Programowanie 2004

Egzamin zasadniczy

14 czerwca 2004

Egzamin trwa trzy godziny zegarowe. Za każde z czterech zadań można otrzymać od -5 do 25 punktów, zatem za cały egzamin można otrzymać do 100 punktów. Za przedstawienie zadania do oceny otrzymuje się -5 punktów. Za brak rozwiązania zadania otrzymuje się 0 punktów. Punkty z egzaminu zasadniczego przeliczają się na oceny następująco: mniej niż 33: ndst, od 33 do 45: dst, od 46 do 58: dst+, od 59 do 71: db, od 72 do 84: db+, od 85: bdb. Ocena z egzaminu zasadniczego jest wystawiana na podstawie sumy punktów z egzaminu i punktów bonusowych. Liczba punktów bonusowych jest częścią całkowitą ilorazu: $(C - 40)/10$, gdzie C oznacza całkowitą liczbę punktów uzyskanych na zaliczenie ćwiczeń. Punkty bonusowe z ćwiczeń dolicza się tylko do wyników egzaminu zasadniczego i tylko w przypadku, gdy ćwiczenia te zaliczono w tym samym semestrze, w którym odbywa się egzamin. Zatem za ćwiczenia zaliczone w poprzednich latach punktów bonusowych nie dopisuje się.

Każde zadanie proszę pisać na osobnych kartkach. Proszę podpisać własnym imieniem i nazwiskiem każdą kartkę przedstawianą do oceny. Na zakończenie egzaminu proszę oddać do sprawdzenia tylko kartki z rozwiązanymi zadaniami. Brudnopisy i treści zadań proszę zatrzymać.

Zadanie 1 (25 pkt). Rozważmy rachunek lambda z typami prostymi, zadany następującą składnią abstrakcyjną:

$$\begin{aligned} M &::= x \mid M_1 M_2 \mid \lambda x.M_1 \\ \sigma &::= \alpha \mid \sigma_1 \to \sigma_2 \end{aligned}$$

oraz regułami typowania:

$$\frac{}{\Gamma, x : \sigma \vdash x : \sigma}\ (\text{Var}) \quad \frac{\Gamma \vdash M_1 : \sigma \to \tau \quad \Gamma \vdash M_2 : \sigma}{\Gamma \vdash M_1 M_2 : \tau}\ (\text{App}) \quad \frac{\Gamma, x : \sigma \vdash M_1 : \tau}{\Gamma \vdash \lambda x.M_1 : \sigma \to \tau}\ (\text{Abs})$$

Niech NF oznacza zbiór termów w postaci normalnej (względem β- i η-redukcji). Udowodnij, że

$$\{M \in \text{NF} \mid \vdash M : \alpha \to (\alpha \to \alpha) \to \alpha\} \quad = \quad \{\lambda x.\lambda y.y^{(n)}x \mid n \in \mathbb{N}\},$$

gdzie

$$M_1^{(n)} M_2 \quad \equiv \quad \underbrace{M_1(\ldots M_1(}_{n} M_2)\ldots),$$

tj., bardziej formalnie,

$$\begin{aligned} M_1^{(0)} M_2 &\equiv M_2, \\ M_1^{(n+1)} M_2 &\equiv M_1(M_1^{(n)} M_2), \end{aligned}$$

dla dowolnych termów M_1 i M_2.

Zadanie 2 (25 pkt). Udowodnij, że jedynym językiem $L \subseteq \{0,1\}^*$ spełniającym równość

$$L \quad = \quad \{\epsilon\} \cup \{0\}L\{1\}$$

jest język

$$L \quad = \quad \{0^n 1^n \mid n \in \mathbb{N}\}.$$

Przypomnijmy, że dla dowolnych języków L_1, L_2 operację konkatenacji zdefiniowaliśmy następująco:

$$L_1 L_2 \quad = \quad \{u_1 u_2 \mid u_1 \in L_1, u_2 \in L_2\}.$$

Zadanie 3 (25 pkt). Rozważmy dwa procesy P_1 i P_2 działające współbieżnie w środowisku z pamięcią dzieloną. Przypuśćmy, że procesy nie potrafią wykonywać żadnych operacji synchronizujących i komunikują się odczytując i zapisując odpowiednie wartości w ustalonych komórkach pamięci. Przypuśćmy dalej, że niepodzielnymi akcjami elementarnymi są odczyt i zapis wartości z/do komórek pamięci, ale operacje te mogą ulec przeplotowi (tj. np. operacja zwiększenia zawartości komórki pamięci o jeden nie jest akcją elementarną). Pokaż, jak w takim środowisku zrealizować wzajemne wykluczanie procesów w sekcji krytycznej. Innymi słowy przedstaw algorytm Dekkera. Możesz użyć dowolnej, byle zrozumiałej składni, np. adopodobnej.

Zadanie 4 (25 pkt). Oto definicja składni abstrakcyjnej języka Prolog (a dokładniej jego podzbioru nie zawierającego arytmetyki i operacji odcięcia), zwanego niekiedy *czystym Prologiem*.
Rozważmy zbiór gatunków $\mathcal{S} = \{\mathit{term}, \mathit{pred}\}$. Typ algebraiczny

$$\pi_n \quad = \quad \underbrace{\mathit{term} \times \ldots \times \mathit{term}}_{n} \to \mathit{pred}$$

będziemy nazywać *typem predykatu o arności n*, zaś typ

$$\sigma_n \quad = \quad \underbrace{\mathit{term} \times \ldots \times \mathit{term}}_{n} \to \mathit{term}$$

typem funktora o arności n. Niech Σ_n oznacza zbiór symboli typu σ_n, zaś Π_n zbiór symboli typu π_n. W szczególności Σ_0, to zbiór *atomów*. Niech $\mathcal{X}$ będzie zbiorem zmiennych gatunku *term*. Zmiennych gatunku *pred* nie mamy. Rozważmy termy $\mathcal{T}^{\mathit{term}}(\{\Sigma_n\}_{n=0}^{\infty}, \mathcal{X})$ i $\mathcal{T}^{\mathit{pred}}(\{\Pi_n\}_{n=0}^{\infty}, \emptyset)$. *Klauzulą hornowską* nazywamy ciąg termów $\mathcal{C} = \langle p_0, p_1, \ldots, p_n \rangle$, gdzie $n \geq 0$ i $p_i \in \mathcal{T}^{\mathit{pred}}(\{\Pi_n\}_{n=0}^{\infty}, \emptyset)$. Klauzule zwykle zapisujemy w postaci p_0`:-`$p_1, \ldots, p_n$. Gdy $n = 0$, to klauzulę nazywamy *faktem*. *Programem w języku Prolog* nazywamy ciąg klauzul $\mathcal{P} = \langle \mathcal{C}_1, \ldots, \mathcal{C}_m \rangle$, $m \geq 0$. *Zapytaniem* (*celem*) nazywamy ciąg termów $\mathcal{Q} = \langle q_1, \ldots, q_n \rangle$, gdzie $n \geq 0$.

Programy, które się nie zapętlają, nazywamy *skończonymi*. Zdefiniuj semantykę denotacyjną lub operacyjną skończonych programów w czystym Prologu.

Wskazówka: Semantyka powinna określać, dla jakich zapytań Prolog odpowie `Yes`, a dla jakich `No`. Dlatego denotacją zapytania może być zbiór wszystkich jego stałych instancji, dla których Prolog odpowie `Yes`. Zbiór ten łatwo zadać jako najmniejszy punkt stały odpowiedniego operatora (semantyka denotacyjna) lub przy pomocy zbioru reguł wnioskowania (semantyka operacyjna).

Powodzenia!