January 1985

Translating an FP dialect to L - A proof of correctness

Dennis M. Volpano

Follow this and additional works at: http://digitalcommons.ohsu.edu/csetech

Recommended Citation
http://digitalcommons.ohsu.edu/csetech/134

This Article is brought to you for free and open access by OHSU Digital Commons. It has been accepted for inclusion in CSE Tech by an authorized administrator of OHSU Digital Commons. For more information, please contact champies@ohsu.edu.
Translating an FP Dialect to L - A Proof of Correctness

Dennis M. Volpano

CSE TR 85-001

Oregon Graduate Center
19600 NW Walker Rd.
Beaverton, OR. 97006

Abstract

A dialect of FP includes FP selectors over tuples and the FP combining forms composition, condition, iteration and tuple construction. The primitives in a dialect are the primitive operations over some abstract data type. In this technical report, the translation of an FP dialect to an abstract imperative language L is formalized. A denotational description of L is given and the translation is proven correct.

Preliminary definitions

Before proving the correctness of the translation, some definitions and conventions must be given.

Definition. (Domains). Let Id be a set of cell names (identifiers) and V be a domain of objects. Let Seqp be the domain of store expressions. A store expression is a cell name or a sequence of store expressions. Let Env denote the domain of environments where an environment is a sequence of cells such that each cell is a triple <CELL, name, contents>.

Definition. Let the signature $\Sigma$ be given by:

\[
\begin{align*}
\text{while} : & \; 2 \rightarrow 1 \\
\sigma_0 : & \; 0 \rightarrow 1 \\
\cdot : & \; 2 \rightarrow 1 \\
(\rightarrow:) : & \; 3 \rightarrow 1 \\
||_{n} : & \; n \rightarrow 1 
\end{align*}
\]

The operator $\sigma_0$ is a family of nullary operators each of which is a primitive operation over some abstract data type. Let $T_{\Sigma}$ be a $\Sigma$-algebra whose carrier is the set of all $\Sigma$-terms and whose operators are those in $\Sigma$.

Definition. Let the signature $\Omega$ be given by:

\[
\begin{align*}
\text{whiledo} : & \; 2 \rightarrow 1 \\
\text{assign} : & \; 0 \rightarrow 1 \\
\text{semi} : & \; 2 \rightarrow 1 \\
\text{cond} : & \; 3 \rightarrow 1 \\
\text{constr}_{n} : & \; n \rightarrow 1 
\end{align*}
\]

The operator assign is a family of nullary operators indexed by elements of $\sigma_0$, Seqp and Id. For example, if $\overline{t} \in \sigma_0$, $x \in \text{Seqp}$ and $n \in \text{Id}$ then assign $(n, \text{apply}(\overline{t}, x))$ is a nullary operator. Let $T_{\Omega}$ be an $\Omega$-algebra whose carrier is the set of all $\Omega$-terms and whose operators are those in $\Omega$.

Conventions. Unless otherwise noted, $p$, $f$ and $g$ are $\Sigma$-terms, $l, l_1, l_2, \ldots$ are $\Omega$-terms and $z, z_1, z_2, \ldots$ are store expressions. Upper-case italic symbols ($I, I', \ldots$) will be used as metavariables ranging over cell names.

Definition. The apply constructor builds an application of a primitive in $\sigma_0$ to a store expression. The usual extractors, operator and operand, can be used on constructions built with apply but do not appear in our proof since no further interpretation is given to apply in the domain of $\Omega$-terms.

