

# Verifikace Modelů a UPPAAL

Radek Mařík

ČVUT FEL, K13132

October 2, 2014



## Obsah

- 1 Úvod
  - Motivace
  - Úvod do verifikace modelů
  - Formální popis
- 2 UPPAAL - Přehled
  - Základní vlastnosti
  - Architektura
- 3 UPPAAL - Vybrané vlastnosti
  - Systém a proces
  - Manuál



# Garantování správného chování systémů <sup>[Cam10]</sup>

- zvyšování složitosti softwarových systémů,
- chyby vedou ke ztrátám
  - finančním,
  - na lidských životech.
- typický problém systémů s kritickou bezpečností
  - letadla,
  - satelity,
  - lékařská zařízení.



# Cíle kladené na požadavky <sup>[Cam10]</sup>

- Demonstrace, že požadavky jsou
  - správné,
  - úplné,
  - přesné,
  - konzistentní,
  - testovatelné.



# Metody zajištění kvality <sup>[Cam10]</sup>

- **Testování a Simulace** poskytuje pouze pravděpodobnostní zajištění.
- **Verifikace za běhu** ... technika kombinující formální verifikaci s během programu.
- **Formální verifikace** ... technika založená na formálních metodách stavějící na matematicky založených jazycích, které umožňují specifikaci a verifikaci systémů.
  - **Specifikace** ... zapsání požadavků na systém v matematickém jazyku.
  - **Verifikace** ... formální důkaz toho, že systém splňuje požadavky.



# Princip formální verifikace <sup>[Cam10, ?]</sup>

## Vstupy

- (matematický) model systému,
  - formální model  $M$ ,
- specifikace požadavků kladených na systém
  - formule  $\varphi$  určité temporální logiky,

## Verifikace

- Ověření, že systém splňuje specifikaci.
  - rozhodnutí, zda-li  $M$  je modelem formule  $\varphi$ , tj.  $M \models \varphi$



## Techniky

- **Statická analýza** . . . ověření chování programu, aniž by se musel spustit.
  - **Abstraktní statická analýza** . . . založená na *abstraktní interpretaci* používající aproximační abstraktní reprezentace k ověřování přibližných vlastností složitých systémů
    - analýza ukazatelů v moderních kompilátorech.
  - **Ověřování modelů** . . . úplné procházení dosažitelných stavů programu.
  - **Omezené ověřování modelů** . . . úplné procházení dosažitelných stavů programu pouze do určité hloubky.
- **Dokazování vět** . . . nalezení důkazu vlastnosti, kdy systém i jeho vlastnosti jsou vyjádřeny jako formule v nějaké matematické logice.



## Ověřování modelů

- Ptáme se, zda daný systém splňuje požadovanou vlastnost.
- Tj. pro strukturu reprezentující systém je třeba zjistit, jestli je modelem zadané formule.
- využitelné pro verifikaci existujících programů.

## Splnitelnost formulí

- Problém rozhodnutí, zda existuje nějaký model zadané formule.
- využitelné při automatické syntéze programů.



# Verifikace modelů <sup>[Cam10]</sup>

## Princip

- budování konečného modelu systému,
- kontrola, zda požadovaná vlastnost je modelem dodržena,
- založeno na úplném prohledání stavového prostoru.

## Základní vlastnosti

- manipulace s obrovskými prohledávacími prostory,
- odpověď je "ano" či "ne", v záporném případě systém poskytuje
  - protipříklad, tj. běh systému, který neodpovídá vlastnosti.
- analýza specifikace softwarových systémů.



# Verifikace modelů v praxi <sup>[Cam10]</sup>

## Aplikace

- ověření hardwaru (obvody),
- ověření protokolů,
- analýza specifikace softwarových systémů.



## Přístupy verifikace modelů [Cam10]

### Temporální verifikace modelů

- použití temporální logiky (vyjádření času),
- systémy modelovány jako přechodové systémy s konečným počtem stavů.

### Automatový přístup

- specifikace i model vyjádřen jako automaty,
- oba automaty se porovnávají
  - jazyková inkluze,
  - zjemňující uspořádání,
  - pozorovací ekvivalence.



## Výhody/nevýhody verifikace modelů [Cam10]

### Výhody

- úplná automatizace,
- vysoká rychlost,
- možnost verifikace i částečných specifikací,
- produkuje protipříklady.

