Verifikation paralleler Programme mit den Methoden Owicki-Gries und Rely-Guarantee in Isabelle/HOL

Leonor Prensa Nieto

Verifikation paralleler Programme

Motivation

- Parallele Programme oft sicherheitskritisch
- Komplexe Programmier- und Spezifikationssprachen
- Verifikation ist notwendig, schwierig und aufwendig

Ziele

- Beweisunterstützung für beliebig komplexe Programme
- Möglichst große Automatisierung

Umfang

Programmiersprache: $c_1 \parallel \cdots \parallel c_k$

- Parallele Programme mit gemeinsamen Variablen
- c_i sequentielle While-Programme mit Synchronisierung
- Parametrisierte parallele Programme:

$$\|_{i=0}^{n} c(i) \equiv c(0) \parallel c(1) \parallel \ldots \parallel c(n)$$

Verifikationsmethode: Hoare Logik für parallele Programme

Sicherheits-Eigenschaften: partielle Korrektheit, gegenseitiger Ausschluß, Deadlock freedom...

Maschinelle Unterstützung: Interaktiver Theorembeweiser

Beitrag

Theorie

- Formalisierung in Isabelle/HOL von zwei Verifikationsmethoden: Owicki-Gries und Rely-Guarantee (Syntax, Semantik, Beweissystem, Korrektheitsbeweis)
- Verifikation parametrisierter paralleler Programme möglich
- Vollständigkeit für parametrisierte parallele Programme

Anwendung

- Weitgehende Automatisierung des Verifikationsprozesses
- Umfangreiche Verifikationsbeispiele

Verifikationsmethode: Hoare Logik

- Inferenzregeln zum Ableiten von gültigen Hoare Tripeln
- $\models \{P\}\ c\ \{Q\} \iff$ jede terminierende Ausführung mit Anfangszustand in P, terminiert in einem Endzustand in Q (partielle Korrektheit)
- Hoare Logik kompositional für sequentielle Programme

$$\frac{\vdash \{P\}\ c_0\ \{M\} \quad \vdash \{M\}\ c_1\ \{Q\}}{\vdash \{P\}\ c_0;\ c_1\ \{Q\}}\,(Sequenz)$$

Interferenzproblem

$\{x{=}0\}\ x{:=}x{+}2\ \{x{=}2\} \iff \{x{=}0\}\ x{:=}x{+}1;\ x{:=}x{+}1\ \{x{=}2\}$

$\{\mathsf{True}\}\ x{:=}0\ \{x{=}0\} \iff \{\mathsf{True}\}\ x{:=}0\ \{x{=}0\}$

$$
\begin{array}{ccc}
\{x{=}0\} & & \{x{=}0\} \\
x{:=}0 \parallel x{:=}x{+}2 & & x{:=}0 \parallel x{:=}x{+}1;\ x{:=}x{+}1 \\
\{x{=}0 \lor x{=}2\} & \neq & \{x{=}0 \lor x{=}1 \lor x{=}2\} \\
=? & & =? \\
\{\mathsf{Op}\ (x{=}0,\ x{=}2)\} & = & \{\mathsf{Op}\ (x{=}0,\ x{=}2)\}
\end{array}
$$

Klassische Hoare Logik nicht auf Parallelismus übertragbar

Die Owicki-Gries Methode

- Adaption des Systems von Hoare für die Verifikation paralleler Programme mit gemeinsamen Variablen
- Hauptidee: Komponenten sind als Beweisskizze (proof outlines) spezifiziert. Diese sollen interferenzfrei sein.

Pre-post Spezifikation

$\{x=0\}$
$x:=x+1;$
$x:=x+1$
$\{x=2\}$

Beweisskizze

$\{x=0\}$
$x:=x+1;$
$\{x=1\}$
$x:=x+1$
$\{x=2\}$

Regel für die parallele Komposition

Wenn die Beweisskizzen für $c(0), \ldots, c(n)$ interferenzfrei sind, dann

$$\frac{\forall i \leq n.\ \vdash \{P(i)\}\ c(i)\ \{Q(i)\}}{\vdash \{\bigwedge_{i=0}^{n}\ P(i)\}\ \|_{i=0}^{n}\ c(i)\ \{\bigwedge_{i=0}^{n}\ Q(i)\}}$$

Beitrag: Parametrisierte parallele Programme lassen sich in einer einzigen Ableitung im System verifizieren

