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.

В тази работа представяме разработка на алгоритъм, който по зададено множество от спецификации на лексеми, създава лексически анализатор. Анализаторът извлича лексимите за линейно време спрямо дължината на входния текст, използвайки бимашина.

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 в текст, както и съобщаване на грешки.

1.1 Мотивация

1.2 Обобщение

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 $\langle q, a, q_2 \rangle \in \Delta$

Фигура 1: Недетерминиран краен автомат

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.

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 ($L(\mathcal{A}_1) = L(\mathcal{A}_2)$)

2.2 Детерминирани крайни автомати

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 - $O(|w|)$.

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 $a \in \Sigma, w \in \Sigma^*$

asd

Фигура 2: Детерминиран краен автомат

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

Доказателство. Строим *детерминиран краен автомат* $\mathcal{A}_D = \langle \Sigma, 2^Q, I, F_D, \delta \rangle$, където

- $F_D = \{S \in 2^Q \mid S \cap F \neq \emptyset\}$
- $\delta(S, a) = \{q \in Q \mid \exists q_1 \in S : \langle q_1, a, q \rangle \in \Delta\}$

□

2.3 Крайни преобразуватели

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

- Σ, Γ 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*
- $\Delta \subseteq Q \times (\Sigma^* \times \Gamma^*) \times Q$ 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 $q_1 \rightarrow_m^w q_2$.

Фигура 3: Краен преобразувател

Дефиниция 2.9. TODO: *Път и успешен път в краен преобразувател*

Дефиниция 2.10. TODO: *Подлежащ автомат на краен преобразувател*

2.4 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 $L_1, L_2 \in \Sigma^*$ 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

Дефиниция 2.12. *Регулярна n-орна стрингова релация* дефинираме индуктивно като множество от n-орки над крайни азбуки $\Sigma_1, \ldots, \Sigma_n$:

- $\emptyset$ е регулярна релация.
- Ако $a \in \Sigma_1 \times ... \times \Sigma_n$, 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. *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 $E_1 \mid E_2$ и E_1E_2, E_1^* също са регулярни изрази.
- Не съществуват други регулярни изрази.

Всеки регулярен израз има сътветстващ регулярен език:

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

Пример 2.1. TODO: Пример за регулярен израз

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 регулярни стрингова релациа можемогат да се изрази, чрез класически краен преобразувател.

Дефиниция 2.14. Регулярна стрнгова функция наричаме регулярна стрингова релация, която е частична функция.

Пример 2.2. TODO: Пример за регулярна стрингова релация

2.5 Свойства на регулярните езици и релации

2.6 Бимашини

Дефиниция 2.15. *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 2.16. 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 $w = a_1 a_2 \dots a_k \in \Sigma^*$ $(k \geq 0)$, 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*

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

$\mathcal{O}_{\mathcal{B}}$ 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 конкатенация 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.

Пример 2.3. *(Бимашина и изпълнение)* TODO: Пример бимашина

Дефиниция 2.17. Нека $\mathcal{B} = \langle \mathcal{A}_L, \mathcal{A}_R, \psi \rangle$ е бимашина. *Разширената изходна функция* ψ^* дефинираме индуктивно:

- $\psi^*(l, \epsilon, r) = \epsilon$ 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 $l \in Q_L, r \in Q_R, w \in \Sigma^*, a \in \Sigma$

Дефиниция 2.18. 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*.

3 Правила за заместване чрез регулярни релации

Дефиниция 3.1. Нека Σ_1, Σ_2 са крайни азбуки. *Двоична стрингова релация* $R \subseteq \Sigma_1^* \times \Sigma_2^*$ наричаме множество от двойки $\langle u, v \rangle$, където $u \in \Sigma_1^*$ и $v \in \Sigma_2^*$.

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

Дефиниция 3.2. 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 3.3. *Двоична стрингова релация* $R \subseteq L_1 \times L_2$ е *регулярна*, ако L_1 и L_2 са регулярни езици.

