Friedrich-Schiller-Universität Jena

Fakultät für Mathematik und Informatik

Programmiersprachen und Compiler

Projektberichte

3.5.2.1 Automatische Verifikation rekursiver Prozeduren

Bei der automatischen Programmverifikation ergänzt der Programmkonstrukteur das Programm um eine formale Spezifikation, z.B. in Form von Vor- und Nachbedingungen. Ein automatischer Programmbeweiser versucht dann die Konsistenz zwischen Spezifikation und Programm ohne weitere Mitwirkung des Programmkonstrukteurs zu beweisen.
Die Aufgabe des Programmkonstrukteurs sollte nicht Dinge umfassen, die aus Spezifikation und Programm abgeleitet werden können. Dazu gehört z.B. eine Terminierungsfunktion zum Nachweis der Terminierung einer rekursiven Prozedur oder einer Schleife. Für Schleifen wurden früher von Kauer Heuristiken zum Nachweis der Terminierung von FOR-Schleifen aufgestellt. In diesem Projekt werden Heuristiken für die Berechnung von Terminierungsfunktionen für rekursive Prozeduren erforscht. Dabei kann die Rekursion direkt oder auch indirekt sein. Bei Einsatz solcher Heuristiken lassen sich manche Programme mit rekursiven Prozeduren automatisch verifizieren.

3.5.2.2 Allgemeine Parametermechanismen

In Programmiersprachen entstanden im Laufe der Zeit eine Reihe verschiedener Parameterarten mit unterschiedlichen Eigenschaften. Häufig wurde versucht, Parameter wie Variable oder Konstanten zu sehen. Parameter haben aber ein wesentliches zusätzliches Merkmal, nämlich die Übergabeart.
Im vorliegenden Projekt werden Parameterarten systematisch nach den Merkmalen „Variabilität“ und „Übergabemechanismus“ entwickelt. Eine praktische Erprobung erfolgt im Rahmen einer Erweiterung der Programmiersprache „Micro“, die auch im Rahmen der Vorlesung „Compilerbau“ eingesetzt wird. Dieses erweiterte Micro-2004 wurde in der Studienarbeit von A. Finn erfolgreich implementiert.

3.5.2.3 Allgemeine Integer-Typen

In Programmiersprachen gibt es meist nur feste Teilintervalle der ganzen Zahlen als Integer-Typen. Manchmal kann der Programmkonstrukteur auch eigene Intervalle als Wertebereiche von Integer-Typen festlegen. Andere Teilmengen wie z.B. alle geraden Zahlen innerhalb eines Bereichs können dagegen nicht als Wertebereich eines Integer-Typs auftreten.
Im Rahmen der Diplomarbeit „Erweiterung von Micro-2004 um Typen“ von S. Jahn wurde die Sprache Micro-2004 um Integer-Typen mit beliebigen Wertemengen $\subseteq$ Int32 erweiteret. Ausserdem wurden Array-Typen eingeführt, in welchen diese neuen Integer-Typen als Indextypen verwendet werden können. Da zur Festlegung der Wertemengen mehrere Mechanismen zur Definition von Ganzzahlmengen eingeführt wurden, entstand gewissermassen als Nebenprodukt die Grundlage von Ganzzahlmengen-Typen. Dies konnte aber innerhalb der Diplomarbeit nicht vertieft werden. Die so erweiterte Form „Micro-2005“ wurde dann innerhalb der Diplomarbeit erfolgreich implementiert.

3.5.2.4 Sharp Adaptation Rules

Viele automatische Programmbeweiser, darunter auch der FPP (Frege Program Prover), benutzen die prädikative bzw. relationale Definition der Semantik von Programmiersprachen. Die darauf basierenden Beweisregeln lassen sich unterschiedlich leicht herleiten. Schwierigkeiten traten besonders bei der Herleitung einer Beweisregel für den Prozeduraufruf auf. Die Beweisregel für den Prozeduraufruf ist ein Spezialfall für die Beweisregel für eine Spezifikation allgemein. Solche Regeln werden auch als Adaptation Rules bezeichnet. In der Vergangenheit wurden von verschiedenen Autoren eine Reihe von Beweisregeln für den Prozeduraufruf und von Adaptation Rules publiziert, die teils nicht korrekt waren (z.B. Meertens, Gries, Bijlsma et al.). Manchmal haben die Regeln auch einen eingeschränkten Anwendungsbereich und liefern ausserhalb dieses Bereiches falsche Ergebnisse. Diese Situation ist besonders ungünstig für das *mechanische* Beweisen von Programmen, wenn das Überprüfen auf Anwendbarkeit aufwendig ist.
Eine grundlegend neue Herleitung von Adaptation Rules auf der Basis von Mengen und Relationen ergab eine neue Regel, welche auch in den Fällen korrekte Ergebnisse ergibt, in welchen frühere Regeln versagen oder unsicher sind.

3.5.2.5 Relational Semantics

Die Probleme mit existierenden Beweisregeln für Prozeduraufrufe (s. 3.5.2.4) führte dazu, einen konsequent auf Relationen zwischen Zuständen basierenden Formalismus zur Definition von Semantik und Verifikationsbedingungen zu entwickeln. Dabei sollen insbesondere auch solche Aspekte realer Programme auf realen Rechnern berücksichtigt werden, die bisher innerhalb der formalen Semantik und Verifikation häufig ignoriert wurden.

3.5.2.6 Teilnahmen an internationalen Programmierwettberwerben

Seit einigen Jahren nehmen Jenaer Studenten regelmässig am ACM „Internatinal Collegiate Programming Contest“ (ICPC) teil (http://psc.informatik.uni-jena.de/ICPC.htm). Im Jahr 2004 und 2005 war unsere Fakultät bei den Regionalausscheiden in Lund (Schweden) und Paris mit jeweils einem Team vertreten und belegte achtbare Plätze im vorderen Drittel.