Analyse von Echtzeit-Systemen mit Uppaal

Dr. Carsten Weise
Ericsson Deutschland GmbH

Übersicht

- Zur Person
- Ericsson
- Dänemark
- Was ist Uppaal? Eine Einführung
- Grundlagen:
  Timed Automata
  Uppaal's Query-Language
  Symbolic Modelchecking
- Uppaals Software-Architektur
- Case Studies
- Einfache Beispiele/Demo

Zur Person

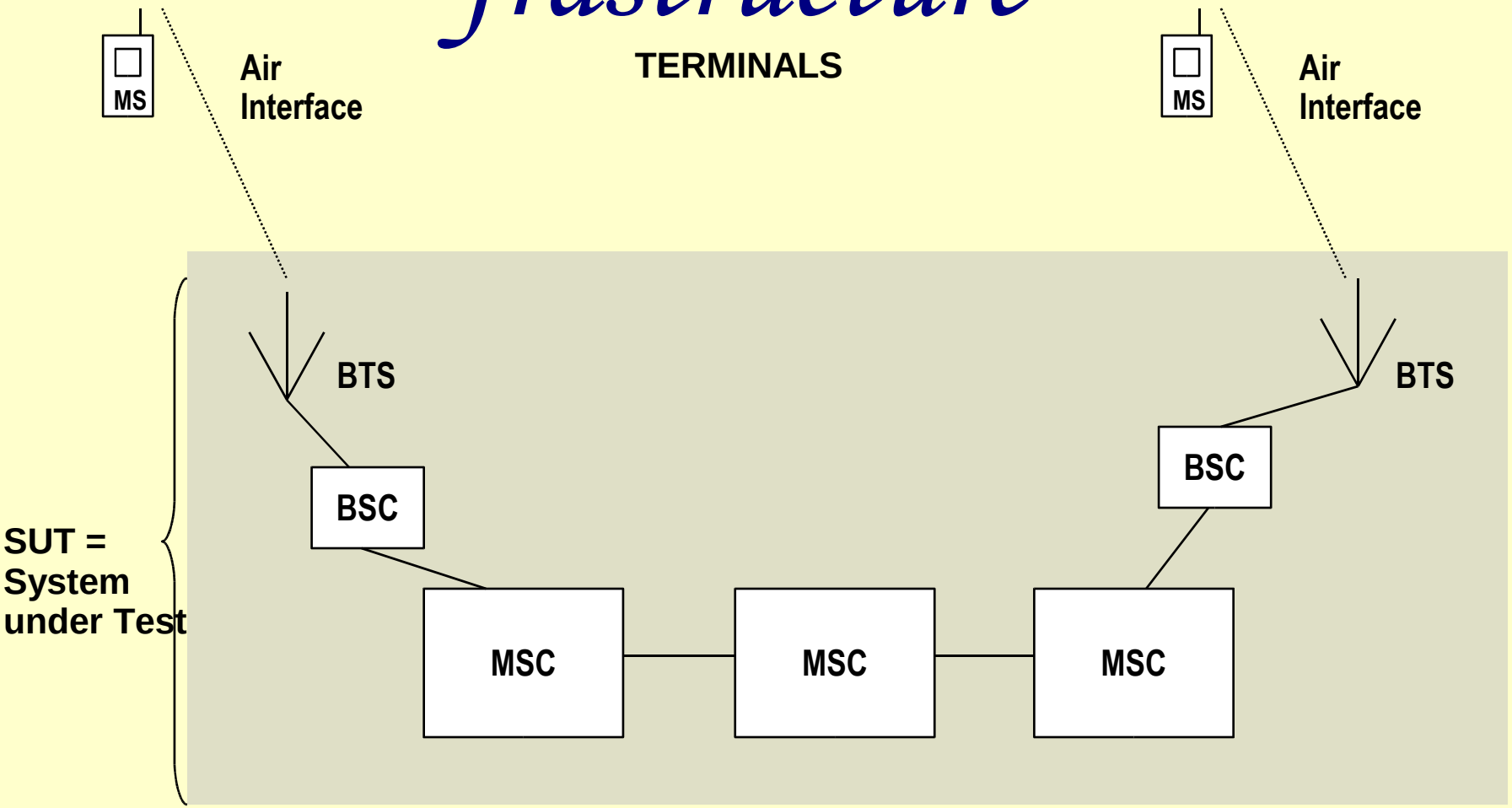
- *Carsten Weise, verheiratet, 3 Kinder, deutsch*
- *1980-87*: Informatik-Studium, RWTH Aachen
  Diplomarbeit: *Prozeßkommunikation durch Remote Procedure Calls in einem verteilten Echtzeitsystem*
- *1987-97*: Assistant am Lehrstuhl für Informatik I
- *1990-94*: Promotion bei Professor Dr. Steffen zur automatischen Analyse von Echtzeitsystemen
- *1997-99:* BRICS Postdoc, Universität Aalborg (DK), bei Professor Kim Larsen, Ph.D.
- *seit 1999*: Ericsson Deutschland GmbH
  - 1999 -2000:Projectmanager und System Designer Aims
  - 2000-2002: Group Manager “Test Tool Design”
  - 2003-2005: Group Manager “MSC Product Handling”
  - seit Mai 2005: Senior Systems Designer „Technical Studies”

