1. (6 punktów) Czy dla każdego regularnego L, język

   $$f(L) = \{w : \text{każdy prefiks } w \text{ długości nieparzystej należy do } L\}$$

   też jest regularny?

   Odpowiedź. Tak, jęsli L jest regularny to też $f(L)$. Niech A będzie alfabetem języka L. Rozważmy dopełnienie $f(L)$, czyli

   $$A^* - f(L) = \{w : \text{pewien prefiks } w \text{ długości nieparzystej nie należy do } L\}.$$

   Dopełnienie to można wyrazić w następujący sposób:

   $$A^* - f(L) = ((A^* - L) \cap (AA)^* A) \cdot A^*,$$

   czyli najpierw dajemy słowo poza L długości nieparzystej, a potem dajemy dowolną ilość liter. Wyrażenie po prawej stronie równości korzysta z języka L, regularnych języków $(AA)^*A$ oraz A^*, oraz operacji dopełnienia, przecięcia języków i konkatenacji. Klasa języków regularnych jest zamknięta na te operacje, a więc dopełnienie $f(L)$ jest regularne. A więc i sam $f(L)$ jest regularny.

2. (6 punktów) Rozważmy język L słów nad alfabetem $\{a, b, (,)\}$ w których:

   - Wszystkie a i b są otoczone nawiasami.
   - Po usunięciu wszystkich a i b otrzymujemy poprawne wyrażenie nawiasowe
   - Dla każdej pary odpowiadających sobie nawiasów (i), liczba liter a i b pomiędzy nimi jest taka sama.

   Na przykład

   $$(aabb) \in L \qquad (a(a(b()a)bab)b) \in L$$

   natomiast

   $$a(()()) \notin L \qquad (a) \notin L \qquad ((aabb) \notin L \qquad (aa)(bb) \notin L$$

   Czy język L jest bezkontekstowy?

   Odpowiedź. Tak, język L jest bezkontekstowy. Oto gramatyka, z symbolem startowym X:

   $$\begin{aligned} X &\to \epsilon | XX | (Y) \\ Y &\to X | aYb | bYa | YY. \end{aligned}$$

   Pokażemy, że dla każdego słowa w, zachodzą równoważności

   (a) $w \in L$ wtedy i tylko wtedy, gdy w można wygenerować z nieterminala X.

   (b) $(w) \in L$ wtedy i tylko wtedy, gdy w można wygenerować z nieterminala Y.

   Dowód jest przez indukcję po długości słowa w. W obu podpunktach implikacja z prawej do lewej jest natychmiastowa, więc skupmy się na implikacji z lewej do prawej. Baza indukcji jest natychmiastowa – to słowo puste. Zróbmy krok indukcyjny.

   (a) Rozważmy niepuste słowo $w \in L$. Rozbijmy je uwzględniając najbardziej zewnętrzne nawiasy

   $$w = (w_1)(w_2)\cdots(w_n).$$

   Z definicji języka, słowa $(w_1), \ldots, (w_n)$ muszą należeć do L, a więc z punktu (b) założenia indukcyjnego słowa $w_1, \ldots, w_n$ można wygenerować z nieterminala Y. Na mocy reguły $X \to (Y)$, słowa $(w_1), \ldots, (w_n)$ można wygenerować z nieterminala X. Korzystając $(n-1)$-krotnie z reguły $X \to XX$, możemy wygenerować całe słowo $(w_1)\cdots(w_n)$ z nieterminala X.

(b) Rozważmy niepuste słowo w takie, że $(w) \in L$. Rozbijmy je uwzględniając najbardziej zewnętrzne nawiasy

$$w = v_0(w_1)v_1(w_2)v_2 \cdots v_{n-1}(w_n)v_n,$$

gdzie słowa $v_0, \ldots, v_n$ nie zawierają nawiasów, a więc należą do $\{a, b\}^*$. Z założenia indukcyjnego wiemy, że słowa $(w_1), \ldots, (w_n)$ można wygenerować z nieterminala X. Teraz pokazujemy, że słowo w można wygenerować z Y, korzystając z tego samego dowodu, który dowodzi że gramatyka

$$Z \to \epsilon | aZb | bZa | ZZ$$

generuje wszystkie słowa, gdzie jest tyle samo a co b.

