

Ćwiczenia 4

23.10

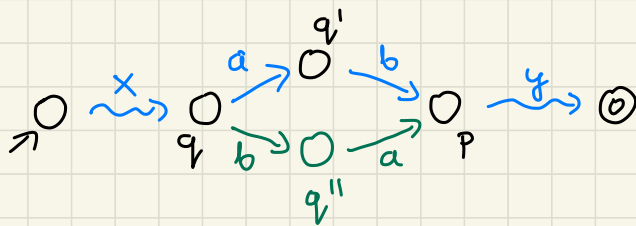
1. Dla danego alfabetu z zależnościami (Σ, D) i języka L rozważmy pytanie, czy L jest zamknięty na równoważność śladową. Pokaż, że pytanie to jest rozstrzygalne dla L regularnego. Udowodnij, że problem staje się nierozstrzygalny, gdy założymy, że L jest bezkontekstowy.

1) L regularny

* udowodniliśmy już, że jeśli regularny język L jest zamknięty na równoważność śladową, to minimalny automat deterministyczny rozpoznający L jest diamentowy

* pytanie: czy można otrzymać automat diamentowy, minimalizując automat rozpoznający język, który nie jest zamknięty na równoważność śladową (\equiv_D)? **NIE**

- * pokażemy, że każdy automat diamentowy A rozpoznaje język L_A zamknięty na \equiv_D
- * dla każdych słów $u \in L_A$ i $v \in [u]$, mając dany bieg akceptujący dla u w A , możemy wygenerować bieg akceptujący dla v
- * wystarczy pokazać, że jeśli akceptowane jest słowo $xaby$, $x, y \in \Sigma^*$, $(a, b) \in \bar{I}$, to akceptowane jest też $xbay$ (bo v otrzymujemy z u przez ciąg takich zamian)
- * powyższe wynika bezpośrednio z własności automatu diamentowego



przejście po b z q istnieje, bo automat jest deterministyczny (i ma stan „śmietnik”)

* stąd, żeby sprawdzić, czy L zamknięty na \equiv_D , wystarczy zminimalizować automat rozpoznający L i sprawdzić, czy jest diamentowy (w każdym stanie sprawdzamy przejścia po każdej parze $(a,b) \in I$)

2) L bezkontekstowy

* dla gramatyki bezkontekstowej problem sprawdzenia, czy generowany przez nią język zawiera wszystkie słowa nad alfabetem symboli terminalnych, jest nierozstrzygalny (problem uniwersalności gramatyki)

* pokazujemy nierozstrzygalność naszego problemu przez redukcję:

uniwersalność \leftarrow zamkniętość na \equiv_D

alfabet Σ

$\hat{\Sigma} = \Sigma \cup \{a,b\}$
 $I = \{(a,b), (b,a)\}$

gramatyka G

$\hat{G}: a \in G, b \in (\Sigma^*)$

język L

\hat{L} generowany przez \hat{G}

czy G uniwersalna? = czy \hat{L} zamknięty na \equiv_D ?

2. Pokaż, że produkty asynchroniczne automatów rozpoznają dokładnie języki prostokątne.

Przypomnienie teorii i rozwiązanie z książki:

Distributed alphabets A distributed alphabet over Σ , or a *distribution* of Σ , is a tuple of nonempty sets $\theta = \langle \Sigma_1, \Sigma_2, \dots, \Sigma_k \rangle$ such that $\bigcup_{1 \leq i \leq k} \Sigma_i = \Sigma$. For each action $a \in \Sigma$, the *locations* of a with respect to the distribution θ is the set $loc_\theta(a) = \{i \mid a \in \Sigma_i\}$. If θ is clear from the context, we write just $loc(a)$ instead of $loc_\theta(a)$.

