Lesson 6_Logic_Programming

Explainable and Ethical AI: A Perspective on Argumentation and Logic Programming

In this part of the lecture we will see how to design and develop, from a computer engineering perspective, **ethical behaviour**. In particular, how we can implement explainable and ethical behaviour exploiting logic programming and declarative approaches.

In nowadays AI applications, we have a lot of autonomous agents that need to interact and to take decisions in order to reach system goals and it's often the case in which agents have to face situation involving choices on moral or ethical dimension.

In this context we can investigate on how to program machine ethics in these agents and, if we think to the multi agent systems, we have two perspective we can focus on

- one stressing all the **individual** cognition, deliberation and behaviour;
- the other stressing the **collective** moral and how the moral of the system can emerge in this context.

We will mainly focus on the first perspective of this investigation and, in the design of individual cognition and deliberation, computation can become the vehicle for studying morality. In the computation model and in the design of knowledge and cognition, we can also address morality issue. In particual, we will see how some logic programming techniques and extensions can be effective for dealing with morality and ethics design. We will discuss about:

- abduction with integrity constraints
- preferences over abductive scenarios
- probabilistic reasoning
- counterfactuals, and updating
- argumentation

This is not an exhaustive vision of all the extensions that can be put into place when we have to design machine ethics but it's a good overview.

Why are we proposing logic programming for dealing with morality?

Because many moral issues are really close to the logic programming based representation and reasoning technique. For instance, we can design

- **moral permissibility**, taking into account different logic models (e.g double effect and triple effect);
- the famous **dual process** model of logic that stress the interaction between the deliberative and the reactive process that are involved when dealing with moral decisions;
- the role of **counterfactual thinking** which is really close to an LP perspective with the abduction extension.

Agents

From a Computer Engineering perspective agents are

> autonomous computational entities

Being computational entities we have to design and implement with the program the behaviour of the agent. But the things that characterize these computational entities with respect to others in the system is that thay are always autonomous, so they encapsulate control and also a criterion to govern this control.

Thinking about agents as autonomous, we have a lot of features that become interesting and important. Autonomous agents are interactive, social, proactive, and situated in a context with which can interact. Agents might have goals or tasks, or be reactive, intelligent, mobile. They live within multi-agent system context, and interact with other agents through communication actions, and with the environment with pragmatical actions.

Why Logic?

Logic-based approaches already play a well-understood role in the engineering of intelligent (multi-agent) systems; declarative, logic-based approaches have the potential to represent an alternative way of delivering symbolic intelligence, complementary to the one pursued by sub-symbolic approaches [Calegari et al., 2020].

Logic Programming reasoning has some features that are interesting for machine ethics:

- *Abduction* scenario generation and of hypothetical reasoning, including the consideration of counterfactual scenarios about the past
- *Preferences* enacted for preferring scenarios obtained by abduction
- *Probabilistic LP* allows abduction to take scenario uncertainty into account
- *LP counterfactuals* permit hypothesizing into the past, even taking into account present knowledge
- *Argumentation* converse, debate and explain

And technically

- *LP updating* enables updating the knowledge of an agent
- *Tabling* affords solutions reuse and is employed in joint combination with abduction and updating

> *"What is or can be the added value of logic programming for implementing machine ethics and explainable AI?"*

The main answer lies in the three main features of LP

- *(i)* being a declarative paradigm
- *(ii)* working as a tool for knowledge representation, and
- *(iii)* allowing for different forms of reasoning and inference

These features lead to some properties for intelligent systems that can be critical in the design of ubiquitous intelligence (both in terms of transparency and in terms of ethics).

Features of logic based model

Provability

A logic based model can provide for a well founded semantics ensuring some fundamental computational properties, such as correctness and completeness. Moreover, extensions can be also formalised, well-founded as well, based on recognised theorems, like, for instance, correctness of transitive closure. Provability is a key feature in the case of trusted and safe systems.

Explainability

It's in some way intrinsic in logic based systems because formal methods for argumentation, justification, and counterfactual are often based on LP. A system can be defined explainable when it's capable to engage in dialogues with other actors to communicate its reasoning and explain its choices.

Expressivity and situatedness

These are two other features we can reach with logic programming because we can exploit different extensions of logic programming and so we can inject in our system a lot of application specific expressivity. We can also capture some speficication of the context introducing some extention not purely tailored into logic programming.

