

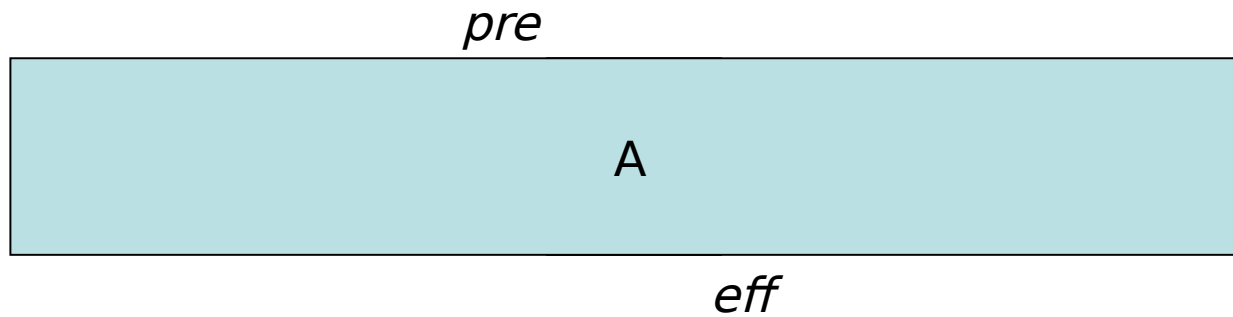
Hybrid Planning

Dr Andrew Coles
Department of Informatics,
King's College London

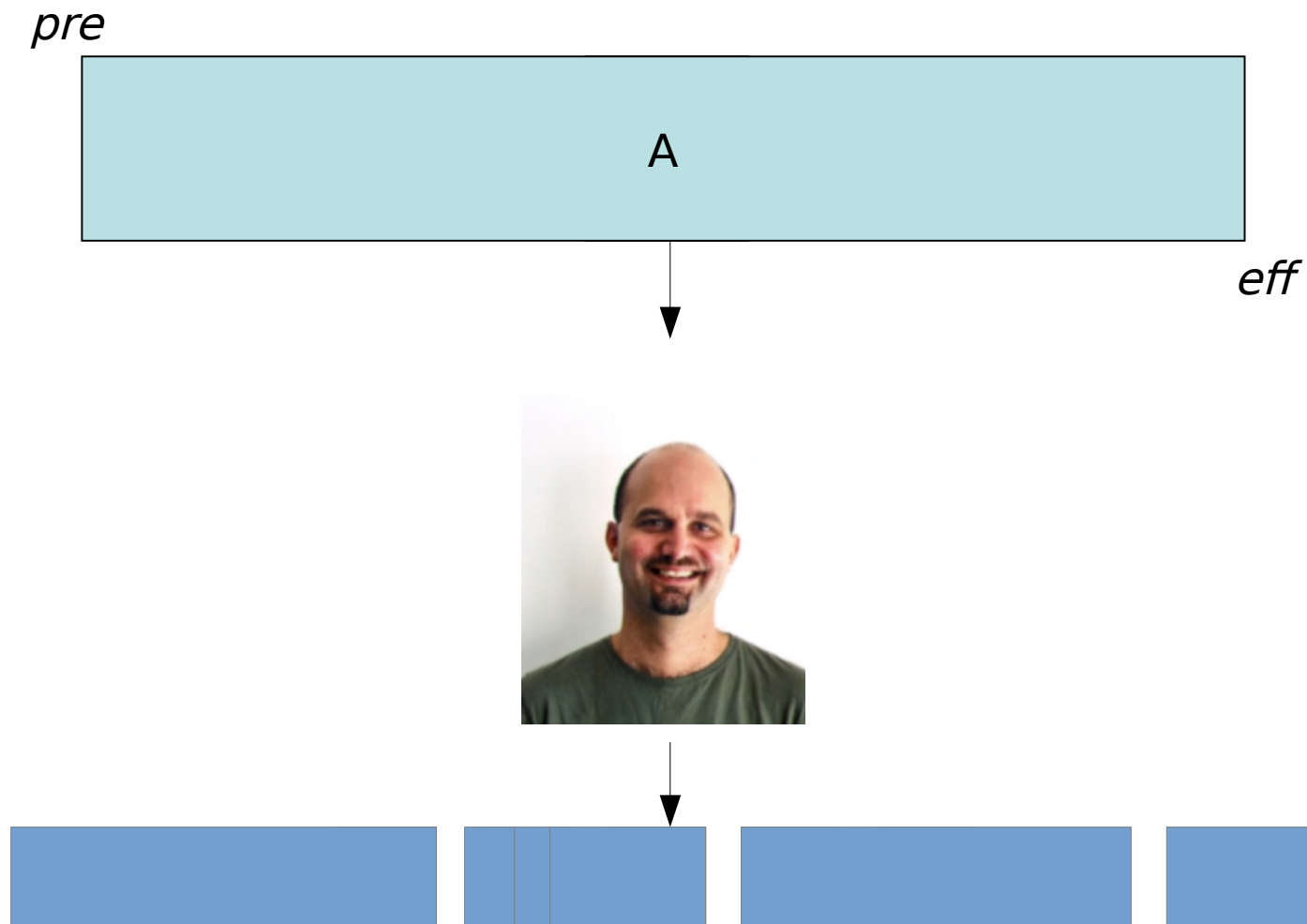
Planning with Time

- Classical planning: find a sequence of actions that are *logically* sound
 - No unsatisfied preconditions or goals
- What if actions have **varying durations**? How would we model this?

Durative Actions?



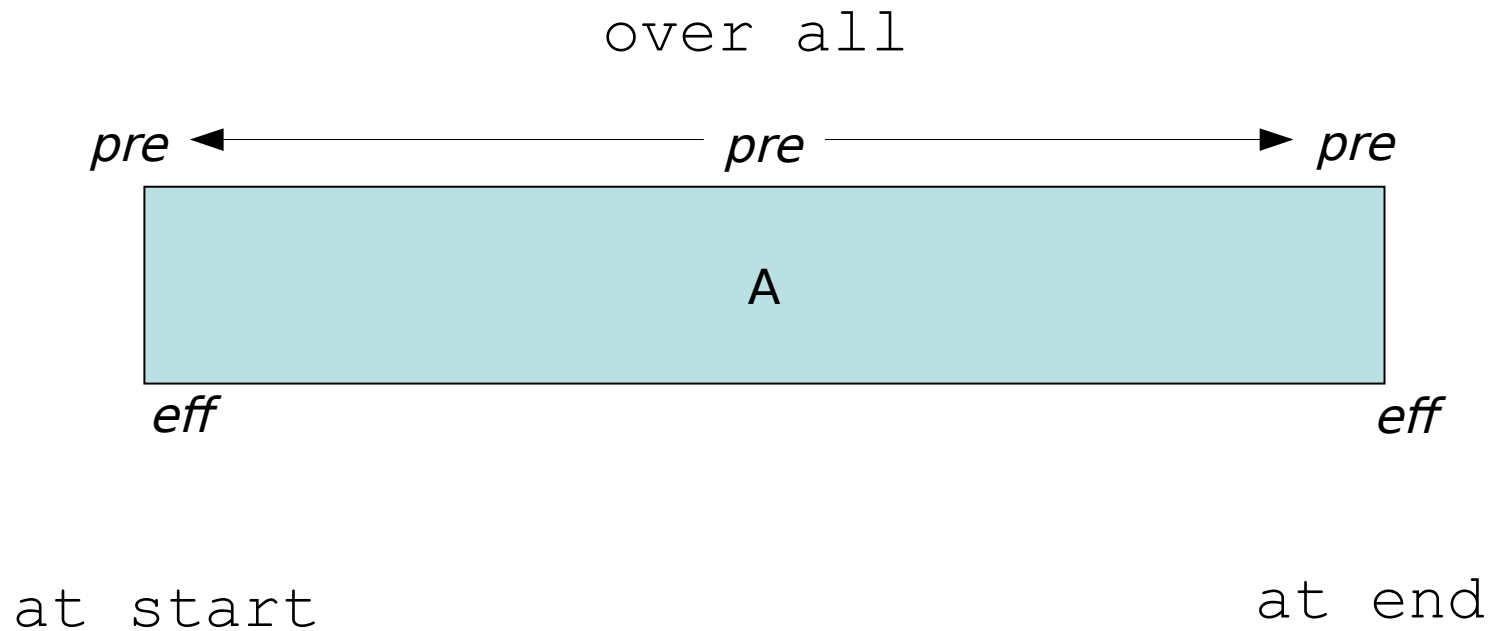
Durative Actions?



Shortest talk ever?