Ericsson

- Multinationaler Telekommunikations-Konzern mit Hauptsitz in Schweden
- Hauptprodukt: Telekom Infrastrukur
  #1 in Telco B2B Infrastructure in the World
- Handys werden bei Sony-Ericsson hergestellt...
- Filialen in aller Welt
- R&D Sites: Aachen, Montreal, Shanghai
- **`carsten.weise at ericsson.com`**

B2B = Business to Business
R&D = Research & Development

Mobile Communication Infrastructure



Dänemark

- *1997-99:*
  BRICS Postdoc, Universität Aalborg (DK), bei Professor Kim Larsen, Ph.D.
  - BRICS = Basic Research in Computer Science
  - *UPPAAL (*`www.uppaal.com`*): Tool für Analyse von Echtzeitsystemen*
  - Tool Integration Platform,
  - *Verification of Hybrid Systems (Esprit)*
  - System-Modellierung
  - Theorie der Echtzeitsysteme

UPPAAL

Tool für die

- Spezifikation
- Simulation
- Verifikation

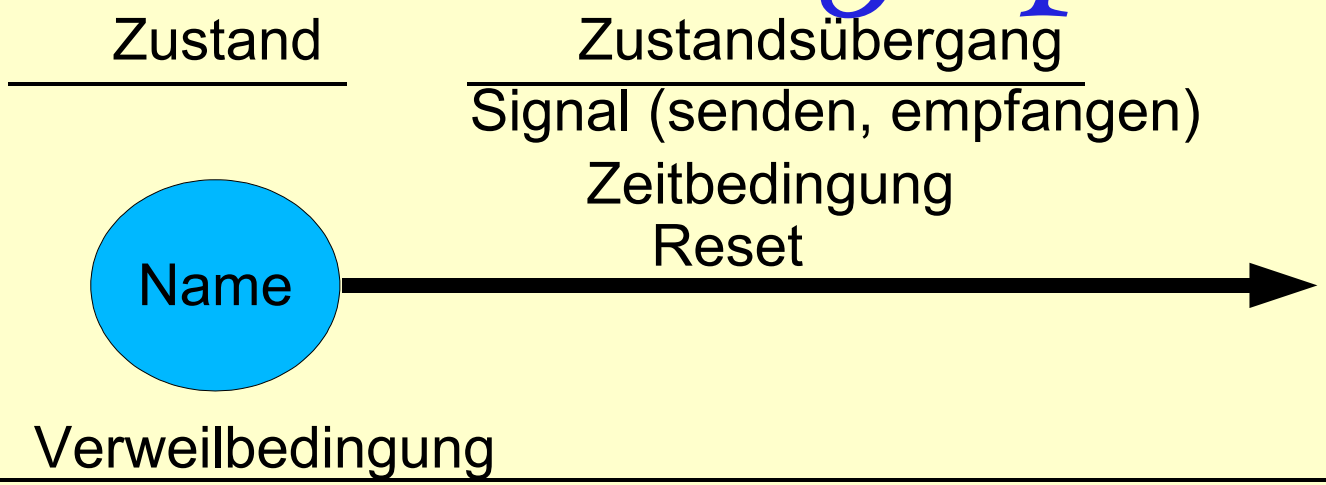
von Echtzeit-Systemen

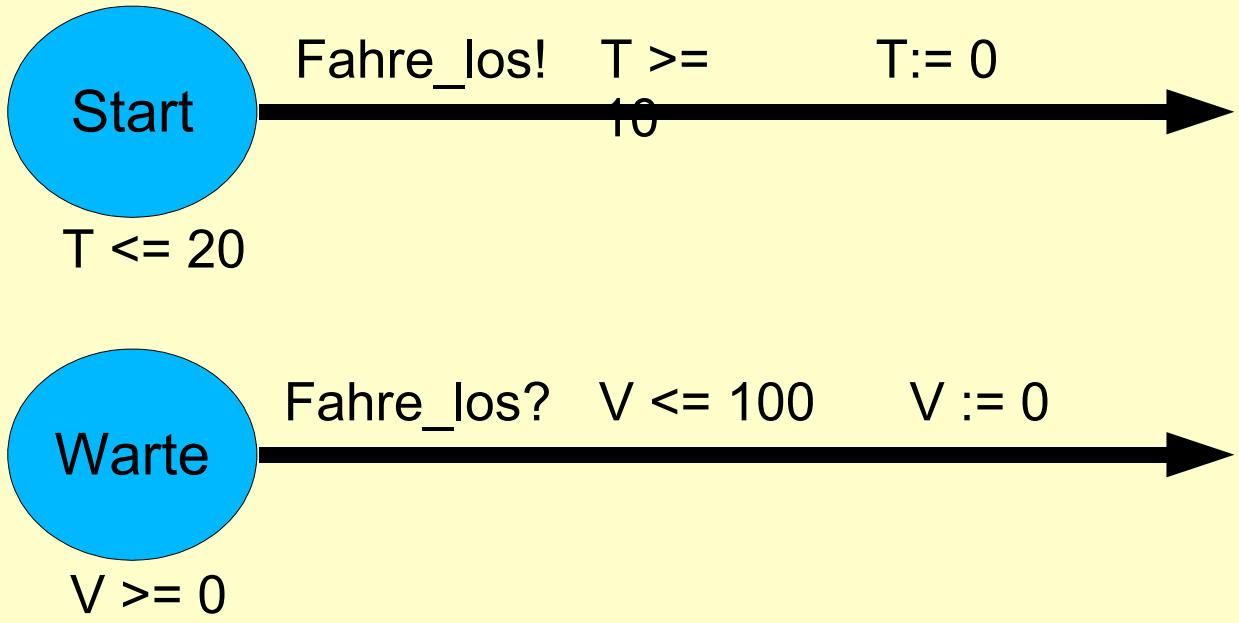
Timed Automata

Formalismus zur Spezifikation von Echtzeitsystemen

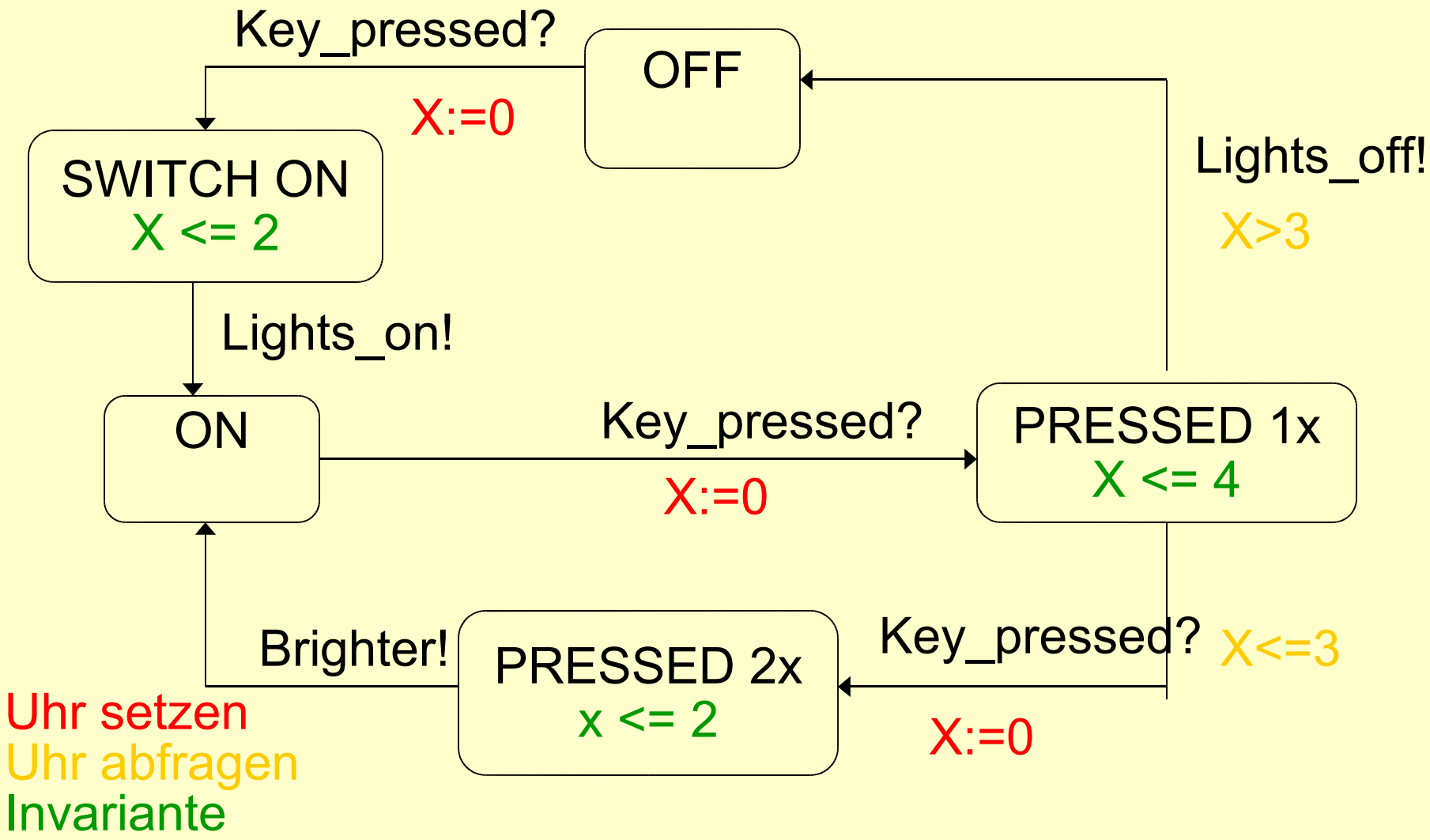
(Endlicher Automat + Uhren)

