MI-FME Cvičení 0

Tomáš Chvosta

Únor 2020

Zadání

Pokuste se zprovoznit software RISCAL, který naleznete na stránce, jejíž odkaz je zde: https://www3.risc.jku.at/research/formal/software/RISCAL/. Je zde několik způsobů, jak program zprovoznit (virtuální systém, instalace Java aplikace), můžete zvolit libovolnou z nich. Pokud během instalace narazíte na jakýkoliv problém, podívejte se do diskuzního fóra na Moodlu. Pokud je zde zmíněný váš problém, ale zatím není vyřešen, jednoduše k němu přidejte své jméno. Pokud zde problém není, přidejte popis vašeho problému.

Pokud se vám úspěšně podařilo nainstalovat program, potom se podívejte také do diskuzního fóra a pokuste se ostatním pomoci s jejich problémy. Ti, co pomohou ostatním, mohou být oceněni bonusovými body.

Upozornění: Tento úkol vypracujte co nejdříve! Pokud nezveřejníte svůj problém do 26. února, budu předpokládat, že jste program úspěšně nainstalovali.

MI-FME Cvičení 1

Tomáš Chvosta

Únor 2020

Zadání

Sdělte ostatním zajímavý příběh o nedorozumění, které vyplynulo z nepřesné specifikace programu nebo z komentáře. Nejlepší příběh bude odměněn bonusovými body. Příběhy, které jste prožili osobně, jsou obvykle zajímavější než příběhy zkopírované z internetu.

MI-FME Cvičení 2

Tomáš Chvosta

Únor 2020

Cvičení 2a

Zadání:

Navrhněte input/output specifikaci pro problém nalezení prvočíselného rozkladu pro zadané celé číslo.

Vstup:

Celé číslo $n \in \mathbb{Z}, n \geq 2$

Výstup:

n-tice prvočísel $(p_1, p_2, \ldots, p_k)$ taková, že platí:

$$(\forall i \in \mathbb{Z})(1 \leq i \leq k)(prvocislo(p_i) \wedge \prod_{i=1}^{k} p_i = n)$$

$(\forall p \in \mathbb{Z})(prvocislo(p)) :\Leftrightarrow (p > 1) \wedge ((\forall x \in \mathbb{N})((p \, mod \, x = 0) \Rightarrow (x = 1) \vee (x = p)))$

Poznámky:

Chci prvočíselný rozklad, takže prostě hledám čísla, která budou prvočísla a zároveň, když je všechna vynásobím, tak získám to číslo, které bylo zadáno. Potom musím zadefinovat prvočíslo, což musí být celé číslo větší než 1 a pokud existuje číslo, které ho dělí beze zbytku, pak je to buď jednička nebo to samotné číslo.

Cvičení 2b

Zadání:

Navrhněte input/output specifikaci pro vyhledávání řetězce.

Vstup:

Pole znaků $text$ délky n
Pole znaků $hledany$ délky k

Výstup:

Pravda - v případě, že se hledaný řetězec vyskytuje v zadaném textu.
Nepravda - v případě, že se hledaný řetězec nevyskytuje v zadaném textu.

$$(\exists i \in \mathbb{Z})(0 \leq i \leq n-k)(\forall j \in \mathbb{Z})(0 \leq j < k)(text[i+j] = hledany[j])$$

Poznámky:

V této úloze vlastně hledám index v textu, kde začíná hledaný řetězec. Mělo by se mi to do toho textu vejít, proto je tam $i \leq n-k$. Když ten index mám, tak už jen stačí zkontrolovat jednotlivé znaky, jestli jsou stejné.

Cvičení 2c

Navrhněte input/output specifikaci pro kontrolu, zda se jeden řetězec shoduje s druhým, který obsahuje právě jeden žolíkový znak '*'.

Zadání:

Vstup:

Pole znaků reg délky n
Pole znaků $text$ délky m

Výstup:

Pravda - v případě, že se řetězce shodují.
Nepravda - v případě, že se řetězce neshodují.

$$(\exists i \in \mathbb{Z})(reg[i] = \ '*') \wedge (((\forall j \in \mathbb{Z})(0 \leq j < i)(reg[j] = text[j])) \wedge$$

$$((\forall k \in \mathbb{Z})(i < k < n)(reg[k] = text[m-n+k]))) \ \wedge \ (m \geq n-1)$$

Poznámky:

V řetězci reg musí být někde právě jedna hvězdička. Tu stačí najít a potom zkontrolovat znaky, které jí předchází a které za ní následují, jestli se vyskytují na správných pozicích v řetězci $text$. Zároveň musí platit $m \geq n-1$, například když má řetězec reg délku 5, znamená to, že má 4 znaky, které nepředstavují

hvězdičku a tyto znaky se nutně musí vyskytovat v řetězci $text$. Z toho můžeme vidět, že délka řetězce $text$ musí být alespoň 4, jinak řetezce req a $text$ nemohou být stejné.

Cvičení 2d

Zadání:

Navrhněte input/output specifikaci pro algoritmus, který ze zadaného textu vrátí ta slova, která představují palindromy.

Vstup:

Pole znaků $text$ délky n

Výstup:

Množina slov z textu:

$$\{slovo \mid palindrom\_v\_textu(text, n, slovo, i, l)\}$$

$$(\forall text, n, i, l)(palindrom\_v\_text(text, n, slovo, i, l)) :\Leftrightarrow$$
$$oddeleni\_slova(text, n, i, l) \wedge slovo = substring(text, i, l) \wedge palindrom(slovo, l)$$

$$(\forall text, n, i, l)(oddeleni\_slova(text, n, i, l)) :\Leftrightarrow$$
$$(0 \leq i < n - l) \wedge$$
$$((i = 0) \ \vee \ (text[i - 1] = \ ' \ ')) \wedge$$
$$((i + l = n - 1) \ \vee \ (text[i + l + 1] = \ ' \ ')) \wedge$$
$$(\nexists x \in \mathbb{Z})(i \leq x \leq i + l)(text[x] = \ ' \ ')$$

$(\forall text, i, l)(substring(text, i, l) := s)$ s.t. $(\forall x \in \mathbb{Z})(0 \leq x \leq l-1)(s[x] = text[i+x])$

$(\forall slovo, l)(palindrom(slovo, l)) :\Leftrightarrow (\forall i \in \mathbb{Z})(0 \leq i < l)(slovo[i] = slovo[l-i-1])$

Poznámky:

Ve své podstatě jen hledám slova, která splňují vlastnost palindromu. Nejprve je tedy potřeba zkontrolovat, že před slovem a za slovem je mezera (s výjimkou slova na pozici 0 a slova na úplném konci textu). To přesně dělá *oddeleni_slova*. Dále funkce *substring* vrátí jednotlivá slova. Nakonec stačí pouze zkontrolovat, že platí vlastnost palindromu, tedy že na i-té pozici je stejný znak jako na $(l - i - 1)$-té pozici.

text ... zadaný text
n ... délka zadaného textu
slovo ... slovo v textu
i ... pozice slova v textu
l ... délka slova

MI-FME Cvičení 3

Tomáš Chvosta

Únor 2020

Cvičení 3a

Zadání:

Dokažte následující formuli:

$$[\neg[[r \vee s] \Rightarrow q] \wedge [[r \vee s] \Rightarrow q]] \Rightarrow [[p \Rightarrow q] \wedge \neg[p \Rightarrow q]]$$

Důkaz:

Jelikož se jedná o implikaci, předpokládáme, že platí levá strana pravidla tedy konjunkce $\neg[[r \vee s] \Rightarrow q]$ a $[[r \vee s] \Rightarrow q]]$. Tyto dva předpoklady představují $\perp$ a jelikož z $\perp$ plyne cokoliv, nezáleží na tom, co máme na pravé straně implikace a formule vždy platí.

Table 1: Důkazová tabulka

Krok	**Předpokládáme**	**Dokazujeme**
1.	$\neg[[r \vee s] \Rightarrow q]$ $[[r \vee s] \Rightarrow q]] \ldots \perp$	$[p \Rightarrow q]$ $\neg[p \Rightarrow q]$

Cvičení 3b

Zadání:

Dokažte následující formuli:

$$[\neg p \Rightarrow p] \Rightarrow p$$

Důkaz:

Jelikož se jedná o implikaci, je předpoklad $[\neg p \Rightarrow p]$ a pokusíme se dokázat p. Použijeme důkazní pravidlo z přednášky, které říká, že když chceme dokázat p, pak můžeme nahradit p za $\neg\neg p$ a následně použít pravidlo pro dokazování negací. Do seznamu předpokladů tedy přidáme $\neg p$ a pokusíme se najít spor.

Předpoklad $[\neg p \Rightarrow p]$ říká, že musí platit p jelikož máme v předpokladech $\neg p$, což je spor. Spor podle přednášky dokončí úspěšně jakýkoliv důkaz.

Table 2: Důkazová tabulka

Krok	**Předpokládáme**	**Dokazujeme**
1.	$[\neg p \Rightarrow p]$	p
2.	$\neg p$	$\neg\neg p$ tedy p
3.	$p \ldots \bot$	

Cvičení 3c

Zadání:

Dokažte následující formuli:

$$\neg[p \Rightarrow q] \Rightarrow [q \Rightarrow p]$$

Důkaz:

Jelikož se jedná o implikaci, je $\neg[p \Rightarrow q]$ předpoklad. Pokusíme se tedy dokázat $[q \Rightarrow p]$. Použijeme stejný postup a předpokládáme, že platí q. Nyní by se mohlo hodit dokázat, že platí $[p \Rightarrow q]$. Jako lemma tedy zvolíme $[p \Rightarrow q]$ a díky předpokladu q je jasné, že toto lemma platí. Můžeme tedy přidat předpoklad $[p \Rightarrow q]$, což společně s předpokladem $\neg[p \Rightarrow q]$ vytvoří $\bot$, ze které plyne cokoliv.

Table 3: Důkazová tabulka

Krok	**Předpokládáme**	**Dokazujeme**
1.	$\neg[p \Rightarrow q]$	$[q \Rightarrow p]$
2.	q	p
3.		lemma $[p \Rightarrow q]$
4.	$[p \Rightarrow q] \ldots \bot$	

Cvičení 3d

Zadání:

Dokažte následující formuli:

$$[p \Rightarrow [[q \vee r] \wedge \neg q \wedge \neg r]] \Rightarrow \neg p$$

Důkaz:

Nejprve předpokládejme $[p \Rightarrow [[q \vee r] \wedge \neg q \wedge \neg r]]$ a dokažme $\neg p$. To uděláme tak, že předpokládáme p a zkusíme najít spor. Pokud platí p, tak můžeme usoudit

$[[q \vee r] \wedge \neg q \wedge \neg r]]$. Z předchozího předpokladu můžeme usoudit, že platí $\neg q$, $\neg r$ a také $[q \vee r]$. Pokud víme, že platí disjunkce $[q \vee r]$, pak postupujeme tak, že nejprve předpokládáme q a následně dokončíme důkaz a poté předpokládáme r a následně dokončíme důkaz. V obou dvou případech získáme $\perp$, čímž získáváme spor a důkaz je úspěšně dokončen.

Table 4: Důkazová tabulka

Krok	**Předpokládáme**	**Dokazujeme**
1.	$[p \Rightarrow [[q \vee r] \wedge \neg q \wedge \neg r]]$	$\neg p$
2.	p	hledáme spor
3.	$[[q \vee r] \wedge \neg q \wedge \neg r]]$	hledáme spor
4.	$\neg q$ $\neg r$ $[q \vee r]$	hledáme spor
5a.	$q \ldots \perp$	hledáme spor
5b.	$r \ldots \perp$	hledáme spor

Cvičení 3e

Zadání:

Dokažte následující formuli:

$$q \Rightarrow [[p \wedge q] \vee [\neg p \wedge q]]$$

Důkaz:

Na počátku předpokládáme, že platí q a dokážeme $[[p \wedge q] \vee [\neg p \wedge q]]$. To můžeme udělat například tak, že předpokládáme $\neg[p \wedge q]$ a dokážeme $[\neg p \wedge q]$ (nebo klidně obráceně, avšak toto je výhodnější a snazší). Při důkazu konjunkce je podtřeba dokázat $\neg p$ i q. Při dokazování q využijeme toho, že máme q v předpokladech a tedy je triviálně dokázáno. Při dokazování $\neg p$ předpokládáme, že platí p a najdeme spor. Ten jsme však již vytvořili přidáním p do předpokladů, poněvadž jistě platí $[p \wedge q]$ díky předpokladům p a q a tedy spolu s $\neg[p \wedge q]$ získáme $\perp$, čímž je důkaz úspěšně dokončen.