- If using a classical planner works, go for it
- But, what about **concurrency**?
- Simple example – mending two fuses in a cellar, using matches for light:
 - Light a match (burns for 8 seconds)
 - Mend the fuse (takes 5 seconds)
 - Mend the second fuse (takes 5 seconds)

Durative Actions in PDDL 2.1



PDDL Example (i)

```
(:      ·action LOAD-TRUCK

:parameters

(?obj - obj ?truck - truck ?loc -
location)

:precondition

  (and      (at ?truck ?loc))
            (at ?obj ?loc)))

:effect

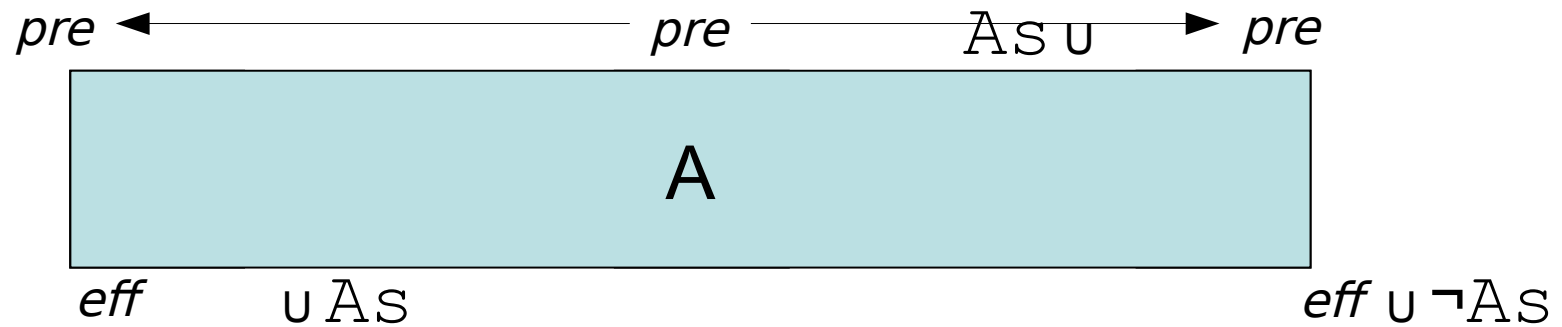
  (and      (not (at ?obj ?loc)))
            (in ?obj ?truck)))
```

PDDL Example (ii)

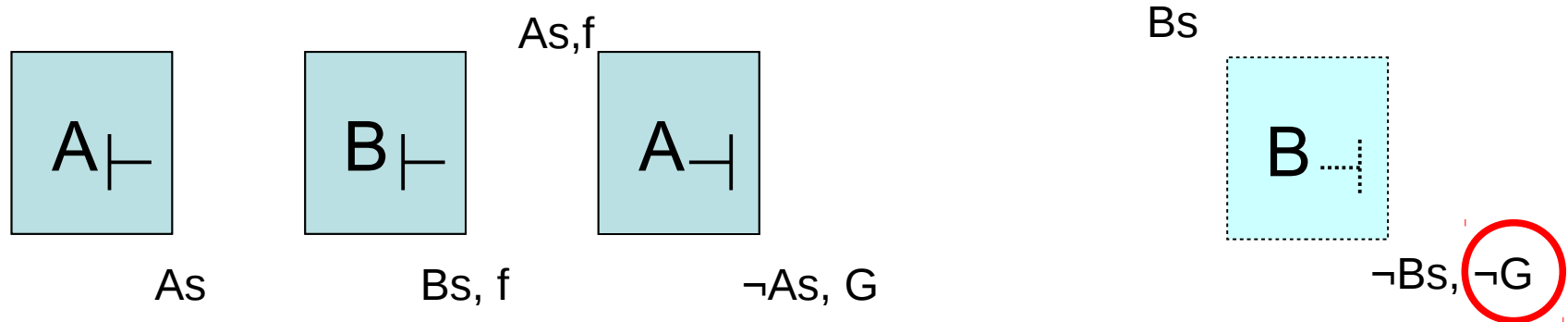
```
(:durative-action open-barrier
  :parameters
    (?loc - location ?p - person)
  :duration (= ?duration 1)
  :condition
    (and (at start (at ?loc ?p)))
  :effect
    (and (at start (door-open ?loc))
          (at end (not (door-open ?loc)))))
```


Durative Actions in LPGP

(Fox and Long, ICAPS 2003)



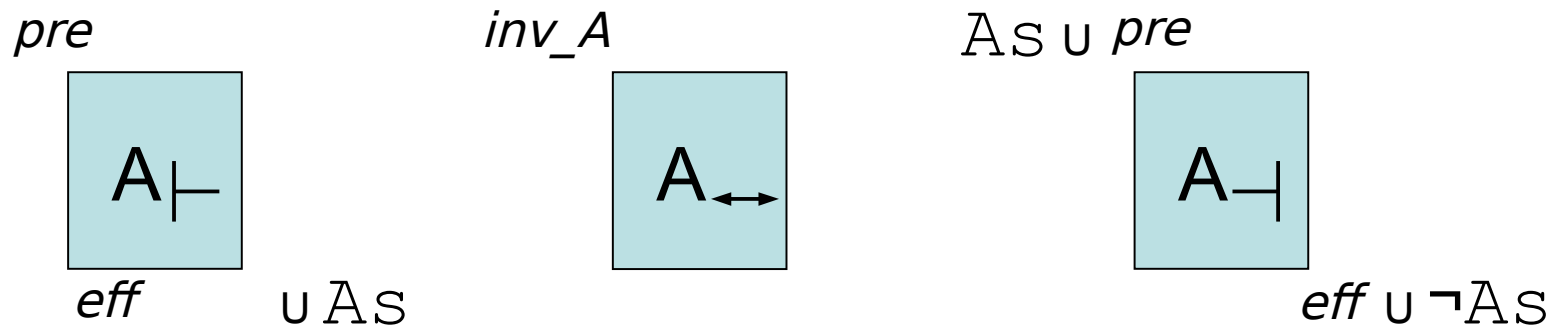
Planning with Snap Actions (i)



Challenge 1: What if $B \dashv$ interferes with the goal?

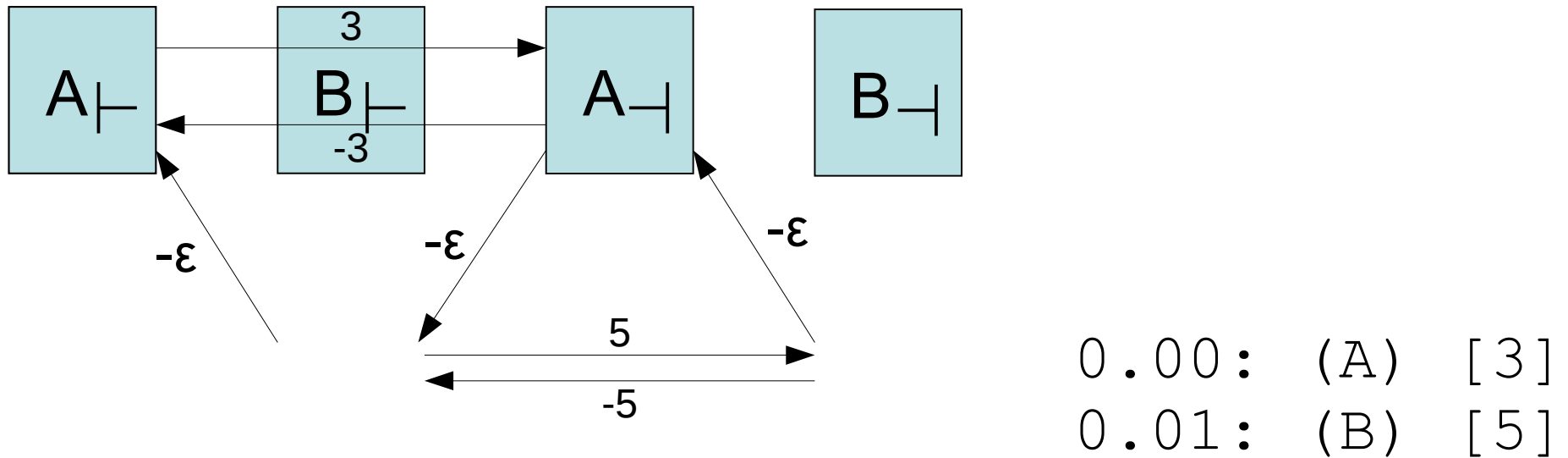
- PDDL 2.1 semantics: **no actions can be executing in a goal state.**
- **Solution:** add $\neg A_s, \neg B_s, \neg C_s \dots$ to the goal
 - (Or make this implicit in a temporal planner.)

Planning with Snap Actions (ii)



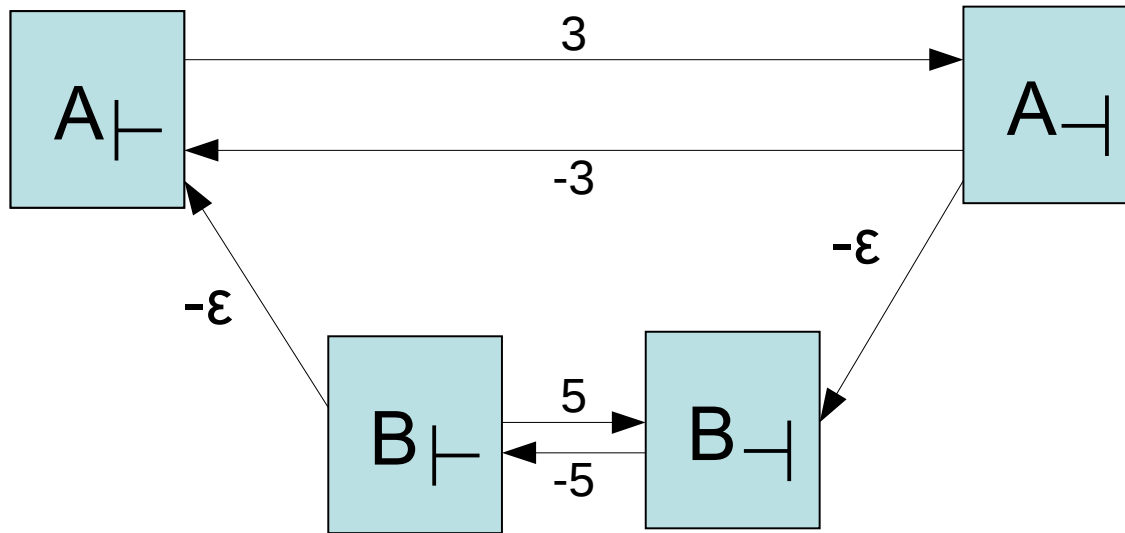
- Challenge 2: what about **over all** conditions?
 - If A is executing, inv_A must hold.
- **Solution:**
 - Add to each snap-action an extra precondition
 $(\text{imply } (A S) \quad inv_A)$
 - Violating an invariant then leads to a **dead-end**.
 - (Or make implicit in a temporal planner.)

Planning with Snap Actions (iii)



- Challenge 3: where did the durations go?
 - More generally, what are the temporal constraints?

Planning with Snap Actions (iii)



No satisfying set of timestamps exists.

