A Generic Framework for Symbolic Execution

Jan Tušil

8. prosince 2017

MojeIntro

```
int x,y;
x = get();
y = -x;
y = -y;
assert(x == y);
```

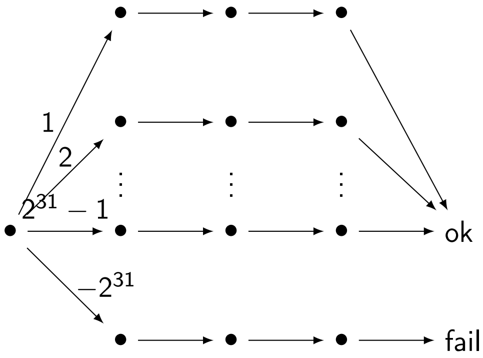
Může assert selhat?

Operační sémantika

$$OpSem : Program \to TransitionSystem$$

Operační sémantika

$$OpSem : Program \to TransitionSystem$$



Konfigurace

$$\langle x = get(); \curvearrowright y = -x; \curvearrowright y = -y; \curvearrowright assert(x == y); \rangle_{\mathsf{k}} \langle x = 0, y = 0 \rangle_{\mathsf{env}}$$

Konfigurace

$$\langle x = get(); \curvearrowright y = -x; \curvearrowright y = -y; \curvearrowright assert(x == y); \rangle_{\mathsf{k}} \langle x = 0, y = 0 \rangle_{\mathsf{env}}$$

Ukázka

Symbolická Konfigurace

$$\langle y = -x; \curvearrowright y = -y; \curvearrowright \mathit{assert}(x == y); \rangle_{\mathsf{k}} \langle x = X, y = 0 \rangle_{\mathsf{env}}$$

Symbolická Konfigurace

$$\langle y = -x; \curvearrowright y = -y; \curvearrowright \mathit{assert}(x == y);\rangle_{\,\mathsf{k}} \langle x = X, y = 0\rangle_{\,\mathsf{env}}$$

Ukázka

Symbolická exekuce

Princip (Pokrytí)

Každému (potenciálně nekonečnému) konkrétnímu běhu odpovídá nějaký symbolický běh.

Symbolická exekuce

Princip (Pokrytí)

Každému (potenciálně nekonečnému) konkrétnímu běhu odpovídá nějaký symbolický běh.

Princip (Přesnost)

Každému konečnému symbolickému běhu odpovídá nějaký konkrétní běh.

Symbolická exekuce

Princip (Pokrytí)

Každému (potenciálně nekonečnému) konkrétnímu běhu odpovídá nějaký symbolický běh.

Princip (Přesnost)

Každému konečnému symbolickému běhu odpovídá nějaký konkrétní běh.

Nekonečné běhy - koindukce

Signatura

Poznámka (Sortování)

Říkáme-li o množině X*, že je* Y*-sortovaná, máme tím na mysli, že existuje funkce* $\mathrm{SortOf} : X \to Y$.

Signatura

Poznámka (Sortování)

Říkáme-li o množině X, že je Y-sortovaná, máme tím na mysli, že existuje funkce $\mathrm{SortOf} : X \to Y$.

Definice

Vícedruhová algebraická signatura je tvořena množinou sortů S spolu s $S^* \times S$-sortovanou množinou Σ funkčních symbolů. Symbol T_Σ označuje Σ-algebru uzavřených termů, $T_{\Sigma,s}$ množinu termů sortu s, $T_\Sigma(Var)$ volnou Σ-algebru termů s proměnnými.

Signatura

Poznámka (Sortování)

Říkáme-li o množině X, že je Y-sortovaná, máme tím na mysli, že existuje funkce $\mathrm{SortOf} : X \to Y$.

