IV113 Validace a verifikace

Formální verifikace algoritmů

Jiří Barnat

Verifikace algoritmů

Validace a Verifikace

- Jeden z obecných cílů V&V je prokázat správné chování algoritmů.

Připomenutí

- Proces testování je neúplný.
- Testováním dokážeme odhalit chybu, ale nedokážeme prokázat bezchybnost.

Závěr

- Je zapotřebí jiného způsobu verifikace systémů.

Formální verifikace

Cíl formální verifikace

- Cílem je prokázat, že systém pracuje správně takovým způsobem, aby míra důvěry ve výsledek procesu verifikace byla stejná, jako míra důvěry v matematický důkaz.

Nutné podmínky pro realizaci

- Formálně přesně definovaná sémantika chování systému.
- Formálně přesně stanovené požadavky na systém.

Formální metody verifikace

- Deduktivní verifikace
- Ověřování modelu (Model Checking)
- Abstraktní interpretace

Deduktivní verifikace

Korektnost programu

Program je korektní pokud

- Pokud pro platný vstup **skončí** a vrátí **korektní** výsledek.
- Dokazují se dvě tvrzení: že je program parciálně korektní, a že výpočet programu vždy skončí.

Parciální korektnost (Korektnost, Soundness)

- Pokud výpočet programu nad vstupními hodnotami, pro které je program definován, skončí, je výsledek výpočtu správný.

Terminace (Úplnost, Konvergence, Completness)

- Pro vstupní hodnoty, pro které je program definován, výpočet programu skončí.

Verifikace sekvenčních programů

Sekvenční programy

- Vstup-výstupně uzavřené konečné programy.
  - Hodnoty vstupu jsou známy před vykonáním programu.
  - Výstup po skončení programu uložen ve výstupní proměnné.
- Quick sort, největší společný dělitel, ...

Princip verifikace

- Na programy a jednotlivé instrukce se nahlíží jako na transformátory stavů.
- Cílem je prokázat, že vstupní a výstupní hodnoty se k sobě mají v odpovídajícím vztahu.
- Tj. verifikovat korektnost postupu aplikovaného za účelem transformace vstupních proměnných na odpovídající výstupní proměnné.

Vyjadřování vlastností programů

Stav výpočtu

- Stav výpočtu programu je jednoznačně určen lokací (hodnotou čítače instrukcí) a valuací proměnných.

Atomické predikáty

- Základní tvrzení o jednotlivých stavech, jejichž pravdivost lze určit na základě konkrétní valuace proměnných.
- Příklady atomických propozic: $(x == 0)$, $(x1 >= y3)$.
- Pozor na rozsah platnosti odkazovaných proměnných!

Popis množiny stavů

- Specifikovány boolovskou kombinací atomických predikátů
- Příklad: $(x == m) \wedge (y > 0)$

Vyjadřování vlastností programů – Assertions

Assertion

- Pro dané místo programu (lokaci) definuje podmínku na valuace proměnných, jež jsou v dané lokaci programu návrhářem algoritmu očekávány.
- Invariant lokace programu.

Assertions – důkazy korektnosti

- Přiřazení vlastností jednotlivým místům grafu toku řízení.
- Robert Floyd: Assigning Meanings to Programs (1967)

Detekce chyb — Assertion Violation

Testování

- Porušení assertu jako orákulum.

Run-Time kontrola

- Ověřování invariantů lokací v době běhu programu.
- Případná chyba ve výpočtu, která vede k assertion violation, je snáze lokalizována díky vazbě invariantu na konkrétní lokaci v programu.

Neodhalené chyby

- Nevhodně zvolená vstupní data.
- Nedeterministické vykonávání programu (paralelismus).

Hoarův dokazovací systém

Hoarův dokazovací systém

Princip

- Programy = transformátory stavů.
- Specifikace = vztah mezi vstupním a výstupním stavem.

