Съдържание

1 Увод

Основна задача на лексическият анализатор е да чете символите на входния текст, да ги групира под формата на лексеми (тоукъни) и извежда като изход редица от тези лексеми. Лексемата е структура от данни, която съдържа нейния тип, текст и позиция във входния текст. Тази информация в последствие се подава на парсер, който от своя страна извършва синтактичният анализ на текста.

Лексическите анализатори могат да се използват и за други цели освен идентификация на лексемите. Други приложения са премахване на сегменти от текст (като нови редове, интервали и пр.), броене на символи, или думи, както и за проверка за грешки.

Лексемите се дефинират чрез регулярни изрази. Генератор за лексически анализ получава множество от регулярни изрази като вход и въз основа на тях строи лексически анализатор.

В тази работа представяме алгоритъм за конструкция на лексически анализатор по зададено множество от регулярни изрази. Анализаторът извлича лексимите за линейно време спрямо дължината на входния текст, като го сканира едновременно от ляво на дясно и от дясно на ляво, използвайки бимашина.

1.1 Мотивация

Съществуващите генератори на лексически анализатор като Lex, Flex и ANTLR намират широко приложение в индустрията. Те позволяват на потребителя да подаде спецификация на лексемите под формата на регулярни изрази и генерират програма, която извършва токенизацията входен текст спрямо тази спецификация.

Общото между тези инструменти е, че конструират крайни автомати, като сканирането на входния текст се случва в една посока - от ляво на дясно, чрез симулация на крайния автомат.

1.2 Обобщение

Съдиржа следните глави

Token	Description
Id	`[a-zA-Z_][a-zA-Z0-9_]*`
Number	`[0-9]+(\.[0-9]+)?`
Boolean	`true|false`
Operator	`=|==|!=|<|<=|>|>=`
If	`if`
Else	`else`
Func	`func`
Return	`return`
BraceOpen	`{`
BraceClose	`}`
WS	`[ \t\r\n]+`

Фигура 1: Спецификация на лексеми

2 Основни дефиниции

2.1 Крайни автомати

Дефиниция 2.1. *Краен автомат* дефинираме като петорка $\mathcal{A} = \langle \Sigma, Q, I, F, \Delta \rangle$, където

- Σ е *крайна азбука от символи*
- Q е *крайно множество от състояния*
- $I \subseteq Q$ е *множество от начални състояния*
- $F \subseteq Q$ е *множество от финални състояния*
- $\Delta \subseteq Q \times \Sigma \times Q$ е *релация на прехода*

Тройки от вида $\langle q_1, m, q_2 \rangle \in \Delta$ наричаме *преходи* и казваме, че започва състояние q_1, има етикет m и завършва в състояние q_2. Алтернативно, тези преходи обозначаваме като $q_1 \to^m q_2$.

Дефиниция 2.2. Нека $\mathcal{A}$ е краен автомат. *Разширена релация на прехода* $\Delta^* \subseteq Q \times \Sigma^* \times Q$ дефинираме индуктивно:

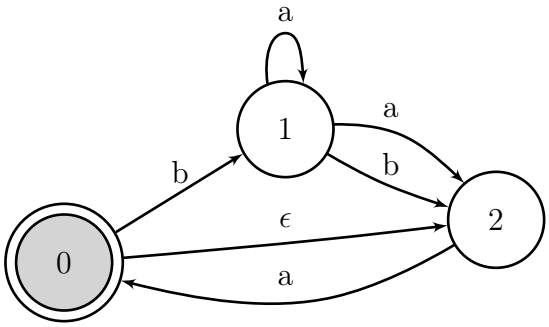
- $\langle q, \epsilon, q \rangle \in \Delta^*$ за всяко $q \in Q$

- $\langle q_1, wa, q_2\rangle \in \Delta^*$ за всяко $q_1, q_2, q \in Q$, $a \in \Sigma, w \in \Sigma^*$, ако $\langle q_1, w, q\rangle \in \Delta^*$ и $\langle q, a, q_2\rangle \in \Delta$

Дефиниция 2.3. Нека $\mathcal{A} = \langle \Sigma, Q, I, F, \Delta\rangle$ е краен автомат. *Път* в $\mathcal{A}$ наричаме крайна редица от преходи с дължина $k > 0$

$$\pi = q_0 \to^{a_1} q_1 \to^{a_2} \ldots \to^{a_k} q_k$$

където $\langle q_{i-1}, a_i, q_i\rangle \in \Delta$ за $i = 1 \ldots k$. Казваме, че *пътят* започва от състояние q_0 и завършва в състояние q_k. Елементите $q_0, q_1, \ldots, q_k$ наричаме *състояния на пътя*, а думата $w = a_1 a_2 \ldots a_k$ наричаме *етикет на пътя*.
Успешен път в автомата е *път*, който започва от начално състояние и завършва във финално състояние.



