



```

X Isabelle/Isar Proof
File Edit View Cmds Tools Options Buffers Proof-General Isabelle/Isar X-Symbol Help

State Proof Back Buffer Next Fix Fold Unfold Stop Restart Help

Achszaehler18.thy Achszaehler17.thy
      S0          : Unbeeinflusst zustand
      S1_war_0    : System 1 beeinflusst - war unbeeinflusst
      S1_war_12   : System 1 beeinflusst - war zweikanalig beeinflusst
      S12         : Zweikanalig beeinflusst
      S2_war_0    : System 2 beeinflusst - war unbeeinflusst
      S2_war_12   : System 2 beeinflusst - war zweikanalig beeinflusst*)
datatype zustand = S0 | S1_war_0 | S1_war_12 | S12 | S2_war_0 | S2_war_12

types zeit_zustand = "zustand*nat"
types uebergang = "zustand*zustand"
constdefs uebergangs :: "uebergang set"
"uebergangs =
  ((S0, S1_war_0), (S0, S2_war_0), (S0, S12),
   (S1_war_0, S12), (S1_war_0, S0), (S1_war_0, S2_war_0),
   (S1_war_12, S12), (S1_war_12, S0), (S1_war_12, S2_war_12),
   (S12, S2_war_12), (S12, S1_war_12), (S12, S0),
   (S2_war_12, S0), (S2_war_12, S12), (S2_war_12, S1_war_12),
   (S2_war_0, S0), (S2_war_0, S12), (S2_war_0, S1_war_0))"

text /* An automaton is a triple of a
Raw: T--*-XEmacs: Achszaehler18.thy (Isar script XS:isabelle/s Font)----1%---
Raw--*-XEmacs: *Warnings* (Fundamental)----Bot-----
  
```

# Theorem Proving mit Isabelle

Olivera Pavlović

# Überblick

- I. **Formale Verifikation**
- II. Achszählsystem
- III. Theorem Prover Isabelle
- IV. Darstellung des Achszählsystems in Isabelle
- V. Abschluss



# Formale Verifikation

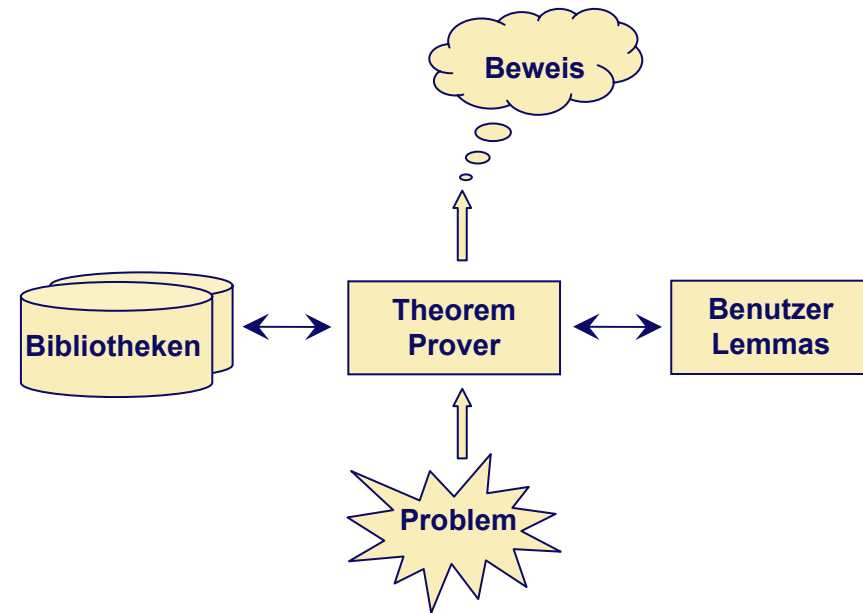
⇒ mathematisch-basiertes Verfahren zur Prüfung der Richtigkeit eines Systems

