IV113 Validace a verifikace

Lehký úvod do analýzy programů

Jiří Barnat

Analýza programů

Cíle programové analýzy

- Odvodit vlastnosti programů z jejich zdrojového kódu a ...
- ... využít je pro optimalizaci programů.
- ... využít je pro (alespoň částečnou) **verifikaci programů**.

Nerozhodnutelnost

- Všechny zajímavé vlastnosti programů zapsaných v obecném programovacím jazyce jsou nerozhodnutelné.
- Henry Gordon Rice (1953) – Riceovy věty.
- Alan Turing (1936) – Problém zastavení.

Nerozhodnutelné – Jak dál?

Abstrakce

- Zakrytí detailů za účelem zjednodušení analýzy.
- Snaha o korektní, byť neúplná řešení.

Využití abstrakce

- Zjednodušené modelovací jazyky pro popis programu. (Typicky vedou na konečně velký stavový prostor.)
- Vykonávání kódu při uvažované abstrakci – **abstraktní interpretace**.

Jiné cesty – připomenutí jiných přístupů

- Fixovaná, či jinak omezená množina vstupů (testování).
- Omezení zkoumaného prostoru (bounded MC).
- Praktická nerozhodnutelnost (řešiče SMT).

Datové a predikátové abstrakce

Datové abstrakce

Motivace

- Stavová exploze způsobená velkými datovými doménami.
- Redukce myšlenkou doménového testování, tj. nahrazení konkrétní velké datové domény abstrahovanou datovou doménou s menším počtem prvků.

Terminologie

- Abstrakce: mapování konkrétních stavů na abstraktní.
- Konkretizace: mapování abstraktních stavů na množiny konkrétních stavů.

Příklad datové abstrakce

- `Int -> { Even, Odd }`
- Konkrétní stav: ⟨ `PC:12, A:15, B:0` ⟩
- Abstraktní stav: ⟨ `PC:12, A:Odd, B:Even` ⟩

Abstraktní přechodový systém

Přechody v konkrétní a abstraktní sémantice

- Příkaz programu na řádku 12: `A := A+A`
- V konkrétní sémantice:
  $\langle$`PC:12, A:15, B:0`$\rangle \longrightarrow \langle$`PC:13, A:30, B:0`$\rangle$
- V abstraktní sémantice:
  $\langle$`PC:12, A:Odd, B:Even`$\rangle \longrightarrow \langle$`PC:13, A:Even, B:Even`$\rangle$

Nedeterminismus v abstraktním přechodovém systému

- Abstraktní stav: $\langle$`PC:13, A:Even, B:Even`$\rangle$
- Příkaz programu na řádku 13: `A := A div 2`
- $\langle$`PC:13, A:Even, B:Even`$\rangle \longrightarrow$

  $\langle$`PC:14, A:Even, B:Even`$\rangle$

  $\langle$`PC:14, A:Odd, B:Even`$\rangle$

Vztah abstraktního a konkrétního přechodového systému

Nad-aproximace (Over-Approximation)

- Každý běh konkrétního systému je obsažen v konkretizaci nějakého abstraktního běhu.
- Mohou existovat běhy, jež jsou obsaženy v konkretizaci nějakého abstraktního běhu, ale nejsou v původním konkrétním přechodovému systému.

Pod-aproximace (Under-Approximation)

- Každá běh obsažený v konkretizaci libovolného abstraktního běhu je během původního konkrétního přechodového systému.
- Mohou existovat běhy konkrétního přechodového systému, které nejsou obsaženy v konkretizaci žádného abstraktního běhu.

Verifikace aproximovaných přechodových systémů

Značení

- APS – abstraktní přechodový systém
- KPS – konkrétní přechodový systém

Verifikace nad-aproximace

- Absence chyby v APS dokazuje absenci chyby v KPS.
- Chyba v APS může a nemusí být chyba v KPS.
- Chyba v APS, která není chybou v KPS, se označuje jako "spurious error" (false positive, false alarm).

Verifikace pod-aproximace

- Chyba v APS dokazuje přítomnost chyby v KPS.
- Absence chyby v APS nedokazuje absenci chyby v KPS.
- Chyba v KPS, která není zachycena v APS, se označuje jako (false negative).

