COMP 409, Lecture 19

March 27, 2003

1 Relational Structure of Predicate Calculus

Let $A = (D, P_1^A, ..., P_i^A, ..., f_1^A, ..., f_i^A, ...)$

where D is the domain of objects (universe), and the P_i^A (relations) and f_i^A (functions) are called the interpretation I.

If P_i is a k-ary relation name, then $P_i^A \subseteq D^k$.

If f_i is a k-ary function name then $f_i^A : D^k \to D$.

If f_i is a 0-ary function then $f_i^0 : D^0 \to D$ (f_i is a constant function), and same for the boundary case for relations. These boundary cases correspond to propositional logic we had before.

Examples of mathematical structures:

- $\mathbb{R} = (\mathbb{R}, <^{\mathbb{R}}, =, +, -, \times)$
- Graph $G = (V, E)$ where V = nodes (universe) and E = edge (binary relation)

2 Semantics of First-Order Logic

To define semantics of First-Order logic we define:

$\alpha : var \to D$ (a variable assignment)

Definition 1 *$\overline{\alpha} : Term \to D$ such that: (Use induction on structure of terms)*

1. *If $x \in Var$, then $\overline{\alpha}(x) = \alpha(x)$.*
2. *If $f(t_1, ..., t_k) \in Term$, $\overline{\alpha}(f(t_1, ..., t_k)) = f^A((\overline{\alpha}(t_1), ..., \overline{\alpha}(t_k))$ (if $t_1, ..., t_k$ are terms).*

Before, truth was a binary relation (a relation between a truth assignment and formula). Now it is a ternary relation. How do we represent this?

We will use A (syntax structure), $\alpha \models \varphi$ (which is equivalent to $A \models_\alpha \varphi$ or $\alpha \models_A \varphi$).

Using induction, we have:

Base: $A, \alpha \models P(t_1, ..., t_k)$ if $(\overline{\alpha}(t_1), ..., \overline{\alpha}(t_k)) \in P^A$ holds (can also be written as $P^A(\overline{\alpha}(t_1), ..., \overline{\alpha}(t_k)))$

Inductive Hypothesis:

- $A, \alpha \models (\neg\varphi)$ if $A, \alpha \not\models \varphi$
- $A, \alpha \models (\theta \wedge \psi)$ if $A, \alpha \models \theta$ and $A, \alpha \models \psi$
- $A, \alpha \models (\exists x)\varphi$ if $A, \alpha[x \mapsto a] \models \varphi$ for some $a \in D$
  $A, \alpha \models (\forall x)\varphi$ if $A, \alpha[x \mapsto a] \models \varphi$ for all $a \in D$.

Recall: The meaning of $(\exists x)$: $(\exists x)(\exists y)(\exists z)(\exists n)\ x^n + y^n = z^n$
is that there are some particular values: $x = a, y = b, n = m$ such that $a^m + b^m = c^m$

- $First - class\ objects$: Elements of D
- $Second - class\ objects$: Relations and functions over D
- $Third - class\ objects$: Takes a function as an argument and returns a function (set of functions)

3 Modeling with Logic: Sequential Circuits

Logic has emerged as an important field because of its great modeling power. We will see some examples.

Definition 2 *A* **sequential circuit** *is a tuple* $C = (I, O, R, Out, Trans, Start)$, *where:*

- I = *input*
- O = *output*
- $Out \in 2^I \times 2^R \rightarrow 2^O$
- $Start \in 2^R$.
- $Trans \in 2^I \times 2^R \rightarrow 2^R$.

Note that we can describe $Trans$ (transition function) as a sequence of functions $Trans_i \in 2^I \times 2^R \rightarrow \{0, 1\}$. Each such function is the transition function for one register. Similarly, we can describe *Output* as a sequence of functions $Output_i \in 2^I \times 2^R \rightarrow \{0, 1\}$.

