

Ověřování modelů II

Radek Mařík

ČVUT FEL

Katedra telekomunikační techniky, K13132

6. prosince 2017



- 1 Temporální logiky
 - LTL logika
- 2 UPPAAL detaily
 - Jazyk modelů
 - Vlastnosti ověřování modelů
 - Čas v UPPAAL
 - Urgentní přechody UPPAAL
- 3 UPPAAL příklady
 - Přejezd vlaků přes most



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 větvení



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.



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 ... $P.l$
 - 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 přijmuta přijímačem.
- v UPPAAL: $E \langle \rangle \varphi$



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
 - $A \square \varphi \equiv \neg E \diamond \neg \varphi \dots \varphi$ by měla být pravdivá ve všech dosažitelných stavech
 - $E \square \varphi \dots$ existuje maximální cesta, podél které φ je vždy pravdivá



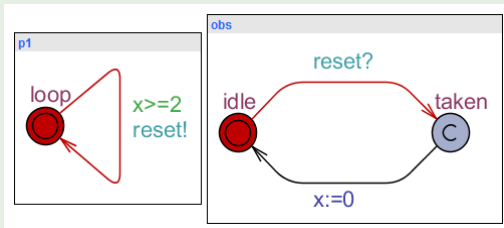
- něco jednoho dne určitě nastane
- příklady
 - stisknutí tlačítka *on* na dálkovém ovladači způsobí, že se televize jednou zapne.
 - v modelu komunikačního protokolu:
jakákoliv vyslaná zpráva bude jednou přijmuta.
- v UPPAAL:
 - $A\langle\rangle\varphi \equiv \neg E\Box\neg\varphi \dots \varphi$ bude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



Pozorovatel ^[BDL05]

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

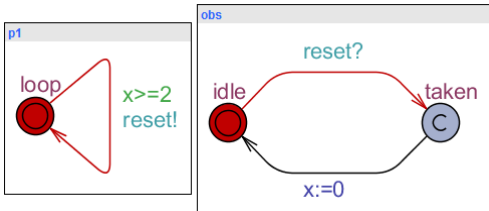
Příklad



- 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();

