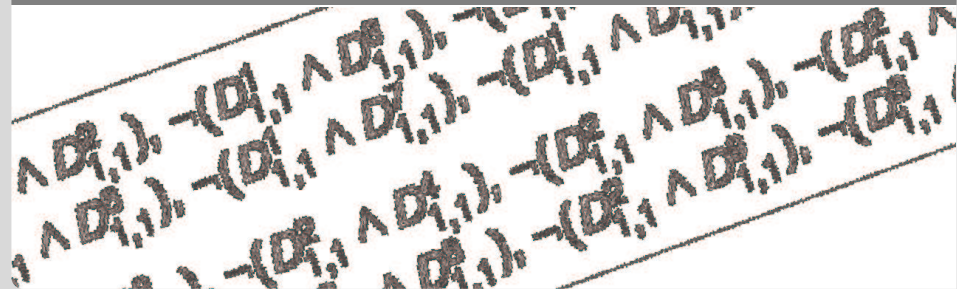


# Anwendungsorientierte Formale Verifikation Praxis der Forschung im SoSe 2017

Bernhard Beckert

Institute for Theoretical Informatics, KIT



# Themenbereiche der Gruppe

Anwendungsbereiche

Security



Safety



Werkzeuge (KeY, Reve)

User Experience/Usability

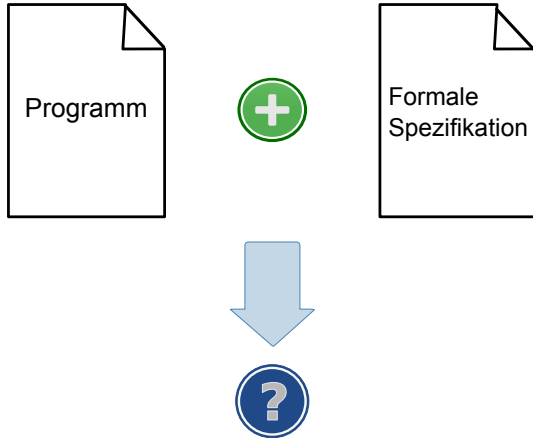
Methoden

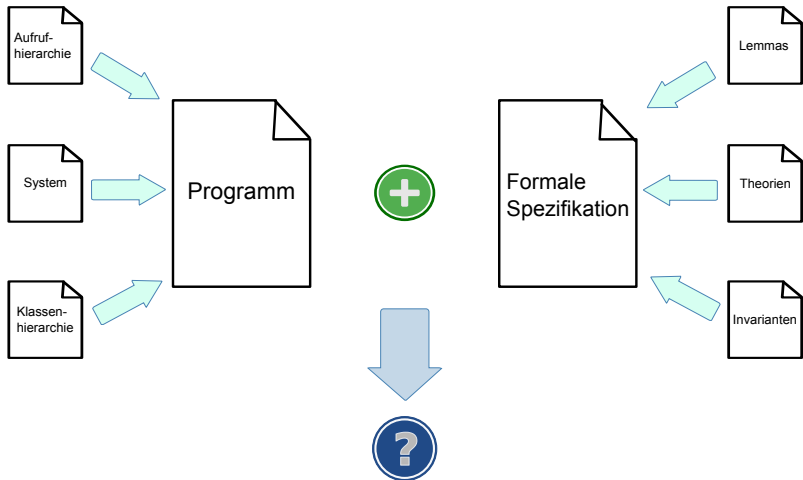
Testen

Verifikation

Bounded Model  
Checking

# Benutzbarkeit von Software- Verifikationssystemen





## Programm-Ebene

```
method MultipleReturn(x: int, y: int) returns (more: int, less: int)
  requires x > 0 && y > 0;
  ensures (y < x) ==> (less > 0) && (x <= y);
  ensures more >= 0;
{
  more := x + y;
  less := x - y;
  assert more > less;
}
```

