Appendix

8. Interaction Equivalence

The Program Translation is defined in Figure 12. The basic definitions are depicted in Figure 13. The configuration translation is defined in Figure 14.

The translation removes sharing and translates automatic updates to pure message passing. The translation function $s2p$, maps a program ${}^{s}p$ (that potentially uses sharing) to ${}^{p}p$ that does not use sharing (EQ. 42). We call ${}^{s}p$ and ${}^{p}p$ sharing and pure programs respectively. The sharing library ${}^{s}Lib$ is translated to the pure library ${}^{p}Lib \triangleq {}^{s}Lib,\ ws2p({}^{s}Lib),\ rs2p({}^{s}Lib)$ (EQ. 43). The functions $ws2p$ (EQ. 44 - EQ. 53) and $rs2p$ (EQ. 54 - EQ. 56) translate each behavior bid in ${}^{s}Lib$ to bid_w and bid_r respectively. (bid_w and bid_r are just

---

$s2p: P \to P$

$s2p(\text{program}(receptionists: \rho, externals: \mathcal{X},$
$\quad library: {}^{s}Lib,$
$\quad actors: \{a_i \mapsto e_i\}_{1 \le i \le m},$
$\quad messages: \{a'_i \lhd m_i\}_{1 \le i \le n},$
$\quad sharing: \langle a_w, A_r \rangle)) \triangleq \qquad A_r = \{a_{r_{i=1..R}}\}$

$\text{program}(receptionists: \rho, externals: \mathcal{X},$ EQ. 42
$\quad library: {}^{p}Lib,$
$\quad actors: \{a_i \mapsto e_i\}_{1 \le i \le m, a_i \notin \{a_w\} \cup A_r},$
$\qquad \{a_i \mapsto rs2p(e_i, 1)\}_{1 \le i \le m, a_i \in A_r},$
$\qquad a_w \mapsto ws2p(e_w, 1),$
$\quad messages: \{a'_i \lhd m_i\}_{1 \le i \le n},$
$\quad sharing:)$

Where

${}^{p}Lib \triangleq {}^{s}Lib, ws2p({}^{s}Lib), rs2p({}^{s}Lib)$ EQ. 43

$ws2p\big(\overline{behDef}\big) \triangleq \overline{ws2p(behDef)}$ EQ. 44

$ws2p\big(\text{behavior } id(\overline{x})\ \overline{methodDef}\big) \triangleq \text{behavior } id_w(\overline{x}, c_w)\ \overline{ws2p(methodDef)}$ EQ. 45

$ws2p(\text{method } id(\overline{x})[\text{enable } e]\ e') \triangleq \text{ method } id(\overline{x})[\text{enable } ws2p(e)]\ ws2p(e')$ EQ. 46

$ws2p(\lambda x. e) \triangleq \lambda x. ws2p(e)$ EQ. 47

$ws2p(e_1\ e_2) \triangleq ws2p(e_1)\ ws2p(e_2)$ EQ. 48

$ws2p(e \lhd [\overline{e}]) \triangleq ws2p(e) \lhd \left[\overline{ws2p(e)}\right]$ EQ. 49

$ws2p\left(ready\big(behId(\overline{e})\big)\right) \triangleq$
$\quad let\ \{\overline{x \coloneqq e}\}$
$\qquad let\ \left\{x'_i \coloneqq a_{r_i} \lhd Update\left(pair\ c_w\ \lambda x''. ready\left(behId_r\big(\overline{x}, inc(c_w)\big)\right)\right)\right\}_{i=1..R} \quad fresh(x'_i)_{i=1..R}$ EQ. 50
$\qquad\quad ready\left(behId_w\big(\overline{x}, inc(c_w)\big)\right)$

$ws2p(\text{self}) \triangleq \text{self}$ EQ. 51

$ws2p(x) \triangleq x \quad x \in X$ EQ. 52

$ws2p(v) \triangleq v \quad v \in V$ EQ. 53

$rs2p\big(\overline{behDef}\big) \triangleq \overline{rs2p(behDef)}$ EQ. 54

$rs2p\big(\text{behavior } id(\overline{x})\ \overline{methodDef}\big) \triangleq \text{behavior } id_r(\overline{x}, c_r)\ \overline{rs2p(methodDef)}\ uMethod$ EQ. 55

$uMethod \triangleq \text{method } Update(x)\ [\text{enable } = (fst\ x)\ c_r]\ (snd\ x)\ (\lambda x. x)$ EQ. 56

$rs2p(\text{method } id(\overline{x})[\text{enable } e]\ e') \triangleq \text{ method } id(\overline{x})[\text{enable } rs2p(e)]\ rs2p(e')$ EQ. 57

$rs2p(\lambda x. e) \triangleq \lambda x. rs2p(e)$ EQ. 58

$rs2p(e_1\ e_2) \triangleq rs2p(e_1)\ rs2p(e_2)$ EQ. 59

$rs2p(e \lhd [\overline{e}]) \triangleq rs2p(e) \lhd \left[\overline{rs2p(e)}\right]$ EQ. 60

$rs2p\left(ready\big(behId(\overline{e})\big)\right) \triangleq ready\big(behId_r(\overline{e}, c_r)\big)$ EQ. 61

