Modelling Computation of Languages
Proofs of Claims

This supplemental document — which is not required reading, but is provided for interested students — provides proofs of various claims that were stated in the notes for Lecture #8.

Phrase Structure Grammars

Theorem 4. *Let $L \subseteq \Sigma^\star$ such that L is the language of a nondeterministic Turing machine*

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\textit{accept}}, q_{\textit{reject}}).$$

Then there exists a phrase-structure grammar G such that $L = L(G)$.

Proof. Suppose first that $q_0 = q_{\text{accept}}$. Then $L(M) = \Sigma^\star$. Then it suffices to set Π to include a rewrite rule

$$S \to \sigma S$$

for all $\sigma \in \Sigma$, as well as a rule

$$S \to \lambda$$

in order to ensure that $L(G) = L(M) = \Sigma^\star$.

From now on we will consider the more challenging (and interesting) case that $q_0 \neq q_{\text{accept}}$.

Let h, S and z be symbols that are not in $Q \cup \Gamma$. Consider the string rewriting system $\Omega(M)$ — recalling, by Theorem #3, that

$$hq_{\text{accept}}h \Rightarrow^\star_{\Omega(M)} hq_0 \sqcup \omega h$$

if and only if $\omega \in L$, for all $\omega \in \Sigma^\star$.

Now consider a phrase-structure grammar

$$G = (V, \Sigma, \Pi, S)$$

that is defined as follows.

- The alphabet Σ for G is the same as the input alphabet Σ for M. Thus $L(G) \subseteq \Sigma^\star$ and $L = L(M) \subseteq \Sigma^\star$ for the same alphabet Σ.
- $V = (Q \cup \Gamma \cup \{h, S, z\}) \setminus \Sigma$.

  Note that since $Q \cup \Gamma \cup \{h\} \subseteq V \cup \Sigma$, all the rewrite rules in $\Omega(M)$ can (and will) be included in Π.
- The new symbol S will be used as the start variable for G.
- Rewrite rules in Π are as follows.
  (a) $S \to hq_{\text{accept}}h$
  (b) $hq_0\sqcup \to z$
  (c) $z\sigma \to \sigma z$ for every symbol $\sigma \in \Sigma$
  (d) $zh \to \lambda$
  (e) every rewrite rule in $\Omega(M)$

Suppose that $\omega \in L = L(M)$, so that

$$hq_{\text{accept}}h \Rightarrow^\star_{\Omega(M)} hq_0 \sqcup \omega h$$

Then

$$\begin{aligned}
S &\Rightarrow_\Pi hq_{\text{accept}}h && \text{(using the rewrite rule } S \to hq_{\text{accept}}h\text{)}\\
&\Rightarrow^\star_\Pi hq_0 \sqcup \omega h && \text{(because } \Omega(M) \subseteq \Pi\text{)}\\
&\Rightarrow_\Pi z\omega h && \text{(using the rewrite rule } hq_0\sqcup \to z\text{)}\\
&\Rightarrow^\star_\Pi \omega z h && \text{(using rewrite rules } z\sigma \to \sigma z \text{ for } \sigma \in \Sigma\text{)}\\
&\Rightarrow_\Pi \omega && \text{(using the rewrite rule } zh \to \lambda\text{).}
\end{aligned}$$

It follows that $L \subseteq L(G)$.

Recall that a ***Post word*** is a string of the form

$$h\mu_1 q \mu_2 h$$

where $\mu_1, \mu_2 \in \Gamma^\star$ and $q \in Q$. Notice that $hq_{\text{accept}}h$ is a Post word.

It is reasonably easy to prove — by induction on the length of a derivation — that if ν is a Post word, $\zeta \in (V \cup \Sigma)^\star$, and $\nu \Rightarrow^\star_{\Omega(\mathcal{M})} \zeta$ then ζ is a Post word too.

