

# **PSE - Abschlusspräsentation**

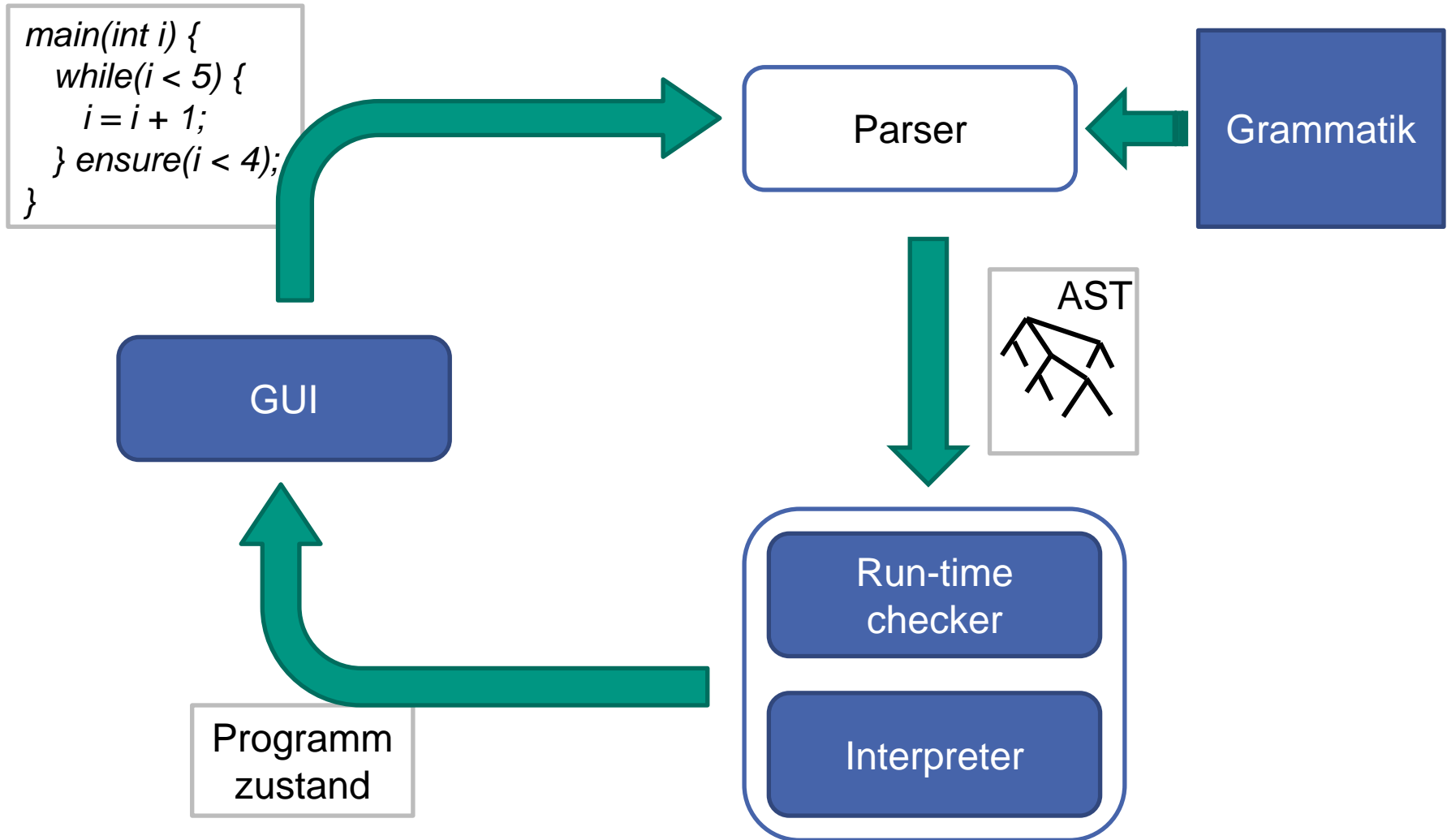
## **Automatisches Prüfen der Korrektheit von Programmen**

