

Lecture Notes for CS 509

Foundations of Computer Science

Lecture 5

Lecturer: Joe Kilian

Scribe: Gayatree Ganu, Darakshan Mir & Crystal Maung

Department of Computer Science

Rutgers, The State University of New Jersey

1 Multivariable Polynomial

The problem is to find the integer roots x_1, x_2, \dots, x_k of the polynomial P . We can describe this in mathematical notation as

$$\exists x_1, x_2, \dots, x_k \in \mathbb{Z} \text{ such that } P(x_1, x_2, \dots, x_k) = 0$$

Note: This problem is also known as Hilbert's 10th Problem.

Example. Consider to find integer roots $x, y, z \in \mathbb{Z}$ for $x^3 + y^3 = z^3$. We can change this into $P(\cdot) = 0$ form by $x^3 + y^3 - z^3 = 0$. This is a special case of Fermat Last Theorem, which is one of the most famous theorems in the history of mathematics. Thanks to Andrew Wile, we know for sure that it has no solution.

Is the problem of finding integer roots for multivariable polynomial Turing recognizable?

Yes.

Proof. For $B = 0, 1, \dots$, try all $0 \leq x_1, \dots, x_k \leq B$.

If there is a solution, B will get big enough to find it.

If there is no solution, it will run forever.

Therefore, it recognizes when the solution exists. □

Is it decidable? No. But we will not be giving the prove in this class.

1.1 Simpler Problem

We are going to prove that the following problem is undecidable.

$\exists x_1, \dots, x_k \forall y_1, \dots, y_k \exists z_1, \dots, z_k P(x_1, \dots, z_k) = 0$ where $x_i, y_i, z_i \in \mathbb{Z}$ where $1 \leq i \leq k$

Proof. We will prove it by reducing PCP(Post Correspondence Problem) into this problem.

An instance of the PCP is a collection P of dominos:

$$P = \left\{ \left[\frac{t_1}{b_1} \right], \left[\frac{t_2}{b_2} \right], \dots, \left[\frac{t_k}{b_k} \right] \right\}$$

and if there is a match, then

$\exists n \geq 1$ such that a sequence i_1, i_2, \dots, i_n , where $t_{i_1} t_{i_2} \dots t_{i_n} = b_{i_1} b_{i_2} \dots b_{i_n}$.

Here n does not have upper bound. Therefore, i grows without bound. But the value of i_j range from 1 to k as there are only k different dominos. We can express this as $1 \leq i_j \leq k$.

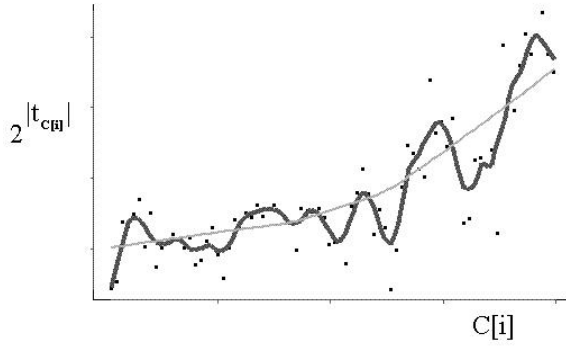
We can represent the list of choices as array C where $C[j] = i_j$. It follows that $1 \leq C[j] \leq k$. Therefore, we have

$$\boxed{\exists n \geq 1, i_1, i_2, \dots, i_n, \text{ where } t_{C[1]} t_{C[2]} \dots t_{C[n]} = b_{C[1]} b_{C[2]} \dots b_{C[n]}}$$

We can change this into

$$\boxed{\begin{array}{l} \exists_{n \geq 1} \forall_{1 \leq i \leq n-1} i \\ T[1] = t_{C[1]} \\ B[1] = b_{C[1]} \\ T[i+1] = T[i] t_{C[i+1]} \\ B[i+1] = B[i] b_{C[i+1]} \\ T[n] = B[n] \end{array}}$$

Let us assume that the strings are comprised of binary characters $\{0, 1\}$. We can replace the string with polynomial as follows:



$$\exists_{n \geq 1} \forall_{1 \leq i \leq n-1} i$$

$$T[1] = 2^{|t_{C[1]}|} + t_{C[1]} \text{ where } |t_{C[1]}| \text{ is the length of } t_{C[1]}$$

$$B[1] = 2^{|b_{C[1]}|} + b_{C[1]}$$

$$T[i+1] = T[i] \cdot 2^{|t_{C[i+1]}|} + t_{C[i+1]}$$

$$B[i+1] = B[i] \cdot 2^{|b_{C[i+1]}|} + b_{C[i+1]}$$

$$T[n] = B[n]$$

For example,

$$t_C[1] = 010$$

$$b_C[1] = 10$$

$$T[0] = 1$$

$$B[0] = 1$$

$$T[1] = 1 \cdot 2^3 + 010 = 1010$$

$$B[1] = 1 \cdot 2^2 + 10 = 110$$

Note: We assume that $T[0] = B[0] = 1$, so that the leading zeros are not neglected in the multiplication, i.e., $001 \neq 01$.

