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 przytaczam te aspekty języka, które są istotne dla prezentowanej logiki.

2.1. Składnia i semantyka

Program w języku jafun jest listą definicji klas. Definicja klasy składa się z listy pól i listy metod. Metody mogą przyjmować dowolną liczbę argumentów i rzucać dowolną liczbę wyjątków, deklarowanych przez słowo kluczowe **throws**, podobnie jak w Javie.

Modyfikatory dostępu ϕ i μ nie mają znaczenia w prezentowanej logice, ale zostały uwzględnione w składni dla kompletności opisu.

$$
\begin{array}{rcl}
\mathsf{Prog} \ni \mathbf{C} & ::= & \mathbf{class}\ C_1\ \mathbf{ext}\ C_2\ \{\overline{\mathbf{F}}\ \overline{\mathbf{M}}\} \\
\mathsf{CId} \ni C & ::= & \langle \mathit{identifier} \rangle \quad (\mathit{class\ name}) \\
\mathbf{F} & ::= & \phi\ C\ x \\
\phi & ::= & \mathtt{rep} \mid \emptyset \\
\mathsf{Id} \ni x & ::= & \langle \mathit{identifier} \rangle \quad (\mathit{variable/field\ name}) \\
\mathsf{arg} & ::= & \mu\ C\ x \qquad \mathsf{argn}\ ::= \emptyset\ C\ x \\
\mathsf{Exc} & ::= & \mu\ C \qquad \mathsf{Excn} ::= \emptyset\ C \\
\mathbf{M} & ::= & \mu\ C\ \mu\ m(\overline{\mathsf{arg}})\ \mathbf{throws}\ \overline{\mathsf{Exc}}\ \{E\} \mid \\
 & & \emptyset\ C\ \emptyset\ m(\overline{\mathsf{argn}})\ \mathbf{throws}\ \overline{\mathsf{Excn}}\ \{E\} \\
\mathsf{AMod} \ni \mu & ::= & \mathtt{rwr} \mid \mathtt{rd} \mid \mathtt{atm} \\
\mathsf{MId} \ni m & ::= & \langle \mathit{identifier} \rangle \quad (\mathit{method\ name}) \\
\mathsf{Expr} \ni E & ::= & \mathbf{new}\ \mu\ C(\overline{\mathsf{v}}) \mid \mathbf{let}\ C\ x = E_1\ \mathbf{in}\ E_2 \mid \\
 & & \mathbf{if}\ \mathsf{v}_1 == \mathsf{v}_2\ \mathbf{then}\ E_3\ \mathbf{else}\ E_4 \mid \mathsf{v}.m(\overline{\mathsf{v}}) \mid \\
 & & \mathsf{fieldref} = \mathsf{v} \mid \mathsf{v} \mid \mathsf{fieldref} \mid \mathbf{throw}\ \mathsf{v} \mid \\
 & & \mathbf{try}\ \{E_1\}\ \mathbf{catch}\ (\mu\ C\ x)\ \{E_2\} \\
\mathsf{v} & ::= & x \mid \mathbf{this} \mid \mathbf{null} \\
\mathsf{fieldref} & ::= & \mathsf{v}.x \\
 & & \\
A & ::= & C \mid \emptyset \\
\mathsf{BCtxt} \ni \mathcal{C} & ::= & [\![\ ]\!]_A \mid \mathbf{let}\ C\ x = \mathcal{C}\ \mathbf{in}\ E \mid \\
 & & \mathbf{try}\ \{\mathcal{C}\}\ \mathbf{catch}\ (\mu\ C\ x)\ \{E\}
\end{array}
$$

Rysunek 2.1: Składnia języka Jafun

Notacje pomocnicze dla deklaracji w $\overline{\mathbf{C}}$

Niech **class** C_1 **ext** C_2 $\{\overline{\mathbf{F}}\ \overline{\mathbf{M}}\}$ będzie deklaracją klasy w $\overline{\mathbf{C}}$. Niech $\phi\ C_3\ x$ będzie deklaracją pola w $\overline{\mathbf{F}}$. Niech $\mu_r\ C_4\ \mu_o\ m(\overline{\mathsf{arg}})$ **throws** $\overline{\mathsf{Exc}}$ $\{E_1\}$ będzie deklaracją metody w $\overline{\mathbf{M}}$, gdzie $\overline{\mathsf{arg}} = \mu'_1\ C'_1\ x_1, \ldots, \mu'_n\ C'_n\ x_n$, $\overline{\mathsf{Exc}} = \mu''_1\ C''_1, \ldots, \mu''_k\ C''_k$, and $\mu_r, \mu_o, \mu'_i, \mu''_j \in$ AMod for all possible i, j. Ustalając $\overline{\mathbf{M}}_1 \cup\!\!\!\cdot\ \overline{\mathbf{M}}_2 = \overline{\mathbf{M}}_1 \cup \{\mathbf{M} \in \overline{\mathbf{M}}_2 \mid \mathsf{name}(\mathbf{M}) \notin \overline{\mathbf{M}}_1\}$, możemy zdefiniować następujące pomocnicze notacje:

$C_1 \in \overline{\mathbf{C}}$	kiedy w deklaracja C_1 istnieje w $\overline{\mathbf{C}}$,
$\mathtt{Object} \in \overline{\mathbf{C}}$, $\mathtt{NPE} \in \overline{\mathbf{C}}$	
$\mathsf{flds}(C_1) = \{x \in \mathsf{Id} \mid \phi\, D_1\, x \in \overline{\mathbf{F}}\} \cup \mathsf{flds}(C_2)$	$\mathsf{flds}(\mathsf{Object}) = \emptyset$
$\overline{\mathsf{flds}}(C_1) = \overline{\mathbf{F}}, \overline{\mathsf{flds}}(C_2)$	$\overline{\mathsf{flds}}(\mathsf{Object}) = \emptyset$
$\mathsf{mthds}(C_1) = \overline{\mathbf{M}} \cup\!\!\!\cdot\ \mathsf{mthds}(C_2)$	$\mathsf{mthds}(\mathsf{Object}) = \emptyset$
$\mathsf{ext}(C_1) = C_2$	$\mathsf{ext}(\mathsf{Object}) = \emptyset$
$x \in C_1$	kiedy deklaracja x istnieje w $\overline{\mathbf{F}}$,
$\mathsf{typeof}(C_1, x) = C_3$	dla $x \in C_1$
$m \in C_1$	kiedy deklaracja m istnieje w $\overline{\mathbf{M}}$,
$\mathsf{body}(C_1, m) = E_1$,	dla $m \in C_1$,
$\mathsf{pars}(C_1, m) = \overline{\mathsf{arg}}$	dla $m \in C_1$,
$\mathsf{parNms}(C_1, m) = x_1, \ldots, x_n$	dla $m \in C_1$,
$\mathsf{class}(h, l)$	klasa obiektu znajdującego się pod lokacją $l \in \mathsf{Loc}$ na stercie $h \in \mathsf{Heap}$
$\mathsf{class}(h, \mathbf{null}) = \bot$	dla każdej sterty h
$\mathsf{alloc} : \mathsf{Heap} \times \mathsf{Prog} \times \mathsf{CId} \to \mathsf{Loc} \times \mathsf{Heap}$	funkcja alokacji
$\mathsf{empty}_C(x) = \mathbf{null}$	dla $x \in \mathsf{flds}(C)$

Rysunek 2.2: Notacje pomocnicze

Semantyka małych kroków języka Jafun jest zdefiniowana przez relację $\to$ na rysunku 2.3, dla ustalonego programu . Relacja $\to$ jest relacją binarną na parach (sterta, stos wywołań). W ogólności ma ona postać

$$\overline{\mathbf{C}},\quad h, \mathcal{C}_1[\![E_1]\!]_{A_1} :: \cdots :: \mathcal{C}_n[\![E_n]\!]_{A_n} \to h', \mathcal{C}'_1[\![E_1]\!]_{A'_1} :: \cdots :: \mathcal{C}'_m[\![E_m]\!]_{A'_m}.$$

Stos wywołań $\mathcal{C}_1[\![E_1]\!]_{A_1} :: \cdots :: \mathcal{C}_n[\![E_n]\!]_{A_n}$, albo w skrócie $\overline{\mathcal{C}}$, to ciąg wyrażeń z rysunku 2.1, w którym aktualnie ewaluowane wyrażenie (redeks) jest oznaczone specjalnym symbolem $[\![\ ]\!]_A$. Indeks A opisuje, czy program wykonuje się w sposób normalny ($A = \emptyset$), czy był rzucony jakiś niezłapany jeszcze wyjątek ($A \in \overline{\mathbf{C}}$).

Dla wygody każda ramka stosu jest podzielona na kontekst $\mathcal{C}_i \in \mathsf{BCtxt}$ i redeks E_i. Kontekst $\mathcal{C}_i$ opisuje wszystkie zagnieżdżone bloki **let** i **catch**, wewnątrz którch znajduje się E_i.