Interferenzfreiheit

Zwei Beweisskizzen P_1 und P_2 sind interferenzfrei $\Longleftrightarrow$

$$\forall p \in \text{Zusicherungen von } P_1 \wedge$$
$$\forall a \in \text{atomare Anweisungen von } P_2,$$
$$\{p \wedge \text{pre}(a)\}\ a\ \{p\}$$

(und umgekehrt)

Probleme

- P_1 und P_2 jeweils n und m Anweisungen
  $\leadsto \mathcal{O}(n \times m)$ Hoare Tripeln
- Interne Implementierung aller Komponenten verwendet
  $\leadsto$ nicht kompositional

Die Rely-Guarantee Methode

- Kompositionale Version der Owicki-Gries Methode
- Hauptidee: jede Komponente ist mit zwei zusätzlichen Bedingungen spezifiziert:

$$\langle rely, guar \rangle : \{P\}\ c\ \{Q\}$$

  - $rely \rightsquigarrow$ erlaubte Interferenzen von der Umgebung
  - $guar \rightsquigarrow$ Wirkung von der Komponente selbst

- $rely$ und $guar$ sind Prädikate über Zustandspaare $(s,\ s')$
  Beispiel: $rely = (x = x')$ $\qquad$ $guar = (x < x')$

Regel für die parallele Komposition

$$\frac{\begin{array}{l}(rely \vee guar_1) \rightarrow rely_2 \\ (rely \vee guar_2) \rightarrow rely_1 \\ (guar_1 \vee guar_2) \rightarrow guar \\ \vdash \langle rely_1,\ guar_1 \rangle : \{P_1\}\ c_1\ \{Q_1\} \\ \vdash \langle rely_2,\ guar_2 \rangle : \{P_2\}\ c_2\ \{Q_2\}\end{array}}{\vdash \langle rely,\ guar \rangle : \{P_1 \wedge P_2\}\ c_1 \parallel c_2\ \{Q_1 \wedge Q_2\}}$$

Vorteile gegenüber Owicki-Gries:

1. Kompositional
2. Geeignet für Verifikation offener Systeme

$\rightsquigarrow$ Skalierbar

Formalisierung der Systeme

Theorien	Spezif.	Lemmata	Interaktionen
Owicki-Gries	220	49	340
Rely-Guarantee	330	93	2240
Total	**550**	**142**	**2580**

Neu beim Korrektheitsbeweis

- Programm-Labels werden nicht benötigt (O-G)
- Modulare Definition von Ausführung (R-G)
- Korrektheitsbeweis für parametrisierte parallele Programme

Vollständigkeit für parametrisierte Programme

$$\models \{x = 0\} \; \|_{i=1}^{n} \; x := x + 1 \; \{x = n\}$$

- Die nicht-parametrisierten Versionen von O-G und R-G Methoden sind vollständig

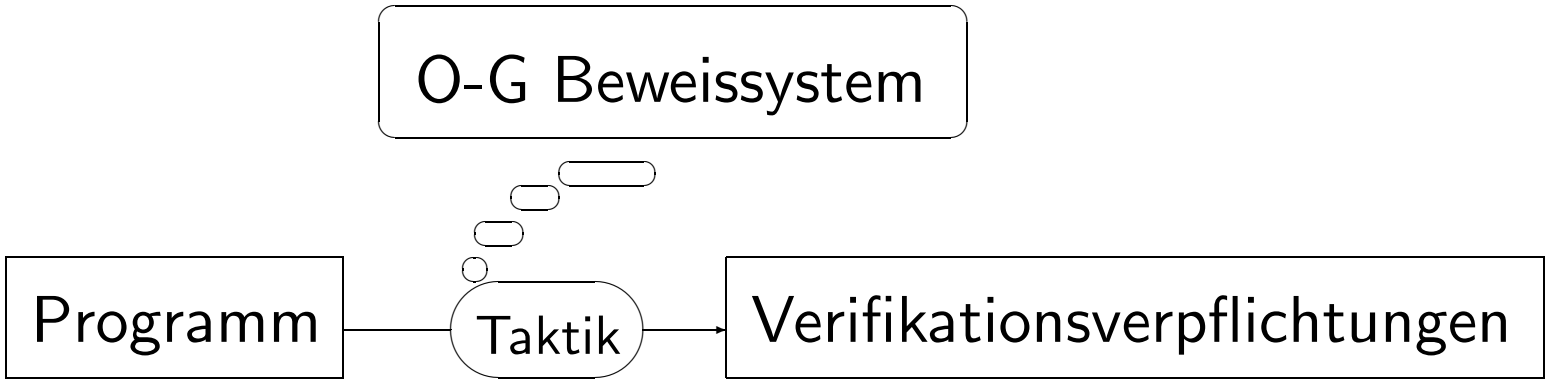
  $\rightsquigarrow$ für jedes n existiert eine richtige O-G (R-G) Annotation