We can represent $2^{|t_{C[i]}|}$ as a polynomial $P(C[i])$ as shown in the figure. Since $1 \leq C[i] \leq k$, we have finite $C[i]$. Therefore, the polynomial function exists for $C[i]$.

$$\exists_{n \geq 1} \forall_{1 \leq i \leq n-1} i$$

$$T[1] = P(C[1]) + Q(C[1])$$

$$B[1] = R(C[1]) + S(C[1])$$

$$T[i+1] = T[i] \cdot P(C[i+1]) + Q(C[i+1])$$

$$B[i+1] = B[i] \cdot R(C[i+1]) + S(C[i+1])$$

$$T[n] = B[n]$$

Now we need to encode array $T[i]$, $B[i]$, $T[i + 1]$, $B[i + 1]$, $T[n]$ and $B[n]$. Before that we will introduce Chinese Remainder Theorem and a Lemma which we will not be proving.

Theorem 1.1 (Chinese Remainder Theorem). *Suppose we have relatively prime integers M_1, \dots, M_n . Then there exists $z \equiv a_i \pmod{M_i}$.*

Therefore $\exists z$ such that $z \equiv T[i] \pmod{M_i}$. Therefore we can replace $T[i]$ with $z \pmod{M_i}$ if we can find relatively prime M_i s.

Lemma 1.2. $\exists C$ such that for $i, j = 1$ to n , $iC + 1$ is relatively prime to $jC + 1$ where $j \neq i$.

Normally C is $n!$.

Using Lemma we can show that there exists z and C , such that

$$\begin{aligned} T[1] &\equiv z \pmod{C + 1} \\ T[i] &\equiv z \pmod{iC + 1} \\ T[i + 1] &\equiv z \pmod{(i + 1)C + 1} \\ T[n] &\equiv z \pmod{nC + 1} \end{aligned}$$

Now, we introduce new variables to get rid of the modulus function. We know that, $\text{mod}(a, b)$ can be expressed as $\exists r, q$ such that $a = qb + r$ and $r < b$.

Therefore, our equations become

$$\begin{aligned} z &= k_1(C + 1) + t_1 && \text{for } T[1] \\ z &= k_i(iC + 1) + t_i && \text{for } T[i] \\ z &= k_{i+1}((i + 1)C + 1) + t_{i+1} && \text{for } T[i+1] \\ z &= k_n(nC + 1) + t_n && \text{for } T[n] \end{aligned}$$

where $t_1 < C + 1$, $t_i < (iC + 1)$, $t_{i+1} < ((i + 1)C + 1)$ and $t_n < nC + 1$. Similarly, we can apply the same to B .

$$\exists_{n \geq 1} \forall_{1 \leq i \leq n-1} i \exists_{z, C, k1, ki, kip, kn, k1', ki', kip', kn'} \exists_{t1, b1, ti, bi, tip, bip, tn, bn}$$

$$t1 = P(C[1]) + Q(C[1])$$

$$b1 = R(C[1]) + S(C[1])$$

$$tip = ti.P(C[i+1]) + Q(C[i+1])$$

$$bip = bi.R(C[i+1]) + S(C[i+1])$$

$$z = k1(C+1) + t1$$

$$z = ki(iC+1) + ti$$

$$z = kip((i+1)C+1) + tip$$

$$z = k1'(C+1) + b1$$

$$z = ki'(iC+1) + bi$$

$$z = kip'((i+1)C+1) + bip$$

$$t1 < C+1$$

$$b1 < C+1$$

$$ti < iC+1$$

$$bi < iC+1$$

$$tip < (i+1)C+1$$

$$bip < (i+1)C+1$$

$$tn < nC+1$$

$$bn < nC+1$$

$$kn(nC+1) + tn = kn'(nC+1) + bn$$

We can change $\forall_{i \leq n-1} P(\cdot)$ with $\forall i P(\cdot) = 0$ or $i > n-1$. We can further change this by introducing a new variable $\lambda > 0$ as $\forall i P(\cdot) = 0$ or $i - \lambda + 1 = n$. Note that we ignore the part $1 \leq i$ because it is already expressed by \exists condition.

We can change all the equations in the form $a = b + c$ into $a - b - c = 0$ form. We can also change all the variables in $a < b$ into $a + c - b = 0$ where $c > 0$. Also we replace $n \geq 1$ with $n - q - 1 = 0$ where $q \geq 0$. If we assume 0 as true and nonzero numbers as false, we can connect all the equations by 'and' operator.

