PFPL Supplement: Relating Transition and Cost Dynamics for Parallel PCF

Robert Harper

September 17, 2019

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 assume that both the computation type and the lazy product type are available.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 $e \mapsto^n v$ with $v \mathtt{val}$, 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 $\tau_1 \mathtt{comp} \otimes \tau_2 \mathtt{comp}$ 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 $\tau \mathtt{comp}$ could be defined as $\tau \& \top$, with its introduction and elimination forms being defined accordingly. Or the computation and lazy product types may be seen as the $n = 1$ and $n = 2$ cases, respectively, of the $n$-ary lazy product type.
3 Validating the Cost Dynamics

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

Proof. By rule induction on the premise of the theorem. For example, consider rule 1f, so that

\[
\begin{align*}
\text{par(} e_1 \& e_2 ; x.e_3) & \downarrow^c v \\
\text{let(} \text{comp(} e ; x.e_2) & \downarrow^1 \oplus \downarrow^c v_2 \\
\text{split(} v_1 \otimes v_2 ; x_1, x_2.e & \downarrow^1 v) \\
\end{align*}
\]

Assume that \( n_1 \) is no larger than \( n_2 \); the other case is handled analogously. Applying the rules of
\[
\begin{align*}
e & \text{ ready} \quad (2a) \\
v & \text{ val} \\
\text{ret}(v) & \text{ done} \quad (2b) \\
\text{ifz}(z)(e_0; x; e_1) & \mapsto e_0 \quad (2c) \\
\text{ap}(\text{fun}\{\tau_2; \tau\}(f.x.e); v_2) & \mapsto \text{fun}\{\tau_2; \tau\}(f.x.e), v_2/f, x/e \quad (2d) \\
& \quad e_1 \mapsto e_1' \\
\text{let}(\text{comp}(e_1); x.e_2) & \mapsto \text{let}(\text{comp}(e_1'); x.e_2) \quad (2e) \\
\text{let}(\text{comp}\left(\text{ret}(v_1)\right); x.e_2) & \mapsto [v_1/x]e_2 \quad (2f) \\
& \quad e_1 \mapsto e_1' \quad e_2 \mapsto e_2' \\
\text{par}(e_1 & e_2; x.e) & \mapsto \text{par}(e_1' & e_2'; x.e) \quad (2g) \\
& \quad e_1 \mapsto e_1' \\
\text{par}(e_1 & \text{ret}(v_2); x.e) & \mapsto \text{par}(e_1' & \text{ret}(v_2); x.e) \quad (2h) \\
& \quad e_2 \mapsto e_2' \\
\text{par}(\text{ret}(v_1) & e_1; x.e) & \mapsto \text{par}(\text{ret}(v_1) & e_1'; x.e) \quad (2i) \\
\text{par}(\text{ret}(v_1) & \text{ret}(v_2); x.e) & \mapsto [v_1 \otimes v_2/x]e \quad (2j) \\
\text{split}(v_1 \otimes v_2; x_1, x_2.e) & \mapsto [v_1, v_2/x_1, x_2]e \quad (2k)
\end{align*}
\]

Figure 2: Parallel Transition Dynamics
Consider only the parallel case; the sequential case is similar. The proof is by induction on the initial transition.

**Theorem 3.3.**

Proof. Proceed by induction on the initial transition.

**Lemma 3.2 (Cost Head Expansion).** If \( e \mapsto e' \) 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. 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 \( e_1 \mapsto e_1' \) and \( e_2 \mapsto e_2' \), and by assumption \( \text{par}(e_1' \& e_2'; x.e_3) \downarrow^c v \) for some cost graph \( c' \). By inversion this implies that \( c' = (e_1' \otimes e_2') \oplus 1 \oplus c_3 \), 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. \([v_1 \otimes v_2/x]e_3 \downarrow^{c_3} v\).

By induction

1. \( e_1 \downarrow^{c_1} v_1 \) with \( dp(c_1) = dp(c_1') + 1 \), and
2. \( e_2 \downarrow^{c_2} v_2 \) with \( dp(c_2) = dp(c_2') + 1 \).

Consequently, by rule 1f, \( e \downarrow^c v \), where \( c = (c_1 \otimes c_2) \oplus 1 \oplus c_3 \). 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.

\[\Box\]

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

Proof. Consider only the parallel case; the sequential case is similar. The proof is by induction on \( n \). If \( n = 0 \), then \( e = \text{ret}(v) \), and take \( c = 1 \) and apply rule 1a. Otherwise, \( e \mapsto e' \mapsto^n \text{ret}(v) \).

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.

\[\Box\]
References