3. (6 punktów) Niech A, B skończone alfabety. Rozważmy funkcję $f : A \to B^*$. Stosujemy ją do słów litera po literze, oraz do języków słowo po słowie. (Ponieważ obraz funkcji f może zawierać słowa puste, to funkcja może skracać słowa.) Czy języki postaci $f(L)$, gdzie L jest w PTIME mogą być nierozstrzygalne?

Odpowiedź. Tak, język potrafi być nierozstrzygalny. Niech $L \subseteq a^*$ będzie dowolnym językiem nierozstrzygalnym, ale częściowo rozstrzygalnym. Np. L może być kodowaniem problemu stopu za pomocą alfabetu jednoliterowego. Niech M będzie deterministyczną maszyną Turinga, która rozpoznaje L, czyli akceptuje dokładnie słowa z L. Maszyna taka istnieje z założenia o częściowej rozstrzygalności, ale będzie się pętlić na niektórych słowach spoza L (no bo inaczej L byłby całkowicie rozstrzygalny). Rozważmy język

$$K = \{a^n b^m : \text{maszyna } M \text{ akceptuje } a^n \text{ w } m \text{ krokach}\}.$$

oraz funkcję f, która wyciera b i zostawia a. Łatwo zobaczyć, że

- Mamy $L = f(K)$, a więc język $f(K)$ jest nierozstrzygalny.
- Język K należy do PTIME. Dostawszy słowo $a^n b^m$, możemy zasymulować m kroków maszyny M i sprawdzić czy maszyna akceptuje. Taka symulacja może być zrobiona w czasie wielomianowym, na kilka sposobów. Oto jeden z nich. Załóżmy, że maszyna M korzysta z taśmy lewostronnie nieskończonej. Wówczas symulujemy maszynę na słowie a^n; przy czym głowica nie potrzebuje zapisywać komórek z b^m, bo maszyna M ma lewostronnie nieskończoną taśmę. Przy każdym kroku, usuwamy jedną z liter b. Przy usunięciu ostatniej litery b, akceptujemy jeśli symulowana maszyna M jest w stanie akceptującym. Taka symulacja zajmuje $O(m^2)$ kroków.

4. (6 punktów *) Czy następujący problem należy do klasy NP?
   - Dane. Automat niedeterministyczny nad alfabetem jednoliterowym.
   - Pytanie. Czy istnieje słowo którego ten automat nie akceptuje?

Odpowiedź. Tak, problem należy do klasy NP. Niech a będzie jedyną literą w alfabecie wejściowym. Dla automatu niedeterministycznego o stanach Q i liczby $k \in \mathbb{N}$, zdefiniujmy

$$\delta_k \subseteq Q \times Q$$

jako zbiór tych par statnów (p, q), że automat ma pewien bieg po słowie a^k, który zaczyna się w p i kończy w q. Zbiór δ_k to zbiór par stanów, a więc ilość miejsca, którą zajmuje w pamięci jest wielomianowa ze względu na automat. Dla słów $k_1, k_2 \in \mathbb{N}$ mamy

$$\delta_{k_1+k_2} = \delta_{k_1} \circ \delta_{k_2} \tag{1}$$

gdzie $\circ$ to złożenie relacji. Jeśli znamy δ_{k_1} i δ_{k_2}, to możemy w czasie wielomianowym obliczyć też $\delta_{k_1+k_2}$, korzystając z powyższego wzoru (jest to coś w rodzaju mnożenia macierzy).

Napiszemy algorytm niedeterministyczny wielomianowy, który działa w dwóch krokach.