Modellierungssprache





Beispiel Timed Automata



Uppaal Modell

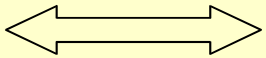
- Menge paralleler, kommunizierender Timed Automata
- Beliebig viele Uhren (alle gleich schnell)
- zusätzlich Variablen mit endlichem Wertebereich

Uppaal's Query Language

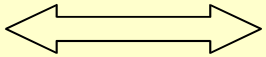
- Atomare Eigenschaften:
  P.loc - Prozess P in Location l
  X == n - Uhr x gleich n (Integerwert)
  X<= n, X <n - entsprechend
- Kombinatoren: e1 and e2, not e1,
  e1 or e2, e1 implies e2
- „Safety Properties“:
  A[] e - „always on all paths e“
  E<> e - „eventually on some path e“

Mutual Exclusion

- In keinem Zustand sollen sowohl P1 wie P2 in ihrer Criticial Section sein



- In allen Zuständen gilt, dass nicht gleich-zeitig P1 und P2 in ihrer CS sind



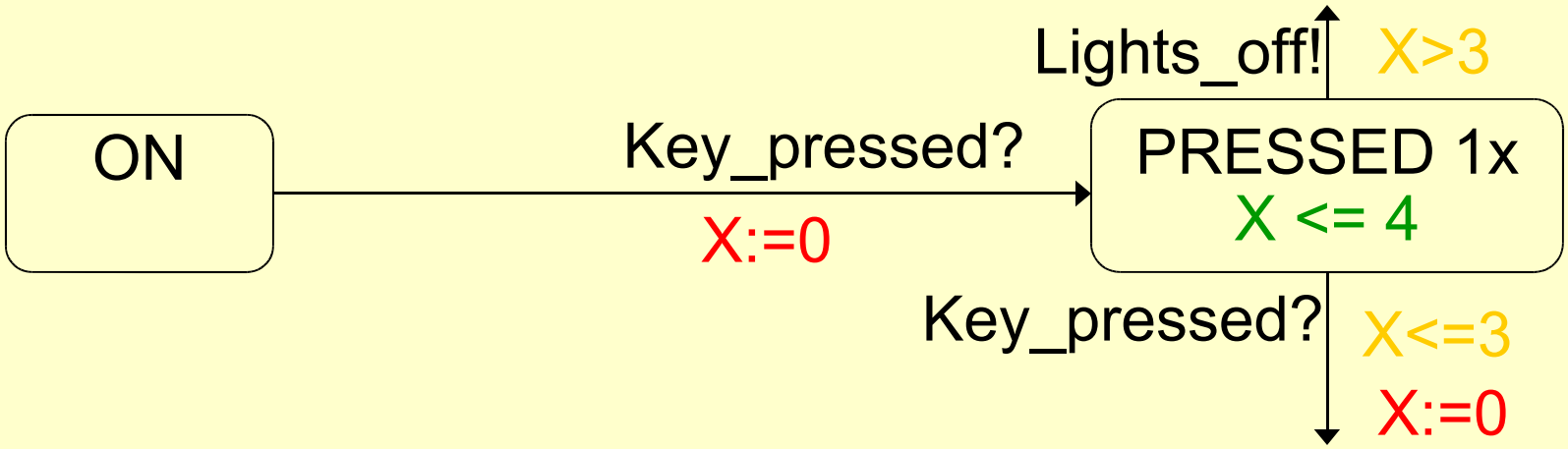
- A[] not (P1.CS and P2.CS)