Therefore, the equations will become

$$\exists_n \forall_i \exists_{z,C,k1,ki,kip,kn,k1',ki',kip',kn',t1,b1,ti,tip,bip,tn,bn} \exists_{ct1,cb1,cti,cbi,ctip,cbip,ctn,cbn} >0, \quad q, \lambda \geq 0$$

$$\begin{aligned}
&t1 - P(C[1]) - Q(C[1]) \\
&b1 - R(C[1]) - S(C[1]) \\
&tip - ti.P(C[i + 1]) - Q(C[i + 1]) \\
&bip - bi.R(C[i + 1]) - S(C[i + 1]) \\
&z - k1(C + 1) - t1 \\
&z - ki(iC + 1) - ti \\
&z - kip((i + 1)C + 1) - tip \\
&z - k1'(C + 1) - b1 \\
&z - ki'(iC + 1) - bi \\
&z - kip'((i + 1)C + 1) - bip \\
&t1 + ct1 - C - 1 \\
&b1 + cb1 - C - 1 \\
&ti + cti - iC - 1 \\
&bi + cbi - iC - 1 \\
&tip + ctip - (i + 1)C - 1 \\
&bip + cbip - (i + 1)C - 1 \\
&tn + ctn - nC - 1 \\
&bn + cbn - nC - 1 \\
&kn(nC + 1) + tn - kn'(nC + 1) - bn \\
&n - q - 1
\end{aligned}$$

OR

$$i - \lambda + 1 - n$$

We can change $P(\cdot) = 0$ and $Q(\cdot) = 0$ to $P^2(\cdot) + Q^2(\cdot) = 0$. Here we need to square them to make sure the addition of these polynomials is zero not because one has positive value p and another has $-p$ but because both are equal to zero. We can also change $P(\cdot) = 0$ or $Q(\cdot) = 0$ to $P(\cdot) + Q(\cdot) = 0$

If we combine everything we will have polynomial in the form $\exists \forall \exists P(\cdot)$.

Therefore, this problem is undecidable. \square

2 Nontriviality Problem for Probabilistic Finite Automata

Given PFA $M \exists x$ such that M accepts x with *Probability* ≥ 0.9 versus $\forall x$ such that M accepts x with *Probability* ≤ 0.1

It is undecidable that which statement is true.

2.1 Configuration of 2 counter finite automata

Example. If the finite automata is in state T with counter $C1 = 7$ and counter $C2 = 5$, we may represent the above configuration as \$aaaaaaaTbbbbbb\$. Suppose we increment counter $C2$ by 1 and leave counter $C1$ unchanged. Then, we are in incrementing configuration which may be represent as \$aaaaaaaQbbbbbb\$. Notice that state has changed to Q.

Recall the elimination game in lecture 2, where Team 1 is rooting for a configuration to be good while Team 2 is rooting for a bad configuration.

Definition. A good transition is the one where we can check whether the number of a's in present configuration is one more than the number of a's in previous configuration if it is in incrementing state Q. Similarly one can check for decrementing state or one that does not change.

If the # of a's after transition is correct given the # of a's before was correct, then you have

$$Pr[\text{Good Team gets a point}] = Pr[\text{Bad Team gets a point}]$$

If the # of a's after the transition is incorrect, given the # of a's in previous is correct, then you have

$$Pr[\text{Good Team gets a point}] \leq 0.1 * Pr[\text{Bad Team gets a point}]$$

The accepting sequence of configurations will always consist of all good configurations. So, one needs to make a statement about the probability of each one of those transitions being correct. Therefore, we modify the game as follow:

You go through all transitions and if the good team gets a point every single transition, then they get a point. Similarly, if there exists a bad transition, the probability of the bad team getting a point is much higher than the probability of a good team getting a point. Therefore, it is more likely that bad team will get a point in end.

Example.

t	1	2	3	...	$n-1$	n	
G	0.01	0.01	0.01	...	0.01	0.01	
B	0.01	0.01	0.01	...	0.01	0.01	← good transition
B	0.01	0.1	0.01	...	0.01	0.01	← bad transition

If configuration sequence is good,
G gets point with probability ε
B gets point with probability ε

If configuration sequence is bad,
G gets point with probability $0.1 * \varepsilon$
B gets point with probability ε

Normally PFA does not have any answer. When there is an answer we will give point to the winning team and keep track of the point. We will consider the team which get specific points (e.g. 10 points) will win the game.

3 Characterization for Recursively Enumerable(RE) Sets

3.1 What is in RE Set?

Theorem 3.1. L is RE if and only if $\exists D(< x, y >)$ (decidable predicate) such that $x \in L$ iff $\exists y D(x, y)$.