  $\rightsquigarrow$ möglicherweise n verschiedene Annotationen

- Frage: Gibt es eine einzige Annotation, die für jeden Wert von n funktioniert?

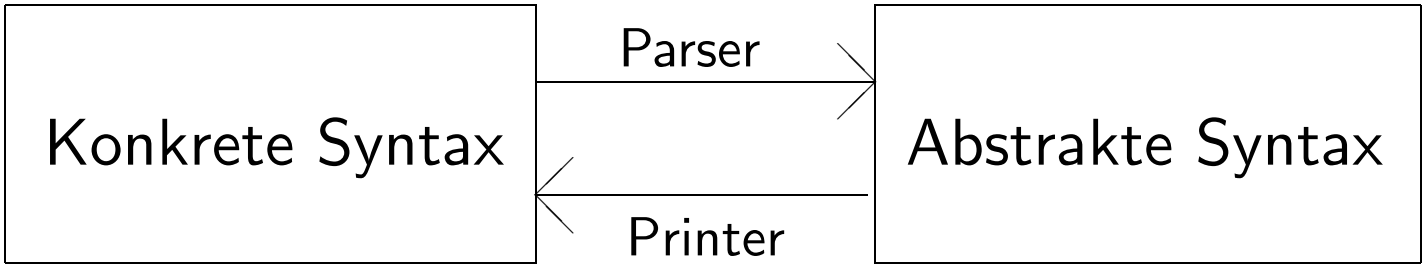
Beitrag: Beide Systeme sind auch für parametrisierte parallele Programme vollständig

