6c Lecture 16: May 21, 2015

The proof of the Tarski-Seidenberg finishes by generalizing Sturm's algorithm from last time so that it can determine whether some finite collection of polynomials satisfies some combination of inequalities. Lets first reduce the types of formulas we need to consider.

Suppose ϕ is quantifier free. We may as well assume that ϕ is in conjunctive normal form:

$$\phi = \psi_1 \vee \theta_2 \vee \ldots \vee \theta_n$$

Then since $\exists x \phi$ is equivalent to

$$\exists x \psi_1 \vee \exists x \theta_2 \vee \ldots \vee \exists x \theta_n$$

Now since each $\phi_i = \theta_1 \wedge \ldots \wedge \theta_{n_i}$ is a conjunction of atomic formulas or their negations, it is enough to eliminate quantifiers from formulas of the form $\exists x(\theta_1 \wedge \ldots \wedge \theta_{n_i})$ where each θ_i is of the form $p(x) = 0$ or $\neg(p(x) = 0)$ or $p(x) > 0$ or $\neg(p(x) > 0)$.

Now,

- $\neg(p(x) = 0)$ is equivalent to $(p(x))^2 > 0$
- $\neg(p(x) > 0)$ is equivalent to $p(x) \leq 0$ which is equivalent to $-p(x) > 0 \vee p(x) = 0$.
- $p_1(x) = 0 \wedge p_2(x) = 0 \wedge \ldots \wedge p_n(x) = 0$ is equivalent to $(p_1(x))^2 + (p_2(x))^2 + \ldots + (p_n(x))^2 = 0$.

Thus, it is enough to eliminate quantifiers for a formula of the form:

$$\exists x(p(x) = 0 \wedge q_1(x) > 0 \wedge \ldots \wedge q_n(x) > 0$$

Now it will be a homework problem for you to adapt Strum's algorithm to eliminate quantifiers for formulas when there is a single q.

Exercise 11.1. Suppose $p(x)$ and $q(x)$ are polynomials in x of degree $\leq n$. Then for each $k \leq n$ there is a quantifier free formula ϕ_k which is true iff there are k different values of x for which $p(x) = 0$ and $q(x) > 0$.

Given this homework problem, we can do the general case as follows. Suppose first that we want to find the number of roots of $p(x) = 0$ where $q_1(x) > 0$ and $q_2(x) > 0$. Then

- Let A be the number of roots of $p(x) = 0$ where $q_1(x) > 0$ and $q_2(x) \neq 0$.
- Let B be the number of roots of $p(x) = 0$ where $q_1(x) \neq 0$ and $q_2(x) > 0$.

- Let C be the number of roots of $p(x) = 0$ where $q_1(x) \neq 0$ and $q_2(x) \neq 0$.
- Let D be the number of roots of $p(x) = 0$ where $q_1(x) > 0$ and $q_2(x) > 0$ or $q_1(x) < 0$ and $q_2(x) < 0$.

Then the number of roots of $p(x) = 0$ where $q_1(x) > 0$ and $q_2(x) > 0$ is equal to $(A + B - (C - D))/2$.

But

- A is the number of roots of $p(x) = 0$ where $q_1(x)q_2^2(x) > 0$.
- B is the number of roots of $p(x) = 0$ where $q_1(x)^2 q_2(x) > 0$. and $q_2(x) > 0$.
- C is the number of roots of $p(x) = 0$ where $q_1(x)^2 q_2^2(x) > 0$.
- D is the number of roots of $p(x) = 0$ where $q_1(x)q_2(x) > 0$.

So by the homework problem, and an inductive argument if $p(x)$ and $q_1(x), \ldots, q_n(x)$ are polynomials in x all of degree $\leq n$, then for each $k \leq n$ there is a quantifier free formula ϕ_k which is true iff there are k different values of x for which $p(x) = 0$ and $q_1(x) > 0 \wedge \ldots \wedge q_n(x) > 0$.

11.1 Beyond Tarski-Seidenberg

How good is the Tarski-Seidenberg algorithm from a practical perspective? When we have a computer execute it, can it quickly solve interesting problems, such as the kissing spheres problem? The answer is that the algorithm is almost completely useless. Each time a quantifier is eliminated we add exponentially many new equations and so the formulas involved become massive.

Fortunately, significant progress has been made on finding faster algorithms, using techniques such as cylindrical algebraic decomposition. There is an algorithm which decides sentences with n symbols in $O(2^{2^n})$ time, and an algorithm for deciding existential formulas (ones beginning with a single block of existential quantifiers, and containing no other quantifiers) in $O(2^n)$ time. This first result is known essentially be optimal.

Alas, even these improved algorithms are still rather slow when run on practical problems. For example, modern implementations of quantifier elimination are able to solve the kissing spheres in 2 dimensions (with a little ingenuity to make the problem slightly easier, such as fixing the position of the first sphere). However, the kissing spheres problem in higher dimensions is completely out of reach for now (the four dimensional version requires a hundred quantifiers). Still, these algorithms are an important part of almost all computer algebra systems and receive a great deal of use for people working on practical mathematics; there are lots of interesting formulas which are rather short.

Another interesting avenue of investigation is how much the Tarski-Seidenberg theorem can be generalized. Does the theorem remain true when we add more functions to our language so that we can discuss more complicated phenomena? For example, Tarski asked in 1940 whether one can prove the same theorem when exponentiation is added to our language:

Open Problem 11.2. *Is there an algorithm for deciding what sentences are true of the reals, in the first order language built from* $\{+, \times, \exp, =, 0, 1\}$.