$$\begin{array}{ll}
\text{(newk)} & \overline{\mathbf{C}}, h, \overline{\mathcal{C}} :: \mathcal{C}[\![\mathbf{new}\ \mu\ C(l_1, \dots, l_k)]\!]_\emptyset \to h'', \overline{\mathcal{C}} :: \mathcal{C}[\![l_0]\!]_\emptyset \\
& \quad \text{gdzie } \mathsf{alloc}(h, \overline{\mathcal{C}}, C) = (l_0, h'),\ \mathsf{flds}(C) = x_1, \dots, x_k, \\
& \qquad o = \mathsf{empty}_C\{x_1 \mapsto l_1, \dots, x_k \mapsto l_k\},\ h'' = h'\{l_0 \mapsto o\} \\
\\
\text{(letin)} & \overline{\mathbf{C}}, h, \overline{\mathcal{C}} :: \mathcal{C}[\![\mathbf{let}\ C\ x = E_1\ \mathbf{in}\ E_2]\!]_\emptyset \to h, \overline{\mathcal{C}} :: \mathcal{C}[\mathbf{let}\ C\ x = [\![E_1]\!]_\emptyset\ \mathbf{in}\ E_2] \\
\text{(letgo)} & \overline{\mathbf{C}}, h, \overline{\mathcal{C}} :: \mathcal{C}[\mathbf{let}\ C\ x = [\![l]\!]_\emptyset\ \mathbf{in}\ E] \to h, \overline{\mathcal{C}} :: \mathcal{C}[\![E\{l/x\}]\!]_\emptyset \\
\\
\text{(ifeq)} & \overline{\mathbf{C}}, h, \overline{\mathcal{C}} :: \mathcal{C}[\![\mathbf{if}\ l_0 == l_1\ \mathbf{then}\ E_1\ \mathbf{else}\ E_2]\!]_\emptyset \to h, \overline{\mathcal{C}} :: \mathcal{C}[\![E_1]\!]_\emptyset \quad \text{gdzie } l_0 = l_1 \\
\text{(ifneq)} & \overline{\mathbf{C}}, h, \overline{\mathcal{C}} :: \mathcal{C}[\![\mathbf{if}\ l_0 == l_1\ \mathbf{then}\ E_1\ \mathbf{else}\ E_2]\!]_\emptyset \to h, \overline{\mathcal{C}} :: \mathcal{C}[\![E_2]\!]_\emptyset \quad \text{gdzie } l_0 \neq l_1 \\
\\
\text{(mthdnpe)} & \overline{\mathbf{C}}, h, \overline{\mathcal{C}} :: \mathcal{C}[\![\mathbf{null}.m(\bar{l})]\!]_\emptyset \to h, \overline{\mathcal{C}} :: \mathcal{C}[\![\mathsf{npe}]\!]_{\mathtt{NPE}} \\
\text{(mthd)} & \overline{\mathbf{C}}, h, \overline{\mathcal{C}} :: \mathcal{C}[\![l.m(\bar{l})]\!]_\emptyset \to h, \overline{\mathcal{C}} :: \mathcal{C}[\![l.m(\bar{l})]\!]_\emptyset :: [\![E]\!]_\emptyset \\
& \quad \text{gdzie } \mathsf{class}(h, l) = D, \mathsf{body}(D, m) = E_0, E = E_0\{l/\mathbf{this}, \bar{l}/\mathsf{parNms}(D, m)\} \\
\text{(mthdret)} & \overline{\mathbf{C}}, h, \overline{\mathcal{C}} :: \mathcal{C}[\![l.m(\bar{l})]\!]_\emptyset :: [\![l']\!]_\emptyset \to h, \overline{\mathcal{C}} :: \mathcal{C}[\![l']\!]_\emptyset \\
\\
\text{(assignnpe)} & \overline{\mathbf{C}}, h, \overline{\mathcal{C}} :: \mathcal{C}[\![\mathbf{null}.x = l]\!]_\emptyset \to h, \overline{\mathcal{C}} :: \mathcal{C}[\![\mathsf{npe}]\!]_{\mathtt{NPE}} \\
\text{(assignev)} & \overline{\mathbf{C}}, h, \overline{\mathcal{C}} :: \mathcal{C}[\![l_1.x = l]\!]_\emptyset \to h', \overline{\mathcal{C}} :: \mathcal{C}[\![l]\!]_\emptyset \\
& \quad \text{gdzie } l_1 \neq \mathbf{null}, o = h(l_1)\{x \mapsto l\}, h' = h\{l_1 \mapsto o\} \\
\\
\text{(varnpe)} & \overline{\mathbf{C}}, h, \overline{\mathcal{C}} :: \mathcal{C}[\![\mathbf{null}.x]\!]_\emptyset \to h, \overline{\mathcal{C}} :: \mathcal{C}[\![\mathsf{npe}]\!]_{\mathtt{NPE}} \\
\text{(var)} & \overline{\mathbf{C}}, h, \overline{\mathcal{C}} :: \mathcal{C}[\![l.x]\!]_\emptyset \to h, \overline{\mathcal{C}} :: \mathcal{C}[\![l']\!]_\emptyset \quad \text{gdzie } l \neq \mathbf{null}, l' = h(l)(x) \\
\\
\text{(thrownull)} & \overline{\mathbf{C}}, h, \overline{\mathcal{C}} :: \mathcal{C}[\![\mathbf{throw\ null}]\!]_\emptyset \to h, \overline{\mathcal{C}} :: \mathcal{C}[\![\mathsf{npe}]\!]_{\mathtt{NPE}} \\
\\
\text{(throw)} & \overline{\mathbf{C}}, h, \overline{\mathcal{C}} :: \mathcal{C}[\![\mathbf{throw}\ l]\!]_\emptyset \to h, \overline{\mathcal{C}} :: \mathcal{C}[\![l]\!]_D \quad \text{gdzie } l \neq \mathbf{null}, \mathsf{class}(h, l) = D \\
\\
\text{(ctchin)} & \overline{\mathbf{C}}, h, \overline{\mathcal{C}} :: \mathcal{C}[\![\mathbf{try}\ \{E_1\}\ \mathbf{catch}\ (\mu\ C\ x)\ \{E_2\}]\!]_\emptyset \to \\
& \qquad h, \overline{\mathcal{C}} :: \mathcal{C}[\mathbf{try}\ \{[\![E_1]\!]_\emptyset\}\ \mathbf{catch}\ (\mu\ C\ x)\ \{E_2\}] \\
\text{(ctchnrml)} & \overline{\mathbf{C}}, h, \overline{\mathcal{C}} :: \mathcal{C}[\mathbf{try}\ \{[\![l]\!]_\emptyset\}\ \mathbf{catch}\ (\mu\ C\ x)\ \{E_2\}] \to h, \overline{\mathcal{C}} :: \mathcal{C}[\![l]\!]_\emptyset \\
\text{(ctchexok)} & \overline{\mathbf{C}}, h, \overline{\mathcal{C}} :: \mathcal{C}[\mathbf{try}\ \{[\![l]\!]_{C'}\}\ \mathbf{catch}\ (\mu\ C\ x)\ \{E_2\}] \to h, \overline{\mathcal{C}} :: \mathcal{C}[\![E_2']\!]_\emptyset \\
& \quad \text{gdzie } E_2' = E_2\{l/x\}, C' \leq: C \\
\\
\text{(letex)} & \overline{\mathbf{C}}, h, \overline{\mathcal{C}} :: \mathcal{C}[\mathbf{let}\ C\ x = [\![l]\!]_{C'}\ \mathbf{in}\ E] \to h, \overline{\mathcal{C}} :: \mathcal{C}[\![l]\!]_{C'} \quad \text{gdzie } C' \neq \emptyset \\
\text{(methodex)} & \overline{\mathbf{C}}, h, \overline{\mathcal{C}} :: \mathcal{C}[\![l.m(\bar{l})]\!]_\emptyset :: [\![l']\!]_C \to h, \overline{\mathcal{C}} :: \mathcal{C}[\![l']\!]_C \quad \text{gdzie } C \neq \emptyset \\
\text{(ctchexnok)} & \overline{\mathbf{C}}, h, \overline{\mathcal{C}} :: \mathcal{C}[\mathbf{try}\ \{[\![l]\!]_{C'}\}\ \mathbf{catch}\ (\mu\ C\ x)\ \{E_2\}] \to h, \overline{\mathcal{C}} :: \mathcal{C}[\![l]\!]_{C'} \\
& \quad \text{gdzie } C' \neq \emptyset, C' \not\leq: C
\end{array}$$

Rysunek 2.3: Semantyka języka Jafun

Ewaluacja programu zaczyna się od stanu $\overline{\mathbf{C}}, h, [\![E']\!]_\emptyset$ gdzie $h \in \mathsf{Heap}$, $E' = E\{l_o/\mathbf{this}\}$, $\mathsf{class}(h, l_o) = C$, a $C'\ m()$ **throws** `NPE` $\{E\}$ jest metodą nieprzyjmującą argumentów w klasie C. Metoda m odpowiada funkcji `main` w zwykłej Javie. Dodatkowo zakładamy, że na stercie h, pod pewną lokacją npe istnieje object klasy `NPE` ("null pointer exception").

2.2. Ewaluacja

Częściową ewaluacją konfiguracji $(h, \overline{\mathcal{C}})$ będziemy nazywać dowolny ciąg par $\mathit{confs} = (h_1, \overline{\mathcal{C}}_1), \dots, (h_n, \overline{\mathcal{C}}_n)$, taki że $h_1 = h$, $\overline{\mathcal{C}}_1 = \overline{\mathcal{C}}$ oraz $(h_i, \overline{\mathcal{C}}_i) \to (h_{i+1}, \overline{\mathcal{C}}_{i+1})$ dla $1 \leq i < n$. Częściowe ewaluacje będziemy oznaczać jako $(h_1, \overline{\mathcal{C}}_1) \overset{confs}{\rightsquigarrow} (h_n, \overline{\mathcal{C}}_n)$.

Ewaluacją wyrażenia e na stercie h będziemy nazywać taką ewaluację konfiguracji $(h, [\![e]\!]_\phi)$, że $\overline{\mathcal{C}}_n = [\![l]\!]_A$ dla pewnych l, A. Jeśli taka ewaluacja istnieje, będziemy to oznaczać jako $(h, e) \overset{confs}{\rightsquigarrow} (h_n, A, l)$ lub

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 i **this** 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. Dodatkowo zakładamy, że wszystkie zmienne wolne występujące w termach w danych osądzie muszą być obecne w środowisku typów dla tego osądu, tj. żeby osąd $\Gamma|P \vdash Q$ mógł pojawić się w regule, wszystkie zmienne wolne w termach P i Q muszą mieć przypisany pewien typ w Γ.

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

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

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

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

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

Rysunek 4.1: Reguły wnioskowania dla tradycyjnych operatorów logicznych

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

$$*\mathrm{I}\ \frac{\Gamma_1 | P_1 \vdash Q_1 \qquad \Gamma_2 | P_2 \vdash Q_2 \qquad \Gamma_1 \cap \Gamma_2 = \emptyset}{\Gamma_1, \Gamma_2 | P_1 * P_2 \vdash Q_1 * Q_2}$$

$$-\!\!*\mathrm{I}\ \frac{R * P \vdash Q}{R \vdash P -\!\!* Q} \qquad -\!\!*\mathrm{E}\ \frac{\Gamma_1 | R_1 \vdash P -\!\!* Q \qquad \Gamma_2 | R_2 \vdash P \qquad \Gamma_1 \cap \Gamma_2 = \emptyset}{\Gamma_1, \Gamma_2 | R_1 * R_2 \vdash Q}$$

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

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

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

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

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

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

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

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

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

Rysunek 4.3: Reguły wnioskowania dla trójek Hoare'a

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

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

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

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

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

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

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

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

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

Rozdział 5

Własności stert

5.1. Definicje

Definicja 5.1.1 (Spójność sterty)**.**
Niech h będzie stertą. Powiemy, że h jest spójna, jeśli każda lokacja będąca wartością pola w pewnym obiekcie na h również jest na h. To znaczy, dla każdego x, l_1, l_2, jeśli $h(l_1)(x) = l_2$, to $l_2 \in \mathsf{Dom}(h)$. $\square$

Definicja 5.1.2 (Suma rozłączna stert)**.**
Niech h, h_1, h_2 będą stertami. Powiemy, że h jest sumą rozłączną stert h_1 i h_2 (zapisywane $h = h_1 \oplus h_2$), jeśli

1. Dla każdej lokacji l, $l \in h$ wtedy i tylko wtedy gdy $l \in h_1$ lub $l \in h_2$
2. Nie istnieje lokacja l, taka że $l \in h_1$ i $l \in h_2$

$\square$

Rozdział 6

Własności ewaluacji

Pokażę teraz twierdzenia o własności ewaluacji, które będą później użyte do udowodnienia poprawności reguł dla trójek Hoare'a.

