ICAPS2020 Summer School: Temporal Planning

Arthur Bit-Monnot[1] and Andrea Micheli[2]
14 October 2020

[1]LAAS-CNRS, Toulouse, France. abitmonnot@laas.fr
[2]Fondazione Bruno Kessler, Trento, Italy. amicheli@fbk.eu

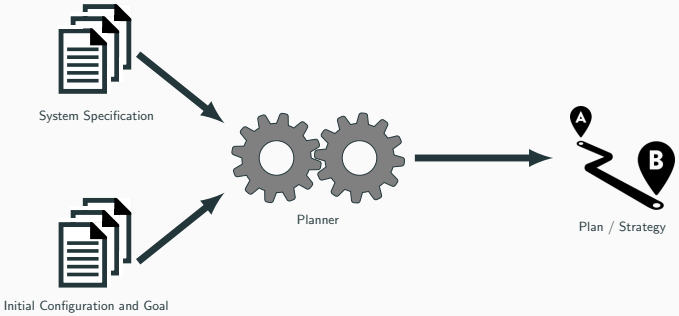
Lecture program

1. Introduction: what is temporal planning?
   - Motivation
   - Applications
   - Complexity
2. Temporal reasoning
   - Temporal Networks Generalities
   - Point Algebra & Allen Algebra
   - Simple Temporal Networks
   - Advanced temporal reasoning frameworks
3. State-oriented techniques
   - Languages & Complexity
   - Time in forward search
   - State-based SAT/SMT/MIP encoding
4. Time-oriented techniques
   - Building blocks for action/timeline representation
   - Search Process
   - Planning as Scheduling
5. Conclusion

Introduction to temporal planning

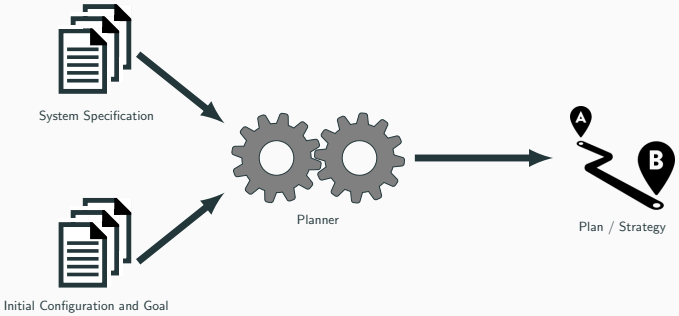
What is AI Planning?

Given a model of a system and a goal to be reached, find a course of actions to drive the system to the goal.



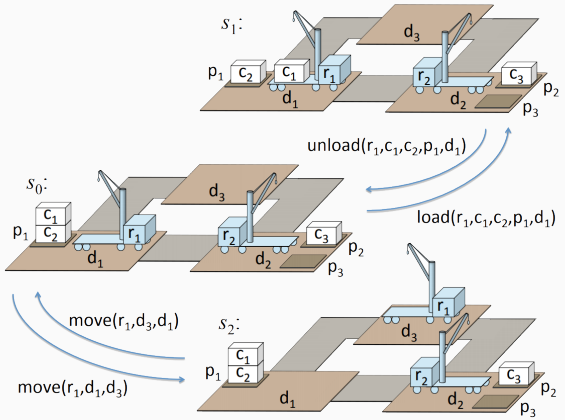
What is AI Planning?

Given a model of a system and a goal to be reached, find a course of actions to drive the system to the goal.



AI Planning is a purely deductive technique: all the information and all the solutions are in the input, the planner reasons on the model and extracts them.

Example



(taken from "Automated Planning and Acting" by M. Ghallab, D. Nau and P. Traverso)

Temporal Planning

- Time and temporal constraints must be considered

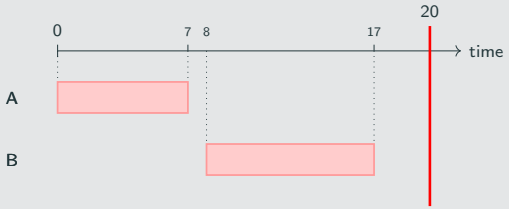
Temporal Planning

- Time and temporal constraints must be considered

Example

Possible Plan:

- A: $move(r1, d1, d3)$ at 0 with duration 7
- B: $load(r2, c3, p2)$ at 8 with duration 9



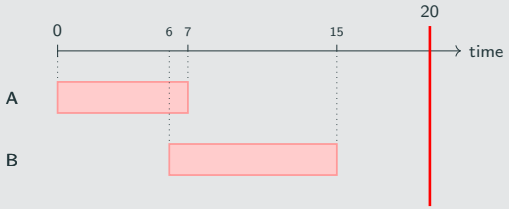
Temporal Planning

- Time and temporal constraints must be considered
- Actions can overlap and interfere

Example

Possible Plan:

- A: $move(r1, d1, d3)$ at 0 with duration 7
- B: $load(r2, c3, p2)$ at 6 with duration 9



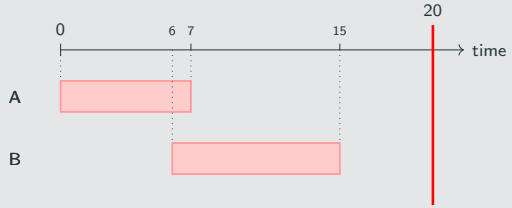
Temporal Planning

- Time and temporal constraints must be considered
- Actions can overlap and interfere
- The key intuition is that we want to decide both **what to do** and **when to do it**: we want to generate a Gantt-chart of activities

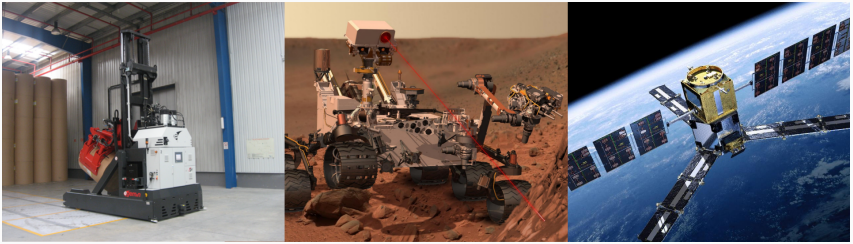
Example

Possible Plan:

- A: $move(r1, d1, d3)$ at 0 with duration 7
- B: $load(r2, c3, p2)$ at 6 with duration 9



Why Temporal Planning?



- Optimal control in known (or partially-known) environments
  (e.g. Factories, Warehouses)
- Mission planning
  (e.g. Exploration rovers, UAV, AUV)
- Safety critical control and recovery
  (e.g. Satellites)

Time interpretation

Time is a first-class citizen and can be interpreted on different domains

- Discrete-time
  - Events happen at discrete steps (e.g. $\mathbb{Z}$)
  - We do not model what happens between two steps
  - A planner has to find a way to schedule events in the discrete steps
  - Q: how many (non-overlapping) actions with duration > 0 can you fit in 10 time-units?
- Dense-time
  - Events can happen in a dense continuum (e.g. $\mathbb{Q}$ or $\mathbb{R}$)
  - Q: how many (non-overlapping) actions with duration > 0 can you fit in 10 time-units?

Time interpretation

Time is a first-class citizen and can be interpreted on different domains

- Discrete-time
  - Events happen at discrete steps (e.g. $\mathbb{Z}$)
  - We do not model what happens between two steps
  - A planner has to find a way to schedule events in the discrete steps
  - Q: how many (non-overlapping) actions with duration > 0 can you fit in 10 time-units?
- Dense-time
  - Events can happen in a dense continuum (e.g. $\mathbb{Q}$ or $\mathbb{R}$)
  - Q: how many (non-overlapping) actions with duration > 0 can you fit in 10 time-units?

Warning

Discrete-time seems simpler (and sometimes is), but consider that linear programming over the rationals is polynomial (ellipsoid algorithm) while over the integers it is NP-complete (integer programming)!

Computational Complexity

Theorem ([Rintanen, 2007, Gigante et al., 2020a, Gigante et al., 2017])

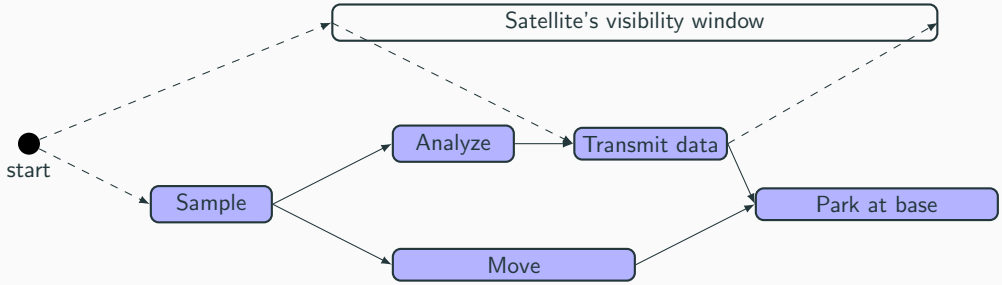
Temporal planning with discrete time is EXPSPACE-Complete.

Theorem ([Gigante et al., 2020a, Bozzelli et al., 2020])

Temporal planning with dense time is undecidable.

Temporal Reasoning

Why Temporal Reasoning?



Answering questions about a (partial?) plan:

- Consistency (no deadlocks, meet deadlines, ...)
- Occurrence times (earliest/latest start times, ...)
- Relative ordering (action A before action B?)
- Dispatchability (strategy to execute the plan)

Intervals and Timepoints

Interval

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

Timepoint

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

Intervals and Timepoints

Interval

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

Timepoint

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

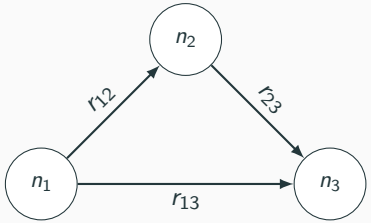
Interval as Timepoints

subject to: $e - s = d$

Temporal Networks

Graphical models where:

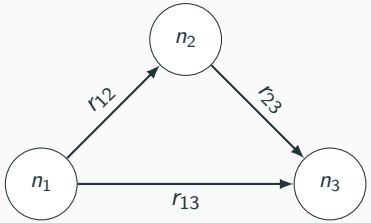
- nodes = **temporal references**
  - timepoints, intervals
- edges = **temporal relations**
  - before, after, ...



Temporal Networks

Graphical models where:

- nodes = **temporal references**
  - timepoints, intervals
- edges = **temporal relations**
  - before, after, ...

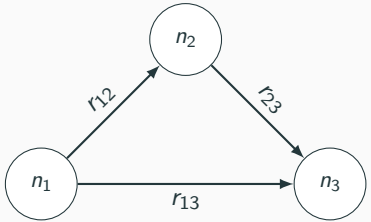


Consistency: there exists a valid instantiation of all references (nodes)

Temporal Networks

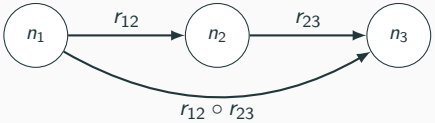
Graphical models where:

- nodes = **temporal references**
  - timepoints, intervals
- edges = **temporal relations**
  - before, after, ...



Consistency: there exists a valid instantiation of all references (nodes)

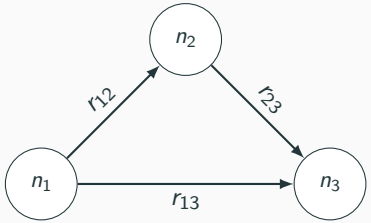
Reasoning rules: Composition



Temporal Networks

Graphical models where:

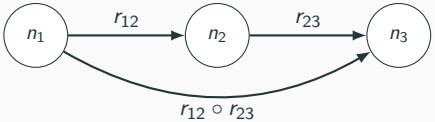
- nodes = **temporal references**
  - timepoints, intervals
- edges = **temporal relations**
  - before, after, ...



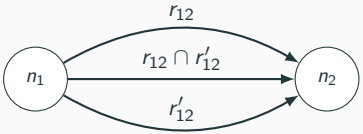
Consistency: there exists a valid instantiation of all references (nodes)

Reasoning rules:

Composition



Intersection



Point Algebra: Qualitative ordering between timepoints

Ordering

Given 2 timepoints a and b, exactly one of the following will be true:

- $a < b$ (a before b)
- $a > b$ (a after b)
- $a = b$ (same occurrence time)

Point Algebra: Qualitative ordering between timepoints

Ordering

Given 2 timepoints a and b, exactly one of the following will be true:

- $a < b$ (a before b)
- $a > b$ (a after b)
- $a = b$ (same occurrence time)

Relation

A relation $a \xrightarrow{R} b$ restricts the set R of possible ordering between a and b.

