Verifikation in der Realität

- In der Industrie wird der Begriff Verifikation häufig im Zusammenhang mit nicht formalen Methoden verwendet:
  - Testen, Strategien:
    - 100% Befehlsabdeckung (Statement Coverage)
    - 100% Zweigüberdeckung (Branch Coverage)
    - 100% Pfadüberdeckung (Path Coverage)
    - Siehe auch http://www.software-kompetenz.de/?10764
  - Code-Reviews
  - Verfolgbarkeitsanalysen

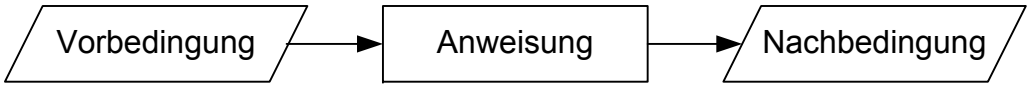
Testen

Mit Testen ist es möglich die Existenz von Fehlern nachzuweisen, nicht jedoch deren Abwesenheit.

- Testen ist von Natur aus unvollständig (non-exhaustive)
- Es werden nur ausgewählte Testfälle / Szenarien getestet, aber niemals alle möglichen.

Deduktive Methoden

- Nachweis der Korrektheit eines Programms durch math.-logisches Schließen
- Anfangsbelegung des Datenraums ⟶ Endbelegung
- Induktionsbeweise, Invarianten
  - klass. Bsp: Prädikatenkalkül von Floyd und Hoare, Betrachten von Einzelanweisungen eines Programms:



- Programmbeweise sind aufwändig, erfordern Experten
- i.A. nur kleine Programme verifizierbar
- Noch nicht vollautomatisch, aber es gibt schon leistungsfähige Werkzeuge

Temporale Logik

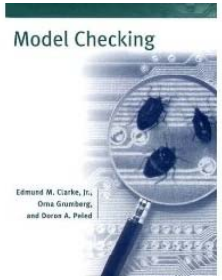
- Mittels Verifikation soll überprüft werden, dass:
  - Fehlerzustände nie erreicht werden
    - Der Aufzug soll nie mit offener Tür fahren.
  - ein System irgendwann einen bestimmten Zustand erreicht (und evtl. dort verbleibt)
    - Nach einer endlichen Initialisierungsphase, geht der Aufzug in den Betriebsmodus über.
  - Zustand x immer nach Eintreten des Zustandes y auftritt.
    - Nach Drücken des Tasters im Stockwerk wird der Aufzug in einem späteren Zustand auch dieses Stockwerk erreichen.
- Um solche Aussagen auch für Rechner lesbar auszudrücken, kann temporale Logik, z.B. in Form von LTL (linear time temporal logic), verwendet werden.
- In LTL werden Zustandsübergänge und damit auch die Zeit als diskrete Folge von Zuständen interpretiert.

