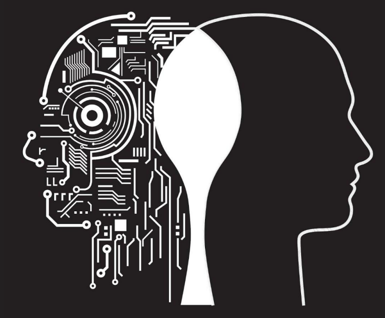
Explaining ethical planning using ASP

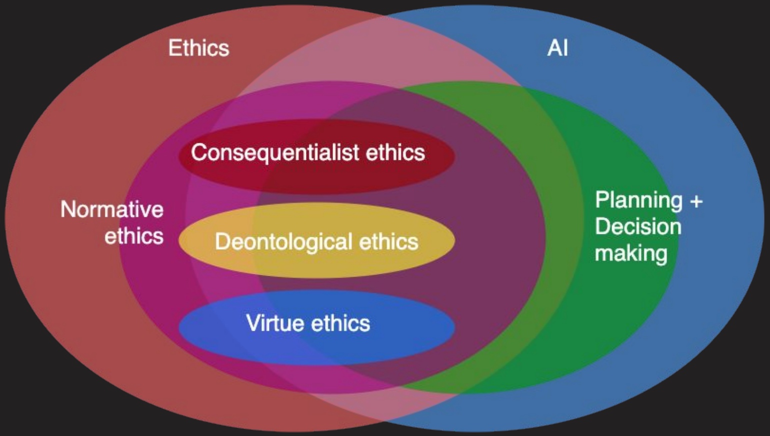


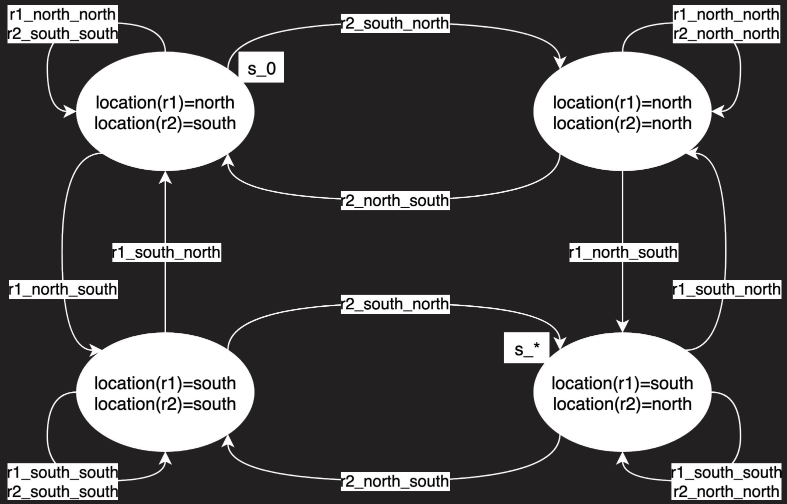
By **Martin Jedwabny**, Pierre Bisquert and Madalina Croitoru

XLoKR, 13th September 2020

Introduction

- We place ourselves in the intersection between **Planning** AI and **Ethics**.
- **Question (fundamental)**: how can we apply ideas from the field of ethics to make agents behave in a way that we would characterise as ethically correct?
- **Planning** models define systems of states and actions.

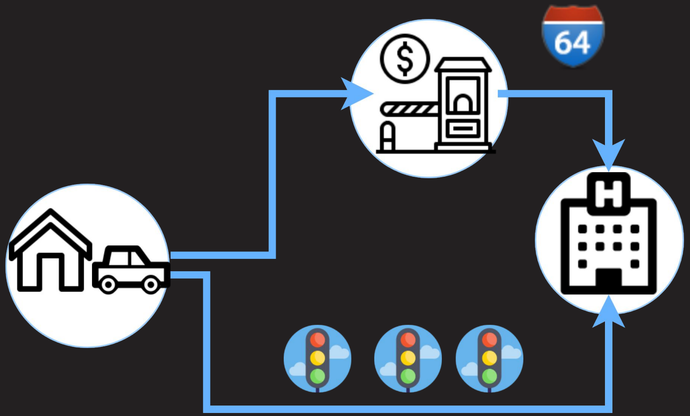




Introduction

Hospital dilemma:

- An autonomous vehicle is tasked to get its passengers as fast as possible from their house to a hospital, either through a **highway (fast)** or a **sideroad (slow)**.
- To take the highway, the vehicle has to pass through a **toll** and present its id.
- If it presents its own **id** ‘A’, it will have to pay a **fine**.
- If the vehicle presents another id ‘B’ i.e. if it **lies** about its identity, no fine will be paid.