- Methoden
  - SAT-Solving
  - Model Checking
  - Theorem Proving

# Formale Verifikation – Theorem Proving

⇒ Verifikation formaler Spezifikationen eines Models

- präzise Beschreibung des Problems in einer Logik
- passende Formulierung und gutes Verstehen des Problems
- Systemsversuch das Problem zu lösen mit der Hilfe der existierenden Bibliotheken
- wenn erfolgreich → Beweis
- wenn nicht erfolgreich → interaktive Beweisführung
- Prozessiteration



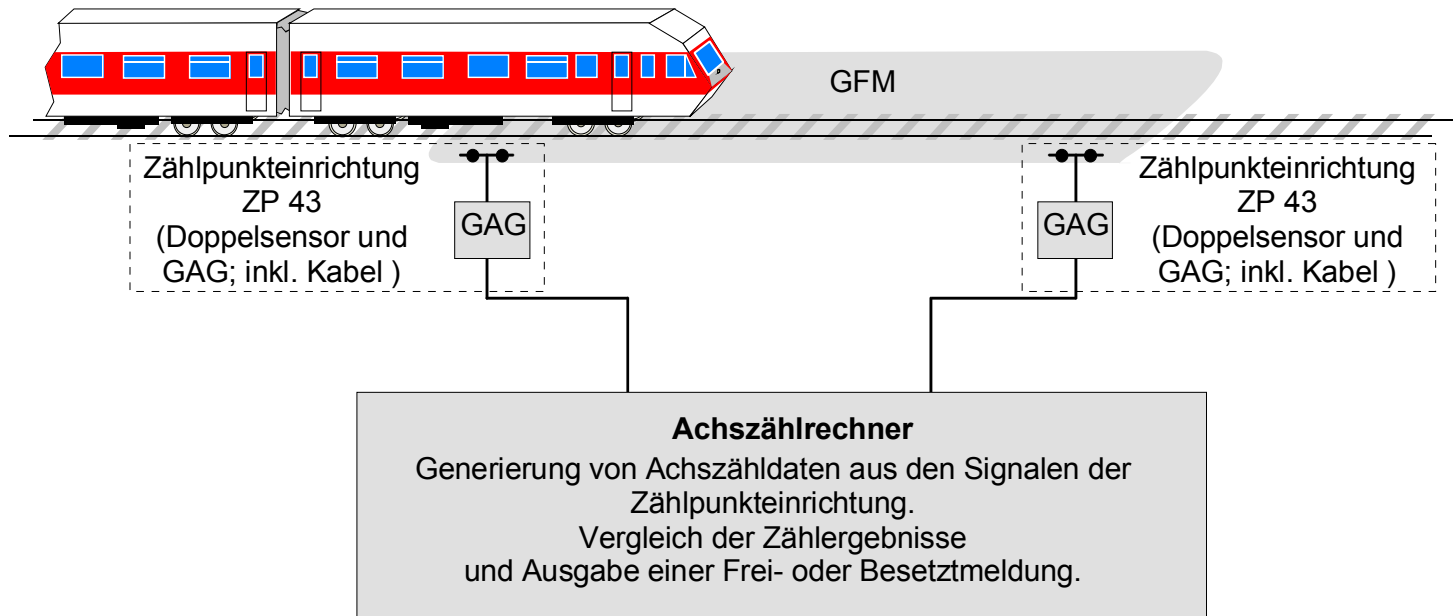


# Überblick

- I. Formale Verifikation
- II. Achszählsystem**
- III. Theorem Prover Isabelle
- IV. Darstellung des Achszählsystems in Isabelle
- V. Abschluss

