8. Automat ze stosem – odpowiedzi

8.1.

$\{ a^{3n+1}b^{5n+1} \mid n \geq 0 \}$

8.2.

$\{ a^{3n+2}b^{7n+2} \mid n \geq 0 \}$

8.3.

$\{ a^{3n+2}b^{4n+3} \mid n \geq 0 \}$

8.4.

$\{ a^{3n+3}b^{5n+3} \mid n \geq 0 \}$

8.5.

$\{ a^{3n}(bdcbba)^n \mid n \geq 1 \}$

8.6.

$\{ a^{3n+1}b((db|a)^2bba)^n \mid n \geq 0 \}$

8.7.

$\{ (abc)^n abd^{7n+3} \mid n \geq 0 \}$

8.8.

$\{ (ab)^{2n+1}ad^{7n+5} \mid n \geq 0 \}$

8.9.

$\{ (aabc)^n aabdaaddbba(ccdddbba)^n \mid n \geq 0 \}$

8.10.

$\{ (aabc)^n ab(baaccc)^n \mid n \geq 0 \}$

8.11.

Model nie jest równoważny. Krótkie uzasadnienie: Taki automat potrafi rozpoznać język $\{a^nb^nc^n \mid n \in \mathrm{N}\}$, który nie jest bezkontekstowy. Najpierw, w zwykły sposób, maszyna sprawdza, czy jest tyle samo *a* co *b*. Potem wraca do pierwszego *b* na wejściu i sprawdza, czy jest tyle samo *b* co *c* (znowu w zwykły sposób).

8.12.

Model nie jest równoważny. Krótkie uzasadnienie: Taki automat potrafi rozpoznać język $\{a^nb^nc^n \mid n \in \mathrm{N}\}$, który nie jest bezkontekstowy. Przy czytaniu prefiksu a^*, wrzucamy go na stos. Przed przeczytaniem pierwszego *b*, wrzucamy na stos #, a potem stos podwajamy. W ten sposób liczbę *n* mamy dwa razy na stosie, co pozwala nam sprawdzić długość bloku liter *b* oraz długość bloku liter *c*.

8.13.

Model nie jest równoważny. Krótkie uzasadnienie: Taki automat potrafi rozpoznać język $\{a^nb^n\#(a|b)^n \mid n \in \mathrm{N}\}$, który nie jest bezkontekstowy. Najpierw, w zwykły sposób, maszyna sprawdza, czy jest tyle samo *a* co *b*, ale przy czytaniu liter *b* automat usuwa ostatnią literę słowa wejściowego. Aby zaakceptować, maszyna musi skończyć (napotkać koniec wejścia, mieć pusty stos) po wczytaniu #.

8.14.
Model nie jest równoważny. Krótkie uzasadnienie: Automat nowego typu może rozpoznać język $\{a^n b^n c^n \mid n \in \mathrm{N}\}$, który nie jest bezkontekstowy. W pierwszym podejściu sprawdza, czy jest tyle samo *a* co *b*, a w drugim podejściu sprawdza czy jest tyle samo *b* co *c*.

8.15.
Model nie jest równoważny. Krótkie uzasadnienie: Taki automat potrafi rozpoznać język $\{a^n b^n c^n \mid n \in \mathrm{N}\}$, który nie jest bezkontekstowy. Najpierw maszyna zapamiętuje wszystkie *a* na pierwszym stosie, po czym czytając litery *b* sprawdza na pierwszym stosie, czy jest ich tyle samo, co leżących tam *a*, równocześnie zapamiętując każdą literę *b* na drugim stosie. Na koniec w zwykły sposób sprawdza, czy na drugim stosie leży tyle liter *b*, ile jest liter *c* na wejściu.

8.16.
Model nie jest równoważny. Krótkie uzasadnienie: Taki automat potrafi rozpoznać język $\{a^n b^m \# c^n d^m \mid n \in \mathrm{N}\}$, który nie jest bezkontekstowy. Przy czytaniu bloku liter *a*, wrzucamy je na stos. Potem zapamiętujemy wszystkie *b* na stosie. Gdy czytamy # to „odwracamy" stos (teraz na stosie mamy najpierw litery *a*, a głębiej litery *b*) i później w zwykły sposób sprawdzamy, czy jest tyle samo liter *a* na stosie co liter *c* na wejściu, a potem czy jest tyle samo liter *b* na stosie co liter *d* na wejściu.

