FO(ID) as an extension of DL with rules

Joost Vennekens, Marc Denecker

Introduction

Topic

Extending Description Logic with rules

- Hot topic currently
- Questions from a knowledge representation point-of-view:
  - Which kind of rules?
    - Default rule
    - Rewrite rules
    - Inference rules
    - ...
  - What do these rules precisely mean? (formally and informally)
  - How do they complement DL?

Our answer

$$\begin{array}{cccc} \text{DL} & + & \text{Rules} & \\ \cap & & \cap & \\ \text{FO} & (& \text{ID} &) \end{array}$$

- We have developed a language FO(ID) that
  - extends first-order logic
  - with a rule-based representation of inductive definitions
- It induces a way of extending DL with definitional rules

Outline

FO(ID)

From FO(ID) to DL(ID)

An example

Inductive definitions

http://wikipedia.org/wiki/Inductive_definition

The prime numbers can be defined as consisting of:

- 2, the smallest prime;
- each positive integer which is not evenly divisible by any of the primes smaller than itself.

In FO(ID):

$$\left\{ \begin{array}{l} \forall x \ \mathit{Prime}(x) \leftarrow x = 2 \\ \forall x \ \mathit{Prime}(x) \leftarrow x > 0 \land \neg \exists y \ y < x \land \mathit{Prime}(y) \land \mathit{Divisible}(x, y) \end{array} \right\}$$

Components:

Inductive definitions

http://wikipedia.org/wiki/Inductive_definition

The prime numbers can be defined as consisting of:

- 2, the smallest prime;
- each positive integer which is not evenly divisible by any of the primes smaller than itself.

In FO(ID):

$$\left\{ \begin{array}{l} \forall x \; \underbrace{Prime(x)} \leftarrow x = 2 \\ \forall x \; Prime(x) \leftarrow x > 0 \land \neg \exists y \; y < x \land Prime(y) \land Divisible(x, y) \end{array} \right\}$$

Components:

- Relation(s) being defined: predicate(s) in head

Inductive definitions

http://wikipedia.org/wiki/Inductive_definition

The prime numbers can be defined as consisting of:

→ - 2, the smallest prime;
→ - each positive integer which is not evenly divisible by any of the primes smaller than itself.

In FO(ID):

$$\begin{Bmatrix} \rightarrow & \forall x \ \underbrace{Prime(x)} \leftarrow x = 2 \\ \rightarrow & \forall x \ \underbrace{Prime(x)} \leftarrow x > 0 \land \neg \exists y \ y < x \land Prime(y) \land Divisible(x, y) \end{Bmatrix}$$

Components:

- Relation(s) being defined: predicate(s) in head
- Set of cases in which it holds: each case is definitional rule

Inductive definitions in FO(ID)

- An inductive definition is a set of rules

$$\forall \mathbf{x}\ P(\mathbf{t}) \leftarrow \varphi,$$

  with φ an FO formula
- Defines certain relations (e.g. *Prime*) in terms of some other relations (e.g. $<$ and *Divisible*)

Formal semantics

- $=$ (parametrized) well-founded semantics from Logic Programming
- Coincides with intuitive meaning of ID

FO(ID)

- Extends FO with IDs:

Formula of FO(ID) = either FO formula or inductive definition

$$\left\{\begin{array}{l} \forall x \; Reachable(x) \leftarrow Initial(x) \\ \forall x \; Reachable(x) \leftarrow \exists y \; Reachable(y) \land Transition(y, x) \end{array}\right\}$$

$$\forall x \; Reachable(x) \land \neg Final(x) \Rightarrow \exists y \; Transition(x, y)$$

$$\ldots$$

- Extends expressive power of FO, which cannot represent induction

FO(ID) versus DL

Formally

(most) DLs $\subseteq$ FO $\subseteq$ FO(ID) and FO(ID) $\supseteq$ nonnested μ, $\cdot^+$

