

# Pixy formal semantics

A. Finn Hackett, Reed Mullanix

February 5, 2018

## 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 program.

## 2 Utilities

$$\text{apply}(E, \text{nil}) = \exists v \in \text{freevariables}(E), \text{apply}(E[v/\text{nil}], \text{nil})$$

$$\text{apply}(E, <>) = E$$

$$\text{apply}(E, << n, v >, R... >) = \text{apply}(E[n/v], R)$$

## 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

$$\frac{C \Rightarrow \text{nil}, \text{apply}(T, \text{nil} \dots) \Rightarrow -, \text{apply}(F, \text{nil} \dots) \Rightarrow -}{\text{if}(C, < T, T_{args} >, < F, F_{args} >) \Rightarrow \text{nil}} \text{Eval - if - nil}$$

$$\frac{C \Rightarrow \text{true}, \text{apply}(T, T_{args}) \Rightarrow T_{val}, \text{apply}(F, \text{nil}) \Rightarrow -}{\text{if}(C, < T, T_{args} >, < F, F_{args} >) \Rightarrow T_{val}} \text{Eval - if - true}$$

$$\frac{C \Rightarrow false, \text{apply}(T, nil) \Rightarrow -, \text{apply}(F, F_{args}) \Rightarrow F_{val}}{\text{if}(C, < T, T_{args} >, < F, F_{args} >) \Rightarrow F_{val}} \text{Eval} - \text{if} - \text{false}$$

### 3.2 Construction

$$\frac{\begin{array}{c} \Gamma \mid - S \mid C \Rightarrow \Gamma \mid - S_1 \mid C_{expr}, C_{vars} \\ \Gamma \mid - S_1 \mid Tsrc \Rightarrow \Gamma \mid - S_2 \mid T_{expr}, T_{vars} \\ \Gamma \mid - S_2 \mid Fsrc \Rightarrow \Gamma \mid - S_3 \mid F_{expr}, F_{vars} \end{array}}{\Gamma \mid - S \mid \begin{array}{l} \text{if } C \text{ then } T \\ \text{else } F \end{array} \Rightarrow \begin{array}{l} \text{if}(C_{expr}, < T_{expr}, T_{vars} >, < F_{expr}, F_{vars} >), \\ C_{vars} \cup T_{vars} \cup F_{vars} \end{array}} \text{Construct} - \text{if}$$

## 4 fby

### 4.1 Evaluation

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

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

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

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

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

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

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

### 4.2 Construction

$$\frac{\begin{array}{c} \Gamma \mid - S \mid L \Rightarrow \Gamma \mid - S_1 \mid L_{expr}, L_{vars} \\ \Gamma \mid - S_1 \mid R \Rightarrow \Gamma \mid - S_2 \mid R_{expr}, R_{vars} \\ \exists t_1, t_2, \end{array}}{\Gamma \mid - S \mid L \text{ fby } R \Rightarrow \Gamma \mid - S_2 \mid \text{fby}(L_{expr}, R_{expr}), L_{vars} \cup R_{vars}} \text{Construct} - \text{fby}$$

## 5 check

### 5.1 Evaluation

$$\frac{E \Rightarrow nil}{\text{check}(E) \Rightarrow false} \text{Eval} - \text{check} - \text{nil}$$

$$\frac{E \Rightarrow v, v \neq nil}{\text{check}(E) \Rightarrow true} \text{Eval} - \text{check} - \text{other}$$

### 5.2 Construction

$$\frac{\Gamma | - S | E \Rightarrow \Gamma | - S_1 | E_{expr}, E_{vars}}{\Gamma | - S | ?E \Rightarrow \text{check}(E_{expr}), E_{vars}} \text{Construct} - \text{check}$$

## 6 where

### 6.1 Evaluation

$$\frac{e_i \Rightarrow v_i \dots, \text{set}(n_i, v_i), E \Rightarrow V}{\text{where}(E, < n_i, e_i > \dots) \Rightarrow V} \text{Eval} - \text{where}$$

### 6.2 Construction

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

## 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. ?