Table 5: Důkazová tabulka

Krok	**Předpokládáme**	**Dokazujeme**
1.	q	$[[p \wedge q] \vee [\neg p \wedge q]]$
2.	$\neg[p \wedge q]$	$[\neg p \wedge q]$
3.		q
4.		$\neg p$ q
5.	p	hledáme spor
6.		lemma $[p \wedge q]$
7.	$[p \wedge q] \ldots \perp$	hledáme spor

Cvičení 3f

Zadání:

Dokažte následující formuli:

$$\neg[p \wedge q] \Rightarrow [\neg p \vee \neg q]$$

Důkaz:

Nejprve předpokládáme, že platí $\neg[p \wedge q]$ a dokážeme $[\neg p \vee \neg q]$. To dokážeme tak, že předpokládáme, že platí p a dokážeme $\neg q$. To dokážeme tak, že předpokládáme q a najdeme spor. Nyní díky předpokladům p a q můžeme usoudit, že platí lemma $[p \wedge q]$, což však společně s $\neg[p \wedge q]$ tvoří $\perp$. Podobně bychom mohli zvolit v druhém kroku q místo p a dostali bychom stejný výsledek.

Table 6: Důkazová tabulka

Krok	**Předpokládáme**	**Dokazujeme**
1.	$\neg[p \wedge q]$	$[\neg p \vee \neg q]$
2a.	p	$\neg q$
2b.	q	$\neg p$
3a.	q	hledáme spor
3b.	p	hledáme spor
4.		lemma $[p \wedge q]$
5.	$[p \wedge q] \ldots \perp$	hledáme spor

Poznámky:

V tabulce buď zvolíme variantu a) nebo b), ale ne obojí!

Cvičení 3g

Zadání:

Dokažte následující formuli:

$$[[p \wedge q] \Rightarrow r] \Rightarrow [[p \Rightarrow r] \vee [q \Rightarrow r]]$$

Důkaz:

Na začátku tohoto důkazu předpokládejme, že platí $[[p \wedge q] \Rightarrow r]$ a dokažme $[[p \Rightarrow r] \vee [q \Rightarrow r]]$. V tomto kroku máme opět dvě volby jako v předchozí úloze, nyní budeme předpokládat, že platí $\neg[q \Rightarrow r]$ a pokusíme se dokázat $p \Rightarrow r$. To dokážeme tak, že předpokládáme p a dokazujeme r. Nyní se hodí využít faktu, že r je ekvivalentní s $\neg\neg r$. Následně můžeme předpokládat $\neg r$ a poté se pokusit najít spor. Aby platil předpoklad $[p \wedge q] \Rightarrow r$ a zároveň předpoklad $\neg r$, musí platit $\neg[p \wedge q]$. Stejně tak, aby platil předpoklad $\neg[q \Rightarrow r]$ a zároveň $\neg r$, musí platit q. Nyní díky předpokladům p a q můžeme dokázat lemma $[p \wedge q]$, což nám vytvoří krásný spor s předpokladem $\neg[p \wedge q]$ a vznikne $\bot$, což úspěšně dokončí důkaz.

Table 7: Důkazová tabulka

Krok	**Předpokládáme**	**Dokazujeme**
1.	$[p \wedge q] \Rightarrow r$	$[p \Rightarrow r] \vee [q \Rightarrow r]$
2.	$\neg[q \Rightarrow r]$	$p \Rightarrow r$
3.	p	r
4.		$\neg\neg r$
5.	$\neg r$	hledáme spor
6.	$\neg[p \wedge q]$	hledáme spor
7.	q	hledáme spor
8.	$[p \wedge q] \ldots \bot$	hledáme spor

Cvičení 3h

Zadání:

Dokažte následující formuli:

$$[p \wedge q] \Rightarrow \neg[\neg p \vee \neg q]$$

Důkaz:

Na začátku předpokládejme $p \wedge q$ a dokažme $\neg[\neg p \vee \neg q]$. Z předpokladu $p \wedge q$ můžeme usoudit p a q. V dalším kroku předpokládejme, že platí $\neg p \vee \neg q$ a pokusíme se najít spor. Pokud víme, že platí $\neg p \vee \neg q$, můžeme nejprve

předpokládat $\neg p$ a dokončit důkaz a poté předpokládat $\neg q$ a dokončit důkaz. V obou případech nalezneme spor a vznikne $\bot$, čímž je důkaz úspěšně dokončen.

Table 8: Důkazová tabulka

Krok	**Předpokládáme**	**Dokazujeme**
1.	$p \wedge q$	$\neg[\neg p \vee \neg q]$
2.	p q	
3.	$\neg p \vee \neg q$	hledáme spor
4.	$\neg p \ldots \bot$	hledáme spor
5.	$\neg q \ldots \bot$	hledáme spor

Cvičení 3i

Zadání:

Dokažte následující formuli:

$$[p \Rightarrow q] \vee [q \Rightarrow r]$$

Důkaz:

Jelikož máme dokázat disjunkci, můžeme postupovat tak, že předpokládáme, že levá část disjunkce neplatí a dokážeme pravou stranu. V tomto konkrétním případě to znamená předpokládat, že platí $\neg[p \Rightarrow q]$ a dokázat $q \Rightarrow r$. Dále pokračujeme předpokladem, že platí q a máme dokázat r. Díky předpokladu q si můžeme dokázat lemma $[p \Rightarrow q]$, díky platnosti q bude lemma vždy platit bez ohledu na to, jestli platí p. Nyní se v předpokladech ocitla formule $[p \Rightarrow q]$, která nám vytvoří spor s prvním předpokladem. Z toho plyne, že první část počáteční formule $[p \Rightarrow q]$ musí platit, čímž jsme důkaz úspěšně dokončili.

Table 9: Důkazová tabulka

Krok	**Předpokládáme**	**Dokazujeme**
1.	$\neg[p \Rightarrow q]$	$q \Rightarrow r$
2.	q	r
3.		lemma $p \Rightarrow q$
4.	$[p \Rightarrow q] \ldots \bot$	r

MI-FME Cvičení 4

Tomáš Chvosta

Březen 2020

Cvičení 4a

Úloha byla vypracována na cvičení.

Cvičení 4b

Zadání:

Dokažte následující formuli:

$$[(\forall x)(P(x) \wedge Q(x))] \Rightarrow [[(\forall x)(P(x))] \wedge [(\forall x)(Q(x))]]$$

Důkaz:

Jelikož se jedná o implikaci, můžeme předpokládat $(\forall x)(P(x) \wedge Q(x))$ a dokázat $[[(\forall x)(P(x))] \wedge [(\forall x)(Q(x))]$. Znamená to tedy, že máme dokázat zvlášť $(\forall x)(P(x))$ a $(\forall x)(Q(x))$. Pokud dokazujeme $(\forall x)(P(x))$, zavedeme novou konstantu, například a a píšeme: „Nechť a je libovolné ale pevné“ a dokážeme $P[x \leftarrow a]$ tedy $P(a)$. Z předpokladu $(\forall x)(P(x) \wedge Q(x))$ můžeme produkovat nové známé věci. Například zvolíme term a a usoudíme $P(a) \wedge Q(a)$. Z tohoto předpokladu můžeme usoudit $P(a)$ a $Q(a)$. Tím máme dokázáno $P(a)$. Stejně tak postupujeme při dokazování $(\forall x)(Q(x))$. Zavedeme novou konstantu, například b a píšeme: „Nechť b je libovolné ale pevné“ a dokážeme $Q(b)$. Z předpokladu $(\forall x)(P(x) \wedge Q(x))$ můžeme produkovat nové známé věci. Například zvolíme term b a usoudíme $P(b) \wedge Q(b)$. Z tohoto předpokladu můžeme usoudit $P(b)$ a $Q(b)$. Tím máme dokázáno $Q(b)$.

Tabulka 1: Důkazová tabulka

Krok	**Předpokládáme**	**Dokazujeme**
1.	$(\forall x)(P(x) \wedge Q(x))$	$[(\forall x)(P(x))] \wedge [(\forall x)(Q(x))]$
2.		$(\forall x)(P(x))$ $(\forall x)(Q(x))$
3.		$P(a)$
4.	$P(a) \wedge Q(a)$	$P(a)$
5.	$P(a)$ $Q(a)$	$P(a)$
6.		$Q(b)$
7.	$P(b)$ $Q(b)$	$Q(b)$

Cvičení 4c

Zadání:

Dokažte následující formuli:

$$[(\exists x)(P(f(x)) \vee Q(g(x)))] \Rightarrow [[(\exists x)(P(x))] \vee [(\exists x)(Q(x))]]$$

Důkaz:

Pro důkaz formule nejprve předpokládejme $(\exists x)(P(f(x)) \vee Q(g(x)))$ a dokažme $[(\exists x)(P(x))] \vee [(\exists x)(Q(x))]$. Pro první předpoklad zvolme novou konstantu tak, že $x \leftarrow a$ a tedy platí $P(f(a)) \vee Q(g(a))$. Disjunkci dokážeme například tak, že předpokládáme $\neg[(\exists x)(P(x))]$ a dokážeme $[(\exists x)(Q(x))]$.

Nyní rozdělíme důkaz na dvě části. Nejprve předpokládejme, že platí $P(f(a))$. Dokážeme lemma $(\exists x)(P(x))$ jelikož se přímo nabízí zvolit term $f(a)$ a dokázat $P[x \leftarrow f(a)]$ tedy $P(f(a))$, což je ale triviálně dokázáno, neboť $P(f(a))$ máme v předpokladech. V této části důkazu získáváme spor.

V druhé části předpokládejme, že platí $Q(g(a))$. Nyní už stačí jen zvolit term $g(a)$ a dokázat, že platí $Q[x \leftarrow g(a)]$ tedy $Q(g(a))$. To je triviálně dokázáno, neboť $Q(g(a))$ už máme v předpokladech.

Tabulka 2: Důkazová tabulka

Krok	Předpokládáme	Dokazujeme
1.	$(\exists x)(P(f(x)) \vee Q(g(x)))$	$[(\exists x)(P(x))] \vee [(\exists x)(Q(x))]$
2.	$P(f(a)) \vee Q(g(a))$	$[(\exists x)(P(x))] \vee [(\exists x)(Q(x))]$
3.	$\neg(\exists x)(P(x))$	$(\exists x)(Q(x))$
4a.	$P(f(a))$	$(\exists x)(Q(x))$
5a.		lemma $(\exists x)(P(x))$
6a.	$(\exists x)(P(x)) \ldots \perp$	
4b.	$Q(g(a))$	$(\exists x)(Q(x))$
5b.		$Q(g(a))$

Cvičení 4d

Zadání:

Dokažte následující formuli:

$$[(\exists x)(S \Rightarrow Q(x))] \Rightarrow [S \Rightarrow (\exists x)(Q(x))]$$

Důkaz:

Jelikož se jedná o implikaci, můžeme předpokládat $(\exists x)(S \Rightarrow Q(x))$ a dokázat $S \Rightarrow (\exists x)(Q(x))$. Opět dokazujeme implikaci, takže předpokládáme S a dokážeme $(\exists x)(Q(x))$. Nyní se hodí vyprodukovat novou znalost z prvního předpokladu. Zavedeme novou konstantu a tak, že $(S \Rightarrow Q(x))[x \leftarrow a]$ a přidáme $S \Rightarrow Q(a)$ do seznamu předpokladů. V předpokladech máme implikaci a také předpokládáme, že platí S, můžeme tedy usoudit $Q(a)$. Pro důkaz $(\exists x)(Q(x))$ zvolíme term a, který dosadíme za x a dokážeme $Q(a)$. To je však již triviálně dokázáno.

Tabulka 3: Důkazová tabulka

Krok	Předpokládáme	Dokazujeme
1.	$(\exists x)(S \Rightarrow Q(x))$	$S \Rightarrow (\exists x)(Q(x))$
2.	S	$(\exists x)(Q(x))$
3.	$S \Rightarrow Q(a)$	$(\exists x)(Q(x))$
4.	$Q(a)$	$(\exists x)(Q(x))$
5.		$Q(a)$

Cvičení 4e

Zadání:

Dokažte následující formuli:

$$[(\neg\exists x)(\exists y)(T(x,y))] \Rightarrow [(\forall x)(\neg T(f(g(x)),f(x)))]$$

Důkaz:

Jelikož dokazujeme implikaci, můžeme předpokládat $(\neg\exists x)(\exists y)(T(x,y))$ a dokázat $(\forall x)(\neg T(f(g(x)),f(x)))$. Zvolme nyní novou konstantu, například a a nechť a je libovolné, ale pevné, a dokažme $\neg T[x \leftarrow a]$ tedy $(\neg T(f(g(a)),f(a)))$. Formuli, před kterou máme negaci, dokážeme tak, že předpokládáme, že platí a pokusíme se najít spor. K tomu abychom našli spor, se nám bude hodit dokázat lemma $(\exists x)(\exists y)(T(x,y))$. To dokážeme tak, že nejprve zvolíme term $f(g(a))$, dosadíme za x a dokážeme $(\exists y)(T(f(g(a)),y))$ a poté zvolíme term $f(a)$, dosadíme za y a dokážeme $T(f(g(a)),f(a))$, což už je ale díky předpokladu triviálně dokázáno. Nový předpoklad $(\exists x)(\exists y)(T(x,y))$ vytvoří spor, což úspěšně dokončí důkaz.

