VCLANG

Валерий Исаев

1. Vccore

Язык vclang состоит из двух частей: ядра (vccore) и фронтенд (vclang). Пользователь пишет код на vclang, и он транслируется в vccore. Как это происходит мы обсудим позже, в этом разделе мы опишем ядро.

Программа на vccore состоит из множества объявлений (то есть *неупорядоченного* списка). Каждое объявление состоит из уникального имени и *определения*, которые бывают трех видов:

- *Функция* содержит тело и сигнатуру, состоящую из списка типов аргументов и типа возвращаемого значения.
- *Алгебраический тип данных* содержит вселенную в которой он лежит, список типов, описывающий параметры типа данных, и множество конструкторов, каждый из которых состоит из уникального имени и списка типов, описывающего параметры конструктора.
- *Класс* содержит список полей. Каждое поле состоит из уникального имени и типа.

Классы в vccore устроенны очень просто, по сути, они являются просто записями. Классы не могут наследоваться друг от друга, но можно использовать анонимное наследование. Если A – некоторый класс, содержащий поля $f_1, \dots f_n$, и $d_1, \dots d_n$ – некоторые термы, то $A\ \{\ f_1 \Rightarrow d_1;\ \dots\ f_n \Rightarrow d_n\ \}$ – анонимный наследник класса A.

Если $A\ \{\ f_1 \Rightarrow d_1;\ \dots\ f_n \Rightarrow d_n\ \}$ – коректный тип и все поля A присутствуют в списке $f_1, \dots f_n$, то его элементы создаются при помощи конструкции **new**:

$$\mathbf{new}\ A\ \{\ f_1 \Rightarrow d_1,\ \dots\ f_n \Rightarrow d_n\ \} : A\ \{\ f_1 \Rightarrow d_1,\ \dots\ f_n \Rightarrow d_n\ \}$$

1.1. Формальное определение термов vccore.

Как обычно, мы сначала опишем множество сырых термов, после чего определим отношения типизации. Выражения и типы vccore тогда будут определятся как типизируемые термы.

При определении функции или связывания в **let** мы можем использовать один из двух способов их описания: либо $f\, x_1\ \dots\ x_n \Rightarrow e$, либо $f\, x_1\ \dots\ x_n \Leftarrow e$. Множество $\{\Leftarrow, \Rightarrow\}$ мы будем обозначать Def. Какой из элементов множества Def используется при определении функции влияет на то, как будет вычисляться эта функция. При выборе варианта $\Leftarrow$, мы можем использовать дополнительные конструкции для задания тела функции. Множество термов $Term$ будет определено ниже. Используя его, можно определить множество $Term_e$ индуктивным образом:

- Если $a \in Term$, то $a \in Term_e$.
- Пусть $k \in \mathbb{N}$, p_i – шаблоны, $d_i \in Def$, $b_i \in Term_e$ и $b_i \in Term$, если $d_i = \Longrightarrow$. Тогда $\mathbf{elim}\, v_k\, \{p_1\, d_1\, b_1; \dots p_n\, d_n\, b_n\} \in Term_e$.

Шаблон – это либо имя конструктора, либо символ var.

Множество термов определяется следующим образом:

$$\begin{aligned} Term := {} & v_i \mid D \mid f \mid D\, a_1 \ldots a_n \,.\, c \\ & \mid \mathbf{new}\, C\, \{S\} \mid C\, \{S\} \\ & \mid \mathbf{let\,it}\, A_1 \ldots A_n : B \Rightarrow b\, \mathbf{in}\, a \\ & \mid \mathbf{let\,it}\, A_1 \ldots A_n : B \Leftarrow e\, \mathbf{in}\, a \\ & \mid b\, a \mid \lambda b \mid \Pi A B \\ & \mid (a, b)_B \mid \mathbf{proj}_1\, p \mid \mathbf{proj}_2\, p \mid \Sigma A B \\ & \mid \mathbf{Type}_i \mid \mathbf{Set}_i \mid \mathbf{Prop} \end{aligned}$$

где $i \in \mathbb{N}$, $a, b, A, B, p, a_1, \ldots a_n, A_1, \ldots A_n \in Term$, $e \in Term_e$, D является именем типа данных, c – именем конструктора типа данных D, C – именем класса, f – именем поля некоторого класса и S – последовательностью пар (f, d), где f – имя поля класса C и d – терм. Такую пару (f, d) мы будем записывать как $f \Rightarrow d$. Как обычно, мы сокращаем $\Pi A(\uparrow B)$ до $A \to B$.

TODO: Добавить конструкции для вселенных **Prop** и $\mathbf{Set}_i$.

Определение редукций: TODO.

