Temporální logiky

Radek Mařík

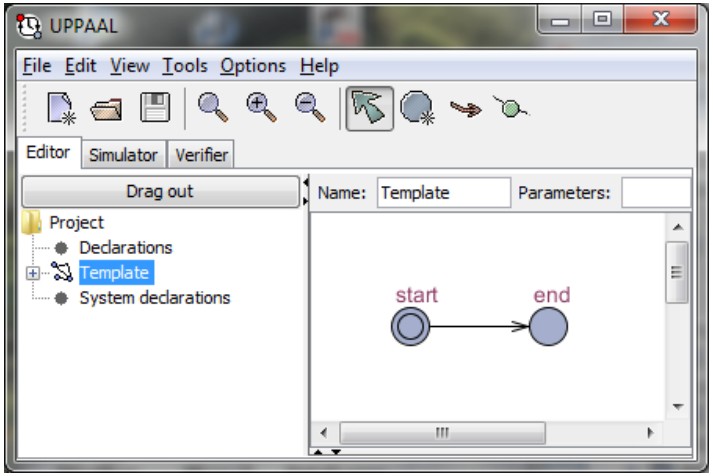
Czech Technical University
Faculty of Electrical Engineering
Department of Telecommunication Engineering
Prague CZ

December 5, 2017

Outline

1. Systém UPPAAL
   - Postup modelování a ověřování
2. Základy temporálních logik
   - Cesty výpočtu a čas
   - CTL* logika
   - CTL logika
   - LTL logika
3. UPPAAL
   - Specifikace požadavků v UPPAAL
   - Jazyk modelů
   - Vlastnosti ověřování modelů
   - Čas v UPPAAL
   - Urgentní přechody UPPAAL
4. UPPAAL příklady
   - Přejezd vlaků přes most
   - Hra NIM
     - Specifikace požadavků hry NIM

Tvorba automatu [UPP09]



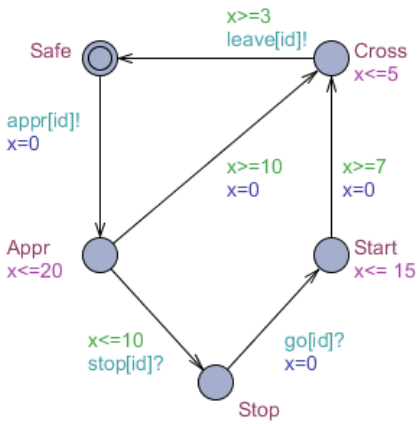
Automat

- 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 [UPP09]

Systém

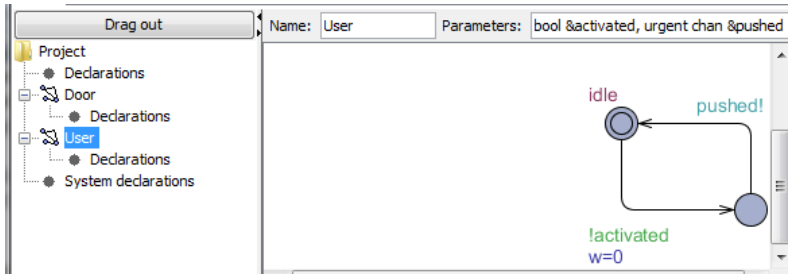
- **Systém** . . . síť paralelních časovaných automatů (procesů).
- **Proces** . . . instance parametrizovaného vzoru.

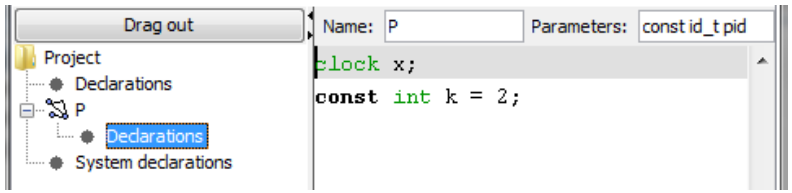


Proces

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

Popis vzoru (template) [UPP09]





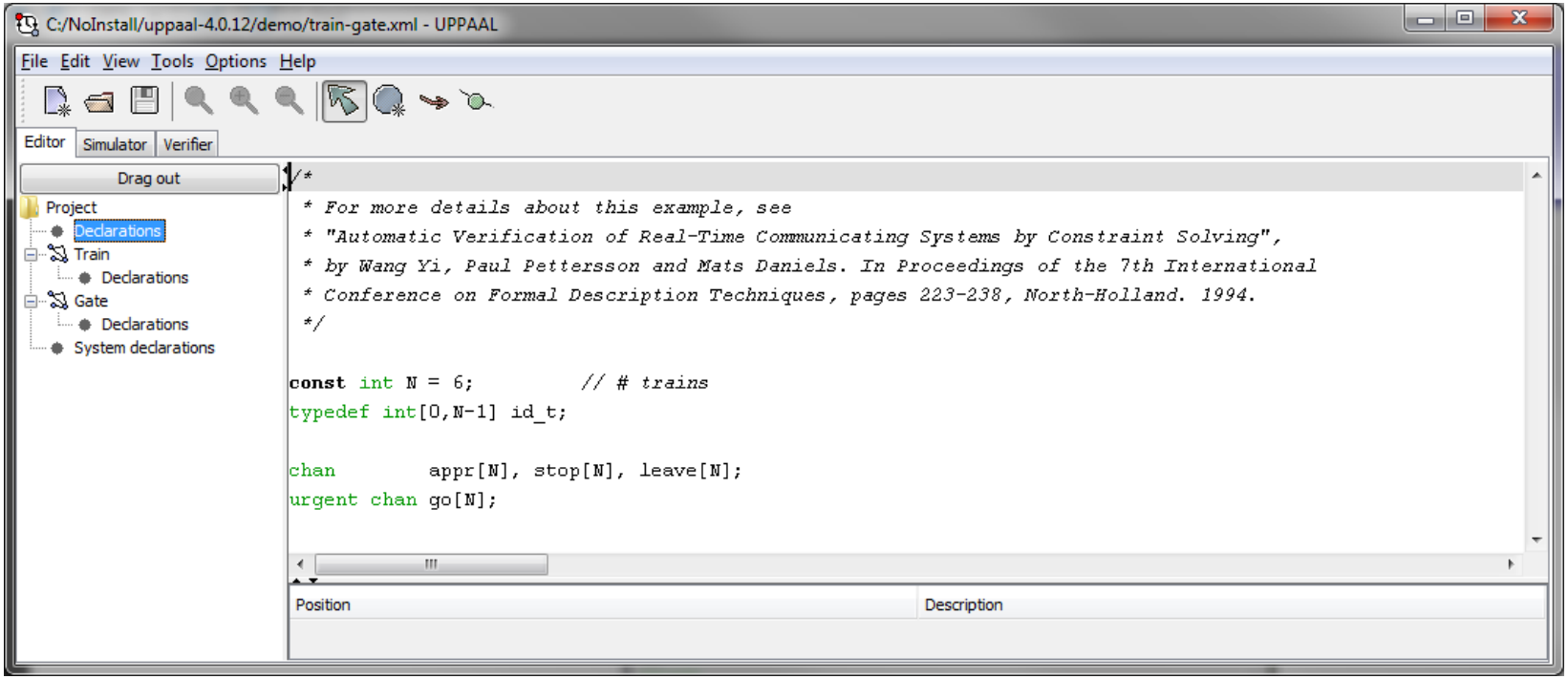
Parametrizovaný časový automat

- jméno,
- parametry,

Lokální deklarace

- proměnné,
- synchronizační kanály,
- konstanty

Popis systému [UPP09]



Globální deklarace

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

Definice systému [UPP09]

```
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;
```

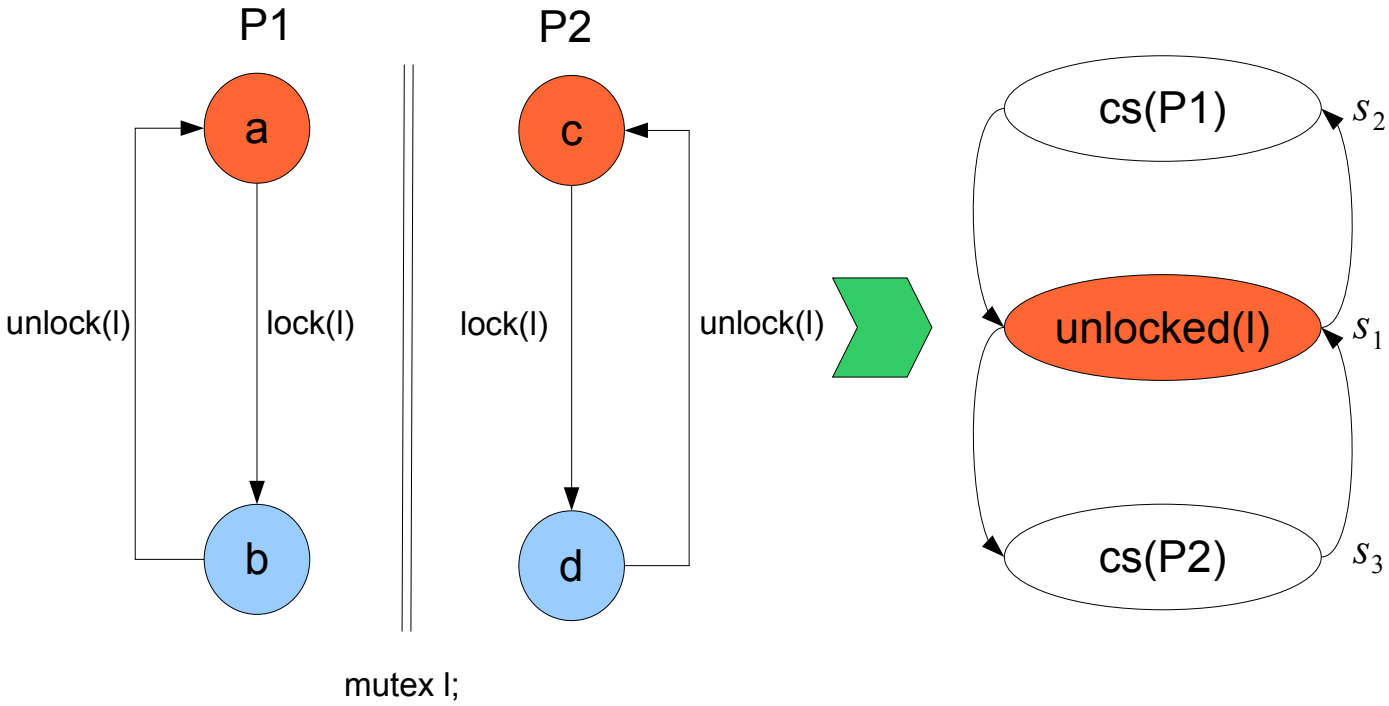
Přiřazení procesů

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

Definice systému

- seznam procesů systému,

Přechody mezi konfiguracemi v Kripkeho struktuře [Voj10]



