Noninvasive regelbasierte Graphtransformation für Java*

James J. Hunt
aicas GmbH, Karlsruhe, Germany
jjh@aicas.com

Arend Rensink and Maarten de Mol
Department of Computer Science, University of Twente
P.O.Box 217, 7500 AE, The Netherlands
rensink@cs.utwente.nl

Der Standardansatz zur Verwendung von Graphtransformationen (GT) in einem Programm ist es, die Daten in eine bekannte Graphenstruktur umzuwandeln, diese zu bearbeiten und sie dann zurück zu wandeln in das Format des Programms. Diese Transformationen stellen sicher, dass die graphischen Operationen korrekt sind. Die Autoren haben eine alternative Methode zur Verwendung der Java-Programme entwickelt, die den Graphtransformator darüber informiert, wie Datenstrukturen eines Programms, die einem idealen Graph entsprechen, so dass in sito Graphtransformationen direkt auf der vorliegenden Datenstruktur durchgeführt werden können, ohne unter einem Korrektheitsverlust zu leiden. Der Vorteil des Ansatzes ist, dass er es ermöglicht, beliebige Java-Programme nicht-invasiv mit deklarativen Graphenregeln zu erweitern. Dieser verbessert die Klarheit, Prägnanz, Nachprüfbarkeit und Leistung.

Schwachstellen der GT sind ihre mangelnde Effizienz und die Notwendigkeit, Daten zwischen dem Anwendungsgebiet und der Graphdomäne zu transformieren. Obwohl mangelnde Effizienz zum Teil der Preis für allgemeine Anwendbarkeit sein kann, so scheint dies nicht der dominierende Faktor zu sein. Viel mehr ist es der Overhead der Transformation. Die beiden "Transformationen" sind eigentlich auch Modelltransformationen, und erhöhen die Komplexität der Technik so weit, dass es unpraktisch für große Graphen wird. Auch eine Anwendung zu erzwingen, die Graphenstruktur des Werkzeugs zu verwenden, hilft nicht viel, da die Graphenstruktur des Werkzeugs nicht ausdrucksfähig genug für die Anwendung sein kann, so dass bestehender Code neu geschrieben werden muss. Daher sind etablierte GT-Techniken invasiv.

Der hier vorgestellte Ansatz stellt diesen Prozess auf den Kopf. Da er Graphensemantik zu dem Problem und nicht das Problem zur Graphensemantik bringt, vermeidet es die invasive Natur der GT, erhält jedoch die Vorteile der GT, einschließlich der allgemeinen Anwendbarkeit und der deklarativen Natur. Es gibt drei Hauptteile dieses Ansatzes.

- Die Graphenstruktur wird spezifiziert durch Einfügen von JAVA-Annotationen in vorhandenen Klassencode. Als Konsequenz bilden die JAVA-Annotationen eine Spezifikationssprache für Graphen.

*Diese Arbeit wurde von der Artemis Joint Undertaking durch das CHARTER-Projekt gefördert, Grant-Nr. 100039. Siehe http://charterproject.ning.com/.

- Graphenmanipulation, wie etwas Hinzufügen oder Löschen von Knoten und Kanten, werden durch applikationsdefinierte Operationen bereitgestellt, die auch Annotationen benötigen, um ihre Wirkung auf die Graphenstruktur zu beschreiben.
- Regeln werden in einer deklarativen Sprache namens CHART geschrieben und anschließend in JAVA-Code übersetzt. Dieser bearbeitet die oben genannten Benutzerklassen mit annotierten Methoden ohne die Übertragung von Datenstrukturen zu und von der Graphendomäne.

Dieser Ansatz ermöglicht es, Komponenten eines bestehenden JAVA-Programms durch deklarative Graphtransformationen zu ersetzten, nur mit der Anforderung, die Datenstrukturen des Programms mit Annotationen zur ergänzen. Der Ansatz wurde in dem CHARTER-Projekt entwickelt, bei dem er in drei verschiedenen Werkzeuge angewendet wurde.

Um Graphentransformationsregeln auf einem vorhandenen JAVA-Programm mit diesem Ansatz anzuwenden, müssen folgende Änderungen vorgenommen werden:

- eine @Node Annotation muss an jeder Klasse und Schnittstelle angefügt werden, die einen Knoten darstellt;
- für jeden Kantentyp muss eine dedizierte Schnittstelle mit einer @Edge Annotation definiert werden und
- für jede gewünschte Operation muss eine Methode zur Verfügung stehen, entweder durch das Annotieren einer bestehende Methode oder das Hinzufügen einer neuen Methode mit der entsprechenden Annotation.

Diese Änderungen bereichern den vorhandenen Code lediglich durch Meta-Daten. Sie können daher an jedes JAVA-Programm angewendet werden. Daher wird dieser Ansatz *noninvasiv* genannt. Dies sollte nicht mit "nonmodifying" verwechselt werden, weil Annotationen und Schnittstellen zugefügt und zusätzliche Manipulationsmethoden implementiert werden müssen.

Eine CHART-Transformation besteht aus Transformationsregeln und kann durch Aufrufen von einer dieser Regeln gestartet werden. Jede Regel hat eine Signatur, die ihren Namen, ihre Parameter und Rückgabewerte definiert. Mehrere Ein- und Ausgabewerte sind erlaubt. Der Körper einer Regel besteht aus jeweils höchstens einem der folgenden Blöcke in der dieser Reihenfolge: ein Match-, ein Update-, ein Sequence- und ein Return-Block.

Ein großer Teil des Aufwands ist in den CHART Compiler geflossen, der den entsprechenden JAVA-Code generiert. Der Compiler wird RDT genannt, was für Regel Driven Transformer seht. Er unterstützt alle beschrieben Funktionen. Er wurde bereits in vier verschiedenen Systemen erfolgreich verwendet. Die Komplexeste davon ist für die Optimierung in einem neuen Byte-Code-Compile von aicas.

Obwohl CHART sich in der Praxis bewährt hat, gibt es noch viel zu tun, um die formalen Grundlagen stärken zu können und die Benutzbarkeit zu verbessern. Eine Theorie, die es einem CHART Programmierer ermöglicht, den Zusammenfluss und die Terminierung seiner Regelsystem abzuleiten, die semantische Erhaltung der Modelltransformationen zu beweisen und eine formale Überprüfung des RDT wären hilfreich. Auf der pragmatischen Seite benötigt der RDT weitere Arbeit an Effizienz, vielleicht mit einem besseren Matching-Algorithmus. Dies könnte die Leistung des erzeugten Codes verbessern. Schließlich könnte die CHART Sprache mit zusätzlicher Funktionalität erweitert werden.