- **Challenge 3: where did the durations go?**
 - More generally, what are the temporal constraints?
 - **Logically sound \neq temporally sound.**

Checking temporal constraints

- All our constraints are of the form:
 - $\varepsilon \leq t(j) - t(i)$ (*c.f. sequence constraints*)
 - $\text{dur}_{\min}(A) \leq t(A_{\perp}) - t(A_{\top}) \leq \text{dur}_{\max}(A)$
- Or, more generally, $lb \leq t(j) - t(i) \leq ub$
 - Is a **Simple Temporal Problem**
 - “Temporal Constraint Networks”,
Dechter, Meiri and Pearl, AIJ, 1991
- Good news – is **polynomial**
 - Bad news – needs checking every state

Various smarts

(c.f. POPF – ICAPS 2010)

- Only need to sequence step j to follow step i if:
 - Step i supports a precondition of j ; or,
 - Step j deletes a precondition of i .
- Heuristics: can use a Temporal Relaxed Planning Graph, where layers have time stamps
 - If the start of A appears at layer t , the end cannot appear before layer $t + \text{dur}_{\min}(A)$

What *really* happens when we light a match?

- We **choose** to take an action: strike the match
- This gives us light, and starts a **process** of burning, gradually reducing the length of the match

We **cannot choose** whether to do this: if the match is lit, it is burning away.

- If the match length hits zero, we burn our fingers. We can **avoid** this by **choosing** to blow out the match.

PDDL+ : What is it and why do we care?

**PDDL+ is the PDDL extension for modelling hybrid systems
(discrete modes + continuous change)**

- Many of our interactions with the world involve us performing actions that initiate, terminate, control or simply avoid continuous processes



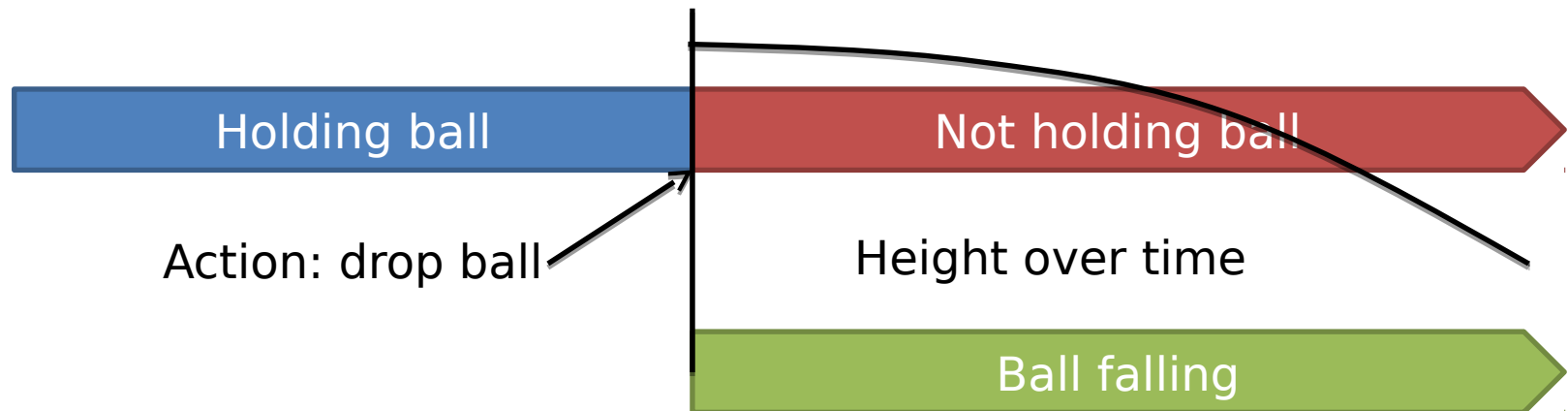
- Processes can represent flows: heat, energy, liquids, gases, traffic, money, information (**continuous change**)
- Reactions, motion, growth and contraction

What happens?

- Executive agents perform **actions** that change the world state in some way
- **Processes** execute continuously under the direction of the world
 - The world decides whether a process is active or not
- **Events** are the world's version of actions: the world performs them when conditions are met
- **Planning** is about deciding which actions the executive agents should perform by anticipating the effects of actions before they are executed
 - Requires a way to predict what will happen when actions are performed (and also when they are not)

What makes it *hybrid*?

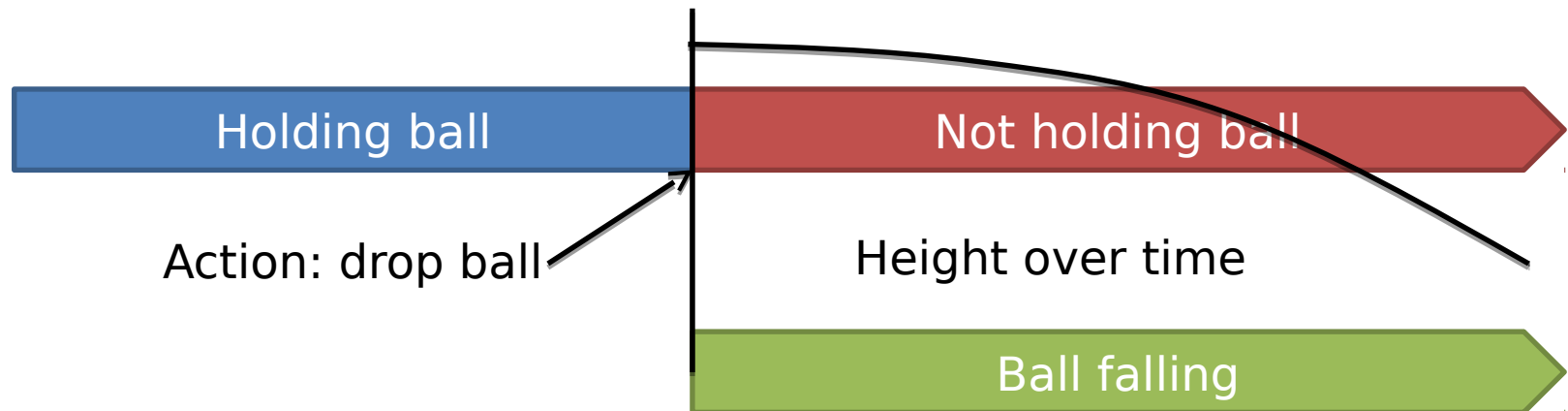
- When actions or events are performed they cause instantaneous changes in the world
 - These are discrete changes to the world state
 - When an action or an event has happened it is over



- Processes are continuous changes
 - Once they start they generate continuous updates in the world state
 - A process will run over time, changing the world at every instant

What makes it *hybrid*?

- When actions or events are performed they cause instantaneous changes in the world
 - These are discrete changes to the world state
 - When an action or an event has happened it is over



```
(:process fall
:parameters (?b - ball)
:precondition (and (not (holding ?b)) (>= (height ?b) 0)))
:effect (and (increase (velocity ?b) (* #t (gravity)))
             (decrease (height ?b) (* #t (velocity ?b)))))
```

“When a ball, bounces off of a wall; ...”

- Events are the world's version of actions: the world performs them when conditions are met

```
(:event bounce
:parameters (?b - ball)
:precondition (and (>= (velocity ?b) 0)
                  (<= (height ?b) 0))
:effect (and (assign (height ?b) (* -1 (height ?b)))
             (assign (velocity ?b) (* -1 (velocity ?b)))))
```

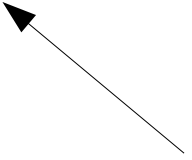
```
(:process fall
:parameters (?b - ball)
:precondition (and (not (holding ?b)) (>= (height ?b) 0)))
:effect (and (increase (velocity ?b) (* #t (gravity)))
             (decrease (height ?b) (* #t (velocity ?b)))))
```

How to plan now?

- The path from the initial state to the goal can be describe as a sequence of **time-stamped happenings**:
 - i. Instantaneous actions; start/end of durative
 - ii. A process starts or ends
 - iii. An event is triggered
- The solution plan only contains (i) but the planner needs to account for the rest.
- Happenings denote **discrete** change; with **continuous** processes running between them.

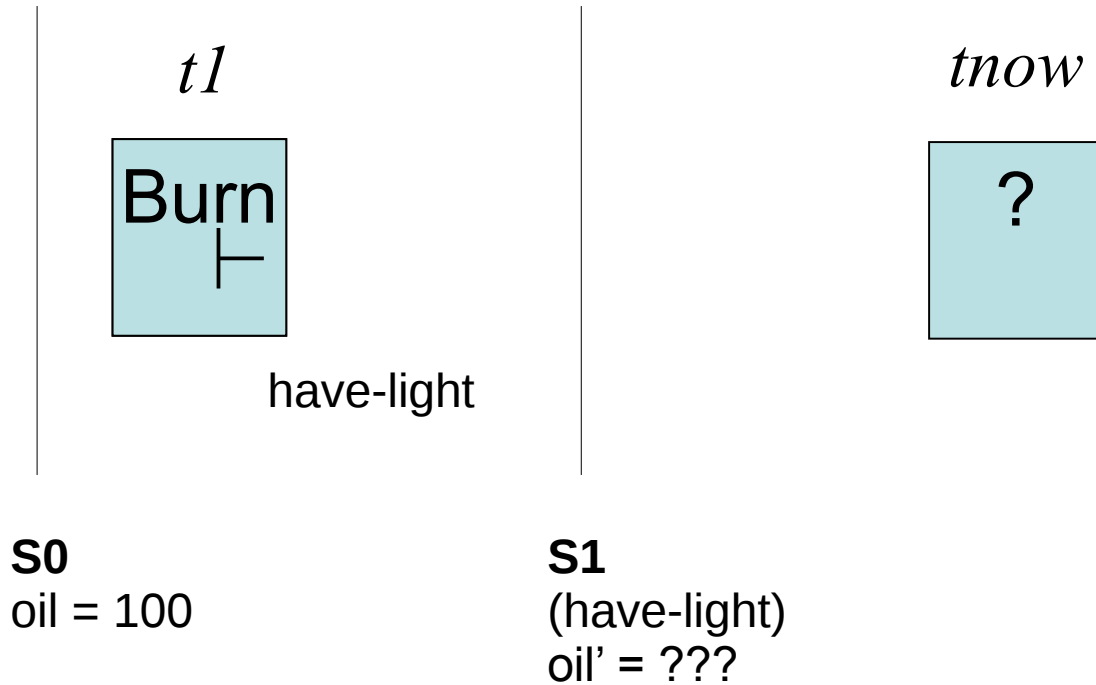
Simple case: processes within actions

```
(:durative-action burn-oil-lamp
:duration (> ?duration 0)
:condition (over all (>= (oil) 0))
:effect (and
  (at start      (have-light) )
  (at end    (not (have-light)))
  (decrease (oil) (* #t 1))
)
)
```