6.1. Łączenie ewaluacji

Podatwowym twierdzeniem, pozwalającym mówić o ewaluacji złożonych wyrażeń, jest twierdzenie o łączeniu ewaluacji.

Twierdzenie 1 (O łączeniu ewaluacji)**.** *Niech* $(h, \overline{\mathcal{C}}), (h', \overline{\mathcal{C}}'), (h'', \overline{\mathcal{C}}'')$ *będą konfiguracjami, a* $confs$ *i* $confs'$ *– ciągami konfiguracji, takimi że* $(h, \overline{\mathcal{C}}) \overset{confs}{\rightsquigarrow} (h', \overline{\mathcal{C}}')$ *i* $(h', \overline{\mathcal{C}}') \overset{confs'}{\rightsquigarrow} (h'', \overline{\mathcal{C}}'')$*. Wtedy* $(h, \overline{\mathcal{C}}) \overset{confs{+}{+}confs'}{\rightsquigarrow} (h'', \overline{\mathcal{C}}'')$*.*

Dowód. TODO □

6.2. Ewaluacja w rozszerzonym kontekście

Reguły takie jak HT-LET pozwalają wnioskować o ewaluacji złożonych wyrażeń na podstawie założeń o ewaluacji ich podwyrażeń. Dowód ich poprawności wymaga jednak skonstruowania ewaluacji wyrażenia opakowanego w dodatkowy kontekst **let** lub **try** przy założeniu, że istnieje ewaluacja tego wyrażenia bez dodatkowego kontekstu. W tej sekcji sformułuję i udowodnię twierdzenie pozwalające mówić o takich ewaluacjach.

Lemat 1 (Redukcja w rozszerzonym kontekście)**.**
Niech $h, h', \overline{\mathcal{C}}, \overline{\mathcal{C}}', \mathcal{C}, \mathcal{C}', E_1, E_1', A, A'$ *będą takie, że* $(h, \mathcal{C}[\![E_1]\!]_A :: \overline{\mathcal{C}}) \to (h', \mathcal{C}'[\![E_1']\!]_{A'} :: \overline{\mathcal{C}}')$*. Wtedy dla dowolnych* μ, C, x, E_2 *istnieją również następujące redukcje:*

- $(h, [\textbf{let}\ C\ x = \mathcal{C}[\![E_1]\!]_A\ \textbf{in}\ E_2] :: \overline{\mathcal{C}}) \to (h', [\textbf{let}\ C\ x = \mathcal{C}[\![E_1']\!]_{A'}\ \textbf{in}\ E_2] :: \overline{\mathcal{C}}')$
- $(h, [\textbf{try}\ \{\mathcal{C}[\![E_1]\!]_A\}\ \textbf{catch}\ (\mu\ C\ x)\ \{E_2\}] :: \overline{\mathcal{C}}) \to$
  $(h, [\textbf{try}\ \{\mathcal{C}'[\![E_1']\!]_{A'}\}\ \textbf{catch}\ (\mu\ C\ x)\ \{E_2\}] :: \overline{\mathcal{C}}')$

Dowód. W pierwszej kolejności, rozpatrzmy dwa przypadki: jeśli stos $\overline{\mathcal{C}}$ jest niepusty, redukowany jest pewnien redeks z $\overline{\mathcal{C}}$, a zatem ramka $\mathcal{C}[\![E_1]\!]_A$ pozostaje bez zmian. Stąd, także w przypadku rozszerzonego kontekstu redukowany będzie redeks ze stosu $\overline{\mathcal{C}}$, a rozszerzona ramka $[\textbf{let}\ C\ x = \mathcal{C}[\![E_1]\!]_A\ \textbf{in}\ E_2]$ pozostanie bez zmian.

Załóżmy więc, że stos $\overline{\mathcal{C}}$ jest pusty, a E_1 jest aktualnym redeksem. Zauważmy, że zdecydowana większość przypadków redukcji w ogóle nie rusza kontekstu, w którym znajduje się

E_1, a tylko modyfikuje samo wyrażenie, dodaje nowy kontekst lub wrzuca nową ramkę na stos. We wszystkich tych przypadkach, redukcja jest niezależna od zewnętrznych kontekstów, a więc będzie przeiegała tak samo przy rozszerzonym kontekście.

Wyjątkiem są redukcje letgo i letex, powodujące zdjęcie kontekstu **let** oraz redukcje ctchnrml, ctchexok, ctchexnok, powodujące zdjęcie kontekstu **try**. Jeśli jednak redukcja $(h, \mathcal{C}[\![E_1]\!]_A :: \overline{\mathcal{C}}) \to (h', \mathcal{C}'[\![E_1']\!]_{A'} :: \overline{\mathcal{C}}')$ jest jedną z nich, to zdejmowany kontekst musi być najgłębszym kontekstem w $\mathcal{C}$.

Istnieją zatem dokładnie te same redukcje konfiguracji $(h, [\mathbf{let}\ C\ x = \mathcal{C}[\![E_1]\!]_A\ \mathbf{in}\ E_2] :: \overline{\mathcal{C}})$ oraz $(h, [\mathbf{try}\ \{\mathcal{C}[\![E_1]\!]_A\}\ \mathbf{catch}\ (\mu\ C\ x)\ \{E_2\}] :: \overline{\mathcal{C}})$, w których również zdejmowany jest jeden z kontekstów w $\mathcal{C}$, a zewnętrzny kontekst **let** lub **try** pozostaje niezmieniony. □

6.3. Ewaluacja zależy tylko od zmiennych wolnych

Twierdzenie o zależności ewaluacji od zmiennych wolnych jest kluczowe w dowodzie poprawności dla reguł *-WEAK i HT-FRAME. Mówi ono, że jeśli dwie sterty zgadzają się na lokacjach odpowiadających zmiennym wolnym w pewnym wyrażeniu E, to ewaluacje wyrażenia E na tych dwóch stertach będą w pewnym sensie równoważne.

Równoważnośc ta nie będzie niestety trywialna, bo nowo zaalokowane lokacje na obu sterach mogą się różnić. Zgodnie z semantyką języka, lokacja zwracana przez operator **new** to (maximum z lokacji na stercie) + 1. Stąd, ponieważ nie zakładamy niczego o lokacjach innych niż te odpowiadające zmiennym wolnym, wartość zwracana przez operator **new** może się różnić pomiędzy stertami. Nowo zaalokowane lokacje mogą następnie zostać zapisane w polach obiektów znajdujących się pod lokacjami odpowiadającymi zmiennym wolnym, co oznacza że nawet te obiekty, początkowo równe na obu stertach, mogą zacząć się różnić w czasie ewaluacji.

6.3.1. Izomorfizmy

Żeby obejść ten problem, zdefiniujemy *izomorfizm stert* jako bijekcję między lokacjami na tych stertach, zachowującą `null`, `npe` i kompozycję.

Definicja 6.3.1 (izomorfizm stert)**.**
Niech $h_1, h_2 : \mathsf{Heap}$. Funkcję $f : \mathrm{Dom}(h_1) \cup \{\mathtt{null}\} \to \mathrm{Dom}(h_2) \cup \{\mathtt{null}\}$ nazwiemy izomorfizmem między tymi stertami, jeśli:

1. f jest bijekcją
2. $f(\mathtt{null}) = \mathtt{null}$
3. $f(\mathtt{npe}) = \mathtt{npe}$
4. f zachowuje kompozycję, to znaczy dla dowolnych lokacji l_1, l_2 i pola x zachodzi

$$h_1(l_1) \hookrightarrow x = l_2 \iff h_2(f(l_2)) \hookrightarrow x = f(l_2)$$

Jeśli taka funkcja f istnieje, powiemy że sterty h_1 i h_2 są izomorficzne. □

Ostatecznie będziemy chcieli pokazać, że jeśli wyrażenie E nie zawiera zmiennych wolnych, a sterty h_1, h_2 są równe na wszystkich lokacjach występujących w E, to ewaluacje E na stertach h_1 i h_2 są równoważne z dokładnością do izomorfizmu.

To oznacza, że potrzebujemy mówić o izomorfizmach ewaluacji (czyli ciągów par (sterta, stos wywołań)), a zatem należy zdefiniować także izomorfizmy między stosami wywołań. Służy temu kolejnych kilka definicji.

Definicja 6.3.2 (izomorfizm wyrażeń).
Intuicyjnie, dwa wyrażenia są izomorficzne, jeśli różnią się tylko lokacjami w nich występującymi i istnieje izomorfizm stert, mapujący lokacje z pierwszego z wyrażeń na odpowiadające im lokacje w drugim. Formalnie zdefiniujemy ten izomorfizm przez indukcję po budowie wyrażeń.

Niech f będzie izomorfizmem między dwiema stertami i niech E_1, E_2 będą wyrażeniami Jafun. Powiemy, że f jest izomorfizmem między tymi wyrażeniami, jeśli

1. E_1 i E_2 są wyrażeniami tego samego rodzaju (np. oba są wyrażeniami **new**)
2. f jest izomorfizmem między odpowiadającymi sobie podwyrażeniami E_1 i E_2
3. Dla każdego **this** występującego w E_1, na odpowiadającej mu pozycji w E_2 też jest **this**
4. Dla każdego identyfikatora występującego w E_1, na odpowiadającej mu pozycji w E_2 jest taki sam identyfikator
5. Dla każdej lokacji l w E_1, na odpowiadającej mu pozycji w E_2 jest $f(l)$

□

Definicja 6.3.3 (izomorfizm kontekstów).
Podobnie jak wyżej, definiujemy izomorfizm kontekstów przez indukcję po budowie kontekstu.

