Formální Metody a Specifikace (LS 2011)
Přednáška 9:
Praktické ověření neomezené správnosti programů

Stefan Ratschan

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

15. duben 2011

Neomezená správnost

Chceme dokázat že

pro každý stav s tak, že $s \models I$,

pro každý stav s' tak, že $s \to^* s'$

$s' \models O$

Takovou formuli O také nazýváme *invariantem* daného programu

Pokud určitá formule V splňuje podmínky induktivity:

- Pro každý stav s, pokud $s \models I$ pak $s \models V$
- Pro každý stav s, s', pokud $s \models V$ a $s \to s'$, pak $s' \models V$

Pak tato formule V je invariantem.

Sice každý induktivní invariant je invariantem,

ale opačný směr bohužel neplatí.

Tudíž pro důkaz že O je invariant, zkusíme

najít induktivní invariant V tak, že

pro každý stav s, pokud $s \models V$, pak $s \models O$.

Obecné systémy

Počítačové programy jsou
jen jeden z mnoha druhů formalizmů se vyvíjejícím se stavem.

Jiné druhy:

- Stav $\mathbb{B}^n$: Modelování digitálních čipů
- Stav $\mathbb{R}^n$: Modelování fyzikálních systémů
- ...

Stejný princip důkazů správnosti induktivními invarianty.

Nalezení induktivního invariantu

1: $r \leftarrow \mathbf{F}$
2: **for** $i \leftarrow 1$
3: **to** 10 **do**
4: **if** $a[i] = 7$ **then** $r \leftarrow \mathbf{T}$
5: **return** r

$$
\begin{aligned}
& pc = 2 \wedge \neg r \\
& pc = 3 \Rightarrow \Big[r \Leftrightarrow \big[\exists k \,.\, 1 \leq k \leq i \wedge a[k] = 7 \big] \Big] \\
& pc = 4 \Rightarrow \Big[r \Leftrightarrow \big[\exists k \,.\, 1 \leq k \leq i - 1 \wedge a[k] = 7 \big] \Big] \\
& pc = 5 \Rightarrow \Big[r \Leftrightarrow \big[\exists k \,.\, 1 \leq k \leq 10 \wedge a[k] = 7 \big] \Big]
\end{aligned}
$$

Požadavek na stav v každém řádku! Můžeme psát přímo do programu

Aserce

1: $r \leftarrow \mathbf{F}$

2: @ $\neg r$

for $i \leftarrow 1$

3: @ $r \Leftrightarrow \left[\exists k \;.\; 1 \leq k \leq i \wedge a[k] = 7\right]$

to 10 **do**

4: @ $r \Leftrightarrow \left[\exists k \;.\; 1 \leq k \leq i - 1 \wedge a[k] = 7\right]$

if $a[i] = 7$ **then** $r \leftarrow \mathbf{T}$

5: @ $r \Leftrightarrow \left[\exists k \;.\; 1 \leq k \leq 10 \wedge a[k] = 7\right]$

return r

Chybějící aserce reprezentuje @ **T**.

Aserce a správnost programů

@ α_i v řádku i reprezentuje invariant

$$pc = i \Rightarrow \alpha_i$$

Výslední celý invariant:

$$\bigwedge_{i \in \{1,\dots,n\}} pc = i \Rightarrow \alpha_i$$

A obráceně.

Tudíž: Aserce nejen zjišťují chyby během exekuce programu, ale navíc

- se můžou používat pro (částečně automaticky) ověřování správnosti,
- a reprezentují podstatu správnosti programu (důležitá dokumentace).

Cíl: Už během programování psát silné (co "nejinduktivnější") aserce!