На множестве термов существует предпорядок $\leq$, порождаемый следующими соотношениями:

- $\mathbf{Prop} \leq \mathbf{Set}_i$.
- $\mathbf{Set}_i \leq \mathbf{Type}_i$.
- Если $i \leq j$, то $\mathbf{Set}_i \leq \mathbf{Set}_j$.
- Если $i \leq j$, то $\mathbf{Type}_i \leq \mathbf{Type}_j$.
- Если $A \Leftrightarrow B$, то $A \leq B$.
- Если $A \leq A'$ и $B \leq B'$, то $\Sigma A B \leq \Sigma A' B'$.
- Если $A' \leq A$ и $B \leq B'$, то $\Pi A B \leq \Pi A' B'$.
- Если любой член последовательности S встречается в S', то $C\, \{S'\} \leq C\, \{S\}$.

Правила типизации будут определены относительно фиксированного множества определений Σ. Как отмечалось ранее, каждое определение является определением либо функции, либо типа данных, либо класса.

Во-первых, опишем стандартные базовые правила для контекстов, индексов и типов:

$$\frac{}{\vdash} \qquad \frac{\Gamma \vdash A}{\Gamma, A \vdash} \qquad \frac{A_1, \ldots A_n \vdash}{A_1, \ldots A_n \vdash v_i \Uparrow (\uparrow^{i+1} A_{n-i})} \qquad \frac{\Gamma \vdash a \Uparrow A}{\Gamma \vdash a \Downarrow B}\, , A \leq B$$

Термы $\mathbf{Type}_i$, $\mathbf{Set}_i$ и **Prop**} мы будем называть *вселенными в нормальной форме*, множество таких термов мы будем обозначать $\mathcal{U}_{nf}$. Мы будем называть *вселенными* термы U, такие что $U \Rightarrow_h^* U'$ для некоторого $U' \in \mathcal{U}_{nf}$. Множество вселенных мы будем обозначать $\mathcal{U}$. Тогда существует функция $nf : \mathcal{U} \to \mathcal{U}_{nf}$ каждой вселенной сопоставляющая ее нормальную форму.

Определим функцию $max : \mathcal{U} \times \mathcal{U} \to \mathcal{U}_{nf}$ следующим образом:

$$\begin{aligned}
max(\mathbf{Type}_i, \mathbf{Type}_j) &= \mathbf{Type}_{max(i,j)} \\
max(\mathbf{Type}_i, \mathbf{Set}_j) &= \mathbf{Type}_{max(i,j)} \\
max(\mathbf{Set}_i, \mathbf{Type}_j) &= \mathbf{Type}_{max(i,j)} \\
max(\mathbf{Set}_i, \mathbf{Set}_j) &= \mathbf{Set}_{max(i,j)} \\
max(\mathbf{Type}_i, \mathbf{Prop}) &= \mathbf{Type}_i \\
max(\mathbf{Prop}, \mathbf{Type}_i) &= \mathbf{Type}_i \\
max(\mathbf{Set}_i, \mathbf{Prop}) &= \mathbf{Set}_i \\
max(\mathbf{Prop}, \mathbf{Set}_i) &= \mathbf{Set}_i
\end{aligned}$$

Для произвольных U и V мы полагаем $max(U, V) = max(nf(U), nf(V))$. Функция max_Π определена так же как и max за исключением того, что $max_\Pi(U, V) = \mathbf{Prop}$ для любого U и V, такого что $nf(V) = \mathbf{Prop}$. Для произвольного конечного множества термов $U_1, \dots U_n$ мы определяем $max(U_1, \dots U_n)$ как $max(U_1, \dots max(U_{n-1}, U_n) \dots)$. На пустом множестве мы определяем max как $\mathbf{Prop}$.

Теперь опишем правила для Π и Σ типов. В правилах ниже термы U и V принадлежат множеству $\{\mathbf{Type}_i, \mathbf{Set}_i, \mathbf{Prop}\}$.

$$\frac{\Gamma \vdash A \Uparrow U \qquad \Gamma, A \vdash B \Uparrow V}{\Gamma \vdash \Pi AB \Uparrow max_\Pi(U, V)} \qquad \frac{\Gamma \vdash A \Uparrow U \qquad \Gamma, A \vdash B \Uparrow V}{\Gamma \vdash \Sigma AB \Uparrow max(U, V)}$$

$$\frac{\Gamma, A \vdash b \Uparrow B}{\Gamma \vdash \lambda b \Uparrow \Pi AB} \qquad \frac{\Gamma \vdash b \Uparrow C \qquad \Gamma \vdash a \Downarrow A}{\Gamma \vdash b\, a \Uparrow B[a]}\, , C \Rightarrow_h^* \Pi AB$$

