Seminar Model Checking SS '07

UPPAAL Einführung: Das Uppaal Tool

Christoph Scheben

9. Juli 2007

Einleitung

- Verifikationstool für Echtzeitsysteme.
- Erfolgreiche Anwendung bei
  - Kommunikations- und Synchronisationsprotokollen,
  - Multimedia-Applikationen sowie
  - weiteren industriellen Fallstudien.
- Erste Version 1995, aktuell 4.0.6.
- Java Benutzeroberfläche, C++ Verifikator.

Übersicht

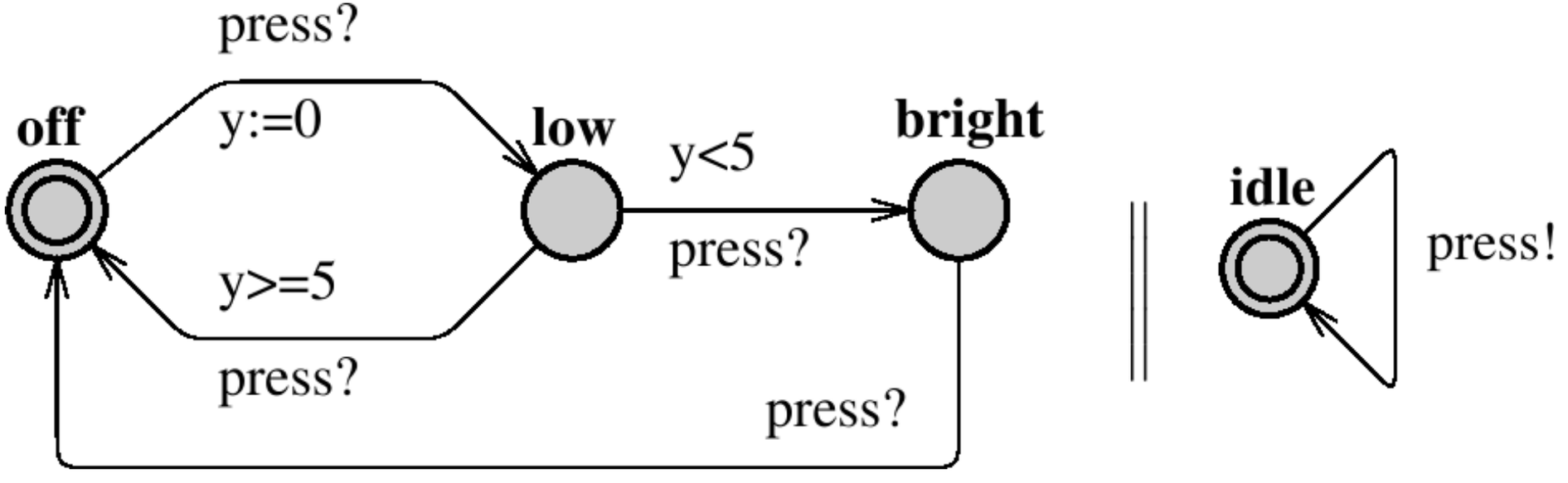
1. Die Modellierungssprache von UPPAAL
2. Die Abfragesprache von UPPAAL
3. Überblick über das Toolkit
4. Beispiel
5. Bemerkungen

Die Modellierungssprache von UPPAAL

- Netzwerk von parallelen erweiterten Timed Automata (TA).
- TA: Endlicher Automat mit Uhr-Variablen.
- Uhren: Reelle Zahlen, alle Uhren schalten synchron.
- Erweiterung des TA-Modells um diskrete Variablen.
- Systemzustand: Zustand der Automaten, Uhren, Variablen.
- Jeder Automat kann eine Kante "feuern" – alleine oder synchron.

Die Modellierungssprache von UPPAAL

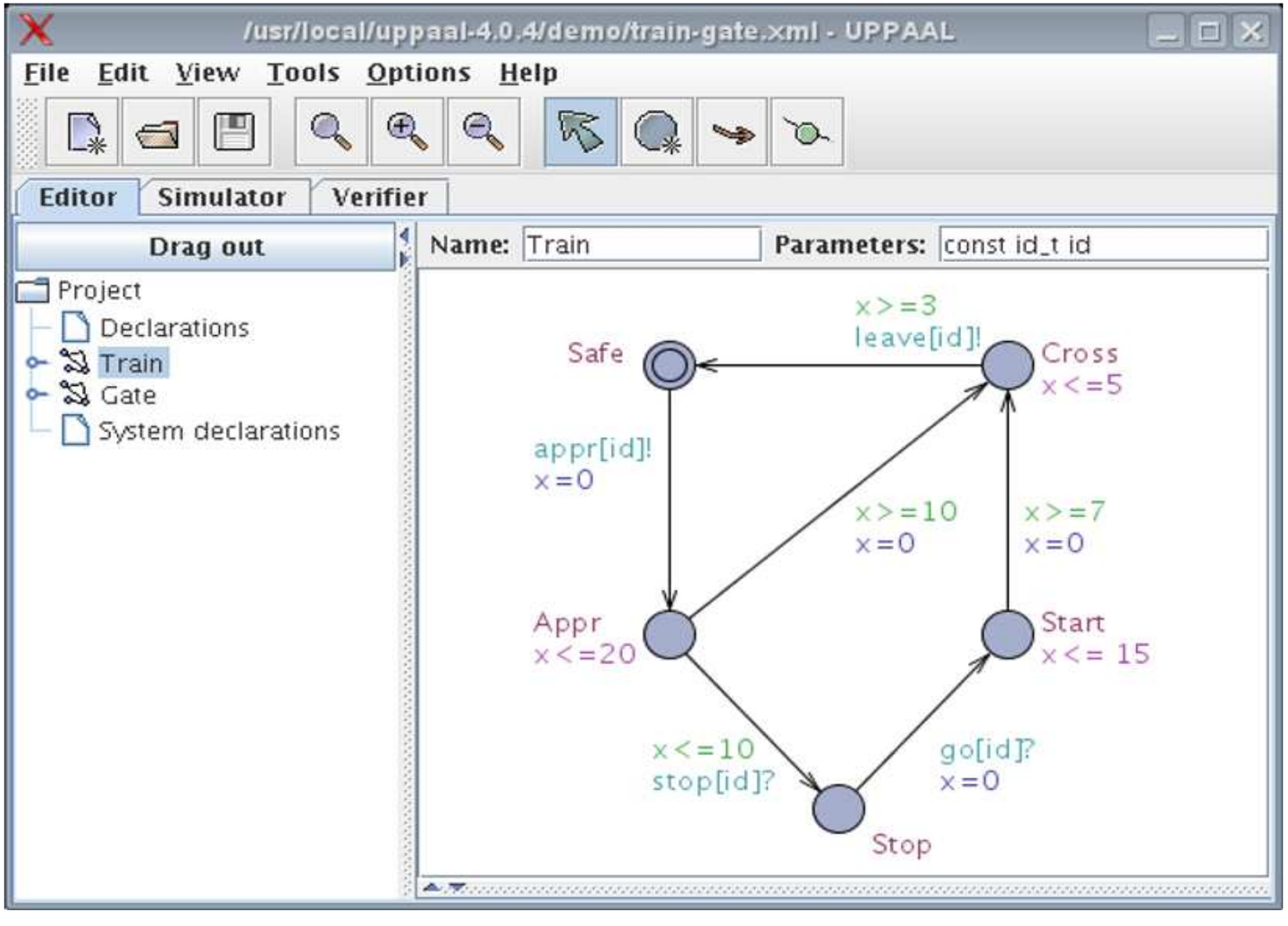
- Beispiel eines Netzwerks von TAs:



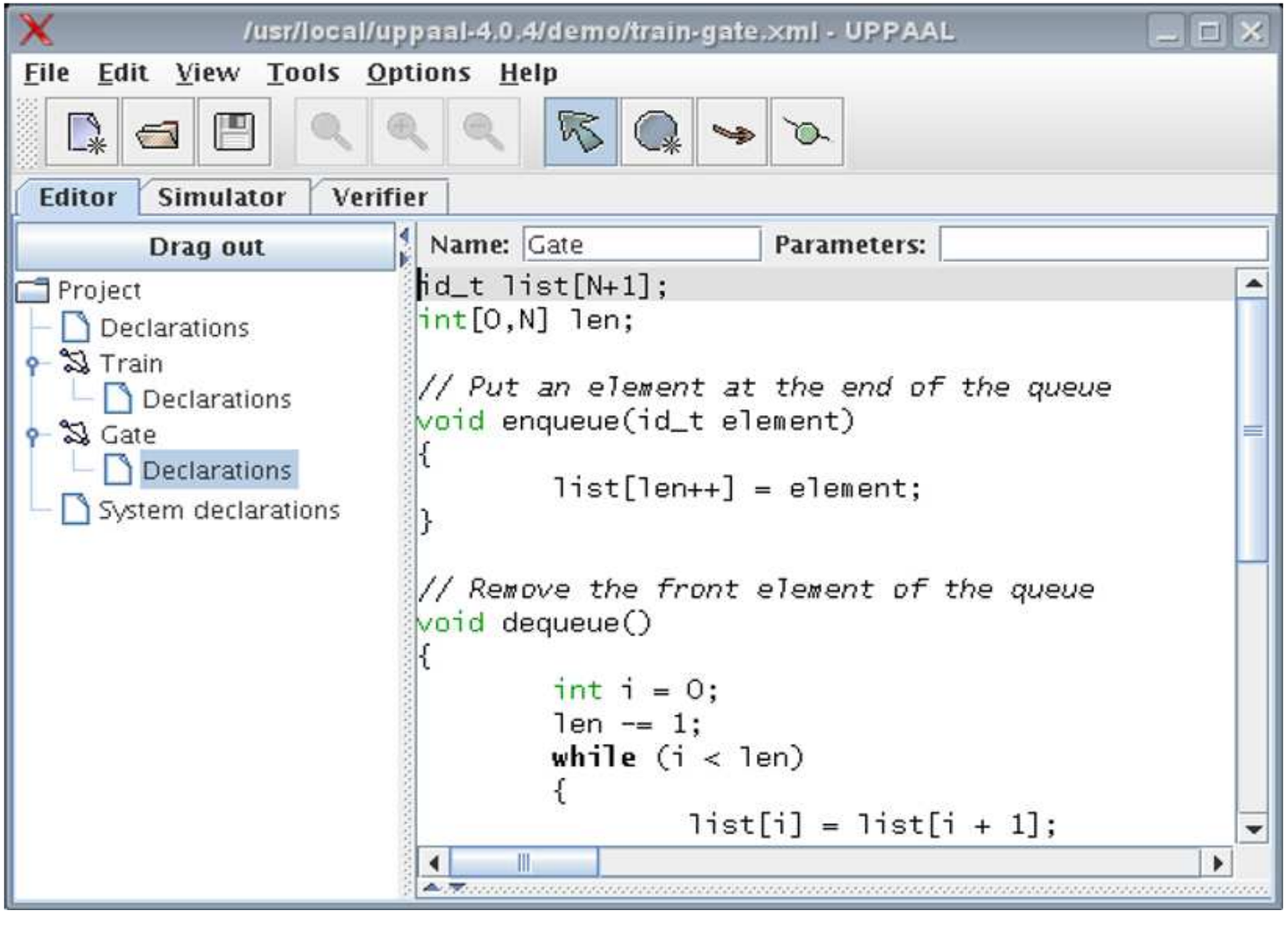
Die Modellierungssprache von UPPAAL

- Definition der TAs über grafische Modelle.
- Jedes Modell besitzt einen Deklarationsteil, indem Variablen (u.ä.) definiert werden können.
- Das graphische Modell kann Variablen (u.ä.) aus dem Deklarationsteil in Wächtern, Zuweisungen, Invarianten und bei Synchronisationen nutzen.

