Automata, Computability, and Complexity

Hamza Bouhelal, Mohamed Reda Arsalane, Mahdi Ouchrahou

SHEET #8:

Exercise 1 :

Every production in Chomsky Normal Form either has the form

S → AB, for non-terminals A and B, or the form

S → x, for terminal x

Let n be the length of a string. We start with the (non-terminal) symbol S which has length n = 1.

Using n − 1 rules of form (non−terminal)→(non−terminal)(non−terminal) we can construct a string containing n non-terminal symbols.

Then on each non-terminal symbol of said string of length n we apply a rule of form (non−terminal)→(terminal). i.e. we apply n rules.

In total we will have applied n − 1 + n = 2n − 1 rules.

Exercise 2 :

A language is decidable iff its complement is decidable.
To show that ALL_{DFA} is decidable, we show that its complement, noted $\sim ALL_{DFA}$, is decidable.
We have $\sim ALL_{DFA}$= {<A> | A is a DFA that rejects some word}
We construct a Turing Machine M that decides on $\sim ALL_{DFA}$.

M = on input <A,x>

1) Store A, x appropriately on tapes, if they do not follow specifications, reject.
2) Convert DFA A into corresponding FA A'.
3) Run A' on TM T that decides the E_{FA} problem (Theorem 4.4 of lecture note).
4) If T accepts <A'>, then accept x.
5) If T accepts <A'>, then reject x.

Since we have seen that every nondeterministic finite automaton has an equivalent deterministic finite automaton, i.e. there exists for every FA N a DFA M such that L(M) = L(N). Step 2 is done in a finite set of steps, as we have seen in the lectures.
Also according to the lecture note E_{FA} is decidable, so is stap 3.
We can conclude that M is a decider for ALL_{DFA}, hence ALL_{DFA} is decidable.

<u>Exercise 3:</u>

If a CFG G = (V, Σ, R, S) includes the rule S → ε, then clearly G can generate ε. But G could still generate ε even if it doesn't include the rule S → ε. So it's not sufficient to simply check if G includes the rule S → ε to determine if ε ∈ L(G).

But if we have a CFG G′ = (V ′, Σ, R′ , S′) that is in Chomsky normal form, then G′ generates ε if and only if S ′ → ε is a rule in G′ . Thus, we first convert the CFG G into an equivalent CFG G′ = (V ′ , Σ, R′ , S′) in Chomsky normal form. If S ′ → ε is a rule in G′ , then clearly G′ generates ε, so G also generates ε since L(G) = L(G′). Since G′ is in Chomsky normal form, the only possible ε-rule in G′ is S ′ → ε, so the only way we can have ε ∈ L(G′) is if G′ includes the rule S ′ → ε in R. Hence, if G′ does not include the rule S ′ → ε, then ε ∉ L(G′). Thus, a Turing machine that decides $A\varepsilon^{cfg}$ is as follows:

M = On input <G>, where G is a CFG:

1. Convert G into an equivalent CFG G′ = (V ′ , Σ, R′ , S′) in Chomsky normal form.

2. If G′ includes the rule S ′ → ε, accept. Otherwise, reject.

We conclude that $A\varepsilon^{cfg}$ is decidable.

Exercise 4:

Let $S_{rever\ \ e}$ = { <M> | M is a DFA that accepts w^R whenever it accepts w}. Show that $S_{rever\ \ e}$ is decidable.
To show that $S_{rever\ \ e}$ is decidable, we construct a decider D for $S_{rever\ \ e}$ (with C a TM that decides EQ^{dfa})
F On input <M, w>:

1. Construct an NFA M’ such that L(M’) = { w^R | w ∈ L(M)},
2. Convert M’ into an equivalent DFA M”,
3. Use C to compare L(M”) and L(M), if L(M”)= L(M), accept, else, reject.

In the TM we just described, the conversion of M into M’ can be done as followed:
-Reverse de directions of all transition arrows in M,
-Create a new state q’ in M’, and connect q’ to each original final states of M with ε-transitions,
-Set the final state of M’ to the starting state of M

The conversion of M into M’ can be done in finite steps too, since each NFA has its equivalent DFA that can be found.

The comparison of L(M) and L(M”) can be done in finite steps too, since both DFAs halt.
We conclude that F runs in finite steps and can therefore conclude that $S_{rever\ \ e}$ is decidable.