Cesta v Kripkeho struktuře [Voj10]

Cesta

- **Cesta** π ... v Kripkeho struktuře M je nekonečná sekvence stavů $\pi = s_0 s_1 s_3 \ldots$ taková, že $\forall i \in N..R(s_i, s_{i+1})$.
- $\Pi(M, s)$ ... množina všech cest v M, které začínají v $s \in S$
- Sufix π^i cesty $\pi = s_0 s_1 s_3 \ldots s_i s_{i+1} s_{i+2}$ je cesta $\pi^i = s_i s_{i+1} s_{i+2}$ začínající v s_i.
- $s_i = \pi[i]$

Pojem času [Voj10]

Abstrakce času

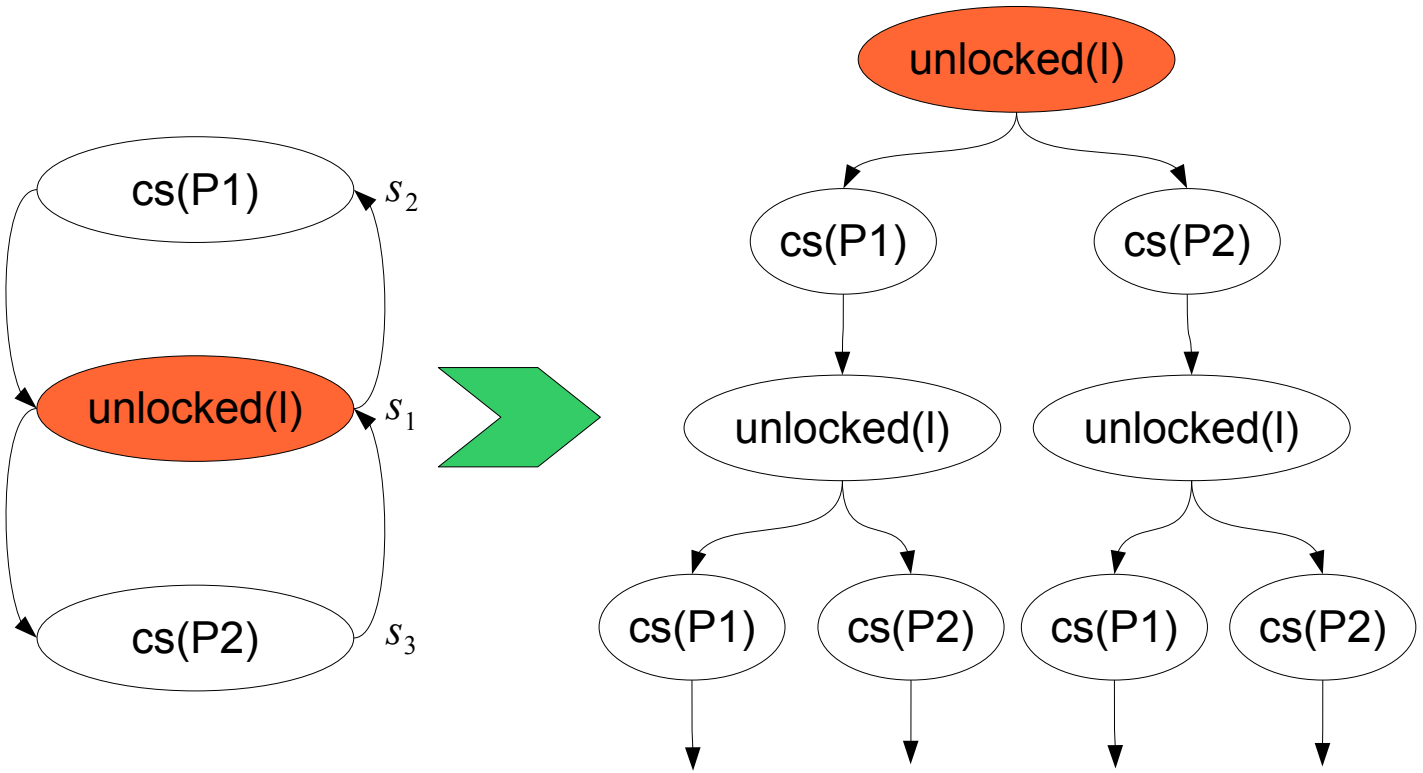
- **Logický čas** . . . pracuje s *(částečným) uspořádáním* stavů/událostí v chování systému.
- **Fyzický čas** . . . *měření* doby uběhlou mezi dvěma stavy/událostí.

Čas ve verifikaci modelů

- **Lineární čas** . . . dovoluje se vyjadřovat pouze o dané *lineární trase* ve stavovém prostoru.
  - Na všech trasách, x musí být následováno y.
  - Na všech trasách, x musí být následováno y nebo z.
- **Větvící se čas** . . . dovoluje kvantifikovat (existenčně i univerzálně) možné budoucnosti počínaje daným stavem. Na stavový prostor se pohlíží jako na rozvinutý *nekonečný strom*.
  - Existuje trasa, kde následující stav je x.

Výpočetní strom [Voj10]

Popisuje vlastnosti výpočetního stromu



CTL* formule [Voj10]

Skládá se z

- atomické výroky
- logické spojky
- kvantifikátory cest
- temporální operátory

CTL* kvantifikátory a operátory [Wik10, Voj10]

Kvantifikátory cest

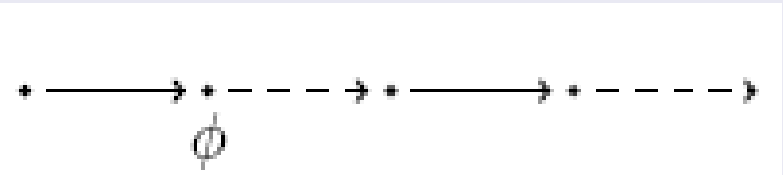
popisují strukturu větvení vypočetního stromu

- E . . . existuje cesta výpočtu z daného stavu.
- A . . . pro všechny cesty výpočtů z daného stavu.

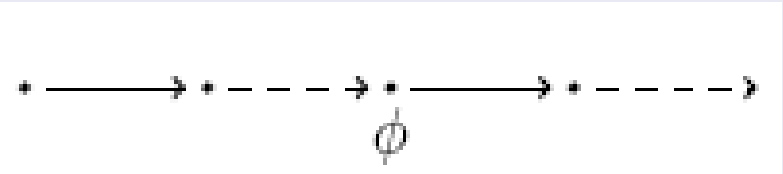
Temporální operátory

určují vlastnosti cesty ve výpočetním stromu

- $X\varphi$ (next time, $\bigcirc$). . . vlastnost φ platí ve druhém (následujícím) stavu cesty..



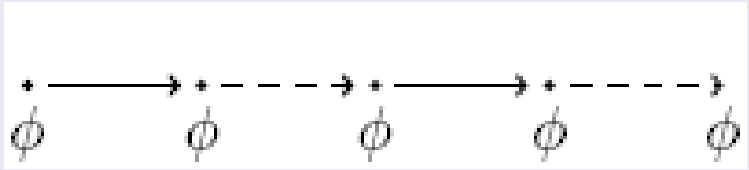
- $F\varphi$ (in future, $\Diamond$). . . vlastnost φ platí v nějakém stavu cesty.



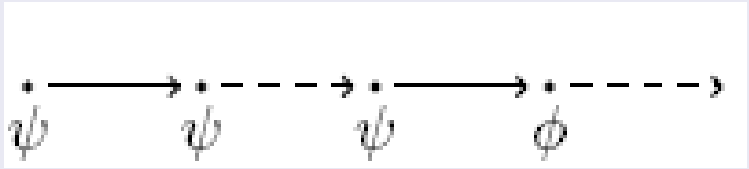
CTL* operátory [Wik10, Voj10]

Temporální operátory

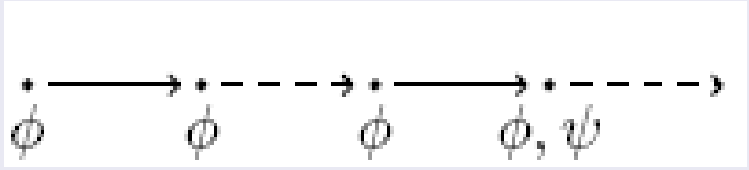
- $G\varphi$ (globally, $\Box$)... vlastnost φ platí ve všech stavech cesty.

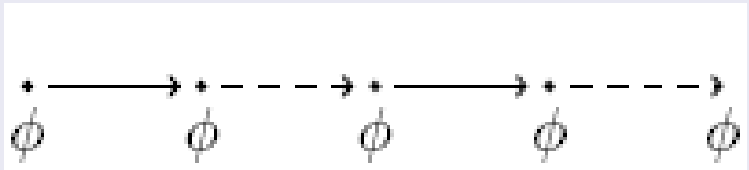


- $\psi U \varphi$ (until)... vlastnost φ platí v nějakém stavu cesty a vlastnost ψ platí přinejmenším ve všech předcházejících stavech této cesty.



- $\psi R \varphi$ (release)... vlastnost φ musí platit do (a včetně) stavu, kdy začne platit vlastnost ψ, pokud takový stav existuje.





CTL* syntax [Voj10]

Nechť AP je neprázdná množina atomických výroků.

Syntax stavových formulí, které jsou pravdivé v daném stavu

- Jestliže $p \in AP$, potom p je stavová formule.
- Jestliže φ a ψ jsou stavové formule, potom $\neg\varphi$, $\varphi \vee \psi$, $\varphi \wedge \psi$ jsou stavové formule.
- Jestliže φ je běhová formule, potom $E\varphi$ a $A\varphi$ jsou stavové formule.

