Egzamin z teorii obliczeń, 8 czerwca 2011

1. *Automat z kolejką* nad alfabetem Σ to krotka $\langle Q, q_0, q_a, \delta\rangle$, gdzie $q_0, q_a \in Q$ oraz
$$\delta : Q - \{q_a\} \times \Sigma \to Q \times (\Sigma \oplus \{\sharp\}).$$
(Znak $\oplus$ oznacza sumę rozłączną.) *Konfiguracja* automatu to para $\langle q, w\rangle \in Q \times \Sigma^*$, a relacja przejścia $\to$ jest taka:
   - Jeśli $\delta(q,a) = (p,b)$, dla $b \neq \sharp$, to $\langle q, aw\rangle \to \langle p, awb\rangle$.
   - Jeśli $\delta(q,a) = (p,\sharp)$, to $\langle q, aw\rangle \to \langle p, w\rangle$.

   Automat *zatrzymuje się* na słowie v jeżeli $\langle q_0, v\rangle \twoheadrightarrow \langle q_a, w\rangle$, dla pewnego w. Dlaczego problem „czy dany automat zatrzymuje się na danym słowie" jest nierozstrzygalny?

2. Czy język flow-diagramów z kolejką nad skończonym alfabetem, zdefiniowaną analogicznie jak w zadaniu 1, jest uniwersalnym językiem programowania?

3. Objaśnić błąd w następującym rozumowaniu:

   Udowodnimy, że dowolny zbiór $A \subseteq \mathbb{N}$ jest rekurencyjny. W tym celu zauważmy, że jeśli $A_d = \{n \in \mathbb{N} \mid n \in A \wedge n \leq d\}$, to $A = \bigcup_{d\in\mathbb{N}} A_d$. Ponadto, każdy ze zbiorów A_d jest skończony, a zatem rekurencyjny. Aby więc sprawdzić, czy $n \in A$, wystarczy wziąć $d \geq n$ i zapytać, czy $n \in A_d$.

4. Funkcja Ackermanna określona równaniami $A(0,x) = x+1$, $A(n+1,0) = A(n,1)$, $A(n+1,x+1) = A(n,A(n+1,x))$ nie jest pierwotnie rekurencyjna. Pokazać, że trójargumentowa relacja $A(n,x) = y$ jest pierwotnie rekurencyjna.[1]

Rozwiązania:

Zadanie 1: Bo za pomocą kolejki można symulować maszynę Turinga. Konfigurację maszyny umieścimy w kolejce w postaci słowa wqv. W każdej kolejnej fazie pracy automat przegląda zawartość kolejki, odkładając przeczytane litery na koniec. Robi to z opóźnieniem jednego kroku, pamiętając ostatnio przeczytany symbol. Po natrafieniu na stan maszyny Turinga automat dokonuje odpowiedniej korekty zawartości kolejki. Po każdej takiej fazie stan kolejki przedstawia nastepną konfigurację maszyny. W ten sposób redukujemy problem stopu dla maszyn Turinga do problemu stopu dla automatów z kolejką: dana maszyna o stanie początkowym s_0 zatrzymuje się na słowie w wtedy i tylko wtedy, gdy skonstruowany przez nas automat zatrzymuje się na słowie s_0w.

Zadanie 2: Nie, bo flow-diagram z kolejką nad skończonym alfabetem nie będzie w stanie obliczyć wartości termu o złożoności kamyczkowej większej niż liczba zmiennych programu.

Zadanie 3: Dla każdego ze zbiorów A_d istnieje algorytm rozstrzygający, ale być może dla każdego inny. Aby opisana procedura mogła rozstrzygać czy $n \in A$ musielibyśmy umieć efektywnie wskazywać taki algorytm[2] w zależności od d.

Zadanie 4: Kłopot z funkcją Ackermanna jest taki, że nie umiemy z góry oszacować jakie wywołania rekurencyjne będą potrzebne dla obliczenia $A(n,x)$. Ale jeśli mamy tylko *sprawdzić*

[1] Wskazówka: funkcja A jest monotoniczna ze względu na oba argumenty: $A(n,x) < A(n+1,x), A(n,x+1)$.
[2] czyli (jak zauważył pan Marcin Kościelnicki) w istocie *numer* zbioru A_d.

czy $A(n,x) = y$ dla danego y, to jest inaczej. Obliczenie $A(n,x)$ odwołuje się bowiem wyłącznie do wartości $A(m,z) \leq y$, dla $m \leq n$ i $z \leq y$. A więc sprawdzenia czy $A(n,x) = y$ można dokonać za pomocą tablicy rozmiaru $n \times y$. Tablica jest początkowo pusta; w kolejnych krokach wpisujemy do niej te wartości $A(m,z)$, które wynikają bezpośrednio z równań definiujących A przez odwołanie do już wypełnionych pól tablicy. Postępujemy tak, aż znajdziemy wartość $A(n,x)$, którą przyrównamy do y. Tablica jest modyfikowana co najwyżej ny razy i zawiera liczby nie większe od y, więc nasz algorytm wymaga tylko rekursji prostej (pętli for).