Pixy formal semantics

A. Finn Hackett, Reed Mullanix

February 23, 2018

Basic context:

1. Γ is the typing context
2. S is the state at $t-1$
3. t is the current timestep
4. $\Rightarrow$ is normal type synthesis
5. $\overset{C}{\Rightarrow}$ is "choked" type synthesis, for use as per my last "if" spec. Watch what where does under those circumstances.
6. $\Downarrow$ is evaluation, not explicitly defined but used in ite
7. fby also uses a choke-based strategy, which should help formalise the idea of "stopping the lhs" and what exactly happens to rhs when we're still waiting for lhs. No buffers!
8. A naive implementation should be terrible but work. I have ideas on how to optimise later. The key to not going forever is to realise that the result will always be the same for the same S, and memoising our way to a stop.
9. **compat** is to ensure that types remain consistent over time - proving type consistency is TODO, needs stronger statements in some places. (right now you can pad with nils to allow random type changes)

$$\frac{\begin{array}{c}\Gamma; S; t \vdash C \Downarrow \mathit{true} \\ \Gamma; S; t \vdash A \Rightarrow T \\ \Gamma; S; t \vdash B \overset{C}{\Leftarrow} 1\end{array}}{\Gamma; S; t \vdash \mathrm{ite}(C, A, B) \Rightarrow T}[\mathtt{Synth-ite-true}]$$

$$\frac{\Gamma; S; t \vdash C \Downarrow false \qquad \Gamma; S; t \vdash A \overset{C}{\Leftarrow} 1 \qquad \Gamma; S; t \vdash B \Rightarrow T}{\Gamma; S; t \vdash \text{ite}(C, A, B) \Rightarrow T}[\mathtt{Synth-ite-false}]$$

$$\frac{\Gamma; S; t \vdash C \Downarrow nil \qquad \Gamma; S; t \vdash A \overset{C}{\Leftarrow} 1 \qquad \Gamma; S; t \vdash B \overset{C}{\Leftarrow} 1}{\Gamma; S; t \vdash \text{ite}(C, A, B) \Rightarrow 1}[\mathtt{Synth-ite-nil}]$$

$$\frac{\Gamma; S; t \vdash C \overset{C}{\Rightarrow} 1 \qquad \Gamma; S; t \vdash A \overset{C}{\Rightarrow} 1 \qquad \Gamma; S; t \vdash B \overset{C}{\Rightarrow} 1}{\Gamma; S; t \vdash \text{ite}(C, A, B) \overset{C}{\Rightarrow} 1}[\mathtt{Synth-ite-C}]$$

$$\frac{S(L) = 0 \qquad \Gamma; S; t \vdash B \overset{C}{\Rightarrow} 1 \qquad \Gamma; S; t \vdash A \Rightarrow T}{\Gamma; S; t \vdash \text{fby}(A, B, L) \Rightarrow T}[\mathtt{Synth-fby-before}]$$

$$\frac{S(L) = 1 \qquad \Gamma; S; t \vdash A \overset{C}{\Rightarrow} 1 \qquad \Gamma; S; t \vdash B \Rightarrow T}{\Gamma; S; t \vdash \text{fby}(A, B, L) \Rightarrow T}[\mathtt{Synth-fby-after}]$$

$$\frac{S(L) = 0 \qquad \Gamma; S; t \vdash B \overset{C}{\Rightarrow} 1 \qquad \Gamma; S; t \vdash A \overset{C}{\Rightarrow} T}{\Gamma; S; t \vdash \text{fby}(A, B, L) \overset{C}{\Rightarrow} T}[\mathtt{Synth-fby-before-C}]$$

$$\frac{S[t](L) = 1 \qquad \Gamma; S; t \vdash A \overset{C}{\Rightarrow} 1 \qquad \Gamma; S; t \vdash B \overset{C}{\Rightarrow} T}{\Gamma; S; t \vdash \text{fby}(A, B, L) \overset{C}{\Rightarrow} T}[\mathtt{Synth-fby-after-C}]$$