Definition 3 *A run of a sequential circuit* $C = (I, O, R, Out, Trans, Start)$ *is a sequence* $(\bar{i_0}, \bar{r_0}, \bar{o_0})(\bar{i_1}, \bar{r_1}, \bar{o_1})...(\bar{i_k}, \bar{r_k}, \bar{o_k})$ *of elements in* $2^I \times 2^R \times 2^O$ *such that:*

1. *Initiation:* $\bar{r}_0 = start$ *(Circuit starts properly)*
2. *Conseqution:* $\bar{r}_{j+1} = Trans(\bar{i}_j, \bar{r}_j)$
3. *Output:* $\bar{o}_j = Out(\bar{i}_j, \bar{r}_j)$
4. $\bar{o}_j \in 2^{Output}$
5. $\bar{r}_j \in 2^{Reg}$
6. $\bar{r}_0 \in Start$
7. $\bar{o}_j = Output(\bar{i}_j, \bar{r}_j)$
8. $\bar{r}_{j+1} = Trans(\bar{i}_j, \bar{r}_j)$

The first step to modeling sequential circuits as relational structures is to pick the domain and vocabulary. Here we choose to pick our domain to be the set of states, that is $D = 2^I \times 2^R$.

We also define the vocabulary: I, R, O as unary relations (for $i \in I$, $i(x) = true$ if high or $false$ if low).

For the behavior of the state, define a binary translation which is a transition relation: $E^{(x)}$

$$\begin{gathered} A_c = (2^I \times 2^R, i^A, ..., {r_1}^A, ..., {o_1}^A, ..., E^A) \\ {i_j}^A(\bar{i}, \bar{r}) \leftrightarrow i_j = 1 \\ {r_j}^A(\bar{i}, \bar{r}) \leftrightarrow r_j = 1 \\ {o_j}^A(\bar{i}, \bar{r}) \leftrightarrow out(\bar{i}, \bar{r}|_j = 1 \\ E^A((\bar{i}, \bar{r}), (\bar{i}', \bar{r}')) \leftrightarrow Trans(\bar{i}, \bar{r}) = \bar{r}' \end{gathered}$$

Now that we have a mathematical model, does this satisfy our requirements? Not yet. The most common requirement we have is *mutual exclusion* (don't allow two processes to write to the same register at the same time). For this, we can introduce the

Safety Property:
r_j and r_k cannot be high at the same time, which leads us to write:

$$(\forall x)[\neg(r_j(x) \wedge r_k(x))]$$

However, this is not sufficient, since we don't have an inherent "ranking" of states. We then say that $(\overline{i_k}, \overline{r_k})$ is *reachable*. So then we can write:

$$(\forall x)[reachable(x) \rightarrow \neg(r_j(x) \wedge r_k(x))]$$

However, reachable is not described in our vocabulary either. The solution for this is not quite clear yet – we will leave this as a problem for now.

4 Distinction between free and bound variables

It is clear that variables used in quantifiers are different from the other variables in that the quantifier prevents the assignment from affecting it.
For example, consider the formula:

$$(\exists x)(\exists x)p(x)$$

This formula will be satisfiable if the following are satisfiable:

- $A, a \models (\exists x)(\exists x)p(x)$ or
- $A, a[x \mapsto a] \models (\exists x)p(x)$ or
- $A, a[x \mapsto a][x \mapsto b] \models p(x)$

That means if there is an $a \in D$ and there is a $b \in D$, we can find a structure A and an assignment α so that $A, \alpha[x \mapsto a][x \mapsto b] \models p(b)$. From the definition of mapping on assignments, this becomes $A, \alpha[x \mapsto b] \models p(b)$. So the outermost quantifier does not affect the formula at all because the inner quantifier somehow makes x "disappear" from the formula.

Let's formalize this. First we try to find the set of occurred variables in a formula:

Definition 4 *Variables in a term t – $var(t)$ – are given by:*

1. $var : Terms \to 2^{var}$
2. $var(x) = \{x\}$
3. $var(f(t_1, \ldots, t_k)) = \bigcup_i var(t_i)$

Definition 5 *Variables in a formula φ – $var(\varphi)$ – are given by:*

1. $var : Form \to 2^{var}$
2. $var(P(t_1, \ldots, t_k)) = \bigcup_i var(t_i)$
3. $var(\neg\varphi) = var(\varphi)$
4. $var(\theta \wedge \psi) = var(\theta) \cup var(\psi)$
5. $var((\exists x)\varphi) = var((\forall x)\varphi) = var(\varphi) \cup \{x\}$

We can see from examples that variables occurring with a quantifier are treated differently in the assignment. To distinguish them, we call the quantified variables bound, and unquantified variables free.