Фигура 2: Недетерминиран краен автомат

Пример 2.1. Нека е зададена азбука $\Sigma = \{a, b, \epsilon\}$ и автомат $\mathcal{A}$ над Σ със състояния $Q = \{0, 1, 2\}$, начални $I = \{0\}$, финални $F = \{0\}$ и релация на прехода

$$\Delta = \{\langle 0, b, 1\rangle, \langle 0, \epsilon, 2\rangle, \langle 1, a, 1\rangle, \langle 1, a, 2\rangle\, \langle 1, b, 2\rangle, \langle 2, a, 0\rangle\}$$

$\mathcal{A}$ е изобразен на Фигура 2. $0 \to^b 1 \to^a 1 \to^b 2 \to^a 0$ е успешен път, разпознавайки думата *baba*.

Дефиниция 2.4. Нека $\mathcal{A}$ е краен автомат. Множеството от етикети на всички успещни пътища в $\mathcal{A}$ наричаме *език на* $\mathcal{A}$ и обозначаваме като $L(\mathcal{A})$.

$$L(\mathcal{A}) = \{w \in \Sigma^* \mid \exists i \in I, f \in F : \langle i, w, f\rangle \in \Delta^*\}$$

Дефиниция 2.5. Нека $\mathcal{A}_1$ и $\mathcal{A}_2$ са крайни автомати. Казваме, че $\mathcal{A}_1$ е еквивалентен на $\mathcal{A}_2$ ($\mathcal{A}_1 \equiv \mathcal{A}_2$), ако езиците им съвпадат ($L(\mathcal{A}_1) = L(\mathcal{A}_2)$)

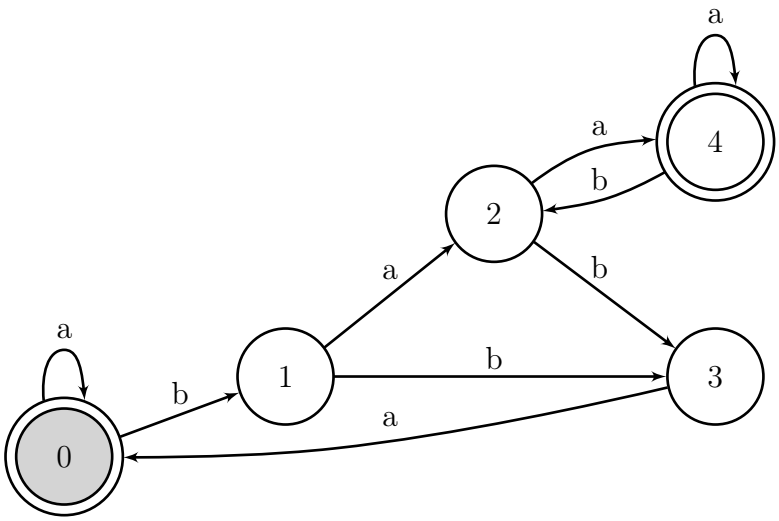
Дефиниция 2.6. Краен автомат $\mathcal{A} = \langle \Sigma, Q, I, F, \Delta \rangle$ е *детерминиран*, ако:

- $\mathcal{A}$ има единствено начално състояние $I = \{q_0\}$.
- За всяко $q_1 \in Q$ и символ $a \in \Sigma$, съществува не повече от едно $q_2 \in Q$, такова че $\langle q_1, a, q_2 \rangle \in \Delta$.

Иначе казано, релацията на прехода може да се представи като частична функция $\delta : Q \times \Sigma \to Q$ и *детерминираните автомати* можем преставим в следния вид

$$\mathcal{A}_D = \langle \Sigma, Q, q_0, F, \delta \rangle$$

Предимството на *детерминираните автомати* се изразява в това, че могат да разпознават дали дума w принадлежи на езика на автомата $L(\mathcal{A}_D)$ за линейно време спрямо дължината ѝ - $O(|w|)$, но в определени случаи могат да имат експоненциален брой състояния спрямо еквивалентният им недетерминиран автомат.



Фигура 3: Детерминиран краен автомат

Дефиниция 2.7. Нека $\mathcal{A}_D = \langle \Sigma, Q, q_0, F, \delta \rangle$ е *детерминиран краен автомат. Разширена функция на прехода* $\delta^* : Q \times \Sigma^* \to Q$ дефинираме индуктивно:

- $\delta^*(q, \epsilon) = q$

- $\delta^*(q, aw) = \delta^*(\delta(q, a), w)$, където $a \in \Sigma, w \in \Sigma^*$

Теорема 2.1. За всеки краен автомат $\mathcal{A} = \langle \Sigma, Q, I, F, \Delta \rangle$, съществува еквивалентен на него, детерминиран краен автомат $\mathcal{A}_D$, където $L(\mathcal{A}) = L(\mathcal{A}_D)$.

