First-Order Logic

Blackburn & Bos, pp. 1-29

Ling335: Computational Semantics

Miriam Butt and Maribel Romero

WiSe2014-15

First-Order Logic

- First-order logic is a formalism used…
  - to represent meaning in natural language, and
  - to carry out various inference tasks:
    - Querying task
    - Consistency checking task
    - Informativity checking task
- Today (first half of chapter 1), we will present first-order logic and describe the three tasks.
- In second half of chapter 1, we will write a first-order model checker performing the querying task.

Roadmap

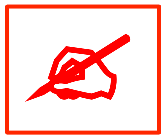
- First-Order Logic
  - Vocabulary
  - First-order models (semantics)
  - First-order languages (syntax)
  - Truth and Satisfaction
  - Adding functions symbols, equality and sorted variables
- Three inference tasks
  - Querying
  - Consistency checking
  - Informativity checking

Vocabulary

- A vocabulary is a set of predicates and individual constants, e.g.

```
{  (LOVE,2)
   (CUSTOMER,1)
   (ROBBER,1)
   (MIA,0)
   (VINCENT,0)
   (HONEY-BUNNY,0)
   (YOLANDA,0)      }
```

- Vocabularies tell us which first-order lgs and which first-order models belong together. E.g. a lg with the vocabulary above cannot be evaluated in a model that is just about cleaning products.
- **Note**: Unlike in Prolog, a given predicate can only be used with a fixed arity.



First-Order Models

- A model is a semantic object: roughly, a situation
- A model for a given vocabulary provides:
  - a non-empty collection of entities (domain D) to be talked about
  - the mapping (interpretation function F) from each symbol in the vocabulary to the appropriate semantic value
- In set-theoretic terms, a model is an ordered pair (D,F).

First-Order Models

Model M_1

$F(MIA) = d_1$

$F(HONEY\text{-}BUNNY) = d_2$

$F(VINCENT) = d_3$

$F(YOLANDA) = d_4$

$F(COSTUMER) = \{d_1, d_3\}$

$F(ROBBER) = \{d_2, d_4\}$

$F(LOVE) = \{(d_4, d_2), (d_3, d_1)\}$

Model M_2

$F(MIA) = d_2$

$F(HONEY\text{-}BUNNY) = d_1$

$F(VINCENT) = d_4$

$F(YOLANDA) = d_1$

$F(COSTUMER) = \{d_1, d_2, d_4\}$

$F(ROBBER) = \{d_3, d_5\}$

$F(LOVE) = \{\ \} = \emptyset$

First-Order Languages: Symbols

- Symbols of a first-order language:
  - Vocabulary symbols (=non-logical symbols)
  - Countably infinite collection of variables: $x, y, z\ldots, x_1, x_2,\ldots$
  - Boolean connectives: $\neg \quad \wedge \quad \vee \quad \rightarrow$
  - Universal quantifier $\forall$ and existential quantifier $\exists$
  - Round brackets and comma

- Among these symbols, we distinguish terms…
  - individual constants (≈ proper names), e.g. MIA
  - Individual variables (≈ pronouns), e.g. x

- … and predicates, e.g. ROBBER.

First-Order Languages: Syntax

- Atomic formulas
  0. If R is a predicate of arity n and τ_1, … ,τ_n are terms, then $R(\tau_1, \ldots, \tau_n)$ is an atomic formula.

- Well-formed formulas (wffs)
  1. All atomic formulas are wffs.
  2. If ϕ and ψ are wffs, then so are $\neg\phi$, $(\phi\wedge\psi)$, $(\phi\vee\psi)$ and $(\phi\rightarrow\psi)$.
  3. If ϕ is a wff and x is a variable, then both $\forall x\phi$ and $\exists x\phi$ are wffs. We call the matrix or scope of such wffs.
  4. Nothing else is a wff.

