Karlsruher Institut für Technologie

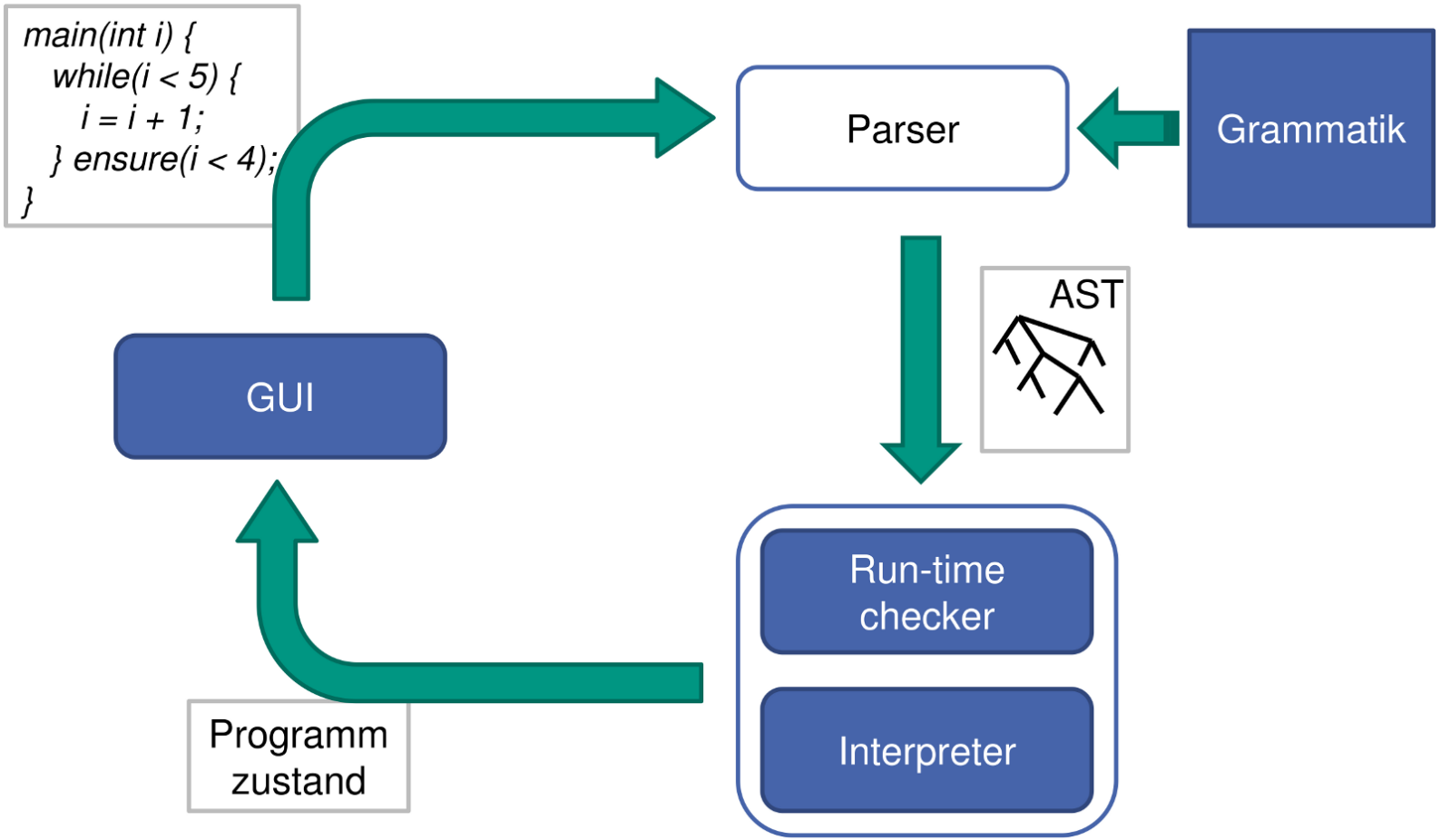
W$_e$Prove

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

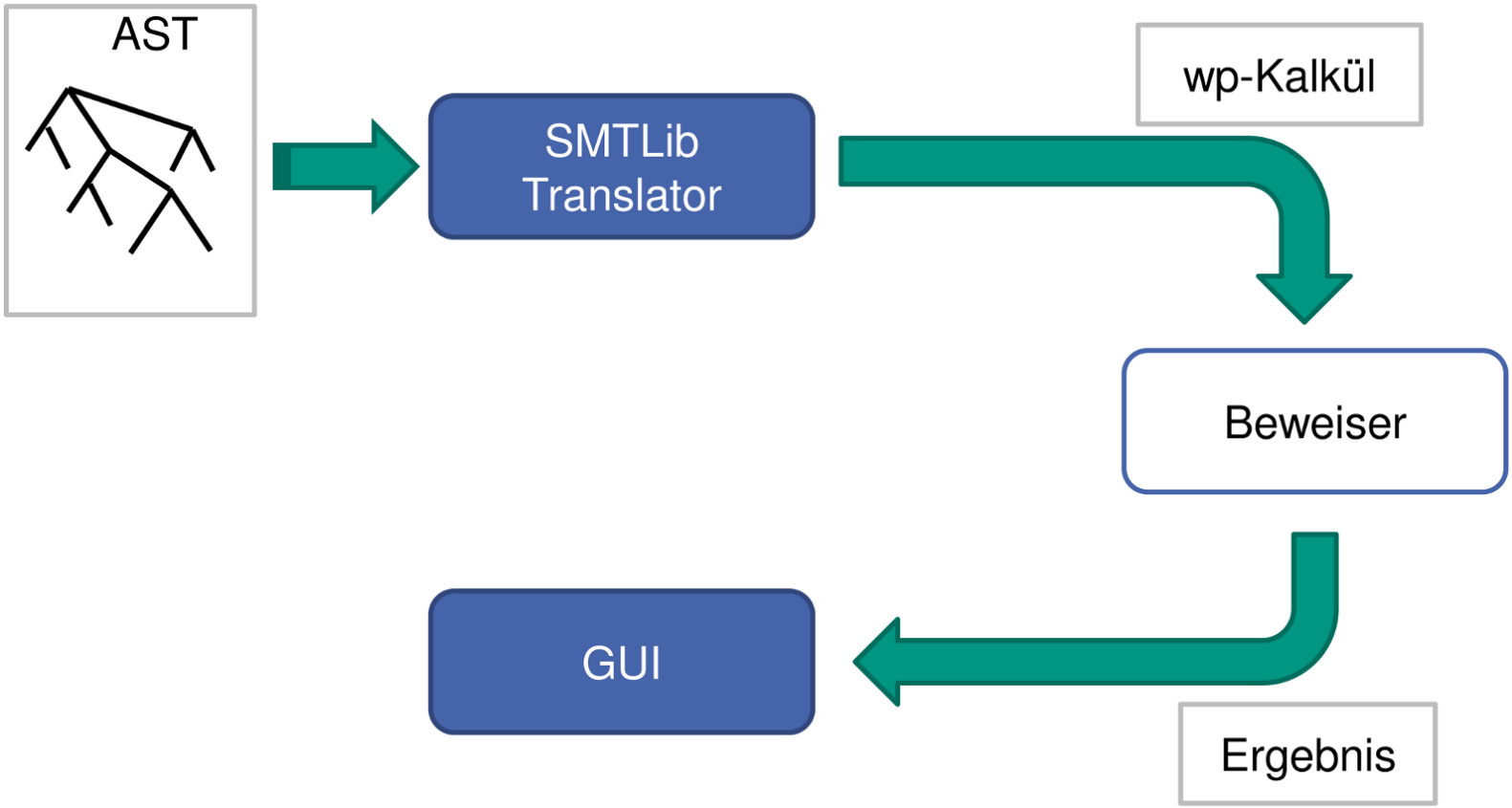
Institut für Theoretische Informatik

Inhalt

Aufgabenstellung: Parser und Interpreter



Aufgabenstellung: Formale Verifikation



Aufgabenstellung: Formale Verifikation

Karlsruher Institut für Technologie

Zur Verifikation eines Programmes sind zwei Schritte nötig:

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

Aufgabenstellung: Formale Verifikation

Karlsruher Institut für Technologie

Zur Verifikation eines Programmes sind zwei Schritte nötig:

1. Ziel des Programms klären
   - 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

Karlsruher Institut für Technologie

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

   1. `Lorem ipsum dolor sit amet...`

Aufgabenstellung: Formale Verifikation

Karlsruher Institut für Technologie

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

2. `Lorem ipsum dolor sit amet...`

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 W$_e$Prove

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

$\Rightarrow$ Tool ist ein guter Einstieg in den Themenbereich der formalen Verifikation (v.a. auch für Studenten)

$\Rightarrow$ Lizensierung unter BSD Lizenz

Tool Demonstration

Karlsruher Institut für Technologie

TODO: Bild

Soll / Haben

Qualitätssicherung

Karlsruher Institut für Technologie

Werkzeuge:

- GitHub Bugtracker
- JUnit
- GUI-Testplan

Probleme:

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

$\Rightarrow$ 108 gefundene Bugs
$\Rightarrow$ Code Coverage: ca. 90 %

Verteilung der Bugs auf die Module

Karlsruher Institut für Technologie

TODO: Graphik

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