$$\frac{T \neq nil}{nil\ \mathbf{compat}\ T}[\mathtt{Compat-nil-T}]$$

$$\frac{T \neq nil}{T\ \mathbf{compat}\ nil}[\mathtt{Compat-T-nil}]$$

$$\frac{}{T\ \mathbf{compat}\ T}[\mathtt{Compat-T-T}]$$

$$\frac{\begin{array}{c}\overline{\oslash; S; t \vdash S(l) \Rightarrow T_l}^{l:c\in C} \\ \overline{\overline{l : T_l}^{l:c\in C}, \Gamma; S; t \vdash c \Rightarrow T'_l}^{l:c\in C} \\ \overline{T_l\ \mathbf{compat}\ T'_l}^{l:c\in C} \\ \overline{l : T_l}^{l:c\in C}, \Gamma; S; t \vdash V \Rightarrow T\end{array}}{\Gamma; S; t \vdash \text{where}(V, C) \Rightarrow T}[\mathtt{Synth-where}]$$

$$\frac{\begin{array}{c}\overline{\oslash; S; t \vdash S(l) \Rightarrow T_l}^{l:c\in C} \\ \overline{\overline{l_2 : T_{l_2}}^{l_2:c_2\in C}, \overline{l_3 : 1}^{l_3:t_3\in\Gamma}; S; t \vdash c_1 \Rightarrow T'_{l_1}}^{l_1:c_1\in C} \\ \overline{T_l\ \mathbf{compat}\ T'_l}^{l:c\in C} \\ \overline{l : T_l}^{l:c\in C}, \overline{l : 1}^{l:t\in\Gamma}; S; t \vdash V \overset{C}{\Rightarrow} T\end{array}}{\Gamma; S; t \vdash \text{where}(V, C) \overset{C}{\Rightarrow} T}[\mathtt{Synth-where-C}]$$

$$\frac{}{\Gamma; S; t \vdash nil \Rightarrow 1}[\mathtt{Synth-nil}]$$

$$\frac{}{\Gamma; S; t \vdash nil \overset{C}{\Rightarrow} 1}[\mathtt{Synth-nil-C}]$$

$$\frac{}{\Gamma; S; t \vdash nil \Leftarrow T}[\mathtt{Check-nil}]$$

$$\frac{}{\Gamma; S; t \vdash \text{num}(N) \Rightarrow Number}[\mathtt{Synth-num}]$$

$$\frac{}{\Gamma; S; t \vdash \text{num}(N) \overset{C}{\Rightarrow} 1}[\mathtt{Synth-num-C}]$$

$$\frac{\Gamma(I) = T}{\Gamma; S; t \vdash \text{id}(I) \Rightarrow T}[\mathtt{Synth-id}]$$

$$\frac{\Gamma(I) = T}{\Gamma; S; t \vdash \text{id}(I) \overset{C}{\Rightarrow} 1}[\mathtt{Synth-id-C}]$$

1 Introduction

The evaluation rules of Pixy are split into two steps: construction and evaluation.

First, any preprocessing is performed such as determining and allocating queue sizes or scanning for free variables.

Then the evaluation rules are applied to the result of this step in order to execute the progam.

2 Utilities

$$\begin{aligned}\text{apply}(E, nil) &= \exists v \in freevariables(E), \text{apply}(E[v/nil], nil)\\ \text{apply}(E, <>) &= E\\ \text{apply}(E, << n, v >, R... >) &= \text{apply}(E[n/v], R)\end{aligned}$$

3 If

If has some quite interesting semantics - unlike in many languages it does not completely skip the evaluation of the subexpression it does not select. Instead, it always executes both subexpressions except that when a subexpression is not selected the inputs are replaced by nil. This has the effect of synchronising time between both branches regardless of which if chosen, while avoiding the catastrophically bad performance of actually providing data for both branches to process.

