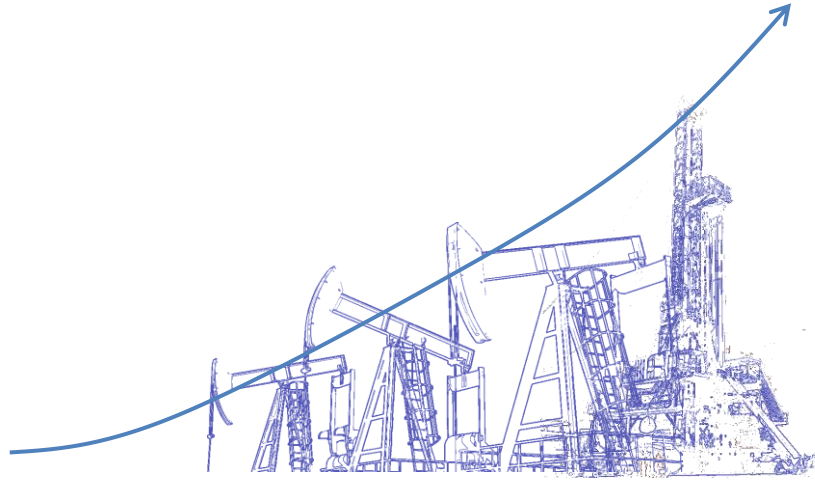
Automated Planning Around Processes

Derek Long
Schlumberger Cambridge Research, UK
King's College London, UK

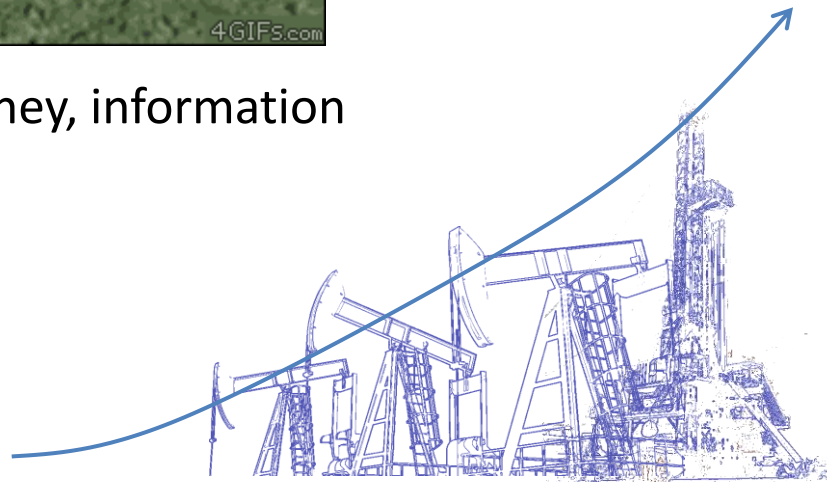
Mikhail Soutchanski
Toronto Metropolitan University, Canada



What is it and why do we care?

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

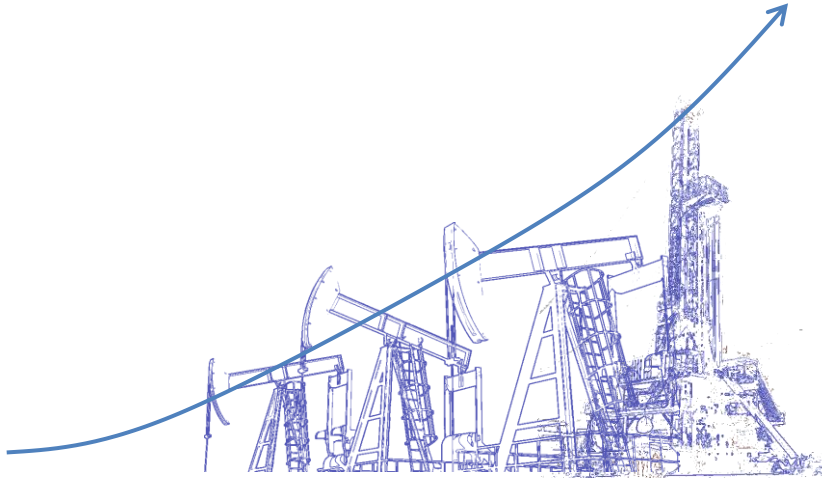
- Processes can represent flows: heat, energy, liquids, gases, traffic, money, information
- Reactions, motion, growth and contraction



Planning and Control

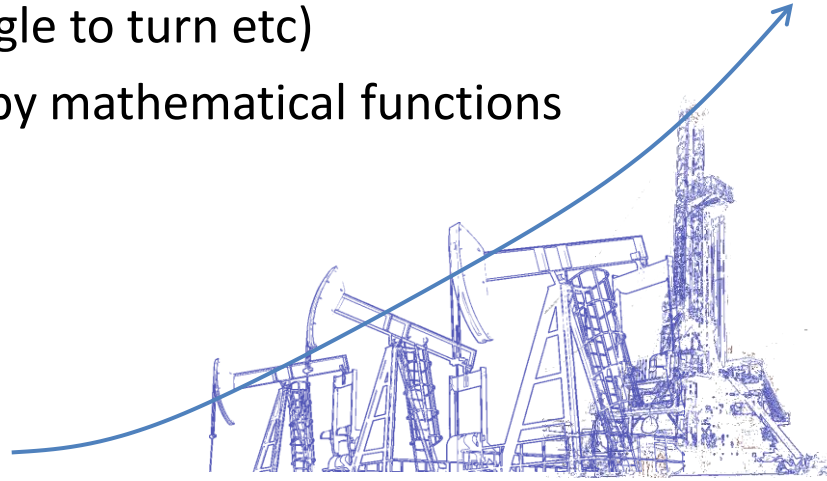
- Planners are designed to tackle “long horizon” discrete control problems – “long” means multi-action
- Mixed discrete-continuous planning problems are hard hybrid control problems at wide ranging granularities

- Planners are traditionally aimed at achieving target end states – the planning “control” problem is about moving efficiently towards a good end state
- Control systems are usually concerned with trajectories – they are designed to keep the behaviour close to a nominal trajectory



Mixed Dynamics

- Control involves making several distinct types of decision:
  - **Action choices** (open this valve or that valve? Use this cutting tool or switch to a different tool? Etc)
    - Discrete choices leading to mode changes
    - Often at longer horizon
  - **Timing choices** (this first or that? Perform this at time *t*? Etc)
    - Discrete or continuous?
    - Depends on temporal model
  - **Parameter choices** (flow rates, power levels, distance to move, angle to turn etc)
    - Generally continuous parameter spaces and choice governed by mathematical functions
    - Often at short horizon

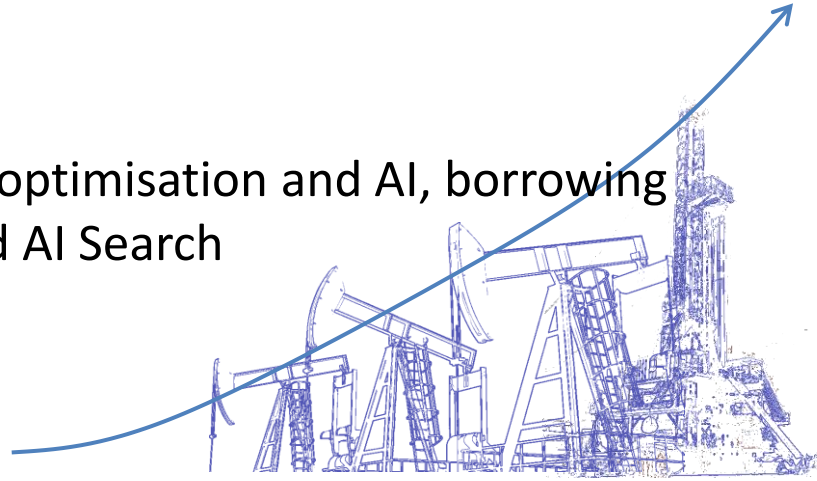


Mixed Discrete-Continuous Systems

- Optimising purely continuous functions is a mathematical problem
  - Linear functions well understood and tractable
  - Non-linear functions manageable under certain conditions
    - Common strategy is to linearise over sufficiently short horizons and use LP

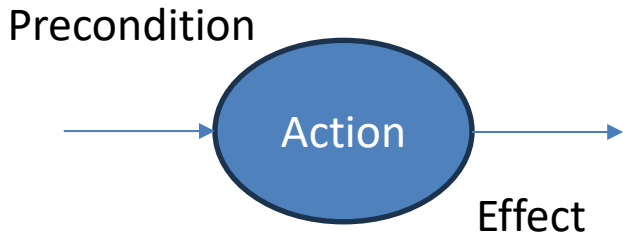
- Discrete control choices introduce combinatorial complexity
  - Control of discrete systems involves search
  - Mixed discrete-continuous optimisation is NP-hard with linear functions (eg adding integer constraints to some variables)

- Mixed Integer Linear Programming (MILP) is a well-developed area of optimisation and AI, borrowing from Operations Research, Mathematics, Constraint Programming and AI Search

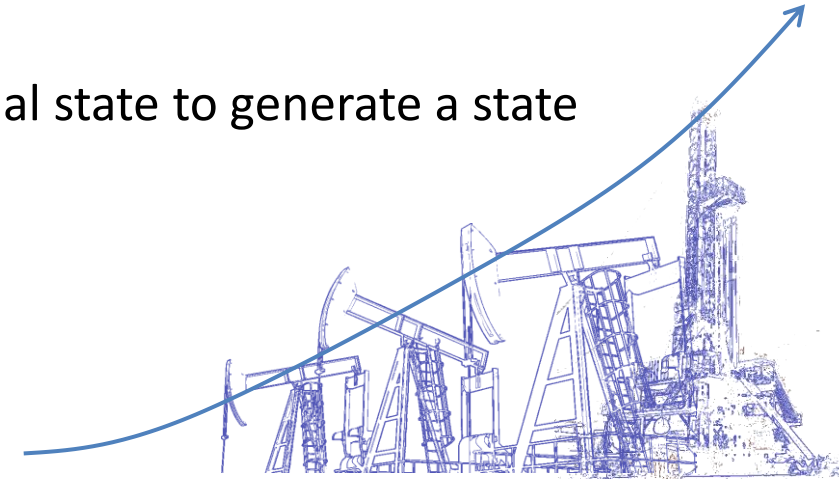


AI Planning

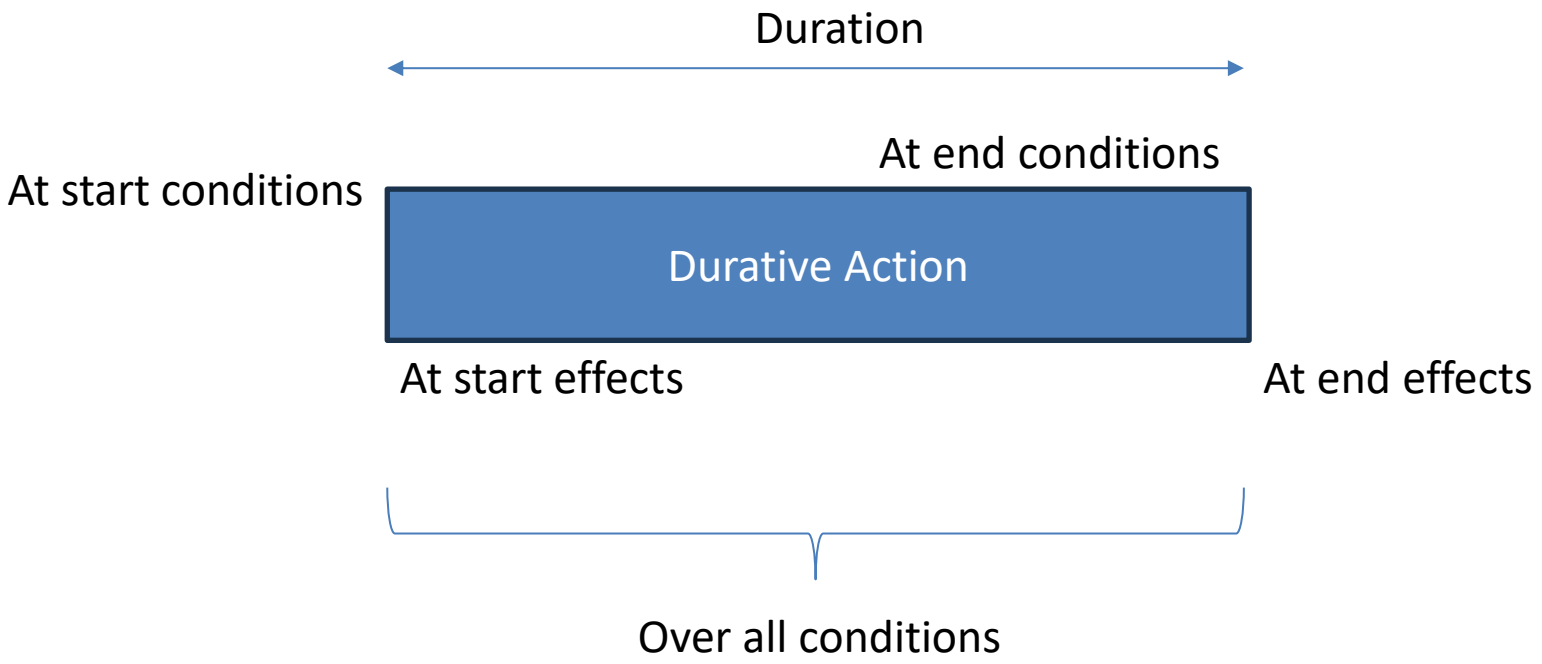
- Given:
  - A domain model (action templates – specify preconditions and effects)
  - An initial state (specifies the objects, their current conditions/states and the goal condition)
- Classically: precondition is a logical sentence over state variables; effect is a partial assignment to state variables – unassigned variables assumed unchanged (STRIPS assumption)



- A plan is a *sequence* of instantiation actions that applies from the initial state to generate a state satisfying the goal
- *Time is implicit only in the ordering of actions!*



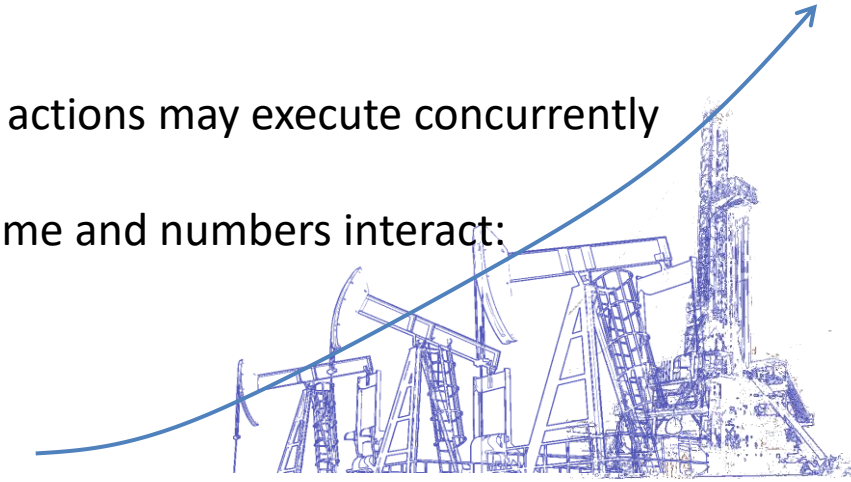
Temporal Planning



Temporal planning with Boolean state variables is only interestingly different if actions may execute concurrently

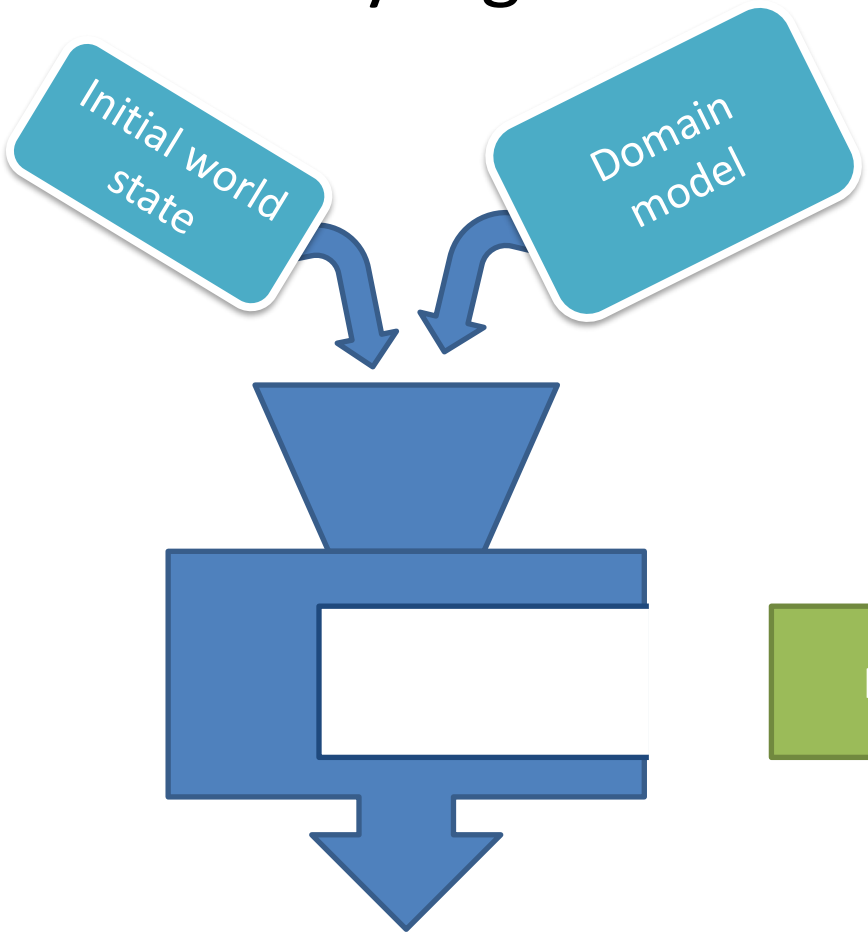
Numeric state variables are hard, but the temporal dimension changes when time and numbers interact:

- Variable durations that also change effects
- Continuous change



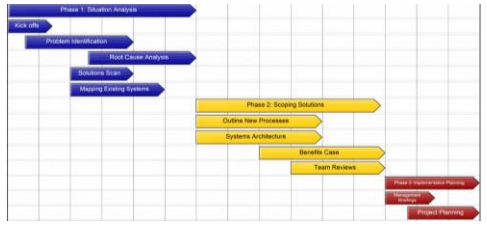
PDDL2. 1: An extension to PDDL for expressing temporal planning domains, M Fox, D Long, Journal of artificial intelligence research 20, 61-124 (2003)

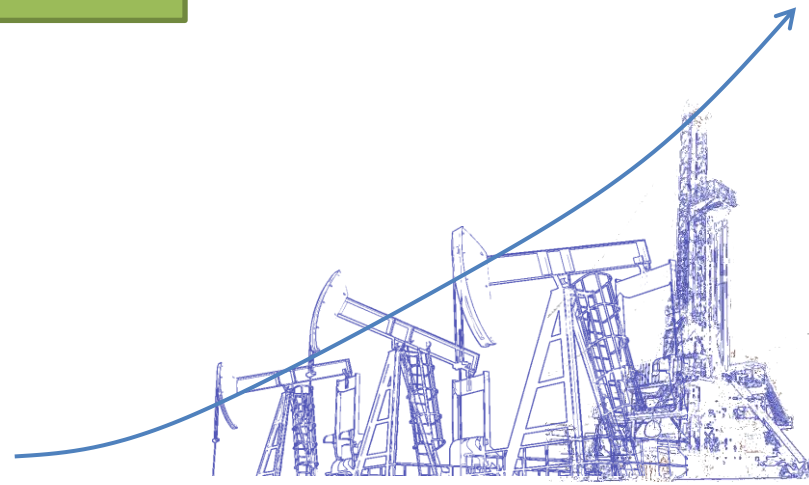
The key ingredients



Planner

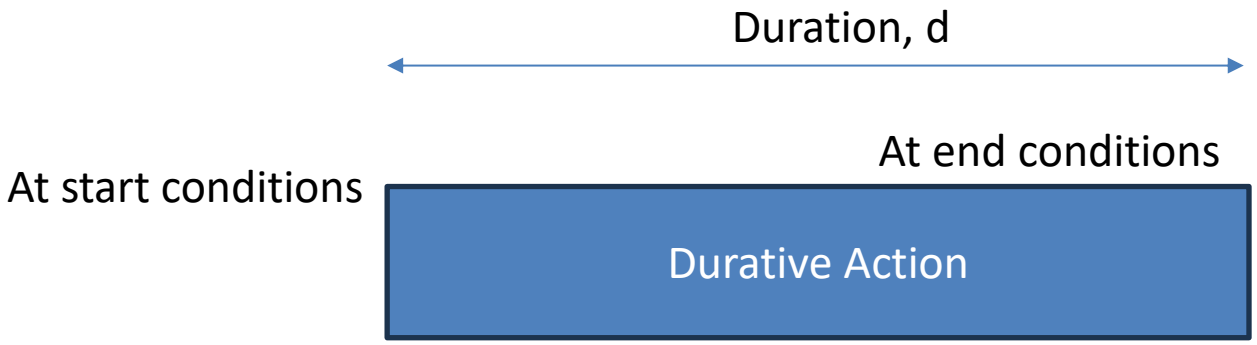
Plan





Capturing Continuous Change

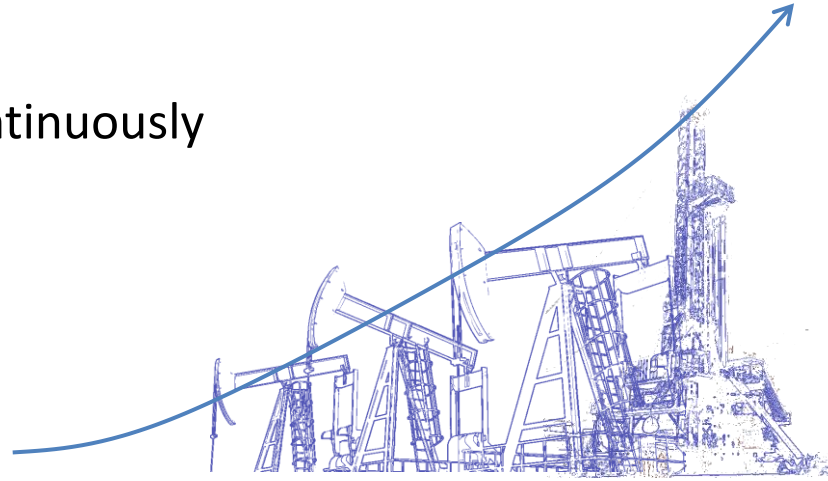
- Black-box: a process is running, but we only see the effect