Knowledge representation

DL	FO(ID)
TBox	
▶ Define concepts ($\equiv$)	Definitions ($\{\leftarrow\}$)
▶ Assert relations between concepts ($\sqsubseteq$)	FO ($\subset$)
ABox	
▶ Assert elements of relations	FO (atoms)

Outline

Motivation

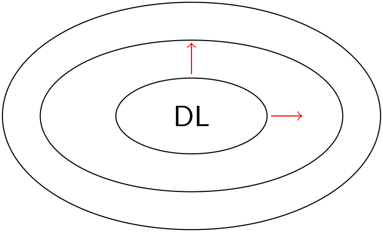
- FO(ID) is strong semantic integration of FO and rules
- In which each component has a clear KR "task"
  - Rules: definitional rules, so define concepts
  - FO: assert additional properties of the defined concepts or of concepts for which you have no definition
- Natural extension of DL

About $\equiv$ in DL handbook

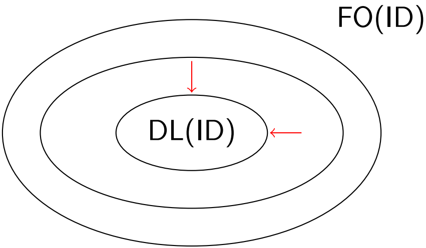
"This form of definition is much stronger than the ones used in other kinds of representations of knowledge, which typically impose only necessary conditions; the strength of this kind of declaration is usually considered a characteristic feature of DL knowledge bases."

FO(ID) as upperbound

Typically:



Our approach:



Reasons for restrictions

DLs are fragments of FO, that are interesting

- Because they enforce concept-centric modeling style
  - What are the relevant concepts?
  - Define some of them
  - Express relations (inclusions) between them
  - Assert facts about objects that belong to the concepts

  For which they offer natural syntactic sugar

- Because they are decidable

Reasons for restrictions

DLs are fragments of FO, that are interesting

- Because they enforce concept-centric modeling style
  - What are the relevant concepts?
  - Define some of them
  - Express relations (inclusions) between them
  - Assert facts about objects that belong to the concepts

  For which they offer natural syntactic sugar

  $\longrightarrow$ Knowledge representation
- Because they are decidable

  $\longrightarrow$ Computation

Concrete proposal: $\mathcal{ALCI}$(ID)

Basic DL $\mathcal{ALCI}$ extended with inductive definitions

- $\doteq$ which translates to singleton definition in FO(ID)
  - Same as $\equiv$ for non-inductive definitions
- $\{\leftarrow\}$ to write definitions with multiple rules
  - Corresponds to FO completion for non-inductive definition
  - Easier to tell structure of definition and to update

And also two new role-constructors:

- Dot connective: $R_1.R_2$
  Stands for: $\exists z\ R_1(x, z) \land R_2(z, y)$

  $$Uncle \doteq Brother.Parent$$

- Cartesian product: $C_1 \times C_2$
  Stands for: $C_1(x) \land C_2(y)$

Properties: Knowledge representation

- DL-like language
- Admitting DL-like modelling style
- Supported with DL-like syntactic sugar
- Only with more expressive definitions

$\longrightarrow$ See example

Properties: Computation

$\mathcal{ALCI}$(ID) is undecidable

If the domain is known and finite, reasoning can still be done

- Reasoning in database context
  (D consists of all object in the database)
- Solving combinatorial problems
  (D is part of the instance that must be solved)

For the other cases

- We have defined a guarded fragment of $\mathcal{ALCI}$(ID)
- Based on guarded fragment of FO(LFP)
- This fragment is decidable

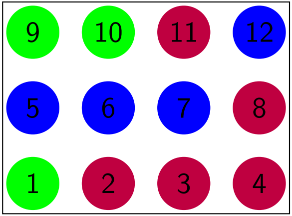
Details in paper

Outline

An example

Game

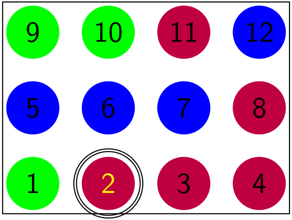
- Select a ball in grid of coloured balls
- All balls in the same colour-group disappear
- Goal: remove all balls, score points by removing large groups



