Другие регулярные модели. Сложность анализа регулярных выражений

Теория формальных языков

2021 г.

Леволинейные грамматики

Класс грамматик, симметричный праволинейным:

- виды правил — $A_i \to a_j$, $A_i \to A_k a_j$ и $S \to \varepsilon$, если S не встречается в правых частях правил;
- описывает тот же самый класс языков, что и праволинейные грамматики.

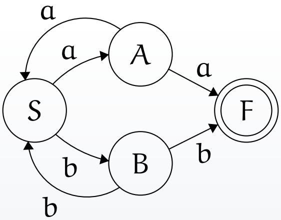
Преобразование в леволинейную форму легко сделать по обратным проходам из финального в начальное состояние в НКА, соответствующем грамматике.

Леволинейные грамматики

Преобразование в леволинейную форму легко сделать по обратным проходам из финального в начальное состояние в НКА, соответствующем грамматике.

$$S \to aA \mid bB \quad A \to aS \mid a \quad B \to bS \mid b$$

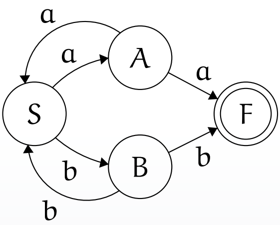
Строим недетерминированный КА по грамматике. Здесь F — финальное состояние, куда добавляются переходы $\langle A_i, a_j \rangle \to F$, соответствующие правилам $A_i \to a_j$ грамматики.

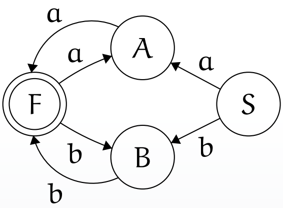


Леволинейные грамматики

$$S \to aA \mid bB \quad A \to aS \mid a \quad B \to bS \mid b$$

Теперь обращаем стрелки и меняем начальное и финальное состояния местами.

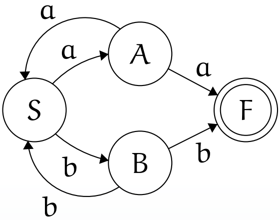


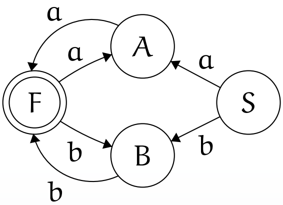


Леволинейные грамматики

$$S \to aA \mid bB \quad A \to aS \mid a \quad B \to bS \mid b$$

Теперь обращаем стрелки и меняем начальное и финальное состояния местами.





По полученному автомату строим леволинейную грамматику:

$$S \to Aa \mid Bb \quad A \to Fa \mid a \quad B \to Fb \mid b \quad F \to Aa \mid Bb$$

Леволинейные грамматики

Формальный алгоритм приведения к леволинейности

1. Добавляем в грамматику новый стартовый символ S' и для всех правил вида $A_i \to a_j$ — правила $S' \to A_i a_j$.
2. Правила вида $A_i \to a_j A_k$ преобразуем в $A_k \to A_i a_j$.
3. По правилам вида $S \to a_j A_k$ дополнительно порождаем правила $A_k \to a_j$.
4. По правилам вида $S \to a_j$ и $S \to \varepsilon$ порождаем правила $S' \to a_j$ и $S' \to \varepsilon$ соответственно.

Если в исходной грамматике S не встречался в правых частях правил, тогда этот алгоритм породит непродуктивные правила $A_k \to S a_j$. Поэтому шаг 2 для правил вида $S \to a_j A_k$ и шаг 1 для $S \to a_j$ в этом случае делать не надо.

Цена детерминизма

Утверждение

Имея регулярную грамматику с N нетерминалами, по ней можно построить ДКА (самое большее) с $O(2^N)$ состояниями. Эта оценка является точной.