Aserce a nalezení induktivních invariant

Substituce $\bigwedge_{i\in\{1,\dots,n\}} pc = i \Rightarrow \alpha_i$ do induktivních podmínek:

První podmínka:

$$\forall r \,.\, I \Rightarrow [\bigwedge_{i\in\{1,\dots,n\}} pc = i \Rightarrow \alpha_i]$$

Program obvykle začíná v prvním řádku, tj. I má podobu $pc = 1 \wedge \dots$:

$$\forall r \,.\, I \Rightarrow \alpha_1$$

Platí zejména pro $\alpha_1 \Leftrightarrow I$.

Tudíž: Programy/procedury/funkce obvykle začínáme asercí @I !

Tato aserce zároveň

- ověřuje správnost konkretního vstupů
  přímo po spuštění programu/procedury/funkce, a
- dává veškerou informaci o vstupy pro ověřování obecné správnosti.

Aserce a nalezení induktivních invariant

Podmínka kroku:

$$\forall r, r' \,.\, \Big[\big[\bigwedge_{i \in \{1,\dots,n\}} pc = i \Rightarrow \alpha_i \big] \wedge \Phi_P \Big] \Rightarrow \Big[\bigwedge_{i \in \{1,\dots,n\}} pc' = i \Rightarrow \alpha_i[r \leftarrow r'] \Big]$$

Substituujeme přechodní podmínku Φ_P:

$$pc = i \Rightarrow \Phi_{P,i}$$

Výsledek substituce:

$$\forall r, r' \,.\, \Big[\big[\bigwedge_{i \in \{1,\dots,n\}} pc = i \Rightarrow \alpha_i \big] \wedge \big[\bigwedge_{i \in \{1,\dots,n\}} pc = i \Rightarrow \Phi_{P,i} \big] \Big] \Rightarrow \Big[\bigwedge_{i \in \{1,\dots,n\}} pc' = i \Rightarrow \alpha_i[r \leftarrow r'] \Big]$$

Aserce a nalezení induktivních invariant

$$\forall r, r' \,.\, \Big[\bigwedge_{i \in \{1,\dots,n\}} pc = i \Rightarrow \big[\alpha_i \wedge \Phi_{P,i}\big]\Big] \Rightarrow$$

$$\Big[\bigwedge_{i \in \{1,\dots,n\}} pc' = i \Rightarrow \alpha_i[r \leftarrow r']\Big]$$

$$\forall r, r' \,.\, \Big[\bigvee_{i \in \{1,\dots,n\}} pc = i \wedge \alpha_i \wedge \Phi_{P,i}\Big] \Rightarrow \Big[\bigwedge_{i \in \{1,\dots,n\}} pc' = i \Rightarrow \alpha_i[r \leftarrow r']\Big]$$

$$\forall r, r' \,.\, \bigwedge_{i \in \{1,\dots,n\}} \Bigg[\Big[pc = i \wedge \alpha_i \wedge \Phi_{P,i}\Big] \Rightarrow \Big[\bigwedge_{i \in \{1,\dots,n\}} pc' = i \Rightarrow \alpha_i[r \leftarrow r']\Big]\Bigg]$$

Podívejme se ne jednotlivé i, tj. druhy příkazů:

i: @ α_i: **goto** l

$$\left[pc = i \wedge \alpha_i \wedge pc' = l \wedge \bigwedge_{u \in V} u' = u\right] \Rightarrow \left[\bigwedge_{i \in \{1,\ldots,n\}} pc' = i \Rightarrow \alpha_i[r \leftarrow r']\right]$$

$$\left[pc = i \wedge \alpha_i \wedge pc' = l \wedge \bigwedge_{u \in V} u' = u\right] \Rightarrow \alpha_l[r \leftarrow r']$$

$$\left[pc = i \wedge \alpha_i \wedge pc' = l \wedge \bigwedge_{u \in V} u' = u\right] \Rightarrow \alpha_l$$

$$\alpha_i \Rightarrow \alpha_l$$

To znamená: Pokud program má na řádku i příkaz **goto** l,
pak musíme ověřit tuto podmínku (*verification condition*).