Niech f będzie izomorfizmem między dwiema stertami i niech $\mathcal{C}_1, \mathcal{C}_2$ będą kontekstami ewaluacji Jafun. Powiemy, że f jest izomorfizmem między tymi kontekstami, jeśli

- $\mathcal{C}_1 = \mathcal{C}_2 = [\![\ ]\!]_A$

  lub

- $\mathcal{C}_1 = \mathbf{let}\ C\ x = \mathcal{C}_1'\ \mathbf{in}\ E_1, \quad \mathcal{C}_2 = \mathbf{let}\ C\ x = \mathcal{C}_2'\ \mathbf{in}\ E_2$,
  a f jest izomorfizmem między $\mathcal{C}_1'$ i $\mathcal{C}_2'$ oraz między E_1 i E_2

  lub

- $\mathcal{C}_1 = \mathbf{try}\ \{\mathcal{C}_1'\}\ \mathbf{catch}\ (\mu\ C\ x)\ \{E_1\}, \quad \mathcal{C}_2 = \mathbf{try}\ \{\mathcal{C}_2'\}\ \mathbf{catch}\ (\mu\ C\ x)\ \{E_2\}$,
  a f jest izomorfizmem między $\mathcal{C}_1'$ i $\mathcal{C}_2'$ oraz między E_1 i E_2

□

Definicja 6.3.4 (izomorfizm stosów wywołań).
f jest izomorfizmem między dwoma stosami wywołań, jeśli są one równej długości i f jest izomorfizmem między każdą parą odpowiadających sobie kontekstów i redeksów z tych stosów.

□

6.3.2. Zależność ewaluacji od zmiennych wolnych

Sformułuję teraz twierdzenie o zależności ewaluacji od zmiennych wolnych, a następnie udowodnię kilka lematów pomocnych w jego dowodzie.

Twierdzenie 2 (O zależności ewaluacji od zmiennych wolnych).
Niech h będzie spójną stertą, env środowiskiem, a E wyrażeniem Jafun, w którym nie występują konkretne lokacje, a wszystkie zmienne wolne są przez env mapowane na lokacje w h. Niech $h_1, h_2, h_1^{rest}, h_2^{rest}$ będą stertami takimi, że $h_1 = h \oplus h_1^{rest}$ i $h_2 = h \oplus h_2^{rest}$. Wreszcie, niech $confs_1, h_{1,n}, A, l_1$ będą takie, że $(h_1, E[/env]) \overset{confs_1}{\rightsquigarrow} (h_{1,n}, A, l_1)$.

Wtedy istnieją $h_{1,n}^{base}, h_{2,n}^{base}, confs_2, h_{2,n}, l_2, f$, takie że

1. $h_{1,n} = h_{1,n}^{base} \oplus h_1^{rest}$

2. $h_{2,n} = h_{2,n}^{base} \oplus h_2^{rest}$

3. *f jest izomorfizmem między $h_{1,n}^{base}$ i $h_{2,n}^{base}$*

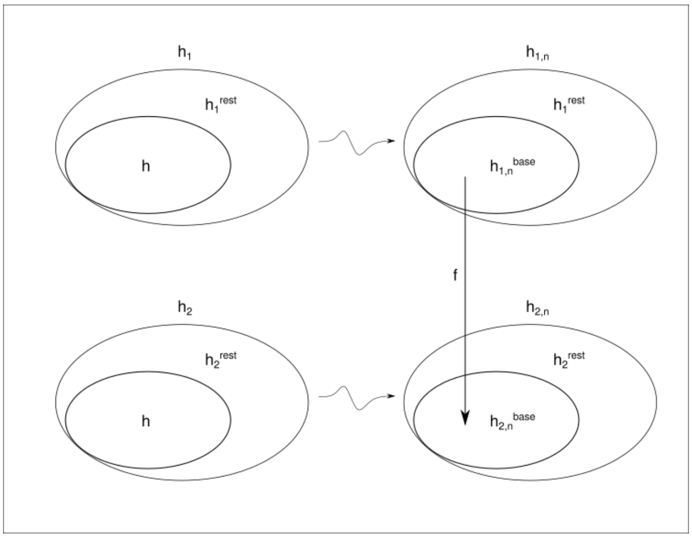
4. *f jest identycznością na lokacjach w env*

5. $f(l_1) = l_2$

6. $(h_2, E[/env]) \overset{confs_2}{\rightsquigarrow} (h_{2,n}, A, l_2)$.

Dowód. Na końcu sekcji. □

Sterty h_1 i h_2 w powyższym twierdzeniu to dwie sterty, o których wiemy, że zgadzają się na wszystkich lokacjach odpowiadających zmiennym wolnym w E (wszystkie one zawierają się w podstercie h). O ich pozostałych fragmentach (odpowiednio h_1^{rest} i h_2^{rest}) nie zakładamy nic. Twierdzenie mówi, że jeśli mamy ewaluację wyrażenia E w środowisku env na stercie h_1, to istnieje też jego analogiczna ewaluacja na stercie h_2, taka że sterty docelowe oraz wyniki ewaluacji są izomorficzne, a fragmenty stert nie mające związku ze zmiennymi wolnymi pozostają niezmienione (Rysunek 6.1).



Rysunek 6.1: Wizualizacja twierdzenia o zależności ewaluacji od zmiennych wolnych

W praktyce oznacza to, że jedynie fragmenty stert odpowiadające zmiennym wolnym w wyrażeniu mają znaczenie dla ewaluacji tego wyrażenia.

Dowód twierdzenia będzie przebiegał przez indukcję po długości ewaluacji *confs₁*. W tym celu jednak musimy sformułować następujący lemat, będący krokiem indukcyjnym w uogólnieniu powyższego twierdzenia na częściowe ewaluacje.

Lemat 2 (O zależności redukcji od zmiennych wolnych)**.**
Niech $\overline{C}_1$ będzie dowolnym stosem wywołań, a $h_1, h_1^{base}, h_1^{rest}$ będą stertami, takimi że h_1^{base} jest spójna , $h_1 = h_1^{base} \oplus h_1^{rest}$, a wszystkie lokacje występujące w wyrażeniach na stosie $\overline{C}_1$ znajdują się na stercie h_1^{base}. Niech teraz $h_{1,n}, \overline{C}_{1,n}$ będą takie, że $(h_1, \overline{C}_1) \to (h_{1,n}, \overline{C}_{1,n})$.

Weźmy teraz dowolną spójną stertę h_2^{base}, stos wywołań $\overline{\mathcal{C}}_2$ oraz izomorfizm f, takie że f jest izomorfizmem między h_1^{base} i h_2^{base} oraz między $\overline{\mathcal{C}}_1$ i $\overline{\mathcal{C}}_2$, zdefiniowanym tylko na lokacjach znajdujących się w h_1^{base}. Jeśli wszystkie lokacje występujące w wyrażeniach na stosie $\overline{\mathcal{C}}_2$ znajdują się na stercie h_2^{base}, to dla dowolnych stert h_2, h_2^{rest}, takich że $h_2 = h_2^{base} \oplus h_2^{rest}$ istnieją sterty $h_{1,n}^{base}, h_{2,n}^{base}, h_{2,n}$, stos $\overline{\mathcal{C}}_{2,n}$ oraz izomorfizm f', takie że

1. *f' rozszerza f*
2. *f' jest zdefiniowane tylko na lokacjach znajdujących się w $h_{1,n}^{base}$*
3. *f' jest izomorfizmem między $h_{1,n}^{base}$ i $h_{2,n}^{base}$ oraz między $\overline{\mathcal{C}}_{1,n}$ i $\overline{\mathcal{C}}_{2,n}$*
4. *$h_{1,n} = h_{1,n}^{base} \oplus h_1^{rest}$*
5. *$h_{2,n} = h_{2,n}^{base} \oplus h_2^{rest}$*
6. *wszystkie lokacje występujące w wyrażeniach na stosie $\overline{\mathcal{C}}_{1,n}$ znajdują się na stercie $h_{1,n}^{base}$*
7. *wszystkie lokacje występujące w wyrażeniach na stosie $\overline{\mathcal{C}}_{2,n}$ znajdują się na stercie $h_{2,n}^{base}$*
8. *$(h_2, \overline{\mathcal{C}}_2) \to (h_{2,n}, \overline{\mathcal{C}}_{2,n})$.*

Dowód. Rozpatrzmy wszystkie możliwe postaci izomorficznych stosów $\overline{\mathcal{C}}_1, \overline{\mathcal{C}}_2$, takich że istnieje redukcja $(h_1, \overline{\mathcal{C}}_1) \to (h_{1,n}, \overline{\mathcal{C}}_{1,n})$.

- $\overline{\mathcal{C}}_1 = \overline{\mathcal{C}}_1' :: \mathcal{C}_1[\![\mathbf{new}\ \mu\ C(l_1, \ldots, l_k)]\!]_\emptyset, \quad \overline{\mathcal{C}}_2 = \overline{\mathcal{C}}_2' :: \mathcal{C}_2[\![\mathbf{new}\ \mu\ C(l_1', \ldots, l_k')]\!]_\emptyset$

  Redukcja $\overline{\mathcal{C}}_1$ jest postaci $(h_1, \overline{\mathcal{C}}_1' :: \mathcal{C}_1[\![\mathbf{new}\ \mu\ C(l_1, \ldots, l_k)]\!]_\emptyset) \to (h_{1,n}, \overline{\mathcal{C}}_1' :: \mathcal{C}_1[\![l_0]\!]_\emptyset)$, gdzie $\mathsf{alloc}(h_1, \overline{\mathcal{C}}_1', C) = (l_0, h)$, $\mathsf{flds}(C) = x_1, \ldots, x_k$, $o = \mathsf{empty}_C\{x_1 \mapsto l_1, \ldots, x_k \mapsto l_k\}$, $h_{1,n} = h\{l_0 \mapsto o\}$. Nowa sterta $h_{1,n}$ jest zatem po prostu stertą h_1 z dodatkowym nowym obiektem o pod nową lokacją l_0. Weźmy zatem $h_{1,n}^{base} = h_1^{base}\{l_0 \mapsto o\}$. Spełnia ona warunki 4 i 6, ponieważ jedyną nową lokacją na stosie jest l_0, które trafiło właśnie do $h_{1,n}^{base}$.

  Weźmy teraz $(l_0', h') = \mathsf{alloc}(h_2, \overline{\mathcal{C}}_2', C)$, $o' = \mathsf{empty}_C\{x_1 \mapsto l_1', \ldots, x_k \mapsto l_k'\}$, $h_{2,n} = h'\{l_0' \mapsto o'\}$. Mamy wtedy $(h_2, \overline{\mathcal{C}}_2) \to (h_{2,n}, \overline{\mathcal{C}}_2' :: \mathcal{C}_2[\![l_0']\!]_\emptyset))$, a zatem biorąc takie $h_{2,n}$ i $\overline{\mathcal{C}}_{2,n} = \overline{\mathcal{C}}_2' :: \mathcal{C}_2[\![l_0']\!]_\emptyset$ mamy spełnione też warunki 7 i 8.