Syntax běhových formulí, které jsou pravdivé podél specifické cesty

- Jestliže φ je stavová formule, pak φ je také běhová formule.
- Jestliže φ a ψ jsou běhové formule, pak $\neg\varphi$, $\varphi \vee \psi$, $\varphi \wedge \psi$, $X\varphi$, $F\varphi$, $G\varphi$, $\varphi U \psi$ a $\varphi R \psi$ jsou běhové formule.

CTL* je množina stavových formulí generovaných výše uvedenými pravidly.

CTL* sémantika [Voj10]

- Nechť je dána Kripkeho struktura $M = (S, T, \mathcal{I}, s_0, L)$ nad množinou atomických výroků AP.
- Pro stavovou formuli φ nad AP, zapisujeme $M, s \models \varphi$ fakt, že φ platí v $s \in S$.
- Pro běhovou formuli φ nad AP, zapisujeme $M, \pi \models \varphi$ fakt, že φ platí podél cesty π v M.
- Nechť $s \in S$, π je cesta v M, φ_1, φ_2 jsou stavové formule nad AP, $p \in AP$, a ψ_1, ψ_2 jsou běhové formule nad AP. Pak relaci $\models$ definujeme induktivně následovně:
  - $\boldsymbol{M, s \models p}$ iff $p \in L(s)$.
  - $\boldsymbol{M, s \models \neg \varphi_1}$ iff $M, s \not\models \varphi_1$.
  - $\boldsymbol{M, s \models \varphi_1 \vee \varphi_2}$ iff $M, s \models \varphi_1$ nebo $M, s \models \varphi_2$.
  - $\boldsymbol{M, s \models \varphi_1 \wedge \varphi_2}$ iff $M, s \models \varphi_1$ a zároveň $M, s \models \varphi_2$.
  - $\boldsymbol{M, s \models E\psi_1}$ iff $\exists \pi \in \Pi(M, s). M, s \models \psi_1$.
  - $\boldsymbol{M, s \models A\psi_1}$ iff $\forall \pi \in \Pi(M, s). M, s \models \psi_1$.

CTL* sémantika [Voj10]

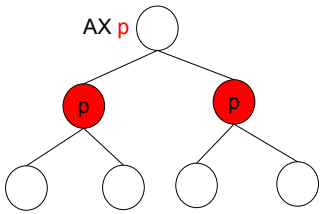
- Pokračování definice relace $\models$:
  - $\boldsymbol{M, \pi \models \varphi_1}$ iff $M, s_0 \models \varphi_1, s_0 = \pi[0]$.
  - $\boldsymbol{M, \pi \models \neg \psi_1}$ iff $M, \pi \not\models \psi_1$.
  - $\boldsymbol{M, \pi \models \psi_1 \vee \psi_2}$ iff $M, \pi \models \psi_1$ nebo $M, \pi \models \psi_2$.
  - $\boldsymbol{M, \pi \models \psi_1 \wedge \psi_2}$ iff $M, \pi \models \psi_1$ a zároveň $M, \pi \models \psi_2$.
  - $\boldsymbol{M, \pi \models X \psi_1}$ iff $M, \pi^1 \models \psi_1$.
  - $\boldsymbol{M, \pi \models F \psi_1}$ iff $\exists i \geq 0 . M, \pi^i \models \psi_1$.
  - $\boldsymbol{M, \pi \models G \psi_1}$ iff $\forall i \geq 0 . M, \pi^i \models \psi_1$.
  - $\boldsymbol{M, \pi \models \psi_1 U \psi_2}$ iff $\exists i \geq 0 . M, \pi^i \models \psi_2$
    a zároveň $\forall 0 \leq j < i . M, \pi^j \models \psi_1$.
  - $\boldsymbol{M, \pi \models \psi_1 R \psi_2}$ iff $\forall i \geq 0 . (\forall 0 \leq j < i . M, \pi^j \not\models \psi_1 \Rightarrow M, \pi^i \models \psi_2$.

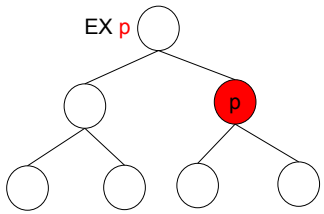
CTL* základní operátory [Voj10]

- Všechny CTL* operátory lze odvodit z $\vee$, $\neg$, X, U a E:
  - Nech $p \in AP$, $true \equiv p \vee \neg p$ (a $false \equiv \neg true$)
  - $\varphi \wedge \psi \equiv \neg(\neg\varphi \vee \neg\psi)$,
  - $F\varphi \equiv true U \varphi$,
  - $G\varphi \equiv \neg F \neg\varphi$,
  - $\varphi R \psi \equiv \neg(\neg\varphi U \neg\psi)$,
  - $A\varphi \equiv \neg E \neg\varphi$.

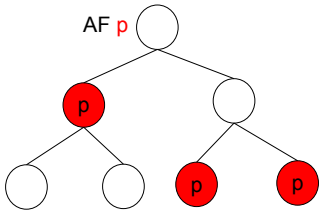
CTL syntaxe [Voj10]

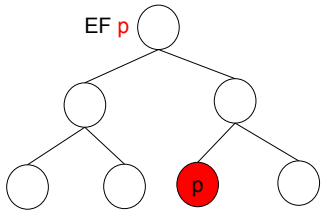
- CTL je sublogikou CTL*
  - běhové formule jsou omezeny na $X\varphi$, $F\varphi$, $G\varphi$, $\varphi U\psi$ a $\varphi R\psi$,
  - kde φ a ψ jsou stavové formule.
- Proto pouze 10 modálních CTL operátorů:
  - AX a EX



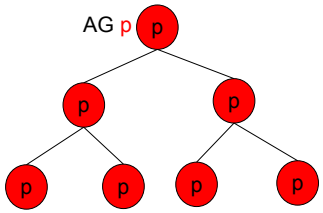


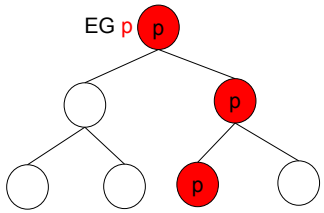
  - AF a EF





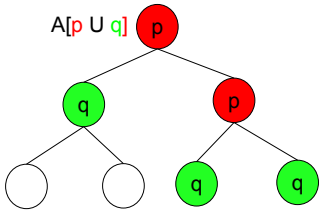
  - AG a EG

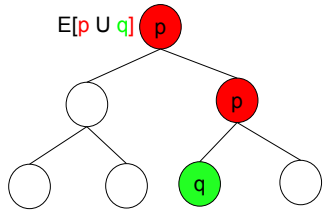




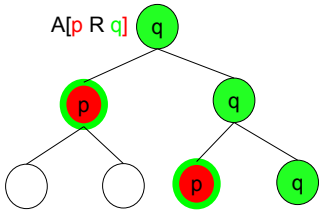
CTL modální operátory [Voj10]

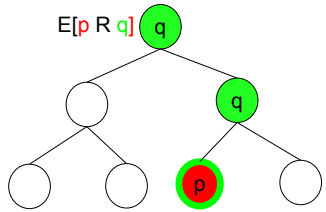
- Modální CTL operátory:
  - AU a EU





  - AR a ER





- Existují 3 základní CTL modální operátory - EX, EG a EU:
  - $\boldsymbol{AX\varphi} \equiv \neg EX\neg\varphi$
  - $\boldsymbol{EF\varphi} \equiv E[trueU\varphi]$
  - $\boldsymbol{AG\varphi} \equiv \neg EF\neg\varphi$
  - $\boldsymbol{AF\varphi} \equiv \neg EG\neg\varphi$
  - $\boldsymbol{A[\varphi U\psi]}$ $\equiv \neg E[\neg\psi U(\neg\varphi \wedge \neg\psi)] \wedge \neg EG\neg\psi$
  - $\boldsymbol{A[\varphi R\psi]} \equiv \neg E[\neg\varphi U\neg\psi]$
  - $\boldsymbol{E[\varphi R\psi]} \equiv \neg A[\neg\varphi U\neg\psi]$

LTL syntaxe [Voj10]

- LTL je sublogikou CTL*
  - Povoluje pouze formule tvaru $A\varphi$, ve kterých stavové podformule jsou atomickými výroky
  - LTL formule se vytváří dle následující gramatiky:
    - $\varphi ::= A\psi$ (A se často vynechává)
    - $\psi ::= p \mid \neg\psi \mid \psi \vee \psi \mid \psi \wedge \psi \mid X\psi \mid F\psi \mid G\psi \mid \psi U \psi \mid \psi R \psi$,
    - kde $p \in AP$.
  - LTL se vyjadřuje o specifických cestách v dané Kripkeho struktuře
    - tj. ignoruje větvění

LTL, CTL, CTL* [Voj10]

- LTL a CTL nelze vůči sobě porovnat:
  - CTL nemůže např. vyjádřit LTL formuli $A(FGp)$
  - LTL nemůže např. vyjádřit CTL formuli $AG(EFp)$
- CTL* pokrývá jak LTL, tak i CTL
  - disjunkce $(A(FGp)) \vee (AG(EFp))$ se nedá vyjádřit ani v LTL, ani v CTL.

BNF gramatika specifikačního jazyka [UPP10]

BNF gramatika

- $A[]Expression$
- $E <> Expression$
- $E[]Expression$
- $A <> Expression$
- $Expression -- > Expression$

Poznámky

- Žadný výraz nesmí mít postranní efekty.
- Výraz $process.location$ testuje, zda určitý proces je v dané pozici.

Příklady specifikačního jazyka [UPP10]

BNF gramatika

- $A[]1 < 2$
  - Invariatně $1 < 2$
- $E <> p1.cs and p2.cs$
  - Pravdivé, pokud systém může dosáhnout stavu, ve kterém procesy $p1$ a $p2$ jsou v jejich pozici cs
- $A <> p1.cs imply not p2.cs$
  - Invariantně process $p1$ v pozici cs implikuje, že proces $p2$ **není** v pozici cs.
- $A[]not\, deadlock$
  - Invariantně, process neobsahuje deadlock.

