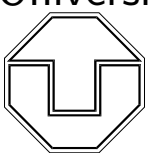
Nachweis der Verhaltensäquivalenz von Feldbus-Komponenten auf unterschiedlichen Abstraktionsebenen

Diplomarbeit

Martin Pitt
martin@piware.de

Technische Universität Dresden



11. November 2004

Stand des Großen Beleges

- Automatische Prüfung von Eigenschaften in Netzwerken, wenn formales CCMB-Modell vorliegt
- Modellerzeugung bei schon fertiger Implementierung
- Zustandsexplosion

Stand des Großen Beleges

- Automatische Prüfung von Eigenschaften in Netzwerken, wenn formales CCMB-Modell vorliegt
- Modellerzeugung bei schon fertiger Implementierung
- Zustandsexplosion

Stand des Großen Beleges

- Automatische Prüfung von Eigenschaften in Netzwerken, wenn formales CCMB-Modell vorliegt
- Modellerzeugung bei schon fertiger Implementierung
- Zustandsexplosion

Aufgabe des Diploms

- Automatische Modellerzeugung
  $\longrightarrow$ Schnittstelle und Verhalten aus C#-Implementierung
- Verhaltensäquivalenz zweier Modelle
  - abstrakte Komponenten mit weniger Zuständen
  - Weglassen von Sonderfällen (z. B. Sicherheitsabschaltung) $\longrightarrow$ bedingte Äquivalenz
  - weitere Möglichkeiten: Teilfunktionalität, qualifizierte Komponentensuche, ...

Aufgabe des Diploms

- Automatische Modellerzeugung
  $\longrightarrow$ Schnittstelle und Verhalten aus C#-Implementierung
- Verhaltensäquivalenz zweier Modelle
  - abstrakte Komponenten mit weniger Zuständen ▸ Automaten
  - Weglassen von Sonderfällen (z. B. Sicherheitsabschaltung) $\longrightarrow$ bedingte Äquivalenz ▸ Automaten
  - weitere Möglichkeiten: Teilfunktionalität, qualifizierte Komponentensuche, . . .

Aufgabe des Diploms

- Automatische Modellerzeugung
  $\longrightarrow$ Schnittstelle und Verhalten aus C#-Implementierung
- Verhaltensäquivalenz zweier Modelle
  - abstrakte Komponenten mit weniger Zuständen ▸ Automaten
  - Weglassen von Sonderfällen (z. B. Sicherheitsabschaltung) $\longrightarrow$ bedingte Äquivalenz ▸ Automaten
  - weitere Möglichkeiten: Teilfunktionalität, qualifizierte Komponentensuche, . . .

Aufgabe des Diploms

- Automatische Modellerzeugung
  $\longrightarrow$ Schnittstelle und Verhalten aus C#-Implementierung
- Verhaltensäquivalenz zweier Modelle
  - abstrakte Komponenten mit weniger Zuständen ▸ Automaten
  - Weglassen von Sonderfällen (z. B. Sicherheitsabschaltung) $\longrightarrow$ bedingte Äquivalenz ▸ Automaten
  - weitere Möglichkeiten: Teilfunktionalität, qualifizierte Komponentensuche, ...

Übersetzung durch `cs2ccmb`

- Parsen des Quellcodes
  → Anleihe des Scanners und der Grammatik aus Mono
  → Repräsentation und Berechnung in Klassenstruktur
- für alle möglichen Kombinationen erlaubter Eingaben:
  → Interpretation der Komponenten-Klasse
  → Ermittlung Endzustand
- erschöpfende Berechnung aller tatsächlich auftretenden Zustände
- Prozessmodell (endlicher Automat) in CCMB-Notation

▸ Algorithmus

Übersetzung durch `cs2ccmb`

- Parsen des Quellcodes
  → Anleihe des Scanners und der Grammatik aus Mono
  → Repräsentation und Berechnung in Klassenstruktur
- für alle möglichen Kombinationen erlaubter Eingaben:
  → Interpretation der Komponenten-Klasse
  → Ermittlung Endzustand
- erschöpfende Berechnung aller tatsächlich auftretenden Zustände
- Prozessmodell (endlicher Automat) in CCMB-Notation

▸ Algorithmus

Übersetzung durch `cs2ccmb`

- Parsen des Quellcodes
  → Anleihe des Scanners und der Grammatik aus Mono
  → Repräsentation und Berechnung in Klassenstruktur
- für alle möglichen Kombinationen erlaubter Eingaben:
  → Interpretation der Komponenten-Klasse
  → Ermittlung Endzustand
- erschöpfende Berechnung aller tatsächlich auftretenden Zustände
- Prozessmodell (endlicher Automat) in CCMB-Notation

▸ Algorithmus

Übersetzung durch `cs2ccmb`

- Parsen des Quellcodes
  → Anleihe des Scanners und der Grammatik aus Mono
  → Repräsentation und Berechnung in Klassenstruktur
- für alle möglichen Kombinationen erlaubter Eingaben:
  → Interpretation der Komponenten-Klasse
  → Ermittlung Endzustand
- erschöpfende Berechnung aller tatsächlich auftretenden Zustände
- Prozessmodell (endlicher Automat) in CCMB-Notation

▸ Algorithmus

Was ist Verhaltensäquivalenz?

Test zweier Modelle auf **Bisimularität**

Zwei Zustände sind bisimular, wenn in ihnen die gleichen Transitionen möglich sind und diese zu jeweils bisimularen Folgezuständen führen. ▸ Exakte Definition

→ Verhaltens**gleichheit**

Erweiterung um vier Konzepte zur Verhaltens**äquivalenz**:

- Zustandsabstraktion
- Transitionsabstraktion
- Zustandsfilter
- einfache Simulation (Teilfunktionalität)

Implementierung: `sim`

Was ist Verhaltensäquivalenz?

Test zweier Modelle auf **Bisimularität**

Zwei Zustände sind bisimular, wenn in ihnen die gleichen Transitionen möglich sind und diese zu jeweils bisimularen Folgezuständen führen. ▸ Exakte Definition

→ Verhaltens**gleichheit**

Erweiterung um vier Konzepte zur Verhaltens**äquivalenz**:

- Zustandsabstraktion
- Transitionsabstraktion
- Zustandsfilter
- einfache Simulation (Teilfunktionalität)

Implementierung: `sim`

Was ist Verhaltensäquivalenz?

Test zweier Modelle auf **Bisimularität**

Zwei Zustände sind bisimular, wenn in ihnen die gleichen Transitionen möglich sind und diese zu jeweils bisimularen Folgezuständen führen. ▸ Exakte Definition

→ Verhaltens**gleichheit**

Erweiterung um vier Konzepte zur Verhaltens**äquivalenz**:

- Zustandsabstraktion
- Transitionsabstraktion
- Zustandsfilter
- einfache Simulation (Teilfunktionalität)

Implementierung: `sim`

Was ist Verhaltensäquivalenz?

Test zweier Modelle auf **Bisimularität**

Zwei Zustände sind bisimular, wenn in ihnen die gleichen Transitionen möglich sind und diese zu jeweils bisimularen Folgezuständen führen. ▸ Exakte Definition

→ Verhaltens**gleichheit**

Erweiterung um vier Konzepte zur Verhaltens**äquivalenz**:

- Zustandsabstraktion
- Transitionsabstraktion
- Zustandsfilter
- einfache Simulation (Teilfunktionalität)

Implementierung: `sim`

Was ist Verhaltensäquivalenz?