Suppose now that $\omega \in L(G)$, so that $S \Rightarrow^\star_\Pi \omega$. Consider a derivation of ω from S — and the intermediate strings in $(V \cup \Sigma)^\star$ that are derived along the way. In this case the derivation of ω from S must begin with an application of the rewrite rule

$$S \to hq_{\text{accept}}h$$

because this is the only rewrite rule with S on the left hand side. Thus

$$S \Rightarrow_{\Pi} hq_{\text{accept}}h \Rightarrow_{\Pi}^{\star} \omega.$$

Note that $hq_{\text{accept}}h$ is a Post word. As noted above, applications of rewrite rules in $\Omega(M)$ (to Post words) produce Post words as well. Since ω is not a Post word (because it does not begin with h), some rewrite rule in Π that is *not* in $\Omega(M)$ must eventually be applied to derive ω.

Consider the ***first*** such rewrite rule that is applied when deriving ω. This rewrite rule cannot be any of

(a) $S \rightarrow hq_{\text{accept}}h$

(c) $z\sigma \rightarrow \sigma z$ for a symbol $\sigma \in \Sigma$

(d) $zh \rightarrow \lambda$

(e) a rewrite rule in $\Omega(M)$

because the string that has been derived, at this point does not include either S or z, and because the rewrite rule being considered is not in $\Omega(M)$.

Thus this rewrite rule must be

(b) $hq_0\sqcup \rightarrow z$.

It follows that if $h\mu_1 q\mu_2 h$ is the Post word that was derived immediately before this (with $\mu_1, \mu_2 \in \Gamma^{\star}$ and $q \in Q$) then $\mu_1 = \lambda$, $q = q_0$, and $\mu_2 = \sqcup\zeta$ for some string $\zeta \in \Gamma^{\star}$.

The derivation of ω therefore has the form

$$S \Rightarrow_{\Pi} hq_{\text{accept}}h \Rightarrow_{\Omega(M)}^{\star} hq_0 \sqcup \zeta h \Rightarrow_{\Pi} z\zeta h \Rightarrow_{\Pi}^{\star} \omega.$$

Let ℓ be the length of the string ζ. It is easy to prove the following by induction on ℓ:

- At least ℓ more rewrite rules must be used after the string $z\zeta h$ has been derived— and each of these is one of the rewrite rules

  $$z\sigma \rightarrow \sigma z$$

  for $\sigma \in \Sigma$.

- $\zeta \in \Sigma^{\star}$ and the application of the rewrite rules mentioned above produces a derivation

  $$z\zeta h \Rightarrow_{\Pi}^{\star} \zeta z h.$$

Our derivation now has the form

$$S \Rightarrow_{\Pi} hq_{\text{accept}}h \Rightarrow_{\Omega(M)}^{\star} hq_0 \sqcup \zeta h \Rightarrow_{\Pi} z\zeta h \Rightarrow_{\Pi}^{\star} \zeta z h \Rightarrow_{\Pi}^{\star} \omega$$

Once again, though, it can be checked that — since $\zeta \in \Sigma^\star$ — the only rewrite rule that can be applied to the string $\zeta z h$ is the rewrite rule

$$zh \to \lambda$$

and no rewrite rule can be applied after that. Thus

$$\zeta z h \Rightarrow_\Pi \zeta = \omega$$

It follows from the above that

$$h q_{\text{accept}} h \Rightarrow^\star_{\Omega(M)} h q_0 \sqcup \omega h$$

and that $\omega \in L(M) = L$.

Since ω was arbitrarily chosen from $L(G)$ it now follows that $L(G) \subseteq L$ — as needed to complete the proof that $L(G) = L$, and to establish the claim. □

Markov Algorithms

Theorem 5. *Let Σ be an alphabet.*

(a) A partial or total function $f : \Sigma^\star \to \Sigma^\star$ is Markov-computable if and only if f is Turing-computable.

(b) A language $L \subseteq \Sigma^\star$ is Markov-recognizable if and only if L is Turing-recognizable.

