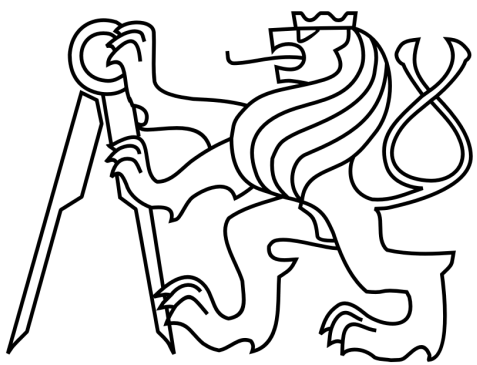
Formální Metody a Specifikace (LS 2011)
Přednáška 7:
Formální metody pro kyber-fyzikální systémy

Stefan Ratschan, Tomáš Dzetkulič

Katedra číslicového návrhu
Fakulta informačních technologií
České vysoké učení technické v Praze

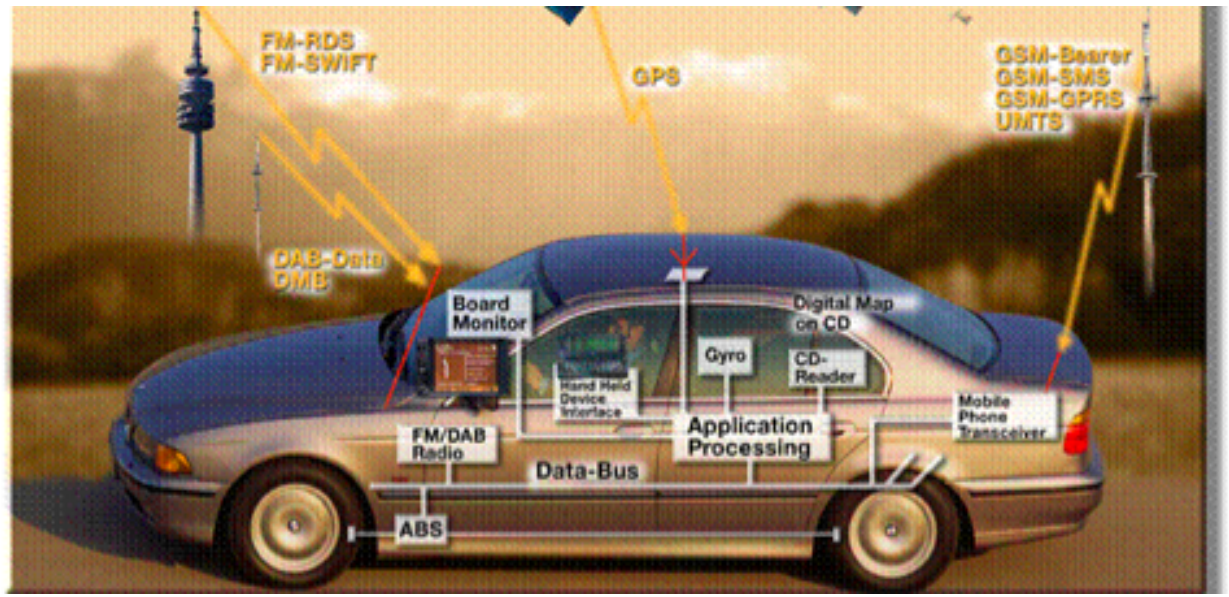
1. duben 2011



Vestavěné systémy

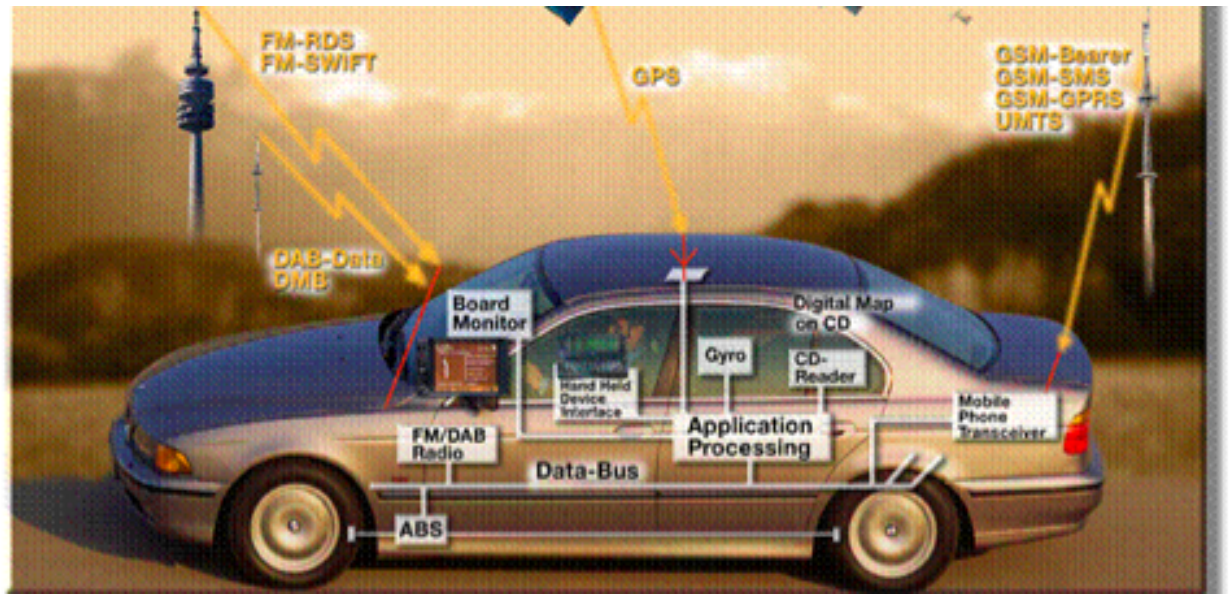
Vestavěné systémy

Převážná většina všech mikroprocesorů se nevyskytuje v desktop počítačích ale vestavěná v technických systémech (např. ve vlacích, autech, robotech, ve Vaší pračce)



Vestavěné systémy

Převážná většina všech mikroprocesorů se nevyskytuje v desktop počítačích ale vestavěná v technických systémech (např. ve vlacích, autech, robotech, ve Vaší pračce)



Náklady na elektroniku v autech brzy víc než 50% (Zpráva pro Evropskou komisi, 2005)

Další integrace

Tradiční vestavěné systémy:

Výpočetní technika řídí fyzikální systém,

jasná hranice mezi obojí.

