Proposal

Zach Sullivan and Gordon Lin

Parametric Polymorphism

We will add parametric polymorphism to our R5 language, which allows us to be more expressive when describing types of functions. Instead of using just our just concrete types like Integer, we can now use type variables when defining functions and lambdas.

Our milestones will be the following: type-checking and moving the code through the compiler. If time permits, we could implement user defined data-types.

Grammar

$$
\begin{aligned}
\mathit{type} &::= \text{Boolean} \mid \text{Integer} \mid (\text{Vector } \mathit{type}^{+}) \mid \text{Void} \mid (\mathit{type}^{*} \rightarrow \mathit{type}) \mid \textbf{tvar} \\
\mathit{expr} &::= \mathit{int} \mid \mathit{var} \mid \#f \mid \#t \mid (\mathit{read}) \mid (\mathit{void}) \mid (- \mathit{expr}) \mid (+ \mathit{expr}\ \mathit{expr}) \\
&\quad \mid (\text{let } ([\mathit{var}\ \mathit{expr}])\ \mathit{expr}) \mid (\text{and } \mathit{expr}\ \mathit{expr}) \mid (\text{or } \mathit{expr}\ \mathit{expr}) \\
&\quad \mid (\text{vector } \mathit{expr}^{+}) \mid (\text{vector-ref } \mathit{expr}\ \mathit{expr}) \mid (\text{vector-set! } \mathit{expr}\ \mathit{expr}\ \mathit{expr}) \\
&\quad \mid (\mathit{expr}\ \mathit{expr}^{*}) \mid (\text{lambda } (\mathit{var}^{*})\ \mathit{expr}) \\
\mathit{defs} &::= (\text{define } (\mathit{var}\ [\mathit{var}\ :\ \mathit{type}]^{*})\ :\mathit{type}\ \mathit{expr}) \\
R_{5.00295} &::= (\text{program } \mathit{defs}^{*}\ \mathit{expr})
\end{aligned}
$$

Examples

```
(define (id [x : a]) : a x)
(id (if (id #f)
        0
        42))
> 42
```

```
(define (swap [v : (Vector a b)]) : (Vector b a)
  (vector (vector-ref v 1) (vector-ref v 0)))
(vector-ref (swap (Vector #t 42)) 0)
> 42
```