(a) Wiemy, że jeśli automat $\mathcal{A}$ odrzuca jakieś słowo, to odrzuca jakieś słowo długości mniejszej niż 2^n. (Bo po zdeterminizowaniu, mamy automat o 2^n stanach, a długość najkrótszego odrzuconego słowa przy automacie deterministycznym szacuje się przez ilość stanów.) Zgadujemy więc liczbę $i \in \{0, \ldots, 2^n - 1\}$, a dokładniej jej zapis binarny

$$i = a_0 \cdot 2^0 + a_1 \cdot 2^1 + \cdots + a_{n-1} \cdot 2^{n-1} \qquad \text{gdzie } a_0, \ldots, a_{n-1} \in \{0, 1\}.$$

Ponieważ zgadujemy tutaj n bitów $a_0, \ldots, a_{n-1}$, operacja może być wykonana przez niedeterministyczną maszynę w czasie wielomianowym (nawet liniowym).

(b) Mając zapis binarny liczby i, chcemy stwierdzić, czy istotnie automat odrzuca słowo a^i. Wystarczy obliczyć δ_i i sprawdzić czy ten zbiór zawiera parę, gdzie pierwszy stan jest początkowy, a drugi stan jest akceptujący. Obliczenie δ_i zrobimy już w deterministycznym czasie wielomianowym. Na mocy (1) mamy

$$\delta_i = \delta_{a_0 \cdot 2^0} \circ \delta_{a_1 \cdot 2^1} \circ \cdots \circ \delta_{a_{n-1} \cdot 2^{n-1}}.$$

Aby obliczyć powyższe wyrażenie w czasie wielomianowym, wystarczy umieć policzyć δ_{2^j} w czasie wielomianowym, dla $j \in \{0, \ldots, n-1\}$. A to robimy korzystając i-krotnie z zasady rekurencyjnej:

$$\delta^{2^j} = \delta^{2^{j-1}} \circ \delta^{2^{j-1}}.$$

Imię, nazwisko, nr indeksu *(kod)*

(9 punktów) Wybierz 9 z poniższych pytań i wybierz odpowiedź tak/nie (bez uzasadnienia). Za prawidłowe odpowiedzi dajemy +1 punkt, za złe -1 punkt. Punkty policzymy za 9 najgorszych odpowiedzi, czyli nie opłaca się odpowiadać na więcej niż 9 pytań.

1. Czy całkowicie rozstrzygalny jest problem: dany automat ze stosem rozpoznający L, pytanie czy $L = L^*$.

   Odpowiedź. Nie. Problem jest nierozstrzygalny, bo jest co najmniej tak samo trudny jak problem uniwersalności dla automatów ze stosem, który jest nierozstrzygalny. Przypomnijmy, że problem uniwersalności brzmi: "dany automat ze stosem rozpoznający L, pytanie czy L zawiera wszystkie słowa". Oto algorytm rozstrzygający problem uniwersalności, który korzysta z algorytmu dla problemu z zadania. Niech L język rozpoznawany przez automat ze stosem $\mathcal{A}$. Algorytm ma dwa kroki.

   (a) Sprawdzamy, czy L zawiera słowo puste i każde słowo jednoliterowe. Jeśli, nie to L nie jest zawiera wszystkich słów i algorytm odpowiada "nie". Ten krok można zrobić w czasie skończonym (nawet wielomianowym), bo problem "czy dany automat ze stosem $\mathcal{A}$ akceptuje słowo w" jest rozstrzygalny w czasie wielomianowym.

   (b) Jeśli L zawiera wszystkie słowa jednoliterowe, to wówczas $L^* = A^*$, a więc problem uniwersalności ma tę samą odpowiedź co problem $L = L^*$, a więc możemy skorzystać z algorytmu dla problemu z zadania.

2. Czy całkowicie rozstrzygalny jest problem: dany automat skończony rozpoznający L, pytanie czy $L^* = (LL)^*$.

   Odpowiedź. Tak. Problem jest całkowicie rozstrzygalny. Najpierw automat zmieniamy na wyrażenie regularne dla L, potem obliczamy wyrażenia regularne dla L^* oraz $(LL)^*$, następnie oba wyrażenia zmieniamy na automaty deterministyczne, minimalizujemy, i sprawdzamy czy w obu przypadkach wyszedł ten sam automat.

3. Jeśli L jest całkowicie rozstrzygalny, oraz $K \subseteq L$, to czy K też jest całkowicie rozstrzygalny?

   Odpowiedź. Nie. Problem może być nierozstrzygalny. Na przykład zbiór wszystkich słów jest całkowicie rozstrzygalny, a ma nierozstrzygalne podzbiory.

