Chapter 5

Appendix

5.1 Algorithm for Recognition Process of a Grammar System

For a given grammar, i.e. a set of rewriting rules, there are some algorithm to decide whether a given word can be accepted by the grammar, if the grammar belongs to the class of context free [46, 86]. In this appendix we introduce one of such algorithms which used in our language game.

The basic problem in rewriting process is the undeterminisity about more than two applicable rules and positions to rewrite. In the derivation process, we introduced randomness to adopt a rule from plural applicable rules and the leftmost derivation. But in the recognition process we must determine without ambiguity whether a given word is involved in a language of a given grammar. To this end we introduce a marker to decide a position to apply a rule and test all possible sub-search-trees until the word is rewritten back to the initial symbol.

At first, we transform each rule $(\alpha \to \beta)$ of a given grammar G to a corresponding reverse rule with a marker which is denoted by $\#$,

$$\beta\# \to \alpha\# \ . \tag{5.1}$$

Second, add two rules to move the marker at the end of rule list,

$$\#0 \to 0\# \ , \ \ \#1 \to 1\# \ . \tag{5.2}$$

A grammar with marker $G^{\#}$ is constructed by these transformation rules for the grammar G. Third, the marker is attached to the head of a given word w tested whether it is involved in the language of the grammar G,

$$\#w \ . \tag{5.3}$$

If the given word w is acceptable by the grammar G or not is attributed to whether the rules in the marked grammar $G^{\#}$ can rewrite the marked word $\#w$ to the target word,

$$S\# \ . \tag{5.4}$$

Note that there is no undeterminisity about the position to apply a rule in this grammar $G^{\#}$, since only one marker $\#$ can appear in a word.

Each agent have its rules in the list form. A rewriting rule which can rewrite the word are searched from the top of the list. If there is an applicable rule, the marked word is rewritten by the rule. No more applicable rules expect for the rules to move the marker can be found, the marker is moved to right by using one of the rules to move the marker. Once a word have been rewritten, then the rule search process is restart from the top of the list. The rewriting process finishes if the word is written as $S\#$ and we know that the given word w can be accepted by the given grammar G. If there is no applicable rule even if the word does not coincide with the target word $S\#$, we put the written word back one rewriting step, and the search process is restarted from the next rule of the last application. This recursive process is continued until there is no rule which is not tried. We know the given grammar G can not accept the given word w, if the written words have never accord with the target word in spite that all rules have tried. It is when the rewritten word become $w\#$.

We show some examples. For a grammar G with rule set,

$$S \to 0,\ S \to A1,\ A \to 10,\ A \to 1S,\ B \to A,\ S \to 10,$$

the corresponding grammar with marker $G^{\#}$ consists of the following rules:

$$0\# \to S\#,\ A1\# \to S\#,\ 10\# \to A\#,\ 1S\# \to A\#,$$
$$A\# \to B\#,\ 10\# \to S\#,\ \#0 \to 0\#,\ \#1 \to 1\#\ .$$

A word 101 is rewritten as the following process:

$$\#101 \Rightarrow 1\#01 \Rightarrow 10\#1 \Rightarrow 1S\#1 \Rightarrow A\#1 \Rightarrow A1\# \Rightarrow S\#\ .$$

Since it can be rewritten to the target word $S\#$, the word 101 is acceptable by the grammar G.

Next example is a rewriting process with putting back. A word 10 is rewritten as following processes:

$$\#10 \Rightarrow 1\#0 \Rightarrow 10\# \Rightarrow A\# \Rightarrow B\# \Rightarrow \times$$

Since the word $B\#$ cannot be rewritten any more, denoted by $\Rightarrow \times$, we put back the word one step to the word $A\#$. And then we restart the search process from the next of the last applied rule in the list. Namely, we begin to search the fitting rule from $10\# \to S\#$, since the word $A\#$ is rewritten by the rule $A\# \to B\#$.

$$A\# \Rightarrow \times$$

But there is not any rule in the remainders of the list. We put back again and the search process is restarted from the next rule of the rule which has been applied to rewrite the word $10\#$ to $A\#$, i.e. $1S\# \to A\#$, then the rewriting process is restarted as,

$$10\# \Rightarrow S\#.$$

Because we can attain to the target word, it is made sure that the grammar G accepts the word 10.

A failure process is displayed as a last example. The rewriting process of the word 00 proceeds as following:

$$\begin{aligned} \#00 \Rightarrow \quad 0\#0 \Rightarrow \quad & S\#0 \Rightarrow S0\# \Rightarrow \times \\ & S\#0 \Rightarrow \times \\ 0\#0 \Rightarrow \quad & 00\# \Rightarrow \times. \end{aligned}$$