

DFA =(<i>Q</i> , <i>Σ</i> , <i>δ</i> , <i>q</i> ₀ , <i>F</i>)
<i>Q</i> skończony zbiór stanów
<i>Σ</i> skończony alfabet wejściowy
<i>δ</i> funkcja przejścia postaci <i>Q</i> × <i>Σ</i> → <i>Q</i>
<i>q</i> ₀ stan początkowy
<i>F</i> ⊆ <i>Q</i> zbiór stanów akceptujących
Minimalizacja DFA
1. forall p końcowy, q niekończowy, oznacz (p,q)
2. forall (<i>p</i> , <i>q</i>) ∈ (<i>F</i> × <i>F</i>) ∪ (<i>Q</i> \ <i>F</i> × <i>Q</i> \ <i>F</i>), <i>p</i> ≠ <i>q</i> if ∃ _{<i>a</i> ∈ <i>Σ</i>} (<i>δ</i> (<i>p</i> , <i>a</i>), <i>δ</i> (<i>q</i> , <i>a</i>)) jest oznaczona, oznacz (p,q) (rekurencyjnie).
3. nieoznaczone scalamy.
PDA <i>M</i> = (<i>Q</i> , <i>Σ</i> , <i>Γ</i> , <i>δ</i> , <i>q</i> ₀ , <i>Z</i> ₀ , <i>F</i>)
<i>Q</i> skończony zbiór stanów
<i>Σ</i> alfabet wejściowy
<i>Γ</i> alfabet stosowy
<i>q</i> ₀ ∈ <i>Q</i> stan początkowy
<i>Z</i> ₀ ∈ <i>Γ</i> symbol początkowy na stosie
<i>F</i> ⊂ <i>Q</i> zbiór stanów akceptujących (jeśli <i>F</i> = ∅ to akceptujemy przez pusty stos)
<i>δ</i> funkcja przejścia postaci <i>δ</i> : <i>Q</i> × (<i>Σ</i> ∪ { <i>ε</i> }) × <i>Γ</i> → 2 ^{<i>Q</i> × <i>Γ</i> *}
LOP Zał., że <i>L</i> regularny. Wtedy istnieje stała <i>n</i> , że jeśli <i>z</i> ∈ <i>L</i> oraz <i>z</i> ≥ <i>n</i> , to można podzielić <i>z</i> na <i>z</i> = <i>uvw</i> takie, że:
1. <i>v</i> ≥ 1
2. <i>uv</i> ≤ <i>n</i>
3. ∀ _{<i>i</i> ∈ <i>N</i>} <i>z</i> ' = <i>uv</i> ^{<i>i</i>} <i>w</i> ∈ <i>L</i>
Podział <i>α</i> = <i>uvw</i> , <i>uv</i> ≤ <i>n</i> oraz <i>v</i> ≥ 1. Wybieramy <i>i</i> dla którego <i>uv</i> ^{<i>i</i>} <i>w</i> ∉ <i>L</i> a powinien.
LOP bezk. Zał., że <i>L</i> bezkontekstowy.Wtedy istnieje stała <i>n</i> , że jeśli <i>z</i> ∈ <i>L</i> oraz <i>z</i> ≥ <i>n</i> , to można podzielić <i>z</i> na <i>z</i> = <i>uvwxy</i> , takie, że:
1. <i>vx</i> ≥ 1
2. <i>vwx</i> ≤ <i>n</i>
3. ∀ _{<i>i</i> ∈ <i>N</i>} <i>z</i> ' = <i>uv</i> ^{<i>i</i>} <i>wx</i> ^{<i>i</i>} <i>y</i> ∈ <i>L</i>
Lemat Ogdena Niech <i>L</i> język bezkontekstowy. Wtedy istnieje stała <i>n</i> taka, że jeśli <i>z</i> ∈ <i>L</i> oraz <i>z</i> ≥ <i>n</i> i oznaczmy w <i>z</i> <i>n</i> lub więcej pozycji jako wyróżnione, to można podzielić <i>z</i> na <i>z</i> = <i>uvwxy</i> takie, że:
1. <i>v</i> i <i>x</i> zawierają łącznie co najmniej jedną wyróżnioną pozycję
2. <i>vwx</i> zawiera co najwyżej <i>n</i> wyróżnionych pozycji
3. ∀ _{<i>i</i> ∈ <i>N</i>} <i>z</i> ' = <i>uv</i> ^{<i>i</i>} <i>wx</i> ^{<i>i</i>} <i>y</i> ∈ <i>L</i>
Klasa języków regularnych jest domknięta na operację sumy, dopełnienia, przecięcia, złożenia i domknięcia Kleene’ego. Gramatyka bezkontekstowa G=(<i>N</i> , <i>T</i> , <i>P</i> , <i>S</i>)
<i>N</i> - skończony zbiór zmiennych (nieterminale)
<i>T</i> - skończony zbiór zmiennych końcowych(terminale, alfabet)
<i>P</i> - skończony zbiór produkcji postaci A → <i>α</i> gdzie A ∈ <i>N</i> i <i>α</i> ∈ (<i>N</i> ∪ <i>T</i>)*

