Proofs of laws for a unified theory of methods

Frank Zeyda and Ana Cavalcanti

March 6, 2014

1 Introduction

This report includes supplementary proofs of laws that have been omitted for brevity in our paper "Laws for a unified theory of methods" [4]. In Section 2, we consider proofs of laws for plain alphabet extension; Section 3 includes proofs of unfolding lemmas; and, Section 4 shows the validity of recast version of relevant relational and design theorems to reason about variable blocks.

2 Plain alphabet extension laws

We first recapture the definition of plain alphabet extension as given in in [4].

Definition 1. *Plain alphabet extension. For any predicate P and alphabet A,*

$$[P_{\oplus A} \Leftrightarrow P] \quad \textit{and} \quad \alpha(P_{\oplus A}) = \alpha P \cup A$$

Table 1 below recaptures the relevant laws for plain alphabet extension. Plain alphabet extension can always be removed, provided that we can show $A \subseteq \alpha P$ (Law 1). It distributes through the logical connectives (Law 2 to Law 5) as well as relational composition (Law 6 and Law 7). Law 8 enables us to remove unused variables from the interface of a relational composition, and Law 9 and Law 10 trade plain alphabet extension for an existentially-quantified variable in the right and left predicate of a relational composition. We next present individual proofs the laws.

Law 1. $P_{\oplus A} \;=\; P$ **provided** $A \subseteq \alpha P$
Law 2. $(\neg P)_{\oplus A} \;=\; \neg(P_{\oplus A})$
Law 3. $(P \;\mathbf{op}\; Q)_{\oplus A} \;=\; P_{\oplus A} \;\mathbf{op}\; Q$ **where** $\mathbf{op} \in \{\wedge, \vee, \Rightarrow, \Leftrightarrow\}$
Law 4. $(P \;\mathbf{op}\; Q)_{\oplus A} \;=\; P \;\mathbf{op}\; Q_{\oplus A}$ **where** $\mathbf{op} \in \{\wedge, \vee, \Rightarrow, \Leftrightarrow\}$
Law 5. $(\Lambda\, x \bullet P)_{\oplus A} \;=\; \Lambda\, x \bullet P_{\oplus A}$ **where** $\Lambda \in \{\forall, \exists\}$ **provided** $x \notin A$
Law 6. $(P \,;\, Q)_{\oplus \{x\}} \;=\; P_{\oplus \{x\}} \,;\, Q$
Law 7. $(P \,;\, Q)_{\oplus \{x'\}} \;=\; P \,;\, Q_{\oplus \{x'\}}$
Law 8. $P_{\oplus \{x'\}} \,;\, Q_{\oplus \{x\}} \;=\; P \,;\, Q$ **provided** $x' \notin \alpha(P) \wedge x \notin \alpha Q$
Law 9. $P_{\oplus \{x'\}} \,;\, Q \;=\; P \,;\, \exists\, x \bullet Q$ **provided** $x' \notin \alpha P$
Law 10. $P \,;\, Q_{\oplus \{x\}} \;=\; (\exists\, x' \bullet P) \,;\, Q$ **provided** $x \notin \alpha Q$

Table 1: Relevant laws for plain alphabet extension.

Proof of Law 1.

$$
\begin{array}{ll}
P \;=\; P_{\oplus A} & \{\text{definition of predicate equality}\} \\
= \; [P \Leftrightarrow P_{\oplus A}] \wedge \alpha(P) = \alpha(P_{\oplus A}) & \{\text{definition of plain alphabet extension}\} \\
= \; \alpha(P) = \alpha(P) \cup A & \{\text{assumption: } A \subseteq \alpha(P)\} \\
= \; \mathit{true} & \square
\end{array}
$$

Proof of Law 2.

$$
\begin{array}{ll}
(\neg P)_{\oplus A} \;=\; \neg P_{\oplus A} & \{\text{definition of predicate equality}\} \\
= \; [(\neg P)_{\oplus A} \Leftrightarrow \neg P_{\oplus A}] \wedge \alpha((\neg P)_{\oplus A}) = \alpha(P_{\oplus A}) & \{\text{simplification of alphabets}\} \\
= \; [(\neg P)_{\oplus A} \Leftrightarrow \neg P_{\oplus A}] \wedge \alpha(P) \cup A = \alpha(P) \cup A & \{\text{using } [(\neg P)_{\oplus A} \Leftrightarrow \neg P] \text{ (Def. 1)}\} \\
= \; [\neg P \Leftrightarrow \neg P_{\oplus A}] & \{\text{logic}\} \\
= \; [P \Leftrightarrow P_{\oplus A}] & \{\text{using } [P_{\oplus A} \Leftrightarrow P] \text{ (Def. 1)}\} \\
= \; [P \Leftrightarrow P] & \{\text{logic}\} \\
= \; \mathit{true} & \square
\end{array}
$$