Definition. Let new be a function that maps a set of cell names $s$ to a single cell name such that new $(s) \notin s$. Let $\text{cells} : \text{Seqp} \rightarrow P(\text{Id})$ be a function defined as follows:

\[
\text{cells} I = \{I\}
\]

\[
\text{cells} <x_1, \ldots, x_n> = \text{cells} x_1 \cup \cdots \cup \text{cells} x_n
\]

Definition. Let store : Id $\rightarrow (\text{Env} \rightarrow \text{Env})$ and fetch : Seqp $\rightarrow (\text{Env} \rightarrow V)$ be functional forms defined as follows:

\[
\text{store} I = \text{apndl} \cdot \mid \text{CELL, I, 1, 2} \mid
\]

\[
\text{fetch} I = \text{eq} \cdot \mid \text{I, 2, 1} \mid \rightarrow 3 \cdot 1;
\]

\[
(\text{fetch} I) \cdot \overline{t}
\]
The functional form `fetch` on store expressions behaves like the function "lift" on sequences in [2].

**Definition.** Let \( \eta : T_\Omega \to \text{Sexp} \) be a function defined as follows:

\[
\eta \text{ assign } (I, \text{ apply } (f, x)) = I
\]
\[
\eta \text{ semi } (l_1, l_2) = \eta l_2
\]
\[
\eta \text{ cond } (l_1, l_2, l_3) = \eta l_2
\]
\[
\eta \text{ whiledo } (l_1, l_2) = \eta l_2
\]
\[
\eta \text{ constr } (l_1, \ldots, l_n) = <\eta l_1, \ldots, \eta l_n>
\]

Intuitively, \( \eta l \) is the store expression representing the array of cells in which results would be "deposited" if \( l \) were evaluated.

**Definition.** Let \( \delta : T_\Omega \times \text{Sexp} \to T_\Omega \) be a function such that \( \eta \delta(I, x) = x \). The mapping \( \delta \) must satisfy an axiom as we shall see. Intuitively, \( \delta \) performs a result-cell coercion by forcing the "result" cells of \( l \) to be \( \text{cells} (x) \).

**Definition.** Let \( \psi : T_\Omega \times \text{Sexp} \to T_\Omega \) be a function. Intuitively, \( \psi(l, x) \) preserves the meaning of the \( \Omega \)-term \( l \) and preserves the store expression \( x \). As we shall see, the mapping \( \psi \) must satisfy two axioms.

**Definition.** Let \( \Phi : T_\Sigma \times \text{Sexp} \times P(f) \to T_\Omega \) be a function. In \( \Phi(f, x, s) \), \( f \) is a \( \Sigma \)-term to be translated, \( x \) is a store expression to which \( f \) is applied and \( s \in P(f) \) is called the reserved set. The set \( s \) is reserved in the sense that any cell names created by virtue of translating \( f \) must be cell names that do not appear in \( s \). Let \( \Phi \) be defined as follows:

\[
\Phi(f, x, s) = \text{assign } (\text{new } (s), \text{apply } (f, x))
\]

where \( f \) is a primitive in \( \sigma_0 \).

\[
\Phi(f \cdot g, x, s) = 
\]
\[
\text{semi } (\Phi(g, x, s), \Phi(f, \eta \Phi(g, x, s), s))
\]

\[
\Phi(p \rightarrow f \cdot g, x, s) =
\]
\[
\text{cond } (\Phi(p, x, s), \delta(\Phi(f, x, s), \eta \Phi(g, x, s)), \Phi(g, x, s))
\]

\[
\Phi(\text{while } p \rightarrow f, x, s) =
\]
\[
\text{whiledo } (\Phi(p, x, s), \delta(\Phi(f, x, s), x))
\]

\[
\Phi([f_1, \ldots, f_n], x, s) =
\]
\[
\text{constr } (\psi(\Phi(f_1, x, v_1), x), \ldots, \psi(\Phi(f_n, x, v_n), x))
\]

where \( f_1, \ldots, f_n \) are \( \Sigma \)-terms, \( v_i = s \) and \( v_j = \text{cells } (\eta \Phi(f_{j-1}, x, v_{j-1})) \cup v_{j-1} \).

