Теоретическая информатика, осень 2020 г. Лекция 6. Применение леммы о накачке. Язык, удовлетворяющий лемме о накачке. Лемма Огдена. Грамматики над односимвольным алфавитом. Невыразимые действия над грамматиками. Удаление пустых правил, удаление единичных правил, нормальный вид Хомского*

Александр Охотин

8 октября 2020 г.

Содержание

1 Ограничения выразительной мощности грамматик

1.1 Лемма о накачке (окончание)

С помощью леммы о накачке можно доказывать несуществование грамматик для ряда языков.

Пример 1. *Язык* $L = \{ a^n b^n c^n \mid n \geqslant 0 \}$ *не задаётся никакой грамматикой.*

Доказательство. Пусть есть, и пусть $p \geqslant 1$ — константа из леммы о накачке. Тогда для строки $w = a^p b^p c^p$ есть разбиение $w = xuyvz$, и нужно рассмотреть несколько случаев возможных разбиений.

- Если одна из строк u,v содержит символы разных типов, т.е., пересекает границу между a, b и c. Тогда $xu^2yv^2z \notin a^*b^*c^*$, и потому эта строка не может принадлежать L.
- Если $u, v \in a^*$, то $xu^0yv^0z = xyz = a^{p-|uv|}b^pc^p \notin L$. Случаи $u, v \in b^*$ и $u, v \in c^*$ такие же.

*Краткое содержание лекций, прочитанных студентам 2-го курса факультета МКН СПбГУ в осеннем семестре 2020–2021 учебного года. Страница курса: `http://users.math-cs.spbu.ru/~okhotin/teaching/tcs_fl_2020/`.

- Если $u \in a^*$ и $v \in b^*$, то $xu^0yv^0z = a^{p-|u|}b^{p-|v|}c^p$; эта строка не принадлежит L, поскольку $p - |u| \neq p$ или $p - |v| \neq p$. Случай $u \in b^*$ и $v \in c^*$ такой же.

В каждом случае получено противоречие. □

На основе этого факта можно сделать выводы о непредставимости, например, языков программирования.

Пример 2 (Флойд [1962]). *Следующая строка — правильная программа на языке C тогда и только тогда, когда $i = j = k$.*

$$\texttt{main() \{ int } \underbrace{\texttt{x...x}}_{i\geqslant 1}\texttt{; } \underbrace{\texttt{x...x}}_{j\geqslant 1} \texttt{ = } \underbrace{\texttt{x...x}}_{k\geqslant 1}\texttt{; \}}$$

Отсюда можно вывести, что множество всех правильных программ на C не описывается грамматикой. Грамматиками описывают существенную часть синтаксиса языков программирования, но не весь синтаксис целиком.


Рис. 1: Роберт Флойд (1936–2001), Дэвид Вайз (род. 1945).

Существуют, однако, языки, удовлетворяющие условию леммы о накачке, но не задаваемые никакой грамматикой. Вот пример такого языка.

Пример 3 (Вайз [1976]). *Язык $\{\, a^\ell b^m c^n \mid \ell, m, n \geqslant 1,\ \ell \neq m,\ m \neq n,\ \ell \neq n \,\}$ удовлетворяет лемме о накачке с константой $p = 7$.*

Доказательство. Для всякой строки $w = a^{\ell_a} b^{\ell_b} c^{\ell_c} \in L$, где $|w| \geqslant 7$, пусть $s \in \{a, b, c\}$ — тот из символов, который встречается в w чаще, чем два других, так что ℓ_s — наибольшее из чисел ℓ_a, ℓ_b и ℓ_c. Тогда $\ell_s \geqslant 4$, поскольку если $\ell_s \leqslant 3$, то строка имела бы длину 6.

Пусть k — наименьшее натуральное число, отличающееся от $\ell_s - \ell_t$, для всех $t \in \{a, b, c\}$. Число k не превосходит 3, и потому меньше, чем ℓ_s. Тогда предполагается накачивать блок из k символов s; иными словами, разбиение $w = xuyvz$, обещанное в лемме о накачке, определяется, полагая $u = s^k$ и $v = \varepsilon$, где x, y и z задаются подобающим образом, чтобы вышло $xuyvz = w$.