Generierung von Beweisverpflichtungen

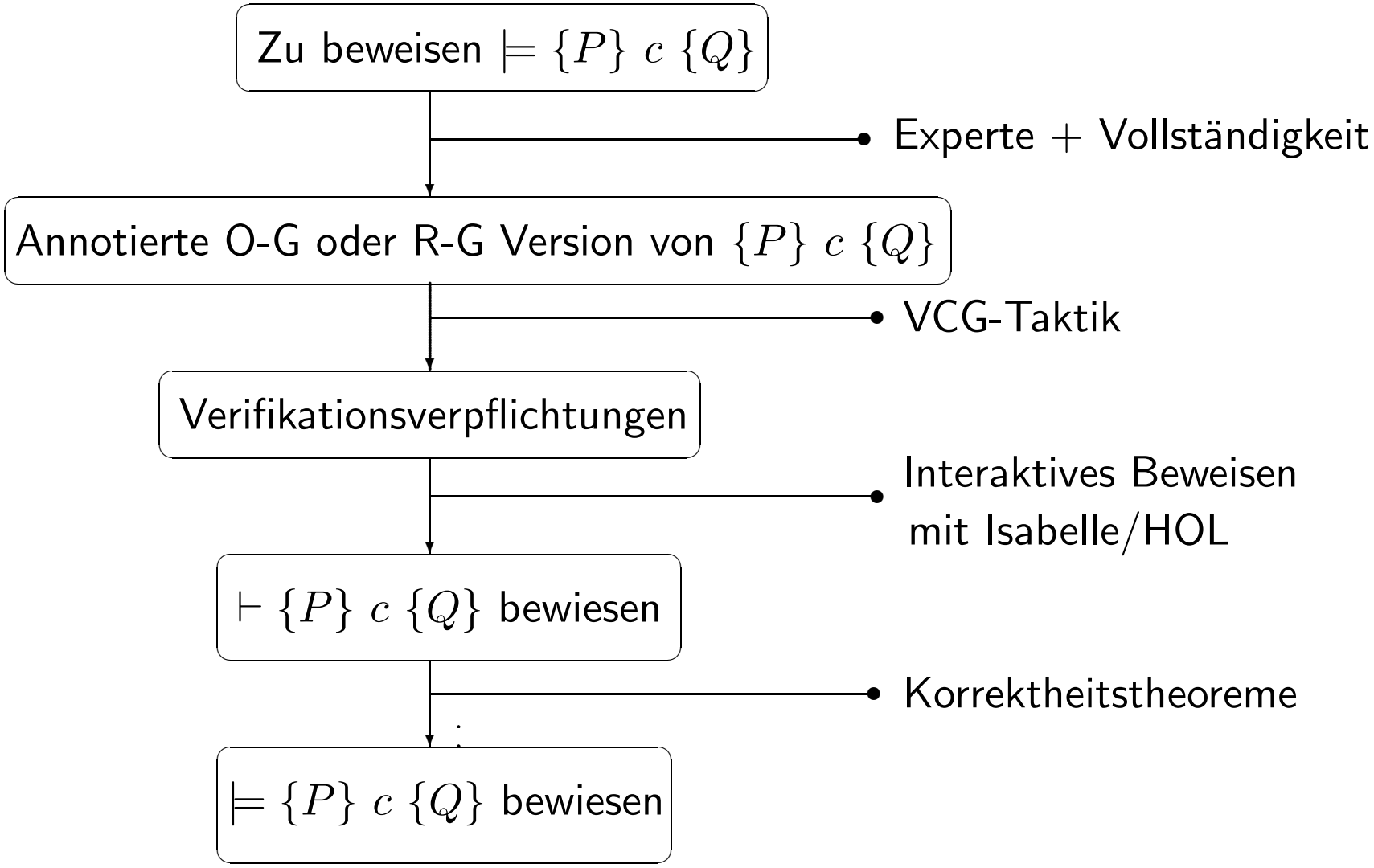


- Inferenzregeln werden systematisch rückwärts angewandt
- Taktik in ML programmiert

Konkrete Syntax



$$
\begin{array}{l}
\{x{=}0 \wedge y{=}0\} \\
\quad x{:=}1 \parallel y{:=}2 \\
\{x{=}1 \wedge y{=}2\}
\end{array}
\quad \longleftrightarrow \quad
\begin{array}{l}
\{(x,y).\ x{=}0 \wedge y{=}0\} \\
[\text{Basic } \lambda(x,\ y).\ (1,\ y), \\
\text{Basic } \lambda(x,\ y).\ (x,\ 2)] \\
\{(x,y).\ x{=}1 \wedge y{=}2\}
\end{array}
$$


Zu beweisen $\models \{P\}\ c\ \{Q\}$
Experte + Vollständigkeit
Annotierte O-G oder R-G Version von $\{P\}\ c\ \{Q\}$
VCG-Taktik
Verifikationsverpflichtungen
Interaktives Beweisen
mit Isabelle/HOL
$\vdash \{P\}\ c\ \{Q\}$ bewiesen
Korrektheitstheoreme
$\models \{P\}\ c\ \{Q\}$ bewiesen

Beispiele

Owicki-Gries

Algorithmus	Beweisverpfl.	Interaktionen
Peterson	122	1
Dijkstra	20	1
Ticket	35	60
Zero search	98	2
Producer/Consumer	138	35
Single mutator gar-coll	289	708
Multi-mutator gar-coll	328	956

Neu: Erste komplette O-G Beweis für Gar-Coll

Rely-Guarantee

Algorithmus	Beweisverpfl.	Interaktionen
Set array to 0 (param)	8	40
Increment variable(param)	14	23
Find least element in array	22	30

Bemerkungen

- Theoretisch können Programme immer kompositional verifiziert werden
- Praktisch Owicki-Gries besser geeignet für Algorithmen mit gemeinsamen Variablen

Zusammenfassung

- Erste Formalisierung der Owicki-Gries und Rely-Guarantee Methoden in einem Theorembeweiser
- Technische Verbesserungen gegenüber den originalen Formalisierungen der Literatur
- Erster Vollständigkeitsbeweis für parametrisierte Programme
- Anwendbarkeit: konkrete Syntax, Automatisierung, Beispiele
- Werkzeug hilfreich auch bei der Suche nach Beweisskizzen

Zukünftige Arbeiten

- Formalisierung der Vollständigkeitsbeweises in Isabelle/HOL
- Erweiterung der Programmiersprache
- Mehr Anwendungsbeispiele
- Systematische Übertragung von O-G Beweisen in R-G Beweise