(b) − start(b) = 2
- end(a) − start(a) = 5
- start(b) ≤ start(a)

STN_{s_3} :
- end(b) − start(b) = 2
- end(a) − start(a) = 5
- start(b) ≤ start(a) ≤ end(a)

STN_{s_4} :
- end(b) − start(b) = 2
- end(a) − start(a) = 5
- start(b) ≤ start(a) ≤ end(a) ≤ end(b)

Temporal Planning with symbolic STN (cont.)

- A family of planners use this approach:
  - Crikey by Coles et al. [14]
  - POPF by Coles et al. [12]
  - COLIN by Coles et al. [13]
  - OPTIC by Benton et al. [4]
  - ...

Temporal Planning with symbolic STN: Heuristics

- Classical planning relaxations by Vidal [42] and Valentini et al. [40]
  - Relax the temporal aspects of the planning domain and use h^+, h^{add}, ...
- Context-enhanced Additive Heuristic for Temporal Planning by Eyerich et al. [20]
  - Transform decision-epoch times into action costs and use h^{cea} by Helmert and Geffner [28]
- Temporal Relaxed Planning Graph by Coles and Coles [11]
  - Compute approximation of the plan length using TRPG generated by computing minimal time of fact and action layers

Remark: state equivalence

- In classical planning we adopted graph search spaces.
- How can we recognize that two plan search states with temporal information are equal?

- Decision-epoch
  - Two states are equal if the have the same fluent assignment, the same agenda and the same timestamp

- Forward heuristic search with STN:
  - We need to check if two STNs are isomorphic
    - We can under-approximate equality as proposed by Coles and Coles [10]

Temporal Planning as Satisfiability (Modulo Theory)

- Encode the bounded planning problem as a formula (as in SATPlan by Kautz and Selman [29]):
  - Any model of the formula encodes a valid plan
  - If the formula is unsatisfiable, either the problem is unsolvable or the bound is insufficient and needs to be increased
  - In order to encode time we need numeric quantities and linear arithmetic: $SMT(\mathcal{LA})$

SMT at a glance

- Satisfiability Modulo Theory (SMT) Barrett et al. [3] is the problem of deciding the satisfiability of a first-order formula expressed in a given (decidable) theory $\mathcal{T}$.
- Example: $\psi = (x > 2) \wedge (x < 8) \wedge ((x < 1) \vee (x > 7))$
  - Is satisfiable in the theory of linear rational arithmetic because $\{x = 7.5\} \models \psi$
  - Is unsatisfiable in the theory of integer arithmetic
- Practical features
  - Many theories are supported
  - Efficient and well-supported implementations
  - Common language to express problems (SMT-LIB) `http://smtlib.cs.uiowa.edu/`
  - Annual solver competitions since 2003 `https://smt-comp.github.io/`

Temporal Planning as SMT

- Two main encodings in the literature:
  - by Shin and Davis discussed in [36] and [37]
  - by Rintanen [34]
  - Each encoding is different and has strengths and weaknesses
- Here we discuss the most important features and the general approach

Temporal Planning as SMT: naïve encoding

- Fluent values are represented at each step by SMT variables: v^i for each $i \in \{1; ...; k\}$
  - A step corresponds to one time point in the timeline
- For every action a and every step i there is a Boolean state variable a^i (snap actions s_a^i, e_a^i) indicating if the action is taken at that step
- The absolute time of each step needs to be represented (e.g. $t^i \in \mathbb{Q}$)
- Action durations are enforced with $\mathcal{LRA}/\mathcal{LIA}$ constraints
  - e.g. $\bigwedge_{i=0}^{k} s_a^i \rightarrow \bigvee_{j=i+1}^{k} (e_a^j \wedge t^j - t^i = dur(a))$
- Snap action conditions are checked at the correct step (e.g. $\bigwedge_{i=0}^{k} a^i \rightarrow [\![pre(a)]\!]^i$)
- The effects of an action are expressed as condition of the (next step) values
  - e.g. $\bigwedge_{i=0}^{k} a^i \rightarrow [\![eff(a)]\!]^{i+1}$
- There are constraints preventing a co-occurrence of two actions that interfere
  - $\bigwedge_{i=0}^{k} a^i \leftrightarrow \neg b^i$
- A "frame-axiom" enforces that values can only change when there is an effect
  - e.g. $\bigwedge_{i=0}^{k-1} v^i \neq v^{i+1} \rightarrow a^i$

