# CS 334 – Hw #7

Yanni Spahi – "I pledge my Honor I have abided by the Stevens Honor System" (+ Vijay Bhamani)

1. Let A = some infinite TM-recognizable language

→ There exists an enumerator $\mathcal{E}$, that enumerates A

→ As we did in class, we'll have $\mathcal{E}$ lexographically run through every input $\Sigma^*$, s.t the TM that recognizes A will check M(1) with 1 transition, M(1) & M(2) with 2 transitions, ... forever printing the lexographic representation of whenever M(i) accepts. (i represents the string. Ex: {1: ε; 2: 0, 3: 1, 4: 00, 5: 01, etc} for $\Sigma = 0,1$)

→ Add each printed string (ordered) to a new language B, ignoring strings already belonging to B. Reorder the printed strings so the first printed string is indexed 1, the second 2, etc.

We first show B is a T-decidable language.

- Construct a Tm M as follows.

M: Given any w

1. If $w \in B$, accept.
2. If $w \notin B$, reject

Thus B is T-decidable. We also have to show B is an infinite subset of A. Notice, $\mathcal{E}$ will never stop printing the language of A. It's infinite. Now at any point in our adding to B, even if we think B is complete, since we know A will continue printing we just have to wait & eventually a new string to add to B will always be printed. Thus B is infinite. Furthermore, since we are continously adding to B from an infinite predefined set A, $B \subset A$.

Thus we find a strict subset B of A which is infinite & decidable.

(Formal proof of $\infty$ is to show bijection from new orderings to $\mathbb{N}$, and that new strings continuosly added)

#2) Show $\{<G>: G$ is a CFG in CNF & $L(G)$ is an infinite language$\}$ is decidable.

First we show any CFG in CNF can be converted to a corresponding directed graph.

We construct such a graph as follows.

First we construct a node for our start nonterminal. Then for each 2-pair combination of non-terminals we construct a node. Finally, we construct a node for each terminal symbol in our CFG + $\varepsilon$.

Now each non-terminal in our CFG is in a combination of 3 forms, $A \to BC$, $A \to \alpha$ or $A \to \varepsilon$ for some $A, B, C, \alpha$.

We define transitions for each node in our graph s.t $A \in$ the node's non-terminal combination.

1. $A \to \alpha$
   → For every node that contains A, make a transition from AX (or XA) to $\alpha$.

2. $A \to \varepsilon$
   → Same as 1 just with $\varepsilon$.

3. $A \to BC$
   → For every node that contains A, make a transition to the node BC.

If a non-terminal A has several optional conversions ie ($A \to AB | BC | \alpha | \varepsilon$), make transitions as shown above for each conversion. Always start at the start symbol, call it S.

Now we've created a graph of of CFG, an example is shown on the following page.

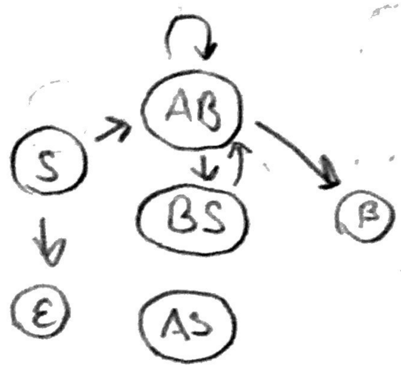
$S \to AB \mid \varepsilon$

$A \to AB \mid BS$

$B \to \beta$

$\Rightarrow$

CFG




Graph

Notice all terminal symbols terminate. This is because a terminal will never add more to the string. Only non-terminals can do that. Also notice, traversing our directed graph is equivalent to starting from our start non-terminal & going through each possible construction path for $L(CFG)$ where every transition to a combination of non-terminals adds an additional nonterminal to our total string. While not outprinting the exact strings of our CFG, we can use this property to find if $L(CFG)$ is infinite. Our graph is in the form of an NFA, thus it is easily converted to a TM.

Now the BIG IDEA is that if our graph of our CFG <u>contains a directed cycle</u>, then it can repeatedly adds a net non-terminal to our string, which will cause its language to be infinite.

We use this property to construct a TM which decides if $L(CFG)$ is infinite. Let M be such a machine.

M: given some CFG in CNF

1. Convert CFG to a directed graph.
2. Perform DFS on our graph, marking non-terminal nodes we transition to. (Don't mark on recursive returns to a node)
3. If we transition to a marked node, our CFG contains a directed cycle & thus an infinite language. Then accept
4. E reject

Since M accepts only $\{ \langle G \rangle : G$ is a CFG in CNF & $L(G)$ is infinite$\}$
& rejects otherwise, we've shown (A) is a Turing decidable language.

∎

---

3. The idea for this proof is straight forward. We can construct a TM to decide on $L = \{ \langle G \rangle : G$ is a CFG over $\{a, b\}$ and $a^* \cap L(G) \neq \phi \}$. Call that TM "M", which we construct as follows.

M: given some CFG w

1. Convert w to be in CNF.
2. If the start non-terminal yields $\varepsilon$, accept.
3. Mark every terminal symbol in w. If we mark a $\beta$, then reject.
4. If $A \to \alpha$ is a rule where every variable in $\alpha$ is marked, ($\alpha$ can be a terminal or 2 non-terminals or several options of the 2) then mark A.
5. Repeat step 3 until no new variable is marked.
6. If the start state is marked accept, else reject.

In other words, we convert G to CNF, reject if it ever uses $\beta$ then recursively eliminate variables we can reach in G. If it can't reach the start variable, we reject. Else we accept. We also accept if the start variable yields $\varepsilon$, since $\varepsilon \in a^*$.

This TM decides on every CFG input into M, thus L is T decidable. ∎