SE 396 Lecture Tue 4/23 Spring 2019

op Hat #
3768

Compare the following two loops, given $G = (V, \Sigma, R, S)$:

```
Set<V> N = ∅   {or init to {A : A → ε is a rule}}
bool changed = true;
while (changed) {
    changed = false;
    for (each A ∈ V \ N) {
        if (A has a rule A → X⃗
            with X⃗ ∈ N*) {
            N = N ∪ {A};
            Changed = true;
        }
    }
}
Output N;
```

```
set<V ∪ Σ> N = Σ
bool changed = true;
while (changed) {
    changed = false;
    for (each A ∈ V \ N) {
        if (A has a rule A → X⃗
            with X⃗ ∈ N*) {
            N = N ∪ {A}
            Changed = true;
        }
    }
}
Output N;
```

laim: Routine is a decider—same reason.

At the end, $S \in N \Leftrightarrow \varepsilon \in L(G)$.

nductively, N becomes $\{A : A \overset{*}{\Rightarrow} \varepsilon\}$. ✓

hus also $\underline{E_{CFG}} = \{G : \varepsilon \in L(G)\}$ is ecidable in $O(|V| \cdot |\langle R \rangle|)$ time.

The while loop must halt within $|V|$ iterations, so this algm is total, ie. a decider. Time $\approx O(|V| \cdot |\langle R \rangle|)$.

Claim: At the end, N equals the set of variables that are "live" in the sense of deriving terminal strings, and $S \in N \Leftrightarrow L(G) \neq \emptyset$.

nce the set N of nullable variables is identified, we can "bypass" -rules by taking every rule $B \to \vec{Y}$ with some number k of nullable variables and making $2^k - 1$ more rules by deleting any subset of their occurrences.

he text does this on-the fly while building N. ither way, this step toward ChNF can take exp! time. But there is a list-based algorithm to do in polynomial $(|V| \cdot |\langle R \rangle|)$ time:

$S \to \varepsilon \mid (\underline{S})\underline{S}$ ← $N = \{S\}$, $k = 2$. Add

$S \to ()S$

$S \to (S)$

$S \to ()$

Then remove $S \to \varepsilon$ to get G'. Then L(G') is L(G) \ {ε}.

A_{CFG}: INST: $\langle G, x\rangle$, ie. a CFG G and an input $x \in \Sigma^*$.
QUES: Is $x \in L(G)$?

Decider: If $x = \varepsilon$, use the decider for "$S \in N$?" Else, convert G to strict Chomsky NF G', so that $x \in L(G) \iff x \in L(G')$. Key fact about G' in ChNF is that if x is derivable at all, it is derivable using $|x|-1$ $\underline{A \to BC}$ kind of rules and $|x|$ $\underline{A \to c}$ kind of rules. Hence with $n = |x|$, we can try all derivations of $2n-1$ steps, and reject if none gives x.

FYI: There is an algorithm called "CKY" or "CYK" that does this in $n^{O(1)}$ time via dynamic programming. Combined with the better conversion to ChNF, this in fact classifies A_{CFG} into polynomial time.

We've seen that E_{CFG}, NE_{CFG}, $\mathcal{E}_{CFG}$, and A_{CFG} are all decidable. How about $ALL_{CFG} = \{\langle G\rangle : L(G) = \Sigma^*\}$? ⚠ ALL_{CFG} is not decidable nor even recognizable.

Can we show that there are undecidable, indeed unrecognizable, languages?

"Diagonalization"

D_{TM}: INST: The code $w = \langle M\rangle$ of a det^c Turing machine M.
QUES: Does M not accept its own code, ie, $w \notin L(M)$?

As a language, $D_{TM} = \{\underbrace{\langle M\rangle}_{q} : \underset{Q}{M} \text{ does not accept } \underset{q}{\langle M\rangle}\}$.

Theorem: There does not exist a TM Q s.t. $L(Q) = D_{TM}$. So D_{TM} is not c-e so undecidable.

Proof: If so, then the "quixotic" TM Q would have a code $q = \langle Q\rangle$. Then

$q \in D_{TM} \iff q \notin L(Q)$ by defn of $q \in D_{TM}$

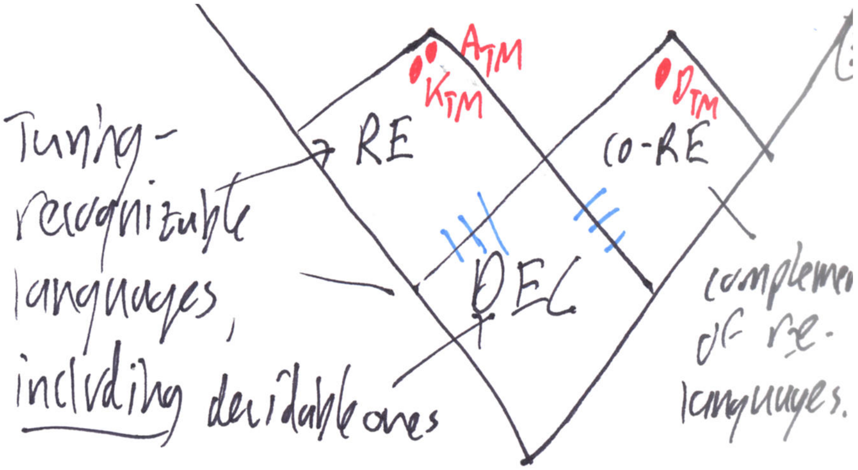
$\iff q \in L(Q)$ by $L(Q) = D_{TM}$.

A logical stmt can never be equivalent to its negation. This is a contradiction. So Q cannot exist. ☒

Conceptual "Landscape"

Convention #1: $\tilde{L}$ is shown in mirror image to L. Hence also

co-RE = $\{\tilde{L} : L$ is recognizable$\}$.



Turing-recognizable languages, including decidable ones

complement of re-languages.

What is the complement of D_{TM}? Essentially (ignoring the issue of invali codes, or lumping them into D

$K_{TM} = \{\langle M\rangle : M$ does accept $\langle M\rangle\}$. Recall that I did define

$A_{TM} = \{\langle M, w\rangle : M$ accepts $w\}$, and showed A_{TM} is re. since it is the language of a univ. TM.

$\boxed{\langle M\rangle \in K_{TM} \Leftrightarrow \langle M, \langle M\rangle\rangle \in A_{TM}.}$

So K_{TM} too is re.

Theorem: L is decidable $\Leftrightarrow$ $\tilde{L}$ is decidable

Proof: If $L = L(M)$ with M total, we can interchange q_{acc} and q_{rej}.

$\therefore$ K_{TM} is (re. but) not decidable, since D_{TM} is undecidable and not even re.

$\therefore$ A_{TM} is likewise (re. but) undecidable.

This finishes Ch 4 coverage — Know as facts for the exam

Added: The diagram also conveys that RE ∩ co-RE = DEC, which says that a language is decidable if (and only if) it is c.e. and co-c.e. If L is decidable then L is automatically c.e., and since $\tilde{L}$ is decidable too, $\tilde{L}$ is c.e., so L is co-c.e. as well. Conversely, if L and $\tilde{L}$ are both c.e., then one can use (non-total) TMs M and M' for them to build a total TM deciding L. Next week we will prove something more general involving reductions to c.e. and co-c.e. languages.