# Migration von Zeigeranalysen in Bauhaus

Michael Schupikov Timm Felden

Universität Stuttgart, Institut für Softwaretechnologie
Universitätsstr. 38, 70569 Stuttgart
tky80099@stud.uni-stuttgart.de, feldentm@informatik.uni-stuttgart.de

## Zusammenfassung

In diesem Papier wird die Durchführung der Migration der Zwischendarstellung von vier Zeigeranalyse-Werkzeugen in Bauhaus beschrieben. Diese wurde von Studenten im Rahmen eines Projekts durchgeführt.

## 1 Einleitung

Bauhaus ist eine Programmanalyse-Werkzeugkette [1], die unter anderem Werkzeuge zur Zeigeranalyse enthält. Seit 2016 wird die Zwischendarstellung von einer Individuallösung auf SKilL migriert, um die Erweiterbarkeit der Werkzeugkette zu verbessern [2]. Die Migration wird überwiegend von Studenten der Universität Stuttgart durchgeführt. In diesem Erfahrungsbericht beschreiben wir die Migration von vier Zeigeranalyse-Werkzeugen im Rahmen eines studentischen Projekts. Aufgabe des Projekts war es den Migrationsprozess für diese Werkzeuge abzuschließen, und dies durch Tests zu belegen, um die Werkzeuge wieder in den Wartungszustand zu überführen.

Vor Beginn der Migration waren die vier Werkzeuge `ecr`(`_tool`), `das`(`_tool`), `andersen`(`_tool`) und `pta`(`_tool`) ausführbar. Allerdings wurde keines automatisch getestet. Ebenso fand sich für keines ein ausreichender Testdatensatz mit Sollwerten. Für `das` und `pta` ist bekannt, dass teilweise kein Ergebnis produziert wird. Ferner beträgt die Laufzeit von `andersen` für große Eingaben teilweise länger als einen Tag.

Zu Projektbeginn waren die Werkzeuge unterschiedlich funktionsfähig. So waren `ecr` und `pta` übersetzbar und ausführbar. Mit `ecr` konnten plausible Debug-Ausgaben, aber keine Zwischendarstellung (IR) erzeugt werden. Die aus `pta` resultierende IR war teilweise korrupt. `das` und `andersen` waren nicht übersetzbar.

## 2 Durchgeführte Anpassungen

In `ecr` wurden beim Lesen manche Container nicht wie durch SKilL spezifiziert initialisiert. Die Debug-Ausgabe funktionierte wie vorgesehen, weil das Problem erst beim Schreiben der Resultate auftrat. Die Lösung besteht in der Initialisierung der Container beim Laden der Eingabedatei. Dadurch können Resultate wie vorgesehen geschrieben werden.

In `das` wurden analog zu `ecr` Container beim Lesen der Eingabe nicht vom API initialisiert. Folglich müssen diese im Werkzeug nach dem Lesen der Eingabe initialisiert werden. Zudem sind Anpassungen aufgrund der Arbeit von Dennis Przytarski [2] notwendig. Einige Funktionen des API werden nicht mehr unterstützt und werfen eine Ausnahme zur Laufzeit. Die Lösung besteht in der Benutzung der alternativen Funktionen, auf die in den Kommentaren verwiesen wird. Eine weitere Änderung des APIs ist, dass die Datei den Speicher besitzt und diesen beim Schreiben freigibt. Danach können auf dem Graphen keine Operationen mehr durchgeführt werden. Als Lösung wurden Debug-Ausgaben aus `das` zur Vermeidung unzulässiger Zugriffe auf den IML-Graphen entfernt.

