

Entscheidungsverfahren mit Anwendungen in der Softwareverifikation

I: Einführung

Carsten Sinz
Institut für Theoretische Informatik

17.10.2018

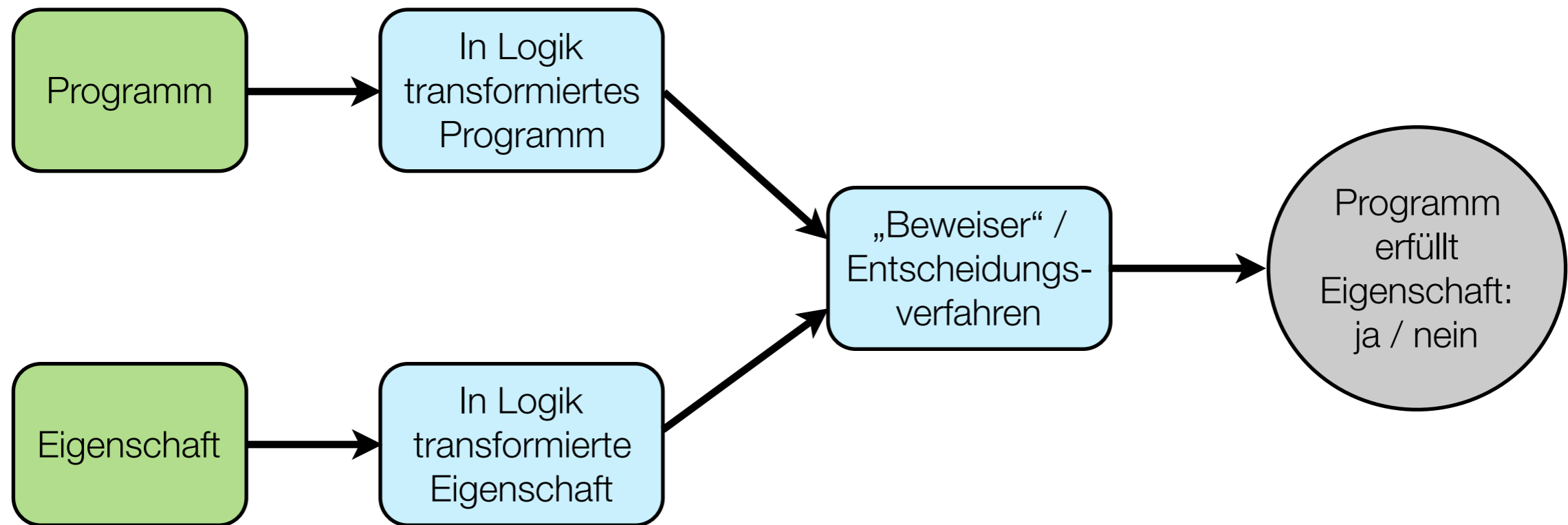
Ist mein Programm korrekt?

- **Beispiel:** Bestimme Anzahl der 1-Bits in einem Wort (*population-count*)

```
uint32_t reference_popcount(uint32_t x)
{
    int i, s = 0;
    for(i = 0; i < 32; ++i)
        if(x & (1 << i))
            s++;
    return s;
}
```

```
uint32_t optimized_popcount(uint32_t x)
{
    x = x - ((x >> 1) & 0x55555555);
    x = (x & 0x33333333) + ((x >> 2) & 0x33333333);
    x = (x + (x >> 4)) & 0x0F0F0F0F;
    x += (x >> 8);
    x += (x >> 16);
    return x & 0x0000003F;
}
```

- **Frage:** Liefert die optimierte Version immer das gleiche Ergebnis wie die Referenzimplementierung?



- **Welche Programme wollen wir prüfen?**

- C, C++, Java, ... (Imperative Programmiersprachen)

- **Welche Eigenschaften wollen wir prüfen?**

- Keine Programmabstürze

- Keine „falschen“ Rechenergebnisse

- Keine Sicherheitslücken

- **Welche Logik ist dafür geeignet?**

- Modellierung von primitiven Datentypen (**int**, **float**, ...) und Operationen darauf

- Modellierung von komplexen Datentypen

- Modellierung von Programmfluss (Daten- und Kontrollfluss)

- Modellierung von Speicher

Welche Logik?

- **Erwünschte Eigenschaften:**
 - Möglichst ausdrucksstark
 - Möglichst entscheidbar
 - Eingebaute Theorien (z.B. Arithmetik)
- **Kandidaten:**
 - Aussagenlogik
 - Prädikatenlogik erster Stufe
 - CTL / LTL
 - Spezielle Logiken

- **Definition (Erfüllbarkeit, Gültigkeit):**
Eine Formel ist *erfüllbar*, wenn es *eine* Variablenbelegung und eine Interpretation der (nicht-logischen) Symbole *gibt*, so dass die Formel zu **wahr** evaluiert. Eine Formel ist gültig, wenn sie für *alle* Variablenbelegungen und *alle* Interpretationen der (nicht-logischen) Symbole zu **wahr** evaluiert.
- **Definition (Entscheidungsproblem):**
Das *Entscheidungsproblem* für eine gegebene Formel (in einer Logik L) ϕ ist zu bestimmen, ob ϕ gültig ist.
- **Definition (Korrektheit):**
Ein Algorithmus für das Entscheidungsproblem ist *korrekt*, falls immer wenn er „gültig“ als Ergebnis liefert die Eingabeformel wirklich gültig ist.
- **Definition (Vollständigkeit):**
Ein Algorithmus für das Entscheidungsproblem ist *vollständig*, wenn er (a) immer terminiert und (b) „gültig“ als Ergebnis liefert, sofern die Eingabeformel gültig ist.
- **Definition (Entscheidungsverfahren):**
Ein Algorithmus A heißt *Entscheidungsverfahren* für eine Logik L, wenn A korrekt und vollständig ist für jede Formel aus L.
- **Definition (Entscheidbarkeit einer Logik):**
Eine Logik L heißt *entscheidbar*, wenn es ein Entscheidungsverfahren für L gibt.