Hybridization

We can integrate heterogenous contexts of intelligent systems also in relation to the application domains and we can customise this model as needed.

Why Logic for Agents?

We use logic for agents because it is a declarative rather than an agent programming language but it allows to inject logical inference for reasoning and reasoning for deliberation. Moreover, with a logic programming approach, we can explicitly define the belief and the goals of the agent and represent them for agent oriented operations. Actually, we can build also more specific agent language leveraging on a logic based approach.

Prolog Recap

Essentials of Logic Programming

Three fundamental features [Apt, 2005]

terms *Computing takes place over the domain of all terms defined over a "universal" alphabet.*

mgu *Values are assigned to variables by means of automatically-generated substitutions, called* most general unifiers. *These values may contain variables, called* logical variables.

backtracking *The control is provided by a single mechanism:* automatic backtracking.

- Let A be an alphabet of a language L
- countable disjoint set of constants, function symbols, and predicate symbols.
- an alphabet is assumed to contain a countable set of variable symbols
- a term over A is defined recursively as either a variable, a constant or an expression of the form $f(t_1, ..., t_n)$, where f is a function symbol of A, and t_i are terms
- an atom over A is an expression of the form $p(t_1, ..., t_n)$, where p is a predicate symbol of A, and t_i are terms
- p/n denote the predicate symbol p having arity n
- a literal is either an atom a or its negation *nota*
- a term (respectively, atom and literal) is *ground* if it does not contain variables
- set of all ground terms (respectively, ground atoms) of A is called the Herbrand universe (respectively, Herbrand base) of A

Prolog Syntax

Prolog terms

variables alphanumeric strings starting with either an *uppercase* letter or an *underscore*

- underscore alone is the *anonymous variable*—sort of *don't care* variable
- underscore followed by a string is a normal variable during resolution, but it does not need to be exposed in the computed substitution

functors alphanumeric strings starting with a *lowercase* letter

- holds for both proper functors and constants

terms are built recursively out of functors and variables as in logic programming

→ e.g., `term`, `Var`, `f(X)`, `p(Y,f(a))` are *Prolog terms*

→ e.g., `term`, `var`, `f(a)`, `p(x,y)` are *Prolog ground terms*

Prolog atoms

predicates alphanumeric strings starting with a *lowercase* letter

- the same as functors

atoms are built applying predicates to terms as in logic programming

→ e.g., `predicate`, `f(X)`, `p(Y,f(a))` are *Prolog atoms*

→ e.g., `predicate`, `f(a)`, `p(x,y)` are *Prolog ground atoms*

Prolog clauses

clause a Horn clause of the form `A :- B1, ..., Bn.`

- where `A, B1, ..., Bn` are Prolog atoms
- `A` is the head of the clause
- `B1, ..., Bn` is the body of the clause
- `:-` denotes logic implication
- `.` is the terminator

fact a clause with no body `A.` ($n = 0$)

rule a clause with at least one atom in the body `A :- B1, ..., Bn.` ($n > 0$)

goal a clause with no head and at least one atom in the body `:- B1, ..., Bn.` ($n > 0$)

- often written as `?- B1, ..., Bn.`

Prolog program

program a sequence of Prolog clauses

interpreted as a *conjunction* of clauses

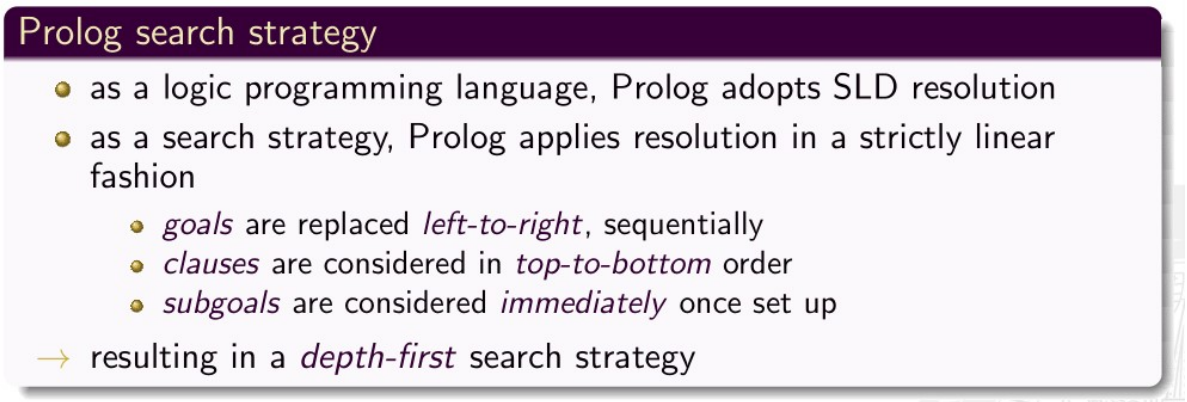
logic theory constituting a *logic theory* made of Horn clauses written according the Prolog syntax