Induces a step change in variable *x*
If duration is variable, then effect is variable

- Exposed: a process is within the durative action – we see its effect continuously

$$\frac{dx}{dt} = m$$



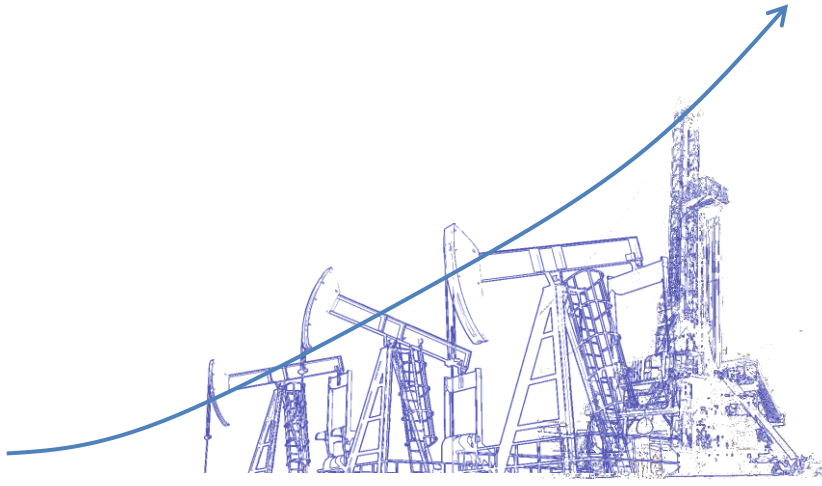
Modelling Continuous Effects in PDDL2.1

- The effect is not associated with start or end (it happens continuously!)
- We write:

  (increase x (m #t))*

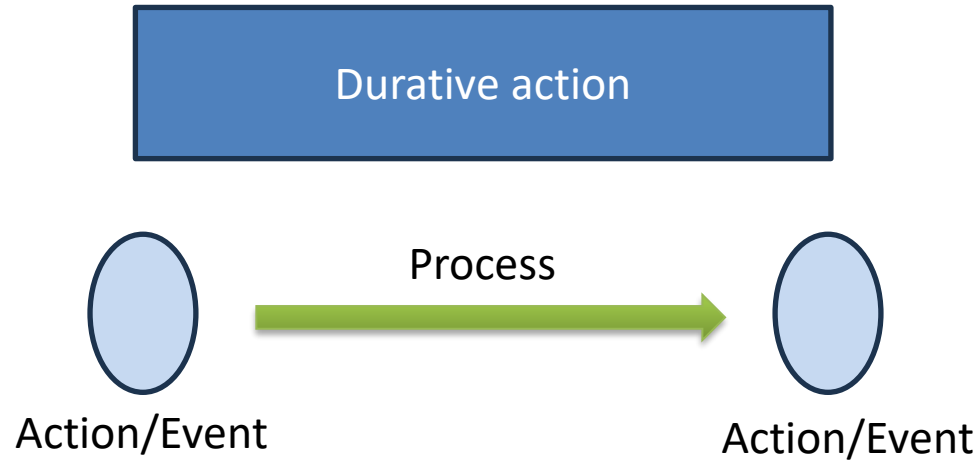
  Note could instead be a *decrease* effect

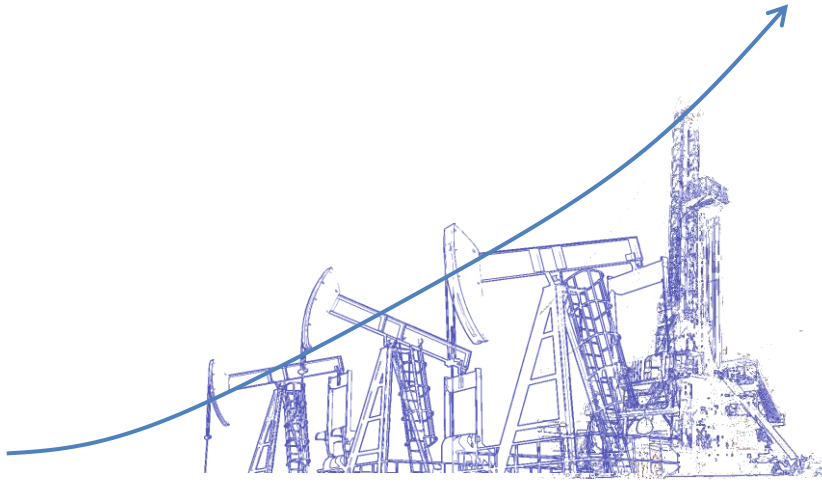
- Why not “dx/dt = m”?
  - Concurrent effects combine continuous contributions to change in x, so single action cannot specify the value of dx/dt, only its contribution to it



What happens in PDDL+?

- Executive agents perform *instantaneous 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



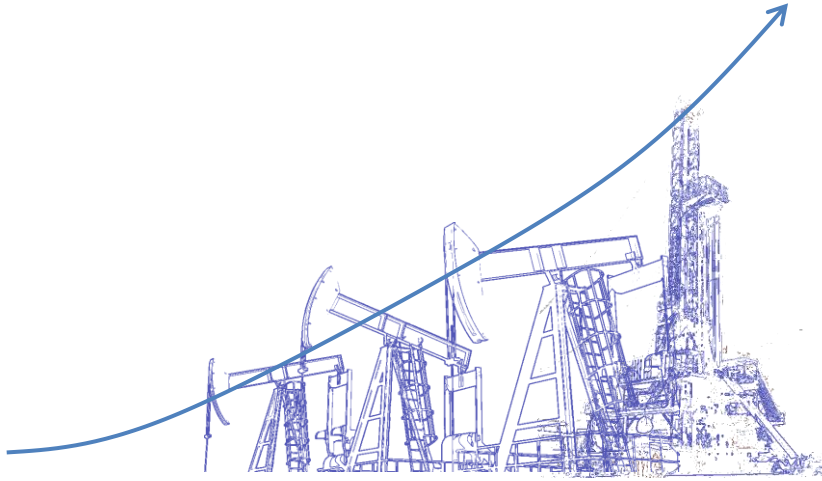


Modelling mixed discrete-continuous domains for planning, M Fox, D Long, Journal of Artificial Intelligence Research 27, 235-297 (2006)

Those action-and-change people

- A significant closely allied research community is that concerned with "Action and Change"
  - Axiomatisations of relationships between actions and change
  - Modelling of events
  - Reasoning about knowledge and what is known to different agents

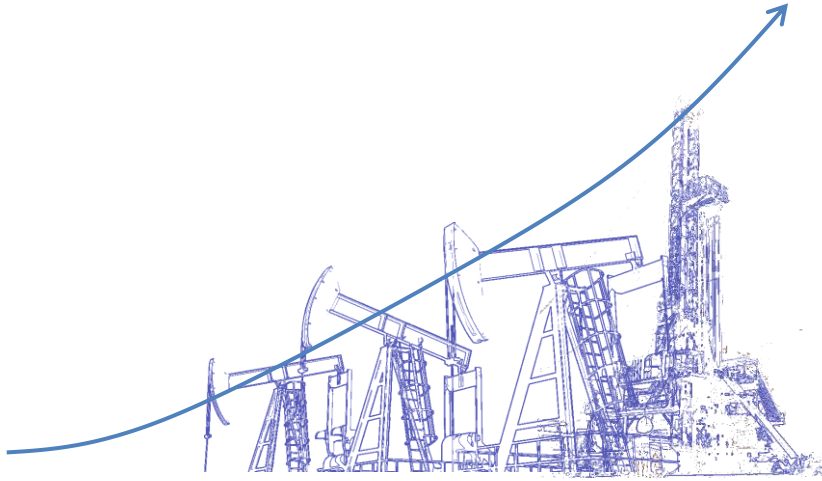
- Focus on logical formalisms underlying the reasoning about action and change (modal temporal logics; branching logics etc)



Those Hybrid-Systems-Modelling People

- Another key community concerned with reasoning about actions and world states is the controls and verification community
  - They are concerned with actions, events and processes
  - Worlds seen as transition systems (finite state hybrid automata)
  - Continuous change modelled as rate-of-change values within states

- Hybrid automata can be model-checked (subject to key constraints)
  - Usually concerned with checking all trajectories within some collection meet particular conditions (eg not violating safety constraint)



A Timed Hybrid Automaton

When the transition occurs, the effects update variables

While the state invariant remains true the system can reside in this state

Jump condition

Invariant

Jump effects

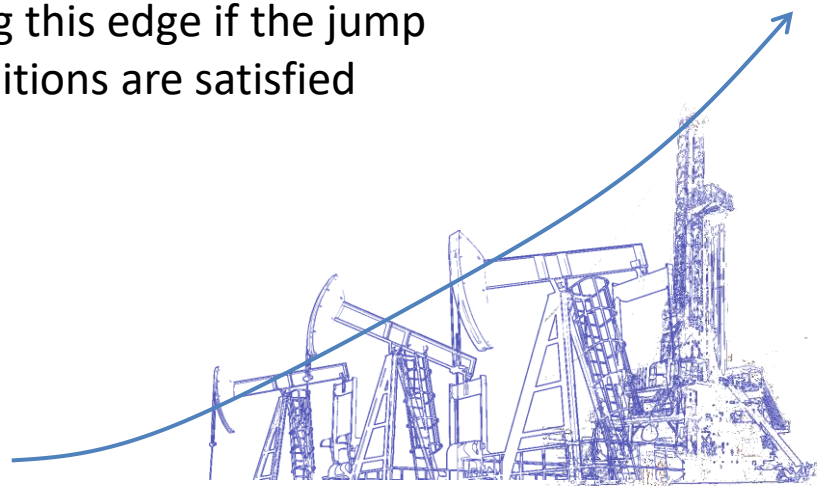
While in this state, the flow conditions are active (which specify rates of change for variables)

Flow conditions

State name

A transition is only possible along this edge if the jump conditions are satisfied

Hybrid automaton state

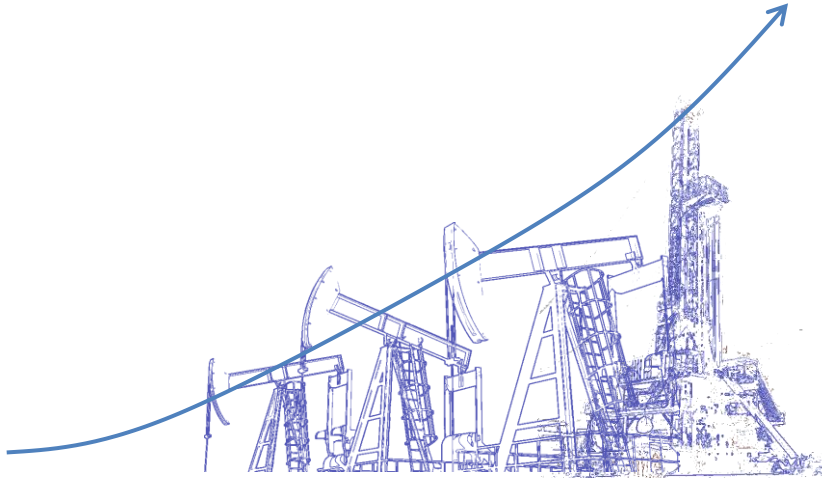


A theory of timed automata
R Alur, DL Dill
Theoretical computer science 126 (2), 183-235 (1994)

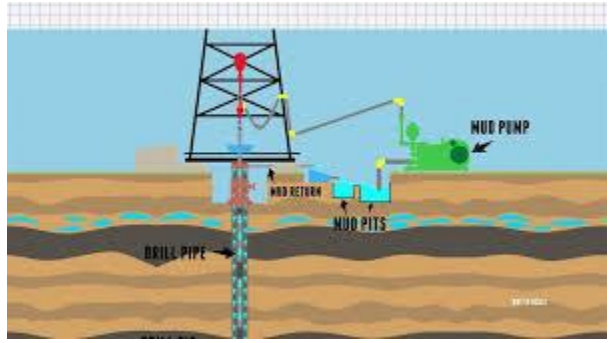
THA and PDDL+

- THAs express uncontrolled uncertainty using flow conditions that specify *ranges* of possible values for rates of change

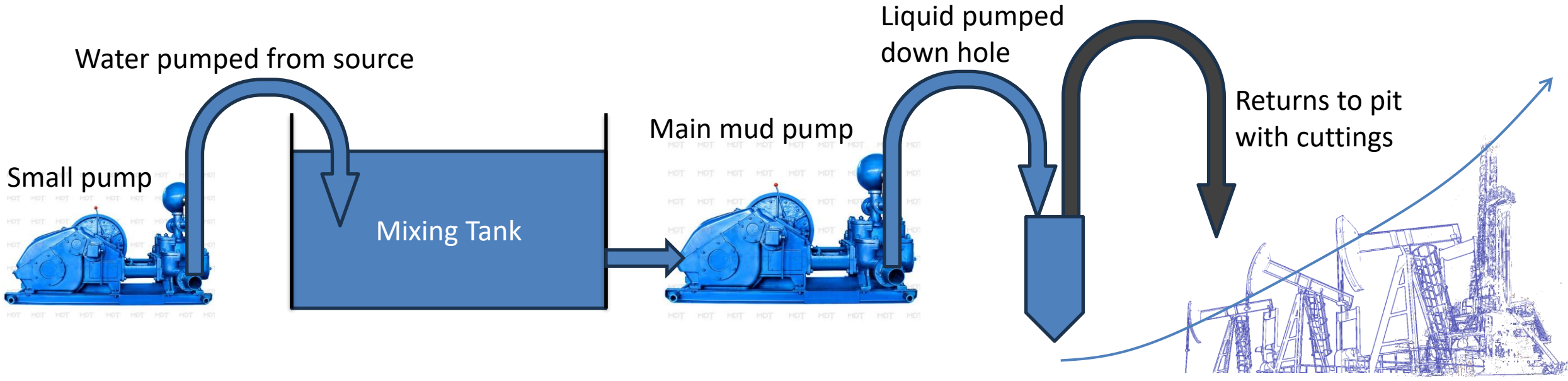
- Events and actions are not distinguished explicitly
  - Deterministic events are triggered when a state invariant is violated by the changes caused by the flow conditions in the same state
  - Other transitions could be non-deterministic events or actions



Simple Example

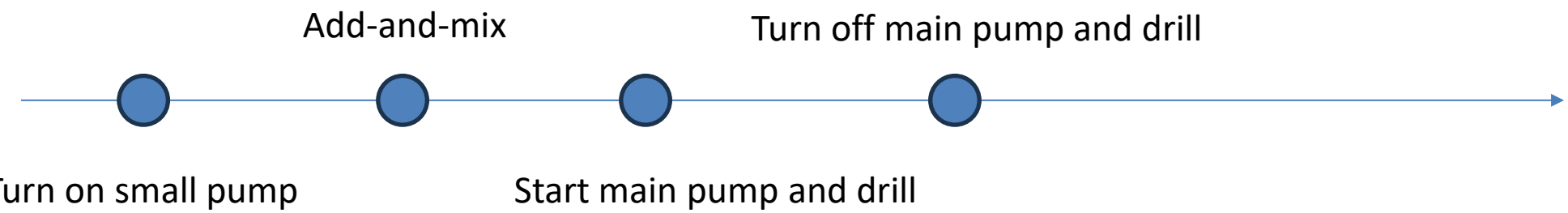


- When starting to drill a well, the first stage is called “spudding” and it is often drilled “open hole”
  - The hole is filled with fluid, pumped down the hole through the drill pipe, to carry the cuttings back up out of the hole
  - The liquid is pumped from a tank and the waste is pumped into a mud return pit
  - The liquid is a mix of water and some chemicals (eg anti-foaming agent) and has to be prepared in advance of drilling

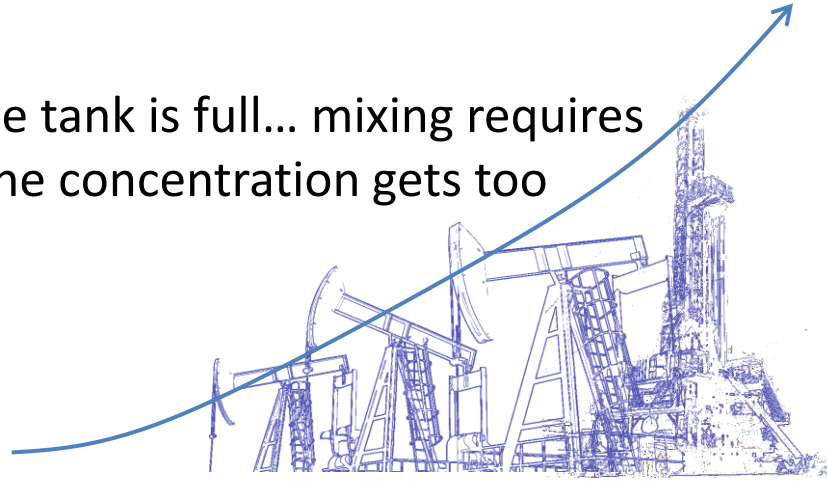


What's the Problem?

- The mud-engineer has to fill the tank with the slower small pump and mix the fluid
- The driller will turn on the main pump when they want to drill, consuming fluid from the tank
- What is the plan for drilling to a depth, d, say?

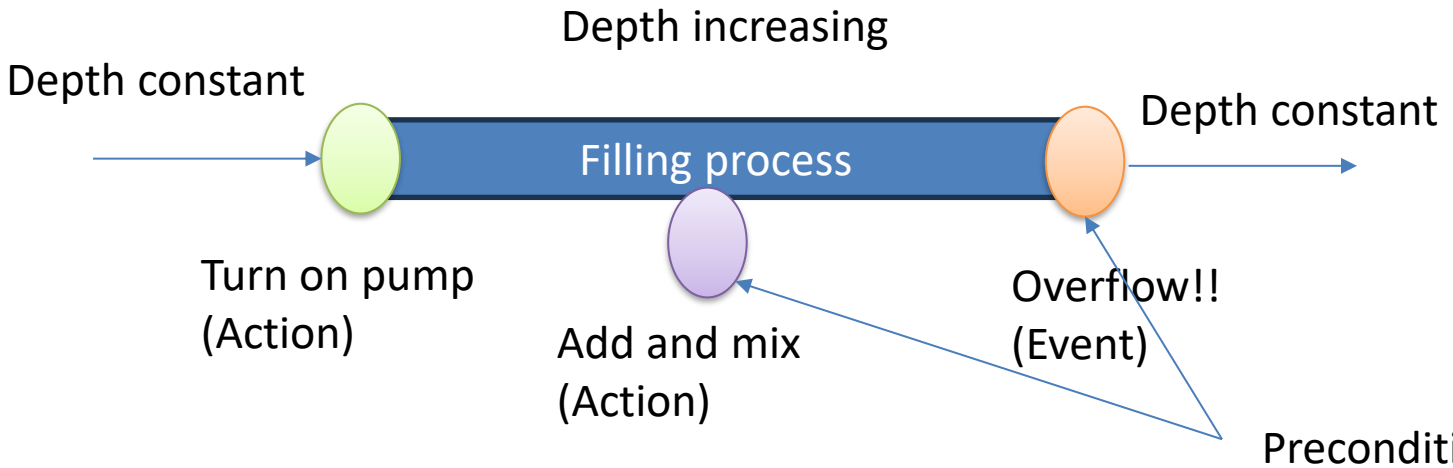


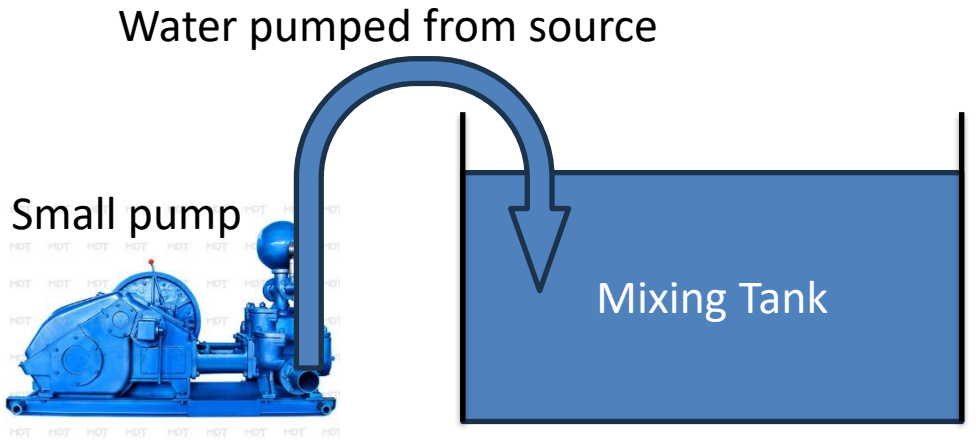
- Drilling must stop if the tank runs dry… the small pump must stop if the tank is full… mixing requires a minimum water level… once mixed, new additive must be added if the concentration gets too low…



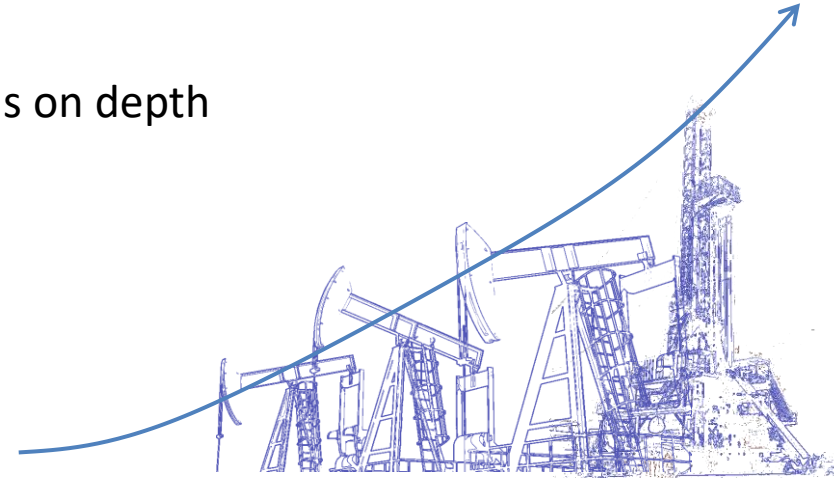
What makes it *hybrid*?