<i>S</i> ∈ <i>N</i> - symbol początkowy
Postać normalna Chomsky’ego postaci: <i>A</i> → <i>BC</i> albo <i>A</i> → <i>a</i> Konstrukcje:
1. If po prawej terminal <i>a</i> to zastępujemy go <i>C</i> _{<i>a</i>} i dopisujemy <i>C</i> _{<i>a</i>} → <i>a</i>
2. If prawa strona dłuższa niz 1 to zastępujemy <i>A</i> → <i>B</i> ₁ . . . <i>B</i> _{<i>n</i>} przez <i>A</i> → <i>B</i> ₁ <i>D</i> ₁ , <i>D</i> ₁ → <i>B</i> ₂ <i>D</i> ₂ , . . . , <i>D</i> _{<i>n</i>-2} → <i>B</i> _{<i>n</i>-1} <i>B</i> _{<i>n</i>}
FIRST(X) - dla symboli
1. X-terminal, to FIRST(X)=X
2. X→ <i>ε</i> to do FIRST(X) dodajemy <i>ε</i>
3. X - nieterminal i <i>X</i> → <i>Y</i> ₁ <i>Y</i> ₂ ... <i>Y</i> _{<i>k</i>} to dodajemy <i>a</i> do <i>FIRST(X)</i> jeśli istnieje <i>i</i> takie, że <i>a</i> ∈ <i>FIRST(Y</i> _{<i>i</i>}) oraz <i>ε</i> ∈ <i>FIRST(Y</i> _{<i>j</i>}) dla każdego <i>j</i> < <i>i</i> . <i>ε</i> ∈ <i>FIRST(X)</i> jeśli należy do wszystkich <i>FIRST(Y</i> _{<i>i</i>}).
4. <i>FIRST(Xα)</i> = <i>FIRST(X)</i> gdy <i>ε</i> ∉ <i>FIRST(X)</i>
5. <i>FIRST(Xα)</i> = <i>FIRST(X)</i> ∪ <i>FIRST(α)</i> gdy <i>ε</i> ∈ <i>FIRST(X)</i>
FOLLOW(A) - dla nieterminali
1. Dla początkowego <i>S</i> do <i>FOLLOW(S)</i> dodajemy \$
2. Jeśli mamy produkcję <i>A</i> → <i>αBβ</i> to do <i>FOLLOW(B)</i> dodajemy wszystkie symbole z <i>FIRST(β)</i> poza <i>ε</i>
3. Jeśli <i>A</i> → <i>αB</i> lub <i>A</i> → <i>αBβ</i> , gdzie <i>ε</i> ∈ <i>FIRST(β)</i> to do <i>FOLLOW(B)</i> dodajemy wszystkie symbole z <i>FOLLOW(A)</i>
LL(1) - <i>A</i> → <i>α</i> Tabela: nazwy kolumn terminale i \$!!!⌈ nazwy wierszy nieterminale ⇔ Najpierw usuń lewostronną rekursję i zrób faktoryzacje
1. ∀ produkcji <i>A</i> → <i>α</i> z gramatyki wykonaj 2 i 3
2. foreach <i>a</i> ∈ <i>T</i> if <i>a</i> ∈ <i>FIRST(α)</i> to wpisz <i>A</i> → <i>α</i> do <i>M</i> [<i>A</i> , <i>a</i>]
3. if <i>ε</i> ∈ <i>FIRST(α)</i> to dla każdego <i>b</i> ∈ <i>FOLLOW(A)</i> wpisz <i>A</i> → <i>α</i> do <i>M</i> [<i>A</i> , <i>b</i>] Jeżeli <i>ε</i> ∈ <i>FIRST(α)</i> oraz \$ ∈ <i>FOLLOW(A)</i> , dodaj <i>A</i> → <i>α</i> do <i>M</i> [<i>A</i> , \$]
4. PROTIP: nie ma w tabeli <i>ε</i> !
SLR Tabela: nazwy kolumn AKCJE (terminale i \$!!!) i PRZEJŚCIA (nieterminale) nazwy wierszy stany
1. zbiory sytuacji <i>C</i> = <i>I</i> ₀ , . . . , <i>I</i> _{<i>n</i>} Zaczynamy od <i>I</i> ₀ = <i>domkniecie</i> ([<i>S</i> ' → • <i>S</i>])
2. tabelka + redukcje (zaznaczyć ew. konflikty) konstrukcja tabelki: w części akcji <i>s</i> _{<i>x</i>} (shift) i <i>r</i> _{<i>x</i>} (reduce), a w części przejść (nieterminale) <i>x</i> (liczba) ACC dla <i>S</i> ' → <i>S</i> •.
3. redukcja do FOLLOW(A) (if redukcja była z <i>A</i> → <i>β</i> •) tzn. jak jest kropka na końcu to do tabeli dodajemy <i>r</i> _{<i>x</i>} , gdzie <i>x</i> to numer produkcji
LR(1)
1. zbiory sytuacji z PODGLĄDEM
2. podgląd początkowy \$