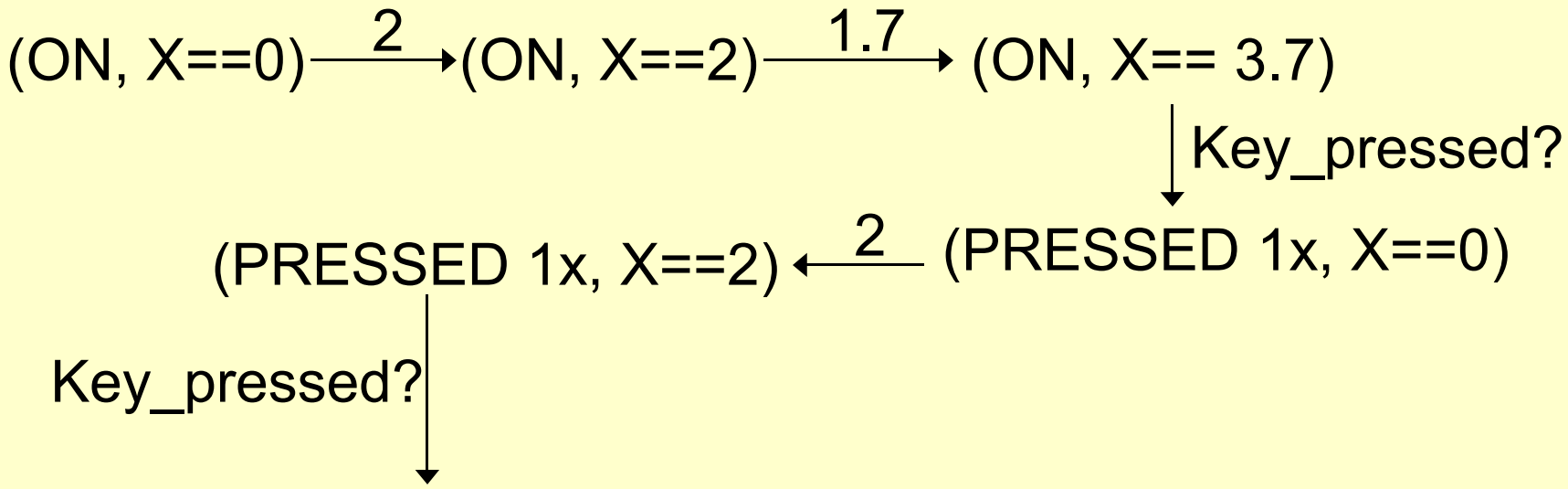
Model Checking

Verfahren zum Überprüfen von Eigenschaften einer Spezifikation
(z.B. Deadlocks, Mutual Exclusion,
toter Code, i.a. Korrektheit)

check if the system
is a ***model*** of the formula

Ausführung


Lights_off!
X>3
ON
Key_pressed?
X:=0
PRESSED 1x
X <= 4
Key_pressed?
X<=3
X:=0


(ON, X==0)
2
(ON, X==2)
1.7
(ON, X== 3.7)
Key_pressed?
(PRESSED 1x, X==0)
2
(PRESSED 1x, X==2)
Key_pressed?