## Logik-Ebene

```
script bool_simp_splits
test := 6;
foo:int;
repeat
  foreach
    call bool_simp;
  cases
    case (test == 6) && (match '==> (?_ && ?_)?P1'):
      andRight on ?P1;
    case match '(?_ || ?_)?P2 ==> ':
      orLeft on ?P2;
    case match '(?_ -> ?_)?P3 ==> ':
      implLeft on ?P3;
    case match '==> ... bsum{?_}{?_,?N+1,?_}@P4 ... ':
      simp;
      bsum_def on ?P4;
      auto;
    case match '(f(c) = ?X)?P5 ==> ':
      applyEq @P5 on '==>... f(c) ...';
      auto;
  default:
    simp;
end repeat
```

Current goal

```
[heap!Pre:=heap || _a:=a || exc:=null || heap:=heap[self.sun := 0][self.max :
wellformed(heap),
self.<created> = TRUE,
SunAndMax:=exactInstance(self) = TRUE,
a.<created> = TRUE,
measuredByfpty.
V_int | allLeftHide | ≥ 0]
==>
self =
a = nu
(heap)
allLeft
hide_left
cut_direct
More rules
Apply rules
Strategy ma
allLeft {
  find { V u; (i < a.length a i = 0 - a[i] = 0) ==> } ,max :
  add { (subst i:1)(i < a.length a i = 0 - a[i] = 0) ==> }
  heuristics { gamma }
```

Copy to clipboard  
Create abbreviation  
Jump to introduction  
Show History

## Programm-Ebene (Annotationen im Code)

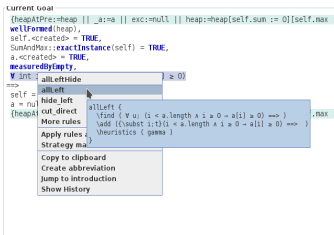
```
method MultipleReturn(x: int, y: int) returns (more: int, less: int)
  requires x > 0 && y > 0;
  ensures (y < x) ==> (less > 0) && (x <= y);
  ensures more >= 0;
{
  more := x + y;
  less := x - y;
  assert more > less;
}
```

## Logik-Ebene

### (Beweisskript)

```
script bool_simp_splits
test := 6;
foo:int;
repeat
  foreach
    call bool_simp;
  cases
    case (test == 6) && (match '==> (?_ && ?_)?P1'):
      andRight on ?P1;
    case match '(?_ || ?_)?P2 ==> ':
      orLeft on ?P2;
    case match '(?_ -> ?_)?P3 ==> ':
      implLeft on ?P3;
    case match '==> ... bsum{?_}{?_,?N+1,?_}@P4 ... ':
      simp;
      bsum_def on ?P4;
      auto;
    case match '(f(c) = ?X)?P5 ==> ':
      applyEq @P5 on '==>... f(c) ...';
      auto;
  default:
    simp;
```

### (Point & Click)



Current goal

```
{heap!Pre:=heap || _a:=a || exc:=null || heap:=heap[self.sun := 0]|self.max;
wellFormed(heap);
self.<created> = TRUE;
SunAndMax::exactInstance(self) = TRUE;
a.<created> = TRUE;
measuredBy!fty.
```

**V.int** | allLeftHide | ≥ 0

==>

- allLeft
- hide\_left
- a = nu
- {heap}
- cut\_direct
- More rules
- Apply rules
- Strategy ma
- Copy to clipboard
- Create abbreviation
- Jump to introduction
- Show History

allLeft {

```
λfind (V u; (i < a.length a i a 0 - a[i] a 0) ==> ) ,max;
λadd ((subst i:1)(i < a.length a i a 0 - a[i] a 0) ==> )
theuristics (gamma )
```

## Programm-Ebene (Annotationen im Code)

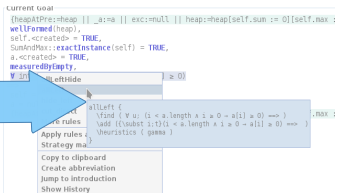
```
method MultipleReturn(x: int, y: int) returns (more: int, less: int)
  requires x > 0 && y > 0;
  ensures (y < x) ==> (less > x <= y);
  ensures more >= 0;
{
  more := x + y;
  less := x - y;
  assert more > less;
}
```

