C++ ist typsicher? Garantiert!

Daniel Wasserrab
Universität Passau
wasserra@fmi.uni-passau.de

Tobias Nipkow
Technische Universität München
nipkow@in.tum.de

Gregor Snelting
Universität Passau
snelting@fmi.uni-passau.de

Frank Tip
IBM T. J.Watson Research Center
ftip@us.ibm.com

Abstract:
Wir präsentieren eine operationelle Semantik mit Typsicherheitsbeweis für Mehrfachvererbung in C++, formalisiert im und maschinengeprüft durch den Maschinenbeweiser Isabelle/HOL. Die Typsicherheit des Vererbungsmechanismus von C++ war lange offen. Der nun vorliegende Beweis erhöht das Vertrauen in die Sprache, erzeugt aber auch neue Einsicht in die Problematik des C++-Vererbungsmechanismus. Er öffnet die Tür für weitergehende Beweise, die bisher unerreichte Sicherheitsgarantien für C++-Programme liefern.

1 Einleitung

"Well-typed programs cannot go wrong"[Mil78]: dieser Slogan beschreibt die Eigenschaft der Typsicherheit. Stark typisierte Programmiersprachen gelten zu Recht als bedeutende Errungenschaften der Softwaretechnik, denn typkorrekte Programme können nicht unkontrolliert mit Laufzeitfehlern abstürzen, sind verständlicher, und sind effizienter zu compilieren. Nur eine Minderheit der Softwareentwickler bevorzugt schwach typisierte Sprachen wie z.B. Skriptsprachen; gerade die wichtigsten objektorientierten Sprachen wie C++, Java und C# sind stark typisiert.

Tatsächlich sind aber Typsysteme für moderne objektorientierte Sprachen sehr komplex. Damit stellt sich die Frage nach der Typsicherheit oder „Soundness" eines Typsystems: können typkorrekte Programme wirklich erfolgreich ausgeführt werden, oder gibt es vielleicht verborgene Defekte im Typsystem, die dazu führen, dass angeblich typkorrekte Programme doch abstürzen? Wer die Feinheiten der Vererbung von Java oder gar C++ kennt, wird sofort glauben, dass definitive Antworten auf diese Frage schwer zu erhalten sind. Die Erfahrung zeigt vielmehr, dass sogar bekannte C++ Compiler inkorrekten Code für eigentlich typkorrekte Programme erzeugen. Wenn man also beweisen kann, dass eine Sprache typsicher ist, wächst automatisch das Vertrauen in die Qualität und Sicherheit von damit implementierten Anwendungen.

Typsicherheitsbeweise für akademische Spielzeugsprachen wurden schon vor Jahrzehnten geführt. Für reale Sprachen sind sie aber so komplex, dass man ohne den Einsatz automatischer Beweiser und maschinengeprüfter Beweise nicht weiterkommt. Denn manuelle

Beweise können Fehler enthalten, und wer kontrolliert das? „Quis custodiet ipsos custodes“ fragte schon Juvenal. Ein großer Schritt wurde vor 10 Jahren erreicht, als Nipkow et al. [NvO98] die Typsicherheit von Java im Hinblick auf eine formale Semantik definierten und mittels des Theorembeweisers Isabelle [NPW02] bewiesen, dass Java tatsächlich typsicher ist. Für C++ war die entsprechende Frage aber lange offen und galt als nicht handhabbar. Denn die Grundlagen von formaler Semantik und Typsystemen einerseits, die Methodik und die Power von maschinellen Beweissystemen andererseits waren unzureichend zur Behandlung einer so komplexen Sprache.

Aufbauend auf den entsprechenden Vorarbeiten für Java bewiesen wir, dass der Vererbungsmechanismus von C++ tatsächlich typsicher ist. Dies ist nichttrivial, zumal C++ die berüchtigte Mixtur aus „virtueller“ und „nichtvirtueller“ Mehrfachvererbung bietet. Der vorliegende Beitrag gibt eine Übersicht über den Beweis. Er hat zum Ziel, auch dem „normalen“ Softwaretechniker jenes Vertrauen in C++ zu vermitteln, das nur durch einen strengen, maschinengeprüften Beweis erreichbar ist; und außerdem zu demonstrieren, zu welchen Leistungen formale Semantik und maschinelles Beweisen heute in der Lage sind.

2 Übersicht

Cardelli [Car04] definiert Typsicherheit dadurch, dass keine ungefangenen Laufzeitfehler auftreten dürfen (wogegen das kontrollierte Auslösen von Ausnahmen erlaubt ist). Die Version der Typsicherheit, die wir hier beweisen, verlangt, dass die Ausführung eines wohlgetypten terminierenden Programms entweder ein Ergebnis des erwarteten Typs liefert oder mit einer Ausnahme endet. Die Semantik und der Typsicherheitsbeweis sind formalisiert und maschinengeprüft mittels des Theorembeweisers Isabelle/HOL [NPW02] und online verfügbar[1].

Ein Hauptgrund für die Komplexität von C++ ist die besondere Art von Mehrfachvererbung mit shared (“virtuell”) und repeated (“nichtvirtuell”) Vererbung, welche beliebig kombiniert werden dürfen. Aufgrund dieser Komplexität wurde das Verhalten von Operationen traditionell informell und mittels Implementierungskonstrukten wie v-tables beschrieben [Str03]. Unsere formale Beschreibung des Verhaltens von C++ ermöglicht maschinengeprüfte Verifizierung bzw. Falsifizierung von Software-Sicherheitsanalysen auf hohem programmiersprachlichem Niveau.

Zusammenfassend leistet der vorliegende Artikel folgende Beiträge:

- Vorstellung einer formalen Semantik mit maschinengeprüften Typsicherheitsbeweis der Mehrfachvererbung in C++.
- Diskussion einiger Besonderheiten der Methodenaufrufe in C++, welche bei der Erstellung der Semantik auftraten.
- Erweiterung des Einsatzbereiches von formalen Semantiken und Theorembeweisern durch die Formalisierung des komplexen Verhalten der C++ Mehrfachvererbung.

[1] `http://afp.sourceforge.net`

- Bereitstellung eines programmiersprachlichen Grundstocks für maschinengeprüfte Beweise von sprachbasierten Software-Sicherheitsanalysen (z.B. Information Flow Control)

Für eine ausführliche Erläuterung der Semantik und ihrer Details siehe [WNST06].

Literatur

[Car04] Luca Cardelli. Type Systems. In *The Computer Science and Engineering Handbook*. 2. Auflage, 2004.

[Mil78] Robin Milner. A theory of type polymorphism in programming. *Journal of Computer and System Sciences*, 17(3):348–375, 1978.

[NPW02] Tobias Nipkow, Lawrence Paulson und Markus Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*, Jgg. 2283 of *Lect. Notes in Comp. Sci.* 2002.

[NvO98] Tobias Nipkow und David von Oheimb. Java$_{light}$ is Type-Safe — Definitely. In *Proc. of POPL'98*, Seiten 161–170. ACM Press, 1998.

[Str03] Bjarne Stroustrup. *The C++ Standard: Incorporating Technical Corrigendum No. 1*. John Wiley, 2. Auflage, 2003.

[WNST06] Daniel Wasserrab, Tobias Nipkow, Gregor Snelting und Frank Tip. An Operational Semantics and Type Safety Proof for Multiple Inheritance in C++. In *Proc. of OOPSLA'06*. ACM Press, 2006.