Theory of Computation 8
Deterministic Membership Testing

Frank Stephan

Department of Computer Science

Department of Mathematics

National University of Singapore

fstephan@comp.nus.edu.sg

Repetition 1: Normal Forms

The normal forms are named after Noam Chomsky (born 1928) and Sheila Greibach (born 1939).

Assume that the language does not contain ε.

Chomsky Normal Form

All rules are of the form $\mathbf{A} \to \mathbf{BC}$ or $\mathbf{A} \to \mathbf{d}$ where $\mathbf{A}, \mathbf{B}, \mathbf{C}$ are non-terminals and $\mathbf{d}$ is a terminal.

Greibach Normal Form

All rules are of the form $\mathbf{A} \to \mathbf{bw}$ where $\mathbf{b}$ is a terminal and $\mathbf{A}$ a non-terminal and $\mathbf{w}$ a (possibly empty) word of non-terminals.

If the language contains ε, one allows in both normal forms $\mathbf{S} \to \varepsilon$ for the start symbol $\mathbf{S}$ which then is not allowed to appear on the right side of a rule.

Repetition 2: Algorithms

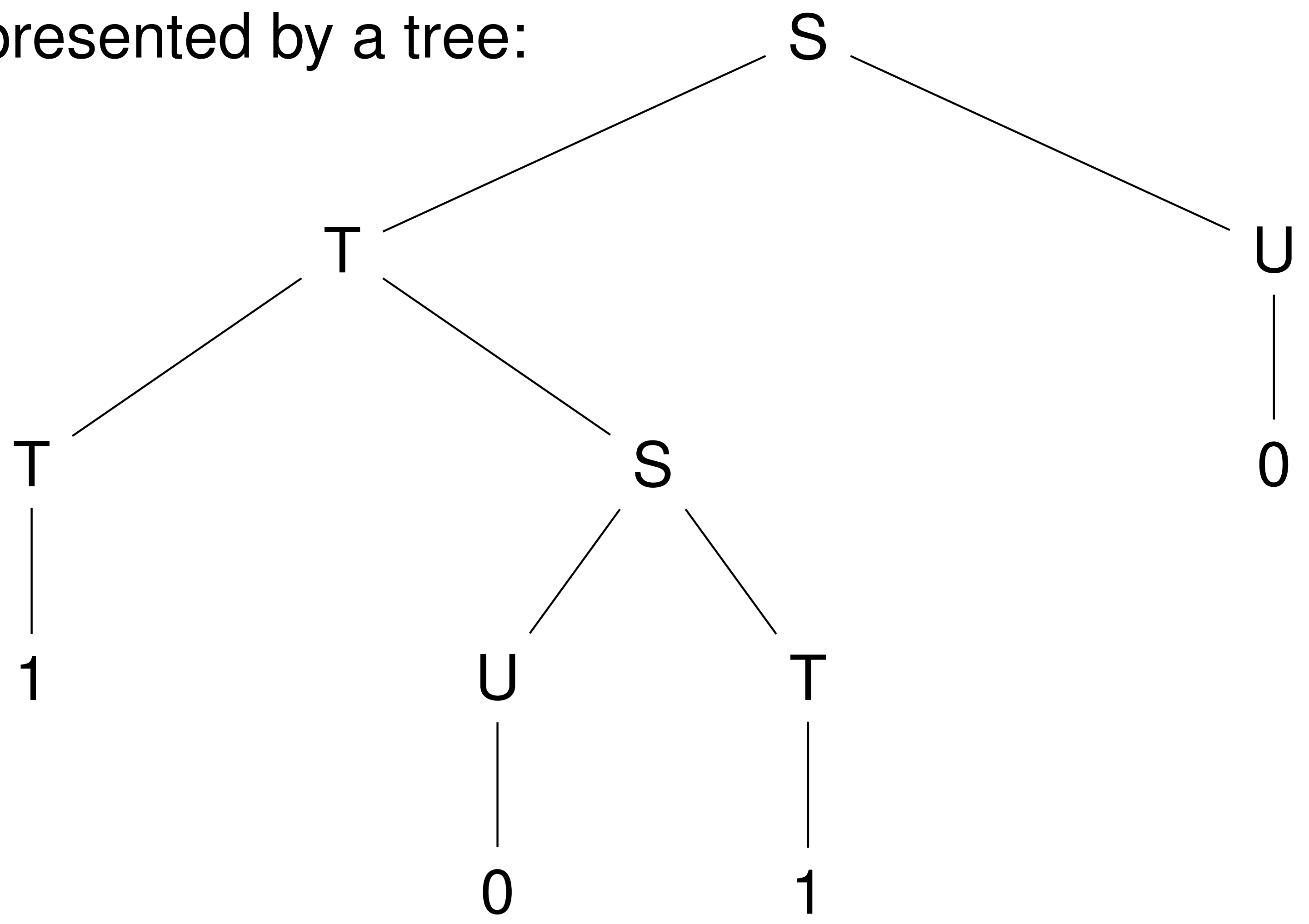
Given a context-free grammar as input, algorithms for the following tasks where presented:

- Converting the grammar into Chomsky Normal Form;
- Testing whether the grammar generates a word;
- Testing whether the grammar generates infinitely many words.

All these algorithms run in time polynomial in the size of the grammar; for grammars in Chomsky Normal Form, the number of non-terminals can serve as the size.

Repetition 3: Derivation Tree

For grammar $(\{\mathbf{S}, \mathbf{T}, \mathbf{U}\}, \{\mathbf{0}, \mathbf{1}\}, \{\mathbf{S} \rightarrow \mathbf{SS}|\mathbf{TU}|\mathbf{UT}, \mathbf{U} \rightarrow \mathbf{0}|\mathbf{US}|\mathbf{SU}, \mathbf{T} \rightarrow \mathbf{1}|\mathbf{TS}|\mathbf{ST}\}, \mathbf{S})$, a derivation $\mathbf{S} \Rightarrow \mathbf{TU} \Rightarrow \mathbf{TSU} \Rightarrow \mathbf{TUTU} \Rightarrow \mathbf{1UTU} \Rightarrow \mathbf{10TU} \Rightarrow \mathbf{101U} \Rightarrow \mathbf{1010}$ can be represented by a tree:



Repetition 4

Marked Symbols

Mark the first four $\mathbf{1}$ in word $\mathbf{0000011111}$ as $\mathbf{0000011111}$. The word can be pumped such that at least one but at most four marked symbols are pumped or between the pumped parts: $\mathbf{000\,0^\ell\,01\,1^\ell\,111}$.

Ogden's Lemma

Let $\mathbf{L} \subseteq \mathbf{\Sigma}^*$ be an infinite context-free language generated by a grammar $(\mathbf{N}, \mathbf{\Sigma}, \mathbf{P}, \mathbf{S})$ in Chomsky Normal Form with $\mathbf{h}$ non-terminals. Then the constant $\mathbf{k} = \mathbf{2^{h+1}}$ satisfies that for every $\mathbf{u} \in \mathbf{L}$ with at least $\mathbf{k}$ marked symbols, there is a representation $\mathbf{vwxyz} = \mathbf{u}$ such that $\mathbf{wxy}$ contains at most $\mathbf{k}$ marked symbols, $\mathbf{wy}$ contains at least $\mathbf{1}$ marked symbol and $\mathbf{vw^\ell xy^\ell z} \in \mathbf{L}$ for all $\ell \in \mathbb{N}$.

Membership Testing

For a language $\mathbf{L}$ and a word $\mathbf{w}$ of length $\mathbf{n}$, one wants to decide whether $\mathbf{w} \in \mathbf{L}$. The following will be shown:

Regular language: Done by a finite automaton, time $\mathbf{O(n)}$.

Linear language: Special case of Cocke, Kasami and Younger's algorithm, time $\mathbf{O(n^2)}$.