↙ produkty asynchroniczne automatów

↖ $dom(a)$

Direct product automaton Let $\langle \Sigma_1, \Sigma_2, \dots, \Sigma_k \rangle$ be a distribution of Σ . For each $i \in [1..k]$, let $A_i = (Q_i, \rightarrow_i, Q_{in}^i, F_i)$ be an automaton over Σ_i . The *direct product automaton* $(A_1 \parallel A_2 \parallel \dots \parallel A_k)$ is the automaton $A = (Q, \rightarrow, Q_{in}, F)$ over $\Sigma = \bigcup_{1 \leq i \leq k} \Sigma_i$, where:

- $Q = Q_1 \times Q_2 \times \dots \times Q_k$.
- Let $\langle q_1, q_2, \dots, q_k \rangle, \langle q'_1, q'_2, \dots, q'_k \rangle \in Q$.
Then $\langle q_1, q_2, \dots, q_k \rangle \xrightarrow{a} \langle q'_1, q'_2, \dots, q'_k \rangle$ if
 - For each $j \in loc(a)$, $q_j \xrightarrow{a} q'_j$.
 - For each $j \notin loc(a)$, $q_j = q'_j$.
- $Q_{in} = Q_{in}^1 \times Q_{in}^2 \times \dots \times Q_{in}^k$.
- $F = F_1 \times F_2 \times \dots \times F_k$.

Direct product language Let $\langle \Sigma_1, \Sigma_2, \dots, \Sigma_k \rangle$ be a distribution of Σ . $L \subseteq \Sigma^*$ is said to be a *direct product language* if there is a direct product automaton $A = (A_1 \parallel A_2 \parallel \dots \parallel A_k)$ such that $L = L(A)$.

Direct product languages can be precisely characterized in terms of their projections onto the local components of the system.

Projections Let $\langle \Sigma_1, \Sigma_2, \dots, \Sigma_k \rangle$ be a distribution of Σ . For $w \in \Sigma^*$ and $i \in [1..k]$, the projection of w onto Σ_i is denoted $w \downarrow_{\Sigma_i}$ and is defined inductively as follows:

- $\varepsilon \downarrow_{\Sigma_i} = \varepsilon$, where ε is the empty string.
- $wa \downarrow_{\Sigma_i} = \begin{cases} (w \downarrow_{\Sigma_i})a & \text{if } a \in \Sigma_i \\ (w \downarrow_{\Sigma_i}) & \text{otherwise} \end{cases}$

Shuffle closure The shuffle closure of L with respect to $\langle \Sigma_1, \Sigma_2, \dots, \Sigma_k \rangle$, $shuffle(L, \langle \Sigma_1, \Sigma_2, \dots, \Sigma_k \rangle)$, is the set

$$\{w \in \Sigma^* \mid \forall i \in [1..k], \exists u_i \in L, w \downarrow_{\Sigma_i} = u_i \downarrow_{\Sigma_i}\}$$

As usual, we write just $shuffle(L)$ if $\langle \Sigma_1, \Sigma_2, \dots, \Sigma_k \rangle$ is clear from the context.

Proposition 1.1. Let $\langle \Sigma_1, \Sigma_2, \dots, \Sigma_k \rangle$ be a distribution of Σ and let $L \subseteq \Sigma^*$ be a regular language. L is a direct product language iff $L = shuffle(L)$.

↪ język prostokątny

Proof Sketch: (\Rightarrow) Suppose that L is a direct product language. It is easy to see that $L \subseteq shuffle(L)$, so we show that $shuffle(L) \subseteq L$. Since L is a direct product language, there exists a direct product automaton $A = (A_1 \parallel A_2 \parallel \dots \parallel A_k)$ such that $L = L(A)$.

Let $w \in shuffle(L)$. For each $i \in [1..k]$, there is a witness $u_i \in L$ such that $w \downarrow_{\Sigma_i} = u_i \downarrow_{\Sigma_i}$. Since $u_i \in L$, there is an accepting run $q \in Q_{in}^i \xrightarrow{u_i \downarrow_{\Sigma_i}} q_f \in F_i$ in A_i . Since this is true for every i , we can “glue” these runs together and construct an accepting run for A on w , so $w \in L(A) = L$.