$rs2p(\text{self}) \triangleq \text{self}$ EQ. 62

$rs2p(x) \triangleq x \quad x \in X$ EQ. 63

$rs2p(v) \triangleq v \quad v \in V$ EQ. 64

Figure 12. Program Translation

$true \triangleq \lambda tf.t$

$false \triangleq \lambda tf.f$

$\neg \triangleq \lambda b.b\ false\ true$

$\wedge \triangleq \lambda bc.b\ c\ false$

$pair \triangleq \lambda fsb.f\ s\ b$

$fst \triangleq \lambda p.p\ true$

$snd \triangleq \lambda p.p\ false$

$0 \triangleq \lambda sz.z$

$1 \triangleq \lambda sz.s(z)$

$2 \triangleq \lambda sz.s\big(s(z)\big)$

$succ \triangleq \lambda nsz.s(n\ s\ z)$

$plus \triangleq \lambda mnsz.m\ s\ (n\ s\ z)$

$inc \triangleq \lambda msz.plus\ m\ 1\ s\ z$

$iszero \triangleq$
$\lambda m.m\ (\lambda x.false)\ true$

$zz \triangleq pair\ 0\ 0$

$ss \triangleq$
$\lambda p.pair\ (snd\ p)\ \big(succ\ (snd\ p)\big)$

$pred \triangleq \lambda m.fst\ (m\ ss\ zz)$

$\geq \triangleq \lambda xy.iszero\ (x\ pred\ y)$

$= \triangleq \lambda xy.\wedge\ (\geq xy)(\geq yx)$

Figure 13. Basic Definitions

new names for behaviors that are obtained from translation of bid.) Let a_w be the writer actor and A_r be the set of reader actors in ${}^{s}p$. The behavior of the writer actor a_w is translated to use bid_w behaviors and the behavior of each reader actor $a_r \in A_r$ is translated to use bid_r behaviors.

The interesting part of translation is the translation of $ready$ for the writer (EQ. 50) and the addition of $uMethod$ to behaviors for the readers (EQ. 55 and EQ. 56). The reduction of $ready$ for a_w (EQ. 13), installs the new behavior for a_w and stores it at the tail of the update store u of each a_r. Later, in $update(a_r)$ transitions (EQ. 15), the behavior at the head of u is taken and installed as the current behavior of a_r. To process updates in order, new behaviors are added at the tail and removed from the head of the update store. The translation simulates the update mechanism with message passing. A $ready$ expression of a_w is translated to sending $Update$ messages to reader actors a_r and then installing the new behavior for a_w (EQ. 50). $Update$ messages contain the new behavior for the reader actors. $uMethod$ receives $Update$ messages at the reader actors. To preserve the order of updates, the update messages should be processed in the same order as they are sent. The translation adds a counter parameter to the definition of the writer and readers behaviors (EQ. 45 and EQ. 55). The writer actor maintains a counter c_w that holds the number of the next update to send and each reader actor a_r maintains the number of the next update message to receive c_r. The selective receive feature of the language is used to check the number of the messages. The $Update$ messages that a_w sends contain a pair of expressions. The first element of the pair is the number of the update message and the second element is a thunk holding the new behavior. The $uMethod$ (EQ. 56) is defined to check the equality of the first element of the pair (that is the number of the message) to c_r (that is the number of the next update message to receive). If the check succeeds, the second element of the pair is applied to a dummy value. The application results in opening the thunk and getting the $ready$ expression that installs the new behavior. c_w and c_r are initialized to 1 for the writer actor and all the reader actors (EQ. 42). The translation of $ready$ for a_w (EQ. 50) passes the value $inc(c_w)$ for the parameter c_w of the new behavior $behId_w$ that is installed for a_w and also passes $inc(c_w)$ as the c_r parameter of the new behavior $behId_r$ that it sends to a_rs. This essentially increments the number of the next message to be sent by a_w and sets the number of the next message to be received by a_rs.

The configuration translation is defined in Figure 14. To reason about intermediate configurations, we lift the definition of translations to the behaviors component B of actor states in a configuration.

$rs2p(\langle e, e', id(\overline{v})\rangle, c_r) = \langle rs2p(e, c_r), rs2p(e', c_r), rs2p(id(\overline{v}), c_r)\rangle$
$rs2p(\lambda x.e, c_r) \triangleq \lambda x.rs2p(e, c_r)$
$rs2p(e_1\ e_2, c_r) \triangleq rs2p(e_1, c_r)\ rs2p(e_2, c_r)$
$rs2p(e \lhd [\overline{e}], c_r) \triangleq rs2p(e, c_r) \lhd \left[\overline{rs2p(e, c_r)}\right]$
$rs2p\big(ready\big(behId(\overline{e})\big), c_r\big) \triangleq ready\big(behId_r(\overline{e}, c_r)\big)$
$rs2p(\mathrm{self}, c_r) \triangleq \mathrm{self}$
$rs2p(x, c_r) \triangleq x \quad x \in X$
$rs2p(v, c_r) \triangleq v \quad v \in V$
$rs2p(id(\overline{v}), c_r) \triangleq id_r(\overline{v}, c_r)$ EQ. 65

