

Tool-Radar: Coverity Prevent und SAT Solver

Jüngste Innovationen bei der statischen Analyse bieten Entwicklern von Multithreaded-Anwendungen Support, um die gut versteckten Defekte, wie Race Condition, Deadlock und Thread Block, schon früh im Entwicklungszyklus einer Anwendung aufzuspüren und zu eliminieren. Mithilfe von Coverity Prevent und der auf Boolescher Satisfiability (SAT) basierenden Softwareanalyse-Engine können Fehler bereits vor der Laufzeit aufgedeckt werden. Das Tool testet automatisch mithilfe statischer Analysen mögliche Pfade und Werte im Code, sodass es den Aufbau von Multithreaded-Anwendungen versteht und komplexe Concurrency-Fehler findet. Herzstück des Tools ist die Software DNA Map – sie erzeugt eine Bit-genaue Repräsentation des Softwaresystems, in der jeder mögliche Softwarevorgang in boolesche Werte (wahr und falsch) und boolesche Operatoren (z.B. und, nicht, oder) übersetzt wird. Diese Repräsentation ermöglicht die Analyse von Quellcode mithilfe von SAT-basierten Solvieren. Ein SAT Solver ist ein Programm mit einer Formel mit Variablen, die durch und, oder, nicht verknüpft sind. Der SAT Solver determiniert, ob sich die einzelnen Variablen so den Werten wahr oder falsch zuordnen lassen, sodass das Ergebnis der gesamten Formel den Wert wahr annimmt und SAT damit erfüllt ist. Erfüllt wenigstens eine Zuordnung diese Voraussetzung, gibt der SAT Solver eine spezielle Erfüllbarkeitsanweisung aus. Wenn nicht, kennzeichnet der Solver die Formel als „Nicht erfüllbar“. Außerdem kann er die Richtigkeit seiner Entscheidung demonstrieren. Voraussetzung für die Anwendung von SAT auf Software ist, dass der Quellcode in einer Form vorliegt, die automatisch an den SAT Solver übergeben werden kann. Hier setzt die Software DNA Map an: Sie macht alle für die Umwandlung in die gewünschte Darstellung benötigten Informationen verfügbar. Da SAT Solver mit den Operatoren wahr, falsch,

und, nicht, oder arbeiten, lassen sich die relevanten Teile eines Programms in die benötigten Konstrukte wandeln. Als Beispiel dient hier die 8-Bit-Variable `char a`; Um `a` als wahr- oder falsch-Wert darzustellen, ist es hilfreich, sich ihre acht Bits (digitale Nullen und Einsen) als wahr beziehungsweise falsch vorzustellen. Damit wird `a` zu einem Array aus acht booleschen Werten:

```
a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7
```

Zusätzlich ist die Übersetzung der Operationen nötig, die im Programmcode Teil von Ausdrücken sind. Alle Ausdrücke im Code lassen sich in eine äquivalente Formel transformieren, die mit den Operatoren und, nicht, oder arbeitet. Das ist erforderlich, da ein Compiler die Operationen in Maschinencodeinstruktionen umwandeln muss, die ein Prozessor abarbeiten kann. Vereinfacht ausgedrückt schickt der Schaltkreis des Chips nur Nullen und Einsen (niedriges und hohes Spannungspotenzial) durch eine Vielzahl von Gates, die sich wie und-, nicht-, oder-Operatoren verhalten. Daher ist die oben gemachte Zuordnung zulässig. Beispielsweise lässt sich der Ausdruck `a = 19` in folgende Formel konvertieren:

```
!a0 ^ !a1 ^ !a2 ^ a3 ^ !a4 ^ !a5 ^ a6 ^ a7
```

`a0` ist hier das höchste Bit von `a` (high bit), `a7` das niedrigste (low bit). Die Eingabe dieser Formel in einen SAT Solver würde die folgende Variablenzuordnung ergeben, mit der die Erfüllbarkeit nachgewiesen wäre:

```
a0 = False (0)
a1 = False (0)
a2 = False (0)
a3 = True (1)
a4 = False (0)
```

```
a5 = False (0)
a6 = True (1)
a7 = True (1)
```

Diese achtstellige Binärzahl (00010011) ist äquivalent zu 19. Nach der Überführung der gesamten Software DNA Map in eine Kombination aus den booleschen Operatoren wahr, falsch, und, nicht, oder lassen sich zahlreiche Formeln daraus ableiten. SAT Solver können den Code dann analysieren und zusätzliche, komplexere Qualitäts- und Security-Schwachstellen aufspüren. Diese Bit-akkurate Darstellung der Software ist die Grundlage für eine präzisere statische Analyse, als bislang basierend auf Datenflusstechniken erreichbar war.

Boolesche Erfüllbarkeit (Boolean Satisfiability)

In der Komplexitätstheorie ist das Boolean-Satisfiability-Problem (SAT) ein Entscheidungsproblem, dessen Instanz ein boolescher Ausdruck ist, formuliert durch und-, oder-, nicht-Operatoren, Variablen und Parenthesen.

Die Kombination von leistungsfähigen Techniken aus dem Bereich der Booleschen Erfüllbarkeit mit Software, die die neuesten Entwicklungen aus der Datenflussanalyse integriert, erzielt durchschlagende Ergebnisse: Diese fortschrittliche Quellcodeanalyse kann – Out-of-the-Box eingesetzt – die Fehlerraten für „false positive“-Defekte auf unter 10 Prozent drücken und gleichzeitig die Skalierbarkeit auf mehrere zehn Millionen Zeilen C/C++- oder Java-Code erweitern. Coverity Prevent mit dem integrierten SAT Solver ist ab sofort für C, C++ und Java verfügbar. Die Preise richten sich nach Projektumfang. *Ben Chelf und John Kodumal*

Links & Literatur

[1] www.coverity.com