Properties of Context-Free Grammar

The Pumping Lemma for CFL

Parse tree

考虑一个 CNF 文法 $G=(V,T,P,S)$ 的 parse tree，且其 yield 为 w，若最长路径长度为 n，则有 $|w|\leqslant 2^{n-1}$

> Proof. 根据 n 归纳即可
>
> Basis. $n=1$，最长路径长为 1 的 parse tree 只有一个 root 和一个标记为 terminal 的叶节点，故 $|w|=1$，而 $2^{n-1}=2^0=1,|w|\leqslant 2^{n-1}$
>
> Induction. 考虑最长路径为 n，则根节点一定使用了产生式 $A\to BC$，则任意子树中最长路径都不超过 $n-1$，根据 I. H.，两颗子树的 yield 长度均不超过 2^{n-2}，则整棵树的 yield 长度不超过 2^{n-1}

Pumping Lemma

令 L 为一个 CFL，则存在常数 n，对于任意属于 L 的 string z，若 $|z|\geqslant n$，则可以将 z 写作 $uvwxy$，并且满足

- $|vwx|\leqslant n$
- $|vx|>0$
- 对任意 $i\geqslant 0$，有 $uv^iwx^iy\in L$

> Proof.
>
> 考虑 L 的 CNF 文法 $G=(V,T,P,S)$，满足 $L-\{\epsilon\}=L(G)$，则假设 G 中有 m 个 variable，令 $n=2^m$，令 z 为 L 中长度至少为 n 的 string，根据上文的结论，任何最长路径为 m 的 parse tree 其 yield 长度不超过 $2^{m-1}=n/2$，则 yield 为 z 的 parse tree 最长路径至少为 $m+1$
>
> 考虑最长路径，设其长度为 $k+1(k\geqslant m)$，则其路径上出现了 $k+1$ 个 variable，记为 $A_0,A_1,\dots,A_k$，而 G 中有 m 个不同的 variable，根据 PHP，路径上至少有两个 variable 是相同的，考虑路径上的最后 $m+1$ 个 variable，即 A_{k-m} 到 A_k，设 $A_i=A_j,k-m\leqslant i<j\leqslant k$

> 则可以划分 parse tree 的 yield，令 w 为子树 A_j 的 yield，令 vwx 为子树 A_i 的 yield，令 $uvwxy$ 为整棵树的 yield。由于 CNF 没有 unit production，故 v, x 不同时为 ϵ
>
> 因此可以构造新的 parse tree，如使用 A_j 的子树代替 A_i的子树，则 yield 为 uwy，对应 uv^iwx^iy 中 $i = 0$ 的情况。或是可以用 A_i 的子树代替 A_j 的子树，即 uv^2wx^2y，同理可以得到 $uv^3wx^3y, uv^4wx^4y, \dots$
>
> 同时由于选择的是最后 $m + 1$ 个 variable，即 $k - m \leqslant i \leqslant k$，则 A_i 的子树中最长路径不会超过 $m + 1$，则其产生的 string $|vwx| \leqslant 2^m = n$

Pumping Lemma 可以用于证明某 language 不是 CFL

Closure Properties of CFL

Substitutions

考虑 alphabet Σ，为其中每个元素 a 选择一个语言 L_a，这个语言可以是定义在任意 alphabet 上的任意语言，则可以定义一个函数 $L_a = s(a)$。对于 string $w = a_1a_2 \dots a_n$

