Metody programowania

Egzamin zasadniczy

24 czerwca 2015

Liczba punktów	Ocena
0 – 16	2.0
17 – 20	3.0
21 – 22	3.5
23 – 24	4.0
25 – 26	4.5
27 – 30	5.0

W każdym pytaniu testowym proszę wyraźnie zaznaczyć dokładnie jedną odpowiedź. Jeśli zostanie zaznaczona więcej niż jedna odpowiedź, to za wybraną zostanie uznana ta, która *nie jest* otoczona kółkiem. Każde pytanie jest warte 1 punkt. Czas trwania egzaminu: 120 minut.

Pytanie 1. Rozważmy predykaty:

```
p(0).
p(1).

q(0).
q(N) :-
   N1 is N+1,
   q(N1).
```

Obliczenie celu

```
?- p(X), q(Y).
```

a. ☐ zakończy się niepowodzeniem.

b. ■ zakończy się sukcesem, a po nawrocie błędem.

c. ☐ będzie mieć nieskończenie wiele sukcesów, w których za zmienną `X` zawsze będzie podstawiona wartość 0.

d. ☐ będzie mieć nieskończenie wiele sukcesów, w których za zmienne (`X`, `Y`) będą kolejno podstawiane pary liczb naturalnych $(0, 0), (1, 0), (0, 1), (1, 1), (0, 2), (1, 2), \ldots$

Pytanie 2. Rozważmy predykat:

```
p([],X,X).
p([H|T],X,Y) :-
   p(T,[H|X],Y),
   !.
```

Wtedy:

- a. ☐ Odcięcie na końcu drugiej klauzuli predykatu `p` jest zielone.
- b. ☐ Odcięcie na końcu drugiej klauzuli predykatu `p` nie ma żadnego wpływu na działanie predykatu.
- c. ■ Cele `p([],X,Y)` oraz `X=Y` mają ten sam efekt.
- d. ☐ Cele `p(X,[],Y)` oraz `reverse(X,Y)` mają ten sam efekt.

Pytanie 3. Efektem obliczenia celu

```
?- append(X,a,[]).
```

jest

- a. ■ niepowodzenie.
- b. ☐ błąd typu, gdyż atom `a` nie jest listą.
- c. ☐ sukces, w którym pod zmienną `X` jest podstawiona lista `[a]`.
- d. ☐ zapętlenie.

Pytanie 4. Efektem obliczenia celu

```
?- X=a, \+ \+ X=b.
```

jest

- a. ☐ sukces, w którym zmienna `X` pozostaje nieukonkretniona.
- b. ☐ sukces, w którym pod zmienną `X` jest podstawiony atom `a`.
- c. ☐ sukces, w którym pod zmienną `X` jest podstawiony atom `b`.
- d. ■ niepowodzenie.

Pytanie 5. Cel

```
?- \+ X=a, X=b.
```

- a. ■ kończy się niepowodzeniem.
- b. ☐ kończy się sukcesem, w którym pod zmienną `X` jest podstawiony atom `a`.
- c. ☐ kończy się sukcesem, w którym pod zmienną `X` jest podstawiony atom `b`.
- d. ☐ ma ten sam efekt, co cel `?- X=b, \+ X=a.`

Pytanie 6. Obliczenie celu

```
?- (N is 0; N is N+1), write(N), nl, fail.
```

spowoduje

a. ☐ wypisywanie w nieskończoność do standardowego strumienia wyjściowego kolejnych liczb naturalnych począwszy od zera, po jednej w wierszu.

b. ☐ wypisywanie w nieskończoność do standardowego strumienia wyjściowego cyfr `0`, po jednej w wierszu.

c. ☐ wypisanie do standardowego strumienia wyjściowego pojedynczej cyfry `0` i znaku nowego wiersza, po czym obliczenie się zakończy.

d. ■ wypisanie do standardowego strumienia wyjściowego pojedynczej cyfry `0` i znaku nowego wiersza, po czym obliczenie zostanie przerwane z komunikatem o błędzie.

Pytanie 7. Obliczenie celu

```
?- \+ (between(0,6,X), X is 13).
```

a. ☐ zakończy się niepowodzeniem.

b. ■ zakończy się pojedynczym sukcesem, w którym zmienna `X` pozostanie nieukonkretniona.

c. ☐ zakończy się pojedynczym sukcesem, w którym pod zmienną `X` zostanie podstawiona liczba 13.

d. ☐ zakończy się błędem arytmetycznym.

Uwaga: Obliczenie celu `between`(m, n, x), gdzie `between/3` jest predykatem dostępnym w SWI-Prologu, a m i n są liczbami całkowitymi, powoduje przy kolejnych nawrotach unifikację termu x z kolejnymi liczbami $m, m+1, m+2, \ldots, n$.

