A NOTE ON INFINITE TREES

Joost ENGELFRIET
Technical University Twente, The Netherlands

Received 15 August 1972

1. Introduction

An infinite object can be specified by a finite set of rules, which generate the object. Example: an infinite subset of $\Sigma^*$ can be specified by a context-free grammar: the set $a^n b c^n; n \geq 0$ is generated by the context-free rules $A \rightarrow aA$ and $A \rightarrow b$. It is of course desirable to know whether two object-specifications specify the same object or not. For context-free grammars this is well-known to be an undecidable question. It is however decidable in the case of regular grammars.

In this note we consider specifications of objects which are infinite not because of the number of their elements but because of their size, viz., infinite sequences of symbols (infinite length) and infinite labeled trees (infinite depth). We show that it is decidable whether two "regular" sequence (tree) specifications specify the same sequence (tree).

2. Notation

$\Sigma$ is a finite alphabet (set of symbols). Elements: $a, b, c, \ldots \in \Sigma^*$ is the set of words (finite sequences of symbols from $\Sigma$) and contains the empty word $e$. Elements: $\alpha, \beta, \gamma, \ldots$ the set $\{\alpha\}$ is identified with the element $\alpha$. A subset $t$ of $\Sigma^*$ is prefix-closed if for all $\alpha, \beta \in \Sigma^*: \alpha \beta \in t \Rightarrow \alpha \in t$.

A regular grammar $G = (N, \Sigma, S, R)$ over $\Sigma$ consists of a finite set $N$ of non-terminals, a start non-terminal $S \in N$ and a finite set $R$ of rules each of which has one of the forms $A \rightarrow \alpha B$ or $A \rightarrow \alpha$ (\(A, B \in N \text{ and } \alpha \in \Sigma^*\)). If $\nu, w \in \Sigma^*((N \cup \&)$ then \(\nu \Rightarrow w\) means that $w$ can be obtained from $\nu$ by the application of one rule, and $\nu \Rightarrow^* w$ the same, but by application of zero or more rules. The language generated by $G$ is $L(G) = \{\alpha \in \Sigma^*; S \Rightarrow^* \alpha\}$.

3. Infinite sequences

Consider an infinite sequence of symbols from $\Sigma$ such as: $abaababaababa \ldots$ This sequence can (intuitively) be generated by the two rules $S \rightarrow abA$ and $A \rightarrow aS$: consider these rules as a regular grammar and start generating

$S \Rightarrow abA \Rightarrow cbaS \Rightarrow abaabA \Rightarrow \ldots$

(the language generated by this grammar is of course empty). In order to be able to apply the methods known from the specification of subsets of $\Sigma^*$ by grammars, we code (or define) a sequence as a subset of $\Sigma^*$:

\textbf{Definition 1.} A sequence $t$ over $\Sigma$ is a non-empty prefix-closed subset of $\Sigma^*$ such that

\[ \forall \alpha \in \Sigma^* [t \cap \alpha \cdot \Sigma^* = \emptyset \Leftrightarrow \exists a \in \Sigma^* : \\
\alpha \cdot \Sigma^* \subseteq t \cap \alpha \cdot \Sigma^*] \]

The above infinite sequence is defined to be the infinite subset \(\{e, a, ab, aba, aba, ababa, ababa, \ldots\}\) of $\Sigma^*$. It should be clear that there is a one-to-one correspondence between "intuitive" sequences and sequences as defined in def. 1.
**Definition 2:** A (regular) sequence specification $SS$ is a 4-tuple $(N, \Sigma, S, R)$ where

- $N$ is a finite set of non-terminals,
- $S \in N$ is the start non-terminal,
- $R$ is a mapping $N \rightarrow \Sigma^* N$.

Instead of $R(A) = \alpha B$ we write $(A \rightarrow \alpha B) \in R$. The method of generating the infinite sequence intuitively has been shown above. Formally:

**Definition 3:** The infinite sequence $t_{SS}$ specified by $SS$ is the language generated by the regular grammar $G_{SS} = (N, \Sigma, S, R')$ with $R'$ defined by:

\[
i \cdot (A \rightarrow a_1 a_2 \ldots a_n B) \in R \quad (A, B \in N, a_i \in \Sigma, n \geq 0),
\]

then $A \rightarrow e, A \rightarrow a_1, A \rightarrow a_1 a_2, \ldots, A \rightarrow a_1 a_2 \ldots a_{n-1} \quad \text{and} \quad A \rightarrow a_1 a_2 \ldots a_n B$ are in $R'$.

