

The Constraint Imperative Programming Language Turtle

Martin Grabmüller

Fachgebiet Übersetzerbau und Programmiersprachen
Fakultät IV – Elektrotechnik und Informatik
Technische Universität Berlin
`magr@cs.tu-berlin.de`

Constraints are conditions in a mathematical sense and are well suited for expressing restrictions on values and relations between objects. Constraint solvers are program components which can recognize violations of constraints and may be able to re-establish them by modifying the objects or adapting the relations participating in these constraints.

The programming language TURTLE [1] combines traditional imperative language constructs and declarative constraints. This blending of paradigms is called *constraint imperative programming* in literature.

The design of the language was started by first defining a base language which includes the basic elements of imperative programming languages: variables, assignments, conditionals, loops and subroutines. TURTLE was then extended with four elements to form a constraint imperative language: *constrainable variables*, *constraint statements*, *user-defined constraints* and *constraint solvers*.

Constrainable variables are not set by assignment statements like normal variables, but instead their values are determined by constraints. Variables must be declared as being constrainable by assigning them special types. Because of the separation between these two kinds of variables, the use and semantics of variables can be deduced from their types.

Constraint statements are used to place constraints on constants and constrainable variables. A constraint statement consists of a constraint conjunction and a sequence of statements. The constraints are satisfied by assigning suitable values to their variables as long as the statements in the body are executed.

User-defined constraints collect simple constraints, just like subroutines combine statements. Constraint conjunctions, which are needed in several constraint statements throughout the program, can be collected and be referenced by a name in order to access the conjunction without the need to specify all individual constraints wherever they are needed.

Constraint solvers check constraints specified by constraint statements and calculate assignments for constrainable variables. The solvers maintain all constraints in *constraint stores*. The number and kind of the used constraint solvers is not specified by the language, it depends on the problems which are to be solved. The TURTLE system implements a generic interface between compiled code and the constraint solvers.

Besides the integration of imperative language properties and constraints, some language constructs mainly known from functional programming have been integrated. Functions in TURTLE are higher-order, which means that they can be arguments and results of functions and can be stored in arbitrary data structures. These higher-order functions are very useful for abstraction and code re-use. The type system allows to define algebraic data types and modules can be parameterized by data types.

A programming system, consisting of a compiler, a run-time system and library modules has been implemented. The resulting programming language is practical and several imperative as well as constraint imperative example programs of varying size have been developed to demonstrate the different language features.

References

- [1] Martin Grabmüller. Constraint-Imperative Programmierung. Diplomarbeit, Technische Universität Berlin, Februar 2003.