Uniwersytet Warszawski
Wydział Matematyki, Informatyki i Mechaniki

Jakub Bujak
Nr albumu: 370737

Logika separacji dla języka programowania Jafun

Praca magisterska
na kierunku INFORMATYKA

Praca wykonana pod kierunkiem
dr hab. Aleksego Schuberta, prof. UW

Wrzesień 2020

Oświadczenie kierującego pracą

Potwierdzam, że niniejsza praca została przygotowana pod moim kierunkiem i kwalifikuje się do przedstawienia jej w postępowaniu o nadanie tytułu zawodowego.

Data Podpis kierującego pracą

Oświadczenie autora (autorów) pracy

Świadom odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam również, że przedstawiona praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu zawodowego w wyższej uczelni.

Oświadczam ponadto, że niniejsza wersja pracy jest identyczna z załączoną wersją elektroniczną.

Data Podpis autora pracy

Streszczenie

W pracy zdefiniowano logikę separacji dla języka Jafun, przedstawiono jej formalizację w systemie Coq i udowoniono jej poprawność względem semantyki języka. Logika separacji pozwala na podział sterty na rozłączne fragmenty. Upraszcza to wnioskowanie o programach, pozwalając na dowodzenie własności podwyrażeń na prostszych fragmentach sterty.

Słowa kluczowe

Logika separacji, Jafun, weryfikacja

Dziedzina pracy (kody wg programu Socrates-Erasmus)

11.3 Informatyka

Klasyfikacja tematyczna

Spis treści

Wprowadzenie

Rozdział 1

Podstawowe pojęcia i definicje

Rozdział 2

Jafun

Jafun to zorientowany obiektowo język programowania podobny do Javy. Jego szczegółowy opis znajduje się w pracy [1]. Poniżej przedstawiam te aspekty języka, które są istotne dla prezentowanej logiki.

2.1. Składnia i semantyka

2.2. Ewaluacja

Ewaluacją konfiguracji (h, st) będziemy nazywać dowolny ciąg par $(h_1, st_1), \ldots, (h_n, st_n)$, taki że $h_1 = h$, $st_1 = st$ oraz $(h_i, st_i) \to (h_{i+1}, st_{i+1})$ dla $1 \leq i < n$.

Ewaluacją wyrażenia e na stercie h będziemy nazywać taką ewaluację konfiguracji $(h, [\![e]\!]_\phi)$, że $st_n = [\![l]\!]_A$ dla pewnych l, A. Jeśli taka ewaluacja istnieje, będziemy to oznaczać jako $(h, e) \rightsquigarrow (h_n, A, l)$

Rozdział 3

Składnia i semantyka

Prezentowana logika separacji dla języka Jafun jest logiką z kwantyfikatorami egzystencjalnymi pierwszego rzędu, trójkami Hoare'a, operatorem $\hookrightarrow$, pozwalającym na opisywanie zawartości sterty i operatorami separacji $*$ i $-\!\!*$.

Iris, na którym wzorowana jest niniejsza logika, jest afiniczną logiką separacyjną, to znaczy własność spełniania termu przez stertę jest domknięta ze względu na rozszerzanie sterty. W celu zachowania zarówno afiniczności, jak i poprawności względem semantyki języka, prezentowana logika nie zawiera kwantyfikatora ogólnego, a kwantyfikator egzystencjalny jest ograniczony do termów najwyższego poziomu (Rysunek 3.1).

Używane będzie także oznaczenie $v_1 \neq v_2$ jako skrót dla $v_1 = v_2 \Rightarrow \texttt{False}$.

$$
\begin{aligned}
\mathbf{P} &::= \exists x : C.\mathbf{P} \mid \mathbf{P} \wedge \mathbf{P} \mid \mathbf{P} \vee \mathbf{P} \mid P \\
P &::= \texttt{True} \mid \texttt{False} \mid P \wedge P \mid P \vee P \mid P \Rightarrow P \mid v = v \mid \\
&\quad\; v \hookrightarrow x = v \mid \{P\}e\{x.P\}_A \mid P * P \mid P -\!\!* P \\
v &::= x \mid \texttt{null} \mid \texttt{this} \\
A &::= C \mid \phi \\
x &::= \langle \textit{identifier} \rangle \ (\textit{variable/field name}) \\
C &::= \langle \textit{identifier} \rangle \ (\textit{class name}) \\
e &::= \langle \textit{Jafun expression} \rangle
\end{aligned}
$$

Rysunek 3.1: Składnia logiki

Środowisko to funkcja częściowa przypisująca identyfikatorom lokacje na stercie lub `null`. Semantyka logiki (Rysunek 3.2) jest standardowa dla kwantyfikatora i operatorów logicznych. Dla uproszczenia zapisu notacja $[\![\cdot]\!]$ została użyta do opisu semantyki obu poziomów termów ($\mathbf{P}$ i P). To, do którego poziomu się odnosi, wynika z kontekstu.