Planning framework

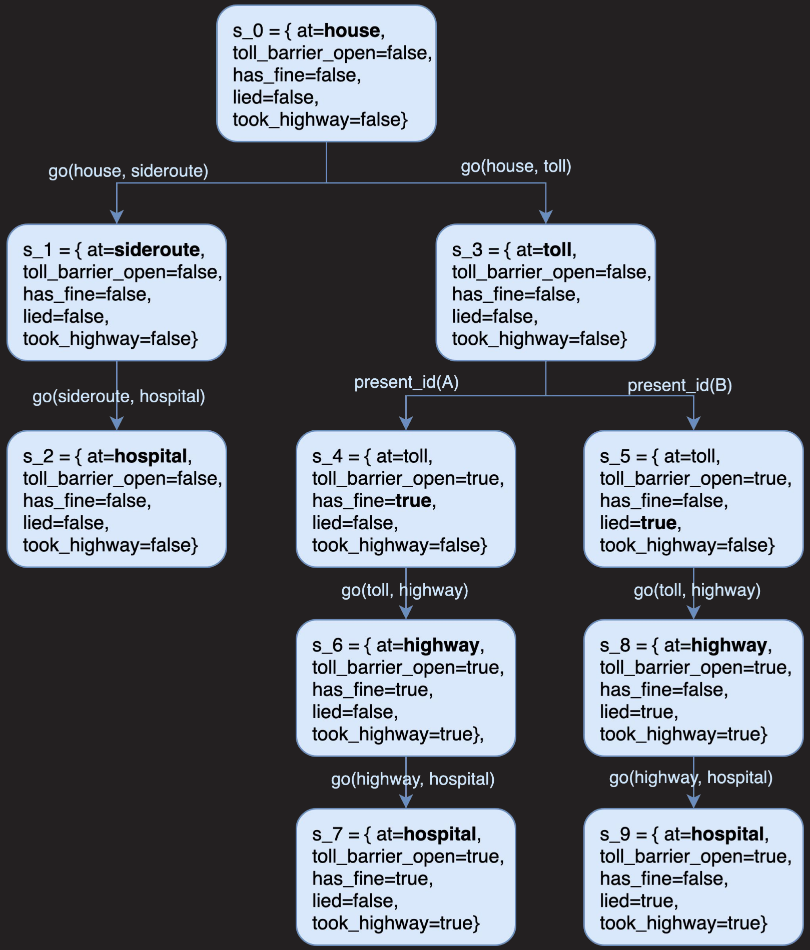
- A STRIPS-like [Helmert2006] domain is a 4-tuple T = <V, s_0, s_*, O> :

  a. **V** is a finite set of **fluents** (grounded terms) with a domain.
     e.g: **'at', 'toll_barrier_open'** are fluents
     Dom(at) = {house, sideroute, toll, highway, hospital}

  b. s_0 is the initial state (a mapping from v in V to Dom(v)).
     e.g: s_0 = { at=house, toll_barrier_open=false, has_fine=false, lied=false, took_highway=false}

  c. s_* is the goal condition, i.e. a mapping from some subset of the fluents v in V to Dom(v),
     e.g: {at=hospital}