Tabulka 4: Důkazová tabulka

Krok	**Předpokládáme**	**Dokazujeme**
1.	$(\neg\exists x)(\exists y)(T(x,y))$	$(\forall x)(\neg T(f(g(x)),f(x)))$
2.		$(\neg T(f(g(a)),f(a)))$
3.	$T(f(g(a)),f(a))$	hledáme spor
4.		lemma $(\exists x)(\exists y)(T(x,y))$
5.	$(\exists x)(\exists y)(T(x,y)) \ldots \bot$	hledáme spor

Tabulka 5: Důkazová tabulka lemmatu

Krok	**Předpokládáme**	**Dokazujeme**
1.	$T(f(g(a)),f(a))$	lemma $(\exists x)(\exists y)(T(x,y))$
2.		$T[x \leftarrow f(g(a))]$
3.		$(\exists y)(T(f(g(a)),y))$
4.		$T[y \leftarrow f(a)]$
5.		$T(f(g(a)),f(a))$

Cvičení 4f

Úloha bude vypracována na cvičení.

MI-FME Cvičení 5

Tomáš Chvosta

Březen 2020

Vyhledávání řetězce z 1. sady úkolů

Zadání:

Navrhněte input/output specifikaci pro vyhledávání řetězce.

Vstup:

Pole znaků *text* délky n
Pole znaků *hledany* délky k

Výstup:

Pravda (resp. 1) - v případě, že se hledaný řetězec vyskytuje v zadaném textu.
Nepravda (resp. 0) - v případě, že se hledaný řetězec nevyskytuje v zadaném textu.

$$(\exists i \in \{0, \ldots, n-k\})(\forall j \in \{0, \ldots, k-1\})(text[i+j] = hledany[j])$$

Poznámky:

V této úloze vlastně hledám index v textu, kde začíná hledaný řetězec. Mělo by se mi to do toho textu vejít, proto je tam $i \leq n - k$. Když ten index mám, tak už jen stačí zkontrolovat jednotlivé znaky, jestli jsou stejné.

Cvičení 5a

Zadání:

Rozšiřte specifikaci uvedenou výše o certifikát pro případ, že je návratová hodnota 1. Popište certifikát formálně, využijte predikátovou logiku.

Řešení:

Certifikát: i kdy platí:

$$(i \in \{0, \ldots, n-k\}) \wedge ((\forall j \in \{0, \ldots, k-1\})(text[i+j] = hledany[j]))$$

Poznámka:

Certifikát představuje index v textu, kde hledaný řetězec začíná. Kdokoliv může zkontrolovat, že se na daném indexu skutečně vyskytuje hledaný řetězec.

Cvičení 5b

Zadání:

Rozšiřte specifikaci uvedenou výše o certifikát pro případ, že je návratová hodnota 0. Popište certifikát neformálně, využijte přirozený jazyk.

Úvaha:

V případě, že návratová hodnota byla 1, to bylo snadné. Stačilo spolu s odpovědí ano vrátit certifikát s indexem, kde se hledaný řetězec nachází. V případě, že je návratová hodnota 0, už to není tak snadné. Znamená to, že takový index neexistuje. Certifikát by měl tuto skutečnost dokazovat, tedy že pro každý index v původním textu platí, že to nemůže být začátek hledaného řetězce. Toho můžeme dosáhnout tak, že pro každý index v původním textu vrátíme hodnotu indexu v hledaném řetězci, kde se písmena liší. Tím certifikát jasně dokáže, že žádný index v původním textu nemůže být začátkem hledaného řetězce. Zároveň pro některé indexi je zbytečné tuto hodnotu hledat, u některých indexů víme, že nidky nemohou být začátkem hledaného řetězce. Například mějme text $text$ ="abcdaccddbc" a hledejme $hledany$ ="acd". Můžeme si všimnout, že $text[9]$ a $text[10]$ nemohou být počáteční písmena pro $hledany$, protože by se tam pak hledaný výraz nemohl vejít vzhledem k jeho délce.

Příklad:

Pojďme si nyní názorně ukázat, jak takový certifikát bude vypadat. Mějme text $text$ ="abcdaccddbc" a hledejme $hledany$ ="acd". Text můžeme reprezentovat jako pole znaků takto:

a	b	c	d	a	c	c	d	d	b	c

Nyní budeme pro každý znak v textu kontrolovat, zda může být začátkem pro hledaný řetězec. Poslední dva znaky můžeme rovnou vyloučit, na těch nemůže hledaný řetězec začínat, neboť by se tam nevešel. Začneme tím, že zkontrolujeme, zda se znak shoduje s prvním znakem v hledaném řetězci. Pokud ne

uložíme si 0 jakožto hodnotu indexu v hledaném řetězci, kde se neshoduje s původním textem:

	0	0	0		0	0	0	0	X	X

Pokračujeme tím, že zkontrolujeme, zda se znak shoduje s druhým znakem v hledaném řetězci. Pokud ne uložíme si 1:

1	0	0	0		0	0	0	0	X	X

A to samé s třetím znakem:

1	0	0	0	2	0	0	0	0	X	X

Nakonec tedy vrátíme tento certifikát:

1	0	0	0	2	0	0	0	0

Řešení

Certifikát je tedy pole délky $n - k + 1$, které má na každé i. té pozici uložený index z hledaného řetězce, na kterém se hledaný řetězec neshoduje s původním textem za předpokladu, že i představuje v původním textu potenciální začátek hledaného řetězce.

Cvičení 5c

Zadání:

Rozšiřte specifikaci uvedenou výše o certifikát pro případ, že je návratová hodnota 0. Popište certifikát formálně, využijte predikátovou logiku.

Řešení:

Pole P délky $n - k + 1$ takové, že platí:

$$(\forall i \in \{0, \ldots, n-k\})(hledany[P[i]] \neq text[i + P[i]])$$

MI-FME Cvičení 6

Tomáš Chvosta

Březen 2020

Zadání

Stáhněte si soubor `search.riscal` z Moodlu. Tento soubor reprezentuje specifikaci vyhledávání řetězce v RISCAL. V této specifikaci jsou řetězce reprezentovány jako pole znaků, kde znaky reprezentují přirozená čísla (včetně nuly).

- Spusťte soubor v RISCAL (checkbox "Nondeterminism" by neměl být zaškrtnut a jako "Operation" zvolte `exec`(), poté klikněte na zelenou šipku v sekci "Analysis".
- Přečtěte si zdrojový kód souboru a pokuste se ho pochopit.
- Upravte zdrojový kód tak, aby výskyt čísla 0 v řetezci a byl interpretován jako žolíkový znak, tedy aby představoval kterýkoliv znak.

Analýza

Program obsahuje jeden predikát *atpos* a jednu funkci *contains*. Predikát *atpos* vypadá následovně:

$$((len\_s \leq N) \wedge (p + len\_a \leq len\_s) \wedge (\forall i \in index)(i < len\_a \Rightarrow (s[p+i] = a[i])))$$

Proměnná s představuje řetězec délky len_s, ve kterém se vyhledává, a představuje hledaný řetězec délky len_a. Funkce *contains* vrací množinu indexů, které splňují predikát *atpos*. Jinými slovy tedy vrací indexy, na kterých se nachází hledaný řetězec a v řetězci s.

Řešení

Původní řešení nabízelo vyhledávání a výpis výskytů řetězce a v řetězci s. V predikátu *atpos* je jasně definováno, že od indexu, který má být výsledkem se všechny znaky musí shodovat pro všechny indexi v řetězci a. Nám tedy stačí rozšířit část predikátu $s[p+i] = a[i]$ o možnost, kdy je $a[i] = 0$. Pokud je totiž

$a[i] = 0$, pak $s[p+i]$ může být kterýkoliv znak. Predikát tedy upravíme na následující tvar:

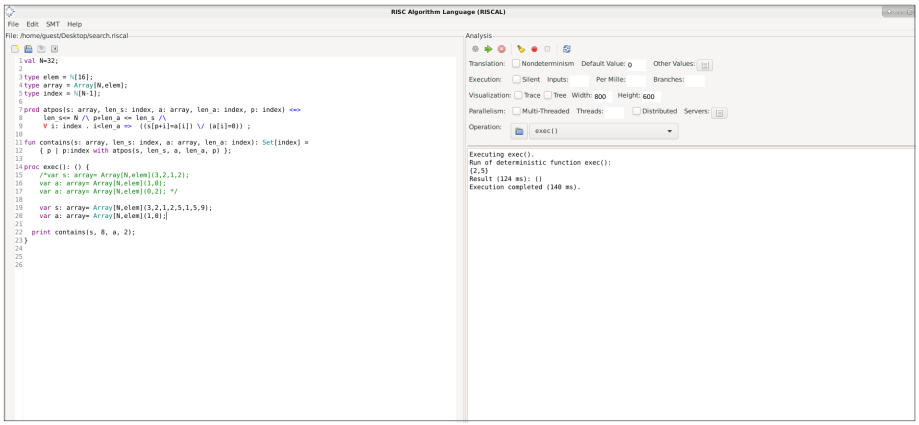
$$((len\_s \leq N) \wedge (p + len\_a \leq len\_s)$$

$$\wedge$$

$$(\forall i \in index)(i < len\_a \Rightarrow ((s[p+i] = a[i]) \ \vee \ (a[i] = 0))))$$

Touto změnou dosáhneme požadovaného výsledku. Na následujícím obrázku je vidět upravený kód a také jeden z běhů programu.

Obrázek 1: Ukázka programu search.riscal



MI-FME Cvičení 7

Tomáš Chvosta

Březen 2020

Cvičení 7a

Úloha byla vypracována na cvičení.

Cvičení 7b

Zadání:

V teorii listů dokažte následující formuli:

$$(\forall l, x, y)(l = cons(x, cons(y, empty())) \Rightarrow first(rest(l)) = y)$$

Důkaz:

Dokazujeme formuli, která má tvar $(\forall l, x, y)(F)$, což znamená, že zavedeme nové konstanty a,b a c. Nechť tyto konstanty a,b,c jsou libovolné ale pevné a dokažme $F[l \leftarrow a, x \leftarrow b, y \leftarrow c]$. Jedná se o implikaci, takže předpokládáme $a = cons(b, cons(c, empty()))$ a dokážeme $first(rest(a)) = c$. Využijeme jedno z ekvivalentních pravidel, že $(\neg\neg(first(rest(a)) = c)) \Leftrightarrow first(rest(a)) = c$. Nyní můžeme postupovat tak, že předpokládáme $\neg(first(rest(a)) = c)$ a najedeme spor. Do posledního předpokladu můžeme dosadit $a = cons(b, cons(c, empty()))$. V tuto chvíli můžeme využít jeden z axiomů, konkrétně $(\forall \mathrm{l}, \mathrm{x})(rest(cons(\mathrm{x}, \mathrm{l})) = \mathrm{l})$ a upravit poslední předpoklad na tvar $\neg(first(cons(c, empty()))) = c)$. Nyní můžeme využít dalšího axiomu, konkrétně $(\forall \mathrm{l}, \mathrm{x})(first(cons(\mathrm{x}, \mathrm{l})) = \mathrm{x})$ a upravit poslední předpoklad na tvar $\neg(c = c)$.

Tabulka 1: Důkazová tabulka

Krok	Předpokládáme	Dokazujeme
1.		$(a = cons(b, cons(c, empty())) \Rightarrow first(rest(a)) = c)$
2.	$a = cons(b, cons(c, empty()))$	$first(rest(a)) = c$
3.		$\neg\neg(first(rest(a)) = c)$
4.	$\neg(first(rest(a)) = c)$	hledáme spor
5.	$\neg(first(rest(cons(b, cons(c, empty())))) = c)$	hledáme spor
6.	axiom č. 1 $[x \leftarrow b, l \leftarrow cons(c, empty())]$	hledáme spor
7.	$\neg(first(cons(c, empty()))) = c)$	hledáme spor
8.	axiom č. 2 $[x \leftarrow c, l \leftarrow empty()]$	hledáme spor
9.	$\neg(c = c) \ldots \bot$	hledáme spor

Tabulka 2: Tabulka využitých axiomů ($\mathrm{l} \in \mathcal{L}[\mathcal{T}], \mathrm{x} \in \mathcal{T}$):

Číslo axiomu	Axiom
1.	$(\forall \mathrm{l}, \mathrm{x})(rest(cons(\mathrm{x}, \mathrm{l})) = \mathrm{l})$
2.	$(\forall \mathrm{l}, \mathrm{x})(first(cons(\mathrm{x}, \mathrm{l})) = \mathrm{x})$

Cvičení 7c

Zadání:

V teorii polí dokažte následující formuli:

$$(\exists a, i, j, x)(write(a, i, x)[j] \neq x)$$

Důkaz:

V tomto důkazu se snažíme dokázat formuli, která má tvar $(\exists a, i, j, x)(F)$, což znamená, že zvolíme nějaké termy, které dosadíme za a, i, j, x a tuto formuli poté dokážeme. Pro tento důkaz můžeme zvolit termy $k, l, w, write(a, l, v)$ a následně dokážeme $F[a \leftarrow write(a, l, v), i \leftarrow k, x \leftarrow w, j \leftarrow l]$. Je nutné podotknout, že $k \neq l$ a také $w \neq v$. Proměnná v značí libovolnou hodnotu. Nyní nastává čas využít axiom $(\forall a, v, i, j)(\neg[i = j] \Rightarrow write(a, i, v)[j] = a[j])$. Víme, že $k \neq l$, pak tedy podle tohoto axiomu platí, že $write(write(a, l, v), k, w)[l] = write(a, l, v)[l]$. Nyní využijeme axiom $(\forall a, v, i, j)(i = j \Rightarrow write(a, i, v)[j] = v)$. Podle tohoto axiomu můžeme usoudit, že $write(a, l, v)[l] = v$. Dostáváme $v \neq w$, což už ale máme triviálně dokázáno.