Proof of Law 3. We first prove the law for conjunction.

$$
\begin{array}{ll}
(P \wedge Q)_{\oplus A} \;=\; P_{\oplus A} \wedge Q & \{\text{definition of predicate equality}\} \\
= \; \left(\begin{array}{l} [(P \wedge Q)_{\oplus A} \Leftrightarrow P_{\oplus A} \wedge Q] \wedge \\ \alpha((P \wedge Q)_{\oplus A}) = \alpha(P_{\oplus A} \wedge Q) \end{array} \right) & \{\text{simplification of alphabets}\} \\
= \; \left(\begin{array}{l} [(P \wedge Q)_{\oplus A} \Leftrightarrow P_{\oplus A} \wedge Q] \wedge \\ \alpha(P) \cup \alpha(Q) \cup A = \alpha(P) \cup \alpha(Q) \cup A \end{array} \right) & \{\text{using } [(P \wedge Q)_{\oplus A} \Leftrightarrow (P \wedge Q)] \text{ (Def. 1)}\} \\
= \; [P \wedge Q \Leftrightarrow P_{\oplus A} \wedge Q] & \{\text{logic}\} \\
= \; [P \wedge Q \Rightarrow P_{\oplus A} \wedge Q] \wedge [P_{\oplus A} \wedge Q \Rightarrow P \wedge Q] & \{Q \text{ appears in assumptions}\} \\
\Leftarrow \; [P \wedge Q \Rightarrow P_{\oplus A}] \wedge [P_{\oplus A} \wedge Q \Rightarrow P] & \{\text{weakening of antecedents}\} \\
= \; [P \Rightarrow P_{\oplus A}] \wedge [P_{\oplus A} \Rightarrow P] & \{\text{logic}\} \\
= \; [P \Leftrightarrow P_{\oplus A}] & \{\text{definition of plain alphabet extension}\} \\
= \; \mathit{true} & \square
\end{array}
$$

The remaining cases for $\mathbf{op} \in \{\vee, \Rightarrow, \Leftrightarrow\}$ are proved by rewriting the respective operators in terms of negation and conjunction, and exploiting the distributivity laws already proved for $\neg$ and $\wedge$.

Proof of Law 4. Follows directly from commutativity of $\{\wedge, \vee, \Leftarrow, \Leftrightarrow\}$.

$$
\begin{array}{ll}
(P \;\mathbf{op}\; Q)_{\oplus A} & \{\mathbf{op} \in \{\vee, \Rightarrow, \Leftrightarrow\} \text{ is commutative}\} \\
= \; (Q \;\mathbf{op}\; P)_{\oplus A} & \{\text{Law 3}\} \\
= \; Q_{\oplus A} \;\mathbf{op}\; P & \{\mathbf{op} \in \{\vee, \Rightarrow, \Leftrightarrow\} \text{ is commutative}\} \\
= \; P \;\mathbf{op}\; Q_{\oplus A} & \square
\end{array}
$$

Proof of Law 5. We first prove the law for existential quantification.