### Nevýhody

- problém exploze stavů,
  - binární rozhodovací diagramy (BDD),
  - nástroje jsou schopny zvládnout systémy s 100 – 200 stavovými proměnnými
  - je možné zvládnout systémy s  $10^{120}$  stavy.



## Rozšíření metod verifikace modelů [Bie08]

## Odstraňování konečnosti

- spojité proměnné,
- spojitý čas,
- práce s pravděpodobností,
- parametrizace velikosti či počtu komponent,
- náhrada konečných automatů zásobníkovými automaty.



## Temporální logika [Bie08]

## Vyšetření sekvenčního či temporálního chování systému

- reaktivní, distribuované či paralelní systémy,
- A. Pnueli upozornil na tuto myšlenku jako první,

## Ověřované vlastnosti

- **Bezpečnost** ... vlastnost stanovující, že určitá chyba či katastrofický stav není dosažitelný.
  - všechny dosažitelné stavy splňují určitý invariant.
- **Živost** ... něco jednou nastane,
- **Férovost** ... ,



# Stavový prostor <sup>[?]</sup>

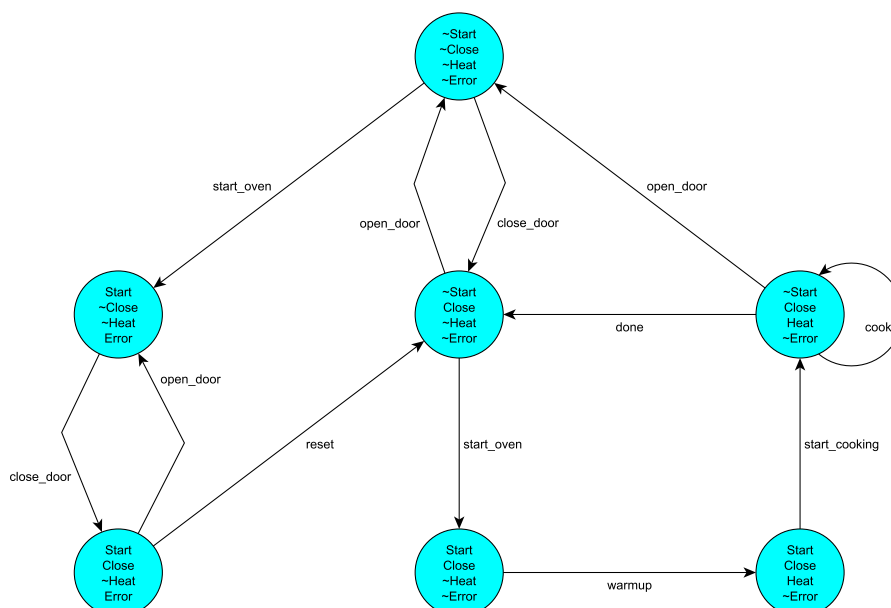
## Aplikovatelné jen na konečné stavové prostory

- Verifikovat se dají pouze ty parametry modelu, které jsou specifikovány.
- Stavový prostor lze formalizovat za použití atomických výroků a Kripkeho struktury

## Atomické výroky

- základní tvrzení popisující daný systém
  - výrazy,
  - konstanty,
  - predikátové symboly.
- Každý atomický výrok je algoritmicky rozhodnutelný na základě daného stavu.
- Stav ... ohodnocení všech proměnných.

# Kripkeho struktura - mikrovlnná trouba <sup>[?]</sup>





# Kripkeho struktura <sup>[?]</sup>

Kripkeho struktura je typ nedeterministického konečného automatu.

## Kripkeho struktura

- Je dána množina atomických propozic  $AP$ .
- Kripkeho struktura je trojice  $(S, T, \mathcal{I})$ , kde
  - $S$  je konečná množina stavů,
  - $T \subseteq S \times S$  je přechodová relace,
  - $\mathcal{I} : S \rightarrow 2^{AP}$  je interpretace  $AP$ .

## Rozšířená Kripkeho struktura

- je čtveřice  $(S, T, \mathcal{I}, s_0)$ , kde
  - $(S, T, \mathcal{I})$  je Kripkeho struktura,
  - $s_0$  je počáteční stav.



# Kripkeho přechodový systém <sup>[?]</sup>

Máme-li danou množinu  $Act$  akcí proveditelných programem, můžeme Kripkeho strukturu rozšířit o označení přechodu.