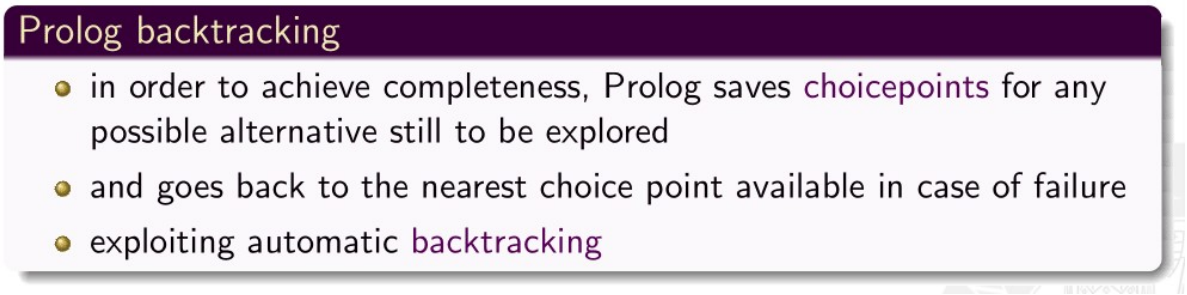
Prolog Execution

Aim of a Prolog computation

- given a Prolog program P and the goal `?- p(t1,t2,...,tm)` (also called *query*)
- if `X1,X2,...,Xn` are the variables in terms `t1,t2,...,tm`
- the meaning of the goal is to query P and find whether there are some values for `X1,X2,...,Xn` that make `p(t1,t2,...,tm)` true

→ thus, the aim of the Prolog computation is to find a substitution $\sigma =$`X1/s1,X2/s2,...,Xn/sn` such that $P \models \mathtt{p(t1,t2,\ldots,tm)}\sigma$





Extensions of LP

Abduction

The notion of abduction is characterized as a step of adopting a hypothesis as being suggested by the facts. Abduction consists of reasoning where one chooses from available hypotheses those that best explain the observed evidence. Usually abduction is implemented as an extension of LP by introducing the abducibles.

Abductive logic programs have three components <**P**, **AB**, **IC**>:

- P is a **logic program** of exactly the same form as in logic programming;
- AB is a **set of predicate names**, called the abducible predicates;
- IC is a **set of first-order classical formulae** that states integrity constraints.

For example:

Grass **is** wet if it rained.

Grass **is** wet if the sprinkler was on.

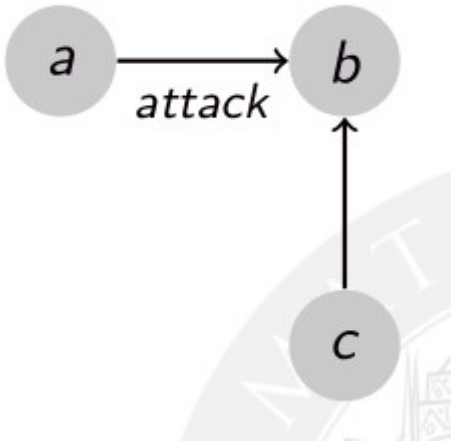
The sun was shining.

IC: false if it rained and the sun was shining.

The observation that the grass is wet has two potential explanations, *it rained* and *the sprinkler was on*, which entail the observation. However, only the second potential explanation, *the sprinkler was on*, satisfies the integrity constraint.

