Lecture 25

1 Satisfiability

Given a φ, is it satisfiable? This is a difficult question to answer because even though the syntax is simple, the sentences can be very complex. Consider the two sentences:

$$(\forall x)(\forall y)(R(x,y) \to (\forall z)(R(y,z) \to R(x,z)))$$

$$(\forall x)(\forall y)((R(x,y) \land R(y,z)) \to R(x,z))$$

Both these sentences define the rule for transitivity. However the second one is much more readily understood. More examples:

$$(\forall x)(\forall y)(\forall z)(\forall w)P(x,y,z,w)$$

defines a complete quaternary relation, while

$$(\forall x)(\exists y)(\forall z)(\exists w)P(x,y,z,w)$$

defines a two-round game between two players in which the second player always wins.

Alternation of quantifiers makes it difficult to understand first order sentences. What we need is to somehow control the occurrences of quantifiers in the sentences we consider. This is done by translating general sentences into more restricted *normal forms*, while still preserving satisfiability. This process is called *normalization*.

1.1 Normalization of Syntax

Universal sentences are sentences that do not contain any existential quantifiers. We will consider universal sentences with the additional restriction that all quantifiers occur up front. This is called *Skolem normal form*. Thus, a sentence is in Skolem normal form if it is of the form $(\forall x_1)\ldots(\forall x_n)\theta$ where θ is a quantifier-free formula.

While translating sentences, two kinds of transformations $\varphi \mapsto \varphi'$ are allowed:

1. $\mapsto$ is semantics preserving: $\varphi \models\!\!=\!\!| \varphi'$. For example, $(\forall x)(\forall y)P(x,y) \mapsto (\forall y)(\forall x)P(x,y)$.
2. Equisatisfiability: φ is satisfiable iff φ' is satisfiable.

Rule 1 implies rule 2. Rule 2 is applied more often.

2 Transformations

2.1 Removing Identity

We assume the following Theorem (proved in homework). Note that this preserves satisfiability but not logical equivalence, which is more stronger and we want that for query transformation in case of databases:

Theorem 1 *Let φ be a sentence in first-order logic with identity, then one can construct a first-order sentence φ' without identity, such that φ is satisfiable iff φ is satisfiable.*

This was done by replacing any occurrences of $(x \approx y)$ with a predicate $E(x, y)$ and conjoining at the end the axioms for equivalence relation and congruence:

$$\phi' = \phi[\approx \mapsto E(\ldots)] \wedge Axioms(E)$$

2.2 Renaming

The first idea that we'll introduce is that of variable renaming. We will need the flexibility to rename variables in our transformations. This is accomplished by replacing *free* variables with other variables; bound variables need not be replaced since the truth for the sub-formula where they appear is determined by the quantifier.

Definition 1 *$\varphi[x \mapsto y]$: replace free occurrences of x in φ by y.*

- **Terms.**
  - *Base Case*: $x[x \mapsto y] = y$ and $z[x \mapsto y] = z$.
  - *Induction*: $f(t_1, \cdots, t_k)[x \mapsto y] = f(t_1[x \mapsto y], \cdots, t_k[x \mapsto y])$
- **Formulas.**
  1. $P(t_1, \cdots, t_k)[x \mapsto y] = P(t_1[x \mapsto y], \cdots, t_k[x \mapsto y])$
  2. $(\neg\varphi)[x \mapsto y] = \neg(\varphi[x \mapsto y])$
  3. $(\varphi \circ \psi)[x \mapsto y] = (\varphi[x \mapsto y] \circ \psi[x \mapsto y])$
  4. $$((\forall u)\varphi)[x \mapsto y] = \begin{cases} (\forall u)\varphi & \text{if } u = x \\ (\forall u)(\varphi[x \mapsto y]) & \text{otherwise} \end{cases}$$

Theorem 2 *Suppose that y is a "fresh" variable (i.e. does not occur in a formula). Then $(\forall x)\varphi \models\dashv (\forall y)(\varphi[x \mapsto y])$ and $(\exists x)\varphi \models\dashv (\exists y)(\varphi[x \mapsto y])$.*

Proof: By simple induction over the structure of formulas using the definition of renaming.

2.3 Prenex Normal Form

The second idea that we introduce is that of *Prenex Normal Form*, which allows us to transform all formulas to forms where all quantifiers precede a quantifier-free term.

Definition 2 φ *is in* prenex form (prenex normal form) if it is of the form $(Q_1x_1)\cdots(Q_nx_n)\theta$ where each $Q_i \in \{\forall, \exists\}$ and θ is a quantifier free formula.

Here again we call $(Q_1x_1)\cdots(Q_nx_n)$ the *prefix* and θ the *matrix* of φ.

Our goal now is to convert every formula into prenex normal form.

Theorem 3 *Every formula φ has a logically equivalent formula φ' in prenex normal form.*

Proof: By induction on the length of the formula as usual.