Context-free language: Cocke, Kasami and Younger's algorithm, time $\mathbf{O(n^3)}$.

Context-sensitive language: Savitch's algorithm, space $\mathbf{O(n^2)}$, time $\mathbf{O(c^{n^2})}$ for some $\mathbf{c}$.

Recursively enumerable language: No algorithm (undecidable).

Cocke, Kasami and Younger

Let (N, Σ, P, S) be in Chomsky Normal Form and $a_1 a_2 \ldots a_n$ be the input word.

1. Initialisation: For all k,

$$E_{k,k} = \{A \in N : A \to a_k \text{ is a rule}\}.$$

2. Loop: Go through all pairs (i, j) such that they are processed in increasing order of $j - i$ and let

$$E_{i,j} = \{A : \exists \text{ rule } A \to BC\, \exists k \\ [i \leq k < j \text{ and } B \in E_{i,k} \text{ and } C \in E_{k+1,j}]\}.$$

3. Decision: Word is generated by the grammar iff $S \in E_{1,n}$.

Set $E_{i,j}$ contains all non-terminals generating $a_i \ldots a_j$.
Time $O(n^3)$: $O(n^2)$ values $E_{i,j}$ with $O(n)$ choices of k.

Example 8.2, Word 0011

Grammar $(\{S, T, U\}, \{0, 1\}, \{S \rightarrow SS|TU|UT, U \rightarrow 0|US|SU, T \rightarrow 1|TS|ST\}, S)$. Entries $E_{i,j}$:

$$\begin{array}{cccc}
 & & E_{1,4} = \{S\} & \\
 & E_{1,3} = \{U\} & E_{2,4} = \{T\} & \\
E_{1,2} = \emptyset & E_{2,3} = \{S\} & E_{3,4} = \emptyset & \\
E_{1,1} = \{U\} & E_{2,2} = \{U\} & E_{3,3} = \{T\} & E_{4,4} = \{T\} \\
0 & 0 & 1 & 1
\end{array}$$

As $S \in E_{1,4}$, the word 0011 is in the language.

Example 8.2, Word 0111

Grammar $(\{\mathbf{S},\mathbf{T},\mathbf{U}\},\{\mathbf{0},\mathbf{1}\},\{\mathbf{S}\rightarrow\mathbf{SS}|\mathbf{TU}|\mathbf{UT},$
$\mathbf{U}\rightarrow\mathbf{0}|\mathbf{US}|\mathbf{SU},\mathbf{T}\rightarrow\mathbf{1}|\mathbf{TS}|\mathbf{ST}\},\mathbf{S})$. Entries $\mathbf{E_{i,j}}$:

$$\begin{array}{cccccccc}
 & & & E_{1,4}=\emptyset & & & & \\
 & & E_{1,3}=\{T\} & & E_{2,4}=\emptyset & & & \\
 & E_{1,2}=\{S\} & & E_{2,3}=\emptyset & & E_{3,4}=\emptyset & & \\
E_{1,1}=\{U\} & & E_{2,2}=\{T\} & & E_{3,3}=\{T\} & & E_{4,4}=\{T\} & \\
0 & & 1 & & 1 & & 1 &
\end{array}$$

As $\mathbf{S}\notin\mathbf{E_{1,4}}$, the word $\mathbf{0111}$ is not in the language.

Quiz 8.3
Make the table for word $\mathbf{1001}$.

Cocke, Kasami and Younger

Exercise 8.4

Consider the grammar $(\{\mathbf{S}, \mathbf{T}, \mathbf{U}, \mathbf{V}, \mathbf{W}\}, \{\mathbf{0}, \mathbf{1}, \mathbf{2}\}, \mathbf{P}, \mathbf{S})$ with $\mathbf{P}$ consisting of the rules $\mathbf{S} \to \mathbf{TT}$, $\mathbf{T} \to \mathbf{UU}|\mathbf{VV}|\mathbf{WW}$, $\mathbf{U} \to \mathbf{VW}|\mathbf{WV}|\mathbf{VV}|\mathbf{WW}$, $\mathbf{V} \to \mathbf{0}$, $\mathbf{W} \to \mathbf{1}$. Make the entries of the Algorithm of Cocke, Kasami and Younger for the words $\mathbf{0011}$, $\mathbf{1100}$ and $\mathbf{0101}$.