$doil / dt = 1$

Time-dependent values



The oil level depends on the **relative timing** of actions:

$$oil' = oil + \delta oil \cdot (tnow - t1)$$

Refilling the lamp

```
(:durative-action refill-oil-lamp
 :duration (= ?duration 10)
 :condition (and
   (at start (have-refill))
   (over all (<= (oil) 90))
 )
 :effect (and
   (at start (not (have-refill)))
   (increase (oil) (* #t 2))
 )
)
```

Starting to refill the lamp

- We can refill when $\text{oil} \leq 90$
- As step 2 of the plan:

$$\text{oil}[1] = 100$$

$$\text{oil}[2] = \text{oil}[1] - (t2 - t1)$$

$$\text{oil}[2] \leq 90$$

$$\text{oil}[2] \geq 0$$

$$t2 \geq t1 + 0.01$$

Solve as a **linear programming** problem (OPTIC, TM-LPSAT)

oil[1] = 100

oil[2] = oil[1] - (t2 - t1)

oil[2] ≤ 90

oil[2] ≥ 0

oil[3] = oil[2] + (t3 - t2)

oil[3] ≤ 90

oil[3] ≥ 0

oil[4] = oil[3] - (t4 - t3)

oil[4] ≥ 0

t2 ≥ t1 + 0.01

t3 - t2 = 10

t4 ≥ t3 + 0.01

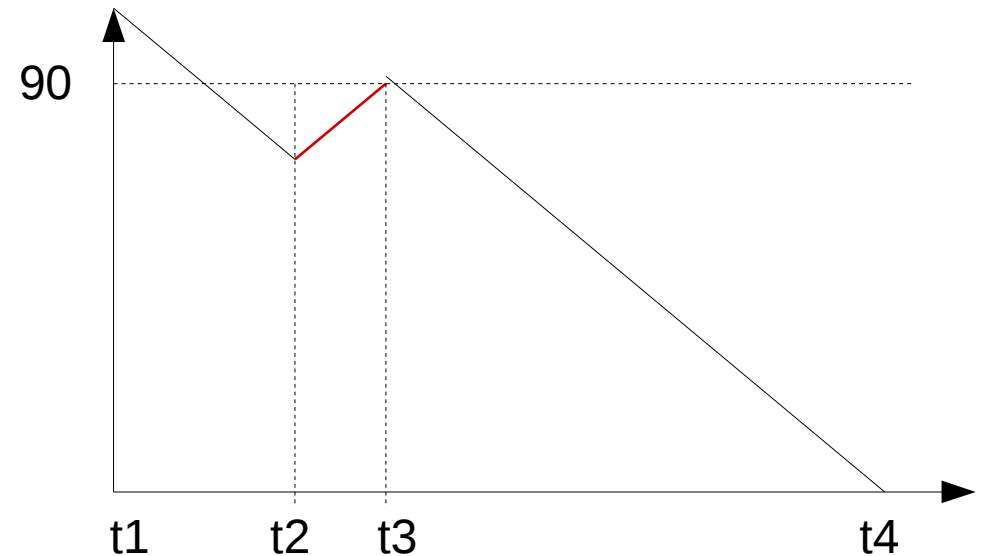
t4 - t1 ≥ 0.01

t1: burn-oil-lamp_start

t2: refill-oil-lamp_start

t3: refill-oil-lamp_end

t4: burn-oil-lamp_end



What about full PDDL+?

- Thought experiment: pretend all happenings are actions
 - Can apply any action/event/process whose conditions are satisfied
 - Can stop any process whose conditions are unsatisfied

?

Issue 1: don't branch over the obvious

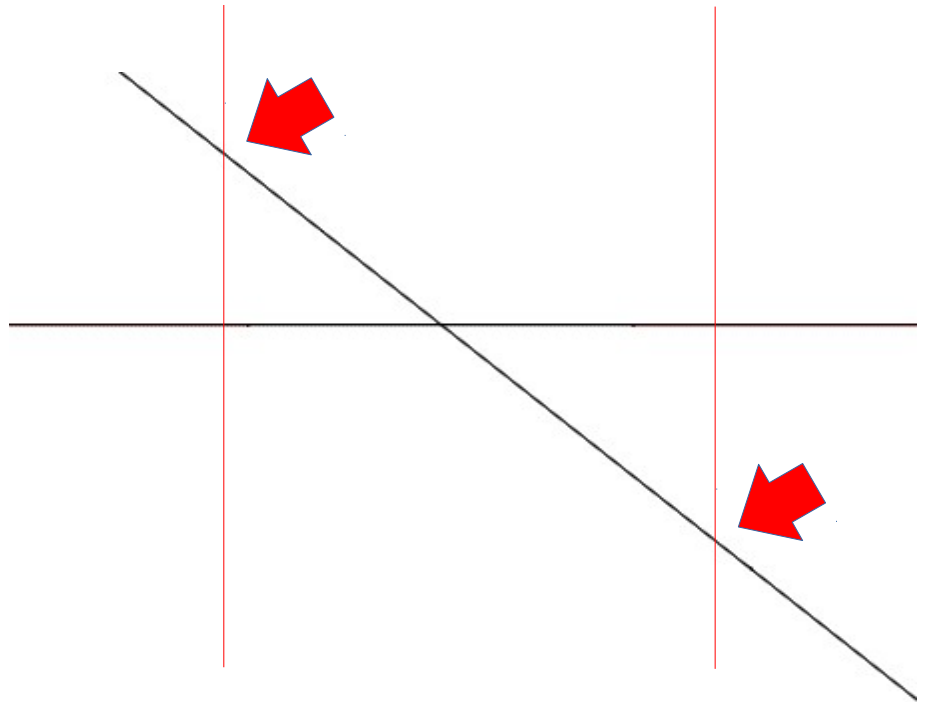
- Search reaches a state S
- LP calculates v to be in the range $[0,50]$
- An event (e) has condition $(\leq (v) 100)$
- **Apply (e) – don't branch over which actions to apply.**

Issue 2: don't overlook happenings

- In classical planning: can apply (a) then (b)
- Now: what if an event (e) should have triggered between (a) and (b)?
- Solution: when applying an action, add the **negation** of the condition on each event (e); solve the LP
 - Same principle as 'over all' conditions
 - If unsolvable: (e) should have been 'applied'

