PFPL Supplement: Relating Transition and Cost Dynamics for Parallel PCF

Robert Harper

October 2018

1 Introduction

A cost dynamics for a language requires validation: what makes the cost assignments sensible? One form of validation is a Brent-type theorem, which provides a concrete bound on the cost in terms of an abstract machine model. The bounds are stated in terms of one or more parameters—such as the number of processing elements—of the machine model. The proof amounts to the description of a scheduler for the specified platform that achieves those bounds.

Another form of validation is to relate a cost dynamics to a transition dynamics for the same language. The assigned cost should determine the number of transitions required to achieve the value of an expression, if it has one. For a sequential SOS the correspondence validates the work measure of a cost graph, and for a parallel SOS the correspondence validates the span measure of a cost graph.

The purpose of this note is to state and prove these correspondences for the modal formulation of PCF enriched with parallelism, as described in Harper (2018). For present purposes we will assume that we have both the computation type and the lazy product type.¹

2 Cost and Transition Dynamics

For convenient reference, the cost dynamics of parallel PCF is given in Figure 1.

The transition dynamics for parallel PCF is given in Figure 2. This dynamics defines the parallel cost of an expression $e$ to be the number, $n$, if such exists, such that $e \rightarrow^n v$ with $v \text{val}$, or $\infty$ otherwise.

Instead of defining a separate sequential transition dynamics for parallel PCF, it may be considered to be defined implicitly by the parallel transition dynamics under the interpretation of the lazy product type $\tau_1 \& \tau_2$ as if it were the eager product type $\tau_1 \text{comp} \otimes \tau_2 \text{comp}$ of two suspended computations. This interpretation forces a sequential (either left-to-right or right-to-left) evaluation order of the components of a pair, so that the length of the transition sequence corresponds to the work, rather than the span, of a computation.

¹The type $\tau \text{comp}$ could be defined as $\tau \& \top$, essentially a “unary product” type, with its introduction and elimination forms being defined accordingly. Or we may derive the computation and lazy product types as the $n = 1$ and $n = 2$ cases, respectively, of the $n$-ary lazy product type.
\[
\begin{align*}
\text{ret}(v) & \Downarrow^1 v \quad \text{(1a)} \\
e_0 & \Downarrow^c v \quad \text{(1b)} \\
\text{if} \{ \tau \} (z; e_0; x.e_1) & \Downarrow^{1 \text{Hc}} v \\
[e/x]e_1 & \Downarrow^c v \quad \text{(1c)} \\
\text{if} \{ \tau \} (s(e); e_0; x.e_1) & \Downarrow^{1 \text{Hc}} v \\
\text{fun} \{ \tau_2; \tau \} (f.x.e), v_2/f, x.e & \Downarrow^c v \quad \text{(1d)} \\
\text{ap} (\text{fun} \{ \tau_2; \tau \} (f.x.e); v_2) & \Downarrow^{1 \text{Hc}} v \\
e & \Downarrow^c v \quad \text{[v/x]}e_2 & \Downarrow^{c_2} v_2 \quad \text{(1e)} \\
\text{let} (\text{comp}(e); x.e) & \Downarrow^{1 \text{Hc}c_2} v_2 \\
e_1 & \Downarrow^{c_1} v_1 \quad e_2 & \Downarrow^{c_2} v_2 \quad [v_1 \otimes v_2/x]e & \Downarrow^c v \quad \text{(1f)} \\
\text{par} (e_1 \& e_2; x.e) & \Downarrow^{(c_1c_2) \oplus 1 \text{Hc}} v \\
[v_1, v_2/x_1, x_2]e & \Downarrow^c v \quad \text{split} (v_1 \otimes v_2; x_1, x_2, e) & \Downarrow^{1 \text{Hc}} v \quad \text{(1g)}
\end{align*}
\]

Figure 1: Cost Dynamics

3 Validating the Cost Dynamics

**Theorem 3.1.** If \( e \Downarrow^c v \), then \( e \rightarrow^n v \), where \( n = dp(e) \) for the parallel dynamics, and \( n = wk(e) \) for the sequential dynamics.

**Proof.** By rule induction on the premise of the theorem. For example, consider rule 1f. We have that \( \text{par} (e_1 \& e_2; x.e_3) \Downarrow^c v_3 \), where

1. \( e_1 \Downarrow^{c_1} v_1 \),
2. \( e_2 \Downarrow^{c_2} v_2 \),
3. \([v_1 \otimes v_2/x]e_3 \Downarrow^{c_3} v_3 \), and
4. \( c = (c_1 \boxtimes c_2) \oplus 1 \boxplus c_3 \).