Для всякого $i \geqslant 1$, строка xu^iyv^iz принадлежит L, поскольку количество символов s остаётся наибольшим, в то время как количество двух других символов не изменяется. Для

$i = 0$, строка xu^0yv^0z, из которой удалена подстока s^k, всё ещё принадлежит языку L, потому что, согласно условию $\ell_s - \ell_t \neq k$, получившееся количество символов s отличается от количества каждого из оставшихся символов. □

1.2 Лемма Огдена

Чтобы доказать несуществование грамматики для языка из примера 3, используется следующий более мощный вариант леммы о накачке.

Лемма 1 (лемма Огдена [1968])**.** *Для всякого языка $L \subseteq \Sigma^*$, задаваемого грамматикой, существует такое число $p \geqslant 1$, что для всякой строки $w \in L$ и для всякого множества $P \subseteq \{1, \ldots, |w|\}$ выделенных позиций в строке w, где $|P| \geqslant p$, существует разбиение $w = xuyvz$, где uv содержит хотя бы одну выделенную позицию, uyv содержит не более p выделенных позиций, и выполняется $xu^iyv^iz \in L$ для всех $i \geqslant 0$.*

Рис. 2: Вильям Огден.

Доказательство. Доказательство дословно повторяет доказательство леммы о накачке, с тем единственным отличием, что размер поддеревьев считается не по числу листьев, а по числу *выделенных* листьев. □

Пример 4. *Язык $L = \{\, a^\ell b^m c^n \mid \ell \neq m,\ m \neq n,\ \ell \neq n \,\}$ не удовлетворяет условию леммы Огдена и потому не задаётся никакой грамматикой.*

Доказательство. Пусть p — число, данное леммой Огдена. Пусть $w = a^{p+p!}b^pc^{p+2p!}$ — строка, в которой выделены символы b^p. Лемма Огдена даёт разбиение $w = xuyvz$.

Если u или v содержит символы двух различных типов, то достаточно накачать дважды, чтобы получить строку xu^2uv^2z, не лежащую в $a^*b^*c^*$ и соответственно не попадающую в L.

Если как u, так и v попадают в b^p, то b^p накачивается $\frac{p!}{|uv|} + 1$ раз до достижения $b^{p+p!}$, так что получается строка $a^{p+p!}b^{p+p!}c^{p+2p!}$, не принадлежащая языку L.

Если u попадает в $a^{p+p!}$ а v — в b^p, то они вместе накачиваются, пока b^p не достигает $b^{p+2p!}$. Что при этом происходит с $a^{p+p!}$ — неважно, поскольку всякая строка $a^kb^{p+2p!}c^{p+2p!}$, где $k \geqslant 0$, не принадлежит языку.

Если u попадает в b^p, а v попадает в $c^{p+2p!}$, то они накачиваются, пока b^p не превратится в $b^{p+p!}$. □

Для леммы Огдена также известны примеры языков, удовлетворяющих её условию, но не задаваемых никакой грамматикой.

1.3 Грамматики над односимвольным алфавитом

Теорема 1 (Парикх [1966]; Гинзбург и Райс [1962]). *Всякая грамматика над односимвольным алфавитом $\Sigma = \{a\}$ задаёт регулярный язык.*

Рис. 3: Рохит Парикх (род. 1936).

Доказательство. По лемме о накачке.

Пусть $L \subseteq a^*$ — язык, пусть p — константа из леммы о накачке. Для всякой строки $a^n \in L$, где $n \geqslant p$, лемма о накачке утверждает, что для некоторого числа ℓ, где $1 \leqslant \ell \leqslant p$, все строки вида $a^{n+i\cdot\ell}$, для всех $i \geqslant 0$, также лежат в L. В частности, раз $p!$ делится на ℓ, все строки вида $a^{n+i\cdot p!}$, где $i \geqslant 1$, принадлежат L.

Для всякого остатка j по модулю $p!$, язык $L \cap a^{\geqslant p}$ или содержит какую-то строку длины j по модулю $p!$, или не содержит. Если он содержит хотя бы одну строку a^{n_j}, где $n_j \equiv j \pmod{p!}$, то он содержит и все более длинные такие строки. Поэтому язык представим так.

$$L = (L \cap a^{<p}) \cup \bigcup_{\substack{j\in\{0,\ldots,p!-1\}:\\ n_j \text{ определено}}} a^{n_j}(a^{p!})^*$$

А это по существу регулярное выражение. □

2 Невыразимые действия над грамматиками

Незамкнутость относительно некой операции над языками: если для аргументов есть грамматика, а для результата операции — нет.

Пересечение: нельзя ли сделать каким-нибудь прямым произведением грамматик? К сожалению, нельзя.