Hoarova logika

- Navržena za účelem ukazování parciální korektnosti programů.
- Nechť P a Q jsou predikáty a S program, pak

$$\{\mathrm{P}\}\ \mathrm{S}\ \{\mathrm{Q}\}$$

je tzv. *Hoarova trojice*.

Zamýšlený význam trojice $\{P\}\ S\ \{Q\}$

- S je program, který transformuje stav splňující *vstupní podmínku* P na stav, který splňuje *výstupní podmínku* Q.

Vstupní a výstupní podmínky

Příklad

- $\{z = 5\}\ x = z * 2\ \{x > 0\}$
- Platná trojice, výstupní podmínka by mohla být přesnější.
- Silnější výstupní podmínka: $\{x > 5 \wedge x < 20\}$, zjevně platí, že $\{x > 5 \wedge x < 20\} \implies \{x > 0\}$.

Nejslabší vstupní podmínka (weakest precondition)

- P je nejslabší vstupní podmínka, pokud
- platí $\{P\}S\{Q\}$ a zároveň
- $\forall P'$, pro které platí $\{P'\}S\{Q\}$, platí $P' \implies P$.

Dokazování v Hoarově systému

Postup dokazování {P} S {Q}

- Zvolíme vhodné podmínky P' a Q'
- Dokazujeme {P'} S {Q'}, P $\Longrightarrow$ P' a Q' $\Longrightarrow$ Q.
- V Hoarově důkazovém systému jsou axiomy a odvozovací pravidla, ze kterých se vytvářejí platné trojice pro strukturálně složitější programy.
- Pokud se podaří vyskládat transformaci P' na Q', tak {P'} S {Q'} je platná trojice.
- P $\Longrightarrow$ P' a Q' $\Longrightarrow$ Q **se dokazují běžným způsobem.**

Hoarův dokazovací systém – Axiom pro přiřazení

Axiom

- Axiom pro přiřazení: $\{\phi[\text{x nahrazeno k}]\}\ x := k\ \{\phi\}$

Význam

- Trojice $\{P\}x := y\{Q\}$ je axiomem v Hoarově dokazovacím systému, pokud platí, že P je shodné s Q, ve kterém byly všechny výskyty x nahrazeny výrazem y.

Příklad

- $\{y+7>42\}\ x:=y+7\ \{x>42\}$ — je axiom
- $\{r=2\}\ r:=r+1\ \{r=3\}$ — není axiom
- $\{r+1=3\}\ r:=r+1\ \{r=3\}$ — je axiom

Hoarova logika – Příklad 1

Příklad

- Dokažte, že následující program vrátí hodnotu větší než 0, pokud je spuštěn pro hodnotu 5.
- Program: $out := in * 2$

Důkaz

1) Formulujeme Hoarovu trojici:
   $\{in = 5\}\ out := in * 2\ \{out > 0\}$

2) Odvodíme vhodnou vstupní podmínku programu:
   $\{in * 2 > 0\}$

3) Dokážeme Hoarovu trojici:
   $\{in * 2 > 0\}\ out := in * 2\ \{out > 0\}$ (axiom)

4) Dokážeme pomocné tvrzení:
   $(in = 5) \implies (in * 2 > 0)$

Hoarův dokazovací systém – Příklad pravidla

Pravidlo

- Sekvenční kompozice: $\frac{\{\phi\}S_1\{\chi\} \wedge \{\chi\}S_2\{\psi\}}{\{\phi\}S_1;S_2\{\psi\}}$

Význam

- Jestliže S_1 transformuje stav splňující ϕ na stav splňující χ a S_2 transformuje stav splňující χ na stav splňující ψ, pak sekvence $S_1; S_2$ transformuje stav splňující ϕ na stav splňující ψ.

Dokazování

- Pro účely důkazu $\{\phi\}S_1; S_2\{\psi\}$ je nutné nalézt χ a ukázat $\{\phi\}S_1\{\chi\}$ a $\{\chi\}S_2\{\psi\}$.