Example. Let $D(M, i)$ be true iff M halts within i steps. That is $M \in Halt_0$ (TM M which halts on no input) iff $\exists i$ such that $D(M, i)$.

How to represent a decidable predicate? We can write the machine that decides it.

Given M computing $D(x,y)$, does M halts on all inputs.

This is undecidable. What is the indication that it is really undecidable? Even though it is recognizable, we cannot know for certain that it halts on all inputs.

We can represent machine which halts on all input using two quantifiers.

4 Arithmetic Hierarchy

Definition. L is in Π_2 iff there exists decidable predicate D such that $x \in L$ iff $\forall y \exists z D(x, y, z)$.

Definition. L is in Σ_2 iff there exists decidable predicate D such that $x \in L$ iff $\exists y \forall z D(x, y, z)$.

Example. M halts on all inputs iff $\forall x \exists i (M \text{ on } x \text{ halts within } i \text{ steps})$. This language is called **Total** and it is in Π_2 .

Note that negation of Σ_2 is Π_2 .

$$\begin{aligned} & \neg \exists y \forall z D(x, y, z) \\ \iff & \forall y \neg \forall z D(x, y, z) \\ \iff & \forall y \exists z \neg D(x, y, z) \end{aligned} \tag{1}$$

4.1 SuperHalt: Halt with a Halt oracle

Let us imagine we have magic subroutine $H(M)$ which computes $\text{Halt}(M)$. In other word, $H(M)$ decides whether M halts. Consider SuperHalt $TM^+ M$ which can make calls to H .

Let $\text{Halt}^+ = \{TM^+ \langle M \rangle \mid M(\langle M \rangle) \text{ halts.}\}$

In other word, Halt^+ is a set of all TM which halts on its own input. There is no TM^+ which decides it. But Halt^+ can be recognized.

Claim 4.0.1. Halt^+ is undecidable.

Proof of claim. Suppose $\exists_{TM^+} M_{\text{Halt}}$ recognizing Halt^+ .

Now consider

Foo($\langle M \rangle$): Run M_{Halt} on $\langle M \rangle$

If $M_{\text{Halt}}(\langle M \rangle)$ accepts, then loop forever.

Otherwise, it halts.

Now Run Foo($\langle Foo \rangle$).

Therefore, Halt^+ is undecidable even if there is an oracle for the halt problem. Similarly, we can make the same argument for Halt^{++} . \square

4.2 SuperHalt is in Σ_2

Theorem 4.1. Halt^+ is in Σ_2 .

Proof. We are going to show that we can represent $Halt^+$ by $\exists x \forall y, z D(x, y, z)$.
 Given some TM^+M with oracle H , while M was running it makes some queries to the oracle of the form

$H(M_1) = ?$
 $H(M_2) = ?$
 \cdot
 \cdot
 \cdot
 $H(M_t) = ?$

The oracle will answer these questions either *True* or *False*.

M halts iff there exists “Correct Halting Transcript” of M ’s computation.

We will consider the transcript to be correct if

- it is logically consistent, and
- all of the H queries are answered correctly.

We can easily check transitions of $TM M^+$ is logically consistent or not. We also need to check the oracle answers correctly or not. We can use the following way.

- When $H(M_1) = \text{True}$, this means that $TM M_1$ will halt.
 Therefore, $TM M^+$ can verify this by running $TM M_1$ to completion. Note that running M_1 to completion will not change whether M^+ is going to halt or not.
- But when $H(M_1) = \text{False}$, $TM M_1$ will not halt.
 Therefore, $TM M^+$ cannot verify the answer of oracle as it cannot run M_1 to completion. Note that if M^+ runs M_1 and the oracle answered correctly, M^+ will not halt as M_1 will not halt.

Since, there is no way to check whether H answers correctly while it answers “False”, there is no such transcript exists.

We can modify this problem as -

$$M \in Halt^+, \text{ iff } \exists T \forall M, i$$

- T is logically consistent,
- T is halting transcript,
- $H(M_k) = True$ answers are verified by running to completion within i steps , and
- $H(M_k) = False$ answers are verified if M is still running after i steps

We can represent this as $D(T, M, i)$.

This is a decidable predicate in T , M and i , because there is a time bounded running function.

Therefore, $Halt^+$ is in Σ_2 . □

Theorem 4.2. **Total** = $\{M \mid M \text{ halts on all inputs}\}$. **Total** is complete for Π_2 .

Proof. We show this by reducing is general form of Π_2 to **Total**.

Suppose L is Π_2 . Therefore, $x \in L$, iff $\forall y \exists z D(x, y, z)$.

Let x be a TM and $x \in L$ iff for all inputs y there exists time z , x halts on all y within z steps. L is **Total**.

Therefore, **Total** is complete for Π_2 . □