Lecture 31

November 18, 2005

1 Truth and Complexity

Recall that while discussing propositional logic, we were particularly interested in the complexity of deciding the truth of sentences. The two kinds of questions were:

1. Given τ, φ does $\tau \models \varphi$? — LOGSPACE
2. Given φ, is $\models \varphi$? — co-NP complete

It is natural to consider similar questions for first order logic. We would like to know how difficult is it, algorithmically, to find out whether a sentence is valid or whether a given structure satisfies a sentence. We also would like to know whether such questions can be answered algorithmically.

The analogous questions for first order logic are:

1. Given A, φ, does $A \models \varphi$? — LOGSPACE to PSPACE-complete for finite A. Note that this is the same problem as query evaluation.
2. Given φ, is $\models \varphi$? — Undecidable. This result is known as the Church-Turing theorem.

What happens if A is infinite in problem 1 above? As an example, consider $A = (N, +, *, exp, =)$. Then Fermat's last theorem can be expressed as a formula ψ and we know (because of Andrew Wiles' celebrated proof) that $A \models \psi$. So if problem 1 above was decidable, the algorithm could be used to prove Fermat's last theorem, and for that matter any theorem in number theory. Thus the decidability of problem 1 has very important consequences for mathematics. At the turn of the century, it was hoped that this question would be answered in the affirmative.

For this particular choice of infinite structure, the problem is undecidable. However, in general, the answer depends on the structure.

- $A = (N, +, *, =)$ is undecidable. Even the highly restricted form of $\varphi = (\exists x P(x) = 0)$, where $P(x)$ denotes a polynomial, is undecidable. That is, there is no algorithm to determine whether a given polynomial has an integer root.

- $A = (R, +, *, <)$ is decidable.

Thus there is a natural problem lying between problems 1 and 2 above:

Given a class C of structures and sentence φ, does $A \models \phi$ for all $A \in C$?

The answer varies and depends on C. An example of this variation, where the answer changes with the class of structures being considered:

- C_1 = class of directed graphs — Undecidable
- C_2 = class of undirected graphs — Undecidable
- C_3 = class of equivalence relations — Decidable

Note that a directed graph is just an general binary relation, an undirected graph is a symmetric binary relation, and an equivalence relation is a symmetric, reflexive and transitive relation. Thus as the underlying structure grows more constrained, the problem shifts from being undecidable to decidable.

2 Undecidability

Consider the two questions:

1. Given a program, does it terminate?
2. Given a program, does it diverge?

There is a difference between the two. Given that a program terminates, we can observe it terminating (by just running it and waiting). However, if the program does not terminate, then we can never verify this fact. This immediately shows that there are different degrees of decidability. We say a problem is *semi-decidable* if there is an algorithm for it that always halts on correct inputs. So semi-decidability has positive witnesses (like NP). This notion is also called *recursive enumerability.*

Going back, $(N, +, *, =)$ is not semi-decidable. Validity ($\models \varphi$), however is undecidable but semi-decidable. So, theorem provers can find proofs if they exist. An immediate problem that arises while considering validity, is how to consider all possible structures (which are uncountable in number). The solution lies in looking at proofs instead. Proofs are finite objects and by Gödels completeness theorem, φ is valid iff φ has a proof.

3 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:

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

$$(\forall x)(\forall y)((R(x,y) \wedge R(y,z)) \rightarrow R(x,z))$$

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.*

3.1 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 *prenex normal form.* Thus, a sentence is in prenex 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: $\varphi \models\!=\!\mid \varphi'$. 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.