

IV101 – Seminář z Verifikace

Jiří Barnat

Organizace kurzu

Cíl kurzu

Katalog předmětů:

V rámci semináře se studenti seznámí s několika nejpoužívanějšími verifikačními nástroji, vhodnými formalizmy pro vyjádření vlastností systémů a vypracují verifikační projekt v rozsahu 15 hod.

Kontext v rámci FI

- IV113 Úvod do validace a verifikace
- IA159 Formal Verification Methods
- Laboratoř ParaDiSe, Modální a Temporální Logiky, Návrh a Verifikace algoritmů, Modelování a simulace

Obsah kurzu

- Úvod do formální verifikace
- Principy modelování
- Seznámení se s několika verifikačními nástroji
- Samostatná práce na verifikačních projektech

Ukončení předmětu

Požadavky

- Vypracování zadaných úkolů
- Schopnost práce s verifikačními nástroji

Schopnost práce s verifikačními nástroji

- Instalace a provoz
- Modelovací jazyk
- Specifikační jazyk

Automatizovaná verifikace

Motivace pro verifikaci

Chceme produkovat

- Bezchybné systémy
- Bezpečné systémy

Chceme zlevnit výrobu

- Úspory na záplatách
- Rychlejší vývoj

Nejdražší softwarové chyby

- Ariane 5, Pentium FDIV bug, Therac-25, ...

Motivace pro automatizovanou verifikaci

- Automatizovaná = s podporou SW nástrojů
- Analýza komplexních systémů
- Částečná eliminace lidského faktoru
- Snadno opakovatelná procedura
- Zrychlení procesu
- ...

Cesty k dosažení spolehlivého produktu

Programátorské techniky

- Ošetření výjimek
- Robustní programování
- Inspekce kódu
- “Štábní” kultura a disciplína
- Paralelní vývoj záložní kopie separátní skupinou
- Využití možností IDE
- ...

Cesty k dosažení spolehlivého produktu

Simulace

- **Pozorování jednoho běhu systému**
- Externí/interní pozorovatel
- Pozorují se statistiky, výsledky, ...
- Profilace (valgrind, callgrind, ...)
- Základní metoda verifikace v okamžiku dosažení bodu, kdy existuje proveditelný kód/model

Cesty k dosažení spolehlivého produktu

Testování

- Odvozování vlastností systému na základě pozorování vybrané množiny běhů
- Opakovaná simulace
- Rychlé odhalení největších chyb
- Nalézá chyby, nedává garanci bezchybnosti
- Může probíhat nad skutečnou implementací
- Ne vždy je realizovatelné
- Existují a používají se podpůrné nástroje
- Chytré metriky na výběr množiny běhů

Nedokonalost testování

- Je obtížné identifikovat/specifikovat všechny kombinace možných vstupů systému.
- Některé systémy, přestože nabývají konečně mnoha různých stavů, mohou provádět nekonečně mnoho různých běhů.
- Souběžné systémy se chovají nedeterministicky (způsobeno vnějšími faktory), chyba systému se nemusí během testování projevit.

Cesty k dosažení spolehlivého produktu

Automatizovaná formální verifikace

Establishing properties of hardware or software designs using logic, rather than (just) testing or informal arguments.

- Dáva garanci
- Neprobíhá nad skutečnou implementací
- Nutné modelování a abstrakce

Metody formální verifikace

- Dokazování (theorem proving)
- Ověřování ekvivalence (equivalence checking)
- Ověřování modelu (model checking)

Dokazování

- Důkaz je konečná posloupnost logicky korektních kroků vycházejících z axiomů a konče dokazovaným tvrzením
- Neexistence důkazu neimplikuje neplatnost tvrzení
- Extrémně náročné na odbornou kvalitu lidí a čas
- Nedostupné v masovém měřítku
- Těžko realizovatelné bez podpůrných nástrojů
- Dokázána korektnost FPU jednoho modelu procesoru AMD
- Nástroje se označují jako *theorem provery*

Ekvivalence checking

- Ověření ekvivalence specifikace a implementace vzhledem k dané relaci
- Nejčastěji používaná ekvivalence je bisimulace
 - Vzájemná shoda ve struktuře pozorovatelného chování
- Různé úrovně abstrakce
- Formálně vymezené obě strany
 - Formálně specifikovat systém může být stejně těžké jako ho implementovat
 - Omezuje praktickou aplikovatelnost
- Používá se především při verifikaci HW a systémů složených z komponent