$$
\begin{array}{ll}
& (\exists\, x \bullet P)_{\oplus A} \;=\; (\exists\, x \bullet P_{\oplus A}) \qquad \{\text{definition of predicate equality}\} \\
= & \begin{pmatrix} [(\exists\, x \bullet P)_{\oplus A} \Leftrightarrow (\exists\, x \bullet P_{\oplus A})] \wedge \\ \alpha((\exists\, x \bullet P)_{\oplus A}) = \alpha((\exists\, x \bullet P_{\oplus A})) \end{pmatrix} \qquad \{\text{simplification of alphabets}\} \\
= & \begin{pmatrix} [(\exists\, x \bullet P)_{\oplus A} \Leftrightarrow (\exists\, x \bullet P_{\oplus A})] \wedge \\ (\alpha(P) \setminus \{x\}) \cup A = (\alpha(P) \cup A) \setminus \{x\} \end{pmatrix} \qquad \{\text{assupmtion } x \notin A\} \\
= & \begin{pmatrix} [(\exists\, x \bullet P)_{\oplus A} \Leftrightarrow (\exists\, x \bullet P_{\oplus A})] \wedge \\ (\alpha(P) \cup A) \setminus \{x\} = (\alpha(P) \cup A) \setminus \{x\} \end{pmatrix} \qquad \{\text{using } [(\exists\, x \bullet P)_{\oplus A} \Leftrightarrow \exists\, x \bullet P] \text{ (Def. 1)}\} \\
= & [(\exists\, x \bullet P) \Leftrightarrow (\exists\, x \bullet P_{\oplus A})] \qquad \{\text{logic}\} \\
= & \begin{pmatrix} [(\exists\, x \bullet P) \Rightarrow (\exists\, x \bullet P_{\oplus A})] \wedge \\ [(\exists\, x \bullet P_{\oplus A}) \Rightarrow (\exists\, x \bullet P)] \end{pmatrix} \qquad \{\text{assumption } x \notin A \text{ and logic, let } x_0 \notin \alpha(P) \cup A\} \\
= & \begin{pmatrix} [P[x \backslash x_0] \Rightarrow (\exists\, x \bullet P_{\oplus A})] \wedge \\ [P[x \backslash x_0]_{\oplus A} \Rightarrow (\exists\, x \bullet P)] \end{pmatrix} \qquad \{\text{using } x_0 \text{ as a witness}\} \\
= & \begin{pmatrix} [P[x \backslash x_0] \Rightarrow P[x \backslash x_0]_{\oplus A}] \wedge \\ [P[x \backslash x_0]_{\oplus A} \Rightarrow P[x \backslash x_0]] \end{pmatrix} \qquad \{\text{logic}\} \\
= & [P[x \backslash x_0] \Leftrightarrow P[x \backslash x_0]_{\oplus A}] \qquad \{\text{using } [P_{\oplus A} \Leftrightarrow P] \text{ (Def. 1)}\} \\
= & [P[x \backslash x_0] \Leftrightarrow P[x \backslash x_0]] \qquad \{\text{logic}\} \\
= & \mathit{true} \qquad \square
\end{array}
$$

As before, the remaining cases for $\forall\, x \bullet P$ is proved algebraically by rewriting the operator in terms of negation and existential quantification amd using the distribution laws already proved.

Proof of Law 6. We assume P and Q are composable with interface v.

$$
\begin{array}{ll}
& (P \,;\, Q)_{\oplus \{x\}} \qquad \{\text{unfolding relational composition}\} \\
= & (\exists\, v'' \bullet P[v' \backslash v''] \wedge Q[v \backslash v''])_{\oplus \{x\}} \qquad \{\text{Law 5 in Table 1 using } x \notin v''\} \\
= & \exists\, v'' \bullet (P[v' \backslash v''] \wedge Q[v \backslash v''])_{\oplus \{x\}} \qquad \{\text{Law 3 in Table 1}\} \\
= & \exists\, v'' \bullet P_{\oplus \{x\}}[v' \backslash v''] \wedge Q[v \backslash v''] \qquad \{\text{folding relational composition}\} \\
= & P_{\oplus \{x\}} \,;\, Q \qquad \square
\end{array}
$$

Proof of Law 7. We assume P and Q are composable with interface v.

$$
\begin{array}{ll}
& (P \,;\, Q)_{\oplus \{x'\}} \qquad \{\text{unfolding relational composition}\} \\
= & (\exists\, v'' \bullet P[v' \backslash v''] \wedge Q[v \backslash v''])_{\oplus \{x'\}} \qquad \{\text{Law 5 in Table 1 using } x \notin v''\} \\
= & \exists\, v'' \bullet (P[v' \backslash v''] \wedge Q[v \backslash v''])_{\oplus \{x'\}} \qquad \{\text{Law 4 in Table 1}\} \\
= & \exists\, v'' \bullet P[v' \backslash v''] \wedge Q_{\oplus \{x'\}}[v \backslash v''] \qquad \{\text{folding relational composition}\} \\
= & P \,;\, Q_{\oplus \{x'\}} \qquad \square
\end{array}
$$

Proof of Law 8. We assume that $x \notin in(\alpha Q)$ and P and Q are composable with interface v.