Hoarův dokazovací systém – částečná korektnost

Axiom pro **skip**: $\{\phi\}$ **skip** $\{\phi\}$

Axiom pro **:=**: $\{\phi[x := k]\}x\mathbf{:=}k\{\phi\}$

Sekvenční kompozice: $\dfrac{\{\phi\}S_1\{\chi\}\wedge\{\chi\}S_2\{\psi\}}{\{\phi\}S_1;S_2\{\psi\}}$

Alternativa: $\dfrac{\{\phi\wedge B\}S_1\{\psi\}\wedge\{\phi\wedge\neg B\}S_2\{\psi\}}{\{\phi\}\textbf{if } B \textbf{ then } S_1 \textbf{ else } S_2 \textbf{ fi}\{\psi\}}$

Iterace: $\dfrac{\{\phi\wedge B\}S\{\phi\}}{\{\phi\}\textbf{while } B \textbf{ do } S \textbf{ od } \{\phi\wedge\neg B\}}$

Důsledek: $\dfrac{\phi \Longrightarrow \phi',\{\phi'\}S\{\psi'\},\psi' \Longrightarrow \psi}{\{\phi\}S\{\psi\}}$

Hoarova logika – Příklad 2

Dokažte, že pro $n \geq 0$ počítá $n!$.

- ```
  r = 1;

  while (n≠ 0) {
     r = r * n;

     n = n - 1;
  }
  ```

Poznámky k důkazu:
  ```

Hoarova logika – Příklad 2

Dokažte, že pro $n \geq 0$ počítá $n!$.

- $\{ \mathsf{n} \geq 0 \wedge \mathsf{t}{=}\mathsf{n} \}$ $\{P\}$

```
r = 1;

while (n≠ 0) {
   r = r * n;