Definition 6 *Free variables in a formula φ – $fvar(\varphi)$ – are given by:*

1. $fvar : Form \to 2^{var}$

2. $fvar(P(t_1, \ldots, t_k)) = \bigcup_i var(t_i)$

3. $fvar(\neg\varphi) = fvar(\varphi)$

4. $fvar(\theta \circ \psi) = fvar(\theta) \cup fvar(\psi)$

5. $fvar((\exists x)\varphi) = fvar((\forall x)\varphi) = fvar(\varphi) - \{x\}$

For assignments, we can show that we only need to worry about free variables. This is the "Relevance Theorem" for first-order logic.

Lemma 1 *Let t be a term, A a structure, and α_1 and α_2 assignments such that $\alpha_1|_{var(t)} = \alpha_2|_{var(t)}$. Then $\overline{\alpha_1}(t) = \overline{\alpha_2}(t)$.*

Proof: By induction.
Base Case: t is x
Then $\overline{\alpha_1}(x) = \alpha_1(x) = \alpha_2(x) = \overline{\alpha_2}(x)$. *Inductive Case:* t is $f(t_1, \ldots, t_k)$ –
Then

$$\begin{aligned} \overline{\alpha_1}(t) &= f^A(\overline{\alpha_1}(t_1), \ldots, \overline{\alpha_1}(t_k)) \\ &= f^A(\overline{\alpha_2}(t_1), \ldots, \overline{\alpha_2}(t_k)) \\ &= \overline{\alpha_2}(t) \end{aligned}$$

Theorem 1 Relevance Theorem: *Let φ be a formula, A a structure, and α_1, α_2 assignments such that $\alpha_1|_{fvar(\varphi)} = \alpha_2|_{fvar(\varphi)}$. Then,*

$$A, \alpha_1 \models \varphi \textit{ iff } A, \alpha_2 \models \varphi.$$

Proof: By induction. *Base Cases:*

1. φ is $t_1 \approx t_2$[1] –
   $A, \alpha_1 \models \varphi$ iff $\overline{\alpha_1}(t_1) = \overline{\alpha_1}(t_2)$ iff $\overline{\alpha_2}(t_1) = \overline{\alpha_2}(t_2)$ iff $A, \alpha_2 \models \varphi$.

2. φ is $P(t_1, \ldots, t_k)$[2] –
   $A, \alpha_1 \models \varphi$ iff $\langle \overline{\alpha_1}(t_1) \ldots \overline{\alpha_1}(t_k) \rangle \in P^A$ iff $\langle \overline{\alpha_2}(t_1) \ldots \overline{\alpha_2}(t_k) \rangle \in P^A$ iff $A, \alpha_2 \models \varphi$.

Inductive Cases:

1. φ is $\neg\psi$ –
   $A, \alpha_1 \models \varphi$ iff $A, \alpha_1 \not\models \psi$ iff $A, \alpha_2 \not\models \psi$ iff $A, \alpha_2 \models \varphi$. We can use similar reasoning for any boolean connective.

[1] $fvar(\varphi) = var(t_1) \cup var(t_2)$
[2] $fvar(\varphi) = \bigcup_i var(t_i)$

2. φ is $(\exists x)\psi$ [3] –
   $A, \alpha_1 \models \varphi$ iff exists $a \in D$ so that $A, \alpha_1[x \mapsto a] \models \psi$ iff[4] exists $a \in D$ so that $A, \alpha_2[x \mapsto a] \models \psi$ iff $A, \alpha_2 \models \varphi$. For $(\forall x)\psi$, simply replace the word "exists" with "for all" in the previous argument.

The consequence of this theorem is that for $A, \alpha \models \varphi$ to be well defined, α needs only be defined on $fvar(\varphi)$.

Notice what will happen:

$$(\forall x)(\forall y)(p(x, y) \to p(y, x))$$

There are no free variables here. All variables are bound, and for any two assignments, they will always agree (even on $\emptyset$).

Definition 7 *Closed formula or sentence is a formula φ such that it has no free variables ($fvars(\varphi) = \emptyset$)*

Theorem 2 *$\varphi|_{A \text{ is a structure}}, \varphi$ is a sentence, and α_1, α_2 be assignments. Then, $A, \alpha_1 \models \varphi$ iff $A, \alpha_2 \models \varphi$. And we can write $A \models \varphi$.*

[3] $fvar(\varphi) = fvar(\psi) - \{x\}$
[4] Since $\alpha_1[x \mapsto a]|_{fvar(\psi)} = \alpha_2[x \mapsto a]|_{fvar(\psi)}$.