i: @ α_i: $v \leftarrow t$

$$\Big[pc = i \wedge \alpha_i \wedge pc' = pc + 1 \wedge v' = t \wedge \bigwedge_{u \in V, u \neq v} u' = u\Big] \Rightarrow$$

$$\Big[\bigwedge_{i \in \{1,\ldots,n\}} pc' = i \Rightarrow \alpha_i[r \leftarrow r']\Big]$$

$$\Big[pc = i \wedge \alpha_i \wedge pc' = pc + 1 \wedge v' = t \wedge \bigwedge_{u \in V, u \neq v} u' = u\Big] \Rightarrow \alpha_{i+1}[v \leftarrow t]$$

$$\alpha_i \Rightarrow \alpha_{i+1}[v \leftarrow t]$$

i: @ α_i: **if** P **then**

$$\left[pc = i \wedge \alpha_i \wedge [P \Rightarrow pc' = pc + 1] \wedge [\neg P \Rightarrow pc' = l] \wedge \bigwedge_{u \in V} u' = u\right] \Rightarrow$$

$$\left[\bigwedge_{i \in \{1,\dots,n\}} pc' = i \Rightarrow \alpha_i[r \leftarrow r']\right]$$

$$P \Rightarrow \left[\left[pc = i \wedge \alpha_i \wedge pc' = pc + 1 \wedge \bigwedge_{u \in V} u' = u\right] \Rightarrow \left[\bigwedge_{i \in \{1,\dots,n\}} pc' = i \Rightarrow \alpha_i[r\right.\right.$$

$$\wedge$$

$$\neg P \Rightarrow \left[\left[pc = i \wedge \alpha_i \wedge pc' = l \wedge \bigwedge_{u \in V} u' = u\right] \Rightarrow \left[\bigwedge_{i \in \{1,\dots,n\}} pc' = i \Rightarrow \alpha_i[r \leftarrow r']\right.\right.$$

i: @ α_i: **if** P **then**

$$P \Rightarrow \left[\left[pc = i \wedge \alpha_i \wedge pc' = pc + 1 \wedge \bigwedge_{u \in V} u' = u\right] \Rightarrow \alpha_{i+1}\right] \wedge$$

$$\neg P \Rightarrow \left[\left[pc = i \wedge \alpha_i \wedge pc' = l \wedge \bigwedge_{u \in V} u' = u\right] \Rightarrow \alpha_l\right]$$

$$\left[P \Rightarrow [\alpha_i \Rightarrow \alpha_{i+1}]\right] \wedge \left[\neg P \Rightarrow [\alpha_i \Rightarrow \alpha_l]\right]$$

$$\left[[\alpha_i \wedge P] \Rightarrow \alpha_{i+1}\right] \wedge \left[[\alpha_i \wedge \neg P] \Rightarrow \alpha_l\right]$$

i: @ α_i: **input** v

$$\left[pc = i \wedge \alpha_i \wedge pc' = pc + 1 \wedge \bigwedge_{u \in V, u \neq v} u' = u\right] \Rightarrow$$

$$\left[\bigwedge_{i \in \{1,\ldots,n\}} pc' = i \Rightarrow \alpha_i[r \leftarrow r']\right]$$

$$\left[pc = i \wedge \alpha_i \wedge pc' = pc + 1 \wedge \bigwedge_{u \in V, u \neq v} u' = u\right] \Rightarrow \alpha_{i+1}[v \leftarrow v']$$

$$\alpha_i \Rightarrow \alpha_{i+1}[v \leftarrow v']$$

Souhrn

Aserce @ $\alpha_1, \ldots,$ @ α_n dokazují správnost programů
(tj. reprezentují induktivní invariant), pokud:

$$\models \forall r \,.\, I \Rightarrow \alpha_1$$

$$\models \forall r, r' \,.\, \bigwedge_{i \in \{1,\ldots,n\}} \mathrm{VC}_i, \text{ přičemž } \mathrm{VC}_i \equiv$$