**Definition.** Let \( \mu : T_\Omega \to (\text{Env} \to \text{Env}) \) be a "meaning map" or representation function giving meaning to \( \Omega \)-terms. The meaning of \( \Omega \)-terms is couched in FP so that the FP algebra can be used in proofs about \( \Omega \)-terms. Let \( \mu \) be defined as follows:

\[
\mu \upharpoonright \bot = \bot
\]

\[
\mu \upharpoonright \text{assign } (I, \text{apply } (\text{select}, x, <x_1, \ldots, x_n>)) =
\]
\[
(\text{store } I) \cdot ([\text{fetch } x_1], \text{id})
\]

\[
\mu \upharpoonright \text{assign } (I, \text{apply } (\text{opr}, x)) =
\]
\[
(\text{store } I) \cdot [\text{opr} \cdot \text{fetch } x], \text{id}
\]

\[
\mu \upharpoonright \text{semi } (l_1, l_2) = \mu [l_2] \cdot \mu [l_1]
\]

\[
\mu \upharpoonright \text{cond } (l_1, l_2, l_3) =
\]
\[
(\text{fetch } \eta l_1) \cdot \mu [l_1] \rightarrow \mu [l_2]; \mu [l_3]
\]

\[
\mu \upharpoonright \text{whiledo } (l_1, l_2) =
\]
\[
(\text{fetch } \eta l_1) \cdot \mu [l_1] \rightarrow
\]
\[
\mu \upharpoonright \text{whiledo } (l_1, l_2) \cdot \mu [l_2]; \text{id}
\]

\[
\mu \upharpoonright \text{constr } (l_1, \ldots, l_n) =
\]
\[
\mu [l_n] \cdot \mu [l_{n-1}] \cdot \ldots \cdot \mu [l_1]
\]

The mappings \( \delta \) and \( \psi \) must satisfy the following axioms:

\[
(\text{fetch } x) \cdot \mu [\delta(I, x)] =
\]
\[
(\text{fetch } \eta l) \cdot \mu [l] \quad (A1)
\]

\[
(\text{fetch } x) \cdot \mu [\psi(l, x)] = (\text{fetch } x) \quad (A2)
\]

\[
(\text{fetch } \eta \psi(l, x)) \cdot \mu [\psi(l, x)] =
\]
\[
(\text{fetch } \eta l) \cdot \mu [l] \quad (A3)
\]

Axiom (A1) is the axiom of "result-cell coercion" and axioms (A2) and (A3) are the axioms of preservation.

---

1 The function \((\text{fetch } \eta l)\) is interpreted as \(\text{fetch } (\eta l)\).
Proof of correctness

The translation is now proven correct by showing that \( \Phi \) preserves the meaning of \( \Sigma \)-terms. The proof proceeds by structural induction on \( \Sigma \)-terms.

Theorem. For any \( \Sigma \)-term \( f \), store expression \( z \) and reserve set \( s \in P \{Id\} \),

\[
(f \eta \Phi(f, z, s)) \cdot \mu [\Phi(f, z, s)] = f \cdot (fetch z)
\]

Proof. Proceed by structural induction on \( \Sigma \)-terms. As basis cases, consider the FP selectors over tuples and the primitive operations over some abstract data type. If \( f \) is a primitive operation and \( \text{new}(s) = I \) then for any environment \( e \):

\[
(f \eta \Phi(f, z, s)) \cdot \mu [\Phi(f, z, s)] : e
\]

If \( f \) is an FP selector, say select,, and \( \text{new}(s) = I \) then for any environment \( e \):

\[
(f \eta \Phi(\text{select},, <x_1, ..., z, >, s)) \cdot \\
\mu [\Phi(\text{select},, <x_1, ..., z, >, s)] : e
\]

Composition.

\[
\begin{align*}
(f \eta \Phi(f \cdot g, z, s)) \cdot \mu [\Phi(f \cdot g, z, s)] \\
= (f \eta \Phi(f, g, z, s)) \cdot \\
\mu [\Phi(f, g, z, s)] \cdot \mu [\Phi(g, z, s)] \\
& \quad \{\text{defn. of } \eta \text{ and } \Phi\}
\end{align*}
\]

Conditional.