3. podgląd przy domknięciu: mamy [<i>A</i> → <i>α</i> • <i>Bβ</i> , <i>a</i>] ∈ <i>I</i> dla każdej produkcji z <i>B</i> → <i>γ</i> dodaj [<i>B</i> → • <i>γ</i> , <i>FIRST(βa)</i>]
4. tabelka jak SLR ale zamiast redukcja do FOLLOW(A) (if redukcja była z <i>A</i> → <i>β</i> •) to redukcja do elementów z podglądu

LALR

1. generujemy rodzinę <i>C</i> = <i>I</i> ₀ , . . . , <i>I</i> _{<i>n</i>} jak w LR(1)
2. sklejamy jądra nie patrząc na podgląd, a podglądy łączymy - tabelka analogicznie do LR(1)

LEADING(A)-pierwsze term. z A

1. <i>a</i> ∈ <i>LEADING(A)</i> jeśli mamy produkcję <i>A</i> → <i>Baβ</i> lub <i>A</i> → <i>aβ</i>
2. if exists prod. <i>A</i> → <i>Bα</i> i <i>a</i> ∈ <i>LEADING(B)</i> to <i>a</i> ∈ <i>LEADING(A)</i>
3. foreach nieterminali liczymy 1 i powtarzamy 2 aż nic się nie zmienia

TRAILING(A)-ostatnie term. z A

1. <i>a</i> ∈ <i>TRAILING(A)</i> jeśli mamy produkcję <i>A</i> → <i>βaB</i> lub <i>A</i> → <i>βa</i>
2. if exists prod. <i>A</i> → <i>αB</i> i <i>a</i> ∈ <i>TRAILING(B)</i> to <i>a</i> ∈ <i>TRAILING(A)</i>
3. foreach nieterminali liczymy 1 i powtarzamy 2 aż nic się nie zmienia