Definice

Vícedruhová algebraická signatura je tvořena množinou sortů S spolu s $S^* \times S$-sortovanou množinou Σ funkčních symbolů. Symbol T_Σ označuje Σ-algebru uzavřených termů, $T_{\Sigma,s}$ množinu termů sortu s, $T_\Sigma(Var)$ volnou Σ-algebru termů s proměnnými.

```
Plant  ::=  favouriteFood(Animal)
Animal ::=  mother(Animal) | father(Animal)
```

Signatura

Poznámka (Sortování)

Říkáme-li o množině X, že je Y-sortovaná, máme tím na mysli, že existuje funkce $\mathrm{SortOf} : X \to Y$.

Definice

Vícedruhová algebraická signatura je tvořena množinou sortů S spolu s $S^* \times S$-sortovanou množinou Σ funkčních symbolů. Symbol T_Σ označuje Σ-algebru uzavřených termů, $T_{\Sigma,s}$ množinu termů sortu s, $T_\Sigma(Var)$ volnou Σ-algebru termů s proměnnými.

```
Plant  ::= favouriteFood(Animal)
Animal ::= mother(Animal) | father(Animal)
```

Příklad.

Vícedruhová FOL

Definice

Signatura provořádové logiky (Σ, Π) *je tvořena algebraickou signaturou* Σ *a* S^**-sortovanou množinou predikátových symbolů* Π.

Vícedruhová FOL

Definice

Signatura provořádové logiky (Σ, Π) *je tvořena algebraickou signaturou* Σ *a* S^**-sortovanou množinou predikátových symbolů* Π*.*

Definice (Formule)

Množina formulí nad signaturou (Σ, Π) *je definována:*

$$\phi ::= \top \mid p(t_1, \ldots, t_n) \mid \neg\phi \mid \phi \wedge \phi \mid (\exists X)\, \phi$$

kde p *označuje predikátové symboly,* X *podmnožiny proměnných a* t_i Σ*-termy s volnými proměnnými.*

Vícedruhová FOL

Definice

Signatura provořádové logiky (Σ, Π) *je tvořena algebraickou signaturou* Σ *a* S^**-sortovanou množinou predikátových symbolů* Π*.*

Definice (Formule)

Množina formulí nad signaturou (Σ, Π) *je definována:*

$$\phi ::= \top \mid p(t_1, \ldots, t_n) \mid \neg\phi \mid \phi \wedge \phi \mid (\exists X)\, \phi$$

kde p *označuje predikátové symboly,* X *podmnožiny proměnných a* t_i Σ*-termy s volnými proměnnými.*

Příklad (StaršíNež).

Vícedruhová FOL

Definice

Signatura provořádové logiky (Σ, Π) *je tvořena algebraickou signaturou* Σ *a* S^**-sortovanou množinou predikátových symbolů* Π.

Definice (Formule)

Množina formulí nad signaturou (Σ, Π) *je definována:*

$$\phi ::= \top \mid p(t_1, \ldots, t_n) \mid \neg\phi \mid \phi \wedge \phi \mid (\exists X)\,\phi$$

kde p *označuje predikátové symboly,* X *podmnožiny proměnných a* t_i Σ*-termy s volnými proměnnými.*

Příklad (StaršíNež). Předpokládáme predikát rovnosti.

Modely

Analogie s realizacemi jazyků v MA007.

Definice

Model signatury (Σ, Π) *je* Σ*-algebra* $\mathcal{T}$ *spolu s realizací* $\mathcal{T}_p \subseteq \mathcal{T}_{s_1} \times \cdots \times \mathcal{T}_{s_n}$ *pro každý predikátový symbol* $p \in \Pi_{s_1 \dots s_n}$.

Modely

Analogie s realizacemi jazyků v MA007.

Definice

Model signatury (Σ, Π)*je* Σ*-algebra* $\mathcal{T}$ *spolu s realizací* $\mathcal{T}_p \subseteq \mathcal{T}_{s_1} \times \cdots \times \mathcal{T}_{s_n}$ *pro každý predikátový symbol* $p \in \Pi_{s_1 \ldots s_n}$.

