Advanced Data Modeling

5: Semantics

Steffen Staab
Overview

- Logic as query language.
- Grounding.
- Minimal Herbrand models.
- Completion.
Given:
- first-order formula $A[x_1, \ldots, x_n]$
- Herbrand interpretation $I$

This first-order formula can be considered as a definition of a relation $R_A$ on $T^\Sigma$ as follows:

$$(t_1, \ldots, t_n) \in R_A := I \models A[t_1, \ldots, t_n]$$
**Clauses as definitions**

We say that a clause

\[ p(t_1, \ldots, t_m) :- L_1, \ldots, L_n \]

defines the relation symbol \( p \).

Let \( C \) be a set of clauses and \( p \) be a relation symbol. We call the definition of \( p \) in \( C \) the set of all clauses in \( C \) that define \( p \).
A deductive database is a set of clauses.

This set of clauses is regarded as a collection of definitions of relations.

The semantics defines the meaning of this definitions by associating with them an interpretation, or a class of interpretations.

Query answering is based on the semantics.
Two key assumptions

- the **unique name assumption**: each name denotes a unique object.

- the **closed world assumption**: a negative statement \( \neg A \) holds if the corresponding positive one \( A \) does not hold.
Minimal Herbrand Models

Let $I$ be a Herbrand model of a set of formulas $S$.

We call $I$ a minimal Herbrand model of $S$ if it is minimal w.r.t. the subset relation, i.e. for every Herbrand model $I´$ of $S$ of the same signature we have $I´ \supseteq I$.

$I$ is called the least Herbrand model of $S$ if for every Herbrand model $I´$ of $S$ of the same signature we have $I \subseteq I´$. 
Does every set of formulas $S$ have a least Herbrand model?
Lecture 5 until here

Lecture 6 from here
**Instance and variant**

Let $E$, $E'$ be a pair of terms or formulas.

- $E'$ is an **instance** of $E$, denoted $E < E'$, if there exists a substitution $\theta$ such that $E \theta = E'$.

- **ground instance**: instance that is ground,

- $E'$ is a **variant** of $E$ if $E'$ is an instance of $E$ and $E$ is an instance of $E'$. 
Examples

- P(x,a) is instance of P(x,y) because of P(x,y)[y|a]

- P(b,a) is a ground instance

- P(x,y) and P(u,v) are variants of each other, because of
  - [x|u, y|v] and
  - [u|x, v|y]
Grounding

Let $C$ be a set of clauses and $\Sigma$ be any signature containing all symbols used in $C$. The **grounding of $C$ w.r.t. $\Sigma$**, denoted $C^*$ is the set of all ground instances of the signature $\Sigma$ of clauses in $C$.

Lemma. Let $I$ be a Herbrand interpretation and $C$ be a set of clauses. Then $I \models C$ if and only if $I \models C^*$. 
Proof
Logic with equality

- Additional atomic formulas \( s = t \), where \( s, t \) are terms.

- Abbreviation: \( x \neq y \) := \( \neg (x = y) \).

- Unlike other relations, the semantics of \( s = t \) is **predefined** in all Herbrand interpretations:
  \( I \models s = t \) if \( s \) coincides with \( t \).
Example valid formulas

\[ f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n) \rightarrow x_1 = y_1 \wedge \ldots \wedge x_n = y_n \]

\[ f(x_1, \ldots, x_n) \neq g(y_1, \ldots, y_n) \]

\[ f(x_1, \ldots, x_n) \neq c \]

\[ d \neq c \]

\[ A[t] \iff \forall x (x = t \rightarrow A[x]) \]
Consider a definition of a relation $r$

$$r(t_1) : \neg G_1$$

$$\ldots$$

$$r(t_m) : \neg G_m$$

What is the meaning of this definition?
Especially in the light of negation as failure?
Idea of completion

man(hans).
man(adam).
person(eva).
woman(X) :- person(X), not man(X).

Is eva a woman?
  - She might be a man and we just don’t know!
  - Minimal model says that she is not a woman!

Trick: Completion:
man(X) ↔ (X=hans ∨ X=adam).
woman(X) ↔ X=eva.

I.e. hans and adam are the only men.
Idea of Completion

\[
\text{man} (\text{hans}). \\
\text{man} (X) :- \text{lovesBeer}(X, Y).
\]

Completion:
\[
\text{man} (X) \iff X = \text{hans} \lor (\exists Y : X = U \land \text{lovesBeer}(U, Y))
\]

A man is a man only if he is hans or if he loves some brand of beer.
Completion. Step 1.

Replace every clause by an equivalent one such that the arguments of \( r \) are \( x_1, \ldots, x_n \):

Given:
\[
r(t_1, \ldots, t_n) \ :- \ G
\]

Replace by:
\[
r(x_1, \ldots, x_n) \ :- \ x_1 = t_1 \land \ldots \land x_n = t_n \land G
\]
Completion. Step 2.

If there are variables $y_1, \ldots, y_k$ occurring in a body but not in the head, apply $\exists$ to these variables, i.e.,

Given
\[ r(x_1, \ldots, x_n) : \neg G \]

Modify to
\[ r(x_1, \ldots, x_n) : \exists y_1 \ldots \exists y_k G \]
If there are several definitions, replace them by one

*Given*
\[ r(x_1, \ldots, x_n) :\!\!\!\!\!:: G_1 \]
\[
\ldots
\]
\[ r(x_1, \ldots, x_n) :\!\!\!\!\!:: G_m \]

*Replace by*
\[ r(x_1, \ldots, x_n) :\!\!\!\!\!:: G_1 \lor \ldots \lor G_m \]
Completion. Step 4.

