1. Napisz gramatykę bezkontekstową generującą język :

$$L_1 \;=\; \{\, 0^i 1 0^j 1 0^p \;:\; i,j,p > 0,\; i+j \;=\; p \,\}$$

Odpowiedź. Gramatyka wygląda tak:

$$\begin{aligned} S \to &\quad 01X0 \quad | \quad 0S0 \\ X \to &\quad 010 \quad | \quad 0X0. \end{aligned}$$

Nieterminalem startowym jest S.

Nieterminal X generuje słowa postaci $0^j 1 0^j$, gdzie $j > 0$. Nieterminal S generuje słowa postaci $0^i 1 w 0^i$, gdzie $i > 0$ i w jest słowem wygenerowanym przez nieterminal X. A więc nieterminal S generuje słowa postaci

$$0^i 1 0^j 1 0^j 0^i = 0^i 1 0^j 1 0^{j+i} \qquad \text{gdzie } i,j > 0,$$

czyli język z treści zadania.

2. Czy język L_1 jest regularny (odpowiedź uzasadnić).

Odpowiedź. Język ten nie jest regularny.

Załóżmy bowiem, że jest regularny. Korzystając z lematu o pompowaniu dojdziemy do sprzeczności. Skoro język jest regularny, to na mocy lematu o pompowaniu, istnieje stała M, taka, że dla każedgo słowa w z języka, jeśli w ma długość przynajmniej M, to istnieje podział na trzy słowa

$$w = xyz \qquad \text{gdzie } y \text{ niepuste}$$

taki, że dla każdego i słowo xy^iz należy do języka. Rozważmy słowo

$$w = 0^M 10^M 10^{2M},$$

które należy do języka i ma długość przynajmniej M. Rozważmy jego podział $w = xyz$ jak wyżej. Na mocy lematu, dla $i = 2$, słowo xy^2z powinno należeć do języka, a pokażemy, że nie należy. Otrzymana sprzeczność dowodzi, że język nie mógł być regularny.

Jeśli słowo y zawiera jedynkę, to słowo xy^2z zawiera trzy lub więcej jedynki, a więc nie może należeć do języka. Jeśli słowo y nie zawiera jedynki, to składa się samych zer, a więc jest w całości zawarte w jednym z trzech bloków zer (bloki te mają długości odpowiednio M, M i $2M$). Wówczas w słowie xy^2z jeden z bloków zer zmienia długość (w porównaniu ze słowem xyz), a pozostałe nie zmieniają długości. Oznacza to, że suma długości pierwszych dwóch bloków zer nie może już być równa długości trzeciego bloku zer.

3. Niech $\phi(L) = \{ w : w \in L \ \& \ w^R \in L \}$. Czy zachodzą implikacje (uzasadnić):

   (a) L jest bezkontekstowy to $\phi(L)$ też jest bezkontekstowy

   (b) L jest regularny to $\phi(L)$ też jest regularny.

Odpowiedź. Implikacja (a) nie zachodzi, lecz implikacja (b) zachodzi. Powód jest z grubsza rzecz biorąc taki, że języki bezkontekstowe nie są zamknięte na przecięcia, a regularne są.