Příklad – konkrétní sémantika

Příklad

- Je dosažitelný chybový stav v následujícím programu?
- % označuje operaci modulo, A je celočíselná proměnná

Kód programu	Hodnota A v konkrétní sémantice po provedení příkazu vlevo	
1 `read(A);`	`[int]`	
2 `A = A % 2;`	`[0]`	`[1]`
3 `A = A + 1;`	`[1]`	`[2]`
4 `if (A==0)`	`<false>`	`<false>`
5 `error;`		
6 `else`		
7 `return;`	`<ret>`	`<ret>`

Příklad – datová abstrakce

Příklad

- Je dosažitelný chybový stav v následujícím programu?
- `A` je abstrahováno do domény `{even,odd}`

Kód programu		Hodnota `A` v abstrahované sémantice po provedení příkazu vlevo	
1	`read(A);`	`[even]`	`[odd]`
2	`A = A % 2;`	`[even]`	`[odd]`
3	`A = A + 1;`	`[odd]`	`[even]`
4	`if (A==0)`	`<false>`	`<true/false>`
5	**`error;`**		**`<error>`**
6	`else`		
7	`return;`	`<ret>`	`<ret>`

Predikátová abstrakce

Predikátová abstrakce

- Predikáty – podmínkové výrazy o valuaci proměnných.
- Jiný způsob jak definovat abstraktní přechodový systém.
- Jedna možná konkrétní definice abstrakce:
  $\langle$čítač instrukcí + valuace zvolených predikátů$\rangle$

Míra abstrakce

- Množství predikátů ovlivňuje přesnost abstrakce.
- Málo predikátů $\rightsquigarrow$ velká nepřesnost, malá stavová exploze
- Více predikátů $\rightsquigarrow$ malá nepřesnost, velká stavová exploze

Příklad

Zadání

- Pro níže uvedený program a uvedené množiny predikátů nakreslete abstraktní přechodový systém, který vznikne použitím metody predikátové abstrakce.
- Ve vámi navrženém přechodovém systému rozhodněte, zda je některá z cest vedoucí do chybového programu realizovatelná.

```
1   read(A);
2   A = A % 2;
3   A = A + 1;
4   if (A==0)
5      error;
6   else
7      return;
```

a) $P1 \equiv A = 0$

b) $P1 \equiv A = 0$,
$P2 \equiv A \geq 0$

Poznámky k predikátové abstrakci

Analýza abstraktních cest vedoucích k chybě

- Rozhodnutí o realizovatelnosti (jedná se o falešný alarm?)
- Odvození nových predikátů, které zpřesní abstrakci.

Problém velikosti abstraktního přechodového systému

- S počtem predikátů roste exponenciálně velikost abstraktního přechodového systému.

Možné řešení

- Predikáty svázané s konkrétní lokací v programu.