- $\alpha_i \Rightarrow \alpha_l$, pokud řádek i má podobu **goto** l
- $\alpha_i \Rightarrow \alpha_{i+1}[v \leftarrow t]$, pokud řádek i má podobu $v \leftarrow t$
- $\Big[[\alpha_i \wedge P] \Rightarrow \alpha_{i+1}]\Big] \wedge \Big[[\alpha_i \wedge \neg P] \Rightarrow \alpha_l]\Big]$ pokud řádek i má podobu **if** P **then**
- $\alpha_i \Rightarrow \alpha_{i+1}[v \leftarrow v']$ pokud řádek i má podobu **input** v

Tradičně: Podobný formalismus "Hoare logic", "Hoare calculus"

Posloupnost přiřazení

@ α
$v_1 \leftarrow t_1; \dots ; v_n \leftarrow t_n$
@ α'

musí existovat $\alpha_1, \dots, \alpha_n$ tak, že

$$\alpha \Rightarrow \alpha_1[v_1 \leftarrow t_1], \dots, \alpha_{n-1} \Rightarrow \alpha_n[v_n \leftarrow t_n], \alpha_n \equiv \alpha'$$

Po volbě

$$\alpha_1 \equiv \alpha_2[v_2 \leftarrow t_2], \dots, \alpha_{n-1} \equiv \alpha_n[v_n \leftarrow t_n]$$

zbývá podmínka

$$\alpha \Rightarrow \alpha'[v_n \leftarrow t_n] \dots [v_1 \leftarrow t_1]$$

while

@ α
while P **do**
    @β
    ...
    @β'
@ α'

- $[\alpha \wedge P] \Rightarrow \beta$
- $[\alpha \wedge \neg P] \Rightarrow \alpha'$
- $[\beta' \wedge P] \Rightarrow \beta$
- $[\beta' \wedge \neg P] \Rightarrow \alpha'$

goto už nepotřebujeme, čísla řádků už nepoužíváme,
aserce jsou samostatný řádky

Celkový induktivný invariant lze rekonstruovat.

for

@ α
for $i \leftarrow l$ **to** u
    @β
    ...
    @β'
@ α'

- $[\alpha \wedge l \leq u] \Rightarrow \beta[i \leftarrow l]$
- $[\alpha \wedge l > u] \Rightarrow \alpha'$
- $[\beta' \wedge i < u] \Rightarrow \beta[i \leftarrow i + 1]$
- $\beta'[i \leftarrow u] \Rightarrow \alpha'$

Pozor: Java/C/C++ for dovoluje víc!

Další kombinace

@ α
if P **then** $v \leftarrow t$
@ α'

Musí existovat β, β' tak, že

$$[\alpha \wedge P] \Rightarrow \beta, \beta \Rightarrow \beta'[v \leftarrow t], \beta' \Rightarrow \alpha', [\alpha \wedge \neg P] \Rightarrow \alpha'$$

Po volbě

$$\beta \equiv \beta'[v \leftarrow t], \beta' \equiv \alpha'$$

zbývá

$$[\alpha \wedge P] \Rightarrow \alpha'[v \leftarrow t], [\alpha \wedge \neg P] \Rightarrow \alpha'$$

Aserce

@ $\mathbf{T}$

$r \leftarrow \mathbf{F}$

@ $\neg r$

for $i \leftarrow 1$ **to** 10 **do**

  @ $r \Leftrightarrow \left[\exists k \,.\, 1 \leq k \leq i-1 \wedge a[k]=7\right]$

  if $a[i]=7$ **then** $r \leftarrow \mathbf{T}$

  @ $r \Leftrightarrow \left[\exists k \,.\, 1 \leq k \leq i \wedge a[k]=7\right]$

@ $r \Leftrightarrow \left[\exists k \,.\, 1 \leq k \leq 10 \wedge a[k]=7\right]$