Доказателство. Нека $\mathcal{A}$ е краен автомат, на който сме премахнали ϵ-преходите. Строим *детерминиран краен автомат* $\mathcal{A}_D = \langle \Sigma, 2^Q, I, F_D, \delta \rangle$, където:

- $F_D = \{S \in 2^Q \mid S \cap F \neq \emptyset\}$
- $\delta(S, a) = \{q \in Q \mid \exists p \in S : \langle p, a, q \rangle \in \Delta\}$

С индукция по дължината на w, ще покажем, че за произволна дума $w \in \Sigma^*$ твръденията $\exists i \in I : \langle i, w, p \rangle \in \Delta^*$ и $p \in \delta^*(I, w)$ са еквивалентни:

- *База*: за $|w| = 0$ имаме $w = \epsilon$. Тогава $\exists i \in I : \langle i, \epsilon, i \rangle \in \Delta^*$. Тъй като $\mathcal{A}$ няма ϵ-преходи, то $\delta^*(I, \epsilon) = I$.
- *Индукция*: $w = w'a, a \in \Sigma, |w| = |w'| + 1$.
  ($\Rightarrow$) Нека допуснем, че $\exists i \in I$, така че $\langle i, w'a, p \rangle \in \Delta^*$, което значи, че $\exists p' \in Q : \langle p', a, p \rangle \in \Delta$. По индуктивното предположение знаем, че и $p' \in \delta^*(I, w')$ и от дефиницията на δ следва, че $p \in \delta(\delta^*(I, w'), a)$.
  ($\Leftarrow$) Нека допуснем, че $p \in \delta^*(I, w'a)$. Тогава $\exists p' : p' \in \delta^*(I, w')$. От индуктиното предположение знаем, че $\langle i, w', p' \rangle \in \Delta^*$ и от дефинициите на δ и Δ следва, че $\langle p', a, p \rangle \in \Delta$, от където следва, че $\exists i \in I : \langle i, w'a, p \rangle \in \Delta^*$.

Така можем да заключим, че за всяка дума $w \in L(\mathcal{A})$, $\exists i \in I, \exists f \in F : \langle i, w, f \rangle \in \Delta^*$ е изпълнено, че $\delta^*(I, w) \in F_D$, следователно $w \in L(\mathcal{A}_D)$ и $L(\mathcal{A}) = L(\mathcal{A}_D)$. $\square$

2.2 Крайни преобразуватели

Дефиниция 2.8. *Краен преобразувател* дефинираме като петорка $\mathcal{T} = \langle \Sigma_1^* \times \Sigma_2^*, Q, I, F, \Delta \rangle$, където

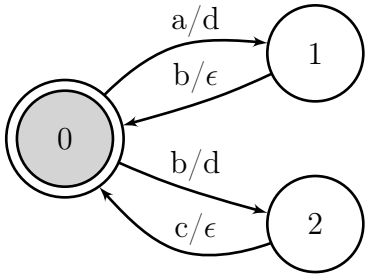
- Σ_1, Σ_2 са *крайни азбуки от символи*
- Q е *крайно множество от състояния*
- $I \subseteq Q$ е *множество от начални състояния*
- $F \subseteq Q$ е *множество от финални състояния*
- $\Delta \subseteq Q \times (\Sigma_1^* \times \Sigma_2^*) \times Q$ е *релация на прехода*

Тройки от вида $\langle q_1, \langle w, m\rangle, q_2\rangle \in \Delta$ наричаме *преходи* и казваме, че започва състояние q_1, има етикет по горната лента w и по долната лента m и завършва в състояние q_2. Алтернативно, тези преходи обозначаваме като $q_1 \to_m^w q_2$.

Дефиниция 2.9. Нека $\mathcal{T} = \langle \Sigma_1^* \times \Sigma_2^*, Q, I, F, \Delta\rangle$ е краен преобразувател. *Път* в $\mathcal{T}$ наричаме крайна редица от преходи с дължина $k > 0$

$$\pi = q_0 \to_{m_1}^{w_1} q_1 \to_{m_2}^{w_2} \ldots \to_{m_k}^{w_k} q_k$$

където $\langle q_{i-1}, \langle w_i, m_i\rangle, q_i\rangle \in \Delta$ за $i = 1 \ldots k$. Казваме, че *пътят* започва от състояние q_0 и завършва в състояние q_k. Елементите $q_0, q_1, \ldots, q_k$ наричаме *състояния на пътя*, а думата $w = w_1 w_2 \ldots w_k$ наричаме *входна дума* на пътя, а $m = m_1 m_2 \ldots m_k$ е *изходна дума* на пътя.
Успешен път в преобразувателя започва от начално състояние и завършва във финално състояние.



Фигура 4: Краен преобразувател

Пример 2.2. Нека са зададени азбукаи $\Sigma_1 = \{a, b, c\}, \Sigma_2 = \{d, \epsilon\}$ и преобразувател $\mathcal{T}$ над $\Sigma_1^* \times \Sigma_2^*$ със състояния $Q = \{0, 1, 2\}$, начални $I = \{0\}$, финални $F = \{0\}$ и релация на прехода $\Delta = \{\langle 0, \langle a, d\rangle, 1\rangle, \langle 1, \langle b, \epsilon\rangle, 0\rangle, \langle 0, \langle b, d\rangle, 2\rangle, \langle 2, \langle c, \epsilon\rangle, 0\rangle\}$.
$\mathcal{T}$ е изобразен на Фигура 4.
$0 \to_d^b 2 \to_\epsilon^c 0 \to_d^a 1 \to_\epsilon^b 0$ е успешен път, превеждайки думата *bcab* в *dd*.