Tabulka 3: Důkazová tabulka

Krok	Předpokládáme	Dokazujeme
1.		$F[a \leftarrow write(a,l,v),$ $i \leftarrow k,$ $x \leftarrow w,$ $j \leftarrow l],$ kde $w \neq v$ a $k \neq l$
2.	axiom č. 1 $[a \leftarrow write(a,l,v),$ $i \leftarrow k,$ $v \leftarrow w,$ $j \leftarrow l]$	$write(write(a,l,v),k,w)[l] \neq w$
3.	axiom č. 2 $[a \leftarrow a,$ $i \leftarrow l,$ $v \leftarrow v,$ $j \leftarrow l]$	$write(a,l,v)[l] \neq w$
4.		$v \neq w$

Tabulka 4: Tabulka využitých axiomů ($\mathrm{a} \in \mathcal{A}[\mathcal{T}], \mathrm{v} \in \mathcal{T}, \mathrm{i,j} \in \mathcal{N}$)):

Číslo axiomu	Axiom
1.	$(\forall a,v,i,j)(i = j \Rightarrow write(a,i,v)[j] = v)$
2.	$(\forall a,v,i,j)(\neg[i = j] \Rightarrow write(a,i,v)[j] = a[j])$

Cvičení 7d

Zadání:

V teorii polí dokažte následující formuli:

$$(\forall a,i,j,x)((write(a,i,x)[j] = x) \Rightarrow (i = j \vee a[j] = x))$$

Důkaz:

Dokazujeme formuli, která má tvar $(\forall a,i,j,x)(F)$, což znamená, že zavedeme nové konstanty p,k,l a v. Nechť tyto konstanty p,k,l,v jsou libovolné ale pevné a dokažme $F[a \leftarrow p, i \leftarrow k, j \leftarrow l, x \leftarrow v]$. Dokazujeme implikaci, takže předpokládáme $write(p,k,v)[l] = v$ a dokážeme $k = l \vee p[l] = v$. Pro důkaz disjunkce předpokládejme, že platí $\neg(k = l)$ a dokažme $p[l] = v$. Nyní využijme následující axiom: $(\forall a,v,i,j)(\neg[i = j] \Rightarrow write(a,i,v)[j] = a[j])$. Jelikož ho můžeme použít jako platný předpoklad, můžeme také vhodně zvolit termy a usoudit $write(p,k,v)[l] = p[l]$. Spolu s předpokladem $write(p,k,v)[l] = v$ můžeme usoudit, že $p[l] = v$, čímž úspěšně dokončíme důkaz.

Tabulka 5: Důkazová tabulka

Krok	**Předpokládáme**	**Dokazujeme**
1.		$write(a, i, x)[j] = x$ $\Rightarrow$ $i = j \vee a[j] = x$
2.	$write(p, k, v)[l] = v$	$k = l \vee p[l] = v$
3.	$\neg(k = l)$	$p[l] = v$
4.	axiom č. 1 $[a \leftarrow p,$ $v \leftarrow v,$ $i \leftarrow k,$ $j \leftarrow l]$	$p[l] = v$
5.	$write(p, k, v)[l] = p[l]$	$p[l] = v$
5.	$p[l] = v$	$p[l] = v$

Tabulka 6: Tabulka využitých axiomů ($\mathrm{a} \in \mathcal{A}[\mathcal{T}], \mathrm{v} \in \mathcal{T}, \mathrm{i}, \mathrm{j} \in \mathcal{N}$)):

Číslo axiomu	**Axiom**
1.	$(\forall a, v, i, j)(\neg[i = j] \Rightarrow write(a, i, v)[j] = a[j])$

Cvičení 7e

Zadání:

Dokažte pomocí Peano axiomů s vyjímkou indukčního axiomu následující formuli:

$$(\forall k)(0 + k = k)$$

Využijte principu slabé matematické indukce.

Důkaz:

Dle principu slabé matematické indukce nejprve provedeme důkaz pro $k = 0$. Dokazujeme tedy $(\forall k)(0 + k = k)$ a zavedeme novou konstantu 0 a dosadíme $(0 + k = k)[k \leftarrow 0]$ a tedy dokážeme $0 + 0 = 0$. Pro dokázání této formule budeme potřebovat jeden z axiomů, konkrétně $(\forall x)(x + 0 = x)$, kdy můžeme zvolit term 0 a usoudit $(x + 0 = x)[x \leftarrow 0]$ tedy $0 + 0 = 0$, což úspěšně dokončí tuto část důkazu.

Tabulka 7: Důkazová tabulka $k = 0$

Krok	**Předpokládáme**	**Dokazujeme**
1.		$(\forall k)(0 + k = k)$ $[k \leftarrow 0]$
2.	$(\forall x)(x + 0 = x)$ $[x \leftarrow 0]$	$0 + 0 = 0$
3.	$0 + 0 = 0$	$0 + 0 = 0$

V druhé části důkazu (matematické indukce) předpokládáme $(0 + k = k)$ a dokážeme formuli pro $k \leftarrow k+1$. Pokusíme se tedy dokázat $0+(k+1) = k+1$, k čemuž se nám bude hodit axiom $(\forall x, y)(x+(y+1) = (x+y)+1)$. V této části zvolíme dva termy 0 a k a usoudíme $(x + (y + 1) = (x + y) + 1)[x \leftarrow 0, y \leftarrow k]$. Vzniknul nám tedy nový předpoklad $0 + (k + 1) = (0 + k) + 1$, do kterého můžeme z prvního předpokladu dosadit $0+k = k$. Tím získáme nový předpoklad $0 + (k + 1) = k + 1$, čímž úspěšně dokončíme důkaz.

Tabulka 8: Důkazová tabulka $k = k + 1$

Krok	**Předpokládáme**	**Dokazujeme**
1.	$0 + k = k$	$(\forall k)(0 + k = k)$ $[k \leftarrow k + 1]$
2.		$0 + (k + 1) = k + 1$
3.	$(\forall x, y)(x + (y + 1) = (x + y) + 1)$ $[x \leftarrow 0, y \leftarrow k]$	$0 + (k + 1) = k + 1$
4.	$0 + (k + 1) = (0 + k) + 1$	$0 + (k + 1) = k + 1$
5.	$0 + (k + 1) = k + 1$	$0 + (k + 1) = k + 1$

Cvičení 7f

Úloha bude vypracována na cvičení.

MI-FME Cvičení 8

Tomáš Chvosta

Březen 2020

Zadání

Napište formuli Φ_P definující přechodovou relaci následujícího programu P:

```
1: i ← 1
2: while i < 10 do
3:     input x
4:     i ← i + (x ≥ 0)?1 : 2
5:     a[i] ← a[i − 1] + x
6: return
```

Řešení

Přechodová relace Φ_P:

$$[pc = 1 \wedge pc' = pc + 1 \wedge i' = 1 \wedge x' = x \wedge a' = a]$$

$$\vee$$

$$[pc = 2 \wedge [i < 10 \Rightarrow pc' = pc + 1] \wedge [i \geq 10 \Rightarrow pc' = 6] \wedge i' = i \wedge x' = x \wedge a' = a]$$

$$\vee$$

$$[pc = 3 \wedge pc' = pc + 1 \wedge i' = i \wedge x' = input() \wedge a' = a]$$

$$\vee$$

$$[pc = 4 \wedge pc' = pc + 1 \wedge [x \geq 0 \Rightarrow i' = i + 1] \wedge$$
$$[x < 0 \Rightarrow i' = i + 2] \wedge x' = x \wedge a' = a]$$

$$\vee$$

$$[pc = 5 \wedge pc' = 2 \wedge i' = i \wedge x' = x \wedge a' = write(a, i, a[i - 1] + x)]$$

$$\vee$$

$$[pc = 6 \wedge \bot]$$

Poznámky opravujícího

Řádek 3: na přednášce jsem podrobně vysvětlil, proč to takto nefunguje, podívejte se prosím ještě jednou na to.

Oprava

Správně tedy opravený 3. řádek vypadá nejspíše takto:

$$[pc = 3 \wedge pc' = pc + 1 \wedge i' = i \wedge a' = a]$$

Vycházím tak ze slajdů přednášky 4, která se týká operační sémantiky programů. Zde je totiž na slajdu 19 uvedeno, že pokud $s(pc)$ odkazuje na řádek **input** v, potom:

$$s \sqcup \pi(s') \models pc' = pc + 1 \wedge \bigwedge_{u \in V, u \neq v} u' = u$$

MI-FME Cvičení 9

Tomáš Chvosta

Březen 2020

Převeďte každou z následujících základních cest na formu SSA (Static Single Assignment Form) a zapište si její podmínku ověření. Zkontrolujte, zda platí podmínka ověření. Pokud platí, napište krátký důkaz. Pokud tomu tak není, najděte protipříklad, to znamená proměnné přiřazení, které ukazuje, že vzorec neplatí, a které zároveň představuje počáteční stav, který ale po následném provedení vede k chybě.

Cvičení 9a

Zadání:

assume $x \geq 0$
$y \leftarrow x$
$z \leftarrow y$
@ $z \geq 0$

Static Single Assignment forma:

V tomto případě není potřeba zavádět nové názvy proměnných, jelikož žádná z proměnných nenabývá podruhé nové hodnoty. SSA forma má tedy následující tvar:

assume $x \geq 0$
$y \leftarrow x$
$z \leftarrow y$
@ $z \geq 0$

Logická formule z SSA:

Logická formule z SSA vypadá následovně:

$$(\forall x, y, z)([x \geq 0 \wedge y = x \wedge z = y] \Rightarrow z \geq 0)$$

Ověřovací podmínky:

Pro důkaz použijeme metodu ručního dokazování z přednášky. Předpokládáme $x \geq 0$, $y = x$, $z = y$ a dokážeme $z \geq 0$. Kvůli předpokladu $z = y$ stačí dokázat $y \geq 0$. Kvůli předpokladu $y = x$ stačí dokázat $x \geq 0$, což je ale zároveň jeden z předpokladů, takže formule platí a můžeme tvrdit, že program je napsán korektně.

Cvičení 9b

Zadání:

assume $x \geq 0$
$z \leftarrow y$
$y \leftarrow x$
@ $z \geq 0$

Static Single Assignment forma:

Můžeme si všimnout, že ve druhém kroku přiřadíme hodnotu y do z a následně přiřadíme do y novou hodnotu x. Je tedy potřeba zavést nový název proměnné pro toto přiřazení. SSA forma má tedy následující tvar:

assume $x \geq 0$
$z \leftarrow y$
$y_1 \leftarrow x$
@ $z \geq 0$

Logická formule z SSA:

Logická formule z SSA vypadá následovně:

$$(\forall x, y, y_1, z)([x \geq 0 \wedge z = y \wedge y_1 = x] \Rightarrow z \geq 0)$$

Ověřovací podmínky:

Jelikož je na první pohled zřejmé, že formule nebude logicky platná, pokusíme se najít protipříklad a tím dokázat, že formule neplatí. Můžeme využít předpokladu $z = y$ a dosadit ho do pravé strany formule. Získáme tedy závěr, že $y \geq 0$. Tedy pro $y < 0$ závěr očividně neplatí. Protipříklad tedy může vypadat například takto:

$$\{x \mapsto 8, y \mapsto -5, y_1 \mapsto 8, z \mapsto -5\}$$

Toto ohodnocení proměnných splňuje levou stranu formule, avšak nesplňuje pravou stranu. Z toho můžeme usoudit, že formule neplatí. Toto ohodnocení klidně může být i počátečním stavem. Můžeme si pro představu ukázat běh programu:

Krok	x	y	z
assume $x \geq 0$	8	-5	
$z \leftarrow y$	8	-5	-5
$y \leftarrow x$	8	8	-5
@ $z \geq 0$	8	8	-5

Z tabulky si můžeme všimnout, že ve chvíli, kdy má být hodnota proměnné $z \geq 0$ je $z = -5$. Program tedy není korektní.