Дефиниция 3.4. *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 4: Приложение на правило на заместване

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

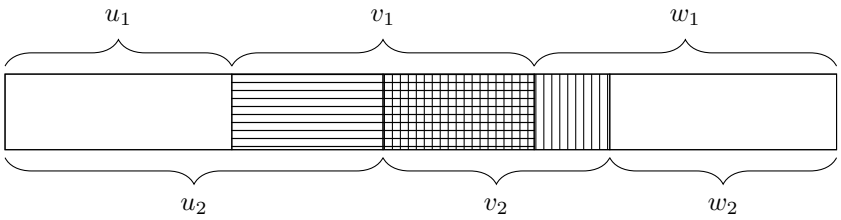
Дефиниция 3.5. Нека разгледаме текст $t \in \Sigma^*$. *Контекст на заместване* наричаме тройка $\langle u, v, w \rangle$, където $t = uvw$ и $u, w \in \Sigma^*$. Също така u и w наричаме съответно *префикс* и *суфикс* на контекста, докато v наричаме *фокус*.

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 единствения 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

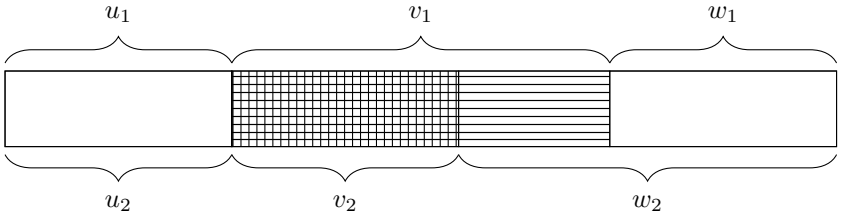
При прилагане на правило за заместване могат да възникнат многозначности в случаите, в които два контекста имат застъпващи се фокуси.

Дефиниция 3.6. 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 $u_1 < u_2 < u_1 v_1$. Израза $u_1 < u_2$ четем като u_1 е *префикс* на u_2.



Фигура 5: Застъпващи се контексти

Пример 3.2. Нека разгледаме правилото $ab|bc \to d$ 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 6: 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 $a^+ \to d$ 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 3.7. *Дефинираме следните функции над множества от контексти. OVER, MOST, LONGEST*

Дефиниция 3.8. *Множество на най-левите и най-дълги контексти* LML(A)

Твърдение 3.1. LML(A) съдържа само незастъпващи се контексти.

Доказателство. ... □

Дефиниция 3.9. *Приложение на правило за заместване над дума в текст под стратегията LML*

Следствие 3.1. Ако релацията е функционална, то LML е функция.

3.2 Токенизация чрез регулярни релации

Токенизираща релация за даден език дефинираме като комбинация на *правила за заместване*. Тези правила биват два вида - такива, които *маркират* дадена лексема, или *нормализиращи правила*, които служат за предварителна обработка входния текст, като на пример за премахване на излишни символи.

Пример 3.4. *Лексическа граматика на аритметичен израз от естествени числа.*

1. `WHITESPACE+` $\rightarrow \epsilon$
2. `SoT [0-9]+` $\rightarrow \ldots$ `EoT`
3. `SoT +|-|*|/` $\rightarrow \ldots$ `EoT`

Правила 2 и 3 маркират лексемите съответно за естествено число и аритметичен оператор като поставят символ за граница на лексема `TB`. Правило 1 е нормализиращо правило, което премахва интервалите от входната дума, тъй като те не носят значение за аритметичния израз.

Всяко от тези правила само по себе си представлява регулярна релация, като релацията за лексически анализ в конкретния случай получаваме като резултат от композицията на нормализиращото правило с обединението на лексическите правила. С други думи, ако всяко правило i представим като регулярна релация R_i, токенизиращата релация под стратегията най-ляво-най-дълго срещане можем да представим както следва

$$R = R^{LML}(R_1) \circ R^{LML}(R_2 \cup R_3)$$

Стратегията най-ляво-най-дълго срещане приложена над функционални релации, каквито са правилата в лексическата граматика, гарантира, че получената релация също е функционална. Това означава, че процесът на токенизация винаги ще има най-много един резултат.

$$R(\texttt{"2 + 14"}) = \texttt{"2 TB + TB 14 TB"}$$

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

Теорема 3.1. TODO: Еквивалентност на регулярните стрингови функции и бимашините ...

4 Реализация

4.1 Дизайн

4.2 Парсер на регулярен израз

4.3 Конструкция на бимашина

4.4 Процедура за токенизация

Литература

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

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