3.1 Evaluation

$$\frac{\begin{array}{c}\Gamma_1 \vdash C \Rightarrow \Gamma_2 \vdash nil\\ \Gamma_2 \vdash \text{choke}(T) \Rightarrow \Gamma_3 \vdash nil\\ \Gamma_3 \vdash \text{choke}(F) \Rightarrow \Gamma_4 \vdash nil\end{array}}{\Gamma \vdash \text{if}(C, T, F) \Rightarrow \Gamma_4 \vdash nil}\texttt{Eval} - \texttt{if} - \texttt{nil}$$

$$\frac{\begin{array}{c}\Gamma_1 \vdash C \Rightarrow \Gamma_2 \vdash true\\ \Gamma_2 \vdash T \Rightarrow \Gamma_3 \vdash V\\ \Gamma_3 \vdash \text{choke}(F) \Rightarrow \Gamma_4 \vdash nil\end{array}}{\Gamma \vdash \text{if}(C, T, F) \Rightarrow \Gamma_4 \vdash V}\texttt{Eval} - \texttt{if} - \texttt{true}$$

$$\frac{\begin{array}{c}\Gamma_1 \vdash C \Rightarrow \Gamma_2 \vdash false\\ \Gamma_2 \vdash \text{choke}(T) \Rightarrow \Gamma_3 \vdash nil\\ \Gamma_3 \vdash F \Rightarrow \Gamma_4 \vdash V\end{array}}{\Gamma \vdash \text{if}(C, T, F) \Rightarrow \Gamma_4 \vdash V}\texttt{Eval} - \texttt{if} - \texttt{false}$$

$$\frac{\begin{array}{c}\Gamma_1 \vdash \text{choke}(C) \Rightarrow \Gamma_2 \vdash nil \\ \Gamma_2 \vdash \text{choke}(T) \Rightarrow \Gamma_3 \vdash nil \\ \Gamma_3 \vdash \text{choke}(F) \Rightarrow \Gamma_4 \vdash nil\end{array}}{\Gamma \vdash \text{if}(C,T,F) \Rightarrow \Gamma_4 \vdash nil}\texttt{Choke} - \texttt{if}$$

3.2 Construction

$$\frac{\begin{array}{c}\Gamma| - S|C => \Gamma| - S_1|C_e \\ \Gamma| - S_1|Tsrc => \Gamma| - S_2|T_e \\ \Gamma| - S_2|Fsrc => \Gamma| - S_3|F_e\end{array}}{\Gamma| - S\left|\begin{array}{c}\text{if}\, C \,\text{then}\, T \\ \text{else}\, F\end{array}\right. => \begin{array}{c}\text{if}(C_{expr}, < T_{expr}, T_{vars} >, < F_{expr}, F_{vars} >), \\ C_{vars} \cup T_{vars} \cup F_{vars}\end{array}}\texttt{Construct} - \texttt{if}$$

4 fby

4.1 Evaluation

$$\frac{S => false, L => nil, R => nil}{\text{fby}(L,R,S,Q) => nil}\texttt{Eval} - \texttt{fby} - \texttt{1}$$

$$\frac{S => false, L => nil, R => R_{val}, R_{val} \neq nil, push(Q, R_{val})}{\text{fby}(L,R,S,Q) => nil}\texttt{Eval} - \texttt{fby} - \texttt{2}$$

$$\frac{S => false, L => L_{val}, L_{val} \neq nil, R => R_{val}, R_{val} \neq nil, push(Q, R_{val}), set(S, true)}{\text{fby}(L,R,S,Q) => L_{val}}\texttt{Eval} - \texttt{fby} - \texttt{3}$$

$$\frac{S => true, R => R_{val}, R_{val} \neq nil, \neg empty(Q), push(Q, R_{val})}{\text{fby}(L,R,S,Q) => pop(Q)}\texttt{Eval} - \texttt{fby} - \texttt{4}$$

$$\frac{S => true, R => R_{val}, R_{val} \neq nil, empty(Q)}{\text{fby}(L,R,S,Q) => R_{val}}\texttt{Eval} - \texttt{fby} - \texttt{5}$$