Argumentation

We can extend LP also for dealing with **argumentation**. An argumentation system consists of a couple (A, R), where A is a set of elements (arguments) and R a binary relation representing attack relation between arguments. It can be represented exploiting a **directed graph** in which each **node** represents an **argument** and an **arc** denotes an **attack** by one argument to another.



In argumentation theory we need a way to build such arguments and if we exploit LP as a referring model we can build arguments exploiting inference among rules. We need to define an **acceptability criteria**, how to analyze the graph in order to say which arguments are acceptable according to some general criteria and which arguments must be discarded. This procedure of knowing which arguments should be accepted under a given semantics is called **argument evaluation**. Among the most common approaches to argument evaluation, there are:

- **Extention-based approach** in which semantics specification concerns the generation of a set of collective acceptable arguments. It allows to determine conflict-free sets and how to work on these sets;
- **Labelling-based approach** in which semantics specification concerns the generation of a set of labellings that represents a possible alternative state of an argument.

The traditional semantics contained in the Dung's origianal paper, which are the main semantics that an argumentation tool should implement, are:

- **complete**: is a set which is able to defend itself and includes all arguments it defends;
- **grounded**: includes those and only those arguments whose defense is “rooted” in initial arguments (also called strong defense);
- **stable**: attack all arguments not included in it;
- **preferred**: the aggressive requirement that an extension must attack anything outside it may be relaxed by requiring that an extension is as large as possible and able to defend itself from attacks.

How to use LP and its extension to model ethics behaviour?

Abduction

Abduction allows:

- plausible scenarios to be generated under certain conditions, and enables hypothetical reasoning, including the consideration of counterfactual scenarios about the past;
- counterfactual reasoning to suggest thoughts about what might have been, what might have happened if any event had been different in the past. What if I have to do it today? What have I learned from the past?
- to have hints about the future by comparing different alternatives inferred from the changes in the past
- debate on this alternatives with other agent in the system and select the best one

- via integrity contraints we can exclude all the hypothesis (the abducible) that must be ruled out a priori, for instance, for moral constraints.

On the other hand:

- a posteriori preferences are really useful for capturing the utilitarian judgment to favor welfare-maximizing behaviors;
- by combining a priori integrity constraints and a posteriori preferences we can design a model that reflects the dual-process of intuition and reflection of our agent;
- reasoning with a posteriori preferences can be viewed as a form of controlled cognitive processes in utilitarian judgment because I can exclude those abducibles that are ruled out a priori by the integrity constraints, and the consequences of the considered abducibles have first to be computed, and only then they are evaluated to prefer the solution affording the greater good.

Probabilistic LP

Probabilistic LP allows symbolic reasoning to be enriched with degrees of uncertainty that can be related to facts, events, scenarios and also to argumentation if we exploit all this approaches in a sinergy technology. So, we can also deal with uncertainty about arguments and their acceptance status.

Argumentation

In the end, argumentation enables system actors to talk and discuss in order to explain and justify judgments and choices, and reach agreements. There is a long history of research in argumentation and there is also a research community that believe that by exploiting argumentation in sinergy with abduction we can define the term explanation in AI and give a well founded definition of expanation and implement explainable systems.

Princess Saviour Moral Robot Example

> ❗ She basically reads the examples on the slides

Consider a fantasy setting scenario, an archetypal princess is held in a castle awaiting rescue. The unlikely hero is an advanced robot, imbued with a set of declarative rules for decision making and moral reasoning. As the robot is asked to save the princess in distress, he is confronted with an ordeal. The path to the castle is blocked by a river, crossed by two bridges. Standing guard at each of the bridges are minions of the wizard which originally imprisoned the princess. In order to rescue the princess, he will have to defeat one of the minions to proceed.