Цена детерминизма

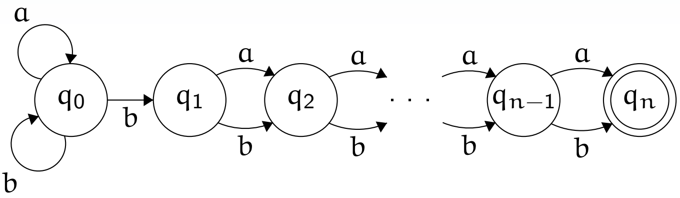
Утверждение

Имея регулярную грамматику с N нетерминалами, по ней можно построить ДКА (самое большее) с $O(2^N)$ состояниями. Эта оценка является точной.

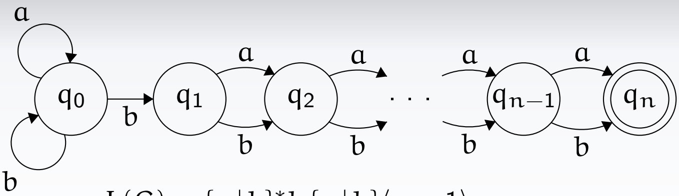
Рассмотрим грамматику G:

$$\begin{array}{llll}
S \to aS & S \to bS & S \to bA_1 & \\
A_1 \to aA_2 & A_1 \to bA_2 & A_2 \to aA_3 & A_2 \to bA_3 \\
\dots & & & \\
A_{n-1} \to aA_n & A_{n-1} \to bA_n & A_n \to a & A_n \to b
\end{array}$$

Грамматике G соответствует следующий НКА. Её язык — это слова вида $\{a \mid b\}^* b\{a \mid b\}\langle n-1\rangle$, то есть слова с n-ой буквой с конца, совпадающей с b.



Построим для этого языка таблицу классов эквивалентности и различающих слов по Майхиллу–Нероуду.



$$L(G) = \{a \mid b\}^* b \{a \mid b\}\langle n-1\rangle.$$

	ε	a	$\ldots$	a^{n-3}	a^{n-2}	a^{n-1}
a^n	$-$	$-$	$\ldots$	$-$	$-$	$-$
$a^{n-1}b$	$-$	$-$	$\ldots$	$-$	$-$	$+$
$a^{n-2}ba$	$-$	$-$	$\ldots$	$-$	$+$	$-$
$a^{n-2}bb$	$-$	$-$	$\ldots$	$-$	$+$	$+$
$a^{n-3}baa$	$-$	$-$	$\ldots$	$+$	$-$	$-$
$a^{n-3}bab$	$-$	$-$	$\ldots$	$+$	$-$	$+$
$a^{n-3}bba$	$-$	$-$	$\ldots$	$+$	$+$	$-$
$a^{n-3}bbb$	$-$	$-$	$\ldots$	$+$	$+$	$+$
$\ldots$			$\ldots$		$\ldots$	
$a\,b^{n-1}$	$-$	$+$	$\ldots$	$+$	$+$	$+$
b^n	$+$	$+$	$\ldots$	$+$	$+$	$+$

$$L(G) = \{a \mid b\}^* b \{a \mid b\}\langle n-1 \rangle.$$

	ε	a	$\dots$	a^{n-3}	a^{n-2}	a^{n-1}
a^n	$-$	$-$	$\dots$	$-$	$-$	$-$
$a^{n-1}b$	$-$	$-$	$\dots$	$-$	$-$	$+$
$a^{n-2}ba$	$-$	$-$	$\dots$	$-$	$+$	$-$
$a^{n-2}bb$	$-$	$-$	$\dots$	$-$	$+$	$+$
$a^{n-3}baa$	$-$	$-$	$\dots$	$+$	$-$	$-$
$a^{n-3}bab$	$-$	$-$	$\dots$	$+$	$-$	$+$
$a^{n-3}bba$	$-$	$-$	$\dots$	$+$	$+$	$-$
$a^{n-3}bbb$	$-$	$-$	$\dots$	$+$	$+$	$+$
$\dots$			$\dots$		$\dots$	
$a\,b^{n-1}$	$-$	$+$	$\dots$	$+$	$+$	$+$
b^n	$+$	$+$	$\dots$	$+$	$+$	$+$