# Achszählsystem

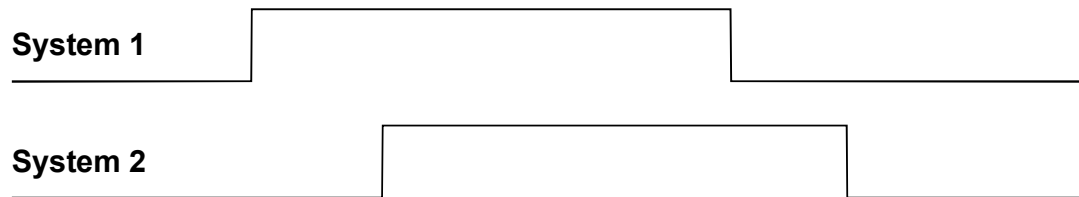
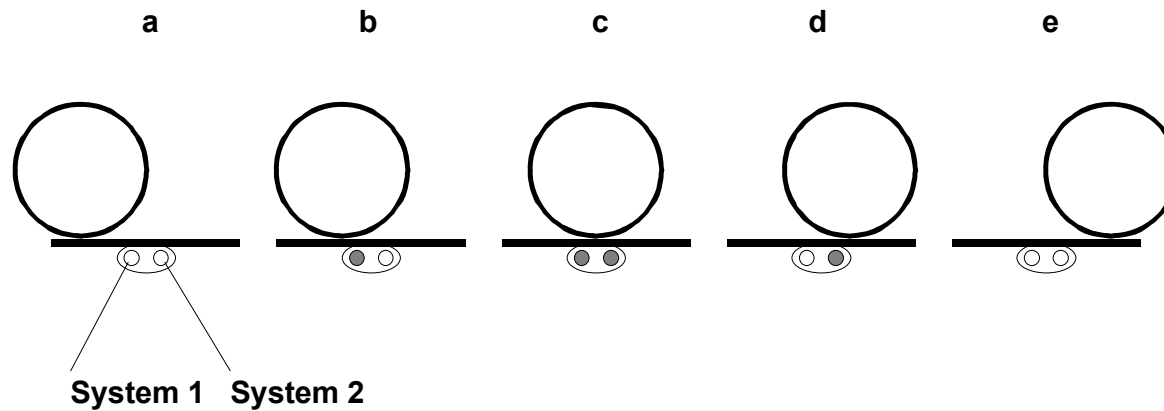
- Das Prinzip der Achszählzählung



