Cartesian Programming with TransLucid

John Plaice

Senior Scientist, GrammaTech, Ithaca

Adjunct, UNSW Australia, Sydney

11 March 2020

The natural numbers

$$n = \langle 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, \ldots \rangle$$

$$\begin{aligned} n_0 &= 0 \\ n_{i+1} &= 1 + n_i \end{aligned}$$

$$n = 0 \ fby.d \ 1 + n$$

Factorial

$$f = \langle 1, 1, 2, 6, 24, 120, 720, \ldots \rangle$$

$$\begin{aligned} f_0 &= 1 \\ f_{i+1} &= (i+1) \times f_i \end{aligned}$$

$$\begin{aligned} &fact\ n = f \\ &\texttt{where} \\ &\quad \texttt{dim}\ d \leftarrow n \\ &\quad f = 1\ fby.d\ index.d \times f \\ &\texttt{end} \end{aligned}$$

Fibonacci

$$fib = \langle 0, 1, 1, 2, 3, 5, 8, 13, 21, \ldots \rangle$$

$$\begin{aligned} f_0 &= 0 \\ f_1 &= 1 \\ f_{i+2} &= f_i + f_{i+1} \end{aligned}$$

```
fib n = f
where
  dim d ← n
  f = 0 fby.d 1 fby.d f + next.d f
end
```

How does it work (1) ?

$$
\begin{aligned}
\mathit{index}.d & = \#d + 1 \\
\mathit{first}.d\ X & = X[d \leftarrow 0] \\
\mathit{next}.d\ X & = X[d \leftarrow \#d + 1] \\
\mathit{fby}.d\ X\ Y & = \texttt{if}\ \#d < 1\ \texttt{then}\ X\ \texttt{else}\ Y[d \leftarrow \#d - 1]
\end{aligned}
$$

Fibonacci, fast

$$
\begin{aligned}
f_0 &= 0 \\
f_1 &= 1 \\
f_2 &= 1 \\
f_{2i} &= f_i \times (2f_{i+1} - f_i) \\
f_{2i+1} &= f_i^2 + f_{i+1}^2
\end{aligned}
$$

```
fib n = f
where
  dim d ← n
  f = 0 fby.d 1 fby.d 1 fby g[d ← #d + 3]
  g = lParent.f × (2 × rParent.d f − lParent.f)
      evenOrOdd.d
      sq(lParent.d f) + sq(rParent.d f)
end
```

How does it work (2) ?

$$
\begin{aligned}
lParent.d\ X &= X\left[d \leftarrow \lfloor \#d/2 \rfloor\right] \\
rParent.d\ X &= X\left[d \leftarrow \lfloor \#d/2 \rfloor + 1\right] \\
lChild.d\ X &= X[d \leftarrow \#d \times 2] \\
rChild.d\ X &= X[d \leftarrow \#d \times 2 + 1]
\end{aligned}
$$

Factorial, tournament computation

Suppose we wish to compute the factorial of n (here $n = 6$), using an approach called *tournament computation*. We will use two *dimensions*, here d (horizontal) and t (vertical):

F	d →								
t	1	1	2	3	4	5	6	1	1	$\cdots$
↓	1	6	20	6	1	1	$\cdots$			
	6	120	1	1	$\cdots$					
	720	1	1	$\cdots$						

The first row is simply the enumeration from 1 to n, with an additional 1 to the left and a sea of 1s to the right. In every other row, the i-th element is the product of the $2i$-th and $(2i + 1)$-st elements from the previous row.

F is the entire, infinite table.

Factorial, tournament computation (2)

F	d $\rightarrow$								
t	1	1	2	3	4	5	6	1	1	$\cdots$
$\downarrow$	1	6	20	6	1	1	$\cdots$			
	6	120	1	1	$\cdots$					
	720	1	1	$\cdots$						

$$
\begin{array}{l}
\mathit{fact}\ n = f \\
\texttt{where} \\
\quad \texttt{dim}\ d \leftarrow 0 \\
\quad f = \texttt{tournamentOp}.d\ n\ \mathit{times}\ \big(\texttt{default}.d\ 1\ n\ 1\ (\#d)\big) \\
\texttt{end}
\end{array}
$$

How does it work (3) ?

$$\texttt{default}.d\ m\ n\ val\ X = Y$$
$$\texttt{where}$$
$$\quad Y = \texttt{if}\ \#d \geq m\ \texttt{and}\ \#d \leq n\ \texttt{then}\ X\ \texttt{else}\ val$$
$$\texttt{end}$$

$$\texttt{tournamentOp}.d\ n\ g\ X = Y$$
$$\texttt{where}$$
$$\quad \texttt{dim}\ t \leftarrow ilog\, n$$
$$\quad Y = X\ fby.d\ g\ (\texttt{lChild}.d\ Y)\ (\texttt{rChild}.d\ Y)$$
$$\texttt{end}$$