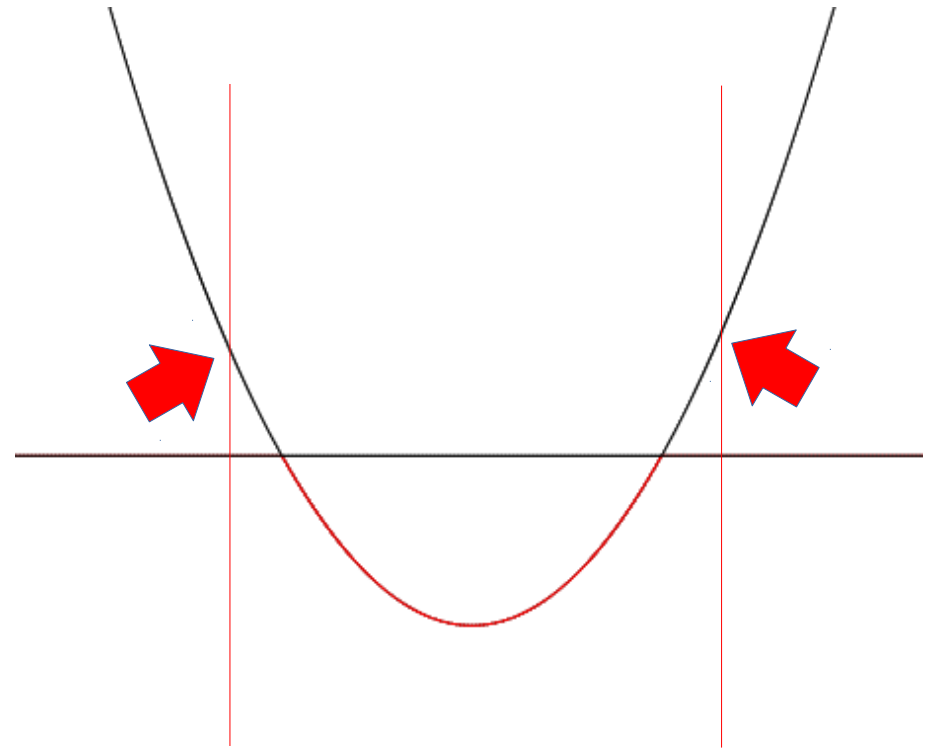
Conditions over linear change

- With **linear** processes: suffices to check conditions at **happenings**



Conditions over non-linear change

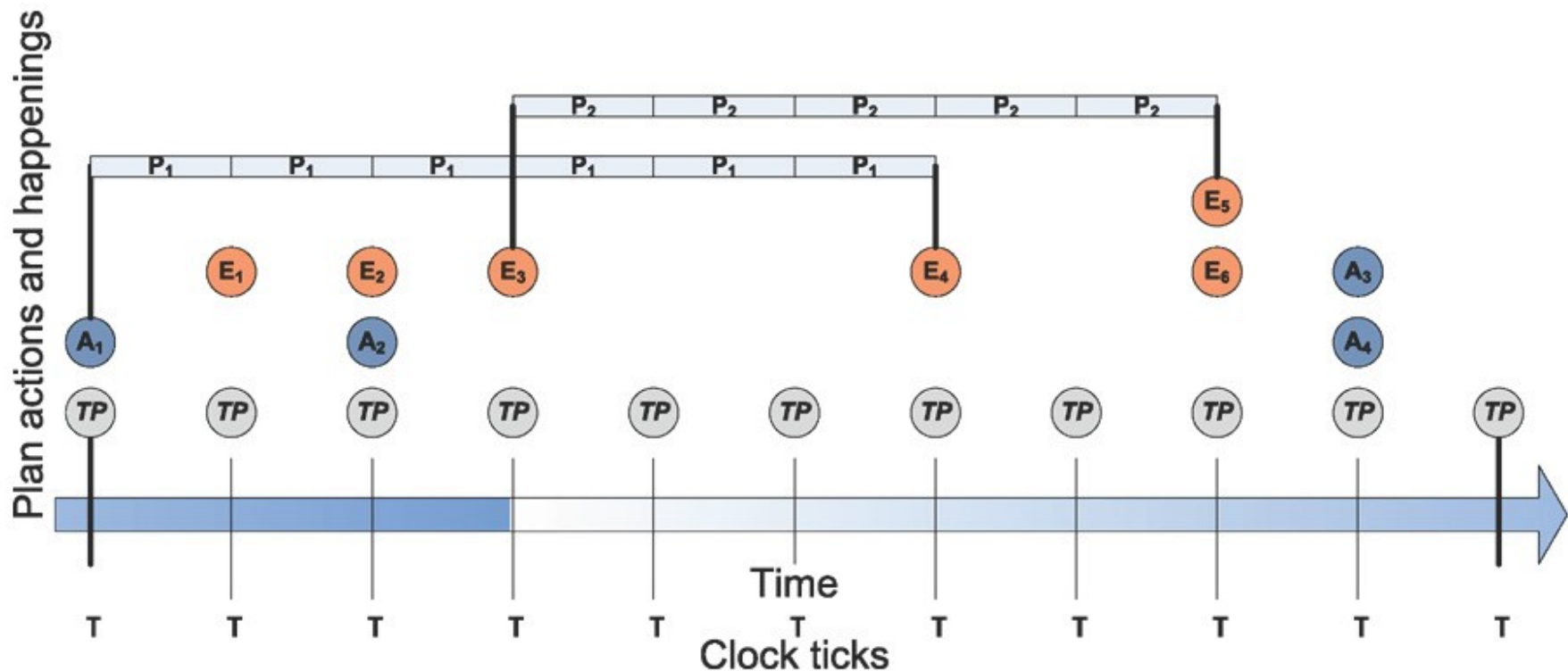
- With **non-linear** processes, this assumption is lost.
- Also, cannot use an LP to determine variable values at happenings – gradients are not constants:



$$oil[2] = oil[1] + \delta oil . (t2 - t1)$$

Solution 1: Discretise time

- UPMurphi (Della Penna *et al*) discretises time
- Add an action **time_passing** that increments t by some amount T :



Initial time discretisation: 0.1 s**5.089**

0.0:	(use b1)	[3.40]	5.60:	(use b2)	[0.20]
3.40:	(use b2)	[0.50]	5.80:	(use b1)	[0.30]
3.90:	(use b1)	[0.10]	6.10:	(use b2)	[0.70]
4.00:	(use b2)	[0.50]	6.80:	(use b1)	[0.20]
4.50:	(use b1)	[0.60]	7.00:	(use b2)	[1.30]
5.10:	(use b2)	[0.20]	8.30:	(use b1)	[0.20]
5.30:	(use b1)	[0.30]	8.50:	(use b2)	[0.20]
			8.70:	(satisfied)	

Checking next happening (time 5.08986)

Updating (gamma b1) (0.502404) by 0.337447 assignment

Updating (delta b1) (0.328362) by 0.550475 assignment

Updating (delta b2) (0.405504) by 0.257052 assignment

EVENT triggered at (time 5.08986)

Triggered event (batterydead b1)

Deleting (switchedon b1)

Adding (dead b1)

Invariant for (use b1) has its condition unsatisfied
between time 5.08986 to 5.1.

Refined time discretisation: 0.01 s

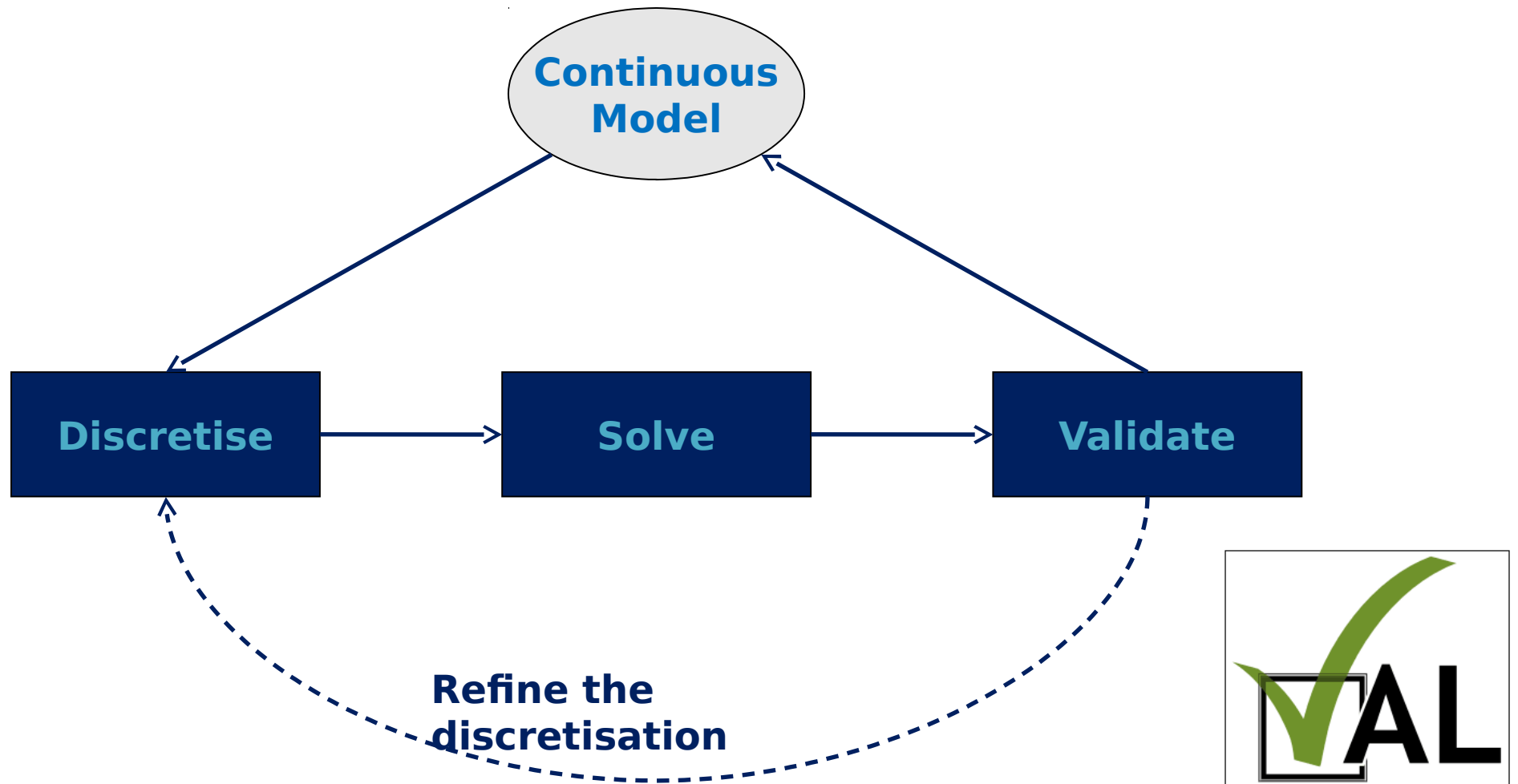
5.089

0.0:	(use b1)	[3.40]
3.40:	(use b2)	[0.50]
3.90:	(use b1)	[0.10]
4.00:	(use b2)	[0.50]
4.50:	(use b1)	[0.60]
5.10:	(use b2)	[0.20]
5.30:	(use b1)	[0.30]
5.60:	(use b2)	[0.20]
5.80:	(use b1)	[0.30]
6.10:	(use b2)	[0.70]
6.80:	(use b1)	[0.20]
7.00:	(use b2)	[1.30]
8.30:	(use b1)	[0.20]
8.50:	(use b2)	[0.20]
8.70:	(satisfied)	

0.0:	(use b1)	[3.40]
3.40:	(use b2)	[0.50]
3.90:	(use b1)	[0.10]
4.00:	(use b2)	[0.50]
4.50:	(use b1)	[0.58]
5.08:	(use b2)	[0.27]
5.35:	(use b1)	[0.08]
5.43:	(use b2)	[0.57]
6.10:	(use b1)	[0.05]
6.15:	(use b2)	[0.40]
6.55:	(use b1)	[0.05]
6.60:	(use b2)	[0.50]
7.10:	(use b1)	[0.05]
7.15:	(use b2)	[1.00]
8.15:	(use b1)	[0.30]
8.45:	(use b2)	[0.20]
8.65:	(use b1)	[0.05]
8.70:	(satisfied)	

