Lesson 1. part II 14/10/2021

Want to be able to express things that are not in our prepositional predicate case

**Rules of programming**

We read the rule H ← B as “*if B is true then H is true*”. If the body is true then the head is true. If you manage to prove than B is true than H is true

This **unidirectionally** propagates knowledge from B to H

But there are **no limits** (trivial models). Construct some trivial models

*Platypus are* ***not*** *ducks;* this constraint is limited. Two classes that are disjunct we can’t put a limit

**Domain closure**

Rules can never introduce new objects

The variable on the head must appear on the body

Cannot represent knowledge with **anonymous individuals** *every parent has a child* (even if you don’t know the child)

**Language extensions**

We want to extend predicate rules to express

* negated facts
* Disjointness of properties/constraints
* Anonymous individuals

Removing some of the constraints on rules

**Negations**

consider the rule

← P(a)

← P(a), Q(b), R(a,b)

↑ those rules were not allowed, we could have an empty body but we needed something in the head

those rules translated to clauses:

( ¬ P(a))

( ¬ P(a) v ¬ Q(b) v ¬ R(a,b)) one of this need to be false, can’t have all tree facts together

The facts in the body cannot appear all together

Do they need to be facts?

← platypus(phineas)

← platypus(x), ducs(x)

← biped(x), mammal(x), oviparous(x) can have at most 2 properties but not 3 true

**Anonymous objects**

Allow new variables in the head of rules

HasChild(x,y) ← parent(x)

New variables are **existentially quantified**

For all x, if x is a parent, then **there exist an object y** which is a child of x

∀ x ∃ y. If x is a parent then exist a y that is the child

← HasChild(x,x) specify that nobody is a child of himself

**Reasoning**

These may seems like very basic extensions but the effect reasoning (technique)

**Inconsistency**

Constraint ran may lead to contradictory knowledge

oru knowledge is contradictory

Platypus(perry) ←

Duck(perry) ←

← platypus(x), duck(x) no object can be a platypus and a duck in the same time

can’t have all those 3 things true in the same time

*Ex falso quodlibet*: from a contradiction you can derive anything you want

we have no model, can’t construct a canonical model

**Infinite models**

Anonymous object may propagate

HasParent(x,y) ← person(x)

person(y) ← HasParent(x,y), person(x)

person(herb) ←

the issue with anonymous object is that they propagate, with big knowledge can be hard

every person have a parent (don’t know who is, but you know they have a parent)

if a person have a parent then that parent is a person

since herb is a person herb have a parent y (don’t know who is)

y have a parent y1 and it propagates to y2 since y2 is a person it has a parent …

it is an infinite model (can’t be constructed and it is impossible to do reasoning)

18. Oct. 2021

Reasoning with inconsistency can be problematic

Infinite models: lack of anonymous object

To represent knowledge we need to introduce objects that must exist but we don’t know

Problem: the introduction of anonymous Brecht might lead to infinite model

**Inconsistency in anonymous object**

When we combine anonymous variables and constraints inconsistencies may be hidden in the anonymous part

Can we stop the construction of the canonical model early? Want to stop to avoid an infinite model, the information is the same and they will act in the same way, in some case we can stop the construction by introducing additional features, sometimes is possible to

How much will it grow? Is it useful?

I can stop the canonical model to be finite instead of infinite. If it grows too much it increases the complexity, if it is too big it can't be constructed.

**Too much freedom**

It turns out that these existential rules are well behaved for reasoning about facts

Well behaved: we can do reasoning

In theory we can always reasoning

But they are difficult to analyse formally: their syntax is too free

The syntax of our rules is too free, we are allowed to express a lot of things

We can have in the body as many variables as we want, can have very complex things

Arbitrary combinations of predicates in the body, the body can be very complicated, can be very long, and inside a lots of variables

New variables in head, these new variable can have a lot of liberty, different body can have different consequences

We have a lot of liberty in our language, if we want to find a well behaved language we need to limit the possibility of the language

**Reasoning about knowledge**

What if we care about general congruences rather than specific facts?