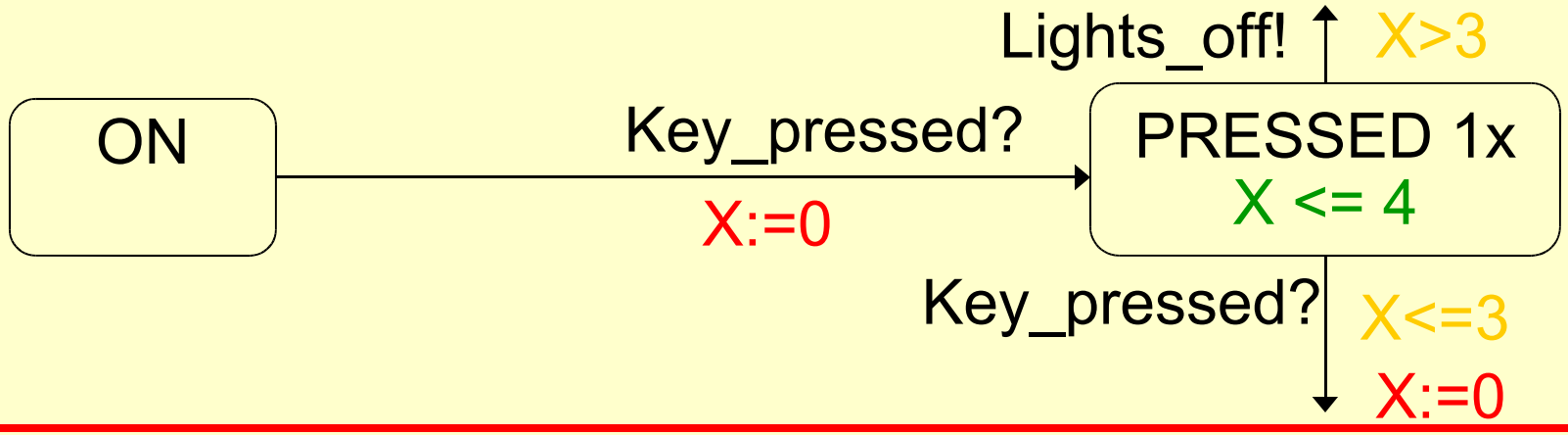
Zustandsraum

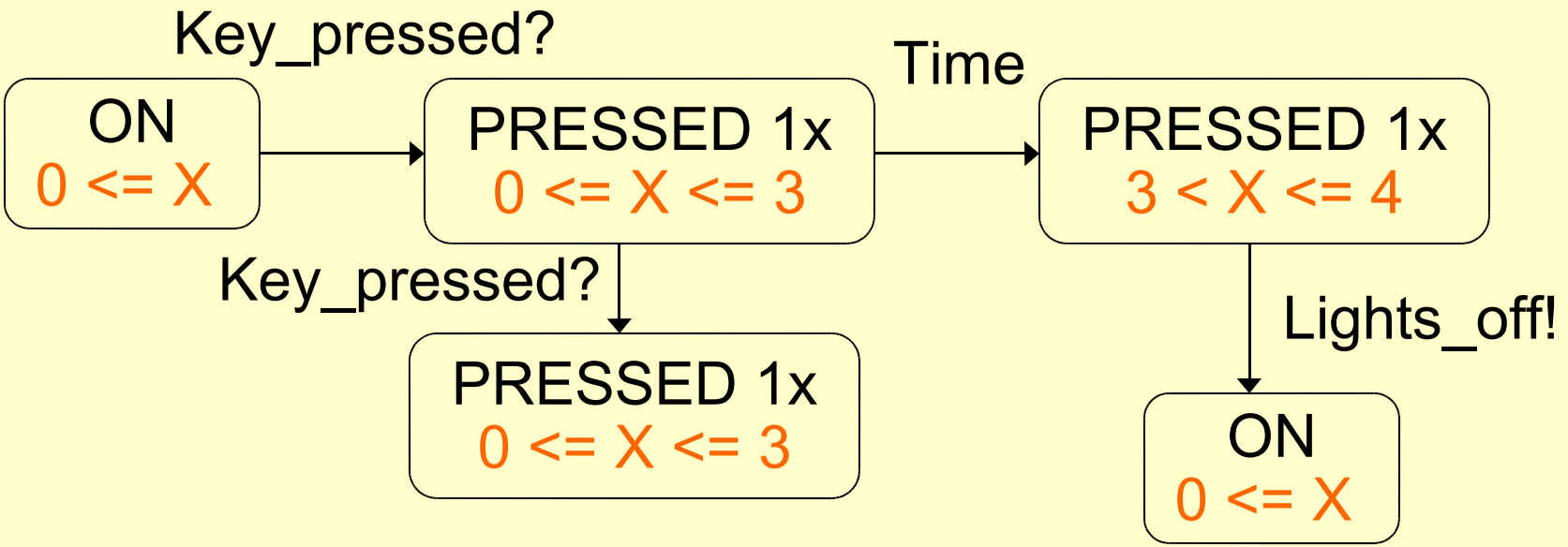
- Zustand eines Timed Automata = (Location, Timer1, … , TimerN)
- Zustandsraum = *Locations* x *Reals* x …. X *Reals*
- Problem: unendlicher Zustandsraum (sogar ueberabzaehlbar)

Methoden

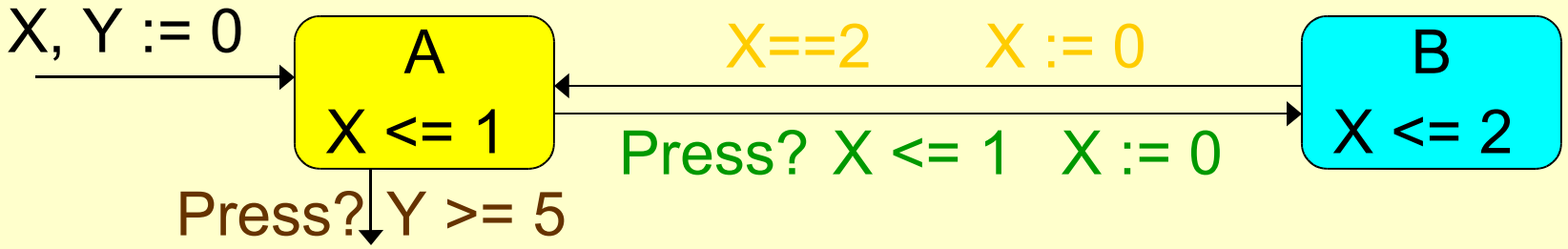
- Model Checking generell:
  Durchsuchen des gesamten Zustandsraums
- Symbolic Model Checking:
  Zusammenfassen ähnlicher Zustände zu Mengen
- Model Checking = Erreichbarkeits-Analyse