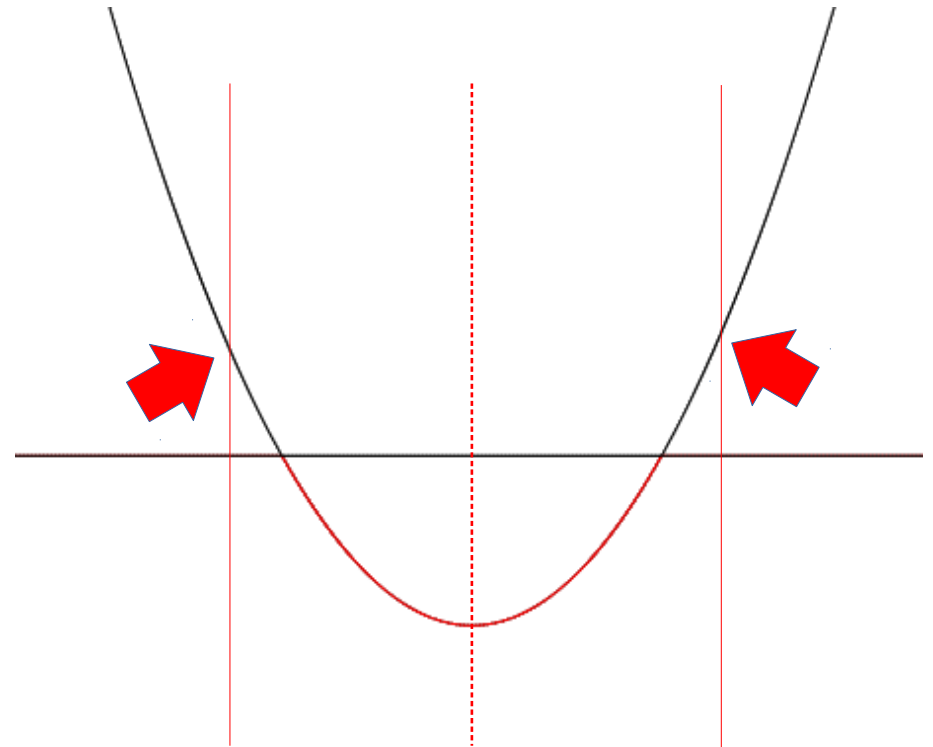
***0.01014 before than
expected***

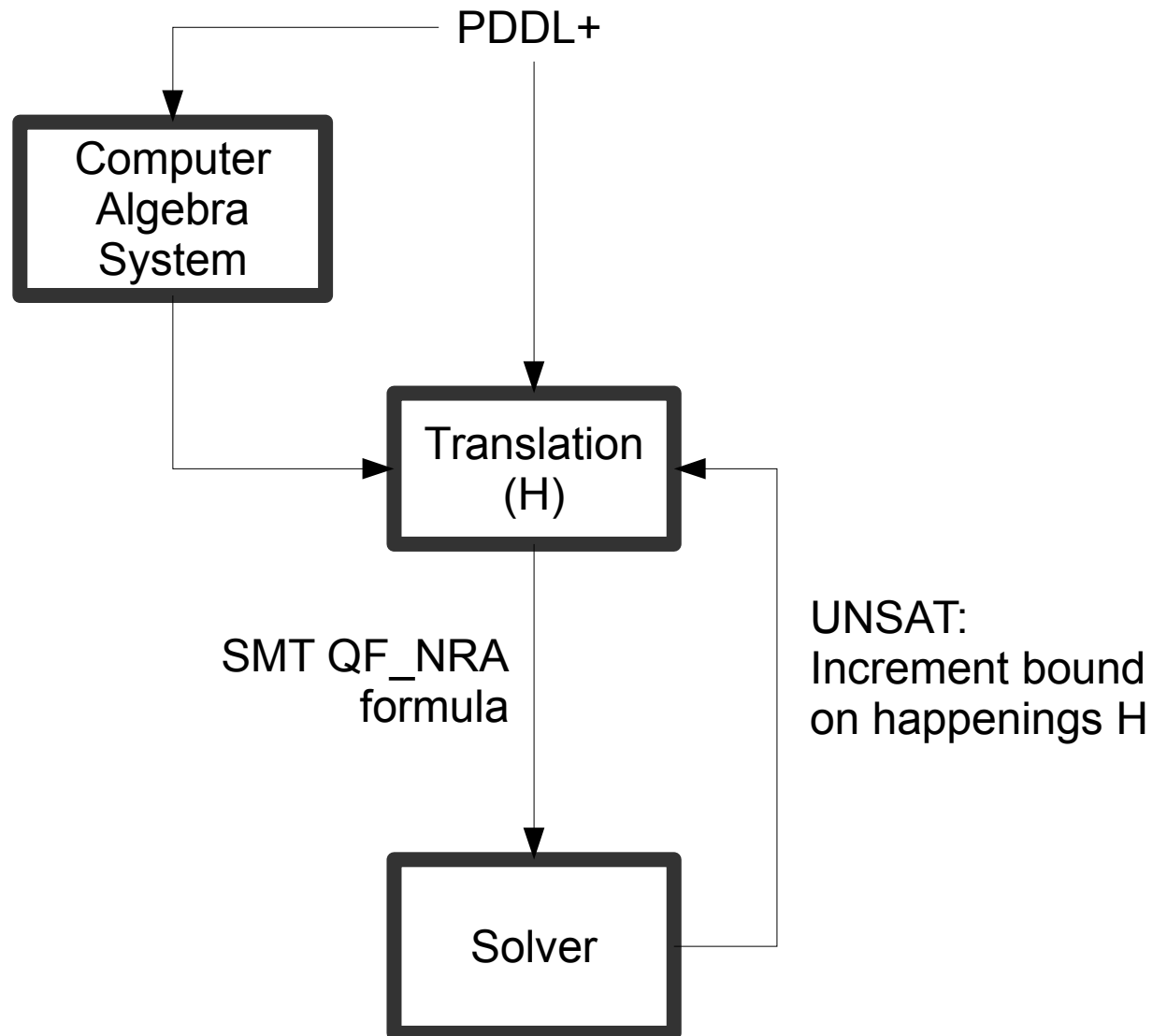
Discretise—Validate



Solution 2: Use a non-linear solver

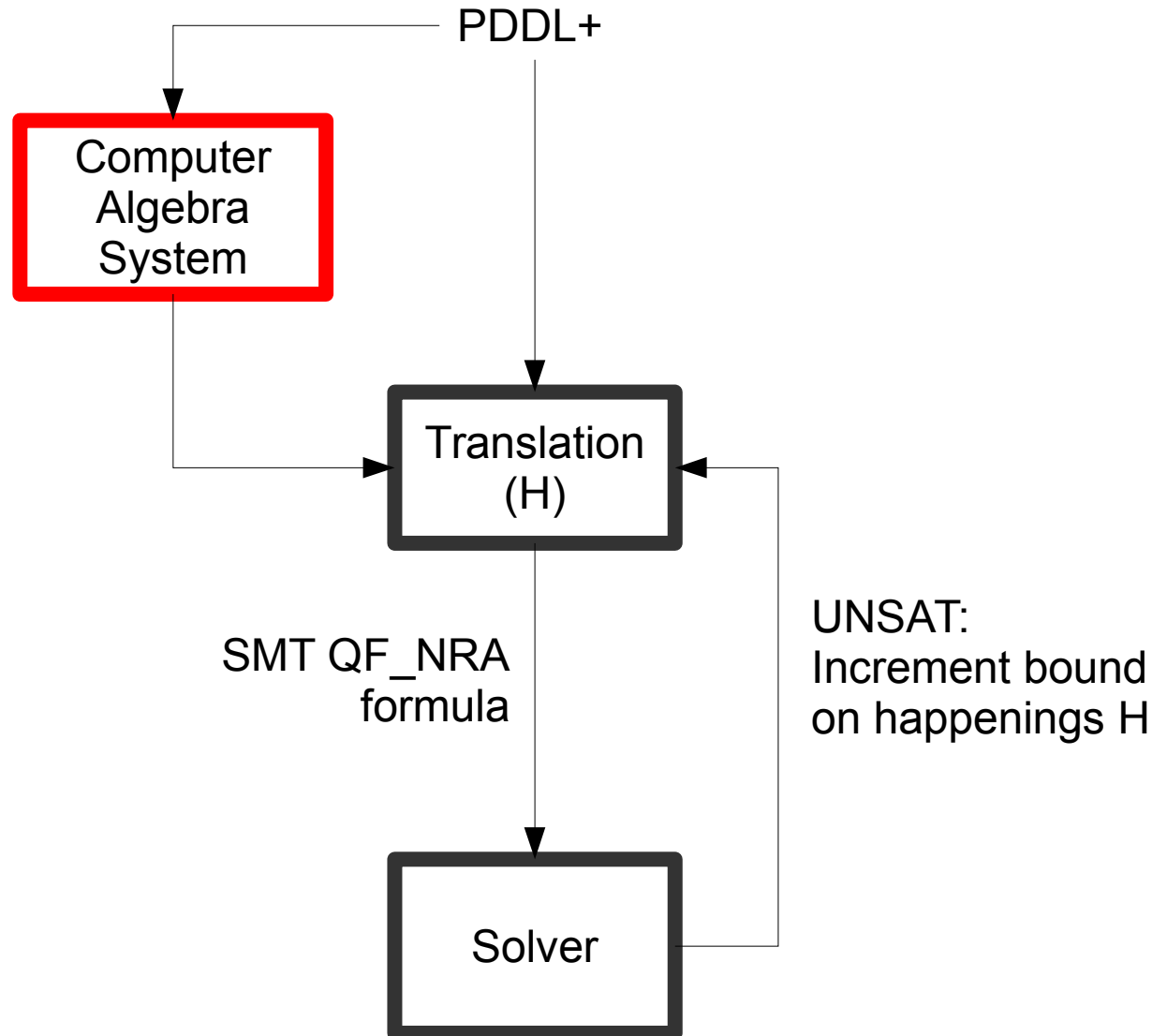
- **SMTPlan+**
(Cashmore *et al*) uses SMT with a non-linear **theory**
- For the zero-crossing problem: the first-derivative **cannot change signs** between two happenings





Continuous effects in the **domain** are translated into formulae.

Integration and differentiation is performed offline. (SymPy).