Relations:

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

Examples: $a \xrightarrow{\{<\}} b$ a must be before b

$a \xrightarrow{\{<,=\}} b$ a must be before or equal to b

Point Algebra: Composition

Given $A \xrightarrow{r_{AB}} B \xrightarrow{r_{BC}} C$

- infer $A \xrightarrow{r_{AC}} C$
- where $r_{AC} = r_{AB} \circ r_{BC}$

Composition table:

$\circ$	$<$	$=$	$>$
$<$	$<$	$<$	$\top$
$=$	$<$	$=$	$>$
$>$	$\top$	$>$	$>$

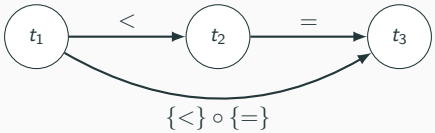
Point Algebra: Composition

Given $A \xrightarrow{r_{AB}} B \xrightarrow{r_{BC}} C$

- infer $A \xrightarrow{r_{AC}} C$
- where $r_{AC} = r_{AB} \circ r_{BC}$

Composition table:

$\circ$	$<$	$=$	$>$
$<$	$<$	$<$	$\top$
$=$	$<$	$=$	$>$
$>$	$\top$	$>$	$>$



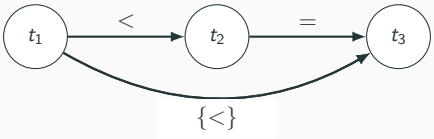
Point Algebra: Composition

Given $A \xrightarrow{r_{AB}} B \xrightarrow{r_{BC}} C$

- infer $A \xrightarrow{r_{AC}} C$
- where $r_{AC} = r_{AB} \circ r_{BC}$

Composition table:

$\circ$	$<$	$=$	$>$
$<$	$<$	$<$	$\top$
$=$	$<$	$=$	$>$
$>$	$\top$	$>$	$>$



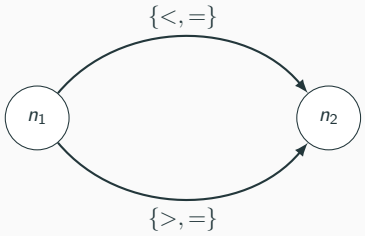
Point Algebra: Intersection

Given $A \xrightarrow{r_1} B$ and $A \xrightarrow{r_2} B$

- infer $A \xrightarrow{r} B$
- where $r = r_1 \cap r_2$

Property: $A \xrightarrow{r_1} B$ and $A \xrightarrow{r_2} B$

- are **weaker** than $A \xrightarrow{r_1 \cap r_2} B$
- become **redundant** and can be removed



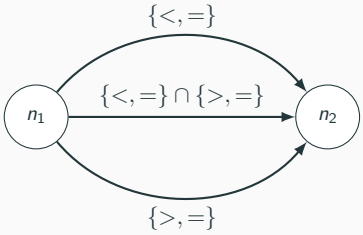
Point Algebra: Intersection

Given $A \xrightarrow{r_1} B$ and $A \xrightarrow{r_2} B$

- infer $A \xrightarrow{r} B$
- where $r = r_1 \cap r_2$

Property: $A \xrightarrow{r_1} B$ and $A \xrightarrow{r_2} B$

- are **weaker** than $A \xrightarrow{r_1 \cap r_2} B$
- become **redundant** and can be removed



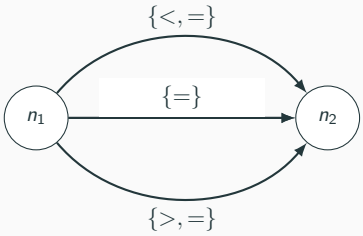
Point Algebra: Intersection

Given $A \xrightarrow{r_1} B$ and $A \xrightarrow{r_2} B$

- infer $A \xrightarrow{r} B$
- where $r = r_1 \cap r_2$

Property: $A \xrightarrow{r_1} B$ and $A \xrightarrow{r_2} B$

- are **weaker** than $A \xrightarrow{r_1 \cap r_2} B$
- become **redundant** and can be removed



Point Algebra: Intersection

Given $A \xrightarrow{r_1} B$ and $A \xrightarrow{r_2} B$

- infer $A \xrightarrow{r} B$
- where $r = r_1 \cap r_2$

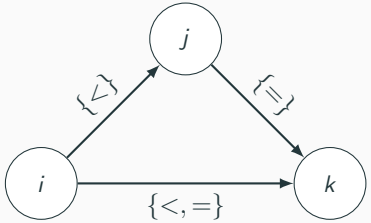
Property: $A \xrightarrow{r_1} B$ and $A \xrightarrow{r_2} B$

- are **weaker** than $A \xrightarrow{r_1 \cap r_2} B$
- become **redundant** and can be removed

Point Algebra: Path Constistency

For each nodes i, j, k:

- $r_{ik} \leftarrow r_{ik} \cap (r_{ij} \circ r_{jk})$
- repeat to quiescence

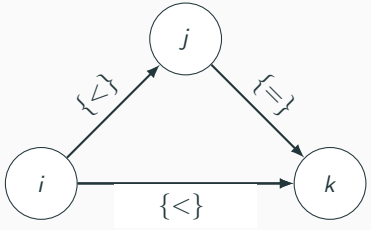


- Infers all pair-wise orderings
- Network is consistent if all pairs of edges have a valid ordering (i.e. no $a \xrightarrow{\{\}} b$)

Point Algebra: Path Constistency

For each nodes i, j, k:

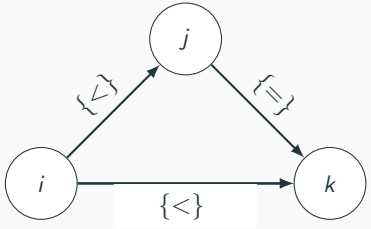
- $r_{ik} \leftarrow r_{ik} \cap (r_{ij} \circ r_{jk})$
- repeat to quiescence



Point Algebra: Path Constistency

For each nodes i, j, k:

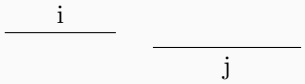
- $r_{ik} \leftarrow r_{ik} \cap (r_{ij} \circ r_{jk})$
- repeat to quiescence



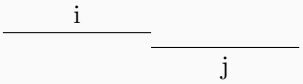
- Infers all pair-wise orderings
- Network is consistent if all pairs of edges have a valid ordering (i.e. no $a \xrightarrow{\{\}} b$)

Allen's Interval Algebra [Allen, 1983]

Qualitative orderings between intervals

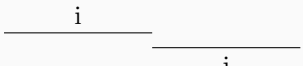
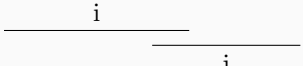


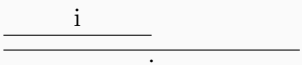
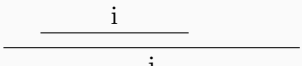
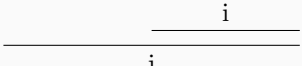
i before j

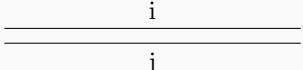


i meet j

Allen's Interval Algebra: Relations

Relation	i ... j
before (b)	
meet (m)	
overlap (o)	

Relation	i ... j
start (s)	
during (d)	
finish (f)	

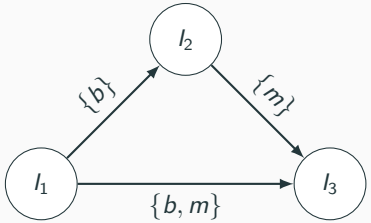
equal (e)	

+ all 6 inverse: i before j $\Leftrightarrow$ j before' i

Allen's Interval Algebra: Intersection & Composition

- Intersection operation (set intersection)
- Composition matrix

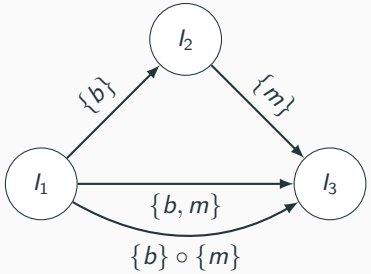
•	b	m	o	s	d	f	b'	d'	f'
b	b	b	b	b	$u \cup v$	$u \cup v$	P	b	b
m	b	b	b	m	v	v	$u' \cup v'$	b	b
o	b	b	u	o	v	v	$u' \cup v'$	$u \cup w'$	u
s	b	b	u	s	d	d	b'	$u \cup w'$	u
d	b	b	$u \cup v$	d	d	d	b'	P	$u \cup v$
f	m	m	v	d	d	f	b'	$u' \cup v'$	x
b'	P	$w \cup u'$	$w \cup u'$	$w \cup u'$	$w \cup u'$	b'	b'	b'	b'
d'	$u \cup w'$	v'	$o' \cup w'$	$o \cup w'$	y	v'	$u' \cup v'$	d'	d'
f'	b	m	o	o	v	x	$u' \cup v'$	d'	f'
s'	$u \cup w'$	$o \cup w'$	$o \cup w'$	$\{s, e, s'\}$	$o' \cup w$	o'	b'	d'	d'



Allen's Interval Algebra: Intersection & Composition

- Intersection operation (set intersection)
- Composition matrix

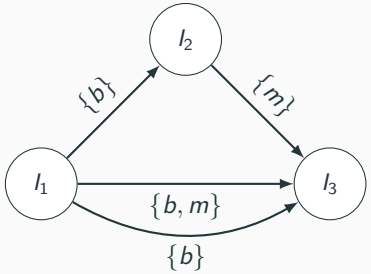
•	b	m	o	s	d	f	b'	d'	f'
b	b	b	b	b	$u \cup v$	$u \cup v$	P	b	b
m	b	b	b	m	v	v	$u' \cup v'$	b	b
o	b	b	u	o	v	v	$u' \cup v'$	$u \cup w'$	u
s	b	b	u	s	d	d	b'	$u \cup w'$	u
d	b	b	$u \cup v$	d	d	d	b'	P	$u \cup v$
f	m	m	v	d	d	f	b'	$u' \cup v'$	x
b'	P	$w \cup u'$	$w \cup u'$	$w \cup u'$	$w \cup u'$	b'	b'	b'	b'
d'	$u \cup w'$	v'	$o' \cup w'$	$o \cup w'$	y	v'	$u' \cup v'$	d'	d'
f'	b	m	o	o	v	x	$u' \cup v'$	d'	f'
s'	$u \cup w'$	$o \cup w'$	$o \cup w'$	$\{s, e, s'\}$	$o' \cup w$	o'	b'	d'	d'



Allen's Interval Algebra: Intersection & Composition

- Intersection operation (set intersection)
- Composition matrix

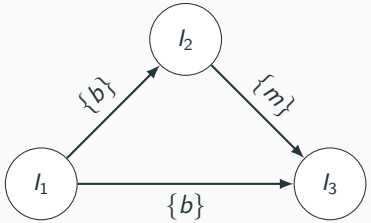
•	b	m	o	s	d	f	b'	d'	f'
b	b	b	b	b	$u \cup v$	$u \cup v$	P	b	b
m	b	b	b	m	v	v	$u' \cup v'$	b	b
o	b	b	u	o	v	v	$u' \cup v'$	$u \cup w'$	u
s	b	b	u	s	d	d	b'	$u \cup w'$	u
d	b	b	$u \cup v$	d	d	d	b'	P	$u \cup v$
f	m	m	v	d	d	f	b'	$u' \cup v'$	x
b'	P	$w \cup u'$	$w \cup u'$	$w \cup u'$	$w \cup u'$	b'	b'	b'	b'
d'	$u \cup w'$	v'	$o' \cup w'$	$o \cup w'$	y	v'	$u' \cup v'$	d'	d'
f'	b	m	o	o	v	x	$u' \cup v'$	d'	f'
s'	$u \cup w'$	$o \cup w'$	$o \cup w'$	$\{s, e, s'\}$	$o' \cup w$	o'	b'	d'	d'



Allen's Interval Algebra: Intersection & Composition

- Intersection operation (set intersection)
- Composition matrix

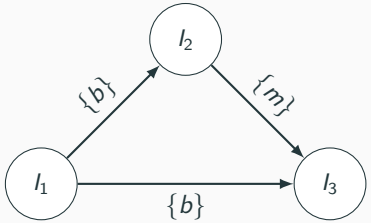
•	b	m	o	s	d	f	b'	d'	f'
b	b	b	b	b	$u \cup v$	$u \cup v$	P	b	b
m	b	b	b	m	v	v	$u' \cup v'$	b	b
o	b	b	u	o	v	v	$u' \cup v'$	$u \cup w'$	u
s	b	b	u	s	d	d	b'	$u \cup w'$	u
d	b	b	$u \cup v$	d	d	d	b'	P	$u \cup v$
f	m	m	v	d	d	f	b'	$u' \cup v'$	x
b'	P	$w \cup u'$	$w \cup u'$	$w \cup u'$	$w \cup u'$	b'	b'	b'	b'
d'	$u \cup w'$	v'	$o' \cup w'$	$o \cup w'$	y	v'	$u' \cup v'$	d'	d'
f'	b	m	o	o	v	x	$u' \cup v'$	d'	f'
s'	$u \cup w'$	$o \cup w'$	$o \cup w'$	$\{s, e, s'\}$	$o' \cup w$	o'	b'	d'	d'



Allen's Interval Algebra: Intersection & Composition

- Intersection operation (set intersection)
- Composition matrix

•	b	m	o	s	d	f	b'	d'	f'
b	b	b	b	b	$u \cup v$	$u \cup v$	P	b	b
m	b	b	b	m	v	v	$u' \cup v'$	b	b
o	b	b	u	o	v	v	$u' \cup v'$	$u \cup w'$	u
s	b	b	u	s	d	d	b'	$u \cup w'$	u
d	b	b	$u \cup v$	d	d	d	b'	P	$u \cup v$
f	m	m	v	d	d	f	b'	$u' \cup v'$	x
b'	P	$w \cup u'$	$w \cup u'$	$w \cup u'$	$w \cup u'$	b'	b'	b'	b'
d'	$u \cup w'$	v'	$o' \cup w'$	$o \cup w'$	y	v'	$u' \cup v'$	d'	d'
f'	b	m	o	o	v	x	$u' \cup v'$	d'	f'
s'	$u \cup w'$	$o \cup w'$	$o \cup w'$	$\{s, e, s'\}$	$o' \cup w$	o'	b'	d'	d'



Can be expressed into *Point Algebra* by relations on start/end timepoints

Simple Temporal Network (STN)