Дефиниция 2.10. Нека $\mathcal{T} = \langle \Sigma_1^* \times \Sigma_2^*, Q, I, F, \Delta\rangle$ е краен преобразувател. *Подлежащ автомат* на $\mathcal{T}$ дефинираме като $\mathcal{A} = \langle \Sigma, Q, I, F, \Delta'\rangle$, където $\Delta' = \{\langle p, w, q\rangle \mid \langle p, \langle w, m\rangle, q\rangle \in \Delta\}$

2.3 Регулярни езици и релации

Дефиниция 2.11. *Регулярен език* е множество над крайна азбука Σ, което дефинираме индуктивно:

- $\emptyset$ е регулярен език.
- ако $a \in \Sigma$, то $\{a\}$ е регулярен език.
- ако $L_1, L_2 \subseteq \Sigma^*$ са регулярни езици, то
  - $L_1 \cup L_2$ (обединение)
  - $L_1 \cdot L_2 = \{a \cdot b \mid a \in L_1, b \in L_2\}$ (конкатенация)
  - $L_1^* = \bigcup_{i=0}^{\infty} L^i$ (звезда на Клини)

  са също регулярни езици.
- Не съществуват други регулярни езици.

Регулярните езици са също така *затворени* относно операциите *допълнение* ($\Sigma^* \setminus L_1$), *разлика* ($L_1 \setminus L_2$), *обръщане* (L_1^{-1}) и *сечение* ($L_1 \cap L_2$).

Дефиниция 2.12. *Двоична регулярна стрингова релация* дефинираме индуктивно като множество от двойки над крайни азбуки Σ_1, Σ_2:

- $\emptyset$ е регулярна релация.
- Ако $a \in \Sigma_1 \times \Sigma_2$, то $\{a\}$ е регулярна релация.
- Ако R_1, R_2 са регулярни релации, то:
  - $R_1 \cup R_2$ (обединение)
  - $R_1 \cdot R_2 = \{a \cdot b \mid a \in R_1, b \in R_2\}$ (конкатенация)
  - $R_1^* = \bigcup_{i=0}^{\infty} R^i$ (звезда на Клини)

  са също регулярни релации.
- Не съществуват други регулярни релации.

Дефиниция 2.13. Нека $R_1, R_2, \in \Sigma^* \times \Sigma^*$ са *двоични стрингови релации. Конкатенацията* на R_1 и R_2 дефинираме както следва

$$R_1 \cdot R_2 = \{\langle u_1 \cdot v_1, u_2 \cdot v_2 \rangle \mid \langle u_1, u_2 \rangle \in R_1, \langle v_1, v_2 \rangle \in R_2\}$$

Дефиниция 2.14. *Регулярен израз* наричаме дума над крайна азбука $\Sigma \cup \{(,), |, *\}$

- ϵ е регулярен израз.

- Ако $a \in \Sigma$, то a е регулярен израз.
- Ако E_1, E_2 са регулярни изрази, то $E_1|E_2$ и E_1E_2, E_1^* също са регулярни изрази.
- Не съществуват други регулярни изрази.

Всеки регулярен израз има сътветстващ регулярен език:

- $L(\epsilon) = \emptyset$
- $L(a) = \{a\}, a \in \Sigma$
- $L(E_1|E_2) = L(E_1) \cup L(E_2)$
- $L(E_1E_2) = L(E_1) \cdot L(E_2)$
- $L(E_1^*) = L(E_1)^*$

Пример 2.3. `(a|b)*c` е *регулярен израз*, разпознаващ думите *c, ac, bc, aac, abc, abbc, bababbc ...*

Теорема 2.2. *(Клини)* За всеки регулярен израз E съществува краен автомат $\mathcal{A}$, за който $L(E) = L(\mathcal{A})$.

Теорема 2.3. Всяка двоична регулярна стрингова релация може да се представи, чрез класически краен преобразувател.

Пример 2.4. $R = \{\langle ab, d\rangle, \langle bc, d\rangle\}^*$ е *регулярна релация*, която е представена чрез чрез крайния преобразувател $\mathcal{T}$ на Фигура 4.
С *dom(R)* бележим домейна на R, който изразяваме чрез *подлежащия автомат* на $\mathcal{T}$, $dom(R) = \{ab, bc, abab, abbc, bcab \dots\}$. С *range(R)* обозначаваме кодомейна на релацията $range(R) = \{\epsilon, d, dd, ddd, dddd \dots\}$.

Пример 2.5. Нека L е *регулярен език.* $Id(L) = \{\langle w, w\rangle \mid w \in L\}$ е *регулярна релация.*

Дефиниция 2.15. *Регулярна стрнгова функция* наричаме регулярна стрингова релация, която е частична функция.

2.4 Бимашини

Дефиниция 2.16. *Класическа бимашина* дефинираме като тройка $\mathcal{B} = \langle \mathcal{A}_L, \mathcal{A}_R, \psi \rangle$, където

- $\mathcal{A}_L = \langle \Sigma, Q_L, s_L, Q_L, \delta_L \rangle$ и $\mathcal{A}_R = \langle \Sigma, Q_R, s_R, Q_R, \delta_R \rangle$ са *детерминирани крайни автомати* и ги наричаме съответно *ляв и десен автомат* на бимашината. Всички състояния на тези автомати са финални.
- $\psi : (Q_L \times \Sigma \times Q_R) \to \Sigma^*$ е частична функция, която наричаме *изходна функция*.