Если в слове w_i в k-ой позиции стоит b, а в w_j стоит a, тогда суффикс a^{k-1} различает w_i и w_j. Все w_i различны $\Rightarrow$ для каждой пары i, j есть такое $k \Rightarrow$ нашлось минимум 2^n классов эквивалентности, и ДКА для L имеет не меньше 2^n состояний.

Ещё раз о лемме о накачке

В отличие от теоремы Майхилла–Нероуда, лемма о накачке использует свойства НКА, а не ДКА. А именно, если константа накачки языка не может быть меньше k, то в НКА, распознающем этот язык, не меньше k состояний.

Рассмотрим язык $L = \{w_1 b w_2 \mid |w_2| = 3\}$. Мы знаем, что распознающий его ДКА имеет минимум 16 состояний. Накачку $w \in L$, такую что $w = xyz$, $xy^n z \in L$, найти очень просто для всякого слова длины $\geqslant 5$: достаточно взять первую букву этого слова в качестве y, а x принять пустым.
Теперь пусть длина накачки p меньше 5. Рассмотрим слово $baaa \in L$. Любая накачка только букв a (нулевая и нет) выводит слово из языка, и нулевая накачка подслова, содержащего букву b, также выводит из языка L. Поэтому НКА, распознающий L, не может иметь меньше 5 состояний.

2-DFA

Двухсторонний DFA

Правила переписывания двухстороннего конечного автомата имеют вид:

$$\delta(q_i, a) = \langle q_j, \{R|L\}\rangle$$

Кроме того, введем символы начала и конца строки $\vdash$, $\dashv$, и кроме конечных состояний также класс отвергающих состояний. Скажем, что $w \in L(\mathscr{A})$, если при распознавании $\vdash w \dashv$ $\mathscr{A}$ оказался в финальном состоянии при чтении символа $\dashv$. Такие 2DFA также описывают регулярные языки.

Регулярность 2DFA

Пусть распознаваемое слово имеет вид $w = xz$. Определим функцию: $T_x(q) = p$, если 2DFA $\mathscr{A}$ зашёл в префикс x справа (из z) в состоянии q и вышел из префикса x обратно в z в состояние p. Состояние p полностью определяется парой $\langle x, q \rangle$. Добавим случаи, когда $\mathscr{A}$ заходит в z из x впервые ($T_x(\bullet) = p$), и когда $\mathscr{A}$ не выходит из x вообще (зацикливается или попадает в отвергающее состояние) — $T_x(p) = \perp$.

Пусть $(T_x(\bullet) = T_y(\bullet))\ \&\ \forall i(T_x(q_i) = T_y(q_i))$. Тогда $xz \in L(\mathscr{A}) \Leftrightarrow yz \in L(\mathscr{A})$, поскольку вся информация об x и y передаётся в z только через состояния. Поскольку функция T_x задана на конечном множестве и действует в конечное множество, таких функций конечное число (а именно, $(n+1)^{n+1}$, где n количество состояний $\mathscr{A}$). Значит, и классов эквивалентности в смысле Майхилла–Нероуда в $L(\mathscr{A})$ конечное число, и $L(\mathscr{A})$ регулярен.

Алфавитные префиксные грамматики

Определение APG

Дана SRS $\mathcal{S}$ с правилами переписывания двух видов:

$$a_i \to b_1 \dots b_n \qquad a_i \to \varepsilon$$

Разрешим применять правила только к первым буквам слова. Пусть дана пара $\langle \mathcal{S}, w_0 \rangle$, где w_0 — слово в алфавите Σ. Эта пара определяет APG.