**Simon Bischof, Jan Haag, Adrian Hermann, Lin Jin, Tobias Schlumberger, Matthias Schnetz**

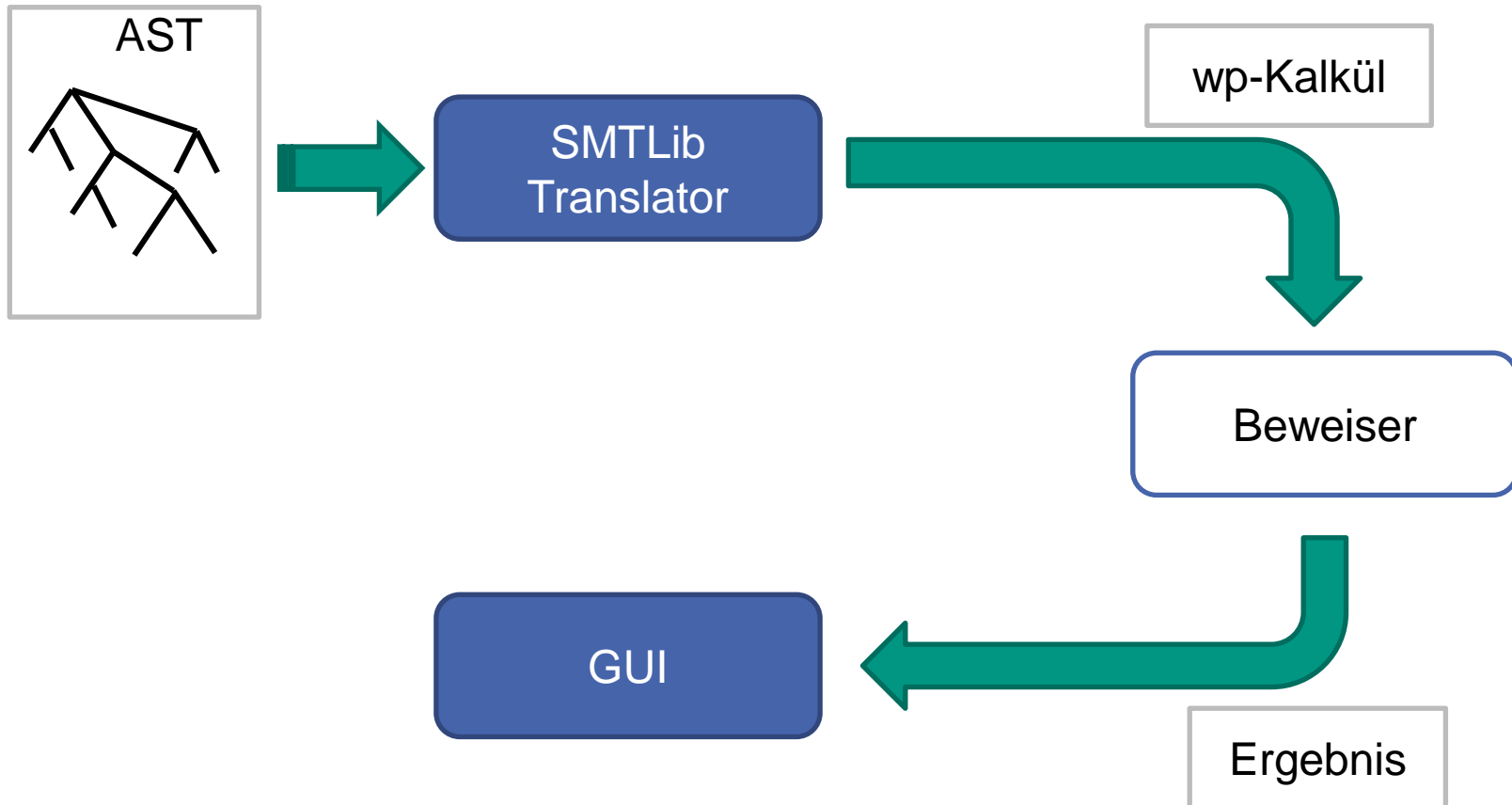
Institut für Theoretische Informatik – Anwendungsorientierte formale Verifikation

PLATZHALTER

# Aufgabenstellung: Parser & Interpreter



# Aufgabenstellung: Formale Verifikation



# Aufgabenstellung: Formale Verifikation

Zur Verifikation eines Programmes sind zwei Schritte nötig:

- ① Ziel des Programms klarmachen
  - Was ist der Sinn dieses Programms?
  - Welches Endergebnis erwarte Ich?
  - Wie soll dieses Ergebnis erreicht werden?

# Aufgabenstellung: Formale Verifikation

Zur Verifikation eines Programmes sind zwei Schritte nötig:

- ① Ziel des Programms klarmachen
  - Was ist der Sinn dieses Programms?
  - Welches Endergebnis erwarte Ich?
  - Wie soll dieses Ergebnis erreicht werden?
  
- ② Das Programm um beweisbare Annotationen erweitern
  - Was gilt für die Variablen?
  - Wie lauten geeignete Invarianten für Schleifen?

# Aufgabenstellung: Formale Verifikation

- 1 Ziel des Programms klarmachen
  - Was ist der Sinn dieses Programms?
  - Welches Endergebnis erwarte Ich?
  - Wie soll dieses Ergebnis erreicht werden?

