# Software Engineering in der Praxis

## Praktische Übungen

# Model Checking I

Florin Pinte Marc Spisländer

Lehrstuhl für Software Engineering
Friedrich-Alexander-Universität Erlangen-Nürnberg

# Model Checking

## Ziel

Fehlerentdeckung in der Systemspezifikation

# Model Checking

## Ziel

Fehlerentdeckung in der Systemspezifikation

## Vorgehensweise

1. Modellierung des Systems durch endliches Zustandstransitionssystem

# Model Checking

## Ziel

Fehlerentdeckung in der Systemspezifikation

## Vorgehensweise

1. Modellierung des Systems durch endliches Zustandstransitionssystem
2. Formulierung der Anforderungen, die das System erfüllen muss, durch logische Formeln

# Model Checking

## Ziel

Fehlerentdeckung in der Systemspezifikation

## Vorgehensweise

1. Modellierung des Systems durch endliches Zustandstransitionssystem
2. Formulierung der Anforderungen, die das System erfüllen muss, durch logische Formeln
3. Automatisches Beweisen, dass das System die Anforderung erfüllt oder nicht erfüllt

# Model Checking im Praktikum

## Werkzeug

NuSMV `http://nusmv.irst.itc.it`

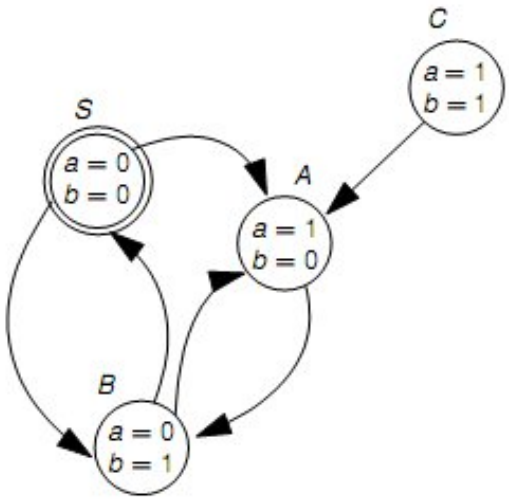
## Formalismus

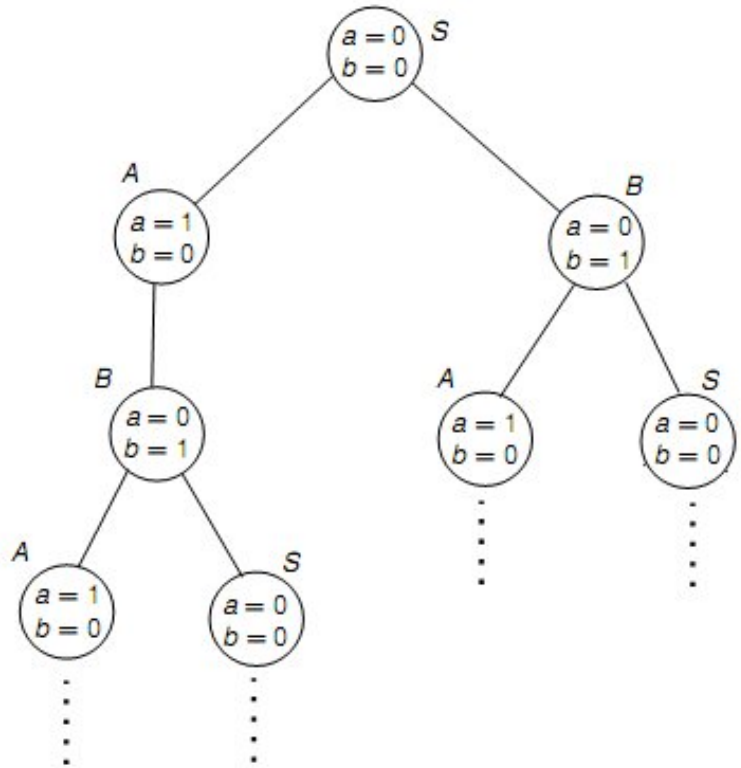
Computational Tree Logic (CTL)

# Computational Tree Logic

## Idee

Betrachte Modell eines Systems:



# Computational Tree Logic

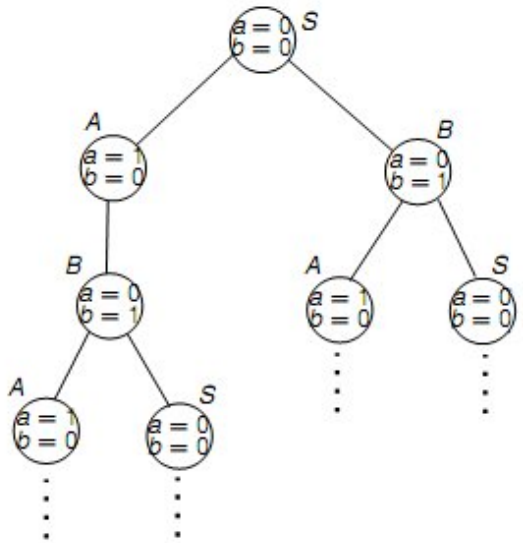
## Idee

### Frage

Gibt es einen Übergangspfad aus dem Startzustand in den Zustand C?

### Formal

Ist die CTL-Formel $S \models \mathrm{EF}\,(a = 1 \wedge b = 1)$ *wahr*?



# Atomare CTL-Formeln

## Definition

Sei $\mathrm{AP}$ eine endliche Menge von *Aussagen*. Falls $A \in \mathrm{AP}$, dann ist $A$ eine *CTL-Formel*.

# Zusammengesetzte CTL-Formeln

## Definition

Falls $F_1$ und $F_2$ CTL-Formeln sind, dann sind folgende Wörter ebenfalls CTL-Formeln:

# Zusammengesetzte CTL-Formeln

## Definition

Falls $F_1$ und $F_2$ CTL-Formeln sind, dann sind folgende Wörter ebenfalls CTL-Formeln:

- $\neg F_1$

# Zusammengesetzte CTL-Formeln

## Definition

Falls $F_1$ und $F_2$ CTL-Formeln sind, dann sind folgende Wörter ebenfalls CTL-Formeln:

- $\neg F_1$
- $F_1 \wedge F_2$

# Zusammengesetzte CTL-Formeln

## Definition

Falls $F_1$ und $F_2$ CTL-Formeln sind, dann sind folgende Wörter ebenfalls CTL-Formeln:

- $\neg F_1$
- $F_1 \wedge F_2$
- AX $F_1$

# Zusammengesetzte CTL-Formeln

## Definition

Falls $F_1$ und $F_2$ CTL-Formeln sind, dann sind folgende Wörter ebenfalls CTL-Formeln:

- $\neg F_1$
- $F_1 \wedge F_2$
- AX $F_1$
- EX $F_1$

# Zusammengesetzte CTL-Formeln

## Definition

Falls $F_1$ und $F_2$ CTL-Formeln sind, dann sind folgende Wörter ebenfalls CTL-Formeln:

- $\neg F_1$
- $F_1 \wedge F_2$
- AX $F_1$
- EX $F_1$
- A$[F_1$ U $F_2]$

# Zusammengesetzte CTL-Formeln

## Definition

Falls $F_1$ und $F_2$ CTL-Formeln sind, dann sind folgende Wörter ebenfalls CTL-Formeln:

- $\neg F_1$
- $F_1 \wedge F_2$
- AX $F_1$
- EX $F_1$
- A$[F_1$ U $F_2]$
- E$[F_1$ U $F_2]$

# Interpretation der CTL-Formeln

CTL-Formeln werden durch *CTL-Strukturen* interpretiert.

## Definition

Eine *CTL-Struktur* ist ein Tripel $(S, R, P)$, mit

| | |
|---|---|
| $S$ | endliche Zustandsmenge |
| $R \subseteq S \times S$ | totale Übergangsrelation |
| $P : S \to 2^{\mathrm{AP}}$ | ordnet jedem Zustand $z \in S$ alle in $z$ wahren Aussagen zu. |

# Atomare CTL-Formeln

## Definition

$A \in \mathrm{AP}$ ist wahr im Zustand $s \in S$, genau dann wenn $A \in P(s)$.
Schreibweise: $s \models A$.