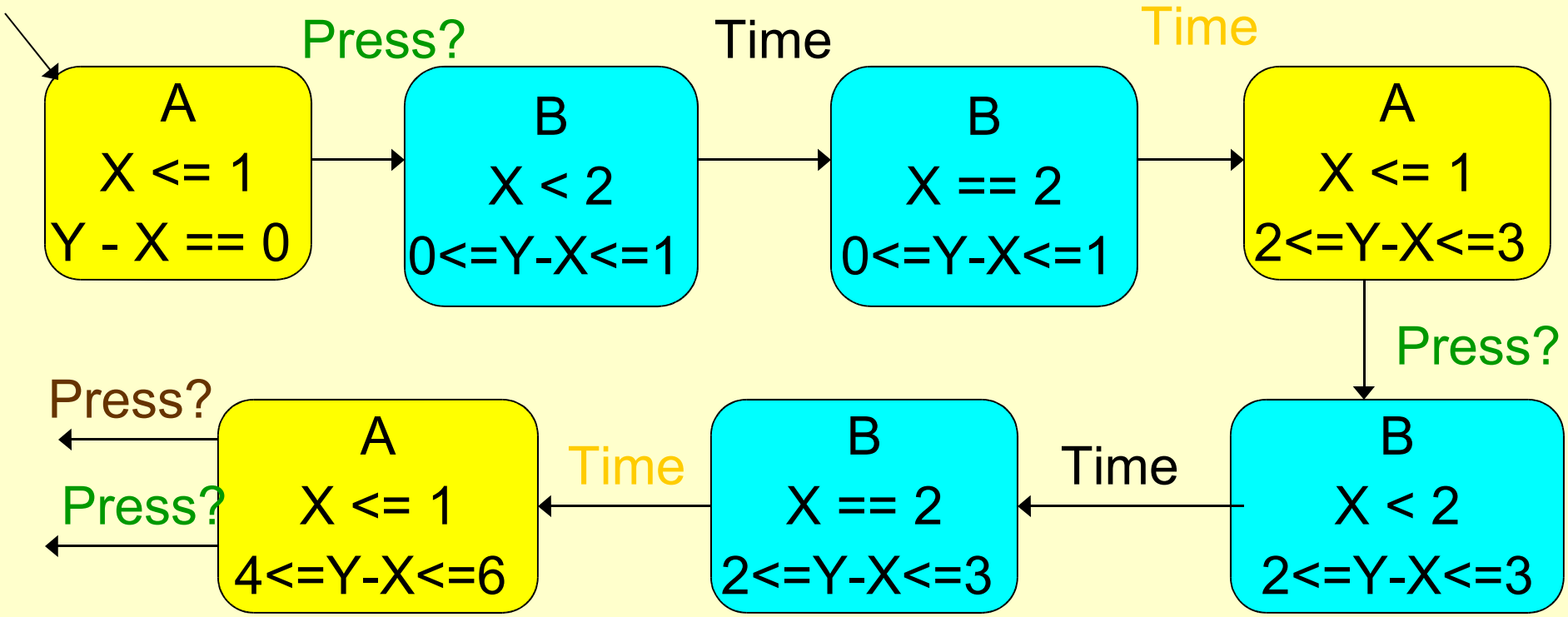
Symbolische Ausführung





Symbolische Ausführung





Der Zug

- Wenn der Zug vom Sensor vor dem Bahnübergang erfaßt wird, wird ein Signal **`Nähere_mich`** gesendet.
- Danach braucht der Zug noch 5 Sekunden bis er im Bereich des Bahnübergangs ist
- Der Zug braucht 10 Sekunden, um den Bereich des Bahnübergangs zu durchqueren

Die Schranke

- Die Schranke reagiert auf ein Steuersignal `Impuls`
- Je nach Position reagiert sie durch Heben oder Senken
- Für Heben und Senken braucht die Schranke jeweils 2 Sekunden

Die Steuereinheit

- Die Steuereinheit wartet auf das Signal **`Nähere_mich`**
- Wenn sie das Signal **`Nähere_mich`** erhält, schickt sie einen Impuls an die Schranke
- Die Steuereinheit braucht 2 Sekunden um das Signal **`Nähere_mich`** zu verarbeiten und das Signal **`Impuls`** zu senden