\[
\begin{align*}
(f \eta \Phi(p \rightarrow f ; g, z, s)) \\
= (f \eta \Phi(p \rightarrow f ; g, z, s)) \cdot \\
\mu [\Phi(p \rightarrow f ; g, z, s)] \cdot \\
\mu [\Phi(g, z, s)] \\
& \quad \{\text{defn. of } \eta \text{ and } \Phi\}
\end{align*}
\]

By definition of \( \Phi \),

\[
\begin{align*}
\mu [\Phi(p \rightarrow f ; g, z, s)] \\
= \mu [\text{cond}(\Phi(p, z, s)), \\
\delta(\Phi(f, z, s), \eta \Phi(g, z, s)), \\
\Phi(g, z, s))] \\
& \quad \{\text{defn. of } \mu \text{ and } \delta\}
\end{align*}
\]

Therefore,

\[
\begin{align*}
(f \eta \Phi(p \rightarrow f ; g, z, s)) \\
= (f \eta \Phi(p \rightarrow f ; g, z, s)) \cdot \\
\mu [\Phi(p \rightarrow f ; g, z, s)] \\
& \quad \{\text{defn. of } \mu \text{ and } \Phi\}
\end{align*}
\]
Assume the inductive hypothesis. Then,

\( (\text{fetch} \ \eta \Phi(\text{while } p \ f \ x \ s)) \cdot \mu [\Phi(\text{while } p \ f \ x \ s)] \)

\( = (\text{while } p \ f \ x \ s) \cdot (\text{fetch} \ x) \)

**Iteration.** Proceed by fixpoint induction.

\( (\text{fetch} \ \eta \Phi(\bot, x, s)) \cdot \mu [\Phi(\bot, x, s)] \)

\( = (\text{fetch} \ \eta \Phi(\bot, x, s)) \cdot \mu \bot \) \{defn. of \( \Phi \)\}

\( = \bot \) \{FP algebra\}

\( = \bot \cdot (\text{fetch} \ x) \) \{FP algebra\}

Now the fixpoint inductive hypothesis is given by:

\( (\text{fetch} \ \eta \Phi(\text{while } p \ f \ x \ s)) \cdot \mu [\Phi(\text{while } p \ f \ x \ s)] \)

\( = (\text{while } p \ f \ x \ s) \cdot (\text{fetch} \ x) \)

Assume the fixpoint inductive hypothesis. Then,

\( (\text{fetch} \ \eta \Phi(\text{while } p \ f \ x \ s)) \cdot \mu [\Phi(\text{while } p \ f \ x \ s)] \)

\( = (\text{fetch} \ \eta \Phi(\text{while } p \ f \ x \ s)) \cdot \mu \{\text{while do} (\Phi(\text{while } p \ f \ x \ s), \delta(\Phi(\text{while } p \ f \ x \ s), x))\} \)

\{defn. of \( \Phi \)\}

\( = (\text{fetch} \ \eta \Phi(\text{while } p \ f \ x \ s)) \cdot \mu [\Phi(\text{while } p \ f \ x \ s)] \rightarrow \\
(\text{fetch} \ \eta \Phi(\text{while } p \ f \ x \ s)) \cdot \mu [\Phi(\text{while } p \ f \ x \ s)] \)

\( = (\text{fetch} \ \eta \Phi(\text{while } p \ f \ x \ s)) \cdot \mu [\Phi(\text{while } p \ f \ x \ s)] \rightarrow \\
(\text{fetch} \ \eta \Phi(\text{while } p \ f \ x \ s)) \cdot \mu [\Phi(\text{while } p \ f \ x \ s)] \)

\( = (\text{fetch} \ \eta \Phi(\text{while } p \ f \ x \ s)) \cdot \mu [\Phi(\text{while } p \ f \ x \ s)] \rightarrow \\
(\text{fetch} \ \eta \Phi(\text{while } p \ f \ x \ s)) \cdot \mu [\Phi(\text{while } p \ f \ x \ s)] \)

\( = (\text{fetch} \ \eta \Phi(\text{while } p \ f \ x \ s)) \cdot \mu [\Phi(\text{while } p \ f \ x \ s)] \)

