Teoretyczne podstawy języków programowania

Lista 12

12.01.2005

Zadanie 1. Java wykorzystuje kowariantne podtypowanie dla tablic.

$$\frac{\sigma <: \tau}{\text{Array } \sigma <: \text{Array } \tau} \ (\text{Array} <:)$$

Napisz możliwie krótki program w Javie pokazujący, że nie jest to reguła bezpieczna (program zgłasza wyjątek `java.lang.ArrayStoreException`).

W zadaniach 2–3 należy udowodnić lematy dotyczące systemu $\lambda_{<:}^{\to,\top,\{\}}$.

Zadanie 2. Lemat

Jeśli $\mathcal{C} :: \sigma <: \tau$, to $\vdash [\![\mathcal{C}]\!] : [\![\sigma]\!] \to [\![\tau]\!]$.

Dowód przez indukcję po $\mathcal{C}$.

Zadanie 3. Lemat

Jeśli $\mathcal{D} :: \Gamma \vdash M : \tau$, to $[\![\Gamma]\!] \vdash [\![\mathcal{D}]\!] : [\![\tau]\!]$, gdzie $[\![\Gamma]\!]$ jest rozszerzeniem tłumaczenia typów na środowiska: $[\![\emptyset]\!] = \emptyset$ i $[\![\Gamma, x : \tau]\!] = [\![\Gamma]\!], x : [\![\tau]\!]$.

Dowód przez indukcję po $\mathcal{D}$, wykorzystując lemat z zadania 2.

Przykład. "Ścinanie" obiektów w C++

Obiekt należy rzutować w górę na wskaźnik lub referencję. Wówczas nie ulega on zmianie (adresy zajmują tyle samo miejsca w pamięci, a wskaźniki i referencje są adresami) i można wykorzystywać polimorfizm inkluzyjny. Rzutowanie w górę na obiekt spowoduje utratę części obiektu, która należy do podklasy. Ta część obiektu zostanie dosłownie "ścięta" (ang. sliced off). Podklasa dodaje zwykle do obiektu nowe pola, które nie mieszczą się w pamięci, przeznaczonej dla obiektu klasy bazowej. Rzutowania w górę na obiekt (być może w argumencie funkcji, przekazywanym przez wartość) uniemożliwia więc wykorzystanie polimorfizmu inkluzyjnego. W poniższym przykładzie obiekt klasy `Point2D` jest rzutowany w górę na obiekt klasy `Point1D`.

```
#include <iostream>
using namespace std;

class Point1D
{ public:
    Point1D(int x=0) { this->x = x; }
    virtual void print() const { cout << "Point1D: [x=" << x << ']'; }
  protected:
    int x;
};

class Point2D : public Point1D
{ public:
    Point2D(int x=0, int y=0) :Point1D(x)
    { this->y = y; }

    virtual void print() const
    { cout << "Point2D: [x=" << x << ", y=" << y << ']'; }
  protected:
    int y;
};

void main()
{ Point2D p2D (2, 5);
  p2D.print();

  Point1D &p1Dref = p2D;
  cout << "\nPoint2D as reference to Point1D (OK)\n";
  p1Dref.print();

  Point1D p1D = p2D;
  cout << "\nPoint2D as Point1D (Point2D has been sliced off to Point1D!)\n";
  p1D.print(); cout << endl;
}
```