Další integrace

Tradiční vestavěné systémy:

Výpočetní technika řídí fyzikální systém,

jasná hranice mezi obojí.

Víc a víc:

Těsná a všudypřítomná integrace mezi

výpočetními a fyzikálními komponenty:

Kyber-fyzikální systémy (cyber-physical systems)

Správnost kyber-fyzikálních systémů

Nejen těsná integrace mezi výpočetními a fyzikálními komponenty,
ale i těsná integrace do každodenního lidského života.

Správnost kyber-fyzikálních systémů

Nejen těsná integrace mezi výpočetními a fyzikálními komponenty, ale i těsná integrace do každodenního lidského života.

Selhání může ohrožovat lidský život (např. letadla)

Správnost kyber-fyzikálních systémů

Nejen těsná integrace mezi výpočetními a fyzikálními komponenty,
ale i těsná integrace do každodenního lidského života.

Selhání může ohrožovat lidský život (např. letadla)

Správnost nepostradatelná (*safety-critical systems*)

Např. Airbus, velká část nákladů: Zabezpečení správnosti vestavěného kódu

Správnost kyber-fyzikálních systémů

Nejen těsná integrace mezi výpočetními a fyzikálními komponenty,
ale i těsná integrace do každodenního lidského života.

Selhání může ohrožovat lidský život (např. letadla)

Správnost nepostradatelná (*safety-critical systems*)

Např. Airbus, velká část nákladů: Zabezpečení správnosti vestavěného kódu

Obrovský zájem i o nejmenší zlepšení zabezpečení správnosti

Návrh a testování kyber-fyzikálních systémů

Další problém: Izolovaný software můžeme snadno testovat.

Návrh a testování kyber-fyzikálních systémů

Další problém: Izolovaný software můžeme snadno testovat.

Ale: Jak otestujeme letadlo, jadernou elektrárnou?

Návrh a testování kyber-fyzikálních systémů

Další problém: Izolovaný software můžeme snadno testovat.

Ale: Jak otestujeme letadlo, jadernou elektrárnou?

Problémy:

- Testování probíhá na fyzikálním systému: drahé, nebezpečné.
- Můžeme testovat jen konečný,
  a v praxi velmi malý počet možných situací.

Návrh a testování kyber-fyzikálních systémů

Další problém: Izolovaný software můžeme snadno testovat.

Ale: Jak otestujeme letadlo, jadernou elektrárnou?

Problémy:

- Testování probíhá na fyzikálním systému: drahé, nebezpečné.
- Můžeme testovat jen konečný,
  a v praxi velmi malý počet možných situací.
- Chyby se najdou pozdě, musíme zopakovat drahé postupy.

Návrh a testování kyber-fyzikálních systémů

Další problém: Izolovaný software můžeme snadno testovat.

Ale: Jak otestujeme letadlo, jadernou elektrárnou?

Problémy:

- Testování probíhá na fyzikálním systému: drahé, nebezpečné.
- Můžeme testovat jen konečný,
  a v praxi velmi malý počet možných situací.
- Chyby se najdou pozdě, musíme zopakovat drahé postupy.
- Zdlouhavý postup (*time to market*).

Model based design

- Co nejdřívější používaní modelů konečného produktu *příklad*

Model based design

- Co nejdřívější používaní modelů konečného produktu *příklad*
- Souběžný vývoj fyzikálního systému a software

Model based design

- Co nejdřívější používaní modelů konečného produktu *příklad*
- Souběžný vývoj fyzikálního systému a software
- Hledání chyb už v modelu

Model based design

- Co nejdřívější používaní modelů konečného produktu *příklad*
- Souběžný vývoj fyzikálního systému a software
- Hledání chyb už v modelu
- Automatizace práce s modelem (např. simulace, testovaní)

Model based design

- Co nejdřívější používaní modelů konečného produktu *příklad*
- Souběžný vývoj fyzikálního systému a software
- Hledání chyb už v modelu
- Automatizace práce s modelem (např. simulace, testovaní)

Následky:

- Testování modelů levnější než testování konkretního výrobku.

Model based design

- Co nejdřívější používaní modelů konečného produktu *příklad*
- Souběžný vývoj fyzikálního systému a software
- Hledání chyb už v modelu
- Automatizace práce s modelem (např. simulace, testovaní)

Následky:

- Testování modelů levnější než testování konkretního výrobku.
- Chyby se najdou brzy (a nemusíme zopakovat drahé postupy).

Model based design

- Co nejdřívější používaní modelů konečného produktu *příklad*
- Souběžný vývoj fyzikálního systému a software
- Hledání chyb už v modelu
- Automatizace práce s modelem (např. simulace, testovaní)

Následky:

- Testování modelů levnější než testování konkretního výrobku.
- Chyby se najdou brzy (a nemusíme zopakovat drahé postupy).
- Větší paralelizace postupu, výrobek rychleji na trhu.