$$s(w) = \{x_1x_2 \dots x_n : x_i \in s(a_i)\}$$

即 $s(w)$ 是一系列 language $s(a_1), s(a_2), \dots, s(a_n)$ 的 concatenation，则对一个语言 L

$$s(L) = \bigcup_{w \in L} s(w)$$

则有：若 L 是一个定义在 Σ 的 CFL，且对于每个 $a \in \Sigma$，有 $s(a)$ 为 CFL，则 $s(L)$ 是 CFL

> Proof. 基本思路为根据 L 的 CFG，将其中每个 terminal a 替换为 $s(a)$ 的 CFG 的开始符号
>
> 设 L 的 CFG 为 $G = (V, T, P, S)$，而 $s(a)$ 的 CFG 为 $G_a = (V_a, T_a, P_a, S_a)$，且为了便于讨论，设 V 以及任意 V_a 的元素不重名。
>
> 则可以为 $s(L)$ 构造一个新的 CFG $G' = (V', T', P', S)$，其中
>
> - V' 是 V 与所有 V_a 的 union
> - T' 是所有 T_a 的 union
> - P' 包含
>   - 所有 P_a 的产生式
>   - 所有 P 的产生式，但将所有 terminal a 用 S_a 代替

则只需证明 $w \in L(G') \iff w \in s(L)$

($\Leftarrow$): 考虑 $w \in s(L)$，则设 $x = a_1 a_2 \dots a_n \in L$，且 $x_i \in s(a_i)$，则有 $w = x_1 x_2 \dots x_n$，则原本从 G 推导出的 $a_1 a_2 \dots a_n$ 在 G' 中将推导出 $S_{a_1} S_{a_2} \dots S_{a_n}$。由于 G' 包含所有 G_a 中的产生式，则 $S_{a_i} \overset{*}{\Rightarrow} x_i$ 同样是 G' 中的推导，故有 $S \overset{*}{\Rightarrow} x_1 x_2 \dots x_n = w$，即 $w \in L(G')$

($\Rightarrow$): 若 $w \in L(G')$，则在推导过程中一定有 $S \overset{*}{\Rightarrow} S_{a_1} S_{a_2} \dots S_{a_n}$，因为在推导的开始将只使用 G 中的产生式，直至推导出某个 S_a，而在 S_a 的子树中将只使用 G_a 中的产生式，因此根据该推导过程的 parse tree，可以得到一个字符串 $a_1 a_2 \dots a_n \in L(G)$，且 $x_i \in s(a_i)$，满足

- $w = x_1 x_2 \dots x_n$
- 在 parse tree 中删除某些子树，其 yield 为 $S_{a_i} S_{a_2} \dots S_{a_n}$

由于 $x_i \in s(a_i)$，故易得 $w \in s(L)$

Applications of the Substitution Theorem

根据 substitution 的封闭性，可以得出 CFL 对于下列操作是封闭的

- union
- concatenation
- closure (*) and positive closure ($^+$)
- homomorphism

Proof. 只需构造特定的 substitution 即可

- union：对于 CFL L_1, L_2，$L_1 \cup L_2 = s(L)$，其中 $L = \{1, 2\}, s(1) = L_1, s(2) = L_2$
- concatenation：对于 CFL L_1, L_2，$L_1 \cup L_2 = s(L)$，其中 $L = \{12\}, s(1) = L_1, s(2) = L_2$
- closure and positive closure：考虑 CFL L_1，令 $L = \{1\}^*, s(1) = L_1$，则 $L_1^* = s(L)$，同理，若 $L = \{1\}^+$，则 $L_1^+ = s(L)$
- homomorphism：考虑 L 是定义在 Σ 的 CFL，而 h 是定义在 Σ 的 homomorphism，则对于任意 $a \in \Sigma$，定义 $s(a) = \{h(a)\}$，则 $h(L) = s(L)$

Reversal

如果 L 是 CFL，则 L^R 也是 CFL

Proof. 构造 L 的文法 $G = (V, T, P, S)$ 使得 $L = L(G)$，则对于 L^R 其 CFG 为 $G^R = (V, T, P^R, S)$，其中 P^R 为每个产生式的 reversal，即若 $A \to \alpha \in G$，则 $A \to \alpha^R \in G^R$

> 对推导长度归纳即可得出 $L(G^R) = L^R$

Intersection

CFL 对 intersection 是不封闭的，考虑

$$L_1 = \{0^n 1^n 2^i : n \geqslant 1, i \geqslant 1\}$$
$$L_2 = \{0^i 1^n 2^n : n \geqslant 1, i \geqslant 1\}$$