$$
\begin{array}{lll}
& P_{\oplus\{x'\}} \;;\; Q_{\oplus\{x\}} & \{\text{unfolding relational composition}\} \\
= & \exists\, v'', x'' \bullet P_{\oplus\{x'\}}[v', x' \backslash v'', x''] \wedge Q_{\oplus\{x\}}[v, x \backslash v'', x''] & \{\text{simplify substitution}\} \\
= & \exists\, v'', x'' \bullet P_{\oplus\{x''\}}[v' \backslash v''] \wedge Q_{\oplus\{x''\}}[v \backslash v''] & \{\text{unused quantified variable } x''\} \\
= & \exists\, v'' \bullet P[v' \backslash v''] \wedge Q[v \backslash v''] & \{\text{folding relational composition}\} \\
= & P \;;\; Q & \square
\end{array}
$$

Proof of Law 9. We assume that $x' \notin \alpha P$ and $P_{\oplus\{x'\}}$ and Q are composable with interface $\{v, x\}$.

$$
\begin{array}{lll}
& P_{\oplus\{x'\}} \;;\; Q & \{\text{unfolding relational composition}\} \\
= & \exists\, v'', x'' \bullet P_{\oplus\{x'\}}[v', x' \backslash v'', x''] \wedge Q[v, x \backslash v'', x''] & \{\text{simplify substitution}\} \\
= & \exists\, v'', x'' \bullet P[v' \backslash v''] \wedge Q[v, x \backslash v'', x''] & \{\text{localising quantifier } \exists\, x'' \bullet \ldots\} \\
= & \exists\, v'' \bullet P[v' \backslash v''] \wedge \exists\, x'' \bullet Q[v, x \backslash v'', x''] & \{\text{extracting substitution}\} \\
= & \exists\, v'' \bullet P[v' \backslash v''] \wedge (\exists\, x'' \bullet Q[x \backslash x''])[v \backslash v''] & \{\text{renaming quantified variable}\} \\
= & \exists\, v'' \bullet P[v' \backslash v''] \wedge (\exists\, x \bullet Q)[v \backslash v''] & \{\text{folding relational composition}\} \\
= & P \;;\; (\exists\, x \bullet Q) & \square
\end{array}
$$

Proof of Law 10. We assume that $x \notin \alpha Q$ and $P_{\oplus\{x\}}$ and Q are composable with interface $\{v, x\}$.

$$
\begin{array}{lll}
& P \;;\; Q_{\oplus\{x\}} & \{\text{unfolding relational composition}\} \\
= & \exists\, v'', x'' \bullet P[v', x' \backslash v'', x''] \wedge Q_{\oplus\{x\}}[v, x \backslash v'', x''] & \{\text{simplify substitution}\} \\
= & \exists\, v'', x'' \bullet P[v', x' \backslash v'', x''] \wedge Q[v \backslash v''] & \{\text{localising quantifier } \exists\, x'' \bullet \ldots\} \\
= & \exists\, v'' \bullet (\exists\, x'' \bullet P[v', x' \backslash v'', x'']) \wedge Q[v \backslash v''] & \{\text{extracting substitution}\} \\
= & \exists\, v'' \bullet (\exists\, x'' \bullet P[x' \backslash x''])[v' \backslash v''] \wedge Q[v \backslash v''] & \{\text{renaming quantified variable}\} \\
= & \exists\, v'' \bullet (\exists\, x' \bullet P)[v' \backslash v''] \wedge Q[v \backslash v''] & \{\text{folding relational composition}\} \\
= & (\exists\, x' \bullet P) \;;\; Q & \square
\end{array}
$$

We note that the above laws have also been proved in our mechanisation of the UTP in Isabelle/HOL [2, 1].

3 Proofs of unfolding lemmas

Table 2 recapture the unfolding lemmas presented in [4]. These lemmas facilitate the proof of recast relational and design laws in Section 4.

Proof of Lemma 1. Without loss of generality let $A = \{v, v', x, x'\}$ and $\alpha(x := e) = A$.

$$\begin{array}{lll} & x := e & \{\text{unfolding assignment}\} \\ = & \mathit{true} \vdash (e \sqsubseteq x' \wedge v \sqsubseteq v') & \{\textbf{Theorem 3.1.5} \text{ in } [3]\} \\ = & (\mathit{true} \vdash e \sqsubseteq x') \wedge (\mathit{true} \vdash v \sqsubseteq v') & \{\text{folding definition of } \mathbf{II}\} \\ = & (\mathit{true} \vdash e \sqsubseteq x') \wedge \mathbf{II}_{\{v,v'\}} & \{\text{using } A = \{v, v', x, x'\}\} \\ = & (\mathit{true} \vdash e \sqsubseteq x') \wedge \mathbf{II}_{A \setminus \{x,x'\}} & \Box \end{array}$$