Test zweier Modelle auf **Bisimularität**

Zwei Zustände sind bisimular, wenn in ihnen die gleichen Transitionen möglich sind und diese zu jeweils bisimularen Folgezuständen führen. ▸ Exakte Definition

→ Verhaltens**gleichheit**

Erweiterung um vier Konzepte zur Verhaltens**äquivalenz**:

- Zustandsabstraktion
- Transitionsabstraktion
- Zustandsfilter
- einfache Simulation (Teilfunktionalität)

Implementierung: `sim`

Was ist Verhaltensäquivalenz?

Test zweier Modelle auf **Bisimularität**

Zwei Zustände sind bisimular, wenn in ihnen die gleichen Transitionen möglich sind und diese zu jeweils bisimularen Folgezuständen führen. ▸ Exakte Definition

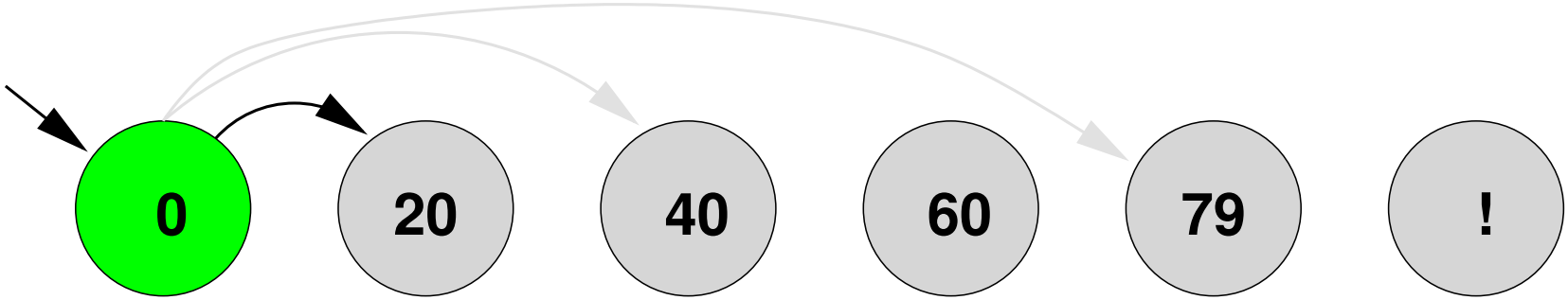
→ Verhaltens**gleichheit**

Erweiterung um vier Konzepte zur Verhaltens**äquivalenz**:

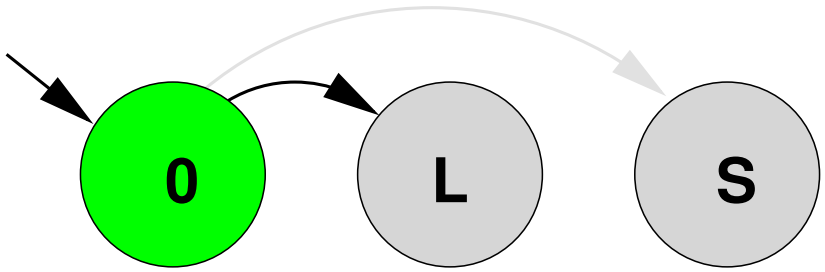
- Zustandsabstraktion
- Transitionsabstraktion
- Zustandsfilter
- einfache Simulation (Teilfunktionalität)

Implementierung: `sim`

Demonstration Verhaltensäquivalenz



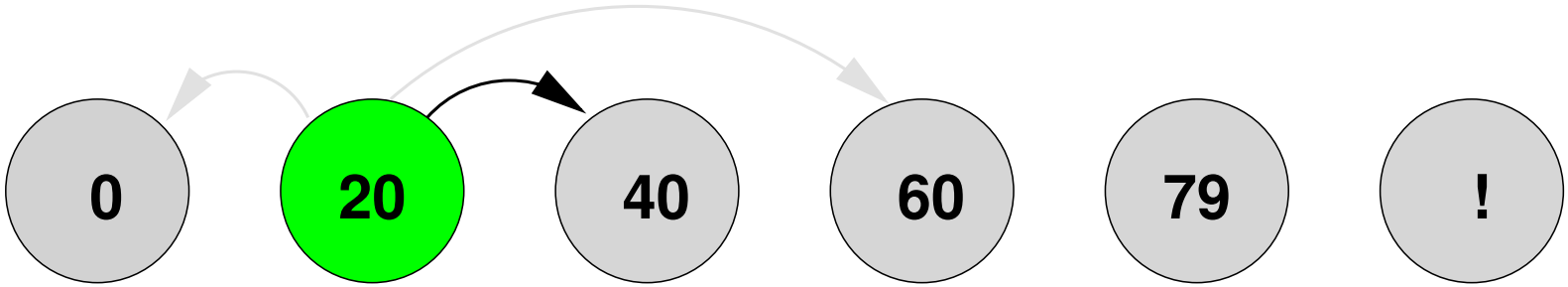
setSpeed(20)



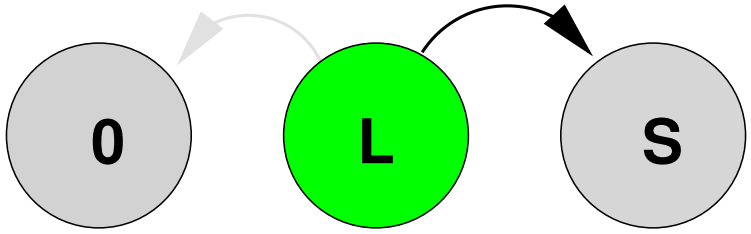
Transitionen:
setSpeed(20)
setSpeed(40)
...
setSpeed(80)

Zustandsabstraktion:
0 = 0
L = (0,20]
S = (20,oo)

Demonstration Verhaltensäquivalenz



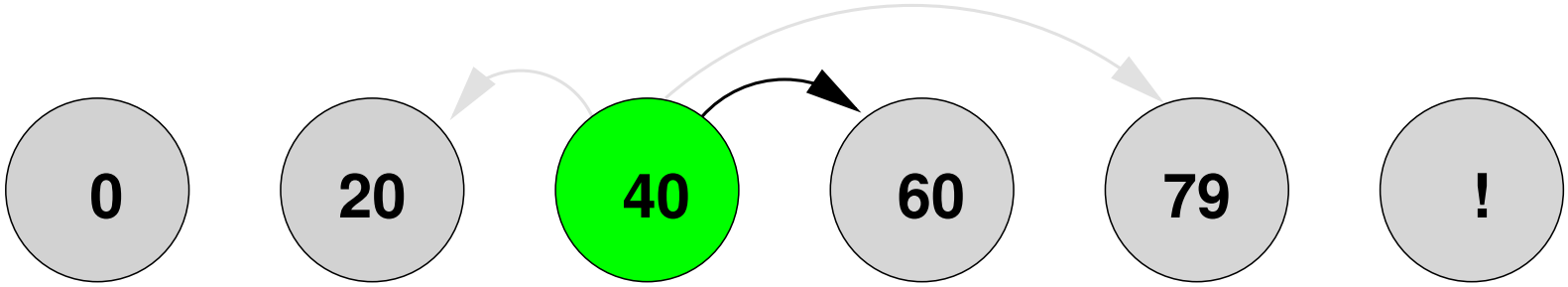
setSpeed(40)