Cvičení 9c

Zadání:

assume $x < 0$
$x \leftarrow x - k$
assume $k \leq 1$
@ $x \geq 0$

Static Single Assignment forma:

Můžeme si všimnout, že na začátku předpokládáme nějakou hodnotu $x < 0$ a poté ve druhém kroku přiřadíme hodnotu $x - k$ do x. Je tedy potřeba zavést nový název proměnné pro toto přiřazení. SSA forma má tedy následující tvar:

assume $x < 0$
$x_1 \leftarrow x - k$
assume $k \leq 1$
@ $x_1 \geq 0$

Logická formule z SSA:

Logická formule z SSA vypadá následovně:

$$(\forall x, x_1, k)([x < 0 \wedge x_1 = x - k \wedge k \leq 1] \Rightarrow x_1 \geq 0)$$

Ověřovací podmínky:

Při dokazování, že platí $x_1 \geq 0$ můžeme využít předpokladu $x_1 = x - k$ a dokazovat $x - k \geq 0$. To můžeme pomocí základních matematických pravidel upravit na tvar $x \geq k$. To znamená, že stačí najít protipříklad, který bude splňovat $x < k$. Protipříklad tedy může vypadat například takto:

$$\{x \mapsto -5, x_1 \mapsto -1, k \mapsto -4\}$$

Toto ohodnocení proměnných splňuje levou stranu formule, avšak nesplňuje pravou stranu. Z toho můžeme usoudit, že formule neplatí. Můžeme si pro představu ukázat běh programu:

Krok	x	k
assume $x < 0$	-5	-4
$x \leftarrow x - k$	-1	-4
assume $k \leq 1$	-1	-4
@ $x \geq 0$	-1	-4

Z tabulky si můžeme všimnout, že ve chvíli, kdy má být hodnota proměnné $x \geq 0$ je $x = -1$. Program tedy není korektní.

Cvičení 9d

Zadání:

assume $k \leq x$
$x \leftarrow x - k$
@ $x \geq 0$

Static Single Assignment forma:

Můžeme si všimnout, že ve druhém kroku přiřazujeme proměnné x novou hodnotu. Je tedy potřeba zavést nový název proměnné pro toto přiřazení. SSA forma má tedy následující tvar:

assume $k \leq x$
$x_1 \leftarrow x - k$
@ $x_1 \geq 0$

Logická formule z SSA:

Logická formule z SSA vypadá následovně:

$$(\forall x, x_1, k)([k \leq x \wedge x_1 = x - k] \Rightarrow x_1 \geq 0)$$

Ověřovací podmínky:

Pro důkaz použijeme metodu ručního dokazování z přednášky. Předpokládáme $k \leq x$, $x_1 = x - k$ a dokážeme $x_1 \geq 0$. Kvůli předpokladu $x_1 = x - k$ stačí dokázat $x - k \geq 0$, tedy $x \geq k$. To však triviálně dokazuje předpoklad $k \leq x$. Můžeme tedy tvrdit, že program je napsán korektně.

Cvičení 9e

Zadání:

$x \leftarrow x - k$
assume $k \leq x$

@ $x \geq 0$

Static Single Assignment forma:

Můžeme si všimnout, že v prvním kroku přiřazujeme proměnné x novou hodnotu. Je tedy potřeba zavést nový název proměnné pro toto přiřazení. SSA forma má tedy následující tvar:

$x_1 \leftarrow x - k$
assume $k \leq x_1$
@ $x_1 \geq 0$

Logická formule z SSA:

Logická formule z SSA vypadá následovně:

$$(\forall x, x_1, k)([x_1 = x - k \wedge k \leq x_1] \Rightarrow x_1 \geq 0)$$

Ověřovací podmínky:

Při dokazování, že platí $x_1 \geq 0$ můžeme využít předpokladu $x_1 = x - k$ a dokazovat $x - k \geq 0$. To můžeme pomocí základních matematických pravidel upravit na tvar $x \geq k$. Zároveň můžeme získat nový předpoklad složení předpokladů $x_1 = x - k$, $k \leq x_1$. Získáme předpoklad $k \leq x - k$ tedy $2k \leq x$. Máme tedy předpoklad $2k \leq x$ a závěr $x \geq k$. Můžeme si však všimnout, že nejspíše bude existovat nějaké x a nějaké k, pro které bude splněn předpoklad, ale nebude platit závěr. Pojďmě ho najít. Hledáme tedy protipříklad, kdy $2k \leq x \wedge x < k$ tedy $2k \leq x < k$. Pro úpravě získáme, že $k < 0$. Zvolme tedy například $k = -1$ a x tak, aby platilo $-2 \leq x < -1$, tedy například $x = -2$. Protipříklad tedy může vypadat například takto:

$$\{x \mapsto -2, x_1 \mapsto -1, k \mapsto -1\}$$

Toto ohodnocení proměnných splňuje levou stranu formule, avšak nesplňuje pravou stranu. Z toho můžeme usoudit, že formule neplatí. Můžeme si pro představu ukázat běh programu:

Krok	x	k
initial state	-2	-1
$x \leftarrow x - k$	-1	-1
assume $k \leq x$	-1	-1
@ $x \geq 0$	-1	-1

Z tabulky si můžeme všimnout, že ve chvíli, kdy má být hodnota proměnné $x \geq 0$ je $x = -1$. Program tedy není korektní.

Cvičení 9f

Zadání:

assume $k \geq 0$
$x \leftarrow x - k$
assume $k \leq x$
@ $x \geq 0$

Static Single Assignment forma:

Můžeme si všimnout, že ve druhém kroku přiřazujeme proměnné x novou hodnotu. Je tedy potřeba zavést nový název proměnné pro toto přiřazení. SSA forma má tedy následující tvar:

assume $k \geq 0$
$x_1 \leftarrow x - k$
assume $k \leq x_1$
@ $x_1 \geq 0$

Logická formule z SSA:

Logická formule z SSA vypadá následovně:

$$(\forall x, x_1, k)([k \geq 0 \wedge x_1 = x - k \wedge k \leq x_1] \Rightarrow x_1 \geq 0)$$

Ověřovací podmínky:

Předpokládáme $k \geq 0$, $x_1 = x - k$, $k \leq x_1$ a dokážeme $x_1 \geq 0$. Spojením prvního a třetího předpokladu získáme předpoklad $0 \leq k \leq x_1$. Závěr $x_1 \geq 0$ je ekvivalentní s $\neg\neg(x_1 \geq 0)$. Jelikož dokazujeme negaci, můžeme předpokládat $\neg(x_1 \geq 0)$ tedy $x_1 < 0$ a najít spor. Spojením předpokladu $0 \leq k \leq x_1$ s předpokladem $x_1 < 0$ získáme předpoklad $0 \leq k \leq x_1 < 0$ tedy po úpravě $0 < 0$, což je spor. Formule tedy platí a můžeme tvrdit, že program je napsán korektně.

Cvičení 9g

Zadání:

assume $x \geq 0$
$y \leftarrow x$
input x
@ $x \geq 0$

Static Single Assignment forma:

Jelikož ve třetím kroku načítáme do x novou hodnotu, je potřeba zavést nový název proměnné. SSA forma má tedy následující tvar:

assume $x \geq 0$
$y \leftarrow x$
input x_1
@ $x_1 \geq 0$

Logická formule z SSA:

Logická formule z SSA vypadá následovně:

$$(\forall x, x_1, y)([x \geq 0 \wedge y = x \wedge \top] \Rightarrow x_1 \geq 0)$$

Ověřovací podmínky:

Na pravé straně formule máme závěr $x_1 \geq 0$, nicméně v předpokladech není x_1 nijak omezeno. Je to způsobeno tím, že bezprostředně po načtení nové hodnoty do x_1 uživatelským vstupem, má být $x_1 \geq 0$. Můžeme tedy velmi snadno najít protipříklad, pro který formule neplatí, například:

$$\{x \mapsto 8, y \mapsto 8, x_1 \mapsto -1\}$$

Toto ohodnocení proměnných splňuje levou stranu formule, avšak nesplňuje pravou stranu. Z toho můžeme usoudit, že formule neplatí. Můžeme si pro představu ukázat běh programu:

Krok	x	y
assume $x \geq 0$	8	
$y \leftarrow x$	8	8
input x	-1	8
@ $x \geq 0$	-1	8

Z tabulky si můžeme všimnout, že ve chvíli, kdy má být hodnota proměnné $x \geq 0$ je $x = -1$. Program tedy není korektní.

Cvičení 9h

Zadání:

$y \leftarrow x$
input x
assume $y \geq 0$
@ $y \geq 0$

Static Single Assignment forma:

Jelikož ve druhém kroku načítáme do x novou hodnotu, je potřeba zavést nový název proměnné. SSA forma má tedy následující tvar:

$y \leftarrow x$
input x_1
assume $y \geq 0$
@ $y \geq 0$

Logická formule z SSA:

Logická formule z SSA vypadá následovně:

$$(\forall x, x_1, y)([y = x \wedge \top \wedge y \geq 0] \Rightarrow y \geq 0)$$

Ověřovací podmínky:

Máme předpoklady $y = x$, $y \geq 0$ a máme dokázat $y \geq 0$. Vidíme, že $y \geq 0$ je zároveň předpoklad i závěr, formule je tedy triviálně dokázána a můžeme tvrdit, že program je napsán korektně.

Cvičení 9i

Zadání:

$y \leftarrow x$
input x
assume $x \geq 0$
@ $y \geq 0$

Static Single Assignment forma:

Jelikož ve druhém kroku načítáme do x novou hodnotu, je potřeba zavést nový název proměnné. SSA forma má tedy následující tvar:

$y \leftarrow x$
input x_1
assume $x_1 \geq 0$
@ $y \geq 0$

Logická formule z SSA:

Logická formule z SSA vypadá následovně:

$$(\forall x, x_1, y)([y = x \wedge \top \wedge x_1 \geq 0] \Rightarrow y \geq 0)$$

Ověřovací podmínky:

Máme předpoklady $y = x$, $x_1 \geq 0$ a máme dokázat $y \geq 0$. Kvůli předpokladu $y = x$ dokazujeme $x \geq 0$. Dále však nemáme žádný předpoklad, který by nějak omezoval x. Stačí tedy najít protipříklad takový, že $x < 0$. Protipříklad tedy může vypadat například takto:

$$\{x \mapsto -1, y \mapsto -1, x_1 \mapsto 8\}$$

Toto ohodnocení proměnných splňuje levou stranu formule, avšak nesplňuje pravou stranu. Z toho můžeme usoudit, že formule neplatí. Můžeme si pro představu ukázat běh programu:

Krok	x	y
$y \leftarrow x$	-1	-1
input x	8	-1
assume $x \geq 0$	8	-1
@ $y \geq 0$	8	-1

Z tabulky si můžeme všimnout, že ve chvíli, kdy má být hodnota proměnné $y \geq 0$ je $y = -1$. Program tedy není korektní.

MI-FME Cvičení 10

Tomáš Chvosta

Duben 2020

Uvažujte následující program (všechny proměnné náleží množině přirozených čísel včetně nuly):

$x_0 \leftarrow x$
$i \leftarrow 0$
while $i < n$ **do**
    @ $x = x_0 + \sum_{k=0}^{i-1} a[k]$
    $x \leftarrow x + a[i]$
    $i \leftarrow i + 1$
@ $x = x_0 + \sum_{k=0}^{n-1} a[k]$

Cvičení 10a

Zadání

Napište všechny základní cesty programu uvedeného výše. Dbejte na to, aby cesty začínající v cyklu měly na začátku odpovídající předpoklad.

Řešení

Nejprve si můžeme všimnout, že program obsahuje while cyklus s podmínkou $i < n$, pomocí které můžeme náš program rozdělit na dvě základní cesty s předpoklady, že tato podmínka platí a neplatí. Dále samotný cyklus obsahuje assertaci na svém začátku, která způsobí, že tělo cyklu také můžeme rozdělit na dvě základní cesty. Celkem tedy získáváme následující 4 základní cesty:

Základní cesta 1:

$x_0 \leftarrow x$
$i \leftarrow 0$
assume $\neg(i < n)$
@ $x = x_0 + \sum_{k=0}^{n-1} a[k]$

Zde tedy předpokládáme, že podmínka cyklu není splněna a rovnou přejdeme k assertaci na konci programu.

Základní cesta 2:

$x_0 \leftarrow x$
$i \leftarrow 0$
assume $(i < n)$
@ $x = x_0 + \sum_{k=0}^{i-1} a[k]$

Zde naopak předpokládáme, že podmínka cyklu je splněna a přejdeme na začátek cyklu k assertaci, kterou tato základní cesta končí.