Утверждение

Язык $L\langle \mathcal{S}, w_0 \rangle$ регулярен.

Алфавитные префиксные грамматики

Утверждение

Язык $L\langle \mathcal{S}, w_0 \rangle$ регулярен.

Скажем, что $a \twoheadrightarrow \varepsilon$ (a коллапсирует), если либо $a \to \varepsilon \in \mathcal{S}$, либо $\exists b_1, \dots, b_n (\forall b_i (b_i \twoheadrightarrow \varepsilon) \,\&\, a \to b_1 \dots b_n \in \mathcal{S})$.

По APG $\langle \mathcal{S}, s_1 \dots s_n \rangle$ породим праволинейную грамматику G.

Каждому символу алфавита a_i сопоставим A_i — нетерминал G.

1. Пусть $a \to b_1 \dots b_n$ и $\exists b_i (\neg (b_i \twoheadrightarrow \varepsilon) \,\&\, \forall j (j < i \Rightarrow b_j \twoheadrightarrow \varepsilon))$. Тогда добавим в G правила $A \to B_1 b_2 \dots b_n$, $A \to B_2 b_3 \dots b_n, \dots$, $A \to B_i b_{i+1} \dots b_n$, $A \to a$.
2. Если такого b_i нет, добавляем в G все правила вида $A \to B_1 b_2 \dots b_n, \dots, A \to B_{n-1} b_n$, $A \to B_n$, $A \to a$.
3. Вводим стартовый нетерминал S и для него добавляем развёртку в исходное слово $s_1 \dots s_m$ по правилам выше.
4. Если все s_i коллапсируют, тогда добавляем в G правило $S \to \varepsilon$.

Алфавитные префиксные грамматики

Скажем, что $a \twoheadrightarrow \varepsilon$ (a коллапсирует), если либо $a \to \varepsilon \in \mathcal{S}$, либо $\exists b_1, \dots, b_n (\forall b_i (b_i \twoheadrightarrow \varepsilon) \;\&\; a \to b_1 \dots b_n \in \mathcal{S})$.

По APG $\langle \mathcal{S}, s_1 \dots s_n \rangle$ породим праволинейную грамматику G.

Каждому символу алфавита a_i сопоставим A_i — нетерминал G.

1. Пусть $a \to b_1 \dots b_n$ и $\exists b_i (\neg(b_i \twoheadrightarrow \varepsilon) \;\&\; \forall j (j < i \Rightarrow b_j \twoheadrightarrow \varepsilon))$.
   Тогда добавим в G правила $A \to B_1 b_2 \dots b_n$,
   $A \to B_2 b_3 \dots b_n, \dots, A \to B_i b_{i+1} \dots b_n$, $A \to a$.
2. Если такого b_i нет, добавляем в G все правила вида
   $A \to B_1 b_2 \dots b_n, \dots, A \to B_{n-1} b_n$, $A \to B_n$, $A \to a$.
3. Вводим стартовый нетерминал S и для него добавляем развёртку в исходное слово $s_1 \dots s_m$ по правилам выше.
4. Если все s_i коллапсируют, тогда добавляем в G правило $S \to \varepsilon$.

Остается сделать развертку правил вида $A \to B_n$, либо перейти от G к NFA с ε-переходами.

Поведение стека в CBV-семантике

Рассмотрим стек с вершиной $\bullet_n$:

$\bullet_n \leftarrow f_{n+1}(\dots), \bullet_{n-1} \leftarrow f_n(\bullet_n \dots) \dots, \bullet_0 \leftarrow f_1(\bullet_1 \dots)$

Опишем его состояние перечислением имён функций в порядке их вхождения: $f_{n+1} f_n \dots f_1$.

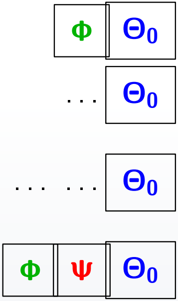
Шаги вычислений над такими состояниями стека описываются как применения правил в APG.