## Kripkeho přechodový systém

- je pětice  $(S, T, \mathcal{I}, s_0, L)$ , kde
  - $(S, T, \mathcal{I}, s_0)$  je rozšířená Kripkeho struktura,
  - $L : T \rightarrow Act$  je značkovácí funkce.



## Nástroj v kostce <sup>[?]</sup>

### Nástroj integrující prostředí

- pro modelování,
- simulaci,
- a verifikaci,
- reálných systémů.

### Vývojové týmy

- **Uppsala University**, Švédsko,
- **Aalborg University**, Dánsko.



## Modely systémů <sup>[?]</sup>

### Vlastnosti modelů

- sada nedeterministických procesů
- s konečnou řídicí strukturou a
- reálnými hodinami,
- komunikující pomocí kanálů nebo
- sdílených proměnných



# Implementace <sup>[?]</sup>

## Hlavní návrhová kritéria

- výkonnost,
  - vyhledávací stroj *za letu*
  - symbolické techniky
- snadno použitelné.
- diagnostický záznam
  - může být generován verifikátorem a přehráván simulátorem

## Dostupnost

- První verze v roce 1995
- Současná verze je 4.0.12
- grafická rozhraní jsou implementována v Java
- verifikátor je implementován v C++
- dostupné pro Linux, SunOS, MS Windows (95/98/NT/2000/XP/Vista/7)

# Průmyslové studie <sup>[?]</sup>

## Případové studie

- audio/video protokol
  - komunikace mezi audio/video komponentami pomocí jediné sběrnice
- protokol vysílání s ohraničeným opakováním,
- protokol pro vyhnutí se kolizím
  - média založená na Eternetu
- řadič spojky automobilů,
- protokol řízení audio komponent (Philips)
- TDMA(Time Division Multiple Access) protokol mechanismu start-up
  - synchronizace 3 komunikujících stanic s libovolného počátečního stavu.

## Typické aplikace

- řadiče reálného času,
- komunikační protokoly.

# Komponenty systému <sup>[?]</sup>

## Jazyk popisu

- jazyk nedeterministických podmíněných příkazů
- jednoduché datové typy (ohrazená celá čísla, pole, atd.)
- sítě automatů s hodinami a datovými proměnnými.

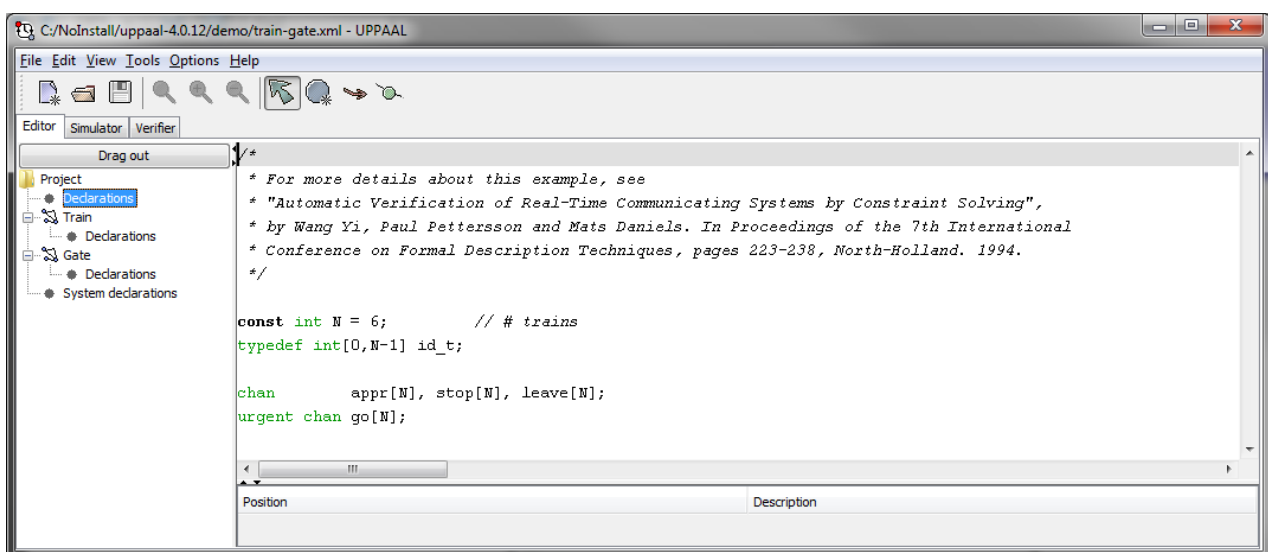
## Simulátor

- vyšetřování možných dynamických běhu nějakého systému,
- detekce vad modelů před jeho verifikací,
- umožňuje analýzu záznamů běhů vedoucích k nežádaným stavům.