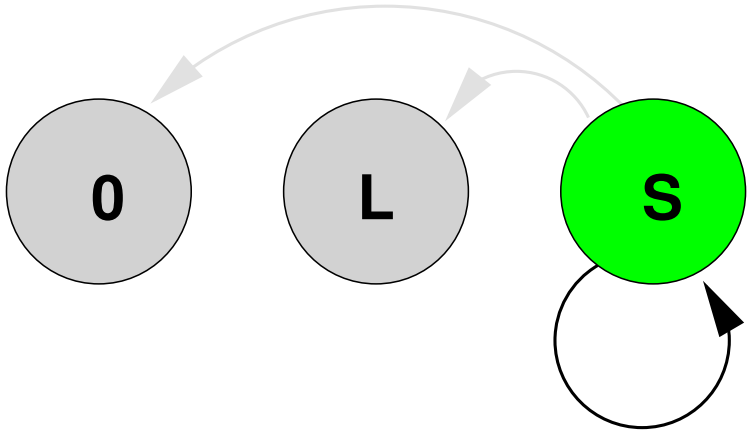
Transitionen:
setSpeed(20)
setSpeed(40)
...
setSpeed(80)

Zustandsabstraktion:
0 = 0
L = (0,20]
S = (20,oo)

Demonstration Verhaltensäquivalenz



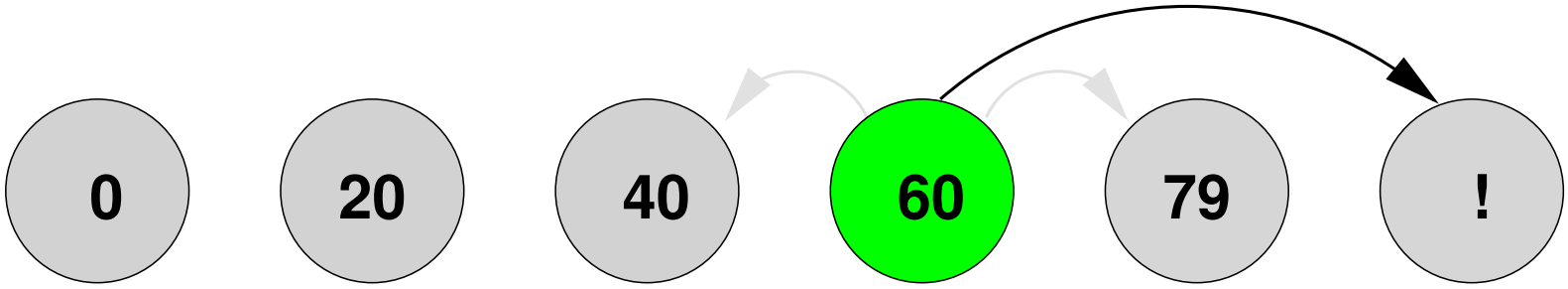
setSpeed(60)



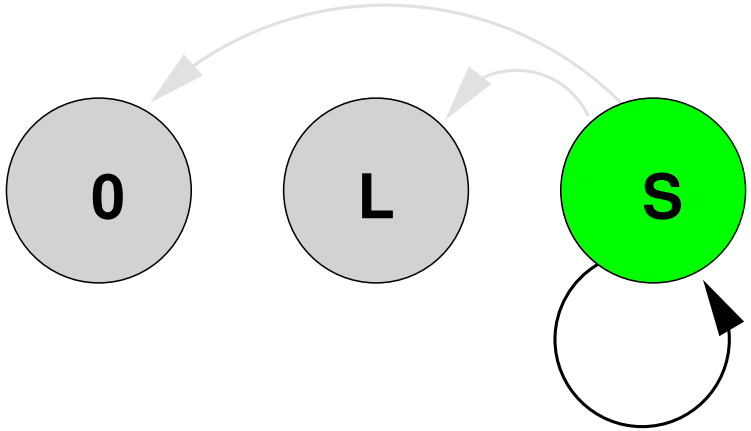
Transitionen:
setSpeed(20)
setSpeed(40)
...
setSpeed(80)

Zustandsabstraktion:
0 = 0
L = (0,20]
S = (20,oo)

Demonstration Verhaltensäquivalenz



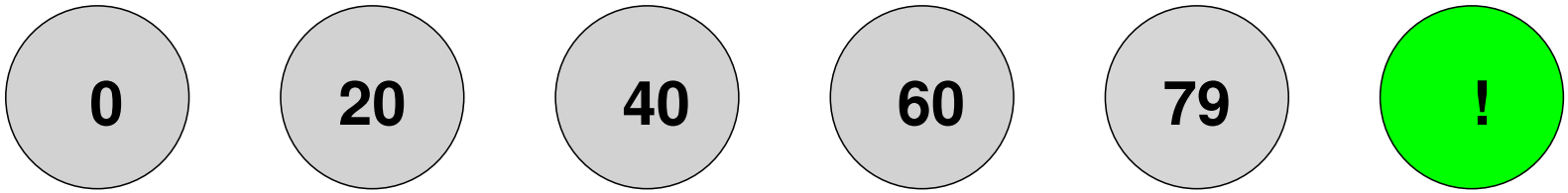
setSpeed(80)



Transitionen:
setSpeed(20)
setSpeed(40)
...
setSpeed(80)

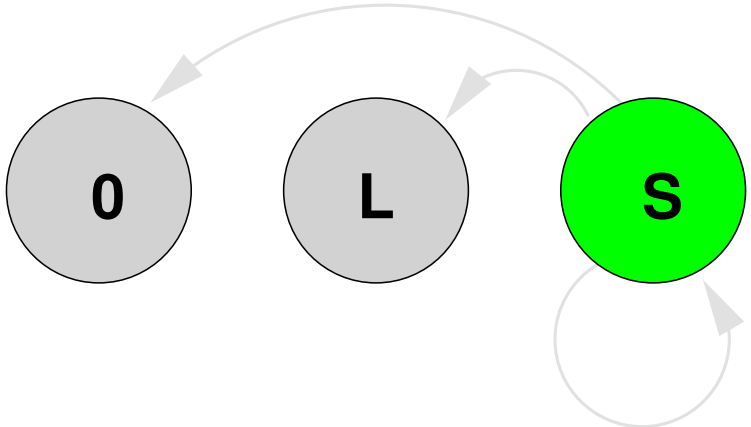
Zustandsabstraktion:
0 = 0
L = (0,20]
S = (20,oo)