Model based design

- Co nejdřívější používaní modelů konečného produktu *příklad*
- Souběžný vývoj fyzikálního systému a software
- Hledání chyb už v modelu
- Automatizace práce s modelem (např. simulace, testovaní)

Následky:

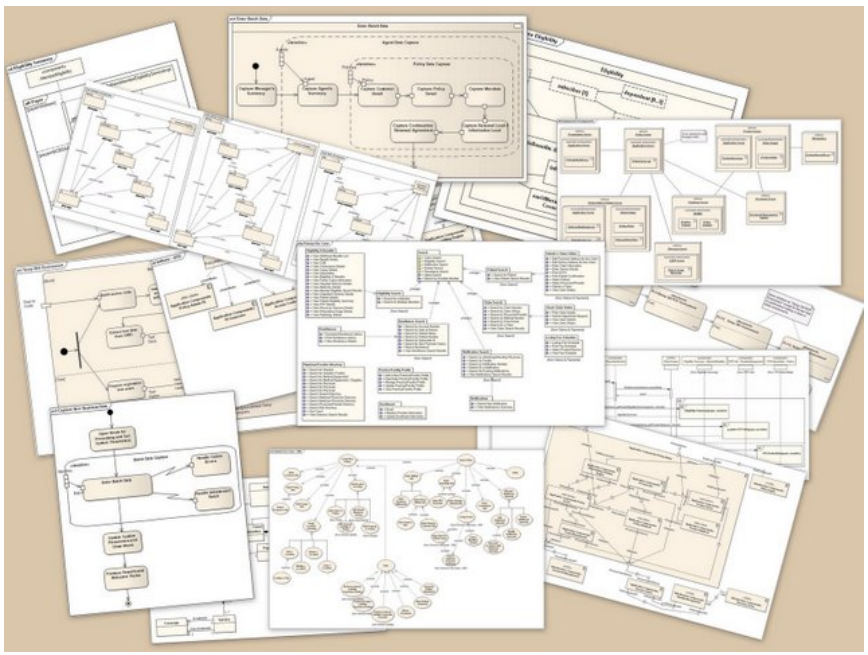
- Testování modelů levnější než testování konkretního výrobku.
- Chyby se najdou brzy (a nemusíme zopakovat drahé postupy).
- Větší paralelizace postupu, výrobek rychleji na trhu.
- Formální důkazy správnosti modelů.

Modely kyber-fyzikálních systémů

Musíme modelovat i

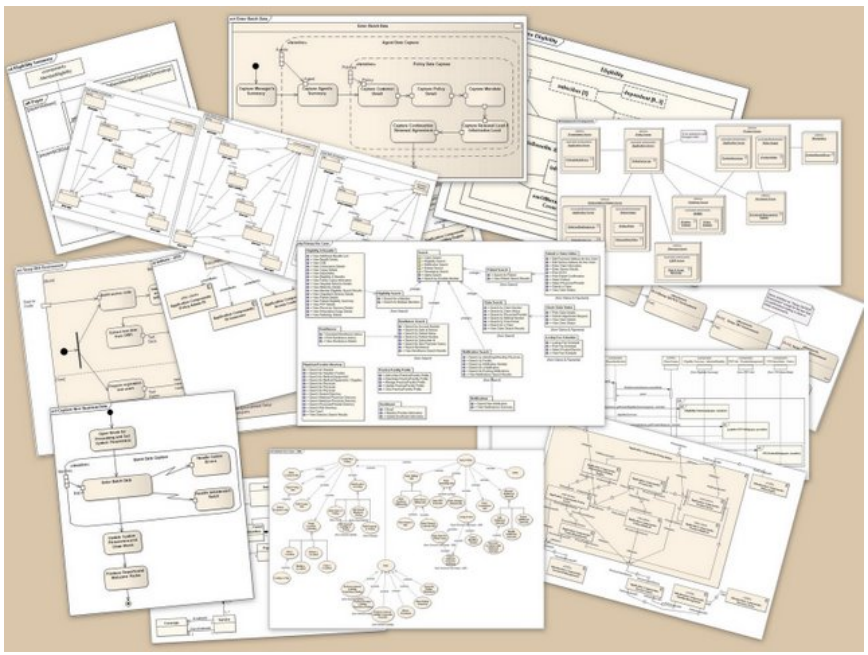
- výpočetní procesy, i
- fyzikální systém.

Modely výpočetních procesů



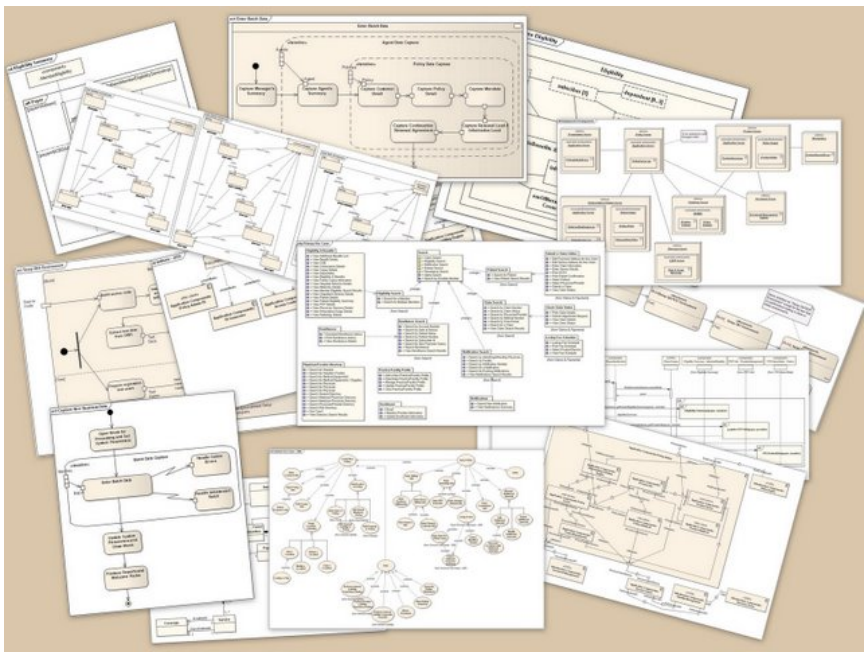
UML (unified modeling language)