Ackermann

	dim d_n $\rightarrow$						
ack	0	1	2	3	4	5	$\cdots$
dim $d_m \downarrow$ 0	1	2	3	4	5	6	$\cdots$
1	2	3	4	5	6	7	$\cdots$
2	3	5	7	9	11	13	$\cdots$
3	5	13	29	61	125	253	$\cdots$
4	13	65533	$\cdots$				
5	65533	$\cdots$					
$\vdots$	$\ddots$						

Ackermann

$$\begin{aligned} ack(0,n) &= n+1 \\ ack(m+1,0) &= ack(m,1) \\ ack(m+1,n+1) &= ack\big(m, ack(m+1,n)\big) \end{aligned}$$

$$\begin{array}{l} \texttt{fun}\ ack\ m\ n = A \\ \texttt{where} \\ \quad \texttt{dim}\ d_m \leftarrow m \\ \quad \texttt{dim}\ d_n \leftarrow n \\ \quad \texttt{var}\ A = (index.d_n)\ fby.d_m \\ \qquad\qquad \big((next.d_n\ A)\ fby.d_n\ (A\ @\ [d_n \leftarrow next.d_m\ A])\big) \\ \texttt{end} \end{array}$$

Matrix multiplication

$$
\begin{aligned}
&\mathit{multiply}.i.j\ n\ A\ B = D \\
&\texttt{where} \\
&\quad \texttt{dim}\ k \leftarrow n \\
&\quad C = \texttt{rotate}.k.j\ A \times \texttt{rotate}.k.i\ B \\
&\quad D = \texttt{sum}.k\ C \\
&\texttt{end}
\end{aligned}
$$

How does it work (4) ?

$$\texttt{rotate}.d_1.d_2\ X = X[d_1 \leftarrow \#d_2]$$

$$\texttt{sum}.d\ X = S$$
$$\texttt{where}$$
$$\quad S = 0\ \mathit{fby}.d\ S + X$$
$$\texttt{end}$$

Sieve of Eratosthenes

The sequence of primes is formed by the first element in each row:

		dim z $\rightarrow$												
	sieve.d	0	1	2	3	4	5	6	7	8	9	10	11	12
dim $d \downarrow$	0	2	3	4	5	6	7	8	9	10	11	12	13	14
	1	3	5	7	9	11	13	15	17	19	21	23	25	27
	2	5	7	11	13	17	19	23	25	29	31	35	37	41
	3	7	11	13	17	19	23	29	31	37	41	43	47	49
	4	11	13	17	19	23	29	31	37	41	43	47	53	59
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Each row corresponds to the removal of the multiples of the first element in the previous row.

Sieve of Eratosthenes

$$
\begin{aligned}
&\mathit{sieve}.d = f \\
&\texttt{where} \\
&\quad \texttt{dim}\ z \leftarrow 0 \\
&\quad f = (\#z + 2)\ \mathit{fby}.d\ \left(f\ \mathit{wvr}.z\ (f\ \%\ \mathit{first}.z \neq 0)\right) \\
&\texttt{end}
\end{aligned}
$$

How does it work (4) ?

$$
\begin{aligned}
&X\ wvr.d\ Y = \\
&\qquad \texttt{if}\ first.d\ Y \\
&\qquad \texttt{then}\ X\ fby.d\ (next.d\ X\ wvr.d\ next.d\ Y) \\
&\qquad \texttt{else}\ next.d\ X\ wvr.d\ next.d\ Y
\end{aligned}
$$

Powers

$$
\begin{array}{l}
powers.p = f \\
\texttt{where} \\
\quad f = \texttt{fn}\ s \to 1\ fby.p\ \texttt{fn}\ s \to s \times f\ s \\
\texttt{end}
\end{array}
$$

It defines the following table of functions:

$powers.p$	dim p $\to$ 0	1	2	3	4	5	$\cdots$
	$\lambda s.s^0$	$\lambda s.s^1$	$\lambda s.s^2$	$\lambda s.s^3$	$\lambda s.s^4$	$\lambda s.s^5$	$\cdots$

Powers

$(powers.d[d \leftarrow 5])2$ yields $2^5 = 32$.

$(powers.d[d \leftarrow 0])3$ yields $3^0 = 1$.

$(powers.d[d \leftarrow 4])3$ yields $3^4 = 81$.