Podmínky nad hodinami [BDL05]

- C ... množina hodin
- $B(C)$ ... množina konjunkcí nad jednoduchými podmínkami typu
  - $x \bowtie c$
  - $x - y \bowtie c$
  - kde
    - $x, y \in C$,
    - $c \in \mathbb{N}$,
    - $\bowtie \in \{<, \leq, =, \geq, >\}$

Dotazovací jazyk [BDL05]

- **Stavové formule** . . . popisují individuální stavy.
- **Běhové formule** . . . vyhodnocují se podél cest a stop modelu.
  - dosažitelnost,
  - bezpečnost,
  - živost.

Stavové formule [BDL05]

- výraz, který lze vyhodnotit pro daný stav, aniž by bylo nutné analyzovat chování modelu.
- nadmnožinou stráží, tj. nemá žádný postranní efekt,
- na rozdíl od stráží, použití disjunkcí není omezeno.
- Test, zda proces je v dané pozici ... $P.\ell$
  - P ... proces
  - ℓ ... pozice
- **zablokování (deadlock)** ...
  - speciální stavová formule, která je splněna pro všechny zablokované stavy,
  - Stav je zablokovaný, jestliže neexistuje žádný akční přechod z daného stavu či jakéhokoliv jeho zpožděného následníka.

Dosažitelnost [BDL05]

- nejjednodušší vlastnost,
- požaduje, zda-li existuje možnost, že daná stavová formule φ je splněná v každém dosažitelném stavu.
- tj. existuje cesta z počátečního stavu taková, že φ bude jednou splněná podél této cesty.
- kontrola základních vlastností modelu
  - že platí alespoň základní chování
  - příklad komunikačního protokolu s jedním vysílačem a jedním přijímačem
    - je vůbec možné odeslat zprávu vysílačem
    - zpráva má nadějí být prijmuta přijímačem.
- v UPPAAL: E<>φ

Bezpečnost [BDL05]

- něco špatného nikdy nenastane
- příklad modelu jaderné elektrárny
  - provozní teplota je vždy (invariantně) pod určitým prahem,
  - nikdy nedojde k roztavení nádoby
- varianta: něco není možné, aby vůbec nastalo
- příklad hraní hry
  - bezpečný stav je takový, že pokud můžeme ještě hru, pak už neexistuje možnost, abychom ji prohráli.
- v UPPAAL:
  - formuluje se pozitivně
  - nechť φ je stavová formule
  - `A[]` $\varphi \equiv \neg E \Diamond \neg \varphi$ ... φ by měla být pravdivá ve všech dosažitelných stavech
  - `E[]` φ ... existuje maximální cesta, podél které φ je vždy pravdivá

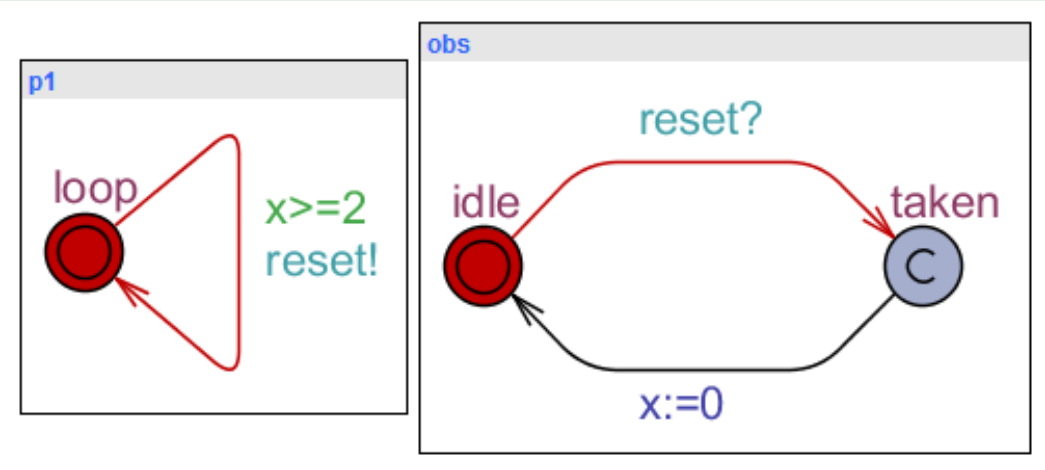
Živost [BDL05]

- něco jednoho dne určitě nastane
- příklady
  - stisknutí tlačítka *on* na dálkovém ovladači způsobí, že se televize jednou zapne.
  - v modelu komunikačního protokolu:
    jakákoliv vyslaná zpráva bude jednou přijmuta.
- v UPPAAL:
  - `A<>`$\varphi \equiv \neg E \Box \neg \varphi \ldots \varphi$ bude vždy jednou splněna
  - φ `-->` $\psi \equiv A \Box (\varphi \Rightarrow A \Diamond \psi) \ldots$
    kdykoliv je splněna φ, potom bude jednou splněna i ψ

Pozorovatel [BDL05]

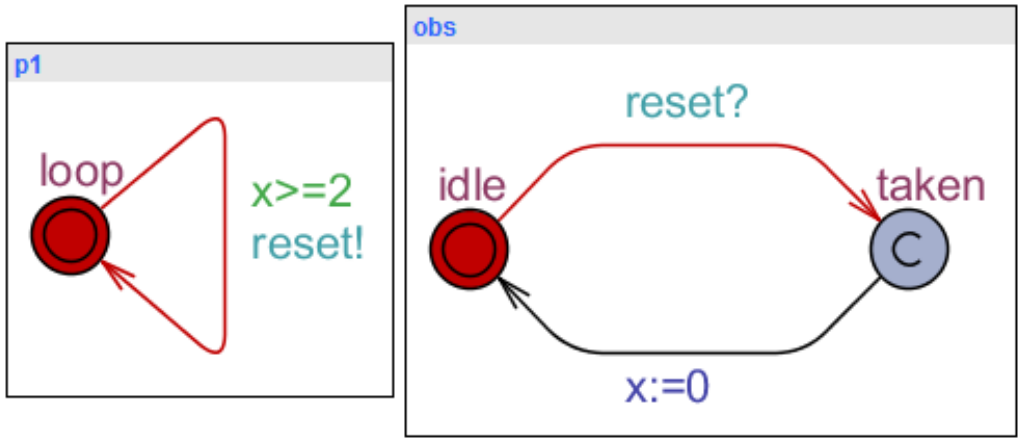
- přidaný automat
- detekuje události, aniž by bylo nutné měnit vlastní model

Příklad



- detekce resetování hodin
- navíc resetování hodin (x:=0)

Výchozí varianta příkladu [BDL05]



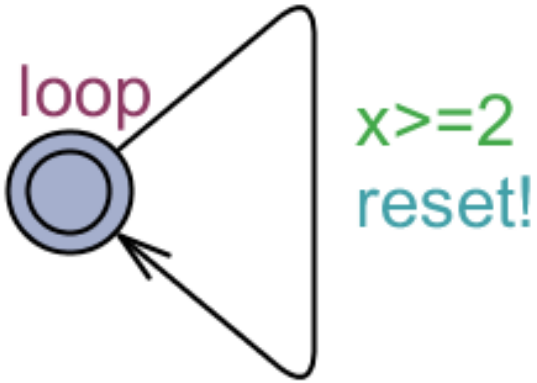

```
// Place global declarations here.
clock x;
chan reset;
```

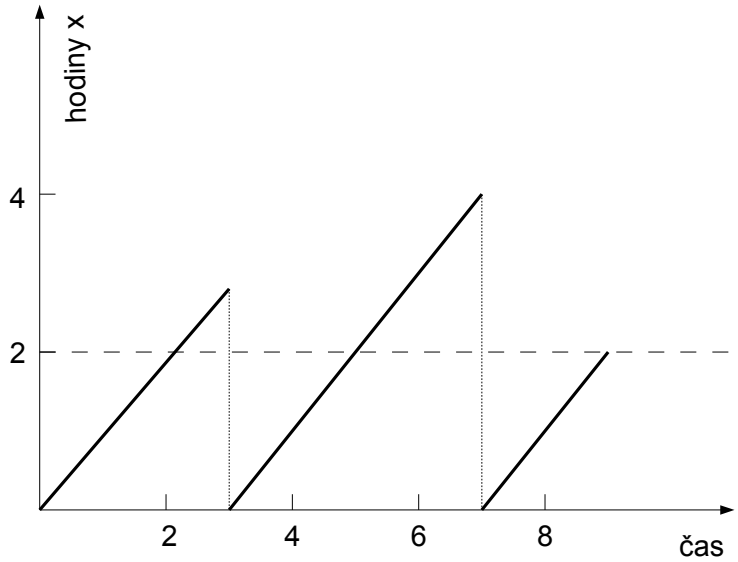
```
// Place template instantiations here.
p1 = P1();
obs = Obs();

// List one or more processes to be comp
system p1, obs;
```

- Cílem je zůstat v pozici, pokud platí podmínka na hodinách a poté opustit pozici
- Varianta 1: bez invariantu

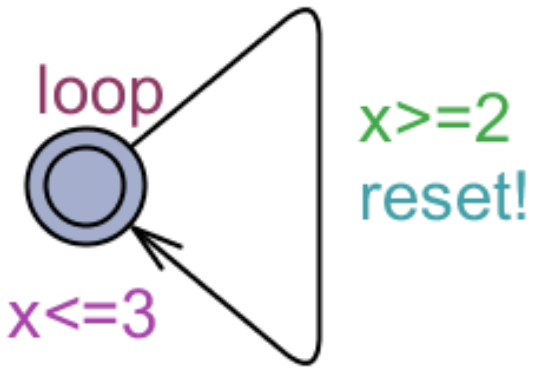
1. varianta příkladu [BDL05]