Sterta spełnia trójkę Hoare'a $\{P\}e\{x.Q\}_A$, jeśli dla każdej sterty spełniającej P, wyrażenie e zostanie obliczone bez błędu, zwróci wyjątek typu A (czyli być może żaden), a wynikowa sterta będzie spełniała Q, w którym za x podstawiony zostanie wynik obliczenia.

Sterta spełnia term $P * Q$, jeśli można ją podzielić na dwa rozłączne fragmenty, z których jeden spełnia P, a drugi Q. Operator $-\!\!*$ to pewnego rodzaju odwrotność operatora $*$ – sterta spełnia $P -\!\!* Q$, jeśli po połączeniu jej z dowolną rozłączną stertą spełniającą P, otrzymana sterta spełnia Q.

$$
\begin{aligned}
[\![\texttt{True}]\!] &\triangleq \top \\
[\![\texttt{False}]\!] &\triangleq \bot \\
[\![\exists x : C.P]\!]_{h,env} &\triangleq \exists l : Loc \;.\; class(h,l) = C \wedge [\![P]\!]_{h,env[x \mapsto l]} \\
[\![P \wedge Q]\!]_{h,env} &\triangleq [\![P]\!]_{h,env} \wedge [\![Q]\!]_{h,env} \\
[\![P \vee Q]\!]_{h,env} &\triangleq [\![P]\!]_{h,env} \vee [\![Q]\!]_{h,env} \\
[\![P \Rightarrow Q]\!]_{h,env} &\triangleq [\![P]\!]_{h,env} \Rightarrow [\![Q]\!]_{h,env} \\
[\![x = y]\!]_{h,env} &\triangleq env(x) = env(y) \\
[\![x \hookrightarrow f = y]\!]_{h,env} &\triangleq h(env(x))(f) = env(y) \\
[\![\{P\}e\{x.Q\}_A]\!]_{h,env} &\triangleq \forall h : Heap \;.\; [\![P]\!]_{h,env} \Rightarrow \\
&\qquad \exists h' : Heap, l : Loc \;.\; (h, e[/env]) \rightsquigarrow (h', A, l) \wedge [\![Q]\!]_{h',env[x \mapsto l]} \\
[\![P * Q]\!]_{h,env} &\triangleq \exists h_1, h_2 : Heap \;.\; h_1 \oplus h_2 = h \wedge [\![P]\!]_{h_1,env} \wedge [\![Q]\!]_{h2,env} \\
[\![P \mathrel{-\!\!*} Q]\!]_{h,env} &\triangleq \forall h' : Heap \;.\; [\![P]\!]_{h',env} \Rightarrow [\![Q]\!]_{h \oplus h',env}
\end{aligned}
$$

Uwaga: $e[/env]$ oznacza wyrażenie powstałe przez podstawienie $env[x]$ w miejsce x dla każdej zmiennej wolnej x w e.

Rysunek 3.2: Semantyka logiki

Rozdział 4

Reguły wnioskowania

Osądy w prezentowanej logice są postaci $\Gamma|P \vdash Q$, gdzie Γ to środowisko typów, przypisujące zmiennym odpowiadające im typy (czyli nazwy klas), a P i Q to termy logiki. Intuicyjnie, osąd $\Gamma|P \vdash Q$ oznacza że Q wynika z P, a więc że każda sterta spełniająca P spełnia też Q.