  Jest to jedyny przypadek, w którym na stercie pojawia się nowa lokacja, a zatem należy zmodyfikować izomorfizm f. Niech więc $h_{2,n}^{base} = h_2^{base}\{l_0' \mapsto o'\}$ i $f' = f\{l_0 \mapsto l_0'\}$. Warunki 1 i 5 są w oczywisty sposób spełnione. Warunek 2 jest spełniony, ponieważ z założenia f było zdefiniowane tylko na lokacjach z h_1^{base}, a l_0 trafiło do $h_{1,n}^{base}$. Wreszcie, warunek 3 jest spełniony, ponieważ z założenia o izomorfizmie $\overline{\mathcal{C}}_1$ i $\overline{\mathcal{C}}_2$, dla $j = 1, \ldots, k$ mamy $f'(l_j) = l_j'$, a z definicji f', $f'(l_0) = l_0'$.

- $\overline{\mathcal{C}}_1 = \overline{\mathcal{C}}_1' :: \mathcal{C}_1[\![\mathbf{let}\ C\ x = e_1\ \mathbf{in}\ e_2]\!]_\emptyset, \quad \overline{\mathcal{C}}_2 = \overline{\mathcal{C}}_2' :: \mathcal{C}_2[\![\mathbf{let}\ C\ x = e_1'\ \mathbf{in}\ e_2']\!]_\emptyset$

  Redukcja $\overline{\mathcal{C}}_1$ jest postaci $(h_1, \overline{\mathcal{C}}_1' :: \mathcal{C}_1[\![\mathbf{let}\ C\ x = e_1\ \mathbf{in}\ e_2]\!]_\emptyset) \to (h_1, \overline{\mathcal{C}}_1' :: \mathcal{C}_1[\mathbf{let}\ C\ x = [\![e_1]\!]_\emptyset\ \mathbf{in}\ e_2])$, a zatem jedyne co się w niej dzieje to dodanie e_2 do kontekstu i zmiana redeksu na e_1.

  Możemy więc wziąć $\overline{\mathcal{C}}_{2,n} = \overline{\mathcal{C}}_2' :: \mathcal{C}_2[\mathbf{let}\ C\ x = [\![e_1']\!]_\emptyset\ \mathbf{in}\ e_2'])$, a sterty i izomorfizm f pozostawić bez zmian.

- $\overline{\mathcal{C}}_1 = \overline{\mathcal{C}}_1' :: \mathcal{C}_1[\textbf{let } C\ x = [\![l]\!]_\emptyset \textbf{ in } e], \quad \overline{\mathcal{C}}_2 = \overline{\mathcal{C}}_2' :: \mathcal{C}_2[\textbf{let } C\ x = [\![l']\!]_\emptyset \textbf{ in } e']$

  Redukcja $\overline{\mathcal{C}}_1$ jest postaci $(h_1, \overline{\mathcal{C}}_1' :: \mathcal{C}_1[\textbf{let } C\ x = [\![l]\!]_\emptyset \textbf{ in } e]) \to (h_1, \overline{\mathcal{C}}_1' :: \mathcal{C}_1[\![e\{l/x\}]\!]_\emptyset)$. Podobnie jak wyżej, możemy pozostawić sterty oraz izomorfizm bez zmian i przyjąć $\overline{\mathcal{C}}_{2,n} = (h_2, \overline{\mathcal{C}}_2' :: \mathcal{C}_2[\![e'\{l'/x\}]\!]_\emptyset)$.

  Wyrażenia $e\{l/x\}$ i $e'\{l'/x\}$ są izomorficzne, ponieważ z założenia f jest izomorfizmem między e i e' oraz $f(l) = l'$.

- $\overline{\mathcal{C}}_1 = \overline{\mathcal{C}}_1' :: \mathcal{C}_1[\![\textbf{if } l_0 == l_1 \textbf{ then } e_1 \textbf{ else } e_2]\!]_\emptyset, \quad \overline{\mathcal{C}}_2 = \overline{\mathcal{C}}_2' :: \mathcal{C}_2[\![\textbf{if } l_0' == l_1' \textbf{ then } e_1' \textbf{ else } e_2']\!]_\emptyset$

  Redukcja $\overline{\mathcal{C}}_1$ jest postaci $(h_1, \overline{\mathcal{C}}_1' :: \mathcal{C}_1[\![\textbf{if } l_0 == l_1 \textbf{ then } e_1 \textbf{ else } e_2]\!]_\emptyset) \to (h_1, \overline{\mathcal{C}}_1' :: \mathcal{C}_1[\![e_i]\!]_\emptyset)$, gdzie $i \in \{1, 2\}$ w zależności od tego czy $l_0 = l_1$.

  Ponieważ $l_0' = f(l_0)$ i $l_1' = f(l_1)$, więc $l_0' = l_1'$ wtedy i tylko wtedy gdy $l_0 = l_1$. Redukcja $\overline{\mathcal{C}}_2$ zatem wybierze tę samą gałąź, co redukcja $\overline{\mathcal{C}}_1$. Możemy więc wziąć $\overline{\mathcal{C}}_{2,n} = \overline{\mathcal{C}}_2' :: \mathcal{C}_2[\![e_i']\!]_\emptyset$, a sterty i izomorfizm f pozostawić bez zmian.

- $\overline{\mathcal{C}}_1 = \overline{\mathcal{C}}_1' :: \mathcal{C}_1[\![l.m(\bar{l})]\!]_\emptyset, \quad \overline{\mathcal{C}}_2 = \overline{\mathcal{C}}_2' :: \mathcal{C}_2[\![l'.m(\bar{l'})]\!]_\emptyset, \quad l \neq \textbf{null}$

  Redukcja $\overline{\mathcal{C}}_1$ jest postaci $(h_1, \overline{\mathcal{C}}_1 :: \mathcal{C}_1[\![l.m(\bar{l})]\!]_\emptyset) \to (h_1, \overline{\mathcal{C}}_1 :: \mathcal{C}_1[\![l.m(\bar{l})]\!]_\emptyset :: [\![e]\!]_\emptyset)$, gdzie e jest ciałem metody m z wartościami $\bar{l}$ podstawionymi w miejsce argumentów.

  Niech teraz e' będzie ciałem metody m z wartościami $\bar{l'}$ podstawionymi w miejsce argumentów. Ponieważ listy argumentów $\bar{l}$ i $\bar{l'}$ są izomorficzne, więc wyrażenia e i e' są izomorficzne.

  Możemy więc wziąć $\overline{\mathcal{C}}_{2,n} = \overline{\mathcal{C}}_2 :: \mathcal{C}_2[\![l'.m(\bar{l'})]\!]_\emptyset :: [\![e']\!]_\emptyset$, a sterty i izomorfizm pozostawić bez zmian.

- $\overline{\mathcal{C}}_1 = \overline{\mathcal{C}}_1' :: \mathcal{C}_1[\![l_1.m(\bar{l})]\!]_\emptyset :: [\![l_2]\!]_\emptyset, \quad \overline{\mathcal{C}}_2 = \overline{\mathcal{C}}_2' :: \mathcal{C}_2[\![l_1'.m(\bar{l'})]\!]_\emptyset :: [\![l_2']\!]_\emptyset$

  Redukcja $\overline{\mathcal{C}}_1$ jest postaci $(h_1, \overline{\mathcal{C}}_1' :: \mathcal{C}_1[\![l_1.m(\bar{l})]\!]_\emptyset :: [\![l_2]\!]_\emptyset) \to (h_1, \overline{\mathcal{C}}_1' :: \mathcal{C}_1[\![l_2]\!]_\emptyset)$. Następuje więc jedynie zdjęcie wywołania metody ze stosu wywołań. Możemy zatem wziąć $\overline{\mathcal{C}}_{2,n} = \overline{\mathcal{C}}_2' :: \mathcal{C}_2[\![l_2']\!]_\emptyset$, a sterty i izomorfizm pozostawić bez zmian.

- $\overline{\mathcal{C}}_1 = \overline{\mathcal{C}}_1' :: \mathcal{C}_1[\![l_1.x = l]\!]_\emptyset, \quad \overline{\mathcal{C}}_2 = \overline{\mathcal{C}}_2' :: \mathcal{C}_2[\![l_1'.x = l']\!]_\emptyset, \quad l_1 \neq \textbf{null}$

  Redukcja $\overline{\mathcal{C}}_1$ jest postaci $(h_1, \overline{\mathcal{C}}_1' :: \mathcal{C}_1[\![l_1.x = l]\!]_\emptyset) \to (h_{1,n}, \overline{\mathcal{C}}_1' :: \mathcal{C}_1[\![l]\!]_\emptyset)$, gdzie $o = h_1(l_1)\{x \mapsto l\}$, $h_{1,n} = h_1\{l_1 \mapsto o\}$

  Ponieważ izomorfizm jest różnowartościowy i zachowuje **null**, więc również $l_1' \neq \textbf{null}$. Weźmy zatem $o' = h_2(l_1')\{x \mapsto l'\}$, $h_{2,n} = h_2\{l_1' \mapsto o'\}$ i $h_{2,n}^{base} = h_2^{base}\{l_1' \mapsto o'\}$.

  Możemy teraz wziąć $\overline{\mathcal{C}}_{2,n} = \overline{\mathcal{C}}_2' :: \mathcal{C}_2[\![l']\!]_\emptyset$ i pozostawić izomorfizm f bez zmian.

- $\overline{\mathcal{C}}_1 = \overline{\mathcal{C}}_1' :: \mathcal{C}_1[\![l_1.x]\!]_\emptyset, \quad \overline{\mathcal{C}}_2 = \overline{\mathcal{C}}_2' :: \mathcal{C}_2[\![l_1'.x]\!]_\emptyset, \quad l \neq \textbf{null}$

  Redukcja $\overline{\mathcal{C}}_1$ jest postaci $(h_1, \overline{\mathcal{C}}_1' :: \mathcal{C}_1[\![l_1.x]\!]_\emptyset) \to (h_1, \overline{\mathcal{C}}_1' :: \mathcal{C}[\![l_2]\!]_\emptyset)$, gdzie $l_2 = h_1(l_1)(x)$.

  Ponieważ sterty h_1 i h_2 są izomorficzne, więc na stercie h_2 istnieje obiekt pod lokacją l_1' zawierający pole x i, co więcej, jeśli $l_2' = h_2(l_1')(x)$, to $f(l_2) = l_2'$.