Temporal Planning as SMT: approach

function TemporalSATPlan(P)
  for each k = 0, 1, 2, ... **do**
    $\phi_k \leftarrow$ Encode(P,k) → *Create an SMT encoding up to k*
    μ, *Status* $\leftarrow$ SAT(ϕ_k)
    if Status == SAT **then** → *Check whether the encoding is SAT*
      $\pi \leftarrow$ ExtractPlan(μ) → *Extract plan from the SAT assignment*
      return π
    end if
  end for
end function

Temporal Planning as SMT: planners

- MIP-based temporal planning by Dimopoulos and Gerevini in [18], and [17]
- TM-LPSAT by Shin and Davis [37] - planning with processes and continuous change using SMT
- Planning with clock variables in SMT by Rintanen [34]
- LCP by Bit-Monnot [5] - an SMT-based planner for ANML

Computational Complexity

- Temporal planning with discrete time is EXPSPACE-Complete as shown by Rintanen [33] and by Gigante et al. [26]

- Temporal planning with dense time is undecidable as shown by Bozzelli et al. [7]

Tips for updating `planutils`

- `pip2,3 install planutils --upgrade`
  This will update `planutils` to the latest version
- `planutils setup`
  To be sure the new planners are properly added to the search path directory
- `planutils install optic`
  To install the `optic` planner (it might be needed to run the command twice since first time it fails)
- `planutils install tfd`
  To install the `tfd` planner (it might be needed to run the command twice since first time it fails)

The OPTIC Temporal Planner

- OPTIC (by Benton et al. [4]) is a temporal planner for use in problems where plan cost is determined by preferences or time-dependent goal-collection costs.
- To install it (with `planutils`)
  `command> planutils install optic`
- How to run OPTIC:
  - `optic -N domain.pddl problem.pddl`
    Avoids optimizing initial found solution!
    Run weighted A* with W = 1.000, not restarting with goal states
  - `optic -N -W1,1 domain.pddl problem.pddl`
    Run weighted A* with W = 1.000, not restarting with goal states
  - `optic -N -E -W1,1 domain.pddl problem.pddl`
    Run weighted A* with W = 1.000, not restarting with goal states
    Skip EHC: go straight to best-first search
- Remarks
  - OPTIC does not support predicates with the same name of a name of an action (other planners do)!

The Temporal Fast Downward (TFD) planner

- TFD is a temporal planning system based on the Fast Downward. It uses an adaptation of the context-enhanced additive heuristic to guide the search in the temporal state space with decision epochs induced by the given planning problem.
- To install it (with `planutils`)
  `command> planutils install tfd`
- How to run TFD:
  - `tfd domain.pddl problem.pddl`
    No particular flags supported
  - It can be enabled anytime as follows:
    `export ANYTIMEOPTS="a T 10 t 5"`
    `tfd domain.pddl problem.pddl`
    This way anytime is enable (`a`), and it is considered a timeout of 5 seconds to optimize the solution if a plan has been found (`t 5`) and a timeout of 10 seconds to find a solution (`T 10`).

Floor Tiles Painting

A set of robots use different colors to paint patterns in floor tiles. The robots can move around the floor tiles in four directions (up, down, left, and right). Robots paint with one color at a time but can change their spray guns to any available color. However, robots can only paint the tile that is in front (up) and behind (down) them, and once a tile has been painted, no robot can stand on it. Author: Tomás de la Rosa

- Domain:
  { Examples/FT/floor-tile.pddl }
- Problem Instance 1:
  { Examples/FT/instance-1.pddl }
- Problem Instance 2:
  { Examples/FT/instance-10.pddl }
- Problem Instance 3:
  { Examples/FT/instance-20.pddl }

Map Analyzer

A temporal version of the *city car* domain. In this domain, cars have different speed, and the time needed for building or removing roads depend on their length. Authors: Mauro Vallati and Lukas Chrpa

- Domain:
  { Examples/MapA/domain.pddl }
- Problem Instance 1:
  { Examples/MapA/instance-1.pddl }
- Problem Instance 2:
  { Examples/MapA/instance-5.pddl }
- Problem Instance 3:
  { Examples/MapA/instance-10.pddl }

Match Cellar

You have some matches and a number of fuses to mend in a dark room. You can only mend a fuse while a match is lit. Domain inspired by [30]. Author Bharat Ranjan Kavuluri.