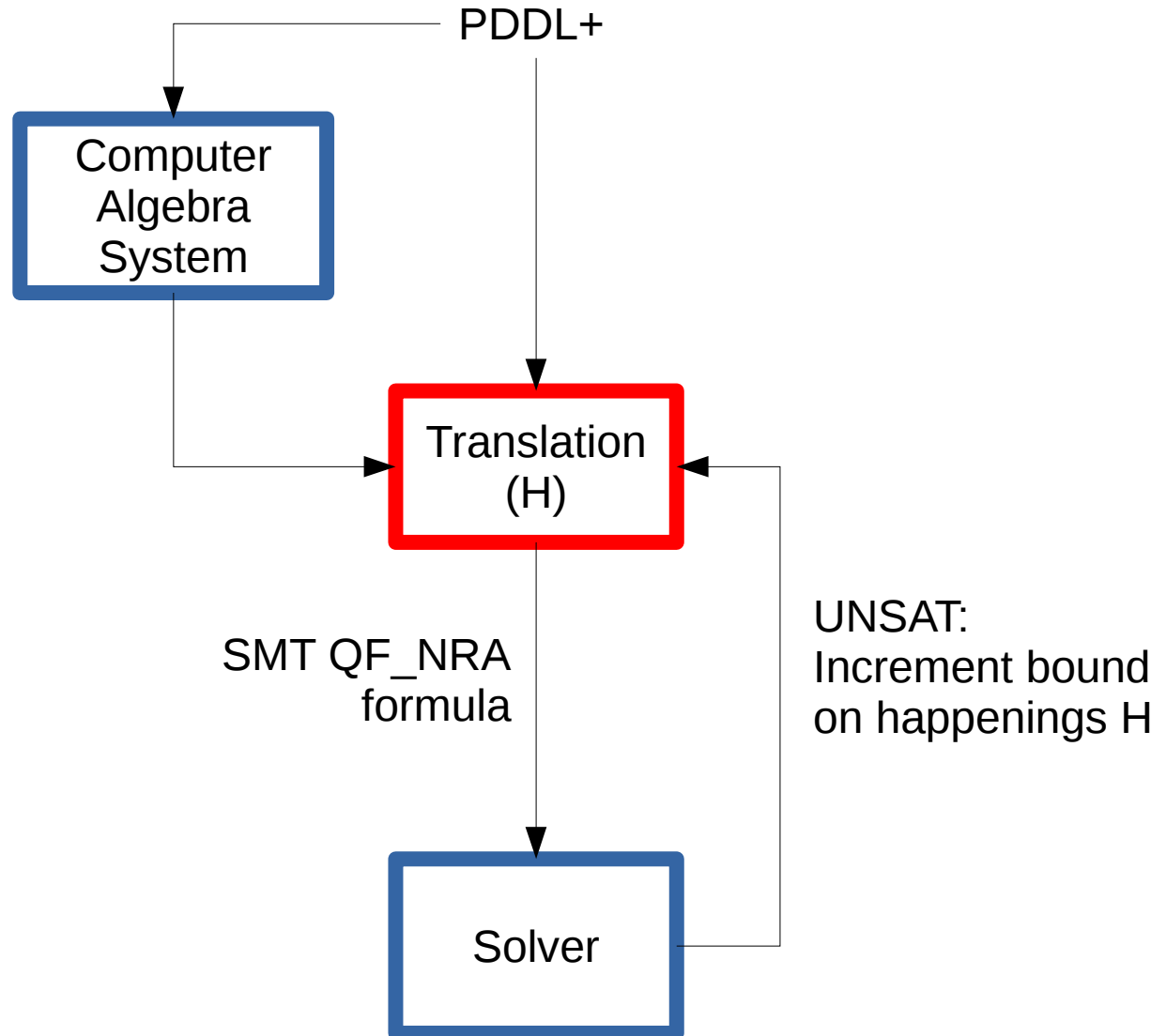
Continuous effects in the **domain** are translated into formulae.

Integration and differentiation is performed offline. (SymPy).

A PDDL+ **problem** is translated into a conjunctive normal form (CNF) SMT formula.

In the logic of *Quantifier-free Nonlinear Real Arithmetic* (QF_NRA)

The number of happenings is bounded (H)



Continuous effects in the **domain** are translated into formulae.

Integration and differentiation is performed offline. (SymPy).

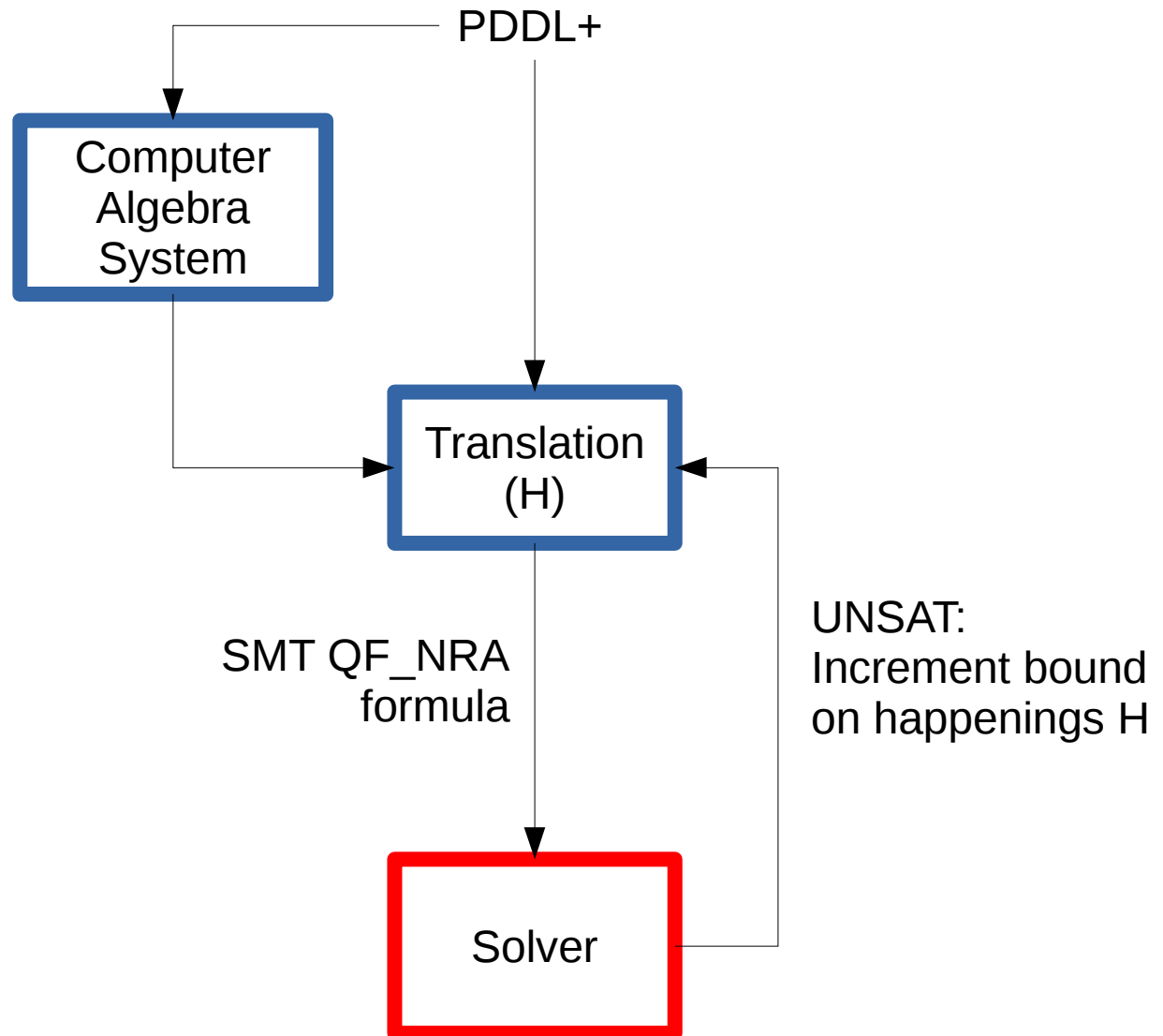
A PDDL+ **problem** is translated into a conjunctive normal form (CNF) SMT formula.

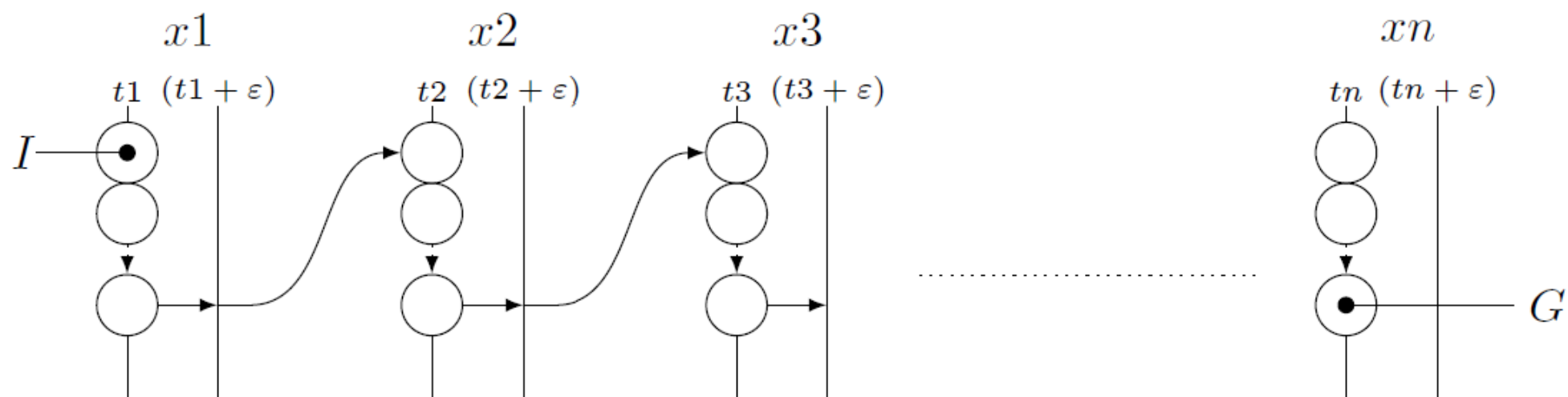
In the logic of *Quantifier-free Nonlinear Real Arithmetic* (QF_NRA)

The number of happenings is bounded (H)

An SMT solver produces a solution: a **plan**.