Exercise 8.5

Consider the grammar $(\{\mathbf{S}, \mathbf{T}, \mathbf{U}, \mathbf{V}, \mathbf{W}\}, \{\mathbf{0}, \mathbf{1}, \mathbf{2}\}, \mathbf{P}, \mathbf{S})$ with $\mathbf{P}$ consisting of the rules $\mathbf{S} \to \mathbf{ST}|\mathbf{0}|\mathbf{1}$, $\mathbf{T} \to \mathbf{TU}|\mathbf{1}$, $\mathbf{U} \to \mathbf{UV}|\mathbf{0}$, $\mathbf{V} \to \mathbf{VW}|\mathbf{1}$, $\mathbf{W} \to \mathbf{0}$. Make the entries of the Algorithm of Cocke, Kasami and Younger for the word $\mathbf{001101}$.

Linear Grammars

Linear languages sit between regular and context-free languages.

Definition

A grammar is linear iff on every left side of a rule exactly one non-terminal and on every right side of a rule at most one non-terminal and arbitrary many terminals.

Normal Form

A linear grammar for a language not containing ε is in normal form iff all rules are of one of the following forms: $\mathbf{A \rightarrow Bc}$, $\mathbf{A \rightarrow cB}$, $\mathbf{A \rightarrow c}$ where $\mathbf{A, B}$ are non-terminals and $\mathbf{c}$ is a terminal.
If ε is in the language then one has the rule $\mathbf{S} \rightarrow \varepsilon$ for the start symbol $\mathbf{S}$ and $\mathbf{S}$ does not appear on any right side of a rule.

Parsing Linear Grammars

Let $(\mathbf{N}, \mathbf{\Sigma}, \mathbf{P}, \mathbf{S})$ be a linear grammar in normal form and $\mathbf{a_1 a_2 \ldots a_n}$ be an input word.

1. Initialisation: For all $\mathbf{k}$,

$$\mathbf{E_{k,k}} = \{\mathbf{A} \in \mathbf{N} : \mathbf{A} \to \mathbf{a_k} \text{ is in } \mathbf{P}\}.$$

2. Loop: Process all pairs $(\mathbf{i}, \mathbf{j})$ with $\mathbf{i} < \mathbf{j}$ in increasing order of $\mathbf{j} - \mathbf{i}$ and let

$$\begin{aligned}\mathbf{E_{i,j}} = \{\mathbf{A} : \quad & \exists \text{ rule } \mathbf{A} \to \mathbf{Bc} \quad [\mathbf{B} \in \mathbf{E_{i,j-1}} \text{ and } \mathbf{c} = \mathbf{a_j}] \text{ or} \\ & \exists \text{ rule } \mathbf{A} \to \mathbf{cB} \quad [\mathbf{c} = \mathbf{a_i} \text{ and } \mathbf{B} \in \mathbf{E_{i+1,j}}]\}.\end{aligned}$$

3. Decision: The word is generated by the grammar iff $\mathbf{S} \in \mathbf{E_{1,n}}$.

Example 8.8, Word 0110

The grammar

$$(\{\mathbf{S},\mathbf{T},\mathbf{U}\},\{\mathbf{0},\mathbf{1}\},\{\mathbf{S}\rightarrow\mathbf{0}|\mathbf{1}|\mathbf{0T}|\mathbf{1U},\mathbf{T}\rightarrow\mathbf{S0}|\mathbf{0},\mathbf{U}\rightarrow\mathbf{S1}|\mathbf{1}\},\mathbf{S})$$

is a linear grammar for palindromes. For the word $\mathbf{0110}$, the entries are

$$\begin{array}{cccccccc}
 & & & E_{1,4}=\{S\} & & & & \\
 & & E_{1,3}=\emptyset & & E_{2,4}=\{T\} & & & \\
 & E_{1,2}=\{U\} & & E_{2,3}=\{S,U\} & & E_{3,4}=\{T\} & & \\
E_{1,1}=\{S,T\} & & E_{2,2}=\{S,U\} & & E_{3,3}=\{S,U\} & & E_{4,4}=\{S,T\} & \\
0 & & 1 & & 1 & & 0 &
\end{array}$$

and as $\mathbf{S}\in\mathbf{E_{1,4}}$, the word is accepted.