## Verifikátor modelu

- prověření všech možností dynamického chování modelu,
- kontrola invariantů a živosti prohledáváním stavového prostoru,
- dosažitelnost symbolických stavů reprezentovaných omezeními.

# Systemový editor <sup>[?]</sup>

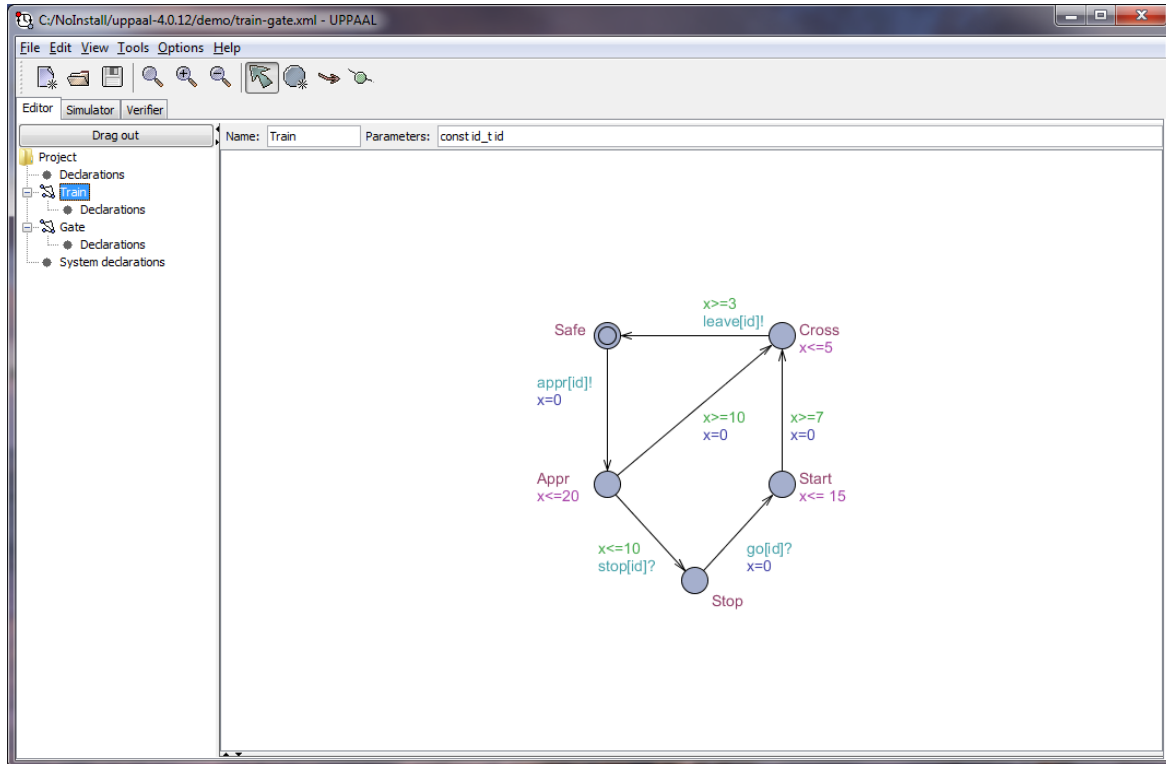


## Editor

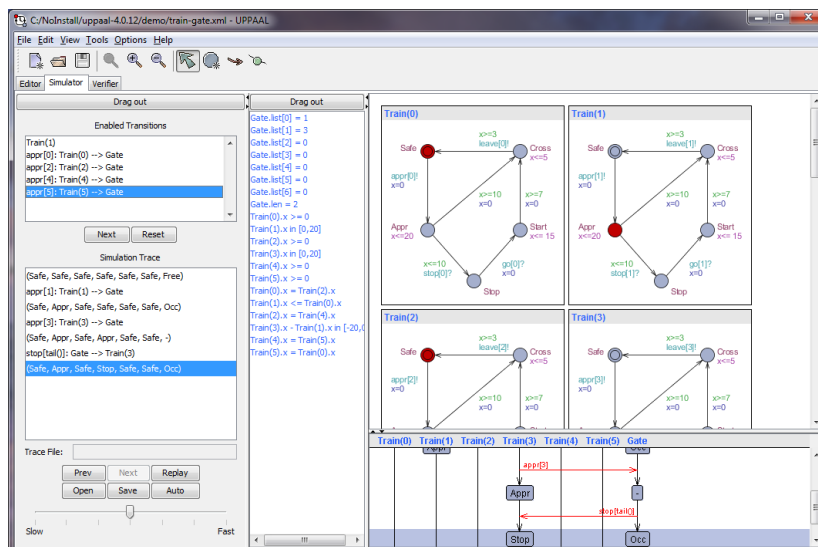
- tvorba grafického i textového popisu systémů



## Grafický systémový editor [?]

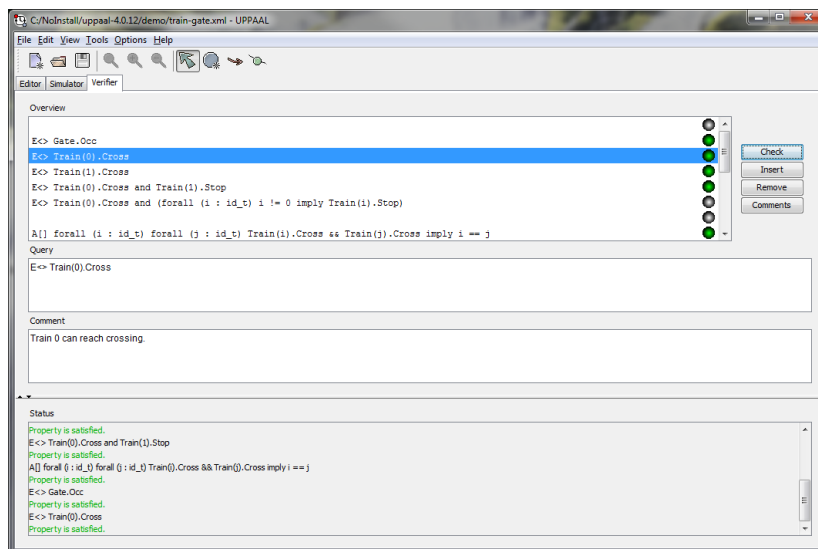


## Grafický simulátor [?]



## Simulátor

- grafická vizualizace a záznam možného dynamického chování popisu systému,
- sekvence symbolických stavů systému,
- možnost vizualizace trasy generované verifikátorem.

Verifikátor <sup>[?]</sup>

## Verifikátor

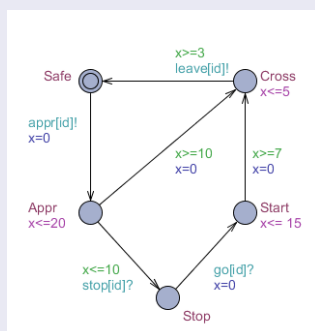
- Editor specifikace požadavků,
- Stroj verifikátoru modelu
  - automatické ověření živosti a ohraničené živosti pomocí dosažitelnosti v symbolickém stavovém prostoru.

Výchozí principy <sup>[?]</sup>

## Model

- Časový automaty
  - konečný stavový automat s hodinami,
  - čas je spojitý,
  - hodiny měří postup času.

## Vzory procesů - Automat



- pozice a hrany,
- symbolické proměnné a konstanty jako parametry,
- lokální proměnné a hodiny,
- daný proces je pak instancí VZORU.

# Časový automat <sup>[BDL05]</sup>

## Časový automat

- je šestice  $(L, \ell_0, C, A, E, \mathcal{I})$ , kde
  - $L$  je množina pozic,
  - $\ell_0 \in L$  je počáteční pozice,
  - $C$  je množina hodin.
  - $A$  je množina akcí, ko-akcí a interní  $\tau$ -akce,
  - $E \subseteq L \times A \times B(C) \times 2^C \times L$  je množina hran mezi pozicemi s akcí, stráží a množinou hodin, které se resetují, a
  - $\mathcal{I} : L \rightarrow B(C)$  přiřazuje invarianty k pozicím.

