Komparatívna štúdia metód na syntézu programov bez rekurzie

Richard Žilinčík

Školiteľka: Mgr. Petra Hozzová, PhD.

25. júna 2024

Čo je to syntéza programov?

- automatizovaný proces
- vstup: špecifikácia programu
- výstup: program ktorý dokázateľne spĺňa špecifikáciu

Špecifikácia programu - logická formula

- zadaná v logickom jazyku - napr. logika prvého rádu
- predpoklady o vstupe pre program
- požiadavky na výstup
- špecifikácia je deklaratívna, výstup je algoritmický

$$\forall x_1, x_2.\ f(x_1, x_2) \geq x_1 \wedge f(x_1, x_2) \geq x_2 \wedge (f(x_1, x_2) = x_1 \vee f(x_1, x_2) = x_2)$$

$$\texttt{if } x_1 < x_2 \texttt{ then } x_2 \texttt{ else } x_1$$

Výstupné programy

- bez rekurzie a cyklov
- ľubovoľne vnorené if-then-else
- využívajúce preddefinované funkcie

Prečo syntéza programov

- napísať deklaratívnu špecifikáciu je jednoduchšie
- šetrí sa čas
- menšia pravdepodobnosť omylu

Prečo porovnávať metódy

- majú odlišné silné a slabé stránky
- používateľ môže urobiť informované rozhodnutie pre konkrétny prípad
- výskumníci získajú lepší prehľad kde konkrétna metóda zaostáva, a kde exceluje
- cieľ práce: porovnať silu špecifikácií a nájsť konkrétne príklady ktoré metódy teoreticky a prakticky dokážu vyriešiť

Porovnávané metódy

- Manna, Z., and Waldinger, R. “A deductive approach to program synthesis”, 1980 - dedukčná metóda
- Reynolds, A., et al.“Refutation-based synthesis in SMT.”, 2019 - metóda na báze SMT
- Hozzová, P., et al.“Program synthesis in saturation”, 2023 - metóda na báze saturácie

Kritéria porovnávania

- sila špecifikačných jazykov
  - vyjadriteľnosť tried špecifikácií
  - efektívnosť kódovania špecifikácií
- porovnanie na spoločnom príklade
- porovnanie silných a slabých stránok vyplývajúcich zo stratégie syntézy
- konkrétne testy na implementáciách

Vypracovaný príklad

	assertions	goals	outputs	
1		$y \geq \sigma_1 \wedge y \geq \sigma_2 \wedge (y = \sigma_1 \vee y = \sigma_2)$	y	
2		$y \geq \sigma_1 \wedge y \geq \sigma_2 \wedge y = \sigma_1$	y	
3		$y \geq \sigma_1 \wedge y \geq \sigma_2 \wedge y = \sigma_2$	y	
4	$x \geq x$			
5		$\neg false \wedge true \wedge \sigma_1 \geq \sigma_2 \wedge \sigma_1 = \sigma_1$	σ_1	GA-resolution 4, 2
6		$\neg false \wedge true \wedge \sigma_2 \geq \sigma_1 \wedge \sigma_2 = \sigma_2$	σ_2	GA-resolution 4, 3
7		$\sigma_1 \geq \sigma_2$	σ_1	logic rules 5
8		$\sigma_2 \geq \sigma_1$	σ_2	logic rules 6
9		$\sigma_2 = \sigma_1 \vee \neg\sigma_1 \geq \sigma_2$	σ_2	logic rules 8
10		$\neg\sigma_1 \geq \sigma_2$	σ_2	orsplit 9
11		$true$	`if` $\sigma_1 \geq \sigma_2$	GG-resolution 7, 10
			`then` σ_1 `else` σ_2	

Vypracovaný príklad

- skolemizujeme: $e \geq a_1 \wedge e \geq a_2 \wedge (e = a_1 \vee e = a_2)$
- hľadáme model
- nájdeme napr. model, kde $e^{\mathcal{I}} = {a_1}^{\mathcal{I}}$
- zapamätáme si, a ďalej hľadáme model kde
  $\neg(a_1 \geq a_1 \wedge a_1 \geq a_2 \wedge (a_1 = a_1 \vee a_1 = a_2))$,
  čo vieme zjednodušiť na $\neg a_1 \geq a_2$
- nájdeme model, kde $e^{\mathcal{I}} = {a_2}^{\mathcal{I}}$
- ďalej by sme potrebovali model, kde $\neg a_1 \geq a_2 \wedge \neg a_2 \geq a_1$, čo nie je možné
- z podmienok vyskladáme program `if` $\neg x_1 \geq x_2$ `then` x_2 `else` x_1

Vypracovaný príklad

(a)	$y < \sigma_1 \vee y < \sigma_2 \vee y \neq \sigma_1 \vee \mathtt{ans}(y)$	[input]
(b)	$y < \sigma_1 \vee y < \sigma_2 \vee y \neq \sigma_2 \vee \mathtt{ans}(y)$	[input]
(c)	$\neg x < x$	[$<$ axiom]
(d)	$\neg x_1 < x_2 \vee \neg x_2 < x_1$	[$<$ axiom]
(e)	$\sigma_1 < \sigma_1 \vee \sigma_1 < \sigma_2 \vee \mathtt{ans}(\sigma_1)$	[ER (a)]
(f)	$\sigma_2 < \sigma_1 \vee \sigma_2 < \sigma_2 \vee \mathtt{ans}(\sigma_2)$	[ER (b)]
(g)	$\sigma_1 < \sigma_2 \vee \mathtt{ans}(\sigma_2)$	[BR (c), (e)]
(h)	$\sigma_2 < \sigma_1 \vee \mathtt{ans}(\sigma_1)$	[BR (c), (f)]
(i)	$\sigma_1 < \sigma_2$	[answer literal removal (g)]
(j)	$\sigma_2 < \sigma_1$	[answer literal removal (h)]
(k)	$\neg\sigma_2 < \sigma_1$	[BR (d), (i)]
(l)	$\square$	[BR (j), (k)]