Дефиниция 2.17. Нека $\mathcal{B} = \langle \mathcal{A}_L, \mathcal{A}_R, \psi \rangle$ е класическа бимашина, Σ е азбуката на на автоматите $\mathcal{A}_L$ и $\mathcal{A}_R$ и $w = a_1 a_2 \dots a_k \in \Sigma^*$ $(k \geq 0)$ е дума и $a_i \in \Sigma$ $(1 \leq i \leq k)$ са букви. Ако $\delta_L^*(a_1 a_2 \dots a_k)$ и $\delta_R^*(a_k a_{k-1} \dots a_1)$ са дефинирани, то можем да получим двата пътя:

$$\pi_L = l_0 \to^{a_1} l_1 \to^{a_2} \dots l_{k-1} \to^{a_k} l_k$$

$$\pi_R = r_0 \leftarrow^{a_1} r_1 \leftarrow^{a_2} \dots r_{k-1} \leftarrow^{a_k} r_k$$

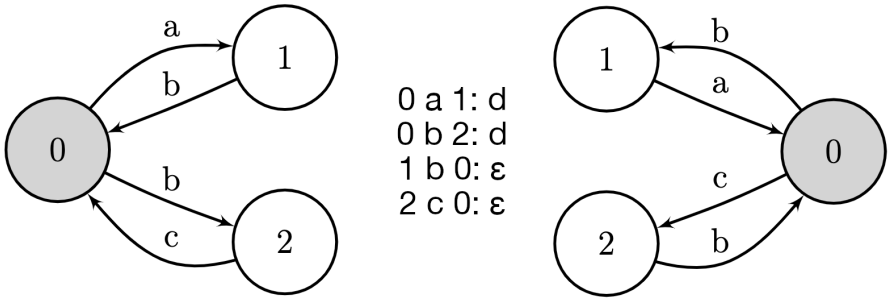
Където π_L и π_R са пътища в съответно левия и десния автомат и думата w се разпознава от $\mathcal{A}_L$ в посока от ляво на дясно, а от $\mathcal{A}_R$, съответно от дясно на ляво. Ако за всички тройки $\langle l_{i-1}, a_i, r_i \rangle$, изходната функция $\psi(l_{i-1}, a_i, r_i)$ е дефинирана, то двойката пътища $\langle \pi_L, \pi_R \rangle$ наричаме *успешно изпълнение* на $\mathcal{B}$ с етикет $w = a_1 a_2 \dots a_k$ и *изход*

$$\mathcal{O}_\mathcal{B}(w) = \psi(l_0, a_1, r_1) \cdot \psi(l_1, a_2, r_2) \cdot \dots \cdot \psi(l_{k-1}, a_k, r_k)$$

$\mathcal{O}_\mathcal{B}$ наричаме *изходна функция на бимашината* и казваме, че бимашината *превежда* w в m, ако $\mathcal{O}_\mathcal{B}(w) = m$, където m е резултат от конкатенацията на всички $\psi(l_{i-1}, a_i, r_i)$ $(1 \leq i \leq k)$.

Бимашината чете входната дума и за всеки символ извежда дума над азбуката си. На всяка стъпка изведената от изходната функция ψ дума зависи от входния символ и двете състояния в които биха преминали левият и десният автомат, четейки входа съответно от ляво на дястно и от дясно на ляво. Крайният резултат е конкатенацията на всички така изведени думи.

Дефиниция 2.18. Нека $\mathcal{B} = \langle \mathcal{A}_L, \mathcal{A}_R, \psi \rangle$ е бимашина. *Разширената изходна функция* ψ^* дефинираме индуктивно:



Фигура 5: Бимашина представяща $\{\langle ab, d\rangle, \langle bc, d\rangle\}^*$

- $\psi^*(l, \epsilon, r) = \epsilon$ за всяко $l \in Q_L, r \in Q_R$
- $\psi^*(l, wa, r) = \psi^*(l, w, \delta_R(r, a)) \cdot \psi(\delta_L^*(l, w), a, r)$, за $l \in Q_L, r \in Q_R, w \in \Sigma^*, a \in \Sigma$

Пример 2.6. *(Бимашина и изпълнение)* Нека разлгледаме бимашината на Фигура 5. Задаваме входна дума *bcabbc*, което води до следното изпълнение на левия и десния автомат съответно.

$$\pi_L = 0 \to^b 2 \to^c 0 \to^a 1 \to^b 0 \to^b 2 \to^c 0$$
$$\pi_R = 0 \leftarrow^b 2 \leftarrow^c 0 \leftarrow^a 1 \leftarrow^b 0 \leftarrow^b 2 \leftarrow^c 0$$

Изходната функция на бимашината $\mathcal{O}_\mathcal{B}$ прилагаме както следва:

$$\psi(0, b, 2) \cdot \psi(2, c, 0) \cdot \psi(0, a, 1) \cdot \psi(1, b, 0) \cdot \psi(0, b, 2) \cdot \psi(2, c, 0) = d \cdot \epsilon \cdot d \cdot \epsilon \cdot d \cdot \epsilon = ddd$$

Дефиниция 2.19. Бимашина $\mathcal{B} = \langle \mathcal{A}_L, \mathcal{A}_R, \psi\rangle$ наричаме *тотална*, ако функциите на прехода $\delta_L : Q_L \times \Sigma \to Q_L$ и $\delta_R : Q_R \times \Sigma \to Q_R$ на левия и десния автомат съответно, както и функцията на изхода $\psi : (Q_L \times \Sigma \times Q_R) \to \Sigma^*$ са *тотални*.