- Domain:
  { Examples/MatchC/domain.pddl }
- Problem Instance 1:
  { Examples/MatchC/instance-1.pddl }
- Problem Instance 2:
  { Examples/MatchC/instance-5.pddl }
- Problem Instance 3:
  { Examples/MatchC/instance-10.pddl }

Turn and Open

In this domain, there are a number of robots with two gripper hands and a set of rooms containing balls. The goal is to find a plan to transport balls from a given room to another. There are doors that must be open to move from one room to another. In order to open a given door, the robot must turn the doorknob and open the door at the same time. Author: Sergio Jiménez Celorrio

- Domain:
  { Examples/TO/domain.pddl }
- Problem Instance 1:
  { Examples/TO/instance-1.pddl }
- Problem Instance 2:
  { Examples/TO/instance-5.pddl }
- Problem Instance 3:
  { Examples/TO/instance-10.pddl }

References I

[1] James F. Allen. Maintaining knowledge about temporal intervals. *Commun. ACM*, 26(11):832–843, 1983. doi: 10.1145/182.358434. URL http://doi.acm.org/10.1145/182.358434. 25, 26, 27, 28, 29, 30

[2] Fahiem Bacchus and Michael Ady. Planning with resources and concurrency: A forward chaining approach. In Bernhard Nebel, editor, *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI 2001, Seattle, Washington, USA, August 4-10, 2001*, pages 417–424. Morgan Kaufmann, 2001. 71

[3] Clark W. Barrett, Roberto Sebastiani, Sanjit A. Seshia, and Cesare Tinelli. Satisfiability modulo theories. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, pages 825–885. IOS Press, 2009. doi: 10.3233/978-1-58603-929-5-825. URL https://doi.org/10.3233/978-1-58603-929-5-825. 78

[4] J. Benton, Amanda Jane Coles, and Andrew Coles. Temporal planning with preferences and time-dependent continuous costs. In Lee McCluskey, Brian Charles Williams, José Reinaldo Silva, and Blai Bonet, editors, *Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling, ICAPS 2012, Atibaia, São Paulo, Brazil, June 25-19, 2012*. AAAI, 2012. URL http://www.aaai.org/ocs/index.php/ICAPS/ICAPS12/paper/view/4699. 74, 85

[5] Arthur Bit-Monnot. A constraint-based encoding for domain-independent temporal planning. In John N. Hooker, editor, *Principles and Practice of Constraint Programming - 24th International Conference, CP 2018, Lille, France, August 27-31, 2018, Proceedings*, volume 11008 of *Lecture Notes in Computer Science*, pages 30–46. Springer, 2018. doi: 10.1007/978-3-319-98334-9_3. URL https://doi.org/10.1007/978-3-319-98334-9_3. 59, 82

References II

[6] Arthur Bit-Monnot, Malik Ghallab, Félix Ingrand, and David E. Smith. FAPE: a constraint-based planner for generative and hierarchical temporal planning. *CoRR*, abs/2010.13121, 2020. URL https://arxiv.org/abs/2010.13121. 59

[7] Laura Bozzelli, Alberto Molinari, Angelo Montanari, Adriano Peron, and Gerhard J. Woeginger. Timeline-based planning over dense temporal domains. *Theor. Comput. Sci.*, 813:305–326, 2020. doi: 10.1016/j.tcs.2019.12.030. URL https://doi.org/10.1016/j.tcs.2019.12.030. 83

[8] Guillaume Casanova, Cédric Pralet, and Charles Lesire. Managing dynamic multi-agent simple temporal network. In Gerhard Weiss, Pinar Yolum, Rafael H. Bordini, and Edith Elkind, editors, *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2015, Istanbul, Turkey, May 4-8, 2015*, pages 1171–1179. ACM, 2015. URL http://dl.acm.org/citation.cfm?id=2773299. 39