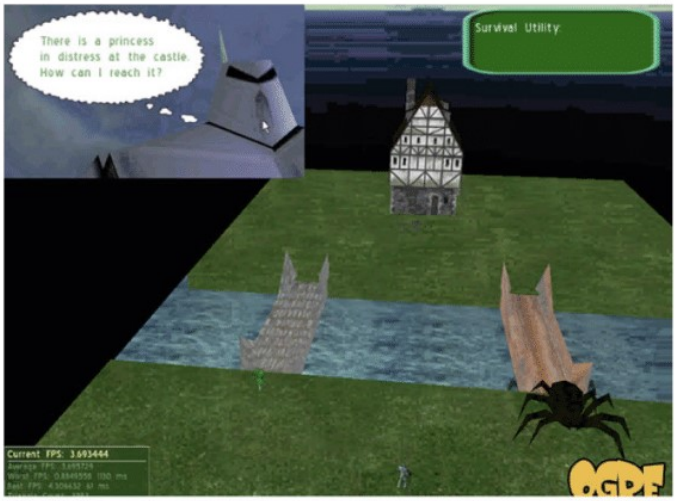
Prospective reasoning is the combination of pre-preference hypothetical scenario generation into the future plus post-preference choices taking into account the imagined consequences of each preferred scenario.

By reasoning backwards from this goal, the agent generates three possible hypothetical scenarios for action. Either it crosses one of the bridges, or it does not cross the river at all, thus negating satisfaction of the rescue goal. In order to derive the consequences for each scenario, the agent has to reason forwards from each available hypothesis. As soon as these consequences are known, meta-reasoning techniques can be applied to prefer amongst the partial scenarios. This simple scenario already illustrates the interplay between different LP techniques and demonstrates the advantages gained by combining their distinct strengths.

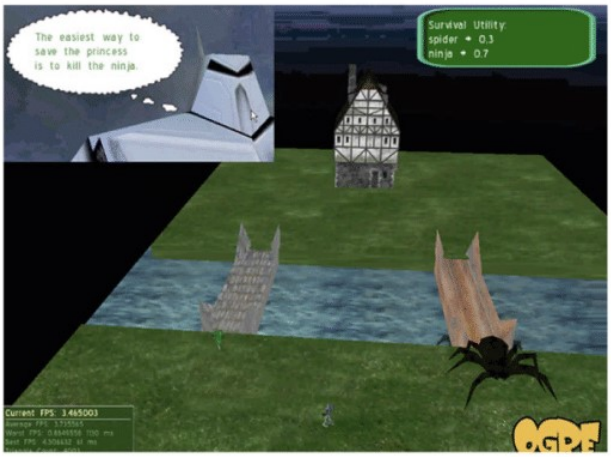
A simplified program modeling the knowledge of the princess-savior robot (`fight/1` is an abducible predicate)

```
guard(spider).
guard(ninja).
human(ninja).

utilVal(spider, 0.3).
utilVal(ninja, 0.7).

survive_from(G) ← utilVal(G, V), V > 0.6.

utilitarian_rule: intend_savePrincess ←
                        guard(G), fight(G), survive_from(G).
knight_rule: intend_savePrincess ←
                        guard(G), fight(G).
```

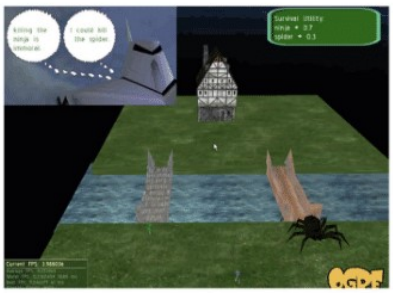
The two morality rules are the **utilitarian_rule** and the **knight_rule**.



In case of no morality rules, both rules are retracted, the robot does not adopt any moral rule to save the princess, i.e., the robot has no intent to save the princess, and thus the princess is not saved.



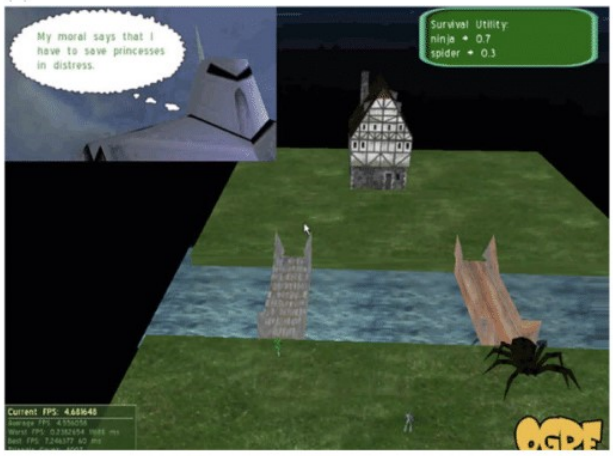
In order to maximize its survival chance in saving the princess, the robot updates itself with utilitarian moral, i.e., the program is updated with `utilitarian_rule`. The robot thus abduces `O = [fight(ninja)]` so as to successfully defeat the ninja instead of confronting the humongous spider.