Powers, fast

$$x^n = \begin{cases} (x^2)^{\frac{n}{2}}, & \text{if } n \text{ is even} \\ x(x^2)^{\frac{n-1}{2}}, & \text{if } n \text{ is odd} \end{cases}$$

$$
\begin{array}{l}
powers.p = f \\
\texttt{where} \\
\quad f = (\texttt{fn}\ s \to 1)\ fby.p\ (\texttt{fn}\ s \to s)\ fby.p\ g[d \leftarrow \#d + 2] \\
\quad g = \texttt{fn}\ s \to sq(\texttt{lParent}.p\ f\ s) \\
\qquad\quad \texttt{evenOrOdd}.d \\
\qquad\quad \texttt{fn}\ s \to s \times sq(\texttt{lParent}.p\ f\ s) \\
\texttt{end}
\end{array}
$$

The core problem I needed to solve

The following should hold:

$$
\begin{aligned}
&(\texttt{fn}\ X \leftarrow \#d + X)(\#e) && \equiv \#d + \#e \\
&(\texttt{fn}\ X \leftarrow \#d + X)\ (\#d + \#e) && \equiv \#d + (\#d + \#e) \\
&\big((\texttt{fn}\ X \leftarrow \#d + X)\ [d \leftarrow 3]\big)\ (\#d + \#e) && \equiv 3 + (\#d + \#e)
\end{aligned}
$$

What is the problem?

Consider the function

$$\texttt{var}\ f = \texttt{fn}.d_1 \to \texttt{fn}.d_2 \to \texttt{fn}\ X \to \texttt{fn}\ Y \to E$$

and the application

$$f.a.b\ P\ Q$$

The evaluation takes place like this:

$$(((((f).a).b)\ P)\ Q)$$

For each pair of parentheses, there is a context of evaluation.

Semantics $[\![\cdot]\!]$

$$
\begin{aligned}
[\![c(E_i)_{i\in m}]\!]\zeta\kappa &= \zeta(c)\left([\![E_i]\!]\zeta\kappa\right)_{i\in m} \\
[\![\#d]\!]\zeta\kappa &= \kappa(\zeta(d)) \\
[\![E\ [d_i \leftarrow E_i]_{i\in m}]\!]\zeta\kappa &= [\![E]\!]\zeta\left(\kappa \dagger \left\{\zeta(d_i) \mapsto [\![E_i]\!]\zeta\kappa\right\}_{i\in m}\right) \\
[\![x]\!]\zeta\kappa &= \begin{cases} \eta'\left(\kappa' \dagger \{\zeta(d_i) \mapsto \kappa(\zeta(d_i))\}_{i\in m}\right), \\ \qquad\qquad \zeta(x) = \left(\{d_i\}_{i\in m}, \eta', \kappa'\right) \\ \zeta(x)(\kappa), \quad \text{otherwise} \end{cases}
\end{aligned}
$$

Semantics $[\![\cdot]\!]$ of functions

$$
\begin{aligned}
[\![E.d']\!]\zeta\kappa &= ([\![E]\!]\zeta\kappa)(d')(\zeta)(\kappa) \\
[\![\mathtt{fn}.d \to E]\!]\zeta\kappa &= \langle\!\langle \mathtt{fn}.d \to E, \zeta, \kappa, \emptyset, \emptyset \rangle\!\rangle
\end{aligned}
$$

$$
\begin{aligned}
[\![E\ E']\!]\zeta\kappa &= ([\![E]\!]\zeta\kappa)(E')(\zeta)(\kappa) \\
[\![\mathtt{fn}\ x \to E]\!]\zeta\kappa &= \langle\!\langle \mathtt{fn}\ x \to E, \zeta, \kappa, \emptyset, \emptyset \rangle\!\rangle
\end{aligned}
$$

Semantics $\langle\!\langle \cdot \rangle\!\rangle$ of functions (1)

$$\langle\!\langle \mathtt{fn}.d \to E, \zeta, \kappa, \{d_i \mapsto (d'_i, \zeta'_i, \kappa'_i)\}_{i \in m}, \{x_j \mapsto (E_j, \zeta_j, \kappa_j)\}_{j \in n} \rangle\!\rangle$$

$$= \lambda d'.\zeta'.\kappa'.\langle\!\langle E, \zeta, \kappa, \{d_i \mapsto (d'_i, \zeta'_i, \kappa'_i)\}_{i \in m} \cup \{d \mapsto (d', \zeta', \kappa')\}, \{x_j \mapsto (E_j, \zeta_j, \kappa_j)\}_{j \in n} \rangle\!\rangle$$

$$\langle\!\langle \mathtt{fn}\ x \to E, \zeta, \kappa, \{d_i \mapsto (d'_i, \zeta'_i, \kappa'_i)\}_{i \in m}, \{x_j \mapsto (E_j, \zeta_j, \kappa_j)\}_{j \in n} \rangle\!\rangle$$

$$= \lambda E'.\zeta'.\kappa'.\langle\!\langle E, \zeta, \kappa, \{d_i \mapsto (d'_i, \zeta'_i, \kappa'_i)\}_{i \in m}, \{x_j \mapsto (E_j, \zeta_j, \kappa_j)\}_{j \in n} \cup \{x \mapsto (E', \zeta', \kappa')\} \rangle\!\rangle$$