$ws2p(\langle e, e', id(\overline{v})\rangle, c_w) = \langle ws2p(e, c_r), ws2p(e', c_r), ws2p(id(\overline{v}), c_w)\rangle$
$ws2p(\lambda x.e, c_w) \triangleq \lambda x.ws2p(e, c_w)$
$ws2p(e_1\ e_2, c_w) \triangleq ws2p(e_1, c_w)\ ws2p(e_2, c_w)$
$ws2p(e \lhd [\overline{e}], c_w) \triangleq ws2p(e, c_w) \lhd \left[\overline{ws2p(e, c_w)}\right]$
$ws2p\big(ready\big(behId(\overline{e})\big), c_w\big) \triangleq$
$let\ \{\overline{x := e}\}$
$let\ \left\{x_i' := a_{r_i} \lhd Update\left(pair\ c_w\ \lambda x''.ready\left(behId_r\big(\overline{x}, inc(c_w)\big)\right)\right)\right\}_{i=1..R} \quad fresh(x_i')_{i=1..R}$ EQ. 66
$ready\left(behId_w\big(\overline{x}, inc(c_w)\big)\right)$
$ws2p(\mathrm{self}, c_w) \triangleq \mathrm{self}$
$ws2p(x, c_w) \triangleq x \quad x \in X$
$ws2p(v, c_w) \triangleq v \quad v \in V$
$ws2p(id(\overline{v}), c_w) \triangleq id_w(\overline{v}, c_w)$ EQ. 67

$rs2p([\![\ ]\!], c_r) \triangleq [\![\ ]\!]$
$\boldsymbol{ws2p}([\![\ ]\!], \boldsymbol{c_w}) \triangleq [\![\ ]\!]$

Figure 14. Configuration Translation

$[\mathcal{R}[\![\text{ready}(bid(\overline{v}))]\!], q_u]_{a_w}, [b,q,u]_{a_{r_{i=1..R}}}$	$\xRightarrow{ready(a_w)}$ $[bid(\overline{v}), \langle q_u, [\,]\rangle]_{a_w}, [b,q,u :: bid(\overline{v})]_{a_{r_{i=1..R}}}$
$[ws2p(\mathcal{R}[\![\text{ready}(bid(\overline{v}))]\!], c_w), q_u]_{a_w}$	$=$ $[ws2p(\mathcal{R}, c_w)[\![e]\!], q_u]_{a_w}$ where $e = let\ \left\{x_i' := a_{r_i} \lhd Update\left(pair\ c_w\ \lambda x''.\text{ready}\left(bid_r(\overline{v}, inc(c_w))\right)\right)\right\}_{i=1..R}$ $\quad \text{ready}\left(bid_w(\overline{v}, inc(c_w))\right)$ $\xRightarrow{(send(a_w), seq(a_w))^*}$ $\left[ws2p(\mathcal{R}, c_w)\left[\!\left[\text{ready}\left(bid_w(\overline{v}, inc(c_w))\right)\right]\!\right], q_u\right]_{a_w},$ $a_{r_i} \lhd Update\left(pair\ c_w\ \lambda x''.\text{ready}\left(bid_r(\overline{v}, inc(c_w))\right)\right)_{i=1..R}$ $\xRightarrow{ready(a_w)}$ $\left[bid_w(\overline{v}, inc(c_w)), q_u\right]_{a_w},$ $a_{r_i} \lhd Update\left(pair\ c_w\ \lambda x''.\text{ready}\left(bid_r(\overline{v}, inc(c_w))\right)\right)_{i=1..R}$ $=$ $\left[ws2p(bid(\overline{v}), inc(c_w)), q_u\right]_{a_w},$ $a_{r_i} \lhd Update\left(pair\ c_w\ \lambda x''.\text{ready}\left(bid_r(\overline{v}, inc(c_w))\right)\right)_{i=1..R}$

Figure 15. $\boldsymbol{ready(a_w)}$

LEMMA 1:
$rs2p(\mathcal{R}[\![e]\!], c_r) = rs2p(\mathcal{R}, c_r)[\![rs2p(e, c_r)]\!]$

LEMMA 2:
$ws2p(\mathcal{R}[\![e]\!], c_w) = ws2p(\mathcal{R}, c_w)[\![ws2p(e, c_w)]\!]$

THEOREM 1:
$\forall p \in P: Isem([\![p]\!]) = Isem([\![s2p(p)]\!])$

PROOF:
We want to prove the interaction equivalence of every program and its translated program that is $\forall p \in P: Isem([\![p]\!]) = Isem([\![s2p(p)]\!])$. Consider an arbitrary program ${}^{s}p$ (potentially using sharing) and its translation ${}^{p}p = s2p({}^{s}p)$ that does not use sharing. We call ${}^{s}p$ and ${}^{p}p$ sharing and pure programs respectively. The configurations corresponding to ${}^{s}p$ and ${}^{p}p$ are $[\![{}^{s}p]\!]$ and $[\![{}^{p}p]\!]$. We need to show that $[\![{}^{s}p]\!]$ and $[\![{}^{p}p]\!]$ are interaction equivalent. We show that the interaction semantics set of each is included in the other. For the sake of simplicity, first, we concentrate on the transition correspondence and then show the fairness.