4. Czy równoważne (pod względem akceptowanych języków) są deterministyczne automaty ze stosem i niedeterministyczne automaty ze stosem?

   Odpowiedź. Nie. Nie są równoważne, bo języki rozpoznawane przez deterministyczne automaty ze stosem są zamknięte na dopełnienie, a niedeterministyczne nie. Oto dłuższe uzasadnienie. Dopełnienie języka $a^n b^n c^n$ to język

   $$\{a^i b^j c^k : \text{liczby } i, j, k \text{ nie są wszystkie równe}\} \cup (\{a, b, c\}^* - a^* b^* c^*),$$

który jest rozpoznawany przez niedeterministyczny automat ze stosem. Gdyby ten język był też rozpoznawany przez deterministyczy automat ze stosem, to i jego dopełnienie byłoby rozpoznawane, a wiemy że język $a^n b^n c^n$ nie jest bezkontekstowy.

5. Czy klasa języków całkowicie rozstrzygalnych zamknięta jest na wszystkie z operacji: suma, dopełnienie, przecięcie, gwiazdka?

   Odpowiedź. Tak. Zróbmy na przykład dopełnienie i gwiazdkę. Dla dopełnienia bierzemy maszynę deterministyczną i zamienamy stany akceptujące z odrzucającymi. Ponieważ nie ma niedeterminizmu ani pętlenia, konstrukcja daje właściwy wynik. Zróbmy teraz gwiazdkę. Dostajemy słowo w, pytanie czy $w \in L^*$, dla języka całkowicie rozstrzygalnego L. Powolny algorytm robi to tak: sprawdza wszsytkie podziały w na konkatenację kilku słów $w = w_1 \cdots w_n$ (wykładniczo wiele sposbów) i dla każdego z nich sprawdza czy wszystkie słowa $w_1, \ldots, w_n$ należą do L. Bardziej algorytimcznie nastawiony czytelnik zauważy, że stosując algorytm dynamiczny można rozważać wielomianowo wiele podziałów.

6. Czy należy do PSPACE problem: dany automat niedeterministyczny, pytanie czy akceptuje wszystkie słowa?

   Odpowiedź. Tak. Korzystamy z tego, że deterministyczny PSPACE jest równy niedeterministycznemu PSPACE. Ponieważ PSPACE jest zamknięty na dopełnienie (z determinizmu), wystarczy pokazać algorytm PSPACE, który sprawdza czy automat odrzuca pewne słowo. Dla problemu odrzucania słowa, możemy natomiast skorzystać z niedeterministycznego algorytmu PSPACE. Niech Q będą stanami danego automatu niedeterministycznego. Algorytm zgaduje kolejne litery $a_1, a_2, \ldots$ słowa, i dla każdego i oblicza zbiór stanów $Q_i \subseteq Q$ osiągalnych po prefiksie $a_1 \cdots a_i$. To wszystko można zrobić w pamięci wielomianowej, bo obliczywszy Q_i można zapomnieć o Q_{i-1} i wszystkich poprzednich literach. Jak algorytm dojdzie do zbioru Q_i który nie zawiera żadnego stanu akceptującego, to akceptuje. Ten algorytm jest już poprawnym rozwiązaniem. Niektórych jednak może irytować fakt, że algorytm potrafi się pętlić. To można poprawić robiąc obserwację (patrz zadanie z gwiazdką spoza testu), że jeśli automat odrzuca słowo, to odrzuca słowo długości $2^{|Q|}$, a więc algorytm może przestać generować nowe litery po $2^{|Q|}$ krokach.

7. Czy dla każdego n istnieje gramatyka bezkontekstowa rozmiaru $O(n)$, której najkrótsze generowane słowo jest długości co najmniej 2^n?