上述两个语言都是 CFL，可以为其构造出对应的文法，但是其 intersection

$$L = L_1 \cap L_2 = \{0^n 1^n 2^n : n \geqslant 1\}$$

不是 CFL（pumping lemma 即可证明）

但是 CFL 和 正则语言的 intersection 仍是 CFL，即

对于 CFL L 和正则语言 R，$L \cap R$ 仍是 CFL

> Proof. 只需要平行地运行 PDA 和 DFA 即可
>
> 令 PDA $P = (Q_P, \Sigma, \Gamma, \delta_P, q_P, Z_0, F_P)$ 满足 $L = L(P)$，令 DFA $A = (Q_A, \Sigma, \delta_A, q_A, F_A)$ 满足 $R = L(A)$
>
> 则可构造 PDA
>
> $$P' = (Q_P \times Q_A, \Sigma, \Gamma, \delta, (q_P, q_A), Z_0, F_P \times F_A)$$
>
> 定义
>
> $$\delta((q,p), a, X) = \{((r,s), \gamma) : s = \delta_A(p,a) \text{ and } (r,\gamma) \in \delta_P(q,a,X)\}$$
>
> 注意当 PDA 在 ϵ 上转换时，DFA 不改变状态
>
> 根据推导步数的归纳可以得出、
>
> $$(q_P, w, Z_0) \underset{P}{\vdash^*} (q, \epsilon, \gamma) \text{ and } \delta_A(q_A, w) = p \iff ((q_P, q_A), w, Z_0) \underset{P'}{\vdash^*} ((q,p), \epsilon, \gamma)$$
>
> 且 (q,p) 为接收状态 $\iff q \in F_P \text{ and } p \in F_A$，故 $w \in L(P') \iff w \in L \text{ and } w \in A$
>
> 可以得出 $L(P') = L \cap R$

同样可以得出对于 CFL L, L_1, L_2 和正则语言 R 满足

- $L - R$ 是 CFL
- $\overline{L}$ 不一定是 CFL
- $L_1 - L_2$ 不一定是 CFL

> Proof.

- $L - R = L \cap \overline{R}$，而 $\overline{R}$ 是正则
- 若 $\overline{L}$ 总是 CFL，则考虑 $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$，这与上述结论矛盾
- 若 $L_1 - L_2$ 总是 CFL，则 $\Sigma^* - L = \overline{L}$ 也是 CFL，这与上述结论矛盾

Inverse Homomorphism

证明思路类似证明正则语言对 inverse homomorphism 是封闭的。构造一个 PDA，每当读入一个输入 a，则将 $h(a)$ 放入一个 buffer 中，每次读取其中一个符号，用于模拟 PDA 的运行，直至读入结束，再读入下一个符号