Die Modellierungssprache von UPPAAL



Die Modellierungssprache von UPPAAL



Die Modellierungssprache von UPPAAL

- UPPAAL-TA-Erweiterungen im Detail:

1. Konstanten, begrenzte Ganzzahlen (Integer), Felder, Funktionen:
   - Konstanten etc. werden im Deklarationsteil definiert
   - C-ähnliche Syntax:

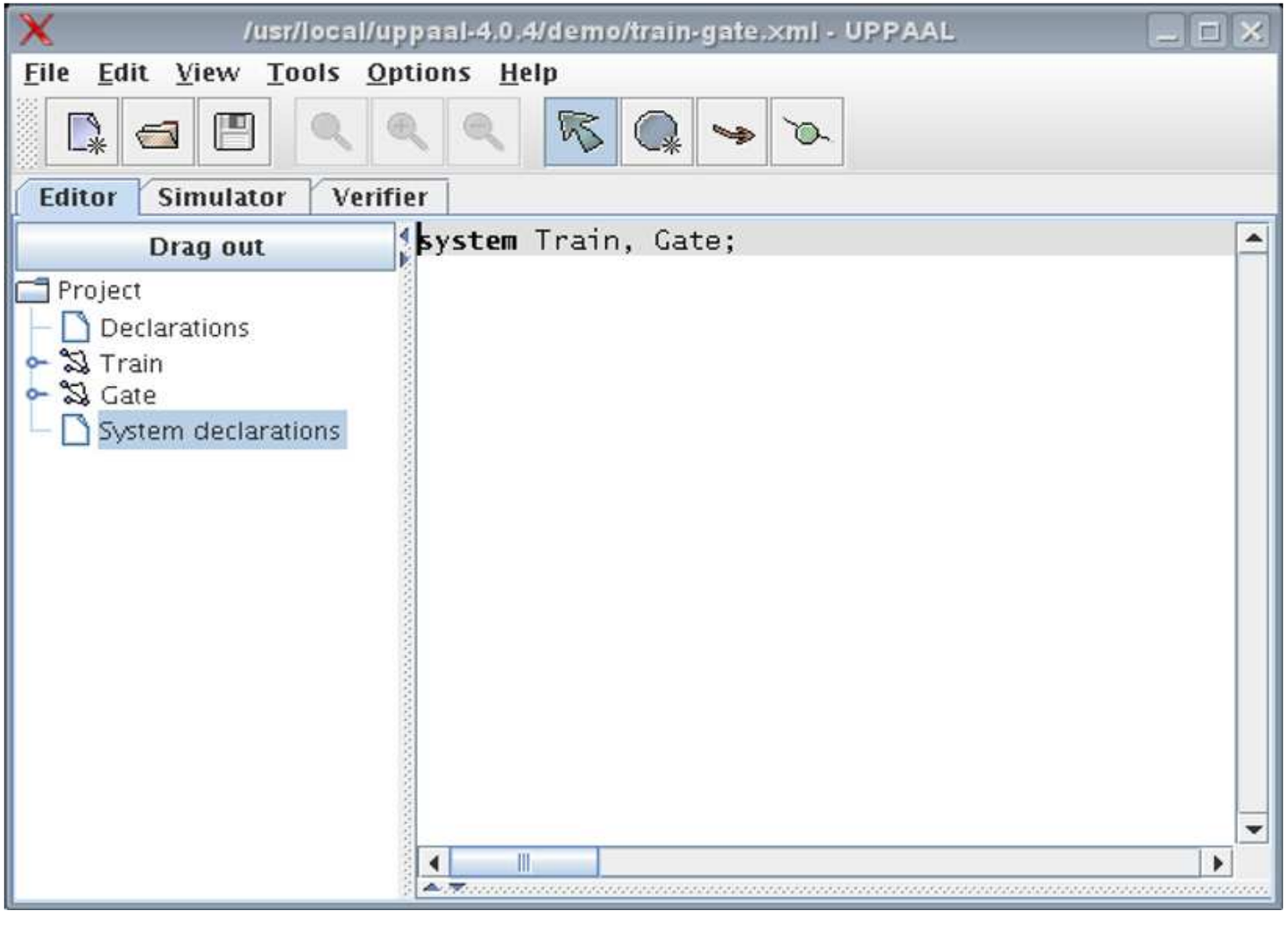
```
const min 3; const max 7;
clock c; int[min,max] myIntArray[12];
int add(int a, int b) { return a + b; }
```

Die Modellierungssprache von UPPAAL

- UPPAAL-TA-Erweiterungen im Detail:

2. Schablonen (Templates):
   - Keine direkte Erstellung von TAs sondern Erstellung von Schablonen.
   - Schablonen können Parameter besitzen.
   - Instanziierung der Schablonen zu TAs (in UPPAAL auch Prozess genannt) in speziellem Systemdeklarationsteil.
   - Beispiel: `TrainA = Train(0); TrainB = Train(1);` `system TrainA, TrainB;`