Tab. priorytetów ≡ <>

TT *T* ≡ *T*

TNT *T* ≡ *T*

TN foreach *a* ∈ *LEADING(N)* do *T* < *a* (wiersze) ⇔

NT foreach *a* ∈ *TRAILING(N)* do *a* > *T* (kolumny) ⌈

\$ zawsze gorszy

Zbiory sytuacji

1. Wzbogacenie <i>S</i> ' → <i>S</i>
2. Ponumerować produkcje (do redukcji!!!).
3. Idziemy od góry z wygenerowanych, jeśli mamy jakąś sytuację <i>A</i> → <i>α</i> • <i>Sβ</i> (kropka nie na końcu) to generujemy d(<i>I</i> _{<i>teraz</i>} , <i>S</i>)
<i>E</i> → <i>ε</i> ⇒ <i>E</i> → .
4. W ostateczności musimy dojść z każdą produkcją z kropką na koniec

Rekurencja

1. <i>A</i> → <i>Aα</i> <i>β</i>
2. <i>A</i> → <i>βA</i> '

A' → *αA*'|*ε*

Faktoryzacja

1. <i>A</i> → <i>αβ</i> ₁ ... <i>αβ</i> _{<i>k</i>}
2. <i>A</i> → <i>αA</i> '
<i>A</i> ' → <i>β</i> ₁ ... <i>β</i> _{<i>k</i>}

język	lem	słowo	notes
$L = \{uvv^Rw : u, v, w \in \{0, 1, 2\}^* v \neq \varepsilon\}$	reg	dwa te same sym obok	
$L = \{wvx : w \in \{0, 1, 2\}^* x \in \{0, 1\}^* w > 0\}$	Ogd	$21^n 0^n 221^n 0^n 2$	
$L = \{wxw^R : w \in \{0, 1, 2\}^* x \in \{0, 1\}^* w > 0\}$	reg	konczy sie tym czym zaczyna	
$L = \{a^n b^k a^n, n \neq k\}$	Ogd	dwa warunki to za mało	
$L = \{a^i b^j c^k, i < j < k\}$	Ogd	nie bezkontekstowy	
$L = \{a^i b^j c^k, k = \max(i, j)\}$	lop/ogd	nie bezkontekstowy	
$L = \{a^i b^j c^k, i + j = k\}$	lop	bezkontekstowy	
$L = \{a^i b^j c^k, j = i + k\}$		bezkontekstowy	
$L = \{a^n b^m c^n, m \neq n\}$		nie jest bezkontekstowy	
$L = \{ww, w \in \{0, 1\}^*\}$		Nie bezkontekstowe, ale dopełnienie bezkontekstowe	
$L = \{w \in \{0, 1, 2\}^* w = \textit{palindrom } i \mid w _0 = w _2 \bmod 13\}$		bezkontekstowy, hybryda	
$L = \{w \in \{a, b, c\}^* w = \textit{palindrom } i \mid w _a = w _b i w _c > 0\}$	Ogden	$a^n b^n c b^n a^n$	
$\omega = xxy \wedge x \neq \varepsilon$	LOP	$ab^n ab^n$	$i = 0$
$\omega = xyz \wedge y \neq \varepsilon$	reg	$len \geq 4$	dobrac krótsze
$\omega \omega^R \wedge \omega _a \equiv \omega _b \equiv 0 \pmod{13}$	LOP	$a^{13n} b^{13n} b^{13n} a^{13n}$	ozn.
$\omega : \omega _a \equiv \omega _b \pmod{3}$	reg	mini	
$\omega = xyy^R \wedge y \neq \varepsilon$	reg	2 obok	
$\omega : \textit{palindrom } \wedge \omega _a = \omega _c$	LOP	$a^n c^n c^n a^n$	
$\omega = xcycz \wedge xy \text{ i } yz \in \{a, b\}^* \textit{palindromy}$	Ogd	$a^m b c a^m c b a^m$	śr. ozn.
$ \omega _a = \omega _b$	bezk.		
$ \omega _a = \omega _b = \omega _c$	LOP	$a^n b^n c^n$	
$\omega : \omega _a \neq \omega _b \neq \omega _c$	Ogd	$a^{m+m!} b^m a^{m+m!}$	ozn b.
$\omega : \omega _a = \omega _b = \omega _c$	LOP	$a^n b^n c^n$	i=0
$\omega : \omega _a = \omega _c > \omega _b$	LOP	$a^{n+1} b^n c^{n+1}$	
$\omega \omega \omega$	LOP	$0^n 1^n 0^n 1^n 0^n 1^n$	i=0
$\omega \omega^R \omega$	LOP	$0^n 1^n 1^n 0^n 0^n 1^n$	i=0
$a^n c^k b^n : n \neq k$	Ogd	$a^{n!+n} c^n b^{n!+n}$	