Quantitative ordering between timepoints

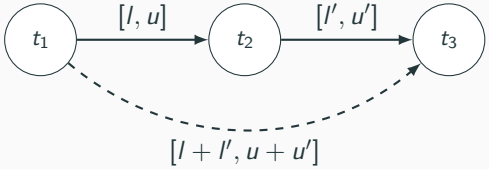
Allowed to bound the delay between two timepoints:

$-10 \leq t_b - t_a \leq 10$	t_a and t_b should not occur more than 10s appart	$a \xrightarrow{[-10,10]} b$
$5 \leq t_b - t_a \leq 7$	t_b must occur between 5 and 7 seconds after t_a	$a \xrightarrow{[5,7]} b$

Also known as *difference logic* in automated reasoning community

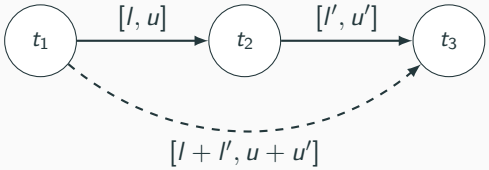
STN: Composition & Intersection

Composition

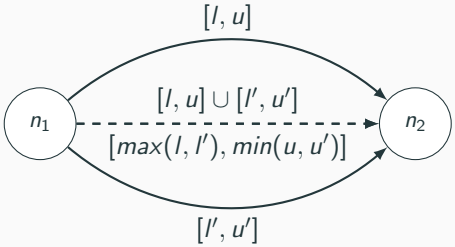


STN: Composition & Intersection

Composition

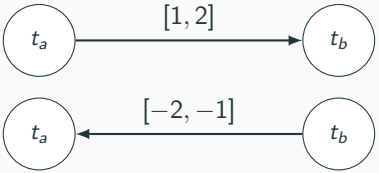


Intersection



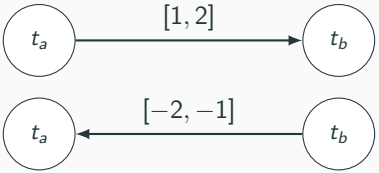
STN: Distance Graph

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



STN: Distance Graph

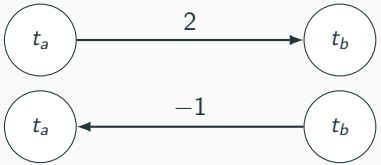
Equivalent $\begin{cases} 1 \leq t_b - t_a \leq 2 \\ -2 \leq t_a - t_b \leq -1 \end{cases}$



Distance graph: only allow upper bounds on difference:

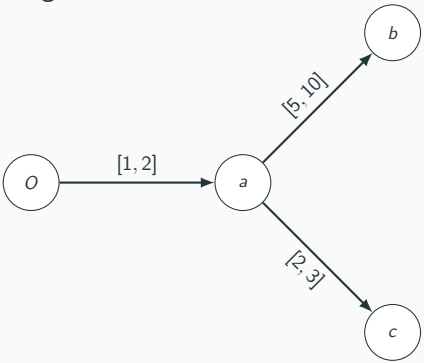
$t_b - t_a \leq 2$

$t_a - t_b \leq -1$

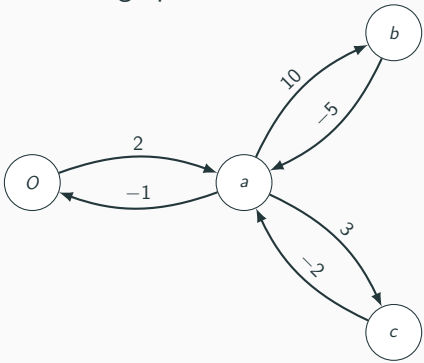


STN: Distance Graph

Original STN:

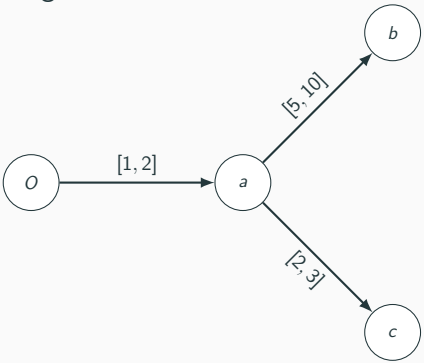


Distance graph:

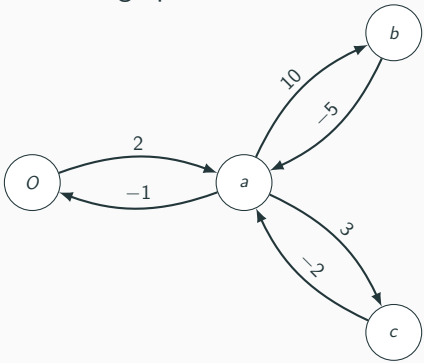


STN: Distance Graph

Original STN:



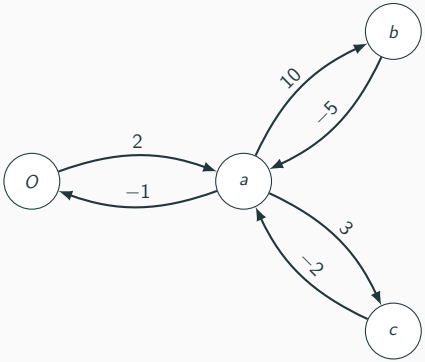
Distance graph:



$$\text{Composition:} \quad (a \xrightarrow{2} b) \circ (b \xrightarrow{10} c) = a \xrightarrow{2+10} c$$

$$\text{Intersection:} \quad (a \xrightarrow{1} b) \circ (a \xrightarrow{5} b) = a \xrightarrow{min(1,5)} b$$

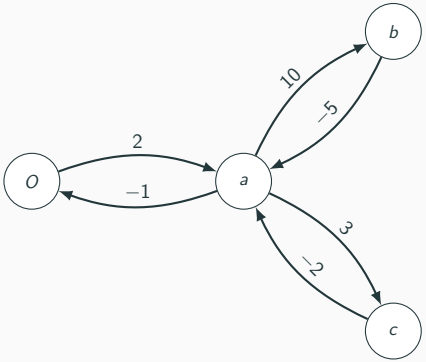
STN: Shortest Paths



Length of shortest path from X to Y:

- = max delay from X to Y
- = −min delay from Y to X

STN: Shortest Paths



Length of shortest path from X to Y:

- = max delay from X to Y
- = −min delay from Y to X

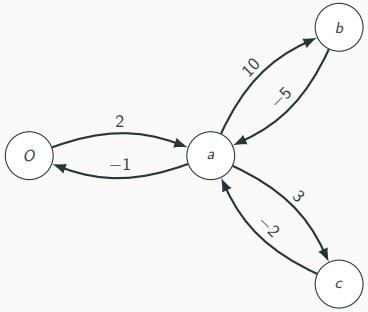
Why?

- Path is a composition (sum) of edges
- Shortest path is the one with minimum length (intersection = min)

STN: Earliest/Latest start time of timepoints

Bellman-Ford Algorithm:

- One-to-All shortest path
  - forward graph: max delay
  - backward graph: min delay
- From ORIGIN timepoint
- Complexity: $O(|N| \times |E|)$
- Incremental: [Cesta and Oddi, 1996]



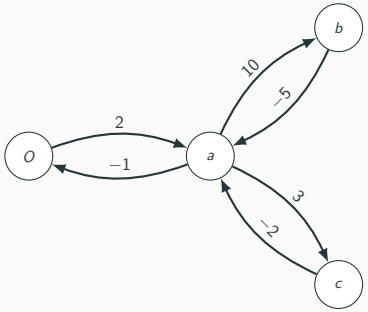
Possible delay from O to ...:

a	$[1, 2]$
b	$[6, 12]$
c	$[3, 5]$

STN: Min/Max delays between all pairs of timepoints

Floyd-Warshall Algorithm:

- All-Pairs shortest path
- Complexity: $O(|N|^3)$
- Incremental: [Planken, 2008]



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

STN Consistency

STN Consistency

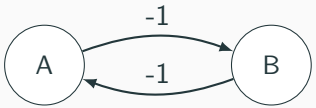
An STN is **consistent** if and only if there is no cycle of negative length in the distance graph.

STN Consistency

STN Consistency

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

Deadlock: A before B & B before A:

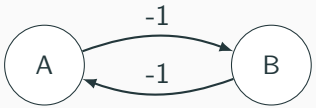


STN Consistency

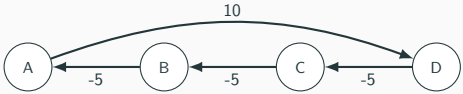
STN Consistency

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

Deadlock: A before B & B before A:



Deadline: D at most 10 after A

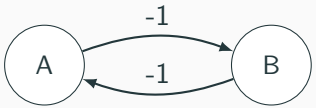


STN Consistency

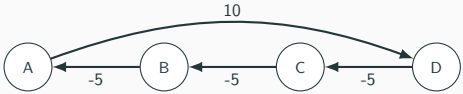
STN Consistency

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

Deadlock: A before B & B before 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, . . .)
- Exploited by some planners [Rankooh and Ghassem-Sani, 2015]

Beyond the Simple Temporal Network

STN extensions:

- STN with Uncertainty [Vidal and Fargier, 1999]
- Disjunctive STN [Tsamardinos and Pollack, 2003]
- Conditional STN [Combi et al., 2013]
- Multi-Agent STN [Casanova et al., 2015]
- Time-Dependent STN [Pralet and Verfaillie, 2012]

Execution of an STN (dispatching): [Tsamardinos and Pollack, 2003]

State-oriented techniques

Action-based languages

PDDL: [Fox and Long, 2003]

```
(define (domain moves)
 (:requirements :strips :typing)
 (:types dock robot)
 (:predicates
    (loc ?r - robot ?d - dock)
    (occupied ?x - dock))

 (:durative-action move
    :parameters(
      ?r - robot
      ?src ?dst - dock)
    :duration (and (>= ?duration 5)
                   (<= ?duration 10))
    :precondition
      (and (at start (not (occupied ?dst)))
           (at start (loc ?r ?src)))
    :effect
      (and (at end (not (occupied ?src)))
           (at start (occupied ?dst))
           (at start (not (loc ?r ?src)))
           (at end (loc ?r ?dst)))))

(define (problem p) (:domain moves)
 (:objects r1 r2 - robot
           d1 d2 d3 - dock)
 (:init (occupied d1) (occupied d2)
        (loc r1 d1) (loc r2 d2))
 (:goal (loc r1 d2)))
```

ANML [Smith et al., 2008]

```
type Dock;

fluent boolean occupied(Dock d);

type Robot with {
  fluent Dock loc;

  action move(Dock dst) {
    duration >= 5 and duration <= 10;
    [start] not occupied(dst);
    [end] loc := dst;
    [end] occupied(loc) := false;
    [end] occupied(dst) := true;
  };
};

instance Robot r1, r2;
instance Dock d1, d2, d3;

[start] occupied(d1) := true;
[start] occupied(d2) := true;
[start] r1.loc := d1;
[start] r2.loc := d2;

[end] r1.loc == d2;
```

Point-based languages: NDL [Rintanen, 2015b] and TPP [Micheli and Scala, 2019].

Epsilon, or non epsilon, that is the question

Some languages (most notably PDDL 2.1 and PDDL+) require "ϵ-separation"

- When two events interfere (e.g. they cannot be freely reordered) they need to be separated by a minimum quantum of time ϵ
  - For example, consider the event of taking the phone and the event of placing a call: one must causally happen before the other
- If ϵ is fixed a-priori, the time interpretation is effectively discrete [Gigante et al., 2020a]
- If the planner needs to find an ϵ that works then time interpretation might be dense

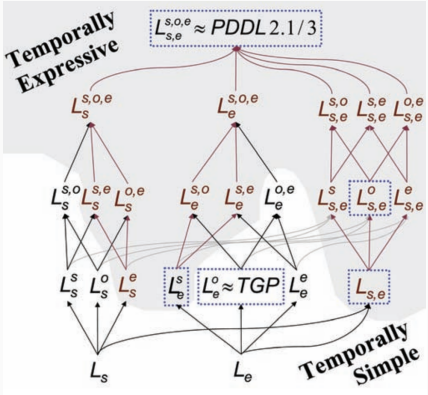
Required concurrency [Cushing et al., 2007]

Definition

A plan is 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.

Matchcellar

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.



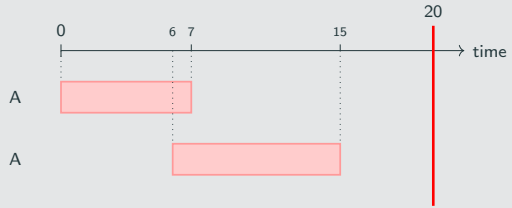
Computational Complexity without Action Self-Overlapping

Theorem ([Rintanen, 2007, Gigante et al., 2020a])

Action-based temporal planning without action self-overlapping is PSPACE-Complete.

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 are running:



Action self-overlapping is rarely useful in practice [Fox and Long, 2007].

Post-process time: the YAHSP2 approach [Vidal, 2011]

For temporally-simple problems, one can just ignore time and temporal constraints

1. Find a plan π for the classical planning problem having one instantaneous action for each durative action with the same (pre)conditions and effects
2. Post-process the plan, attaching durations to actions and de-ordering [Bäckström, 1998] the events to reduce the makespan.

Post-process time: the YAHSP2 approach [Vidal, 2011]

For temporally-simple problems, one can just ignore time and temporal constraints

1. Find a plan π for the classical planning problem having one instantaneous action for each durative action with the same (pre)conditions and effects
2. Post-process the plan, attaching durations to actions and de-ordering [Bäckström, 1998] the events to reduce the makespan.