Теорема 2 (Шейнберг [1960]). *Класс языков, задаваемых грамматиками, не замкнут относительно пересечения и дополнения.*

Доказательство. Языки $L_1 = \{\, a^i b^\ell c^\ell \mid i, \ell \geqslant 0 \,\}$ и $L_2 = \{\, a^m b^m c^j \mid j, m \geqslant 0 \,\}$, задаются следующими грамматиками.

$$\begin{array}{rcl} S_1 & \to & aS_1 \mid A \\ A & \to & bAc \mid \varepsilon \end{array} \qquad\qquad \begin{array}{rcl} S_2 & \to & S_2 c \mid B \\ B & \to & aBb \mid \varepsilon \end{array}$$

Если предположить, что семейство замкнуто относительно пересечения, то тогда пересечение двух вышеприведённых языков также будет описывается грамматикой.

$$L_1 \cap L_2 = \{\, a^i b^\ell c^\ell \mid i, \ell \geqslant 0 \,\} \cap \{\, a^m b^m c^j \mid j, m \geqslant 0 \,\} = \{\, a^n b^n c^n \mid n \geqslant 0 \,\}$$

Однако ранее было доказано, что грамматики для этого пересечения не существует — противоречие.

Незамкнутость относительно дополнения следует из этого результата, используя замкнутость относительно объединения. □

Ещё одно действие, не выразимое в грамматиках — это так называемое *деление* языков один на другой. Конечно, это не полноценная обратная операция к конкатенации (умножению) — для конкатенации обратного элемента нет. Но если строка u — префикс строки uv, то uv можно «поделить» на u слева, получив строку v. Эту операцию обозначают через $u^{-1} \cdot uv = v$, и её можно распространить на языки, записав множество всех возможных «частных» такого вида, для аргументов взятых из двух языков.

$$K^{-1} \cdot L = \{\, v \mid \exists u \in K :\ uv \in L \,\}$$

Теорема 3 (Гинзбург)**.** *Множество языков, задаваемых грамматиками, не замкнуто относительно операции деления.*

Доказательство. Языки $K = \{\, a^m b^{2m} \mid m \geqslant 1 \,\}^*$ и $L = a\{\, b^\ell a^\ell \mid \ell \geqslant 1 \,\}^*$ задаются следующими двумя грамматиками.

$$\begin{array}{rcl} S_2 & \to & S_2 A \mid \varepsilon \\ A & \to & aAbb \mid abb \end{array} \qquad\qquad \begin{array}{rcl} S_1 & \to & S_1 B \mid a \\ B & \to & bAa \mid ba \end{array}$$

Пусть их частное $K^{-1}L$ также задаётся грамматикой; тогда есть грамматика и для его пересечения с регулярным языком a^*. Какие строки попадут в это пересечение? Это все строки, полученные откусыванием от строки вида $ab^{m_1}a^{m_1}b^{m_2}a^{m_2}\dots a^{m_{n-1}}b^{m_n}a^{m_n}$ из L некоторой строки из K, причём так, чтобы остался только заключительный кусок, $v = a^{m_n}$. Это возможно только в том случае, если строка $u = ab^{m_1}a^{m_1}b^{m_2}a^{m_2}\dots a^{m_{n-1}}b^{m_n}$ принадлежит языку K. Строка u принадлежит K тогда и только тогда, когда $2 \cdot 1 = m_1$ (то есть, $m_1 = 2$), и далее, $2m_1 = m_2$, $2m_2 = m_3$, …, $2m_{n-1} = m_n$. Отсюда вытекает, что $m_n = 2^n$, и потому рассматриваемая произвольная строка, принадлежащая языку $K^{-1}L \cap a^*$, имеет вид $v = a^{2^n}$.

Поэтому язык имеет следующий вид.

$$K^{-1}L \cap a^* = \{\, a^{2^n} \mid n \geqslant 0 \,\}$$

Это нерегулярный язык над односимвольным алфавитом, и потому, по ранее доказанной теореме, он не задаётся никакой грамматикой. Тем самым получено противоречие. □

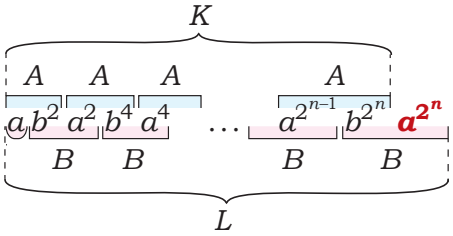

Рис. 4: Незамкнутость относительно деления.

