CSC1018 - Logic

Predicate Logic

David Sinclair

First-Order Logic

First-order logic is an extension of propositional logic that includes predicates that are interpreted as relations on a domain.

Example: `prime(x)` is a unary relation that maps from $\mathcal{N} \mapsto \{T, F\}$.

`prime(0)=F` `prime(1)=F` `prime(2)=T`
`prime(3)=T` `prime(4)=F` `prime(5)=T`
`prime(6)=F` `prime(7)=T` `prime(8)=F`
...

Example: `square(x,y)` is a binary relation that maps from $\mathcal{N}^2 \mapsto \{T, F\}$.

`square(0,0)=T`, `square(1,1)=T`, `square(2,4)=T`,
`square(3,9)=T`, `square(4,16)=T`, ...
`square(0,1)=F`, `square(1,0)=F`, `square(0,2)=F`,
`square(1,2)=F`, `square(2,1)=F`, ...

First-Order Logic (2)

Let $\mathcal{P}, \mathcal{A}$ and $\mathcal{V}$ be a countable set of *predicates*, *constant symbols* and *variables*. Each predicate symbol, $p^n \in \mathcal{P}$, has an associated *arity*, n, that defines the number of *arguments* it takes.

By convention, we will use p, q, r as predicate symbols, x, y, z as variables and a, b, c as constant symbols.

Let $\forall$ be the *universal quantifier*. $\forall x.p(x)$ is read as "for all x $p(x)$ is true".
Let $\exists$ be the *existential quantifier*. $\exists x.p(x)$ is read as "there exists an x $p(x)$ is true".

First-Order Logic (3)

An *atomic formula* is an n-ary predicate with n arguments $p(t_1, t_2, \ldots, t_n)$ where each argument t_i is either a variable or a constant.

A *formula* is recursively defined as:

- A formula is an atomic formula.
- $\neg F$ is a formula if F is a formula.
- F_1 *op* F_2, where *op* is a binary operator, is a formula if F_1 and F_2 are formulae.
- $\forall x.F$ is formula if F is a formula. $\forall x.F$ is called a *universally quantified formula* or *universal formula*.'pause
- $\exists x.F$ is formula if F is a formula. $\exists x.F$ is called an *existentially quantified formula* or *existential formula*.

Scope of Variables

In the formulae $\forall x.F$ and $\exists x.F$ the *scope* of the *quantified variable* x is the formula F.

Let F be a formula, then an occurrence of a variable x in F is a *free variable of* F if and only if x is not in the scope of a quantified variable x.

A variable that is not free is *bound*.

A formula with no free variable is *closed*.

Example: $\exists y.p(x, y)$ has one free variable x and one bound variable y.

Example: $\forall y.\exists x.p(x, y)$ is a closed formula as both x and y are bound variables.

Interpretations

If A is a formula with predicates $\{p_1, p_2, \ldots, p_m\}$ and constants $\{a_1, a_2, \ldots, a_k\}$, then an *interpretataion* $\mathcal{I}_A$ for A is a triple

$$(D, \{R_1, R_2, \ldots, R_m\}, \{d_1, d_2, \ldots, d_k\})$$

where D is a *domain*, R_i is an n_i-ary relation on D that is assigned to n_i-ary predicate p_i and $d_i \in D$ is assigned to constant a_i.

Example: The formula $\forall x.p(a, x)$ has multiple interpretations:

$\mathcal{I}_1 = (\mathcal{N}, \{\leq\}, \{0\})$, $\mathcal{I}_2 = (\mathcal{S}, \{substr\}, \{""\})$

The domain $\mathcal{N}$ is the set of natural numbers and the domain $\mathcal{S}$ is the set of strings.

Validity and Satisfiability

Let A be a closed first-order formula.

- A is true in $\mathcal{I}$, $\mathcal{I} \vDash A$, if and only if $v_{\mathcal{I}}(A) = T$.
- A is *valid*, $\vDash A$, if A is true for all interpretations.
- A is *satisfiable* if for some interpretation, $\mathcal{I} \vDash A$.
- A is *unsatisfiable* if it is not satisfiable.
- A is *falsifiable* if it is not valid.