   Odpowiedź. Tak. Gramatyka ma n nieterminali $X_1, \ldots, X_n$ i następujące reguły:

   $$X_1 \to X_2 X_2 \qquad X_2 \to X_3 X_3 \qquad \cdots \qquad X_{n-1} \to X_n X_n \qquad X_n \to aa.$$

   Gramatyka ta generuje dokładnie jedno słowo, jest nim a^{2^n}. To jest właściwie granica możliwości każdej gramatyki: najkrótsze generowane słowo jest zawsze długości wykładniczej ze względu na rozmiar gramatyki.

8. Czy dla każdego n istnieje gramatyka bezkontekstowa rozmiaru $O(n)$, której najkrótsze niegenerowane słowo jest długości co najmniej 2^n?

   Odpowiedź. Tak. Gramatyka ma n nieterminali $X_1, \ldots, X_n$ i następujące reguły:

   $$X_1 \to X_2X_2 \qquad X_2 \to X_3X_3 \qquad \cdots \qquad X_{n-1} \to X_nX_n \qquad X_n \to aa|a|\epsilon.$$

   Gramatyka te generuje słowa postaci a^i dla $i \leqslant 2^n$. Wykraczając poza zadanie, można też zrobić gramatykę, gdzie najkrótsze niegenerowane słowo jest rozmiaru 2^{2^n}, a nawet dłuższe niż $f(n)$ dla dowolnej funkcji obliczalnej.

9. Rozważmy (skierowany) graf konfiguracji maszyny Turinga (wierzchołki to konfiguracje, krawędzie to jeden krok obliczeń). Konfiguracja osiągalna to taka, do której można dojść z konfiguracji postaci: słowo wejściowe na taśmie, głowica na początku taśmy w stanie początkowym. Maszynę Turinga nazwiemy odwracalną, jeśli do każdej osiągalnej konfiguracji wchodzi dokładnie jedna krawędź. Czy każdy język całkowicie rozstrzygalny można rozpoznawać maszyną deterministyczną odwracalną (maszyną totalną, czyli taką, która każdy bieg kończy akceptacją lub odrzuceniem w skończonym czasie)?

   Odpowiedź. Tak. Niech L będzie językiem całkowicie rozstrzygalnym, rozpoznawanym przez maszynę deterministyczną i totalną M. Przeróbmy maszynę M tak, żeby z każdej konfiguracji można było odtworzyć słowo wejściowe. Na przykład maszyna może zostawić słowo wejściowe na taśmie i pisać tylko symbole spoza alfabetu wejściowego. To znaczy, jeśli maszyna dostaje na wejściu słowo w, to wszystkie kolejne zawartości taśmy w trakcie obliczenia są postaci wv, gdzie v korzysta tylko z tej części alfabetu roboczego, która nie jest aflabetem wejściowym. Nowa maszyna ma tę własność, że patrząc na jej konfigurację (zawartość taśmy, położenie głowicy i stan) możemy odtworzyć poprzednią konfigurację. Wystarczy bowiem preczytać słowo wejściowe (które jest do odtworzenia z zawartości taśmy), wykonać od nowa całe obliczenie aż do bieżącej konfiguracji i zwrócić przedostatnią konfigurację. A więc maszyna jest odwracalna.

10. Rozważmy wariant wyrażeń regularnych, gdzie oprócz sumy, dostępna jest też część wspólna języków. Czy takie wyrażenia generują tylko języki regularne?

    Odpowiedź. Tak, bo klasa języków regularnych jest zamknięta na przecięcie.

11. Rozważmy deterministyczne automaty z wieloma stosami, które akceptują przez stan akceptujący. Czy dla każdego automatu z 3 stosami istnieje równoważny automat z 2 stosami?

    Odpowiedź. Tak. Oto dwa możliwe uzasadnienia. Pierwsze jest takie, że już z 2 stosami dostajemy model tak samo silny jak maszyny Turinga,

a maszyny Turinga umieją symulować 3 a nawet więcej stosów. Drugie uzasadnienie jest konstrukcją bezpośrednią. Załóżmy, że mamy zawartość 3 stosów w słowach w_1, w_2, w_3. Wówczas trzymamy na pierwszym stosie słowo $w_1\#w_2\#w_3$. Korzystając z drugiego stosu jako stosu pomocniczego, jesteśmy w stanie zasymulować dowolną operację stosową.