- 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

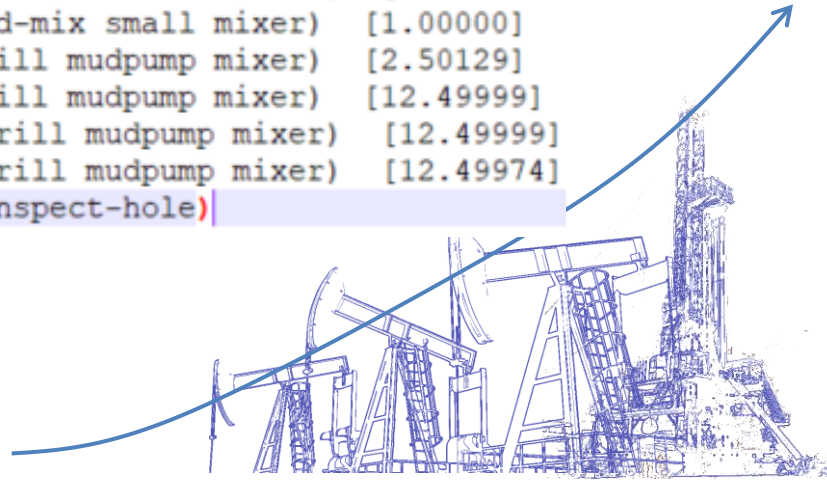


PDDL2.1 Tanks and Pumps

```
(:durative-action fill-tank
 :parameters (?p - pump ?t - tank)
 :duration (<= ?duration  1000)
 :condition (and (at start (not (is-on ?p)))
                 (at start (feeds ?p ?t))
                 (over all (< (tank-level ?t) (max-depth ?t)))
                 (over all (is-on ?p))
                 )
 :effect (and (increase (tank-level ?t) (* #t (pump-rate ?p)))
                 (at start (is-on ?p))
                 (at end (not (is-on ?p)))))

(:durative-action add-mix
 :parameters (?p - pump ?t - tank)
 :duration (= ?duration 1)
 :condition (and
         (at start (feeds ?p ?t))
         (at start (not (blended ?t)))
         (over all (is-on ?p))
          (over all (<= (tank-level ?t) (* (max-depth ?t) 0.75)))
          (over all (>= (tank-level ?t) (/ (max-depth ?t) 2))))
 :effect (and
            (at end (blended ?t))
           ))

(:durative-action drill
 :parameters (?p - pump ?t - tank)
 :duration (<= ?duration 1000)
 :condition (and (at start (feeds-hole ?p ?t))
                 (at start (>= (tank-level ?t) 80))
                     (over all (>= (tank-level ?t) 0))
                     (over all (blended ?t)))
 :effect (and (increase (hole-depth) (* #t (drill-rate)))
            (decrease (tank-level ?t) (* #t (pump-rate ?p)))))
```

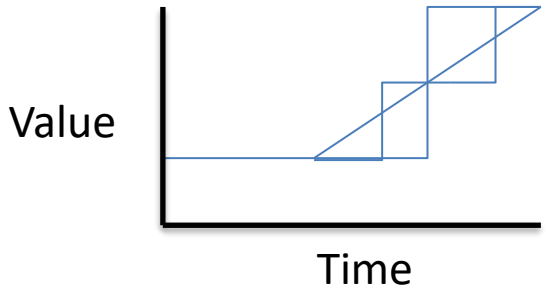
```
(define (problem fill-and-mix)
    (:domain mudtank)
    (:objects small mudpump - pump mixer - tank)
    (:init   (= (tank-level mixer)  0)
        (= (pump-rate small)  2)
        (= (max-depth mixer)  100)
            (= (hole-depth) 0)
            (= (pump-rate mudpump) 10)
            (= (drill-rate) 1)
        (feeds small mixer)
        (feeds-hole mudpump mixer)
        (= (target) 40)
    )
    (:goal (and (drilled))))
```

```
0.00000: (fill-tank small mixer)  [200.00500
25.00000: (add-mix small mixer)  [1.00000]
49.99995: (drill mudpump mixer)  [2.50129]
62.50639: (drill mudpump mixer)  [12.49999]
125.00632: (drill mudpump mixer)  [12.49999]
187.50626: (drill mudpump mixer)  [12.49974]
200.00500: (inspect-hole)
```



Wait! I still don't get it…

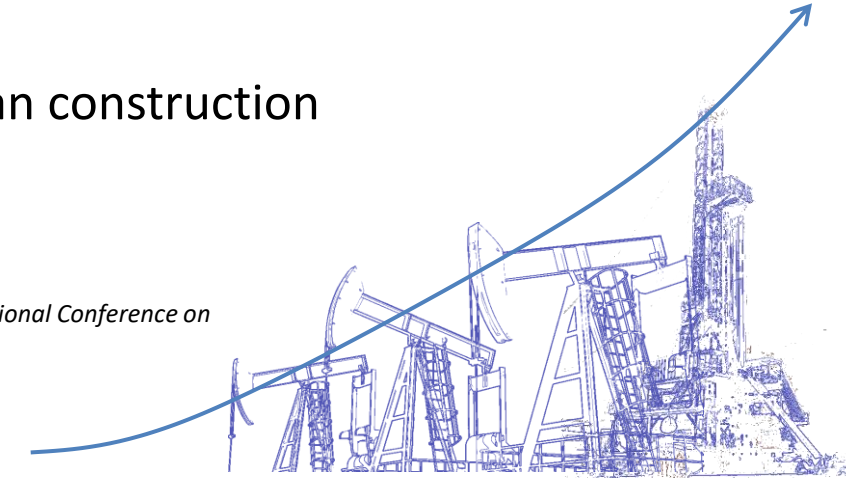
- Why not simply abstract continuous change: discretise it at fixed time points?



Step function discretisation

Refined step function discretisation

- Two problems with this idea:
  - Not always easy to predict what discretisation is necessary in order to find a plan without building the plan first…
  - Fine-grained discretisation leads to many choice points during plan construction



Della Penna, G., Magazzeni, D., Mercorio, F., & Intrigila, B. (2009). UPMurphi: A Tool for Universal Planning on PDDL+ Problems. *Proceedings of the International Conference on Automated Planning and Scheduling*, *19*(1), 106-113.
Heuristic Planning for PDDL+ Domains Wiktor Piotrowski,1 Maria Fox,1 Derek Long,1 Daniele Magazzeni,1 Fabio Mercorio (IJCAI 2016)
A Practical Approach to Discretised PDDL+ Problems by Translation to Numeric Planning
Francesco Percassi, Enrico Scala, Mauro Vallati (JAIR 76, 2023)

How can we work with this?

First, note that the process effect in this model is of the form:

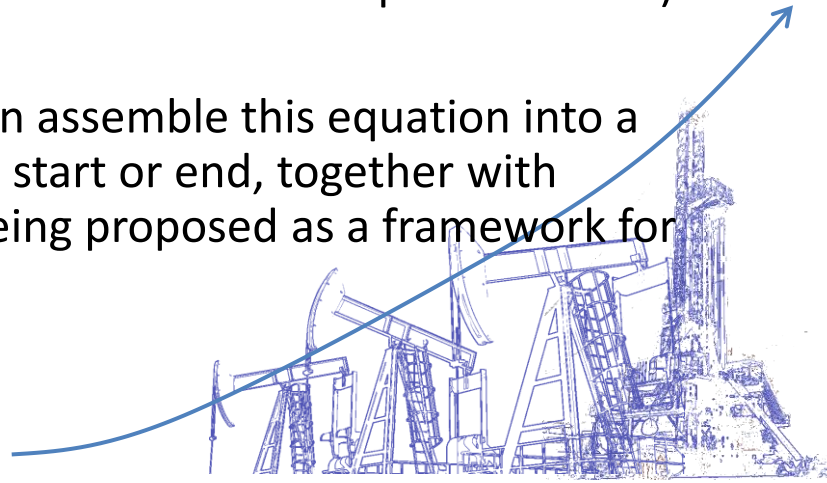
(increase (var) (<constant> #t))*

This can be interpreted as meaning: $\frac{d}{dt}v = m$ for a variable, *v*, and a constant, *m*

We solve this differential equation by integration: v(t) = mt + c (for some new constant, c)

We establish the value of c by considering the value of v at t = 0 (the initial value of v when the process starts)

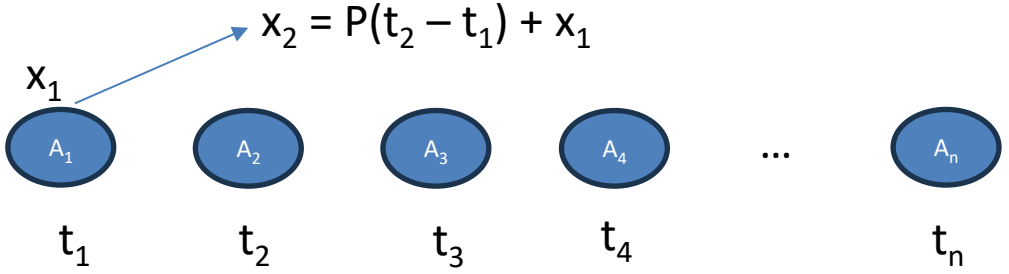
This gives us a *linear* equation determining the value of *v* at time *t* and we can assemble this equation into a linear program that captures the constraints on the times at which processes start or end, together with constraints on the values of the variable v, based on a collection of actions being proposed as a framework for a plan – a solution to the LP will then establish the actual plan



Forward-chaining partial-order planning
A Coles, A Coles, M Fox, D Long - Proceedings of the International Conference on Automated Planning and Scheduling 2010

Planning with Time and Processes

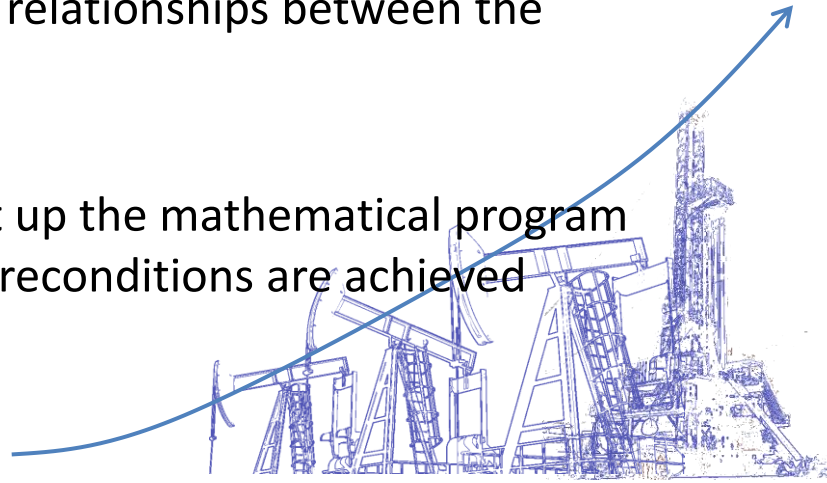
- A classical plan is a sequence of instantaneous actions



- A temporal plan is a classical plan in which the actions are also assigned an execution time

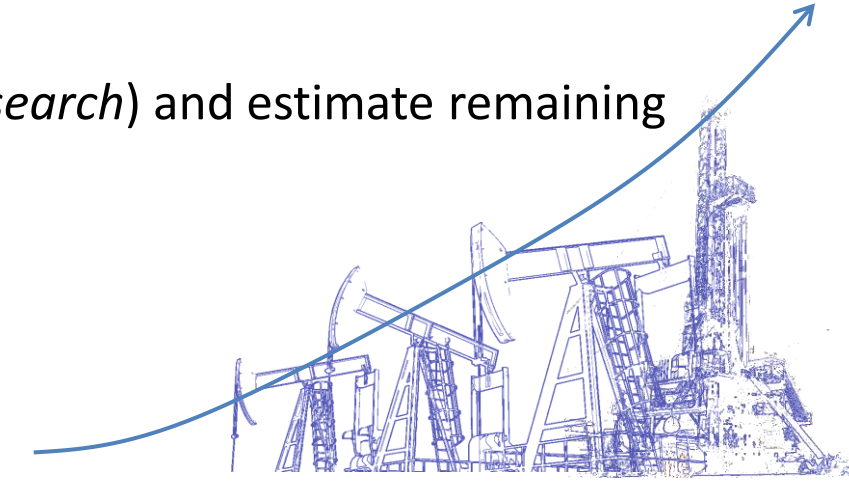
- A temporal plan with processes is a temporal plan with constraints on the relationships between the action times and numeric variables in the plan state

- So… we can search over classical plans that achieve the goals and then set up the mathematical program over action start times and the corresponding process effects, to ensure preconditions are achieved



Some efficiency improvements

- Avoid ordering time points where possible (*partial order structure*)
- Check constraints *incrementally* (don't build the whole classical plan before checking it)
- Keep the incremental solutions and use them as the starting point to check the next incremental constraint set (*warm start the solver*)
- Use bounds from the solution to prune the search space for the classical plan (*smart inference*)
- Use approximations of the process effects to suggest actions (*smart search*) and estimate remaining plan duration (*smart pruning*)



Extending the Model – Non-Linear Change

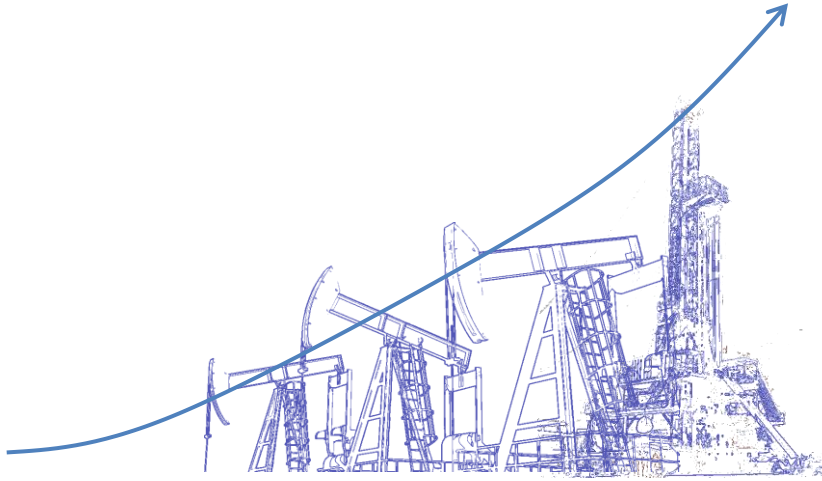
- If we consider the concentration of anti-foaming agent in the tank, we can add constraints:
  - No drilling if the concentration is lower than some minimum (wait until agent is added)
  - No adding agent if the concentration is higher than some level (wait until agent is diluted)

- Concentration c(t) can be taken to be proportional to the mass of agent in the tank, m(t), divided by the depth, d(t) (the cross sectional area is then part of the constant of proportionality)
  - We have: $c(t) = Km(t)/d(t)$

- The mass in the tank decreases as the content is pumped out, depending on the concentration:
  - We have: $dm/dt = -p_o\, c(t)$ where p_o is the output pump rate

- A bit of maths and we get:

  $dc/dt = -p_i\, c(t)/d(t)$ where p_i is the input pump rate



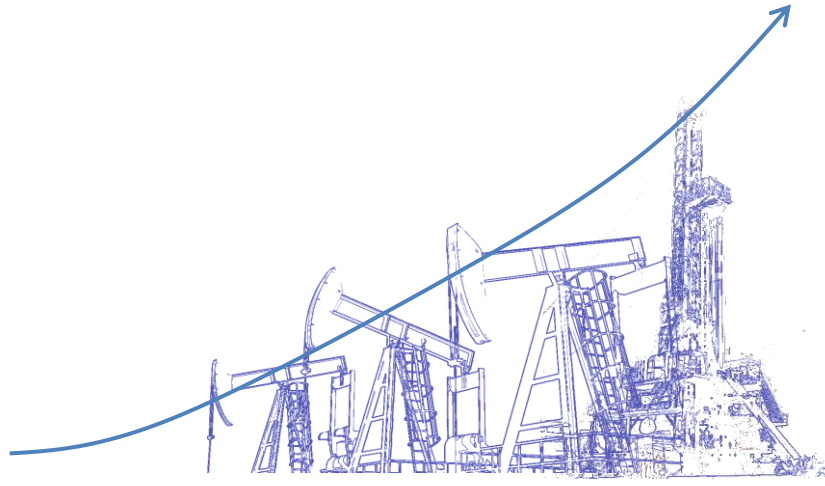
And in PDDL…

- But this is too hard for planners – and even for VAL!

```
(:durative-action fill-tank
 :parameters (?p - pump ?t - tank)
 :duration (<= ?duration  1000)
 :condition (and (at start (not (is-on ?p)))
                 (at start (feeds ?p ?t))
                 (over all (< (tank-level ?t) (max-depth ?t)))
                 (over all (is-on ?p))
                 )
 :effect (and (increase (tank-level ?t) (* #t (pump-rate ?p)))
              (decrease (concentration) (* #t (/ (* (pump-rate ?p) (concentration))
                                                 (tank-level ?t))))
                 (at start (is-on ?p))
                 (at end (not (is-on ?p)))))

(:durative-action add-mix
 :parameters (?p - pump ?t - tank)
 :duration (= ?duration 1)
 :condition (and
        (at start (<= (concentration) 0.1))
        (at start (feeds ?p ?t))
        ;(at start (not (blended ?t)))
        (over all (is-on ?p))
         (over all (<= (tank-level ?t) (* (max-depth ?t) 0.75)))
         (over all (>= (tank-level ?t) (/ (max-depth ?t) 2))))
 :effect (and
            ;(at end (blended ?t))
            (at end (assign (concentration) (/ 100 (tank-level ?t))))
          ))

(:durative-action drill
 :parameters (?p - pump ?t - tank)
 :duration (<= ?duration 1000)
 :condition (and (over all (>= (concentration) 0.2))
                 (at start (feeds-hole ?p ?t))
                 (at start (>= (tank-level ?t) 80))
                     (over all (>= (tank-level ?t) 0))
                     ;(over all (blended ?t))
                     )
 :effect (and (increase (hole-depth) (* (drill-rate) #t))
              (decrease (tank-level ?t) (* (pump-rate ?p) #t))))
```

Note the coupled effects imply coupled differential equations

Solution has d(t) linear in t, but c(t) in the form: $k(A\,t + b)^{-1/A}$ for constants A, b, k while drilling and filling concurrently

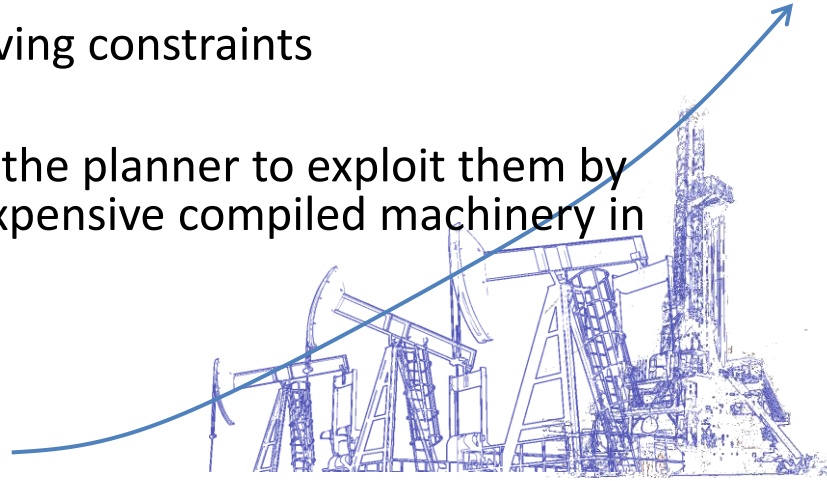


Time passes too fast!

- Things I did not have time to cover…

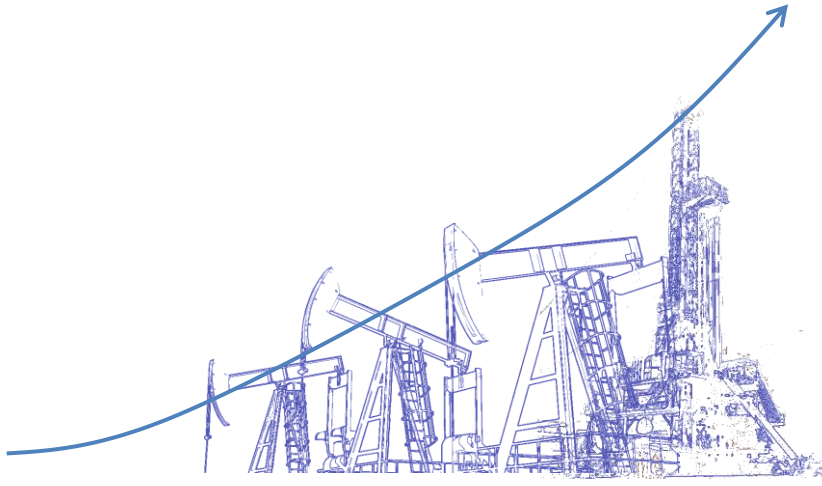
- Modelling and semantics:
  - Epsilon – the minimum time that the executive requires to react to an observed event and initiate an action, but also to model that no executive can execute actions *simultaneously and yet ordered*
  - How do we ensure that continuous change cannot lead to agents measuring time with greater precision than the epsilon suggests?
  - Time paradoxes for events – for the world, epsilon = 0? What happens if we have events that cascade infinitely often in a single instance?