Model Checking

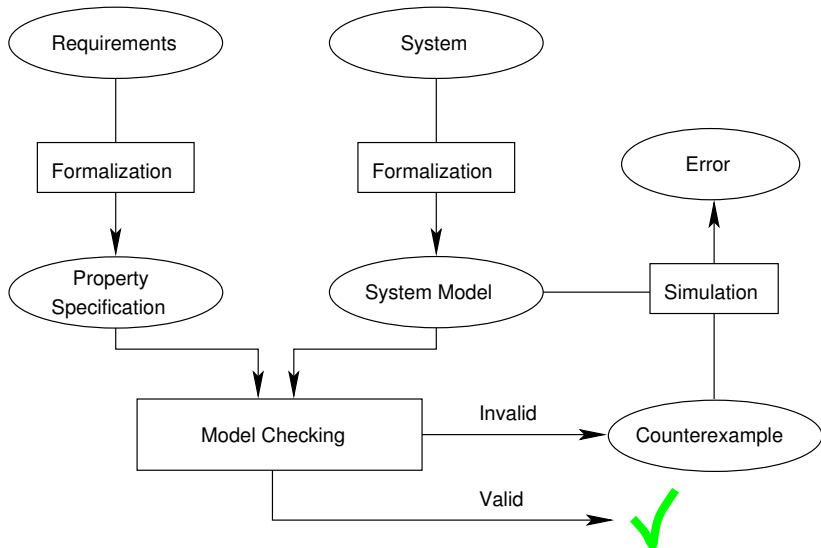
Model checking

- Systematické zkoumání všech možných běhů systému.
- Vyžaduje, aby systém nabýval konečně mnoha různých stavů, anebo s nějakým takovým systémem korespondoval.
- Zkoumaný systém je nutné formalizovat a to včetně prostředí, které systém ovlivňuje.

Model Checking

- Automatizovaná procedura ověřující jednu konkrétní vlastnost systému.

Model checking schéma



Výhody a nevýhody metody

Nevýhody

- Zkoumaný systém je nutné pro účely provedení automatizované analýzy modelovat.
- Z platnosti dokazovaných vlastností neplyne bezchybnost systému.
- Vlastnosti systému je nutné formálně specifikovat, nejčastěji s využitím nějaké temporální logiky.

Výhody

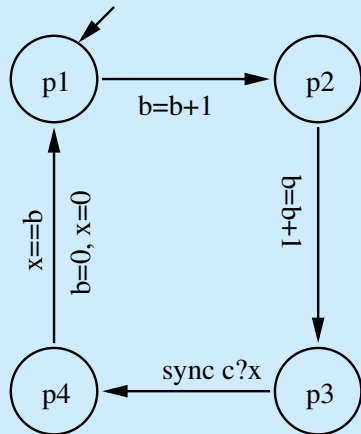
- Pracuje se s modelem, nevyžaduje funkční implementaci.
- Je možné aplikovat na navrhované koncepty ve velmi ranném stádiu vývoje produktu.

Příklad modelovacího jazyka (DiVinE)

- Konečný automat
 - Stavy (Lokace)
 - Iničiální stav
 - Přechody
 - (Akceptující stavy)
- Přechody rozšířené o
 - Strážce (guards)
 - Komunikační primitiva
 - Efekt
- Lokální proměnné
 - integer, byte
 - channel

Process B

byte b,x;



State: $[\]$; A:[q1, a:0]; B:[p1, b:0, x:0]
 0 $\langle 0.0 \rangle$: q1 \rightarrow q2 { effect a = a+1; }
 1 $\langle 1.0 \rangle$: p1 \rightarrow p2 { effect b = b+1; }
 Command:1

State: $[\]$; A:[q1, a:0]; B:[p2, b:1, x:0]
 0 $\langle 0.0 \rangle$: q1 \rightarrow q2 { effect a = a+1; }
 1 $\langle 1.1 \rangle$: p2 \rightarrow p3 { effect b = b+1; }
 Command:1

State: $[\]$; A:[q1, a:0]; B:[p3, b:2, x:0]
 0 $\langle 0.0 \rangle$: q1 \rightarrow q2 { effect a = a+1; }
 Command:0

State: $[\]$; A:[q2, a:1]; B:[p3, b:2, x:0]
 0 $\langle 0.1 \rangle$: q2 \rightarrow q3 { effect a = a+1; }
 Command:0

State: $[\]$; A:[q3, a:2]; B:[p3, b:2, x:0]
 0 $\langle 0.2 \& 1.2 \rangle$: q3 \rightarrow q1 { sync c!a; effect a = 0; }
 p3 \rightarrow p4 { sync c?x; }
 Command:0

State: $[\]$; A:[q1, a:0]; B:[p4, b:2, x:2]