1. Base case: $P(t_1, \cdots, t_k)$ is already in prenex normal form.
2. φ is $(\neg\theta)$: By I.H. there is a quantifier-free ψ such that $\theta \models\!\!\!=\!\!\!| \; (Q_1x_1)\cdots(Q_mx_m)\psi$. Notice that $\neg(\forall x)\phi \models\!\!\!=\!\!\!| \; (\exists x)(\neg\phi)$. So we push the negation through. Therefore $\varphi' = (\overline{Q_1}x_1)\cdots(\overline{Q_m}x_m)(\neg\psi)$ where $\overline{\forall} = \exists$ and $\overline{\exists} = \forall$.
3. φ is $(\exists x)\theta$: Let θ' be the prenex form of θ and rename $\theta'[x \mapsto x']$, where x' is a variable that does not occur in ψ so that we satisfy the prenex requirement of all quantified variables being different. Now $\varphi' = (\exists x')\theta'$
4. (for simplicity assume only one binary connective $\wedge$). Assume $\varphi = \varphi_1 \wedge \varphi_2$. Take φ_1 to φ_1' <u>and</u> φ_2 to φ_2' using renaming to make sure that the quantified variables of φ_1' do not occur in φ_2' and vice versa. Let us call p the quantified prefix of a formula in prenex form. Then $\varphi_1' = p_1\theta_1$ and $\varphi_2' = p_2\theta_2$. Then $\varphi' = (p_1)(p_2)(\theta_1 \wedge \theta_2)$. (In fact we can randomly shuffle p_1 and p_2 in order to get a quantified prefix for φ').

3 Skolemization

Skolemization is the process of eliminating existential quantifiers from a formula in prenex normal form. We say that a formula $\varphi_\forall$ is in *skolem normal form* if it is of the form $(\forall y_1)\ldots(\forall y_n)\theta$, where θ is quantifier-free formula. This transformation does not preserve semantics but it does preserve satisfiability: for every formula φ, there exists a universal sentence φ_U such that φ is satisfiable iff φ_U is satisfiable. Thus, skolemization is an equisatisfiable transformation. We begin with a simple case.

Theorem 4 *Let ϕ be a first order formula $\phi = (\forall x)(\exists y)\theta$. Then, there exists $f(x)$ such that ϕ is satisfiable iff $\phi' = (\forall x)\theta[y \mapsto f(x)]$ is satisfiable (where y is free in θ).*

Proof.
$\boxed{\Rightarrow}$: Assume $A \models (\forall x)\,(\exists y)\,\theta$. That is, for all $a \in D^A$ there exists $b \in D^A$ such that $A, \alpha[x \mapsto a, y \mapsto b] \models \theta$. Notice that selecting b depends only on the choice for a. Let f be a function that returns the choice for b, given a.

Let the structure A' be the result of adding the function f^A to A. We claim that $A' \models \forall x\, \theta[y \mapsto f(x)]$. In other words, we want to show that for all $a \in D^A$, we have $A', \alpha[x \mapsto a] \models \theta[y \mapsto f(x)]$ where $f^A(a)$ is some b such that $A, \alpha[x \mapsto a, y \mapsto b] \models \theta$. Note that b may not be unique, and this proof depends upon the Axiom of Choice to choose a particular b.

By a simple inductive argument, it can be shown that $A', \alpha[x \mapsto a] \models \theta[y \mapsto f(x)]$ iff $A, \alpha[x \mapsto a, y \mapsto b] \models \theta$.

$\boxed{\Leftarrow}$: Suppose A is a structure such that $A, \alpha \models \forall x\, \theta[y \mapsto f(x)]$. That is, for all $a \in D^A$ we have $A, \alpha[x \mapsto a] \models \theta[y \mapsto f(x)]$. We need to show that for all $a \in D^A$, there is some $b \in D^A$ such that $A, \alpha[x \mapsto a, y \mapsto b] \models \theta$.

Take $b = f^A(a)$. We need to prove that $A, \alpha[x \mapsto a, y \mapsto f^A(a)] \models \theta$ iff $A, \alpha[x \mapsto a] \models \theta[y \mapsto f(x)]$. We can do this by structural induction on a first order formula.

Hint: use a "Little Lemma" which says that

$$\overline{\alpha[x \mapsto a, y \mapsto b]}(t) = \overline{\alpha[x \mapsto a]}(t[y \mapsto b])$$

(Recall that $\overline{\alpha}$ is the $Term \to D$ mapping.)

$\square$

This transformation can be easily generalized. Suppose $\phi = (\forall x_1) \ldots (\forall x_n)(\exists y)\theta$ and let A be a structure such that $A \models \phi$. Then we know that for every n-tuple $\langle a_1, \ldots, a_n \rangle \in (D^A)^n$ there exists a $b \in D^A$ such that $A \models \theta[\alpha]$ where the assignment α maps x_i to a_i and y to b. This allows us to construct a table that for every n-tuple indicates the value of y that satisfies θ. In cases where y could have more than one value, we simply choose one of them.

Observe that this table maps each n-tuple in D^A to a unique element of D^A. Therefore, it defines a function $f^A(n)$ on the domain of A. We now define a new structure A' which assoiciates the unique symbol f to the function defined above. Then our transformation replaces every occurrence of y with $f(x_1, \ldots, x_n)$.