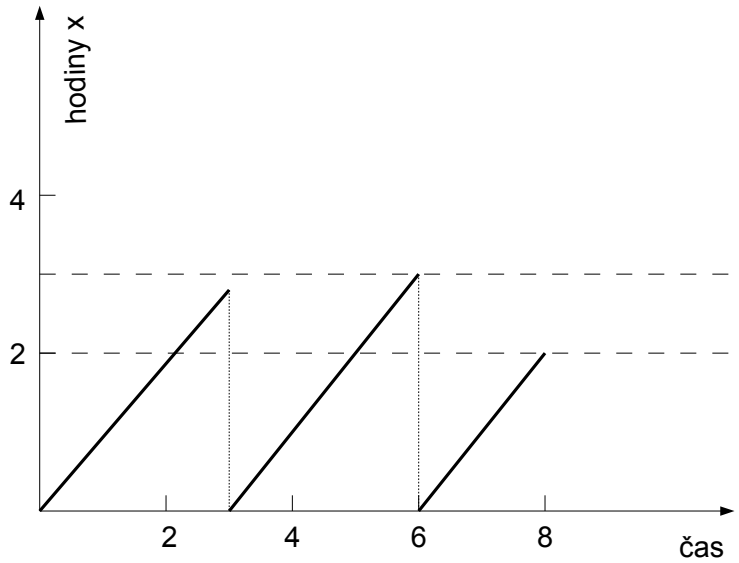




- Cílem je zůstat v pozici, pokud platí podmínka na hodinách a poté opustit pozici
- Varianta 1: bez invariantu
- `A[] obs.taken imply x>=2`
- `E<> obs.idle and x>3`

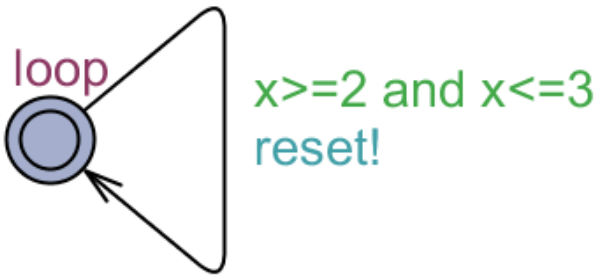
2. varianta příkladu [BDL05]

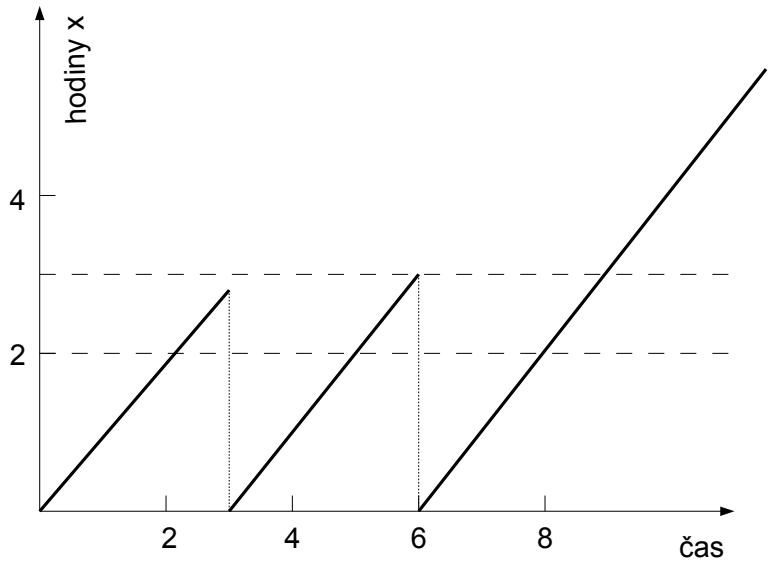




- Cílem je zůstat v pozici, pokud platí podmínka na hodinách a poté opustit pozici
- Varianta 2: s invariantem
- `A[] obs.taken imply (x>=2 and x<=3)`
- `E<> obs.idle and x>2`
  - `E<> obs.idle and x>3` ...neplatí
- `A[] obs.idle imply x<=3`

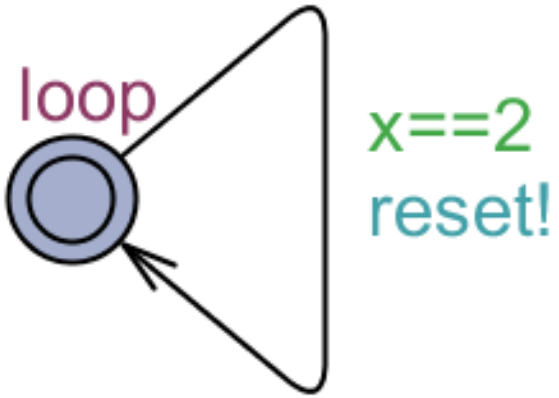
3. varianta příkladu [BDL05]

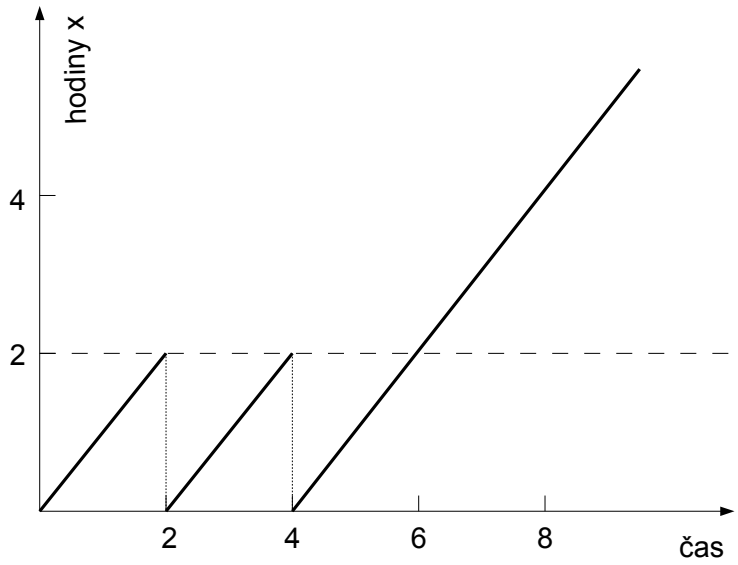




- Cílem je zůstat v pozici, pokud platí podmínka na hodinách a poté opustit pozici
- Varianta 3: bez invariantu se stráží
- `A[] x>3 imply not obs.taken` ... zablokování
- `A[] not deadlock` ... neplatí

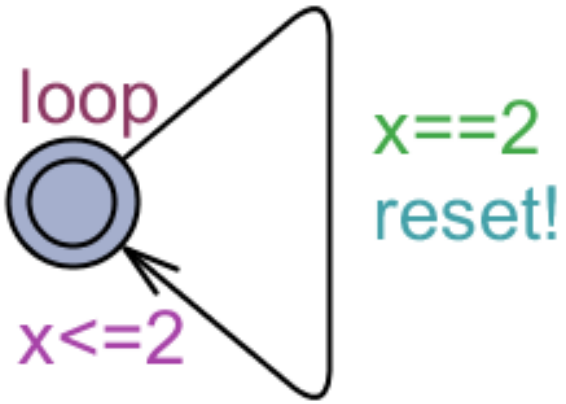
4. varianta příkladu [BDL05]

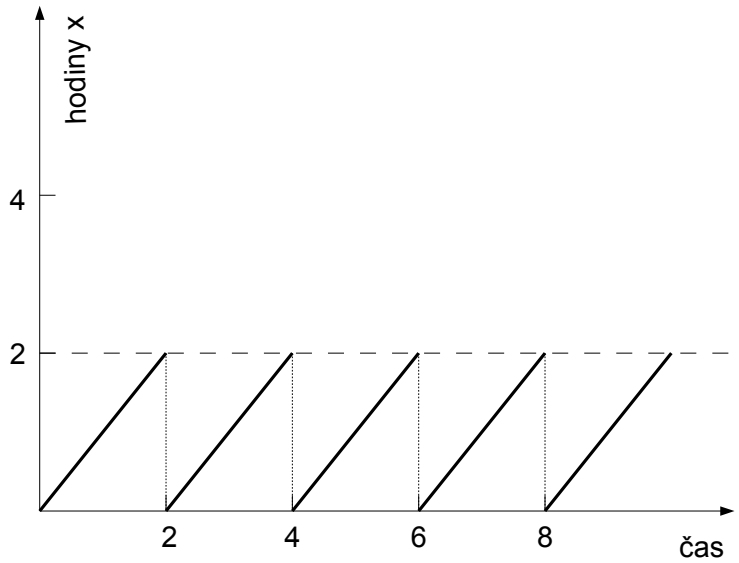




- Cílem je zůstat v pozici, pokud platí podmínka na hodinách a poté opustit pozici
- Varianta 4: bez invariantu se stráží s rovností
- `A[] x>2 imply not obs.taken` ...zablokování
- `A[] not deadlock` ...neplatí

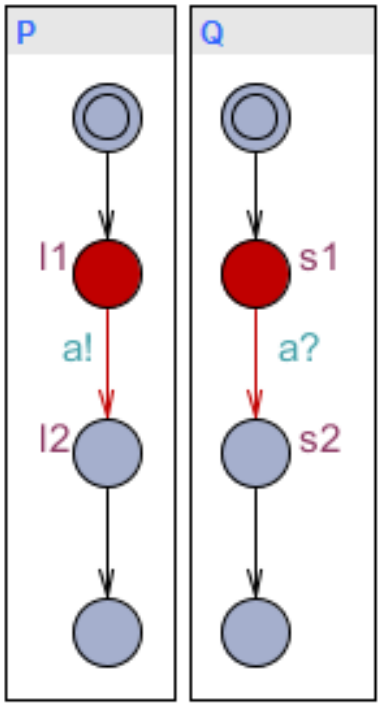
5. varianta příkladu [BDL05]





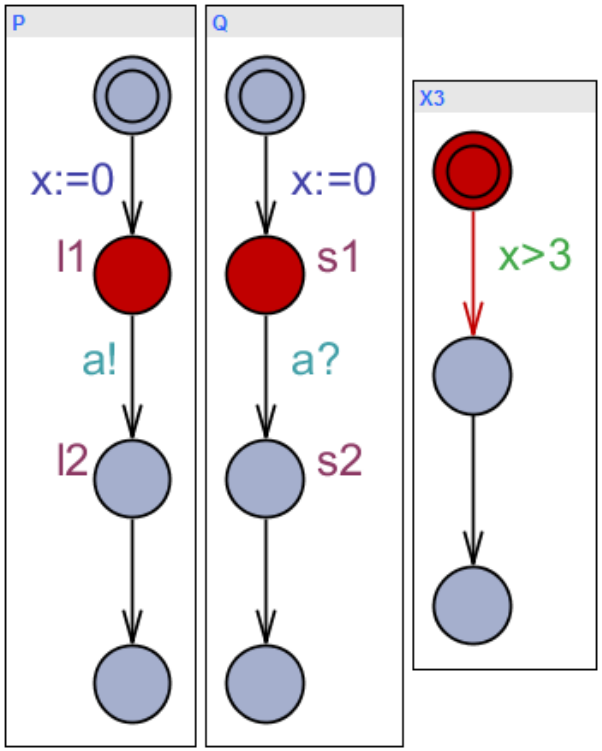
- Cílem je zůstat v pozici, pokud platí podmínka na hodinách a poté opustit pozici
- Varianta 5: s invariantem a se stráží s rovností
- `A[] obs.taken imply x==2`
- `E<> obs.idle and x>2` ...neplatí
- `A[] obs.idle imply x<=2`