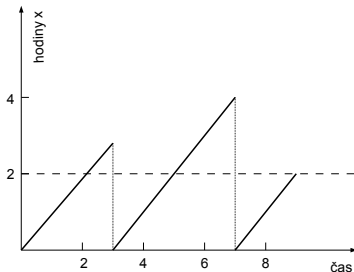
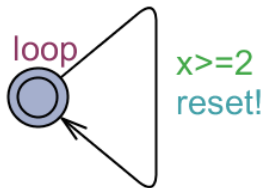
// List one or more processes to be comp
system p1, obs;

```

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



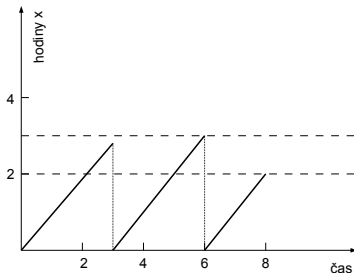
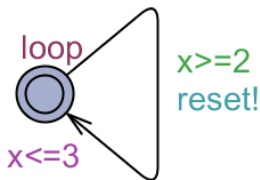
1. varianta příkladu ^[BDL05]



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



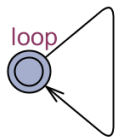
2. varianta příkladu ^[BDL05]



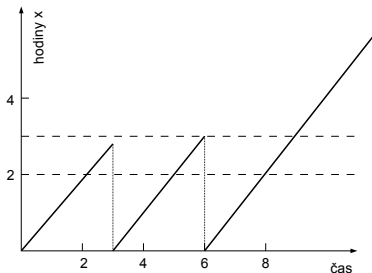
- Cílem je zůstat v pozici, pokud platí podmínka na hodinách a poté opustit pozici
- Varianta 2: s invariantem
- $A[]$ obs.taken imply $(x \geq 2$ and $x \leq 3)$
- $E\langle\rangle$ obs.idle and $x > 2$
 - $E\langle\rangle$ obs.idle and $x > 3$...neplatí
- $A[]$ obs.idle imply $x \leq 3$



3. varianta příkladu ^[BDL05]

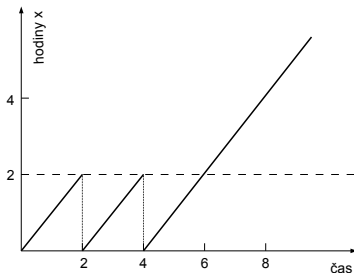
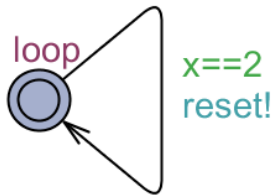


$x \geq 2$ and $x \leq 3$
reset!



- 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

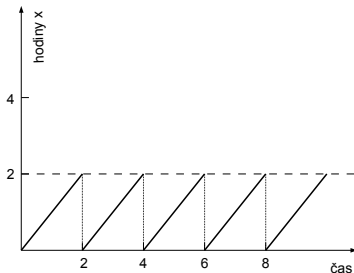
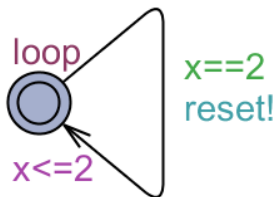


4. varianta příkladu ^[BDL05]

- 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

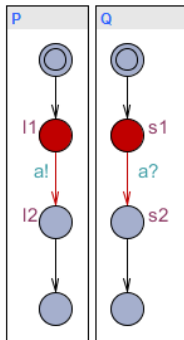