Demonstration Verhaltensäquivalenz



S ≁ ! aber Filter: v < 80

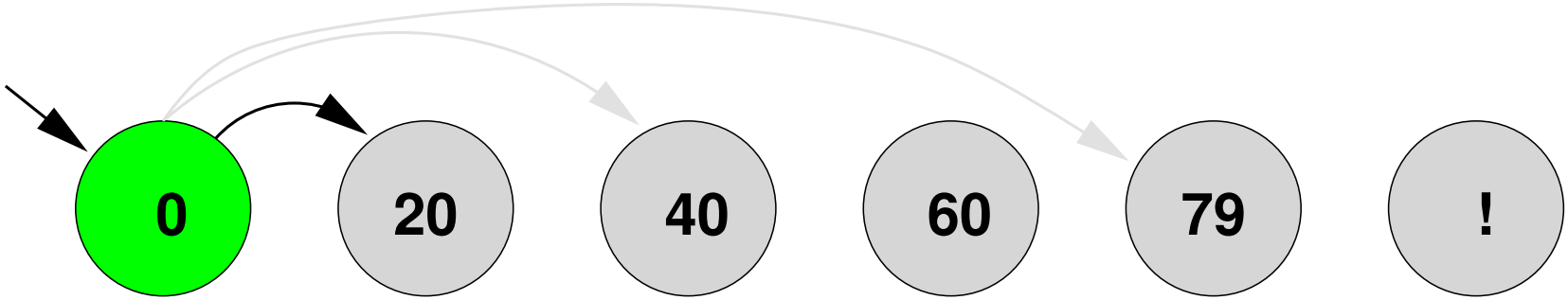
bedingte Verhaltensäquivalenz



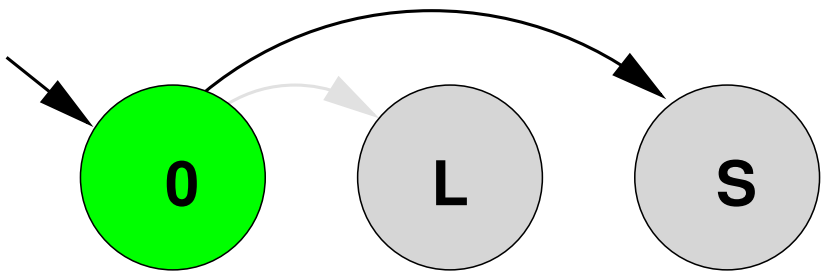
Transitionen:
setSpeed(20)
setSpeed(40)
...
setSpeed(80)

Zustandsabstraktion:
0 = 0
L = (0,20]
S = (20,oo)

Demonstration Gegenbeispiel Verhaltensäquivalenz



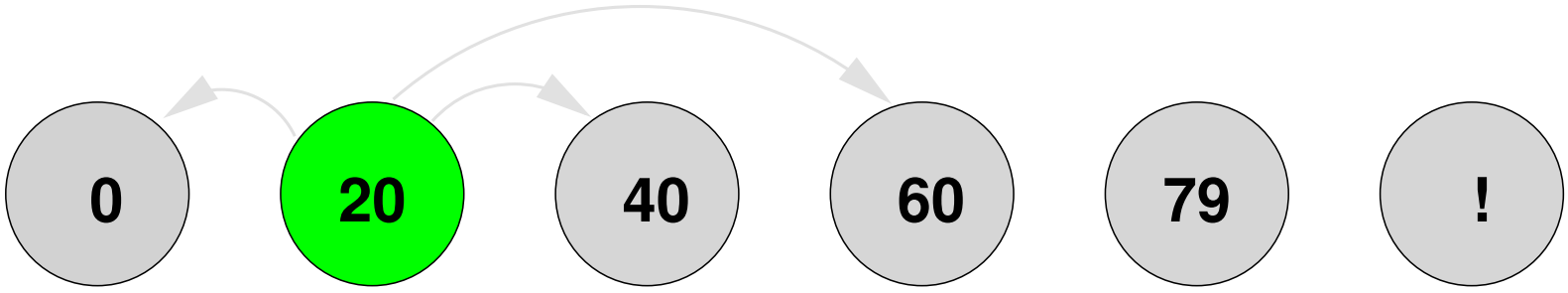
setSpeed(20)



Transitionen:
setSpeed(20)
setSpeed(40)
...
setSpeed(80)

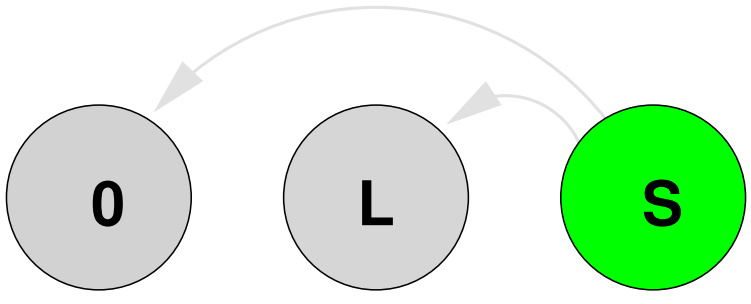
Zustandsabstraktion:
0 = 0
L = (0,20]
S = (20,oo)

Demonstration Gegenbeispiel Verhaltensäquivalenz



S ≁ 20 und erfüllt v < 80

⇒ nicht bisimular



Transitionen:
setSpeed(20)
setSpeed(40)
...
setSpeed(80)

Zustandsabstraktion:
0 = 0
L = (0,20]
S = (20,oo)

Programmaufruf

```
cs2ccmb.exe -bin SpeedSafe.bin VFS.cs SpeedSafe.cs
cs2ccmb.exe -bin SpeedAbst.bin VFS.cs SpeedAbst.cs
```

```
sim.exe -b SpeedAbst.bin SpeedSafe.bin
```

$\longrightarrow$ nicht verhaltensäquivalent

```
sim.exe -b SpeedAbst.bin SpeedSafe.bin 'speed<80.0'
```

$\longrightarrow$ äquivalent

Programmaufruf

```
cs2ccmb.exe -bin SpeedSafe.bin VFS.cs SpeedSafe.cs
cs2ccmb.exe -bin SpeedAbst.bin VFS.cs SpeedAbst.cs
```

```
sim.exe -b SpeedAbst.bin SpeedSafe.bin
```

$\longrightarrow$ nicht verhaltensäquivalent

```
sim.exe -b SpeedAbst.bin SpeedSafe.bin 'speed<80.0'
```

$\longrightarrow$ äquivalent

Programmaufruf

```
cs2ccmb.exe -bin SpeedSafe.bin VFS.cs SpeedSafe.cs
cs2ccmb.exe -bin SpeedAbst.bin VFS.cs SpeedAbst.cs

sim.exe -b SpeedAbst.bin SpeedSafe.bin
```

$\longrightarrow$ nicht verhaltensäquivalent

```
sim.exe -b SpeedAbst.bin SpeedSafe.bin 'speed<80.0'
```

$\longrightarrow$ äquivalent

Modellgröße

- vorgestelltes Beispiel: sehr einfach, nur wenige Zustände
- größtes berechnetes Beispiel: `Courtesy`
  - Implementierung: 3 Mio. Zustände
  - Abstraktion (ohne Sicherheitsabschaltung): 1 Mio. Zustände
  - 60% der Berechnung in 20 Minuten; Rest: mehrere Stunden
- Speicher ist begrenzende Größe

▸ Benchmarks

Modellgröße

- vorgestelltes Beispiel: sehr einfach, nur wenige Zustände
- größtes berechnetes Beispiel: `Courtesy`
  - Implementierung: 3 Mio. Zustände
  - Abstraktion (ohne Sicherheitsabschaltung): 1 Mio. Zustände
  - 60% der Berechnung in 20 Minuten; Rest: mehrere Stunden
- Speicher ist begrenzende Größe

▸ Benchmarks

Modellgröße

- vorgestelltes Beispiel: sehr einfach, nur wenige Zustände
- größtes berechnetes Beispiel: `Courtesy`
  - Implementierung: 3 Mio. Zustände
  - Abstraktion (ohne Sicherheitsabschaltung): 1 Mio. Zustände
  - 60% der Berechnung in 20 Minuten; Rest: mehrere Stunden
- Speicher ist begrenzende Größe

▸ Benchmarks

Noch zu lösende Probleme

- effizientere Speicherung des Modells
  → Zusammenfassung in OBDDs?
- Vervollständigung `cs2ccmb`
  → Propertys, Destruktoren, Indexer, Operatoren, Enums, ...
- Abstraktion von Teilsystemen
  → Semantik in Beleg
  → Transitionsexplosion durch Interleaving