# Zusammengesetzte CTL-Formeln

## Definition

Für $s \in S$ und CTL-Formeln $F_1$ und $F_2$ definiert man:

$s \models \neg F_1$ gdw. $F_1$ in $s$ *nicht* wahr ist

# Zusammengesetzte CTL-Formeln

## Definition

Für $s \in S$ und CTL-Formeln $F_1$ und $F_2$ definiert man:

$s \models \neg F_1$ gdw. $F_1$ in $s$ *nicht* wahr ist

$s \models F_1 \wedge F_2$ gdw. $F_1$ *und* $F_2$ in $s$ wahr sind

# Zusammengesetzte CTL-Formeln

## Definition

Für $s \in S$ und CTL-Formeln $F_1$ und $F_2$ definiert man:

$s \models \neg F_1$ gdw. $F_1$ in $s$ *nicht* wahr ist

$s \models F_1 \wedge F_2$ gdw. $F_1$ *und* $F_2$ in $s$ wahr sind

$s \models \mathrm{AX}\, F_1$ gdw. $F_1$ in *allen direkten Nachfolgern* von $s$ wahr ist

# Zusammengesetzte CTL-Formeln

## Definition

Für $s \in S$ und CTL-Formeln $F_1$ und $F_2$ definiert man:

$s \models \neg F_1$ gdw. $F_1$ in $s$ *nicht* wahr ist

$s \models F_1 \wedge F_2$ gdw. $F_1$ *und* $F_2$ in $s$ wahr sind

$s \models \mathsf{AX}\, F_1$ gdw. $F_1$ in *allen direkten Nachfolgern* von $s$ wahr ist

$s \models \mathsf{EX}\, F_1$ gdw. $F_1$ in *einem direkten Nachfolger* von $s$ wahr ist

# Zusammengesetzte CTL-Formeln

## Definition

$s \models \mathrm{A}[F_1 \ \mathrm{U} \ F_2]$ gdw. für alle Pfade $(z_0 = s, z_1, z_2, \ldots)$ gilt:

Es existiert ein $i \geq 0$ mit $z_i \models F_2$ und $z_j \models F_1$ für alle $j \leq i$.

# Zusammengesetzte CTL-Formeln

## Definition

$s \models \mathsf{A}[F_1 \ \mathsf{U} \ F_2]$ gdw. für alle Pfade $(z_0 = s, z_1, z_2, \ldots)$ gilt:

Es existiert ein $i \geq 0$ mit $z_i \models F_2$ und $z_j \models F_1$ für alle $j \leq i$.

## Definition

$s \models \mathsf{E}[F_1 \ \mathsf{U} \ F_2]$ gdw. es gibt einen Pfad $(z_0 = s, z_1, z_2, \ldots)$ für den gilt:

Es existiert ein $i \geq 0$ mit $z_i \models F_2$ und $z_j \models F_1$ für alle $j \leq i$.

# Abkürzungen für CTL-Formeln

AF $F$ $\mathsf{A}[\mathit{True}\ \mathsf{U}\ F]$

$F$ ist unvermeidbar

EF $F$ $\mathsf{E}[\mathit{True}\ \mathsf{U}\ F]$

$F$ ist möglich

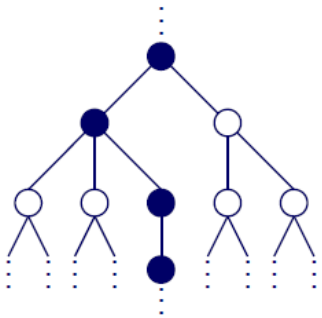
AG $F$ $\neg\mathsf{EF}\ \neg F$

Auf allen Pfaden, die im aktuellen Zustand starten, ist $F$ in jedem Zustand des Pfades wahr.

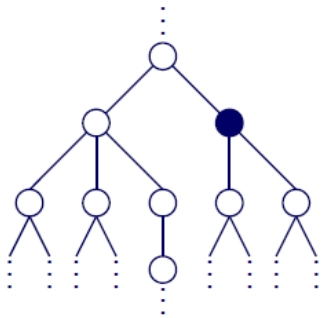
EG $F$ $\neg\mathsf{AF}\ \neg F$

Es ex. ein Pfad, der im aktuellen Zustand startet, so dass $F$ in jedem Zustand des Pfades wahr ist.

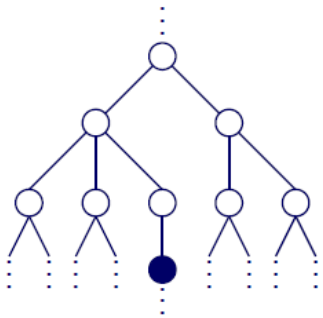
# Grafische Veranschaulichung der CTL-Formeln