Im `andersen`-Werkzeug wurden Zuordnungen über das werkzeugspezifische, generische Mapping-Paket realisiert. Dessen Verwendung wurde an das nun vollständig generierte API angepasst. Zudem war der Typ `Andersen_Result` fälschlich in der Spezifikation als `Storable` deklariert. Aufgrund eines Fehlers in der Bauhaus-Reflection werden spezifizierte Felder des Typs Storables ignoriert, weshalb der Fehler zuvor nicht erkannt wurde. Den Typ `Andersen_Result` korrekt zu spezifizieren, löst die dadurch verursachten Probleme. Schließlich ist eine Anpassung bzgl. des API analog zu `das` notwendig, um nicht mehr unterstützte Funktionen durch Alternativen zu ersetzen.

In `pta` wurde die korrupte Ausgabe der Resultate durch die Anpassung der entsprechenden Array-Initialisierung beseitigt. Es wird ein Fehler im Compiler vermutet, der zur korrupten Ausgabe geführt hat. Darüber hinaus wurden in einigen Funktionsaufrufen nicht korrekte Bereiche für Indizes angenommen. Durch die Verwendung korrekter Indizes ist das Werfen entsprechender Ausnahmen beseitigt. Schließlich wurden einige optionale Checks am IML-Graphen nach dessen Schreiben durchgeführt, also nachdem die SKilL-Datei invalidiert wurde. Analog zu `das` führt dies zum Zugriff auf freigegebenen Speicher. Die Lösung besteht im Durchführen der Checks vor dem Schreiben der Resultate.

## 3 Tests

Für die Durchführung der Tests haben wir zwei Werkzeuge entwickelt. `tooltest` automatisiert Testdurchläufe für einen ganzen Ausführungspfad der Werk-

| | Erfolg | Abbruch | Ign. | Fehl. |
|---|---|---|---|---|
| `ecr` | 2286 | 326 | 26 | 34 |
| `das` | 2289 | 327 | 30 | 26 |
| `andersen` | 2292 | 329 | 19 | 32 |
| `pta` | 2283 | 349 | 19 | 21 |

Tabelle 1: Testergebnisse vor Migration

| | Erfolg | Abbruch | Ign. | Fehl. |
|---|---|---|---|---|
| `ecr` | 2318 | 246 | 106 | 2 |
| `das` | 2313 | 248 | 109 | 2 |
| `andersen` | 2322 | 246 | 102 | 2 |
| `pta` | 242 | 564 | 22 | 1844 |
| `iml2sf + pta` | 1892 | 36 | 4 | 194 |

Tabelle 2: Testergebnisse nach Migration

zeugkette. Am Ende der Ausführung wird eine konfigurierbare Testbedingung evaluiert und mit einem Sollwert verglichen. Der Sollwert wird über den Stand vor der Migration erhoben. Dadurch wird das Testresultat vierwertig: *Erfolg* bedeutet, Soll und Ist erfüllen die Testbedingung; *Abbruch* bedeutet, beide erfüllen diese nicht bzw. brechen vorher ab. *Ignoriert* wird ein negatives Soll und positives Ist. Ein *Fehlschlag* entspricht einem positiven Soll und negativen Ist. `ecrCR` erzeugt eine kanonische Darstellung der Analyseergebnisse von `ecr`, die für die alte und neue Zwischendarstellung gleich ist.

Als Testdaten werden 2672 Programme in C verwendet, die überwiegend der gcc-Testsuite[1] entnommen sind. Dabei kommen nur C-Dateien zum Einsatz, die gcc ohne Parameter übersetzen kann.

### Existenz von Resultaten

Zunächst wird die Existenz von Analyseergebnissen geprüft. Um die Stabilität der Testergebnisse zu untersuchen, werden die Resultate mit dem vorhergegangenen Durchlauf verglichen. Die Ergebnisse vor der Migration sind in Tabelle 1 aufgeführt.

Wir stellen fest, dass die Tests nicht stabil sind, da es ignorierte und fehlschlagende Tests gibt. Da hier Testdaten und Werkzeuge für Ist- und Soll-Werte identisch sind, kann eine Migration bezogen auf diesen Test nicht zu einem perfekten Ergebnis kommen.

