

# Semantic Web Technologies

## F-Logic Semantics

Jos de Bruijn

`jos.debruijn@deri.org`

Digital Enterprise Research Institute (DERI)  
University of Innsbruck, Austria

May 25, 2006

# Outline

F-Logic Semantics

Reducing F-Logic Programming to standard LP

## F-Structures (I)

- ▶ Interpretations in F-Logic are called **F-structures**.

### F-Structure

An **F-structure** is a tuple  $\mathbf{I} = \langle U, \prec_U, \in_U, \mathbf{I}_F, \mathbf{I}_{\rightarrow}, \mathbf{I}_{\leftrightarrow} \rangle$ .

- ▶  $U$  is the domain,
- ▶  $\prec_U$  is an irreflexive partial order on  $U$ ,
  - ▶  $a \preceq_U b$  when  $a \prec_U b$  or  $a = b$
- ▶  $\in_U$  is a binary relation over  $U \times U$ , and
- ▶  $a \in_U b$  and  $b \preceq_U c$  then  $a \in_U c$ .

## Interpreting Symbols with $\mathbf{I}_F$

- ▶  $\mathcal{F}$  is the set of function symbols of some F-Logic language

### Interpretation of function symbols

$\mathbf{I}_F$  interprets every  $n$ -ary function symbol  $f \in F$  as an  $n$ -ary function over the domain  $U$ :  $\mathbf{I}_F(f) : U^n \rightarrow U$

- ▶ Thus, constants are interpreted as elements of the domain

## Interpreting Attributes with $\mathbf{I}_{\rightarrow}$

### Interpretation of single-valued attributes

$\mathbf{I}_{\rightarrow}$  interprets every **element of the domain**  $u \in U$  as a partial function from  $U$  to  $U$ :  $\mathbf{I}_{\rightarrow}(u) : U \rightarrow U$ .

- ▶ Thus,  $\mathbf{I}_{\rightarrow}$  associates with **elements of the domain**,
- ▶ pairs  $\langle u_0, u_1 \rangle$ .

## Interpreting Attributes with $\mathbf{I}_{\rightarrow}$

### Interpretation of set-valued attributes

$\mathbf{I}_{\rightarrow}$  interprets every **element of the domain**  $u \in U$  as a partial function from  $U$  to  $\mathcal{P}(U)$ :  $\mathbf{I}_{\rightarrow}(u) : U \rightarrow \mathcal{P}(U)$ .

- ▶ Thus,  $\mathbf{I}_{\rightarrow}$  associates with **elements of the domain**,
- ▶ pairs  $\langle u_0, \{u_1, \dots, u_n\} \rangle$ .

## Interpreting Variables

- ▶ Same as in FOL

### Interpreting Variables

Given interpretation  $\mathbf{I}$ , a variable assignment  $B$ , and a term  $t$ ,  $t^{\mathbf{I},B}$  is defined as:

- ▶  $x^{\mathbf{I},B} = x^B$  for variable symbol  $x$  and
- ▶  $t^{\mathbf{I},B} = \mathbf{I}_F(f)(t_1^{\mathbf{I},B}, \dots, t_n^{\mathbf{I},B})$  for  $t$  is  $f(t_1, \dots, t_n)$ .

## Satisfaction in F-Structures

### Satisfaction of atomic formulas

Given  $\mathbf{I} = \langle U, \prec_U, \in_U, \mathbf{I}_F, \mathbf{I}_P, \mathbf{I}_{\rightarrow} \rangle$ ,  $B$  a variable assignment, and  $\phi$  a formula.  $\mathbf{I}, B \models \phi$  if:

- ▶  $\mathbf{I}, B \models t_1 : t_2$  iff  $t_1^{\mathbf{I}, B} \in_U t_2^{\mathbf{I}, B}$
- ▶  $\mathbf{I}, B \models t_1 :: t_2$  iff  $t_1^{\mathbf{I}, B} \preceq_U t_2^{\mathbf{I}, B}$
- ▶  $\mathbf{I}, B \models t_1[t_2 \rightarrow t_3]$  iff  $t_3^{\mathbf{I}, B} = \mathbf{I}_{\rightarrow}(t_2^{\mathbf{I}, B})(t_1^{\mathbf{I}, B})$ .
- ▶  $\mathbf{I}, B \models t_1[t_2 \twoheadrightarrow t_3]$  iff  $\mathbf{I}_{\rightarrow}(t_2^{\mathbf{I}, B})(t_1^{\mathbf{I}, B})$  is defined and  $t_3^{\mathbf{I}, B} \in \mathbf{I}_{\rightarrow}(t_2^{\mathbf{I}, B})(t_1^{\mathbf{I}, B})$ .

