Erweiterung der Programmiersprache Java um Constraint-Programmierung

Jan C. Dageförde und Herbert Kuchen, WWU Münster

In der Praxis hat sich objektorientierte Programmierung mit Sprachen wie Java an vielen Stellen durchgesetzt, u. a. durch Features wie Kapselung von Struktur und Verhalten sowie Vererbung. Java ist allerdings nur bedingt geeignet zur Lösung von Suchproblemen, wie man sie beispielsweise bei der Personaleinsatzplanung oder der Produktionsplanung vorfindet. Zwar lassen sich Constraint Solver als Bibliotheken einbinden; jedoch sind diese wenig intuitiv in Programme einzubetten und nicht standardisiert[1]. Auch sind dann Constraint Solving-Abschnitte von der restlichen objektorientierten Programmierung sprachlich getrennt. Gegebenenfalls wird die Lösungsfindung sogar an Prolog-Programme mit `clpfd` ausgelagert, wodurch die Integration in bestehende Java-Programme zusätzlich erschwert wird.

Wir schlagen daher die Sprache *Muli Lang* (Muenster Logic Imperative Language) vor, die die genannten Features von Java und Constraint Programming vereint. Sie verfügt über logische Variablen, eingekapselte Suche und Backtracking. Das zugehörige Laufzeitsystem basiert auf einer symbolischen Java Virtual Machine (SJVM), die die übliche Java Virtual Machine (JVM) um Komponenten wie Choice Points und Trail erweitert, wie sie von abstrakten Maschinen für logische Programmiersprachen, wie der Warren Abstract Machine (WAM), bekannt sind.

Muli Lang ist mit dieser Kombination von Eigenschaften beispielsweise für die Realisierung von Testfallgeneratoren oder auch für die Implementierung von Planungsproblemen geeignet, bei denen sich im zeitlichen Verlauf inkrementell neue Restriktionen ergeben und andere hinfällig werden, wie das in der Praxis häufig der Fall ist.

[1] Bestrebungen zur Schnittstellenstandardisierung wie das JSR 331 scheinen nicht mehr aktiv verfolgt zu werden, vgl. `https://jcp.org/en/jsr/detail?id=331`.