Pytanie 8. Struktura `[[a|b]|[c|d]]`

a. ☐ jest listą dwuelementową.

b. ☐ jest listą trzyelementową.

c. ☐ unifikuje się ze strukturą `[[a|b],[c|d]]`.

d. ■ unifikuje się ze strukturą `[[a|b],c|d]`.

Pytanie 9. Obliczenie celu

```
?- append(X,[a],X).
```

a. ☐ zakończy się niepowodzeniem.

b. ☐ zakończy się pojedynczym sukcesem.

c. ■ zapętli się.

d. ☐ będzie mieć nieskończenie wiele sukcesów.

Pytanie 10. Niech

```
p :- write(a).
p :-
   p.
```

Rozważmy cel

```
?- !, p.
```

a. ■ Cel będzie spełniony na nieskończenie wiele sposobów. Przed każdym sukcesem do standardowego strumienia wyjściowego będzie wypisywana pojedyncza litera a.

b. □ Cel będzie spełniony na nieskończenie wiele sposobów. Przed pierwszym sukcesem do standardowego strumienia wyjściowego zostanie wypisana pojedyncza litera a.

c. □ Cel będzie spełniony na jeden sposób, a do standardowego strumienia wyjściowego zostanie wypisana pojedyncza litera a.

d. □ Obliczenie celu zapętli się, a do standardowego strumienia wyjściowego zostanie wypisana nieskończenie wiele razy litera a.

Pytanie 11. Rozważmy predykat

```
collect(N,N,[N]).
collect(N,M,[N|L]) :-
   N1 is N+1,
   collect(N1,M,L).
```

i cel

```
?- collect(0,5,L), !, member(X,L).
```

Odcięcie, które występuje w podanym celu

a. □ nie ma wpływu na wynik jego obliczenia.

b. □ powoduje, że przy kolejnych nawrotach pod zmienną `X` są podstawiane jedynie liczby mniejsze od 5.

c. ■ powoduje, że obliczenie tego celu nie zapętli się przy żadnym nawrocie.

d. □ powoduje, że cel jest spełniony tylko na jeden sposób.

Pytanie 12. Niech

```
form --> [p].
form --> [q].
form --> form, binop, form.
binop --> [and].
binop --> [or].
```

Obliczenie celu

```
?- form([p,and,q],[]).
```

a. ☐ zakończy się niepowodzeniem.

b. ☐ zakończy się pojedynczym sukcesem.

c. ■ zakończy się sukcesem, a po nawrocie zapętli się.

d. ☐ będzie mieć nieskończenie wiele sukcesów.

Pytanie 13. Rozważmy predykat

```
combine([],X,[],X).
combine([A|TA],X,[A,B|R],TB) :-
   combine(TA,X,R,[B|TB]).
```

Obliczenie celu

```
?- combine([1,2,3],[4,5,6],R,[]).
```

a. ☐ zakończy się niepowodzeniem.

b. ☐ zakończy się pojedynczym sukcesem, w którym zmienna `R` zostanie zunifikowana z listą `[1,2,3,4,5,6]`.

c. ■ zakończy się pojedynczym sukcesem, w którym zmienna `R` zostanie zunifikowana z listą `[1,6,2,5,3,4]`.

d. ☐ zakończy się pojedynczym sukcesem, w którym zmienna `R` zostanie zunifikowana z listą `[3,2,1,4,5,6]`.

Pytanie 14. Niech

```
p([H|T]-[H|Y],T-Y).
```

Dla zadanej otwartej listy różnicowej `X` obliczenie celu

```
?- p(X,Y).
```

spowoduje, że zmienna `Y` zunifikuje się z:

a. ☐ listą różnicową zawierającą elementy listy `X` w odwrotnej kolejności.

b. ☐ listą różnicową zawierającą te same elementy, co lista `X`.

c. ■ listą różnicową zawierającą te same elementy, co ogon listy `X` z dodaną na końcu głową listy `X`.

d. ☐ listą różnicową otrzymaną przez przestawienie ostatniego elementu listy `X` na jej początek.

Pytanie 15. Obliczenie celu

```
?- X is 1, Y is 2, X \= Y.
```

a. ☐ będzie miało taki sam efekt, jak obliczenie celu

```
?- X \= Y, X is 1, Y is 2.
```

b. ■ będzie miało taki sam efekt, jak obliczenie celu

```
?- X = 1, Y = 2.
```

c. ☐ zakończy się niepowodzeniem.

d. ☐ zakończy się błędem arytmetycznym.