Dla poprawienia czytelności, jeśli Γ jest wspólne dla wszystkich osądów występujących w danej regule, to jest ono pomijane.

$$\textsc{Asm}\ \frac{}{P \vdash P} \qquad \textsc{Trans}\ \frac{P \vdash Q \qquad Q \vdash R}{P \vdash R} \qquad \textsc{Eq-refl}\ \frac{}{P \vdash v = v} \qquad \textsc{Eq-sym}\ \frac{P \vdash v = w}{P \vdash w = v}$$

$$\bot\text{E}\ \frac{P \vdash \texttt{False}}{P \vdash Q} \qquad \top\text{I}\ \frac{}{P \vdash \texttt{True}} \qquad \wedge\text{I}\ \frac{R \vdash P \qquad R \vdash Q}{R \vdash P \wedge Q} \qquad \wedge\text{EL}\ \frac{R \vdash P \wedge Q}{R \vdash P}$$

$$\wedge\text{ER}\ \frac{R \vdash P \wedge Q}{R \vdash Q} \qquad \vee\text{IL}\ \frac{R \vdash P}{R \vdash P \vee Q} \qquad \vee\text{IR}\ \frac{R \vdash Q}{R \vdash P \vee Q}$$

$$\vee\text{E}\ \frac{S \vdash P \vee Q \qquad P \vdash R \qquad Q \vdash R}{S \vdash R} \qquad \Rightarrow\text{I}\ \frac{R \wedge P \vdash Q}{R \vdash P \Rightarrow Q} \qquad \Rightarrow\text{E}\ \frac{R \vdash P \Rightarrow Q \qquad R \vdash P}{R \vdash P \Rightarrow Q}$$

$$\exists\text{I}\ \frac{\Gamma, x : C|Q \vdash P[t/x]}{Q \vdash \exists x : C.P} \qquad \exists\text{E}\ \frac{\Gamma|R \vdash \exists x : C.P \qquad \Gamma, x : C|R \wedge P \vdash Q}{\Gamma|R \vdash Q}$$

Rysunek 4.1: Reguły wnioskowania dla tradycyjnych operatorów logicznych

$$\textsc{Weak}\ \frac{}{P * Q \vdash P} \qquad \textsc{Sep-assoc}\ \frac{}{P * (Q * R) \dashv\vdash (P * Q) * R} \qquad \textsc{Sep-sym}\ \frac{}{P * Q \vdash Q * P}$$

$$*\text{I}\ \frac{P_1 \vdash Q_1 \qquad P_2 \vdash Q_2}{P_1 * Q_1 \vdash P_2 * Q_2} \qquad -\!\!*\text{I}\ \frac{R * P \vdash Q}{R \vdash P -\!\!* Q} \qquad -\!\!*\text{E}\ \frac{R_1 \vdash P -\!\!* Q \qquad R_2 \vdash P}{R_1 * R_2 \vdash Q}$$

Rysunek 4.2: Reguły wnioskowania dla operatorów separacyjnych

Reguły strukturalne dla trójek Hoare'a

$$\textsc{Ht-frame}\ \frac{S \vdash \{P\}e\{v.Q\}_A \qquad S \text{ jest trwały}}{S \vdash \{P * R\}e\{v.Q * R\}_A} \qquad \textsc{Ht-ret}\ \frac{}{S \vdash \{\texttt{True}\}w\{v.v = w\}_\phi}$$

$$\textsc{Ht-csq}\ \frac{\Gamma|S \vdash P \Rightarrow P' \qquad \Gamma|S \vdash \{P'\}e\{v.Q'\}_A \qquad \Gamma, v : C|S \vdash Q' \Rightarrow Q \qquad S \text{ jest trwały}}{S \vdash \{P\}e\{v.Q\}_A}$$

$$\textsc{Ht-disj}\ \frac{S \vdash \{P\}e\{v.Q\}_A \qquad S \vdash \{Q\}e\{v.Q\}_A}{S \vdash \{P \vee Q\}e\{v.Q\}_A}$$

$$\textsc{Ht-pers}\ \frac{S \wedge R \vdash \{Q\}e\{v.Q\}_A}{S \vdash \{Q \wedge R\}e\{v.Q\}_A}\ \text{jeśli R trwały}$$

Reguły dla trójek Hoare'a opisujących konstrukcje języka

$$\textsc{Ht-new-null}\ \frac{}{S \vdash \{\texttt{True}\}\textbf{new}\ C(\overline{v})\{w.w \neq \textbf{null}\}_\phi}$$

$$\textsc{Ht-new-field}\ \frac{\text{flds}(C) = f_1, \ldots, f_n}{S \vdash \{\texttt{True}\}\textbf{new}\ C(v_1, \ldots, v_n)\{w.w \hookrightarrow f_i = v_i\}_\phi}$$

$$\textsc{Ht-let}\ \frac{\Gamma|S \vdash \{P\}E_1\{x.Q\}_\phi \qquad \Gamma, x : C|S \vdash \{Q\}E_2\{w.R\}_A}{\Gamma|S \vdash \{P\}\textbf{let}\ C\ x = E_1\ \textbf{in}\ E_2\{w.R\}_A}\ \text{jeśli S trwały}$$