## Příklady

- $y := 0$  ... resetování hodin  $y$ ,
- *press?* a *press!* ... označují akci a ko-akci (zde kanálovou synchronizaci).



# Hodiny časového automatu <sup>[BDL05]</sup>

## Hodiny

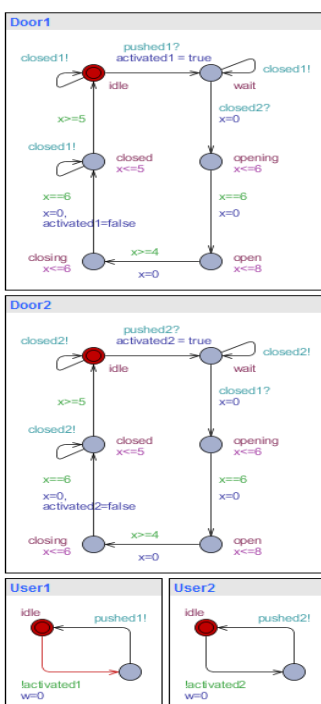
- **Ohodnocení hodin** je funkce  $u : C \rightarrow \mathbb{R}_{\geq 0}$  z množiny hodin do nezáporných reálných čísel.
- Nechť  $\mathbb{R}^C$  je množina všech ohodnocení hodin.
- Nechť  $u_0(x) = 0$  pro všechna  $x \in C$ .
- Zápis  $u \in \mathcal{I}(\ell)$  bude znamenat, že  $u$  splňuje  $\mathcal{I}(\ell)$ .
- Z daného stavu je možné provést přechod pomocí *akce* nebo *zpoždění*.



Sémantika časového automatu <sup>[BDL05]</sup>

## Sémantika časového automatu

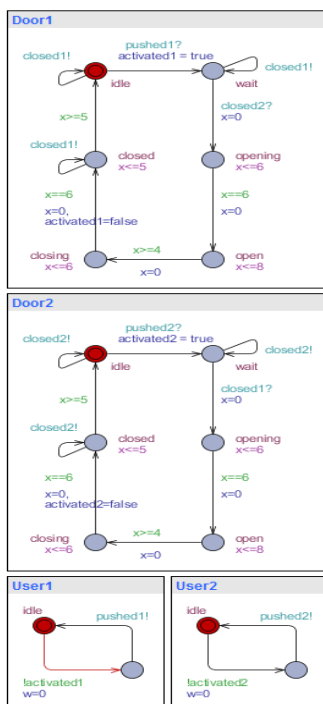
- Nechť  $(L, l_0, C, A, E, \mathcal{I})$  je časový automat.
- Sémantika ... přechodový systém s označením  $\langle S, s_0, \rightarrow \rangle$ , kde
- $S \subseteq L \times \mathbb{R}^C$  je množina stavů,
- $s_0 = (l_0, u_0)$  je počáteční stav,
- $\rightarrow \subseteq S \times (\mathbb{R}_{\geq 0} \cup A) \times S$  je přechodová relace taková, že
  - $(l, u) \xrightarrow{d} (l, u + d)$  if  $\forall d' : 0 \leq d' \leq d \implies u + d' \in \mathcal{I}(l)$
  - $(l, u) \xrightarrow{a} (l', u')$  if  $\exists e = (l, a, g, r, l') \in E$   
 $| e \in g, u' = [r \mapsto 0]u, u' \in \mathcal{I}(l')$ ,
- $u + d$  zobrazuje každé hodiny  $x \in C$  na hodnotu  $u(x) + d$ , pro  $d \in \mathbb{R}_{\geq 0}$ ,
- $[r \mapsto 0]u$  označuje ohodnocení hodin, která mapuje každé hodiny  $v$  na 0 a souhlasí s  $u$  nad  $C \setminus r$ .

Specifikace systému <sup>[?]</sup>

## Systém je kompozice souběžných procesů

- každý process je modelován jako automat
- automat má množinu pozic,
- změny pozic se dějí pomocí hran/přechodů.
- stav systému je charakterizován pomocí aktuální
  - pozice každého automatu,
  - hodnot proměnných, a
  - stavu hodin.
- přechody je možné řídit pomocí stráží a synchronizací
- stráž je podmínka nad proměnnými a hodinami specifikující, kdy je přechod možný.