Example

Time-abstracted plan: A;B

(Suppose start(A) supports start(B))

Post-process time: the YAHSP2 approach [Vidal, 2011]

For temporally-simple problems, one can just ignore time and temporal constraints

1. Find a plan π for the classical planning problem having one instantaneous action for each durative action with the same (pre)conditions and effects
2. Post-process the plan, attaching durations to actions and de-ordering [Bäckström, 1998] the events to reduce the makespan.

Example

Time-abstracted plan: A;B

(Suppose start(A) supports start(B))

Post-process time: the YAHSP2 approach [Vidal, 2011]

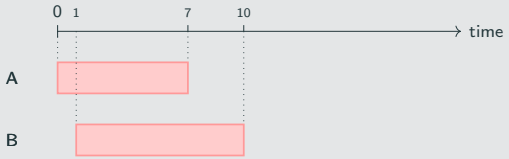
For temporally-simple problems, one can just ignore time and temporal constraints

1. Find a plan π for the classical planning problem having one instantaneous action for each durative action with the same (pre)conditions and effects
2. Post-process the plan, attaching durations to actions and de-ordering [Bäckström, 1998] the events to reduce the makespan.

Example

Time-abstracted plan: A;B

(Suppose start(A) supports start(B))



Post-process time: the YAHSP2 approach [Vidal, 2011]

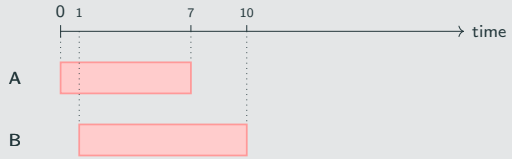
For temporally-simple problems, one can just ignore time and temporal constraints

1. Find a plan π for the classical planning problem having one instantaneous action for each durative action with the same (pre)conditions and effects
2. Post-process the plan, attaching durations to actions and de-ordering [Bäckström, 1998] the events to reduce the makespan.

Example

Time-abstracted plan: A;B

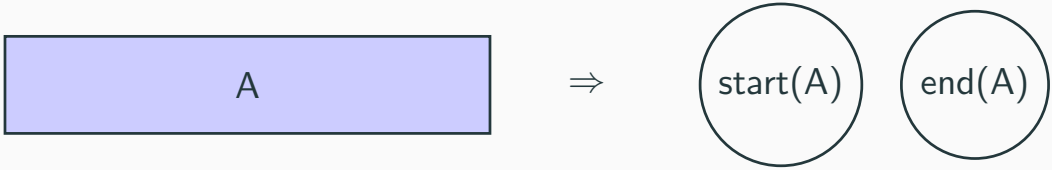
(Suppose start(A) supports start(B))



Drawback: incomplete for problems with required concurrency; non-makespan optimal for temporally-simple

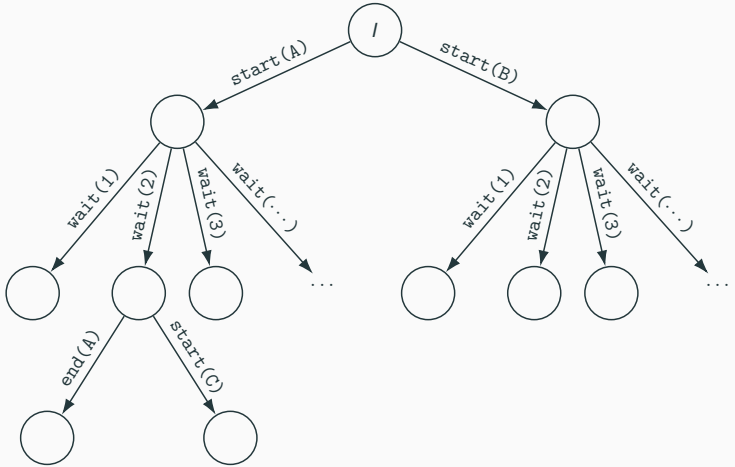
Snap actions

Basic idea: split each durative action A into two instantaneous actions: start(A) and end(A)



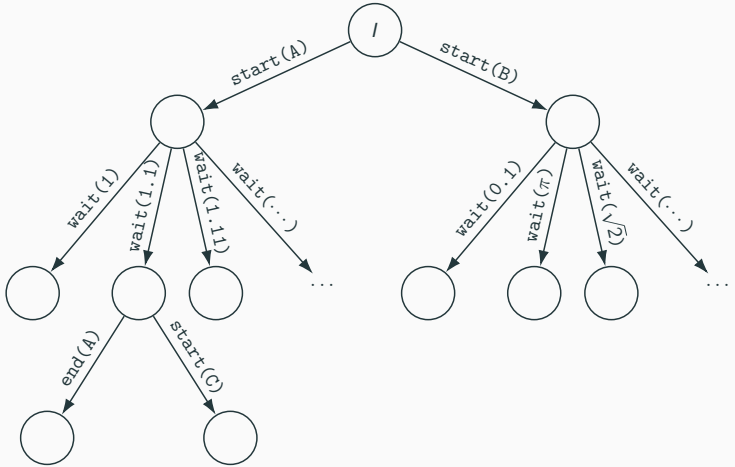
The search method needs to ensure that for each start(A) there exists exactly one end(A) and that durative conditions are respected.

The problem with time and heuristic search



Splitting over (dense) time!

The problem with time and heuristic search



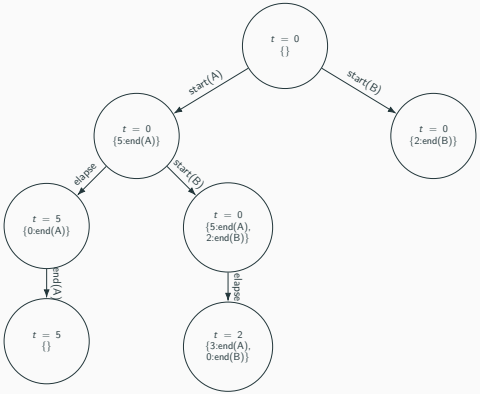
Splitting over (dense) time!

Decision Epoch Planning

Idea

- Augment the planning state with an agenda of open actions and a timestamp t
- At each step, we can either start a new action instance or advance time
- Time can advance only to the first 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

Suppose dur(A) = 5 and dur(B) = 2

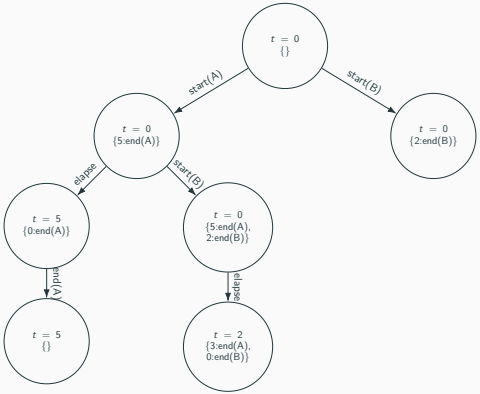


Decision Epoch Planning

Idea

- Augment the planning state with an agenda of open actions and a timestamp t
- At each step, we can either start a new action instance or advance time
- Time can advance only to the first 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

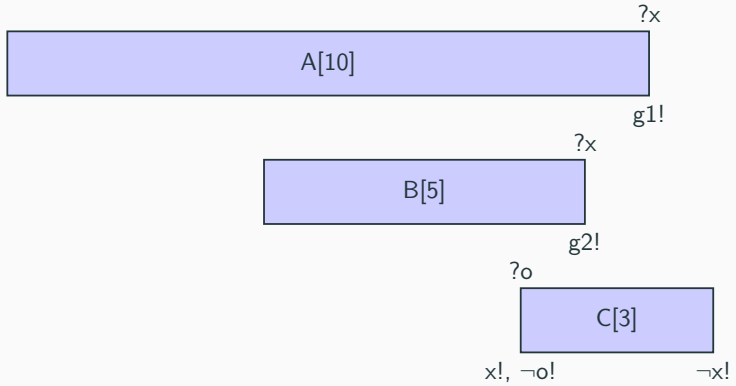
Suppose dur(A) = 5 and dur(B) = 2



Several planners use decision epoch: SAPA [Do and Kambhampati, 2003], TLPlan [Bacchus and Ady, 2001], TFD [Eyerich et al., 2012], ...

Decision Epoch is still incomplete [Cushing et al., 2007]

Decision Epoch planners cannot start actions at arbitrary times: only at the start or end of another action in the agenda



Forward heuristic search with symbolic time

Idea

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

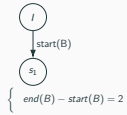
Suppose dur(A) = 5 and dur(B) = 2

Forward heuristic search with symbolic time

Idea

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

Suppose dur(A) = 5 and dur(B) = 2

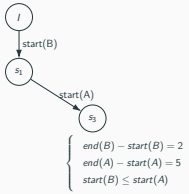


Forward heuristic search with symbolic time

Idea

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

Suppose dur(A) $= 5$ and dur(B) $= 2$

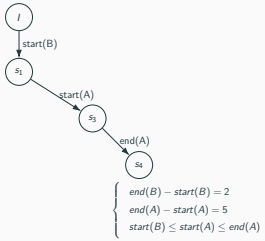


Forward heuristic search with symbolic time

Idea

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

Suppose dur(A) = 5 and dur(B) = 2

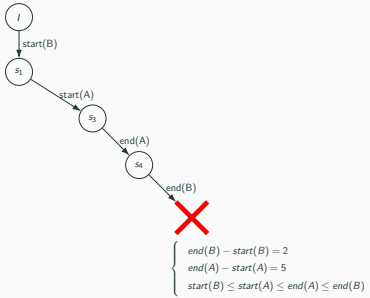


Forward heuristic search with symbolic time

Idea

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

Suppose dur(A) = 5 and dur(B) = 2

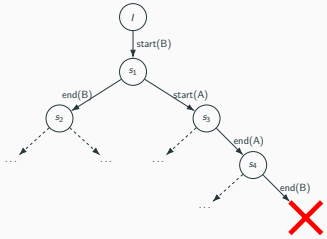


Forward heuristic search with symbolic time

Idea

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

Suppose dur(A) = 5 and dur(B) = 2

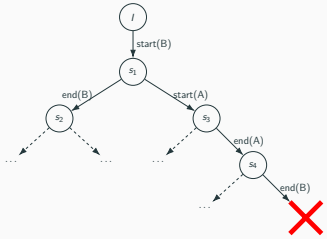


Forward heuristic search with symbolic time

Idea

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

Suppose dur(A) = 5 and dur(B) = 2



A family of planners use this approach: Crikey [Coles et al., 2009], POPF [Coles et al., 2010], COLIN [Coles et al., 2012], OPTIC [Benton et al., 2012].

TAMER [Valentini et al., 2020] also uses this idea for ANML.

Have I been here before?

One important note: in planning we are used to graph search spaces, but how can we recognize that two states with temporal information are equal?

- In Decision-epoch is simple: two states are equal if the have the same fluent assignment, the same agenda and the same timestamp
- In forward heuristic search with symbolic time it is hard: we need to understand if two STNs are isomorphic to decide if two states are equal
  - We can under-approximate equality [Coles and Coles, 2016]

Heuristics for temporal planning

- Classical planning relaxations [Vidal, 2011, Valentini et al., 2020]
  - Relax the temporal aspects of the planning domain and use h^+, h^{add}, ...
- Context-enhanced Additive Heuristic for Temporal Planning [Eyerich et al., 2012]
  - Transform decision-epoch times into action costs and use h^{cea} [Helmert and Geffner, 2008]
- Temporal Relaxed Planning Graph [Coles et al., 2010, Coles and Coles, 2017]
  - Compute approximated the plan length using TRPG generated by computing minimal time of fact and action layers

Temporal Planning as satisfiability

As in SATPlan [Kautz and Selman, 1992]: encode the bounded planning problem as a formula

- 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, SAT is not enough, we need numeric quantities and linear arithmetic: SMT($\mathcal{LA}$)

SMT at-a-glance

Satisfiability Modulo Theory (SMT) [Barrett et al., 2009] is the problem of deciding the satisfiability of a first-order formula expressed in a given (decidable) theory T.

SMT at-a-glance

Satisfiability Modulo Theory (SMT) [Barrett et al., 2009] is the problem of deciding the satisfiability of a first-order formula expressed in a given (decidable) theory T.

Example

$\phi \doteq (x > 2) \wedge (x < 8) \wedge ((x < 1) \vee (x > 7))$

- Is satisfiable in the theory of linear rational arithmetic because $\{x \doteq 7.5\} \models \phi$
- Is unsatisfiable in the theory of integer arithmetic

SMT at-a-glance

Satisfiability Modulo Theory (SMT) [Barrett et al., 2009] is the problem of deciding the satisfiability of a first-order formula expressed in a given (decidable) theory T.