- Planning with continuous change, processes and events:
  - Linear change – exploit a linear program solver to find time points solving constraints
  - Non-linear constraint solvers for *some* more complex constraints
  - Processes and events – “benign” versions can be handled by allowing the planner to exploit them by need… “harmful” effects have to be forced and this usually involves expensive compiled machinery in the domain model…

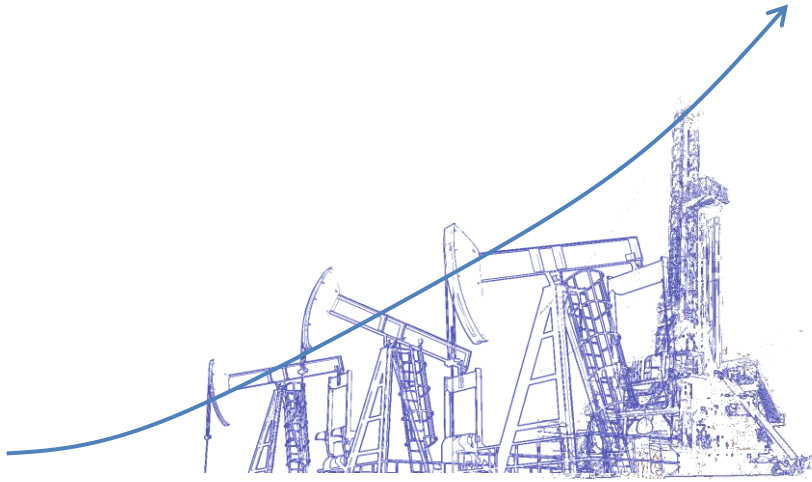


And this, too, shall pass

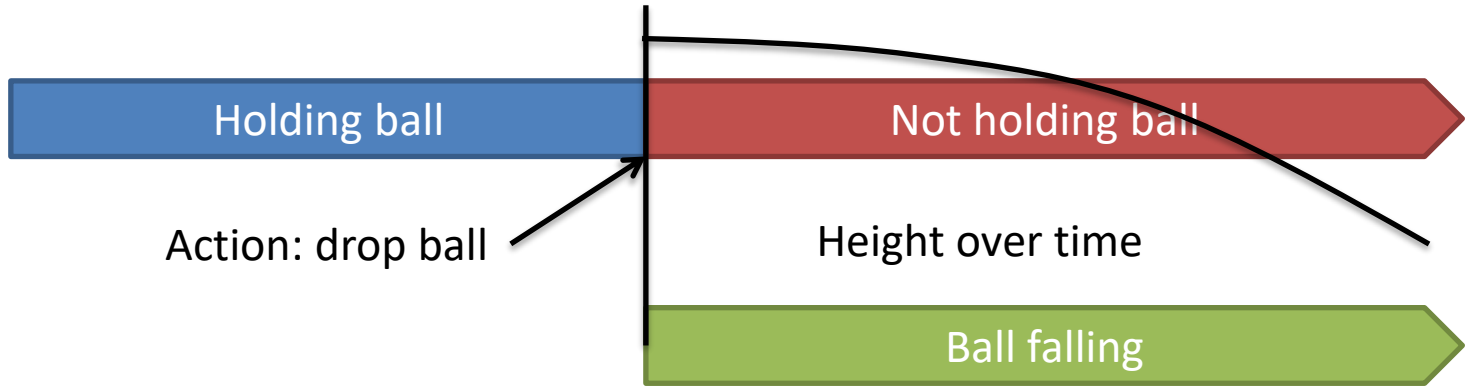
- Plans and execution:
  - How does an executive *use* a plan? Particularly one that models continuous change?
  - How do we resolve the differences between the model and reality – tiny changes/fluctuations in rates of change, measurement precision, observation accuracy…

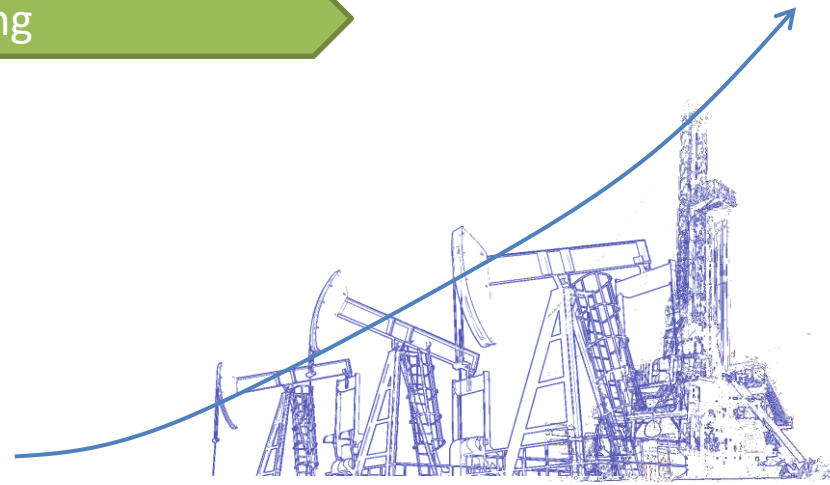


Spare slides…



What about non-linear?



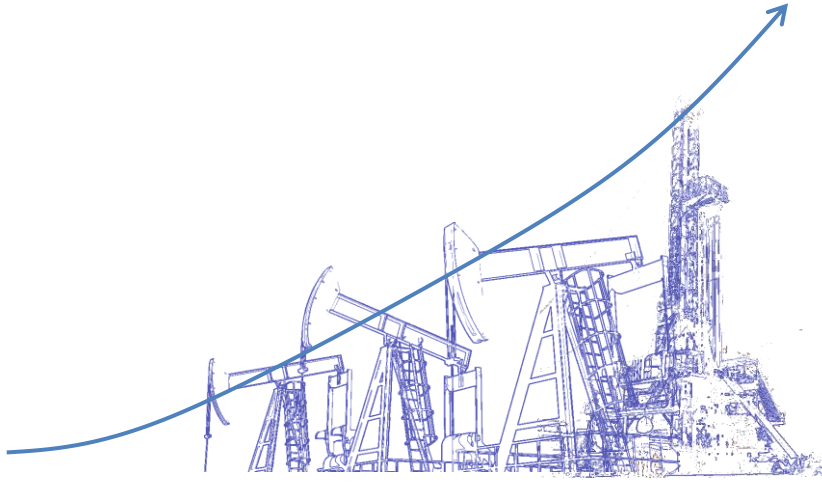


PDDL2.1: Where the planning meets the metal

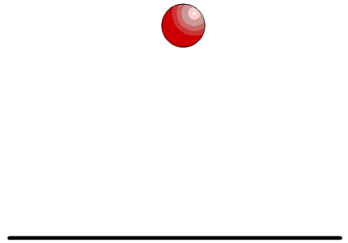
- PDDL2.1 adds durative actions to PDDL
  - Keep in mind: durative actions can include continuous change effects and variable durations

```
(:durative-action drop-ball
 :parameters (?b - ball)
 :duration (> ?duration 0)
 :condition (and (at start (holding ?b)) (at start (= (velocity ?b) 0)))
 :effect (and (at start (not (holding ?b)))
                (decrease (height ?b) (* #t (velocity ?b)))
                (increase (velocity ?b) (* #t (gravity)))))
```

Notice how the two continuous effects interact



PDDL2.1: Dropping the ball

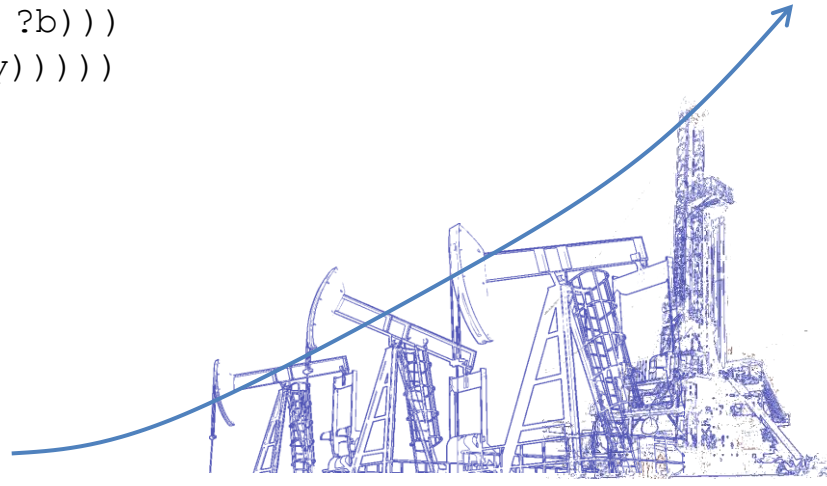


- What happens if there is a floor?

- Duration of the drop-ball action is limited by the distance to the floor...

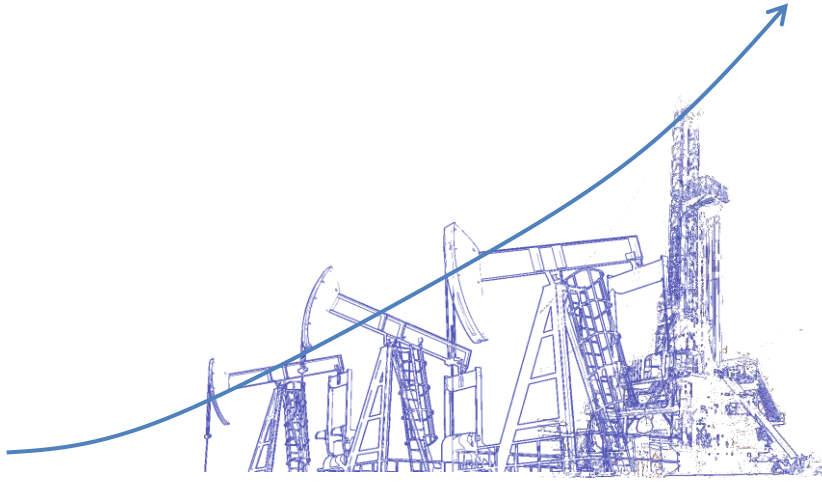
```
(:durative-action drop-ball
 :parameters (?b - ball)
 :duration (> ?duration 0)
 :condition (and (at start (holding ?b)) (at start (= (velocity ?b) 0))
                 (over all (>= (height ?b) 0)))
 :effect (and (at start (not (holding ?b)))
                 (decrease (height ?b) (* #t (velocity ?b)))
                 (increase (velocity ?b) (* #t (gravity)))))
```

But what happens if we leave the ball to fall?



PDDL+: Let’s see it bounce

- A “better” model is to see releasing the ball as separated from the fate of the ball after it falls
  - Release initiates a process of falling
- The falling process can be terminated by various possible actions (catching, hitting, ...) or events (bouncing)
- How does this look in PDDL+



PDDL+: Let it go

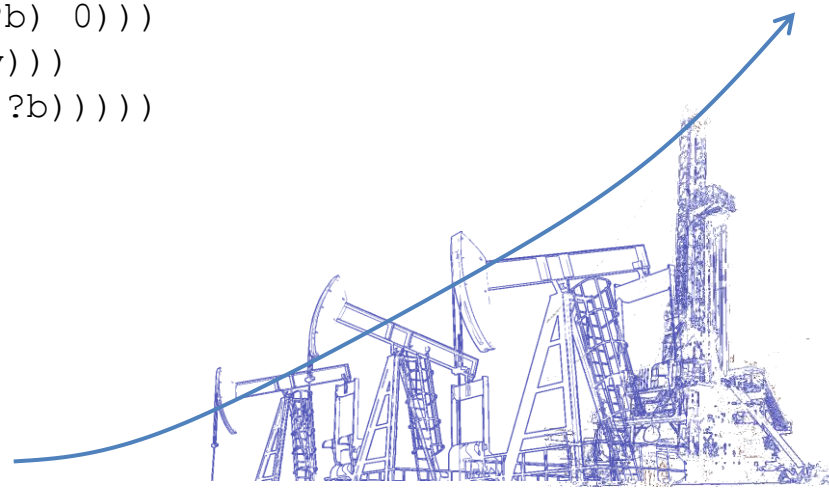
- First drop it...

```
(:action release
 :parameters (?b - ball)
 :precondition (and (holding ?b) (= (velocity ?b) 0))
 :effect (and (not (holding ?b))))
```

- Then watch it fall...

```
(: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)))))
```

- And then?



PDDL+: See it bounce

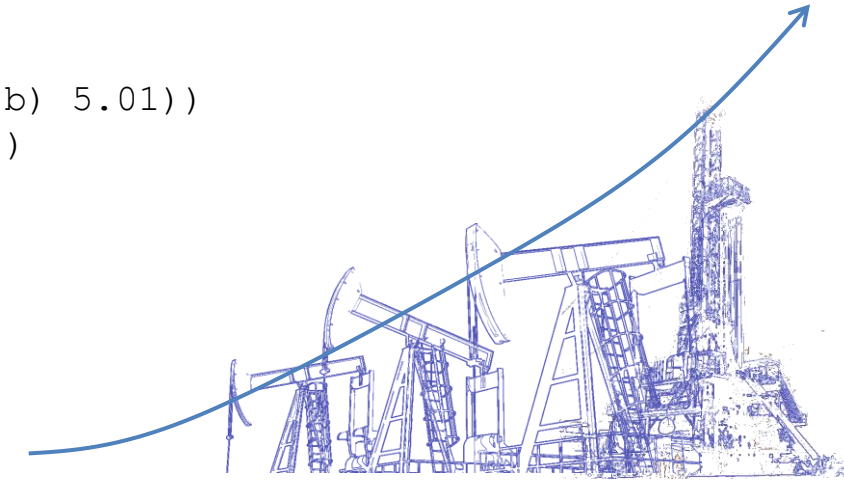
- Bouncing...

Note the slight fudge here

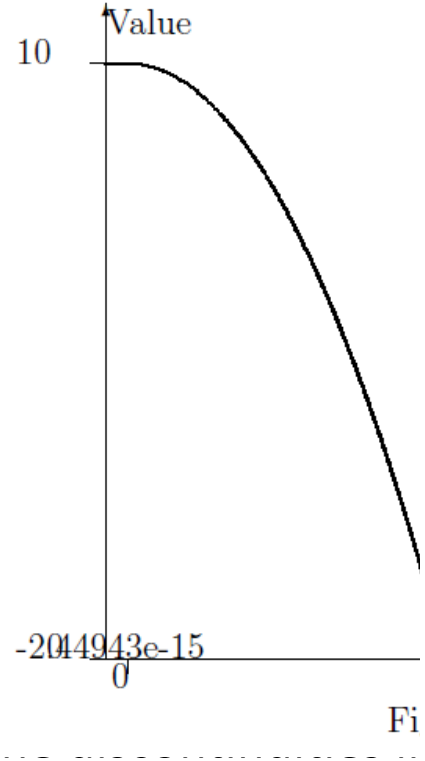
```
(:event bounce
 :parameters (?b - ball)
 :precondition (and  (>= (velocity ?b) 0)
                     (<= (height ?b) 0))
 :effect (and (assign (height ?b) (* -1 (height ?b)))
              (assign (velocity ?b) (* -1 (velocity ?b)))))
```

- Now let's plan to catch it...

```
(:action catch
 :parameters (?b - ball)
 :precondition (and (>= (height ?b) 5) (<= (height ?b) 5.01))
 :effect (and (holding ?b) (assign (velocity ?b) 0)))
```



- Let it
- Wha
- Note



1.51421: **Event triggered!**
Triggered event (bounce b1)
Unactivated process (fall b1)
Updating **(height b1)** (-2.22045e-15) by 2.22045e-15 assignment.
Updating **(velocity b1)** (14.1421) by -14.1421 assignment.

1.51421: **Event triggered!**
Activated process (fall b1)

4.34264: Checking Happening... ...OK!
(height b1)$(t) = -5t^2 + 14.1421t + 2.22045e - 15$
(velocity b1)$(t) = 10t - 14.1421$
Updating **(height b1)** (2.22045e-15) by -2.44943e-15 for continuous update.
Updating **(velocity b1)** (-14.1421) by 14.1421 for continuous update.

4.34264: **Event triggered!**
Triggered event (bounce b1)
Unactivated process (fall b1)
Updating **(height b1)** (-2.44943e-15) by 2.44943e-15 assignment.
Updating **(velocity b1)** (14.1421) by -14.1421 assignment.

4.34264: **Event triggered!**
Activated process (fall b1)

4.757: Checking Happening... ...OK!
(height b1)$(t) = -5t^2 + 14.1421t + 2.44943e - 15$
(velocity b1)$(t) = 10t - 14.1421$
Updating
update.
Updating
update.

4.757: Checking
Adding (
Updating

4.757: **Event t**
Unactiva

Plan executed succe

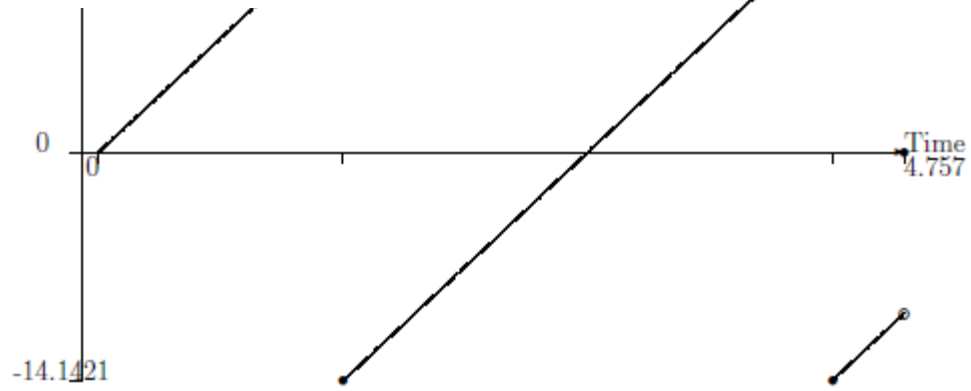
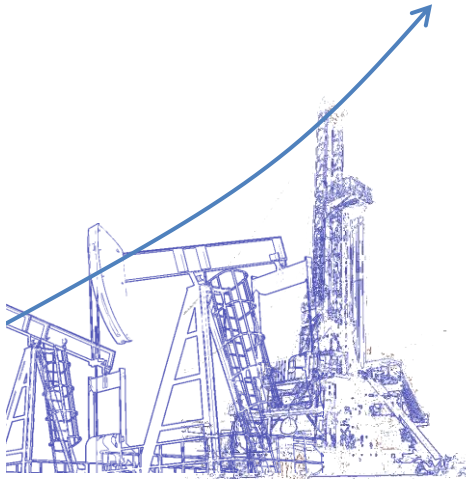


Figure 2.2: Graph of (velocity b1).



What happens if the ball is not perfectly elastic?

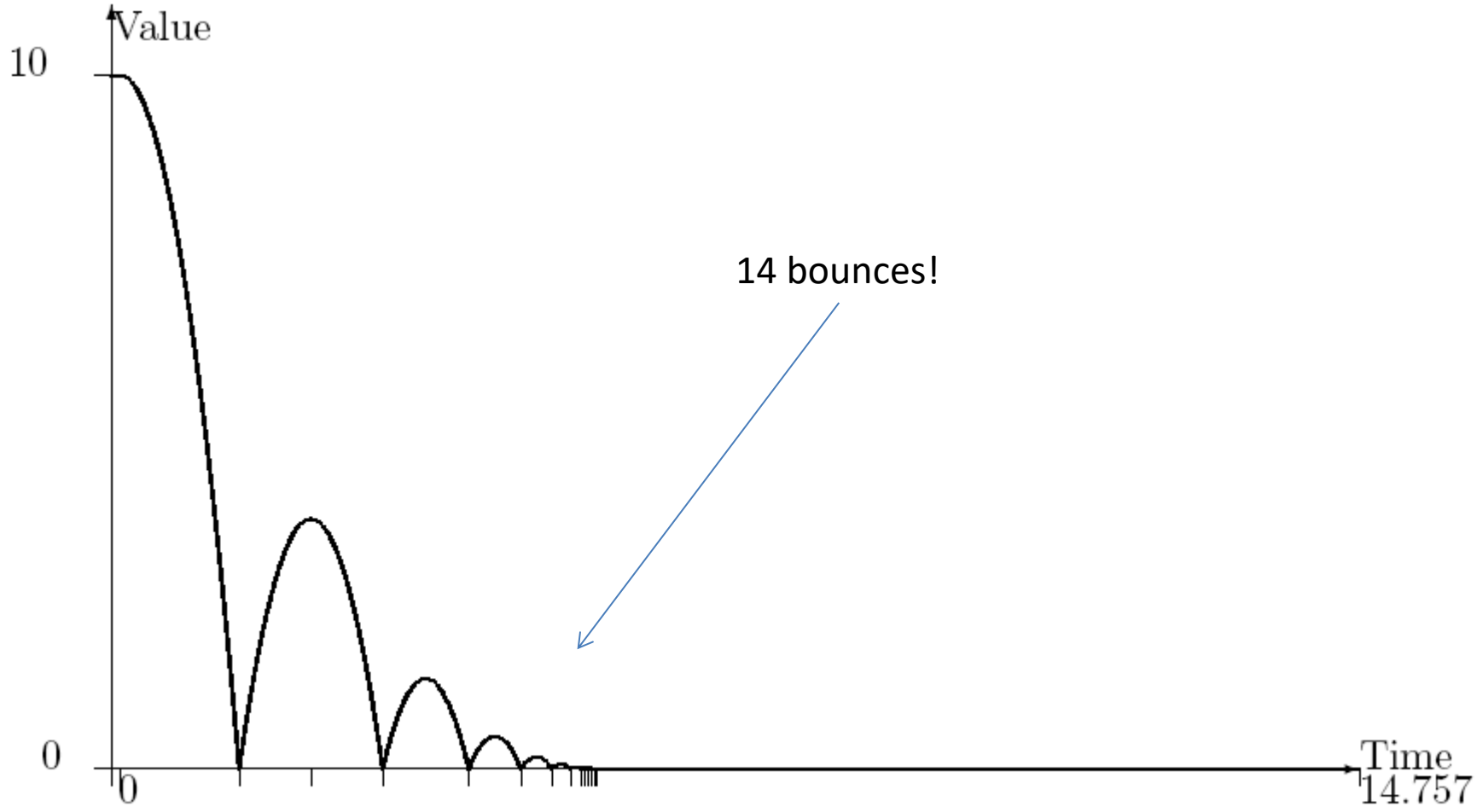


Figure 2.1: Graph of (height b1).

