Lecture L&M 8

Unrestricted Grammars, Undecidability

8.1 Unrestricted grammars

A. Give an unrestricted grammar that generates the language $\{a^i b^j a^i b^j \mid i, j \geq 0\}$.

First construct the context-free language $\{a^i b^j D^{j-1} a^i b \mid i > 0, j > 0\}$, then let the Ds "travel" to the right and convert into a b as soon as they "encounter" a b. The sublanguages $\{a^{2i} \mid i \geq 0\}$ and $\{b^{2j} \mid j \geq 0\}$ are generated separately via the nonterminals A and B.

$$\begin{aligned} S &\to A \mid B \mid aCab \\ A &\to aaA \mid \lambda \\ B &\to bbB \mid \lambda \\ C &\to aCa \mid bD \\ D &\to bDE \mid \lambda \\ Ea &\to aE \\ Eb &\to bb \end{aligned}$$

An example derivation of $aabbbaabbb$:

$$\begin{aligned} S &\Rightarrow aCab \Rightarrow aaCaab \Rightarrow aabDaab \Rightarrow aabbDEaab \Rightarrow aabbbDEEaab \Rightarrow aabbbEEaab \\ &\Rightarrow aabbbEaEab \Rightarrow aabbbEaabb \Rightarrow aabbbaEabb \Rightarrow aabbbaaEbb \Rightarrow aabbbaabbb \end{aligned}$$

B. *(extra)* Give an unrestricted grammar that generates the language $\{a^i b^i c^i d^i \mid i \geq 0\}$.

Somewhat comparable to the previous one. We first generate the context-free language $\{a^{i+1} b D^i cd \mid i \geq 0\}$; the Ds then travel to the right, and produce a b upon encountering a c and continue as E; an E turns into cd upon encountering a d.

$$\begin{array}{llll} S & \to aCcd \mid \lambda & Db & \to bD \\ C & \to aCD \mid b & Dc & \to bcE \end{array} \qquad \begin{array}{ll} Ec & \to cE \\ Ed & \to cdd \end{array}$$

An example derivation:

$$\begin{aligned} S &\Rightarrow aCcd \Rightarrow aaCDcd \Rightarrow aaaCDDcd \Rightarrow aaabDDcd \Rightarrow aaabDbcEd \Rightarrow aaabDbccdd \\ &\Rightarrow aaabbDccdd \Rightarrow aaabbbcEcdd \Rightarrow aaabbbccEdd \Rightarrow aaabbbcccddd \end{aligned}$$

C. Prove that the language generated by the following grammar is $\{a^i b^i c^i \mid i \geq 0\}$:

$$\begin{aligned} S &\to aAbc \mid \lambda \\ A &\to aAbC \mid \lambda \\ Cb &\to bC \\ Cc &\to cc \end{aligned}$$

We first observe that every derivation can be done by

1. rewriting S once. If $S \to \lambda$ was applied here, we're done ($\lambda = a^0b^0c^0$); otherwise the word is $aAbc$
2. applying $A \to aAbC$ i times, then applying $A \to \lambda$ once; then the word is$aa^i(bC)^ibc$
3. applying the rule $Cb \to bC$ a number of times; then the word is $aa^ib^ibC^ic$
4. applying $Cc \to cc$ i times; then the word is $a^{i+1}b^{i+1}c^{i+1}$.

The reason that the rules can always be applied in this order is that they don't affect each other: especially the rules for Cb and Cc can never be applied on overlapping parts of a word at the same time. (Formally: these rules are *locally confluent*.)

8.2 Undecidability

D. The *n-step halting problem* is as follows: Given a Turing machine M and a word w, decide whether M halts after performing at most n transitions when run on input w.

Is this problem decidable?

Yes, it is decidable. We can modify the universal Turing machine to also count the number of transitions executed (e.g. on an extra tape) while simulating M on input w and, once that number reaches $n + 1$, terminate with result "no, does not halt in at most n transitions". This modified universal Turing machine always terminates and decides the n-step halting problem.

E. Let L be a (fixed) language that is recursively enumerable but not recursive. Let M be a (fixed) Turing machine that recognises L. The *halting problem for M* is defined as follows: Given a word w, decide whether M halts on input w.

Prove that this problem is undecidable (for any L and M).

Assume that there is a machine H_M that solves the halting problem for M. We can use H_M and M to construct a machine M' that works as follows, given input w:

1. Run H_M on input w. If the result is "no, does not halt", then halt in a non-accepting state.
2. Otherwise, run M on w (i.e. simulate M, like the universal Turing machine). Because of the previous step, we know that M will halt. If it halts in an accepting state, then let M' also halt in an accepting state; otherwise, let it halt in a non-accepting state.

M' thus decides L. This contradicts that L is not recursive. Thus the machine H_M cannot exist, which means that the halting problem for M is undecidable.

F. Consider the following potential argument for the undecidability of the blank tape problem:

The blank tape problem is a subproblem of the halting problem,
which is undecidable and therefore must be undecidable itself.

Why is this *not* a valid argument?

The argument is essentially a reduction from the blank tape problem to the halting problem. But that is the "wrong direction": a subproblem may be easier than the general problem.

For example, every DFA can also be seen as a Turing machine (that only reads its input once from left to right and that does not modify the tape). The halting problem for DFA-as-Turing-machines is decidable (every DFA halts on every input), yet it is a subproblem of the undecidable general halting problem for Turing machines.

G. Prove that the *101 halting problem* of deciding whether an arbitrary given Turing machine halts when run on input 101 is undecidable. Use a reduction.

Let Q be a Turing machine that solves this problem. We reduce the blank tape problem to it. The input to the blank tape problem is (the encoding of) a Turing machine P. Let R be a third Turing machine that takes such a(n encoding of) P and modifies it to (the encoding of) P', which

1. first reads the full input from the tape, erases it, moves back to the start of the tape, and
2. if the input was 101, behaves like P, otherwise diverges.

Then R transforms (reduces) each input P for the blank tape problem into an input P' for Q such that P halts on a blank tape if and only if Q answers "yes" on input P'. Since the blank tape problem is undecidable, therefore also the 101 halting problem is undecidable.