## Extension to complex formulas

### Satisfaction of Complex formulas

- ▶  $\mathbf{I}, B \models t_1 = t_2$  iff  $t_1^{\mathbf{I}, B} = t_2^{\mathbf{I}, B}$ ,
- ▶  $\mathbf{I}, B \models \phi_1 \wedge \phi_2$  iff  $\mathbf{I}, B \models \phi_1$  and  $\mathbf{I}, B \models \phi_2$ ,
- ▶  $\mathbf{I}, B \models \phi_1 \vee \phi_2$  iff  $\mathbf{I}, B \models \phi_1$  or  $\mathbf{I}, B \models \phi_2$ ,
- ▶  $\mathbf{I}, B \models \neg\phi_1$  iff  $\mathbf{I}, B \not\models \phi_1$ ,
- ▶  $\mathbf{I}, B \models \forall x(\phi_1)$  iff for every  $B'$  which is an  $x$ -variant of  $B$ ,  $\mathbf{I}, B' \models \phi_1$ , and
- ▶  $\mathbf{I}, B \models \exists x(\phi_1)$  iff for some  $B'$  which is an  $x$ -variant of  $B$ ,  $\mathbf{I}, B' \models \phi_1$ .

## Reducing F-Logic to LP

- ▶ F-Logic Programming can be reduced to Logic Programming
- ▶ Thus, F-Logic is syntactic sugar; it does not add anything in expressiveness
- ▶ Reducing terms:
  - ▶ Map object IDs to constants
  - ▶ Map constructed terms to functions
  - ▶ Map variables to variables

Thus, terms look exactly the same!

- ▶ Map atoms to atoms

## Reducing F-Logic to LP (2)

- ▶ Each type of molecule is mapped to an atom:
  - ▶  $A:B$  maps to `_member(A,B)`
  - ▶  $A::B$  maps to `_subclass(A,B)`
  - ▶  $A[B \rightarrow C]$  maps to `_svatt(A,B,C)`
  - ▶  $A[B \rightarrow \rightarrow C]$  maps to `_mvatt(A,B,C)`
  - ▶  $A[B = \rightarrow C]$  maps to `_svattsig(A,B,C)`
  - ▶  $A[B = \rightarrow \rightarrow C]$  maps to `_mvattsig(A,B,C)`

Note that the name of the predicates does not really matter.

- ▶ You obtain LP rules by simply replacing terms, atoms and molecules as specified.
- ▶ Axiomatize the semantics of the F-Logic is-a molecules:
 

```

_subclass(X,Z) :- _subclass(X,Y),_subclass(Y,Z).
_member(X,Z) :- _member(X,Y),_subclass(Y,Z).
      
```

## Reducing F-Logic to LP (2)

- ▶ Axiomatize the semantics of the F-Logic is-a molecules:  
 $\_subclass(X,Z) :- \_subclass(X,Y), \_subclass(Y,Z).$   
 $\_member(X,Z) :- \_member(X,Y), \_subclass(Y,Z).$
- ▶ Axiomatize the semantics of the F-Logic functions:
  - ▶  $X \doteq Y :- \_svatt(V,W,X), \_svatt(V,W,Y).$
  - ▶ **or:**  $!- \_svatt(V,W,X), \_svatt(V,W,Y), X \neq Y.$

## Reducing F-Logic to LP (4)

```
_member(bob, empl).
_svatt(bob,name,"Bob").
_svatt(bob,age,40).
_svatt(bob,affiliation,cs1).
_member(mary,faculty).
_svatt(mary,affiliation,cs1).
_member(cs1,dept).
_svatt(cs1,dname,"CS").
_svatt(cs1,mngr,bob).

bob:empl[name->"Bob";
          age->40;
          affiliation->cs1].
mary:faculty[affiliation->cs1].
cs1:dept[dname->"CS";
          mngr->bob]].
```

## Class information:

```
_svattsig(person,name,string).  
_mvattsig(person,friends,person).  
_mvattsig(person,children,child(person)).  
_subclass(empl,person).  
_svattsig(empl,affiliation,department).  
_svattsig(empl,boss,empl).  
_subclass(faculty,empl).  
_svattsig(faculty,boss,faculty).  
_svattsig(faculty,boss,manager).  
_svatt(faculty,avgSalary,50000).  
_svattsig(dept,mngr,empl).
```

```
person[name=>string;  
    friends=>>person;  
    children=>>child(person)].  
empl::person[affiliation=>department;  
    boss=>empl].  
faculty::empl[boss=>(faculty,manager);  
    avgSalary->50000].  
dept[mngr=>empl].
```

## Rule

```
_svatt(E,boss,M) :- _member(E,empl),_member(D,dept),  
_svatt(E,affiliation,D), _svatt(D,mngr,M),  
_member(M,empl).
```

```
E[boss->M] :- E:empl, D:dept,  
E[affiliation->D[mngr->M:empl]].
```

# Summary

F-Logic Semantics

Reducing F-Logic Programming to standard LP

## Required reading

- ▶ Michael Kifer: Rules and Ontologies in F-Logic. Reasoning Web 2005: 22-34

## Further reading

- ▶ Michael Kifer, Georg Lausen, James Wu: Logical Foundations of Object-Oriented and Frame-Based Languages. J. ACM 42(4): 741-843 (1995)
- ▶ Guizhen Yang, Michael Kifer: Reasoning about Anonymous Resources and Meta Statements on the Semantic Web. J. Data Semantics 1: 69-97 (2003)
- ▶ Guizhen Yang, Michael Kifer: Well-Founded Optimism: Inheritance in Frame-Based Knowledge Bases. CoopIS/DOA/ODBASE 2002: 1013-1032