Zeno behaviour

- If we look c … an idealised view, at lea…
- This is Zenc…
  - Cannot … anywhere else…
- Check the \…

5.74146: Checking Happening... ...OK!
(**height b1**)$(t) = -5t^2 + 0.0513073t + 1e - 08$
(**velocity b1**)$(t) = 10t - 0.0513073$
Updating (**height b1**) (1e-08) by 1e-08 for continuous update.
Updating (**velocity b1**) (-0.0513073) by 0.0513073 for continuous update.

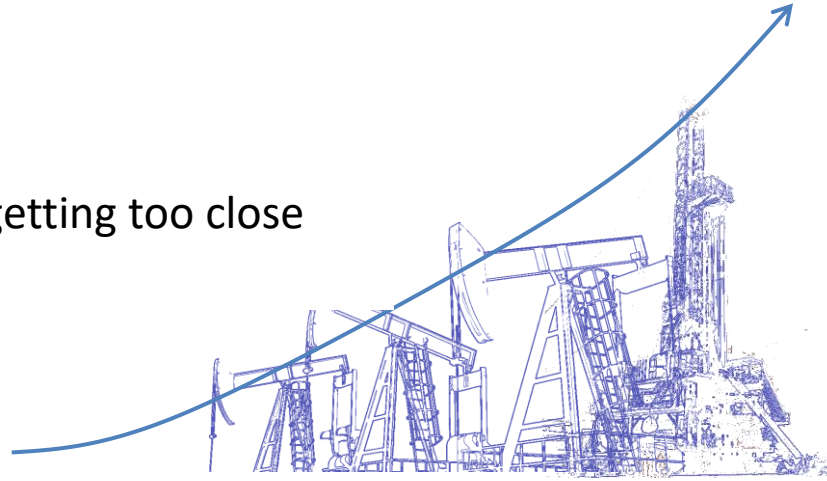
5.74146: **Event trigger…**
Triggered eve…
Updating (**velo…**.0513073) by -0.0307844 assignment.

...Error!

Attempt to trigger event (bounce b1) twice

Plan failed to execute

This is where we see events getting too close

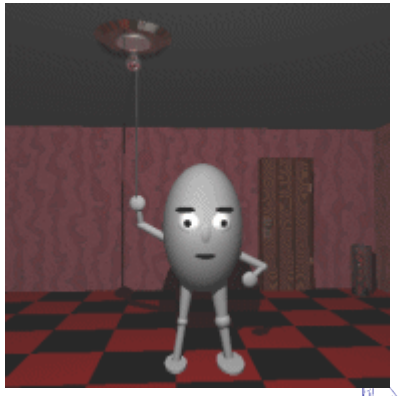


Cascading events

- Events (and processes) can be problematic in lots of interesting ways

- It is easy to set up events that trigger each other:
  - Imagine a light connected to a light-sensitive switch, so that when the light is on it triggers an event which turns the light off...
  - But, when the light is off an event [illegible] that turns it on!

- What is the best model?
  - Processes measuring time between switches?
  - Other states and events?

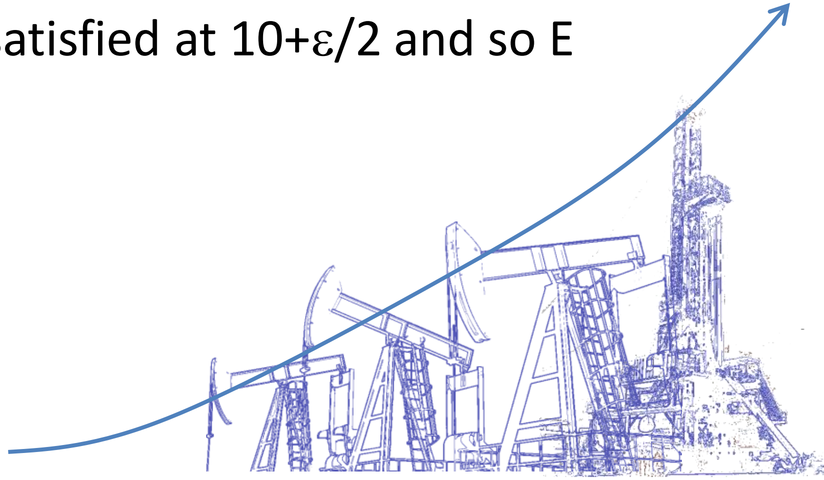


Other Semantic Issues

- Events with strict inequality preconditions can create an interesting problem

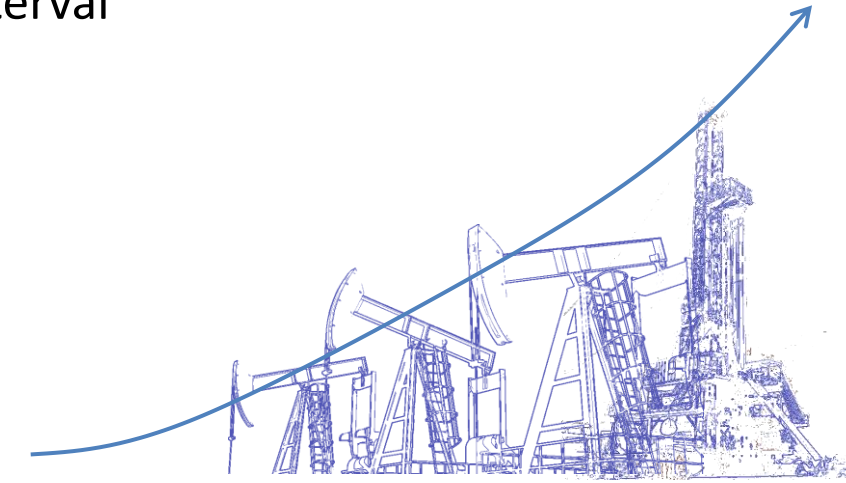
```
(:event E
  :precondition (> 10 (x))
  :effect (and (not (active)) (explode)))

(:process
  :precondition (active)
  :effect (increase (x) (* #t 1)))

:init (= (x) 0) (active)
```

- When does E occur?
  - At time 10, (x) = 10, so the precondition of E is not satisfied
  - At time 10+ε, for any ε>0, the precondition is satisfied, but it was satisfied at 10+ε/2 and so E should trigger before 10+ε



PDDL+ is expressive!

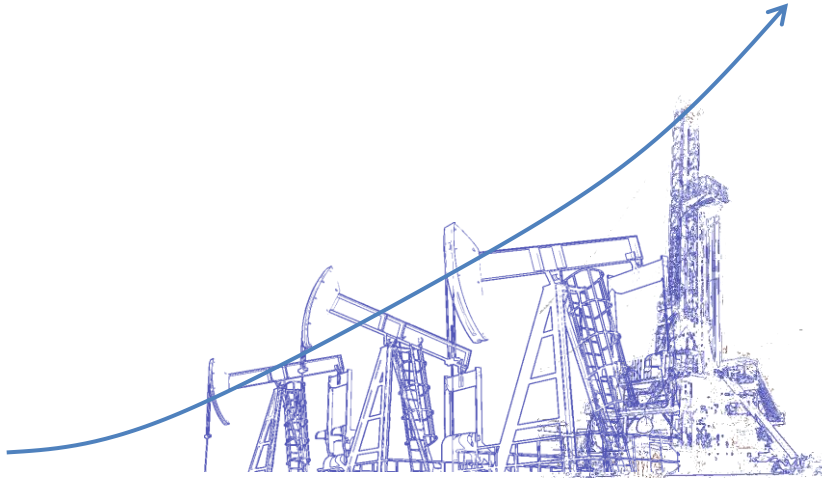
- PDDL+ is strictly more expressive than PDDL2.1
- This is because of events, which mean that plan validation is undecidable, which is not the case for (propositional) PDDL2.1
- Adding continuous effects can make PDDL2.1 plan validation undecidable if the behaviours are particularly unpleasant
  - Functions with many turning points in an interval over which an invariant condition must hold will require that all the zeros of a function be found within the interval



We can say so much, but what can we do?

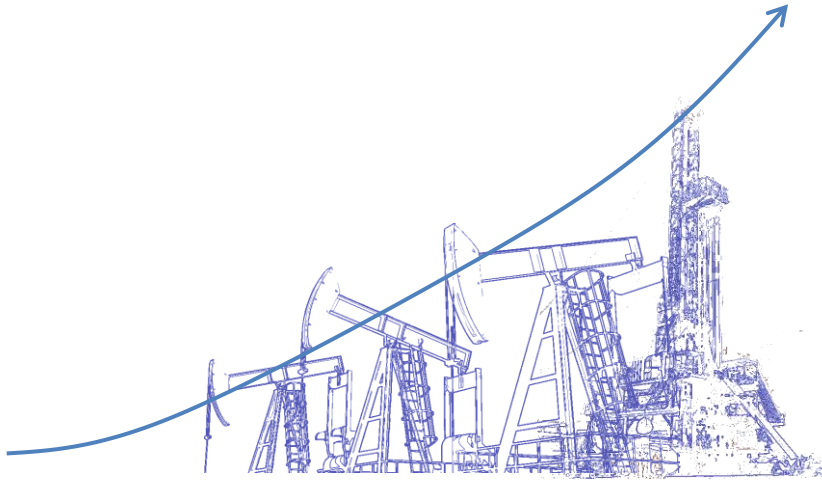
- Planning with *linear* processes is possible
  - Exploit linear programming to do the reasoning about process effects

- Observe that plans define a sequence of *happenings* which are actions, events or process triggers, each at specific time points
  - The time points need not be evenly spaced, but a plan can be built by deciding *how* effects should be achieved (or managed) and *when* the actions used to do this should be executed



The Things We Need (for Forward State Space search)

- State progression when there are active processes and invariants
- A heuristic to select good actions when we search for a plan (forward state space search)
- However, if we choose *when* to execute an action too quickly, we face an infinite choice parameter:
  - Delay commitment to actual times, but simply commit to order
  - Use the LP to help us to narrow down choices of when to do things until the choice is removed, or the plan is complete



Another simple example

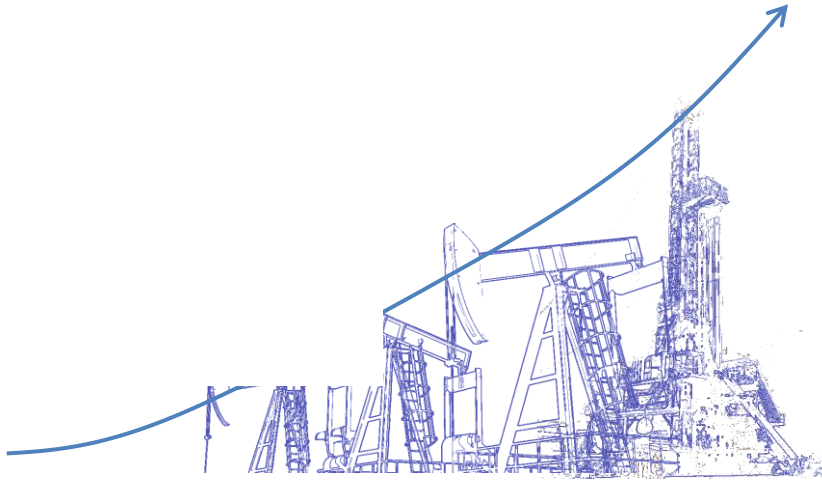
- A borrower/saver wishes to buy a house and pay off the mortgage
  - Can save a deposit and then, depending on how much is saved, select between different mortgages
  - Objective is to eventually take a “life audit” and confirm debt is paid and home is owned

Domain model

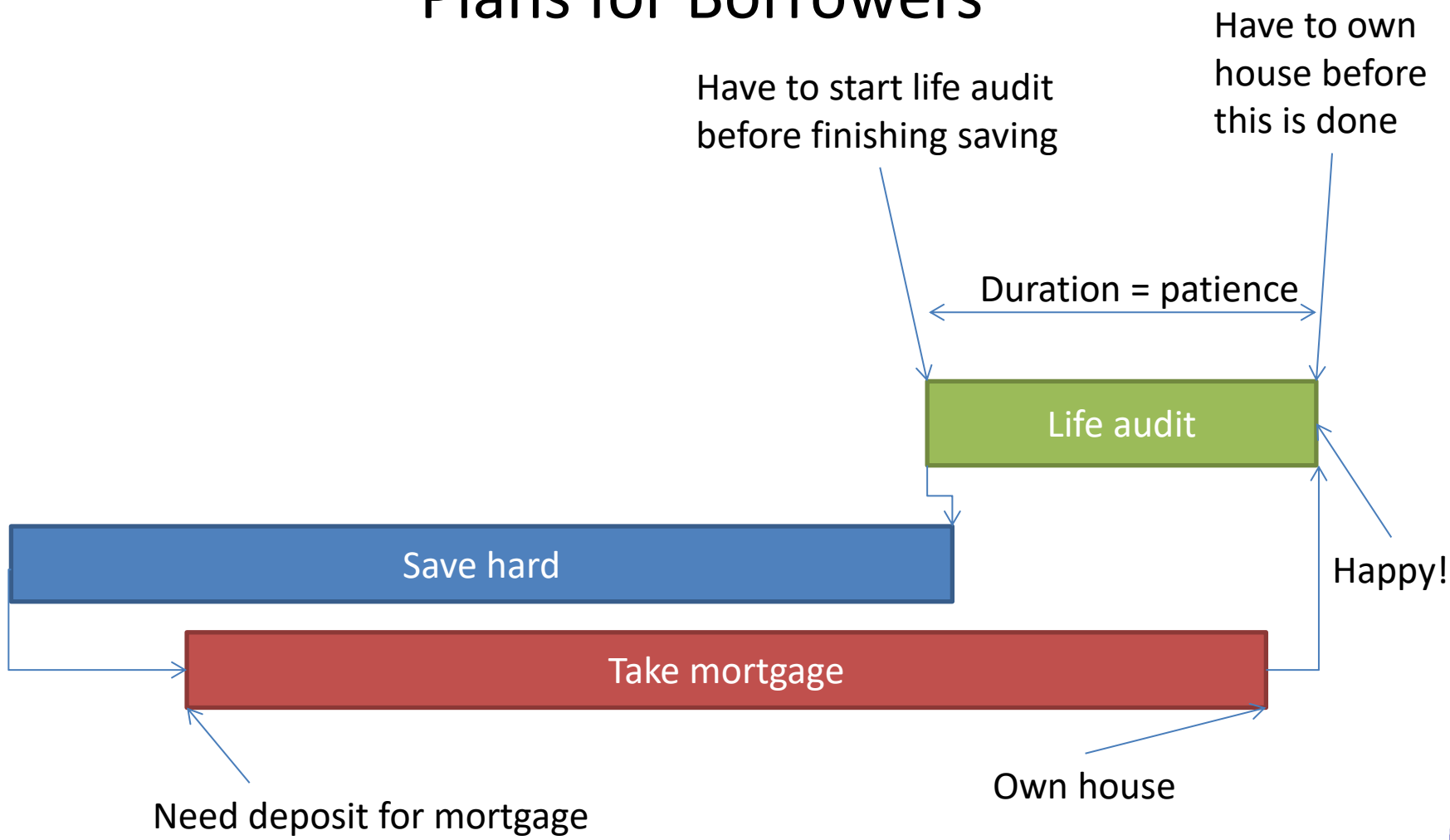
```
(:durative-action saveHard
 :parameters ()
 :duration (= ?duration 10)
 :condition
    (and (at start (canSave))
         (over all (>= (money) 0)))
 :effect
    (and (at start (not (canSave)))
         (at end (canSave))
         (at start (saving))
         (at end (not (saving)))
         (increase (money) (* #t 1))))
```

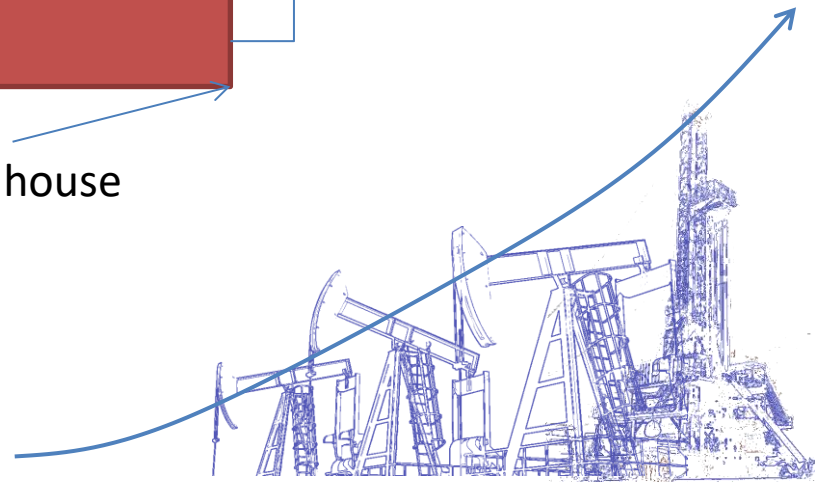
```
(:durative-action lifeAudit
 :parameters ()
 :duration (= ?duration (patience))
 :condition
    (and (at start (saving))
         (at end (boughtHouse))
         (at end (>= (money) 0)))
 :effect (and (at end (happy))))
```

```
(:durative-action takeMortgage
 :parameters (?m - mortgage)
 :duration (= ?duration (durationFor ?m))
 :condition
    (and (at start (saving))
         (at start (>= (money) (depositFor ?m)))
         (over all (<= (money) (maxSavings ?m))))
 :effect
    (and (at start (decrease (money) (depositFor ?m)))
         (decrease (money) (* #t (interestRateFor ?m)))
         (at end (boughtHouse))))
```

