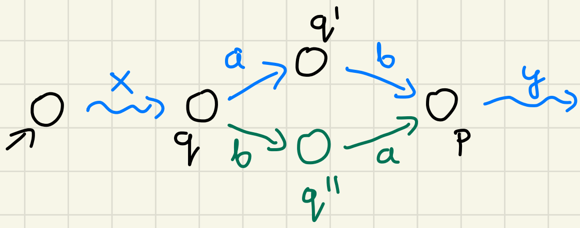
Ćwiczenia 4 2.11

1. Dla danego alfabetu z zależnością (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 $xaby$, $x, y \in \Sigma^*$, $(a,b) \in I$, 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 – w pesymistycznym przypadku prowadzi do stanu „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 $\leftsquigarrow$ zamkniętość na $\equiv_D$

alfabet Σ $\qquad$ $\hat{\Sigma} = \Sigma \uplus \{a,b\}$

$I = \{(a,b),(b,a)\}$

gramatyka G $\qquad$ $\hat{G}$: abG, $ba(\Sigma^*)$

język L $\qquad$ $\hat{L}$ 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 $\theta = \langle \Sigma_1, \Sigma_2, \ldots, \Sigma_k \rangle$ such that $\bigcup_{1 \le i \le k} \Sigma_i = \Sigma$. 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 $loc_\theta(a)$.

dom(a)

produkty asynchroniczne automatów

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

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

Direct product language Let $\langle \Sigma_1, \Sigma_2, \ldots, \Sigma_k \rangle$ be a distribution of Σ. $L \subseteq \Sigma^*$ is said to be a direct product language if there is a direct product automaton $A = (A_1 \parallel A_2 \parallel \cdots \parallel A_k)$ 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 $\langle \Sigma_1, \Sigma_2, \ldots, \Sigma_k \rangle$ 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.
- $wa{\downarrow_{\Sigma_i}} = \begin{cases} (w{\downarrow_{\Sigma_i}})a \text{ if } a \in \Sigma_i \\ (w{\downarrow_{\Sigma_i}}) \quad \text{otherwise} \end{cases}$

Shuffle closure The shuffle closure of L with respect to $\langle \Sigma_1, \Sigma_2, \ldots, \Sigma_k \rangle$, $\mathit{shuffle}(L, \langle \Sigma_1, \Sigma_2, \ldots, \Sigma_k \rangle)$, 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 $\mathit{shuffle}(L)$ if $\langle \Sigma_1, \Sigma_2, \ldots, \Sigma_k \rangle$ is clear from the context.

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

język prostokątny

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

Let $w \in \mathit{shuffle}(L)$. 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 $q \in Q^i_{\text{in}} \xrightarrow{u\downarrow_{\Sigma_i}}_i q_f \in F_i$ 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 $L = \mathit{shuffle}(L)$. 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 $L = L(A_1 \parallel A_2 \parallel \cdots \parallel A_k)$. $\square$

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 $L \neq \mathit{shuffle}(L)$. For instance, $abb \in \mathit{shuffle}(L) \setminus L$.

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

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

uogólniony produkt asynchroniczny

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

Synchronized product language Let $\langle \Sigma_1, \Sigma_2, \ldots, \Sigma_k \rangle$ 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 $A = (Q, \rightarrow, Q_{\text{in}}, F)$ be a synchronized product such that $\langle TS_1, TS_2, \ldots, TS_k \rangle$ are the component transition systems over $\langle \Sigma_1, \Sigma_2, \ldots, \Sigma_k \rangle$. For each $f = \langle f_1, f_2, \ldots, f_k \rangle \in F$, extend TS_i to an automaton $A_i^f = (TS_i, f_i)$ and construct the direct product $A_f = (A_1^f \parallel A_2^f \parallel \cdots \parallel A_k^f)$. 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 $A^i = (A_1^i \parallel A_2^i \parallel \cdots \parallel A_k^i)$. For $j \in [1..k]$, let $A_j^i = (Q_j^i, \rightarrow_j^i, Q_{\text{in}}^{i_j}, F_j)$ be the j^{th} component of A^i. We construct a synchronous product $\hat{A} = (\hat{A}_1 \parallel \hat{A}_2 \parallel \cdots \parallel \hat{A}_k)$ as follows. For each component j, we let $\widehat{Q}_j$ be the disjoint union $\biguplus_{i \in [1..m]} Q_j^i$ and define the set of initial states of component j be $\bigcup_{i \in [1..m]} Q_{\text{in}}^{i_j}$. 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 $(F_1^1 \times F_2^1 \times \cdots \times F_k^1) \cup (F_1^2 \times F_2^2 \times \cdots \times F_k^2) \cup \cdots \cup (F_1^m \times F_2^m \times \cdots \times F_k^m)$. This ensures that the synchronized product accepts only if all components agree on the choice of L_i. $\square$

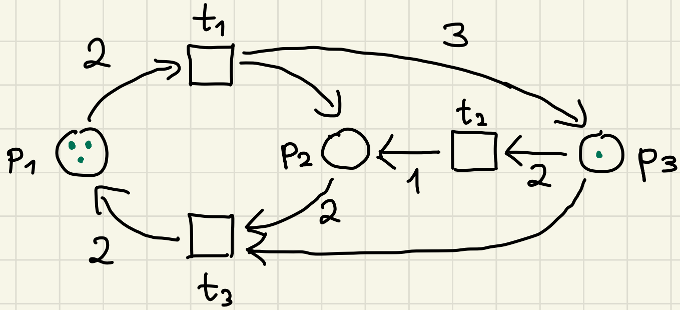
4. Modele równoważne: sieci Petriego, VAS, VASS i automaty licznikowe bez testów 0.