Entscheidbarkeit für verschiedene Logiken

Logik	Entscheidbarkeit
Prädikatenlogik (erster Stufe)	unentscheidbar
Monadische Prädikatenlogik	entscheidbar
Polynomgleichungen über den ganzen Zahlen	unentscheidbar
Presburger-Arithmetik	entscheidbar
Quantorenfreie Logik der Bitvektoren	entscheidbar

- **Beispiel 1: Lösen quadratischer Gleichungen**

- Wann besitzt die Gleichung $ax^2 + bx + c = 0$ eine Lösung / Lösungen?

- Wir wissen: $x_{1/2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ beschreibt die Lösungen

- Lösbar?

- Hängt von Grundbereich ab, über dem wir die Formel interpretieren!

- Über \mathbb{R} : Lösbar (d.h. erfüllbar), wenn $b^2 - 4ac \geq 0$.

- Über \mathbb{C} : Immer lösbar.

- Über \mathbb{Q} ? Über \mathbb{Z} ? Über algebraischen Körpererweiterungen von \mathbb{Q} ?

- Entscheidungsverfahren / Algorithmus (zumindest für \mathbb{R}, \mathbb{C}) sehr einfach

- Wir sind häufig nicht nur an der Lösbarkeit interessiert, sondern auch an einer oder sogar allen Lösungen.

- **Beispiel 2: Lineare Gleichungssysteme**

- Wann besitzt das lineare Gleichungssystem

$$\begin{aligned} a_{n,1}x_n + a_{n-1,1}x_{n-1} + \cdots + a_{1,1}x_1 &= b_1 \\ &\vdots \\ a_{n,m}x_n + a_{n-1,m}x_{n-1} + \cdots + a_{1,m}x_1 &= b_m \end{aligned}$$

eine Lösung / Lösungen?

- Wie sieht ein Lösungsverfahren aus?
- Gaußsches Eliminationsverfahren (für \mathbb{Q})
- Funktioniert dies auch über \mathbb{Z} ?
- Komplexität?
 - Was ist eine elementare Operation? Koeffizienten können möglicherweise sehr groß werden \rightarrow Bit-Komplexität

- **Entscheidungsverfahren sind abhängig von:**
 - Erlaubten Funktions- und Relationssymbolen (**F**, **R**)
 - Grundbereich, über dem eine Formel interpretiert wird (**U**)
 - Axiomen (**A**), die gelten sollen
- **Satisfiability Modulo Theory (SMT)**
 - SMT-Solver implementieren Entscheidungsverfahren für gegebene **F**, **R**, **U** und **A** (d.h., Axiome und Grundbereich sind „eingebaut“).
 - Standard für SMT-Theorien: SMT-LIB
- **Komplexität in Praxis wichtig**
 - für arithmetische Theorien häufig exponentiell

Theorie	17.10.	Einführung
	24.10.	Theoretische Grundlagen, Beispiele
	31.10.	SAT-Solving: Fortgeschrittene Algorithmen
	07.11.	DPLL(T): DPLL mit Theorie-Atomen
	14.11.	Uninterpretierte Funktionen mit Gleichheit (EUF)
	21.11.	Lineare Arithmetik der ganzen Zahlen (Fourier-Motzkin, Omega-Test)
	28.11.	Logik der Bitvektoren
	05.12.	Array-Logik
	12.12.	Modulare Arithmetik
	19.12.	Quantorenelimination für reell abgeschlossene Körper (CAD)
09.01.	Kombination von Theorien	
Anwendung	16.01.	Polyspace: Abstract Interpretation
	23.01.	LLBMC: Software Bounded Model Checking
	30.01.	KLEE: Symbolic Execution, SATABS: Predicate Abstraction
	06.02.	Lab: Polyspace, LLBMC, KLEE, SATABS

Übungen 14-tägig, erster Termin: 30.10.

Enthält dieser C-Code einen Fehler?

- **Array-index-out-of-bounds error**

- What are the consequences?

Unexpected behavior!

- How can we detect this error?

By testing? **Perhaps (depending on the compiler)**

array index out of bounds in iteration 16

```
int satd = 0, dd, k;
for (dd=d[k=0]; k<16; dd=d[++k]) {
    satd += (dd < 0 ? -dd : dd);
}
```

gcc pre-4.8 generates the following code:

```
SATD:
    ...
.L2:
    jmp .L2
```

[Code from SPEC CPU 2006 Benchmark 464.h264ref; via John Regehr's blog *Embedded in Academia*; March 22, 2013]

- Daniel Kröning / Ofer Strichman:
Decision Procedures – An Algorithmic Point of View
(Springer, 2008)
[Innerhalb des Uni-Netzes ist eine Online-Version im Volltext verfügbar.]
- Aaron R. Bradley / Zohar Manna:
The Calculus of Computation: Decision Procedures with Applications to Verification
(Springer, 2007)
- Weitere Literatur wird während der Vorlesung genannt