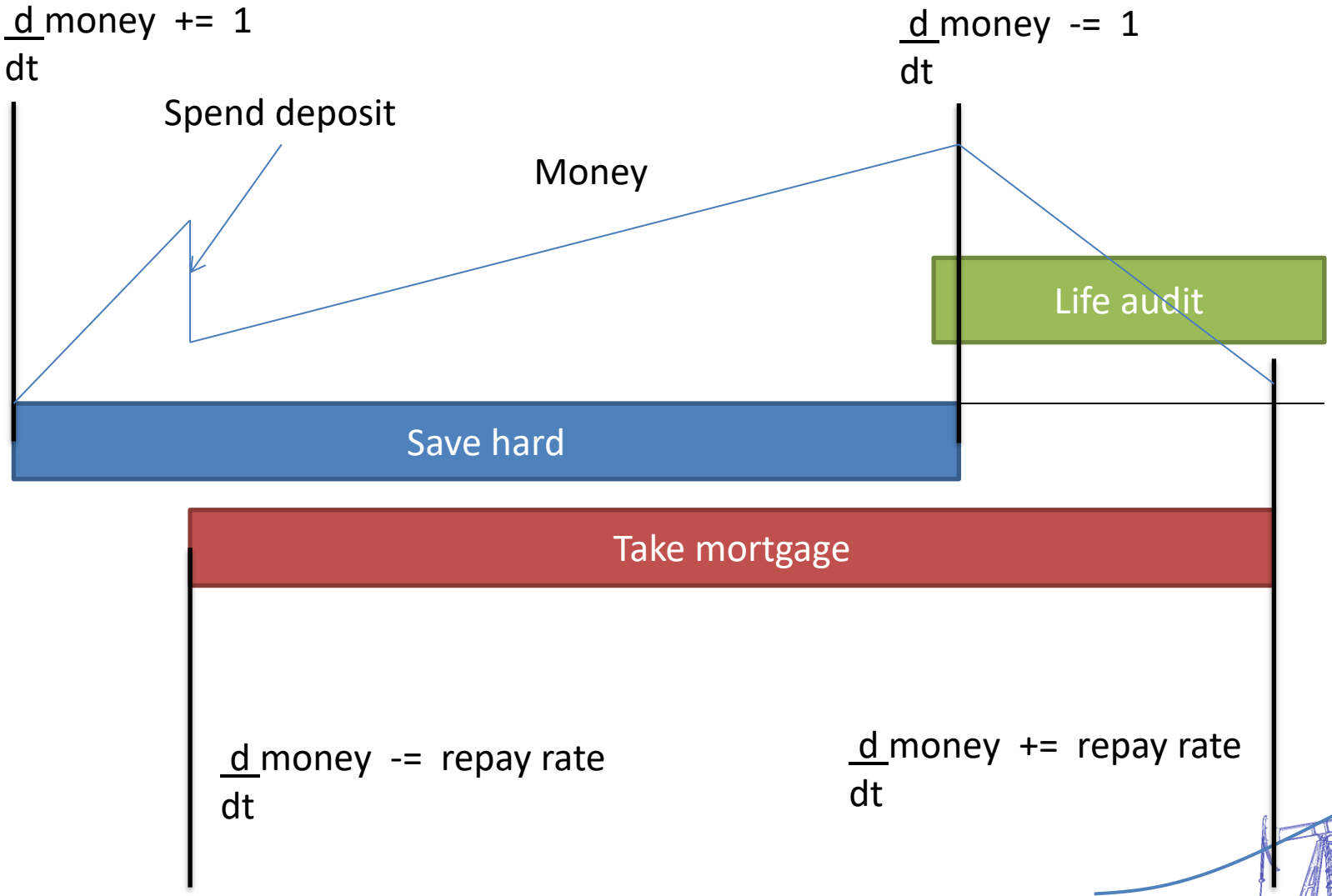


Plans for Borrowers

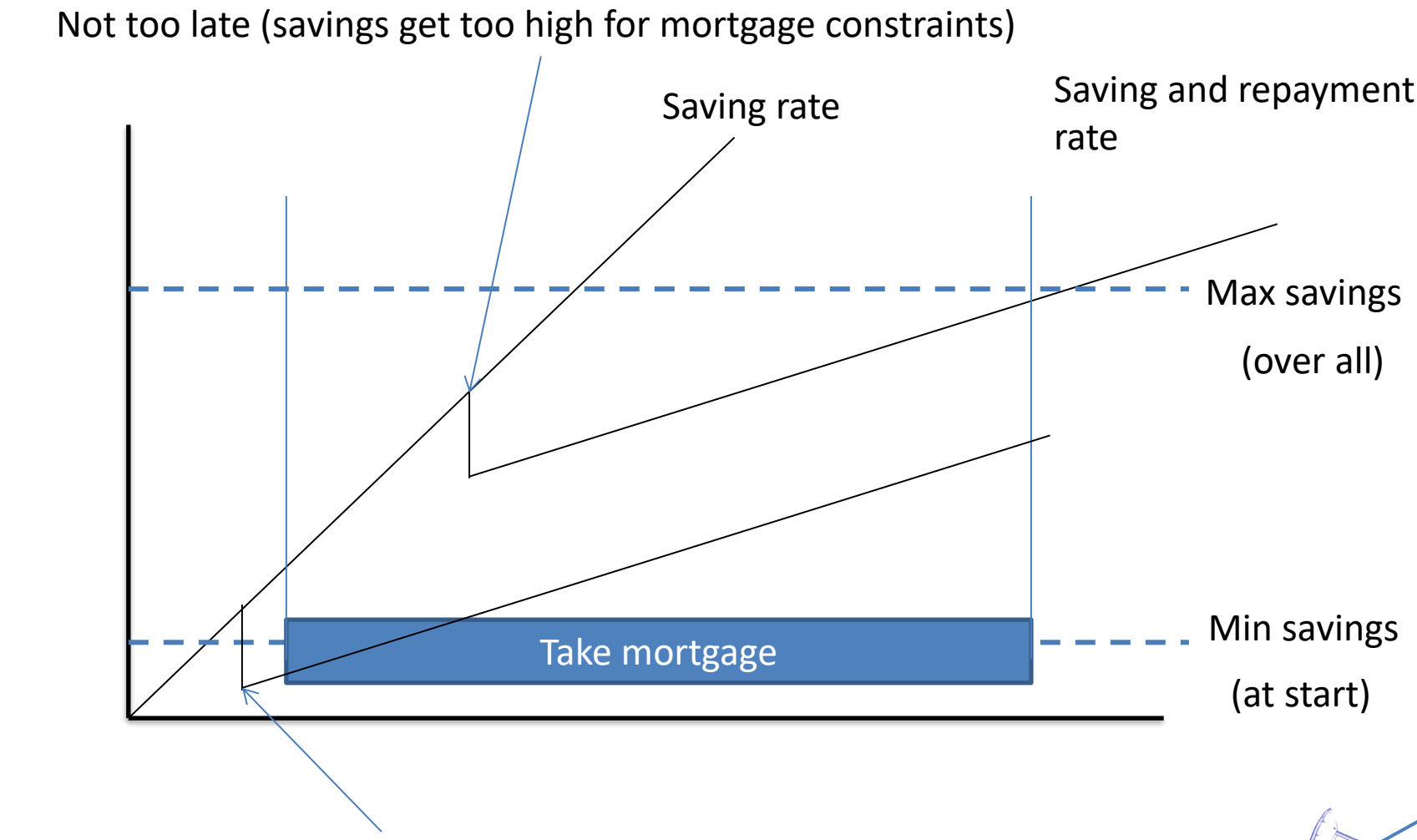


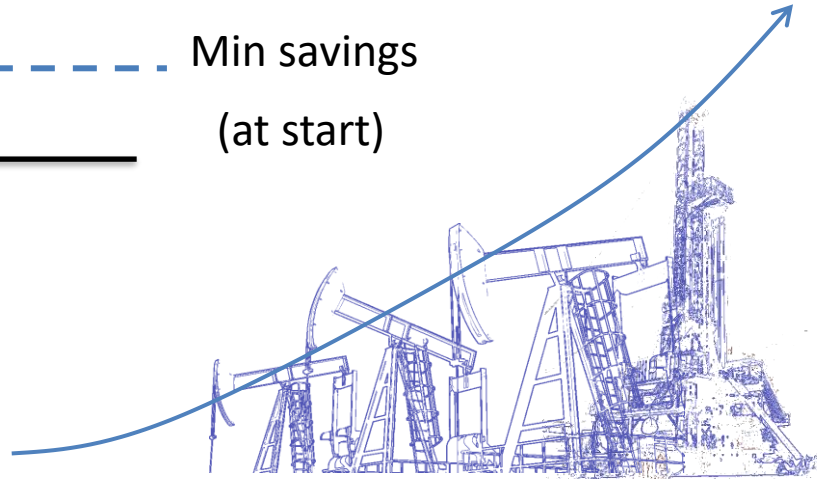


Plans for Borrowers

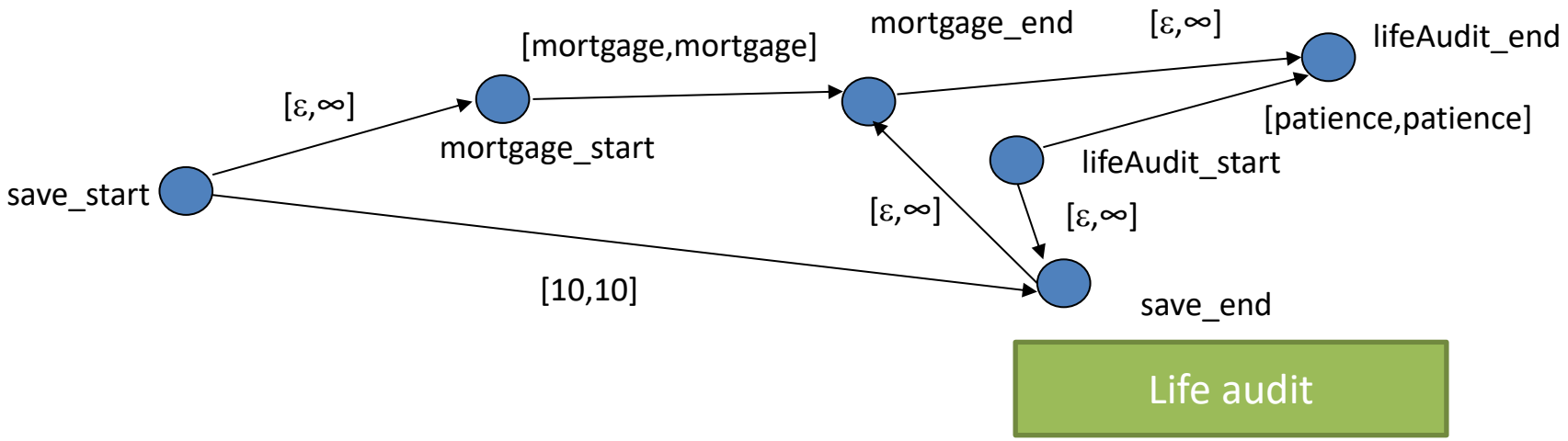


Sensitive bounds…





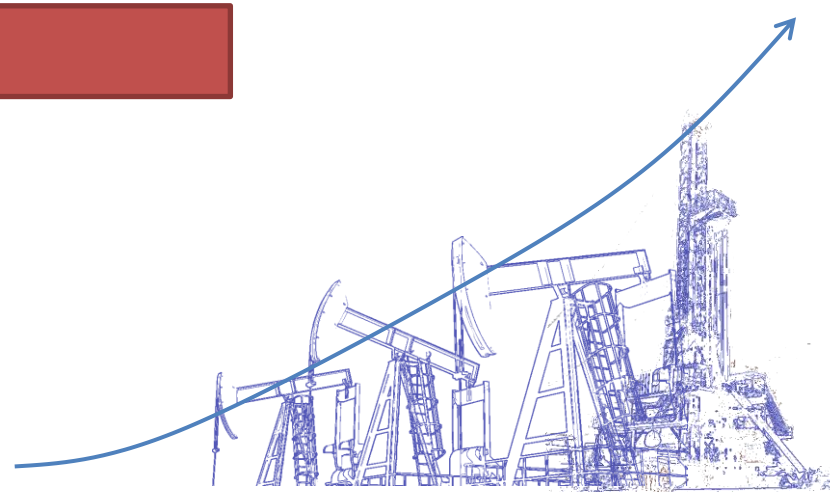
Planning with Simple Temporal Networks



Save hard

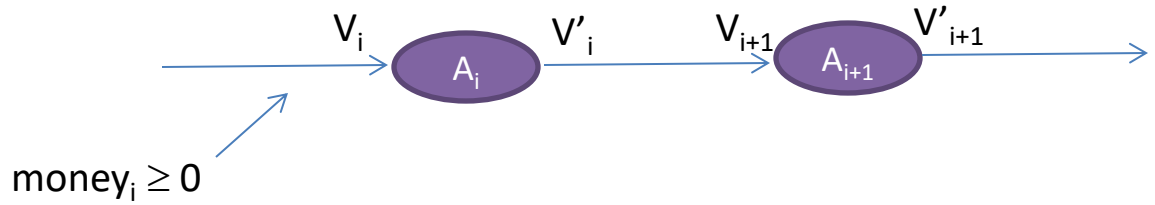
Take mortgage

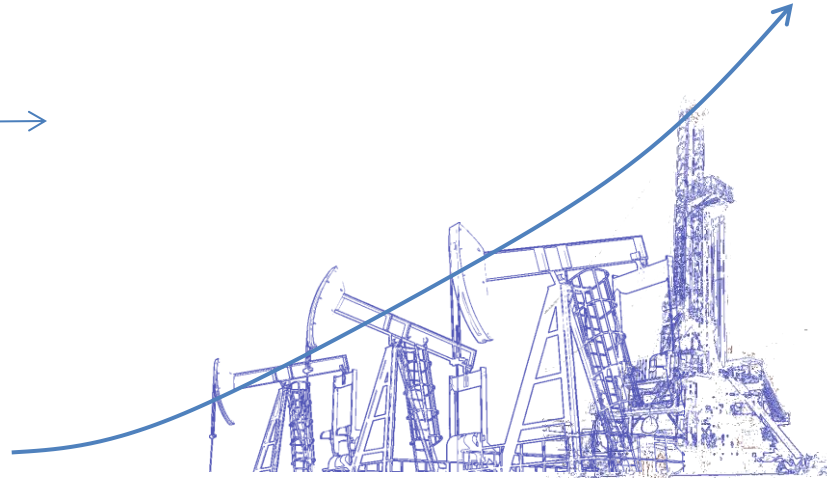
STN can be expressed as an LP



Encoding numeric state variables in an LP

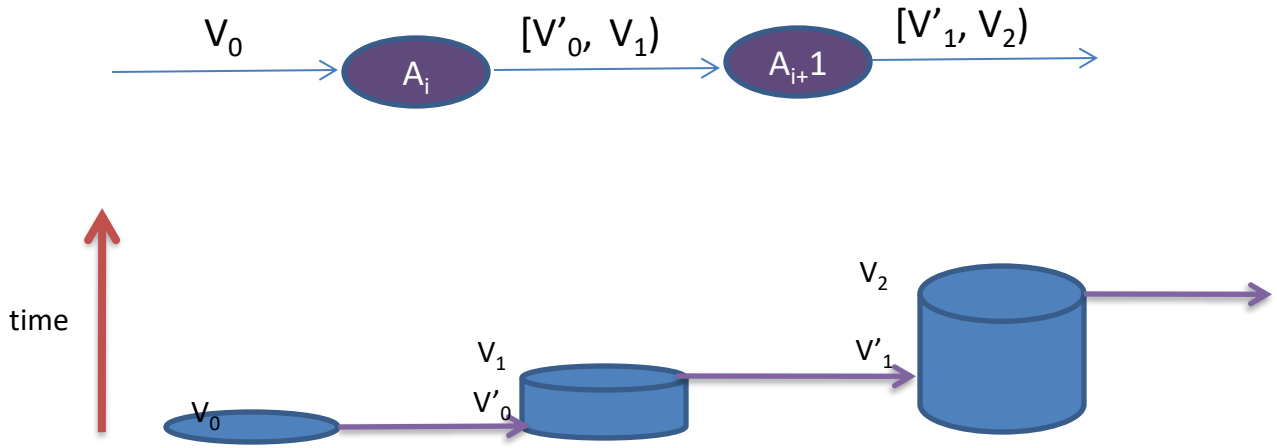
- To handle temporal-numeric interaction, we build an LP to model the numeric behaviour of actions
- For each action A_i:
  - add a set of variables V_i denoting the values of the numeric state variables before A_i
  - and a set V'_i denoting the values after the discrete effects of A_i
- Preconditions of the action A_i can be added as constraints over V_i:

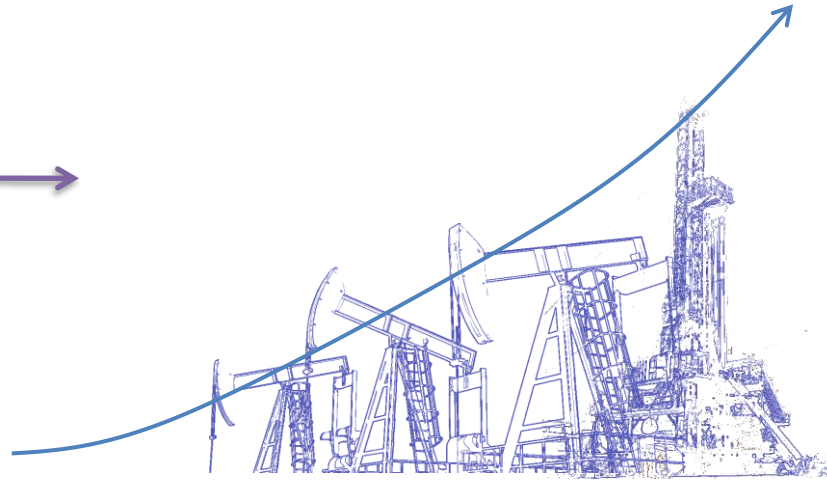




Encoding linear continuous change in the LP

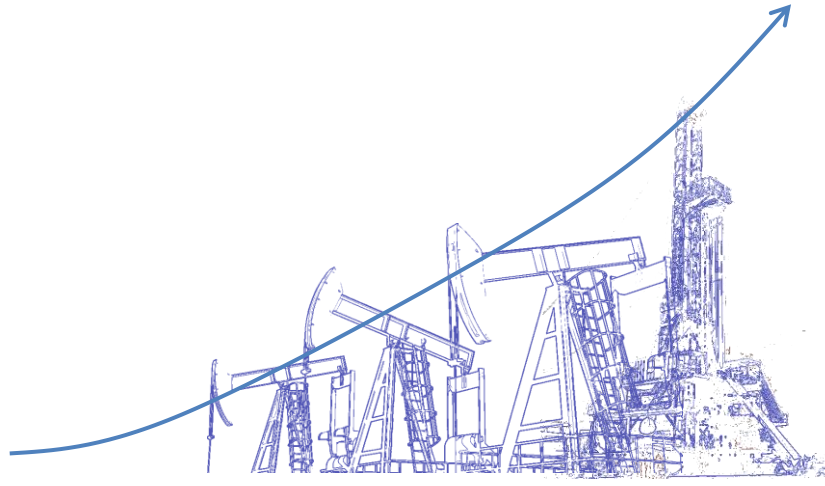
- So far, focus has been on instantaneous change: between V_i and V'_i
- In a continuous setting change also occurs between V'_i and V_{i+1} while remaining in a single state





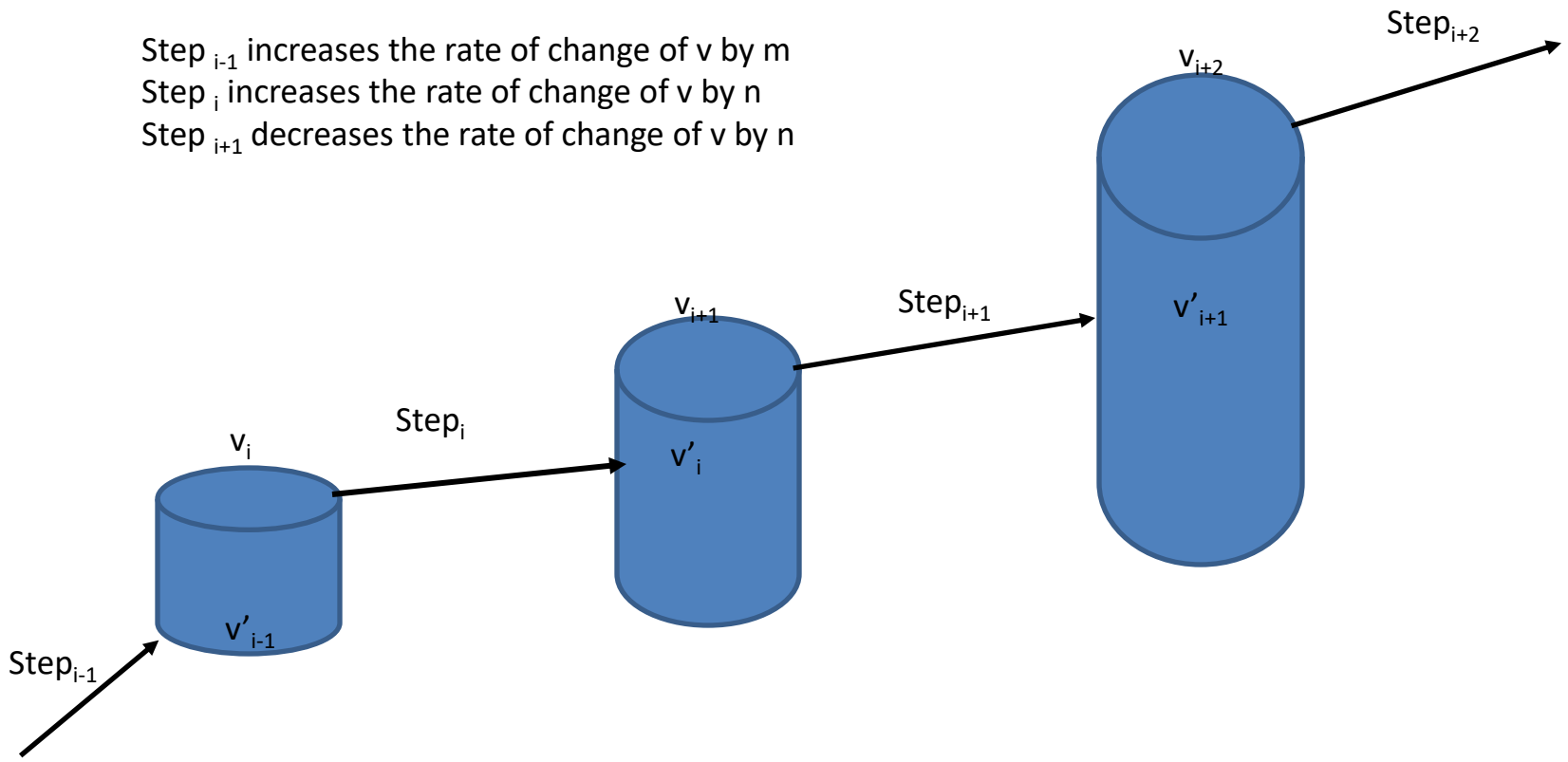
LP Constraints

- The LP is constructed by tracking the active continuous change and combining multiple updates on the same variable
  - Initially, for each variable v, $\delta v_0 = 0$ – no change active
  - If $step_i$ is the start of an action that increases the rate of change of v by m, $\delta v_{i+1} = \delta v_i + m$
  - If $step_i$ is the end of such an action, $\delta v_{i+1} = \delta v_i - m$

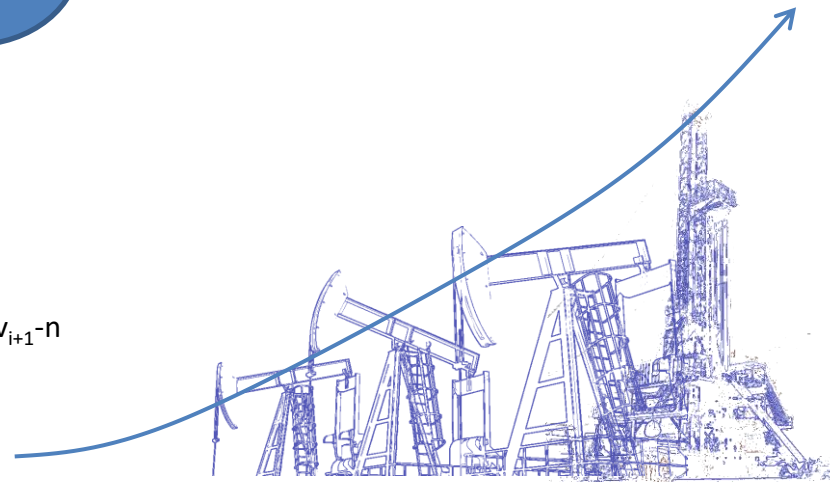


Change within a state

Step $_{i-1}$ increases the rate of change of v by m
Step $_{i}$ increases the rate of change of v by n
Step $_{i+1}$ decreases the rate of change of v by n



$\delta v_{i-1}=0$ $\delta v_i=\delta v_{i-1}+m$ $\delta v_{i+1}=\delta v_i+n$ $\delta v_{i+2}=\delta v_{i+1}-n$