Zvířecí příklad.

Modely

Analogie s realizacemi jazyků v MA007.

Definice

Model signatury (Σ, Π)*je* Σ*-algebra* $\mathcal{T}$ *spolu s realizací* $\mathcal{T}_p \subseteq \mathcal{T}_{s_1} \times \cdots \times \mathcal{T}_{s_n}$ *pro každý predikátový symbol* $p \in \Pi_{s_1 \ldots s_n}$.

Zvířecí příklad.

Definice (Relace splnitelnosti)

Pro (Σ, Π)*-formuli* ϕ, (Σ, Π)*-model* $\mathcal{T}$ *a valuaci* $\rho : Var \to \mathcal{T}$ *definujeme relaci splnitelnosti* $\rho \models \phi$ *jako obvykle. (Valuaci* ρ *přirozeně rozšiřujeme na morfismus* Σ*-algeber* $\rho : T_\Sigma(Var) \to \mathcal{T}$*.)*

Modely

Analogie s realizacemi jazyků v MA007.

Definice

Model signatury (Σ, Π)*je* Σ*-algebra* $\mathcal{T}$ *spolu s realizací* $\mathcal{T}_p \subseteq \mathcal{T}_{s_1} \times \cdots \times \mathcal{T}_{s_n}$ *pro každý predikátový symbol* $p \in \Pi_{s_1 \ldots s_n}$.

Zvířecí příklad.

Definice (Relace splnitelnosti)

Pro (Σ, Π)*-formuli* ϕ, (Σ, Π)*-model* $\mathcal{T}$ *a valuaci* $\rho : Var \to \mathcal{T}$ *definujeme relaci splnitelnosti* $\rho \models \phi$ *jako obvykle. (Valuaci* ρ *přirozeně rozšiřujeme na morfismus* Σ*-algeber* $\rho : T_\Sigma(Var) \to \mathcal{T}$*.)*

Zkusme to na tabuli.

Modely

Analogie s realizacemi jazyků v MA007.

Definice

Model signatury (Σ, Π)je Σ-algebra $\mathcal{T}$ spolu s realizací $\mathcal{T}_p \subseteq \mathcal{T}_{s_1} \times \cdots \times \mathcal{T}_{s_n}$ pro každý predikátový symbol $p \in \Pi_{s_1 \dots s_n}$.

Zvířecí příklad.

Definice (Relace splnitelnosti)

Pro (Σ, Π)-formuli ϕ, (Σ, Π)-model $\mathcal{T}$ a valuaci $\rho : Var \to \mathcal{T}$ definujeme relaci splnitelnosti $\rho \models \phi$ jako obvykle. (Valuaci ρ přirozeně rozšiřujeme na morfismus Σ-algeber $\rho : T_\Sigma(Var) \to \mathcal{T}$.)

Zkusme to na tabuli.
Příklad.

Modely

Analogie s realizacemi jazyků v MA007.

Definice

Model signatury (Σ, Π)*je* Σ*-algebra* $\mathcal{T}$ *spolu s realizací* $\mathcal{T}_p \subseteq \mathcal{T}_{s_1} \times \cdots \times \mathcal{T}_{s_n}$ *pro každý predikátový symbol* $p \in \Pi_{s_1 \ldots s_n}$.

Zvířecí příklad.

Definice (Relace splnitelnosti)

Pro (Σ, Π)*-formuli* ϕ, (Σ, Π)*-model* $\mathcal{T}$ *a valuaci* $\rho : Var \to \mathcal{T}$ *definujeme relaci splnitelnosti* $\rho \models \phi$ *jako obvykle. (Valuaci* ρ *přirozeně rozšiřujeme na morfismus* Σ*-algeber* $\rho : T_\Sigma(Var) \to \mathcal{T}$*.)*

Zkusme to na tabuli.
Příklad.
Formule ϕ je validní (v $\mathcal{T}$), když je splněná všemi valuacemi. Označujeme: $\models \phi$.