Modely výpočetních procesů



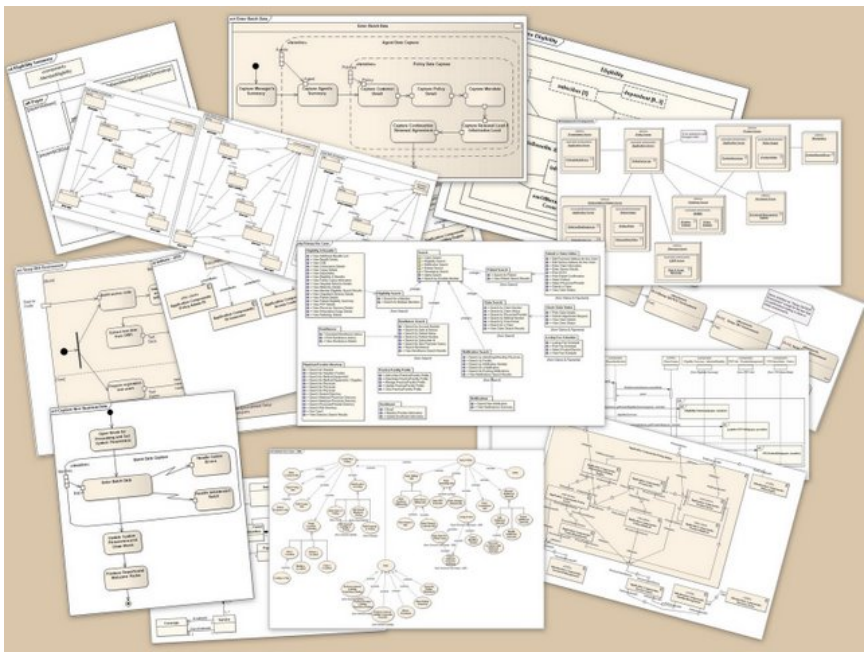
UML (unified modeling language),
konečné automaty

Modely výpočetních procesů



UML (unified modeling language), konečné automaty, C++

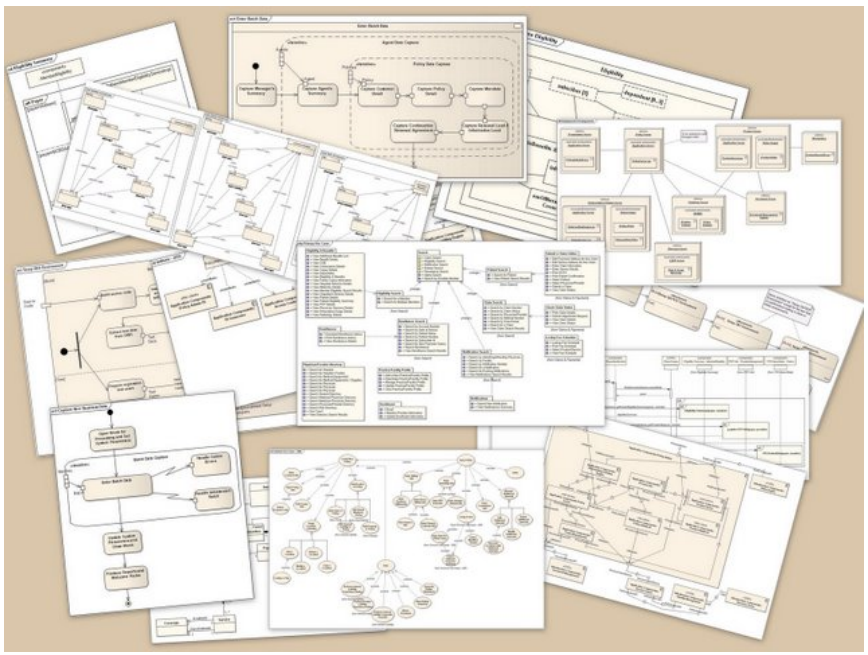
Modely výpočetních procesů



UML (unified modeling language), konečné automaty, C++

- Tradiční předmět informatiky

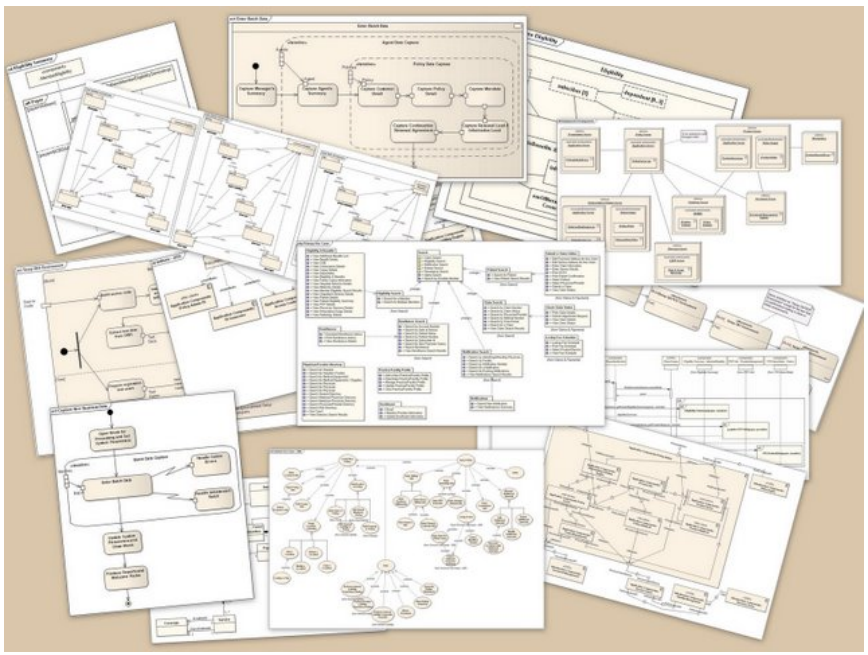
Modely výpočetních procesů



UML (unified modeling language), konečné automaty, C++

- Tradiční předmět informatiky
- Základní objekty: konečné množiny, celá čísla, pole, seznamy a podobné datové struktury (diskrétní)