Proof of Lemma 2. Without loss of generality let $A = \{v, v', x, x'\}$ and $\alpha(\mathbf{var}\, x) = A \setminus \{x\}$.

$$\begin{array}{lll} & \mathbf{var}\, x & \{\text{unfolding definition of } \mathbf{var}\} \\ = & \exists\, x \bullet \mathbf{II}_A & \{\text{unfolding definition of } \mathbf{II}\} \\ = & \exists\, x \bullet \mathit{true} \vdash (x \sqsubseteq x' \wedge v \sqsubseteq v') & \{\textbf{Theorem 3.1.5} \text{ in } [3]\} \\ = & \exists\, x \bullet (\mathit{true} \vdash x \sqsubseteq x') \wedge (\mathit{true} \vdash v \sqsubseteq v') & \{\text{localising quantifier}\} \\ = & (\exists\, x \bullet (\mathit{true} \vdash x \sqsubseteq x')_{\{x,x'\}}) \wedge (\mathit{true} \vdash v \sqsubseteq v')_{\{v,v'\}} & \{\text{using } x' \text{ as a witness}\} \\ = & \mathit{true}_{\{x'\}} \wedge (\mathit{true} \vdash v \sqsubseteq v')_{\{v,v'\}} & \{\text{folding definition of } \mathbf{II}\} \\ = & \mathit{true}_{\{x'\}} \wedge \mathbf{II}_{A \setminus \{x,x'\}} & \{\text{Law 3 in Table 1 and logic}\} \\ = & (\mathbf{II}_{A \setminus \{x,x'\}})_{\oplus \{x'\}} & \{\text{using } A \setminus \{x, x'\} = \alpha(\mathbf{var}\, x) \setminus \{x'\}\} \\ = & (\mathbf{II}_{\alpha(\mathbf{var}\, x) \setminus \{x'\}})_{\oplus \{x'\}} & \Box \end{array}$$

Proof of Lemma 3. Without loss of generality let $A = \{v, v', x, x'\}$ and $\alpha(\mathbf{end}\, x) = A \setminus \{x'\}$.

$$\begin{array}{lll} & \mathbf{end}\, x & \{\text{unfolding definition of } \mathbf{end}\} \\ = & \exists\, x' \bullet \mathbf{II}_A & \{\text{unfolding definition of } \mathbf{II}\} \\ = & \exists\, x' \bullet \mathit{true} \vdash (x \sqsubseteq x' \wedge v \sqsubseteq v') & \{\textbf{Theorem 3.1.5} \text{ in } [3]\} \\ = & \exists\, x' \bullet (\mathit{true} \vdash x \sqsubseteq x') \wedge (\mathit{true} \vdash v \sqsubseteq v') & \{\text{localising quantifier}\} \\ = & (\exists\, x' \bullet (\mathit{true} \vdash x \sqsubseteq x')_{\{x,x'\}}) \wedge (\mathit{true} \vdash v \sqsubseteq v')_{\{v,v'\}} & \{\text{using } x \text{ as a witness}\} \\ = & \mathit{true}_{\{x\}} \wedge (\mathit{true} \vdash v \sqsubseteq v')_{\{v,v'\}} & \{\text{folding definition of } \mathbf{II}\} \\ = & \mathit{true}_{\{x\}} \wedge \mathbf{II}_{A \setminus \{x,x'\}} & \{\text{Law 3 in Table 1 and logic}\} \\ = & (\mathbf{II}_{A \setminus \{x,x'\}})_{\oplus \{x\}} & \{\text{using } A \setminus \{x, x'\} = \alpha(\mathbf{end}\, x) \setminus \{x\}\} \\ = & (\mathbf{II}_{\alpha(\mathbf{end}\, x) \setminus \{x\}})_{\oplus \{x\}} & \Box \end{array}$$

Lemma 1. $x := e \ = \ (\mathit{true} \vdash e \sqsubseteq x') \wedge \mathbf{I\!I}_{A \setminus \{x,x'\}}$ **where** $A = \alpha(x := e)$
Lemma 2. $\mathbf{var}\, x \ = \ (\mathbf{I\!I}_{A \setminus \{x'\}})_{\oplus\{x'\}}$ **where** $A = \alpha(\mathbf{var}\, x)$
Lemma 3. $\mathbf{end}\, x \ = \ (\mathbf{I\!I}_{A \setminus \{x\}})_{\oplus\{x\}}$ **where** $A = \alpha(\mathbf{end}\, x)$

Table 2: Unfolding lemmas for higher-order UTP operators.