Co bylo na začátku?

Co bylo na začátku? Program, sémantika, přechodový systém, konfigurace.

Co bylo na začátku? Program, sémantika, přechodový systém, konfigurace.

$$\langle \langle x = 5; \curvearrowright y = -x; \rangle_{\mathsf{k}} \langle x \mapsto 0, y \mapsto 0 \rangle_{\mathsf{env}} \rangle_{\mathsf{cfg}}$$

Co bylo na začátku? Program, sémantika, přechodový systém, konfigurace.

$$\langle \langle x = 5; \curvearrowright y = -x; \rangle_{\mathsf{k}} \langle x \mapsto 0, y \mapsto 0 \rangle_{\mathsf{env}} \rangle_{\mathsf{cfg}}$$

Tohle je nějaký term.

Co bylo na začátku? Program, sémantika, přechodový systém, konfigurace.

$$\langle \langle x = 5; \curvearrowright y = -x; \rangle_{\mathsf{k}} \langle x \mapsto 0, y \mapsto 0 \rangle_{\mathsf{env}} \rangle_{\mathsf{cfg}}$$

Tohle je nějaký term.

$$\langle \langle X = I; \curvearrowright y = -x; \rangle_{\mathsf{k}} \langle X \mapsto 0, y \mapsto 0 \rangle_{\mathsf{env}} \rangle_{\mathsf{cfg}}$$

Co bylo na začátku? Program, sémantika, přechodový systém, konfigurace.

$$\langle \langle x = 5; \curvearrowright y = -x; \rangle_{\mathsf{k}} \langle x \mapsto 0, y \mapsto 0 \rangle_{\mathsf{env}} \rangle_{\mathsf{cfg}}$$

Tohle je nějaký term.

$$\langle \langle X = I; \curvearrowright y = -x; \rangle_{\mathsf{k}} \langle X \mapsto 0, y \mapsto 0 \rangle_{\mathsf{env}} \rangle_{\mathsf{cfg}}$$

Term s volnými proměnnými (X, I).

Co bylo na začátku? Program, sémantika, přechodový systém, konfigurace.

$$\langle \langle x = 5; \curvearrowright y = -x; \rangle_{\mathsf{k}} \langle x \mapsto 0, y \mapsto 0 \rangle_{\mathsf{env}} \rangle_{\mathsf{cfg}}$$

Tohle je nějaký term.

$$\langle \langle X = I; \curvearrowright y = -x; \rangle_{\mathsf{k}} \langle X \mapsto 0, y \mapsto 0 \rangle_{\mathsf{env}} \rangle_{\mathsf{cfg}}$$

Term s volnými proměnnými (X, I). Můžeme popsat strukturu takovýchto termů?

Co bylo na začátku? Program, sémantika, přechodový systém, konfigurace.

$$\langle \langle x = 5; \curvearrowright y = -x; \rangle_{\mathsf{k}} \langle x \mapsto 0, y \mapsto 0 \rangle_{\mathsf{env}} \rangle_{\mathsf{cfg}}$$

Tohle je nějaký term.

$$\langle \langle X = I; \curvearrowright y = -x; \rangle_{\mathsf{k}} \langle X \mapsto 0, y \mapsto 0 \rangle_{\mathsf{env}} \rangle_{\mathsf{cfg}}$$

Term s volnými proměnnými (X, I). Můžeme popsat strukturu takovýchto termů? Třeba sorty: $S = \{Cfg, Map, Stmt, Expr, Id, Int\}$

Co bylo na začátku? Program, sémantika, přechodový systém, konfigurace.

$$\langle\langle x = 5; \curvearrowright y = -x; \rangle_{\mathsf{k}} \langle x \mapsto 0, y \mapsto 0 \rangle_{\mathsf{env}} \rangle_{\mathsf{cfg}}$$