We have by induction

1. \( e_1 \rightarrow^{n_1} v_1 \) with \( n_1 = dp(c_1) \).
2. \( e_2 \rightarrow^{n_2} v_2 \) with \( n_2 = dp(c_2) \).
3. \( e_3 \rightarrow^{n_3} v_3 \) with \( n_3 = dp(c_3) \).

Assume that \( n_1 \) is no larger than \( n_2 \); the other case is handled analogously. We have the following
Figure 2: Parallel Transition Dynamics
transitions in the parallel transition system:

\[ \text{par}(e_1 \& e_2; x.e_3) \rightarrow^{n_1} \text{par}(\text{ret}(v_1) \& e_2'; x.e_3) \text{ (for some } e_2') \]
\[ \rightarrow^{n_2-n_1} \text{par}(\text{ret}(v_1) \& \text{ret}(v_2); x.e_3) \]
\[ \rightarrow^1 [v_1 \otimes v_2/x]e_3 \]
\[ \rightarrow^{n_3} v_3. \]

The length of this sequence is exactly the depth, \( dp(c) = \max(n_1, n_2) + 1 + n_3 \). The validation of the work is similar, albeit using the sequential interpretation of the lazy product. In that case the transition sequence has length \( n_1 + n_2 + 1 + n_3 \), which is \( wk(c) \).

**Lemma 3.2** (Cost Head Expansion). If \( e \rightarrow e' \) in parallel and \( e' \downarrow^c v \), then \( e \downarrow^c v \) for some \( c \) such that \( dp(c) = dp(c') + 1 \), and similarly for sequential transitions.

**Proof.** We proceed by induction on the initial transition.

Consider, for example, rule 2g, wherein \( e = \text{par}(e_1 \& e_2; x.e_3) \), \( e' = \text{par}(e_1' \& e_2'; x.e_3) \), with \( e_1 \rightarrow e_1' \) and \( e_2 \rightarrow e_2' \), and we have by assumption \( \text{par}(e_1' \& e_2'; x.e_3) \downarrow^{c'} v \) for some cost graph \( c' \).

By inversion this implies that \( c' = (c_1 \boxplus c_2) \boxplus 1 \boxplus c_3 \), where

1. \( e_1' \downarrow^{c_1} v_1 \) for some \( v_1 \),
2. \( e_2' \downarrow^{c_2} v_2 \) for some \( v_2 \), and
3. \([v_1 \otimes v_2/x]e_3 \downarrow^{c_3} v \).

We have by induction

1. \( e_1 \downarrow^{c_1} v \) with \( dp(c_1) = dp(c'_1) + 1 \), and
2. \( e_2 \downarrow^{c_2} v \) with \( dp(c_2) = dp(c'_2) + 1 \).

Consequently, by rule 1f, \( e \downarrow^c v \), where \( c = (c_1 \boxplus c_2) \boxplus 1 \boxplus c_3 \). Calculate as follows:

\[
dp(c) = \max(dp(c_1), dp(c_2)) + dp(c_3) + 1
\]
\[
= \max(dp(c'_1) + 1, dp(c'_2) + 1) + dp(c_3) + 1
\]
\[
= \max(dp(c'_1), dp(c'_2)) + 1 + dp(c_3) + 1
\]
\[
= dp(c') + 1.
\]

The other cases are handled similarly, as are the cases for the sequential transition system related to the work.

**Theorem 3.3.** If \( e \rightarrow^n \text{ret}(v) \) in parallel, then there exists \( c \) with \( dp(c) = n \) such that \( e \downarrow^c v \). Similarly, if \( e \rightarrow^n \text{ret}(v) \) sequentially, then there exists \( c \) with \( wk(c) = n \) and \( e \downarrow^c v \).

**Proof.** We will consider the parallel case; the sequential case is handled similarly. The proof is by induction on \( n \). If \( n = 0 \), then \( e = \text{ret}(v) \), and we may take \( c = 1 \) and apply rule 1a. Otherwise, we have \( e \rightarrow e' \rightarrow^{n'} \text{ret}(v) \). By induction \( e' \downarrow^{c'} v \) for some \( c' \) such that \( dp(c') = n' \). By Lemma 3.2 \( e \downarrow^c v \) for some \( c \) such that \( dp(c) = dp(c') + 1 \). But then \( dp(c) = n' + 1 = n \), as desired.
References
