Egzamin z teorii obliczeń, 24 czerwca 2017

1. Udowodnić, że następujące problemy są nierozstrzygalne:

   Dana deterministyczna maszyna Turinga M. Czy istnieje takie słowo $w \in L(M)$, że:

   (a) Liczba zer w słowie w jest podzielna przez 13?

   (b) Akceptując słowo w maszyna M używa wszystkich swoich stanów?

2. Liczba n jest *pozornym punktem stałym* funkcji f, gdy $\varphi_n = \varphi_{f(n)}$. Udowodnić, że każda funkcja rekurencyjna ma co najmniej dwa pozorne punkty stałe. A więcej?

   Wskazówka: Niech n będzie pozornym punktem stałym funkcji f. Jakie pozorne punkty stałe ma funkcja $g = \lambda i.\, \texttt{if}\ i = n\ \texttt{then}\ m\ \texttt{else}\ f(i)$, jeśli odpowiednio wybierzemy m?

3. Flow-diagramy z (unarnymi) *tablicami* mogą, oprócz zwykłych instrukcji, używać podstawień postaci $F(x) := t$ oraz $y := F(x)$, gdzie F jest dodatkowym symbolem funkcyjnym (nazwą zmiennej tablicowej). Mamy więc dodatkową pamięć zorganizowaną w tablicę. Pokazać, że klasa flow-diagramów z tablicami:

   (a) jest „semi-uniwersalna", tj. dla dowolnej sygnatury istnieje program P z jedną zmienną wejściową, który zatrzymuje się wtedy i tylko wtedy, gdy dana wejściowa generuje skończoną podalgebrę;[1]

   (b) ale nie jest uniwersalna, bo ma rozstrzygalny problem stopu na skończonych interpretacjach.

   (c) I nie będzie nawet semi-uniwersalna, jeśli zamiast tablic funkcyjnych będzie można używać tylko tablic relacyjnych o wartościach zerojedynkowych.[2]

[1] Można założyć, że sygnatura składa się tylko z jednego symbolu funkcyjnego f, który ma dwa argumenty.

[2] Przyjmijmy za oczywiste to, że program P, o którym mowa w definicji (3a), musi zawsze obliczać wszystkie elementy takiej skończonej podalgebry, bo inaczej mógłby się „pomylić" i zatrzymać kiedy nie trzeba.

Rozwiązania:

Zadanie 1a: Własność „*istnieje takie $w \in L(M)$, że liczba zer w słowie w jest podzielna przez 13*" jest nietrywialna i adekwatna (jest to własność języka, a nie maszyny). Dlatego nierozstrzygalność tego problemu wynika wprost z twierdzenia Rice'a.

Zadanie 1b: Zredukujemy do tego pytania problem „*czy deterministyczna maszyna N akceptuje słowo puste*", konstruując maszynę M_N. Ta maszyna ma jeden dodatkowy symbol pomocniczy # i jeden dodatkowy stan f, który jest jej jedynym stanem akceptującym. Pozostałe symbole i stany są takie same jak symbole i stany maszyny N. (Bez straty ogólności można założyć, że N ma tylko jeden stan akceptujący.) Maszyna M_N działa początkowo tak samo jak N. Po wejściu w stan akceptujący maszyny N, zamiast się zatrzymać, maszyna M_N wpisuje na taśmie # i kolejno przebiega wszystkie swoje stany w pewnej ustalonej kolejności (nie przesuwając głowicy i nic nie pisząc). Ostatnim z tych stanów jest nowy stan akceptujący f.

Zadanie 2: Na mocy twierdzenia o rekursji każda funkcja rekurencyjna ma co najmniej jeden pozorny punkt stały. Niech więc n będzie pozornym punktem stałym funkcji f i niech m będzie takie, że $\varphi_n \neq \varphi_m$. Funkcja g ze wskazówki jest rekurencyjna i też ma pozorny punkt stały k. Nie może być tak, że $k = n$, bo wtedy byłoby $\varphi_n = \varphi_{g(n)} = \varphi_m$. Zatem $k \neq n$ i mamy $\varphi_k = \varphi_{g(k)} = \varphi_{f(k)}$, czyli k jest drugim pozornym punktem stałym funkcji f. W istocie jest ich nieskończenie wiele, gdyby bowiem zbiór A pozornych punktów stałych f był skończony, to moglibyśmy zastosować to samo rozumowanie do funkcji $g = \lambda i.\, \texttt{if}\ i \in A\ \texttt{then}\ m\ \texttt{else}\ f(i)$, wybierając takie m, że $\varphi_m \neq \varphi_a$, dla $a \in A$.

Zadanie 3a: Mając jednowymiarową tablicę F możemy konstruować różnowartościowy ciąg a_n elementów podalgebry generowanej przez wartość wejściową a_0 i zapisywać go w tablicy w ten sposób, że $F(a_n) = a_{n+1}$. W każdej fazie pracy dodajemy do tablicy jeden nowy element, a jeśli takiego nie znajdziemy, to zatrzymujemy się. Na początku takiej fazy mamy określone wartości $F(a_i) = a_{i+1}$ dla $i = 0, \ldots, n-1$ i mamy zmienne *first* i *last* o wartościach odpowiednio a_0 i a_n. Zaczynając od $x, y := \mathit{first}$, obliczamy wartości $f(x, y)$ dla x i y pomiędzy *first* i *last* (na przykład w porządku leksykograficznym) i sprawdzamy (w wewnętrznej pętli) czy występują w tablicy. Po wykryciu nowej wartości wpisujemy ją jako $F(\mathit{last})$, po czym wykonujemy $\mathit{last} := F(\mathit{last})$ i zaczynamy kolejną fazę.

Zadanie 3b: Znaczeniem tablicy w interpretacji A jest funkcja z A do A. Jeśli interpretacja jest skończona, to takich funkcji jest skończenie wiele. Zatem program z tablicami może przyjąć tylko ograniczoną liczbę różnych konfiguracji. Długość obliczenia skończonego nie może być większa niż to ograniczenie, więc można w skończonym czasie rozstrzygnąć, czy takie istnieje.

Zadanie 3c: Mając skończoną liczbę n zmiennych, program z tablicami może obliczać tylko wartości termów o złożoności rejestrowej (kamyczkowej) co najwyżej n. Tablice binarne nic tu nie pomogą. Dlatego nie da się przeglądać wszystkich elementów podstruktury generowanej przez wartość wejściową.