3 Нормальный вид Хомского

Несмотря на то, что определение грамматик в целом очень просто, тем не менее, в них можно выразить некоторые конструкции, анализ которых может вызывать затруднения — как в алгоритмах, так и в математических доказательствах, и даже просто при чтении грамматики человеком.

Например, правило $A \to A$: алгоритм, глядя на него, может уйти в бесконечный цикл, доказательства индукцией по длине строки перестают проходить, да и человек, читающий грамматику, прежде чем он обнаружит, что это правило ни на что не влияет, может запутаться. А обнаружить подобные затруднения не всегда так просто. Скажем, правило $A \to EAE$, если E определено правилами $E \to EE \mid \varepsilon$, тоже ни на что не влияет, но обнаружение этого обстоятельства может потребовать некоторых усилий.

В более сложном варианте, грамматика может содержать *цепочку зависимостей*, образуемую, например, правилами $A \to EBE$ и $E \to \varepsilon$, связывающую представление строки w в виде A и в виде B. Прослеживание цепочки зависимостей может вылиться в *круговую зависимость*, такую как в правилах $A \to EBE$, $E \to \varepsilon$, $B \to EAE$, согласно которым представление w в виде A зависит от представления w в виде B, и наоборот.

Однако как цепочки зависимостей, так и круговые зависимости, могут быть удалены путём преобразования грамматики к *нормальному виду*.

3.1 Удаление пустых правил

Цель: по данной грамматике построить другую, задающую тот же язык и не содержащую «пустых правил» вида $A \to \varepsilon$.

Пример 5. *Грамматика, определяющая язык* $\{ab, b\}$.

$$S \to AB$$
$$A \to \varepsilon \mid a$$
$$B \to b$$

Правило $A \to \varepsilon$ *хочется убрать. Однако его удаление потребует присовокупить к правилу* $S \to AB$ *дополнительное правило* $S \to B$*, соответствующее случаю, когда* A *задаёт строку* ε*. Поэтому в итоге получится следующая грамматика.*

$$S \to AB \mid B$$
$$A \to a$$
$$B \to b$$

Сперва надо узнать, какие нетерминальные символы задают пустую строку ε. Для грамматики $G = (\Sigma, N, R, S)$ это следующее множество.

$$\textsc{Nullable}(G) = \{\, A \mid A \in N,\ \varepsilon \in L_G(A) \,\}$$

Алгоритм 1 Построение множества $\textsc{Nullable}(G)$.

Грамматика $G = (\Sigma,\ N,\ R,\ S)$, переменная: $W \subseteq N$.

1: $W = \varnothing$
2: **while** W можно изменить **do**
3: $\quad W = \{\, A \in N \mid$ есть правило $A \to B_1 \dots B_\ell,$ где $B_1, \dots, B_\ell \in W \,\}$

На выходе $W = \textsc{Nullable}(G)$.

Последовательно строятся следующие множества.

$$\textsc{Nullable}_0(G) = \varnothing$$
$$\textsc{Nullable}_{i+1}(G) = \{\, A \in N \mid \text{есть правило } A \to B_1 \dots B_\ell,\ \text{где } B_1, \dots, B_\ell \in \textsc{Nullable}_i(G) \,\}$$

Это последовательность вложенных множеств; рано или поздно множество насыщается.

Лемма 2. *Грамматика $G = (\Sigma, N, R, S)$, множество $\textsc{Nullable}(G)$ для неё. Новая грамматика $G' = (\Sigma, N, R', S)$, где R' содержит следующие правила.*

$$A \to X_1 \dots X_\ell \qquad \text{(для всех } \ell \geqslant 1,\ X_1, \dots, X_\ell \in \Sigma \cup N \text{ и } \theta_0, \dots, \theta_\ell \in \textsc{Nullable}(G)^*\text{).}$$
$$\text{для которых в } R \text{ есть правило } A \to \theta_0 X_1 \theta_1 \dots X_\ell \theta_\ell$$

Тогда, для каждого $A \in N$, $L_{G'}(A) = L_G(A) \setminus \{\varepsilon\}$.