Not only is the question an open problem, but we don't even know if there is an algorithm for deciding the truth of sentences such as $e^{-e^2} - 60e^{-15} = e^{-3e^1+2e^{-1}}$ involving no variables or quantifiers! Are there any surprising identities involving exponentiation and the integers beyond obvious ones that follow from the fact that $e^x e^y = e^{x+y}$? This is a difficult problem in transcendental number theory. However, there is a widely believed conjecture due to Schanuel which implies that indeed, the only such true identities are the obvious ones, and that there is an algorithm for deciding quantifier-free sentences. In fact, if Schanuel's conjecture is true, then Macintyre and Wilkie have shown that there is an algorithm for deciding all sentences in the language with exponentiation, settling the entire problem.

What about if we change what number system we use to something other than the real numbers? For example, if we work over the complex numbers instead, then Tarski showed in 1948 that the analogous theorem is true: there is algorithm to decide the truth of sentences in the first order language built from $\{+, \times, =, <, 0, 1\}$ about the complex numbers[1]. How about the natural numbers? In this setting, we can state many difficult open problems like Goldbach's conjecture:

$$\begin{aligned}&\forall n((n \geq 2 \land \exists k(n = 2k)) \to \exists p \exists q(n = p + q \land \\ &\qquad \forall r \forall s((p = rs \to (r = 1 \lor s = 1)) \land (q = rs \to (r = 1 \lor s = 1)))))\end{aligned}$$

Is there a similar algorithm to decide which of these statements are true or false? In his famous speech outlining 23 important problems for twentieth century mathematics, Hilbert made it a goal to find a process to solve a simple class of such problems: find an algorithm for determining whether a a multivariable polynomial has any integer roots, (perhaps similar to the one we have given above to determine whether there are any *real* roots to such a polynomial). This is known as Hilbert's 10th problem.

We finish this section by stating one more famously open problem:

Open Problem 11.3 (Hilbert's tenth problem over $\mathbb{Q}$)**.** *Is there an algorithm for deciding whether a multivariable polynomial with integer coefficients has any rational roots?*

12 Computability

12.1 Decision problems

Definition 12.1 (A first attempt)**.** Suppose D is a set. Given $P \subseteq D$, the decision problem for P is the question:

[1] A year later, Abraham Robinson gave a very pretty model-theoretic proof of this fact

$$\text{Given } I \in D, \text{ is } I \in P \text{ or is } I \notin P?$$

We are interested in an algorithm which answers this question in finitely many steps.

A few things to note. We don't have a definition of algorithm yet, so the above is still fuzzy. We should also note that Hilbert's 10th problem can be stated as such a decision problem: let D be the set of polynomials in several variables with integer coefficients, and let P be such polynomials which have integer roots.

Example 12.2. Let D be the set of formulas in propositional logic, and P be the set of *satisfiable* formulas in propositional logic. We've seen this come up before. There is definitely an algorithm for this one: just try every valuation. There are only finitely many, so that will work. It's not very efficient, though. We'll talk more about such considerations in a couple of lectures.

Definition 12.3. If an algorithm exists for the decision problem P, we call P decidable. Otherwise we call P undecidable.

At this point, it is worth getting more precise about what an algorithm is. Without a precise definition, how could we possibly hope to show a given decision problem is undecidable?

Definition 12.4. An alphabet is an arbitrary nonempty set, whose elements we call symbols.

Definition 12.5. A word (or string) in A is a finite sequence $a_1 \dots a_n$ where each $a_i \in A$. If $w = a_1 \dots a_n$ is a word, then the length of w (which is n here) is written $|w|$.

We write A^n to denote the set of words in A of length n. We let θ denote the empty word, which has length 0. And we let $A^* = \cup_{n \geq 0} A^n$.

Note that we can really encode just about any mathematical objects as words on an appropriately chosen alphabet.

Example 12.6. We are very used to writing the natural numbers as words in $\{0, 1, 2, \dots, 9\}$. We could just as easily use their binary expansions to write them as words in $\{0, 1\}$. One could even use unary notation to write them as words in $\{1\}$.

Example 12.7. We can encode a graph $G = (V, E)$ with vertices $\{v_1, \dots, v_n\}$ as a word in $\{0, 1\}$ as follows. First we identify G with its adjacency matrix $A = [a_{ij}]$, i.e. the matrix where $a_{ij} = 0$ if there is no edge between v_i and v_j, and $a_{ij} = 1$ otherwise. We could just represent G by the word $a_{11}a_{12} \dots a_{1n}a_{21} \dots a_{nn}$.

Example 12.8. Suppose we have a finite first-order language $\mathcal{L} = \{f_1, \dots, f_n, R_1, \dots, R_m\}$. If we represent the variable x_n as xn, with n written in binary, then we can represent any first-order formula in $\mathcal{L}$ as a word in

$$A = \mathcal{L} \cup \{x, 0, 1, \neg, \vee, \wedge, \rightarrow, \leftrightarrow, \exists, \forall, (,), =\}$$

Now we can consider decision problems of a much more definite form than our original definition.

Definition 12.9. A decision problem consists of a finite alphabet A and a set of words $P \subseteq A^*$. We write (A, P) for the problem.

Example 12.10. VALIDITY$_{\mathcal{L}} = (A, P)$, where A is as in the previous example, and

$$P = \{S \mid S \text{ is a sentence in } \mathcal{L} \text{ and } \vDash S\}.$$

Next time, we will give a precise definition of what an algorithm is.