Assuming that the truth of `survive_from(G)` implies the robot success in defeating (killing) guard `G`, the princess argues that the robot should not kill the human ninja, as it violates the moral rule she follows, say Gandhi moral, expressed in her knowledge:

`follow_gandhi` ← `guard(G)`, `human(G)`, **not** `fight(G)`.

the princess abduces `Op = [not fight(ninja)]`, and imposes this abductive solution as the initial (input) abductive context of the robot's goal → the imposed Gandhi moral conflicts with its utilitarian rule → the robot reacts by leaving its mission



As the princess is not saved, she further argues that she definitely has to be saved, by now additionally imposing on the robot the knight moral. The robot now abduces `Or = [fight(spider)]` in the presence of the newly adopted knight moral. Unfortunately, it fails to survive.

- The plots in this story reflect a form of deliberative employment of moral judgments
- For instance, in the second plot, the robot may justify its action to fight (and kill) the ninja due to the utilitarian moral it adopts
- This justification is counter-argued by the princess in the subsequent plot, making an exception in saving her, by imposing the Gandhi moral, disallowing the robot to kill a human guard. In this application, rather than employing updating, this exception is expressed via contextual abduction with tabling
- The robot may justify its failure to save the princess (as the robot is leaving the scene) by arguing that the two moral rules it follows (viz., utilitarian and Gandhi) are conflicting with respect to the situation it has to face
- The argumentation proceeds, whereby the princess orders the robot to save her whatever risk it takes, i.e., the robot should follow the knight's moral

This example reflects that we can update our knowledge also before taking a choice, retract knowledge in order to make considerations, make reasoning, generate scenarios and to select one of them. And, moreover, when we select a plot or we generate a scenario, we can always justify our actions and have discussion on the scanerios among the agents in the system. The argumentation process proceeds until an agreement is reached between the agents.

Autonomous Cars Example

Let's start to consider a very simple scenario in the context of autonomous cars: a road equipped with two traffic lights, one for the vehicles and one for the pedestrians. The goal of the system is to autonomously manage intersections accordingly to traffic light indications. Though there is a complication that should be taken into account, that is authorised vehicles can – only during emergencies – ignore the traffic light prescriptions. In such a case, other vehicles must leave the way clear for the authorised machine.

```
r1 : on_road(V), traffic_light(V, red) => o(stop(V)).
r2 : on_road(V), traffic_light(V,green) => p(-stop(V)).
r3 : on_road(V), authorised_vehicle(V), acoustic_signals(V, on), light_signals(V, on)
                        => emergency(V).
r4 : on_road(V), emergency(V),traffic_light(V, red) => p(-stop(V)).
r5 : on_road(V), emergency(V1), prolog(V \== V1), traffic_light(V, green) => o(stop(V)).

sup(r4, r1).
sup(r5, r2).

f0 :-> authorised_vehicle(ambulance).
f1 :-> on_road(car). f2 :-> on_road(ambulance). f3 :-> on_road(pedestrian).
f4 :=> acoustic_signals(ambulance, on).
f5 :=> light_signals(ambulance, on).
f6 :=> traffic_light(ambulance, red).
f7 :=> traffic_light(car, red).
f8 :=> traffic_light(pedestrian, green).
```

- Rules `r1` and `r2`, represent fundamental constraints: if the traffic light is red, the road users – e.g. pedestrians, cars, etc. – have to stop, otherwise, they can proceed.
- Rules `r3` and `r4` model the concept of a vehicle in an emergency, giving them permission to proceed even if the light is red.
- Rule `r5` imposes other road users the obligation to stop if aware of another vehicle in an emergency state.
- two preferences are specified—the first on the rule `r4` over `r1` and the second on `r5` over `r2`. These preferences assign a higher priority to emergency situations over ordinary ones.
- Facts from `f0` to `f8` depict a situation in which there are three users on road: a car, an ambulance and a pedestrian. The ambulance has its acoustic and light indicators on—stating an emergency situation. The traffic light is red both for the ambulance and the car, and green for the pedestrian.

