A Proof of Equivalence of Operational Semantics

This appendix contains a proof that the two operational semantics systems defined in Sections 4.1 and 4.2 are equivalent. We begin our discussion by formally distinguishing between complex clauses which have already been wired and those which have not.

▶ **Definition 1.1.** A complex clause a in a graph D is *complete* iff, for all f, x_1, and x_2, we have that $D \longrightarrow^1 D \cup \textsc{Wire}(a, f, x_1, x_2)$ only if $D = D \cup \textsc{Wire}(a, f, x_1, x_2)$. A complex clause which is not *complete* is *incomplete*. Simple clauses are neither complete nor incomplete.

In showing equivalence, we are only concerned with graphs that were evaluated from an embedding of a real expression. These graphs exhibit certain properties – such as the uniqueness of active, incomplete clauses – that are important for showing alignment. Essentially, we wish to demonstrate that, up to the point we are currently evaluating, the graph looks like the expression from which it was embedded except that the graph has "bumps" where complete clauses were never deleted. We formalize this intuition as the following well-formedness property:

▶ **Definition 1.2.** A graph D is *well-formed* iff all of the following are true.

1. It contains at most one incomplete clause.
2. All clauses which are not complete (including simple clauses) are totally ordered.
3. For all active clauses a appearing in D and any x, $|D(x, a)| \leq 1$.

Of course, embeddings of expressions should be well-formed. Also, well-formedness should be preserved by evaluation.

▶ **Lemma 1.3.** *For any e, $\textsc{Embed}(e)$ is well-formed.*

Proof. Conditions 1 and 2 follow immediately from Definition 1.1 and the fact that $\textsc{Embed}(e)$ is totally ordered. The proof of condition 3 follows by induction on the number of nodes between a and the $\textsc{Start}$ node; we know that such a path must exist and be unique, as $\textsc{Embed}(e)$ is totally ordered and a is active. ◀

▶ **Lemma 1.4.** *If D is well-formed and $D \longrightarrow^1 D'$ then D' is well-formed.*

Proof. Each rule of Figure 12 conditions upon an active clause a. If that clause is complete, then $D = D'$ and this property is trivial. Otherwise, the conditions of well-formedness can be demonstrated to hold by showing that a' is unique (i.e. that D cannot step to any other graph but itself). We observe that a is both active and incomplete; by condition 1 of well-formedness, it is the only such clause in D. By condition 3, there is at most one way to satisfy each rule in Figure 12. By inspection, these rules have exclusive premises; thus D' is unique.

Using similar logic, a can be shown to be complete in D'. Wiring inserts a totally ordered sequence of nodes from the predecessors of a (at most one of which is not complete) to the successors of a (at most one of which is not complete). Thus, condition 2 holds. As a result, we know by Definition 3.5 that at most one incomplete clause is active (which may have been in or after the inserted wiring); thus, condition 1 holds.

Demonstrating that condition 3 of well-formedness holds is more tedious but not complex; it proceeds by case analysis on Definition 4.4 by using the well-formedness of D. ◀

With the preservation of well-formedness, we can now prove the equivalence of these operational semantics. Key to this equivalence proof is a bisimulation relation which we

establish between an expression and its embedding. We can then show that this bisimulation is preserved throughout evaluation. The bisimulation shows that, at a given point in evaluation, the contents of each variable from the current point of evaluation appear the same and all future evaluation steps are identical. We formalize this bisimulation as follows:

▶ **Definition 1.5.** Let $e' = E \,||\, e$. Let D be a well-formed graph with a node a which is not complete and has no active successors. (This is either a unique, active, incomplete node or the End node, depending on whether D has finished evaluating.) We write $e' \cong D$ when the following conditions are met:

1. For all $x = v \in E$, $D(x, a) = \{v\}$.
2. For any path $a \ll a' \ll \ldots \ll \textsc{End}$ in D, we have $\textsc{Embed}(e) = \{\textsc{Start} \ll a \ll a' \ll \ldots \ll \textsc{End}\}$.

The first condition of bisimulation ensures that each variable in the environment matches its lookup value in the graph (and vice versa); the second condition ensures that the unevaluated portions of the expression and the graph are identical. We then prove that the operational semantics are equivalent by showing that evaluation preserves our bisimulation; this proves Lemma 4.6.

Proof. Each part of the proof proceeds by case analysis on the appropriate relation. In the first part, for instance, we proceed by case analysis on the rule used to prove $e \longrightarrow^1 e'$. If it works on a complex clause, then the corresponding graph evaluation rule can be used to show that $D \longrightarrow^1 D'$ using the conditions of bisimulation and well-formedness. If the proof of $e \longrightarrow^1 e'$ uses the Variable Lookup rule, then $e' \cong D$ (since graph evaluation is lazy in alias clauses).

The second part of the proof is similar except that the latter relation may take many steps. For any $D \longrightarrow^1 D'$, we proceed by case analysis on the rule used and apply the appropriate rule of $e \longrightarrow^1 e''$, again satisfying premises from bisimulation and well-formedness. This step may introduce some number of alias clauses, which we then evaluate ($e'' \longrightarrow^* e'$) to reach our result. ◀