PLATZHALTER FÜR BEISPIEL – KOMMT BIS  
MITTWOCH

# Aufgabenstellung: Formale Verifikation

- 2 Das Programm um beweisbare Annotationen erweitern
  - Was gilt für die Variablen?
  - Wie lauten geeignete Invarianten für Schleifen?

PLATZHALTER FÜR BEISPIEL – KOMMT BIS  
MITTWOCH

# Kenndaten

- 6 Entwickler
- 17.000 LOC
  - 100 Klassen
  - 15 Pakete
- Lauffähig unter:
  - Windows XP & Windows 7
  - Linux
  - Mac OS X
- Beweisbare Programme:
  - Summe von 1 bis  $n$
  - Russische Multiplikation

9.500 Zeilen  
Code

3.500 Zeilen  
generierter Code

3.500 Zeilen  
Code aus der QS

800 Zeilen  
Dokumentation



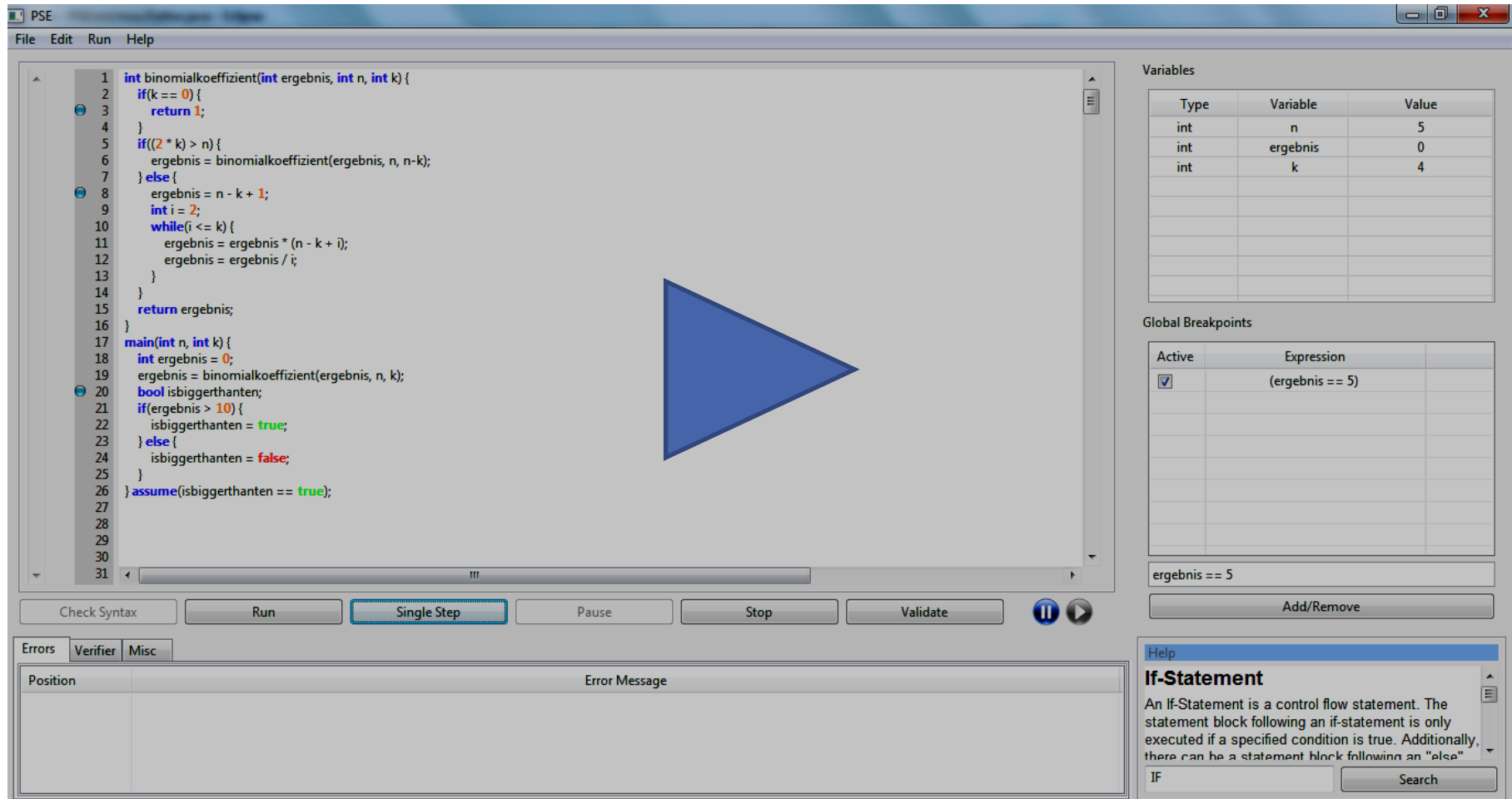
# Zukunft von wProof

- Einsatzgebiete von formaler Verifikation:
  - Hardware-Bereich: Entwicklung von Prozessoren
  - Software-Bereich: Systeme bei denen Zuverlässigkeit wichtig
  