Semantics $\langle\langle\cdot\rangle\rangle$ of functions (2)

$$
\begin{aligned}
&\left\langle\left\langle \underline{E}, \zeta, \kappa, \left\{d_i \mapsto (d'_i, \zeta'_i, \kappa'_i)\right\}_{i\in m}, \left\{x_j \mapsto (E_j, \zeta_j, \kappa_j)\right\}_{j\in n}\right\rangle\right\rangle \\
= \quad & \text{let } \nu = \max\left(\left\{\mathrm{rk}(\zeta), \mathrm{rk}(\kappa), \mathrm{rk}(\zeta'_i), \mathrm{rk}(\kappa'_i), \mathrm{rk}(\zeta_j), \mathrm{rk}(\kappa_j)\right\}_{i,j\in m,n}\right) \\
& \quad \sigma(i) = \min\left\{k \in m \mid d'_k = d'_i\right\}, \qquad i \in m \\
& \quad \kappa'' = \kappa \dagger \left\{\nu + \sigma(i) \mapsto \kappa'_{\sigma(i)}\left(\zeta'_{\sigma(i)}\left(d'_{\sigma(i)}\right)\right)\right\}_{i\in m} \\
& \quad \zeta''_j = \zeta_j \dagger \left\{d'_{\sigma_i} \mapsto \nu + \sigma(i)\right\}_{i\in m,} \qquad j \in n \\
& \quad \zeta'' = \zeta \dagger \left\{d_i \mapsto \nu + \sigma(i)\right\}_{i\in m} \\
& \qquad\qquad \dagger \left\{x_j \mapsto \left(\{d_i\}_{i\in m}, [\![E_j]\!]\zeta''_j, \kappa_j\right)\right\}_{j\in n} \\
& \text{in } [\![E]\!]\zeta''\kappa''
\end{aligned}
$$

Semantics $[\![\cdot]\!]$ of `where` clauses (1)

$$[\![E \texttt{ where} \quad \texttt{dim}_{i \in p} \quad d_i \leftarrow E_i \quad \texttt{var}_{j \in p+1..q} \quad x_j = E_j \quad \texttt{var}_{k \in q+1..r} \quad x_k.(d_{i,k})_{i \in m_k}(x_{j,k})_{j \in n_k} = E_k \quad \texttt{end}]\!] \zeta\kappa$$

Semantics $[\![\cdot]\!]$ of `where` clauses (2)

$$
\begin{aligned}
= \quad \text{let } \nu &= \max\big(\mathrm{rk}(\zeta), \mathrm{rk}(\kappa)\big) \\
\kappa' &= \kappa \dagger \left\{(\nu + i) \mapsto [\![E_i]\!]\zeta\kappa\right\}_{i \in p} \\
\zeta_0 &= \zeta \dagger \left\{d_i \mapsto \nu + i\right\}_{i \in p} \\
&\qquad \dagger \left\{x_j \mapsto \bot_{\mathbf{H}}\right\}_{j \in p+1..q} \\
&\qquad \dagger \left\{x_k \mapsto \bot_{D^{m_k} \times \mathbf{H}^{n_k} \to \mathbf{H}}\right\}_{k \in q+1..r} \\
\zeta_{\alpha+1} &= \zeta \dagger \left\{d_i \mapsto \nu + i\right\}_{i \in p} \\
&\qquad \dagger \left\{x_j \mapsto [\![E_j]\!]\zeta_\alpha\right\}_{j \in p+1..q} \\
&\qquad \dagger \left\{x_k \mapsto [\![\mathtt{fn}.(d_{i,k})_{i \in m_k} \to \mathtt{fn}\ (x_{j,k})_{j \in n_k} \to \underline{E_k}]\!]\zeta_\alpha\right\}_{k \in q+1..r} \\
\zeta' &= \lim_{\alpha \to \infty} \zeta_\alpha \\
\text{in } [\![E]\!]\zeta'\kappa'
\end{aligned}
$$

Conclusion

We now have for TransLucid:

- Currified dimensionally-abstract functions.
- Standard equalities are valid:
  $[\![E_1\ E2]\!]\zeta\kappa \equiv \left([\![E_1]\!]\zeta\kappa\right)\left([\![E_2]\!]\zeta\kappa\right).$

The following are Works in Progress:

- Semantics paper
- Haskell interpreter: `https://github.com/plaice/TLghc`
- Example programs
- Extensions to TransLucid
- Revived blog: `https://cartesianprogramming.com`
- Email: `johnplaice@gmail.com`