Modely výpočetních procesů



UML (unified modeling language), konečné automaty, C++

- Tradiční předmět informatiky
- Základní objekty: konečné množiny, celá čísla, pole, seznamy a podobné datové struktury (diskrétní)
- Typické chyby: Array overflow, dělení nulou, ne-terminace

Modely fyzikálních procesů

$$
\begin{aligned}
\dot{x}_1 &= x_1^2 - x - 2 \\
\dot{x}_2 &= \sin x_1 \\
\dot{x}_3 &= x_1 + x_2
\end{aligned}
$$

Modely fyzikálních procesů

$$
\begin{aligned}
\dot{x}_1 &= x_1^2 - x - 2 \\
\dot{x}_2 &= \sin x_1 \\
\dot{x}_3 &= x_1 + x_2
\end{aligned}
$$

- Tradiční předmět řídící techniky, fyziky, a matematiky

Modely fyzikálních procesů

$$\begin{aligned} \dot{x}_1 &= x_1^2 - x - 2 \\ \dot{x}_2 &= \sin x_1 \\ \dot{x}_3 &= x_1 + x_2 \end{aligned}$$

- Tradiční předmět řídící techniky, fyziky, a matematiky
- Základní objekty: reálná čísla, ... (spojité, nespočetné)

Modely fyzikálních procesů

$$
\begin{aligned}
\dot{x}_1 &= x_1^2 - x - 2 \\
\dot{x}_2 &= \sin x_1 \\
\dot{x}_3 &= x_1 + x_2
\end{aligned}
$$

- Tradiční předmět řídící techniky, fyziky, a matematiky
- Základní objekty: reálná čísla, ... (spojité, nespočetné)
- Typické chyby: Dosažení nebezpečných stavů, nedosažený cílového stavu.

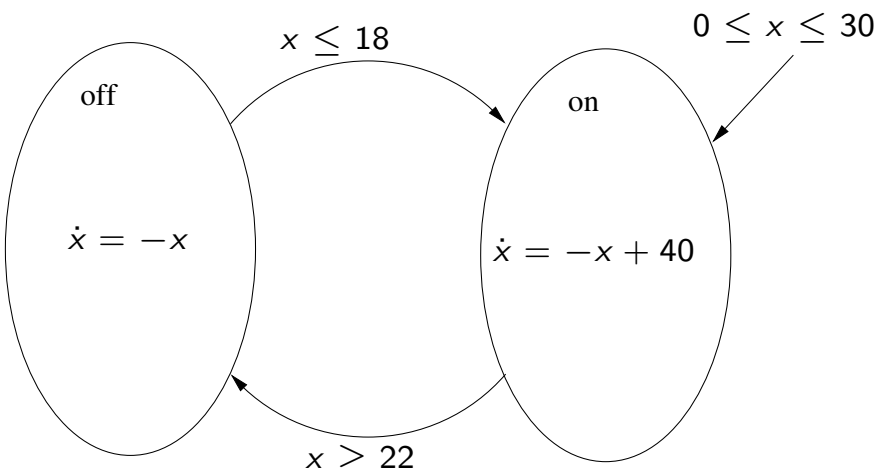
Hybridní modely

Problém: oddělené modely fyzikálních a výpočetních procesů nestačí (vzájemný vliv)

Hybridní modely

Problém: oddělené modely fyzikálních a výpočetních procesů nestačí (vzájemný vliv)