Example

$\phi \doteq (x > 2) \wedge (x < 8) \wedge ((x < 1) \vee (x > 7))$

- Is satisfiable in the theory of linear rational arithmetic because $\{x \doteq 7.5\} \models \phi$
- 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)
- Annual solver competitions since 2003

Encoding temporal planning in SMT [Shin and Davis, 2005, Rintanen, 2015a]

Each encoding is different and has strengths and weaknesses, here is a general gist:

- Fluent values are represented at each step by SMT variables: v^i for each $i \in 1, \cdots, k$
  - A step corresponds to one time point in the timeline

Encoding temporal planning in SMT [Shin and Davis, 2005, Rintanen, 2015a]

Each encoding is different and has strengths and weaknesses, here is a general gist:

- Fluent values are represented at each step by SMT variables: v^i for each $i \in 1, \cdots, k$
  - 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 indicating if the action is taken at that step

Encoding temporal planning in SMT [Shin and Davis, 2005, Rintanen, 2015a]

Each encoding is different and has strengths and weaknesses, here is a general gist:

- Fluent values are represented at each step by SMT variables: v^i for each $i \in 1, \cdots, k$
  - 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 indicating if the action is taken at that step
- The absolute time of each step needs to be represented (e.g. $t_i \in \mathbb{Q}$)

Encoding temporal planning in SMT [Shin and Davis, 2005, Rintanen, 2015a]

Each encoding is different and has strengths and weaknesses, here is a general gist:

- Fluent values are represented at each step by SMT variables: v^i for each $i \in 1, \cdots, k$
  - 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 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 $\mathcal{LRA}/\mathcal{LIA}$
  - e.g. $\bigwedge_{i=0}^{k} s_a^i \rightarrow \bigvee_{j=i+1}^{k}(e_a^j \wedge t^j - t^i = dur(a))$

Encoding temporal planning in SMT [Shin and Davis, 2005, Rintanen, 2015a]

Each encoding is different and has strengths and weaknesses, here is a general gist:

- Fluent values are represented at each step by SMT variables: v^i for each $i \in 1, \cdots, k$
  - 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 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 $\mathcal{LRA}/\mathcal{LIA}$
  - 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. $\bigwedge_{i=0}^{k} a^i \rightarrow [\![pre(a)]\!]^i$)

Encoding temporal planning in SMT [Shin and Davis, 2005, Rintanen, 2015a]

Each encoding is different and has strengths and weaknesses, here is a general gist:

- Fluent values are represented at each step by SMT variables: v^i for each $i \in 1, \cdots, k$
  - 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 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 $\mathcal{LRA}/\mathcal{LIA}$
  - 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. $\bigwedge_{i=0}^{k} a^i \rightarrow [\![eff(a)]\!]^{i+1}$

Encoding temporal planning in SMT [Shin and Davis, 2005, Rintanen, 2015a]

Each encoding is different and has strengths and weaknesses, here is a general gist:

- Fluent values are represented at each step by SMT variables: v^i for each $i \in 1, \cdots, k$
  - 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 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 $\mathcal{LRA}/\mathcal{LIA}$
  - 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
  - e.g. $\bigwedge_{i=0}^{k} a^i \leftrightarrow \neg b^i$

Encoding temporal planning in SMT [Shin and Davis, 2005, Rintanen, 2015a]

Each encoding is different and has strengths and weaknesses, here is a general gist:

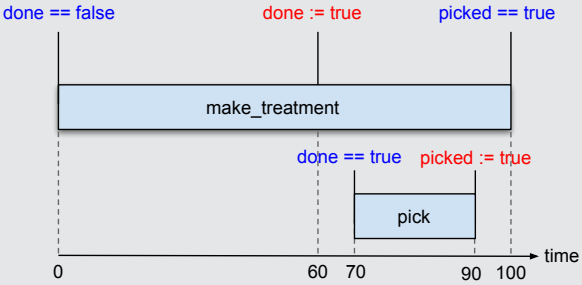
- Fluent values are represented at each step by SMT variables: v^i for each $i \in 1, \cdots, k$
  - A step corresponds to one time point in the timeline
- For every action a and every step i there is a Boolean state variable a^i indicating if the action is taken at that step
- The absolute time of each step needs to be represented (e.g. $t_i \in \mathbb{Q}$)
- Action durations are enforced with $\mathcal{LRA}/\mathcal{LIA}$
  - 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
  - e.g. $\bigwedge_{i=0}^{k} a^i \leftrightarrow \neg b^i$
- A "frame-axiom" enforces that values can only change when there is an effect
  - e.g. $\bigwedge_{i=0}^{k-1} v^i \neq v^{i+1} \rightarrow a^i$

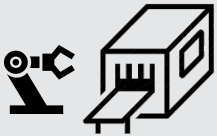
Planners using satisfiability encodings

- TM-LPSAT [Shin and Davis, 2005] planning with processes and continuous change using SMT
- Planning with clock variables in SMT [Rintanen, 2015a]
- LCP [Bit-Monnot, 2018], an SMT-based domain independent planner for ANML
- MIP-based temporal planning [Dimopoulos and Gerevini, 2002]

Intermediate conditions and effects (ICEs)

Actions with conditions and effects defined at arbitrary moments during the action's duration.





Supported by ANML: LCP [Bit-Monnot, 2018] and TAMER [Valentini et al., 2020].

Time-oriented techniques

Reflecting on state-oriented forward search

Key questions in (temporal) planning:

- What actions?
- Which parameters?

} Tied by grounding into operators

- When?

} Fixed by progression search

Reflecting on state-oriented forward search

Key questions in (temporal) planning:

- What actions?
- Which parameters?

} Tied by grounding into operators

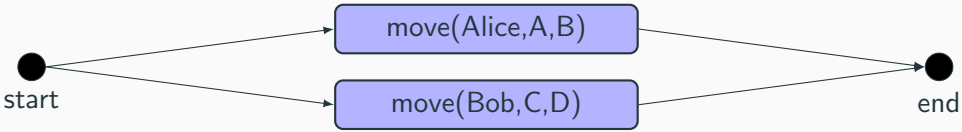
- When?

} Fixed by progression search

In state-oriented forward search, all 3 decisions made at once:

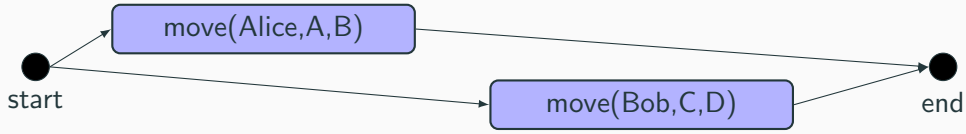
- necessary for extracting a state
- error prone (high branching factor)
- enables extremely helpful search heuristics

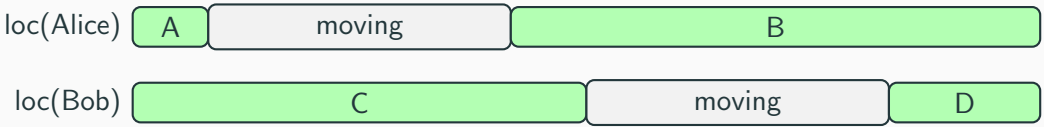
Plans & Partial Orders



The sequence of traversed states changes with the (arbitrary) ordering between actions.

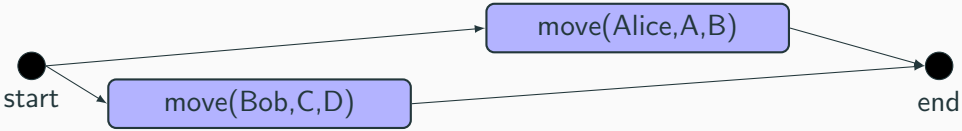
Plans & Partial Orders

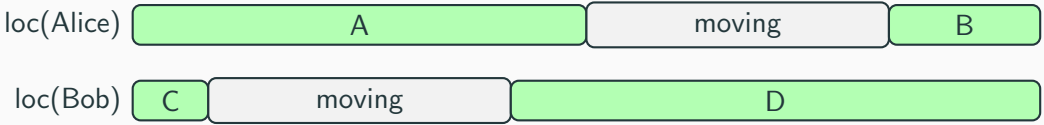




The sequence of traversed states changes with the (arbitrary) ordering between actions.

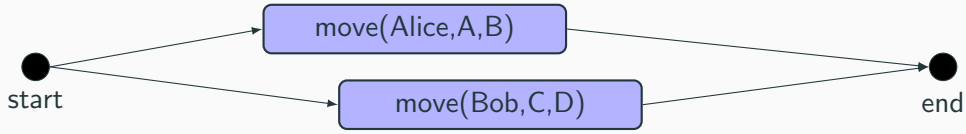
Plans & Partial Orders

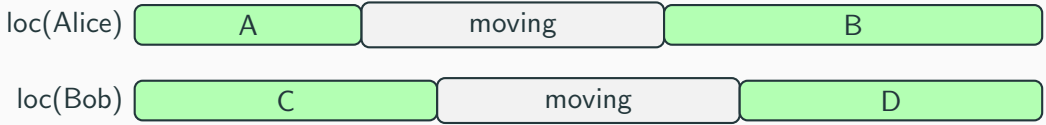




The sequence of traversed states changes with the (arbitrary) ordering between actions.

Plans & Partial Orders



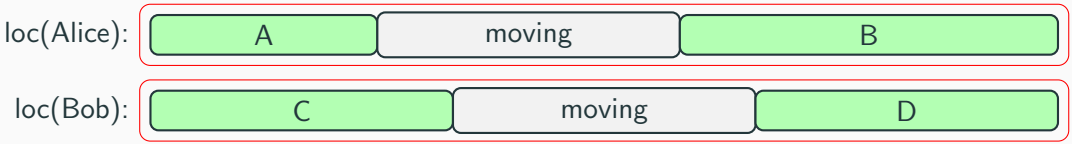


The sequence of traversed states changes with the (arbitrary) ordering between actions.

Timelines

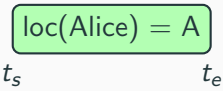
Timeline

A timeline denotes the evolution of a particular state variable through time.



Token

A timeline is filled with tokens, that give the value of the state variable over an interval of time.



Action-based vs Timeline-based models

Action-based model:

When some conditions hold, then you can produce this effect

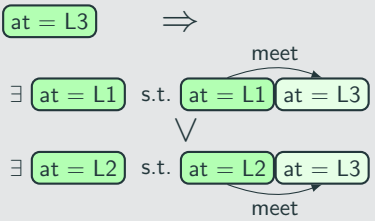
action A:
pre: at = L1
eff: at = L3

action B:
pre: at = L2
eff: at = L3

Planners: IxTeT, FAPE, CPT, LCP

Timeline-based model:

The presence of a token implies additional conditions.



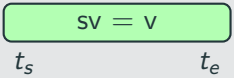
Planners: Europa, ASPEN, CHIMP

Action-based models can be expressed as timeline-based models [Gigante et al., 2016]

Tokens: Effects & Conditions

Effect tokens

Value v of a state variable sv over a temporal interval $[t_s, t_e]$



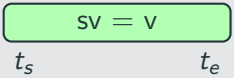
Usage:

- Action effects
- Initial facts
- Timed initial literals (e.g. satellite visible during [120,420])

Tokens: Effects & Conditions

Effect tokens

Value v of a state variable sv over a temporal interval $[t_s, t_e]$

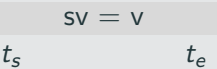


Usage:

- Action effects
- Initial facts
- Timed initial literals (e.g. satellite visible during [120,420])

Condition tokens

State variable sv must have value v over $[t_s, t_e]$ but it has not been established yet.



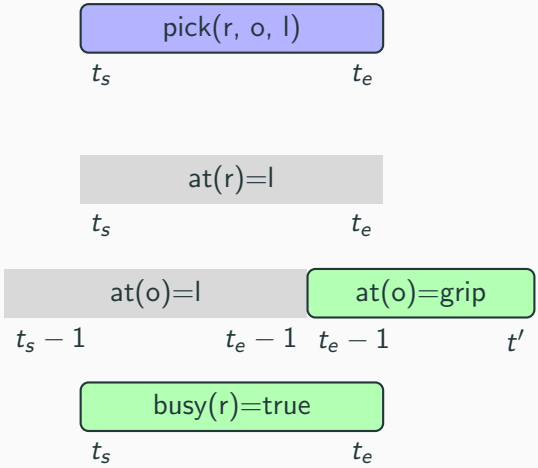
Usage:

- Action's condition
- Goals
- Temporally extended goals

Actions as a set of tokens

pick(r: robot, o: object, l: loc)

start:	t_s
end:	t_e
duration:	4 $(= t_e - t_s)$
conditions:	$[t_s, t_e]$ at(r) = l
	$[t_s - 1, t_e - 1]$ at(o) = l
effects:	$[t_e - 1, t']$ at(o) = grip
	$[t_s, t_e]$ busy(r) = true



t': artificial, unbounded, timepoint to reason about the end of the effect

Timeline consistency: Threat (Conflicting assignment)

Two distinct tokens

should not concurrently impose a value to the same variable

$$sv^1 \neq sv^2 \quad \vee \quad t_e^1 < t_s^2 \quad \vee \quad t_e^2 < t_s^1$$

Timeline consistency: Unsupported conditions

A condition token

$$\underset{t_s^1}{\mathrm{sv}^1} = \underset{t_e^1}{\mathrm{v}^1}$$

should be established by an effect token

$$\underset{t_s^2}{\mathrm{sv}^2} = \underset{t_e^2}{\mathrm{v}^2}$$

such that

$$sv^1 = sv^2 \quad \wedge \quad v^1 = v^2 \quad \wedge \quad t_s^2 \leq t_s^1 \leq t_e^1 \leq t_e^2$$

Search Process

- Start from a **empty plan**, containing only the **tokens representing the initial facts ands goals**
- Such **partial plan** is **flawed**: it might contains violations of the consistency rules
  - unsupported condition
  - conflicting assignment
- PSP algorithm: refines the partial plan until no flaw remains

Partial plan:

- Set of actions
- Set of timelines and associated tokens
- STN to maintain constraints between timepoints