(c) A language $L \subseteq \Sigma^\star$ is Markov-decidable if and only if L is Turing-decidable.

Proof. It is a reasonably straightforward ***exercise*** to take a Markov algorithm, with an input alphabet Σ, with some language $L \subseteq \Sigma^\star$, and design a deterministic Turing machine, M, with the same input alphabet Σ, which satisfies the following properties for every string $\omega \in \Sigma^\star$:

- If the Markov algorithm accepts ω then M accepts ω as well.
- If the Markov algorithm rejects ω then M rejects ω too.
- If the Markov algorithm loops on ω then M also loops on ω.

Similarly, a Markov algorithm that computes a partial or total function $f : \Sigma^\star \to \Sigma^\star$ can be used to produce a Turing machine that computes the same function.

In order to establish a converse (of the claim that this would prove), consider a ***deterministic*** Turing machine

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}}).$$

We may assume without loss of generality that $q_0 = q_{\text{accept}}$, since $L(M) = \Sigma^\star$ and it is easy to replace M with another Turing machine, with input alphabet Σ, that decides this language, and whose start state and accept states are different.

Consider a Markov algorithm, with "input alphabet" Σ, whose substitution formulas include strings over a larger alphabet

$$\widehat{\Gamma} = Q \cup \Gamma \cup \{h, \ell, r, \mathtt{T}\}$$

— where the states in Q should be renamed, as needed, to ensure that $Q \cap \Gamma = \emptyset$ and $h, \ell, r, T \notin Q$.

Suppose that the following substitution formulas are included in this Markov algorithm's scheme.

(i) The ***last*** of the substitution formula is as follows:

$$\lambda \to r$$

Every other substitution formula in this scheme will have a string, one the left, that includes at least one of the symbol "ℓ", "r", or the symbol for one of the states in Q. Since the input string ω does not include any of these symbols this substitution formula will be the first (and only) substitution formula can be applied at the beginning of the algorithm's computation. If the input is a string $\omega \in \Sigma^\star$ then the computation begins with

$$\omega \Rightarrow r\omega$$

(ii) ***Immediately above*** this bottom substitution formula, the scheme should include a substitution formula

$$r\sigma \to \sigma r$$

for every symbol $\sigma \in \Sigma$ — with one more substitution formula

$$r \to \ell h$$

appearing between these and the substitution rule (shown above) at the end.

Every other substitution formula in the scheme will have a string, on the left, that at includes either the symbol "ℓ" or the symbol for one of the states in Q. Suppose, now, that

$$\omega = \tau_1\tau_2 \dots \tau_n \in \Sigma^\star, \tag{1}$$

where $n \geq 0$ and $\tau_1, \tau_2, \dots, \tau_n \in \Sigma$ — so that the sting obtained, so far, is

$$r\tau_1\tau_2 \dots \tau_n$$

One can show, by induction on i, that if $0 \leq i \leq n$ then, after another i rules have been applied, the string that has been obtained is

$$\tau_1\tau_2 \dots \tau_i r \tau_{i+1}\tau_{i+2} \dots \tau_n$$

Thus, if $i \leq n-1$ then the first substitution formula that can be applied is

$$r\tau_{i+1} \to \tau_{i+1} r$$

which results in the string

$$r\tau_1\tau_2 \ldots \tau_{i+1} r \tau_{i+2}\tau_{i+3} \ldots \tau_n$$

On the other hand, if $i = n$, then the first substitution formula that can be applied is

$$r \rightarrow \ell h$$

and the application of this produces the string

$$\tau_1\tau_2 \ldots \tau_n \ell h$$

(iii) ***Immediately above*** the substitution formulas that have now been described, the scheme should include a substitution formula

$$\sigma\ell \rightarrow \ell\sigma$$

for every symbol $\sigma \in \Sigma$ — with one more substitution formula

$$\ell \rightarrow hq_0\sqcup$$

appearing between these and the substitution rules that have already been described.