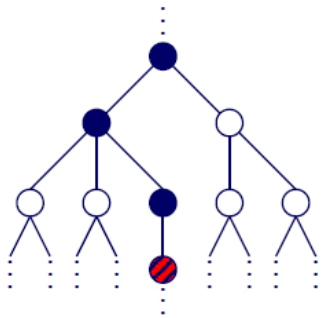
EG $F$



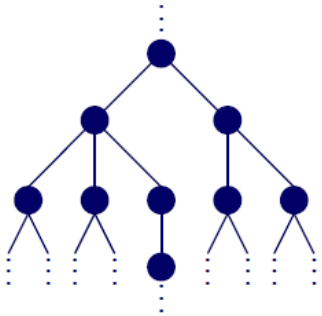
EX $F$



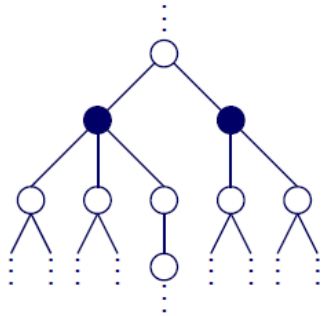
EF $F$



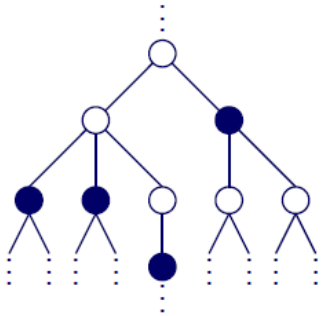
E$[F_1$ U $F_2]$



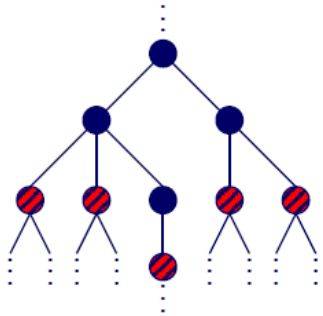
AG $F$



AX $F$



AF $F$



A$[F_1$ U $F_2]$

# Das Model-Checking-Problem

## Gegeben

CTL-Struktur, Zustand $s$ und CTL-Formel $F$.

# Das Model-Checking-Problem

## Gegeben

CTL-Struktur, Zustand $s$ und CTL-Formel $F$.

## Gesucht

Gilt $s \models F$?

# Model Checking mit NuSMV

NuSMV erlaubt:

- die Formulierung einer CTL-Struktur
- die Simulation von Zustandsübergängen in der CTL-Struktur
- Model Checking (Lebendigkeit, Deadlocks, . . . )

# Das NuSMV-Modell

- Das NuSMV-Modell besteht aus benannten Modulen; eins davon muss `main` heißen.
- Innerhalb von Modulen kann man Variablen deklarieren, ihnen einen Initialwert zuweisen und ihren Wert im sog. *nächsten Schritt* definieren.
- Variablen haben endlichen Definitionsbereich

# Das NuSMV-Modell

## Der Zustandsraum von CTL-Strukturen

- Module werden instanziiert.
- Alle Modulinstanzen zusammen definieren den Zustandsraum einer CTL-Struktur, wie z. B.:

$$S = D_{x_a} \times D_{y_a} \times D_{x_b} \ldots$$

wobei $D_{x_a}$ der Definitionsbereich der Variablen $x$ in der Modulinstanz $a$ ist.

# Das NuSMV-Modell

## Die Übergangsrelation von CTL-Strukturen

- Die *Instanziierungsart* der Module definiert die Übergangsrelation der CTL-Struktur.
- Zwei Arten von Modulinstanziierungen:

  **synchron** Der nächste Zustand wird definiert, indem alle Variablen aller Modulinstanzen auf ihren Wert *im nächsten Schritt* gesetzt werden.

  **asynchron** Der nächste Zustand wird definiert, indem alle Variablen *genau* einer Instanz auf ihren Wert *im nächsten Schritt* gesetzt werden.

# Beispiel

```
1  MODULE main
   VAR
3      semaphore: {red, red_yellow, green, yellow};
   INIT
5      semaphore = red;
   ASSIGN
7      next(semaphore) :=
           case
9              semaphore = red : red_yellow;
               semaphore = red_yellow : green;
11             semaphore = green : yellow;
               semaphore = yellow : red;
13         esac;
```

# Model Checking mit NuSMV

## Spezifikation von Anforderungen

```
1 SPEC AG
      ((semaphore = green -> AF semaphore = red) &
3     (semaphore = red -> AF semaphore = green));

5 SPEC AG AF semaphore = green;
```

# Wichtige NuSMV-Befehle

- NuSMV im interaktiven Modus starten:

```
NuSMV -int Modellname
```

- Übersicht über alle Befehle:

```
help
```

- Beschreibung eines Befehls:

```
befehl -h
```

- Initialisierung des System für die Verifikation:

```
go
```