Metoda CEGAR

Counter-Example Guided Abstraction Refinement

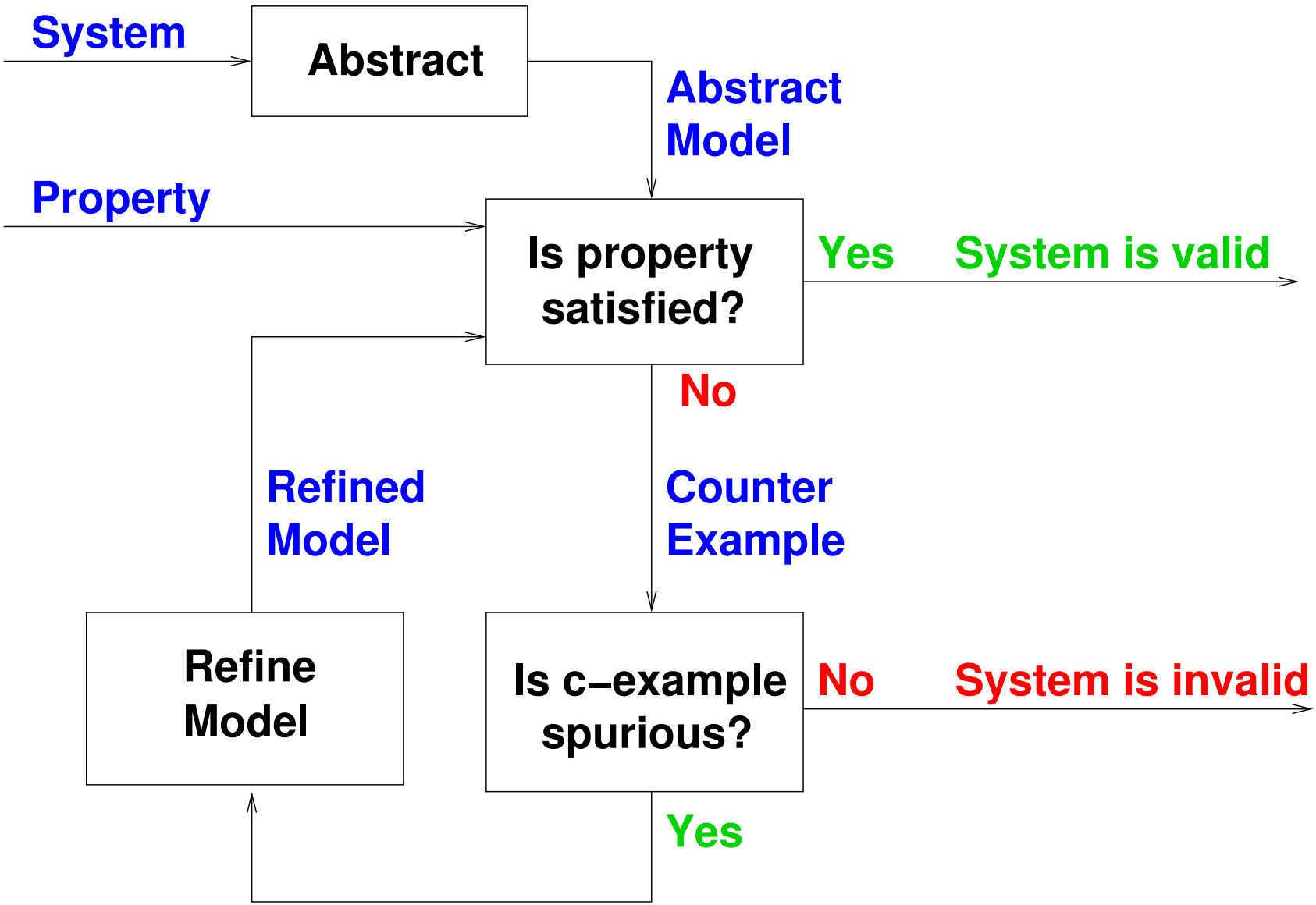
Princip metody CEGAR

- Systém je abstrahován metodou predikátové abstrakce pro iniciální množinu predikátů.
- Abstraktní přechodový systém (nad-aproximace) verifikován metodou ověřování modelu.
- V případě nalezení "spurious" protipříkladu je tento použit k odvození nových predikátů a zpřesnění abstrakce.
- Po zpřesnění abstrakce se postup opakuje.

Poznámky

- Odvozování nových vhodných predikátů je velmi složité.
- Odvozování predikátů v době běhu verifikace (on-the-fly).
- Berkeley Lazy Abstraction Software Verification Tool (BLAST).

Schéma metody CEGAR



Základy abstraktní interpretace

Programová analýza abstraktní interpretací

Reprezentace programu – Flow Graph

- "Speciální verze" grafu toku řízení (Control-Flow Graph).
- Každá hrana má buď právě jednu stráž (podmínku), nebo právě jeden efekt (přiřazení).

Cíl

- Vypočítat vlastnosti jednotlivých vrcholů flow-grafu.

Příklady cílů

- V jakém rozsahu hodnot se v daném místě programu může vyskytnout vybraná proměnná.
- Které proměnné jsou v daném místě programu živé.
- ...

Obecný postup výpočtu

Požadavky

- Doména možných řešení pro jednotlivé vrcholy grafu.
- Iniciální řešení asociované s každým vrcholem grafu.
- Definice toho, jakým způsobem ovlivňuje (aktualizuje) hrana mezi dvěma vrcholy řešení asociované k dotčeným vrcholům.
- Aktualizace řešení způsobené (opakovaným) zpracováním hrany flow grafu je monotónní funkce.

Výpočet

- Vyberu nezpracovanou hranu a aktualizuji řešení u přidružených vrcholů, pokud se řešení změnilo, označím hranu znovu jako nezpracovanou.
- Opakuji, dokud existují nezpracované hrany.

Příklad – výpočet živých proměnných

Konfigurace výpočtu

- Doména = potenční množina množiny všech proměnných.
- Iniciální hodnota asociovaná s vrcholy je $\emptyset$.
- Pro hranu z vrcholu u do vrcholu v je definován update řešení pro vrchol u takto:

$$V(u) = V(u) \cup \Big(V(v) \setminus \mathit{assigned}(u, v) \cup \mathit{used}(u, v) \Big),$$

kde $V(x)$ označuje množinu řešení asociovaných s vrcholem x, $\mathit{assigned}(u, v)$ a $\mathit{used}(u, v)$ označují proměnné předefinované a použité hranou z u do v.