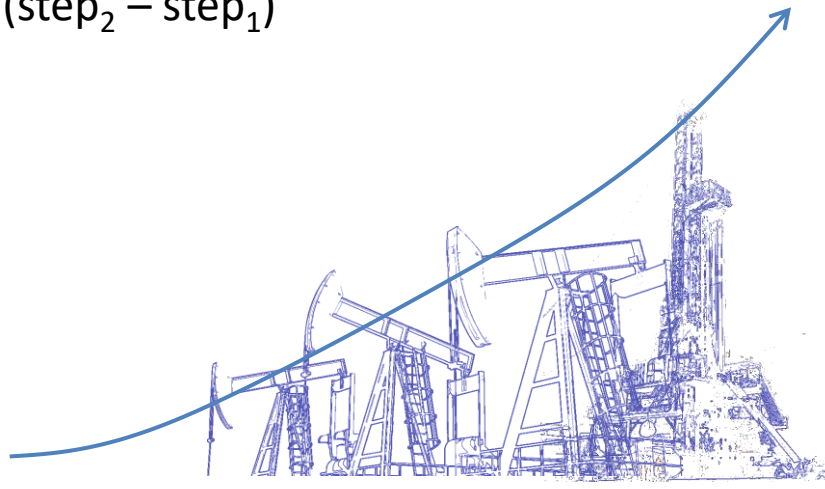
Encoding linear continuous change

- Constraints combining the discrete and time-dependent change can now be written as:

$$V_{i+1} = V'_i + \delta V_{i+1}.(step_{i+1} - step_i)$$

- The LP then encodes interacting *temporal-numeric consistency*

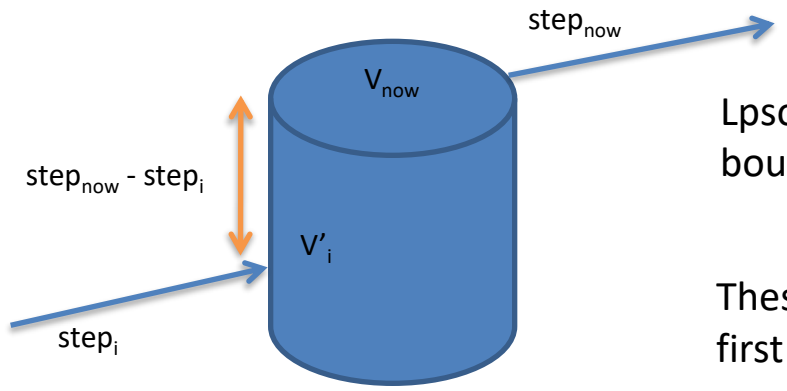
$$money_2 = money'_1 + (saving\text{-}rate - repayment\text{-}rate).(step_2 - step_1)$$



Extending the Temporal RPG

Because of continuous change we don't know the exact values of variables or time points of actions

We can compute bounds on these from constraints



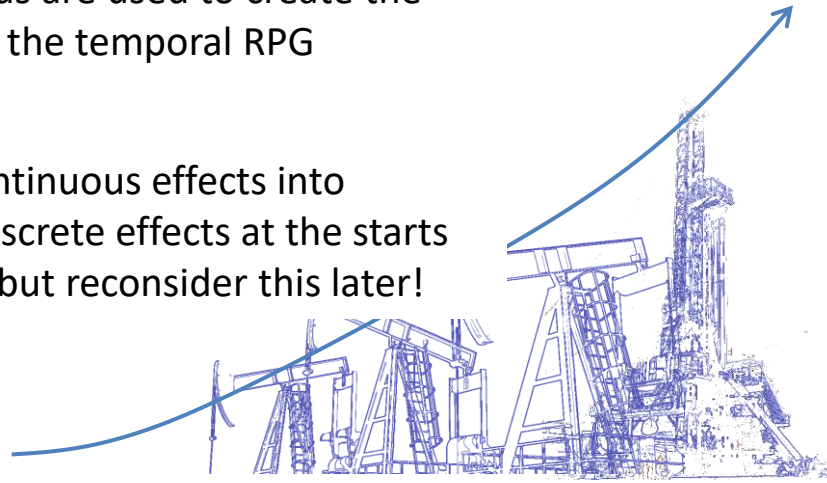
$step_{now} \geq step_i + \varepsilon$

$V_{now} = V'_i + \delta V_{i+1} (step_{now} - step_i)$

Lpsolve computes upper and lower bounds for V_{now} and $step_{now}$

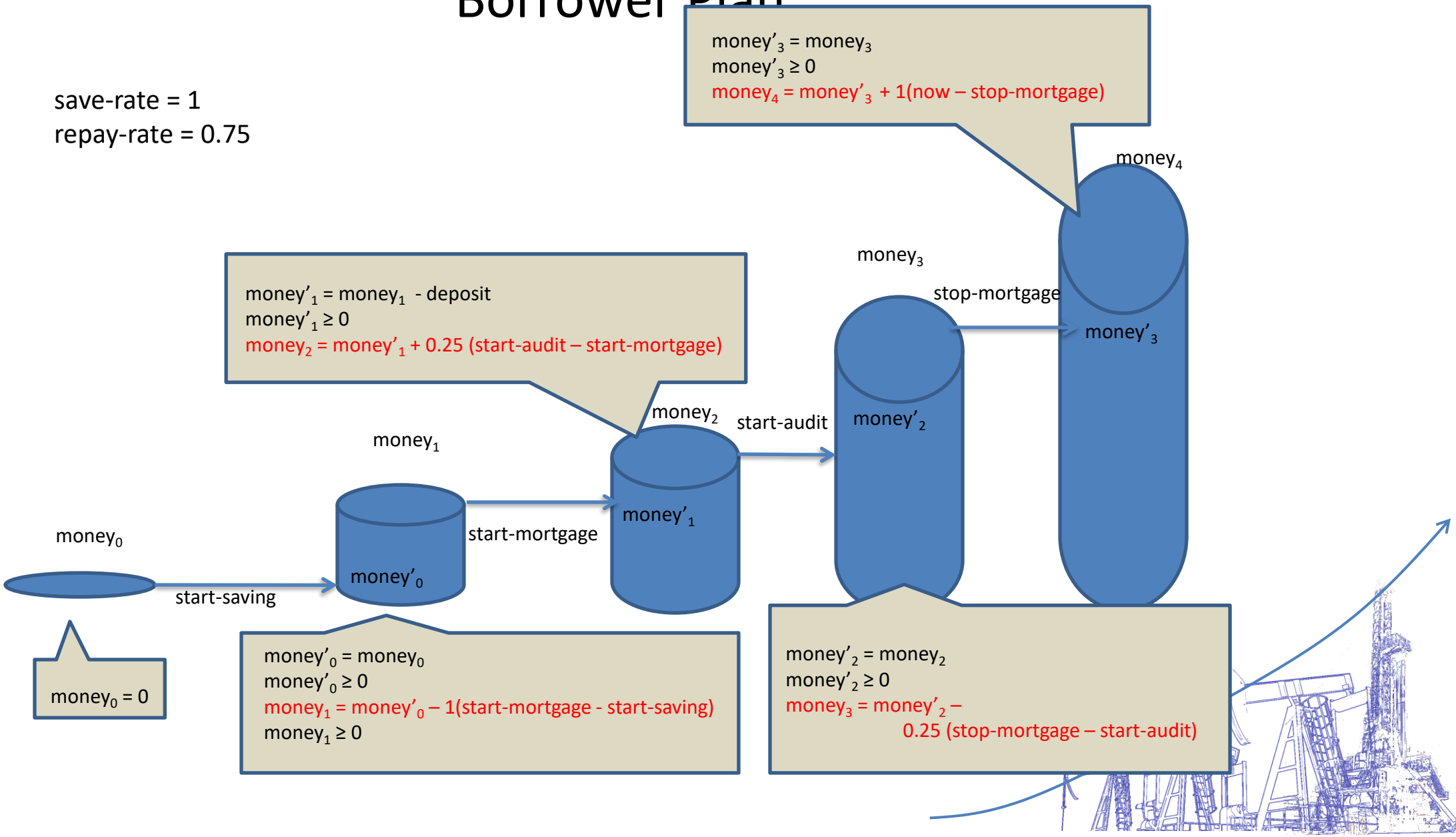
These bounds are used to create the first layer of the temporal RPG

We relax continuous effects into optimistic discrete effects at the starts of actions… but reconsider this later!



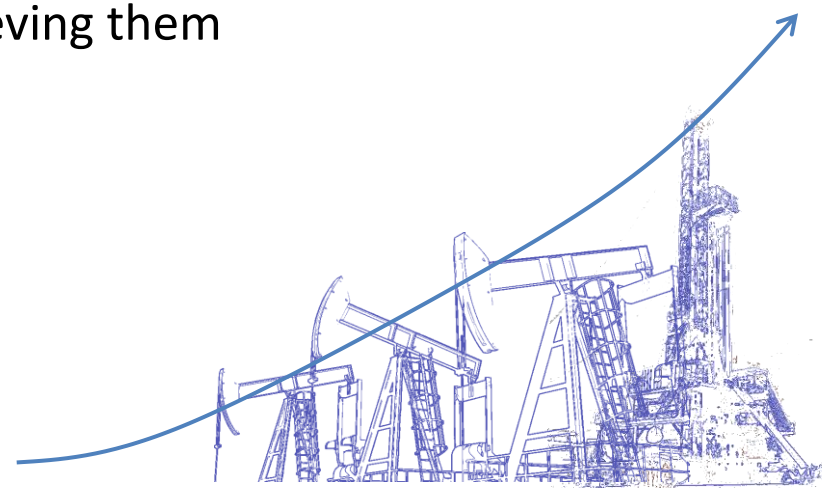
Borrower Plan

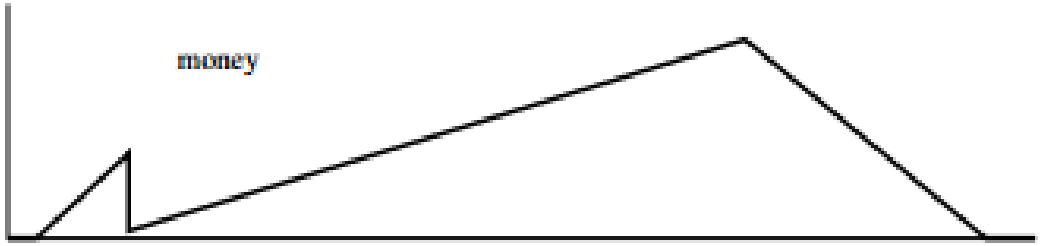
save-rate = 1
repay-rate = 0.75

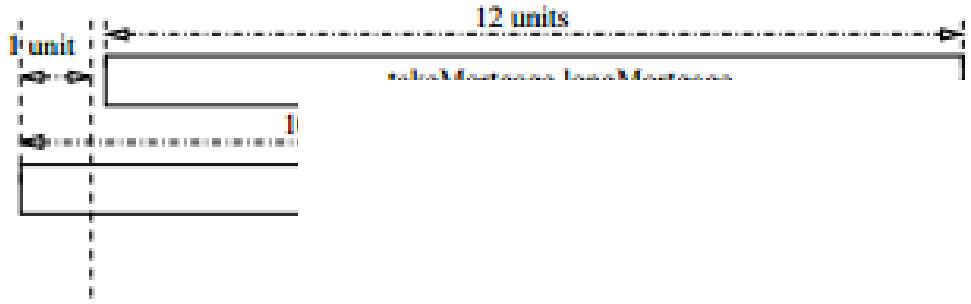


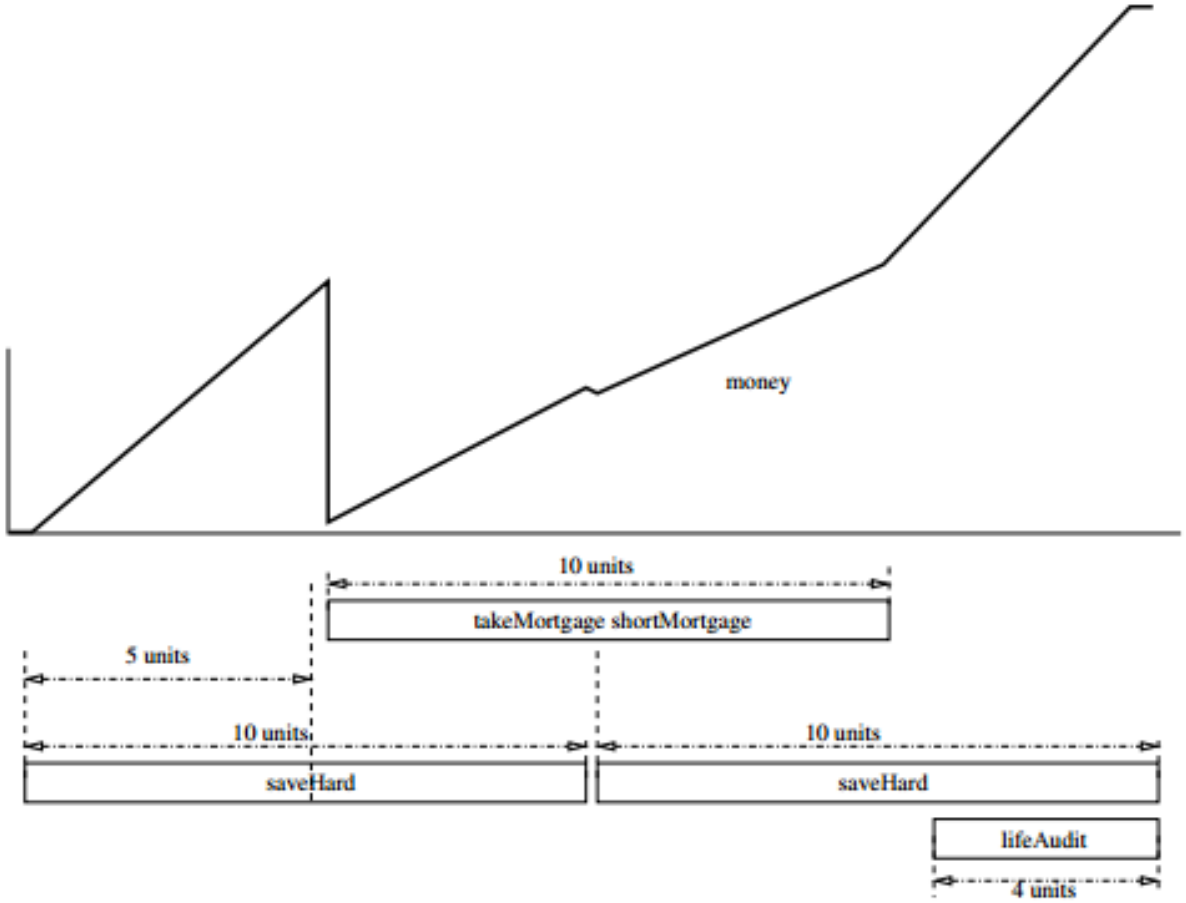
Heuristic Values

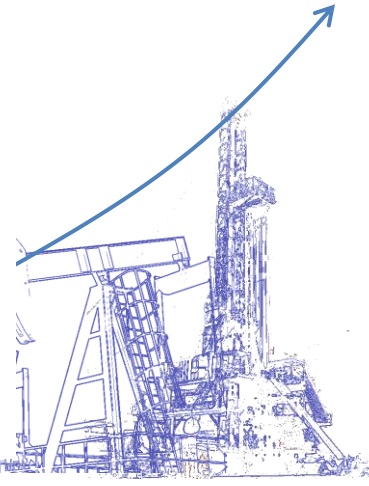
- A state is assigned a value based on the relaxed plan to achieve the goals, but using makespan (in the relaxed plan) as a tie-breaker between states
  - Different schemes are possible to weight more heavily towards makespan
- The makespan is estimated by building a Temporal Relaxed Planning Graph
  - Layers are timestamped with the earliest time they can be reached
- Process effects interact with makespan: active processes determine an earliest time when preconditions can be satisfied, where the process contributes to achieving them
  - Simple heuristic: assume all process effects available immediately
  - More complex: compute earliest time process effect available



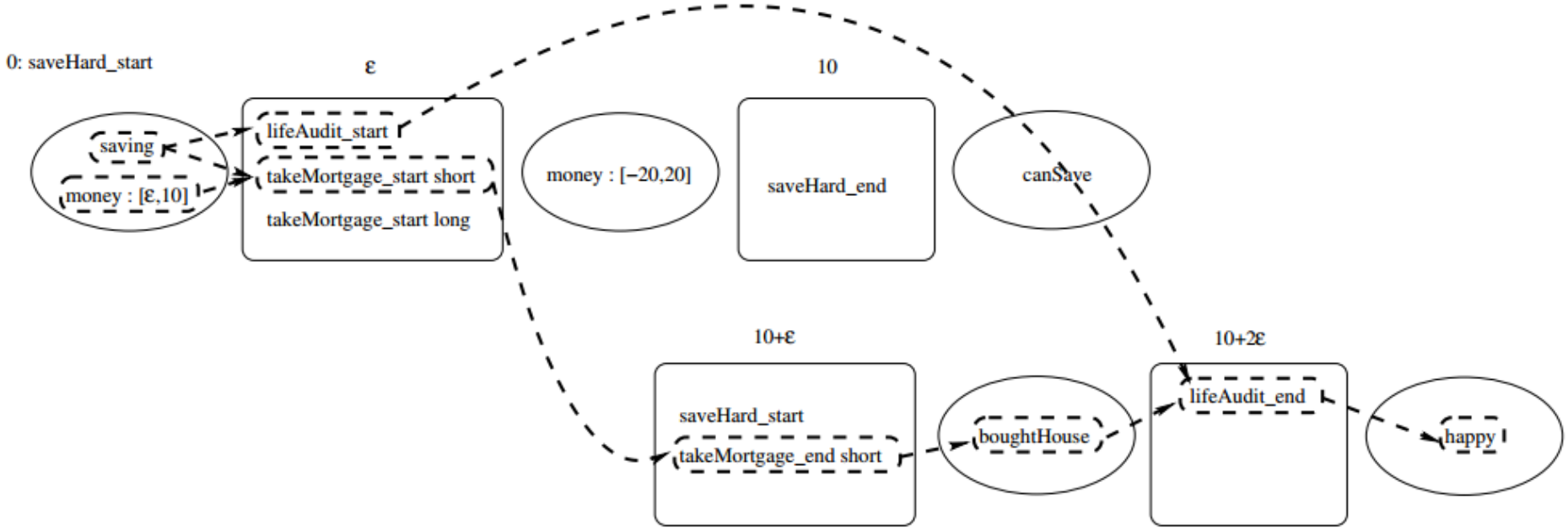

money

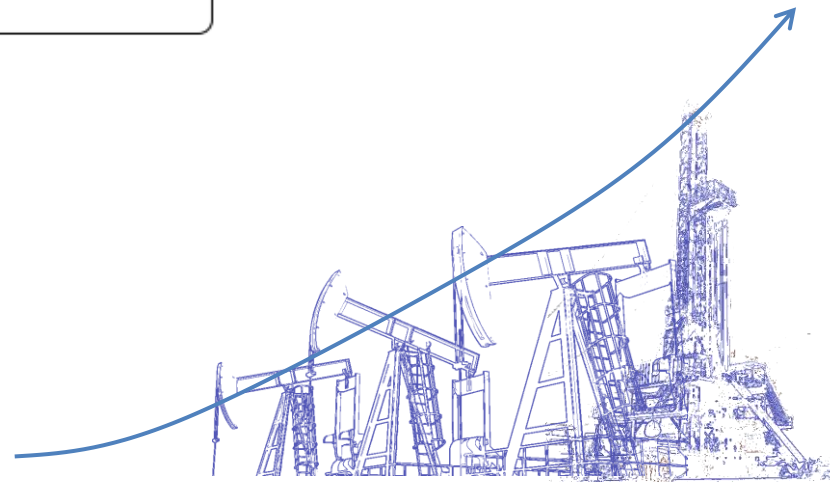

12 units
1 unit


money
10 units
takeMortgage shortMortgage
5 units
10 units
10 units
saveHard
saveHard
lifeAudit
4 units