GAG = Gleisanschlussgehäuse  
GFM = Gleisfreimeldeabschnitt

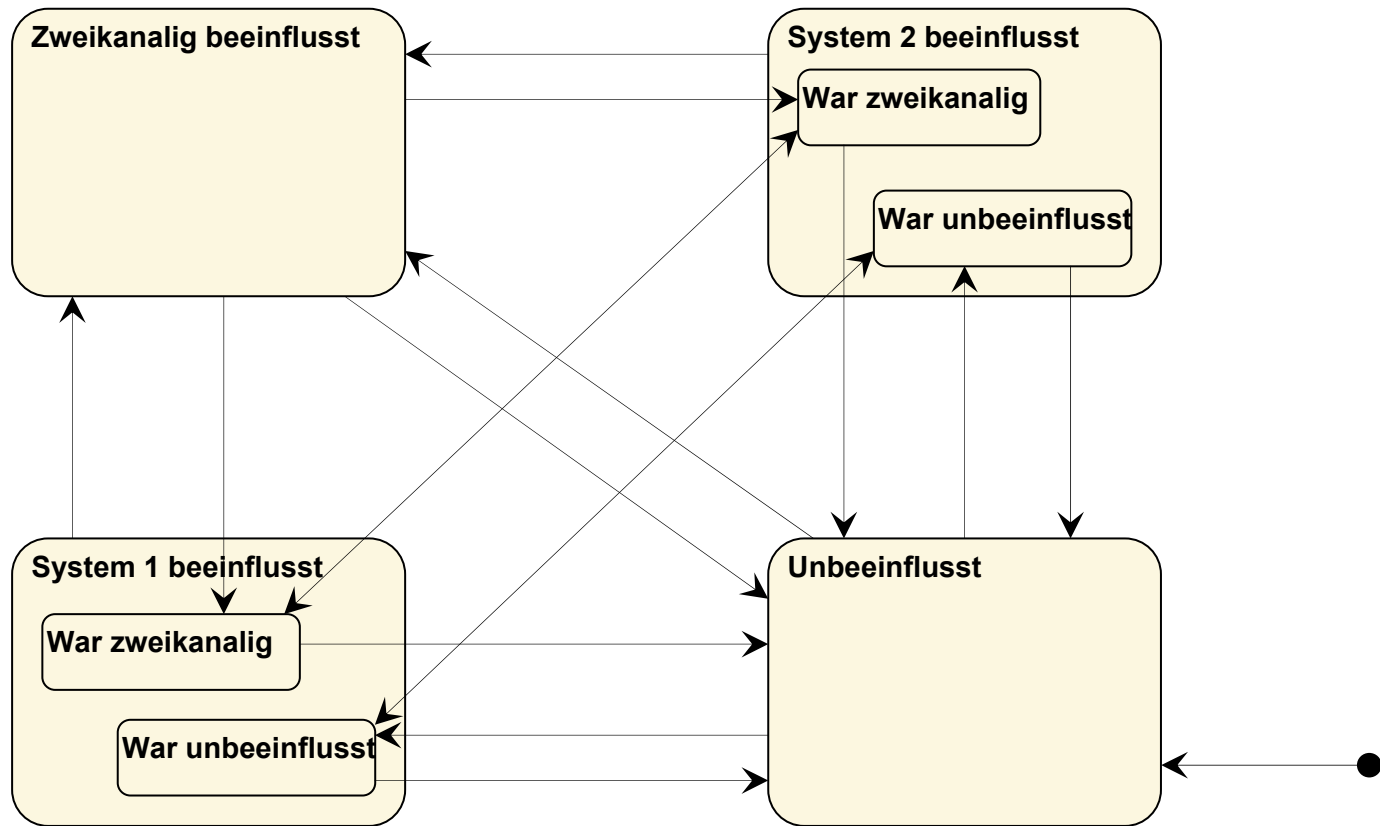
# Achszählsystem

- Achszählung mit Doppelsensor



# Achszählsystem

- Zustandsautomat des Zählpunkts



# Überblick

- I. Formale Verifikation
- II. Achszählsystem
- III. Theorem Prover Isabelle**
- IV. Darstellung des Achszählsystems in Isabelle
- V. Abschluss

# Theorem Prover Isabelle

- Cambridge Universität und TU München
- Logiken: HOL, FOL, ZF...
- Isabelle/HOL

HOL = Funktionale Programmierung + Logik

- Implementierungssprache ML
- Betriebssystem
  - Linux, Solaris, MacOS X
  - Windows

# Theorem Prover Isabelle

- Theorien
- `theory T=B1 + ... + Bn :`  
*declarations, definitions and proofs*  
`end`

```
header (* Konstanten für Achszähler.thy *)

theory Achszaehler_const = Main:

constdefs
  t_coll      :: nat
  "t_coll    == 2500"
  t_trail     :: nat
  "t_trail   == 30"
  t_stat      :: nat
  "t_stat    == 250"
  t_short     :: nat
  "t_short   == 30"
  t_pow1      :: nat
  "t_pow1    == 125"
  t_pow2      :: nat
  "t_pow2    == 8"
  t_dsupmax   :: nat
  "t_dsupmax == 30"
  t_dsupidle  :: nat
  "t_dsupidle== 8"
  t_idle      :: nat
  "t_idle    == 25000"
  s_max       :: nat
  "s_max     == 1"
  c_max       :: int
  "c_max     == 3"
  rl_max      :: nat
  "rl_max    == 1"
  gx_max      :: nat
  "gx_max    == 4"
```

end

# Theorem Prover Isabelle

- Beweisdurchführung = Anwenden von
  - Regeln  
  `apply (rule hd_append)`
  - Taktiken  
  `apply (auto)`
  - Methoden  
  `apply (simp)`
  