5. varianta příkladu ^[BDL05]



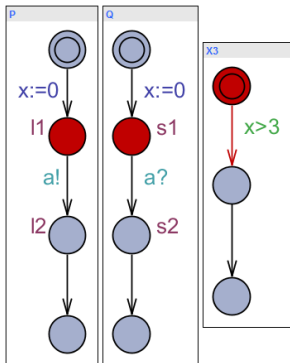
- Cílem je zůstat v pozici, pokud platí podmínka na hodinách a poté opustit pozici
- Varianta 5: s invariantem a se stráží s rovností
- $A[]$ obs.taken imply $x == 2$
- $E \langle \rangle$ obs.idle and $x > 2$...neplatí
- $A[]$ obs.idle imply $x \leq 2$



Příklad 1, procesy P, Q [Dav05]

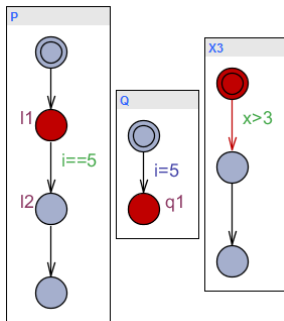
- Cílem je provést přechod se synchronizací co možná nejdříve.
- tj. jakmile jsou oba automaty P a Q připraveny (současně v pozicích l_1 a s_1).
- Jak zvolit model, když se do pozic dostanou v jiný okamžik?



Příklad 1, procesy P , Q , $X3$ [Dav05]

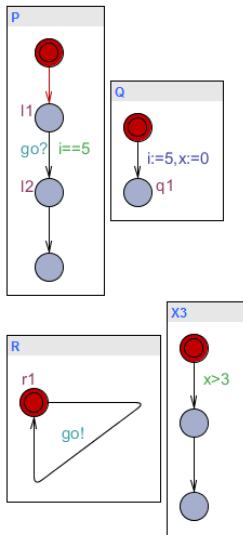
- Cílem je provést přechod se synchronizací co možná nejdříve.
- tj. jakmile jsou oba automaty P a Q připraveny (současně v pozicích l_1 a s_1).
- Jak zvolit model, když se do pozic dostanou v jiný okamžik?
- **Řešení:** urgent chan a



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!

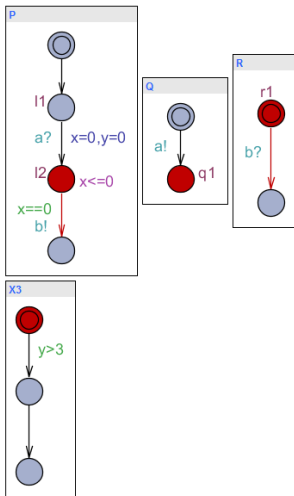


Urgentní kanály ^[Dav05]

- 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.



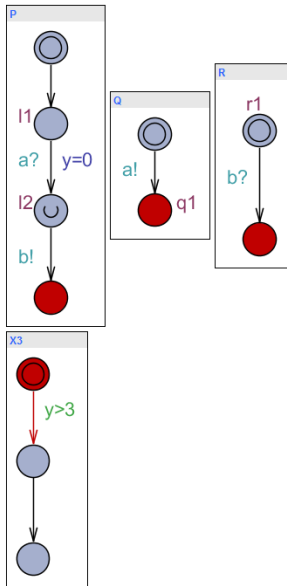
Urgentní pozice pomocí hodin [Dav05]



