

Lecture 16
ECE 664

PROPERTIES OF PHRASE-STRUCTURE GRAMMARS and The LANGUAGES GENERATED BY THEM

Avi KAK

- What is the difference between a process and a phrase structure grammar?
- We say a Turing Machine accepts a language, whereas a grammar generates a language.
- What are the main differences between the alphabet of a Turing Machine, the alphabet of a process, and the alphabet of a grammar?

Observation about grammars: Let the alphabet of a phrase structure grammar Γ be

$$\{ \underbrace{s_1, s_2, \dots, s_n}_{\text{Terminals } T}, \underbrace{v_1, \dots, v_k}_{\text{Variables } V} \}$$

We will consider the alphabet of Γ ordered as shown. Obviously, the strings on this alphabet are notations for integers in base $n+k$.

• The predicate $u \xRightarrow{\Gamma} v$ is primitive recursive. and therefore guaranteed to be computable

PROOF: Let the productions of Γ be $g_i \rightarrow h_i$ for $i = 1, 2, \dots, l$. For each production, we construct the following primitive recursive function:

$$\text{PROD}_i(u, v) \iff (\exists r, s)_{\leq u} [u = \text{CONCAT}(r, g_i, s) \ \& \ v = \text{CONCAT}(r, h_i, s)]$$

\uparrow for the i th production \rightarrow bounded quantifiers are primitive recursive \leftarrow There are primitive recursive

We can talk about r and s as being bounded by u because the strings can be thought of as ~~integers~~ notations of integers in base $n+k$.
Now we can say

$$u \xRightarrow{\Gamma} v \iff \text{PROD}_1(u, v) \vee \text{PROD}_2(u, v) \vee \dots \vee \text{PROD}_l(u, v)$$

DEFINITION The predicate $\text{DERIV}(u, y)$ with $u \in T^*$ means that there exists a derivation

$$S = u_1 \xRightarrow{\Gamma} u_2 \xRightarrow{\Gamma} u_3 \dots \xRightarrow{\Gamma} u_m = u$$

so that $y = [u_1, u_2, \dots, u_m, 1]$

• $\text{DERIV}(u, y)$ is primitive recursive.

PROOF: $\text{DERIV}(u, y) \iff (\exists m)_{\leq y} \left(\begin{aligned} & m+1 = \text{lt}(y) \ \& \ (y)_i = n+1 \\ & \& \ (y)_m = u \ \& \ (y)_{m+1} = 1 \\ & \& \ (\forall j)_{< m} \{ j=0 \vee [(y)_j \xRightarrow{\Gamma} (y)_{j+1}] \} \end{aligned} \right)$

Gödel number \leftarrow $\prod_{l=1}^m u_l \cdot p_{n+1}$
 $p_1=2 \ p_2=3 \ \dots$
value of S is $n+1$ in base $n+k$

\leftarrow to take care of the situation when $u = u_m = 0$

is therefore guaranteed to be computable

• We can obviously write $S \xRightarrow{\Gamma}^* u \iff (\exists y) \text{DERIV}(u, y)$ unbounded quantifier
 or, equivalently, $S \xRightarrow{\Gamma}^* u \iff \min_y \text{DERIV}(u, y) \downarrow$ unbounded minimization

• Therefore $\{u \mid S \xRightarrow{\Gamma}^* u\}$ is r.e. only partially computable

• Since $L(\Gamma) = T^* \cap \{u \mid S \xRightarrow{\Gamma}^* u\}$, $L(\Gamma)$ is r.e. (T^* is recursive, and therefore, r.e.)

• Therefore, the language generated by a grammar is r.e.

- Now let's consider a language L . The only thing we know about this language is that it is r.e.
- By lecture 13, for every r.e. L there exists a nondeterministic Turing Machine that accepts L .
- By lecture 14, for every nondeterministic Turing Machine, there exists a semi-Thue process that accepts the same language. And by Lecture 15, there is guaranteed to be a phrase structure grammar that will ~~be~~ generate this language.
- Combining the conclusion at the bottom of the previous page with the above conclusion, we get:

A language U is r.e. iff there is a grammar T' such that $U = L(T')$

CONTEXT-SENSITIVE GRAMMARS

- A grammar is called context-sensitive if for each production $g_i \rightarrow h_i$ we have $|g_i| \leq |h_i|$.
- If T' is context sensitive, then the ~~productive~~ set

is recursive.

$$\{u \mid S \xrightarrow{T'}^* u\}$$

Compare to the case of a phrase-structure grammar where the same set is r.e.

PROOF:

Consider the derivation

$$S = u_1 \Rightarrow u_2 \Rightarrow u_3 \dots \Rightarrow u_m = u$$

Since $|S|=1$, we have

$$1 = |u_1| \leq |u_2| \leq \dots \leq |u_m| = |u|$$

It must therefore be the case that, considered as notations for integers in base $n+k$,

$$u_1, u_2, u_3, \dots, u_m \leq g(u)$$

where $g(u)$ is the smallest number which represents a string of length $|u|+1$ in base $n+k$:

$$g(u) = \sum_{i=0}^{|u|} (n+k)^i$$

Since when considered as a base $n+k$ integer, u_i cannot be larger than u_{i+1} , the length m of the derivation is bounded by the total number of strings of length $\leq |u|$. But this number is $g(u)$. Therefore, the Gödel number

$$[u_1, \dots, u_m, 1] = \prod_{i=1}^m p_i^{u_i} \cdot p_{m+1} \leq h(u) = \prod_{i=1}^{g(u)} p_i^{g(u)} \cdot p_{g(u)+1}$$

So we can write

$$S \xrightarrow{T'}^* u \iff (\exists y)_{\leq h(u)} \text{ DERIV}(u, y)$$

bounded minimization

FAMOUS UNSOLVABLE PROBLEMS

- | | | |
|---|---|---|
| <p>① The Halting Problem is unsolvable</p> <p>② Given a Turing Machine with alphabet A and $L \subseteq A^*$ as the language accepted by the Turing Machine. Given a word $u \in A^*$, determining whether $u \in L$ is unsolvable.</p> | <p>③ Given a semi-Thue process Π with alphabet A and given two words $u, v \in A^*$, the problem of determining whether $u \xrightarrow{\Pi}^* v$ is unsolvable.</p> <p>④ Given L as the language generated by a phrase structure grammar T whose alphabet A contains the set of terminals T. That is $L = \{u \in T^* \mid S \xrightarrow{T}^* u\}$. Determining whether a word $u \in T^*$ belongs to L is unsolvable.</p> | <p>⑤ For a phrase structure grammar T', determining $L(T') = \emptyset$ or $L(T')$ is infinite is unsolvable.</p> <p>⑥ Given a pair of grammars Δ and T', determining whether</p> <ol style="list-style-type: none"> 1. $L(\Delta) \subseteq L(T')$ 2. $L(\Delta) = L(T')$ <p>are unsolvable.</p> |
|---|---|---|