Logical Equivalences

Let A and B be closed first-order formula, then A *is logically equivalent to* B, $A \equiv B$, if and only if $v_{\mathcal{I}_U}(A) = v_{\mathcal{I}_U}(b)$ for all interpretations $\mathcal{I}_U$.

$$A \equiv B \text{ if and only if } \vDash A \leftrightarrow B$$

Let A be a closed first-order formula and $U = \{A_1, \ldots, A_n\}$ be a set of closed first-order formulae, then A *is a logical consequence of* U, $U \vDash A$, if and only if for all $A_i \in U$ $v_{\mathcal{I}_{U \cup A}}(A_i) = T$ implies $v_{\mathcal{I}_{U \cup A}}(A) = T$.

$$U \vDash A \text{ if and only if } \vDash (A_1 \wedge \ldots \wedge A_n) \rightarrow A$$

Logical Equivalences (2)

Duality

$$\models \forall x.A(x) \leftrightarrow \neg\exists x.\neg A(x)$$
$$\models \exists x.A(x) \leftrightarrow \neg\forall x.\neg A(x)$$

Commutativity

$$\models \forall x.\forall y.A(x,y) \leftrightarrow \forall y.\forall x.A(x,y)$$
$$\models \exists x.\exists y.A(x,y) \leftrightarrow \exists y.\exists x.A(x,y)$$

but the combination of $\forall$ and $\exists$ commute only one way.

$$\models \exists x.\forall y.A(x,y) \rightarrow \forall y.\exists x.A(x,y)$$

Logical Equivalences (3)

Distributivity

Universal quantification distributes over conjunction and existential quantification distributes over disjunction.

$$\models \forall x.(A(x) \wedge B(x)) \leftrightarrow \forall x.A(x) \wedge \forall x.B(x)$$
$$\models \exists x.(A(x) \vee B(x)) \leftrightarrow \exists x.A(x) \vee \exists x.B(x)$$

Universal quantification distributes only one way over disjunction and existential quantification distributes only one way over conjunction.

$$\models \forall x.(A(x) \vee B(x)) \rightarrow \forall x.A(x) \vee \forall x.B(x)$$
$$\models \exists x.(A(x) \wedge B(x)) \rightarrow \exists x.A(x) \wedge \exists x.B(x)$$

Logical Equivalences (4)

Distribution over equivalence works in one direction only.

$$\models \forall x.(A(x) \leftrightarrow B(x)) \rightarrow \forall x.A(x) \leftrightarrow \forall x.B(x)$$
$$\models \exists x.(A(x) \leftrightarrow B(x)) \rightarrow \exists x.A(x) \leftrightarrow \exists x.B(x)$$

When distributing over implication the following formulae hold:

$$\models \forall x.(A(x) \rightarrow B(x)) \rightarrow \forall x.A(x) \rightarrow \exists x.B(x)$$
$$\models \forall x.(A(x) \rightarrow B(x)) \rightarrow \exists x.A(x) \rightarrow \exists x.B(x)$$
$$\models \exists x.(A(x) \rightarrow B(x)) \leftrightarrow \forall x.A(x) \rightarrow \exists x.B(x)$$
$$\models (\exists x.A(x) \rightarrow \forall x.B(x)) \rightarrow \forall x.(A(x) \rightarrow B(x))$$

When a subformula does not contain the quantified variable as a free variable, i.e. x is not free in B, then distribution can be applied. For example:

$$\models \forall x.(A(x) \vee B) \leftrightarrow \forall x.A(x) \vee B \qquad \models \exists x.(A(x) \vee B) \leftrightarrow \exists x.A(x) \vee B$$
$$\models \forall x.(A(x) \wedge B) \leftrightarrow \forall x.A(x) \vee B \qquad \models \exists x.(A(x) \wedge B) \leftrightarrow \exists x.A(x) \vee B$$
$$\models \forall x.(A(x) \leftrightarrow B) \leftrightarrow \forall x.A(x) \leftrightarrow B \qquad \models \exists x.(A(x) \rightarrow B) \leftrightarrow \exists x.A(x) \rightarrow B$$