Pozorování

- V každém bodě výpočtu mám nějaké (aproximující) řešení.
- Dosažení pevného bodu není garance nejlepšího řešení.

Abstraktní interpretace

Pozorování

- Výše uvedený postup je ve své podstatě velmi obecný, vhodnou volbou abstrakce a dalších parametrů lze ověřovat mnoho různých věcí.
- Označuje se jako **abstraktní interpretace**.

Co lze parametrizovat

- Abstraktní doménu řešení.
- Směr aktualizace (po směru, proti směru, obojí).
- Definice update funkcí.
- Jak kombinovat hodnoty v místě spojení cest flow-grafu.
- Pořadí vyhodnocování nezpracovaných hran.
- Detekce časného ukončení.

Relevantní otázky abstraktní interpretace

Existuje pevný-bod?

- Struktura možných řešení asociovaných s jednotlivými vrcholy tvoří úplný svaz.
- Knaster-Tarskiho teorém říká, že na takové doméně má každá monotónní funkce pevný bod.

Terminuje výpočet?

- Pokud neexistuje nekonečná rostoucí posloupnost možných řešení, pak ano.
- V opačném případě nemusí.

Rozšíření (Widening)

- Pomocná transformace vypočítaných mezi-řešení tak, aby zachovalo korektnost, ale zabránilo existenci nekonečně rostoucí posloupnosti, respektive zkrátilo její délku.

Příklad

- Určení číselného intervalu, ve kterém budou hodnoty zpracovány přesně, mimo tento interval budou reprezentovány hodnotou $+\infty$ nebo $-\infty$.
- Posloupnost

  $$[0,1] \subset [0,2] \subset [0,3] \subset [0,4] \subset [0,5] \subset [0,6] \ldots$$

  se pro interval přesnosti [0,3] změní na:

  $$[0,1] \subset [0,2] \subset [0,3] \subset [0,+\infty]$$

Pomocné techniky – Zúžení

Zúžení (Narrowing)

- Použití rozšíření vede na velmi nepřesné výsledky.
- Může být použit pouze dočasně k terminaci analýzy cyklů.
- Po analýze cyklu možno provést analýzu cyklu znovu a přesně, avšak s iniciální hodnotou získanou po dokončení analýzy cyklu s rozšířením.

Příklad

- Hodnota $[0,+\infty]$ se provedení zúžení dostane na reálnou hodnotu $[0,n]$.

Komentář

- Přesné a další použití technik rozšíření a zúžení je nad rámec tohoto kurzu.

Finální poznámky k abstraktní interpretaci

Další oblasti programové analýzy

- Mezi-procedurální analýza.
- Analýza paralelních programů.
- Generování invariantů.
- Analýza ukazatelů a dynamických datových struktur.
- ...

CPA checker

- The Configurable Software-Verification Platform
- `http://cpachecker.sosy-lab.org/`
- Vítěz mnoha kategoriích v Software Verification Competition.

Domácí úkol

Připomenutí

- Pro možnost udělení hodnocení stupněm A, je nutné vypracovat všechny domácí úlohy.

Zadání domácí úlohy

- Identifikujte ve slajdech místo, které by stálo za to doplnit nějakým vysvětlujícím obrázkem, tento obrázek vytvořte a nejpozději v den konání zkoušky pošlete emailem vyučujícímu spolu s identifikací místa umístění.
- Přednášející si vyhrazuje právo, odmítnout nevhodně, či "lacině" připravený obrázek.

Sekce

A to je konec ...

Shrnutí IV113 a reklama

IV113 – Přehled verifikačních přístupů

- Black-box testing.
- White-box testing a symbolická exekuce.
- Principy deduktivní verifikace.
- Model checking LTL a CTL.
- Bounded model checking.
- Úvod do programové analýzy.

IA159 – Formal Verification Methods

- Detaily vybraných verifikačních metod.
- Programová analýza.
- Verifikace nekonečně stavových systémů.

IV101 – Seminář z verifikace

- Použití verifikačních nástrojů.

Závěrem

Zkoušky

- Bez pomocných materiálů na veškerý odpřednášený obsah.
- Nutno se přihlásit skrze IS.

Výzva

- Prosím o zpětnou vazbu skrze studentskou anketu.
- Uvítám náměty a obrázky na doplnění slajdů.

Děkuji za pozornost!