

Jméno: Ing. Martin Kot, Ph.D.

E-mail: martin.kot@vsb.cz

Místnost: EA413

Web: <http://www.cs.vsb.cz/kot>

- **Zápočet** (30 bodů):

- Dva miniprojekty (2×15 bodů)

- Pokud je miniprojekt uznán, je hodnocen 10–15 body.
- Podmínkou získání zápočtu je alespoň 1 uznáný miniprojekt

- **Zkouška** (70 bodů)

- Nejprve je potřeba vyřešit jeden příklad podobný příkladům ze cvičení označených symbolem hvězdičky
- Poté následuje ústní zkouška - cca 20 minut dlouhá prezentace jednoho z 5 předem známých témat. U této prezentace je možné využívat donesené rukou psané podklady v rozsahu maximálně 1 strany A4 na téma.

- Přechodové systémy
- Kalkulus komunikujících systémů (CCS)
- Ekvivalence chování - především silná a slabá bisimulační ekvivalence
- Hennessy-Milner logika
- Tarského věta o pevném bodu
- Hennessy-Milner logika s rekurzí
- Časované automaty a jejich sémantika
- Sítě časovaných automatů
- Temporální logiky

Hlavním výukovým textem je:

- Luca Aceto, Anna Ingólfssdóttir, Kim Guldstrand Larsen, Jiří Srba
Reactive Systems: Modelling, Specification and Verification,
Cambridge University Press, August 2007 .

Poznámka: Většinu knihy dala jedna z autorek k dispozici ve formě pdf souboru na adrese

<http://www.cs.ioc.ee/yik/schools/win2007/ingolfsdottir>

- Testování - snaha najít chyby
 - Běh programu na vybraných vstupech sloužící k odhalení chyb
 - Výhody: vhodné k odhalení chyb, relativně levné
 - Nevýhody: obvykle negarantuje bezchybnost
- Verifikace - snaha formálně dokázat korektnost
 - Formálními metodami ověřuje systém
 - Výhody: částečně vhodná k odhalení chyb, může garantovat korektnost
 - Nevýhody: obvykle drahá (na čas, prostředky, kvalifikované pracovníky, ...)
- Typy chyb:

	časté	vzácné
neškodné	testování	nedůležité
katastrofické	testování, verifikace	verifikace

- Raketa Ariane 5
 - výbuch při prvním letu 37 sekund po startu
 - konverze 64-bit float na 16-bit signed integer
 - využití části kódu ze SW pro Ariane 4 bez řádné verifikace
 - na palubě byly družice Cluster za 500 milionů dolarů
- Therac-25
 - počítačem řízený přístroj na radioterapii z roku 1982
 - chyby v souběhu (concurrent programming errors)
 - několik pacientů vystaveno až 100násobně vyšším dávkám radiace
 - dřívější modely měly HW pojistku, tento spoléhal na SW kontrolu
 - několik případů nemoci z ozáření, 3 pacienti na následky zemřeli
- Mars Pathfinder
 - využíval VxWorks RTOS
 - časté restarty způsobené chybně navrženými prioritami procesů a prioritami přístupu ke sdílenému médiu (komunikační sběrnici)
 - řešení: program v C změnil 3 globální proměnné z False na True

- Nové mezinárodní letiště v Denveru
 - pokus o první plně automatický zavazadlový systém
 - chyby v návrhu (podcenění složitosti projektu, nevytvoření záložních kapacit)
 - oddálilo otevření letiště o 16 měsíců
 - náklady na údržbu neotevřeného letiště a úroky z půjček ve výši 1,1mil dolarů denně
 - nakonec se systém nějakou dobu využíval jen ve velmi omezené míře (1 terminál ze 3, 1 aerolinie) a zbytek se řešil tradičně (manuálně)
- Intel Pentium FDIV bug
 - chyba v návrhu HW jednotky pro práci s plovoucí řádovou čárkou (FPU - floating point unit)
 - objevena v květnu 1994 (procesory vyráběny od března 1993)
 - chyba se projevila v asi 1 z 9 miliard dělení s náhodnými hodnotami
 - Intel nabídl v prosinci 1994 výměnu procesorů
 - přímá škoda - 475 milionů dolarů za výměnu procesorů

- Ruční - člověk se snaží vytvořit důkaz korektnosti
- Poloautomatické
 - Dokazování vět (theorem proving)
 - nástroj COQ - interaktivní, využit pro CompCert (překladač jazyka C)
- Automatické
 - Ověřování ekvivalence (equivalence checking)
 - algoritmus porovnává 2 systémy (programy, modely)
 - ověřuje, zda se systémy chovají stejně (z vnějšího pohledu)
 - využívají se ekvivalence chování (behavioral equivalence)
 - použití např. na porovnání implementace vs. specifikace
 - Ověřování modelů (model checking)
 - algoritmus zjišťuje zda systém (program, model) má danou vlastnost
 - využívají se různé logiky (modální, temporální) pro popis vlastností

Charakterizace klasického programu

Program transformuje vstup na výstup.

- Denotační sémantika: popis významu programu pomocí funkcí
 - $\text{program} :: \text{Vstup} \rightarrow \text{Výstup}$
 - využívá λ -kalkul
 - popis prostředky funkcionálního jazyka
- **Nekonečný běh programu je považován za chybu!**
- V případě konečného běhu je výsledek jednoznačný.