Continuing the computation — when ω is as shown at line (1) — one can show, by induction on i, that if $0 \leq i \leq n$ then, after i of the above rules, the string that has been obtained is

$$\tau_1\tau_2 \ldots \tau_{n-i} \ell \tau_{n-i+1}\tau_{n-i+2} \ldots \tau_n h$$

In particular, after n of these rules have been applied, the string is

$$\ell\tau_1\tau_2 \ldots \tau_n h$$

At this point, the first substitution rule (in the listing) that can be applied is the last substitution rule in this group,

$$\ell \rightarrow hq_0\sqcup$$

The string produced, after applying this rule, is

$$hq_0 \sqcup \tau_1\tau_2 \ldots \tau_n h$$

— the ***Post word*** corresponding to M's initial configuration on the input string ω.

(iv) ***Immediately above*** the substitution rules that have been introduced so far, include all the rewrite rules in the String Rewriting System $\Sigma(M)$, described in the lecture notes. Since the string that has now been obtained is the Post word for the initial configuration for M, one can establish the following (which slightly the results of Lemma #2, from the lecture notes, but which follows by the same argument):

- If M accepts ω then, after a finite number of additional applications of substitution formulas, the string

$$hq_{\text{accept}}h$$

  is obtained.
- If M accepts ω then, after a finite number of additional applications of substitution formulas, a Post word for a ***rejecting*** configuration of M is obtained. None of the substitution formulas that have been provided, so far, can be applied at this point.
- If M accepts ω then the application of substitution rules never ends — that is, there is aways a substitution rule that can be applied.

(v) Finally, let us add the *final* substitution formula

$$hq_{\text{accept}}h \to \cdot\, \mathtt{T}$$

at the beginning of the Markov algorithm's scheme.

A consideration of the above confirms that the following properties are satisfied, for every string $\omega \in \Sigma^\star$:

- If M accepts ω then this Markov algorithm accepts ω.
- If M rejects ω then this Markov algorithm rejects ω.
- If M loops ω then this Markov algorithm loops on ω too.

Thus the Markov algorithm has the same language, $L \subseteq \Sigma^\star$ — and the Markov algorithm decides L if and only if M does, as needed to establish parts (b) and (c) of the claim.

A proof of part (a) can be completed by beginning with a Turing machine

$$M = (Q, \Sigma, \Sigma, \Gamma, \delta, q_0, q_{\text{halt}})$$

which computes some partial or total function $f : \Sigma^\star \to \Sigma^\star$, and by using a Markov algorithm to compute the same function. A Markov algorithm resembling the one above — with a slightly larger alphabet $\widehat{\Gamma}$, and the substitution formulas described in parts (i), (ii) and (iii), above, can be used for this. Recall, as described in the lecture notes

$$\Sigma(M) = \Pi_1 \cup \Pi_2$$

for a pair of sets of production rules, Π_1 and Π_2. Only the rewrite rules in Π_1 should be included in the Markov algorithm's scheme, so that the desired output is not erased by the application of substitution formulas — and so that the application of rules leads to a Post word

$$hq_{\text{halt}} \sqcup \zeta h$$

if the function f is defined on the input, and the function's value is the string $\zeta \in \Sigma^\star$.

Suppose we introduce another two new symbols, $\widehat{\ell}$ and $\widehat{r}$, into the Markov algorithm's alphabet $\widehat{\Gamma}$. Then the substitution formula shown in part (v), above, can be replaced by a small collection of substitution formulas, including

$$q_{\mathsf{halt}} \to \widehat{r}$$

to begin a "cleanup" process, along with a substitution formula

$$h q_{\mathsf{halt}} \widehat{\ell} \to \cdot \lambda$$

to end it. The completion of the Markov algorithm (by adding more substitution formulas involving the new symbols $\widehat{r}$ and $\widehat{\ell}$, and the completion of the proof of part (a) of the claim, is left as an exercise. □