An example

Game

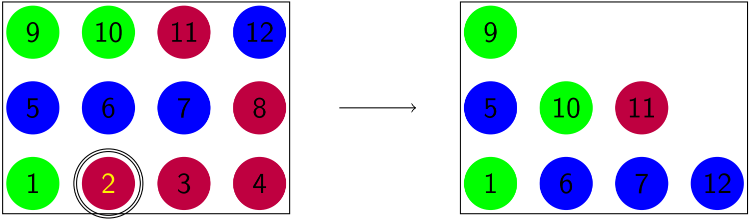
- Select a ball in grid of coloured balls
- All balls in the same colour-group disappear
- Goal: remove all balls, score points by removing large groups



An example

Game

- Select a ball in grid of coloured balls
- All balls in the same colour-group disappear
- Goal: remove all balls, score points by removing large groups



An example (2)

We will define the effect of a single move

Given:

- A grid described by $\mathit{Left}/2, \mathit{Up}/2$
- The colours of the balls: $\mathit{Colour}/2$
- The player's move: $\mathit{Selected}/1$

We write a theory that defines

- The set of remaining balls: $\mathit{Remains}/1$
- The new grid layout by $\mathit{Left}'/2, \mathit{Up}'/2$

The disappearing balls

The balls that disappear are those in the same colour group as the selected ball

- In DL/FO:

$$Disappears \equiv \exists SameColourGroup.Selected$$

$$\forall x\ Disappears(x) \equiv \exists y\ SameColourGroup(x, y) \wedge Selected(y)$$

- In FO(ID)/DL(ID) also:

$$Disappears \doteq \exists SameColourGroup.Selected$$

$$\{\forall x\ Disappears(x) \leftarrow \exists y\ SameColourGroup(x, y) \wedge Selected(y)\}$$

Colour groups

Two balls are in the same colour group

- if they are either adjacent and of the same colour, or
- if there exists a ball they are both already in the same colour group with.

$$\left\{ \begin{array}{l} SameColourGroup \leftarrow SameColour \sqcap Adjacent \\ SameColourGroup \leftarrow SameColourGroup.SameColourGroup \end{array} \right\}$$

$$\left\{ \begin{array}{l} \forall x, y\ SameColourGroup(x, y) \leftarrow SameColour(x, y) \wedge Adjacent(x, y) \\ \forall x, y\ SameColourGroup(x, y) \leftarrow \exists z\ SameColourGroup(x, z) \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad \wedge\ SameColourGroup(z, y) \end{array} \right\}$$

Colour groups

Two balls are in the same colour group

- if they are either adjacent and of the same colour, or
- if there exists a ball they are both already in the same colour group with.

$$\left\{\begin{array}{l} SameColourGroup \leftarrow SameColour \sqcap Adjacent \\ SameColourGroup \leftarrow SameColourGroup.SameColourGroup \end{array}\right\}$$

$$\left\{\begin{array}{l} \forall x,y\ SameColourGroup(x,y) \leftarrow SameColour(x,y) \land Adjacent(x,y) \\ \forall x,y\ SameColourGroup(x,y) \leftarrow \exists z\ SameColourGroup(x,z) \\ \qquad\qquad\qquad\qquad\qquad\qquad \land\ SameColourGroup(z,y) \end{array}\right\}$$

$\rightarrow$ Inductive definition: not possible in FO

Defining Up'

Above is the transitive closure of *Up*

$$Above \doteq Up \sqcup Up.Above$$

Apart from the balls that disappear, *Above* remains the same

$$Above' \doteq Above \sqcap (Reamins \times Remains)$$

Up' is the intransitive relation of which the $Above'$ is the transitive closure

$$Up' \doteq Above' \sqcap \neg(Above'.Above')$$

Defining Up'