Základní cesta 3:

assume $x = x_0 + \sum_{k=0}^{i-1} a[k]$
$x \leftarrow x + a[i]$
$i \leftarrow i + 1$
assume $(i < n)$
@ $x = x_0 + \sum_{k=0}^{i-1} a[k]$

Zde vidíme, že této části musela předcházet buď základní cesta 2, nebo 3 a tedy musíme na začátek zapsat příslušný předpoklad. V tomto případě se jedná o předpoklad $x = x_0 + \sum_{k=0}^{i-1} a[k]$. Na konci této základní cesty předpokládáme, že hodnota i je pořád menší než hodnota n a tedy přejdeme k assertaci na začátku cyklu, kterou tato základní cesta končí.

Základní cesta 4:

assume $x = x_0 + \sum_{k=0}^{i-1} a[k]$
$x \leftarrow x + a[i]$
$i \leftarrow i + 1$
assume $\neg(i < n)$
@ $x = x_0 + \sum_{k=0}^{n-1} a[k]$

Zde musí být na začátku opět stejný předpoklad jako u předchozí základní cesty. Na konci však předpokládáme, že podmínka cyklu není splněna, tedy hodnota i není menší než hodnota n. Přejdeme tedy k assertaci na konci programu.

Cvičení 10b

Zadání

Pro každou základní cestu vytvořte ověřovací podmínku a případně se ji pokuste dokázat.

Řešení

Všechny základní cesty vytvořené v předchozí sekci byly převedeny do SSA formy a následně byla z této formy vytvořena logická formule. Zde už budu uvádět pouze logické formule.

Základní cesta 1:

$$(\forall x, x_0, i, n, a)([x_0 = x \wedge i = 0 \wedge \neg(i < n)] \Rightarrow x = x_0 + \sum_{k=0}^{n-1} a[k])$$

Máme tedy předpoklady $x_0 = x$, $i = 0$, $\neg(i < n)$ a pokusíme se dokázat $x = x_0 + \sum_{k=0}^{n-1} a[k]$. Předpoklad $\neg(i < n)$ můžeme zapsat jako $i \geq n$. Pomocí předpokladu $i = 0$ můžeme dosazením získat nový předpoklad $n \leq 0$, a díky tomuto předpokladu víme, že horní hranice sumy $\sum_{k=0}^{n-1} a[k]$ bude menší než 0. To však nutně znamená, že $\sum_{k=0}^{n-1} a[k] = 0$ a že potřebujeme dokázat $x = x_0$, což je ale jeden z předpokladů. Ověřovací podmínka pro základní cestu 1 platí.

Základní cesta 2:

$$(\forall x, x_0, i, n, a)([x_0 = x \wedge i = 0 \wedge i < n] \Rightarrow x = x_0 + \sum_{k=0}^{i-1} a[k])$$

Máme tedy předpoklady $x_0 = x$, $i = 0$, $i < n$ a pokusíme se dokázat pravou stranu formule $x = x_0 + \sum_{k=0}^{i-1} a[k]$. Můžeme využít předpokladu $i = 0$, který dosadíme do sumy a získáme $\sum_{k=0}^{-1} a[k] = 0$. Máme tedy dokázat $x = x_0$, což je ale jeden z předpokladů. Ověřovací podmínka pro základní cestu 2 platí.

Základní cesta 3:

$$(\forall x, x_0, x_1, i, i_1, n, a)$$

$$([x = x_0 + \sum_{k=0}^{i-1} a[k] \wedge x_1 = x + a[i] \wedge i_1 = i + 1 \wedge i_1 < n] \Rightarrow x_1 = x_0 + \sum_{k=0}^{i_1-1} a[k])$$

Máme tedy předpoklady $x = x_0 + \sum_{k=0}^{i-1} a[k]$, $x_1 = x + a[i]$, $i_1 = i + 1$, $i_1 < n$ a potřebujeme dokázat $x_1 = x_0 + \sum_{k=0}^{i_1-1} a[k]$. Můžeme využít předpokladů $x = x_0 + \sum_{k=0}^{i-1} a[k]$ a $x_1 = x + a[i]$, dosadit první do druhého, čímž získáme nový předpoklad $x_1 = x_0 + \sum_{k=0}^{i} a[k]$. Pokud použijeme předpoklad $i_1 = i + 1$ a dosadíme ho do dokazovaného výrazu $x_1 = x_0 + \sum_{k=0}^{i_1-1} a[k]$, získáme $x_1 = x_0 + \sum_{k=0}^{i} a[k]$, což už je ale jeden z předpokladů. Formule je tedy dokázána a ověřovací podmínka pro základní cestu 3 platí.

Základní cesta 4:

$$(\forall x, x_0, x_1, i, i_1, n, a)$$

$$([x = x_0 + \sum_{k=0}^{i-1} a[k] \wedge x_1 = x + a[i] \wedge i_1 = i + 1 \wedge \neg(i_1 < n)] \Rightarrow x_1 = x_0 + \sum_{k=0}^{n-1} a[k])$$

Máme tedy předpoklady $x = x_0 + \sum_{k=0}^{i-1} a[k]$, $x_1 = x + a[i]$, $i_1 = i + 1$, $\neg(i_1 < n)$ a potřebujeme dokázat $x_1 = x_0 + \sum_{k=0}^{n-1} a[k]$. Můžeme využít předpokladů $x = x_0 + \sum_{k=0}^{i-1} a[k]$ a $x_1 = x + a[i]$, dosadit první do druhého, čímž získáme nový předpoklad $x_1 = x_0 + \sum_{k=0}^{i} a[k]$. Dále můžeme podobným způsobem využít předpoklady $i_1 = i + 1$, $\neg(i_1 < n)$ a získat nový předpoklad $i \geq n - 1$. Nyní můžeme dosadit předpoklad $x_1 = x_0 + \sum_{k=0}^{i} a[k]$ do dokazované formule $x_1 = x_0 + \sum_{k=0}^{n-1} a[k]$ a po úpravě získáme $\sum_{k=0}^{i} a[k] = \sum_{k=0}^{n-1} a[k]$, což je třeba dokázat. To však platí pouze pro $i = n - 1$, ale dle předpokladu je $i \geq n - 1$. Pro $i > n - 1$ tedy formule nemusí platit a tedy ověřovací podmínka pro základní cestu 4 neplatí.

Můžeme si ukázat jednoduchý protipříklad, pro který formule neplatí. Například tento protipříklad:

$$\{x \mapsto 40, x_0 \mapsto 10, x_1 \mapsto 70, i \mapsto 2, i_1 \mapsto 3, n \mapsto 2, a \mapsto [10, 20, 30]\}$$

Toto ohodnocení proměnných splňuje levou stranu formule, avšak nesplňuje pravou stranu. Z toho můžeme usoudit, že formule neplatí.

Cvičení 10c

Zadání

Zkontrolujte všechny ověřovací podmínky. Pokud nějaká ověřovací podmínka neplatí, upravte assertaci v cyklu tak, aby všechny ověřovací podmínky platily. Neměňte však assertaci na konci programu.

Řešení

V předchozí sekci si můžeme všimnout, že platí všechny podmínky kromě ověřovací podmínky u základní cesty 4. Tato ověřovací podmínka neplatí v případě, že $i > n - 1$ v průběhu cyklu. Podíváme-li se do původního programu, zjistíme, že k takové situaci nemůže dojít, jelikož je u while cyklu podmínka $i < n$. Stačí tedy tuto podmínku přidat i do assertace na začátku while cyklu. Program tedy po úpravě bude vypadat následovně:

$x_0 \leftarrow x$
$i \leftarrow 0$
while $i < n$ **do**

@ $x = x_0 + \sum_{k=0}^{i-1} a[k] \wedge i < n$
$x \leftarrow x + a[i]$
$i \leftarrow i + 1$
@ $x = x_0 + \sum_{k=0}^{n-1} a[k]$.

Pojďme se nyní podívat, jak se změní ověřovací podmínky v předchozí sekci po této úpravě.

Změna v základní cestě 1:
Základní cesta 1 zůstává beze změny a tedy i její ověřovací podmínka.

Změna v základní cestě 2:
V závěru základní cesty přibyde $\wedge(i < n)$, což je triviálně dokázáno díky předpokladu $i < n$.

Změna v základní cestě 3:
Přibyde předpoklad $(i < n)$, který v našem případě nemá žádný vliv na platnost ověřovací podmínky.

Změna v základní cestě 4:
Přibyde předpoklad $(i < n)$, díky kterému ověřovací podmínka platí.

MI-FME Cvičení 11

Tomáš Chvosta

Duben 2020

Uvažujte následující specifikaci programu:

- Input: $x, y \in \mathbb{N}_0, y \geq 1$
- Output: xy

$r \leftarrow x$
$i \leftarrow 1$
while $i < y$ **do**
    @
    $r \leftarrow r + x$
    $i \leftarrow i + 1$
@ $r = xy$
return r

Cvičení 11a

Zadání

Vytvořte tabulku, která bude ukazovat hodnoty proměnných r a i v místě programu s chybějící assertací ve všech iteracích cyklu pro nějaké netriviální vstupy x a y.

Řešení

Nejprve pojďme zvolit dvě libovolné hodnoty x a y (samozřejmě tak, aby platilo, že $x, y \in \mathbb{N}_0, y \geq 1$). Například $x = 10$ a $y = 5$. Následující tabulka zobrazuje hodnoty proměnných i a r v místě programu s chybějící assertací pro celý běh programu:

i	r
1	10
2	20
3	30
4	40

Nyní si pojďme ukázat, jak bude tabulka vypadat pro obecné hodnoty x, y a $n \in \{1, ..., y-1\}$:

i	r
1	x
2	$2x$
3	$3x$
.	.
.	.
.	.
n	nx
.	.
.	.
.	.
$y-1$	$(y-1)x$

Cvičení 11b

Zadání

Pokuste se odhadnout chybějící assertaci uvnitř cyklu.

Řešení

Z předchozích tabulek se můžeme pokusit odhadnout, který výraz bychom mohli doplnit do assertace na začátku cyklu. Z tabulky lze vypozorovat následující vlastnosti:

- $r = ix$
- $i < y$

Pojďme tedy tyto dvě vlastnosti doplnit do assertace na začátku cyklu. Doplněním vznikne následující program:

$r \leftarrow x$
$i \leftarrow 1$
while $i < y$ **do**
    @ $r = ix \wedge i < y$
    $r \leftarrow r + x$
    $i \leftarrow i + 1$
@ $r = xy$
return r

Cvičení 11c

Zadání

Zapište odpovídající základní cesty programu a zkontrolujte, zda platí jejich ověřovací podmínky.

Řešení

Z programu v předchozí sekci lze vytvořit následující základní cesty programu.

Základní cesta 1

$r \leftarrow x$
$i \leftarrow 1$
assume $\neg(i < y)$
@ $r = xy$

SSA forma 1

V tomto případě není třeba zavádět nové názvy proměnných.

Logická formule 1

$$(\forall x, y, r, i \in \mathbb{N}_0, y \geq 1)([r = x \wedge i = 1 \wedge \neg(i < y)] \Rightarrow r = xy)$$

Ověřovací podmínka 1

Máme předpoklady $r = x$, $i = 1$, $\neg(i < y)$ a máme dokázat $r = xy$. Z předpokladů $\neg(i < y)$ a $i = 1$ můžeme vytvořit předpoklad $y \leq 1$. Jelikož však zadání definuje, že $y \geq 1$, může y nabývat pouze hodnoty $y = 1$. Použijeme tedy předpoklady $r = x$ a $y = 1$ a dosadíme je do dokazované části formule $r = xy$, čímž dostaneme $x = x$, což je triviálně dokázáno. Ověřovací podmínka pro základní cestu 1 tedy platí.

Základní cesta 2

$r \leftarrow x$
$i \leftarrow 1$
assume $i < y$
@ $r = ix \wedge i < y$

SSA forma 2

V tomto případě není třeba zavádět nové názvy proměnných.

Logická formule 2

$$(\forall x, y, r, i \in \mathbb{N}_0, y \geq 1)([r = x \wedge i = 1 \wedge i < y] \Rightarrow [r = ix \wedge i < y])$$

Ověřovací podmínka 2

Máme předpoklady $r = x$, $i = 1$, $i < y$ a máme dokázat $r = ix \wedge i < y$, tedy dokázat zvlášť $r = ix$ a $i < y$. Můžeme si všimnout, že $i < y$ je zároveň i předpoklad, tedy tuto část máme triviálně dokázanou. Nyní použijeme předpoklady $r = x$ a $i = 1$ a dosadíme je do dokazované části formule $r = ix$, čímž dostaneme $x = x$, což je triviálně dokázáno. Ověřovací podmínka pro základní cestu 2 tedy platí.