- Předpokládejme, že modelujeme jednoduchý 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 jednoduchý 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.

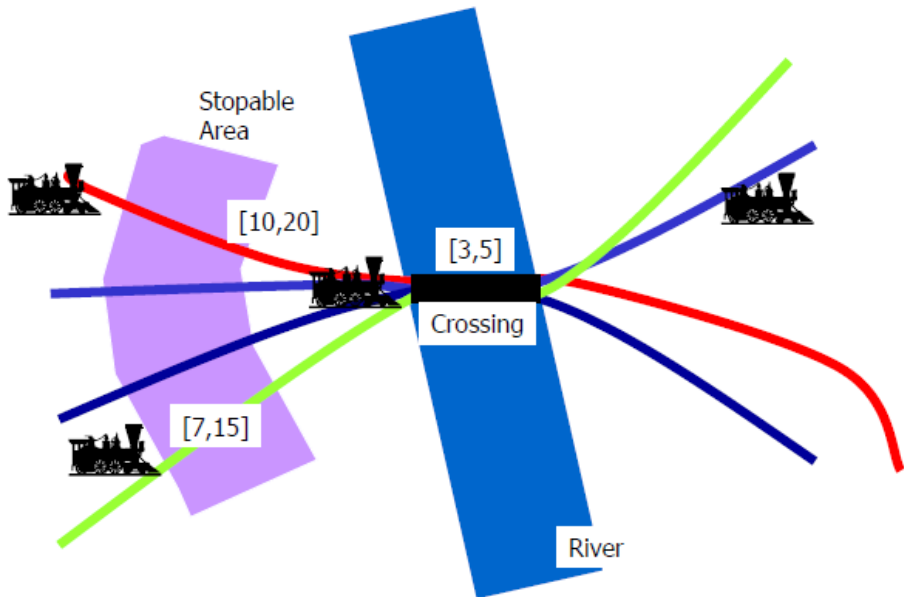


- **Poznámka:**

- Použití urgentních 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.



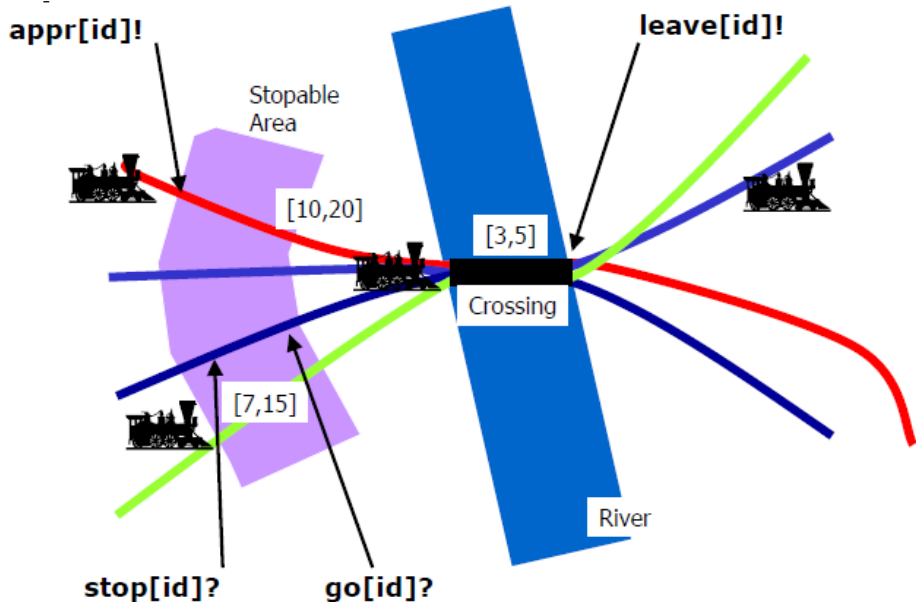
Časování a komunikace ^[BDL05]

Časová omezení a komunikace

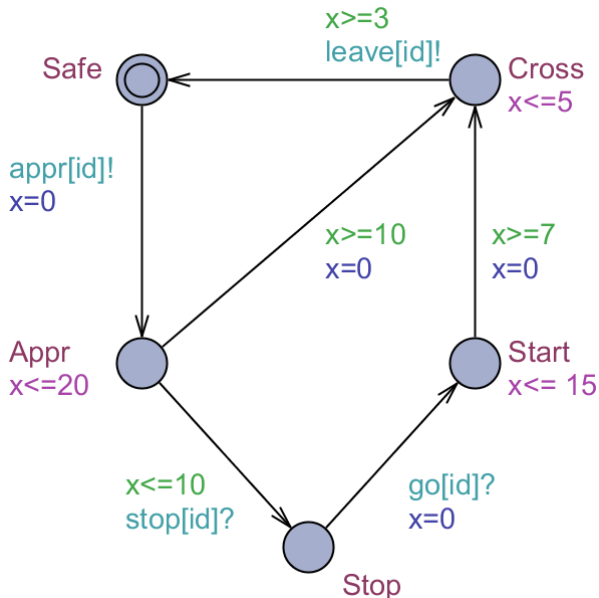
- při příjezdu k mostu vlak včas vyšle **appr!** signál
- poté vlak má 10 časových jednotek, aby přijal signál k zastavení
 - umožňuje bezpečné zastavení před mostem
- po těchto 10 časových jednotkách, trvá dalších 10 jednotek, než vlak dojde k mostu, pokud není zastaven
- jestliže je vlak zastaven,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



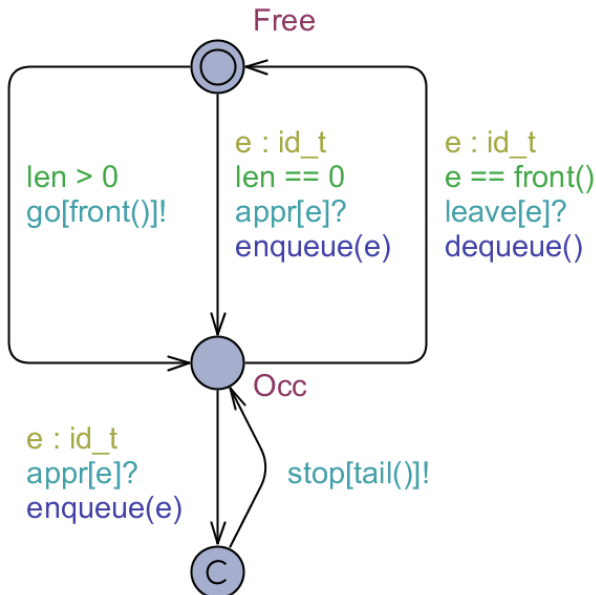
Synchronizační signály [BDL05]



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`



Literatura I



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.



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



Tomas Vojnar.

Formal analysis and verification.

Lecture handouts, <http://www.fit.vutbr.cz/study/courses/FAV/public/>, August 2010.