Suppose I have all my rules, these is a general knowledge we + **completare**

Are all bipedal mammals primates? We can do it with our rules, it is not always possible

**EL⊥**

We will introduce a new KR language which

* limits the expressively of predicate rules
* Allows for constraints. Allows to introduce others features and limitations for not happenings things together
* Allows for anonymous objects
* Has an effective and efficient reasoning methods. Reasoning can be done with a very precise algorithm

This language EL⊥ belongs to the family of **Description logics (DLs)**

**What are DLs?**

Description logics are a **family** of KR (knowledge representation) languages

Before it was taking existing logics and try to use them to represent some knowledge, DLs was design to represent knowledge

They are build in a way to ***+compeltare***

Characterised by

* clear syntax. Very precise and well defined. Totally different syntax
* Formal **un**ambiguous semantics. The semantics should not leave space for multiple interpretations. Precise meaning for all the formulas

**A bit of history**

Description logics evolved from semantics networks as language to reprint the terminological knowledge of a domain

Didn’t have a formal understanding or meaning

DLs are an evolution, try to represent classes but now we introduce a precise meaning for connections, have one and only one meaning

Represent the properties of the objects, represent the terms and create a dictionary, try to describe the vocabulary, introduces classes

Describe the vocabulary (classes and their relationship) specificity restrictions on their interpretations

We have two classes, primates and mammals, in therms of symbols they are different, no relations between them

The vocabulary are describes to constraints, we are limiting the interpretations, the constraints are usually not complete

If I know that somebody is a parent there is a liberty, don’t know how many children, **open world assumptions, whatever is not especially stated doesn't mean is false**

Clause of assumption: whatever doesn't follow from my knowledge is false

DLs if you can’t prove then you don’t know, it is only false if you can prove it it false

**KL-ONE**

An early representation and reasoning system was KL-ONE

Early system that implement reasoning techniques still used by modern system

It was considered inefficient but it performed well, it showed that the techniques could be used in practice

First DLs were called KL-ONE-like

**Trade-offs**

The family structure answered to the known trade-off between

* expressivity and
* Complexity

If I can say more things I need to do more spots, the algorithm is more complex…

If I extend my expressivity it will take longer to do reansong, if you want a fast system you need to limit it

More expressivity means more cases to analyse during reasonings, mean more time and more memory

**Trade off II**

Rather then defining

* a very expressive language including all possible constructors, or
* A language with very efficient reasoning services

Either very expressive or very efficient

DLs study a whole spectrum has the languages that goes from low expressivity to high one

Spectrum: dosen’t meaning a linear relationship we have also incomparable cases

Pick and mix depending on the specific use-case

Depending on the use case you should pick the right language

There is no language useful for everyone, depending on what you are trying to represent (and the resources that you have) you can pick the right language

If you are getting a lot of data you need a system that will give you the answer in few seconds, on the other hand if you are doing reansong to decide if you are going to implement a system for the next year doesn't matters if it takes 2 days for an answer

**Global constraint**

The scope of DLs is to provide reasoning

The main goal is to have decidable reasoning services

There are sound, complete and terminating algorithm

It has to give the answer in finite time

We have to guarantee it will stop

Most DLs fall wishing the **two-variable fragment** of first-order predicate logic

Two variable fragment: unwary and binary, the fragment is that every formulas used at most two variables, can express formulas that have to variables, anything can be representing only using two variables

**First DLs**

The first DL was called attribute language (AL)

Which was fast extended to include complements (ALC)

Complements ↔ negations

ALC can represent complex classes

* ¬ mortal class of immortal (**non** mortal) objects
* Green ∏ fragile object that are green **and** fragile (conjunction)
* ∃hasChild.Female people with female children. It is a relationship between classes (maybe one female and one boy)
* ∀ hasChild.Female people with **only** female children

**First results**

First theoretical results:

* complexity results
* Optimal algorithms
* Relationship to other logics

**The push for growth**

At first, the goal was to express more

How much can we push expressivity while remaining decidable?

Example

* quantification (at most three children)
* Inverse relations (hasChild as inverse of hasParent)
* Transitivity (hasAnchestor). If we have a person A that have an ancestor B or that person have an ancerts C than A have C as ancestor
* ***completare names***

**The counter flow**

Small is beautiful again

Take away some constructor

EL (only conjunction ∏ and ∃)

Fl0 (only conjunction ∏ and ∀ )

Dl lite

Turned out to be very useful for some applications

**The semantic web**

The W3C chose OWL (based on an expressive DL) as the standard ontology language for semantic web

OWL is based on a very expressive language

Introduce several profiles

OWL 2 EL (based on ELt)

OWL 2 QL (based on DL-lite)

OWL 2 RL (based on rules)

…

**Further developments**

Non standard reasoning:

* explanations
* Inconsistency-tolerance
* …

Non-standard extensions

* time
* Uncertainty
* …