$$\frac{S => true, R => nil, empty(Q)}{\text{fby}(L,R,S,Q) => nil}\texttt{Eval} - \texttt{fby} - \texttt{6}$$

$$\frac{S => true, R => nil, \neg empty(Q)}{\text{fby}(L,R,S,Q) => pop(Q)}\texttt{Eval} - \texttt{fby} - \texttt{7}$$

$$\frac{\begin{array}{c}\Gamma \vdash \text{choke}(L) \Rightarrow \Gamma_1 \vdash nil \\ \Gamma_1 \vdash \text{choke}(R) \Rightarrow \Gamma_2 \vdash nil\end{array}}{\Gamma \vdash \text{fby}(L,R,S,Q) \Rightarrow \Gamma_2 \vdash nil}\texttt{Choke} - \texttt{fby}$$

4.2 Construction

$$\frac{\begin{array}{c}\Gamma| - S|L => \Gamma| - S_1|L_{expr}, L_{vars} \\ \Gamma| - S_1|R => \Gamma| - S_2|R_{expr}, R_{vars} \\ d = maxdistance(L_{expr}, R_{expr}) \\ < Q_f, \Gamma' >= fresh(Q, \Gamma) \\ < P_f, \Gamma'' >= fresh(P, \Gamma') \\ S_3 = alloc(d, Q_f, S_2) \\ S_4 = alloc(P_f, S_3)\end{array}}{\Gamma| - S|L \,\mathrm{fby}\, R => \Gamma''| - S_4|\,\mathrm{fby}(L_{expr}, R_{expr}, P_f, Q_f), L_{vars} \cup R_{vars}}\texttt{Construct} - \texttt{fby}$$

5 check

5.1 Evaluation

$$\frac{E => nil}{\mathrm{check}(E) => false}\texttt{Eval} - \texttt{check} - \texttt{nil}$$

$$\frac{E => v, v \neq nil}{\mathrm{check}(E) => true}\texttt{Eval} - \texttt{check} - \texttt{other}$$

$$\frac{\Gamma \vdash \mathrm{choke}(E) \Rightarrow \Gamma_1 \vdash nil}{\Gamma \vdash \mathrm{check}(E) \Rightarrow \Gamma_1 \vdash nil}\texttt{Choke} - \texttt{check}$$

5.2 Construction

$$\frac{\Gamma| - S|E => \Gamma| - S_1|E_{expr}, E_{vars}}{\Gamma| - S|?E => \mathrm{check}(E_{expr}), E_{vars}}\texttt{Construct} - \texttt{check}$$

6 where

6.1 Evaluation

$$\frac{\mathbf{foreach}\, e_i => v_i ..., set(n_i, v_i), E => V}{\mathrm{where}(E, < n_i, e_i > ...) => V}\texttt{Eval} - \texttt{where}$$

6.2 Construction

$$\dfrac{\begin{array}{c} < nf_i, \Gamma' >= fresh(n_i, \Gamma)... \\ S' = alloc(nf_i, S)... \\ E_s = E[n_i/nf_i...] \\ \Gamma'| - S'|E_s => \Gamma'| - S'_0|E_{expr}, E_{vars} \\ es_i = e_i[n_i/nf_i...]... \\ \Gamma'| - S'_{i-1}|es_i => \Gamma'| - S'_i|e_{i,expr}, e_{i,vars}... \end{array}}{\Gamma| - S \left| \begin{array}{l} E\,\text{where} \\ n_i = e_i; ... \\ end \end{array} \right. => \Gamma'| - S'_n \left| \begin{array}{c} \text{where}(E_{expr}, < nf_i, e_{i,expr} > ...), \\ E_{vars} \cup e_{i,vars}... \setminus \{nf_i...\} \end{array} \right.} \texttt{Construct} - \texttt{where}$$

7 hold

TODO: how to achieve nested iteration; current theory: specify a set of streams to sample from and hold constant while the nested iteration finishes. ?