  Zatem możemy wziąć $\overline{\mathcal{C}}_{2,n} = \overline{\mathcal{C}}_2' :: \mathcal{C}[\![l_2']\!]_\emptyset$, a sterty i izomorfizm pozostawić bez zmian.

- $\overline{\mathcal{C}}_1 = \overline{\mathcal{C}}_1' :: \mathcal{C}_1[\![\textbf{throw } l]\!]_\emptyset, \quad \overline{\mathcal{C}}_2 = \overline{\mathcal{C}}_2' :: \mathcal{C}_2[\![\textbf{throw } l']\!]_\emptyset, \quad l \neq \textbf{null}$

  Redukcja $\overline{\mathcal{C}}_1$ jest postaci $(h_1, \overline{\mathcal{C}}_1' :: \mathcal{C}_1[\![\textbf{throw } l]\!]_\emptyset) \to (h_1, \overline{\mathcal{C}}_1' :: \mathcal{C}_1[\![l]\!]_D)$, gdzie $\mathsf{class}(h_1, l) = D$, zatem jedynym efektem jest zmiana trybu wykonania na wyjątkowy z wartością wyjątku l i typem D. Możemy więc po prostu wziąć $\overline{\mathcal{C}}_{2,n} = \overline{\mathcal{C}}_2' :: \mathcal{C}_2[\![l']\!]_D$, a sterty i izomorfizm pozostawić bez zmian.

- $\overline{\mathcal{C}}_1 = \overline{\mathcal{C}}'_1 :: \mathcal{C}_1[\![\mathbf{try}\ \{e_1\}\ \mathbf{catch}\ (\mu\ C\ x)\ \{e_2\}]\!]_\emptyset, \quad \overline{\mathcal{C}}_2 = \overline{\mathcal{C}}'_2 :: \mathcal{C}_2[\![\mathbf{try}\ \{e'_1\}\ \mathbf{catch}\ (\mu\ C\ x)\ \{e'_2\}]\!]_\emptyset$

  Redukcja $\overline{\mathcal{C}}_1$ jest postaci $(h_1, \overline{\mathcal{C}}'_1 :: \mathcal{C}_1[\![\mathbf{try}\ \{e_1\}\ \mathbf{catch}\ (\mu\ C\ x)\ \{e_2\}]\!]_\emptyset) \to (h_1, \overline{\mathcal{C}}'_1 :: \mathcal{C}_1[\mathbf{try}\ \{[\![e_1]\!]_\emptyset\}\ \mathbf{catch}\ (\mu\ C\ x)\ \{e_2\}])$, a zatem przebiega podobnie, jak analogiczna redukcja dla **let** – do kontekstu dodawane jest wyrażenie e_2, a redeks zostaje zmieniony na e_1. Podobnie jak dla **let**, możemy wziąć $\overline{\mathcal{C}}_{2,n} = \overline{\mathcal{C}}'_2 :: \mathcal{C}_2[\mathbf{try}\ \{[\![e'_1]\!]_\emptyset\}\ \mathbf{catch}\ (\mu\ C\ x)\ \{e'_2\}])$, a sterty i izomorfizm pozostawić bez zmian.

- $\overline{\mathcal{C}}_1 = \overline{\mathcal{C}}'_1 :: \mathcal{C}_1[\mathbf{try}\ \{[\![l]\!]_\emptyset\}\ \mathbf{catch}\ (\mu\ C\ x)\ \{e_2\}], \quad \overline{\mathcal{C}}_2 = \overline{\mathcal{C}}'_2 :: \mathcal{C}_2[\mathbf{try}\ \{[\![l']\!]_\emptyset\}\ \mathbf{catch}\ (\mu\ C\ x)\ \{e'_2\}]$

  Przy normalnym wykonaniu wyrażenia wewnątrz blocku **try**, jest on po prostu usuwany z kontekstu. Redukcja $\overline{\mathcal{C}}_1$ jest wtedy postaci $(h_1, \overline{\mathcal{C}}'_1 :: \mathcal{C}_1[\mathbf{try}\ \{[\![l]\!]_\emptyset\}\ \mathbf{catch}\ (\mu\ C\ x)\ \{e_2\}]) \to (h_1, \overline{\mathcal{C}}'_1 :: \mathcal{C}_1[\![l]\!]_\emptyset)$.

  Wystarczy zatem wziąć $\overline{\mathcal{C}}_{2,n} = \overline{\mathcal{C}}'_2 :: \mathcal{C}_2[\![l']\!]_\emptyset$, a sterty i izomorfizm pozostawić bez zmian.

- $\overline{\mathcal{C}}_1 = \overline{\mathcal{C}}'_1 :: \mathcal{C}_1[\mathbf{try}\ \{[\![l]\!]_{C'}\}\ \mathbf{catch}\ (\mu\ C\ x)\ \{e_2\}], \quad \overline{\mathcal{C}}_2 = \overline{\mathcal{C}}'_2 :: \mathcal{C}_2[\mathbf{try}\ \{[\![l']\!]_{C'}\}\ \mathbf{catch}\ (\mu\ C\ x)\ \{e'_2\}]$, gdzie $C' \leq: C$

  Redukcja w tym przypadku oznacza złapanie rzuconego wcześniej wyjątku i obsłużenie go przez e_2. Jest ona postaci $(h_1, \overline{\mathcal{C}}'_1 :: \mathcal{C}_1[\mathbf{try}\ \{[\![l]\!]_{C'}\}\ \mathbf{catch}\ (\mu\ C\ x)\ \{e_2\}]) \to (h_1, \overline{\mathcal{C}}'_1 :: \mathcal{C}_1[\![e]\!]_\emptyset)$, gdzie $e = e_2\{l/x\}$.

  Niech $e' = e'_2\{l'/x\}$. Wtedy wystarczy wziąć $\overline{\mathcal{C}}_{2,n} = \overline{\mathcal{C}}'_2 :: \mathcal{C}_2[\![e']\!]_\emptyset$, a sterty i izomorfizm pozostawić bez zmian.

- $\overline{\mathcal{C}}_1 = \overline{\mathcal{C}}'_1 :: \mathcal{C}_1[\mathbf{try}\ \{[\![l]\!]_{C'}\}\ \mathbf{catch}\ (\mu\ C\ x)\ \{e_2\}], \quad \overline{\mathcal{C}}_2 = \overline{\mathcal{C}}'_2 :: \mathcal{C}_2[\mathbf{try}\ \{[\![l']\!]_{C'}\}\ \mathbf{catch}\ (\mu\ C\ x)\ \{e'_2\}]$, gdzie $C' \neq \emptyset, C' \not\leq: C$

  W przypadku niezłapanie wyjątku, kontekst **try** jest usuwany, a wyjątek przekazywany dalej. Redukcja jest postaci $(h_1, \overline{\mathcal{C}}'_1 :: \mathcal{C}_1[\mathbf{try}\ \{[\![l]\!]_{C'}\}\ \mathbf{catch}\ (\mu\ C\ x)\ \{e_2\}]) \to (h_1, \overline{\mathcal{C}}'_1 :: \mathcal{C}_1[\![l]\!]_{C'})$. Bierzemy zatem $\overline{\mathcal{C}}_{2,n} = \overline{\mathcal{C}}'_2 :: \mathcal{C}_2[\![l']\!]_{C'}$, a sterty i izomorfizm pozostawiamy bez zmian.

- $\overline{\mathcal{C}}_1 = \overline{\mathcal{C}}'_1 :: \mathcal{C}_1[\mathbf{let}\ C\ x = [\![l]\!]_{C'}\ \mathbf{in}\ e], \quad \overline{\mathcal{C}}_2 = \overline{\mathcal{C}}'_2 :: \mathcal{C}_2[\mathbf{let}\ C\ x = [\![l']\!]_{C'}\ \mathbf{in}\ e'], \quad C' \neq \emptyset$

  Przypadek wyjątku w czasie ewaluacji wyrażenia wewnętrzego **let** jest analogiczny jak poprzedni. Możemy wziąć $\overline{\mathcal{C}}_{2,n} = \overline{\mathcal{C}}'_2 :: \mathcal{C}_2[\![l']\!]_{C'}$, a sterty i izomorfizm zostawić bez zmian.

- $\overline{\mathcal{C}}_1 = \overline{\mathcal{C}}'_1 :: \mathcal{C}_1[\![l_1.m(\bar{l})]\!]_\emptyset :: [\![l_2]\!]_C, \quad \overline{\mathcal{C}}_2 = \overline{\mathcal{C}}'_2 :: \mathcal{C}_2[\![l'_1.m(\bar{l}')]\!]_\emptyset :: [\![l'_2]\!]_C, \quad C \neq \emptyset$

  Podobnie w przypadku wyjątku podczas wykonywania metody. Wywołanie jest zdejmowane ze stosu, a wyjątek przekazywany wyżej. Redukcja jest postaci $(h_1, \overline{\mathcal{C}}'_1 :: \mathcal{C}_1[\![l_1.m(\bar{l})]\!]_\emptyset :: [\![l_2]\!]_C) \to (h_1, \overline{\mathcal{C}}'_1 :: [\![l_2]\!]_C)$.

  Bierzemy $\overline{\mathcal{C}}_{2,n} = (h_1, \overline{\mathcal{C}}'_2 :: [\![l'_2]\!]_C)$, a sterty i izomorfizm pozostawiamy bez zmian.

- Redukcja $\overline{\mathcal{C}}_1$ powoduje odwołanie do lokacji **null**

  Jest tak, gdy $\overline{\mathcal{C}}_1 = \overline{\mathcal{C}}'_1 :: \mathcal{C}_1[\![e]\!]_\emptyset$, gdzie e jest postaci **null**.x, **null**.$x = l$, **null**.$m(\bar{l})$ lub **throw null**.

  Redukcja $\overline{\mathcal{C}}_1$ jest wtedy postaci $(h_1, \overline{\mathcal{C}}'_1 :: \mathcal{C}_1[\![e]\!]_\emptyset) \to (h_1, \overline{\mathcal{C}}'_1 :: \mathcal{C}_1[\![\mathsf{npe}]\!]_{\mathtt{NPE}})$.

  Wówczas $\overline{\mathcal{C}}_2 = \overline{\mathcal{C}}'_2 :: \mathcal{C}_2[\![e']\!]_\emptyset$, gdzie e' jest izomorficzne z e, a ponieważ izomorfizm zachowuje **null** więc redukcja e' również powoduje odwołanie do **null**.

