Lecture 32

November 21, 2005

1 Classification

Bruce Lindsay: "Relational databases provide the foundation for western civilization". And relational databases are indeed very important to our day-to-day life. In this lecture our goal will be to classify formulas according to their satisfiability and/or validity with respect to a class C. We recall that doing one of the two is sufficient since φ is satisfiable iff $\neg\varphi$ is not valid and φ is valid iff $\neg\varphi$ is not satisfiable.

1.1 Search for truth

For any propositional formula φ we have the following cases:
$\tau \models \varphi$ where $\tau \in 2^{Prop}$
$\Gamma \models \varphi$ and φ is true for every truth assignment in, $\Gamma \subseteq 2^{Prop}$
$\models \varphi$ and φ is true for all possible assignments

We have similar sort of cases in first order logic, depending on the form of C; at one end of the spectrum, we have a class which consists of a single structure: $C = \{\mathcal{A}\}$. The answer then depends on the structure. An example of this is a database as a structure and C as a query on that. For finite structures, truth is decidable: simply examine the formula recursively, with each quantifier generating the full domain for each quantified variable. On the other hand, for an arithmetical structure $(\mathbb{N}, +, \times, 0)$, truth is not even recursively enumerable (Gödel's Incompleteness Theorem). The truth is also known to be decidable in geometrical structure $(\mathbb{R}, +, \times, 0)$ and Dresbuger arithmetical structure $(\mathbb{N}, +, 0)$ where we don't have multiplication.

Moving to the other end of the spectrum, we have the case where the class C includes infinite structures over the language of φ. In this case, the problem changes to that of validity; does every structure satisfy the sentence? It is shown that for the class of all structures validity is undecidable but recursively enumerable by Turing and Church.Gödel actually proved that validity for first order logic is R_E.We can also prove that any deductive system that is both sound and complete is R_E, as $\models \varphi \leftrightarrow \vdash \varphi$ and strings of all lengths can be enumerated and checked if they constitute a proof.

2 Transformations

In general, sentences can be written in a very convoluted and complex manner even when they convey simple meanings. For example consider the formula:
$\forall x(P(x) \to \forall y(E(x,y) \to \forall x(E(y,x) \to P(x))))$
Here P(x) is true iff x is labeled "P" and E(x,y) tells if there is an edge between x and y. This formula says that -if you start from a node x, which is labeled "P" and take two edges, you will reach another vertex with same label. But this is a very complicated way of writing that. An easier way would be: $\forall x \forall y \forall z(P(x) \land E(x,y) \land E(y,z) \to P(z))$. Hence It would be nice if we could rewrite all formulas to some normal form, which would be easier to compute with. In particular, formulas where all the quantifiers precede everything else (such as $\forall x \forall y \forall z P(x,y,z)$) seem to be simpler to work with. Moreover it would be desirable to have all quantifiers be universal and not intermixed, since it is easier to test satisfiability for such formulas. For example, formula $\forall x_1 \forall x_2 \dots \forall x_n P(x_1, x_2, \dots, x_n)$ is much easier to understand and compute than $\forall x_1 \exists x_2 \dots \forall x_{n-1} \exists x_n P(x_1, x_2, \dots, x_n)$, which has alternating quantifiers.

Definition 1 *A first-order sentence φ is* universal *if it is of the form $\forall x_1 \dots \forall x_n\ \theta$ where θ is quantifier-free. Here, $\forall x_1 \dots \forall x_n$ is called Quantifiers and θ is called Matrix.*

Our ultimate goal will be to transform all formulas to universal formulas while preserving the satisfiability property. To this end we introduce the following 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:

$$\phi' = \phi[\approx \mapsto E(\dots)] \land Axioms(E)$$

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 2 *$\varphi[x \mapsto y]$: 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 Forms*, which allows us to transform all formulas to forms where all quantifiers precede a quantifier-free term.

Definition 3 *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 $\theta \models\!\dashv (Q_1x_1)\cdots(Q_mx_m)\psi$. 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').