Základní cesta 3

assume $r = ix \wedge i < y$
$\quad r \leftarrow r + x$
$\quad i \leftarrow i + 1$
assume $i < y$
@ $r = ix \wedge i < y$

SSA forma 3

assume $r = ix \wedge i < y$
$\quad r_1 \leftarrow r + x$
$\quad i_1 \leftarrow i + 1$
assume $i_1 < y$
@ $r_1 = i_1 x \wedge i_1 < y$

Logická formule 3

$$(\forall x, y, r, i \in \mathbb{N}_0, y \geq 1)$$
$$([r = ix \wedge i < y \wedge r_1 = r + x \wedge i_1 = i + 1 \wedge i_1 < y] \Rightarrow [r_1 = i_1 x \wedge i_1 < y])$$

Ověřovací podmínka 3

Máme předpoklady $r = ix$, $i < y$, $r_1 = r + x$, $i_1 = i + 1$, $i_1 < y$ a máme dokázat $r_1 = i_1 x \wedge i_1 < y$, tedy dokázat zvlášť $r_1 = i_1 x$ a $i_1 < y$. Můžeme si všimnout, že $i_1 < y$ je zároveň i předpoklad, tedy tuto část máme triviálně dokázanou. Z předpokladů $r = ix$ a $r_1 = r+x$ můžeme vytvořit nový předpoklad $r_1 = (i + 1)x$. Do tohoto předpokladu můžeme dosadit předpoklad $i_1 = i + 1$, čímž vznikne předpoklad $r_1 = i_1 x$, což jsme zároveň měli dokázat. Ověřovací podmínka pro základní cestu 3 tedy platí.

Základní cesta 4

assume $r = ix \wedge i < y$
$\quad r \leftarrow r + x$
$\quad i \leftarrow i + 1$
assume $\neg(i < y)$
@ $r = xy$

SSA forma 4

assume $r = ix \wedge i < y$
$r_1 \leftarrow r + x$
$i_1 \leftarrow i + 1$
assume $\neg(i_1 < y)$
@ $r_1 = xy$

Logická formule 4

$$(\forall x, y, r, i \in \mathbb{N}_0, y \geq 1)$$
$$([r = ix \wedge i < y \wedge r_1 = r + x \wedge i_1 = i + 1 \wedge \neg(i_1 < y)] \Rightarrow r_1 = xy)$$

Ověřovací podmínka 4

Máme předpoklady $r = ix$, $i < y$, $r_1 = r + x$, $i_1 = i + 1$, $\neg(i_1 < y)$ a máme dokázat $r_1 = xy$. Z předpokladů $r = ix$ a $r_1 = r + x$ můžeme vytvořit nový předpoklad $r_1 = (i + 1)x$. Do tohoto předpokladu můžeme dosadit předpoklad $i_1 = i + 1$, čímž vznikne nový předpoklad $r_1 = i_1 x$. Dále můžeme získat z předpokladu $\neg(i_1 < y)$ předpoklad $y \leq i_1$, který můžeme následně spojit s předpokladem $i < y$, čímž získáme předpoklad $i < y \leq i_1$ z čehož zjistíme, že $y = i_1$. Tento nový předpoklad můžeme dosadit do předpokladu $r_1 = i_1 x$ a tím získat $r_1 = xy$, což jsme zároveň měli dokázat. Ověřovací podmínka pro základní cestu 4 tedy platí.

Cvičení 11d

Zadání

Pokud jsou všechny základní cesty programu korektní, stejně tak jako celý algoritmus, pak je úkol dokončen. V opačném případě upravte assertaci uvnitř cyklu a pokračujte předchozími dvěma úkoly, dokud nejsou všechny základní cesty korektní.

Řešení

V předchozí sekci si můžeme všimnout, že všechny základní cesty programu jsou korektní a tím tedy i celý algoritmus.

MI-FME Cvičení 12

Tomáš Chvosta

Duben 2020

Zadání

Uvažujte funkci s následujícím chováním:

function $s(a, k)$
Input: $a \in \mathcal{A}, k \in \mathcal{N}$ s.t. $k \geq 1$
Output: $1 + \sum_{i \in \{0,\ldots,k-1\}} a[i]$

Dokažte že je následující program zcela korektní (p je pole proměnných typu integer, ostatní proměnné jsou typu integer):

assume $\forall i \ . \ p[i] \geq 5$
$r \leftarrow s(p, 2)$
@ $r \geq 8$

Dodržujte metody pro zpracování volání funkcí, které jsou uvedené v přednáškách. Kromě toho také používejte dokazovací pravidla pro kvantifikátory. Můžete však libovolně využívat jakékoliv znalosti ohledně proměnných typů pole a integer.

Řešení

Nejprve si můžeme všimnout, že výstup ve specifikaci funkce nemá přiřazený žádný název, tedy žádnou výstupní proměnnou. Upravíme tedy specifikaci na následující tvar:

function $s(a, k)$
Input: $a \in \mathcal{A}, k \in \mathcal{N}$ s.t. $k \geq 1$
Output: r s.t. $r = 1 + \sum_{i \in \{0,\ldots,k-1\}} a[i]$

Upravená specifikace poté odpovídá následující logické formuli:

$$(\forall a \in \mathcal{A}, k, r \in \mathcal{N})((k \geq 1 \wedge r = s(a, k)) \Rightarrow (r = 1 + \sum_{i=0}^{k-1} a[i]))$$

Tuto formuli můžeme nyní brát jako předpoklad pro důkazy, ve kterých se bude vyskytovat naše funkce s. Pojďme si nyní převést do logické formule i náš program. Program je v SSA formě, takže rovnou získáváme logickou formuli:

$$(\forall i, r \in \mathcal{N}, p \in \mathcal{A})((p[i] \geq 5 \wedge r = s(p, 2)) \Rightarrow (r \geq 8))$$

Z předpokladu, který popisuje funkci s po volbě $a \leftarrow p,\ k \leftarrow 2$, víme:

$$(\forall i, r \in \mathcal{N}, p \in \mathcal{A})((p[i] \geq 5 \wedge r = 1 + \sum_{i=0}^{1} p[i]) \Rightarrow (r \geq 8))$$

Máme tedy předpoklady $p[i] \geq 5$ a $r = 1 + p[0] + p[1]$ a máme dokázat $r \geq 8$. Víme, že předpoklad $p[i] \geq 5$ platí pro všechna i, po volbách $i \leftarrow 0$ a $i \leftarrow 1$ získáváme nové předpoklady $p[0] \geq 5$ a $p[1] \geq 5$. Dále můžeme využít předpoklad $r = 1 + p[0] + p[1]$ a upravit dokazovaný výraz na tvar $1 + p[0] + p[1] \geq 8$ tedy $p[0] + p[1] \geq 7$. To můžeme dokázat například sporem. Předpokládáme $\neg(p[0] + p[1] \geq 7)$ tedy $p[0] + p[1] < 7$ a pokusíme se najít spor. Tento nový předpoklad můžeme upravit na tvar $p[0] - 7 < -p[1]$. Dále pak předpoklad $p[1] \geq 5$ upravíme a získáme předpoklad $-p[1] \leq -5$ a také upravíme předpoklad $p[0] \geq 5$ a získáme předpoklad $p[0] - 7 \geq 5 - 7$ tedy $p[0] - 7 \geq -2$. Pokud spojíme předpoklady $p[0] - 7 < -p[1]$, $-p[1] \leq -5$ a $p[0] - 7 \geq -2$ získáme $-2 \leq p[0] - 7 < -p[1] \leq -5$ tedy $-2 \leq -5$, čímž jsme došli ke sporu a naše formule platí. Program je tedy korektní.

MI-FME Cvičení 13

Tomáš Chvosta

Duben 2020

Zadání

Uvažujte proceduru s následujícím chováním:

procedure $p(a, r, x)$
Input: $a \in \mathcal{N}, r \in \mathcal{N}, x \in \mathcal{N}$ s.t. $r \geq 0$
Output: a^* s.t. $a^* = a + rx$

Dokažte že je následující fragment programu zcela korektní (všechny proměnné jsou typu integer):

assume $x \geq 10 \wedge k \geq 5$
$p(x, 2, k)$
@ $x \geq 15$

Dodržujte metody pro zpracování volání procedur, které jsou uvedené v přednáškách. Kromě toho také používejte dokazovací pravidla pro kvantifikátory. Můžete však libovolně využívat jakékoliv znalosti ohledně proměnných typů pole a integer.

Řešení

Nejprve vytvoříme logickou formuli pro proceduru p. Jelikož procedura mění a a potřebujeme odlišný název pro vstupní a výstupní proměnnou, uvedeme vstupní a výstupní hodnotu zvlášť. Zároveň v logické formuli změníme význam p na predikát:

$$(\forall a, a^*, r, x \in \mathcal{N})((r \geq 0 \wedge p(a, a^*, r, x)) \Rightarrow a^* = a + rx)$$

Tuto formuli můžeme nyní brát jako předpoklad pro důkazy, ve kterých se bude vyskytovat naše procedura p. Pojďme si nyní převést do logické formule i náš program. Abychom něco takového mohli udělat, budeme nejprve potřebovat SSA formu:

assume $x \geq 10 \wedge k \geq 5$
assume $p(x, x_1 2, k)$
@ $x_1 \geq 15$

Je potřeba brát ohled na to, že p v SSA formě představuje predikát. Logická formule z této SSA formy bude vypadat následovně:

$$(\forall x, x_1, k \in \mathcal{N})((x \geq 10 \wedge k \geq 5 \wedge p(x, x_1, 2, k)) \Rightarrow x_1 \geq 15)$$

Z předpokladu, který popisuje proceduru p pomocí logické formule po volbě $a \leftarrow x,\ a^* \leftarrow x_1,\ r \leftarrow 2,\ x \leftarrow k$, víme:

$$(\forall x, x_1, k \in \mathcal{N})((x \geq 10 \wedge k \geq 5 \wedge x_1 = x + 2k) \Rightarrow x_1 \geq 15)$$

Máme tedy tři předpolady $x \geq 10$, $k \geq 5$, $x_1 = x + 2k$ a máme dokázat $x_1 \geq 15$. Můžeme využít předpokladu $x_1 = x + 2k$ a upravit dokazovaný výraz na tvar $x + 2k \geq 15$. To můžeme snadno dokázat sporem. Předpokládáme tedy $\neg(x + 2k \geq 15)$, což je $x + 2k < 15$ a najdeme spor. Tento předpoklad můžeme upravit na tvar $2k < 15 - x$. Dále předpoklad $k \geq 5$ upravíme na tvar $2k \geq 10$. Složením předpokladů $2k < 15 - x$ a $2k \geq 10$ získáme $10 \leq 2k < 15 - x$ tedy $10 < 15 - x$, což můžeme upravit na tvar $x < 5$. To je však spor s předpokladem $x \geq 10$. Logická formule tedy platí a program je korektní.

MI-FME Cvičení 14

Tomáš Chvosta

Duben 2020

Zadání

Přidejte logické formule představující assertace na řádky programu začínající symbolem @ tak, aby platily všechny ověřovací podmínky. Není potřeba cokoliv dokazovat, neformální argumenty bohatě postačí. Neměňte však samotný program.

assume $r = 0 \wedge s = 0$
for $i \leftarrow 0$ **to** n **do**
    @
    if $a[i] < 0$ **then** $r \leftarrow r + 1$
    @
    if $a[i] > 0$ **then** $s \leftarrow s + 1$
    @
@ $r + s = |\{i \in \{0, \ldots, n\} \mid a[i] \neq 0\}|$

Řešení

Pojďme si nyní znázornit ukázkový běh programu. Nejprve si označíme program následujícím způsobem:

assume $r = 0 \wedge s = 0$
for $i \leftarrow 0$ **to** n **do**
    @ A1
    if $a[i] < 0$ **then** $r \leftarrow r + 1$
    @ A2
    if $a[i] > 0$ **then** $s \leftarrow s + 1$
    @ A3
@ $r + s = |\{i \in \{0, \ldots, n\} \mid a[i] \neq 0\}|$

Dále dosadíme například $a \leftarrow [4, -8, 0, 3]$, $n \leftarrow 3$. Běh programu je znázorněn v následující tabulce:

i	**Assertace**	r	s	$r+s$
0	A1	0	0	0
0	A2	0	0	0
0	A3	0	1	1
1	A1	0	1	1
1	A2	1	1	2
1	A3	1	1	2
2	A1	1	1	2
2	A2	1	1	2
2	A3	1	1	2
3	A1	1	1	2
3	A2	1	1	2
3	A3	1	2	3

Můžeme si všimnout toho, že proměnná r obsahuje aktuální počet prvků pole a v jednoltivých iteracích cyklu, kdy jednotlivé prvky jsou menší než 0. Naopak proměnná s obsahuje aktuální počet prvků pole a v jednoltivých iteracích cyklu, kdy jednotlivé prvky jsou větší než 0. V bodě A1 jsou však v proměnných r, s uloženy počty z předchozích iterací, tedy pro iteraci, kde $i = k$, jsou započítány prvky v poli a na pozicích 0 až $k - 1$. V bodě A2 je v proměnné r uložen počet z aktuální iterace, tedy pro iteraci, kde $i = k$, jsou započítány prvky v poli a na pozicích 0 až k, v proměnné s je uložen počet z předchozí iterace. V bodě A3 jsou zase v proměnných r, s uloženy počty z aktuální iterace.