Tohle je nějaký term.

$$\langle\langle X = I; \curvearrowright y = -x; \rangle_{\mathsf{k}} \langle X \mapsto 0, y \mapsto 0 \rangle_{\mathsf{env}} \rangle_{\mathsf{cfg}}$$

Term s volnými proměnnými (X, I). Můžeme popsat strukturu takovýchto termů? Třeba sorty: $S = \{Cfg, Map, Stmt, Expr, Id, Int\}$ Symboly:

Co bylo na začátku? Program, sémantika, přechodový systém, konfigurace.

$$\langle\langle x = 5; \curvearrowright y = -x; \rangle_{\mathsf{k}} \langle x \mapsto 0, y \mapsto 0 \rangle_{\mathsf{env}} \rangle_{\mathsf{cfg}}$$

Tohle je nějaký term.

$$\langle\langle X = I; \curvearrowright y = -x; \rangle_{\mathsf{k}} \langle X \mapsto 0, y \mapsto 0 \rangle_{\mathsf{env}} \rangle_{\mathsf{cfg}}$$

Term s volnými proměnnými (X, I). Můžeme popsat strukturu takovýchto termů? Třeba sorty: $S = \{Cfg, Map, Stmt, Expr, Id, Int\}$ Symboly: např.

$$\langle\langle\_\rangle_k \langle\_\rangle_{env}\rangle_{cfg} \in \Sigma_{Stmt,Map,Cfg}$$

Co bylo na začátku? Program, sémantika, přechodový systém, konfigurace.

$$\langle\langle x = 5; \curvearrowright y = -x; \rangle_{\mathsf{k}} \langle x \mapsto 0, y \mapsto 0 \rangle_{\mathsf{env}} \rangle_{\mathsf{cfg}}$$

Tohle je nějaký term.

$$\langle\langle X = I; \curvearrowright y = -x; \rangle_{\mathsf{k}} \langle X \mapsto 0, y \mapsto 0 \rangle_{\mathsf{env}} \rangle_{\mathsf{cfg}}$$

Term s volnými proměnnými (X, I). Můžeme popsat strukturu takovýchto termů? Třeba sorty: $S = \{Cfg, Map, Stmt, Expr, Id, Int\}$ Symboly: např.

$$\langle\langle \_ \rangle_k \langle \_ \rangle_{env} \rangle_{cfg} \in \Sigma_{Stmt, Map, Cfg}$$

Mohli bychom použít vícedruhovou logiku prvního řádu k uvažování o konfiguracích?

Co bylo na začátku? Program, sémantika, přechodový systém, konfigurace.

$$\langle\langle x = 5; \curvearrowright y = -x; \rangle_{\mathsf{k}} \langle x \mapsto 0, y \mapsto 0 \rangle_{\mathsf{env}} \rangle_{\mathsf{cfg}}$$

Tohle je nějaký term.

$$\langle\langle X = I; \curvearrowright y = -x; \rangle_{\mathsf{k}} \langle X \mapsto 0, y \mapsto 0 \rangle_{\mathsf{env}} \rangle_{\mathsf{cfg}}$$

Term s volnými proměnnými (X, I). Můžeme popsat strukturu takovýchto termů? Třeba sorty: $S = \{Cfg, Map, Stmt, Expr, Id, Int\}$ Symboly: např.

$$\langle\langle \_ \rangle_k \langle \_ \rangle_{env} \rangle_{cfg} \in \Sigma_{Stmt,Map,Cfg}$$

Mohli bychom použít vícedruhovou logiku prvního řádu k uvažování o konfiguracích? Jak by vypadaly modely?