12. Załóżmy P $\neq$ NP, bo inaczej zadanie jest oczywiste. Niech A, B skończone alfabety. Rozważmy funkcję $f : A \to B$. Stosujemy ją do słów litera po literze (funkcja zachowuje długość słowa), oraz do języków słowo po słowie. Czy języki postaci $f(L)$, gdzie L jest w PTIME, mogą być NP-trudne?

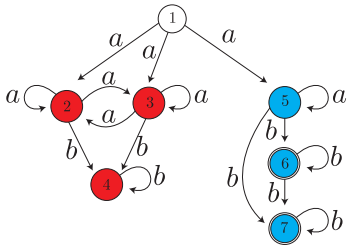
    Odpowiedź. Tak. Rozważmy język L słów postaci

    $$b_1 \cdots b_n \varphi$$

    gdzie $b_1, \ldots, b_n$ są w $\{0, 1\}$, oraz φ jest zapisem formuły o n zmiennych, prawdziwej przy wartościowaniu $b_1 \ldots b_n$. Załóżmy, że formułę φ zapisujemy za pomocą alfabetu A, który nie zawiera liter $\{0, 1\}$. Język L jest w PTIME, bo to jest po prostu ewaluacja formuł. Z drugiej strony, po zastosowaniu funkcji f, która jest identycznością na A, a która obie litery $\{0, 1\}$ zastępuje symbolem ?, dostajemy problem SAT, który jest NP-zupełny.

13. Rozważmy model niedeterministycznego automatu, gdzie kryterium akceptacji to "co najmniej połowa biegów akceptuje, spośród tych które do dochodzodzą do końca słowa". Czy taki automat rozpoznaje wyłącznie (choć niekoniecznie wszystkie) języki bezkontekstowe?

    Odpowiedź. Nie. Rozważmy automat, który wygląda tak:

    Będzie nas głównie interesowało zachowanie automatu na słowach postaci a^+b^+. Pokażemy, że z nowym kryterium akceptacji, automat akceptuje język L o własności:

    $$L \cap a^+b^+ = \{a^n b^m : 2^n \leqslant m + 1, m \geqslant 1, n \geqslant 1\}.$$

    Kluczowa własność automatu jest taka, że w czerwonej części jest niedeterminizm wykładniczy, a w niebieskiej niedeterminizm liniowy (patrz niżej co to znaczy). Przy każdej literze a, automat rozdwaja się na dwie kopie w czerwonej części (bo ze stanu 2 są przejścia do stanów $\{2, 3\}$, a ze stanu

3 są też przejścia do stanów $\{2, 3\}$). A więc przy n literach a, automat stworzy 2^n biegów, które kończą się w stanie 4. Z kolei w niebieskiej części, automat robi tylko jeden krok niedeterministycznie: raz może wybrać pomiędzy stanem 6 a 7. A więc przy m literach b, automat stworzy $m+1$ biegów, które kończą się w stanach $\{6, 7\}$. Na końcu słowa z a^+b^+, każdy bieg skończy w jednym ze stanów $\{4, 6, 7\}$. Aby automat zaakceptował, musi być więcej biegów w stanach $\{6, 7\}$ niż biegów w stanie 4.

Język L nie może być bezkontekstowy, bo i wtedy przecięcie L z regularnym językiem byłoby bezkontekstowe, a wiadomo, że $\{a^n b^m : 2^n \geqslant m+1\}$ nie jest bezkonstekstowy.

14. Czy każdy język generowany przez wyrażenie regularne jest w klasie PTIME?

    Odpowiedź. Tak. Skoro wyrażenie regularne można przerobić na automat deterministyczny, to języki te można rozpoznawać nawet w czasie liniowym.

15. Załóżmy P $\neq$ NP, bo inaczej zadanie jest oczywiste. Czy istnieje NP-trudny język rozpoznawany przez niedeterministyczny automat ze stosem?

    Odpowiedź. Nie. Algorytm CYK, który był na wykładzie, pokazuje, że każdy język bezkontekstowym można rozpoznawać w czasie wielomianowym, konkretnie $O(n^3)$.