Programmieren ohne syntaktisches Korsett

Christian Heinlein
Studiengang Informatik
Hochschule Aalen − Technik und Wirtschaft

Gängige Programmiersprachen zwingen ihre Benutzer in ein syntaktisches Korsett von Funktions- oder Methodenaufrufen: Wann immer man ein Stück Programmcode kapseln möchte, um davon zu abstrahieren oder um es an mehreren Stellen wiederverwenden zu können, muss man es in eine Funktion oder Methode schreiben und diese dann mit der jeweiligen Syntax aufrufen. Dies führt beispielsweise dazu, dass Matrixoperationen in Java umständlich als `A.negate().multiply(B)` anstelle von `-A * B` oder einfach nur `-A B` geschrieben werden müssen. In Sprachen wie z. B. C++ oder Haskell, die das Überladen vorhandener Operatorsymbole oder auch die Definition neuer Symbole erlauben, könnte dieses Beispiel zwar „natürlicher“ formuliert werden, aber wie das folgende Beispiel zeigt, ist C++ diesbezüglich auch nicht viel besser als Java: Um aus einer Menge `s` von ganzen Zahlen alle negativen Werte zu entfernen, muss man `remove_if(s.begin(), s.end(), bind2nd(less<int>(), 0))` schreiben, um das ganze kompakt in einem Funktionsaufruf auszudrücken. Könnte man einfach `remove x from s where x < 0` schreiben, so wäre die Bedeutung auch für Nicht-C++-Experten und wahrscheinlich sogar für Nichtinformatiker verständlich. Zwar erlaubt Haskell, Dinge dieser Art elegant durch „list comprehensions“ auszudrücken, aber früher oder später stößt man auch hier, wie in jeder anderen Sprache, an unüberwindbare syntaktische Grenzen − es sei denn, die Sprache ist „MOSTflexiPL“ und bietet ihren Benutzern nahezu beliebige syntaktische Gestaltungsmöglichkeiten.

Im Vortrag sollen diese außergewöhnlichen Möglichkeiten der Sprache MOSTflexiPL (Modular, Statically Typed, Flexibly Extensible Programming Language) anhand einiger Beispiele skizziert werden. Bei Interesse können viele weitere Beispiele im Rahmen des Workshops „live“ demonstriert werden, um die Thematik zu vertiefen.

http://flexipl.info