To však nejsou jediné zákonitosti, které můžeme vypozorovat. Jelikož jsou assertace uvnitř cyklu, je třeba přidat do assertace podmínku $i \leq n$. Výsledný program s doplněnými assertacemi bude vypadat následovně:

assume $r = 0 \wedge s = 0$
for $i \leftarrow 0$ **to** n **do**
  @ $r = |\{k \in \{0, \ldots, i-1\} \mid a[k] < 0\}| \wedge$
    $s = |\{k \in \{0, \ldots, i-1\} \mid a[k] > 0\}| \wedge$
    $i \leq n$
  if $a[i] < 0$ **then** $r \leftarrow r + 1$
  @ $r = |\{k \in \{0, \ldots, i\} \mid a[k] < 0\}| \wedge$
    $s = |\{k \in \{0, \ldots, i-1\} \mid a[k] > 0\}| \wedge$
    $i \leq n$
  if $a[i] > 0$ **then** $s \leftarrow s + 1$
  @ $r = |\{k \in \{0, \ldots, i\} \mid a[k] < 0\}| \wedge$
    $s = |\{k \in \{0, \ldots, i\} \mid a[k] > 0\}| \wedge$
    $i \leq n$
@ $r + s = |\{i \in \{0, \ldots, n\} \mid a[i] \neq 0\}|$

Po vypsání všech základních cest programu a všech logických formulí vycházejících z těchto základních cest lze snadno zjistit, že všechny ověřovací podmínky platí. Dle zadání to však už není součástí tohoto úkolu.

MI-FME Cvičení 15

Tomáš Chvosta

Duben 2020

Zadání

Přidejte logické formule představující assertace na řádky programu začínající symbolem @ tak, aby platily všechny ověřovací podmínky. Není potřeba cokoliv dokazovat, neformální argumenty bohatě postačí. Pokud se stane, že najdete chybu v programu (předpokládám, že ji najdete), opravte ji.

$found \leftarrow \bot$
for $i \leftarrow 0$ **to** $n - k$ **do**
    @
    $p \leftarrow \top$
    for $j \leftarrow 0$ **to** $k - 1$ **do**
        @
        if $a[i + j] \neq s[j]$ **then**
            $p \leftarrow \bot$
        @
    if p **then** $found \leftarrow \top$
    @
@ $found \Leftrightarrow \exists r \ . \ substr(a, n, s, k, r)$

Je použitá následující definice:

$$(\forall a \in \mathcal{A}, n \in \mathcal{N}, s \in \mathcal{A}, k \in \mathcal{N}, p \in \mathcal{N})$$
$$(substr(a, n, s, k, p) :\Leftrightarrow (\forall i \in \{0, \ldots, k - 1\})(p + i < n \wedge a[p + i] = s[i]))$$

Řešení

Můžeme si všimnout, že program obsahuje dva for cykly a assertace, které máme za úkol doplnit, jsou vždy na začátku a na konci těchto cyklů. Vnější for cyklus se snaží najít index i v poli a, který značí výskyt pole s. Pokud ho najde změní hodnotu proměnné $found$ na $\top$. Vnitřní cyklus kontroluje, zda se na daném indexu i v poli a všechny prvky z pole s skutečně vyskytují. Pokud ne, uloží do proměnné p hodnotu $\bot$.

V assertaci na začátku vnitřního cyklu by tedy mělo platit, že p platí, právě když pro všechny doposud zkontrolované j se prvky polí shodovali. K tomu můžeme využít definovaný predikát *substr*. To samé bude i na konci cyklu, akorát zde si musíme uvědomit, že jsme zkontrolovali o jednu hodnotu j navíc.

V assertaci na začátku vnějšího cyklu by zase mělo platit, že $found$ platí, právě když mezi doposud projitými indexi i existuje takový, který představuje počáteční index v poli a, kde se vyskytuje s. Opět můžeme využít definici predikátu *substr*.

Nesmíme opomenout ještě jednu věc. Do assertací bychom měli doplnit podmínky, že $i \leq n - k$ a také $j \leq k - 1$, abychom měli zaručeno, že ve všech základních cestách programu máme v proměnných korektní hodnoty. Program s doplněnými assertacemi bude vypadat následovně:

$found \leftarrow \bot$
for $i \leftarrow 0$ **to** $n - k$ **do**
  @ $(found \Leftrightarrow ((\exists r \in \{0, \ldots, i-1\})(substr(a, n, s, k, r)))) \wedge i \leq n - k$
  $p \leftarrow \top$
  for $j \leftarrow 0$ **to** $k - 1$ **do**
    @ $(p \Leftrightarrow substr(a, n, s, j, i)) \wedge i \leq n - k \wedge j \leq k - 1$
    if $a[i + j] \neq s[j]$ **then**
      $p \leftarrow \bot$
    @ $(p \Leftrightarrow substr(a, n, s, j + 1, i)) \wedge i \leq n - k \wedge j \leq k - 1$
  if p **then** $found \leftarrow \top$
  @ $(found \Leftrightarrow ((\exists r \in \{0, \ldots, i\})(substr(a, n, s, k, r)))) \wedge i \leq n - k$
@ $found \Leftrightarrow \exists r \,.\, substr(a, n, s, k, r)$

Po podrobnějším zkoumání programu si můžeme všimnout, že program nefunguje korektně v případě, že $n = 0$ a zároveň $k = 0$. Proměnná $found$ je triviálně $\top$, nicméně neexistuje r takové, že by splňovalo $substr(a, n, s, k, r)$. To však dokážeme jednoduše opravit upravením podmínky v assertaci na konci programu. Program pak bude vypadat následovně:

$found \leftarrow \bot$
for $i \leftarrow 0$ **to** $n - k$ **do**
  @ $(found \Leftrightarrow ((\exists r \in \{0, \ldots, i-1\})(substr(a, n, s, k, r)))) \wedge i \leq n - k$
  $p \leftarrow \top$
  for $j \leftarrow 0$ **to** $k - 1$ **do**
    @ $(p \Leftrightarrow substr(a, n, s, j, i)) \wedge i \leq n - k \wedge j \leq k - 1$
    if $a[i + j] \neq s[j]$ **then**
      $p \leftarrow \bot$
    @ $(p \Leftrightarrow substr(a, n, s, j + 1, i)) \wedge i \leq n - k \wedge j \leq k - 1$
  if p **then** $found \leftarrow \top$
  @ $(found \Leftrightarrow ((\exists r \in \{0, \ldots, i\})(substr(a, n, s, k, r)))) \wedge i \leq n - k$
@ $(found \Leftrightarrow \exists r \,.\, substr(a, n, s, k, r)) \vee ((n = 0) \wedge (k = 0))$

Po vypsání všech základních cest programu a všech logických formulí vycházejí-

cích z těchto základních cest lze snadno zjistit, že všechny ověřovací podmínky platí. Dle zadání to však už není součástí tohoto úkolu.

MI-FME Cvičení 16

Tomáš Chvosta

Květen 2020

Zadání

1: $x \leftarrow x + 1$
2: **input** y
3: **if** $x > 7$ **then**
4: $\quad y \leftarrow xy + z$
5: $\quad$ **if** $y > 3$ **then**
6: $\quad\quad x \leftarrow x + 1$
7: $\quad$ **else**
8: $\quad\quad x \leftarrow x - 1$

Napište logické formule pro symbolické provedení řádků programu 1, 2, 3, 4, 5, 6 a poté řádků programu 1, 2, 3, 4, 7, 8. Použijte zjednodušenou verzi této formule, která je ve tvaru $X_{l_1,\ldots,l_n}(P) \equiv F_{pre}(SSA(BP_{l_1\ldots,l_n}(P)))$. V této úloze není potřeba psát mezikroky (tj. ani $BP_{l_1\ldots,l_n}(P)$, ani SSA formu). Také není třeba řešit výsledek formule.

Řešení

V této úloze budeme používat značení **program A** pro řádky programu 1, 2, 3, 4, 5, 6 a **program B** pro řádky programu 1, 2, 3, 4, 7, 8. Přestože je v zadání uvedeno, že není potřeba psát $BP_{l_1\ldots,l_n}(P)$, ani SSA formu, bude lepší a přehlednější si tyto kroky uvést.

Základní cesty programu A

$x \leftarrow x + 1$
input y
assume $x > 7$
$y \leftarrow xy + z$
assume $y > 3$
$x \leftarrow x + 1$
@ $\bot$

SSA forma programu A

$x_2 \leftarrow x_1 + 1$
input y_1
assume $x_2 > 7$
$y_2 \leftarrow x_2 y_1 + z_1$
assume $y_2 > 3$
$x_3 \leftarrow x_2 + 1$
@ $\bot$

Výsledná logická formule pro program A

$$[x_2 = x_1 + 1 \wedge x_2 > 7 \wedge y_2 = x_2 y_1 + z_1 \wedge y_2 > 3 \wedge x_3 = x_2 + 1] \Rightarrow \bot$$

Základní cesty programu B

$x \leftarrow x + 1$
input y
assume $x > 7$
$y \leftarrow xy + z$
assume $\neg(y > 3)$
$x \leftarrow x - 1$
@ $\bot$

SSA forma programu B

$x_2 \leftarrow x_1 + 1$
input y_1
assume $x_2 > 7$
$y_2 \leftarrow x_2 y_1 + z_1$
assume $\neg(y_2 > 3)$
$x_3 \leftarrow x_2 - 1$
@ $\bot$

Výsledná logická formule pro program B

$$[x_2 = x_1 + 1 \wedge x_2 > 7 \wedge y_2 = x_2 y_1 + z_1 \wedge y_2 \leq 3 \wedge x_3 = x_2 - 1] \Rightarrow \bot$$

MI-FME Cvičení 17

Tomáš Chvosta

Květen 2020

Zadání

```
1:  x ← x + 1
2:  input y
3:  if x > 7 then
4:      y ← ext(y)
5:      if y > 3 then
6:          x ← x + 1
7:      else
8:          x ← x − 1
```

Demonstrujte, jak metoda „concolic testing“ zkouší nalézt testovací případ pro provedení řádků programu 1, 2, 3, 4, 7, 8. K provedení externí funkce využijte přiložený program `rand.c` v jazyce C (jednoduše zkompilujte, spusťte, zadejte číslo a použijte výsledek z výpisu). V této úloze je třeba vyřešit logické formule (tedy najít takové ohodnocení formule, pro které je splnitelná). Řešte však pouze ty formule potřebné k nalezení vstupu a volání externí funkce. Neřešte žádné jiné formule.

Řešení

Metoda „concolic testing“ postupuje podle následujících kroků:

- Vytvoříme logickou formuli $X_{l_1,\ldots,l_n}(P)$ avšak nikoliv najednou ale inkrementálně.
- Jakmile během vytvoření formule narazíme na funkci, kterou nedokážeme vyřešit, použijeme program k obdržení výsledku funkce.
- Pokud potřebuje funkce k provedení nějaké vstupy, vypočítáme je vyřešením symbolických formulí (té části, která odpovídá programu před zavolání externí funkce)
- Použijeme výsledek konkrétního provedení části programu a nahradíme volání externí funkci odpovídající získanou hodnotou.

Postupujeme tedy podle výše zmíněných bodů. Nejprve vytváříme inkrementálně logickou formuli $X_{l_1,\ldots,l_n}(P)$ do chvíle, než narazíme na funkci, kterou nedokážeme vyřešit. Získáváme:

$$[x_2 = x_1 + 1 \wedge x_2 > 7]$$

Na dalším řádku se nachází volání externí funkce $ext(y)$, pro které potřebujeme nějaký vstup y. Zvolme tedy například $y = 4$, pro které je $f(y) = -5$. Nyní se můžeme vrátit zpět k prvnímu kroku a dál tvořit inkrementálně logickou formuli, čímž získáme:

$$[x_2 = x_1 + 1 \wedge x_2 > 7 \wedge y_1 = 4 \wedge y_2 = -5 \wedge y_2 \leq 3 \wedge x_3 = x_2 - 1] \Rightarrow \bot$$

.

Poznámka

I přesto, že to zadání nevyžaduje, si pojďme ukázat, jak je to se splnitelností levé strany výsledné formule. V tomto případě můžeme snadno najít ohodnocení proměnných, pro které je levá strana formule splněna (například $x_1 \leftarrow 7$, $x_2 \leftarrow 8$, $x_3 \leftarrow 7$, $y_1 \leftarrow 4$, $y_2 \leftarrow -5$), a tedy nenacházíme žádnou chybu v programu.

Pokud bychom však při konstrukci logické formule metodou „concolic testing“ zvolili například $y = 2$, pro které je $f(y) = 5$, získali bychom logickou formuli, která by obsahovala $y_2 = 5 \wedge y_2 \leq 3$, což by vedlo k závěru, že levá strana výsledné formule není splnitelná a program tedy nutně obsahuje chybu. Plyne z toho, že je velmi důležité zvolit správně počáteční a vstupní hodnoty potřebné k zavolání externích funkcí.