For the forward direction, we need to show that for every computation path ${}^{s}\pi$ starting from $[\![{}^{s}p]\!]$, there is a computation path ${}^{p}\pi$ starting from $[\![{}^{p}p]\!]$ with the same interaction semantics. For each ${}^{s}\pi$, we construct a path ${}^{p}\pi$ inductively. We define an invariant that holds between the initial configurations $[\![{}^{s}p]\!]$ and $[\![{}^{p}p]\!]$. Assuming the invariant between ${}^{s}K_1$ and ${}^{p}K_1$, for each step ${}^{s}\pi_i = {}^{s}K_1 \to {}^{s}K_2$, we introduce a multi-step ${}^{p}\pi_i = {}^{p}K_1 \to^* {}^{p}K_n$ such that the invariant between ${}^{s}K_2$ and ${}^{p}K_n$ is preserved. The invariant between ${}^{s}K$ and ${}^{p}K$ ensures that there is a correspondence between the actors of ${}^{s}K$ and ${}^{p}K$ such that if a transition is possible from ${}^{s}K$, a corresponding transition is possible from ${}^{p}K$ specifically if $update(a_r)$ is enabled in ${}^{s}K$, the selective receive condition is enabled for the corresponding $Update$ message in ${}^{p}K$. The invariant states that: If the state of the writer actor a_w is $[b,q]$ in ${}^{s}K$, there exists c_w such that the state of a_w is $[ws2p(b, c_w), q]$ in ${}^{p}K$. If the state of a reader actor a_r is $[b,q,u]$ in ${}^{s}K$, there exists c_r such that the state of a_r is $[rs2p(b, c_r), q]$ in ${}^{p}K$. If the element at position $i \in \{0..n\}$ of u of a_r is $bid_i(\overline{v})$ in ${}^{s}K$, there is a message $Update\left(pair\ c\ \lambda x.ready\left(bid_{i_r}(\overline{v}, inc(c))\right)\right)$ in ${}^{p}K$ such that $c = plus\ c_r\ i$. If there is a message in ${}^{s}K$, the same message is in ${}^{p}K$. In addition, $c_w = plus\ c_r\ |u|$.

Let us look at the corresponding transition sequences. The interesting cases are transitions with labels $ready(a_w)$ and $update(a_r)$. There is a one-to-one mapping for other transitions.

Consider Figure 15. If ${}^{s}\pi = {}^{s}K_1 \to {}^{s}K_2$ has the label $ready(a_w)$ (EQ. 13), $bid(\overline{v})$ is backed up at the tail of the update list of each a_r. The behavior component of a_w is $\mathcal{R}[\![\text{ready}(bid(\overline{v}))]\!]$ in ${}^{s}K_1$, thus, by the invariant, there exists c_w such that the behavior of a_w is $ws2p(\mathcal{R}[\![\text{ready}(bid(\overline{v}))]\!], c_w)$ in ${}^{p}K_1$. Thus, by LEMMA 2, the behavior of a_w is $ws2p(\mathcal{R}, c_w)[\![e]\!]$ where e is defined by EQ. 66. By $send(a_w)$ transitions, an $Update$ message is sent to each a_r and finally a $ready(a_w)$ transition installs the new behavior.

The resulting behavior of a_w in ${}^{p}K_n$ is $bid_w(\overline{v}, inc(c_w))$. By EQ. 67, $bid_w(\overline{v}, inc(c_w))$ is equal to $ws2p(bid(\overline{v}), inc(c_w))$. Thus, the resulting behavior of a_w in ${}^{p}K_n$ is $ws2p(bid(\overline{v}), inc(c_w))$. The new counter of a_w is $c_w' = inc(c_w)$.

An element $bid(\overline{v})$ is added at position $|u|$ of each reader in ${}^{s}K_2$. The sent message is $Update\left(pair\ c_w\ \lambda x.ready\left(bid\left(\overline{v}, inc(c_w)\right)\right)\right)$. By the invariant, we have $c_w = plus\ c_r\ |u|$. Thus, there is a message $Update\left(pair\ c\ \lambda x.ready\left(bid\left(\overline{v}, inc(c)\right)\right)\right)$ where $c = plus\ c_r\ |u|$ in ${}^{p}K_n$.

By the invariant, we have $c_w = plus\ c_r\ |u|$. The new update list of a_{r_i} is $u :: bid(\overline{v})$. We need to show that the invariant is preserved that is $c'_w = plus\ c_r\ |u :: bid(\overline{v})|$. This is obvious from $c'_w = inc(c_w)$ and $|u :: bid(\overline{v})| = inc(|u|)$. The interaction semantics of the two paths are equivalent as there is no in or out labels in any of them.