Example 8.8, Word 1110

For processing the word 1110, one gets the following table:

$$\begin{array}{cccc}
 & & E_{1,4} = \{T\} & \\
 & E_{1,3} = \{S, U\} & E_{2,4} = \{T\} & \\
E_{1,2} = \{S, U\} & E_{2,3} = \{S, U\} & E_{3,4} = \{T\} & \\
E_{1,1} = \{S, U\} & E_{2,2} = \{S, U\} & E_{3,3} = \{S, U\} & E_{4,4} = \{S, T\} \\
1 & 1 & 1 & 0
\end{array}$$

Exercise 8.9

Make the entries for the word 0110110 for above grammar.

Exercises 8.10 and 8.11

Exercise 8.10

Consider the following linear grammar:

$$(\{S, T, U\}, \{0, 1\}, \{S \to 0T|T0|0U|U0,\ T \to 0T00|1,\ U \to 00U0|1\}, S).$$

Convert the grammar into the linear grammar normal form and determine the $E_{i,j}$ for input 00100.

Exercise 8.11

Which two of the following languages are linear? Provide linear grammars for these two languages:

- $L = \{0^n 1^m 2^k : n + k = m\}$;
- $H = \{0^n 1^m 2^k : n + m = k\}$;
- $K = \{0^n 1^m 2^k : n \neq m \text{ or } m \neq k\}$.

Kleene Star

Algorithm 8.12

Let $\mathbf{L}$ be generated by a linear grammar and $\mathbf{a_1 a_2 \ldots a_n}$ be a word. To check whether $\mathbf{a_1 a_2 \ldots a_n \in L^*}$, do the following:

First Part: Compute for each $\mathbf{i, j}$ with $\mathbf{1 \leq i \leq j \leq n}$ the set $\mathbf{E_{i,j}}$ of all non-terminals which generate $\mathbf{a_i a_{i+1} \ldots a_j}$.

Initialise Loop for Kleene Star: Let $\mathbf{F_0 = 1}$.

Loop for Kleene Star: For $\mathbf{m = 1, 2, \ldots, n}$ Do
Begin If there is a $\mathbf{k < m}$ with $\mathbf{S \in E_{k+1,m}}$ and $\mathbf{F_k = 1}$
Then let $\mathbf{F_m = 1}$ Else let $\mathbf{F_m = 0}$ End.

Decision: $\mathbf{w \in L^*}$ iff $\mathbf{F_n = 1}$.

First Part is $\mathbf{O(n^2)}$, Loop for Kleene Star is $\mathbf{O(n^2)}$.

Other Combinations

Assume that $\mathbf{H}, \mathbf{K}, \mathbf{L}$ are linear languages and that one has computed for $\mathbf{a_1 \ldots a_n}$ the entries $\mathbf{E^L_{i,j}}, \mathbf{E^H_{i,j}}, \mathbf{E^K_{i,j}}$ say whether the word $\mathbf{a_i \ldots a_j}$ is in $\mathbf{L}, \mathbf{H}, \mathbf{K}$, respectively. This information can be computed in $\mathbf{O(n^2)}$. Complete the algorithm to do the following check in $\mathbf{O(n^2)}$ time:

Exercise 8.13
Is $\mathbf{a_1 a_2 \ldots a_n} \in \mathbf{L} \cdot \mathbf{H} \cdot \mathbf{K}$?

Exercise 8.14
Is $\mathbf{a_1 a_2 \ldots a_n} \in (\mathbf{L} \cap \mathbf{H})^* \cdot \mathbf{K}$?