return r

- $\mathbf{T} \Rightarrow \neg r[r \leftarrow \mathbf{F}]$
- $\neg r \Rightarrow \left[r \Leftrightarrow \left[\exists k \,.\, 1 \leq k \leq i-1 \wedge a[k]=7\right]\right][i \leftarrow 1]$
- $\left[r \Leftrightarrow \left[\exists k \,.\, 1 \leq k \leq i-1 \wedge a[k]=7\right]\right] \wedge a[i]=7 \Rightarrow$ $\left[\exists k \,.\, 1 \leq k \leq i \wedge a[k]=7\right]$
- $\left[r \Leftrightarrow \left[\exists k \,.\, 1 \leq k \leq i-1 \wedge a[k]=7\right]\right] \wedge a[i] \neq 7 \Rightarrow \dots$
- $\dots$

Paralelní vývoj asercí a programů: Dekompozice

Např:

- Vstup: Pole příjmení a_0 a jmen b_0 délky n
- Výstup: $a \times b$ je permutací $a_0 \times b_0$ (píšeme $(a, b) \sim (a_0, b_0)$),
  $\forall i \in \{1, \ldots, n-1\} \, . \, a[i] < a[i+1] \vee [a[i] = a[i+1] \wedge b[i] \leq b[i+1]]$

Nejdřív píšeme aserci na vstup a výstup:

$\ldots$

$$@ \; (a, b) \sim (a_0, b_0) \wedge \forall i \in \{1, \ldots, n-1\} \, . \, a[i] < a[i+1] \vee [a[i] = a[i+1] \wedge b[i] \leq b[i+1]]$$

Potom, rozklad úkolů:

$\ldots$

$$@ \; (a, b) \sim (a_0, b_0) \wedge \forall i \in \{1, \ldots, n-1\} \, . \, a[i] \leq a[i+1]$$

$\ldots$

$$@ \; (a, b) \sim (a_0, b_0) \wedge \forall i \in \{1, \ldots, n-1\} \, . \, a[i] < a[i+1] \vee [a[i] = a[i+1] \wedge b[i] \leq b[i+1]]$$

Paralelní vývoj asercí a programů: Cykly

Potom, cyklus s asercemi:

for $k \leftarrow 1$ **to** $n-1$ **do**

$\quad$ @ $(a,b) \sim (a_0,b_0) \wedge \forall i \in \{1,\dots,k-1\} \,.\, a[i] \leq a[i+1]$

$\quad$ ...

$\quad$ @ $(a,b) \sim (a_0,b_0) \wedge \forall i \in \{1,\dots,k\} \,.\, a[i] \leq a[i+1]$

@ $(a,b) \sim (a_0,b_0) \wedge \forall i \in \{1,\dots,n\} \,.\, a[i] \leq a[i+1]$

...

@ $(a,b) \sim (a_0,b_0) \wedge$

$\quad \forall i \in \{1,\dots,n-1\} \,.\, a[i] < a[i+1] \vee [a[i] = a[i+1] \wedge b[i] \leq b[i+1]]$

atd.

V praxi

- Většina programovacích jazyků nedovoluje psaní asercí v predikátové logiky
- Většinou nemáme podporu pro automatické ověřování vyplývajících podmínek
- Psaní asercí sice může zajišťovat správnost ale může brát hodně času

Musíme dělat kompromisy, např.

- psát aserce částečně jako komentáře
- ověřovat jen část ověřovacích podmínek
- přidat aserce až během testování (tak, aby vyloučily určité důvodu pro danou chybu)

Explicitní podpora pro psaní a ověřovaní asercí roste, např:

Eiffel, Spec$\sharp$, JML, ESC/Java, KeY (http://www.key-project.org), ...

Parciální vs. totální správnost

$x \leftarrow 564$
while T do
@ $x = 564$
return x

Správný program!

Chybí: Ověřování terminace