Теорема 2.4. Класическите бимашини са еквивалентни по изразителност на регулярните функции. [4]

3 Правила за заместване

За двоичните стрингови релации можем да си мислим като множество от преводи, като на пример за двойката $\langle u, v \rangle$ казваме, че думата u се превежда като v.

Дефиниция 3.1. *Правило за заместване* представяме във вида

$$E \to \beta$$

където E е регулярен израз над крайна азбука Σ, а $\beta \in \Sigma^*$ е дума. *Приложение на правилото върху текст* $t \in \Sigma^*$ представлява заместването на поднизовете на t, които са в езика $L(E)$ с β.

$$a_1 a_2 \ldots \overbrace{\underbrace{a_i \ldots a_{i+k}}_{\beta}}^{\in L(E)} \ldots \overbrace{\underbrace{a_j \ldots a_{j+l}}_{\beta}}^{\in L(E)} \ldots a_n$$

Фигура 6: Приложение на правило на заместване

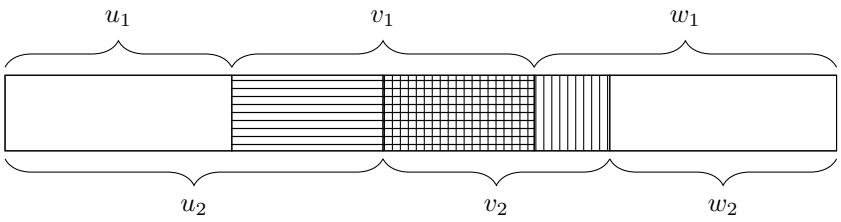
Правила за заместване можем да представим като *регулярни стрингови релации*, използвайки единствено алгебрата на регулярните езици и релации. Те се реализират програмно чрез крайни преобразуватели. [1]

Дефиниция 3.2. Нека разгледаме текст $t \in \Sigma^*$. *Контекст на заместване* наричаме тройка $\langle u, v, w \rangle$, където $t = uvw$. Също така u и w наричаме съответно *префикс* и *суфикс* на контекста, докато v наричаме *фокус*.

Пример 3.1. Нека разгледаме правилото `ab|bc` $\to d$. След приложението му над текста abb, ще получим db, като $\langle \epsilon, ab, b \rangle$ е единственият контекст на заместване. Приложението на правилото над $abbacbca$ ще доведе до $dbacda$ с два контекста на заместване $\langle \epsilon, ab, bacbca \rangle$ и $\langle abbac, bc, a \rangle$.

3.1 Разрешаване на многозначности

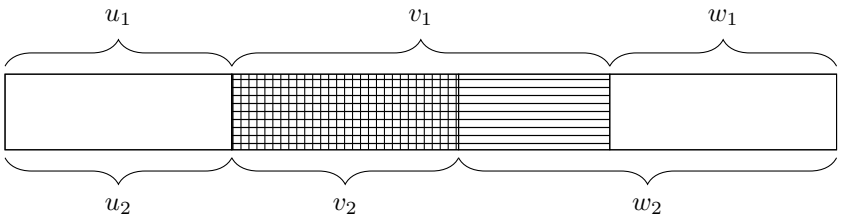
При прилагане на правило за заместване могат да възникнат многозначности в случаите, в които два контекста имат застъпващи се фокуси.



Фигура 7: Застъпващи се контексти

Дефиниция 3.3. Два контекста на заместване $\langle u_1, v_1, w_1\rangle$ и $\langle u_2, v_2, w_2\rangle$ за даден текст t *се застъпват*, ако $u_1 < u_2 < u_1v_1$. Израза $u_1 < u_2$ четем като u_1 е *префикс* на u_2.

Пример 3.2. Нека разгледаме правилото `ab|bc` $\rightarrow d$) приложено над текста $t = aabcb$. Получаваме контекстите $\langle a, ab, cb\rangle$ и $\langle aa, bc, b\rangle$, които очевидно се застъпват и съответно стигаме до две различни валидни заместания $adcb$ и $aadb$.



Фигура 8: Застъпващи се контексти с еднакво начало

Пример 3.3. Друг вид многозначност може да получим, когато фокусите на два контекста имат еднакво начало. Например, ако приложим правилото $a+ \rightarrow d$ над тескт $t = aa$, може получим превода dd, на който отговарят контекстите $\langle a, a, \epsilon\rangle$ и $\langle \epsilon, a, a\rangle$, и d с контекст $\langle \epsilon, aa, \epsilon\rangle$.

