

Logic programming and description logics

G. Falquet,

Source :

B. Groszof et al. Description Logic Programs : combining Logic Programs with Description Logic. In proc. WWW2003, Budapest, May 2003.

Motivations

- ▶ Add a system of inference rules to the Description logics (logic programming)
- ▶ Rule bases for reasoning that refer to the vocabulary defined in an ontology
- ▶ Ability to make requests on instances (not very effective in DL)
- ▶ Use logic programming techniques to access relational databases
- ▶ Logical description of Web services
 - ▶ ontology for the categorisation of services and their i/o
 - ▶ representing business rules and the relationship between input and output

First Order (Predicate) Logic (FOL) and ontologies

Defining the semantics of a DL by showing its translation to FOL

DL	FOL
Individual	Constant
Class (expressions)	Formula with one free variable
Property	Formula with two free variables
$C \sqsubseteq D$	$\forall x(C(x) \Rightarrow D(x))$
$C(a)$	ground atom $C(a)$
$P(a, b)$	ground atom $P(a, b)$
transitivity of P	$\forall x, y, z(P(x, y) \wedge P(y, z)) \Rightarrow P(x, z)$
functionality of P	$\forall x, y, z(P(x, y) \wedge P(x, z)) \Rightarrow y = z$
Q is the inverse of P	$\forall x, y(P(x, y) \Leftrightarrow Q(y, x))$
...	...

Expression equivalence

DL	FOL
$C \sqcap D$	$C(x) \wedge D(x)$
$C \sqcup D$	$C(x) \vee D(x)$
$\neg C$	$\neg C(x)$
$\exists P.C$	$\exists y(P(x, y) \wedge C(y))$
$\forall P.C$	$\forall y(P(x, y) \Rightarrow C(y))$
$\geq nP.C$	$\exists y_1, \dots, y_n$ $P(x, y_1) \wedge C(y_1) \wedge \dots \wedge P(x, y_n) \wedge C(y_n)$ $\wedge y_1 \neq y_2 \wedge \dots \wedge y_{n-1} \neq y_n$

Logic Programs

Set of rules each having the form

$$H \leftarrow B_1 \wedge \dots \wedge B_m \wedge \sim B_{m+1} \wedge \dots \wedge \sim B_n$$

where

H and the B_i 's are atoms ($Predicate(term_1, \dots)$)

" \sim " means "negation as failure" $\sim B_i$ means " B_i is not believed" (i.e., is unknown or false)

(variables are implicitly quantified by \forall)

Definite LP and Horn clauses

A definite logic program is a LP without \sim .

A rule in a definite LP is a definite Horn clause, it can be written as

$$H \vee \neg B_1 \vee \dots \vee \neg B_m$$

or equivalently as

Example

$grandParent(x, y) \leftarrow parent(x, z) \wedge parent(z, y)$

$parent(a, b)$

$parent(b, c)$

$connected(x, y) \leftarrow line(x, y)$

$connected(x, y) \leftarrow line(x, z) \wedge connected(z, y)$

$line(a, b), line(b, c), line(c, d), line(b, e), line(e, a)$

Semantics

- ▶ Let HB stand for the Herbrand base of the logic program \mathcal{R} .
 - ▶ all the atoms $P(t_1, \dots, t_k)$ where the t_i s are ground terms (built with functions and constants)
- ▶ The conclusion set is the smallest subset \mathcal{S} of HB such that for any rule
 $H \leftarrow B_1 \wedge \dots \wedge B_m,$
if $B_1 \wedge \dots \wedge B_m \in \mathcal{S}$ then $H \in \mathcal{S}$.

Example

Meaning of the program

1. $k(x, y) \leftarrow \text{line}(x, y)$
2. $k(x, y) \leftarrow \text{line}(x, z) \wedge k(z, y)$
3. $\text{line}(a, b), \text{line}(b, c), \text{line}(c, d), \text{line}(c, a)$

By (3) S must contain $\text{line}(a, b), \text{line}(b, c), \text{line}(c, d), \text{line}(c, a)$

By (1) it must contain $k(a, b), k(b, c), k(c, d), k(c, a)$

By (2) it must contain $k(a, c), k(b, d), k(b, a), k(c, b)$

By (2) it must contain $k(a, d), k(a, a), k(b, b), k(c, c)$

Expressive power

def-LP is a mildly weaker version of the def-Horn ruleset.

Every conclusion of the def-LP must have the form of a fact.

By contrast, the entailments of the def-Horn ruleset are not restricted to be facts.

Example. suppose \mathcal{RH} consists of the two rules

1. $kiteDay(Tues) \leftarrow sunny(Tues) \wedge windy(Tues)$
2. $sunny(Tues)$

Then it entails $kiteDay(Tues) \leftarrow windy(Tues)$, a non-unit derived clause.