[9] Amedeo Cesta and Angelo Oddi. Gaining efficiency and flexibility in the simple temporal problem. In Luca Chittaro, Scott D. Goodwin, Howard J. Hamilton, and Angelo Montanari, editors, *Proceedings of the Third International Workshop on Temporal Representation and Reasoning, TIME-96, Key West, Florida, USA, May 19-20, 1996*, pages 45–50. IEEE Computer Society, 1996. doi: 10.1109/TIME.1996.555676. URL https://doi.org/10.1109/TIME.1996.555676. 36

[10] Amanda Jane Coles and Andrew Ian Coles. Have I been here before? state memoization in temporal planning. In Amanda Jane Coles, Andrew Coles, Stefan Edelkamp, Daniele Magazzeni, and Scott Sanner, editors, *Proceedings of the Twenty-Sixth International Conference on Automated Planning and Scheduling, ICAPS 2016, London, UK, June 12-17, 2016*, pages 97–105. AAAI Press, 2016. URL http://www.aaai.org/ocs/index.php/ICAPS/ICAPS16/paper/view/13187. 76

References III

[11] Amanda Jane Coles and Andrew Ian Coles. A temporal relaxed planning graph heuristic for planning with envelopes. In Laura Barbulescu, Jeremy Frank, Mausam, and Stephen F. Smith, editors, *Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling, ICAPS 2017, Pittsburgh, Pennsylvania, USA, June 18-23, 2017*, pages 47–55. AAAI Press, 2017. URL https://aaai.org/ocs/index.php/ICAPS/ICAPS17/paper/view/15722. 75

[12] Amanda Jane Coles, Andrew Coles, Maria Fox, and Derek Long. Forward-chaining partial-order planning. In Ronen I. Brafman, Hector Geffner, Jörg Hoffmann, and Henry A. Kautz, editors, *Proceedings of the 20th International Conference on Automated Planning and Scheduling, ICAPS 2010, Toronto, Ontario, Canada, May 12-16, 2010*, pages 42–49. AAAI, 2010. URL http://www.aaai.org/ocs/index.php/ICAPS/ICAPS10/paper/view/1421. 74

[13] Amanda Jane Coles, Andrew Coles, Maria Fox, and Derek Long. COLIN: planning with continuous linear numeric change. *J. Artif. Intell. Res.*, 44:1–96, 2012. doi: 10.1613/jair.3608. URL https://doi.org/10.1613/jair.3608. 74

[14] Andrew Coles, Maria Fox, Keith Halsey, Derek Long, and Amanda Smith. Managing concurrency in temporal planning using planner-scheduler interaction. *Artif. Intell.*, 173(1):1–44, 2009. doi: 10.1016/j.artint.2008.08.003. URL https://doi.org/10.1016/j.artint.2008.08.003. 74

[15] Carlo Combi, Luke Hunsberger, and Roberto Posenato. An algorithm for checking the dynamic controllability of a conditional simple temporal network with uncertainty. In Joaquim Filipe and Ana L. N. Fred, editors, *ICAART 2013 - Proceedings of the 5th International Conference on Agents and Artificial Intelligence, Volume 2, Barcelona, Spain, 15-18 February, 2013*, pages 144–156. SciTePress, 2013. 39

References IV

[16] William Cushing, Subbarao Kambhampati, Mausam, and Daniel S. Weld. When is temporal planning really temporal? In Manuela M. Veloso, editor, *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, pages 1852–1859, 2007. URL http://ijcai.org/Proceedings/07/Papers/299.pdf. 64, 65, 71

[17] Yannis Dimopoulos and Alfonso Gerevini. Temporal planning through mixed integer programming. In Maria Fox and Alexandra M. Coddington, editors, *AIPS 2002 Workshop on Planning for Temporal Domains, Toulous, France, April 24, 2002*, pages 2–8, 2002. 82

[18] Yannis Dimopoulos and Alfonso Gerevini. Temporal planning through mixed integer programming: A preliminary report. In Pascal Van Hentenryck, editor, *Principles and Practice of Constraint Programming - CP 2002, 8th International Conference, CP 2002, Ithaca, NY, USA, September 9-13, 2002, Proceedings*, volume 2470 of *Lecture Notes in Computer Science*, pages 47–62. Springer, 2002. doi: 10.1007/3-540-46135-3_4. URL https://doi.org/10.1007/3-540-46135-3_4. 82