$$\frac{\Gamma, A \vdash B \qquad \Gamma \vdash a \Uparrow A \qquad \Gamma \vdash b \Downarrow B[a]}{\Gamma \vdash (a, b)_B \Uparrow \Sigma AB}$$

$$\frac{\Gamma \vdash p \Uparrow C}{\Gamma \vdash \mathbf{proj}_1\, p \Uparrow A}\, , C \Rightarrow_h^* \Sigma AB \qquad \frac{\Gamma \vdash p \Uparrow C}{\Gamma \vdash \mathbf{proj}_2\, p \Uparrow B[\mathbf{proj}_1\, p]}\, , C \Rightarrow_h^* \Sigma AB$$

Следующие правила для вселенных:

$$\frac{\Gamma \vdash}{\Gamma \vdash \mathbf{Type}_i} \qquad \frac{\Gamma \vdash}{\Gamma \vdash \mathbf{Set}_i} \qquad \frac{\Gamma \vdash}{\Gamma \vdash \mathbf{Prop}} \qquad \frac{\Gamma \vdash A \Downarrow U}{\Gamma \vdash A}$$

$$\frac{\Gamma \vdash}{\Gamma \vdash \mathbf{Prop} \Uparrow \mathbf{Set}_0} \qquad \frac{\Gamma \vdash}{\Gamma \vdash \mathbf{Set}_i \Uparrow \mathbf{Type}_{i+1}} \qquad \frac{\Gamma \vdash}{\Gamma \vdash \mathbf{Type}_i \Uparrow \mathbf{Type}_{i+1}}$$

Если D является типом данных в сигнатуре Σ с параметрами $A_1, \dots A_n$ и вселенной U, то у нас есть следующее правило вывода:

$$\frac{\Gamma \vdash}{\Gamma \vdash D \Uparrow \Pi A_1(\dots \Pi A_n U \dots)}$$

Если c является конструктором D с типами аргументов $B_1, \dots B_k$, то у нас есть следующее правило вывода:

$$\frac{\Gamma \vdash \qquad \Gamma \vdash a_i \Downarrow A_i[a_1, \dots a_{i-1}]}{\Gamma \vdash D\, a_1\, \dots\, a_n\, .\, c \Uparrow \Pi(B_1[a_1, \dots a_n])(\dots \Pi(B_k[a_1, \dots a_n])(D\, a_1\, \dots\, a_n) \dots)}$$

Если C – класс в сигнатуре Σ и $f_1, \dots f_n$ – поля C с типами $B_1, \dots B_n$ соответственно, то для любого $1 \leq k \leq n$ у нас есть следующие правила вывода:

$$\frac{\Gamma \vdash \qquad \Gamma, C\,\{f_1 \Rightarrow b_1; \dots f_{i-1} \Rightarrow b_{i-1}\} \vdash b_i \Downarrow B_i,\ 1 \leq i < k}{\Gamma \vdash C\,\{f_1 \Rightarrow b_1; \dots f_{k-1} \Rightarrow b_{k-1}\}}$$

$$\frac{\begin{array}{c}\Gamma, C\,\{f_1 \Rightarrow b_1; \dots f_{i-1} \Rightarrow b_{i-1}\} \vdash b_i \Downarrow B_i,\ 1 \leq i < k \\ \Gamma \vdash \qquad \Gamma, C\,\{f_1 \Rightarrow b_1; \dots f_{k-1} \Rightarrow b_{k-1}\} \vdash B_i \Uparrow U_i,\ k \leq i \leq n\end{array}}{\Gamma \vdash C\,\{f_1 \Rightarrow b_1; \dots f_{k-1} \Rightarrow b_{k-1}\} \Uparrow max(U_k, \dots U_n)}$$

$$\frac{\Gamma \vdash \qquad \Gamma, C\,\{f_1 \Rightarrow b_1; \dots f_{i-1} \Rightarrow b_{i-1}\} \vdash b_i \Downarrow B_i,\ 1 \leq i < k}{\Gamma \vdash \mathbf{new}\, C\,\{f_1 \Rightarrow b_1; \dots f_{k-1} \Rightarrow b_{k-1}\} \Uparrow C\,\{f_1 \Rightarrow b_1; \dots f_{k-1} \Rightarrow b_{k-1}\}}$$

Если f является полем класса C типа B, то у нас есть следующее правило вывода:

$$\frac{\Gamma \vdash}{\Gamma \vdash f \Uparrow \Pi(C\,\{\})B}$$

Для конструкции **elim** мы вводим отношение $\Gamma \vdash^n a \Downarrow A$, где Γ – последовательность термов из $Term$, $n \in \mathbb{N}$, $a \in Term_e$ и $A \in Term$. Правило для **elim** тогда выглядит следующим образом: TODO.