(\Leftarrow) Suppose that $L = shuffle(L)$. We prove that L is a direct product language. For $i \in [1..k]$, $L_i = L \downarrow_{\Sigma_i}$ is a regular language, since homomorphic images of regular languages are regular. For each $i \in [1..k]$, there exists a deterministic automaton A_i such that $L_i = L(A_i)$. It is then easy to see that $L = L(A_1 \parallel A_2 \parallel \dots \parallel A_k)$. \square

Źródło: Madhavan Mukund. Automata on Distributed Alphabets

Dodatek o produktach asynchronicznych:

Proposition 1.2. Direct product languages are not closed under boolean operations.

Example 1.3.

Let $\theta = \langle \{a\}, \{b\} \rangle$ and let $L = \{ab, ba, aabb, abab, abba, baab, baba, bbaa\}$. Then L is clearly the union of $\{ab, ba\}$ and $\{aabb, abab, abba, baab, baba, bbaa\}$, both of which are direct product languages. However, L is not itself a direct product language because $L \neq shuffle(L)$. For instance, $abb \in shuffle(L) \setminus L$.

3. Udowodnij, że uogólnione produkty asynchroniczne rozpoznają dokładnie sumy języków prostokątnych.

Synchronized product automaton Let $\langle \Sigma_1, \Sigma_2, \dots, \Sigma_k \rangle$ be a distribution of Σ . For each $i \in [1..k]$, let $TS_i = (Q_i, \rightarrow_i, Q_{in}^i)$ be a transition system over Σ_i . The *synchronized product automaton* of $(TS_1, TS_2, \dots, TS_k)$ is an automaton $A = (Q, \rightarrow, Q_{in}, F)$ over $\Sigma = \bigcup_{1 \leq i \leq k} \Sigma_i$, where:

- $Q = Q_1 \times Q_2 \times \dots \times Q_k$
- Let $\langle q_1, q_2, \dots, q_k \rangle, \langle q'_1, q'_2, \dots, q'_k \rangle \in Q$.
Then $\langle q_1, q_2, \dots, q_k \rangle \xrightarrow{a} \langle q'_1, q'_2, \dots, q'_k \rangle$ if
 - For each $j \in \text{loc}(a)$, $q_j \xrightarrow{a} q'_j$.
 - For each $j \notin \text{loc}(a)$, $q_j = q'_j$.
- $Q_{in} = Q_{in}^1 \times Q_{in}^2 \times \dots \times Q_{in}^k$.
- $F \subseteq Q_1 \times Q_2 \times \dots \times Q_k$.

uogólniony produkt
asynchroniczny

Synchronized product language Let $\langle \Sigma_1, \Sigma_2, \dots, \Sigma_k \rangle$ be a distribution of Σ . $L \subseteq \Sigma^*$ is said to be a synchronized product language if there is a synchronized product automaton A such that $L = L(A)$.

Proposition 1.5. *A language is a synchronized product language if and only if it can be written as a finite union of direct product languages.*

Proof Sketch: (\Rightarrow) Let $A = (Q, \rightarrow, Q_{in}, F)$ be a synchronized product such that $\langle TS_1, TS_2, \dots, TS_k \rangle$ are the component transition systems over $\langle \Sigma_1, \Sigma_2, \dots, \Sigma_k \rangle$. For each $f = \langle f_1, f_2, \dots, f_k \rangle \in F$, extend TS_i to an automaton $A_i^f = (TS_i, f_i)$ and construct the direct product $A_f = (A_1^f \parallel A_2^f \parallel \dots \parallel A_k^f)$. Then, $L(A) = \bigcup_{f \in F} L(A_f)$.

(\Leftarrow) Conversely, let L be a finite union of direct product languages $\{L_i\}_{i \in [1..m]}$, where each L_i is recognized by a direct product $A^i = (A_1^i \parallel A_2^i \parallel \dots \parallel A_k^i)$. For $j \in [1..k]$, let $A_j^i = (Q_j^i, \rightarrow_j^i, Q_{in}^{ij}, F_j)$ be the j^{th} component of A^i . We construct a synchronous product $\hat{A} = (\hat{A}_1 \parallel \hat{A}_2 \parallel \dots \parallel \hat{A}_k)$ as follows. For each component j , we let \hat{Q}_j be the disjoint union $\bigsqcup_{i \in [1..m]} Q_j^i$ and define the set of initial states of component j be $\bigcup_{i \in [1..m]} Q_{in}^{ij}$. The local transition relations of each component are given by the union $\bigcup_{i \in [1..m]} \rightarrow_j^i$. The crucial point is to define the global set of final states as $(F_1^1 \times F_2^1 \times \dots \times F_k^1) \cup (F_1^2 \times F_2^2 \times \dots \times F_k^2) \cup \dots \cup (F_1^m \times F_2^m \times \dots \times F_k^m)$. This ensures that the synchronized product accepts only if all components agree on the choice of L_i . \square