So $t_{SS} = L(G_{SS})$. It should of course be checked that $t_{SS}$ is a sequence. The $SS$ with rule $S \rightarrow e, S \rightarrow a$, $S \rightarrow a B, A \rightarrow e, A \rightarrow a S$. This grammar generates the language $e \cup a(ba)^* \{e, b, ba\}$, i.e. the (intuitive) sequence $abaabaaba \ldots$ It should be clear that $G_{SS}$ generates formally the sequence which $SS$ specifies intuitively.

As a corollary to our definitions we obtain:

It is decidable whether two (regular) sequence specifications specify the same infinite sequence. (This is a direct consequence of the fact that equivalence of regular grammars is decidable.)

Remark: The sentences which can be specified by an $SS$ are easily seen to be all ultimately periodic sequences.

---

**4. Infinite trees**

The trees which we will consider here are of the "selector type" (cf. [1]), i.e. the branches of the tree are labeled with elements of $\Sigma$ (selectors) such that branches which leave the same node are labeled with different selectors (and there is no order between the branches). See fig. 1. The infinite tree of fig. 1b can intuitively be generated by the two rules of fig. 2 in the way shown in fig. 3 (replace a non-terminal by the right-hand side of the corresponding rule). In the same way as a sequence specification looks like a regular grammar, a tree specification looks like a regular tree grammar (cf. [1]).

Again, we define a tree as a subset of $\Sigma^*$:

**Definition 4:** A tree $t$ over $\Sigma$ is a non-empty prefix-closed subset of $\Sigma^*$. The tree of fig. 1b is defined as the infinite prefix-closed set $\{e, a, b, aa, ac, aaab, aaaca, \ldots\}$. An the tree of fig. 1a is $\{e, a, b, c, aa, ac\}$. There is again a one-to-one correspondence between intuitive trees and formal trees (each element of the prefix-closed set indicates a possible path in the tree). A sequence is a special case of a tree (compare def.'s 1, 4). Fig. 1c gives a picture of the sequence $abc$ as a tree.

Before giving the definition of a tree-specification we need to know what a tree with non-terminals is.

The set $T_N$ of finite trees with non-terminals from $N$ is defined as a subset of $[\Sigma \cup N \cup \{(,\},,\}]^*$ by

1) $* \in T_N$ (the tree with one node and no branches);
2) If $A \in N$, then $A \in T_N$;
3) If $t_1, t_2, \ldots, t_k \in T_N$ (for $k \geq 1$) and $\{a_1, a_2, \ldots, a_k\} \subset \Sigma$ with $i \neq j \rightarrow a_i \neq a_j$, then $a_1 \cdot t_1 \cdot a_2 \cdot t_2 \cdots a_k \cdot t_k \in T_N$ (this is the tree with root from which $k$ branches leave: the $i$th branch is labeled $a_i$ and leads to the subtree $t_i$).