Die Sicherheitsbedingung

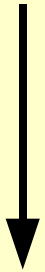
- Wenn ein Zug sich im Bereich des Bahnübergangs befindet, muß die Schranke unten sein
-
- Zug@BÜ => Schranke@Unten

Parallele Komposition

Parallele Komposition macht aus vielen Graphen einen Graphen

(Fern | Oben | Warten)

Nähere_mich



(Nah | Oben | Signal_verarbeiten)

(Nah | Unten | Warten)

Betrete_BÜ

(BÜ | Unten | Warten)

Erreichbarkeitsanalyse

Gibt es im Graphen des Gesamtzsystems
einen vom Anfangszustand

("Fern" | "Oben" | "Warten")

erreichbaren Zustand

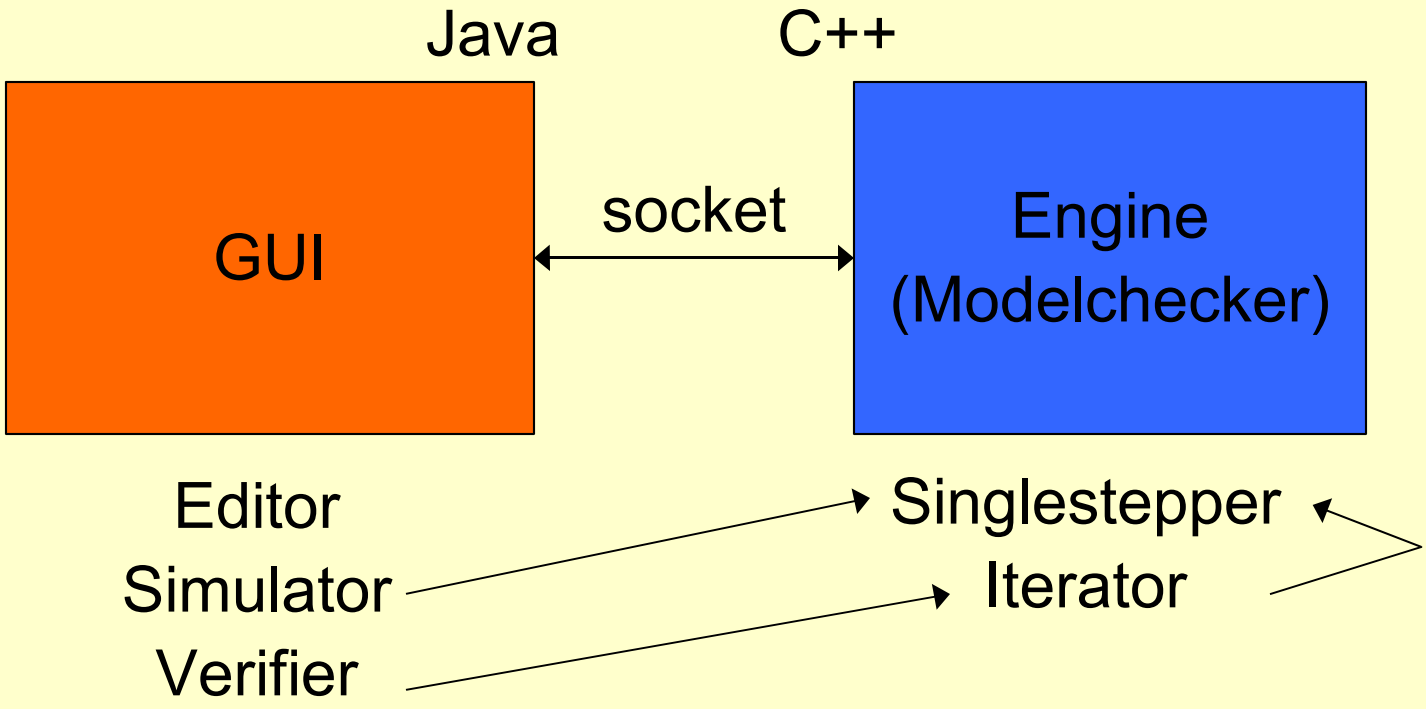
(Z | S | SE)

mit Z == "BÜ" und S != "Unten" ?

Wichtige Eigenschaften der Timed Automata

- Endliche symbolische Darstellung existiert
- Symbolische Darstellung erhält die Semantik
- Erreichbarkeit und Bisimulation entscheidbar
-

Software Architecture



Runs on Linux, Windows, Solaris

Runs remotely

Uppaal Demo

Zusammenfassung

- Mächtiger Formalismus für Echtzeit-systeme
- effektive und effiziente Analyse
- praktikabel (versch. Industrieanwndgn)
- Erweiterung der klassischen Theorie endlicher Automaten
- Erweiterung auf komplexere Beispiele möglich (driftende Uhren, Scheduling, hybride Systeme)