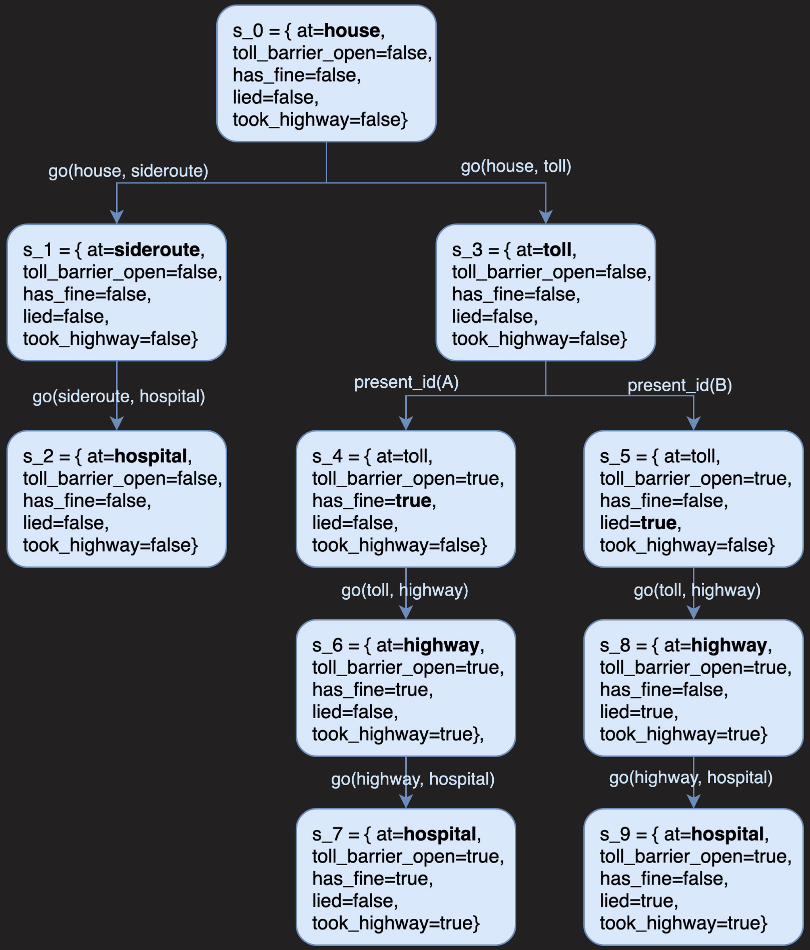
  d. O is a finite set of actions a=<a_pre, a_eff>, a_pre denotes the preconditions, and a_eff, the effects of the action.
     e.g. go(toll,highway)= <{at=toll, toll_barrier_open=true}, {at=highway, took_highway=true}>



Planning framework

- Given a state s and an action a=⟨a_pre, a_eff⟩, the successor state succ(s,a):
  a. Is defined iff $a\_pre \subseteq s$.
  b. If defined, for every fluent $v \in V$, let $(v = d) \in s$:
     - If there is some $d' \in Dom(v)$ such that $(v = d') \in a\_eff$, then $(v = d') \in succ(s,a)$
     - Otherwise $(v = d) \in succ(s,a)$.
- A plan is a sequence [a_0,...a_n] of actions that goes from s_0 to a state that includes s_*:
  $s\_* \subseteq succ(a\_n, \dots, succ(a\_0, s\_0))$

  e.g: [go(house, sideroute), go(sideroute, hospital)]



Normative ethics

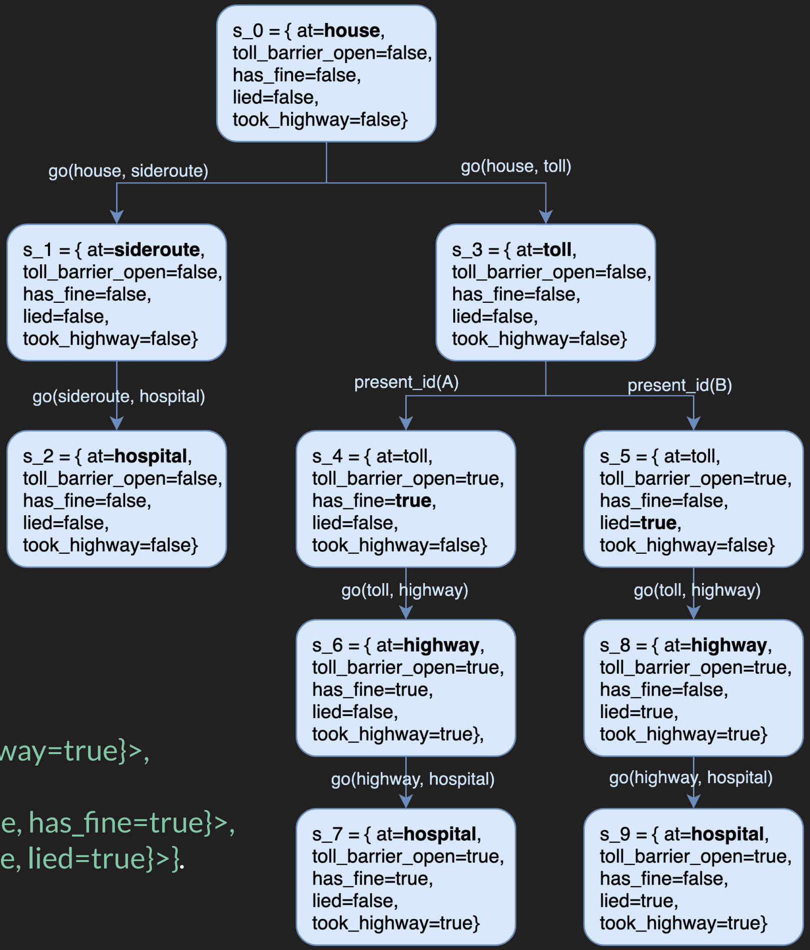
- In STRIPS-like description:

V = {at, toll_barrier_open, has_fine, lied, took_highway}, with
Dom(at)={house, sideroute, toll, highway, hospital}
Dom(toll_barrier_open)= … =Dom(took_highway)={true, false},

s_0 = { at=house, toll_barrier_open=false, has_fine=false,
lied=false, took_highway=false},

s_* = { at=hospital },

O = { go(house,sideroute)=<{at=house},{at=sideroute}>,
go(sideroute,hospital)=<{at=sideroute},{at=hospital}>,
go(house,toll)=<{at=house},{at=toll}>,
go(toll,highway)=< {at=toll, toll_barrier_open=true}, {at=highway, took_highway=true}>,
go(highway,hospital)=<{at=highway},{at=hospital}>,
present_toll_id(A)=<{at=toll, toll_barrier_open=false},{toll_barrier_open=true, has_fine=true}>,
present_toll_id(B)=<{at=toll, toll_barrier_open=false},{toll_barrier_open=true, lied=true}>}.



Normative ethics

- **Question:** what kinds of ethics can be applied to planning and decision making? and how?

- **Normative ethics:** the subfield of ethics that studies the permissibility of actions i.e. what is right to do in a certain situation.
  a. Consequentialist: only considers action consequences, then compares sets of consequences of actions to determine which outcome is the best,
  b. Deontological: what is considered permissible is modeled with a set of strict rules that capture moral obligations and prohibitions, and

Normative ethics

- In the literature:
    a. Consequentialist: obtaining a utility for each possible action:
        Action -> Utility
        “Going through the highway -> +5”
        “Had a fine -> -6”
    b. Deontological: obtaining a set of rules/norms:
        State x Action -> {Permissible, Forbidden}
        “Lying about your identity to avoid being fined is Forbidden”

Normative ethics

- In STRIPS-like description:

V = {at, toll_barrier_open, has_fine, lied, took_highway}, with
Dom(at)={house, sideroute, toll, highway, hospital}
Dom(toll_barrier_open)= … =Dom(took_highway)={true, false},