Die Modellierungssprache von UPPAAL



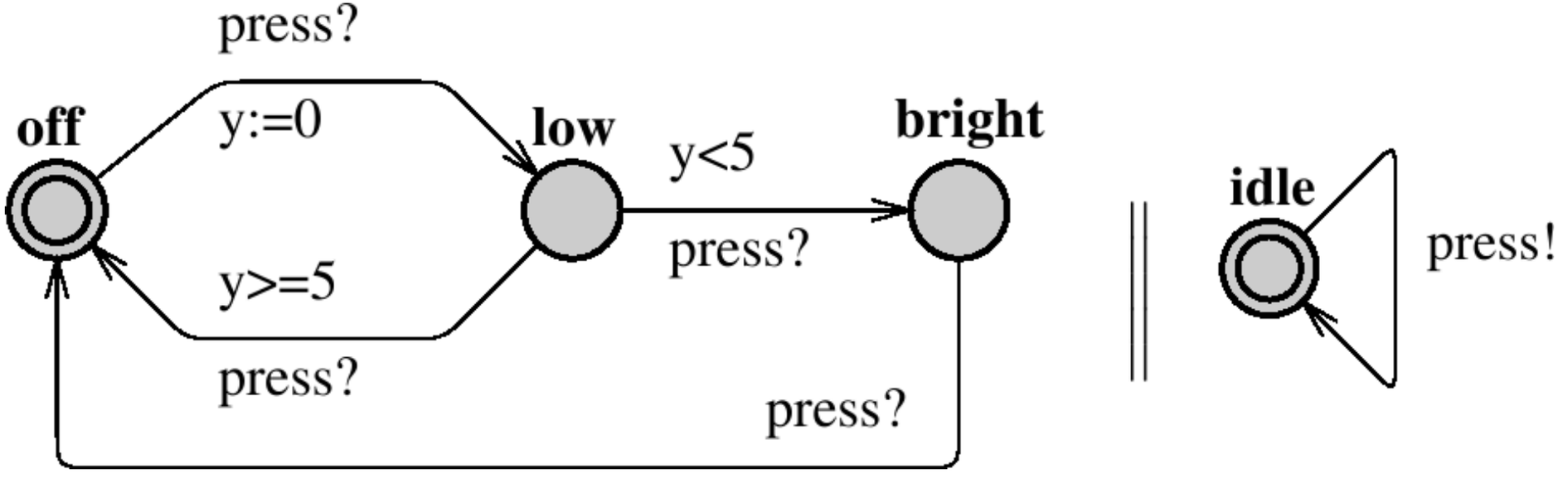
Die Modellierungssprache von UPPAAL

- UPPAAL-TA-Erweiterungen im Detail:

  3. Binäre Kanäle
     - Dienen zur Synchronisation zweier TAs.
     - Deklaration: `chan c`
     - Mit `c!` beschriftete Kante synchronisiert sich mit `c?` beschrifteter Kante.
     - Synchronisationspaare werden nichtdeterministisch ausgewählt.

Die Modellierungssprache von UPPAAL

- Beispiel binäre Synchronisation:



Die Modellierungssprache von UPPAAL

- UPPAAL-TA-Erweiterungen im Detail:

4. Broadcast-Kanäle
   - Dienen zur Synchronisation von mehreren TAs
   - Deklaration: `broadcast chan c`
   - Sender `c!` kann sich mit beliebiger Anzahl von Empfängern `c?` synchronisieren.
   - Jeder Empfänger, der sich mit dem Sender synchronisieren kann, muß dies tun.
   - Sender kann Kante `c!` auch ohne Empfänger nehmen.

Die Modellierungssprache von UPPAAL

- UPPAAL-TA-Erweiterungen im Detail:

  5. Dringende (urgent) Kanäle
     - Sobald Kanal "aktiviert" wird, muß dieser auch ohne Zeitverzögerung genommen werden.
     - Kanten, die dringende Kanäle verwenden, dürfen keine Uhr-Wächter besitzten.
     - Schlüsselwort: `urgent`

Die Modellierungssprache von UPPAAL

- UPPAAL-TA-Erweiterungen im Detail:

  6. Dringende (urgent) Lokalisationen
     - Sobald Lokalisation besucht wird, muss diese auch ohne Zeitverzögerung wieder verlassen werden.
     - Äquivalent: Hinzufügen einer Uhr x, die auf allen Eingangskanten auf Null gesetzt wird, und hinzufügen der Invariante $x \leq 0$ zur Lokalisation.
     - Symbol:

Die Modellierungssprache von UPPAAL

- UPPAAL-TA-Erweiterungen im Detail:

7. Verpflichtete (commited) Lokalisationen
   - Zustand verpflichtet, falls dieser verpflichtete Lokalisation enthält.
   - Sobald Zustand verpflichtet, muss der nächste Zustandsübergang Ausgangskante einer verpflichteten Lokalisation nutzen.
   - Restriktiver als dringende Lokalisation.
   - Symbol: ©

Die Abfragesprache von UPPAAL

- Untermenge von CTL (*Computation Tree Logic* – verzweigende temporale Logik/modale Aussagenlogik)
- Syntax:
  `E[]` φ, `A[]` φ, `E<>` φ, `A<>` φ oder $\psi \rightsquigarrow \varphi$,
  wobei φ, ψ *Zustandsformeln*.
- Im Gegensatz zu CTL keine Verschachtelung von modalen Operatoren möglich.

Die Abfragesprache von UPPAAL

- Zustandsformel:
  - Seiteneffekt freier Ausdruck, der nach `boolean` auswertet (mit gewissen weiteren Einschränkungen); kann
    - Prozess-Ausdruck (z. B. `P.s1`) oder
    - Deadlock-Ausdruck (`deadlock`) enthalten.
  - Beispiel: `(P.s1 and (P.i == 7)) or (clock > 200)`
- Deadlock-Ausdruck nicht mit dem Operator $\rightsquigarrow$ verwendbar.