Above is the transitive closure of *Up*

$$Above \doteq Up \sqcup Up.Above$$

→Inductive, so cannot be translated to FO

Apart from the balls that disappear, *Above* remains the same

$$Above' \doteq Above \sqcap (Reamins \times Remains)$$

→Not inductive, so could be translated to FO

Up' is the intransitive relation of which the $Above'$ is the transitive closure

$$Up' \doteq Above' \sqcap \neg(Above'.Above')$$

→Not inductive, so could be translated to FO

Defining *Left′*

A ball is in the column to the left of the column of another ball if

- it is either directly to the left of it, or
- it is below or above a ball directly to the left of it
- it is to left of a ball that is below or above it

$$\left\{\begin{array}{l} \mathit{InLeftColumn} \leftarrow \mathit{Left} \\ \mathit{InLeftColumn} \leftarrow \mathit{Left}.(\mathit{Above} \sqcup \mathit{Above}^{-}) \\ \mathit{InLeftColumn} \leftarrow (\mathit{Above} \sqcup \mathit{Above}^{-}).\mathit{Left} \end{array}\right\}$$

In the new situation, the bottom layer is formed by those remaining balls that do not have a remaning ball below them

$$\mathit{OnGround}' \doteq \mathit{Remains} \sqcap \neg\exists \mathit{Above}^{-}.\mathit{Remains}$$

Defining *Left'*

A ball is in the column to the left of the column of another ball if

- it is either directly to the left of it, or
- it is below or above a ball directly to the left of it
- it is to left of a ball that is below or above it

$$\left\{ \begin{array}{l} InLeftColumn \leftarrow Left \\ InLeftColumn \leftarrow Left.(Above \sqcup Above^{-}) \\ InLeftColumn \leftarrow (Above \sqcup Above^{-}).Left \end{array} \right\}$$

→ Not inductive, but natural fit with case based structure

In the new situation, the bottom layer is formed by those remaining balls that do not have a remaning ball below them

$$OnGround' \doteq Remains \sqcap \neg \exists Above^{-}.Remains$$

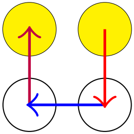
Defining Left' (2)

In the new situation, a ball is to the left of another ball if
it was in the column to its left and they are now on the same level

$$\mathit{Left}' \doteq \overbrace{\mathit{InLeftColumn}} \sqcap \overbrace{((\mathit{OnGround}' \times \mathit{OnGround}') \sqcup \mathit{Up}'^{-}.\mathit{Left}'.\mathit{Up}')}$$

Is an inductive definition over the rows of the grid:

- Base case: both balls on the ground
- Inductive step:



Computing state transitions

- Simulate game
- Or plan a winning strategy using backtracking

Given interpretation S

- Domain $D = \{b_1, \ldots, b_n\}$
- *Colour*, *Selected*
- *Left*, *Up*

Find interpretation S'

- With same D
- *Remains*
- $Left'$, Up'

Computing state transitions (2)

Finite model expansion

Extend interpretation S (with finite domain) for Σ with interpretation S' for Σ' such that $S \cup S' \models T$

Complexity:

- For FO and FO(ID): captures NP
- In FO(ID), if all predicates have definition: in P

We don't need decidability

Implementation: IDP (competitive with best ASP solvers)

- `http://www.cs.kuleuven.be/~dtai/krr/software.html`

Conclusions

$$\begin{array}{ccc} \text{DL} & + & \text{Rules} \\ \cap & & \cap \\ \text{FO} & (& \text{ID} \quad) \end{array}$$

Investigated extension of DL with rules induced by FO(ID)

- Fragment of FO(ID) and syntactic sugar that allows similar modeling style to DL
- But extends $\equiv$ of DL by allowing definitions that
  - Can be inductive
  - Can consist of multiple rules
- Has decidable guarded fragment (which is also useful?)

Questions or comments

$\left.\begin{array}{r}\texttt{joost.vennekens}\\ \texttt{marc.denecker}\end{array}\right\}$ @cs.kuleuven.be