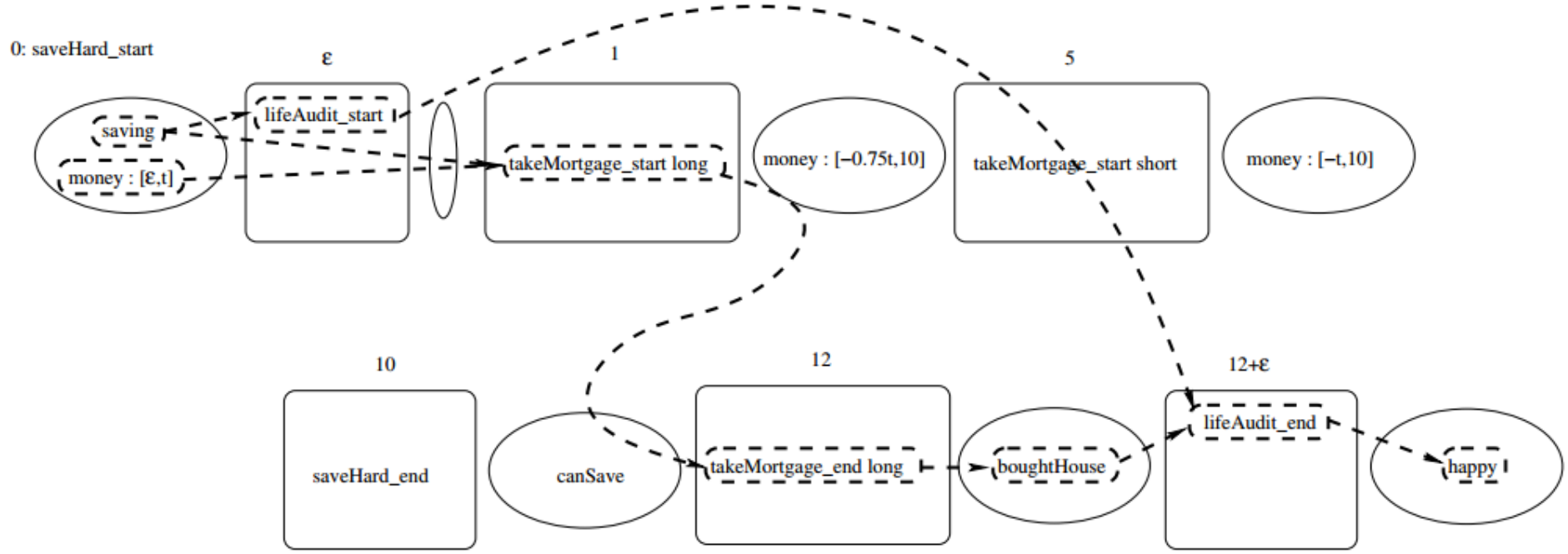


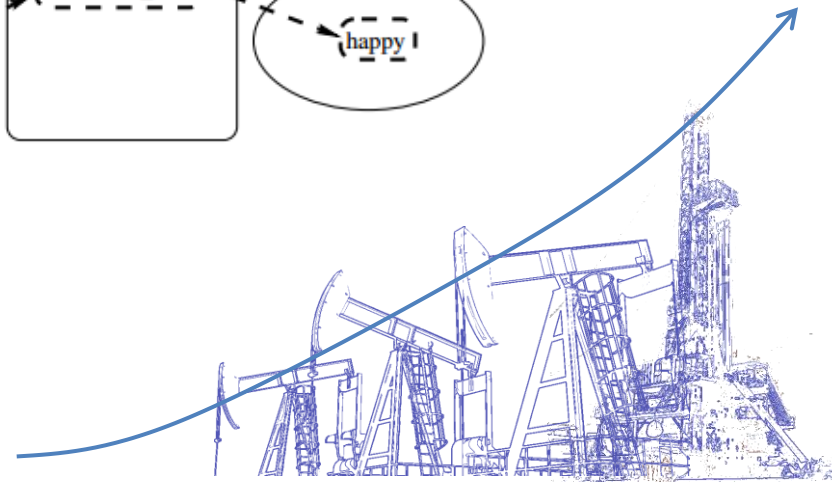
Simple Heuristic





Better Heuristic

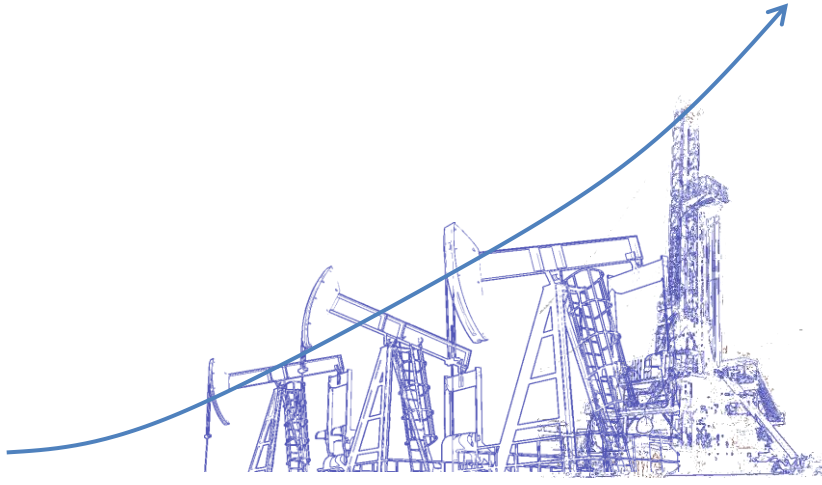




Events?

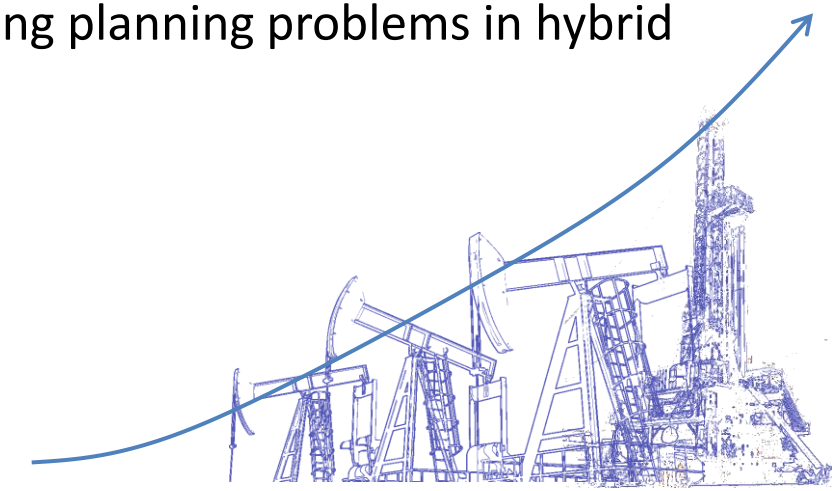
- Events and true processes are harder to handle, because they can generate happenings that are not part of the plan

- The COLIN approach has been extended to handle them (for linear processes)
  - See Coles&Coles



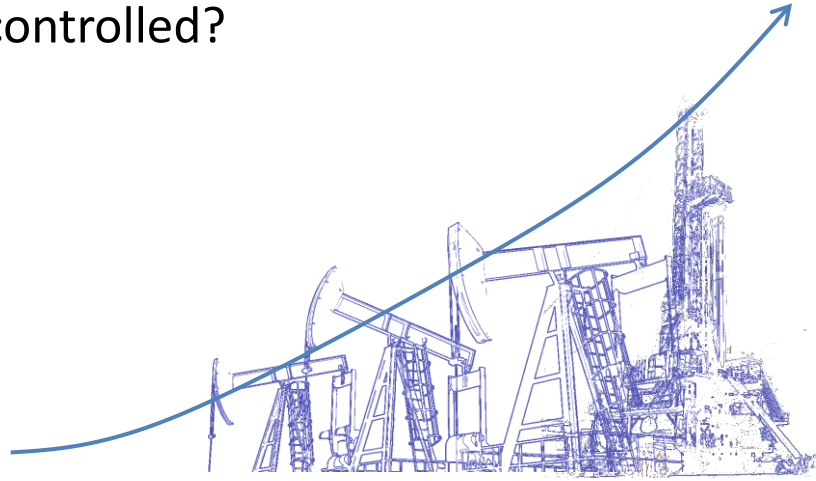
Closing Remarks

- Hybrid planning involves reasoning about continuous physical process models alongside discrete actions and events
- Hybrid systems introduce granularity challenges: some reasoning on long horizons and some very fine-grained
- Modelling processes and events gives a powerful language with subtle complexities
- Although we have some techniques for tackling these problems, solving planning problems in hybrid systems still offers lots of open challenges



Controlling Continuous Systems in PDDL+

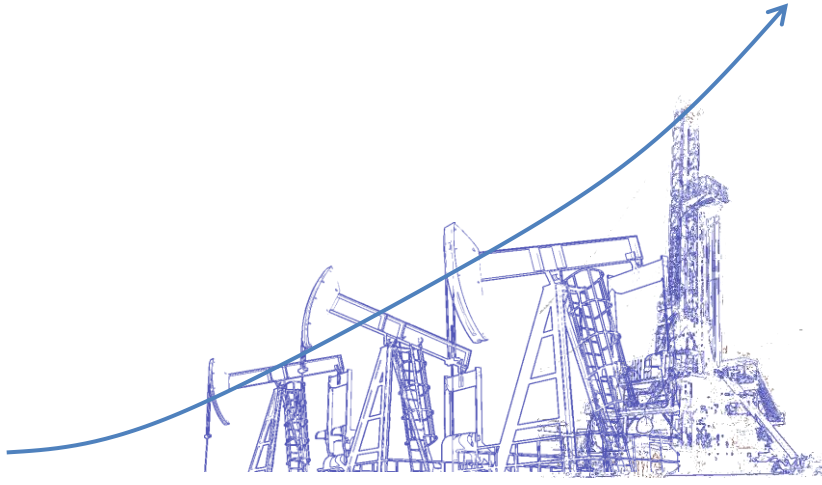
- A planner attempts to control a system by selection of actions
  - The actions are the means of control
- A PDDL action generates a discrete change in a finite propositional space and, possibly, an infinite discretised metric space
- How are continuous processes controlled in PDDL models?
- Eg: When refuelling the generator – how might the refuelling rate be controlled?



Discrete Control of Continuous Processes

- Could have a preset selection of flow rates

```
(:action setFlowRate
 :parameters (?f ?f1 – flow)
 :precondition (flowRate ?f)
 :effect (and (not (flowRate ?f)) (flowRate ?f1)
              (assign (theRateOfFlow) (valueFor ?f1))))

(:durative-action refuel
 :parameters (?f – flow)
 :duration (= ?duration (/ (canContent)  (theRateOfFlow)))
 :condition (over all (flowRate ?f))
 :effect (increase (fuelLevel) (* #t (theRateOfFlow))))
```

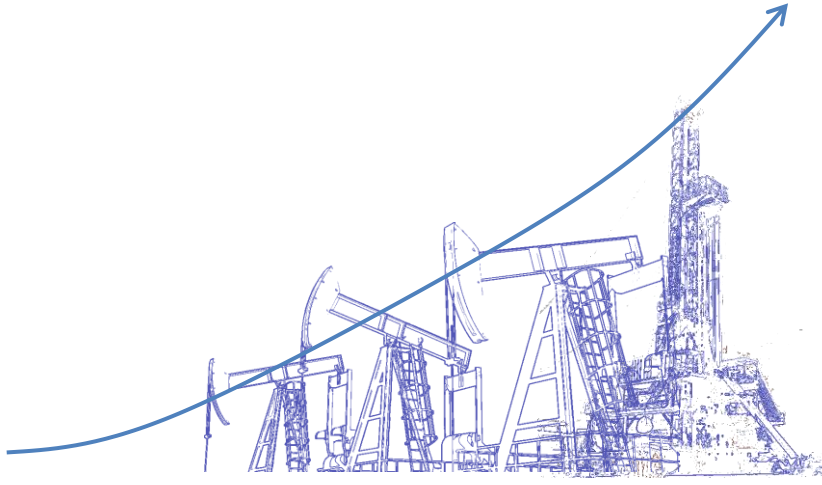


Discrete Control of Continuous Processes

- Could extend this to a discrete metric control:

```
(:action increaseFlowRate
 :parameters ()
 :precondition ()
 :effect (increase (theRateOfFlow) 1))
```

```
(:process refuel
 :parameters ()
 :precondition (and (> (theRateOfFlow) 0) (> (content) 0))
 :effect (and (increase (fuelLevel) (* #t (theRateOfFlow)))
              (decrease (content) (* #t (theRateOfFlow)))))
```

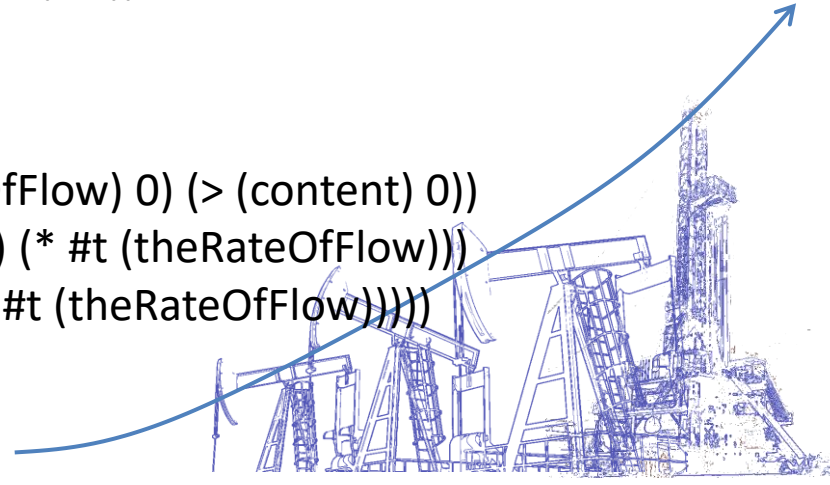


Continuous Control of Continuous Processes

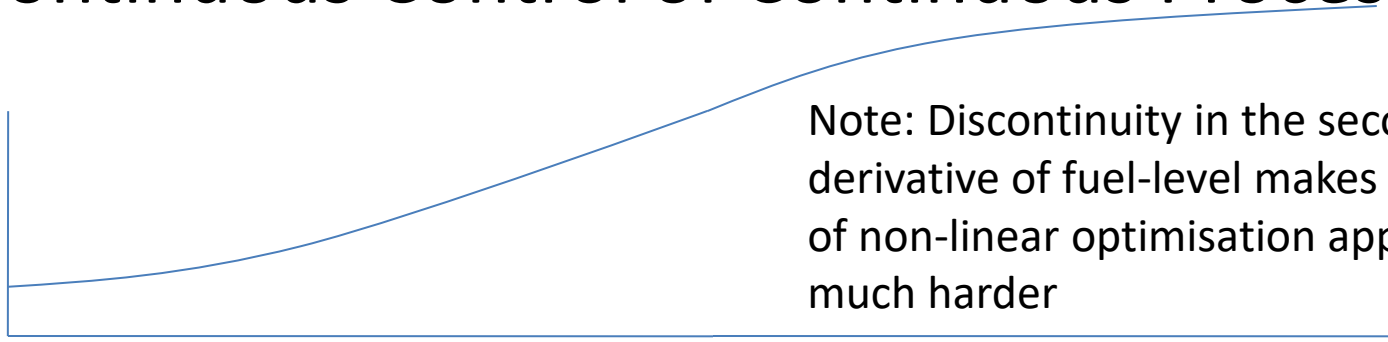
- Finally, we could allow continuous control:

```
(:action tip
 :parameters ()
 :precondition ()
 :effect (and (not (levelling))
              (tipping))

(:action lower
 :parameters ()
 :precondition (tipping)
 :effect (and (levelling)
              (not (tipping)))

(:event levelled
 :parameters ()
 :precondition (and (levelling)
              (= (theRateOfFlow) 0))
 :effect (not (levelling)))
```

```
(:process increaseFlowRate
 :parameters ()
 :precondition (tipping)
 :effect (increase (theRateOfFlow) #t))

(:process decreaseFlowRate
 :parameters ()
 :precondition (and (> (theRateOfFlow) 0) (levelling))
 :effect (decrease (theRateOfFlow) #t))

(:process refuel
 :parameters ()
 :precondition (and (> (theRateOfFlow) 0) (> (content) 0))
 :effect (and (increase (fuelLevel) (* #t (theRateOfFlow)))
              (decrease (content) (* #t (theRateOfFlow)))))
```



Continuous Control of Continuous Processes



Note: Discontinuity in the second derivative of fuel-level makes application of non-linear optimisation approaches much harder

$$\frac{d}{dt}(\text{fuel-level}) = (\text{theRateOfFlow})$$

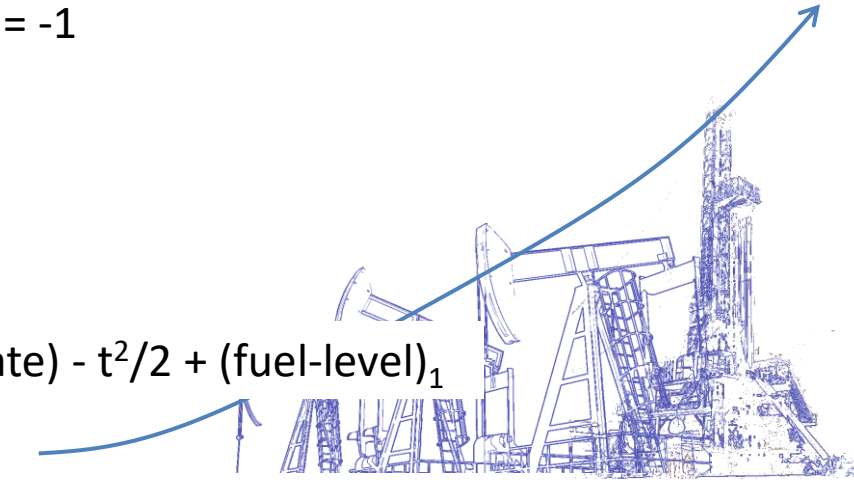
$$\frac{d}{dt}(\text{theRateOfFlow}) = 1$$

$$\frac{d}{dt}(\text{fuel-level}) = (\text{theRateOfFlow})$$

$$\frac{d}{dt}(\text{theRateOfFlow}) = -1$$

$$(\text{fuel-level}) = t^2/2 + (\text{fuel-level})_0$$

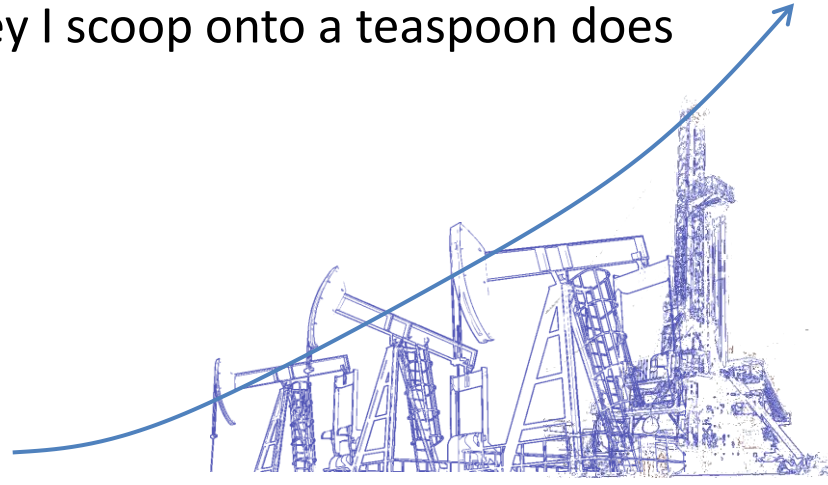
$$(\text{fuel-level}) = t*(\text{maxFlowRate}) - t^2/2 + (\text{fuel-level})_1$$



Continuous Control

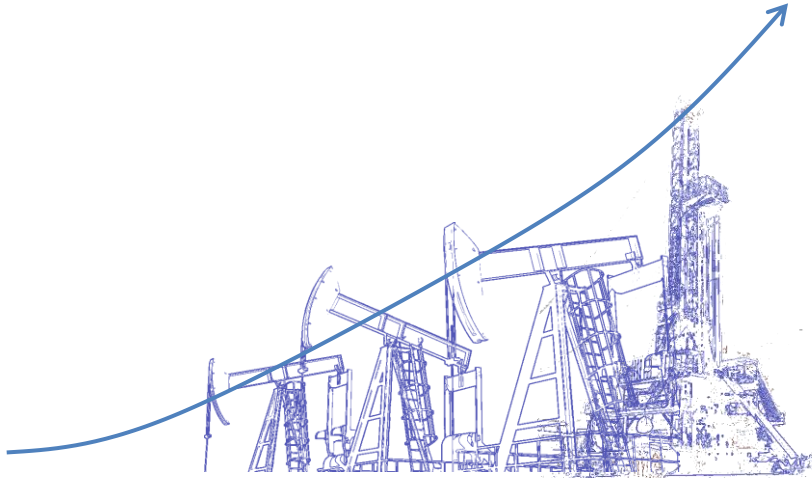
- Control is only through the choice of *when* to execute actions
- Numbers are not first-class objects, so cannot be selected as parameter values
- Intuitively acceptable model of analogue control
  - Setting control level takes time – the level is proportional to the time spent changing it

- But....
  - Inconvenient for highly variable control levels that can be changed very fast relative to process being influenced (eg accelerator setting on car)
  - Seems counter-intuitive for some controls (Eg the amount of honey I scoop onto a teaspoon does not seem to be proportional to the time I spend scooping it)



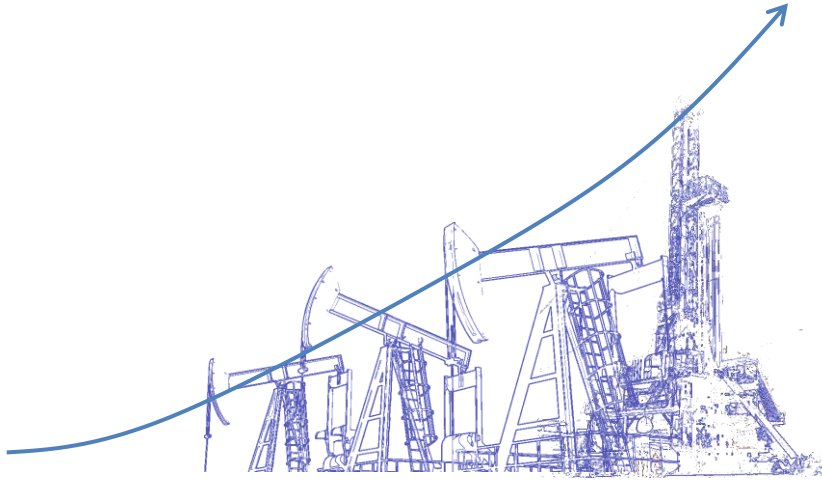
Key Observations

- PDDL+ is (by design) able to express only finitely many ground actions
- Therefore, choices are always finite, except for the choice of *when* to execute an action
  - The timeline is continuous, real-valued
- The (deterministic) Timed Hybrid Automaton offers the same model



A Different Model

- An alternative view is that actions could have metric parameters
- A Discrete Time Automaton uses discrete time points corresponding to “clock ticks” and continuous control parameters for its actions
  - Thus, there is less choice about when things can be done, but more choice about what can be done at those times
- Arguably, a better model of a digital controller (which can only act on clock cycles), although parameters are, in practice, also discretised for such a device



Yet Another View

- Real processes are never precisely predictable, so it is most useful to imagine having a lower level controller that will use a tight feedback loop to manage process behaviours
- Therefore, the choice of control parameters lies in a range and the interesting question is only about the envelope of the process controlled by that range over time

- A hybrid planner is concerned with:
  - selecting discrete actions and
  - the timing of such actions
- But it will rely on a lower level system to realise the actual control of processes within their envelopes in order to meet constraints in the plan