Die Resultate nach der Migration sind in Tabelle 2 dargestellt. Wir stellen fest, dass für `ecr`, `das` und `andersen` mehr Tests erfolgreich sind. Das ist hauptsächlich auf Verbesserungen in Compiler und Linker zurückzuführen. Da der Code nur minimal angepasst wurde, sind die Werkzeuge also prinzipiell einsetzbar.

Bei `pta` schlagen die Tests überwiegend fehl. Da die Tests auf migrierten Testdaten (`iml2sf + pta`) überwiegend erfolgreich sind, wurden offenbar uns unbekannte Annahmen über die IR seitens `pta` durch das neue Front-End verletzt.

### Korrektheit der Resultate

Um die Korrektheit der Resultate zu prüfen, kombinieren wir `tooltest` mit `ecrCR`. Dadurch können wir prüfen, ob alle Zeigerbeziehungen mit einer gültigen Quellposition von der Migration unberührt bleiben. Um die Effekte des verbesserten Front-Ends auszublenden, wurden die Testdaten gelinkt und in das neue Format konvertiert, was die Zahl der Testdatensätze reduziert. Die Ausführung ergibt 2286 Erfolge, 0 Abbrüche, 25 Ignorierte und 10 Fehlschläge.

Alle Fehlschläge wurden manuell überprüft. Hier handelt es sich immer um Selbstreferenzen, z.B. durch "`void *p = &p;`". Da die Ergebnisse in Soll- und Ist-Werten falsch waren, ist davon auszugehen, dass die Werkzeugimplementierung an sich fehlerhaft ist.

Der einzige nennenswerte Unterschied zwischen beiden `ecr`-Versionen ist die Traversierungsreihenfolge der Knoten. Bezogen auf die Zeigeranalyse nach Steensgard bedeutet das einen Fehler in der Implementierung, da der Algorithmus fussinsensitiv ist [3].

Für `das` und `andersen` haben wir mit ähnlichen Werkzeugen vergleichbare Resultate erzielt. Daher wird die Migration hier beendet und die Entwicklung der Werkzeuge wird ohne Berücksichtigung des Migrationsaspekts fortgeführt. Zukünftige Tests werden ohne Vergleich mit den Ergebnissen vor der Migration durchgeführt.

## 4 Fazit

Es wurden die Werkzeuge `ecrCR`, `dasCR`, `andersenCR` zur Beurteilung des Migrationsfortschritts entwickelt. Diese nutzen die SKilL-inhärente Änderungstoleranz und Sprachunabhängigkeit, um den Arbeitsaufwand zu reduzieren. Die Werkzeuge wurden mit reduzierten Spezifikationen in Scala in etwa zwei Stunden entwickelt. Dadurch konnten wir erkennen, dass die Werkzeuge in unserem Sinne migriert wurden.

Um für Zeigeranalysewerkzeuge leichter beurteilen zu können, in welchen Fällen ein entsprechendes Werkzeug schon vor der Migration kein korrektes Resultat erzeugt hat, wäre ein umfangreicher Testdatensatz wünschenswert. Dieser könnte beispielsweise basierend auf dem Algorithmus von Steensgard [3] als Überapproximation für den von uns gewählten Testdatensatz realisiert werden.

## Literatur


[1] Aoun Raza, Gunther Vogel und Erhard Plödereder. Bauhaus – A Tool Suite for Program Analysis and Reverse Engineering. In: *Reliable Software Technologies – Ada-Europe 2006*, *Lecture Notes in Computer Science*, volume 4006, (pages 71–82). Springer Berlin Heidelberg, 2006.

[2] Dennis Przytarski. SKilLed Bauhaus. Masterarbeit: Universität Stuttgart, Institut für Softwaretechnologie, 2016.

[3] Bjarne Steensgaard. Points-to Analysis in Almost Linear Time. In: *Proceedings of the 23rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '96, (pages 32–41). ACM, 1996.


[1] github.com/gcc-mirror/gcc/tree/master/gcc/testsuite