$$\textsc{Ht-let-ex}\ \frac{S \vdash \{P\}E_1\{w.Q\}_A \qquad A \neq \phi}{\Gamma|S \vdash \{P\}\textbf{let}\ C\ x = E_1\ \textbf{in}\ E_2\{w.Q\}_A}$$

$$\textsc{Ht-field-set}\ \frac{}{S \vdash \{x \neq \texttt{null}\}x.f = v\{\_.x \hookrightarrow f = v\}_\phi}$$

$$\textsc{Ht-null-set}\ \frac{}{S \vdash \{x = \texttt{null}\}x.f = v\{w.w = \texttt{npe}\}_{\texttt{NPE}}}$$

$$\textsc{Ht-field-get}\ \frac{}{S \vdash \{x \hookrightarrow f = v\}x.f\{w.w = v\}_\phi}$$

$$\textsc{Ht-null-get}\ \frac{}{S \vdash \{x = \texttt{null}\}x.f\{w.w = \texttt{npe}\}_{\texttt{NPE}}}$$

Rysunek 4.3: Reguły wnioskowania dla trójek Hoare'a

$$\textsc{Ht-if}\ \frac{S \vdash \{P \wedge v_1 = v_2\} E_1 \{w.Q\}_A \qquad S \vdash \{P \wedge v_1 \neq v_2\} E_2 \{w.Q\}_A}{S \vdash \{P\}\textbf{if } v_1 = v_2 \textbf{ then } E_1 \textbf{ else } E_2 \{w.Q\}_A}$$

$$\textsc{Ht-invoke}\ \frac{\Gamma \vdash x : C \qquad \begin{array}{c} \{P'\} \cdot \{w.Q'\}_A \in \texttt{invariants}(C, m) \\ S \wedge \{P'\} x.m(\overline{v})\{w.Q'\}_A \vdash \{P\} x.m(\overline{v})\{w.Q\}_A \end{array}}{S \vdash \{P\} x.m(\overline{v})\{w.Q\}_A}$$

$$\textsc{Ht-null-invoke}\ \frac{}{S \vdash \{x = \texttt{null}\} x.m(\overline{v})\{w.w = \texttt{npe}\}_{\texttt{NPE}}}$$

$$\textsc{Ht-throw}\ \frac{\Gamma \vdash x : C}{S \vdash \{x \neq \texttt{null}\}\textbf{throw } x\{w.w = x\}_C}$$

$$\textsc{Ht-null-throw}\ \frac{}{S \vdash \{x = \texttt{null}\}\textbf{throw } x\{w.w = \texttt{npe}\}_{\texttt{NPE}}}$$

$$\textsc{Ht-catch-normal}\ \frac{S \vdash \{P\} E_1 \{w.Q\}_\phi}{S \vdash \{P\}\textbf{try } E_1 \textbf{ catch } (C\ x)\ E_2 \{w.Q\}_\phi}$$

$$\textsc{Ht-catch-ex}\ \frac{\Gamma | S \vdash \{P\} E_1 \{x.Q\}'_C \qquad \Gamma, x : C' | S \vdash \{Q\} E_2 \{w.R\}_A \qquad C' \leq C}{\Gamma | S \vdash \{P\}\textbf{try } E_1 \textbf{ catch } (C\ x)\ E_2 \{w.R\}_A} \text{ jeśli S trwały}$$

$$\textsc{Ht-catch-pass}\ \frac{S \vdash \{P\} E_1 \{w.Q\}'_C \qquad C' \not\leq C}{S \vdash \{P\}\textbf{try } E_1 \textbf{ catch } (C\ x)\ E_2 \{w.Q\}'_C}$$

Rysunek 4.4: Reguły wnioskowania dla trójek Hoare'a - c.d.

Rozdział 5

Własności ewaluacji

Rozdział 6

Poprawność

Rozdział 7

Formalizacja w systemie Coq

Rozdział 8

Podsumowanie

Bibliografia

[1] J. Chrząszcz and A. Schubert. Function definitions for compound values in object- oriented languages. In *Proc. of the 19th International Symposium on Principles and Practice of Declarative Programming*, PPDP '17, pp. 61–72. ACM, 2017.

[2] J. Chrząszcz and A.Schubert. Formalisation of a frame stack semantics for a Java-like language. 2018