Replace :- by $\leftrightarrow$:

Given

$$r(x_1, \ldots, x_n) :- G_1 \lor \ldots \lor G_m$$

Replace by

$$r(x_1, \ldots, x_n) \leftrightarrow G_1 \lor \ldots \lor G_m$$

The formula

$$r(x_1, \ldots, x_n) \leftrightarrow G_1 \lor \ldots \lor G_m$$

is called the **completed definition** of the original set of clauses.
Example

\[ r(u,v):-p(u,1,z). \]
\[ r(u,v):-p(2,v,z). \]

\[ r(x,y) :- x=u \land y=v \land p(x,1,z). \]
\[ r(x,y) :- x=u \land y=v \land p(2,y,z). \]

\[ r(x,y) :- \exists z \ x=u \land y=v \land p(x,1,z). \]
\[ r(x,y) :- \exists z \ x=u \land y=v \land p(2,y,z). \]

\[ r(x,y) \leftrightarrow (\exists z \ x=u \land y=v \land p(x,1,z)) \lor (\exists z \ x=u \land y=v \land p(2,y,z)) \]
Properties

- All steps preserve Herbrand models, except for the last one.
  - Why?

- Gives a unique semantics to non-recursive definitions
  - What about recursive definitions?

- Logic programming semantics and first-order semantics coincide for non-recursive definitions
Recursive definitions

odd(1).
even(f(X)) :- odd(X).
odd(f(X)) :- even(X).

Completion:

odd(X) ↔ X=1 ∨ (X=f(Y) ∧ even(Y)).
even(X) ↔ X=f(Y) ∧ odd(Y).
Recursive definitions

person(adam).
person(eva).
woman(X) :- person(X), not man(X).
man(X) :- person(X), not woman(X).

Completion:
woman(X) ↔ (Y=X ∧ person(Y) ∧ not man(Y)).
man(X) ↔ (Y=X ∧ person(Y) ∧ not woman(Y)).

Semantics not unique in logic programming:
Models are
1. I={woman(adam),woman(eva)}
2. I={man(adam),man(eva)}
3. I={woman(adam),man(eva)}
4. I={man(adam),woman(eva)}

What is the semantics in first order logics?
Recursive definitions

person(adam).
person(eva).
woman(X) :- person(X), not man(X).
man(X) :- person(X), not woman(X).

Completion:
woman(X) ↔ (Y=X ∧ person(Y) ∧ not man(Y)).
man(X) ↔ (Y=X ∧ person(Y) ∧ not woman(Y)).

Semantics not unique in logic programming:
Models are
1. I={woman(adam),woman(eva)}
2. I={man(adam),man(eva)}
3. I={woman(adam),man(eva)}
4. I={man(adam),woman(eva)}

What is the semantics in first order logics?
Models are: woman I= {} = man I
Simple characterization of completion

Let $C$ be a definition of $r$, $I$ be a Herbrand model of the corresponding completed definition, and $r(t_1, \ldots, t_n)$ be a ground atom.

Then

$I \models r(t_1, \ldots, t_n) \iff \exists (r(t_1, \ldots, t_n) :- L_1, \ldots, L_m) \in C^* (I \models L_1 \land \ldots \land L_n)$:
Immediate consequence operator

\[
T_C(I) := \{ A \mid \text{there exists } (A :- G) \in C^* \text{ such that } I \models G \}
\]

Fixpoint: an interpretation such that \( T_C(I) = I \).
Definite clauses have the least model

Let $C$ be a set of definite clauses.

Define

$$I_0 := \{\}$$
$$I_{n+1} := T_C(I_n), \text{ for all } n \geq 0,$$
$$I_\omega := \bigcup_{i=0}^\omega I_i$$

Then $I_\omega$ is the least fixpoint of $T_C$ and also the least Herbrand model of $C$. 
Let C be a non-recursive database and K be an arbitrary interpretation.

Define

\[ I_0 := K \]
\[ I_{n+1} := T_C(I_n), \text{ for all } n \geq 0, \]
\[ I_\omega := \bigcup_{i=0}^\omega I_i \]

Then \( I_\omega \) is the only fixpoint of \( T_C \). Moreover, for some \( n \) we have \( I_\omega = I_n \).
Non-recursive sets of clauses

Let \( C \) be a set of clauses.

Its **dependency graph** consists of pairs \( p \rightarrow r \) such that \( p \) occurs in the body of a clause which defines \( r \) in \( C \).

A set of clauses is **non-recursive** if the dependency graph contains no cycles.
 Dependency graph of $C$ consists of pairs $p \rightarrow r$ such that $p$ occurs in the body of a clause which defines $r$ in $C$.

$C$ is non-recursive if the dependency graph contains no cycles.

A relation $p$ depends on a relation $q$ in $C$ if there exists a path of length $\geq 1$ from $q$ to $p$ in the dependency graph of $C$.

A set of clauses is non-recursive if and only if no relation depends on itself.
**Level mapping**

- Level mapping: mapping ` from a set of relation symbols to N.
- \( l(r) \) is called the level of \( r \).

**Theorem.** Let \( C \) be a finite non-recursive set of clauses. Then there exists a level mapping \( l \) such that for every clause \( c \in C \),
  
  if \( q \) occurs in the body of \( c \) and \( c \) defines \( r \),
  
  then \( l(r) > l(q) \).