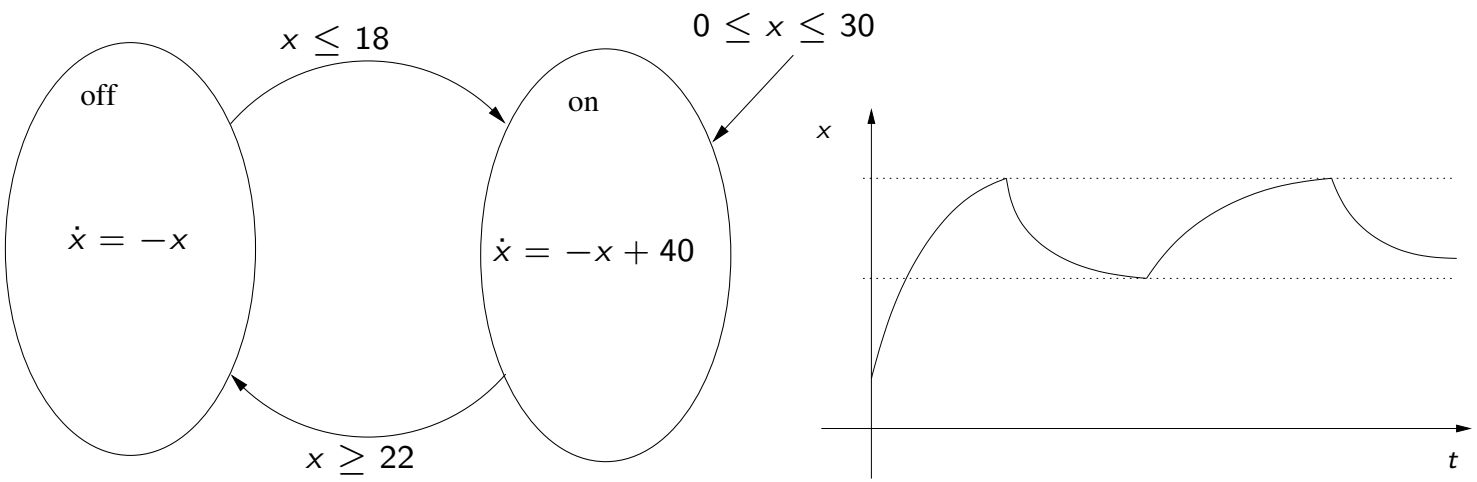
Např.: Termostat



Hybridní modely

Problém: oddělené modely fyzikálních a výpočetních procesů nestačí (vzájemný vliv)

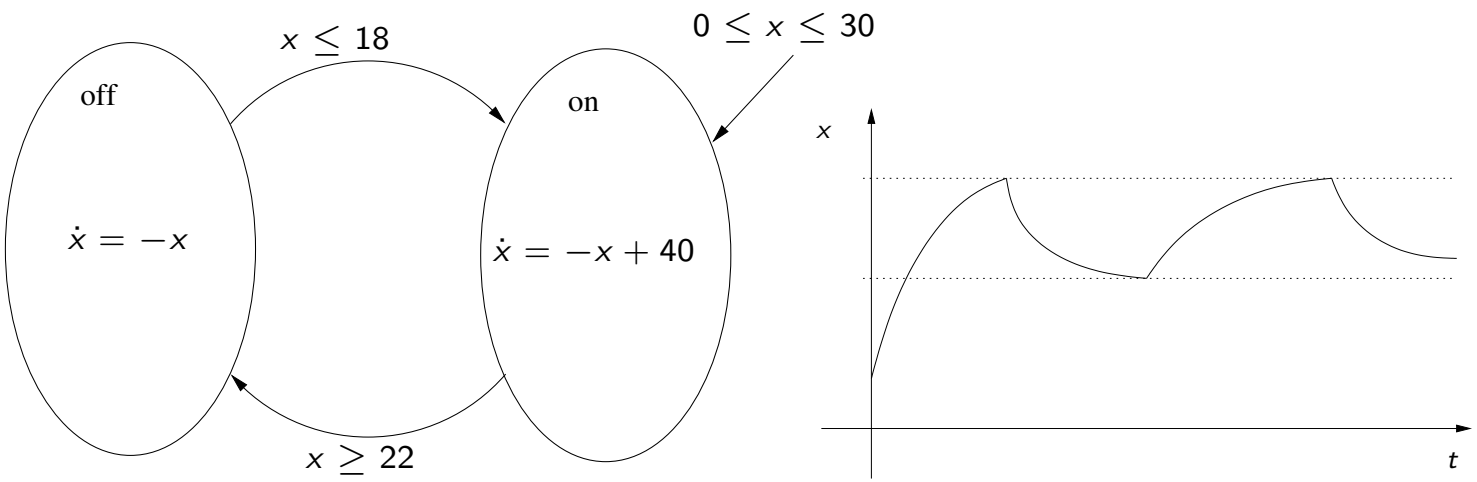
Např.: Termostat



Hybridní modely

Problém: oddělené modely fyzikálních a výpočetních procesů nestačí (vzájemný vliv)

Např.: Termostat



Hybridní systém.

Hybridní systémy: další příklady

Občas, fyzikální systém sám má částečně diskrétní povahu.

Například:

- fyzikální kontakt: míč

Hybridní systémy: další příklady

Občas, fyzikální systém sám má částečně diskrétní povahu.

Například:

- fyzikální kontakt: míč
- diskrétní zařízení v systému: vypínač, řadicí páka

Hybridní systémy: další příklady

Občas, fyzikální systém sám má částečně diskrétní povahu.

Například:

- fyzikální kontakt: míč
- diskrétní zařízení v systému: vypínač, řadicí páka
- diskrétní modelování: linearizace

Hybridní systémy: další příklady

Občas, fyzikální systém sám má částečně diskrétní povahu.

Například:

- fyzikální kontakt: míč
- diskrétní zařízení v systému: vypínač, řadicí páka
- diskrétní modelování: linearizace

Občas se spojitost vyskytuje už v čistě výpočetních procesech:

- požadavky na reálný čas: protokoly (po 9.8 sekundách, udělej ...)
- výpočet spojitých výstupů: hudba, simulace spojitých jevů
- spojitá abstrakce výpočetních systémů

Hybridní systémy: popis na základě logických formulí

Konečná množina *módů* (např. $\{\texttt{off}, \texttt{on}\}$)

Hybridní systémy: popis na základě logických formulí

Konečná množina *módů* (např. $\{\texttt{off}, \texttt{on}\}$)

Speciální proměnná *mode*

nabývající hodnotu z těchto módů (viz. čítač programu)

Hybridní systémy: popis na základě logických formulí

Konečná množina *módů* (např. $\{\texttt{off}, \texttt{on}\}$)

Speciální proměnná *mode*
nabývající hodnotu z těchto módů (viz. čítač programu)

Konečná množina proměnných nabývajících reálné hodnoty
(např. modelování teploty).

Hybridní systémy: popis na základě logických formulí

Konečná množina *módů* (např. $\{\texttt{off}, \texttt{on}\}$)

Speciální proměnná *mode*
nabývající hodnotu z těchto módů (viz. čítač programu)

Konečná množina proměnných nabývajících reálné hodnoty
(např. modelování teploty).

