Drei Dinge braucht ein generischer Operator: explizite, implizite und deduzierte Parameter

Christian Heinlein
Studiengang Informatik
Hochschule Aalen − Technik und Wirtschaft

Explizite Parameter sind ein fundamentaler Bestandteil von Funktionen (oder Methoden) in allen höheren Programmiersprachen. *Implizite Parameter*, deren Wert beim Aufruf einer Funktion vom Programmierer nicht explizit angegeben wird, sondern vielmehr vom Compiler implizit aus dem Aufrufkontext ermittelt wird, sind weit weniger verbreitet. Defaultargumente in C++ sind zwar auf den ersten Blick sehr ähnlich zu impliziten Parametern, aber es besteht ein wesentlicher Unterschied: Ein Defaultargument wird bei der Deklaration bzw. Definition einer Funktion angegeben und kann nur Dinge verwenden, die dort statisch sichtbar sind; der Wert eines impliziten Parameters wird jedoch im jeweiligen Kontext eines Funktionsaufrufs ermittelt, der bei jedem Aufruf unterschiedlich sein kann. *Deduzierte Parameter*, deren Wert im Normalfall ebenfalls nicht explizit angegeben wird, sondern aus dem Typ eines anderen Parameters ermittelt (deduziert) wird, findet man z. B. in C++ als „Template-Parameter“ und in Java als „Typ-Parameter“. In beiden Sprachen spielen sie jedoch gegenüber „normalen“ Parametern eine Sonderrolle, was u. a. daran liegt, dass Typen dort keine „first class citizens“ sind.

Die momentan von mir entwickelte *statisch typisierte erweiterbare Ausdruckssprache* STEEL (Statically Typed Extensible Expression Language) bietet alle drei o. g. Parameterarten − explizit, implizit und deduziert − in einheitlicher Weise an, und ein *generischer Operator*[1] − wie z. B. ein Maximumoperator für beliebige Datentypen − benötigt typischerweise eine Kombination aller drei Arten: Die beiden Werte `x` und `y`, deren Maximum bestimmt werden soll, werden als explizite Parameter übergeben (was der Compiler daran erkennt, dass sie in der Signatur des Operators auftreten); ihr gemeinsamer Typ `T` wird durch einen deduzierten Parameter repräsentiert (was der Compiler daran erkennt, dass er im Typ von anderen Parametern auftritt); und der zur Ermittlung des Maximums benötigte Vergleichsoperator `>` (der selbst wiederum zwei explizite Parameter `a` und `b` besitzt) wird als impliziter Parameter übergeben (was der Compiler daran erkennt, dass er weder explizit noch deduziert ist):

```
[ "T" : type;                              Deduzierter Parameter
  "x" : T; "y" : T;                        Explizite Parameter
  ["a" : T; "b" : T] "a > b" : bool        Impliziter Parameter
]
"max x y" : T                              Signatur des Operators
{ if x > y then x else y end }             Implementierung des Operators
```

Der implizite Parameter kann auch als *Einschränkung* oder *Nebenbedingung* für den deduzierten Parameter aufgefasst werden: Die statische Typprüfung wird eine Anwendung des `max`-Operators auf zwei Werte `x` und `y` eines Typs `T` nur dann akzeptieren, wenn es im Kontext des Aufrufs einen Operator `>` gibt, der auf zwei Werte `a` und `b` des Typs `T` angewandt werden kann und als Resultattyp `bool` besitzt.

Da der Operator, der als „Belegung“ eines impliziten Parameters verwendet wird, selbst wieder implizite Parameter besitzen kann, die bei einem Aufruf belegt werden müssen, ergeben sich zum Teil interessante Abhängigkeitsbeziehungen. Beispielsweise kann man einen generischen Vergleichsoperator für Sequenzen (Listen) definieren, der genau dann anwendbar ist, wenn es einen Vergleichsoperator für den Elementtyp der Sequenzen gibt. Wendet man diesen Operator z. B. auf Sequenzen von Sequenzen von Zahlen an, so gelingt dies nur, wenn es einen Vergleichsoperator für Sequenzen von Zahlen gibt; diesen wiederum „gibt“ es genau dann, wenn es einen Vergleichsoperator für Zahlen gibt, was standardmäßig der Fall ist.

[1] In STEEL besteht kein Unterschied zwischen Operatoren und Funktionen, und es wird einheitlich die Bezeichnung „Operator“ verwendet. Bei der Deklaration von Operatoren und ihren Parametern müssen aus syntaktischen Gründen Anführungszeichen verwendet werden.