Implikacja (a) nie zachodzi. Rozważmy język

$$L = \{a^i b a^j b a^k : i = j\}.$$

Język ten jest bezkontekstowy, generuje go gramatyka

$$\begin{aligned} S &\to Xb|Sa \\ X &\to b|aXa. \end{aligned}$$

Lustrzane odbicie tego języka to

$$L^R = \{a^i b a^j b a^k : j = k\}.$$

A więc słowo należy do L i L^R wtedy i tylko wtedy, gdy jest postaci

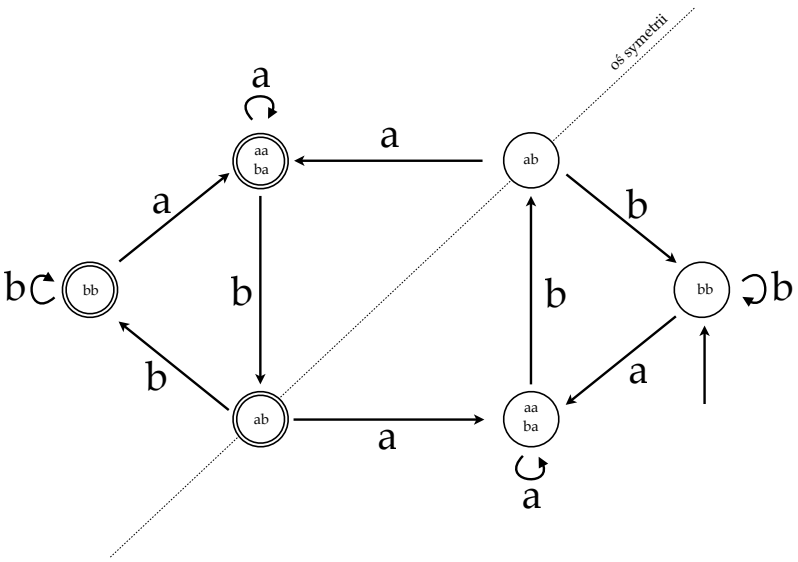
$$a^i b a^j b a^k \qquad \text{gdzie } i = j = k.$$

Zbiór słów takiej postaci nie jest bezkontekstowy z tego samego powodu, dla którego bezkontekstowy nie jest język $\{a^n b^n c^n : n \in \mathbb{N}\}$.

Implikacja (b) zachodzi. Język $\phi(L)$ to przecięcie języków L i L^R. Języki regularne są zamknięte na przecięcia (wystarczy rozważyć produkt dwóch automatów). A więc wystarczy pokazać, że języki regularne są zamknięte na operację lustrzanego odbicia $L \mapsto L^R$. Automat (niedeterministyczny) dla L^R otrzymujemy z automatu dla L poprzez: zamianę stanów akceptujących z początkowymi i odwrócenie kierunku strzałek w tranzycjach.

4. Narysować minimalny deterministyczny automat skończony akceptujący słowa nad alfabetem $\{a, b\}$, w których podsłowo aba występuje nieparzystą liczbę razy. **Nie należy** w tym zadaniu pisać uzasadnienia minimalności.

Odpowiedź.



Dla słów przynajmniej dwuliterowych, automat pamięta dwie rzeczy:

- czy aba wystąpiło parzystą czy nieparzystą ilość razy.
- ostatnie dwie litery, przy czym końcówki aa i ba są utożsamione.

Każda kombinacja jest możliwa, stąd $2 \times 3 = 6$ stanów. Nie potrzebujemy stanów dla słów o długości zero lub jeden, ponieważ:

- słowa ϵ i b są równoważne słowu bb;
- słowo a jest równoważne słowu aa.

5. Dla automatu skończonego A przez $L_u(A)$ oznaczmy zbiór wszystkich długości słów akceptowanych przez A. Czy problem sprawdzania równości $L_u(A) = L_u(B)$ jest rozstrzygalny dla automatów skończonych (uzasadnić).

Odpowiedź. Problem ten jest rozstrzygalny. Poniżej przedstawimy algorytm rozstrzygający problem, który działa w czasie wykładniczym ze względu na rozmiar automatów A i B.

Zbiór $L_u(A)$ można widzieć jako zbiór słów nad alfabetem jednoliterowym $\{a\}$, mianowicie jako zbiór słów postaci a^n, gdzie n jest długością pewnego słowa akceptowanego przez A. Automat rozpoznający ten zbiór słów, oznaczmy go A', otrzymujemy z automatu A poprzez zastąpienie każdej litery w przejściach przez literę a. (Automat A' może być niedeterministyczny, nawet jeśli A był deterministyczny.) Oznaczmy wreszcie przez A'' automat otrzymany z A' przez determinizację. Automat A'' można obliczyć na podstawie A w czasie wykładniczym.

Przy tej notacji, problem z zadania brzmi: mając dane dwa automaty A i B, sprawdzić czy języki rozpoznawane przez A'' i B'' są takie same. Dla automatów deterministycznych, sprawdzanie równości języków można zrobić w czasie wielomianowym, opis tego algorytmu jest poniżej.