«Подозрительное» поведение — такое, при котором вершина стека повторяется, выбрасывая промежуточные вычисления.

Поиск бесконечных циклов

Отношение Турчина

Пусть на пути развертки программы имеются два состояния стеков: $c_1 : \Phi\Theta_0$, $c_2 : \Phi\Psi\Theta_0$, такие что Θ_0 неизменна на всём отрезке пути от c_1 до c_2. Тогда скажем, что $c_1 \preceq c_2$ (связаны отношением Турчина).



Если вершина Φ действительно входит в бесконечный цикл, порождая всё новые состояния вида $\Phi\Psi^n\Theta_0$, тогда правдоподобно, что $c_1 \preceq c_2$. Однако может случиться, что $c_1 \preceq c_2$ и на развертке завершающегося вычисления (ложное срабатывание).

Теорема Турчина

Вариант для CBV

На любом бесконечном пути вычислений имеются два состояния стека, такие что $c_1 \preceq c_2$.

Теорема Турчина гарантирует, что существование $\preceq$-пар — необходимое условие бесконечного (зацикливающегося) вычисления. Поэтому $\preceq$ может использоваться для приблизительного анализа завершаемости программ (наряду с другими условиями).

Пинг-понг протоколы

Определение

Пусть дано множество одноместных операций $\mathcal{V}_x = \mathcal{O}_x \cup \mathcal{P}_x$, задаваемое для участника x, причём для некоторых $p_1, p_2 \in \mathcal{V}_x$ выполняются тождества $p_1 \circ p_2 = \mathrm{id}$, и для всех $p_1, p_2, p_3 ((p_1 \circ p_2) \circ p_3 = p_1 \circ (p_2 \circ p_3))$. Пинг-понг протокол для двух участников — это конечная последовательность инструкций $[p_1 \dots p_n, [x, y]]$, $p_i \in \mathcal{V}_x \cup \mathcal{O}_y$.

$\mathcal{O}_x$ — публичные операции; $\mathcal{P}_x$ — приватные операции.

Модель угрозы Долева–Яо

Д. Долев & А. Яо — первая формальная модель угрозы и первое формальное понятие криптографического протокола (1983).

Злоумышленник по Долеву–Яо:

- Может перехватывать, пересылать и изменять любое сообщение в сети;
- Может играть роль любого пользователя (маскарад);
- Может убедить пользователей начать любой дозволенный протоколом сеанс передачи сообщений.

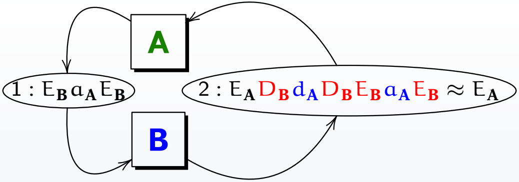
- Не может совершать битовые операции над сообщениями;
- Не может угадать свойства секретных операций.

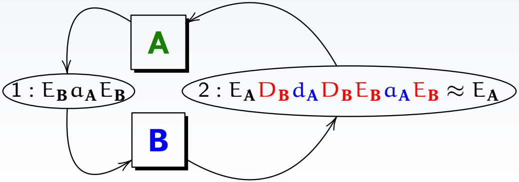
Протокол для двух участников

Легальные пользователи — **A**, **B**. **Злоумышленник** — **Z** (одного всегда достаточно).
Изначальное сообщение — M (обычно засекреченное).
Σ_x — словарь операторов x. E_x — зашифровка открытым ключом x, D_x — расшифровка E_x, a_x — приписывание к сообщению имени x, d_x — удаление префикса сообщения, совпадающего с именем x.