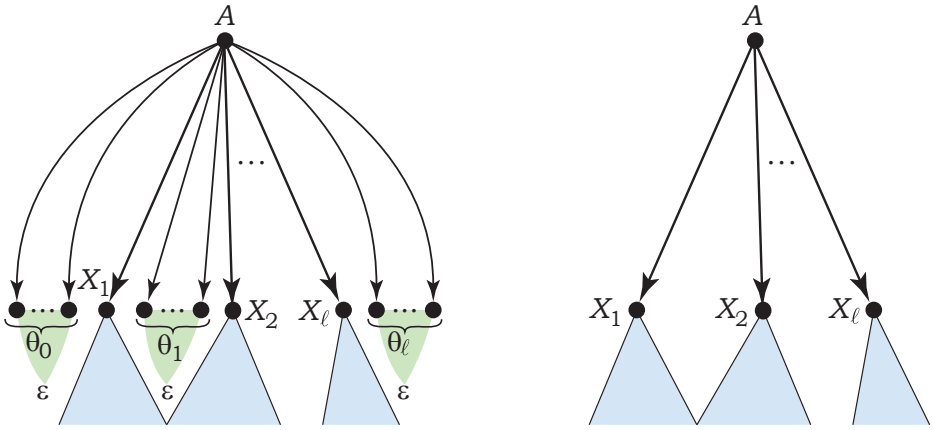

Рис. 5: Удаление пустых правил по лемме 2: (слева) правило $A \to \theta_0 X_1 \theta_1 \dots X_\ell \theta_\ell$ при разборе согласно исходной грамматике, где все θ_i описывают пустую строку; (справа) правило $A \to X_1 \dots X_\ell$ при разборе той же строки согласно построенной грамматике.

Доказательство. Преобразованием деревьев разбора. В одну сторону: если есть дерево непустой строки w в грамматике G, то у всякой его вершины, у которой есть потомки,

задающие пустую строку, как на рис. 5(левом), все эти потомки просто удаляются, как показано на рис. 5(правом) — причём в грамматике G' по построению найдётся правило без выброшенных нетерминальных символов. Получится дерево той же строки в грамматике G'.

И обратно: всякий раз, когда в дереве строки w согласно грамматике G' используется правило, которого не было в G, это правило было получено выбрасыванием нетерминальных символов, задающих пустую строку. Тогда каждой вершине, как на рис. 5(правом), добавляются эти выброшенные потомки с деревьями разбора пустой строки, как на рис. 5(левом).

Строгое доказательство: в каждую сторону индукцией по высоте дерева разбора, считая вершину A корнем. □

Построение из леммы 2 может привести к экспоненциальному росту размера грамматики.

Пример 6. *Для всякого $n \geqslant 1$, следующая грамматика описывает множество всех строк над односимвольным алфавитом длины не более чем n.*

$$\begin{aligned} S &\to A_1 \dots A_n \\ A_i &\to a \mid \varepsilon \qquad\qquad (i \in \{1, \dots, n\}) \end{aligned}$$

По построению из леммы 2, из единственного правила для S получаются $2^n - 2$ новых правил, соответствующих всем способам удаления нетерминальных символов, задающих пустую строку.

Подобный экспоненциальный рост может получится только для правил с длинными правыми частями. Чтобы его избежать, следует сперва преобразовать грамматику, обрезав правые части правил до длины не более чем 2. Для этого всякое правило $A \to XY\alpha$, где $X, Y \in \Sigma \cup N$ и $\alpha \in (\Sigma \cup N)^+$, заменяется на следующие два правила, где A' — новый нетерминальный символ.

$$\begin{aligned} A &\to A'\alpha \\ A' &\to XY \end{aligned}$$

Это преобразование применяется, пока все правила не станут короткими. Оно приводит лишь к линейному росту размера грамматики. Далее, построение из леммы 2, применённое к получившейся грамматике, также вызывает лишь линейный рост размера.

3.2 Удаление единичных правил

Цель: по данной грамматике без пустых правил построить грамматику, задающую тот же язык и не содержащую «единичных правил» вида $A \to B$.