Zadania domowe

(nieobowiązkowe)

VAS - vector addition system

- * nieujemny wektor początkowy $u \in \mathbb{N}^d$
- * skończony zbiór wektorów $V \subseteq \mathbb{Z}^d$ oznaczających "możliwe przejścia"; z wektora u można osiągnąć $u+v$ w jednym kroku, tylko jeśli $v \in V$ i $u+v \in \mathbb{N}^d$ (nie można spaść na żadnej współrzędnej wektora poniżej zera)
- * wektor w jest osiągalny z u , jeśli istnieje sekwencja kroków, którymi można przejść do w

VASS - vector addition system with states

- * graf skierowany (Q, T) , gdzie Q są stanami (wierzchołkami), a T to krawędzie etykietowane wektorami (formalnie $T \subseteq Q \times \mathbb{Z}^d \times Q$)
- * z konfiguracji początkowej $(p, u) \in Q \times \mathbb{N}^d$ można w jednym kroku osiągnąć $(q, u+v)$, jeśli $(p, v, q) \in T$ i $u+v \in \mathbb{N}^d$

Automat licznikowy bez testów 0

- * zbiór liczników (komórek przechowyujących nieujemne liczby całkowite)
- * skończony niedeterministyczny automat
- * w automacie przejścia są etykietowane akcją zmniejszenia / zwiększenia wybranego licznika o 1
- * gdy automat jest w stanie p i ma wychodzącą krawędź (p, dec_i, q) , a $c_i = 0$, to przejście jest niemożliwe

1.^D Zaproponuj transformację n -wymiarowego VASSu do równoważnego $(n+s)$ -wymiarowego VASu.

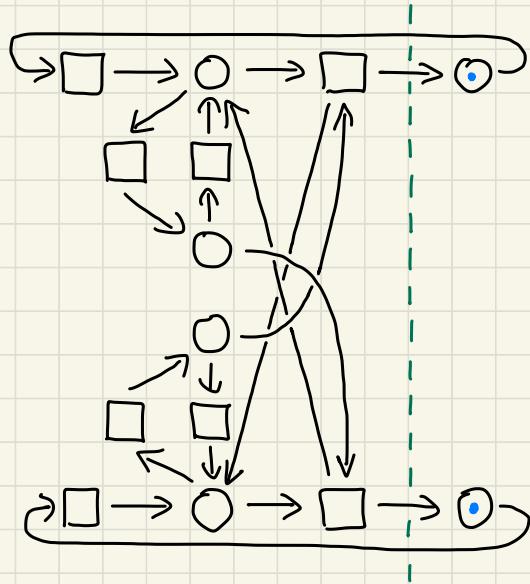
Jak może być s ?

2.^D O ile większą siłę mają automaty licznikowe z testami na zero? Pokaż, że można nimi zasymulować taśmę, maszyny Turinga.

Uwaga: na następnej stronie jest
rozwiązanie zadania domowego
2^D z poprzednich ćwiczeń

2^D. Ogólna sieć Petriego N z konfiguracją M jest żywa i 1-ograniczona. Czy wynika stąd, że z dowolnej osiągalnej konfiguracji M' można wrócić do M ?

* nie, wystarczy spojrzeć na kontrprzykład po prawej



- * w powyższej sieci jedynie początkowa konfiguracja ma dwa żetony po prawej stronie zielonej linii
- * cały czas zachowany jest niezmiennik: jeden żeton w górnej części sieci i jeden w dolnej
- * każde przeniesienie żetona na prawą stronę zielonej linii wymaga, aby drugi żeton był po lewej stronie