Zusammenfassung

Das Thema dieser Arbeit ist ein neuer Ansatz zur Bestimmung von may-alias-Beziehungen im Rahmen von ANSI-C-Programmen. Von einem may-alias spricht man, wenn zwei verschiedene Variablen (oder auch komplexere Ausdrücke) ein und dieselbe Speicherstelle verwenden. Aliase entstehen durch die Verwendung von call-by-reference-Parametern oder Zeigern. Im Rahmen von C-Programmen entstehen hier besonders schwerwiegende Probleme, da C-Programme in der Regel intensiven Gebrauch von Zeigern machen und darüber hinaus nur sehr wenige Restriktionen für den Einsatz von Zeigern existieren.

Im Rahmen dieser Arbeit wird ein Verfahren beschrieben, daß die Zusammenfassung der für die Alias-Analyse relevanten Effekte mit Hilfe von Graphen (function interface graphs) realisiert. Diese Graphen stellen eine statische Repräsentation der von einem Programm benutzten Speicherstellen, sowie der darin enthaltenen Werte dar. Basierend auf einer Reihe von Standard-Verfahren (Normalisierung der Aufrufe und Rückgabewerte einer Funktion, Kontrollflußgraphen, Static-Single-Assignment-Form) wird während der intraprozeduralen Phase zunächst für jede Funktion ein Graph gebildet, der ihre Effekte zusammenfaßt. Hierzu werden einzelne Graphen für die rein sequentiellen Blöcke (basic blocks) einer Funktion berechnet, die dann nachher unter Zuhilfenahme des Kontrollflußgraphen vereinigt werden. Nachdem die Graphen für einzelne Funktionen erzeugt worden sind, werden anschließend die Teile der Graphen entfernt, die nicht zur Repräsentation der von außen sichtbaren Effekten dienen. Die eine Funktion repräsentierenden Graphen werden auf diese Weise deutlich kleiner, was zu einer Effizienssteigerung für die nachfolgenden Schritte führt. Die reduzierten Graphen werden dann gemäß der existierenden Funktionsaufrufe miteinander vereinigt (interprozedurale Analyse). Hierbei werden indirekte Funktionsaufrufe (über Zeiger auf Funktionen) zunächst ignoriert. Erst wenn im Rahmen der Analyse festgestellt wird welche Funktionen ein solcher Aufruf aufrufen könnte, werden auch die zu diesem Aufruf gehörenden Graphen vereinigt. Funktionen die an mehreren Stellen aufgerufen werden, werden hierbei solange wie möglich unter Berücksichtigung des jeweiligen Aufruf-Kontext betrachtet.

Ein wesentlicher Vorteil dieses Algorithmus besteht darin, daß er für viele reale ANSI-C-Programme verwendet werden kann, da nur sehr wenige Einschränkungen notwendig waren (Assembler-Code, Interrupt-Handling, volatile-Attribute und I/O-basierte Aliase). Der Algorithmus behandelt die Effekte von Strukturen und Unions, beliebiger Verwendung von Zeigern, Typ-Umwandlungen und Zeigern auf Funktionen. Da dies dazu führt, daß sich der Algorithmus nicht auf die vorhanden Typ-Informationen stützen kann, basieren die erzeugten Graphen auf einem Speichermodell auf niedriger Ebene. Der Algorithmus wurde mit Hilfe des SUIF-Compilers implementiert und erfolgreich mit einer Reihe von realen Programmen getestet.