Technische Universität München
Institut für Informatik
Prof. Tobias Nipkow, Ph.D.
Dr. Clemens Ballarin

Vorlesung Semantik
Sommersemester 2004
Übungsblatt 9
Besprechung am 23. Juni 2004

Aufgabe 1 (H) (*Zugiff auf die Oberklasse:* `super`)

Bei Feldzugriff und Methodenaufruf kann in Java `super` anstelle von `this` verwendet werden. Taucht `super` in einem Methodenrumpf in der Deklaration der Klasse C auf, so wird nicht auf das von C aus sichtbare Feld zugegriffen, sondern auf das von der direkten Oberklasse C' von C aus sichtbare Feld. Ebenso für Methoden.

a) Führen Sie in Jinja die neuen Ausdrücke `super`.$F\{D\}$ und `super`.$F\{D\} := e$ für Feldzugriff und Feldzuweisung ein. Wie bei den normalen Feldoperationen bezieht sich D auf eine Oberklasse von C. Im Unterschied zu `this` muss es sich aber um eine echte Oberklasse handeln.

b) Führen Sie in Jinja den Ausdruck `super`.$M\{D\}(es)$ zum Methodenaufruf ein. Wie bei den Feldoperationen wird jetzt die Klasse, aus der die Methode stammt, explizit angegeben.

Geben Sie jeweils Typisierungsregeln und die Big-Step-Semantik für die neuen Sprachkonstrukte an.

Aufgabe 2 (Ü) (*Mehrfachvererbung und Interfaces*)

In Java wird Mehrfachvererbung mittels Interfaces realisiert. Im Gegensatz zu Klassen kann ein Interface mehrere Interfaces gleichzeitig erweitern. Inferfaces enthalten keine Felder, und die Methoden enthalten keine Implementierung, also keine Methodenrümpfe. Klassendeklarationen können sowohl eine Klasse importieren, als auch ein Interface implementieren. In einem wohlgeformten Programm muss eine Klasse C, die das Interface I implementiert, alle in I und dessen Ober-Interfaces vorkommende Methoden implementieren.

a) Erweitern sie Jinja-Programme um Interfaces. Geben Sie hierzu genau an, aus welchen Bestandteilen ein Programm jetzt besteht. Zur Vereinfachung gehen Sie davon aus, dass eine Klasse immer genau eine Klasse erweitert und ein Interface implementiert. Analog zur Klasse `Object` steht das leere Interface `Any` zur Verfügung.

Im Folgenden soll die Typisierungsregel für den Methodenaufruf auf die neue Situation angepasst werden. Gehen Sie dabei analog zu *James Gosling, Bill Joy, Guy Steele: The Java Language Specification*, Abschnitte 15.11.1 und 15.11.2 vor.

b) Definieren Sie die Begriffe *Applicable Method*, *More Specific Method* und *Most Specific Method*.

c) Geben Sie die Typisierungsregel für den Methodenaufruf an. Damit zur Laufzeit die richtige Methode aufgerufen werden kann, wird der Methodenaufruf mit den Argumenttypen parametrisiert. Das heisst,

$$e.M\{Ts\}(es)$$

ruft im Objekt e die Methode M mit den Argumenttypen Ts und den Argumenten es auf. Dabei ist Ts der Argumenttyp der spezifischsten, anwendbaren Methode, die durch die Typisierungsregeln bestimmt wird.

Aufgabe 3 (P) (*Ausnahmebehandlung in Jinja*)

Erweitern Sie die Big-Step-Semantik von Jinja um die Regeln zur Ausnahmebehandlung.

`/usr/proj/semantik/prog/prolog/jinja-exn/`.

Ergänzen Sie die Regelrümpfe in der Datei `semantik.pl` und geben Sie Ihre Datei ab. Einige Testbeispiele finden Sie wieder in `dialog.txt`.