Правило для **let**: TODO.

Теперь мы опишем как расширять сигнатуру новыми определениями. Мы будем писать $- \vdash_\Sigma -$ для обозначения того факта, что это отношение верно в сигнатуре Σ. Сначала мы введем вспомогательное понятие абстрактной функции, необходимое для описания рекурсивных функций. Пусть Σ – корректная сигнатура, f – имя функции и $B_1, \dots B_n, A \in Term$. Тогда $(f, (B_1, \dots B_n), A)$ является корректным определением абстрактной функции в сигнатуре Σ, если

$$B_1, \dots B_n \vdash^n_\Sigma A.$$

Пусть Σ – корректная сигнатура, f – имя функции, $d \in Def$, $B_1, \dots B_n, A \in Term$, $a \in Term_e$ и если $d ==\Rightarrow$, то $a \in Term$. Тогда $(f, (B_1, \dots B_n), A, d, a)$ является корректным определением функции в сигнатуре Σ, если $(f, (B_1, \dots B_n), A)$ является корректной абстрактной функцией в Σ и

$$B_1, \dots B_n \vdash^n_{\Sigma,(f,(B_1,\dots B_n),A)} a \Downarrow A.$$

Кроме того, мы требуем, чтобы определение было завершающимся, что проверяется отдельно.

Пусть Σ – корректная сигнатура, D – имя типа данных, $A_1, \dots A_n \in Term$ и C – множество пар (c, B), где c – имя конструктора и $B = B_1, \dots B_k$ – список типов. Тогда $(D, (A_1, \dots A_n), C)$ является корректным определением типа данных в сигнатуре Σ, если

$$A_1, \dots A_n \vdash_\Sigma$$

и для любой пары $(c, (B_1, \dots B_k)) \in C$ верно, что

$$A_1, \dots A_n, B_1, \dots B_k \vdash_{\Sigma,(D,(A_1,\dots A_n),\varnothing)} .$$

Пусть Σ – корректная сигнатура, C – имя класса и F – конечное множество пар (f, B), где f – имя поля и $B \in Term$. Тогда (C, F) является корректным

определением класса, если элементы F можно упорядочить $(f_1, B_1), \dots (f_n, B_n)$ таким образом, что для любого $1 \leq i \leq n$ выполняется следующее свойство:

$$C\,\{\} \vdash_{\Sigma,(C,\{(f_1,B_1),\dots(f_{i-1},B_{i-1})\})} B_i.$$

2. Vclang

Язык vclang имеет ряд отличий от vccore:

- В vclang есть полноценные классы. Они могут быть вложенными, а также могут наследоваться. В vclang, так же как и в vccore, есть анонимное наследование.
- В vclang есть неявные аргументы.
- В vclang будет реализован механизм, аналогичный классам типов.

Множество термов $Term_{vc}$ языка vclang определяется следующим образом:

$$\begin{aligned} Term_{vc} := {} & x \mid D \mid D\,a_1\ \dots\ a_n\,.\,c \\ & \mid \mathbf{new}\,C\,\{S\} \mid C\,\{S\} \\ & \mid b\,.\,D \mid b\,.\,f \mid b\,.\,D\,a_1\ \dots\ a_n\,.\,c \\ & \mid \mathbf{new}\,b\,.\,C\,\{S\} \mid b\,.\,C\,\{S\} \\ & \mid \mathbf{let}\,x\,A_1\ \dots\ A_n : B \Rightarrow b\,\mathbf{in}\,a \\ & \mid \mathbf{let}\,x\,A_1\ \dots\ A_n : B \Leftarrow e\,\mathbf{in}\,a \\ & \mid b\,a \mid \lambda x.b \mid \lambda x : A.b \mid \Pi(x : A)B \\ & \mid (a,b) \mid \Sigma(x : A)B\,.\,(a,b) \mid \mathbf{proj}_1\,p \mid \mathbf{proj}_2\,p \mid \Sigma(x : A)B \\ & \mid \mathbf{Type}_i \mid \mathbf{Set}_i \mid \mathbf{Prop} \end{aligned}$$

где x – имя переменной, $a, b, A, B, p, a_1, \dots a_n, A_1, \dots A_n \in Term$, $e \in Term_e$, D является именем типа данных, c – именем конструктора типа данных D, C – именем класса, f – именем поля некоторого класса и S – последовательностью пар (f, d), где f – имя поля класса C и d – терм. Такую пару (f, d) мы будем записывать как $f \Rightarrow d$. Как обычно, мы сокращаем $\Pi(x : A)B$ до $A \to B$, если $x \notin FV(B)$.