Search Process

- Start from a **empty plan**, containing only the **tokens representing the initial facts ands goals**
- Such **partial plan** is **flawed**: it might contains violations of the consistency rules
  - unsupported condition
  - conflicting assignment
- PSP algorithm: refines the partial plan until no flaw remains

Partial plan:

- Set of actions
- Set of timelines and associated tokens
- STN to maintain constraints between timepoints

PSP(pplan):

1. **if** flaws(pplan) $= \emptyset$

Search Process

- Start from a **empty plan**, containing only the **tokens representing the initial facts ands goals**
- Such **partial plan** is **flawed**: it might contains violations of the consistency rules
  - unsupported condition
  - conflicting assignment
- PSP algorithm: refines the partial plan until no flaw remains

Partial plan:

- Set of actions
- Set of timelines and associated tokens
- STN to maintain constraints between timepoints

PSP(pplan):

1. **if** flaws(pplan) $= \emptyset$
   - return pplan **(solution)**

Search Process

- Start from a **empty plan**, containing only the **tokens representing the initial facts ands goals**
- Such **partial plan** is **flawed**: it might contains violations of the consistency rules
  - unsupported condition
  - conflicting assignment
- PSP algorithm: refines the partial plan until no flaw remains

Partial plan:

- Set of actions
- Set of timelines and associated tokens
- STN to maintain constraints between timepoints

PSP(pplan):

1. **if** flaws(pplan) $= \emptyset$
   - return pplan **(solution)**
2. flaw = **pick arbitrary** in flaws(pplan)

Search Process

- Start from a **empty plan**, containing only the **tokens representing the initial facts ands goals**
- Such **partial plan** is **flawed**: it might contains violations of the consistency rules
  - unsupported condition
  - conflicting assignment
- PSP algorithm: refines the partial plan until no flaw remains

Partial plan:

- Set of actions
- Set of timelines and associated tokens
- STN to maintain constraints between timepoints

PSP(pplan):

1. **if** flaws(pplan) $= \emptyset$
   - return pplan **(solution)**
2. flaw = **pick arbitrary** in flaws(pplan)
3. if resolvers(flaw) $= \emptyset$

Search Process

- Start from a **empty plan**, containing only the **tokens representing the initial facts ands goals**
- Such **partial plan** is **flawed**: it might contains violations of the consistency rules
  - unsupported condition
  - conflicting assignment
- PSP algorithm: refines the partial plan until no flaw remains

Partial plan:

- Set of actions
- Set of timelines and associated tokens
- STN to maintain constraints between timepoints

PSP(pplan):

1. **if** flaws(pplan) $= \emptyset$
   - return pplan **(solution)**
2. flaw = **pick arbitrary** in flaws(pplan)
3. if resolvers(flaw) $= \emptyset$
   - return **failure (unsolvable flaw)**

Search Process

- Start from a **empty plan**, containing only the **tokens representing the initial facts ands goals**
- Such **partial plan** is **flawed**: it might contains violations of the consistency rules
  - unsupported condition
  - conflicting assignment
- PSP algorithm: refines the partial plan until no flaw remains

Partial plan:

- Set of actions
- Set of timelines and associated tokens
- STN to maintain constraints between timepoints

PSP(pplan):

1. **if** flaws(pplan) $= \emptyset$
   - return pplan **(solution)**
2. flaw = **pick arbitrary** in flaws(pplan)
3. if resolvers(flaw) $= \emptyset$
   - return **failure (unsolvable flaw)**
4. **nondeterministically choose** $R \in resolvers(pplan)$

Search Process

- Start from a **empty plan**, containing only the **tokens representing the initial facts ands goals**
- Such **partial plan** is **flawed**: it might contains violations of the consistency rules
  - unsupported condition
  - conflicting assignment
- PSP algorithm: refines the partial plan until no flaw remains

Partial plan:

- Set of actions
- Set of timelines and associated tokens
- STN to maintain constraints between timepoints

PSP(pplan):

1. **if** flaws(pplan) $= \emptyset$
   - return pplan **(solution)**
2. flaw = **pick arbitrary** in flaws(pplan)
3. if resolvers(flaw) $= \emptyset$
   - return **failure (unsolvable flaw)**
4. **nondeterministically choose** $R \in$ *resolvers*(*pplan*)
5. pplan' = **apply-resolver**(R, pplan)

Search Process

- Start from a **empty plan**, containing only the **tokens representing the initial facts ands goals**
- Such **partial plan** is **flawed**: it might contains violations of the consistency rules
  - unsupported condition
  - conflicting assignment
- PSP algorithm: refines the partial plan until no flaw remains

Partial plan:

- Set of actions
- Set of timelines and associated tokens
- STN to maintain constraints between timepoints

PSP(pplan):

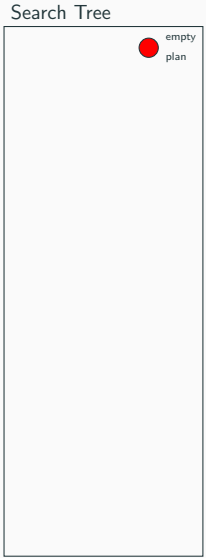
1. **if** flaws(pplan) $= \emptyset$
   - return pplan **(solution)**
2. flaw = **pick arbitrary** in flaws(pplan)
3. if resolvers(flaw) $= \emptyset$
   - return **failure (unsolvable flaw)**
4. **nondeterministically choose** $R \in$ *resolvers*(*pplan*)
5. pplan' = **apply-resolver**(R, pplan)
6. return **PSP(pplan')**

Searching in the space of partial plans/timelines

at(r)=l1

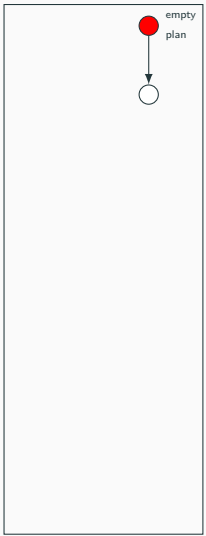
at(o)=l1

at(o)=l2



Searching in the space of partial plans/timelines

Search Tree



at(r)=l1

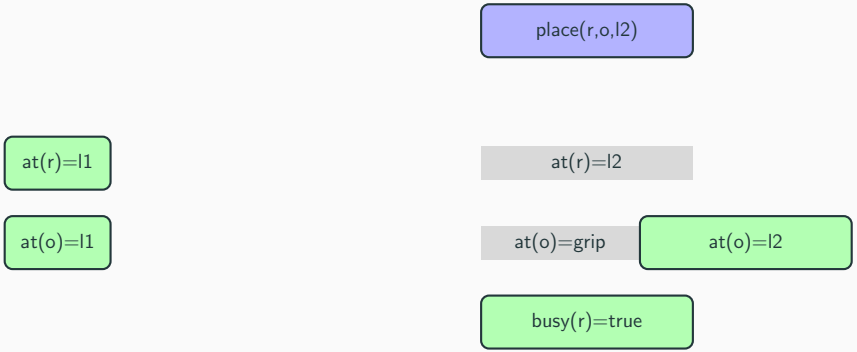
at(o)=l1

at(o)=l2

Flaw unsupported condition at(o)=l2 :

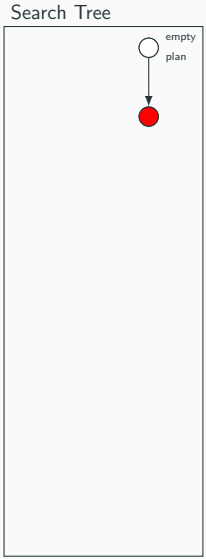
→ Insert action place(r,o,l2) and unify with effect at(o)=l2

Searching in the space of partial plans/timelines



Flaw unsupported condition at(o)=l2 :

→ Insert action place(r,o,l2) and unify with effect at(o)=l2



Searching in the space of partial plans/timelines

place(r,o,l2)

at(r)=l1

at(r)=l2

at(o)=l1

at(o)=grip

at(o)=l2

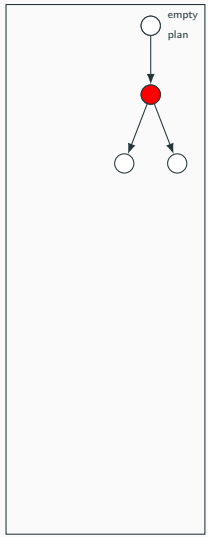
busy(r)=true

Flaw unsupported condition at(o)=grip :

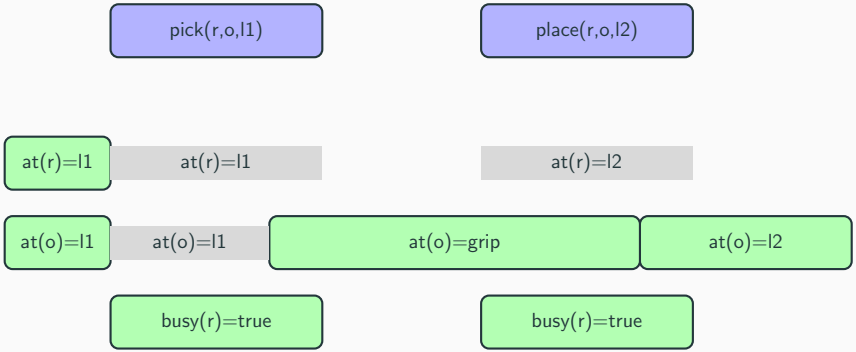
→ Insert action pick(r,o,l1) and unify with effect at(o)=grip

Insert action pick(r,o,l2) and unify with effect at(o)=grip

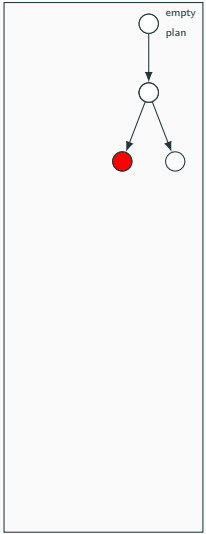
Search Tree



Searching in the space of partial plans/timelines



Search Tree

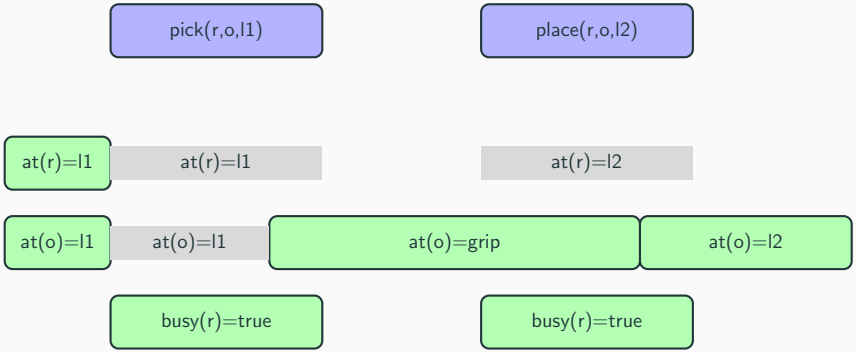


Flaw unsupported condition at(o)=grip :

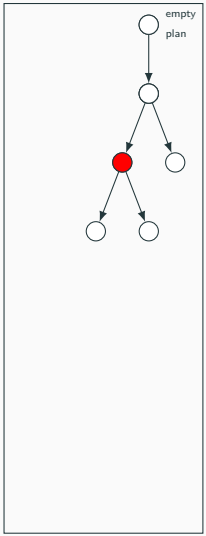
→ Insert action pick(r,o,l1) and unify with effect at(o)=grip

Insert action pick(r,o,l2) and unify with effect at(o)=grip

Searching in the space of partial plans/timelines



Search Tree

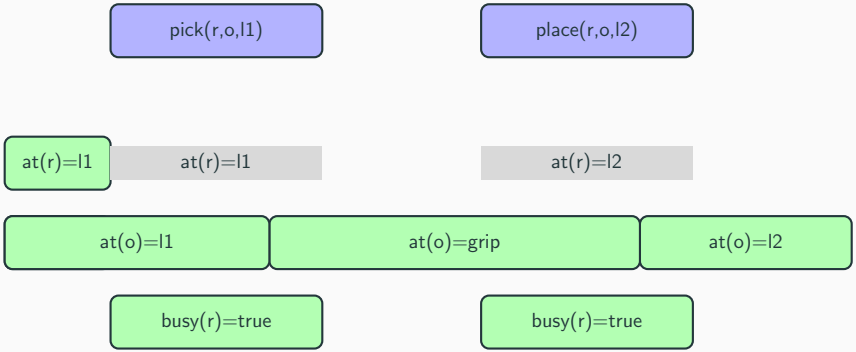


Flaw unsupported condition at(o)=l1 :

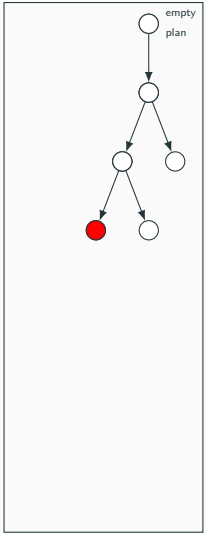
→ Unify with existing at(o)=l1

Insert action place(r,o,l1) and unify with effect at(o)=l1

Searching in the space of partial plans/timelines



Search Tree

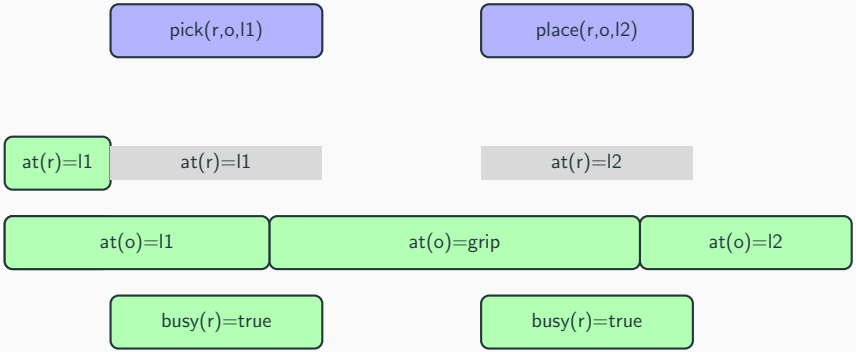


Flaw unsupported condition at(o)=l1 :

