Ein Ansatz zum Erkennen von Schleifen in Graphersetzungssystemen mit Hilfe einer aussagenlogischen Kodierung und seine Anwendung auf Haskell*

Marcus Ermler

Arbeitsgruppe Theoretische Informatik
Universität Bremen
maermler@informatik.uni-bremen.de

1 Motivation

Graphersetzung ist ein Turing-vollständiges Berechnungsmodell, welches insbesondere in solchen Gebieten Verwendung findet, in denen Probleme durch Graphen modelliert werden können. Eine bekannte Anwendung ist die Ersetzung von Termgraphen in der funktionalen Programmierung. Die Frage nach der Terminierung ist ein wichtiger Punkt, da zum Beispiel eine Endlosschleife selten ein intendiertes Verhalten darstellt. Für Graphersetzungssysteme ist diese Frage im Allgemeinen jedoch unentscheidbar (vgl. [Plu98]). Die Idee der Übersetzung von Graphersetzung in aussagenlogische Formeln wurde in [KKW10] eingeführt, in [Erm13] mit einem Ansatz zur induktiven Verifikation von Haskell-Programmen verbunden und wird hier im Sinne des Erkennens von Schleifen in Ableitungen mit einer Anwendung auf Haskell weiter vorangetrieben.

2 Erkennen von Schleifen: Von Graphersetzung über SAT zu Haskell

Seien G_0 und H_0 zwei Graphen und $g\colon G_0 \to H_0$ ein injektiver Graphmorphismus. Dann setzt das JOIN-Theorem aus [Kre78] die Bedingungen fest, unter denen eine Ableitung $G_0 \overset{n}{\Longrightarrow} G_n$ zu einer Ableitung $H_0 \overset{n}{\Longrightarrow} H_n$ erweitert werden kann. Einfach gesagt, wird $H_0 - G_0$ mit jedem G_i verbunden. Dieses Ergebnis kann dazu eingesetzt werden, um Ableitungen der Form $G_0 \overset{n}{\Longrightarrow} G_n \Longrightarrow H_0 \supsetneq G_0$ zu finden, d.h. um Schleifen in Ableitungen zu erkennen. Dem JOIN-Theorem folgend, würde so eine Ableitung $G_0 \overset{n}{\Longrightarrow} G_n$ zu einer Ableitung $H_0 \overset{n}{\Longrightarrow} H_n$ erweitert werden. Wichtig hierbei ist, dass Knoten, die $g(G_0)$ und $H_0 - g(G_0)$ verbinden, nicht während des Ableitungsprozesses gelöscht werden.

Definition 1 *Ein Graphersetzungssystem GRS mit initialem Graphen G_0 und Regelmenge P ohne knotenlöschende Regeln enthält eine* Schleife *genau dann, wenn Graphen G, H und ein injektiver Graphmorphismus $g\colon G \to H$ existieren, so dass $G_0 \overset{*}{\underset{P}{\Longrightarrow}} G \overset{+}{\underset{P}{\Longrightarrow}} H$ gilt.*

In der SAT-Kodierung bleibt die Knotenmenge invariant, wobei Knotenaddition durch spezielle Markierungen umgesetzt wird. Dass ein Graph in einer Ableitung der Länge $m > 0$ isomorph zu einem Teilgraphen des letzten Graphen der Ableitung ist, wird kodiert durch:

$$\mathsf{loop(m)} = \bigvee_{k=0}^{m-1} \bigvee_{g \in \mathcal{M}(n,n)} \bigwedge_{(v,a,v') \in E} \Big(edge(v,a,v',k) \to edge(g(v),a,g(v'),m) \Big),$$

wobei $\mathcal{M}(n,n)$ die Menge aller Abbildungen zwischen den Knoten der invarianten Knotenmenge ist und Kanten in E aus zwei Knoten und einer Markierung bestehen. Das Erkennen einer Schleife in allen Ableitungen bis zu einer Länge $k \in \mathbb{N}$ wird kodiert durch

$$\mathsf{loop\_det(G_0, k)} = \bigvee_{m=1}^{k} \big(\mathsf{der(G_0, m)} \wedge \mathsf{loop(m)} \big),$$

wobei $\mathsf{der(G_0, m)}$ die aussagenlogische Kodierung einer Ableitung der Länge m beginnend in G_0 gemäß [KKW10] ist. Der Zusammenhang zu Schleifen ist der folgende.

Satz 2 *GRS enthält eine Schleife, falls es ein k gibt, so dass* $\mathsf{loop\_det(G_0, k)}$ *erfüllbar ist.*

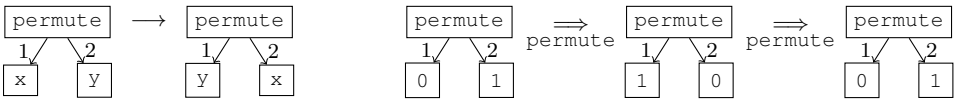

Abbildung 1: Anwendung der Regel `permute` (links) führt zu einer Schleife (rechts)

Betrachten wir eine Funktion `permute x y = permute y x` zur Permutation von Argumenten, die links in Abbildung 1 als Termgraphersetzungsregel gegeben ist. Rechts in Abbildung 1 findet man eine Ableitung, beginnend beim Termgraphen von `permute 0 1`, in der die zweifache Anwendung von `permute` zu einer Schleife führt. Eine erfüllende Belegung für die Teilformel $\mathsf{der}(\texttt{permute 0 1}, 2) \wedge \mathsf{loop}(2)$ erkennt die Schleife.

Literatur

[Erm13] Marcus Ermler. Towards a Verification Framework for Haskell by Combining Graph Transformation Units and SAT Solving. Bericht 1306, Christian-Albrechts-Universität zu Kiel, September 2013. Michael Hanus und Ricardo Rocha, Hrsg., Seiten 138-152.

[KKW10] Hans-Jörg Kreowski, Sabine Kuske und Robert Wille. Graph Transformation Units Guided by a SAT Solver. In Hartmut Ehrig, Arend Rensink, Grzegorz Rozenberg und Andy Schürr, Hrsg., *ICGT 2010*, Jgg. 6372 of *LNCS*, Seiten 27–42. Springer, 2010.

[Kre78] Hans-Jörg Kreowski. *Manipulationen von Graphmanipulationen*. Dissertation, TU Berlin, 1978.

[Plu98] Detlef Plump. Termination of Graph Rewriting is undecidable. *Fundamenta Informaticae*, 33(2):201–209, 1998.