Pytanie 16. Nech

```
newtype Number = Number Integer
nonzero (Number 0) = False
nonzero _ = True
```

Wtedy

a. ☐ typem funkcji `nonzero` jest `Number a => a → Bool`.

b. ☐ wyrażenia `Number undefined` i `undefined` mają ten sam typ.

c. ■ wyrażenia `Number undefined` i `undefined` mają tę samą wartość.

d. ☐ wartością wyrażenia `nonzero (Number undefined)` jest `True`.

Pytanie 17. Jednym z aksjomatów równościowych monady jest

a. ☐ $\lambda\, x \to (p >\!\!>\!\!= (q >\!\!>\!\!= r)) \;=\; (\lambda\, x \to (p >\!\!>\!\!= q)) >\!\!>\!\!= r$.

b. ■ $\mathrm{do}\ \{\ x \leftarrow p;\ q\ x >\!\!>\!\!= r\ \} \;=\; \mathrm{do}\ \{\ x \leftarrow p;\ q\ x\ \} >\!\!>\!\!= r$.

c. ☐ $\mathtt{return} >\!\!>\!\!= p \;=\; p$.

d. ☐ $p >\!\!>\!\!= (\lambda\, x \to \mathtt{return}) \;=\; p$.

Pytanie 18. Jeżeli funkcja f jest *strict*, to

a. ☐ $f\ 1 \neq \perp$.

b. ☐ $f \perp \neq \perp$.

c. ■ $f\ (\mathtt{head}\ [\,]) = \perp$.

d. ☐ $f\ (\mathtt{head}\ [\perp]) \neq f \perp$.

Pytanie 19. Niech

```
data T = T (Bool,Bool)
newtype S = S (Bool,Bool)
```

Wtedy

- a. ■ $\texttt{S}\perp = \perp$.
- b. □ $\texttt{S}\perp = \texttt{T}\perp$.
- c. □ $\texttt{S}\,(\perp,\perp) = \texttt{S}\perp$.
- d. □ $\texttt{T}\,(\perp,\perp) = \texttt{T}\perp$.

Pytanie 20. Równość

$$\texttt{foldl}\ (+)\ 0\ \mathit{xs}\ =\ \texttt{foldr}\ (+)\ 0\ \mathit{xs},$$

gdzie $\mathit{xs} :: \texttt{Num}\ a \Rightarrow [a]$,

- a. ■ zachodzi dla każdej listy *xs*.
- b. □ nie zachodzi dla dowolnej nieskończonej listy *xs*.
- c. □ nie zachodzi dla pewnej częściowej listy *xs*.
- d. □ zachodzi tylko dla tych spośród nieskończonych list *xs*, w których występuje jedynie skończenie wiele elementów różnych od 0.

Pytanie 21. Niech

```
f x = x (f x)
```

- a. □ Funkcja `f` nie posiada typu.
- b. ■ Najogólniejszym typem funkcji `f` jest `(a → a) → a`.
- c. □ Wyrażenie `(head . f)(1:)` nie posiada typu.
- d. □ Wartością wyrażenia `(head . f)(1:)` jest $\perp$.

Pytanie 22. Niech dla pewnych funkcji f i g funkcja h będzie zdefiniowana wzorem

$h\ x\ y = f\ (g\ x\ y)$

Wtedy funkcja h jest równa

- a. □ $f\ .\ g$
- b. ■ $(f\ .)\ .\ g$
- c. □ $(.)\ f\ (.\ g)$
- d. □ $f\ (.\ g)$

Pytanie 23. Funkcja f . g jest *strict* wtedy i tylko wtedy, gdy

a. ☐ f jest *strict*.

b. ☐ g jest *strict*.

c. ☐ f i g są *strict*.

d. ☐ Żaden z powyższych warunków nie jest konieczny do tego, by funkcja f . g była strict.

Pytanie 24. Równość

$$\texttt{foldr}\ (\oplus)\ a\ xs = \texttt{foldl}\ (\oplus)\ a\ xs,$$

gdzie $(\oplus) :: t \to t \to t$, $a :: t$, $xs :: [t]$, dla pewnego typu t, zachodzi dla

a. ☐ dowolnej funkcji $(\oplus)$, dowolnej wartości a i dowolnej listy xs.

b. ☐ dowolnej funkcji $(\oplus)$, dowolnej wartości a i dowolnej skończonej listy xs.

c. ☐ dowolnej listy xs, jeśli $(t, (\oplus), a)$ jest monoidem.

d. ■ dowolnej skończonej listy xs, jeśli $(t, (\oplus), a)$ jest monoidem.