Špecifikačné jazyky - syntaktické obmedzenia

```
(synth-fun f ((x Int) (y Int)) Int
    ((I Int) (V Int))
    ((I Int ((+ I I) V))
    (V Int (x y))))
```

- môžu napomôcť syntéze (zúžiť priestor možných riešení), alebo proces skomplikovať (vylúčiť jednoduché riešenia)
- metóda na báze SMT solverov je najexpresívnejšia
- deduktívna metóda na syntaktické obmedzenia vôbec nemá prostriedky

Špecifikačné jazyky - sémantické obmedzenia

- deduktívna metóda a metóda na báze saturácie podporujú iba vlastnosti s jedným volaním funkcie

$$\forall x_1, x_2 . \exists y.\ y \geq x_1 \wedge y \geq x_2 \wedge (y = x_1 \vee y = x_2)$$

- metóda na báze SMT solverov podporuje v špecifikácií logiku vyšších rádov
- metóda na báze SMT solverov priamo nepodporuje použitie neinterpretovaných funkcií
  - ľavý inverzný prvok $\forall x.\ x \cdot x^{-1} = e$
  - ľavá identita $\forall x.\ e \cdot x = x$
  - asociativita $\forall x, y.\ x \cdot y = y \cdot x$
- našli sme kódovanie špecifikácií s neinterpretovanými funkciami pomocou funkcií vyššieho rádu

Silné a slabé stránky

- deduktívna metóda je neúplná, na automatizáciu chýbajú kľúčové časti
- metóda na báze SMT solverov exceluje v zadaniach s množstvom aritmetiky a iných operácií z konkrétnych teórií
- metóda na báze saturácie si vie lepšie poradiť s kvantifikátormi

Praktická skúška implementácií

- 1 príklad prevzatý, 5 nových príkladov
- navrhnuté tak, aby pokryli triedy príkladov, kde sme očakávali rozdiely
- potvrdili sa očakávania podľa silných a slabých stránok metód

Name	Vampire	cvc5
Square of Sum	✓	✗(✓with restrictions)
Absolute Value	✓	✓
Same Quotient, Different Remainder	✗	✓
Invert Bitvector Addition	✗	✓
Field Theory	✓	✗
Quotient 1	✗	✗

Praktická skúška implementácií

Absolútna hodnota

$$\exists f.\forall x.\ f(x)^2 = x^2 \wedge f(x) \geq 0$$

$$f(x) = \texttt{if } x > 0 \texttt{ then } x \texttt{ else } -x.$$

Teória polí

$$\forall x_1, x_2.\ (-x_1) \cdot x_2 = -f(x_1, x_2)$$

$$\forall x_1, x_2.\ (-x_1) \cdot (-x_2) = g(x_1, x_2)$$

$$f(x_1, x_2) = g(x_1, x_2) = x_1 \cdot x_2$$

Quotient 1 - na pohľad jednoduchý príklad, nezvládla ani jedna implementácia

$$\exists f.\forall x.\ x \neq 0 \Rightarrow \mathrm{div}(f(x), x) = 1$$

$$f(x) = x$$

Možnosti ďalšej práce

- Porovnanie viacerých metód a špecifikačných jazykov
- Rozšírenie na programy s rekurziou
- Automatická konverzia medzi špecifikačnými jazykmi
- Porovnávanie syntetizovaných programov na báze dĺžky alebo komplexity

Computable unification

Algorithm 2 Computable Unification with Abstraction

`function` $\mathtt{mgu}_{\mathtt{comp}}(E_1, E_2, E_3)$
  `if` E_3 is uncomputable `then` fail
  let $\mathcal{E}$ be a set of equations and θ be a substitution; $\mathcal{E} := \{E_1 = E_2\}$; $\theta := \{\}$
  let $\mathcal{D}$ be a set of disequalities; $\mathcal{D} := \emptyset$
  `repeat`
    `if` $\mathcal{E}$ is empty `then`
      `return` (θ, D) where D is the disjunction of literals in $\mathcal{D}$
    Select an equation $s = t$ in $\mathcal{E}$ and remove it from $\mathcal{E}$
    `if` s coincides with t `then` do nothing
    `else if` s is a variable and s does not occur in t `then`
      `if` s does not occur in E_3 or t is computable `then` $\theta := \theta \circ \{s \mapsto t\}; \mathcal{E} = \mathcal{E}\{s \mapsto t\}$
      `else if` $t = f(t_1, \ldots, t_n)$ and f is computable `then`
        $\theta := \theta \circ \{s \mapsto f(x_1, \ldots, x_n)\}$; $\mathcal{E} := \mathcal{E}\{s \mapsto f(x_1, \ldots, x_n)\} \cup \{x_1 = t_1, \ldots, x_n = t_n\}$
          where $x_1, \ldots, x_n$ are fresh variables
      `else if` $t = f(t_1, \ldots, t_n)$ and f is uncomputable `then` $\mathcal{D} := \mathcal{D} \cup \{s \not\approx t\}$
    `else if` s is a variable and s occurs in t `then` fail
    `else if` t is a variable `then` $\mathcal{E} := \mathcal{E} \cup \{t = s\}$
    `else if` s and t have different top-level symbols `then` fail
    `else if` $s = f(s_1, \ldots, s_n)$ and $t = f(t_1, \ldots, t_n)$ `then` $\mathcal{E} := \mathcal{E} \cup \{s_1 = t_1, \ldots, s_n = t_n\}$