Regular Closure of CTF

A language $\mathbf{H}$ is said to be in the regular closure of the context-free languages iff it is obtained by combining finitely many context-free languages with intersection, union, concatenation, set-difference, Kleene Star and Kleene Plus. An example is

$$\mathbf{H} = (\mathbf{L_1} \cap \mathbf{L_2})^* \cdot \mathbf{L_3} \cdot (\mathbf{L_1} \cap \mathbf{L_2})^* - \mathbf{L_4}$$

and here $\mathbf{L_1}, \mathbf{L_2}, \mathbf{L_3}, \mathbf{L_4}$ are context-free.

Exercise 8.15
Prove by structural induction that every language $\mathbf{H}$ which is in the regular closure of the context-free languages has an $\mathbf{O(n^3)}$ decision algorithm.

Counting Derivation Trees

One can modify the algorithm of Cocke, Kasami and Younger to count derivation trees.

Given grammar in CNF $(\mathbf{N}, \mathbf{\Sigma}, \mathbf{P}, \mathbf{S})$, a word $\mathbf{w} = \mathbf{a_1 a_2 \ldots a_n}$ and all $\mathbf{A} \in \mathbf{N}$, let $\mathbf{E_{i,j}}$ denote the set of all nonterminals which generate the characters $\mathbf{a_i \ldots a_j}$ and $\mathbf{D_{i,j,A}}$ denote the number of derivation trees which can, with root $\mathbf{A}$, derive a word $\mathbf{a_i \ldots a_j}$. Let $\mathbf{P}$ be the set of rules. For each $\mathbf{A} \in \mathbf{N}$, if $\mathbf{A} \in \mathbf{E_{i,i}}$, there is exactly one tree with $\mathbf{A} \to \mathbf{a_i}$ and so $\mathbf{D_{i,i,A} = 1}$ else $\mathbf{D_{i,i,A} = 0}$. If $\mathbf{i < j}$ then

$$\mathbf{D_{i,j,A}} = \sum_{\mathbf{(B,C):\ A \to BC \in P}} \ \sum_{\mathbf{k:\ i \leq k < j}} \mathbf{D_{i,k,B} \cdot D_{k+1,j,C}}$$

and the overall number of derivation trees is $\mathbf{D_{1,n,S}}$.

Exercises 8.17 - 8.20

Let $\mathbf{P}$ contain the rules $\mathbf{V} \rightarrow \mathbf{VV}|\mathbf{WW}|\mathbf{0}$ and $\mathbf{W} \rightarrow \mathbf{VW}|\mathbf{WV}|\mathbf{1}$. Consider the grammars $\mathbf{G} = (\{\mathbf{V},\mathbf{W}\},\{\mathbf{0},\mathbf{1}\},\mathbf{P},\mathbf{W})$ and $\mathbf{H} = (\{\mathbf{U},\mathbf{V},\mathbf{W}\},\{\mathbf{0},\mathbf{1},\mathbf{2}\},\mathbf{P}\cup\{\mathbf{U}\rightarrow\mathbf{VU}|\mathbf{UV}|\mathbf{2}\},\mathbf{U})$.

Exercise 8.17
How many derivation trees has $\mathbf{0011100}$ in $\mathbf{G}$?

Exercise 8.18
How many derivation trees has $\mathbf{0000111}$ in $\mathbf{G}$?

Exercise 8.19
How many derivation trees has $\mathbf{021111}$ in $\mathbf{H}$?

Exercise 8.20
How many derivation trees has $\mathbf{010012}$ in $\mathbf{H}$?

Polynomial Space

An algorithm can be measured by

- The time needed to do the computation;
- The space (size of variables and arrays and ...) needed to do the computation.

Let $\mathbf{n}$ be a parameter to measure the size of the input.

If one can be computed in time $\mathbf{F(n)}$ then, under certain assumptions to the machine model, it can also be done in space $\mathbf{F(n)}$.