Consider Figure 16. If ${}^{s}\pi = {}^{s}K_1 \rightarrow {}^{s}K_2$ has the label $update(a_r)$ (EQ. 15), the behavior $bid(\overline{v})$ at the head of the update list u is installed as the current behavior. By the invariant, there is a message $Update\left(pair\ c_0\ \lambda x.ready\left(bid_r\left(\overline{v}, inc(c_0)\right)\right)\right)$ in ${}^{p}K_1$ such that $c_0 = plus\ c_r\ 0$. This means that $Update\left(pair\ c_r\ \lambda x.ready\left(bid_r\left(\overline{v}, inc(c_r)\right)\right)\right)$ is in ${}^{p}K_1$. The $Update$ message can be delivered and processed by $deliver(a_r)$ and $traverse(a)$ transitions. The selective receive clause of $uMethod$ (EQ. 56) checks equality of c_r with the first element of the pair contained in the $Update$ message (that is c_r). By a sequence of $check(a_r)$ transitions, the selective receive expression is evaluated to true and so the $enable(a_r)$ rule can install the body of $uMethod$. By the sequence of a $seq(a_r)$ and then a $ready(a_r)$ transition, The body of $uMethod$ is reduced to $bid_r\left(\overline{v}, inc(c)\right)$. By EQ. 65, $bid_r\left(\overline{v}, inc(c)\right)$ is equivalent to $rs2p\left(bid(\overline{v}), inc(c_r)\right)$. The new counter for a_r is $c'_r = inc(c_r)$.

By the invariant, we have $c_w = plus\ c_r\ |bid(\overline{v}) :: u|$. We need to show that $c_w = plus\ c'_r\ |u|$. This is obvious from $c'_r = inc(c_r)$ and $|bid(\overline{v}) :: u| = inc(|u|)$. The interaction semantics of the two paths are equivalent as there is no in or out labels in any of them.

For the reverse direction, we need to show that for every computation path ${}^{p}\pi$ that is resulted starting from $[\![{}^{p}p]\!]$, there is a computation path ${}^{s}\pi$ starting from $[\![{}^{s}p]\!]$ such that $Isem({}^{s}\pi) = Isem({}^{p}\pi)$.

For every path ${}^{p}\pi$, there is a path ${}^{p}\pi'$ where the selective receive guards for $Update$ messages are never disabled and $Isem({}^{p}\pi') = Isem({}^{p}\pi)$. The path ${}^{p}\pi'$ can be constructed from ${}^{p}\pi$ by removing all the $Update$ message delivery and failed guard evaluations from ${}^{p}\pi$ and delivering each $Update$ message just before its successful guard evaluation in ${}^{p}\pi$. Thus, in the rest of the proof, it is assumed that no selective receive guard for $Update$ messages is disabled in ${}^{p}\pi$.

We define a big-step transformation on ${}^{p}\pi$ that results in the equivalent big-step path ${}^{p}\pi^{\dagger}$.

The translation function $ws2p$ translates each $ready\left(behId(\overline{v})\right)$ expression for a_w to the following expression (EQ. 66):

$$let\ \left\{x'_i \coloneqq a_{r_i} \lhd Update\left(pair\ c_w\ \lambda x''.ready\left(behId_r\left(\overline{v}, inc(c)\right)\right)\right)\right\}_{i=1..R} \quad fresh(x'_i)_{i=1..R}$$
$$ready\left(behId_w\left(\overline{v}, inc(c_w)\right)\right)$$

The big-step transformation takes back each transition of the expression above just after its preceding transition. In the big step form, transitions of the expression above are in sequence.

The translation function $rs2p$ adds $uMothod$ to behavior definitions (EQ. 55). $uMethod$ is defined (EQ. 56) as follows:

$uMethod \triangleq$ mehod $Update(x)$ [enable $= (fst\ x)\ c_r$] $(snd\ x)\ (\lambda x.x)$

Consider the delivery $deliver(a_r, m)$, traverse $traverse(a_r)$, constraint check $check(a_r)^{*}\ enable(a_r)$ and body transitions $seq(a_r)\ ready(a_r)$ of $uMethod$ for an $Update$ message. The big-step transformation moves left each of these transitions to the right of its preceding transition.

Each of the two above sequences of transitions and each other transition is a step in the big-step path.

We show that the transitions that are moved commute to the left. Each transition is moved to the left until it reaches its preceding transition by the same actor. All the transitions that are passed are on other actors. Thus, condition of EQ. 38 is satisfied. The only transition

