KU LEUVEN

A Compositional Typed Higher-Order Logic with Definitions

Ingmar Dasseville - Matthias Van Der Hallen - Bart Bogaerts - Gerda Janssens - Marc Denecker

International Conference on Logic Programming, 2016

Goal in KR:

- build expressive logics
- by integrating useful and expressive language constructs
- in a meaning preserving way

To add aggregate expressions to logic programming and ASP: many effort years, several PhD's and many papers.

To add a nested cardinality aggregate $Card$ to classical logic:

- New syntactical rule in definition of term:
  - $Card(\{x : \varphi\})$ is a term if φ is a formula
- New semantical rule in definition of term evaluation:
  - $(Card(\{x : \varphi\}))^{\mathscr{I}} = \#(\{d \mid \mathscr{I}[x : d] \models \varphi\})$

To add a nested cardinality aggregate $Card$ to classical logic:

- New syntactical rule in definition of term:
  - $Card(\{x:\varphi\})$ is a term if φ is a formula
- New semantical rule in definition of term evaluation:
  - $(Card(\{x:\varphi\}))^{\mathscr{I}} = \#(\{d \mid \mathscr{I}[x:d] \models \varphi\})$

We are ready.

Developing a compositional method to extend rule sets under well-founded and stable semantics with new language constructs.

Last Year: Adding templates to KR languages

Result: Framework for adding language constructs and building logics

This Year: Building a general logic including compositionality principles

Definition (Compositionality according to Frege)

The meaning of a complex expression is determined by the meanings of its constituent expressions and the rules used to combine them

The semantics for a logic L and a language constructs C must satisfy:

$$Sem_L(C(e_1,...,e_n)) = Sem_C(Sem_L(e_1),...,Sem_L(e_n))$$

What is $Sem_L(C(e_1, ..., e_n))$ mathematically?

- Logic expressions express "information"
- Infon : mathematical semantical object to express information
  - Function from structures to values
  - $=$ A quantum of information
  - Confer intensional objects (e.g., Montague)

KU LEUVEN

What is $Sem_L(C(e_1, ..., e_n))$ mathematically?

- Logic expressions express "information"
- Infon : mathematical semantical object to express information
  - Function from structures to values
  - = A quantum of information
  - Confer intensional objects (e.g., Montague)

Infon of $p \vee q$
Maps $\{p\}$ to *True*
Maps $\{\}$ to *False*

KU LEUVEN

What is $Sem_L(C(e_1, ..., e_n))$ mathematically?

- Logic expressions express “information”
- Infon : mathematical semantical object to express information
  - Function from structures to values
  - $=$ A quantum of information
  - Confer intensional objects (e.g., Montague)

Infon of $p \vee q$
Maps $\{p\}$ to *True*
Maps $\{\}$ to *False*

Infon of $c + 3$
Maps $\{c = 5\}$ to 8

KU LEUVEN

Syntax: Extend the set of valid expressions

Syntax: Extend the set of valid expressions

Typing: Not all expressions within the grammar are sensible (e.g. 1+“hello”)

Syntax: Extend the set of valid expressions

Typing: Not all expressions within the grammar are sensible (e.g. 1+“hello”)

Semantics: What infon corresponds to the expression?

A language expression C is:

- Abstract syntax $C(e_1, ..., e_n)$

A language expression C is:

- Abstract syntax $C(e_1, ..., e_n)$
- Typing function on types: $Typ_C(type_1, \ldots, type_n)$ such that

$$Typ_L(C(e_1, ..., e_n)) = Typ_C(Typ_L(e_1), ..., Typ_L(e_n))$$

A language expression C is:

- Abstract syntax $C(e_1,...,e_n)$
- Typing function on types: $Typ_C(type_1,\dots,type_n)$ such that

$$Typ_L(C(e_1,...,e_n)) = Typ_C(Typ_L(e_1),...,Typ_L(e_n))$$

- Semantic function on infons: $Sem_C(Infon_1,\dots,Infon_n)$ such that

$$Sem_L(C(e_1,...,e_n)) = Sem_C(Sem_L(e_1),...,Sem_L(e_n))$$

Simply typed lambda calculus

- Higher order types
- Lambda Abstractions

Definitions

- Higher order Rules
- Well-founded/stable semantics, lifted

Higher Order Definitions

```
{
∀cur ∀Move ∀ IsWon:
win(cur, Move, IsWon) ← IsWon(cur) ∨
    ∃ nxt : Move(cur,nxt) ∧ lose(nxt,Move,IsWon).

∀cur ∀Move ∀IsWon:
lose(cur,Move, IsWon) ← ¬IsWon(cur) ∧
     ∀ nxt : Move(cur,nxt) ⇒ win(nxt,Move,IsWon).
}
```

- Meaning of a logical expression is an infon.
- Composionality obtained using Frege's principle.
- Integration of common logical and functional language constructs.
- Simplifying current and enabling new applications.
- But we need solvers!