---

**Fig. 1.**

**Fig. 2.**
J. Engelfriet. A note on infinite trees

\[ \begin{align*}
\text{Definition 5: A (regular) tree specification TS is a} & \quad 4\text{-tuple } (N, \Sigma, S, R) \text{ where } N, \Sigma, S \text{ are the usual things} \\
& \quad \text{and } R \text{ is a mapping } N \rightarrow T_N. \text{ If } R(A) = t \text{ then we write} \\
& \quad (A \rightarrow t) \in R. \\
\text{The example of fig. 2 is now formally the TS with} & \quad \text{rules } S \rightarrow a(A) b(*) \text{ and } A \rightarrow a(S) c(*). \\
\text{Definition 6: Let } t \in T_N, \text{ then } [t] \subset \Sigma^* (N \cup e) \text{ is} & \quad \text{defined by:} \\
& \quad 1) \ [e] = e; \\
& \quad 2) \ [A] = A \text{ for } A \in N; \\
& \quad 3) \ [a_1(t_1) a_2(t_2) \ldots a_k(t_k)] = e \cup [a_1[t_1] \cup a_2[t_2] \cup \ldots \\
& \quad \quad \quad \cup a_k[t_k]. \\
\text{Example: } [a(A) b(*)] = e \cup a [A] \cup b[\] & \quad \text{is } e \cup a A \cup b e = \{e, a, A, b\}. \\
\text{Definition 7: The infinite tree } t_{\text{TS}} \text{ specified by TS} & \quad \text{is the language generated by the regular grammar} \\
& \quad G_{\text{TS}} = (N, \Sigma, S, R') \text{ with } R' \text{ defined by} \\
& \quad \text{if } (A \rightarrow t) \in R \text{ then } (A \rightarrow w) \in R' \text{ for all } w \in [t]. \\
\text{In the } \text{example of fig. 2 the corresponding } G_{\text{TS}} & \quad \text{has rules } S \rightarrow e, S \rightarrow b, S \rightarrow aA, A \rightarrow e, A \rightarrow c, A \rightarrow aS. \\
\text{This grammar generates the set } (aA)^* \{e, a, b, ac\}, & \quad \text{which is the tree of fig. 1b. It is left to the reader to} \\
\text{show that the tree which is intuitively generated by a} & \quad \text{TS (using replacement of non-terminals) corresponds} \\
\text{to the formal tree generated by } G_{\text{TS}}. \\
\text{Again as a corollary to our definitions we obtain} & \quad \text{Theorem: It is decidable whether two (regular)} \\
\text{tree specifications specify the same infinite tree.} & \quad \text{The TS with one rule } A \rightarrow a(a(A) c(*)) b(*) \text{ has a}
\end{align*} \]

\[ G_{\text{TS}} \text{ with rules } A \rightarrow a, A \rightarrow b, A \rightarrow c, A \rightarrow a, A \rightarrow aA, \text{ and} \\
A \rightarrow aaA. \text{ This grammar generates } (aa)^* \{e, a, b, ac\} \text{ which is the tree of fig. 1b.} \]

5. Examples

\( a \) \ The meta-grammar of ALGOL-68 produces also certain “infinite sequences” (especially from the non-terminal MODE). In these “sequences” infinities occur in the middle. In fact it is better to consider these sequences as infinite trees. The infinite MODE’s which can occur in a program are always of “regular” type, because of the nature of a mode-declaration.

Consider the mode cell defined by the mode-declaration mode cell = struct (ref cell next, int item). Take \( \Sigma = \{S, r, i\} \) where the letters have obvious meanings, and \( N = \{\text{cell}\} \). Let the only rule of the TS be cell \( \rightarrow S(r(\text{cell})) S(i(*)) \). This TS specifies the tree \( (S, r)^* \{e, S, rS, rS, S\} \) which is a representation of the mode cell. The mode ref cell is then represented by the tree \( e \cup r(S, r)^* \{e, S, rS, S\} \).

Let us now look at the mode-declaration

\[ \text{mode thing = ref struct (thing next, int item).} \]

The corresponding TS with rule thing \( \rightarrow r(S, r(\text{thing})) S(i(*)) \) specifies the tree \( (rS)^* \{e, r, rS, rS, i\} \), which is easily seen to be the same tree as that representing ref cell. Therefore, the modes ref cell and thing are equal. From these considerations it follows that it is decidable whether two modes (declared by evtl. recursive mode-declarations) are the same (cf. \([4, 6, 7]\)).

\( b \) A program with loops is in fact a method of specifying an infinite program without loops. The program (with test p and transformations f, g) of

\[ \begin{align*}
\text{Fig. 4.} \\
\text{Fig. 5.} \\
\end{align*} \]
fig. 4a specifies the infinite tree generated by the TS with the rule from fig. 4b (where $\bar{p}$ stands for not-$p$).

From the theorem it follows that it is decidable whether two programs specify the same infinite tree. For example, the program (with corresponding TS) of fig. 5 gives the same tree as that in fig. 4, viz. $(p \lor \bar{e})^*$ (e, fi, $\bar{g}$, $\bar{p}$). This decidability is well known from the literature (cf. [2, 3, 8] where this is worked out further).

**Remark:** The other way around, a TS could have been defined as a directed graph with branches labeled with elements of $\Sigma$. The graph corresponding to the TS of fig. 2 is shown in fig. 6 (cf. also [9]).

6. Conclusion

If we can code (infinite) objects as subsets of some $\Sigma^*$ in such a way that (regular) object-specifications are coded as regular grammars, then it is decidable whether two object-specifications specify the same object. This is especially true for infinite trees labeled with elements from $\Sigma$.

**References**