## Logik-Ebene

### (Beweisskript)

```
script bool_simp_splits
test := 6;
foo:int;
repeat
  foreach
    call bool_simp;
  cases
    case (test == 6) && (match '==> (?_ && ?_@?P1)'):
      andRight on ?P1;
      case match '(?_ || ?_)?P2 ==> ':
        orLeft on ?P2;
      case match '(?_ -> ?_)?P3 ==> ':
        impLeft on ?P3;
      case match '==> ... bsum{?_}{?N+1,?_}@P4 ... ':
        simp;
        bsum_def on ?P4;
        auto;
      case match '(f(c) = ?X)?P5 ==> ':
        applyEq @P5 on '==>... f(c) ...';
        auto;
    default:
      simp;
```

### (Point & Click)



Current goal

```
(!heap[Pre:=heap || _a:=a || exc:=null || heap:=heap[self.sun := 0][self.max := 0]
wellFormed(heap,
self.<created> = TRUE,
SunAndMax:=exactInstance(self) = TRUE,
a.<created> = TRUE,
measuredBy!apt,
V := ...]
```

Apply rules  
Strategy ma  
Copy to clipboard  
Create abbreviation  
Jump to introduction  
Show History



## Verbesserung der Benutzbarkeit von Programm-Beweissystemen

### Aufgaben

- Entwicklung eines integrierten Darstellungs-Konzepts basierend auf den verschiedenen Interaktionsparadigmen, bspw.:
  - Darstellung von Abhängigkeiten
  - Fließender Wechsel zwischen den Sichten
- Umsetzung des Konzepts in bestehendem Framework
- Evaluation durch Nutzerstudien (z.B. Eye-Tracking, Fragebogen, etc.)

### Bei Interesse:

- Sarah Grebing [grebing@ira.uka.de](mailto:grebing@ira.uka.de) (ITI Beckert)
- Andrea Schankin [schankin@teco.edu](mailto:schankin@teco.edu) (TM Beigl)
- Mattias Ulbrich [ulbrich@kit.edu](mailto:ulbrich@kit.edu) (ITI Beckert)

# Verifikation OO-Software für Produktionsanlagen

# Verifikation objektorientierter Software für Produktionsanlagen

## Hintergrund

- Steigende **Komplexität** von Anlagensteuerungen
- Einführung von **Objekt-Orientierung**

## Regressionverifikation

Formale Methode zur Garantie der **Verhaltensequivalenz**

## Methoden

Parserbau, Static Analysis, Model-Checking

Bei Interesse: Alexander Weigl ([weig@kit.edu](mailto:weig@kit.edu))

# Verifikation objektorientierter Software für Produktionsanlagen

## Hintergrund

- Steigende **Komplexität** von Anlagensteuerungen
- Einführung von **Objekt-Orientierung**

## Regressionverifikation

**Formale Methode** zur Garantie der **Verhaltensequivalenz**

## Methoden

Parserbau, Static Analysis, Model-Checking

Bei Interesse: Alexander Weigl ([weig@kit.edu](mailto:weig@kit.edu))

# Verifikation objektorientierter Software für Produktionsanlagen

## Hintergrund

- Steigende **Komplexität** von Anlagensteuerungen
- Einführung von **Objekt-Orientierung**

## Regressionverifikation

**Formale Methode** zur Garantie der **Verhaltensequivalenz**

## Methoden

Parserbau, Static Analysis, Model-Checking

Bei Interesse: Alexander Weigl ([weig@kit.edu](mailto:weig@kit.edu))

# Verifikation objektorientierter Software für Produktionsanlagen

## Hintergrund

- Steigende **Komplexität** von Anlagensteuerungen
- Einführung von **Objekt-Orientierung**

## Regressionverifikation

**Formale Methode** zur Garantie der **Verhaltensequivalenz**

## Methoden

Parserbau, Static Analysis, Model-Checking

Bei Interesse: Alexander Weigl ([weig@kit.edu](mailto:weig@kit.edu))