Popis systému, logické formule s těmito predikáty a proměnnými:

Hybridní systémy: popis na základě logických formulí

Konečná množina *módů* (např. $\{\texttt{off}, \texttt{on}\}$)

Speciální proměnná *mode*
nabývající hodnotu z těchto módů (viz. čítač programu)

Konečná množina proměnných nabývajících reálné hodnoty
(např. modelování teploty).

Popis systému, logické formule s těmito predikáty a proměnnými:

- *Jump* (např. $[mode = \texttt{off} \wedge x \geq 10] \Rightarrow [mode' = \texttt{on} \wedge x' = 0]$)
  viz. přechodová podmínka programů

Hybridní systémy: popis na základě logických formulí

Konečná množina *módů* (např. $\{\texttt{off}, \texttt{on}\}$)

Speciální proměnná *mode*
nabývající hodnotu z těchto módů (viz. čítač programu)

Konečná množina proměnných nabývajících reálné hodnoty
(např. modelování teploty).

Popis systému, logické formule s těmito predikáty a proměnnými:

- *Jump* (např. $[mode = \texttt{off} \wedge x \geq 10] \Rightarrow [mode' = \texttt{on} \wedge x' = 0]$)
  viz. přechodová podmínka programů
- *Flow* (např.
  $[mode = \texttt{off} \Rightarrow \dot{x} = x \sin(x) + 1] \wedge [mode = \texttt{on} \Rightarrow \ \ldots]$)

Hybridní systémy: popis na základě logických formulí

Konečná množina *módů* (např. $\{\texttt{off}, \texttt{on}\}$)

Speciální proměnná *mode*
nabývající hodnotu z těchto módů (viz. čítač programu)

Konečná množina proměnných nabývajících reálné hodnoty
(např. modelování teploty).

Popis systému, logické formule s těmito predikáty a proměnnými:

- *Jump* (např. $[mode = \texttt{off} \wedge x \geq 10] \Rightarrow [mode' = \texttt{on} \wedge x' = 0]$)
  viz. přechodová podmínka programů
- *Flow* (např.
  $[mode = \texttt{off} \Rightarrow \dot{x} = x \sin(x) + 1] \wedge [mode = \texttt{on} \Rightarrow \ \ldots]$)

Podobně jako u programů můžeme na základě toho definovat
jestli systém může udělat krok ze stavu s do stavu s'.

Vývoj hybridních systémů

$s \rightarrow s'$ pokud

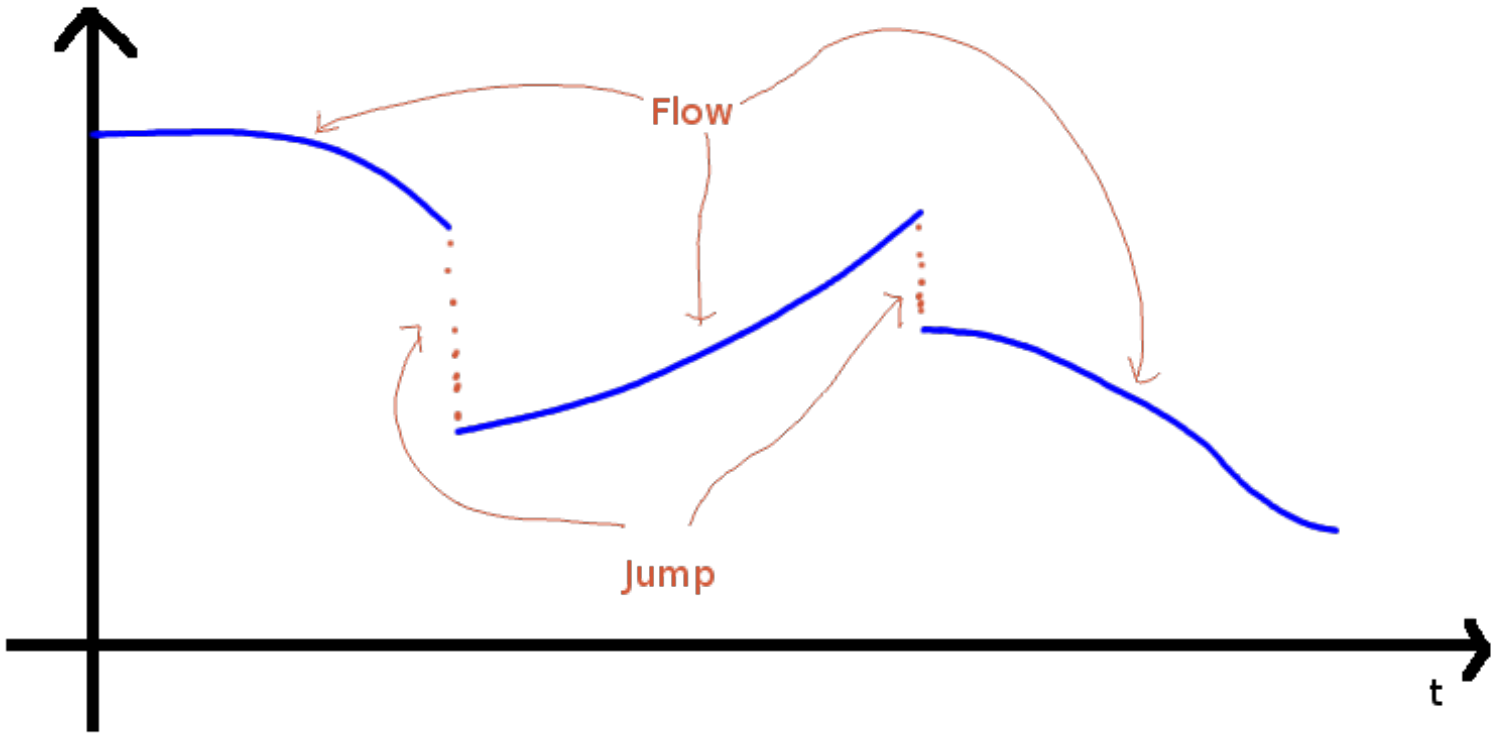
- s, s' splňují formuli *Jump*, nebo
- je možný spojitý vývoj podle *Flow*.