Komunikace procesů <sup>[?]</sup>

## Synchronizace

- synchronizace je mechanismus kdy dva procesy provedou současný přechod
  - 1 synchronizační kanál  $a$ ,
  - 2 první proces vyvolá přechod zprávou  $a!$
  - 3 druhý proces provede přechod přijetím zprávy  $a?$
- během přechodu je možné přiřadit do proměnných nebo resetovat hodiny.

Sít časových automatů <sup>[BDL05]</sup>

## Sada automatů

- Společná množina hodin a akcí.
- $n$  časových automatů  $\mathcal{A}_i = (L_i, \ell_i^0, C, A, E_i, \mathcal{I}_i)$ ,  $1 \leq i \leq n$
- poziční vektor  $\bar{\ell} = (\ell_1, \dots, \ell_n)$
- společná funkce invariantů  $\mathcal{I}(\bar{\ell}) = \bigwedge_i \mathcal{I}_i(\ell_i)$
- $\bar{\ell}[l'_i/l_i]$  ...  $i$ -tý element  $\ell_i$  vektoru  $\bar{\ell}$  je nahrazen  $\ell'_i$



Sémantika sítě časových automatů <sup>[BDL05]</sup>

## Sít časových automatů

- $n$  časových automatů  $\mathcal{A}_i = (L_i, \ell_i^0, C, A, E_i, \mathcal{I}_i)$
- počáteční vektor pozic  $\bar{\ell}^0 = (\ell_1^0, \dots, \ell_n^0)$
- Sémantika ... přechodový systém s označením  $\langle S, s_0, \rightarrow \rangle$ , kde
- $S \subseteq (L_1 \times \dots \times L_n) \times \mathbb{R}^C$  je množina stavů,
- $s_0 = (\bar{\ell}_0, u_0)$  je počáteční stav,
- $\rightarrow \subseteq S \times S$  je přechodová relace taková, že
  - $(\bar{\ell}, u) \xrightarrow{d} (\bar{\ell}, u + d)$  if  $\forall d' : 0 \leq d' \leq d \implies u + d' \in \mathcal{I}(\bar{\ell})$ , a
  - $(\bar{\ell}, u) \xrightarrow{a} (\bar{\ell}[l'_i/l_i], u')$  if  $\exists l_i \xrightarrow{\tau g_i} l'_i$   
 $| u \in g, u' = [r \mapsto 0]u, u' \in \mathcal{I}(\bar{\ell}[l'_i/l_i])$ ,
  - $(\bar{\ell}, u) \xrightarrow{a} (\bar{\ell}[l'_j/l_j, l'_i/l_i], u')$  if  $\exists l_i \xrightarrow{c?g_i r_i} l'_i$  a  $l_j \xrightarrow{c!g_j r_j} l'_j$   
 $| u \in (g_i \wedge g_j), u' = [r_i \cup r_j \mapsto 0]u, u' \in \mathcal{I}(\bar{\ell}[l'_j/l_j, l'_i/l_i])$ ,

Typy jazyka <sup>[BDL05]</sup>

## Typy

- **Konstanty** ... const name value, celočíselná hodnota.
- **Omezené celočíselné hodnoty** ... int[min, max] name, výchozí nastavení -32768 až 32768.
- **Pole** ... hodiny, kanály, konstanty, celočíselné proměnné  
chan c[4]; clock a[2]; int[3,5] u[7];.
- **Iniciátory** ... nastavení hodnot celočíselných proměnných a polí s celočíselnými proměnnými  
int i := 2; int k[3] := {1, 2, 3};.



Speciální přechody <sup>[BDL05]</sup>

## Řídící elementy

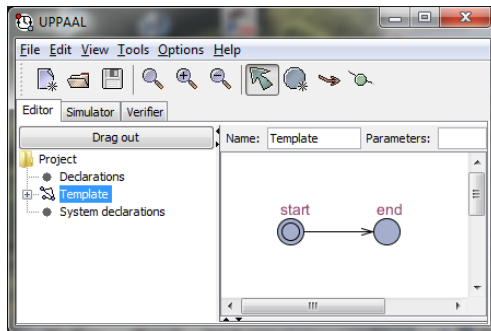
- **Binární synchronizace** ... `chan c`, hrany `c!` a `c?`, nedeterministicky pár.
- **Broadcast synchronizace** ... `broadcast chan c`, jedna hrana `c!` se všemi možnými `c?`, neblokuje.
- **Urgentní synchronizace** ... `urgent chan c`.
  - Zpoždění není dovoleno, pokud je možný přechod na urgentním kanálu.
- **Urgentní pozice** ... Čas systému nemůže plynout, pokud se systém nachází v urgentní pozici.
- **Prováděcí pozice**
  - **Prováděcí stav** ... alespoň jedna z pozic je prováděcí.
  - Prováděcí stav se nemůže zpožďovat.
  - Následující přechod musí zahrnovat jednu výstupní hranu vedoucí z prováděcí pozice.

