

Modul “Algorithmen zur Software-Verifikation”

Ziel

In der Vorlesung werden neuere Algorithmen vorgestellt, die eine automatische Analyse und Verifikation komplexer Strukturen ermöglichen, wie sie bei Software-Systemen vorkommen. In den Übungen werden diese Algorithmen implementiert und an Fallstudien erprobt. Es werden folgende Kenntnisse, Fertigkeiten und Kompetenzen vermittelt:

- **Kenntnisse:**
Kripke-Strukturen, Transitionssysteme, temporale Logiken CTL und CTL*, Fixpunkt-Algorithmen für rekursive CTL-Operatoren, Model-Checking Algorithmus für CTL, Simulation und Bisimulation auf Kripke-Strukturen, Sätze über die Erhaltung von Eigenschaften bei (Bi-) Simulationen, existenzielle und universelle Abstraktion von Kripke-Strukturen, Abstraktions-Verfeinerungs-Schleife (CEGAR-Methode)
- **Fertigkeiten:**
CTL-Model-Checking an Beispielen durchführen, Konstruktion abstrakter Kripke-Strukturen an Hand vorgegebener Datenabstraktionen, Abstraktions-Verfeinerungs-Schleife am Beispiel durchführen
- **Kompetenzen:**
Spezifikation von reaktiven Systemen mit Hilfe von Kripke-Strukturen und CTL-Formeln, Model-Checking-Verfahren in praktische Algorithmen (in Java) umsetzen, Konzept der Simulation und Bisimulation verstanden, Konzept der Abstraktion von Daten und Transitionen verstanden, Model-Checking-Verfahren als Instanzen von Fixpunkt-Algorithmen sehen

Inhalt

Software-Systeme weisen komplexe Daten- und Kontrollstrukturen sowie immer größere Zustandsräume auf, so dass sie durch Testen nur unzulänglich auf ihre Korrektheit überprüft werden können. Es ist daher eine große Herausforderung an die Informatik, automatische Methoden zur Analyse und Verifikation von Verhaltenseigenschaften für Software-Systeme zu entwickeln.

In dieser Lehrveranstaltung werden State-of-the-Art-Algorithmen aus den Bereichen der Programmanalyse und des Model-Checkings vorgestellt und praktisch erprobt. Die Algorithmen verarbeiten Transitionssysteme, wie sie aus Software-Systemen entstehen, und benutzen Abstraktionstechniken für Daten und Transitionen, um die Zustandsräume analysierbar zu machen.

Insbesondere werden folgende Konzepte erklärt: Transitionssysteme, Temporale Logik CTL, (Bi-) Simulationen, abstrakte Interpretation, Shape Analysis für Pointer-Strukturen, Abstraktions-Verfeinerungs-Schleife.

Literatur

E.M. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 2000.

F. Nielson, H.R. Nielson, and C. Hankin. *Principles of Program Analysis*, Springer, 2005

E.M. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith, *Counterexample-guided abstraction refinement for symbolic model checking*, Journal of the ACM 50(5) 752-794 (2003)