  Ponieważ izomorfizm zachowuje także npe, możemy wziąć $\overline{\mathcal{C}}_{2,n} = \overline{\mathcal{C}}'_2 :: \mathcal{C}_2[\![\mathsf{npe}]\!]_{\mathtt{NPE}}$, a sterty i izomorfizm pozostawić bez zmian.

$\square$

Korzystając z tego lematu w kroku indukcyjnym udowodnimy teraz następujący lemat, mówiący o ewaluacjach dowolnej długości.

Lemat 3 (O zależności częsciowej ewaluacji od zmiennych wolnych).
Niech $\overline{\mathcal{C}}_1$ będzie dowolnym stosem wywołań, a $h_1, h_1^{base}, h_1^{rest}$ będą stertami, takimi że h_1^{base} jest spójna , $h_1 = h_1^{base} \oplus h_1^{rest}$, a wszystkie lokacje występujące w wyrażeniach na stosie $\overline{\mathcal{C}}_1$ znajdują się na stercie h_1^{base}. Niech teraz $h_{1,n}, \overline{\mathcal{C}}_{1,n}, confs_1$ będą takie, że $(h_1, \overline{\mathcal{C}}_1) \overset{confs_1}{\rightsquigarrow} (h_{1,n}, \overline{\mathcal{C}}_{1,n})$.

Weźmy teraz dowolną spójną stertę h_2^{base}, stos wywołań $\overline{\mathcal{C}}_2$ oraz izomorfizm f, takie że f jest izomorfizmem między h_1^{base} i h_2^{base} oraz między $\overline{\mathcal{C}}_1$ i $\overline{\mathcal{C}}_2$, zdefiniowanym tylko na lokacjach znajdujących się w h_1^{base}. Jeśli wszystkie lokacje występujące w wyrażeniach na stosie $\overline{\mathcal{C}}_2$ znajdują się na stercie h_2^{base}, to dla dowolnych stert h_2, h_2^{rest}, takich że $h_2 = h_2^{base} \oplus h_2^{rest}$ istnieją sterty $h_{1,n}^{base}, h_{2,n}^{base}, h_{2,n}$, stos $\overline{\mathcal{C}}_{2,n}$, izomorfizm f' oraz ciąg konfiguracji $confs_2$, takie że

1. *f' rozszerza f*
2. *f' jest zdefiniowane tylko na lokacjach znajdujących się w $h_{1,n}^{base}$*
3. *f' jest izomorfizmem między $h_{1,n}^{base}$ i $h_{2,n}^{base}$ oraz między $\overline{\mathcal{C}}_{1,n}$ i $\overline{\mathcal{C}}_{2,n}$*
4. *$h_{1,n} = h_{1,n}^{base} \oplus h_1^{rest}$*
5. *$h_{2,n} = h_{2,n}^{base} \oplus h_2^{rest}$*
6. *wszystkie lokacje występujące w wyrażeniach na stosie $\overline{\mathcal{C}}_{1,n}$ znajdują się na stercie $h_{1,n}^{base}$*
7. *wszystkie lokacje występujące w wyrażeniach na stosie $\overline{\mathcal{C}}_{2,n}$ znajdują się na stercie $h_{2,n}^{base}$*
8. *$(h_2, \overline{\mathcal{C}}_2) \overset{confs_2}{\rightsquigarrow} (h_{2,n}, \overline{\mathcal{C}}_{2,n})$.*

Dowód. Przez indukcję po długości ciągu konfiguracji $confs_1$.

- Jeśli $confs_1 = [\,]$ jest pustym ciągiem, to znaczy że $h_{1,n} = h_1$ oraz $\overline{\mathcal{C}}_{1,n} = \overline{\mathcal{C}}_1$. Możemy wtedy wziąć $h_{1,n}^{base} = h_1^{base}$, $h_{2,n}^{base} = h_2^{base}$, $h_{2,n} = h_2$, $\overline{\mathcal{C}}_{2,n} = \overline{\mathcal{C}}_2$, $f' = f$ oraz $confs_2 = [\,]$.
- Jeśli $confs_1 = confs'_1 :: (h_{1,n-1}, \overline{\mathcal{C}}_{1,n-1})$, to Z założenia indukcyjnego będziemy mieli sterty $h_{1,n-1}^{base}, h_{2,n-1}^{base}, h_{2,n-1}$, stos $\overline{\mathcal{C}}_{2,n-1}$, izomorfizm f' oraz ciąg konfiguracji $confs'_2$, takie że spełnione są warunki z treści zadania, w szczególności $(h_2, \overline{\mathcal{C}}_2) \overset{confs'_2}{\rightsquigarrow} (h_{2,n-1}, \overline{\mathcal{C}}_{2,n-1})$. Z lematu 2 dostaniemy sterty $h_{1,n}^{base}, h_{2,n}^{base}, h_{2,n}$, stos $\overline{\mathcal{C}}_{2,n}$ oraz izomorfizm f'', takie że spełnione są warunki 1-7 oraz $(h_{2,n-1}, \overline{\mathcal{C}}_{2,n-1}) \to (h_{2,n}, \overline{\mathcal{C}}_{2,n})$. Składając tę redukcję z ewaluacją z założenia indukcyjnego dostaniemy, że w istocie $(h_2, \overline{\mathcal{C}}_2) \overset{confs_2}{\rightsquigarrow} (h_{2,n}, \overline{\mathcal{C}}_{2,n})$.

$\square$

Możemy wreszcie udowodnić twierdzenie 2

Dowód. (twierdzenia o zależności ewaluacji od zmiennych wolnych) Niech h, env, E, h_1, h_2, h_1^{rest}, h_2^{rest}, $confs_1, h_{1,n}, A, l_1$ będą jak w treści twierdzenia. Pokażę, że istnieją istnieją $h_{1,n}^{base}, h_{2,n}^{base}$, $confs_2, h_{2,n}, l_2, f$, spełniające warunki z treści.

W tym celu zastosujemy lemat 3 dla $h_1^{base} = h_2^{base} = h$, ewaluacji $(h_1, [\![E[/env]]\!]_\emptyset) \overset{confs_1}{\rightsquigarrow} (h_{1,n}, [\![l_1]\!]_A)$ i $\overline{\mathcal{C}}_2 = [\![E[/env]]\!]_\emptyset$. Żeby jednak móc go zastosować, musimy pokazać że wszystkie lokacje z wyrażenia $E[/env]$ występują na stercie h. Wynika to jednak wprost z założeń, że w wyrażeniu E nie ma konkretnych lokacji, a wszystkie zmienne wolne mapowane są przez env na lokacje z h. Stąd, po podstawieniu za zmienne wolne lokacji z env, założenie jest spełnione.

Ostatnim problemem jest wybranie odpowiedniego początkowego f. Wystarczy jednak, żeby f była identycznością na **null**, npe i lokacjach występujących w h lub $E[/env]$. Ponieważ h jest spójna, więc tak zdefiniowana f jest w oczywisty sposób automorfizmem na h oraz na $E[/env]$. □

Rozdział 7

Poprawność

Głównym twierdzeniem, dowodzonym w ramach niniejszej pracy, jest poprawność prezentowanej logiki. Mówi ono, że jeśli możemy udowodnić pewne wynikanie $\Gamma|P \vdash Q$, to każda sterta spełniająca P będzie również spełniała Q. Dla ścisłego sformułowania tego twierdzenia potrzebujemy jednak kilku dodatkowych założeń o stertach, o których chcemy wnioskować. Będziemy rozważać jedynie spójne sterty, a ponieważ język Jafun jest silnie typowany, więc potrzebne jest jeszcze założenie o odpowiednim otypowaniu obiektów na stercie.

Powiemy więc, że środowisko *env* zgadza się ze środowiskiem typów Γ na stercie h, jeśli mają te same dziedziny i typ każdej zmiennej w Γ zgadza się z jej typem na stercie h. To znaczy, dla każdego x, jesli $\Gamma \vdash x : C$, to obiekt $h(env(x))$ jest typu C.

Możemy teraz ściśle sformułować twierdzenie o poprawności.

Twierdzenie 3 (O poprawności logiki separacji dla języka Jafun)**.**
Niech P, Q będą dowolnymi termami logiki separacji, takimi że $\Gamma|P \vdash Q$. Wtedy dla każdej spójnej sterty h i środowiska env, takich że env zgadza się z Γ na stercie h, jeśli $[\![P]\!]_{h,env} = \top$, to również $[\![Q]\!]_{h,env} = \top$. □

Dowód tego twierdzenia przebiega przez indukcję po budowie drzewa dowodu dla $\Gamma|P \vdash Q$. Najpierw należy pokazać, poprawność przypadków bazowych, czyli udowodnić tezę twierdzenia dla wszystkich reguł niezawierających osądów w przesłankach. Następnie należy pokazać, że wszystkie pozostałe reguły zachowują poprawność. W tym celu będziemy zakładać, że wszystkie osądy z przesłanek są poprawne, a następnie korzystając z tych założeń będziemy dowodzić poprawność osądu będącego wnioskiem reguły. Takie założenie jest możliwe dzięki temu, że drzewa dowodów przesłanek są właściwymi poddrzewami dowodu dla wniosku, a zatem założenie o poprawności przesłanek jest założeniem indukcyjnym.

Pełen dowód tego twierdzenia w systemie Coq znajduje się w pliku `JaIrisSoundeness.v` jako dowód twierdzenia `JFIOuterSoundness`.

7.1. Poprawność reguł dla tradycyjnych operatorów logicznych

Dowody poprawności reguł przedstawionych na rysunku 4.1 są do siebie bardzo podobne i, oprócz kilku szczegółów techniczych, mało interesujące, jako że wprost wynikają z własności operatorów logicznych. Na przykład, poprawność reguły Asm jest oczywista – każda sterta spełniająca P spełnia P.

Dowód poprawności innych reguł jednak wymaga doprecyzowania. Poprawność reguły Trans wymaga pokazania, że zmienna v istnieje w środowisku *env*. Jest to prawdą, ponieważ z faktu, że v jest zmienną wolną w osądzie wynika, że v ma przypisany typ w środowisku typów Γ. Istnienie v w środowisku env wynika więc z założenia o zgodności env i Γ

Kolejną regułą, której poprawność jest nieoczywista, jest reguła $\exists$I, wprowadzająca kwantyfikator $\exists$. Należy tu pokazać, że, zakładając poprawność przesłanki, każda sterta spełniająca Q spełnia też $\exists x : C.P$. Kluczową rolę gra tutaj wartość v, którą podstawiamy za zmienną x. Należy rozpatrzeć przypadki, w których v jest równa **null**, **this** lub jest zmienną. Następnie możemy dodać do środowiska mapowanie x na odpowiednio **null**, $env(\mathbf{this})$ lub $env(v)$. Powstałe środowisko jest wtedy zgodne ze środowiskiem $(\Gamma, x : C)$ na stercie h, a zatem można po prostu zastosować założenie indukcyjne.