Výrazy jazyka <sup>[BDL05]</sup>

## Hodiny, celočíselné proměnné a konstanty

- **Stráž** ... výsledkem je logická hodnota.
- **Synchronizace** ... synchronizační návěští `Expression!` nebo `Expression?` nebo prázdné.  
Výsledkem je kanál.  
Odkazovat může celá čísla, konstanty, kanály.
- **Přiřazení** ... výrazy oddělená čárkou.  
Odkazovat může hodiny, celočíselné proměnné, konstanty.  
Hodinám může přiřadit pouze celočíselné hodnoty.
- **Invariant** ... Konjunkce podmínek tvaru  $x < e$  nebo  $x \leq e$ , kde
  - $x$  je odkaz na hodiny,
  - $e$  se vyčíslí do celého čísla.
 Odkazovat může hodiny, celočíselné proměnné, konstanty.

# Tvorba automatu <sup>[?]</sup>



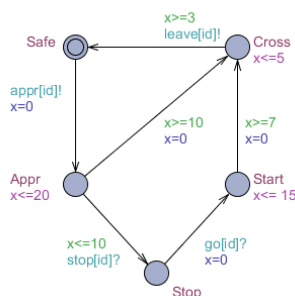
## Automat

- počáteční pozice (dvojitá kružnice)
- "Add Location" pro přidání pozice
- "Selection Tool" pro pojmenování pozice
- "Add Edge" pro přidání hrany, prohnutí hran pomocí myši v okolí konců
- dolní tabulka "Position" a "Description" pro analýzu chyb

# Kompozice systému <sup>[?]</sup>

## System

- **System** ... síť paralelních časovaných automatů (procesů).
- **Proces** ... instance parametrizovaného vzoru.

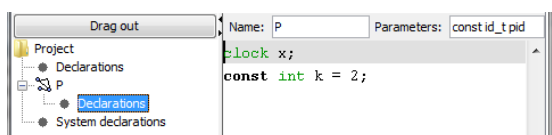
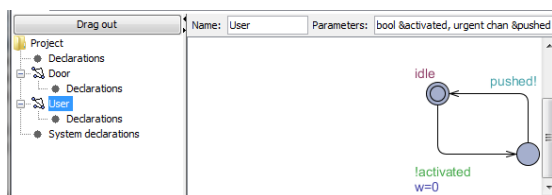


## Proces

- **Pozice** ...
  - jméno,
  - invarianty
- **Hrany** ...
  - podmínky stráží ( $x \geq 7$ ),
  - synchronizace ( $go[id]?$ ),
  - přiřazení ( $x = 0$ ),



# Popis vzoru (template) <sup>[?]</sup>



## Parametrizovaný časový automat

- jméno,
- parametry,

## Lokální deklarace

- proměnné,
- synchronizační kanály,
- konstanty



# Popis systému <sup>[?]</sup>

```

const int N = 6;          // # trains
typedef int[0,N-1] id_t;

chan      appr[N], stop[N], leave[N];
urgent chan go[N];

```

## Globální deklarace

- globální celočíselné proměnné,
- globální hodiny,
- synchronizační kanály,
- konstanty

## Definice systému <sup>[?]</sup>

```
bool activated1, activated2;
urgent chan pushed1, pushed2;
urgent chan closed1, closed2;

Door1 = Door(activated1, pushed1, closed1, closed2);
Door2 = Door(activated2, pushed2, closed2, closed1);
User1 = User(activated1, pushed1);
User2 = User(activated2, pushed2);

system Door1, Door2, User1, User2;
```

### Přiřazení procesů

- deklarace instancí procesu,
- vzory s úplně/částečně specifikovanými parametry,

### Definice systému

- seznam procesů systému,

## Literatura I