Vývoj hybridních systémů

$s \to s'$ pokud

- s, s' splňují formuli *Jump*, nebo
- je možný spojitý vývoj podle *Flow*.

Opět: systém může vykonat libovolný počet takovýchto kroků ($\to^*$)



Specifikace chování

Podobně jako u programů můžeme specifikovat
žádané chování hybridních systémů:

Specifikace chování

Podobně jako u programů můžeme specifikovat
žádané chování hybridních systémů:

- *Init* (e.g., $mode = \texttt{firstgear} \wedge 0 \leq x \wedge x \leq 10$)

Specifikace chování

Podobně jako u programů můžeme specifikovat
žádané chování hybridních systémů:

- *Init* (e.g., $mode = \texttt{firstgear} \wedge 0 \leq x \wedge x \leq 10$)
- *Safe* (např. $x \geq 8000$)

Specifikace chování

Podobně jako u programů můžeme specifikovat
žádané chování hybridních systémů:

- *Init* (e.g., $mode = \texttt{firstgear} \wedge 0 \leq x \wedge x \leq 10$)
- *Safe* (např. $x \geq 8000$)

A můžeme se zeptat jestli to systém splňuje, tj. pokud

- začíná v stavu splňujícím *Init*, a
- dělá libovolný počet kroků podle $\rightarrow$ (tj. *Jump*, a *Flow*)

bude vždy splňovat *Safe*?

Ověřování omezené správnosti hybridních systémů

Podobně jako u programů, můžeme zakódovat
správnost během omezeného počtu kroků do logické formule:

$$\neg\exists v_1, \ldots, v_n \,.\, \Big[\mathit{Init}(v_1) \wedge \bigwedge_{i=1,\ldots,n-1} \Phi_H[v \leftarrow v_i, v' \leftarrow v_{i+1}] \wedge \neg \mathit{Safe}(v_n) \Big]$$

Tady, Φ_H zakóduje že systém dělá jeden krok:

$$\mathit{Jump} \vee \Phi_{\mathit{Flow}},$$

přičemž *Jump* je přímo z popisu hybridního systému,
ale Φ_{Flow} je složitá formule
(nad-aproximace diferenciálních rovnic
na základě Taylorových polynómů).

Ověřování omezené správnosti hybridních systémů

Podobně jako u programů, můžeme zakódovat
správnost během omezeného počtu kroků do logické formule:

$$\neg\exists v_1, \ldots, v_n \,.\, \Big[\mathit{Init}(v_1) \wedge \bigwedge_{i=1,\ldots,n-1} \Phi_H[v \leftarrow v_i, v' \leftarrow v_{i+1}] \wedge \neg \mathit{Safe}(v_n) \Big]$$

Tady, Φ_H zakóduje že systém dělá jeden krok:

$$\mathit{Jump} \vee \Phi_{\mathit{Flow}},$$

přičemž *Jump* je přímo z popisu hybridního systému,
ale Φ_{Flow} je složitá formule
(nad-aproximace diferenciálních rovnic
na základě Taylorových polynómů).

Celá formule je formulí v teorií reálných čísel,
a můžeme je dokázat příslušnou rozhodující procedurou.

Ověřování omezené správnosti hybridních systémů

Podobně jako u programů, můžeme zakódovat
správnost během omezeného počtu kroků do logické formule:

$$\neg\exists v_1, \ldots, v_n \,.\, \Big[\mathit{Init}(v_1) \wedge \bigwedge_{i=1,\ldots,n-1} \Phi_H[v \leftarrow v_i, v' \leftarrow v_{i+1}] \wedge \neg \mathit{Safe}(v_n) \Big]$$

Tady, Φ_H zakóduje že systém dělá jeden krok:

$$\mathit{Jump} \vee \Phi_{\mathit{Flow}},$$

přičemž *Jump* je přímo z popisu hybridního systému,
ale Φ_{Flow} je složitá formule
(nad-aproximace diferenciálních rovnic
na základě Taylorových polynómů).

Celá formule je formulí v teorií reálných čísel,
a můžeme je dokázat příslušnou rozhodující procedurou.

Tudíž: Automatický důkaz omezené správnosti hybridních systémů.

Implementace správného systému

Správnost modelu nutně neznamená správnost konečného produktu

Implementace správného systému

Správnost modelu nutně neznamená správnost konečného produktu

Může se stát i pro software pro který máme důkaz správnosti.

Proč?

Implementace správného systému

Správnost modelu nutně neznamená správnost konečného produktu

Může se stát i pro software pro který máme důkaz správnosti.

Proč?

Chybný překladač, chybný operační systém, chybný čip atd..

Neomezená správnost

Ve našem výzkumu vyvíjíme metody které umí ověřovat správnost hybridních systémů přes neomezený počet kroků.

Neomezená správnost

Ve našem výzkumu vyvíjíme metody které umí ověřovat
správnost hybridních systémů přes neomezený počet kroků.

`http://hsolver.sourceforge.net`

Neomezená správnost

Ve našem výzkumu vyvíjíme metody které umí ověřovat
správnost hybridních systémů přes neomezený počet kroků.

`http://hsolver.sourceforge.net`

Demo

Neomezená správnost

Ve našem výzkumu vyvíjíme metody které umí ověřovat správnost hybridních systémů přes neomezený počet kroků.

`http://hsolver.sourceforge.net`

Demo

Příští přednáška: neomezená správnost programů.