8.21.
$\{ x \mid x \in \{a,b,c\}^*, x = x^R, |x| \geq 2 \}$ czyli chodzi o akceptację nietrywialnych palindromów o długości co najmniej 2. Wejście ma postać: $a_1 a_2 ... a_n$.

(1) Automat przesuwa głowicę wejściową w prawo do znacznika prawego końca, w każdym ruchu kopiuje przeczytany symbol wejściowy na stos. Stos zawiera teraz $Z_0 a_1 a_2 ... a_n$. Jeśli $n<2$, automat odrzuca łańcuch wejściowy natychmiast.

(2) Automat przesuwa głowicę do znacznika lewego końca; nie zmienia przy tym symboli stosu. Automat umieszcza głowicę nad symbolem leżącym na prawo od znacznika lewego końca.

(3) **while** symbol na szczycie stosu jest identyczny z symbolem czytanym z wejścia **do**
begin
zdejmij symbol ze szczytu stosu;
przesuń głowicę wejściową o jedną pozycję w prawo;
end

(4) Jeśli stos zawiera tylko symbol Z_0, automat zatrzymuje się i akceptuje wejście.
W przeciwnym przypadku automat zatrzymuje się odrzucając łańcuch wejściowy.

Automat wymaga wykonania $O(n)$ kroków dla wejścia o długości *n*.

8.23.
$\{ xw \mid w \in \{a,b,c\}^*, x \in \{a,b,c\}^*, x = x^R, |x| \geq 2 \}$ czyli chodzi o akceptację łańcuchów posiadających przedrostek o długości 2 lub większej, będący nietrywialnym palindromem. Wejście ma postać: $a_1 a_2 ... a_n$.

(1) Automat przesuwa głowicę wejściową w prawo do znacznika prawego końca, w każdym ruchu kopiuje przeczytany symbol wejściowy na stos. Stos zawiera teraz $Z_0 a_1 a_2 ... a_n$. Jeśli $n<2$, automat odrzuca łańcuch wejściowy natychmiast.

(2) Automat przesuwa głowicę do znacznika lewego końca; nie zmienia przy tym symboli stosu. Automat umieszcza głowicę nad symbolem leżącym na prawo od znacznika lewego końca.

(3) **while** symbol na szczycie stosu jest identyczny z symbolem czytanym z wejścia **do**
begin
zdejmij symbol ze szczytu stosu;
przesuń głowicę wejściową o jedną pozycję w prawo;
end

(4) Jeśli stos zawiera tylko symbol Z_0, automat zatrzymuje się i akceptuje wejście. W przeciwnym przypadku (symbol wejściowy nie jest identyczny z symbolem na szczycie stosu) automat odbudowuje stos kopiując łańcuch wejściowy z powrotem na stos przesuwając głowicę w lewo, aż dotrze do znacznika lewego końca (do tej pory wszystko pasowało, więc można w ten sposób odbudować stos). Stos zawiera teraz $Z_0a_1a_2...a_i$ dla pewnego i. (Po dotarciu do tego miejsca po raz pierwszy $i=n$, lecz za każdym następnym razem wartość i będzie malała o jeden). Jeżeli $i=2$ automat zatrzymuje się i odrzuca wejście. W przeciwnym razie zdejmuje ze stosu a_i i przesuwa głowicę wejściową na prawo o jedną klatkę do symbolu bezpośrednio na prawo od znacznika lewego końca. Potem automat wraca do kroku (3).

Procedura wymaga od automatu wykonania $O(n^2)$ ruchów dla wejścia o długości n.

8.24.

