Unrestricted Grammars

Type 0 or unrestricted grammars are the most general class of grammars in the Chomsky hierarchy. This grammar can generate arbitrary recursively enumerable languages.

An unrestricted grammar G is a four-tuple, $G = (\Sigma, N, S, P)$ with the symbols having their usual meanings, and P is a finite set of productions of the form $\alpha \to \beta$, for any $\alpha, \beta \in (N \cup \Sigma)^*$, with α containing at least one non terminal symbol. As usual, $L(G) = \left\{ w \in \Sigma^* \mid S \overset{*}{\underset{G}{\to}} w \right\}$.

Example 1 The following grammar generates the language $L_1 = \{a^{2^n} \mid n \geq 0\}$:

$$\begin{aligned} S &\to TaU \\ U &\to AU \mid \varepsilon \\ aA &\to Aaa \\ TA &\to T \\ T &\to \varepsilon \end{aligned}$$

A derivation of the string a^8 using the above grammar is as follows:

$$\begin{aligned} S &\to TaU \to TaAU \to TaAAU \to TaAAAU \\ &\to TAaaAAU \to TAaAaaAU \to TAAaaaaAU \\ &\to TAAaaaAaaU \to TAAaaAaaaaU \to TAAaAaaaaaaU \\ &\to TAAAaaaaaaaaU \overset{*}{\to} aaaaaaaa \end{aligned}$$

□

Example 2 The following grammar generates the language $L_2 = \{a^n b^n c^n \mid n \geq 0\}$.

$$\begin{aligned} S &\to UT \\ U &\to aUbC \mid \varepsilon \\ Cb &\to bC \\ CT &\to Tc \\ T &\to \varepsilon \end{aligned}$$

For example, the string $a^2b^2c^2$ can be derived as:

$$\begin{aligned} S &\to UT \to aUbCT \to aaUbCbCT \to aabCbCT \to aabbCCT \\ &\to aabbCTc \to aabbTcc \to aabbcc \end{aligned}$$

□

We will now show, that the class of languages generated by Type 0 grammars are exactly the Turing recognisable (recursively enumerable) languages.

Theorem X.1 *If $L = L(G)$ for some unrestricted grammar G, then L is r.e.*

Proof: Let us construct a four-tape nondeterministic Turing machine to recognise L.

We can non-deterministically simulate the grammar G and try to derive the string x. If we successfully do so, we accept. The first tape contains the input (and is never changed), the second tape contains the sentential form in the derivation process. The third and fourth tape are used to store α and β for the production $\alpha \to \beta$ being used.

M on input x first writes S on the second tape, then loops:

- Non-deterministically choose a position on the second tape.
- Non-deterministically choose a production rule $\alpha \to \beta$.
- Write α to tape 3 and β to tape 4.
- Compare tape 2 and tape 3 starting from the chosen position in tape 2.
- If the comparison succeeds, replace α with β in tape 2, shifting the contents if required.
- Compare tape 1 and tape 2, if they are identical then accept. $\square$

Theorem X.2 *If L is r.e., then $L = L(G)$ for some unrestricted grammar G.*

Proof: Let M be a DTM recognising L. WLOG, assume that M uses a right end marker ($\dashv$) as its rightmost non-blank symbol on its tape throughout its working. Also, modify M so that it erases the tape contents when it accepts a string, so that the final accepting configuration is always $\vdash \mathrm{t}' \dashv$, where t' is the new accepting state.

G will simulate the working of the Turing machine in reverse. If the following represents the sequence of configurations reached by the TM to accept x:

$$\mathrm{s} \vdash x \dashv \quad \longrightarrow \quad ... \quad \longrightarrow \quad C_i \quad \longrightarrow \quad C_j \quad \longrightarrow \quad ... \quad \longrightarrow \quad \vdash \mathrm{t}' \dashv$$

Then, the grammar will derive x by the following derivation:

$$S \quad \longrightarrow \quad \vdash \mathrm{t}' \dashv \quad \longrightarrow \quad ... \quad \longrightarrow \quad C_j \quad \longrightarrow \quad C_i \quad \longrightarrow \quad ... \quad \longrightarrow \quad \mathrm{s} \vdash x \dashv \quad \longrightarrow \quad x$$

M could have transitioned from configuration C_i to C_j by a transition:

- $\delta(\mathrm{p}, a) = (\mathrm{q}, b, L)$

$$\cdots x\mathrm{p}ay\cdots \quad \longrightarrow \quad \cdots \mathrm{q}xby\cdots$$

- or, $\delta(\mathrm{p}, a) = (\mathrm{q}, b, R)$

$$\cdots x\mathrm{p}ay\cdots \quad \longrightarrow \quad \cdots xb\mathrm{q}y\cdots$$

So, we can define our grammar as $G = (\Sigma, V = Q \cup (\Gamma \setminus \Sigma) \cup \{S\}, S, P)$ with production rules

$$\begin{aligned} S \quad &\to \quad \vdash t' \dashv \\ s \vdash \quad &\to \quad \varepsilon \\ \vdash \quad &\to \quad \varepsilon \end{aligned}$$

For every transition $\delta(\mathrm{p}, a) = (\mathrm{q}, b, L)$, $\forall x \in \Gamma$,

$$\mathrm{q}xb \to x\mathrm{p}a$$

For every transition $\delta(\mathrm{p}, a) = (\mathrm{q}, b, R)$,

$$b\mathrm{q} \to \mathrm{p}a$$

The equivalence of languages can be shown by induction. $\square$