Noch zu lösende Probleme

- effizientere Speicherung des Modells
  → Zusammenfassung in OBDDs?
- Vervollständigung `cs2ccmb`
  → Propertys, Destruktoren, Indexer, Operatoren, Enums, ...
- Abstraktion von Teilsystemen
  → Semantik in Beleg
  → Transitionsexplosion durch Interleaving

Noch zu lösende Probleme

- effizientere Speicherung des Modells
  → Zusammenfassung in OBDDs?
- Vervollständigung `cs2ccmb`
  → Propertys, Destruktoren, Indexer, Operatoren, Enums, ...
- Abstraktion von Teilsystemen
  → Semantik in Beleg
  → Transitionsexplosion durch Interleaving

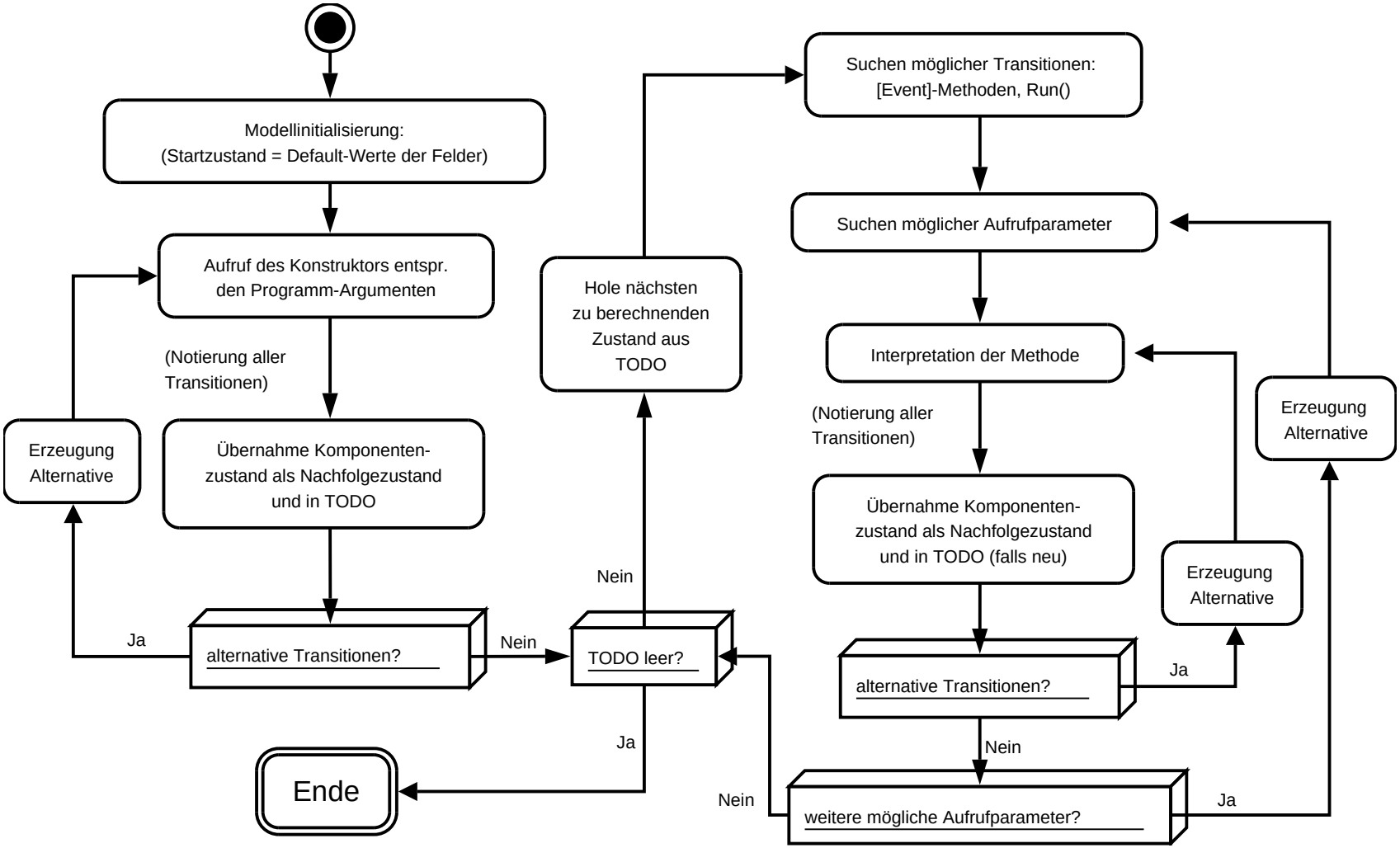
Vielen Dank für Ihre Aufmerksamkeit!

Download: `http://piware.de/docs.shtml`

email-Kontakt: `martin@piware.de`

Jetzt ist Zeit für Fragen und Diskussion.

Algorithmus Modellgenerierung



$\mathcal{F}_1 = (S_1, s_1^0, next_1 \subseteq S_1 \times \Delta \times S_1)$, $\mathcal{F}_2 = (S_2, s_2^0, next_2 \subseteq S_2 \times \Delta \times S_2)$: KRIPKE-Strukturen über Transitionsmenge Δ

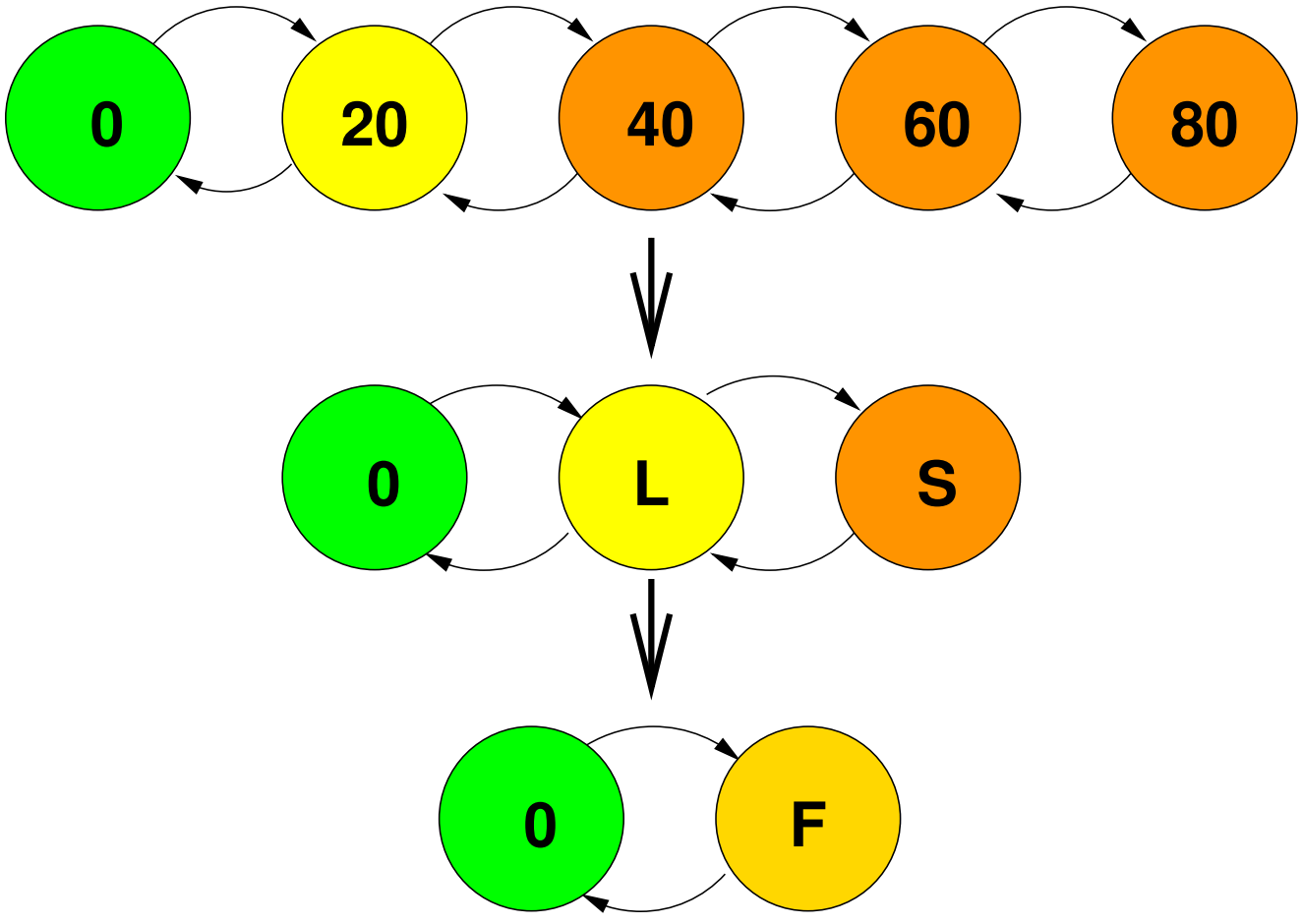
Relation $R \subseteq S_1 \times S_2$ ist eine **Bisimulation**, gdw.

