Lecture 25: COMP 409 – April 21, 2003

Sumit Mittal

April 28, 2003

1 Satisfiability

We had proved the following theorem in the last class - *Validity of FOL is recursively enumerable.*

Let us review the proof sketch:
a) Simplifying structure.

1. Convert to prenex-normal form – preserves semantics.
2. Eliminate $\approx$ – preserves satisfiability.
3. Eliminate $\exists$ – preserves satisfiability.
4. Skolemize - gives universal formula.

This leaves us with sentences of fairly simple syntax, i.e., $\forall y_1 \ldots \forall y_n \theta$. The price of this transformation is that it introduces additional functions.

b) Herbrand theorem: - If a universal sentence φ is satisfiable, then it has a Herbrand model.

c) If A is an Herbrand structure, then $A \models \varphi$ iff $ground(\varphi)$ is satisfiable.

d) Propositionalize ground(φ) and use Compactness theorem.

Consider SPL - Simple Programming Language, having the following
1. Assignments: X = X + 1, X = X -1
2. Branching: goto l
3. Conditional: If X = 0, then goto l

Decision problem for a Counter Machine (CM) - Starting at statement 0, with all variables initialized to 0, will machine reach a statement k with all variables (counters) zeroes ??

Problem is decidable for 1CM. (1CM is similar to push down automaton)

Problem for 2CM is undecidable.(simulates a Turing Machine)

Consider $S_i(t_1, t_2)$ to mean that the machine is in line i with t_1, t_2 in its counters.
Let our starting state be $S_0(0,0)$.

We can model the properties of the machine as:

1. $(\forall x)(\forall y)(S_i(x,y)) \rightarrow S_{i+1}(f(x),y))$ for X = X + 1.
2. $(\forall x)(\forall y)(S_i(f(x),y)) \rightarrow S_{i+1}(x,y))$ for X = X - 1.
3. $(\forall x)(\forall y)(S_i(x,y)) \rightarrow S_l(x,y)$ for goto l.
4. $(\forall x)(\forall y)(S_i(0,y)) \rightarrow S_l(0,y)$ for If X = 0, goto l.

Lemma 1 *The machine M reaches statement k with zeroed variables iff*
$\models (\varphi_M \rightarrow S_k(0,0))$

Here, φ_M corresponds to the formula represented by the machine M.
Let us try to prove a stronger lemma.

Lemma 2 *If in step 'i' of the execution, M is at statement p with counters r, s, then*
$\models \varphi_M \rightarrow S_p(f^r(0), f^s(0))$

The proof follows by induction on $'i'$.

Now, suppose M does not reach k with zeroes variables. We need to construct a structure A such that $A \models \varphi_M$ but $A \not\models S_k(0,0)$
i.e. $A \models \varphi_M \wedge \neg S_k(0,0)$

It is easy to see that we can construct a Herbrand structure A such that
$A \models \varphi_M \wedge \neg S_k(0,0)$, since the specification of the machine and execution has all universal sentences.

Coming back to validity of FOL.

Theorem 1 (Church Turing theorem) *Validity of FOL is undecidable.*

φ is satisfiable iff $\neg\varphi$ is not valid.

Corollary 1 *Satisfaction is not recursively enumerable.*

This follows from the fact that if a problem is both CO-RE and RE, it is recursive. This corollary is known as an *Incompleteness* Theorem, since it implies that there is no sound and complete proof system for satisfiability.

Now, what about validity over all finite structures. Unfortunately, there we have another negative result.

Definition 1 *φ is valid in the finite if $A \models \varphi$ holds whenver A is finite.*

Theorem 2 (Trakhtenbrot's Theorem) *Validity over finite structures for first-order logic is not recursive.*

Definition 2 *φ is satisfiable in the finite if there is a finite structure A, such that $A \models \varphi$*

What about satisfiability in a finite structure? If the vocabulary is finite, then there are finitely many structures (up to an isomorphism) of a given finite cardinality.

Corollary 2 *Satisfiability in some finite structure is recursively enumerable.*

Hence, validity over finite structures cannot be recursively enumerable.

Let $\text{Arith} = (\mathbf{N}, +, *)$. The truth problem for arithmetics is then $\{\varphi : \text{Arith} \models \varphi\}$.

Theorem 3 (Gödel's Incompleteness Theorem) *The truth problem for arithmetics is not recursive. Hence, it is also not recursively enumerable.*