- Examples

  ¬LOVE(YOLANDA,VINCENT)

  (ROBBER(MIA) → LOVE(MIA,HONEY-BUNNY))

  ∀x (CUSTOMER(x) → ∃yLOVE(y,x))

First-Order Languages: some syntactic conventions

- We often drop outer brackets:
  E.g. instead of $(\phi \wedge \psi)$, we write $\phi \wedge \psi$.

- Negation $\neg$ has more "glue" than $\wedge$ and $\vee$, which in turn have more glue than $\rightarrow$.

First-Order Languages: free vs. bound variables

- An occurrence of a variable x is *bound* if it occurs in the scope of ∀x or ∃x. A variable is *free* if it is not bound.
- A formula with no free variables is a special kind of formula called *sentence*.

∀xROBBER(x) = ∀yROBBER(y)	'Everybody is a robber'

ROBBER(x) ≠ ROBBER(y)	'He is a robber' ≠ 'She is a robber'

Place holders

Need more information!

Truth and Satisfaction

- 2-place relation *truth* that holds –or doesn't– between a sentence and a model of the same vocabulary
- 3-place relation *satisfaction* that holds –or doesn't– between a formula, a model M of the same vocabulary and an assignment function g from variables to values

Formula (description)	**M = (D,F)** (situation)	**g: variables → D** (context)
∀xROBBER(x) ROBBER(x)	M_1	g = [x → YOLANDA y → MIA z → HONEY-BUNNY]

Satisfaction

- Interpretation function for vocabulary and variables: $I_F^g(.)$

  i. If τ is a constant term, then $I_F^g(\tau) = F(\tau)$

  ii. If τ is a variable term, then $I_F^g(\tau) = g(\tau)$

  iii. If P is a predicate, then $I_F^g(P) = F(P)$

- x-variant of an assignment

  If g and g’ are assignments in M and, for all variables y other that x, g(y)= g’(y), then g’ is an x-variant of g

- $M,g \models \phi$ is read as "ϕ is satisfied in M wrt assignment g"

Satisfaction (cont'd)

- Definition of satisfaction:

0. $M,g \models R(\tau_1,\ldots,\tau_n)$	iff	$(I_F^g(\tau_1),\ldots,I_F^g(\tau_n)) \in F(R)$
2.1 $M,g \models \neg\phi$	iff	not $M,g \models \phi$
2.2 $M,g \models (\phi \wedge \psi)$	iff	$M,g \models \phi$ and $M,g \models \psi$
2.3 $M,g \models (\phi \vee \psi)$	iff	$M,g \models \phi$ or $M,g \models \psi$
2.4 $M,g \models (\phi \rightarrow \psi)$	iff	not $M,g \models \phi$, or $M,g \models \psi$
3.1 $M,g \models \forall x\phi$	iff	$M,g' \models \phi$ for all x-variants g' of g
3.2 $M,g \models \exists x\phi$	iff	$M,g' \models \phi$ for some x-variant g' of g

Truth

- Definition of truth

  `A sentence` ϕ `is true in a Model M iff, for any assignment g from variables to values in M, we have that M,g` $\models$ ϕ.

- $M \models \phi$ is read as "ϕ is true in M",

Some additions

- Function symbols
- Equality predicate
- Sorted variables

Adding function symbols

- FATHER(BUTCH) not as "Butch is a father"
  but as "the father of Butch"
- An n-place function symbol f is interpreted as a function that takes an n-tuple of elements of D as input and yields an element of D as output.
- Additional syntactic rule:
  - 1. If f is a function symbol of arity n and τ_1, … ,τ_n are terms, then $f(\tau_1, \ldots ,\tau_n)$ is a term.
- Additional semantic rule:
  - 1. If τ is a term of the form $f(\tau_1, \ldots ,\tau_n)$, then $I_F^g(\tau) = F(f)(I_F^g(\tau_1),\ldots,I_F^g(\tau_n))$

Adding equality

- Two-place relation symbol **=**, with infix notation, e.g. $\tau_1 = \tau_2$.