$\forall \delta \in \Delta; s_1, s_1' \in S_1; s_2, s_2' \in S_2.$

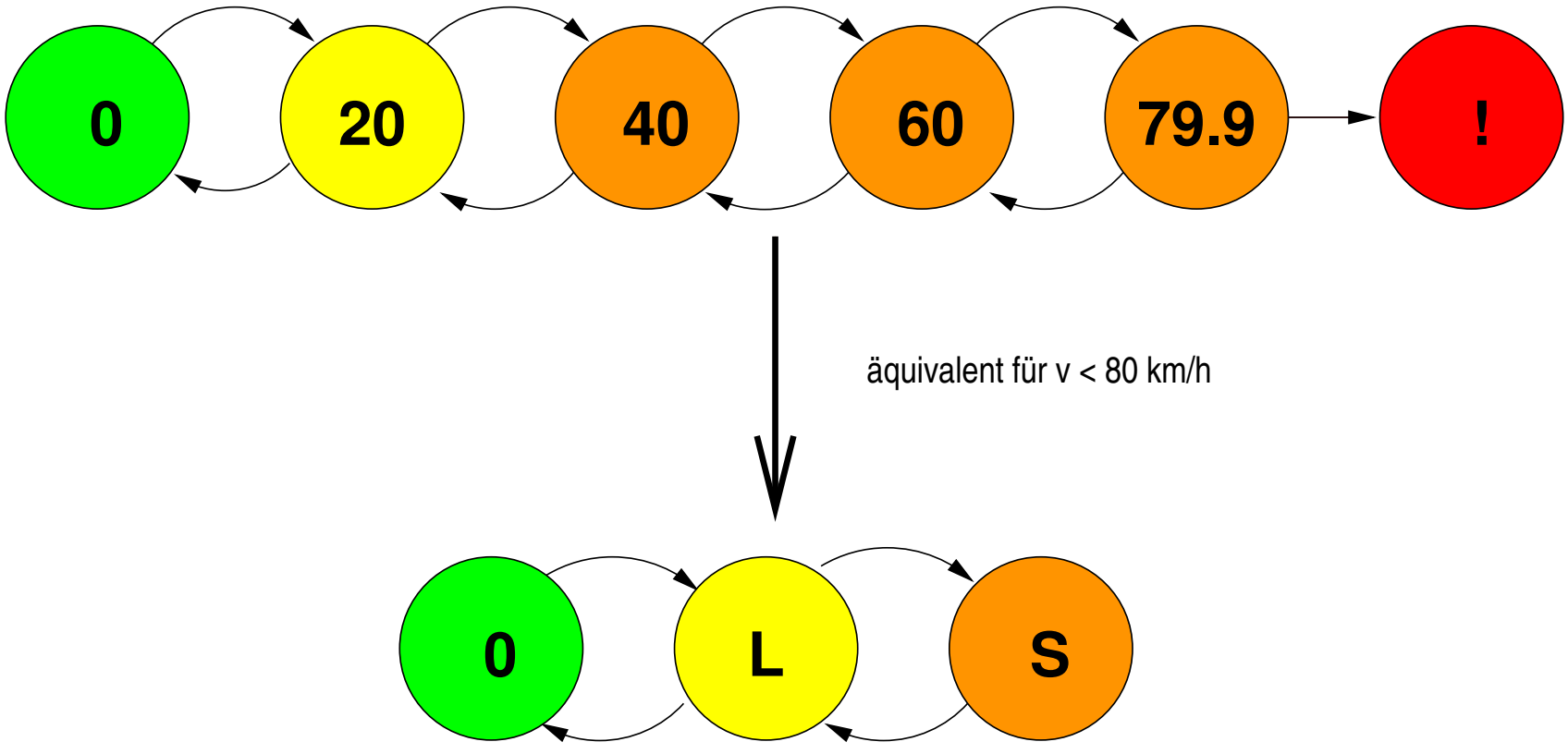
$$(s_1, \delta, s_1') \in next_1 \ \wedge s_1 R s_2 \quad \Rightarrow \quad \exists s_2'.\ (s_2, \delta, s_2') \in next_2 \ \wedge \ s_1' R s_2'$$
$$\wedge \ (s_2, \delta, s_2') \in next_2 \ \wedge s_1 R s_2 \quad \Rightarrow \quad \exists s_1'.\ (s_1, \delta, s_1') \in next_1 \ \wedge \ s_1' R s_2'$$

Zwei Zustände s_1, s_2 sind **bisimular** (notiert als $s_1 \sim s_2$), wenn es eine Bisimulation R gibt mit $s_1 R s_2$. Zwei KRIPKE-Strukturen $\mathcal{F}_1$ und $\mathcal{F}_2$ sind bisimular ($\mathcal{F}_1 \sim \mathcal{F}_2$), wenn ihre Startzustände bisimular sind, d. h. $s_1^0 \sim s_2^0$.