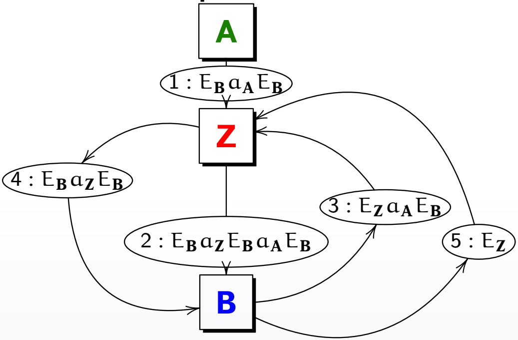
Протокол — набор α_i (**слов протокола**) и указаний, кто посылает α_i. **Атака** — последовательность подстановок в α_i, порождающая пустое слово (т.е. демаскирующая сообщение M).

Пример протокола

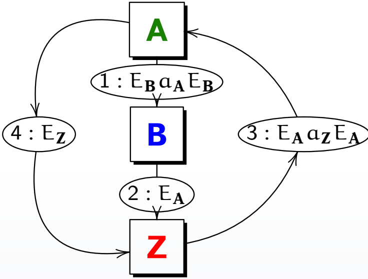




Первая атака



Вторая атака



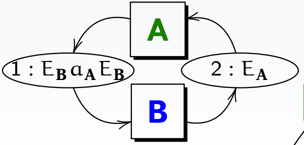
Автоматная модель $\mathscr{A}(\mathrm{P})$

- Строим все возможные подстановки в протокол P пар участников (включая злоумышленника);
- Строим начальное состояние 0 и конечное состояние 1, между ними — путь, соответствующий обращению первого слова протокола с двумя легальными участниками $\mathbf{A}$, $\mathbf{B}$ (чтобы было что атаковать);
- Строим пути из 0 в 0, соответствующие реверсам (обращенным) словам-подстановкам в протокол P;
- Строим пути из 0 в 0, соответствующие всем возможным индивидуальным действиям злоумышленника $\mathbf{Z}$ — т.е. элементам $\mathcal{O}_{\mathbf{A}}$, $\mathcal{O}_{\mathbf{B}}$, $\mathcal{O}_{\mathbf{Z}}$ и $\mathcal{P}_{\mathbf{Z}}$.

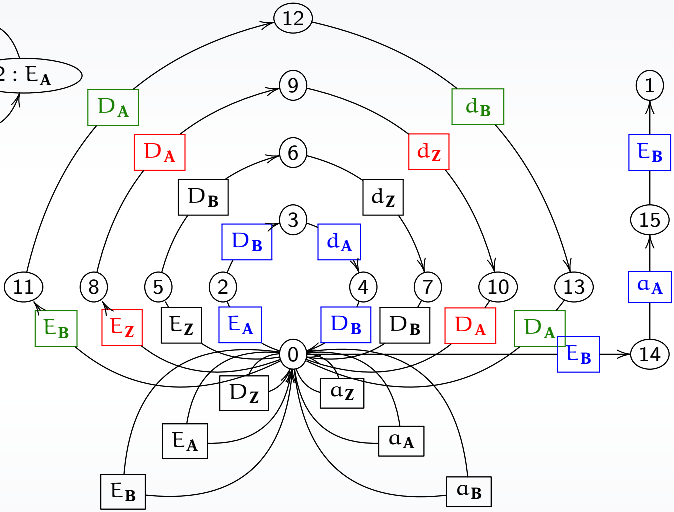
Утверждение

Протокол P ненадёжен в модели угрозы Долева–Яо тогда и только тогда, когда $\varepsilon \in \mathrm{L}(\mathscr{A}(\mathrm{P}))$.

Протокол



Автоматная модель



Случаи

$\mathbf{P}_{\text{Double}}[\mathbf{A}, \mathbf{B}]$

$\mathbf{P}_{\text{Double}}[\mathbf{B}, \mathbf{A}]$

$\mathbf{P}_{\text{Double}}[\mathbf{A}, \mathbf{Z}]$

$\mathbf{P}_{\text{Double}}[\mathbf{Z}, \mathbf{B}]$