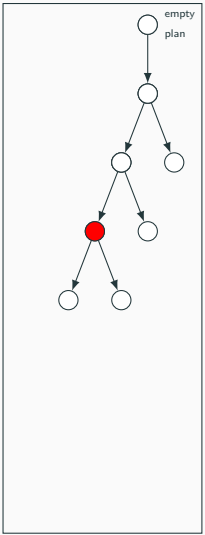
→ Unify with existing at(o)=l1

Insert action place(r,o,l1) and unify with effect at(o)=l1

Searching in the space of partial plans/timelines



Search Tree

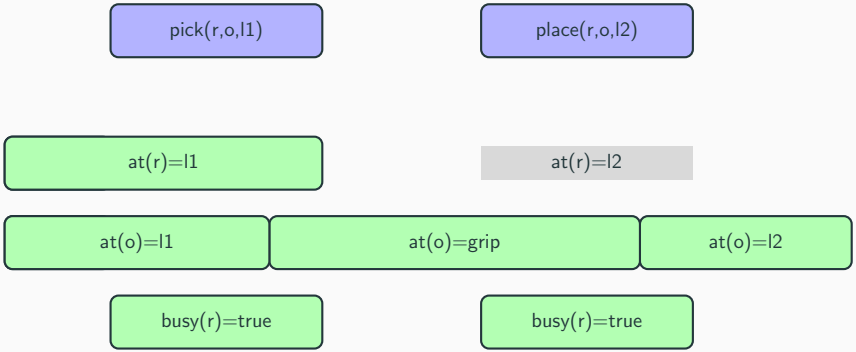


Flaw unsupported condition at(r)=l1 :

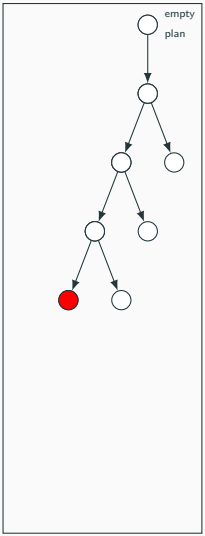
→ Unify with existing at(r)=l1

Insert action move(r,l2,l1) and unify with effect at(r)=l1

Searching in the space of partial plans/timelines



Search Tree

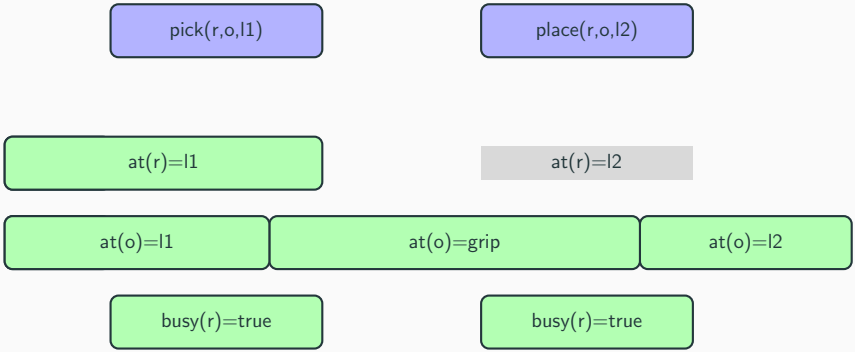


Flaw unsupported condition at(r)=l1 :

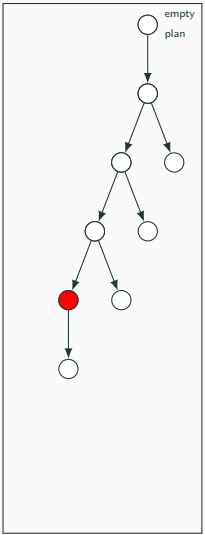
→ Unify with existing at(r)=l1

Insert action move(r,l2,l1) and unify with effect at(r)=l1

Searching in the space of partial plans/timelines



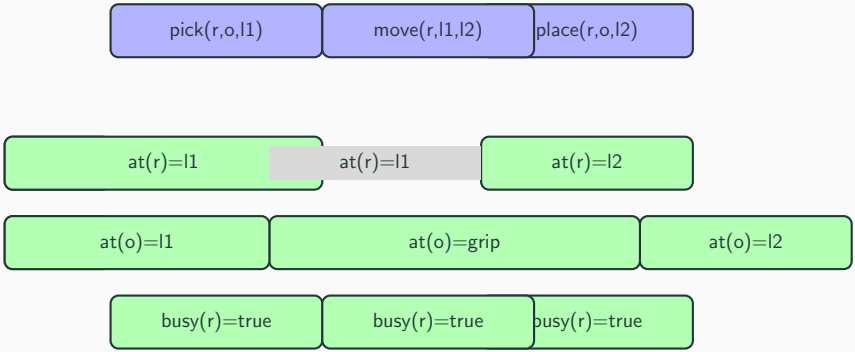
Search Tree



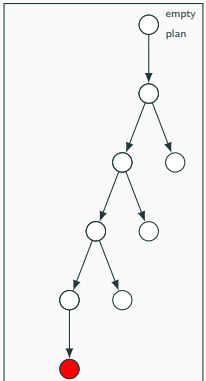
Flaw unsupported condition at(r)=l2 :

→ Insert action move(r,l1,l2) and unify with effect at(r)=l2

Searching in the space of partial plans/timelines



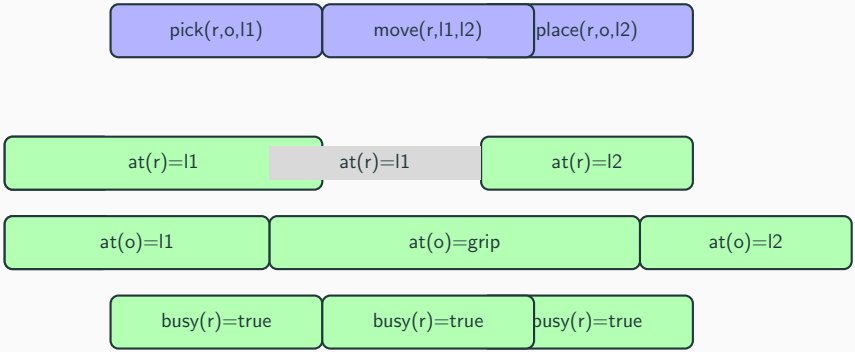
Search Tree



Flaw unsupported condition at(r)=l2 :

→ Insert action move(r,l1,l2) and unify with effect at(r)=l2

Searching in the space of partial plans/timelines

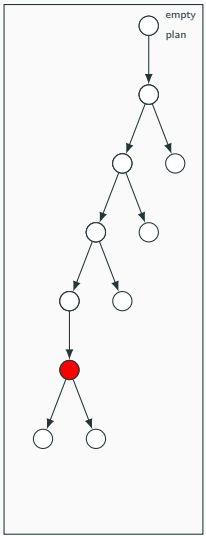


Flaw unsupported condition at(r)=l1 :

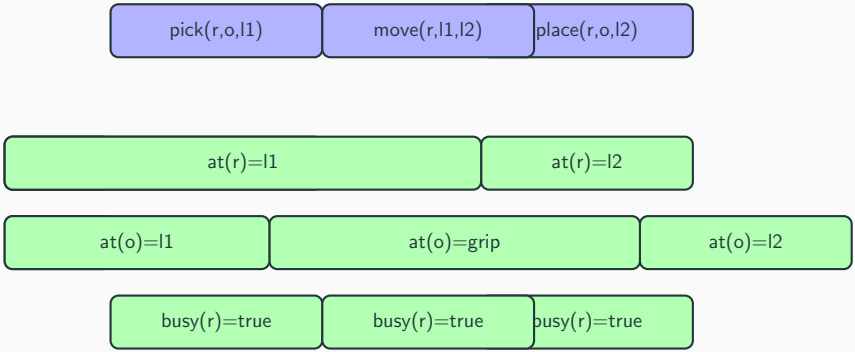
→ Unify with existing at(r)=l1

Insert action move(r,l2,l1) and unify with effect at(r)=l1

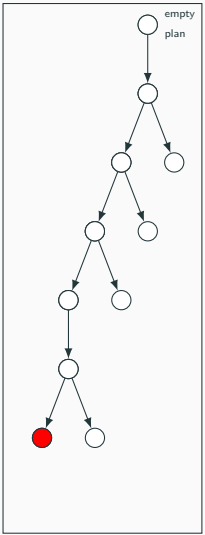
Search Tree



Searching in the space of partial plans/timelines



Search Tree

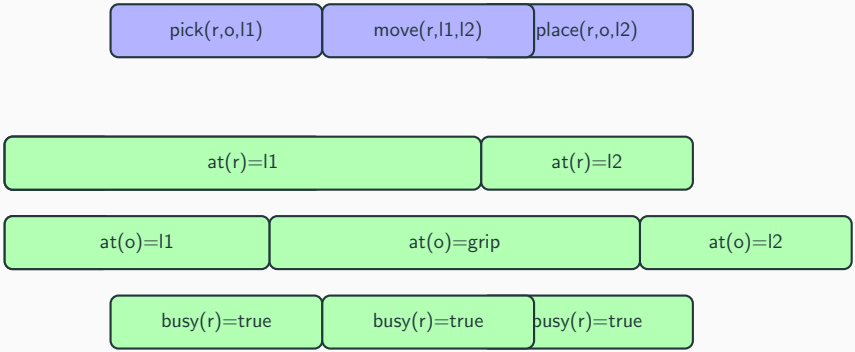


Flaw unsupported condition at(r)=l1 :

→ Unify with existing at(r)=l1

Insert action move(r,l2,l1) and unify with effect at(r)=l1

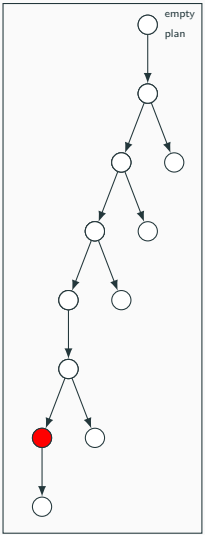
Searching in the space of partial plans/timelines



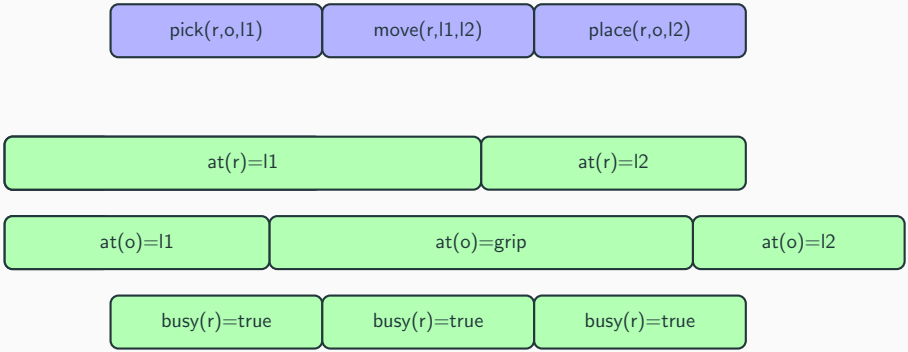
Flaw conflicting busy(r)=true and busy(r)=true :

→ Force ordering

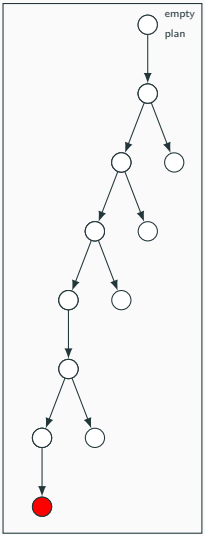
Search Tree



Searching in the space of partial plans/timelines



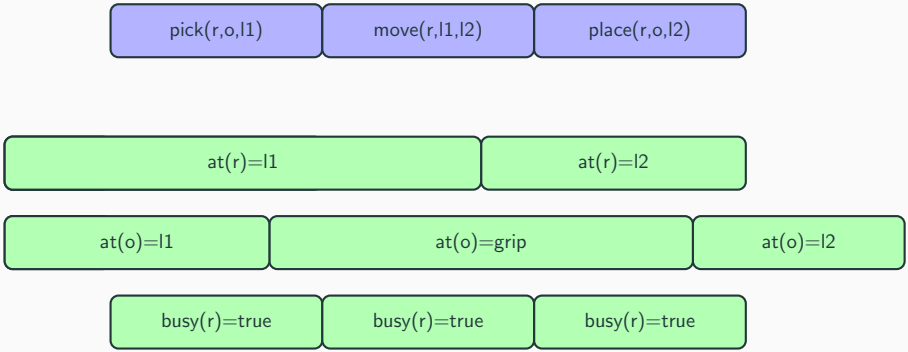
Search Tree



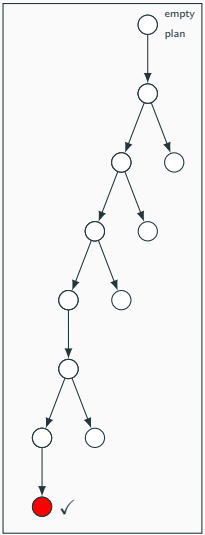
Flaw conflicting busy(r)=true and busy(r)=true :

→ Force ordering

Searching in the space of partial plans/timelines



Search Tree



Search Control

PSP generates a search tree which can be explored:

- In Depth-First (common option among timeline-based planners)
- Through heuristic search ($A^*, A\epsilon$, weighted A^*)

Search Control

PSP generates a search tree which can be explored:

- In Depth-First (common option among timeline-based planners)
- Through heuristic search (A^*, $A\epsilon$, weighted A^*)

Choices to make:

- Which flaw to fix?
  - (often) with least resolvers (minimize branching factor)
- Which partial plan (A^*) or resolver (DFS) ?
  - Domain specific (common in timeline-based)
  - Least-commitment (IxTeT)
  - Distance to consistency [Bernardini and Smith, 2008, Bit-Monnot et al., 2020]

Lifted planning: delay parameter bindings

Lifted planning

- Action parameters are variables whose value are constrained
- The STN is complemented with a binding constraint network that maintain the possible values of parameters.