   n = n - 1;
}
```

$\{ \mathsf{r}{=}\mathsf{t}! \}$ $\{Q\}$

Poznámky k důkazu:

- Formulace dokazovaného jako Hoareho trojice.
- Všimněme si použití pomocné proměnné t.

Hoarova logika – Příklad 2

Dokažte, že pro $n \geq 0$ počítá $n!$.

- $\{ \mathsf{n} \geq 0 \wedge \mathsf{t}{=}\mathsf{n} \}$ $\{P\}$
  r = 1;
  $\{ \mathsf{n} \geq 0 \wedge \mathsf{t}{=}\mathsf{n} \wedge r = 1 \}$ $\{I_1\}$
  while (n≠ 0) {
  r = r * n;

  n = n - 1;
  }
  $\{ \mathsf{r}{=}\mathsf{t}! \}$ $\{Q\}$

Poznámky k důkazu:

- $\{\mathsf{n} \geq 0 \wedge \mathsf{t}{=}\mathsf{n} \wedge 1{=}1\}$ r=1 $\{ \mathsf{n} \geq 0 \wedge \mathsf{t}{=}\mathsf{n} \wedge \mathsf{r}{=}1 \}$
- $(\mathsf{n} \geq 0 \wedge \mathsf{t}{=}\mathsf{n}) \implies (\mathsf{n} \geq 0 \wedge \mathsf{t}{=}\mathsf{n} \wedge 1{=}1)$

Hoarova logika – Příklad 2

Dokažte, že pro $n \geq 0$ počítá $n!$.

- $\{ \mathsf{n} \geq 0 \wedge \mathsf{t}{=}\mathsf{n} \}$ $\{P\}$
  r = 1;
  $\{ \mathsf{n} \geq 0 \wedge \mathsf{t}{=}\mathsf{n} \wedge r = 1 \}$ $\{I_1\}$
  while (n≠ 0) $\{ \mathsf{r}{=}\mathsf{t}!/\mathsf{n}! \wedge \mathsf{t} \geq \mathsf{n} \geq 0 \}$ { $\{I_2\}$
    r = r * n;

    n = n - 1;
  }
  $\{ \mathsf{r}{=}\mathsf{t}! \}$ $\{Q\}$

Poznámky k důkazu:

- Invariant cyklu: $\{I_2\} \equiv \{ \mathsf{r}{=}\mathsf{t}!/\mathsf{n}! \wedge \mathsf{t} \geq \mathsf{n} \geq 0 \}$
- $I_1 \implies I_2$ $\quad$ $(I_2 \wedge \neg(\mathsf{n}{\neq}0)) \implies Q$

Hoarova logika – Příklad 2

Dokažte, že pro $n \geq 0$ počítá $n!$.

- $\{ \; n \geq 0 \wedge t{=}n \; \}$ $\{P\}$
  r = 1;
  $\{ \; n \geq 0 \wedge t{=}n \wedge r = 1 \; \}$ $\{I_1\}$
  while (n$\neq$ 0) $\{ \; r{=}t!/n! \wedge t \geq n \geq 0 \; \}$ { $\{I_2\}$
    r = r * n;
    $\{ \; r{=}t!/(n\text{-}1)! \wedge t \geq n > 0 \; \}$ $\{I_3\}$
    n = n - 1;
  }
  $\{ \; r{=}t! \; \}$ $\{Q\}$

Poznámky k důkazu:

- $\{ \; r{*}n = t!/(n\text{-}1)! \wedge t \geq n > 0 \; \}$ r=r*n $\{I_3\}$
- $I_2 \wedge (n{\neq}0) \implies (\; r{*}n = t!/(n\text{-}1)! \wedge t \geq n > 0 \;)$

Hoarova logika – Příklad 2

Dokažte, že pro $n \geq 0$ počítá $n!$.

- $\{ \; n \geq 0 \wedge t{=}n \; \}$ $\{P\}$
  r = 1;
  $\{ \; n \geq 0 \wedge t{=}n \wedge r = 1 \; \}$ $\{I_1\}$
  while (n≠ 0) $\{ \; r{=}t!/n! \wedge t \geq n \geq 0 \; \}$ { $\{I_2\}$
    r = r * n;
    $\{ \; r{=}t!/(n\text{-}1)! \wedge t \geq n > 0 \; \}$ $\{I_3\}$
    n = n - 1;
  }
  $\{ \; r{=}t! \; \}$ $\{Q\}$

Poznámky k důkazu:

- $\{ \; r = t!/(n\text{-}1)! \wedge t \geq (n\text{-}1) \geq 0 \; \}$ n=n-1 $\{I_2\}$
- $I_3 \implies (\; r = t!/(n\text{-}1)! \wedge t \geq (n\text{-}1) \geq 0 \;)$

Hoareho logika a úplnost

Pozorování

- Díky Hoareho logice jsme převedli důkaz korektnosti programu na sadu matematických tvrzení, typicky využívající Peanovu aritmetiku.

Poznámka o korektnosti a (ne)úplnosti

- Hoarova logika je korektní, tj. pokud je možné dokázat $\{P\}S\{Q\}$, pak vykonání programu S ze stavu splňujícím P může skončit pouze ve stavu splňujícím Q.
- Je-li důkazový systém dostatečně silný na popis aritmetiky celých čísel, je nutně neúplný, tj. existují tvrzení, které nelze v systému dokázat a nelze dokázat ani jejich negaci.

Hoareho logika a obecně dokazování v praxi

Potíže s tvorbou důkazů

- Pro potřeby důkazu je často nutné vhodně zesílit vstupní podmínku nebo oslabit výstupní podmínku.
- Je velmi obtížné hledat invarianty cyklů.

Důkaz v praxi – částečná korektnost

- Často se zjednodušuje na konstatování invariantů a prokázání, že se skutečně jedná o invarianty cyklu (typicky indukcí).

Více o Hoarově logice na FI