Semantic Tableaux

The semantic tableaux method can be extended to first-order logic, but we need to systematic in the construction of the tableaux and we need to be aware of the following special cases:

- Universal quantifiers need to work for all constants.
- Existential quantifiers shouldn't be instantiated with the same constant twice.
- Universal quantifiers are not used up during the construction of the tableaux.
- A branch in the tableaux may not terminate.
- Open branches with universal quantifiers may terminate.

Semantic Tableaux (2)

The rules for semantic tableaux for first-order logic are the same as proposition logic with the addition of the following two sets of rules.

γ	$\gamma(a)$
$\forall x.A(x)$	$A(a)$
$\neg\exists x.A(x)$	$\neg A(a)$

δ	$\delta(a)$
$\exists x.A(x)$	$A(a)$
$\neg\forall x.A(x)$	$\neg A(a)$

A *semantic tableau* of a formula ϕ is a tree $\mathcal{T}$ where each node is a pair $(U(n), C(n))$ where $U(n) = \{A_1, \ldots, A_{n_k}\}$ is a set of formulae and $C(n) = \{c_{n_1}, \ldots, c_{n_m}\}$ is a set of constants.

Initially, $\mathcal{T}$ consists of a single node n_0

$$(\{\phi\}, \{a_{0_1}, \ldots, a_{0_k}\})$$

where $\{a_{0_1}, \ldots, a_{0_k}\}$ is the set of constants that appear in ϕ.
If ϕ has no constants, take the first constant in the set $\mathcal{A}$, say a_0, initially we have $(\{\phi\}, \{a_0\})$.

Semantic Tableaux (3)

The semantic tableau $\mathcal{T}$ is built inductively by choosing an unmarked leaf $l = (U(l), C(l))$ and applying the **first** applicable rule from the following:

- If $U(l)$ contains a complementary pair of literal, mark the leaf l as *closed*.
- If $U(l)$ is not a set of literal, choose a formula A in $U(l)$ that is an α-, β- or δ-formula.
  - If A is an α-formula, create a new child node l' of l
    $$((U(l) - \{A\}) \cup \{\alpha_1, \alpha_2\}, C(l))$$
  - If A is an β-formula, create two new children nodes l' and l'' of l
    $$((U(l) - \{A\}) \cup \{\beta_1\}, C(l))$$
    $$((U(l) - \{A\}) \cup \{\beta_2\}, C(l))$$
  - If A is an δ-formula, create a new child node l' of l
    $$((U(l) - \{A\}) \cup \{\delta_1(a')\}, C(l) \cup \{a'\})$$
    where a' is some constant that does not appear in $U(l)$.

Semantic Tableaux (4)

- Let $\{\gamma_{l_1}, \ldots, \gamma_{i_m}\} \subseteq U(l)$ be all the γ-formulae in $U(l)$ and let $C(l) = \{c_{l_1}, \ldots, c_{l_k}\}$. Create a new child node l' of l

$$(U(l) \cup \{\bigcup_{i=1}^{m} \bigcup_{j=1}^{k} \gamma_{l_i}(c_{l_j})\}, C(l))$$

  If $U(l)$ only consists of literals and γ-formulae and if $U(l')$ would be the same as $U(l)$ then do not create $U(l')$ and mark the leaf as *open*.

If all the branches of the semantic tableau $\mathcal{T}$ are closed then the semantic tableau $\mathcal{T}$ is *closed*, otherwise $\mathcal{T}$ is *open*.

Remember that semantic tableau is a *refutation procedure*, so to show that A is *valid* we need to show that the semantic tableau for $\neg A$ is closed.

Semantic Tableau Example

Example: Show $\forall x.p(x) \vee \forall x.q(x) \rightarrow \forall x.(p(x) \vee q(x))$ is valid.