$[b, \langle[\,], q_r\rangle, bid(\overline{v}) :: u]_{a_r}$	$\xRightarrow{update(a_r, bid(\overline{v}))}$ $[bid(\overline{v}), \langle q_r, [\,]\rangle, u]_{a_r}$
$[rs2p(b, c_r), \langle[\,], q_r\rangle]_{a_r}, a_r \lhd m$ where $m = Update($ $pair$ c_r $\lambda x''.\text{ready}\left(bid_r\left(\overline{v}, inc(c_r)\right)\right)$ $)$	$\xRightarrow{deliver(a_r, m)}$ $[rs2p(b, c_r), \langle[m], q_r\rangle]_{a_r} \xRightarrow{traverse(a_r)}$ $[\langle= (fst\ m)\ c_r, (snd\ m)\ (\lambda x.x), rs2p(b, c_r)\rangle, \langle m, [\,], q_r\rangle]_{a_r} \xRightarrow{check(a_r)^{*}}$ $[\langle= c_r\ c_r, (snd\ m)\ (\lambda x.x), rs2p(b, c_r)\rangle, \langle m, [\,], q_r\rangle]_{a_r} \xRightarrow{check(a_r)}$ $[\langle\text{true}, (snd\ m)\ (\lambda x.x), rs2p(b, c_r)\rangle, \langle m, [\,], q_r\rangle]_{a_r} \xRightarrow{enable(a_r)}$ $[(snd\ m)\ (\lambda x.x), q_r]_{a_r} \xRightarrow{seq(a_r)}$ $\left[\text{ready}\left(bid_r\left(\overline{v}, inc(c_r)\right)\right), q_r\right]_{a_r} \xRightarrow{ready(a_r)}$ $\left[bid_r\left(\overline{v}, inc(c_r)\right), \langle q_r, [\,]\rangle\right]_{a_r} =$ $\left[rs2p\left(bid(\overline{v}), inc(c_r)\right), \langle q_r, [\,]\rangle\right]_{a_r}$

Figure 16. $update(a_r)$

in the transitions above that consumes a message is $deliver(a)$ that is the first transition in the sequence of receiving an $Update$ message. The transitions that move to the left consume no messages. Thus, EQ. 39 is satisfied. All the above transitions are silent transitions. Thus, EQ. 40 is satisfied. Therefore, the moved transitions commute. Thus, the big-step path ${}^{p}\pi^{\dagger}$ is equal to ${}^{p}\pi$. Thus, for every path ${}^{p}\pi$, there exists a path ${}^{p}\pi^{\dagger}$ where $Isem({}^{p}\pi^{\dagger}) = Isem({}^{p}\pi)$.

Now, we show that for every big-step path ${}^{p}\pi^{\dagger}$ starting from $[\![{}^{p}p]\!]$, there is a path ${}^{s}\pi$ starting from $[\![{}^{s}p]\!]$ such that $Isem({}^{s}\pi) = Isem({}^{p}\pi^{\dagger})$. Similar to the forward direction, for each path ${}^{p}\pi$, we construct a path ${}^{s}\pi$ inductively. We define an invariant that holds between the initial configurations $[\![{}^{p}p]\!]$ and $[\![{}^{s}p]\!]$. Assuming the invariant between ${}^{p}K_1$ and ${}^{s}K_1$, for each big-step ${}^{p}\pi_i = {}^{p}K_1 \rightarrow^{*} {}^{p}K_n$, we introduce a step ${}^{s}\pi_i = {}^{s}K_1 \rightarrow {}^{s}K_2$ such that the invariant between ${}^{p}K_n$ and ${}^{s}K_2$ is preserved.

The invariant between ${}^{p}K$ and ${}^{s}K$ ensures that there is a correspondence between the actors of ${}^{p}K$ and ${}^{s}K$ such that if a big-step is possible from ${}^{p}K$, a corresponding transition is possible from ${}^{s}K$ specifically if an $Update$ message is received by a_r in ${}^{p}K$, then the label $update(a_r)$ is enabled in ${}^{s}K$ and the first element of the update list of a_r in ${}^{s}K$ corresponds to the $Update$ message. The invariant states that: If the state of the writer actor a_w is $[b', q]$ in ${}^{p}K$, there exists b and c_w such that $b' = ws2p(b, c_w)$ and the state of a_w is $[b, q]$ in ${}^{s}K$. If the state of a reader actor a_r is $[b', q]$ in ${}^{p}K$, there exists b, c_r and u such that $b' = rs2p(b, c_r)$ and the state of a_r is $[b, q, u]$ in ${}^{s}K$. There exists n such that the set of update messages for a_r in ${}^{p}K$ is $Update\left(pair\ c_i\ \lambda x.ready\left(bid_{i_r}(\overline{v}, inc(c_i))\right)\right)_{i=0..n-1}$ such that $c_i = plus\ c_r\ i$ and the element at position $i \in \{0..n-1\}$ of u of a_r is $bid_i(\overline{v})$ in ${}^{s}K$. If there is a message other than $Update$ messages in ${}^{p}K$, the same message is in ${}^{s}K$. In addition, $c_w = plus\ c_r\ |u|$.

The interesting cases are the two big steps for installing a new behavior of the writer and receiving an $Update$ message. There is a one-to-one mapping for other transitions.