- IV022 Návrh a verifikace algoritmů
- IA159 Formal Verification Methods

Prokazování terminace

Dobře založená doména

- Částečně uspořádaná množina prvků, ve které neexistuje nekonečně dlouhá klesající posloupnost.
- Příklady: $(\mathbf{N},<)$, $(\mathit{FinPowerSet}(\mathbf{N}),\subseteq)$

Prokázání terminace

- Každému vrcholu grafu toku řízení je přiřazen výraz, jež se vyhodnocuje nad společnou vhodně zvolenou dobře založenou doménou.
- Prokáže se, že hodnota výrazu asociovaného s vrcholem grafu toku řízení neroste podél žádné hrany grafu.
- Prokáže se, že hodnota výrazu asociovaného s vrcholem grafu toku řízení striktně klesá podél alespoň jedné hrany každého cyklu v grafu.

Automatizace deduktivní verifikace

Princip automatizace deduktivní verifikace

Předzpracování

- Přepis programu (transformace) do vhodného mezi-jazyka.
- Příklady jazyků: Boogie (Microsoft Research), Why3 (INRIA)

Strukturální analýza a konstrukce kostry důkazu

- Nalezení trojic Hoareho logiky a invariantů cyklů.
- Vstupní a výstupní podmínky dodány spolu s verifikovaným programem.
- Generování znění pomocných lemat (proof obligations).

Řešení pomocných lemat

- Využití nástroju pro automatické dokazování k prokázání pomocných lemat.

Řešení pomocných lemat

Nástroje pro automatizované dokazování

- Uživatel vede nástroj k sestrojení důkazu požadovaného.
- HOL, ACL2, Isabelle, PVS, Coq, ...

Redukce na problém splnitelnosti negace

- Využití řešičů splnitelnosti.
- Z3, ...

Automatizované dokazování

Důkaz

- Konečná posloupnost transformací předpokladů ψ na požadovaný závěr φ s využitím axiomů daného dokazovacího systému a již dokázaných tvrzení.

Pozorování

- Pro systémy s konečným počtem axiomů a odvozovacích pravidel lze důkazy dané délky systematicky generovat, tj. pro platná tvrzení lze v konečném čase nalézt důkaz.
- Všechny rozumné dokazovací systémy mají nekonečně mnoho axiomů. Uvažme např. axiom $x = x$, ve skutečnosti je to pouze zkratka za axiomy 1=1, 2=2, 3=3, ...).
- Důkazy lze generovat, pokud všechny axiomy a pravidla lze alespoň enumerovat.

Automatizované dokazování

Hledání důkazu platného tvrzení

- Potenciálních konečných posloupností k ověření je příliš (nekonečně) mnoho.
- Obecně pro nalezení důkazu daného tvrzení v daném dokazovacím systému neexistuje algoritmus, je např. nerozhodnutelné, zda program zastaví pro každý vstup.
- Bez rozumné strategie, nelze očekávat nalezení důkazu platného tvrzení v rozumně krátké době.
- Strategie hledání důkazu je udávána uživatelem s odpovídající kvalifikací a matematickým cítěním.

Verifikace s nástrojem pro dokazování

Nástroje pro podporu dokazování (Theorem Provery)

- Cílem je pro danou množinu axiomů a důkazový systém nalézt důkaz daného tvrzení.
- Důkaz se hledá střídavě dvěma způsoby:
  - Algoritmický mód – aplikace odvozených pravidel a axiomů
    - Řízen uživatelem nástroje
    - Dedukce, resoluce, unifikace, přepisování, ...
  - Hledací mód – hledá nová platná tvrzení
    - Využití hrubé výpočetní síly.

Existující nástroje

- Popis dokazovacího systému, programu i požadovaného tvrzení probíhá ve vstupním jazyce dokazovacího nástroje.

Možné výsledky procesu dokazování

Výstup procesu dokazování

a) Důkaz nalezen a ověřen.

b) Důkaz nenalezen.
   - Tvrzení platí, lze dokázat, ale důkaz se nám nepodařilo nalézt.
   - Tvrzení platí, ale nelze jej v daném systému dokázat.
   - Tvrzení neplatí.

Pozorování

- V případě nenalezení důkazu metoda nedává žádnou nápovědu k tomu, proč se tvrzení nepodařilo dokázat.

Dafny

http://rise4fun.com/dafny