Лемма 3. *Пусть $G = (\Sigma, N, R, S)$ — грамматика, в которой всякое правило имеет вид $A \to \alpha$, где $|\alpha| \geqslant 1$, и, соответственно, $\varepsilon \notin L_G(A)$ для всех $A \in N$. Тогда тот же язык задаётся грамматикой $G' = (\Sigma, N, R', S)$, где R' содержит правило $A \to \alpha$ тогда и только тогда, когда $\alpha \in (\Sigma \cup N)^* \setminus N$ и есть последовательность правил $A_0 \to A_1$, $A_1 \to A_2$, …, $A_{k-1} \to A_k$, $A_k \to \alpha$ в R, где $A_0 = A$ и $k \geqslant 0$.*

Доказательство. Преобразованием деревьев разбора в обе стороны, как в лемме 2. □

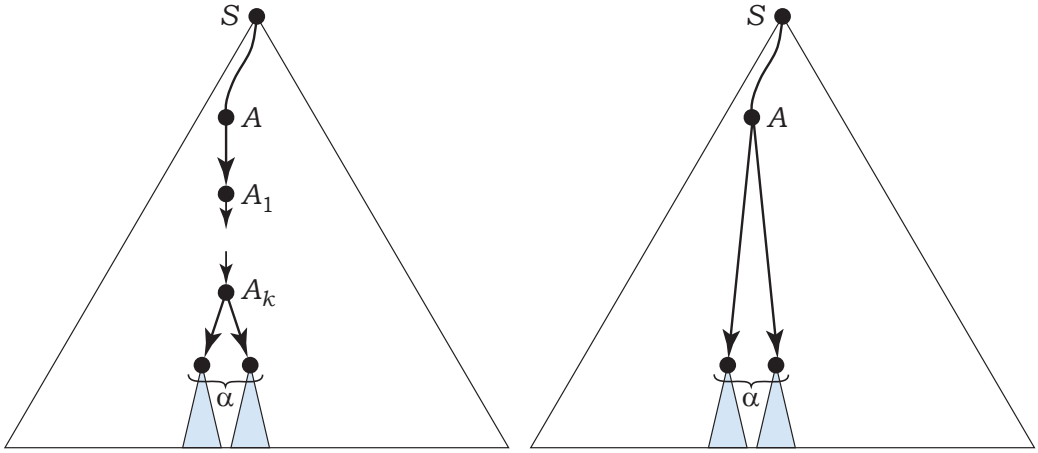

Рис. 6: Удаление единичных правил: (слева) цепочка правил $A \to A_1$, $A_1 \to A_2$, $\ldots$, $A_{k-1} \to A_k$, $A_k \to \alpha$ в исходной грамматике; (справа) немедленное применение правила $A \to \alpha$ в построенной грамматике.

Похоже на преобразование ε-NFA в NFA. Рост размера грамматики: $O(n^2)$, так как в худшем случае каждый нетерминальный символ получит по собственному экземпляру почти каждого правила исходной грамматики.

Нижняя оценка на сложность: не менее чем $n^{\frac{3}{2}-o(1)}$ (Норберт Блум [1983]).

3.3 Приведение к нормальному виду

Определение 1. *Грамматика $G = (\Sigma, N, R, S)$ — в нормальном виде Хомского, если всякое правило из R имеет следующий вид.*

$A \to BC$	$(B, C \in N)$
$A \to a$	$(a \in \Sigma)$
$S \to \varepsilon$	(*только если S не используется в правых частях никаких правил*)

Теорема 4 (Хомский [1959])**.** *Для всякой грамматики можно построить грамматику в н.в.Хомского, задающую тот же язык.*

Набросок доказательства. Подготовка: нарезать правила, чтобы длина правой части не превосходила 2.

Шаг 1: удалить все *пустые правила* вида $A \to \varepsilon$.

Шаг 2: удалить *единичные правила* вида $A \to B$.

Последний шаг: переместить символы из Σ в отдельные правила. □

Список литературы

[1983] N. Blum, “More on the power of chain rules in context-free grammars”, *Theoretical Computer Science*, 27 (1983), 287–295.

[1959] N. Chomsky, “On certain formal properties of grammars”, *Information and Control*, 2:2 (1959), 137–167.

[1962] R. W. Floyd, “On the non-existence of a phrase structure grammar for ALGOL 60”, *Communications of the ACM*, 5 (1962), 483–484.

[1962] S. Ginsburg, H. G. Rice, “Two families of languages related to ALGOL”, *Journal of the ACM*, 9 (1962), 350–371.

[1968] W. F. Ogden, “A helpful result for proving inherent ambiguity”, *Mathematical Systems Theory* 2:3 (1968), 191–194.

[1966] R. J. Parikh, “On context-free languages”, *Journal of the ACM* 13:4 (1966), 570–581.

[1960] S. Scheinberg, “Note on the boolean properties of context free languages”, *Information and Control*, 3 (1960), 372-375.

[1976] D. S. Wise, “A strong pumping lemma for context-free languages”, *Theoretical Computer Science*, 3:3 (1976), 359–369.