Beispiel: Geschwindigkeitsregler



Beispiel: Geschwindigkeitsregler mit Sicherheitsabschaltung



Komponentenklasse SpeedSafe

```
1  public class SpeedSafe : Comp
2  {
3      public VFS OUT = new VFS(1);
4
5      float speed;
6
7      public SpeedSafe() {
8          OUT.State = 0;
9      }
10
11     [Event] public void setSpeed( [Range(0.0,80.0,20.0)] float speed) {
12         this.speed = speed;
13         EventOccured();
14     }
15
16     protected override void Run()
17     {
18         if( OUT.StateFb == 0 ) {
19             if( speed < 80.0 )
20                 OUT.Cont = speed;
21             else {
22                 OUT.Cont = 0.0;
23                 OUT.State = 1;
24             }
25         }
26     }
27 }
```

CCMB-Modell von SpeedSafe

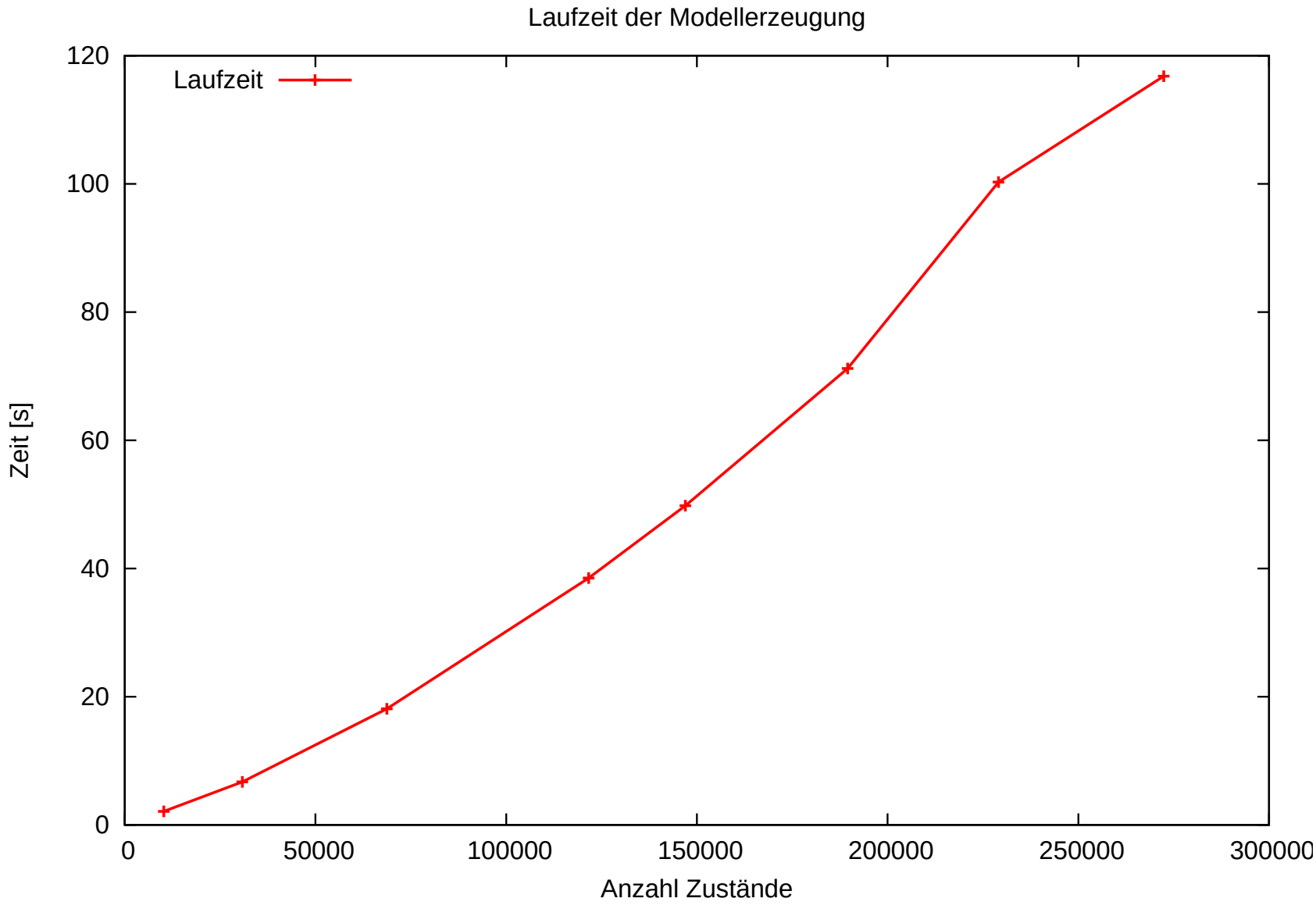
```
SpeedSafe = OUT.State:=0 -> (
              OUT.StateFb=0 -> OUT.Cont:=0 -> [0,"VFS"]
            | OUT.StateFb=1 -> [0,"VFS"])

[0,"VFS"] = (
          setSpeed(0) -> (
              OUT.StateFb=0 -> OUT.Cont:=0 -> [0,"VFS"]
            | OUT.StateFb=1 -> [0,"VFS"])
        | setSpeed(20) -> (
              OUT.StateFb=0 -> OUT.Cont:=20 -> [20,"VFS"]
            | OUT.StateFb=1 -> [20,"VFS"])
        | [...]
        | OUT.StateFb=0 -> OUT.Cont:=0 -> [0,"VFS"]
        | OUT.StateFb=1 -> [0,"VFS"])

[...]
```

Abstraktion von SpeedSafe

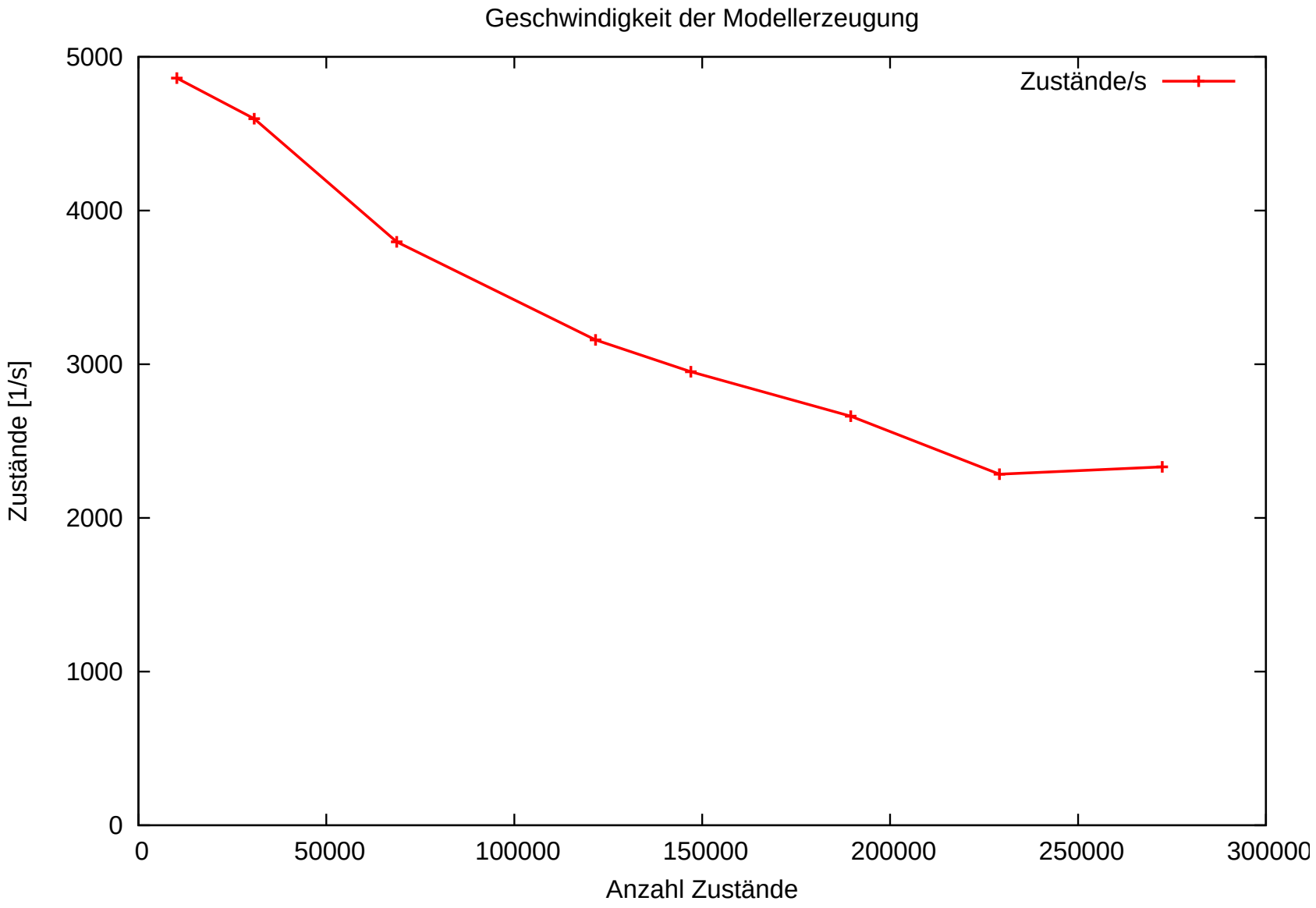
```
1  public class SpeedAbst : Comp
2  {
3      [Property("Cont"), Eq( 0.0, 0.0 ), Eq( 10.0, "(0.0,oo)" )]
4      public VFS OUT = new VFS(1);
5
6      [Abstracts("speed"), Eq( false , 0.0 ), Eq( true , "(0.0,oo)" )]
7      bool go;
8
9      public SpeedAbst() {
10         OUT.State = 0;
11     }
12
13     [Event] public void setSpeed( [Range(0.0,80.0,20.0)] float speed) {
14         this.go = ( speed > 0.0 );
15         EventOccured();
16     }
17
18     protected override void Run()
19     {
20         if ( go )
21             OUT.Cont = 10.0;
22         else
23             OUT.Cont = 0.0;
24     }
25 }
```

