Linda
(zadanie zaliczeniowe)

Michał R. Przybyłek

Należy zaprojektować serwer „Lindy" oferujący następujące usługi:

- tworzenie/usuwanie przestrzeni krotek o zadanej strukturze
- wstawianie/usuwanie/podglądanie krotek w zadanej przestrzeni

Serwer jako parametr startowy pobiera numer portu, na którym będzie nasłuchiwał komunikatów z żądaniami od klientów. Komunikaty mają strukturę dokumentów XML'owych opisaną w pliku „linda.dtd".

W świecie Lindy istnieją typy proste (w szczególności: integer, float, string, binary) i złożone (tj. tuple). Na wszystkich występujących typach określony jest częściowy porządek. Na typach prostych jest to ich naturalny porządek. Na tuplach porządek określamy po współrzędnych, tj. dla tupli t, s zachodzi $t \leq s$ witw. gdy $\forall_i t_i \leq s_i$, gdzie przez x_i oznaczamy i-tą współrzędną tupli x. Ponadto istnieje pewna pseudowartość $\perp$, która przy dowolnych porównaniach zawsze daje prawdę.

Semantyka komunikatów określona jest następująco:

1. $[\![\langle\text{simple type="T"}\rangle x\langle\backslash\text{simple }\rangle]\!] = x{:}\,T$
2. $[\![\langle\text{tuple}\rangle x_1, x_2, \ldots, x_n\langle\backslash\text{tuple}\rangle]\!] = ([\![x_1]\!], [\![x_2]\!], \ldots, [\![x_n]\!])$
3. $[\![\langle\text{blank }\backslash\rangle]\!] = \perp$
4. $[\![\langle\text{value test=ęq-less"}\rangle x\langle\backslash\text{value }\rangle]\!] = y \mapsto y \leq [\![x]\!]$ i analogicznie dla pozostałych operatorów
5. $[\![\langle\text{and}\rangle x_1, x_2, \ldots, x_n\langle\backslash\text{and}\rangle]\!] = [\![x_1]\!] \wedge [\![x_2]\!] \wedge \ldots \wedge [\![x_n]\!]$ i analogicznie dla pozostałych operatorów

Komunikat „request” przyjmuje jako atrybut „space” nazwę przestrzeni krotek, na której bedzie wykonywał żądanie klienta i jako atrybut „kind” typ żadania: „get”/„lookup” (wyjęcie/podejrzenie w przestrzeni krotek dowolnej tupli spełniającej wyspecyfikowaną formułę), „put” (włożenie do przestrzeni krotek tupli o zadanej wartości). Przy żądaniu „put” w ciele znacznika „request” może wystąpić tylko znacznik „value” z wartością „test” określoną na „equal” i nie mogą wystąpić jakiekolwiek pseudowartości. Odpowiedzią na żądania jest komunikat „response” ze znacznikiem „ok” (z dołączoną wartością przy komunikacie „get”) w przypadku operacji zakończonej sukcesem . Żądania mają być nieblokujące, tj. przy niemożliwości wykonania zadanego żądania zwracana jest natychmiastowo odpowiedź „fail”.

Komunikaty „create” i „drop” przyjmują w atrybucie „name” nazwę przestrzeni krotek do utworzenia, bądź usunięcia. Specyfikacja struktury tupli mogących być przechowywanych w przestrzeni określana jest znacznikami „simple-type” (dla typów prostych) i „tuple-type” (dla tupli). Dla „simple-type” konkretny typ prosty ustala się w atrybucie „type”. Przy udanym utworzeniu/usunięciu przestrzeni zwracany jest pusta odpowiedź „ok”, natomiast przy jakimkolwiek błędzie — „fail”.