However, for the converse, only a rough estimate is known: If something can be computed in space $\mathbf{F(n)}$ then one can compute it in time $\mathbf{2^{F(n)}}$.

Theorem of Savitch

Algorithm 8.21
Context-senstive grammar (N, Σ, P, S), input word w.

Recursive Call: Function $Check(u, v, t)$

```
    Begin If $u = v$ or $u \Rightarrow v$ Then Return$(1)$;
    If $t \leq 1$ and $u \neq v$ and $u \not\Rightarrow v$ Then Return$(0)$;
    Let $t' = t/2$; Let $r' = 0$;
    For all $u' \in (N \cup \Sigma)^*$ with $|u| \leq |u'| \leq |v|$ Do
    Begin If $Check(u, u', t') = 1$ and $Check(u', v, t') = 1$
    Then $r' = 1$ End; Return$(r')$ End.
```

Decision: If $Check(S, w, k^n) = 1$ Then $w \in L$ Else $w \notin L$.

Space Complexity, per call $O(n)$, in total $O(n^2)$;
Value of t: $k^n/2^h$ in depth h of recursion $(k = |\Sigma| + |N| + 1)$;
Number of nested calls: $O(\log(k^n)) = O(\log(k) \cdot n)$.

Runtime: $O(c^{n^2})$ for any $c > (2k)^{\log(k)}$.

Example

Check whether $\mathbf{S} \Rightarrow^* \mathbf{w}$ within $\mathbf{8}$ steps. One call $\mathbf{Check}(\mathbf{S}, \mathbf{w}, \mathbf{8})$.

Is there a word $\mathbf{v}$ such that $\mathbf{S} \Rightarrow^* \mathbf{v}$ and $\mathbf{v} \Rightarrow^* \mathbf{w}$ both within $\mathbf{4}$ steps? For each $\mathbf{v}$, two calls $\mathbf{Check}(\alpha, \beta, \mathbf{4})$ one after each other with $(\alpha, \beta) = (\mathbf{S}, \mathbf{v}), (\mathbf{v}, \mathbf{w})$.

Is there a word $\mathbf{v}'$ such that $\alpha \Rightarrow^* \mathbf{v}'$ and $\mathbf{v}' \Rightarrow^* \beta$ both within $\mathbf{2}$ steps? For each word $\mathbf{v}'$, two calls $\mathbf{Check}(\alpha', \beta', \mathbf{2})$ one after each other with $(\alpha', \beta') = (\alpha, \mathbf{v}'), (\mathbf{v}', \beta')$.

Is there a word $\mathbf{v}''$ such that $\alpha' \Rightarrow^* \mathbf{v}''$ and $\mathbf{v}'' \Rightarrow^* \beta'$ both within $\mathbf{1}$ steps? For each word $\mathbf{v}''$, two calls $\mathbf{Check}(\alpha'', \beta'', \mathbf{1})$ one after each other with $(\alpha'', \beta'') = (\alpha', \mathbf{v}''), (\mathbf{v}'', \beta')$.

Is $\alpha'' = \beta''$ or $\alpha'' \Rightarrow \beta''$ true? No further call needed, bottom of recursion reached.

Example 8.22

Consider grammar $(\{S, T, U, V, W\}, \{0, 1\}, P, S)$ with rules P being the following:

$S \to 0S|U$, $U \to V|0$, $0V \to 1U$, $V \to 1$, $1V \to WU$, $1W \to W0$, $0W \to 10$.

This gives a binary counter with $S \to 0S$ generating enough digits before doing $S \to U$. U stands for last digit 0, V stands for last digit 1, W stands for a digit 0 still having a carry bit to pass on.

Deriving a binary number k needs at least k steps; as the length n of k is logarithmic in k, the derivation length can be exponential in n. In particular deriving 1^n needs more than 2^n steps.

Exercises 8.23 and 8.24

Exercise 8.23

Give a proof that there are k^n or less words of length up to n over the alphabet $\Sigma \cup N$ with $k - 1$ symbols.