Příklad 1, procesy P, Q [Dav05]



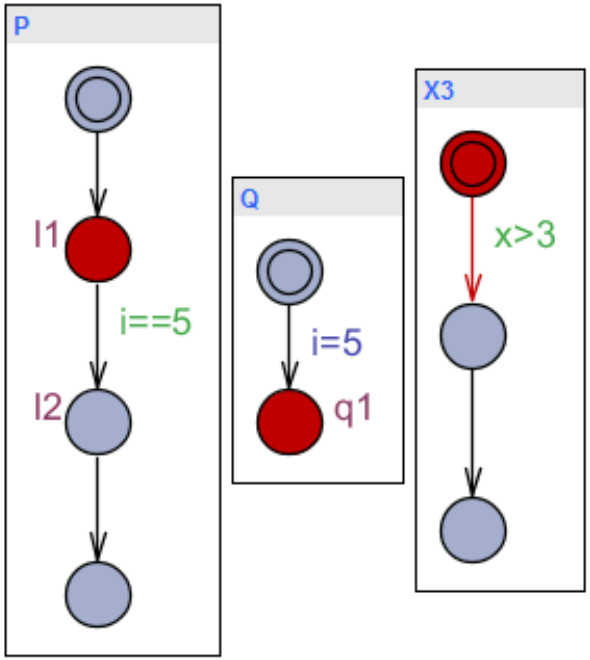
- Cílem je provést přechod se synchronizací co možná nejdříve.
- tj. jakmile jsou oba automaty P a Q připraveny (současně v pozicích l_1 a s_1).
- Jak zvolit model, když se do pozic dostanou v jiný okamžik?

Příklad 1, procesy $P, Q, X3$ [Dav05]



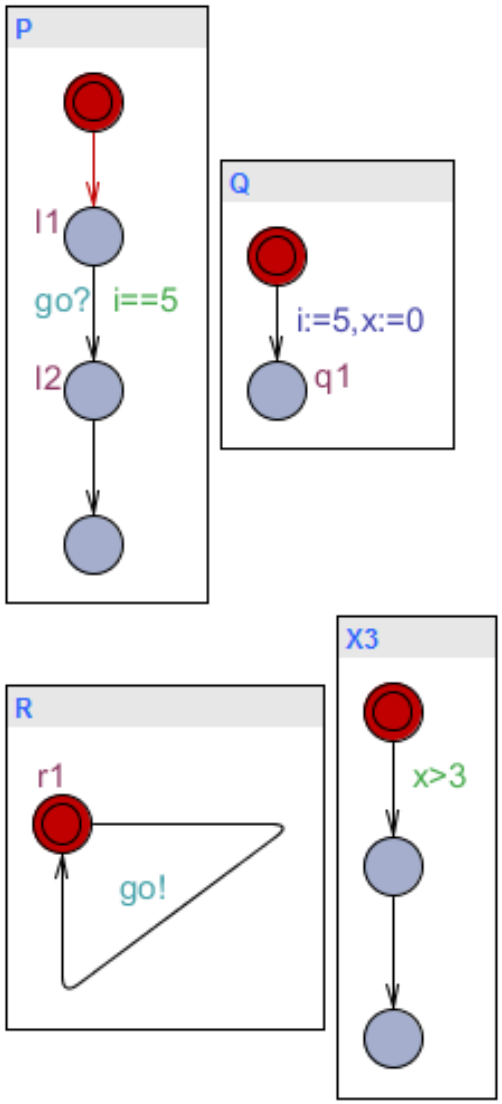
- Cílem je provést přechod se synchronizací co možná nejdříve.
- tj. jakmile jsou oba automaty P a Q připraveny (současně v pozicích l_1 a s_1).
- Jak zvolit model, když se do pozic dostanou v jiný okamžik?
- **Řešení:** `urgent chan a`

Příklad 2, procesy $P, Q, X3$ [Dav05]



- Cílem je provést přechod s podmínkou $i == 5$, jakmile je splněna.

Příklad 2, procesy $P, Q, R, X3$ [Dav05]

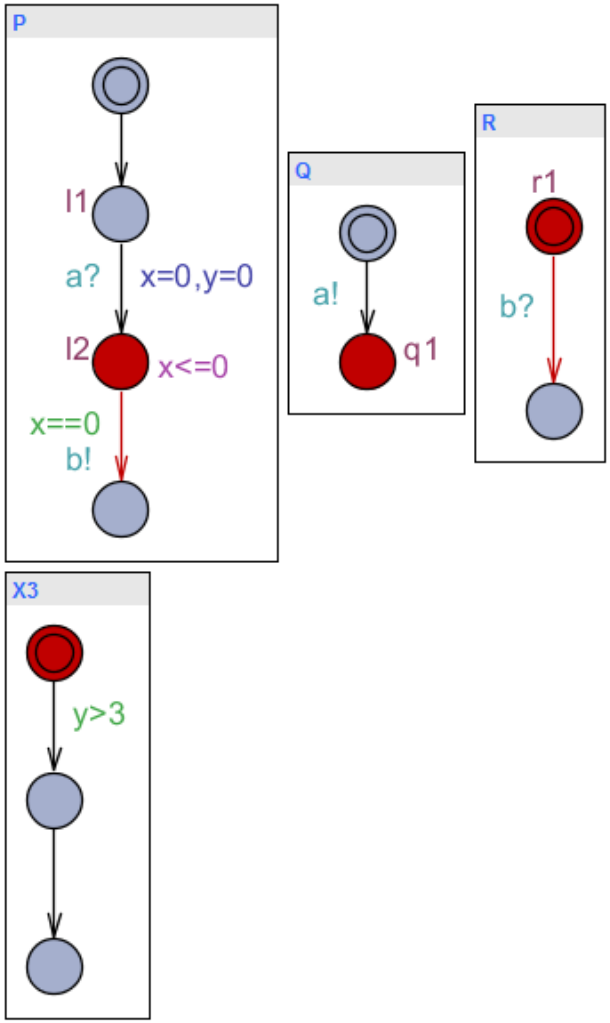


- Cílem je provést přechod s podmínkou $i == 5$, jakmile je splněna.
- tj. jakmile jsou oba automaty P a Q připraveny (současně v pozicích l_1 a s_1).
- Jak zvolit model, když se do pozic dostanou v jiný okamžik?
- **Řešení:**
- `urgent chan go`
- další proces emitující akci $go!$

Urgentní kanály [Dav05]

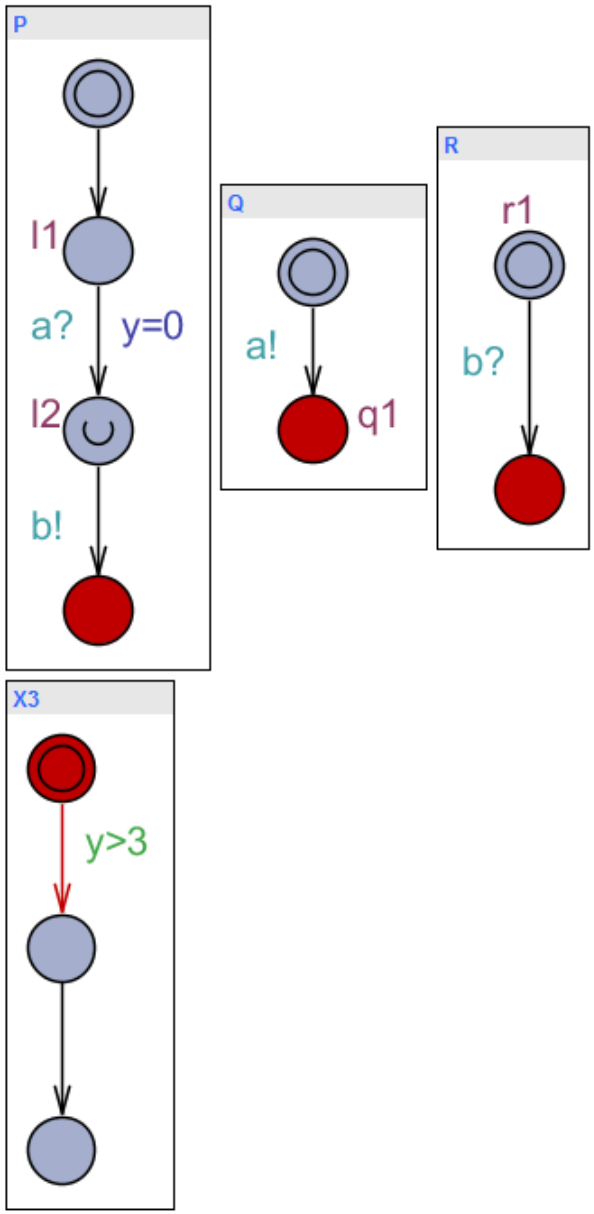
- `urgent chan hurry`
- **Semantika:**
- Nenastane žádné zpoždění, pokud hrana s urgentní akcí může být provedena.
-
- **Omezení:**
  - Na hranách s urgentní akcí nejsou povoleny žádné stráže s hodinami.
  - Invarianty a stráže na datovými proměnnými jsou povoleny.