Pomysł: sieć Petriego $\rightsquigarrow$ VAS $\rightsquigarrow$ VASS $\rightsquigarrow$ automat licznikowy $\rightsquigarrow$ sieć Petriego

równoważność w znaczeniu osiągalności oraz pokrywalności

sieć Petriego $\rightsquigarrow$ VAS

* zakładamy, że nie ma ciasnych pętli



* symulacja VASem:

- wektor początkowy $u = (3, 0, 1)$ (początkowa konfiguracja)
- zbiór wektorów $V = \{t_1, t_2, t_3\}$, gdzie

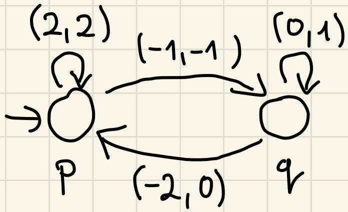
$t_1 = (-2, 1, 3)$, $t_2 = (0, 1, -2)$,

$t_3 = (2, -2, 1)$

VAS $\rightsquigarrow$ VASS

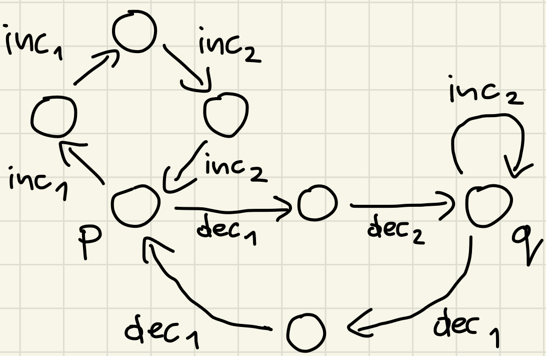
* dany VAS: (u, V)
* definiujemy VASS tak, żeby miał jeden stan q i dajemy mu przejścia (q, v, q) dla każdego $v \in V$
* czy w osiągalne z u? $\rightsquigarrow$ czy (q, w) osiągalne z (q, u)?

VASS $\rightsquigarrow$ automat licznikowy bez testów 0



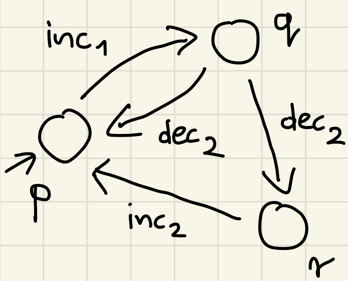
$q, (1,2)$ osiągalne z $p, (0,0)$

$q, (0,0)$ nieosiągalne z $p, (0,0)$

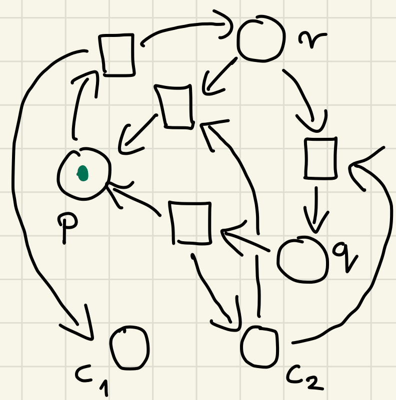


automat licznikowy bez testów 0 $\rightsquigarrow$ sieć Petriego

* definiujemy po jednym miejscu dla każdego stanu automatu i po jednym dla liczników
* tranzycja (w sieci Petriego) reprezentująca przejście (p, inc_i, q) bierze jeden żeton z p i kładzie po jednym żetonie na q i miejscu c_i reprezentującym licznik i



automat licznikowy



odpowiadająca sieć Petriego

* gdy automat jest w stanie p i istnieje krawędź $t = (p, dec_i, q)$, a obecnie $c_i = 0$, to t jest nieaktywne (nie można go użyć)
* istnieje rodzaj sieci Petriego pozwalający na testy na zero – dodajemy krawędzie „inhibitor arcs"

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

6. Dla każdego $m > 0$ skonstruuj sieć Petriego rozmiaru $O(n+m)$, która jest ograniczona, ale przez wartość nie mniejszą niż $F_m(n)$, gdzie $F_1(n) = 2n$, a $F_{m+1}(n) = F_m^n(1)$ jest n-krotnym złożeniem funkcji F_m zaaplikowanym do 1.

zadania do pomyślenia w domu (nieobowiązkowe)