Pytanie 25. Niech

```
evil :: [Integer] → Bool
evil [] = False
evil (666:xs) = True
evil (_:xs) = evil xs
lucky :: [Integer]
lucky = 7:lucky
```

Wtedy wyrażenie `evil` (`lucky` ++ [666]) jest równe wyrażeniu

a. ☐ `True`.

b. ☐ `False`.

c. ■ `evil` (`lucky` ++ `lucky`).

d. ☐ `evil` ([666] ++ `lucky`).

Pytanie 26. Typ `(a → b → c) → (a → b) → (a → c)` jest najogólniejszym typem funkcji `f` zdefiniowanej następująco:

a. ☐ `f x y z = y (x y) z`

b. ☐ `f x y z = x y (x z)`

c. ■ `f x y z = x z (y z)`

d. ☐ `f x y z = x y (y z)`

Pytanie 27. Typem wyrażenia `map map` jest

a. ☐ `[a → b] → ([a] → [b]) → (a → b) → [[a] → [b]]`

b. ☐ `((a → b) → [a] → [b]) → [a → b] → [[a] → [b]]`

c. ■ `[a → b] → [[a] → [b]]`

d. ☐ `(a → b) → ([a] → [b])`

Pytanie 28. Wyrażenia

`[(x,y) | x ← xs, p x, y ← ys]` oraz `[(x,y) | x ← xs, y ← ys, p x]`

gdzie `xs :: [a]`, `ys :: [b]` i `p :: a → Bool`, dla pewnych typów a i b, są równe:

a. ☐ dla dowolnych list `xs` i `ys` i dowolnej funkcji `p`, która jest *strict*.

b. ☐ dla dowolnych skończonych list `xs` i `ys` i dowolnej funkcji `p`.

c. ☐ dla dowolnych częściowych list `xs` i `ys` i dowolnej funkcji `p`.

d. ■ Żadna z powyższych odpowiedzi nie jest poprawna.

Pytanie 29. Wartością wyrażenia `show (42,True)` jest

a. ☐ `"(\"42\",\"True\")"`

b. ■ `"(42,True)"`

c. ☐ `("42","True")`

d. ☐ `"\"(42,True)\""`

Pytanie 30. Niech

```
fst :: (a,b) → a
fst (x,_) = x
snd :: (a,b) → b
snd (_,y) = y
pair :: (a → b, a → c) → a → (b,c)
pair (f,g) x = (f x, g x)
k, h :: (a → b, c → d) → (a,c) → (b,d)
k (f,g) = pair (f . fst, g . snd)
h (f,g) (x,y) = (f x, g y)
```

Wtedy

a. ☐ $\mathtt{k} = \mathtt{h}$.

b. ☐ $\mathtt{k}\,(f,g) = \mathtt{h}\,(f,g)$ dla dowolnych f :: `a → b` i g :: `c → d`.

c. ■ $\mathtt{k}\ p\,(x,y) = \mathtt{h}\ p\,(x,y)$ dla dowolnego p :: (`a → b`, `c → d`) i dowolnych x :: `a` i y :: `c`.

d. ☐ $\mathtt{k}\,(f,g)\ p = \mathtt{h}\,(f,g)\ p$ dla dowolnych f :: `a → b` i g :: `c → d` i dowolnego p :: (a, c).

Uwagi do zadań

W zadaniu 23 warunek f . g jest *strict* implikuje, że f jest *strict*, a w drugą stronę warunek f i g są *strict* implikuje, że f . g jest *strict*. Zatem żadna z czterech odpowiedzi nie jest poprawna. *Zadanie jest wadliwe i zostało anulowane. Wszystkie osoby otrzymały za to zadanie punkt niezależnie od wybranej odpowiedzi.*

W zadaniu 28 kontrprzykładem dla opowiedzi a. i b. jest np. $\mathtt{xs} = [0]$, $\mathtt{ys} = [\,]$, $\mathtt{p} = \perp$. Kontrprzykładem dla odpowiedzi c. jest np. $\mathtt{xs} = 0 : 1 : \perp$, $\mathtt{ys} = 0 : \perp$ i $\mathtt{p}\ x = x\ \text{`}\mathtt{mod}\text{`}\ 2 \mathrel{/=} 0$.

W zadaniu 29, jeśli założyć, że prawdziwa jest tylko jedna z odpowiedzi, to łatwo wywnioskować, że jest nią odpowiedź c., gdyż na mocy zasady ekstensjonalności odpowiedź a. implikuje pozostałe, zaś odpowiedzi b. i d. są równoważne. Można więc wskazać właściwą dopowiedź nie rozwiązując zadania! Funkcje `h` i `k`, to funkcje `cross` i `cross'` ze stron 42–43 podręcznika. Tam znajduje się dokładna analiza tego zadania.