- Additional syntactic rule:

  00. If τ_1 and τ_n are terms, then $\tau_1 = \tau_n$ is an atomic formula.

- Additional semantic rule:

  00. $M, g \models \tau_1 = \tau_2$ iff $I^g_F(\tau_1)$ equals $I^g_F(\tau_2)$

Adding sorted variables

- $\forall x(\text{ANIMATE}(x) \rightarrow \text{BREATH}(x))$ abbreviated as $\forall a\ \text{BREATH}(a)$
- $\neg\exists x(\text{INANIMATE}(x) \wedge \text{TALK}(x))$ abbreviated as $\neg\exists i\ \text{TALK}(i)$
- Not incorporated into the current fragment. Some use for this is chapter 3.

Roadmap

- First-Order Logic
  - Vocabulary
  - First-order models (semantics)
  - First-order languages (syntax)
  - Truth and Satisfaction
  - Adding functions symbols, equality and sorted variables
- Three inference tasks
  - Querying
  - Consistency checking
  - Informativity checking

Querying Task

Given a model M (, and assignment g) and a first-order formula ϕ , is ϕ satisfied in M (with respect to g) or not?

- Is querying a task we can compute? Yes, if we fix what the free variables stand for (i.e., if we spell out g at least for the variables used) and if we confine ourselves to finite models.
- Model checker: program that performs this task

Consistency Checking Task

- A formula ϕ is consistent/satisfiable if it is satisfied in at least one model.
- A finite set of formulas $\{\phi_1,\ldots, \phi_n\}$ is consistent/satisfiable if the formula $(\phi_1 \wedge \ldots \wedge \phi_n)$ is consistent/satisfiable.

Given a first-order formula ϕ , is ϕ consistent/ satisfied or inconsistent/unsatisfiable?

Consistency Checking Task

- Is this task computationally decidable? No.
  - vast mathematical universe of models
  - some satisfiable formulas only have infinite satisfying models
- But a partial solution can be reached by moving from a model-theoretic (semantic) perspective to a proof-theoretic (syntactic) perspective (Chapters 4-5)

Informativity Checking Task

- A formula ϕ is valid if it is satisfied in all models given any variable assignment. $\models \phi$
- Valid formulas are uninformative, as they do not rule out possibilities.
- A formula that is not valid is called invalid. $\not\models \phi$
- Invalid formulas are informative, as they rule out possibilities.

Given a first-order formula ϕ ,
is ϕ informative/invalid or uninformative/valid?

Informativity Checking Task

- An argument with a finite set of premises $\phi_1, \ldots, \phi_n$ and a conclusion ψ is valid if the formula $(\phi_1 \wedge \ldots \wedge \phi_n) \rightarrow \psi$ is valid.
- More formally:

Semantic Deduction Theorem:

$\phi_1, \ldots, \phi_n \models \psi$ iff $\models (\phi_1 \wedge \ldots \wedge \phi_n) \rightarrow \psi)$

Given an argument μ with a finite set of premises $\phi_1, \ldots, \phi_n$ and a conclusion ψ,
is μ informative/invalid or uninformative/valid?

Informativity Checking Task

- Is the informativity checking task computationally decidable? No, as before.

- But a partial solution can be reached by moving from a model-theoretic (semantic) perspective to a proof-theoretic (syntactic) perspective (Chapters 4-5)

Relating Consistency and Informativity

- ϕ is consistent/satisfiable iff $\neg\phi$ is informative/invalid.
- ϕ is inconsistent/unsatisfiable iff $\neg\phi$ is uninformative/valid.
- ϕ is informative/invalid iff $\neg\phi$ is consistent/satisfiable.
- ϕ is uninformative/valid iff $\neg\phi$ is inconsistent/unsatisfiable.

Exercises

- Mandatory: 1.1.1, 1.1.3-1.1.5, 1.1.7, 1.1.10
- Optional: 1.1.6, 1.1.11, 1.1.17, 1.2.1