Law 11. $\mathbf{var}\, x \,;\, \mathbf{end}\, x \ = \ \mathbf{I\!I}$
Law 12. $x := E \,;\, \mathbf{end}\, x \ = \ \mathbf{end}\, x$
Law 13. $\mathbf{var}\, x \,;\, D_{+x} \ = \ D \,;\, \mathbf{var}\, x$ **provided** $\{x, x'\} \cap \alpha D = \varnothing$
Law 14. $D_{+x} \,;\, \mathbf{end}\, x \ = \ \mathbf{end}\, x \,;\, D$ **provided** $\{x, x'\} \cap \alpha D = \varnothing$

Table 3: Recast of relational laws adopted from [3] for higher-order UTP.

4 Proofs of recast relational theorems

Table 3 summarises relational laws required for some of the proofs in [4]. We note that we cannot take for granted that those laws, albeit proved for relations and designs in [3], persist to hold in HO UTP because of the modified definition of $\mathbf{I\!I}$, assignment, and alphabet extension (Table **??**). Redefining $\mathbf{I\!I}$ moreover indirectly incurs a change in the definition of variable blocks, since the semantics of $\mathbf{var}\, x$ and $\mathbf{end}\, x$ depend on $\mathbf{I\!I}$. We next present proofs of the laws in Table 3.

We first use Lemma 1 in [4] to prove the idempotence law $\mathbf{I\!I}_A \,;\, \mathbf{I\!I}_A \ = \ \mathbf{I\!I}_A$.

Law 15. *Let A be a homogeneous alphabet. Then,*

$$\mathbf{I\!I}_A \,;\, \mathbf{I\!I}_A \ = \ \mathbf{I\!I}_A$$

Proof. Without loss of generality let $A = \{v, v'\}$. Then,

$$\begin{array}{lll}
 & \mathbf{I\!I}_A \,;\, \mathbf{I\!I}_A & \{\text{unfolding definition of } \mathbf{I\!I}_A\} \\
= & (\mathit{true} \vdash v \sqsubseteq v') \,;\, (\mathit{true} \vdash v \sqsubseteq v') & \{\text{Corollary 1 in [4]}\} \\
= & \mathit{true} \vdash (v \sqsubseteq v' \,;\, v \sqsubseteq v') & \{\text{Lemma 1 in [4]}, \sqsubseteq \text{ is a preorder}\} \\
= & \mathit{true} \vdash v \sqsubseteq v' & \{\text{folding definition of } \mathbf{I\!I}_A\} \\
= & \mathbf{I\!I}_A & \square
\end{array}$$

The above law shows that the higher-order $\mathbf{I\!I}$ is a fixed point of **P1** and **P2** and thus a healthy predicate in HO UTP. We proceed proving the recast relational laws.

Name	Syntax	Definition in HO UTP	Caveat
Skip	$\mathbf{II}_A$	$\mathit{true} \vdash v \sqsubseteq v'$	$A = \{v, v'\}$
Assignment	$x :=_A e$	$\mathit{true} \vdash e \sqsubseteq x' \land v \sqsubseteq v'$	$A = \{x, x', v, v'\}$
Parallel composition	$(P_1 \vdash Q_1) \parallel (P_2 \vdash Q_2)$	$P_1 \land P_2 \vdash Q_1 \land Q_2$	$\alpha(P_1 \vdash Q_1) = \alpha(P_2 \vdash Q_2)$
Alphabet extension	D_{+v}	$D \parallel (\mathit{true} \vdash v \sqsubseteq v')$	$\{v, v'\} \cap \alpha D = \varnothing$
Variable declaration	$\mathbf{var}\, x$	$\exists\, x \bullet \mathbf{II}_A$	$\{x, x'\} \subseteq A$
Variable erasure	$\mathbf{end}\, x$	$\exists\, x' \bullet \mathbf{II}_A$	$\{x, x'\} \subseteq A$

Table 4: Modified definitions of operators in higher-order UTP.