Die Abfragesprache von UPPAAL

- Definition Ausdruck:

```
Expr ::= ID
       | NAT
       | Expr '[' Expr ']'
       | '(' Expr ')'
       | Expr '++' | '++' Expr
       | Expr '--' | '--' Expr
       | Expr Assign Expr
       | Unary Expr
```

Die Abfragesprache von UPPAAL

- (Vorsetzung Definition Ausdruck:)

```
| Expr Binary Expr
| Expr '?' Expr ':' Expr
| Expr '.' ID
| Expr '(' Args ')'
| 'forall' '(' ID ':' Type ')' Expr
| 'exists' '(' ID ':' Type ')' Expr
| 'deadlock' | 'true' | 'false'
```

Die Abfragesprache von UPPAAL

- (Vorsetzung Definition Ausdruck:)

```
Args :  := [ Expr ( ',' Expr )* ]
Assign ::= '=' | ':=' | '+=' | '-=' | '*=' | '/='
         | '|=' | '&=' | '≙' | '<<=' | '>>='
Unary : := '+' | '-' | '!' | 'not'
Binary ::= '<' | '<=' | '==' | '!=' | '>=' | '>'
         | '+' | '-' | '*' | '/' | '%' | '&'
         | '|' | '^' | '<<' | '>>' | '&&' | '||'
         | '<?' | '>?' | 'or' | 'and' | 'imply'
```

Die Abfragesprache von UPPAAL

Semantik:

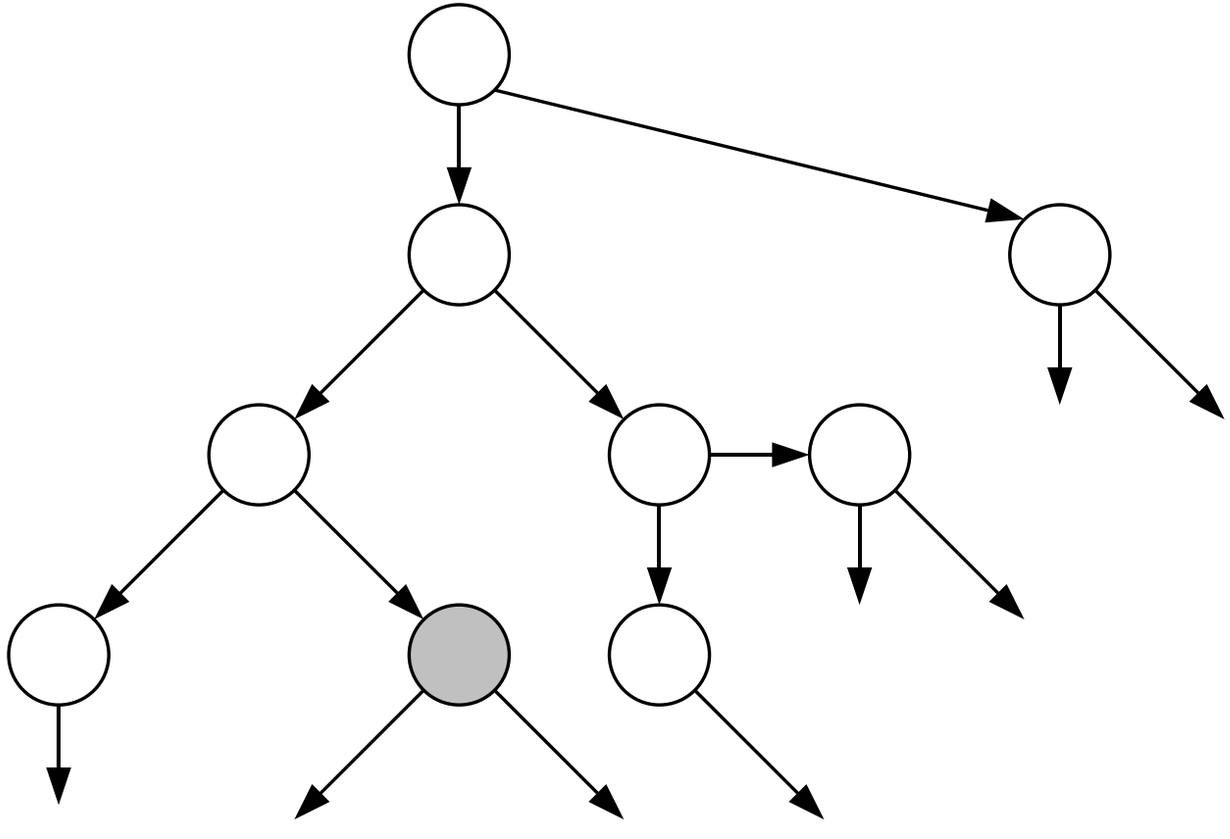
- Grundlage spezielle Kripke Strukturen (G, R, I), wobei
  - R partielle Ordnung auf G,
  - für alle $g \in G$ existiert direkter Nachfolger und
  - (G, R) Wurzelbaum
- (G, R, I) heißt auch Berechnungsbaum.
- Festlegung des Berechnungsbaums über das jeweilige TA-Netzwerk; Wurzel ist dabei der eindeutige Startzustand.

Die Abfragesprache von UPPAAL

- Zustandsformeln φ werden in einer Welt $g \in G$ mit Hilfe der Interpretation I wie üblich ausgewertet.