If the formula has no solution, the bound is increased (H++)





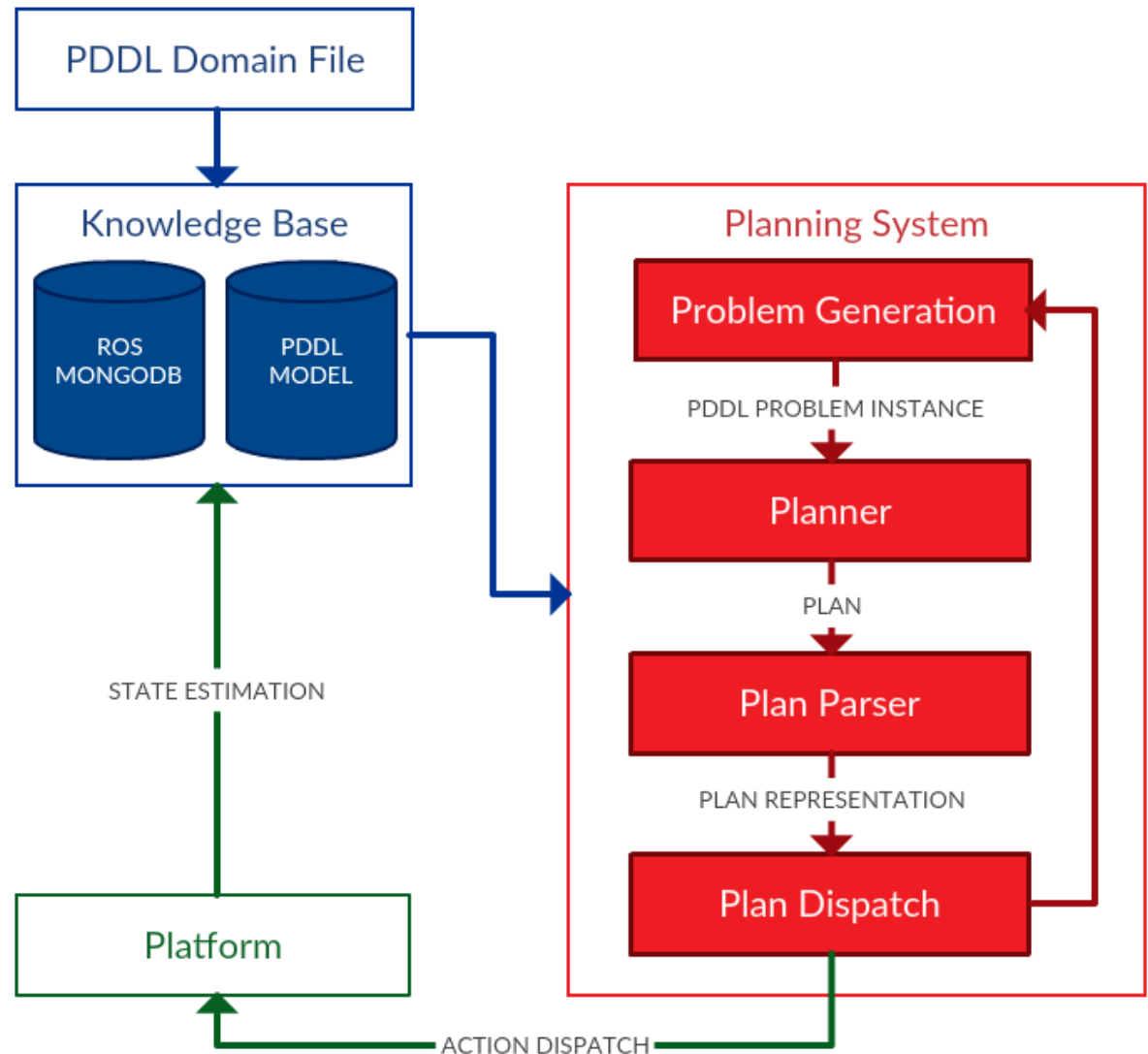
PANDORA



The AUV plans for inspection missions, recording images of pipes and welds. It navigates through a probabilistic roadmap. The environment is uncertain, and the roadmap might not be correct.

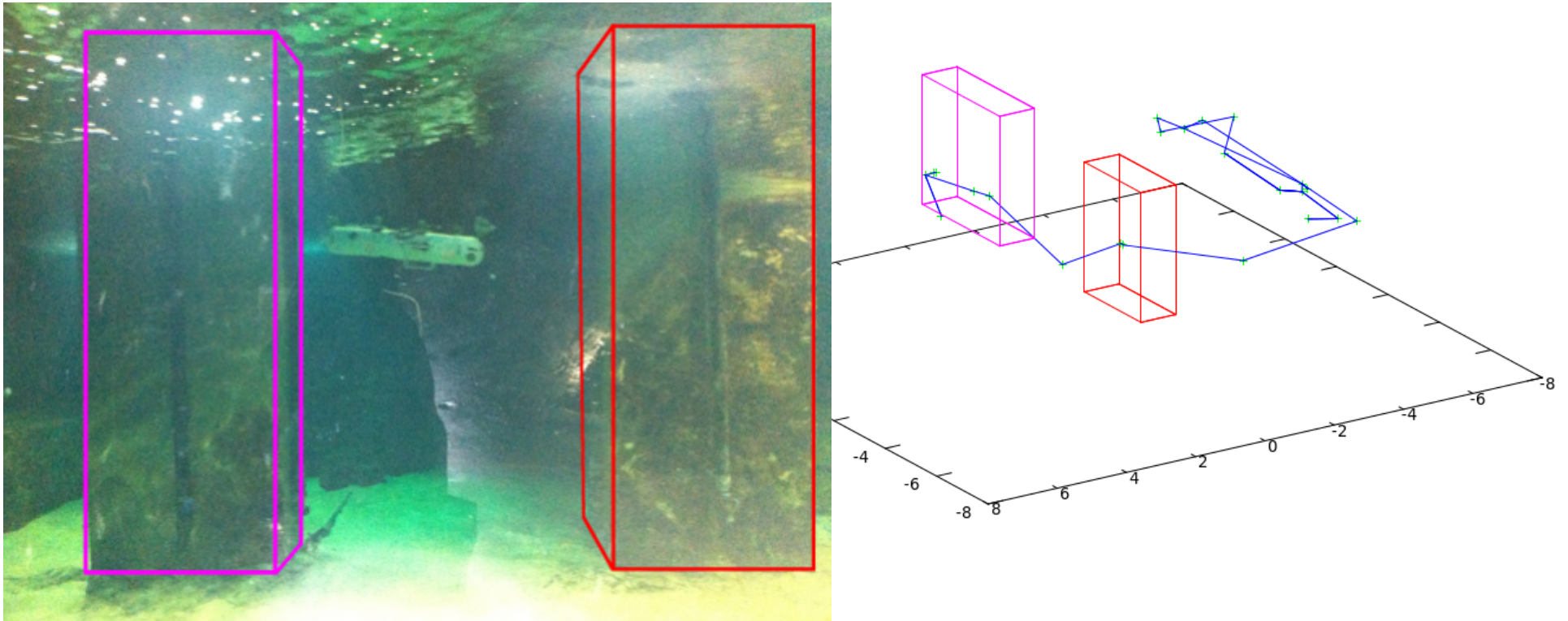
<http://kcl-planning.github.io/ROSPlan/>

- PDDL+ model is continuously updated from sensor data.
- problem file is automatically generated.
- the planner generates a plan.
- the plan is dispatched action-by-action.
- feedback on action success and failure.
- the plan is validated against the current model.



Replanning for the AUV

The plan is continuously validated against the model.



The planned inspection path is shown on the right. The AUV will move around to the other side of the pillars before inspecting the pipes on their facing sides.

After spotting an obstruction between the pillars, the AUV should re-plan early.

TL;DR

- Need to model time? Planners using STNs are your friend
- Linear continuous change? Planners using LPs
- Non-linear? Discretise, or non-linear solver
- Even a hybrid PDDL+ model approximates an uncertain environment – ROSPlan designed to fill this gap

Acknowledgements

- ERGO has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. 730086
- PANDORA is an EU FP7-ICT STREP funded research project

Further reading

Generative planning for hybrid systems based on flow tubes.

Li, H. X., and Williams, B. C.; In Proceedings of ICAPS 2008.

COLIN: Planning with continuous linear numeric change.

Coles, A. J., Coles, A.; Fox, M.; and Long, D. 2012. Journal of Artificial Intelligence Research (JAIR)

PDDL+ Planning with Events and Linear Processes

Coles, A.J.; Coles, A.I.; In Proceedings of ICAPS 2014

Processes and continuous change in a SAT-based planner.

Shin, J.-A., and Davis, E. Artificial Intelligence; 2005

Planning as model checking in hybrid domains

Bogomolov, S.; Magazzeni, D.; Podelski, A.; and Wehrle; In Proceedings of AAAI 2014

PDDL+ planning with hybrid automata: Foundations of translating must behavior

Bogomolov, S.; Magazzeni, D.; Minopoli, S.; and Wehrle, M.; In Proceedings of ICAPS; 2015

UPMurphi Released: PDDL+ Planning for Hybrid Systems.

G. Della Penna, B. Intrigila, D. Magazzeni, F. Mercorio. MOCHAP 2015

Heuristic Planning for PDDL+ Domains (DiNo)

Piotrowski, W.; Fox, M.; Long, D.; Magazzeni, D.; Mercorio, F.; In Proceedings of IJCAI 2016

A compilation of the full PDDL+ language into SMT

Cashmore, M.; Fox, M.; Long, D.; Magazzeni, M. In Proceedings of ICAPS 2016

SMT-based nonlinear PDDL+ planning.

Bryce, D.; Gao, S.; Musliner, D. J.; and Goldman, R. P.; In Proceedings of AAAI 2015