Planners: FAPE, IxTeT, Europa, LCP

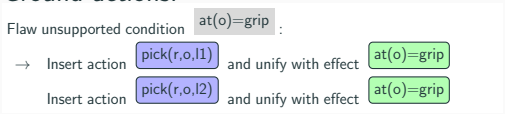
Lifted planning: delay parameter bindings

Lifted planning

- Action parameters are variables whose value are constrained
- The STN is complemented with a binding constraint network that maintain the possible values of parameters.

Planners: FAPE, IxTeT, Europa, LCP

Ground actions:



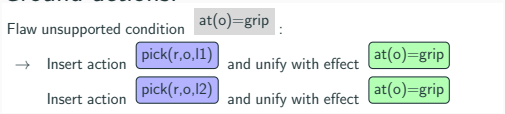
Lifted planning: delay parameter bindings

Lifted planning

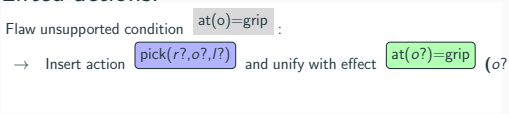
- Action parameters are variables whose value are constrained
- The STN is complemented with a binding constraint network that maintain the possible values of parameters.

Planners: FAPE, IxTeT, Europa, LCP

Ground actions:



Lifted actions:



(single resolver, choice is postponed)

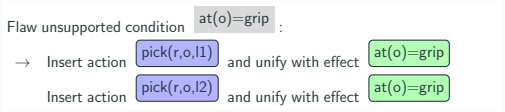
Lifted planning: delay parameter bindings

Lifted planning

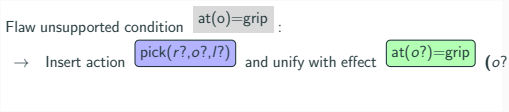
- Action parameters are variables whose value are constrained
- The STN is complemented with a binding constraint network that maintain the possible values of parameters.

Planners: FAPE, IxTeT, Europa, LCP

Ground actions:



Lifted actions:



(single resolver, choice is postponed)

Other CSP-based extensions:

- Resource reasoning [Laborie, 2003]
- Spatial reasoning [Mansouri and Pecora, 2014]

Planning as Scheduling

Temporal Planning { What? (which actions?) / How? (which parameters?) / When? } Scheduling

Planning as Scheduling

$$\text{Temporal Planning} \left\{ \begin{array}{l} \text{What? (which actions?)} \\ \text{How? (which parameters?)} \\ \text{When?} \end{array} \right\} \text{Scheduling}$$

- Mature and efficient tooling in *Operations Research* and *Constraint Programming* communities
- Many (most?) planning problems are naturally handle as scheduling.

A good entry point: CPOptimizer [Laborie et al., 2018]

Conclusions

Take-away message

Summary

- Temporal Planning = Deciding what to do + deciding when to do it
- State-oriented techniques
  - Strong heuristic guidance
  - Risk of temporal over-commitment
  - Harder to support complex temporal constraints
- Time-oriented techniques
  - Harder to provide heuristic guidance
  - Least commitment planning
  - Can support complex temporal constraints

Take-away message

Summary

- Temporal Planning = Deciding what to do + deciding when to do it
- State-oriented techniques
  - Strong heuristic guidance
  - Risk of temporal over-commitment
  - Harder to support complex temporal constraints
- Time-oriented techniques
  - Harder to provide heuristic guidance
  - Least commitment planning
  - Can support complex temporal constraints

Temporal planning is an active area of research

- Continuous change: resources changing continuously
- Temporal uncertainty
- Combination of temporal planning with other extensions of classical planning:
  - Optimal temporal planning
  - Temporal planning with trajectory preferences

Thank you!

References i

Allen, J. (1983).
Maintaining Knowledge About Temporal Intervals.
Communications of the ACM, 26(11):823–832.

Bacchus, F. and Ady, M. (2001).
Planning with resources and concurrency: A forward chaining approach.
In Nebel, B., 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.

Bäckström, C. (1998).
Computational aspects of reordering plans.
J. Artif. Intell. Res., 9:99–137.

Barrett, C. W., Sebastiani, R., Seshia, S. A., and Tinelli, C. (2009).
Satisfiability modulo theories.
In *Handbook of Satisfiability*, pages 825–885. IOS Press.

References ii

Benton, J., Coles, A. J., and Coles, A. (2012).
Temporal planning with preferences and time-dependent continuous costs.
In *ICAPS 2012*.

Bernardini, S. and Smith, D. E. (2008).
Automatically Generated Heuristic Guidance for EUROPA2.
In *ISAIRAS*.

Bit-Monnot, A. (2018).
A constraint-based encoding for domain-independent temporal planning.
In *Principles and Practice of Constraint Programming - 24th International Conference, CP 2018, Lille, France, August 27-31, 2018, Proceedings*.

Bit-Monnot, A., Ghallab, M., Ingrand, F., and Smith, D. E. (2020).
FAPE: A Contraint-based Planner for Generative and Hierarchical Planning.

References iii

Bozzelli, L., Molinari, A., Montanari, A., Peron, A., and Woeginger, G. J. (2020).
Timeline-based planning over dense temporal domains.
Theor. Comput. Sci., 813:305–326.

Casanova, G., Pralet, C., and Lesire, C. (2015).
Managing Dynamic Multi-Agent Simple Temporal Network.
AAMAS, pages 1171–1179.

Cesta, A. and Oddi, A. (1996).
Gaining Efficiency and Flexibility in the Simple Temporal Problem.
In *TIME*, pages 45–50.

Coles, A., Fox, M., Halsey, K., Long, D., and Smith, A. (2009).
Managing concurrency in temporal planning using planner-scheduler interaction.
Artificial Intelligence, 173(1):1–44.


Coles, A. J., Coles, A., Fox, M., and Long, D. (2010).
Forward-chaining partial-order planning.
In *ICAPS*, pages 42–49.


Coles, A. J., Coles, A., Fox, M., and Long, D. (2012).
Colin: Planning with continuous linear numeric change.
Journal of Artificial Intelligence Research, 44:1–96.


Coles, A. J. and Coles, A. I. (2016).
Have I been here before? state memoization in temporal planning.
In Coles, A. J., Coles, A., Edelkamp, S., Magazzeni, D., and Sanner, S., 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.


Coles, A. J. and Coles, A. I. (2017).
A temporal relaxed planning graph heuristic for planning with envelopes.
In Barbulescu, L., Frank, J., Mausam, and Smith, S. F., 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.


Combi, C., Hunsberger, L., and Posenato, R. (2013).
An Algorithm for Checking the Dynamic Controllability of a Conditional Simple Temporal Network with Uncertainty.
In *ICAART*.


Cushing, W., Kambhampati, S., Mausam, and Weld, D. S. (2007).
When is temporal planning really temporal?
In Veloso, M. M., editor, *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, pages 1852–1859.


Della Monica, D., Gigante, N., La Torre, S., and Montanari, A. (2020).
Complexity of qualitative timeline-based planning.
In Muñoz-Velasco, E., Ozaki, A., and Theobald, M., editors, *27th International Symposium on Temporal Representation and Reasoning, TIME 2020, September 23-25, 2020, Bozen-Bolzano, Italy*, volume 178 of *LIPIcs*, pages 16:1–16:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.


Dimopoulos, Y. and Gerevini, A. (2002).
Temporal planning through mixed integer programming.
In Fox, M. and Coddington, A. M., editors, *AIPS 2002 Workshop on Planning for Temporal Domains, Toulous, France, April 24, 2002*, pages 2–8.


Do, M. B. and Kambhampati, S. (2003).
Sapa: A multi-objective metric temporal planner.
JAIR.


Eyerich, P., Mattmüller, R., and Röger, G. (2012).
Using the context-enhanced additive heuristic for temporal and numeric planning.
In *Towards Service Robots for Everyday Environments - Recent Advances in Designing Service Robots for Complex Tasks in Everyday Environments*.


Fox, M. and Long, D. (2003).
PDDL2.1: An extension to PDDL for expressing temporal planning domains.
Journal of artificial intelligence research.


Fox, M. and Long, D. (2007).
A note on concurrency and complexity in temporal planning.
In *PlanSIG 2007*, page 32.


Gigante, N., Micheli, A., Montanari, A., and Scala, E. (2020a).
Decidability and complexity of action-based temporal planning over dense time.
In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 9859–9866. AAAI Press.


Gigante, N., Montanari, A., Cialdea Mayer, M., and Orlandini, A. (2017).
Complexity of Timeline-based Planning.
In *Proc. of the 27th International Conference on Automated Planning and Scheduling*, pages pages 116–124.


Gigante, N., Montanari, A., Mayer, M. C., and Orlandini, A. (2016).
Timelines Are Expressive Enough to Capture Action-Based Temporal Planning.
In *TIME*, pages 100–109.


Gigante, N., Montanari, A., Orlandini, A., Mayer, M. C., and Reynolds, M. (2020b).
On timeline-based games and their complexity.
Theor. Comput. Sci., 815:247–269.


Helmert, M. and Geffner, H. (2008).
Unifying the causal graph and additive heuristics.
In Rintanen, J., Nebel, B., Beck, J. C., and Hansen, E. A., editors, *Proceedings of the Eighteenth International Conference on Automated Planning and Scheduling, ICAPS 2008, Sydney, Australia, September 14-18, 2008*, pages 140–147. AAAI.


Kautz, H. A. and Selman, B. (1992).
Planning as satisfiability.
In *ECAI*, pages 359–363.


Laborie, P. (2003).
Algorithms for propagating resource constraints in AI planning and scheduling: Existing approaches and new results.
Artificial Intelligence, 143(2):151–188.


Laborie, P., Rogerie, J., Shaw, P., and Vilím, P. (2018).
IBM ILOG CP optimizer for scheduling.
Constraints, 23(2):210–250.


Mansouri, M. and Pecora, F. (2014).
More Knowledge on the Table: Planning with Space, Time and Resources for Robots.
In *ICRA*.


Micheli, A. and Scala, E. (2019).
Temporal planning with temporal metric trajectory constraints.
In *AAAI 2019*, pages 7675–7682.


Planken, L. (2008).
New Algorithms for the Simple Temporal Problem.
Master Thesis, Delft University.


Pralet, C. and Verfaillie, G. (2012).
Time-dependent simple temporal networks.
In Milano, M., editor, *Principles and Practice of Constraint Programming*, pages 608–623, Berlin, Heidelberg. Springer Berlin Heidelberg.


Rankooh, M. F. and Ghassem-Sani, G. (2015).
ITSAT: An Efficient SAT-Based Temporal Planner.
JAIR, 53:541–632.


Rintanen, J. (2007).
Complexity of concurrent temporal planning.
In *ICAPS*.


Rintanen, J. (2015a).
Discretization of temporal models with application to planning with SMT.
In *AAAI*, pages 3349–3355.


Rintanen, J. (2015b).
Models of action concurrency in temporal planning.
In *IJCAI*, pages 1659–1665.


Shin, J. and Davis, E. (2005).
Processes and continuous change in a sat-based planner.
Artif. Intell., 166(1-2):194–253.

Smith, D., Frank, J., and Cushing, W. (2008).
The anml language.
In *KEPS 2008*.

Tsamardinos, I. and Pollack, M. E. (2003).
Efficient solution techniques for disjunctive temporal reasoning problems.
Artificial Intelligence, 151(1):43 – 89.

Valentini, A., Micheli, A., and Cimatti, A. (2020).
Temporal planning with intermediate conditions and effects.
In *AAAI*.

Vidal, T. and Fargier, H. (1999).
Handling Contingency in Temporal Constraint Networks: From Consistency to Controllabilities.
Journal of Experimental and Theoretical Artificial Intelligence, 11.


Vidal, V. (2011).
Yahsp2: Keep it simple, stupid.
In *Proceedings of the 7th International Planning Competition (IPC-2011)*, pages 83–90.

Backup Slides

Complexity Results for Timeline-based planning

- Timeline-based planning over discrete time is EXPSPACE-complete, NEXPTIME-complete if plan horizon is known in advance (see [Gigante et al., 2017])
- *Qualitative* timeline-based planning over discrete time is PSPACE-complete (see [Della Monica et al., 2020])
- Timeline-based planning over dense time is undecidable, with many decidable fragments ranging from EXPSPACE- to NP-complete (see [Bozzelli et al., 2020])
- Timeline-based games (i.e. playing against the nondeterministic environment) are 2EXPTIME-complete (see [Gigante et al., 2020b])