Дефиниция 3.4. Въвеждаме следните оператори над множества от контексти:

$$AFTER(A, B) = \{\langle u, v, w\rangle \in A \mid \forall\langle u', v', w'\rangle \in B : u' \cdot v' \leq u \wedge u' < u\}$$

$AFTER$ избира измежду всички контексти в множеството A тези, в които фокусът v започва след всички фокуси в множеството B.

$$LEFTMOST(A) = \{\langle u, v, w\rangle \in A \mid \forall \langle u', v', w'\rangle \in A : u \leq u'\}$$

$LEFTMOST$ избира измежду всички контексти в A, тези с чийто фокус се намира възможно най в ляво. Може да имаме повече от един такъв контекст.

$$LONGEST(A) = \{\langle u, v, w\rangle \in A \mid \forall \langle u', v', w'\rangle \in A : u \neq u' \vee v' \leq v\}$$

Измежду контекстите, чиито фокуси започват от една и съща позиция, $LONGEST$ избира тези, които имат най-дълъг фокус.

Дефиниция 3.5. Въвеждаме оператора $LML(A)$ (leftmost-longest), чрез който ще елиминираме многозначностите. По дадено множество от контексти A, $LML(A)$ избира най-левите, най-дълги измежду тях.

$$LML(A) := \bigcup_{i=0}^{\infty} C_i$$

Където междинните множества C_i строим по индукция:

$$C_0 = \emptyset$$
$$C_{i+1} = C_i \cup LONGEST(LEFTMOST(AFTER(A, C_i)))$$

$LML(A)$ е крайно множество, защото A е крайно, т.е. след дадем момент редицата ще престане да нараства.

Твърдение 3.1. Нека $t \in \Sigma^*$ е текст и A е множество от контексти в t. $LML(A)$ съдържа само незастъпващи се контексти.

Доказателство. Нека $LML(A)$ съдържа застъпващите се контексти $\langle u_1, v_1, w_1\rangle$ и $\langle u_2, v_2, w_2\rangle$ т.е. $u_1 < u_2 < u_1v_1$. Също така $\langle u_1, v_1, w_1\rangle \in C_{i_1}$ и $\langle u_2, v_2, w_2\rangle \in C_{i_2}$. Нека допуснем, че $i_1 \geq i_2$. В такъв случай:

$$\langle u_2, v_2, w_2\rangle \in C_{i_2} = C_{i_2-1} \cup LONGEST(LEFTMOST(AFTER(A, C_{i_2-1})))$$

След като $\langle u_2, v_2, w_2\rangle \notin C_{i_2-1}$, то $\langle u_2, v_2, w_2\rangle \in LEFTMOST((AFTER(A, C_{i_2-1}))$. От допускането, че $i_1 \geq i_2$ следва, че $\langle u_1, v_1, w_1\rangle \in AFTER(A, C_{i_2-1})$, което е в противоречие с дефиницията на $LEFTMOST$ и следователно $i_1 < i_2$.
След като $i_1 < i_2$, то $\langle u_2, v_2, w_2\rangle \notin C_i$ за $i \leq i_1$. При $i > i_1$, $\langle u_1, v_1, w_1\rangle \in C_i$, значи $\langle u_2, v_2, w_2\rangle \in AFTER(A, C_i)$, което не е възможно спрямо дефиницията на $AFTER$ т.е. $\langle u_2, v_2, w_2\rangle \notin C_{i+1}$. Това противоречи с факта, че $\langle u_2, v_2, w_2\rangle \in C_{i_2}$ $(i_1 < i_2)$, следователно $LML(A)$ не съдържа застъпващи се контексти. □

Дефиниция 3.6. Нека $T \subseteq \Sigma^* \times \Sigma^*$ е релация и $t \in \Sigma^*$ е текст. С $A_{dom(T)}$ бележим множеството на всички поднизове в t, които са в $dom(T)$. Нека

$$t = u_1 v_1 x_2 v_2 \dots x_k v_k w_k$$

е каноничното представяне на t за множеството $LML(A_{dom(T)})$. Тогава t' е *заместването* на t с T под стратегията най-ляво-най-дълго срещане

$$t = u_1 v'_1 x_2 v'_2 \dots x_k v'_k w_k$$

и $\langle v, v' \rangle \in T$ за $1 \leq i \leq k$. С $R^{LML}(T)$ бележим релацията на заместване под стратегията най-ляво-най-дълго срещане за T. Тя съдържа всички двойки $\langle t, t' \rangle \in \Sigma^* \times \Sigma^*$, така че t' е заместване на t с T под стратегията най-ляво-най-дълго срещане.

Следствие 3.1. Нека $T : \Sigma^+ \to \Sigma^*$ е функция, тогава релацията на заместване $R^{LML}(T)$ е функционална.

Пример 3.4. Нека разгледаме правилото `ab|bc` $\to d$ съответстващо на релацията $T = \{\langle ab, d \rangle, \langle bc, d \rangle\}$, приложено над текста $t = aabcbab$. Получаваме контекстите $\langle a, ab, cbab \rangle$, $\langle aa, bc, bab \rangle$ и $\langle aabcb, ab, \epsilon \rangle$. Очевидно първите два се застъпват, но стратегията най-ляво-най-дълго срещане определя първият и третият за валидни. Резултатът от заместането е $R^{LML}(T)(aabcbab) = adcbd$.