Co bylo na začátku? Program, sémantika, přechodový systém, konfigurace.

$$\langle\langle x = 5; \curvearrowright y = -x;\rangle_{\mathsf{k}}\langle x \mapsto 0, y \mapsto 0\rangle_{\mathsf{env}}\rangle_{\mathsf{cfg}}$$

Tohle je nějaký term.

$$\langle\langle X = I; \curvearrowright y = -x;\rangle_{\mathsf{k}}\langle X \mapsto 0, y \mapsto 0\rangle_{\mathsf{env}}\rangle_{\mathsf{cfg}}$$

Term s volnými proměnnými (X, I). Můžeme popsat strukturu takovýchto termů? Třeba sorty: $S = \{Cfg, Map, Stmt, Expr, Id, Int\}$ Symboly: např.

$$\langle\langle \_ \rangle_k \langle \_ \rangle_{env}\rangle_{cfg} \in \Sigma_{Stmt,Map,Cfg}$$

Mohli bychom použít vícedruhovou logiku prvního řádu k uvažování o konfiguracích? Jak by vypadaly modely? Skoro jako algebry termů. Ukázka.

Matching Logic - logika konfigurací

Definice (Signatura ML)

Signatura Matching Logiky (ML) je trojice $\Phi = (\Sigma, \Pi, Cfg)$, *kde* (Σ, Π) *je prvořádová signatura a* $Cfg \in \Sigma$ *je speciální sort pro konfigurace.*

Matching Logic - logika konfigurací

Definice (Signatura ML)

Signatura Matching Logiky (ML) je trojice $\Phi = (\Sigma, \Pi, Cfg)$, kde (Σ, Π) je prvořádová signatura a $Cfg \in \Sigma$ je speciální sort pro konfigurace.

Definice

Množina formulí ML nad signaturou Φ je definována:

$$\varphi ::= \pi \mid \top \mid p(t_1, \ldots, t_n) \mid \neg\varphi \mid \varphi \wedge \varphi \mid (\exists V)\,\varphi$$

kde π může být z $T_{\Sigma,Cfg}(Var)$ a ostatní podobně jako u formulí FOL.

Matching Logic - logika konfigurací

Definice (Signatura ML)

Signatura Matching Logiky (ML) je trojice $\Phi = (\Sigma, \Pi, Cfg)$, *kde* (Σ, Π) *je prvořádová signatura a* $Cfg \in \Sigma$ *je speciální sort pro konfigurace.*

Definice

Množina formulí ML nad signaturou Φ *je definována:*

$$\varphi ::= \pi \mid \top \mid p(t_1, \ldots, t_n) \mid \neg\varphi \mid \varphi \wedge \varphi \mid (\exists V)\,\varphi$$

kde π *může být z* $T_{\Sigma, Cfg}(Var)$ *a ostatní podobně jako u formulí FOL.*

Jaký je vztah mezi formulemi ML a FOL?

Matching Logic - logika konfigurací

Definice (Signatura ML)

Signatura Matching Logiky (ML) je trojice $\Phi = (\Sigma, \Pi, Cfg)$, *kde* (Σ, Π) *je prvořádová signatura a* $Cfg \in \Sigma$ *je speciální sort pro konfigurace.*

Definice

Množina formulí ML nad signaturou Φ *je definována:*

$$\varphi ::= \pi \mid \top \mid p(t_1, \ldots, t_n) \mid \neg\varphi \mid \varphi \wedge \varphi \mid (\exists V)\, \varphi$$

kde π *může být z* $T_{\Sigma,Cfg}(Var)$ *a ostatní podobně jako u formulí FOL.*

Jaký je vztah mezi formulemi ML a FOL? Příklad.

Motivace

```
int x,y;
x = get();
y = -x;
y = -y;
assert(x == y);
```

Může assert selhat?

A co funkce?

```
int foo(int x) {
  int y = -x;
  y = -y;
  return y;
}
// ...
int x = get();
assert(foo(x) == x);
```

A co šablony?

```
template < typename T >
T foo(T x) {
  T y = -x;
  y = -y;
  return y;
}

template < typename T >
void check() {
    T x = get<T>();
    assert(foo(x) == x);
}
```