We consider the case that ${}^{p}\pi_i = {}^{p}K_1 \rightarrow^{*} {}^{p}K_n$ is the big step for installing a new behavior for the writer actor. Consider Figure 15, row 2. The result of the transitions is $\left[ws2p\left(bid(\overline{v}), inc(c_w)\right), q_u\right]_{a_w}, a_{r_i} \lhd Update\left(pair\ c_w\ \lambda x''.\text{ready}\left(bid_r(\overline{v}, inc(c_w))\right)\right)_{i=1..R}$. From the invariant, we have that the set of update messages for a_r is $Update\left(pair\ c_i\ \lambda x.ready\left(bid_{i_r}(\overline{v}, inc(c_i))\right)\right)_{i=0..n-1}$ in ${}^{p}K_1$. From the invariant, we have $c_w = plus\ c_r\ |u|$ that is $c_w = plus\ c_r\ n$. The new $Update$ message for each a_r is $Update\left(pair\ c_w\ \lambda x''.\text{ready}\left(bid_r(\overline{v}, inc(c_w))\right)\right)$. Thus, we have that the new $Update$ message for each a_r is $Update\left(pair\ c_n\ \lambda x''.\text{ready}\left(bid_r(\overline{v}, inc(c_n))\right)\right)$ where $c_n = plus\ c_r\ n$. Thus, the set of update messages for a_r in ${}^{p}K_n$ is $Update\left(pair\ c_i\ \lambda x.\text{ready}\left(bid_{i_r}(\overline{v}, inc(c_i))\right)\right)_{i=0..n}$ where $bid_n = bid$. By the invariant, as the writer is of the form $\left[ws2p\left(\mathcal{R}[\![\text{ready}(bid(\overline{v}))]\!], c_w\right), q_u\right]_{a_w}$ in ${}^{p}K_1$, it is of the form $\left[\mathcal{R}[\![\text{ready}(bid(\overline{v}))]\!], q_u\right]_{a_w}$ in ${}^{s}K_1$. Let ${}^{s}K_1 \rightarrow {}^{s}K_2$ be the transition by label $ready(a_w)$. By $ready(a_w)$ rule, shown in Figure 15, row 1, the writer becomes $[bid(\overline{v}), \langle q_u, [\,]\rangle]_{a_w}$ and each reader a_r becomes $[b, q, u']_{a_r}$ where $u' = u :: bid(\overline{v})$. Now, the element at location n of u' is $bid(\overline{v})$. As we defined $bid_n = bid$, the element at location n of u' is $bid_n(\overline{v})$. From the invariant we have, $c_w = plus\ c_r\ |u|$. The new counter c_w' of the writer a_w is $c_w' = inc(c_w)$ and $|u'| = inc(|u|)$. Thus, we have $c_w' = plus\ c_r\ |u'|$. The interaction semantics of the two paths are equivalent as all the transitions are internal.

We consider the case that ${}^{p}\pi_i = {}^{p}K_1 \rightarrow^{*} {}^{p}K_n$ is the big step for receiving an $Update$ message. Consider Figure 16, row 2. By the invariant, the set of update messages in ${}^{p}K_1$ is $Update\left(pair\ c_i\ \lambda x.ready\left(bid_{i_r}(\overline{v}, inc(c_i))\right)\right)_{i=0..n-1}$ such that $c_i = plus\ c_r\ i$. The first $Update$ message from this sequence is consumed. The result of the transitions is $\left[rs2p\left(bid_0(\overline{v}), inc(c_r)\right), \langle q_r, [\,]\rangle\right]_{a_r}$. The new counter c_r' of a_r is $inc(c_r)$. The set of messages in ${}^{p}K_n$ is $Update\left(pair\ c_i\ \lambda x.ready\left(bid_{i_r}(\overline{v}, inc(c_i))\right)\right)_{i=1..n-1}$ such that $c_i = plus\ c_r\ i$. From $c_r' = inc(c_r)$ we have, the set of messages in ${}^{p}K_n$ is $Update\left(pair\ c_i\ \lambda x.ready\left(bid_{i_r}(\overline{v}, inc(c_i))\right)\right)_{i=0..m-1}$ such that $c_i = plus\ c_r'\ i$ and $m = n - 1$. By the invariant, as $[rs2p(b, c_r), \langle [\,], q_r\rangle]_{a_r}$ is the state of a_r in ${}^{p}K_1$, its state is $[b, \langle [\,], q_r\rangle, u]_{a_r}$ in ${}^{s}K_1$ where the element at position 0 of u is $bid_0(\overline{v})$ i.e. $u = bid_0(\overline{v}) :: u'$. Let ${}^{s}K_1 \rightarrow {}^{s}K_2$ be the transition by label $update(a_r)$. As Figure 16, row 1 shows an $update(a_r)$ transition takes a_r to state $[bid_0(\overline{v}), \langle q_r, [\,]\rangle, u']_{a_r}$. By the invariant, we have that the element at position $i \in \{0..n-1\}$ of u is bid_i. It is trivial that the element at position $i \in \{1..n-1\}$ of u is bid_i that is the element at position $i \in \{0..n-2\}$ of u' is bid_i. Thus, the element at position $i \in \{0..m-1\}$ of u' (where $m = n - 1$) is bid_i.