Urgentní pozice pomocí hodin [Dav05]



- Předpokládejme, že modelujeme jednoduché systém M, které přijímá balíky na kanálu a a ihned je odesílá na kanál b
- P_1 modeluje systém pomocí hodin x

Urgentní pozice [Dav05]

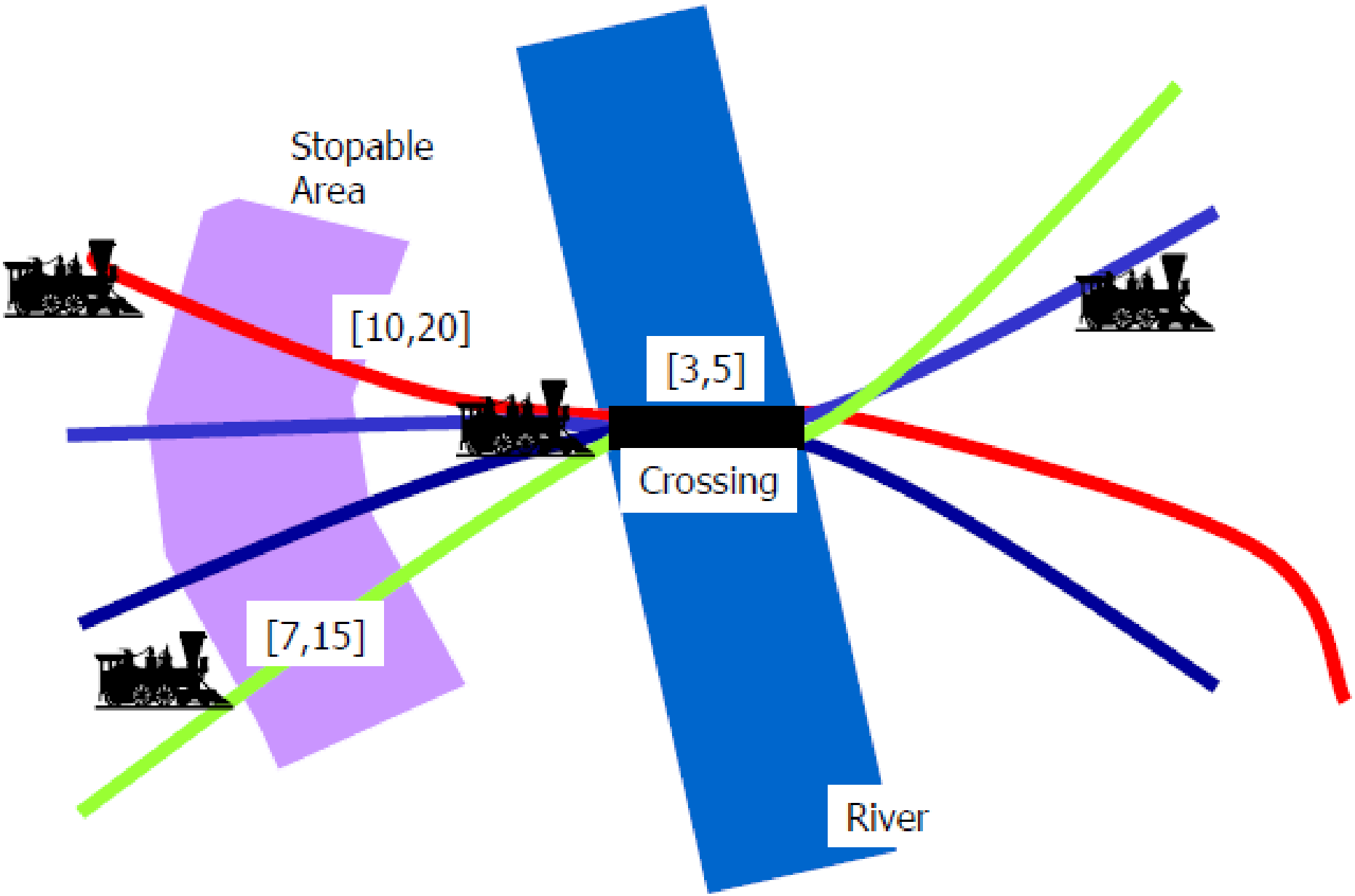


- Předpokládejme, že modelujeme jednoduché systém M, které přijímá balíky na kanálu a a ihned je odesílá na kanál b
- P_2 modeluje systém pomocí urgentní pozice
- P_1 a P_2 mají totožné chování

Urgentní kanály [Dav05]

- **Semantika:**
- Nenastane žádné zpoždění v urgentní pozici.
-
- **Poznámka:**
  - Použití urgetních pozic **redukuje** počet hodin v modelu a tím i složitost analýzy.

Myšlenka příkladu [BDL05]



Slovní zadání příkladu [BDL05]

Zadání

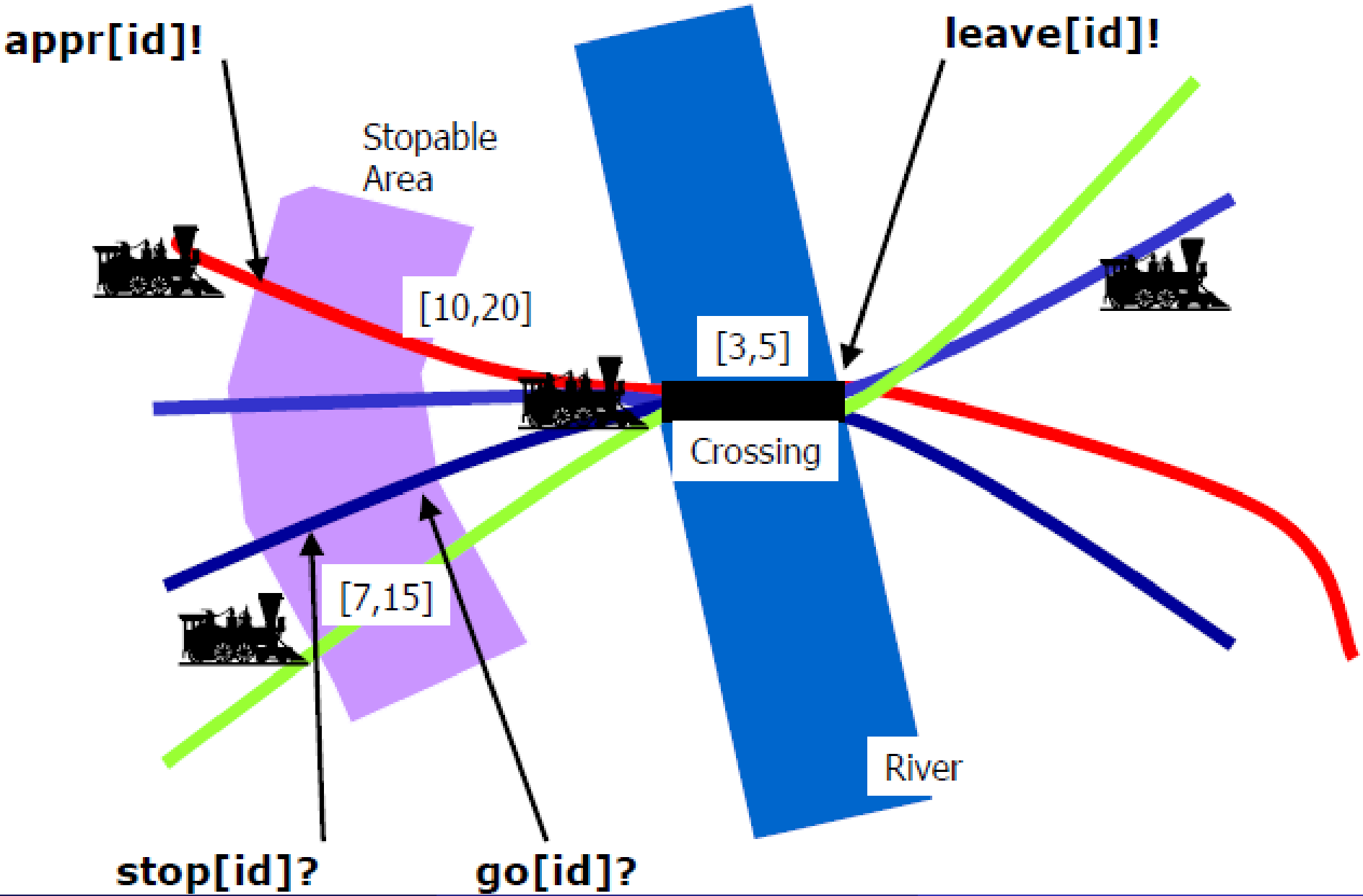
- řízení přístupu k mostu pro několik vlaků
- most jako kriticky sdílený zdroj může být přejížděn pouze jedním vlakem
- systém je definován jako několik vlaků a řadič
- vlak nemůže být zastaven okamžitě, rovněž rozjezd trvá dobu.

Časování a komunikace [BDL05]