$L = \{w \in \{0, 1, 2\}^* w = \textit{palindrom } i \mid w|_0 = |w|_2 \bmod 13\}$ - hybryda palindromów (da sie zrobic automat) oraz przejście po stanach

$L = \{w \in \{a, b\}^* |w|_a = |w|_b + 5\} < \text{bezkontekstowy}$

$L = \{w \in \{a, b\}^* |w|_a = |w|_b = 2 \bmod 5 < \text{regularny}$

Majac $L = \{w \in \{a, b, c\}^* i |w|_a \neq |w|_b \textit{lub} |w|_a \neq |w|_c \textit{lub} |w|_b \neq |w|_c\}$ mamy niedeterministyczny automat ze stosem ktory zgaduje poprawny wynik.

$L_1 = \{w : w \in \{a, b, c\}^* \wedge |w|_a \neq |w|_b \neq |w|_c\}$
Niech n stała z lematu Ogdena. Niech m > n. Wybieramy słowo $z = a^{m+m!} b^m c^{m+m!}$ i oznaczamy m liter b jako wyróżnione.

- nie możemy pompować samego a ani samego c (brak wyróżnionych).
- nie możemy pompować jednocześnie a oraz c (pomiędzy nimi jest więcej niż n wyróżnionych liter).

Pozostają nam do rozpatrzenia podziały, w których:

- pompujemy b:
wyznaczamy i: $|vx|_b = p$
 $m + m! = m + (i - 1)p$
 $m! = ip - p$
 $i = \frac{m!}{p} + 1$

 $|z'|_b = |z|_b + (i - 1)|vx|_b = m + (\frac{m!}{p} + 1 - 1)p = m + m!$
Długość b jest taka sama jak długość reszty więc wyszliśmy z języka.
- pompowanie a i b. Równamy ilość b do ilości c.
pomowanie b i c. Równamy ilość b do ilości a.

FIRST

- Szukamy produkcji gdzie na początku stoi terminal i ten terminal dodajemy do zbioru FIRST od nieterminala przed strzałką.
- Szukamy produkcji z eps i dodajemy ten eps do zbioru FIRST od nieterminala przed strzałką.
- Szukamy produkcji gdzie na początku stoi nieterminal i FIRST od tego nieterminala dodajemy do FIRST od nieterminala stojącego przed strzałką (bez epsilon). Jeżeli w kopiowanym zbiorze jest epsilon to dodajemy FIRST od następnego symbolu. (Jeśli w każdym symbolu jest epsilon to na końcu dodajemy epsilon).

FOLLOW

- Do zbioru FOLLOW od symbolu początkowego dodajemy \$
- Szukamy produkcji gdzie za nieterminalem będzie stał jakiś symbol i do FOLLOW od tego nieterminala dodajemy FIRST od następnego symbolu (pomijając eps). Jeśli w dodawanym zbiorze był eps to sprawdzamy kolejny symbol.
- Szukamy produkcji gdzie na końcu znajduje się nieterminal i do FOLLOW od tego nieterminala kopiujemy zawartość FOLLOW od nieterminala przed strzałką.
- Szukamy produkcji gdzie za jakimś nieterminalem cała prawa strona będzie się zerowała (czyli w FIRST od całej strony będzie epsilon). Wtedy do FOLLOW od tego nieterminala dodajemy FOLLOW od nieterminala przed strzałką.
Powtarzaj 3 i 4 dopóki są zmiany.

LL FIRST(alfa) - FIRST od pierwszego znaku eps, jeśli jest epsilon to wchodzimy do kolejnego
bin(n)bin(n+1), n > 0 - nie jest bezkontekstowe
bin(n)bin(n+1)^R, n > 0 – bezkontekstowe
bin(n)hex(n)^R, n > 0 > korzystamy z odpowiedniosci bin >> hex i ucieczka od zernapoczatku