考虑 CFL L 和 homomorphism h，则 $h^{-1}(L)$ 也是 CFL

> Proof. 假设 h 定义在 Σ 上，且输出的 string 属于 T^*，且假设 L 是定义在 T 上的语言，则有 PDA $P = (Q, T, \Gamma, \delta, q_0, Z_0, F)$ 满足 $L = L(P)$，则构造一个新的 PDA
>
> $$P' = (Q', \Sigma, \Gamma, \delta', (q_0, \epsilon), Z_0, F \times \{\epsilon\})$$
>
> 其中
>
> Q' 中的元素 (q, x) 满足 $q \in Q$ 且 x 是某个 $h(a)$ 的后缀，即用 x 来模拟 buffer
>
> 由于 $a \in \Sigma$ 是有限的，故 $h(a)$ 也是有限的，则 Q' 中的状态也是有限的
>
> 对于 δ'，其定义分为两种情况
>
> - buffer 为空，即 $\delta'((q, \epsilon), a, X) = \{((q, h(a)), X)\}$，此处的 $a \neq \epsilon$。即当 buffer 为空，P' 读入下一个输入 a 并将 $h(a)$ 放入 buffer
> - buffer 不为空，则若 $(p, \gamma) \in \delta(q, b, X), b \in T \text{ or } b = \epsilon$，则
>
> $$((p, x), \gamma) \in \delta'((q, bx), \epsilon, X)$$
>
> 即 P' 用 buffer 中第一个 symbol 模拟 P 的运行
>
> 则有
>
> $$(q_0, h(w), Z_0) \vdash_P^* (p, \epsilon, \gamma) \iff ((q_0, \epsilon), w, Z_0) \vdash_{P'}^* ((p, \epsilon), \epsilon, \gamma)$$
>
> 证明基于对推导步数的归纳即可。需要注意的是 P' 在 buffer 非空时一定模拟 P 的运行，但是当 buffer 为空时仍有可能继续模拟 P 的运行
>
> 故 $w \in L(P') \iff h(w) \in L(P)$，即 $L(P') = h^{-1}(L(P))$

Decision Properties of CFL

对于 CFL 的 decision properties，有一些是有确定的算法可以解决的

- 判断 w 是否在 CFL L 中
- 判断 CFL L 是否为空
- 判断 CFL L 是否无限

而有一些没有算法可以解决，换言之这些问题是 undecidable 的

- 判断两个 CFL 是否相等
- 判断两个 CFL 是否 disjoint

Emptiness of CFL

测试 CFL L 是否为空只需要测试其开始符号 S 是否为 generating 即可

CFL L 非空 $\iff$ 其开始符号 S 为 generating

Membership of CFL

有一种算法利用 DP 的思想，若 string 的长度为 n，可以在 $O(n^3)$ 的时间内确定这个 string 是否在给定的 CFL 中，即 CYK 算法。

考虑 CFL L 及其 CNF 形式的文法 $G=(V,T,P,S)$，设 $|w|=n, w=a_1a_2\dots a_n$，算法构造一个三角矩阵形如

$$\begin{matrix} x_{1n} & & & \\ x_{1n-1} & x_{2n} & & \\ \vdots & \vdots & \ddots & \\ x_{11} & x_{22} & \dots & x_{nn} \end{matrix}$$

其中

$$x_{ij}=\{A:A\overset{*}{\Rightarrow}a_ia_{i+1}\dots a_j\}$$

则

$$S\in x_{1n}\iff S\overset{*}{\Rightarrow}w$$

从底向上，第 i 行的条目都代表了一系列 variable，可以推导出长为 i 的 substring

计算表中条目是一个从底向上的归纳的过程

Basis. 即最底下一行，$x_{ii}=\{A:A\to a_i\in G\}$

Induction. 假设需要计算 x_{ij}，位于第 $j-i+1$ 行，且其下的条目都已计算，则任意推导过程

$$A\overset{*}{\Rightarrow}a_ia_{i+1}\dots a_j$$

都从 $A\to BC$ 开始，则对某个 $i\leqslant k<j$ 有

$$B \overset{*}{\Rightarrow} a_i \dots a_k$$
$$C \overset{*}{\Rightarrow} a_{k+1} \dots a_j$$

则需要找到所有满足

- $i \leqslant k < j$
- $B \in x_{ik}$
- $C \in x_{k+1j}$
- $A \to BC \in G$

的 A，这样的 A 即为 x_{ij} 的成员

CYK 算法的正确性从其归纳的计算过程中即可得出。

算法计算每个条目需要 $O(n)$ 的时间，因为其要检查每个 (x_{ik}, x_{k+1j}) 对，而表中一共有 $\frac{n(n+1)}{2}$ 个条目。故其运行时间为 $O(n^3)$

Infiniteness of CFL

Infiniteness 的测试思想与正则语言相同，对于 pumping lemma 的常数 n，只需测试长度在 $[n, 2n-1]$ 的 string 是否都在 L 中即可

存在一个满足条件的 string 则 L 为无穷