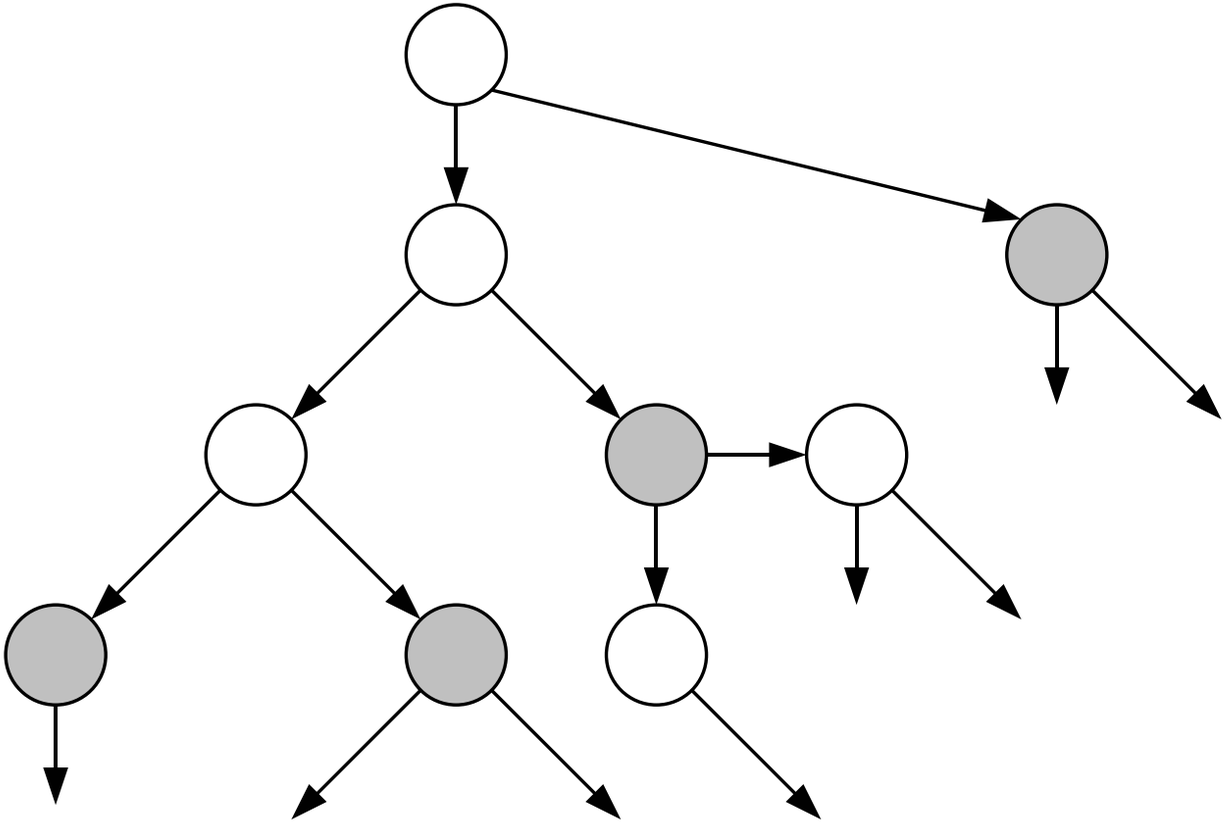
Die Abfragesprache von UPPAAL

- Auswertung von E<>φ:



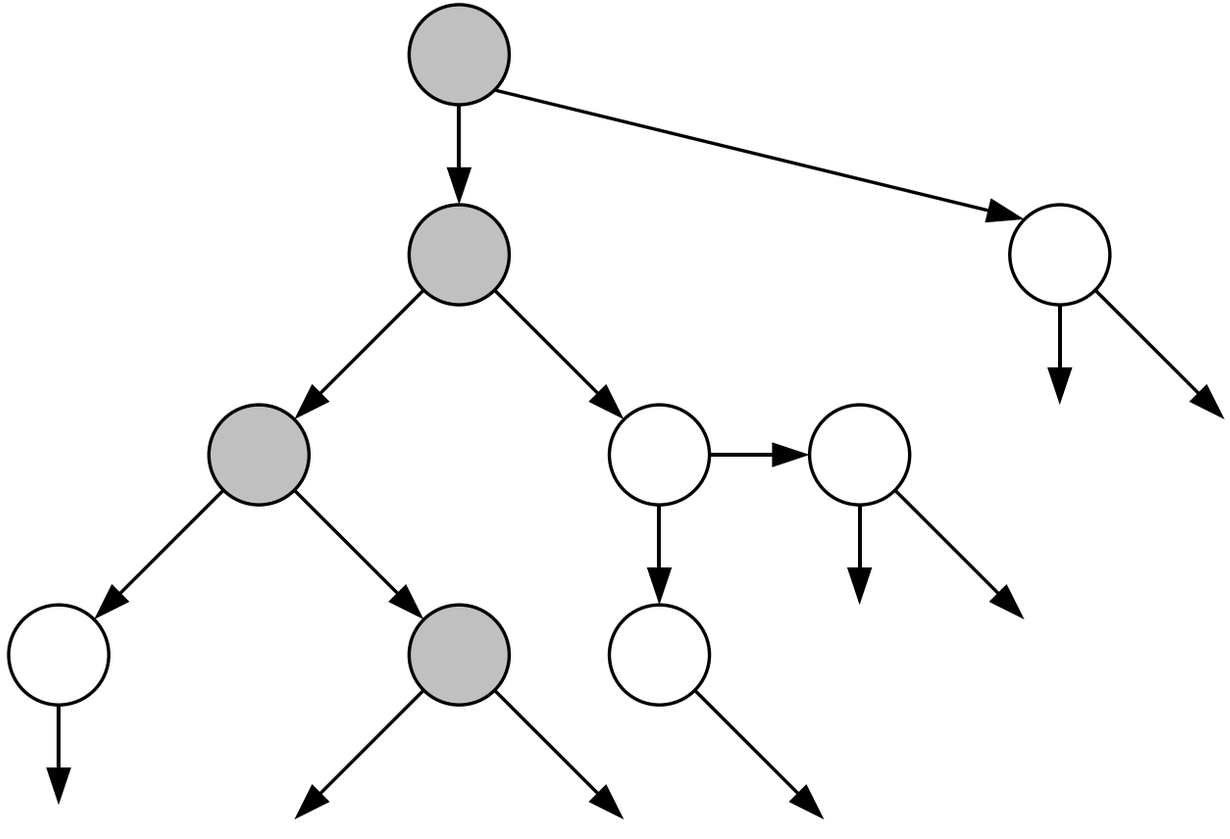
Die Abfragesprache von UPPAAL

- Auswertung von A<>φ:



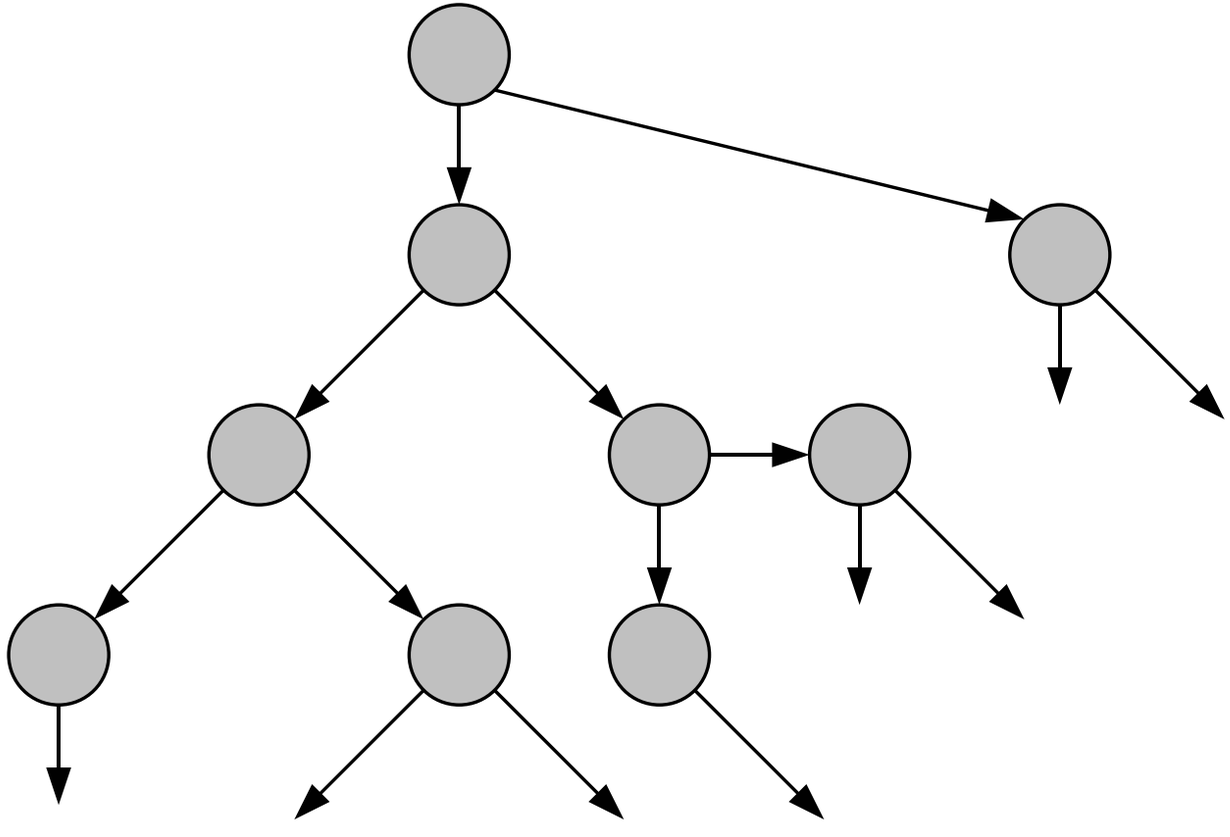
Die Abfragesprache von UPPAAL

- Auswertung von $\mathtt{E[]}\,\varphi$:



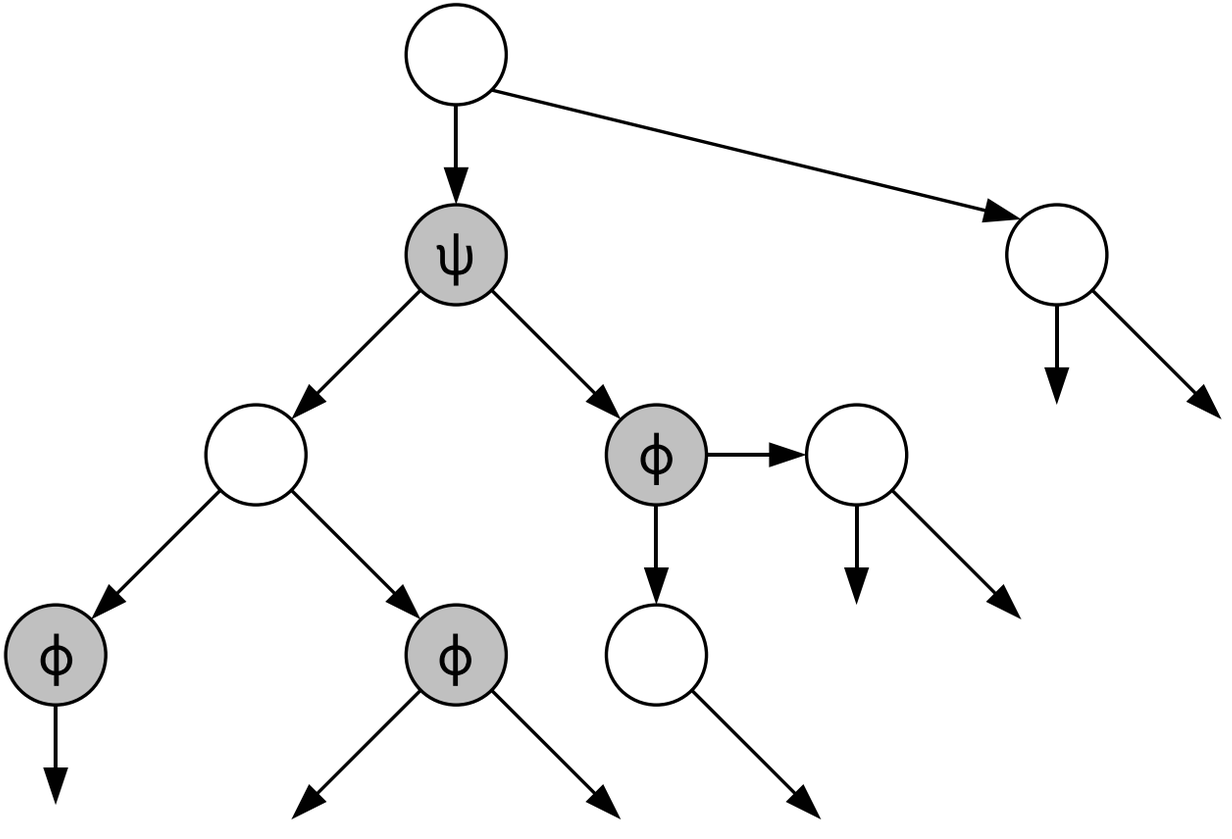
Die Abfragesprache von UPPAAL

- Auswertung von $\texttt{A[]}\,\varphi$:



Die Abfragesprache von UPPAAL

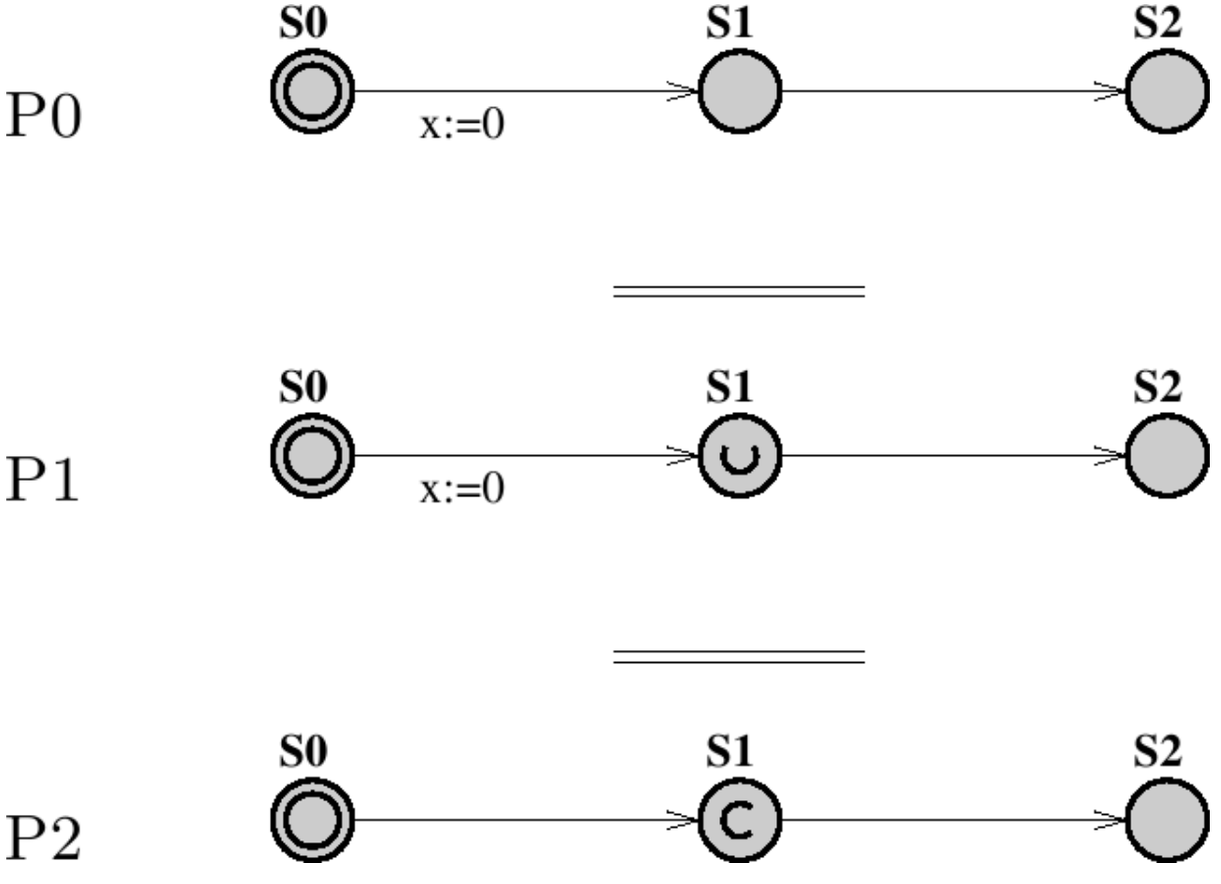
- Auswertung von $\psi \rightsquigarrow \phi$:



Überblick über das Toolkit

Beispiel

- Vergleich normale, dringende, verpflichtete Lokalisationen



Beispiel

- Folgende Eigenschaften sind erfüllt:
  - `E<> P0.S1 and P0.x > 0`, d. h. es ist erlaubt in `P0.S1` zu warten.
  - `A[] P1.S1 imply P1.x == 0`, d. h. es ist nicht möglich in `P1.S1` zu warten.
- Verpflichtete Lokalisationen restriktiver als dringende Lokalisationen:
  Sobald `P2.S1` aktiv, muss als nächstes ohne Verzögerung Übergang nach `P2.S2` genommen werden.

Bemerkungen

- Auch in Uppaal gibt es Entwurfsmuster!
- Das Zeitmodell von Uppaal ist kontinuierlich, auch wenn Uhren nur mit Ganzzahlen (integern) vergleichbar sind.

Ende

Vielen Dank für Ihre Aufmerksamkeit!