- Es existieren andere Tools mit größerem Funktionsumfang und Entwicklerteams mit mehr Erfahrung/Mitteln
  - KeY
  - Isabelle/HOL



- ➡ Tool ist ein guter Einstieg in den Themenbereich der formalen Verifikation (v.a. auch für Studenten)
- ➡ Lizenzierung unter BSD Lizenz

# Tool Demonstration



**Variables**

Type	Variable	Value
int	n	5
int	ergebnis	0
int	k	4

**Global Breakpoints**

Active	Expression
<input checked="" type="checkbox"/>	(ergebnis == 5)

ergebnis == 5

Add/Remove

**Help**

### If-Statement

An If-Statement is a control flow statement. The statement block following an if-statement is only executed if a specified condition is true. Additionally, there can be a statement block following an "else".

IF  Search

```
1 int binomialkoeffizient(int ergebnis, int n, int k) {
2   if(k == 0) {
3     return 1;
4   }
5   if((2 * k) > n) {
6     ergebnis = binomialkoeffizient(ergebnis, n, n-k);
7   } else {
8     ergebnis = n - k + 1;
9     int i = 2;
10    while(i <= k) {
11      ergebnis = ergebnis * (n - k + i);
12      ergebnis = ergebnis / i;
13    }
14    return ergebnis;
15  }
16 }
17 main(int n, int k) {
18   int ergebnis = 0;
19   ergebnis = binomialkoeffizient(ergebnis, n, k);
20   bool isbiggerthanten;
21   if(ergebnis > 10) {
22     isbiggerthanten = true;
23   } else {
24     isbiggerthanten = false;
25   }
26   assume(isbiggerthanten == true);
27 }
28
29
30
31
```

Check Syntax Run Single Step Pause Stop Validate

Errors Verifier Misc

Position	Error Message
----------	---------------

**(KEIN Teil der Präsentation)**

---

HIER BEGINNT DER  
„INTERNE“ TEIL

# Soll / Haben

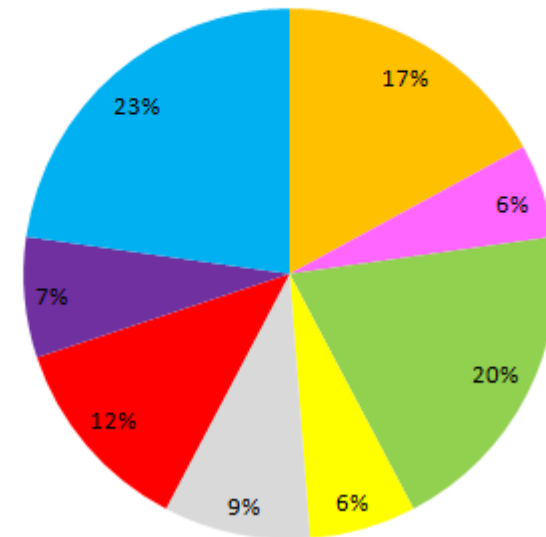
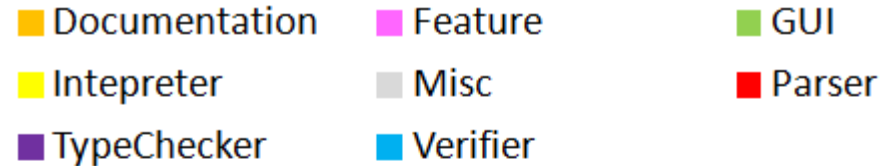
---

- Wird noch ausgearbeitet bis Mittwoch

# Qualitätssicherung

## ■ Werkzeuge

- Bug - Tracker
- JUnit
- GUI - Testplan



➡ 108 gefundene Bugs

➡ Code Coverage: ca. 90 %

## Probleme:

- Klare Abgrenzung der einzelnen Module zum Teil schwierig
- Testen von generiertem Code nur teilweise möglich

# Herausforderungen & Erfahrungen

---

## ■ Herausforderungen

- Grammatikerstellung für Parser erfordert Behandlung vieler Spezialfälle
- Themengebiet der formalen Verifikation sehr abstrakt / komplex
- In manchen Phasen hoher Zeitdruck

## ■ Erfahrungen

- Fehler / ungelöste Probleme des Entwurfs wirken sich sehr stark auf die Implementierung aus
- Zeitplanung
- Beteiligung/Engagement schwankt zwischen wenig und viel
- Qualitätssicherung bringt mehr Fehler zum Vorschein als erwartet