- Goal buffer
  - G
  - 1. G1
  - ...
  - n. Gn

# Überblick

- I. Formale Verifikation
- II. Achszählsystem
- III. Theorem Prover Isabelle
- IV. Darstellung des Achszählsystems in Isabelle**
- V. Abschluss

# Darstellung des Achszählsystems in Isabelle

- Darstellung des Automaten in Isabelle

- Zustände

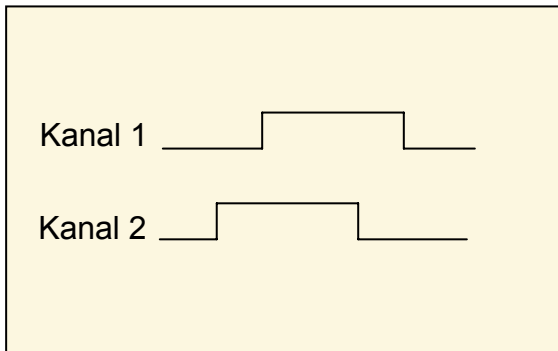
```
datatype  
zustand = S0 | S1_war_0 | S1_war_12 | S12 | S2_war_0 | S2_war_12
```

- Übergänge

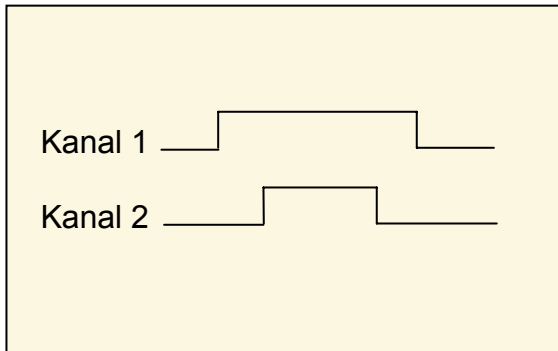
```
types uebergang = "zustand*zustand"  
constdefs uebergangs :: "uebergang set"  
"uebergangs ≡  
{(S0, S1_war_0), (S0, S2_war_0), (S0, S12),  
(S1_war_0, S12), (S1_war_0, S0), (S1_war_0, S2_war_0),  
(S1_war_12, S12), (S1_war_12, S0), (S1_war_12, S2_war_12),  
(S12, S2_war_12), (S12, S1_war_12), (S12, S0),  
(S2_war_12, S0), (S2_war_12, S12), (S2_war_12, S1_war_12),  
(S2_war_0, S0), (S2_war_0, S12), (S2_war_0, S1_war_0)}"
```

# Darstellung des Achszählsystems in Isabelle

- Testfall



**INormaleBefahrung = [S0,  
S2\_war\_0,  
S12,  
S1\_war\_12,  
S0]**



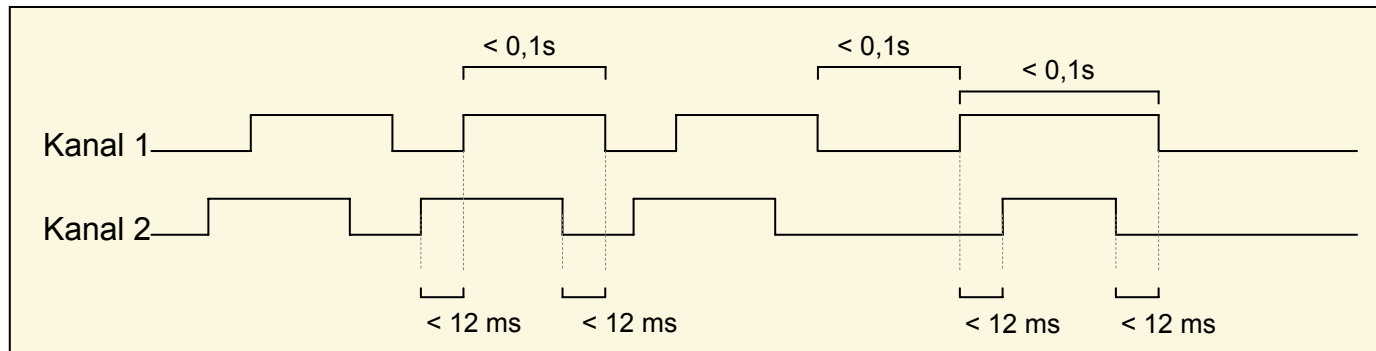
**rEntgegenFahrtrichtung = [S0,  
S1\_war\_0,  
S12,  
S1\_war\_12,  
S0]**