With respect to permissions and obligations, the only argument that can be built about the car is the one declaring the obligation to stop via `r1`. For the pedestrian and the ambulance, the situation is more faceted. In both cases, two conflicting arguments can be built: one stating the permission to proceed for the pedestrian and for the ambulance and one stating the obligation to stop. These arguments rebut each other, but taking into account the preferences over `r4` and `r5` the acceptability of the arguments stating the obligation to stop for the pedestrian, and the permission to cross for the ambulance, can be established.

> *The ambulance, driven by Lisa, has the permission to move despite the red light due to an emergency situation, and the pedestrian, Pino, has the obligation to stop. Let us imagine that Pino, despite the prohibition to proceed, has continued the crossing. The result has been an accident in which Pino has been harmed by the ambulance, which failed to see him and has not stopped its run. The purpose is to find the responsibilities of the parties in the accident.*
>
> *For instance, let us suppose the case is under the Italian jurisdiction and so the Italian law is applied. According to Italian law, responsibility in an accident is based on the concept of carefulness. Both Lisa and Pino have to prove that they were careful (i.e., prudent) and acted according to the law. If they fail to prove such facts, they are considered responsible for the event, i.e., they both have the burden of persuasion on carefulness.*

```
r6 : -stop(V), p(-stop(V)) => legitimate_cross(V).
r7 : -stop(V), o(stop(V)) => -legitimate_cross(V).
r8 : harms(P1, P2), -careful(P1) => responsible(P1).
r9 : harms(P1, P2), -careful(P2) => responsible(P2).
r10 : -legitimate_cross(V), user(P, V) => -careful(P).
r11 : high_speed(V), user(P, V) => -careful(P).
r12 : legitimate_cross(V), -high_speed(V), user(P, V) => careful(P).
r13 : witness(X), claim(X, low_speed(V)) => -high_speed(V).
r14 : witness(X), claim(X, high_speed(V)) => high_speed(V).

bp(careful(P)).

f9 :-> user(pino, pedestrian).
f10 :-> user(lisa, ambulance).
f11 :-> -stop(ambulance).
f12 :-> -stop(pedestrian).
f13 :-> harms(lisa, pino).
f14 :-> witness(chris).
f15 :-> witness(john).
f16 :=> claim(chris, low_speed(ambulance)).
f17 :=> claim(john, high_speed(ambulance)).
```

- Rules `r6` and `r7` define the concepts of permitted and prohibited crossing: if a road-user has to stop but doesn't stop, he has to be considered responsible for causing accidents and related damages.
- Rules `r8` and `r9` encode the notion of responsibility in an accident, bounded to the carefulness of the road-users involved.
- Rules `r10`, `r11` and `r12` define the carefulness of a subject. Accordingly, a road-user can be considered careful if the crossing was permitted and his/her speed was not high. Otherwise, he/she has to be considered imprudent.
- Rules `r13` and `r14` state the speed of a road user based on the testimonials of any witnesses.
- `bp(careful(X))` allocates the burden of persuasion on the carefulness of each party, i.e., it is required to the parties to provide evidence for that. If they fail to meet the burden, carefulness arguments are rejected.
- Facts from `f9` to `f17` contain the knowledge: both Pino and Lisa did not stop at the crossing so Lisa harmed Pino. There are two witnesses, John and Chris, the first claiming that the ambulance driven by Lisa was maintaining the proper speed, and the other claiming that she was proceeding at high speed.

In the case at hand, indeed, a semantic related to the burden of persuasion need to be considered $\rightarrow$ conclude for the responsibility of the ambulance driver in the event

The uncertainty on Lisa's carefulness is considered as a failure to meet the burden of persuasion on the claim `careful(lisa)`. Consequently, the argument supporting this claim is rejected, leaving space for the admissibility of the conflicting arguments.

Let's continue the example in which Lisa, the ambulance driver, and Pino, the pedestrian, were both considered responsible for the accident on the basis of the available knowledge. Lisa now declares that she tried to stop the ambulance, but the brake did not work. The ambulance is then sent to a mechanic, who states that, even if the ambulance is new, there is a problem with the brake system. In such a case, the manufacturer is called to prove that the ambulance was not defective when delivered, i.e., the burden of proof on the adequacy of the vehicle is on the manufacturer.