Kripke-Struktur

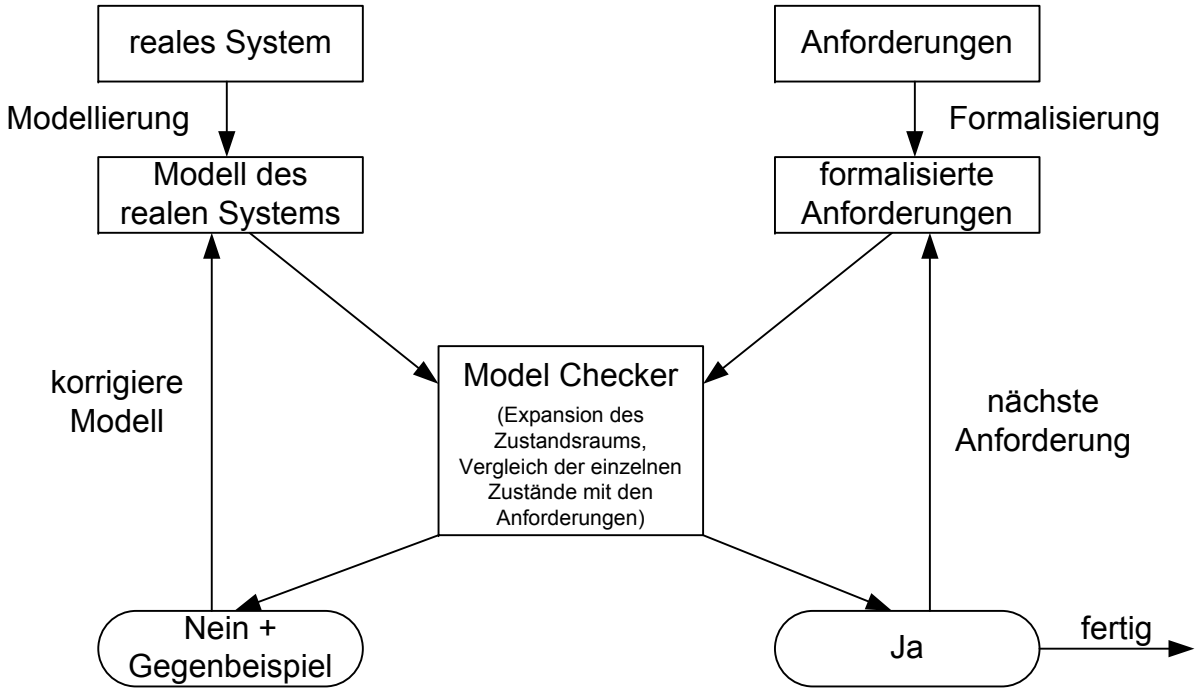
- Zur Darstellung eines Systems werden Kripke-Strukturen K = (V,I,R,B) und eine endliche Menge P von atomaren logischen Aussagen verwendet.
  - V: Menge binärer Variablen (z.B. Tür offen, Aufzug fährt)
  - Die Zustandsmenge S ergibt sich aus allen möglichen Kombinationen über V, somit gilt $S = 2^V$
  - Menge der möglichen Anfangszustände $I \sqsubseteq S$
  - R: Transitionsstruktur $R \sqsubseteq S \times S$
  - B: Bewertungsfunktion S x P $\rightarrow$ {true,false} zur Feststellung, ob ein Zustand eine Eigenschaft auf P erfüllt
- Mittels Model-Checking muss nun nachgewiesen werden, dass eine gewisse Eigenschaft P ausgehend von den Anfangszuständen
  - immer gilt
  - schließlich erfüllt wird
  - …

Explizites Model Checking: Verfahren

- Ausgehend von den Startzuständen exploriert der Model Checker mögliche Nachbarzustände:
  - Auswahl eines noch nicht evaluierten Zustandes
  - Prüfung aller möglichen Zustandsübergange:
    - bereits bekannter Zustand: verwerfen
    - unbekannter Zustand, Eigenschaft prüfen
      - falls Eigenschaft nicht erfüllt, Abbruch und Präsentation eines Gegenbeispiels
      - falls erfüllt, zur Menge der nicht evaluierten Zustände hinzufügen
  - Abbruchbedingung: alle erreichbaren Zustände wurden überprüft
- Problem: Zustandsexplosion
- Literaturhinweis: Edmund M. Clarke, Orna Grumberg, Doron A. Peled, *Model Checking*, 1999, MIT Press



Umgang mit Model Checking



Probleme mit formalen Methoden

- Entwickler empfinden formale Methoden häufig als zu kryptisch
- Beispiel TLA:

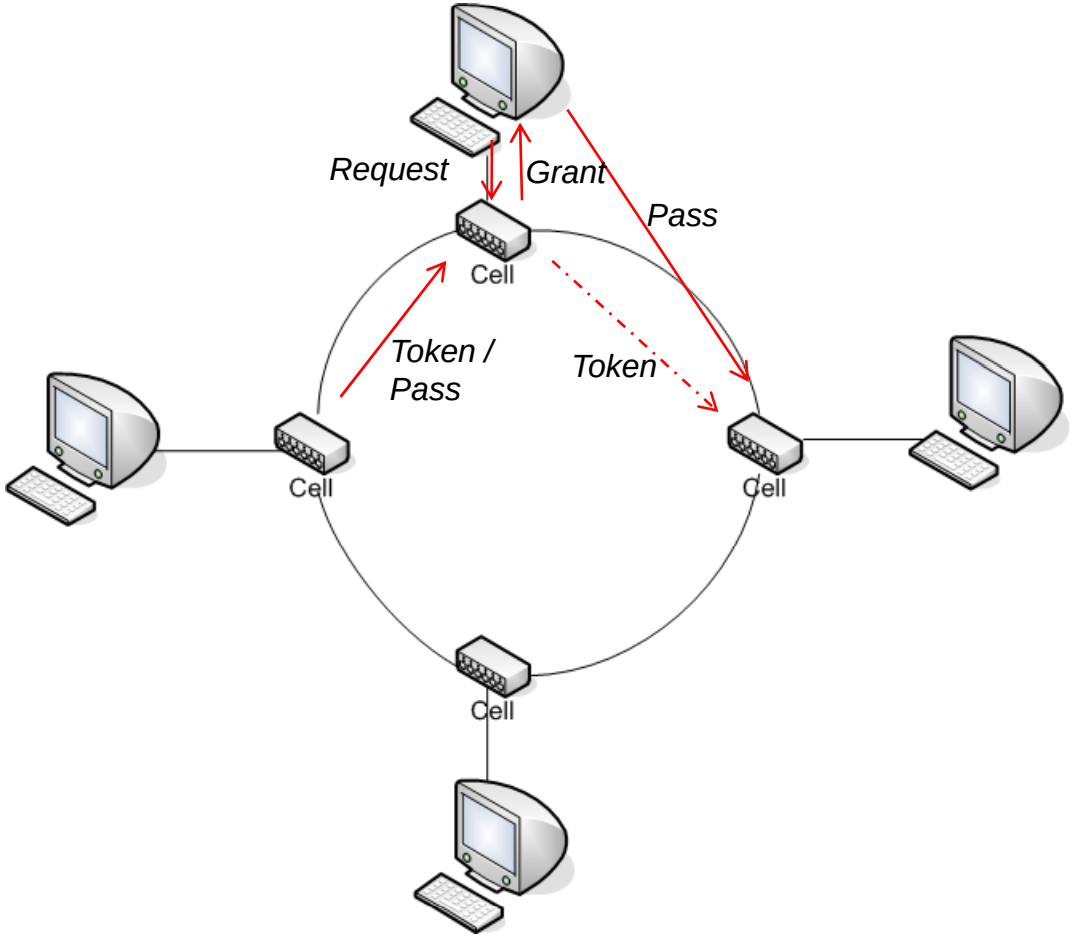
$$
\begin{aligned}
HCini \triangleq &\quad \wedge \quad hr \in \{0, \ldots, 23\} \\
HCnxt \triangleq &\quad \wedge \quad hr' = IF\ hr \neq 23\, THEN\ hr + 1\ ELSE\ 0 \\
HC \triangleq &\quad \wedge \quad HCini \\
&\quad \wedge \quad \Box HCnxt
\end{aligned}
$$

- Neue Ansätze: Erweiterung der Programmier / Modellierungssprachen, automatische Übersetzung

Beispiel: Verifikation von Esterel-Programmen

- Esterel-Verifier `xeve`
  - Einfach: Verifikation der sicheren (Nicht-)Auftretens von Signalen
  - Komplexe Bedingungen: Codierung als Esterel-Programm (Observer), dessen Verhalten mit dem des Ausgangsprogrammes geschnitten wird
- Grundsätzliche Vorgehensweise:
  - Finden von Fehlern in den Annahmen / Modellen mit begrenztem Model Checker
  - Nachweis der Korrektheit des verbesserten Modells in Bezug auf die korrigierten Eigenschaften mit unbegrenztem Model Checking / symbolischen Model Checking
- Esterel Studio
  - Eingebaute Verifikationsfunktionalität zur einfachen Verifikation von Programmen
  - Zur Modellierung der verschiedenen Eigenschaften kann das Schlüsselwort `assert` verwendet werden.
  - Im Verifikationsmodus können die Eigenschaften dann getestet werden, dabei stehen Methoden zum unbegrenzten / in der Testtiefe begrenzten Modell Checking, sowie zum symbolischen Model Checking zur Verfügung.
- Details siehe Demonstration: Bus-Arbiter (`xeve`)

Bus-Arbiter Beispiel



Verifikation ist immer nur auf Modellebene

```
int16_t number;
Signal e,end;
```

```
float share;
```

Prozess 1:

```
number:=1;
while (true)
{
    wait(e);
    number++;
}
```

Prozess 2:

```
while (true)
{
    wait(end);
    share:=1/number;
    print(share);
    number:=1;
}
```

Was kann hier schiefgehen (unabhängig von inhaltlicher Korrektheit)?
Problem Überlauf: Annahme Ereignisse zwischen zwei Endsignalen begrenzt (<1000)?
Problem: Atomare Erhöhung, korrekt auf 16-Bit Controller, falsch auf 8-Bit Controller.

Kapitel 3

Nebenläufigkeit