CCMB-Modell von SpeedAbst

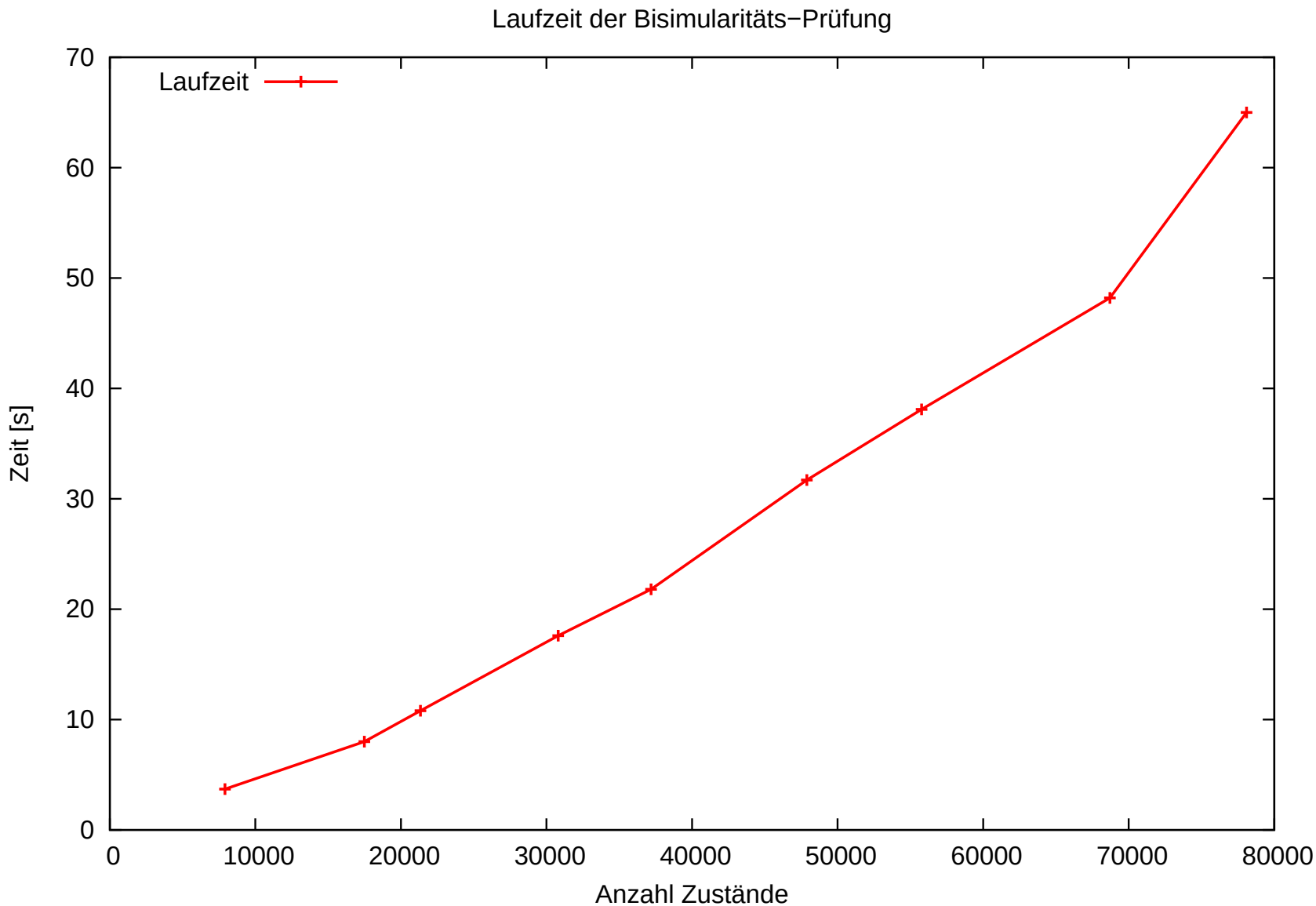
```
SpeedAbst = OUT.State:=0 -> OUT.Cont:=0 -> [False,"VFS"]

[True,"VFS"] = (
          setSpeed(0) -> OUT.Cont:=0 -> [False,"VFS"]
        | setSpeed(20) -> OUT.Cont:=10 -> [True,"VFS"]
        | setSpeed(40) -> OUT.Cont:=10 -> [True,"VFS"]
        | setSpeed(60) -> OUT.Cont:=10 -> [True,"VFS"]
        | setSpeed(80) -> OUT.Cont:=10 -> [True,"VFS"]
        | OUT.Cont:=10 -> [True,"VFS"])

[False,"VFS"] = (
          setSpeed(0) -> OUT.Cont:=0 -> [False,"VFS"]
        | setSpeed(20) -> OUT.Cont:=10 -> [True,"VFS"]
        | setSpeed(40) -> OUT.Cont:=10 -> [True,"VFS"]
        | setSpeed(60) -> OUT.Cont:=10 -> [True,"VFS"]
        | setSpeed(80) -> OUT.Cont:=10 -> [True,"VFS"]
        | OUT.Cont:=0 -> [False,"VFS"])
```



Zustandsraum: Z, Laufzeit: $\mathcal{O}(|Z|)$


Geschwindigkeit der Modellerzeugung
Zustände/s
Zustände [1/s]
Anzahl Zustände
0
1000
2000
3000
4000
5000
0
50000
100000
150000
200000
250000
300000

Benchmarks Bisimulations-Prüfung



Laufzeit: $\mathcal{O}(|Z_1| + |Z_2|)$

Benchmarks Bisimulations-Prüfung