Wreszcie, najbardziej najbardziej skomplikowanym przypadkiem, spośród reguł dla klasycznych operatorów, jest poprawność reguły $\exists$E, eliminującej kwantyfikator $\exists$.

Kluczowym krokiem w dowodzie poprawności $\exists$E jest zauważenie, że dodanie świeżej, nie będącej zmienną wolną w termie, zmiennej do środowiska nie zmienia spełnialności tego termu przez stertę. To znaczy jeśli $x \notin FV(P)$, to $[\![P]\!]_{h,env} = [\![P]\!]_{h,env[x \mapsto l]}$. Formalny dowód tego faktu wymaga indukcji po budowie termu P, ale intuicyjnie jest to prawda, ponieważ skoro x nie jest wolne w P, więc P nie ma możliwości odwoływania się do niego, a zatem podczas obliczania semantyki P w środowisku env nigdy nie odwołamy się do lokacji $env(x)$. Nie ma zatem znaczenia, czy x występuje w środowisku env i, jeśli tak, to jaka jest wartość $env(x)$.

Kiedy udowodnimy ten fakt, reszta dowodu przebiega już w prosty sposób: mamy stertę h oraz środowisko env zgadzające się na h z Γ, takie że h spełnia term R w środowisku env. Skoro tak, to z założenia indukcyjnego dla pierwszej przesłanki mamy, że h spełnia też w tym środowisku term $\exists x : C.P$, a zatem istnieje taka lokacja l, że $h(l)$ jest typu C i h spełnia P w środowisku $env[x \mapsto l]$.

Możemy więc teraz zastosować założenie indukcyjne dla drugiej przesłanki. Jak zauważyliśmy przed chwilą, $env[x \mapsto l]$ zgadza się z $(\Gamma, x : C)$ na stercie h oraz h spełnia P w tym środowisku. Pozostaje pokazać, że h spełnia R w tym środowisku. W tym momencie korzystamy z obserwacji o zachowaniu spełniania przy dodaniu świeżej zmiennej. Wiemy, że h spełnia R w środowisku env, oraz że x jest świeże w R (bo x nie jest w Γ, a wszystkie zmienne wolne z R są). h spełnia więc też R w środowisku $env[x \mapsto l]$, a zatem z założenia indukcyjnego dla drugiej przesłanki, h spełnia Q w środowisku $env[x \mapsto l]$.

Możemy teraz ponownie skorzystać z faktu o świeżych zmiennych, bo, podobnie jak w R, x jest świeże w Q. Ostatecznie dostajemy więc, że h spełnia Q w środowisku env, a zatem reguła $\exists$E jest poprawna.

7.2. Poprawność reguł dla operatorów separacji

Poprawność reguł Sep-assoc i Sep-sym wynika w prosty sposób z własności sterty, takich jak przemienność i łączność operatora $\oplus$ oraz faktu że suma spójnych stert jest spójna. Poprawność pozostałych reguł jest nieco bardziej skomplikowanym zagadnieniem.

7.2.1. Reguły eliminacji i wprowadzania operatorów separacji

Dowód poprawności eliminacji i wprowadzania operatorów separacji nie jest skompliowany, ale kryje się w nim kilka subtelności związanych z manipulowaniem środowiskami, tak żeby było możliwe skorzystanie z nich używając założeń indukcyjnych. Pokażę je na przykładzie reguły $*$I, ale podobne rozumowanie jest też stosowane w pozostałych regułach.

Aby pokazać poprawność tej reguły, bierzemy dowolne spójne h spełniające $P_1 * P_2$ i środowisko env zgadzające się z (Γ_1, Γ_2) na h. Ponieważ h spełnia $P_1 * P_2$, więc istnieje podział

h na rozłączne sterty h_1 i h_2, spełniające odpowiednio P_1 i P_2. Chcemy teraz skorzystać z założeń indukcyjnych, żeby otrzymać, że spełniają one też odpowiednio Q_1 i Q_2, co zakończy dowód.

Nie możemy jednak zrobić tego używając środowiska *env*, bo nie musi ono zgadzać się z Γ_1 ani Γ_2 – jest na to zbyt duże. O ile oba ze środowisk typów są niepuste, to *env* zawiera zarówno zmienne niebędące w Γ_1, jak i zmienne niebędące w Γ_2. Należy więc ostrożenie wziąc podzbiór środowiska *env*, zgodny z Γ_1 na h_1 i drugi, zgodny z Γ_2 na h_2 w taki sposób, żeby zachować spełnianie termów przez sterty h_1 i h_2 w tych okrojonych środowiskach.

Aby to osiągnąć, wystarczy wziąć env_1 i env_2 będące obcięciami środowiska *env* odpowiednio do zmiennych znajdujących się w Γ_1 i Γ_2. Wtedy oczywiście env_1 zgadza się z Γ_1 na h_1 i analogicznie dla env_2. Pozostaje więc pokazać, że h_1 spełnia P_1 w środowisku env_1. W tym celu możemy znów skorzystać z faktu, że dodawanie świeżych zmiennych do środowiska nie wpływa na spełnianie termu przez stertę. Ponieważ wszystkie zmienne wolne w P_1 znajdują się w Γ_1, więc znajdują się też w env_1. Możemy więc otrzymać *env* przez dodawanie do env_1 kolejnych zmiennych, z których każda jest świeża w P_1, a zatem dodanie żadnej z nich nie zmienia spełniania P_1 przez h_1. Stąd mamy więc $[\![P1]\!]_{h_1,env} = [\![P1]\!]_{h_1,env_1}$ i podobnie $[\![Q1]\!]_{h_1,env} = [\![Q1]\!]_{h_1,env_1}$.

Korzystając więc z powyższych równości i z założenia indukcyjnego, dostajemy że h_1 spełnia Q_1 w środowisku *env*. Przeprowadzenie analogicznego rozumowania dla h_2 kończy dowód.

7.2.2. Reguła osłabiania

Reguła osłabiania *-Weak jest jedną z dwóch reguł, obok reguły Ht-frame, dzięki którym operatory separacji są tak przydatne. Sprawia ona, że logika separacji jest afiniczna, to znaczy że jeśli sterta spełnia pewien term, to każda jej spójna nadsterta też go spełnia.

Istotnie, skoro mamy $P_1 * P_2 \vdash P_1$ dla dowolnych P_1, P_2, to po przyjęciu $P_2 = \texttt{True}$ możemy w pewnym sensie zapomnieć o niepasującej nam części sterty. Jeśli mamy więc stertę h_1 spełniającą term P_1, to dla dowolnej sterty h_2 mamy $h_1 \oplus h_2$ spełnia $P_1 * \texttt{True}$, a zatem, korzystając z reguły osłabiania, także $h_1 \oplus h_2$ spełnia P_1.

Jest to niewątpliwie silna własność logiki, jednak zachować jej poprawność przy zadanej semantyce języka, niezbędne były pewne kompromisy. Przede wszystkim, logika ta nie zawiera kwantyfikatora ogólnego $\forall$. Przy naszej definicji sterty kwantyfikator ten zaburzyłby poprawność reguły osłabiania. Bez dodatkowych założeń nie ma bowiem gwarancji, że $h_1 \oplus h_2$ spełnia $\forall x.P$, nawet jeśli samo h_1 spełnia $\forall x.P$. Podczas rozszerzania sterty moglibyśmy dodać nowe lokacje, niespełniające P, i w ten sposób zepsuć poprawność tej reguły.

Jednocześnie, nięzbędne jest też zadbanie, żeby nie było możliwe symulowanie kwantyfikatora ogólnego przy pomocy innych kontrukcji logiki. Możliwe byłoby to na przykład przez zastąpienie termu $\forall x : C.P$ przez $\neg\exists x : C.\neg P$. Z tego powodu kwantyfikator $\exists$ został w logice ograniczony do termów najwyższego poziomy, tak że niemożliwe staje się jego zanegowanie.

Te dwa ograniczenia wystarczają do zapewnienia poprawności reguły osłabiania przy zachowaniu semantyki języka bez zmian.

Wciąż jednak, pomimo tych ograniczeń, dowód poprawności reguły osłabiania nie jest trywialny. Wynika ona wprost z pomocniczego twierdzenia, mówiącego że jeśli pewne środowisko mapuje wszystkie zmienne wolne termu na lokacje na pewnej stercie, to możemy dowolnie rozszerzać zarówno środowisko jak i stertę bez zmiany semantyki tego termu, to znaczy jeśli $h_1 \subseteq h$, $env_1 \subseteq env$ i wszystkie zmienne wolne z termu P mapowane są przez env_1 na lokacje w h_1, to $[\![P]\!]_{h_1,env_1} = [\![P]\!]_{h,env}$.

Twierdzenie to, mimo że wygląda podobnie to przytaczanego wcześniej faktu o rozszerzaniu środowiska o świeże zmienne, jest od niego istotnie trudniejsze do udowodnienia, ponieważ tutaj chcemy zagwarantować także możliwość rozszerzania sterty. Powoduje to utrudnienia w dowodzie dla przypadku trójki Hoare'a. W pozostałych przypadkach dowód nadal nie jest trudny. Dzięki temu, że operator $*$ może występować jedynie w wyrażeniach wewnętrznych, nie musimy przejmować się kwantyfikatorem $\exists$.

W przypadku trójek Hoare'a jednak musimy skorzystać z udowodnionego wcześniej twierdzenia 2 (O zależności ewaluacji od zmiennych wolnych). Zapewnia nam ono, że dla dowolnej ewaluacji na stercie h_1 będziemy mieli odpowiednią ewaluację na stercie h i, co więcej, otrzymane na końcu sterty będą izomorficzne. Pozostaje więc pokazać, że dla semantyka termu dla izomorficznych stert jest taka sama. Jest to prawda tylko przy założeniu, że środowiska, w których rozpatrujemy semantykę, również są izomorficzne, przynajmniej na zmiennych wolnych w tym termie. Na szczęście jednak twierdzenie 2 zapewnia nam, że otrzymany izomorfizm jest identycznością na lokacjach przypisanych w środowisku, a zatem wyjściowe środowiska env i env_1 spełniają to założenie.

Rozdział 8

Formalizacja w systemie Coq

Rozdział 9

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