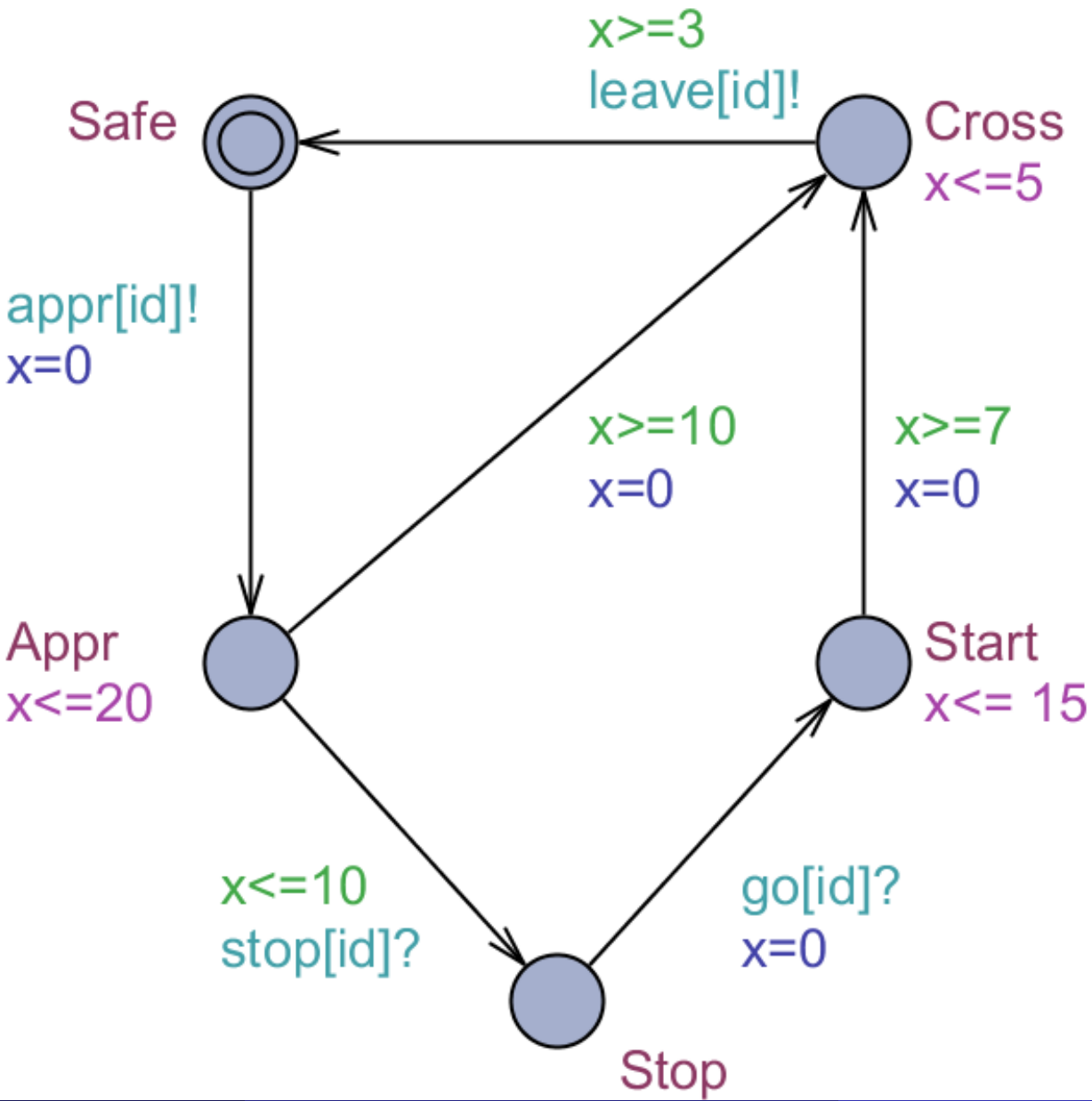
Časová omezení a komunikace

- při příjezdu k mostu vlak včas vyšle **appr!** signál
- poté vlak má 10 časových jednotek, aby přijal signál k zastavení
  - umožňuje bezpečné zastavení před mostem
- po těchto 10 časových jednotkách, trvá dalších 10 jednotek, než vlak dojede k mostu, pokud není zastaven
- jestliže je vlak zastaven, vlak se rozjede po té, co přijme signál **go!** z řadiče mostu
- když vlak opouští most, vyšle signál **leave!**

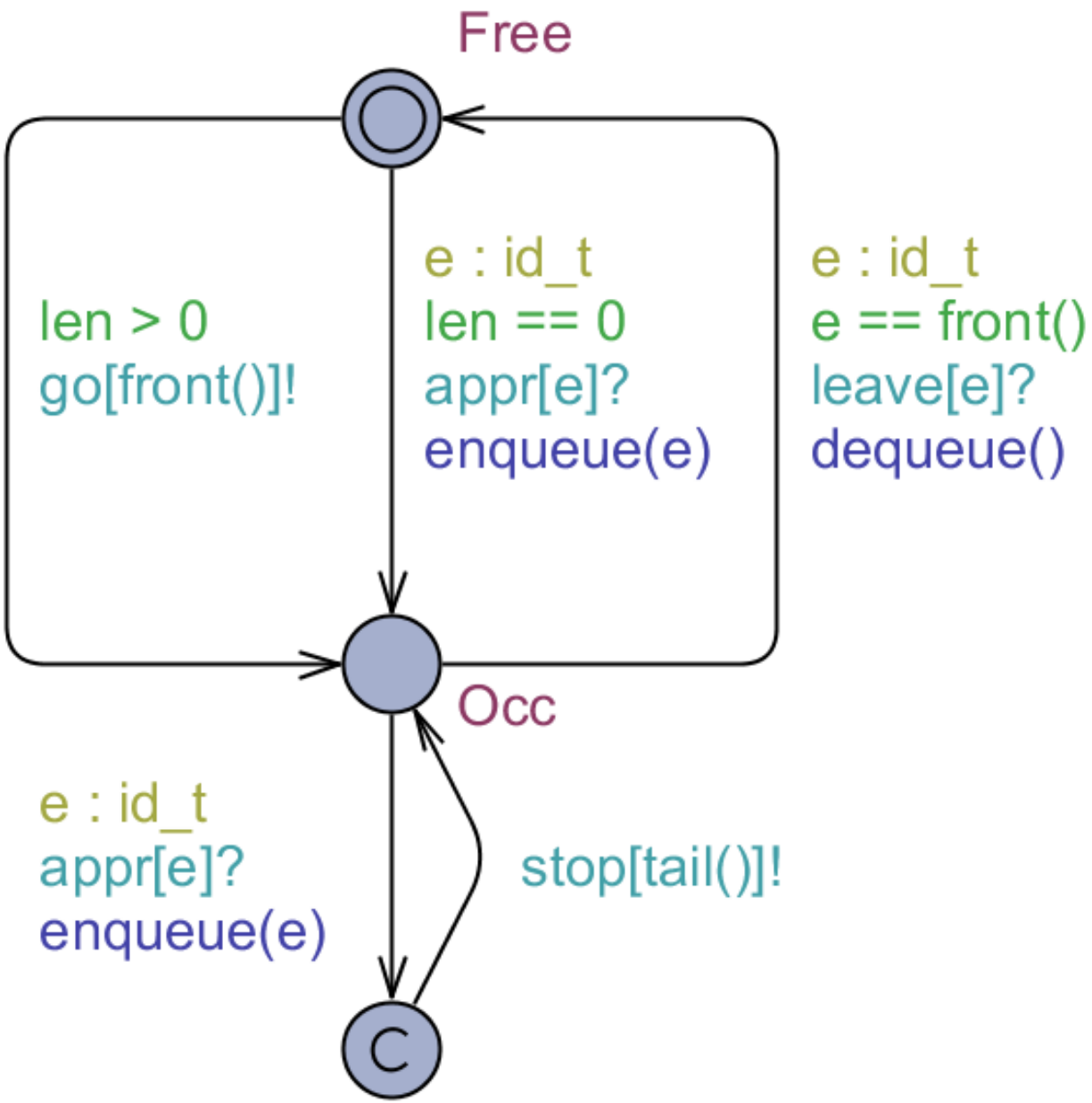
Synchronizační signály [BDL05]



Šablona vlaku [BDL05]



Šablona řadiče mostu [BDL05]



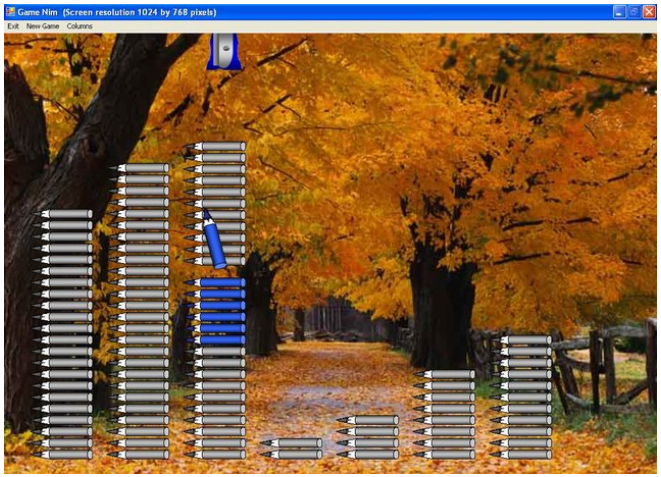
Ověření modelu [BDL05]

- `E<> Gate.Occ`
- `E<> Train(0).Cross`
- `E<> Train(1).Cross`
- `E<> Train(0).Cross and Train(1).Stop`
- `E<> Train(0).Cross and (forall (i : id_t) i != 0 imply Train(i).Stop)`
- `A[] forall (i : id_t) forall (j : id_t) Train(i).Cross && Train(j).Cross imply i == j`
- `A[] Gate.list[N] == 0`
- `Train(0).Appr --> Train(0).Cross`
- `Train(1).Appr --> Train(1).Cross`
- `Train(2).Appr --> Train(2).Cross`
- `Train(3).Appr --> Train(3).Cross`
- `Train(4).Appr --> Train(4).Cross`
- `Train(5).Appr --> Train(5).Cross`
- `A[] not deadlock`

Jednoduchá varianta NIM

- NIM je hra založená na logice a strategii.
- Hrají 2 hráči.
- Hráč při svém tahu odstraní jednu až MAX (2) věci (zápalky, protony) z řady.
- Vyhrává ten hráč, který odstraní poslední věc.

Klasická varianta NIM



- NIM je hra založená na logice a strategii.
- Hrají 2 hráči.
- Hráči odebírají objekty z různých hromádek/řad.
- Hráč musí odstranit při svém tahu alespoň jeden objekt.
- Hráč při svém tahu odstraní libovolný počet objektů, které náleží všechny k jedné hromádce.
- Základní varianty hry:
  - **Normální** . . . Vyhrává ten hráč, který odstraní poslední věc.
  - **Prohra** . . . Prohrává ten hráč, který odstraní poslední věc.

Literatura I

[BDL05] Gerd Behrmann, Alexandre David, and Kim G. Larsen. A tutorial on UPPAAL, updated 25th october 2005. Technical report, Department of Computer Science, Aalborg University, Denmark, October 2005.

[Dav05] UPPAAL tutorial at rtss'05 (), December 2005.

[UPP09] UPPAAL 4.0: Small tutorial, November 2009.

[UPP10] Tool environment for validation and verification of real-time systems (UPPAAL pamphlet). http://www.it.uu.se/research/group/darts/papers/texts/uppaal-pamphlet.pdf, September 2010.

[Voj10] Tomas Vojnar. Formal analysis and verification. Lecture handouts, http://www.fit.vutbr.cz/study/courses/FAV/public/, August 2010.

[Wik10] Linear temporal logic. http://en.wikipedia.org/wiki/Linear_temporal_logic, November 2010.