\( = (\text{fetch} \ \eta \Phi(\text{while } p \ f \ x \ s)) \cdot \mu [\Phi(\text{while } p \ f \ x \ s)] \)

**Construction.** The following proposition is needed in the proof of construction.

**Proposition.** For any \( \Omega \)-term of the form, constr\( * (\psi(l_1, x), ..., \psi(l_n, x)) \) where \( x \) is a store expression and \( i \neq j \),

\( (\text{fetch} \ \eta \psi(l, x)) \cdot \mu [\psi(l, x)] \)

\( = (\text{fetch} \ \eta \psi(l, x)) \)

**Proof.** If \( \eta \psi(l, x) \neq \eta \psi(l_j, x) \) then by definition of fetch the proposition holds. Suppose \( l = \Phi(f, x, v) \) and \( l_j = \Phi(f, x, v_j) \) where \( v \) and \( v_j \) are reserved sets such that...
\[ v_{i+1} = \text{cells} (\eta \Phi(f_i, x, v_i)) \cup v_i. \]

If \( i < j \) then without result-cell coercion \( \eta \psi(l_i, x) \in v_j \).

Therefore \( \eta \psi(l_j, x) \neq \eta \psi(l_j, x) \) since \( v_j \) is a reserved set. Similarly, if \( i > j \) then without coercing result cells \( \eta \psi(l_i, x) \in v_i \). Therefore \( \eta \psi(l_i, x) \neq \eta \psi(l_j, x) \) since \( v_i \) is a reserved set. Hence \( \eta \psi(l_i, x) = \eta \psi(l_j, x) \) only if \( l_i \) and \( l_j \) are the products of a result-cell coercion such that \( \eta \psi(l_i, x) = \eta \psi(l_j, x) \).

By axiom (A2),
\[
(f\text{etch } x) \cdot \mu \{ \psi(l_j, x) \} = (f\text{etch } x)
\]

and since \( x = \eta \psi(l_i, x) \), the proposition holds.

The proof of construction now proceeds as follows:
\[
(f\text{etch } \eta \psi(\Phi(f_i, x, v_i), x)) \cdot \\
\mu \{ \Phi([f_1, \ldots, f_n], x, s) \}
\]
\[
= (f\text{etch } \eta \psi(\Phi(f_i, x, v_i), x)) \cdot \\
\mu \{ \text{cons}_{\text{constr}} (\psi(\Phi(f_1, x, v_1), x), \ldots, \\
\psi(\Phi(f_n, x, v_n), x)) \}
\]
{defn. of \( \Phi \)}
\[
= (f\text{etch } \eta \psi(\Phi(f_i, x, v_i), x)) \cdot \\
\mu \{ \psi(\Phi(f_1, x, v_1), x) \} \cdot \ldots \\
\mu \{ \psi(\Phi(f_n, x, v_n), x) \}
\]
{defn. of \( \mu \)}
\[
= (f\text{etch } \eta \psi(\Phi(f_i, x, v_i), x)) \cdot \\
\mu \{ \psi(\Phi(f_i, x, v_i), x) \} \cdot \ldots \\
\mu \{ \psi(\Phi(f_i, x, v_i), x) \}
\]
{proposition}
\[
= (f\text{etch } \eta \Phi(f_i, x, v_i)) \cdot \\
\mu \{ \Phi(f_i, x, v_i) \} \cdot \ldots \\
\mu \{ \psi(\Phi(f_1, x, v_1), x) \}
\]
{axiom A3}
\[
= f, \cdot (f\text{etch } x) \cdot \\
\mu \{ \psi(\Phi(f_i, x, v_i), x) \} \cdot \ldots \\
\mu \{ \psi(\Phi(f_i, x, v_i), x) \}
\]
{ind. hyp.}
\[
= f, \cdot (f\text{etch } x) \} \} \text{axiom A2, } i-1 \text{ times}
\]

This concludes the proof of correctness.

References