[19] Minh Binh Do and Subbarao Kambhampati. Sapa: A multi-objective metric temporal planner. *J. Artif. Intell. Res.*, 20:155–194, 2003. doi: 10.1613/jair.1156. URL https://doi.org/10.1613/jair.1156. 71

References V

[20] Patrick Eyerich, Robert Mattmüller, and Gabriele Röger. Using the context-enhanced additive heuristic for temporal and numeric planning. In Erwin Prassler, Johann Marius Zöllner, Rainer Bischoff, Wolfram Burgard, Robert Haschke, Martin Hägele, Gisbert Lawitzky, Bernhard Nebel, Paul-Gerhard Plöger, and Ulrich Reiser, editors, *Towards Service Robots for Everyday Environments - Recent Advances in Designing Service Robots for Complex Tasks in Everyday Environments*, volume 76 of *Springer Tracts in Advanced Robotics*, pages 49–64. Springer, 2012. doi: 10.1007/978-3-642-25116-0_6. URL https://doi.org/10.1007/978-3-642-25116-0_6. 71, 75

[21] Maria Fox and Derek Long. PDDL2.1: an extension to PDDL for expressing temporal planning domains. *J. Artif. Intell. Res.*, 20: 61–124, 2003. doi: 10.1613/jair.1129. URL https://doi.org/10.1613/jair.1129. 60

[22] Maria Fox and Derek Long. A note on concurrency and complexity in temporal planning. 01 2007. 66

[23] Hector Geffner and Blai Bonet. *A Concise Introduction to Models and Methods for Automated Planning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2013. ISBN 9781608459698. doi: 10.2200/S00513ED1V01Y201306AIM022. URL https://doi.org/10.2200/S00513ED1V01Y201306AIM022.

[24] Malik Ghallab, Dana S. Nau, and Paolo Traverso. *Automated planning - theory and practice*. Elsevier, 2004. ISBN 978-1-55860-856-6.

[25] Malik Ghallab, Dana S. Nau, and Paolo Traverso. *Automated Planning and Acting*. Cambridge University Press, 2016. ISBN 978-1-107-03727-4. URL http://www.cambridge.org/de/academic/subjects/computer-science/artificial-intelligence-and-natural-language-processing/automated-planning-and-acting?format=HB.

References VI

[26] Nicola Gigante, Angelo Montanari, Marta Cialdea Mayer, and Andrea Orlandini. Complexity of timeline-based planning. In Laura Barbulescu, Jeremy Frank, Mausam, and Stephen F. Smith, editors, *Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling, ICAPS 2017, Pittsburgh, Pennsylvania, USA, June 18-23, 2017*, pages 116–124. AAAI Press, 2017. URL https://aaai.org/ocs/index.php/ICAPS/ICAPS17/paper/view/15758. 83

[27] Patrik Haslum, Nir Lipovetzky, Daniele Magazzeni, and Christian Muise. *An Introduction to the Planning Domain Definition Language*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2019. doi: 10.2200/S00900ED2V01Y201902AIM042. URL https://doi.org/10.2200/S00900ED2V01Y201902AIM042.

[28] Malte Helmert and Hector Geffner. Unifying the causal graph and additive heuristics. In Jussi Rintanen, Bernhard Nebel, J. Christopher Beck, and Eric A. Hansen, editors, *Proceedings of the Eighteenth International Conference on Automated Planning and Scheduling, ICAPS 2008, Sydney, Australia, September 14-18, 2008*, pages 140–147. AAAI, 2008. URL http://www.aaai.org/Library/ICAPS/2008/icaps08-018.php. 75

[29] Henry A. Kautz and Bart Selman. Planning as satisfiability. In Bernd Neumann, editor, *10th European Conference on Artificial Intelligence, ECAI 92, Vienna, Austria, August 3-7, 1992. Proceedings*, pages 359–363. John Wiley and Sons, 1992. 77

[30] Derek Long and Maria Fox. Exploiting a graphplan framework in temporal planning. In Enrico Giunchiglia, Nicola Muscettola, and Dana S. Nau, editors, *Proceedings of the Thirteenth International Conference on Automated Planning and Scheduling (ICAPS 2003), June 9-13, 2003, Trento, Italy*, pages 52–61. AAAI, 2003. URL http://www.aaai.org/Library/ICAPS/2003/icaps03-006.php. 89