Limitations of the Description Logics

- ▶ Usual DLs correspond to a very limited version of the FOL (guarded quantifiers)
- ▶ We cannot express the fact that an individual must be connected to another individual (anonymous) via two different paths.
- ▶ For example, a local worker works and lives in the same city.
- ▶ Easy to express in LP

$$\text{localWorker}(x) \leftarrow \text{worksFor}(x, y) \wedge \text{locatedAt}(y, z) \wedge \text{livesIn}(x, z)$$

Limitations of def-Horn

All the variables are universally quantified

Impossible to assert the existence of individuals that are unknown

Example. “every person has a (biological) mother”

Easy in DL

$$\text{Personn} \sqsubseteq \exists \text{mother}.\top$$

Impossible with Horn clauses

Absence of negation and existential quantifier \Rightarrow impossible to represent statements like

Every person is either a man or a woman, but not both

Which is easy in DL :

$$Person \sqsubseteq Man \sqcup Woman$$

$$Man \sqsubseteq \neg Woman$$

No equality \Rightarrow impossible to represent functional properties

Recursive mapping \mathcal{T} from DL to def-Horn

$Expr$	$\mathcal{T}(Expr)$	
$C \sqsubseteq D$	$D(x) \leftarrow C(x)$	subproperty
$Q \sqsubseteq P$	$P(x, y) \leftarrow Q(x, y)$	
$\top \sqsubseteq \forall P.C$	$C(y) \leftarrow P(x, y)$	range restriction
$\top \sqsubseteq \forall P^-.C$	$C(y) \leftarrow P(y, x)$	domain restriction
$C(a)$	$C(a)$	indiv. assertion
$P(a, b)$	$P(a, b)$	prop. assertion
$C \equiv D$	$D(x) \leftarrow C(x)$ $C(x) \leftarrow D(x)$	equivalence
$P^+ \sqsubseteq P$	$P(x, z) \leftarrow P(x, y) \wedge P(y, z)$	transitivity of P

... class construction

$C_1 \cap C_2 \subseteq D$	$D(x) \leftarrow C_1(x) \wedge C_2(x)$	
$C \subseteq D_1 \cap D_2$	$D_1(x) \leftarrow C(x)$ $D_2(x) \leftarrow C(x)$	
$C_1 \sqcup C_2 \subseteq D$	$D(x) \leftarrow C_1(x)$ $D(x) \leftarrow C_2(x)$	
$C \subseteq D_1 \sqcup D_2$	impossible	

...class construction with quantifiers

$C \sqsubseteq \forall P.D$	$(D(y) \leftarrow P(x, y)) \leftarrow C(x)$ $\equiv D(y) \leftarrow C(x) \wedge P(x, y)$	
$\forall P.C \sqsubseteq D$	impossible	
$\exists P.C \sqsubseteq D$	$D(x) \leftarrow P(x, y) \wedge C(y)$	
$C \sqsubseteq \exists P.D$	impossible	

DHL Ontologies

A DHL ontology is a set of axioms of the form $C \sqsubseteq D$, $A \equiv B$, $Q \sqsubseteq P$, $\top \sqsubseteq \forall P.C$, $\top \sqsubseteq \forall P^-.C$, $C(a)$, $P(a, b)$, $P^+ \sqsubseteq P$.

with restrictions on $C \sqsubseteq D$

- ▶ no \sqcup or \exists in the right-hand side
- ▶ no \forall in the left-hand side
- ▶ no \neg, \leq, \geq

Theorem

A DHL ontology can be translated to def-Horn while preserving its semantics (same models and logical consequences)

DLP

- ▶ We say that a def-logic program \mathcal{RP} is a Description Logic Program (DLP) when it is the LP-correspondent of some DHL ruleset \mathcal{RH} .
- ▶ A DLP is directly defined as the LP-correspondent of a def-Horn ruleset that results from applying the DHL \rightarrow def-Horn.
- ▶ Semantically, a DLP is the f-weakening of that DHL ruleset
- ▶ The DLP expressive class is thus the expressive f-subset of DHL.
- ▶ By Theorem 1, DLP can be viewed precisely as an expressive subset of DL.
- ▶ expressively DLP is contained in DHL which in turn is contained in the expressive intersection of DL and Horn.

Practical consequences

An OWL profile, like OWL-RL, that satisfies the DHL syntactic restrictions

- ▶ can be translated to a def-Horn program
- ▶ all the inferences can be obtained with a rule-based system

OWL 2 RL¹

An OW 2 profile with syntactic restrictions

- ▶ Aimed at efficient reasoning with rule-based systems

With a set of inference rules for reasoning

- ▶ complete reasoning for the OWL 2 RL profile (see Theorem PR1 in [1])
- ▶ incomplete reasoning for OWL 2

1. [1] https://www.w3.org/TR/owl2-profiles/#OWL_2_RL

Sample rules

```
T(?x, owl:someValuesFrom, ?y)
T(?x, owl:onProperty, ?p)
T(?u, ?p, ?v)
T(?v, rdf:type, ?y)
-----
T(?u, rdf:type, ?x)
```

```
T(?c, owl:unionOf, ?x)
LIST[?x, ?c1, ..., ?cn]
T(?y, rdf:type, ?ci)
-----
T(?y, rdf:type, ?c)
```