- Operační systémy
- Komunikační protokoly
- Řídící programy
 - Výrobní linky
 - Automobily
 - Chytrá domácnost
- Prodejní automaty, bankomaty

Charakterizace reaktivního systému

Reaktivní systém průběžně reaguje na stimuly z jeho okolí.

- Klíčové záležitosti:
 - komunikace a interakce
 - paralelismus
- **Nekonečný běh programu je většinou velmi žádoucí!**
- Pokud dává nějaké výstupy, nemusí být jednoznačné.

Klasické vs. reaktivní systémy

	Klasické	Reaktivní/paralelní
Interakce	ne	ano
Nekonečný běh	nežádoucí	často žádoucí
Jednoznačný výsledek	ano	ne
Sémantika	denotační, operační	?

Definice

Ohodnocený přechodový systém (labelled transition system - LTS) je trojice $(Proc, Act, \{\xrightarrow{a} \mid a \in Act\})$, kde:

- $Proc$ je množina **stavů** (nebo **procesů**)
 - Act je množina **akcí** (or **actions**)
 - pro každé $a \in Act$, $\xrightarrow{a} \subseteq Proc \times Proc$ je binární relace na stavech nazývaná **přechodová relace**
-
- Budeme používat infixovou notaci $s \xrightarrow{a} s'$ znamenající $(s, s') \in \xrightarrow{a}$.
 - Někdy je navíc určen **počáteční** stav.

Nechť $(Proc, Act, \{\xrightarrow{a} \mid a \in Act\})$ je LTS.

- rozšíříme \xrightarrow{a} na prvky Act^*
- $\longrightarrow = \bigcup_{a \in Act} \xrightarrow{a}$
- \longrightarrow^* je reflexivní tranzitivní uzávěr \longrightarrow
- $s \xrightarrow{a}$ – existuje s' tž. $s \xrightarrow{a} s'$
- $s \not\xrightarrow{a}$ – neexistuje s' tž. $s \xrightarrow{a} s'$

- LTS mohou být nekonečně velké
- LTS jsou sémantikou reaktivních systémů
- Potřebujeme vhodnou syntaxi pro reaktivní systémy
- Používá se mnoho různých modelů reaktivních systémů:
 - různé typy automatů
 - Kripkeho struktury - stavy mají přiřazené atomické výroky, které v nich platí
 - automaty na nekonečných slovech - Büchi, Muller, Rabin, Streett
 - bezkontextové gramatiky (v Greibachové normální formě)
 - Petriho sítě
 - procesní algebry
 - kalkulus komunikujících systémů (CCS)

Příklady použití bezkontextových gramatik

$$A \longrightarrow b$$

$$A \longrightarrow bAB$$

$$B \longrightarrow a$$

Příklady použití bezkontextových grammatik

$$A \longrightarrow b$$

$$A \longrightarrow bAB$$

$$B \longrightarrow a$$

$$A \xrightarrow{b} \varepsilon$$

$$A \xrightarrow{b} AB$$

$$B \xrightarrow{a} \varepsilon$$

Příklady použití bezkontextových grammatik

$$\begin{array}{ll} A \longrightarrow b & A \xrightarrow{b} \varepsilon \\ A \longrightarrow bAB & A \xrightarrow{b} AB \\ B \longrightarrow a & B \xrightarrow{a} \varepsilon \end{array}$$

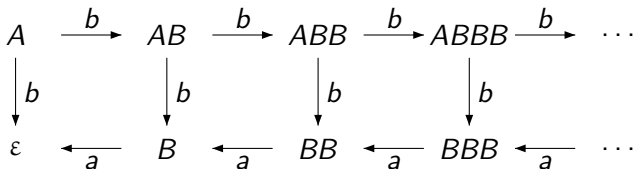
- BPA - Basic Process Algebra

$$\begin{array}{ccccccc} A & \xrightarrow{b} & AB & \xrightarrow{b} & ABB & \xrightarrow{b} & ABBB \xrightarrow{b} \dots \\ \downarrow b & & \downarrow b & & \downarrow b & & \downarrow b \\ \varepsilon & \xleftarrow{a} & B & \xleftarrow{a} & BB & \xleftarrow{a} & BBB \xleftarrow{a} \dots \end{array}$$

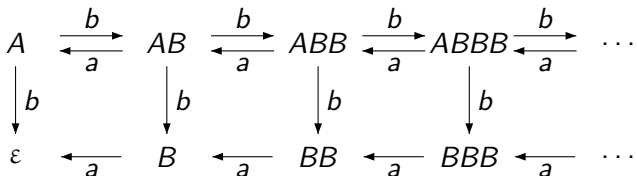
Příklady použití bezkontextových grammatik

$$\begin{array}{ll} A \longrightarrow b & A \xrightarrow{b} \varepsilon \\ A \longrightarrow bAB & A \xrightarrow{b} AB \\ B \longrightarrow a & B \xrightarrow{a} \varepsilon \end{array}$$

- BPA - Basic Process Algebra



- BPP - Basic Parallel Processes



Základní princip

- 1 Definovat několik **atomických procesů** (modelování nejjednoduššího chování procesů).
- 2 Definovat **operace** (tvorba složitějších chování procesů z jednodušších).

Příklad

- 1 atomická instrukce: přiřazení (např. $x:=2$ a $x:=x+2$)
- 2 nové operátory:
 - sekvenční kompozice ($P_1; P_2$)
 - paralelní kompozice ($P_1 \mid P_2$)

Nyní např. $(x:=1 \mid x:=2); x:=x+2; (x:=x-1 \mid x:=x+5)$ je proces.