References VII

[31] Léon Planken, Mathijs de Weerdt, and Neil Yorke-Smith. Incrementally solving stns by enforcing partial path consistency. In Ronen I. Brafman, Hector Geffner, Jörg Hoffmann, and Henry A. Kautz, editors, *Proceedings of the 20th International Conference on Automated Planning and Scheduling, ICAPS 2010, Toronto, Ontario, Canada, May 12-16, 2010*, pages 129–136. AAAI, 2010. URL http://www.aaai.org/ocs/index.php/ICAPS/ICAPS10/paper/view/1447. 37

[32] Cédric Pralet and Gérard Verfaillie. Time-dependent simple temporal networks. In Michela Milano, editor, *Principles and Practice of Constraint Programming - 18th International Conference, CP 2012, Québec City, QC, Canada, October 8-12, 2012. Proceedings*, volume 7514 of *Lecture Notes in Computer Science*, pages 608–623. Springer, 2012. doi: 10.1007/978-3-642-33558-7_44. URL https://doi.org/10.1007/978-3-642-33558-7_44. 39

[33] Jussi Rintanen. Complexity of concurrent temporal planning. In Mark S. Boddy, Maria Fox, and Sylvie Thiébaux, editors, *Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling, ICAPS 2007, Providence, Rhode Island, USA, September 22-26, 2007*, pages 280–287. AAAI, 2007. URL http://www.aaai.org/Library/ICAPS/2007/icaps07-036.php. 66, 83

[34] Jussi Rintanen. Discretization of temporal models with application to planning with SMT. In Blai Bonet and Sven Koenig, editors, *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, pages 3349–3355. AAAI Press, 2015. URL http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9428. 79, 82

References VIII

[35] Jussi Rintanen. Models of action concurrency in temporal planning. In Qiang Yang and Michael J. Wooldridge, editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 1659–1665. AAAI Press, 2015. URL http://ijcai.org/Abstract/15/237. 60

[36] Ji-Ae Shin and Ernest Davis. Continuous time in a sat-based planner. In Deborah L. McGuinness and George Ferguson, editors, *Proceedings of the Nineteenth National Conference on Artificial Intelligence, Sixteenth Conference on Innovative Applications of Artificial Intelligence, July 25-29, 2004, San Jose, California, USA*, pages 531–536. AAAI Press / The MIT Press, 2004. URL http://www.aaai.org/Library/AAAI/2004/aaai04-085.php. 79

[37] Ji-Ae Shin and Ernest Davis. Processes and continuous change in a sat-based planner. *Artif. Intell.*, 166(1-2):194–253, 2005. doi: 10.1016/j.artint.2005.04.001. URL https://doi.org/10.1016/j.artint.2005.04.001. 79, 82

[38] David E. Smith, Jeremy Frank, and William Cushing. The ANML Language. 2007. https://ti.arc.nasa.gov/m/profile/de2smith/publications/ICAPS08-ANML.pdf. 60

[39] Ioannis Tsamardinos and Martha E. Pollack. Efficient solution techniques for disjunctive temporal reasoning problems. *Artif. Intell.*, 151(1-2):43–89, 2003. doi: 10.1016/S0004-3702(03)00113-9. URL https://doi.org/10.1016/S0004-3702(03)00113-9. 39

References IX

[40] Alessandro Valentini, Andrea Micheli, and Alessandro Cimatti. Temporal planning with intermediate conditions and effects. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 9975–9982. AAAI Press, 2020. URL https://aaai.org/ojs/index.php/AAAI/article/view/6553. 75

[41] Thierry Vidal and Hélène Fargier. Handling contingency in temporal constraint networks: from consistency to controllabilities. *J. Exp. Theor. Artif. Intell.*, 11(1):23–45, 1999. doi: 10.1080/095281399146607. URL https://doi.org/10.1080/095281399146607. 39

[42] Vincent Vidal. YAHSP2: Keep it Simple, Stupid. In SJ Angel Garcıa-Olaya and CL eds. López, editors, *International Planning Competition*, 2011. 67, 75