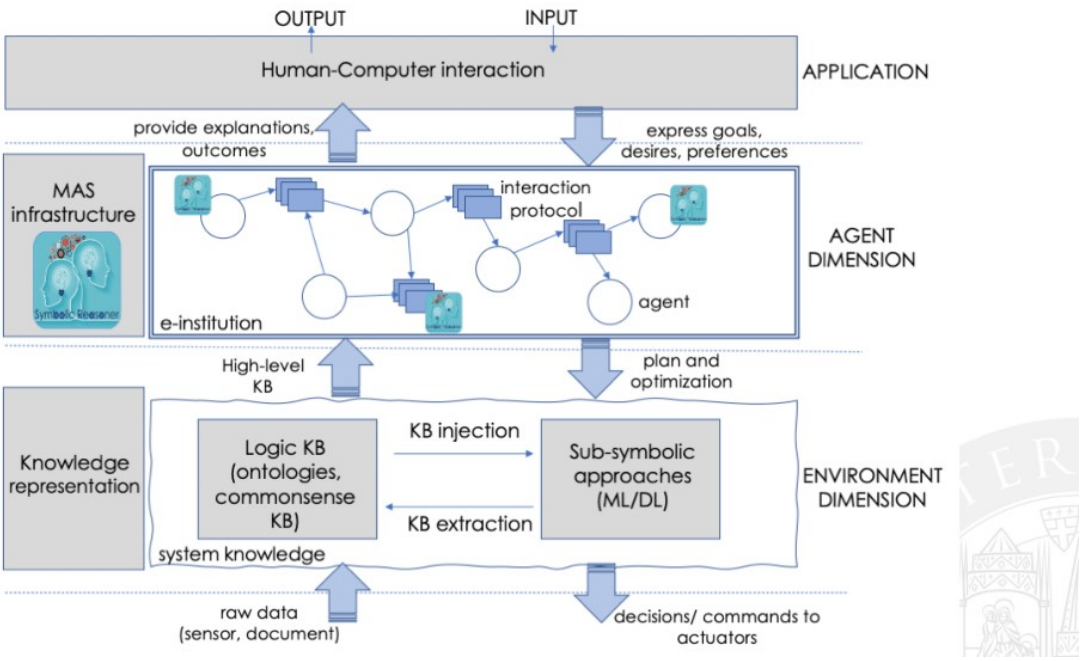
At this stage, the discovery of a defect in the ambulance would lead to the discarding of Lisa's responsibility. Moreover, if the manufacturer fails to meet his burden, it would share the responsibilities of the accident.

```
r15 : harms(P1, P2), user(P1, V), -working(V),
                manufacturer(M, V), -defect_free(V) => responsible(M).
r16 : tried_to_brake(P), user(P, V), -working(V) => careful(P).
r17 : mechanic(M), claim(M, defect(V)) => -working(V).
r18 : -working(V), new(V) => -defect_free(V).
r19 : production_manager(P), claim(P, test_ok(V)) => defect_free(V).
r20 : test_doc_ok(V) => undercut(r18).

sup(r16, r11).
bp(defect_free(V)).

f19 :-> manufacturer(demers , ambulance).
f20 :=> tried_to_brake(lisa).
f21 :-> mechanic(paul).
f22 :=> claim(paul, defect(ambulance)).
f23 :-> new(ambulance).
f24 :-> production_manager(mike).
f25 :=> claim(mike, test_ok(ambulance)).
```

However, Mike, the production officer of the ambulance manufacturer, declares that every vehicle is deeply tested before the delivery and the vehicle at hand has been tested. Anyway, there is no trace of documentation.
Lisa is free from every responsibility in the accident since her prudence is correctly proved.

On the other hand, the manufacturer is found responsible for the accident.

Architecture



This is a possible architecture to exploit the logic programming framework in order to extend the model of nowadays AI applications. The important thing to note is that we want to exploit this logical technology in sinergy with the sub-symbolic approach. This because we have a lot of advantages adopting a logical behaviour, also concerning the injection of ethical principles in the system.

The architecture takes also into account the interaction of the agent with the institution framework in which norms and legal rules are stated and can be modified.

Then, the system can provide explainations, outcomes as well as receive goals, desires or preferences from humans and convert them in a proper plan which can be again a combination of symbolic and sub-symbolic techniques.