Proof of Law 11. Without loss of generality, we assume $\alpha(\mathbf{var}\, x) = A \backslash \{x\}$ and $\alpha(\mathbf{end}\, x) = A \backslash \{x'\}$ for some homogeneous A with $\{x, x'\} \subseteq A$.

$$
\begin{aligned}
& \mathbf{var}\, x \,;\, \mathbf{end}\, x && \{\text{unfolding Lemma 2 and 3 in Table 2}\} \\
= \;& (\mathbf{II}_{A\backslash\{x,x'\}})_{\oplus\{x'\}} \,;\, (\mathbf{II}_{A\backslash\{x,x'\}})_{\oplus\{x\}} && \{\text{Law 8 in Table 1}\} \\
= \;& \mathbf{II}_{A\backslash\{x,x'\}} \,;\, \mathbf{II}_{A\backslash\{x,x'\}} && \{\text{Law 15}\} \\
= \;& \mathbf{II}_{A\backslash\{x,x'\}} && \square
\end{aligned}
$$

Proof of Law 12. We assume that e only references undashed variables.

$$
\begin{aligned}
& x := e \,;\, \mathbf{end}\, x && \{\text{unfolding assignment and Lemma 3}\} \\
= \;& (\mathit{true} \vdash e \sqsubseteq x' \land v \sqsubseteq v') \,;\, (\mathbf{II}_{\{v,v'\}})_{\oplus\{x\}} && \{\text{unfolding definition of } \mathbf{II}\} \\
= \;& (\mathit{true} \vdash e \sqsubseteq x' \land v \sqsubseteq v') \,;\, (\mathit{true} \vdash v \sqsubseteq v')_{\oplus\{x\}} && \{\text{Corollary 1 in [4]}\} \\
= \;& \mathit{true} \vdash (e \sqsubseteq x' \land v \sqsubseteq v') \,;\, (v \sqsubseteq v')_{\oplus\{x\}} && \{\text{unfolding relational composition}\} \\
= \;& \mathit{true} \vdash \exists\, x'', v'' \bullet (e \sqsubseteq x'' \land v \sqsubseteq v'') \land (v'' \sqsubseteq v')_{\oplus\{x''\}} && \{\text{localising quantifier}\} \\
= \;& \mathit{true} \vdash \exists\, v'' \bullet (\exists\, x'' \bullet e \sqsubseteq x'') \land v \sqsubseteq v'' \land v'' \sqsubseteq v' && \{\text{using } e \text{ as a witness}\} \\
= \;& \mathit{true} \vdash \exists\, v'' \bullet \mathit{true}_{\{x,v\}} \land v \sqsubseteq v'' \land v'' \sqsubseteq v' && \{\text{logic simplification}\} \\
= \;& (\mathit{true} \vdash \exists\, v'' \bullet v \sqsubseteq v'' \land v'' \sqsubseteq v')_{\oplus\{x\}} && \{\text{folding relational composition}\} \\
= \;& (\mathit{true} \vdash v \sqsubseteq v' \,;\, v \sqsubseteq v')_{\oplus\{x\}} && \{\text{Lemma 1 in [4]}, \sqsubseteq \text{ is a preorder}\} \\
= \;& (\mathit{true} \vdash v \sqsubseteq v')_{\oplus\{x\}} && \{\text{Lemma 3 in Table 2}\} \\
= \;& \mathbf{end}\, x && \square
\end{aligned}
$$

Proof of Law 13. Let D be a **P1** and **P2**-healthy design with $\alpha(D) = \{v, v'\}$.

$$
\begin{array}{lll}
 & \mathbf{var}\, x \,;\, D_{+x} & \{\text{every design can be written as } P \vdash Q\} \\
= & \mathbf{var}\, x \,;\, (P \vdash Q)_{+x} & \{\text{Lemma 2 in Table 2}\} \\
= & (\mathbf{II}_{\{v,v'\}})_{\oplus\{x'\}} \,;\, (P \vdash Q)_{+x} & \{\text{definition of alphabet extensions}\} \\
= & (\mathbf{II}_{\{v,v'\}})_{\oplus\{x'\}} \,;\, ((P \vdash Q) \,||\, (\mathit{true} \vdash x \sqsubseteq x')) & \{\text{unfolding parallel composition}\} \\
= & (\mathbf{II}_{\{v,v'\}})_{\oplus\{x'\}} \,;\, (P \wedge \mathit{true} \vdash Q \wedge x \sqsubseteq x') & \{\text{Law 9 in Table 1}\} \\
= & \mathbf{II}_{\{v,v'\}} \,;\, (\exists\, x \bullet P \vdash Q \wedge x \sqsubseteq x') & \{\text{localising quantifier, } x \notin \alpha(P \vdash Q)\} \\
= & \mathbf{II}_{\{v,v'\}} \,;\, (P \vdash Q \wedge \exists\, x \bullet x \sqsubseteq x') & \{\text{using } x' \text{ as a witness}\} \\
= & \mathbf{II}_{\{v,v'\}} \,;\, (P \vdash Q \wedge \mathit{true}_{\{x'\}}) & \{\text{extracting plain alphabet extension}\} \\
= & (\mathbf{II}_{\{v,v'\}} \,;\, (P \vdash Q))_{\oplus\{x'\}} & \{D \text{ is fixed point of } \mathbf{P2}\} \\
= & (P \vdash Q)_{\oplus\{x'\}} & \{D \text{ is fixed point of } \mathbf{P1}\} \\
= & (P \vdash Q \,;\, \mathbf{II}_{\{v,v'\}})_{\oplus\{x'\}} & \{\text{distributing plain alphabet extension}\} \\
= & P \vdash Q \,;\, (\mathbf{II}_{\{v,v'\}})_{\oplus\{x'\}} & \{\text{Lemma 2 in Table 2}\} \\
= & P \vdash Q \,;\, \mathbf{var}\, x & \square
\end{array}
$$