Exercise 8.24

Modify Savitch's Algorithm such that it computes the length of the shortest derivation of a word w in the context-sensitive grammar, provided that such derivation exists. If it does not exist, the algorithm should return the special value ∞.

Naive Algorithm

Exercise 8.25
What is the time and space complexity of this naive algorithm?

Recursive Call: Function $\mathbf{Check}(\mathbf{u}, \mathbf{w}, \mathbf{t})$

```
    Begin If $\mathbf{u} = \mathbf{w}$ or $\mathbf{u} \Rightarrow \mathbf{w}$ Then Return$(\mathbf{1})$;
    If $\mathbf{t} \leq \mathbf{1}$ and $\mathbf{u} \neq \mathbf{v}$ and $\mathbf{u} \not\Rightarrow \mathbf{w}$ Then Return$(\mathbf{0})$;
    Let $\mathbf{r}' = \mathbf{0}$;
    For all $\mathbf{v} \in (\mathbf{N} \cup \mathbf{\Sigma})^*$ with $\mathbf{u} \Rightarrow \mathbf{v}$ and $|\mathbf{v}| \leq |\mathbf{w}|$ Do
    Begin If $\mathbf{Check}(\mathbf{v}, \mathbf{w}, \mathbf{t} - \mathbf{1}) = \mathbf{1}$ Then $\mathbf{r}' = \mathbf{1}$ End;
    Return$(\mathbf{r}')$ End;
```

Decision: If $\mathbf{Check}(\mathbf{S}, \mathbf{w}, \mathbf{k}^{\mathbf{n}}) = \mathbf{1}$ Then $\mathbf{w} \in \mathbf{L}$ Else $\mathbf{w} \notin \mathbf{L}$.

Growing Grammars

Definition [Dahlhaus and Warmuth 1986]
A grammar $(\mathbf{N}, \Sigma, \mathbf{P}, \mathbf{S})$ is growing context-sensitive iff $|\mathbf{l}| < |\mathbf{r}|$ for all rules $\mathbf{l} \to \mathbf{r}$ in the grammar.

Theorem [Dahlhaus and Warmuth 1986]
Given a growing context-senstive grammar there is a polynomial time algorithm which decides membership of the language generated by this growing grammar.

In this result, polynomial time means here only with respect to the words in the language, the dependence on the size of the grammar is not polynomial time. So if one asks the uniform decision problem for an input consisting of a pair of a grammar and a word, no polynomial time algorithm is known for this problem. As the problem is NP-complete, the algorithm is unlikely to exist.

Example 8.28

Consider the grammar

$$(\{S, T, U\}, \{0, 1\}, \{S \to 011|T11, T \to T0U|00U, U0 \to 0UU, U1 \to 111\}, S)$$

which is growing. This grammar has derivations like $S \Rightarrow T11 \Rightarrow 00U11 \Rightarrow 001111$ and $S \Rightarrow T11 \Rightarrow T0U11 \Rightarrow 00U0U11 \Rightarrow 00U01111 \Rightarrow 000UU1111 \Rightarrow 000U111111 \Rightarrow 00011111111$. The language of the grammar is

$$\{0^n 1^{2^n} : n > 0\} = \{011, 001111, 0^3 1^8, 0^4 1^{16}, 0^5 1^{32}, \ldots\}$$

and not context-free, as infinite languages satisfying the context-free pumping lemma can only have constant gaps (sequence of lengths without a word). This grammar has growing gaps.

Exercises 8.29 - 8.31

Exercise 8.29

Show that every context-free language is the union of a language generated by a growing grammar and a language containing only words up to length 1.

Exercise 8.30

Modify the proof of Theorem 6.11 to prove that every recursively enumerable language, that is, every language generated by some grammar is the homomorphic image of a language generated by a growing context-sensitive grammar.

Exercise 8.31

Construct a growing grammar for the language $\{1^{2^n}0^{2n}1^{2^n} : n > 0\}$ which is the “palindromisation” of the language from Example 8.28.