# Darstellung des Achszählsystems in Isabelle

- **Testfall 203:**

Es werden bis zu  $1+a/rl_{\max}$  Zustandsübergänge entgegen der Fahrtrichtung zugelassen. Um das Zählverhalten zu überprüfen, folgt in den Fällen 1 bis 3 der Simulation einer ordnungsgemäßen Befahrung von 2 bzw. 3 Achsen eine Anzahl zweikanaliger Störungen...

**Fall 1:** Im Fall 1 bewirkt eine zweikanalige Störung 2 ( $<1+a/rl_{\max}$ ,  $a=4$ ) Zustandsübergänge entgegen der Fahrtrichtung und wird toleriert. Die zweikanalige Störung wird als Achse eingezählt.



# Darstellung des Achszählsystems in Isabelle

```
lemma testfall1203F1 : "x = fAutomat (fVerketteten
  [lNormaleBefahrung, lNormaleBefahrung, lNormaleBefahrung,
  rEntgegenFahrtrichtung])
  (((0,0,0,0,0), (0,0,0,0,0), 0), (0,0,False)) False (0,False) ==>
  (snd(snd(fst(fst x)))=-4  $\square$  fFreimeldung x=True)"

apply(simp add: lNormaleBefahrung_def
  rEntgegenFahrtrichtung_def
  null_z1z2z3sc_def
  t_trail_def
  t_coll_def
  t_stat_def
  t_short_def
  fAddZaehlers_def
  fAuswertZaehlers_def
  fZustandsuebergang_def
  s_max_def c_max_def
  fFreimeldung_def
  Let_def)

done
```



# Überblick

- I. Formale Verifikation
- II. Achszählsystem
- III. Theorem Prover Isabelle
- IV. Darstellung des Achszählsystems in Isabelle
- V. **Abschluss**

# Abschluss

- Zusammenfassung
  - Formale Verifikation
    - Methode: Theorem Proving
  - Achszählsystem – ein Beispiel in Eisenbahntechnik
    - Achszählrechner und Zählpunkteinrichtungen
    - Zustandsautomat des Zählpunkts

# Abschluss

- Theorem Prover Isabelle
  - Theorie
  - Beweisdurchführung
  
- Darstellung des Achszählsystems in Isabelle
  - Zustände
  - Übergänge
  - Testfälle

# Abschluss

## ■ Kritikpunkte

- + keine Begrenzung der Größe eines Modells
- + keine Zustandsexplosion
- + Beweisen abstrakter Probleme
- + bessere Kontrolle des Beweises
- + gutes Verständnis des Problems
- Halb-Automatik
- gute Kenntnisse der Mathematik
- keine Gegenbeispiele

# Abschluss

- Weitere Schritte
  - 100% Implementierung des Mehrachsenzählverfahrens
  - wie viel Achsen könnten gezählt werden?
  - Durchführung aller Testfällen
  - evtl. Zusammenfassung einigen Testfällen
  - bis jetzt sind Eingangswerte für den Automat Konstanten; könnten sie Variablen sein?
    - Optimierung des Automaten?
    - Entdeckung der Lücken?
  - Auf Isabelle in vollem Umfang zugreifen!