Proof of Law 14. Let D be a **P1** and **P2**-healthy design with $\alpha(D) = \{v, v'\}$.

$$
\begin{array}{lll}
 & D_{+x} \,;\, \mathbf{end}\, x & \{\text{every design can be written as } P \vdash Q\} \\
= & (P \vdash Q)_{+x} \,;\, \mathbf{end}\, x & \{\text{Lemma 3 in Table 2}\} \\
= & (P \vdash Q)_{+x} \,;\, (\mathbf{II}_{\{v,v'\}})_{\oplus\{x\}} & \{\text{definition of alphabet extensions}\} \\
= & ((P \vdash Q) \,||\, (\mathit{true} \vdash x \sqsubseteq x')) \,;\, (\mathbf{II}_{\{v,v'\}})_{\oplus\{x\}} & \{\text{unfolding parallel composition}\} \\
= & (P \wedge \mathit{true} \vdash Q \wedge x \sqsubseteq x') \,;\, (\mathbf{II}_{\{v,v'\}})_{\oplus\{x\}} & \{\text{Law 9 in Table 1}\} \\
= & (\exists\, x' \bullet P \vdash Q \wedge x \sqsubseteq x') \,;\, \mathbf{II}_{\{v,v'\}} & \{\text{localising quantifier, } x \notin \alpha(P \vdash Q)\} \\
= & (P \vdash Q \wedge \exists\, x' \bullet x \sqsubseteq x') \,;\, \mathbf{II}_{\{v,v'\}} & \{\text{using } x \text{ as a witness}\} \\
= & (P \vdash Q \wedge \mathit{true}_{\{x\}}) \,;\, \mathbf{II}_{\{v,v'\}} & \{\text{extracting plain alphabet extension}\} \\
= & ((P \vdash Q) \,;\, \mathbf{II}_{\{v,v'\}})_{\oplus\{x\}} & \{D \text{ is fixed point of } \mathbf{P1}\} \\
= & (P \vdash Q)_{\oplus\{x\}} & \{D \text{ is fixed point of } \mathbf{P2}\} \\
= & (\mathbf{II}_{\{v,v'\}} \,;\, P \vdash Q)_{\oplus\{x\}} & \{\text{distributing plain alphabet extension}\} \\
= & (\mathbf{II}_{\{v,v'\}})_{\oplus\{x\}} \,;\, P \vdash Q & \{\text{Lemma 3 in Table 2}\} \\
= & \mathbf{end}\, x \,;\, P \vdash Q & \square
\end{array}
$$

References

[1] S. Foster and R. Payne. Theorem Proving Support - Developers Manual. Technical Report D33.2b, COMPASS: Comprehensive Modelling for Advanced Systems of Systems, FP7 Grant Agreement 287829, September 2013. Available from http://www.compass-research.eu/deliverables.html.

[2] S. Foster, F. Zeyda, and J. Woodcock. Isabelle/UTP: A mechanised theory engineering framework , February 2014. Under review for *UTP 2014, 5th International Symposium on Unifying Theories of Programming*.

[3] C. A. R. Hoare and H. Jifeng. *Unifying Theories of Programming*. Prentice Hall Series in Computer Science. Prentice Hall, Upper Saddle River, NJ, USA, 1998.

[4] F. Zeyda and A. Cavalcanti. Laws for a unified theory of methods, February 2014. Under review for *UTP 2014, 5th International Symposium on Unifying Theories of Programming*.