$\{ xx^Rw \mid w \in \{a,b,c\}^*, x \in \{a,b,c\}^*, |x| \geq 1 \}$ Automat próbuje znaleźć najkrótszy możliwy łańcuch x taki by całe wejście miało postać xx^Rw. Poniżej uproszczony zarys procedury działania automatu. Automat wchodzi w tryb kopiowania wejścia na stos i czytając wejście od lewej w prawą stronę kopiuje każdy przeczytany symbol na stos, aż spotka na wejściu dwa identyczne symbole pod rząd. (W łańcuchu xx^R ostatni symbol x i pierwszy symbol x^R są identyczne). Jeśli nie napotka na wejściu dwóch identycznych symboli pod rząd i dojdzie do prawego znacznika końca wejścia, to odrzuca wejście. Jeśli napotka dwa kolejne identyczne symbole na wejściu, to przechodzi do trybu dopasowywania wejścia do zawartości stosu, czytając dalej wejście w prawo zdejmuje ze stosu symbole identyczne z przeczytanymi tak długo, jak się da. Jeśli dojdzie do dna stosu, akceptuje. Jeśli dojdzie do prawego znacznika końca wejścia, a na stosie nie będzie wyłącznie znacznika dna stosu, to odrzuca wejście. Jeśli symbol czytany z wejścia nie odpowiada symbolowi na szczycie stosu, to odtwarza zawartość stosu, czytając wejście w lewo, aż dojdzie do pierwszego miejsca, w którym są dwa identyczne symbole. Teraz stos zawiera to samo, co zawierał przed zmianą trybu. Więc automat wraca do trybu kopiowania wejścia na stos, czytając w prawo znowu szuka następnego wystąpienia dwóch kolejnych identycznych symboli. Jeśli znajdzie, przechodzi do trybu dopasowywania wejścia, jak wcześniej. Jeśli w trybie kopiowania nie napotka na wejściu dwóch identycznych symboli pod rząd i dojdzie do prawego znacznika końca wejścia, to jak poprzednio, odrzuca wejście.

Czytelnik zechce dopracować powyższy zarys procedury, podobnie jak to uczyniono w odpowiedziach do zadań 8.21, 8.23 i 8.26.

8.26. $\{ w\#x \mid w \in \{a,b,c\}^*, x \in \{a,b,c\}^*, x$ jest podłańcuchem łańcucha $w \}$ czyli chodzi o rozpoznanie „słowa kluczowego” x w napisie w. Stanowiący część wejścia łańcuch w ma postać: $a_1a_2...a_n$.

(1) Automat przesuwa głowicę wejściową w prawo aż napotka #.

(2) Automat przesuwa głowicę w lewo, kopiując $a_n a_{n-1} \ldots a_1$ na stos, aż głowica dotrze do znacznika lewego końca. Stos zawiera teraz $Z_0 a_n a_{n-1} \ldots a_1$; na szczycie stosu leży pierwszy od lewej strony symbol łańcucha *w*.

(3) Automat przesuwa głowicę wejściową do pierwszego na prawo od # symbolu, który jest początkiem łańcucha *x*. i zaczyna czytając wejście w prawo dopasowywać przeczytane symbole *x* do symboli *w* leżących na stosie.

(4) **while** symbol na szczycie stosu jest identyczny z symbolem czytanym z wejścia **do**
begin
zdejmij symbol ze szczytu stosu;
przesuń głowicę wejściową o jedną pozycję w prawo;
end

(5) Jeśli głowica wejściowa dotarła do znacznika prawego końca *$*, automat akceptuje wejście.
W przeciwnym przypadku (symbol wejściowy nie jest identyczny z symbolem na szczycie stosu) automat odbudowuje stos kopiując łańcuch wejściowy z powrotem na stos przesuwając głowicę w lewo, aż dotrze do symbolu # (do tej pory wszystko pasowało, więc można w ten sposób odbudować stos). Stos zawiera teraz $Z_0 a_n a_{n-1} \ldots a_i$ dla pewnego $1 \leq i \leq n$. (Po dotarciu do tego miejsca po raz pierwszy $i=1$, lecz za każdym następnym razem wartość i będzie rosła o jeden). Jeżeli $i=n$ automat zatrzymuje się i odrzuca wejście. W przeciwnym razie zdejmuje ze stosu a_i i przesuwa głowicę wejściową w prawo o jedną klatkę do pierwszego symbolu łańcucha *x*. Potem automat wraca do kroku (4).

Procedura może wymagać od automatu wykonania $O(|x| \cdot |w|)$ ruchów.