Aby sprawdzić równość dwóch języków, konstruujemy produkt kartezjański $A'' \times B''$ i sprawdzamy, czy w tym automacie osiągalna jest para stanów, gdzie na jednej współrzędnej stan jest akceptujący, a na drugiej współrzędnej stan nie jest akceptujący. Słowo prowadzące do tej pary stanów jest świadkiem na różność języków rozpoznawanych przez A'' i B''. Ważny jest tutaj determinizm automatów, w przeciwnym przypadku dla słowa prowadzącego do złej pary (jeden automat akceptuje, drugi automat odrzuca), mogłyby istnieć inne biegi pokazujące, że oba automaty akceptują słowo.

6. Maszyna Turinga jest typu *write-once* jeśli w każdym obliczeniu co najwyżej raz zmienia zawartość każdej komórki taśmy. Czy problem $L(M) = \emptyset$ jest rozstrzygalny dla deterministycznych jednotaśmowych maszyn Turinga M typu *write-once* ? Odpowiedź uzasadnić.

Odpowiedź. Problem nie jest (całkowicie) rozstrzygalny. Pokażemy rozwiązanie dla definicji write-once, w której zmiana litery "blank" na inną też liczy się jako zapis. (Trochę łatwiej zadanie jest rozwiązać dla większej klasy maszyn, gdzie zmiana litery "blank" na inną się nie liczy.)

Do problemu z zadania zredukujemy problem Posta, który nie jest całkowicie rozstrzygalny. Przypomnijmy, że problem Posta brzmi tak:

- Dane: zbiór par słow P.
- Pytanie: czy istnieje skończony ciąg par $(x_1, y_1), \ldots, (x_n, y_n) \in P$, być może z powtórzeniami, taki że

$$x_1 \cdots x_n = y_1 \cdots y_n$$

Równoważne sformułowanie pytania brzmi, czy niepusty jest język, oznaczmy go przez L_P, który zawiera słowa postaci

$$\#x_1\$y_1\#x_2\$y_2\# \cdots \#x_n\$y_n,$$

(zakładamy, że specjalne searatory # i \$ nie występują w słowach ze zbioru P) takie, że spełnione są następujące warunki (a) i (b):

(a) dla każdego $i \in \{1, \ldots, n\}$, para (x_i, y_i) należy do P.

(b) $x_1 \cdots x_n = y_1 \cdots y_n$.

Pokażemy, że na podstawie zbioru P można w skończonym czasie obliczyć maszynę Turinga, nazwijmy ją M_P, która rozpoznaje język L_P i jest write-once. A więc gdybyśmy umieli rozstrzygać niepustość języka dla maszyn write-once, to byśmy także umieli rozstrzygać problem Posta.

Maszyna M_P najpierw sprawdza warunek (a) nie pisząc nic taśmie, a potem sprawdza warunek (b) pisząc na taśmie w sposób write-once.

Warunek (a) jest językiem regularnym, opisanym wyrażeniem regularnym

$$(\sum_{(x,y)\in P} \#x\$y)^*.$$

Można więc warunek ten rozpoznawać deterministycznym automatem skończonym, który jest szczególnym przypadkiem maszyny Turinga nie piszącej na taśmie. Po sprawdzeniu warunku (a), maszyna M_P wraca na początek taśmy i sprawdza warunek (b) w sposób opisany poniżej.

Aby sprawdzić warunek (b), maszyna M_P postępuje w sposób następujący. Idea jest taka, że litery nie będące separatorami są stopniowo zastępowane znacznikiem @. Maszyna, będąc na początku taśmy, szuka głowicą (od lewej do prawej) pierwszej litery pochodzącej ze słowa typu x (a więc litery która nie jest separatorem, ale jej najbliższy z lewej separator to # a nie \$), która jeszcze nie została zastąpiona znacznikiem @. Znalazłwszy taką literę, maszyna zapamiętuje ją w stanie i zastępuje ją znacznikiem @ (tutaj występuje jednokrotny zapis). Następnie głowica maszyny wraca na początek taśmy i szuka (znowu od lewej do prawej) pierwszej litery pochodzącej ze słowa typu y (a więc litery która nie jest separatorem, ale jej najbliższy z lewej separator to \$ a nie #) i zastępuje ją @ o ile jest taka sama jak litera zapamiętana w stanie (w przeciwnym przypadku maszyna natychmiast odrzuca). Następnie maszyna powtarza proces, aż nie doprowadzi do sytuacji, w której na taśmie są wyłącznie symbole @, # i \$, wówczas akceptuje.