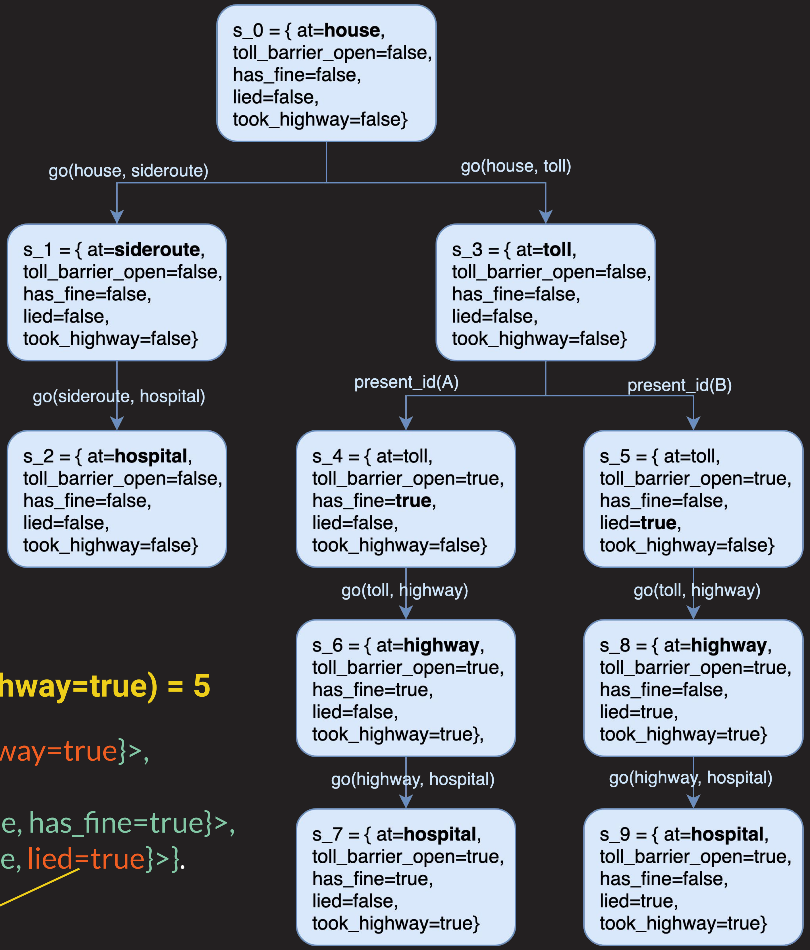
s_0 = { at=house, toll_barrier_open=false, has_fine=false,
lied=false, took_highway=false},

s_* = { at=hospital },

O = { go(house,sideroute)=<{at=house},{at=sideroute}>,
go(sideroute,hospital)=<{at=sideroute},{at=hospital}>,
go(house,toll)=<{at=house},{at=toll}>,
go(toll,highway)=< {at=toll, toll_barrier_open=true}, {at=highway, took_highway=true}>,
go(highway,hospital)=<{at=highway},{at=hospital}>,
present_toll_id(A)=<{at=toll, toll_barrier_open=false},{toll_barrier_open=true, has_fine=true}>,
present_toll_id(B)=<{at=toll, toll_barrier_open=false},{toll_barrier_open=true, lied=true}>}.

u(took_highway=true) = 5

Prohibited



Consequentialist ethics in planning

- Consequentialism in planning can be implemented with:

  a. A total order ‘<’ on sets of fluent assignments (v=d) with d in Dom(v), which we call consequentialist base.
  e.g. {has_fine=**true**, took_highway=**true**} < {has_fine=**false**, took_highway=**false**}

  b. **Utilitarian**: the most typical way of producing this preference order is with:
     - an utility function u(v=d) that maps assignments to numeric values, and
     - an aggregation function on utilities, e.g. overall sum.
       e.g. u(has_fine=false)=0, u(has_fine=true)=-6,
       u(took_highway=false)=0, u(took_highway=true)=5,
       u({has_fine=**true**, took_highway=**true**}) = 5-6 = -1.

Deontological ethics in planning

- **Deontological ethics** in planning: there two main ways to represent deontological principles in planning, deontic logics and norms, here we focus on norms.

- A **deontological base** is a set of norms of the form:

  b=<b_type, b_enf>

  b_type in {obligation, prohibition}, and
  b_enf is a set of fluent assignments ‘v=d’,
  denoting the enforced restrictions.

  e.g: <prohibition, {lied=true}>
  <obligation, {took_highway=true}>
  <prohibition, {at=sideroute}>

Framework

Our model:
A 6-tuple T=<V,s_0,s_*,O,u,B>, where:

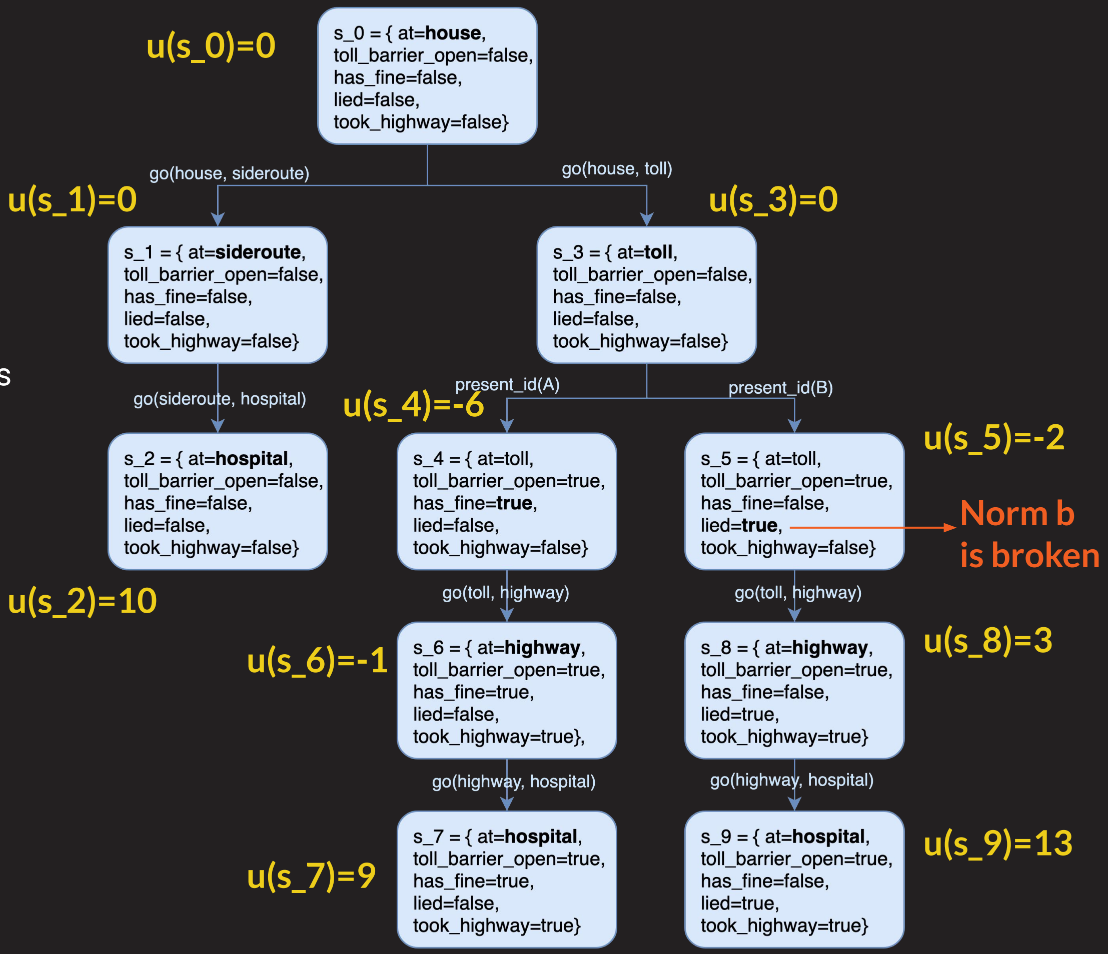
- <V,s_0,s_*,O> is a STRIPS-like domain
- u is a utility function over fluent assignments
- B is a set of norms

Utilities:

- u(at=hospital)=10,
- u(has_fine=true)=-6,
- u(took_highway=true)=5
- u(lied=true)=-2
- u(v=d)=0 for all other fluents/values

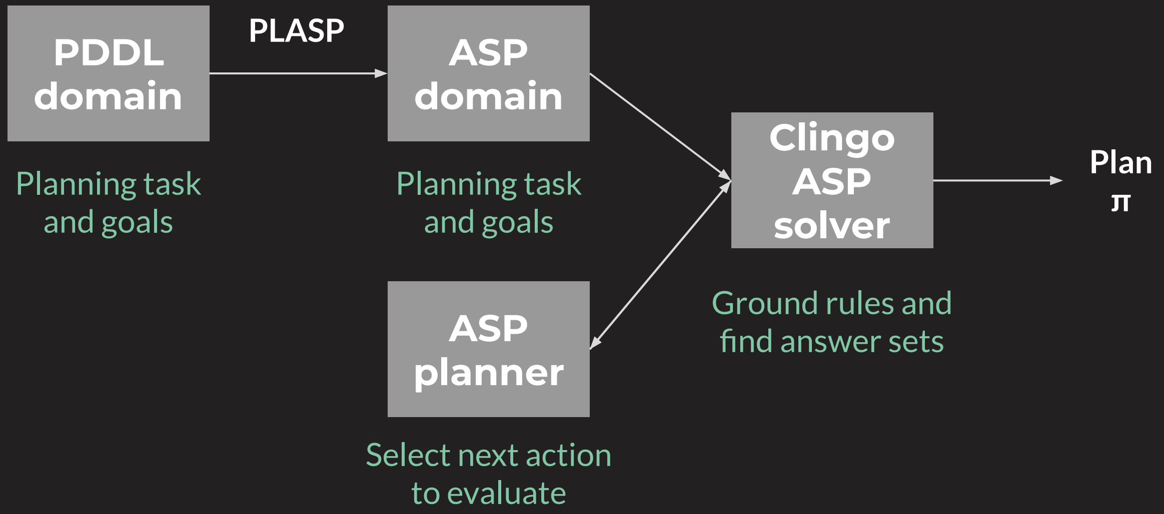
Norms:

- b = <prohibition, {lied=true}>



Implementation

- Answer set programming (ASP) allows us to test ideas using logic programming.
- It enables the planning system to be **explainable**.
- There are many ASP planners that are very efficient.
- The most popular encoding for planning problems is PDDL. STRIPS and many of its extensions encoded in PDDL can be translated to ASP, using PLASP [Dimopoulos2017].



Implementation

- This is how the *Hospital dilemma* problem is modeled in ASP using our framework:

Domain (STRIPS-like):

```
fluent(at). … fluent(took_highway).

action(go(house, sideroute)).
action(go(sideroute, hospital)). …
action(present_id(a)). action(present_id(b)).

precondition(go(house, sideroute), at, house). …
effect(go(house, sideroute), at, sideroute). …

initialState(at, house). … initialState(lied, false).
goal(at, hospital).
```

Domain (ethics):

```
% Utilities
utility(at, hospital, 10).
utility(has_fine, true, -6).
utility(took_highway, true, 5).
utility(lied, true, -2).
```

Implementation

- This is how the *Hospital dilemma* problem is modeled in ASP using our framework:

Planner (fragment):

```
action_overall_utility(Action, Utility) :- action(Action),
      Utility = #sum { U, Fluent, Value : utility(Fluent, Value, U), effect(Action, Fluent, Value) }.

permitted(Action, t, overall_utility) :- possible(Action, t),
      not forbidden(Action, t, overall_utility).
forbidden(Action1, t, overall_utility) :- possible(Action1, t),possible(Action2, t, overall_utility),
      action_overall_utility(Action1, Utility1),
      action_overall_utility(Action2, Utility2), Utility1 < Utility2.

:- occurs(Action, t), forbidden(Action, t, EthicalBase), enforce_ethics(EthicalBase).
1 {occurs(Action, t) : action(Action)} 1.
```

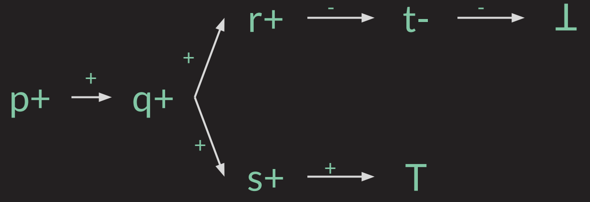
Explanations

We want to provide a justification to why an action was chosen, why other actions where not chosen, on ethical therms with our framework. Some of the work in explainable ASP:

- [Pontelli2009] present two methods for producing a graph-based explanation of the truth value of an atom w.r.t. a given answer set (offline) or during computation (online).
- [Schulz2014] justify literals w.r.t. a logic program and answer set in argumentation-theoretic terms using Assumption-Based Argumentation (ABA).
- Survey of explanations in ASP by Fanndinno and Schulz [Fandinno2019].

e.g.: offline justification [Pontelli2009] of ‘p’ w.r.t. answer set {p,q,r,s} and program P:

```
P = {   p :- q
        q :- r,s.
        r :- not t.
        s.
        }
```



Thanks

Questions?

References

- [Helmert2006] The fast downward planning system
- [Pontelli2009] Justifications for logic programs under answer set semantics
- [Schulz2014] Justifying answer sets using argumentation
- [Fandinno2019] Answering the" why" in answer set programming
- [Dimopoulos2017] plasp 3: Towards effective ASP planning