4 Лексически анализ чрез регулярни релации

За да разбием даден текст на редица от тоукъни, е нужно да представим тези тоукъни под формата на регулярни изрази. Това представяне наричаме *лексическа граматика.*

Пример 4.1. *Лексическа граматика и извличане на тоукъни.*

Token	**Description**
Number	`[0-9]+(\.[0-9]+)?`
Operator	`+|-|*|/`
Equal	`=`

Фигура 9: Лексическа граматика на аритметичен израз

Символите на думата *15+9-3=21* се групират в следната редица от тоукъни спрямо граматиката: `15`, `+`, `9`, `-` , `3`, `=`, `21`, докато за *+-**3232*, получаваме `+`, `-`, `*`, `*`, `3232`.

Регулярните релации намират своето приложение в множество домейни, като лексическият анализ е един от тях [3]. Идеята е да сведем процеса на токенизация до заместване на думи в текст, като от граматиката построим релация на заместване (под стратегията най-ляво-най-дълго срещане). Тази релация представяме програмни като краен преобразувател, който поставя маркер след всеки разпознат токън от лексическата граматика. В последствие от преобразувателят строим еквивалентна бимашина.

$$E \rightarrow \ldots \texttt{EoT}$$

Тази нотация представлява правило на заместване, което за дума в езика $L(E)$ на регулярния израз, добавя маркер в края ѝ (*end-of-token*). Със символа $\ldots$ означаваме, че разпознатата дума се замества със себе си (т.е. входът остава непроменен). Формално, такъв тип релаци по зададен регулярен получаваме израз като конкатенираме идентитета на неговия език със синглетон, който представлява заместване на празната дума с маркер за край на токън (end-of-token).

$$Id(L(E)) \cdot \{\langle \epsilon, \texttt{EoT} \rangle\}$$

Граматиката от Пример 4.1, представяме по следния начин:

1. Number: `[0-9]+` $\rightarrow \ldots$ `EoT`
2. Operator: `+|-|*|/` $\rightarrow \ldots$ `EoT`
3. Equal: `=` $\rightarrow \ldots$ `EoT`

Всяко правило в граматиката е регулярна релация като *релацията за лексически анализ* получаваме като обединим релациите, съответстващи на тези правила и от това обединение построим релация на заместване под стратегията *най-ляво-най-дълго* срещане.

$$R^{LML}(R_1 \cup R_2 \cup \ldots \cup R_n)$$

Както установихме, $R^{LML}(T)$ е функционална, ако T е функция. Тъй като една коректно дефинирана лексическа граматика не може да има две правила, които разпознават една и съща дума, то условието е изпълнено.
Нека се върнем на граматиката от Пример 4.1. Релацията за лексически анализ представяме по следния начин:

$$R := R^{LML}(R_{Num} \cup R_{Op} \cup R_{Eq})$$

$$R(42\text{-}15*5) = \texttt{"42 EoT - EoT 15 EoT * EoT 5 EoT"}$$

След като маркерите са коректно поставени, извличането на тоукъните става тривиално.

Работата на лексическият анализатор не се изчерпва с извличането на тоукъните от входния текст. Освен съдържанието му, често се нуждаем от информация и за неговия тип, както и позиция в текста. Типът е необходим при извършването на синтактичният анализ, докато позицията е полезна, когато съобщаваме за грешки във входния текст било то по време на синтактичния, или лексическия анализ.
Нека променим правилото на заместване, така че да включва и типа.

$$E \rightarrow \texttt{Type SoT}\ldots\texttt{EoT}$$

`Type` е индикатор, който носи информация за това кое правило на заместване е било приложено, докато `SoT` маркира началото на лексемата. Релациите за тези правила получаваме по следния начин:

$$\{\langle \epsilon, \texttt{Type SoT} \rangle\} \cdot Id(L(E)) \cdot \{\langle \epsilon, \texttt{EoT} \rangle\}$$

Съответно граматиката от Пример 4.1 придобива следния вид:

1. Number: `@1 SoT [0-9]+` $\rightarrow \ldots$ `EoT`
2. Operator: `@2 SoT +|-|*|/` $\rightarrow \ldots$ `EoT`
3. Equal: `@3 SoT =` $\rightarrow \ldots$ `EoT`

Релацията за лексически анализ R получаваме по същия начин, но изходната дума вече съдържа повече информация.

$$R(999+1) =$$

$$\texttt{"@1 SoT 999 EoT @2 SoT + EoT @1 SoT 1 EoT"}$$

5 Реализация

5.1 Дизайн

5.2 Парсер на регулярен израз

5.3 Конструкция на бимашина

5.4 Процедура за токенизация

Литература

[1] Kaplan, Ronald and Kay, Martin. 1994. *Regular models of phonological rule systems.* Computational Linguistics 20(3):331-378

[2] Gerdemann, Dale and van Noord, Gertjan. 1999. *Transducers from rewrite rules with backreferences* EACL '99: Proceedings of the ninth conference on European chapter of the Association for Computational Linguistics June 1999 Pages 126–133

[3] Karttunen, Lauri. 1996. *Directed Replacement.*

[4] Schützenberger, M.-P. 1961. *A remark on finite transducers.* Information and Control, 4:185–196.