8.1. Fairness

To reason about fairness, we need to show correspondence between enabled and fired labels of the sharing and translated pure programs. To have a correspondence, we need to have counter values in the labels of the original semantics. Consider the following transition rules that only add dummy counters for the number of reductions of ready expressions for each actor and the number of updates that reader actors have installed.
The state of actor is augmented with a number at the end to count the number of $ready$ reductions. U is augmented with a expression to count the number of installed updates. U is a pair of a number and a sequence of updates for a reader actor. The first element of the pair is the number of the past updates and the second one is the waiting updates.
$S = B, Q, E \cup B, Q, U, E$

$U = \langle E, Id(\overline{V})^{*} \rangle$

$$deliver(a,m) \qquad [bid(\overline{v}), \langle [\,], q_r \rangle, c]_a, a \lhd m \Rightarrow [bid(\overline{v}), \langle [m], q_r \rangle, c]_a \qquad \text{Eq. 68}$$

$$traverse(a) \qquad \begin{array}{l} [bid(\overline{v}), \langle m :: q_u, q_r \rangle, c]_a \Rightarrow \\ [\langle e_c, e_b, bid(\overline{v}) \rangle, \langle m, q_u, q_r \rangle, c]_a \quad if\ methMatch(Lib, bid, \overline{v}, m) = \langle e_c, e_b \rangle \end{array} \qquad \text{Eq. 69}$$

$$check(a) \qquad \frac{e \rightarrow_a e'}{[\langle e, e_b, bid(\overline{v}) \rangle, \langle m, q_u, q_r \rangle, c]_a \Rightarrow [\langle e', e_b, bid(\overline{v}) \rangle, \langle m, q_u, q_r \rangle, c]_a} \qquad \text{Eq. 70}$$

$$disable(a) \qquad [\langle v, e_b, bid(\overline{v}) \rangle, \langle m, q_u, q_r \rangle, c]_a \Rightarrow [bid(\overline{v}), \langle q_u, q_r :: m \rangle, c]_a \quad if\ v \neq \text{true} \qquad \text{Eq. 71}$$

$$enable(a) \qquad [\langle \text{true}, e_b, bid(\overline{v}) \rangle, \langle m, q_u, q_r \rangle, c]_a \Rightarrow [e_b, q_u . q_r, c]_a \qquad \text{Eq. 72}$$

$$seq(a) \qquad \frac{e \rightarrow_a e'}{[e, q_u, c]_a \Rightarrow [e, q_u, c]_a} \qquad \text{Eq. 73}$$

$$send(a, c, mid[\overline{v}]) \qquad [\mathcal{R}[\![v \lhd mid[\overline{v}]]\!], q_u, c]_a \Rightarrow [\mathcal{R}[\![0]\!], q_u, c]_a, v \lhd mid[\overline{v}] \quad if\ v \in A \qquad \text{Eq. 74}$$

$$ready(a) \qquad \left[\mathcal{R}[\![\text{ready}(bid(\overline{v}))]\!], q_u, c\right]_a \Rightarrow [bid(\overline{v}), \langle q_u, [\,] \rangle, c]_a \quad if\ behMatch(Lib, bid, \overline{v})\ and\ a \neq a_w \qquad \text{Eq. 75}$$

The added rules are:

$$ready(a_w) \qquad \begin{array}{l} \left[\mathcal{R}[\![\text{ready}(bid(\overline{v}))]\!], q_u\right]_{a_w}, [b, q, \langle c, u \rangle, c]_{a_{r_{i=1..n}}} \Rightarrow \\ [bid(\overline{v}), \langle q_u, [\,] \rangle]_{a_w}, [b, q, \langle c, u :: bid(\overline{v}) \rangle, inc(c)]_{a_{r_{i=1..n}}} \end{array} \quad if\ behMatch(Lib, bid, \overline{v}) \qquad \text{Eq. 76}$$

$$l \qquad \frac{l\colon [b, q, c]_a \Rightarrow [b', q', c]_a}{[b, q, u, c]_a \Rightarrow [b', q', u, c]_a} \qquad \text{Eq. 77}$$

$$update(a_r, c, bid(\overline{v})) \qquad \begin{array}{l} \left[bid'(\overline{v'}), \langle [\,], q_r \rangle, \langle c, bid(\overline{v}) :: u \rangle, c'\right]_{a_r} \Rightarrow \\ [bid(\overline{v}), \langle q_r, [\,] \rangle, \langle inc(c), u \rangle, c']_{a_r} \end{array} \qquad \text{Eq. 78}$$

It is trivial to see the equivalence of the original semantics and this semantics.

A slight modification is needed in the definition of enabled $update$ labels (adding the counter):
A label l is enabled in K if $l \in L \cup out(A, M)$ and
K has a transition with label l' where l' is the same as l up to choice of names for new actors or
K has a transition with label $l' = update(a_r, c_r, bid(\overline{v})_0)$ and $[b, q, bid(\overline{v})_{i=0..n}]_{a_r} \in K$ and $l = update(a_r, c_r + i, bid(\overline{v})_i)$, $i = 1..n$.
$Enabled(\pi, i)$ is the set of labels that are enabled in the source K_i of $\pi(i)$.

To show the equivalence of the sharing and translated pure programs, the same correspondence of transitions that was explained above can be built here. Now, the counter c_w is the value of the last element of the state of the writer actor $[b, q, c_w]_{a_w}$ and for each reader actor, the counter c_r is the value of the first element of the update component $[b, q, \langle c_r, u \rangle, c]_{a_r}$. We define the following to correspondence functions on the labels:

$$lp2s(l) = ls2p^{-1}(l) = \begin{cases} update(a_r, c_r, bid(\overline{v})) & if\ \left(l = deliver\left(a_r, Update(pair\ c_r\ \lambda x. ready\ bid(\overline{v}))\right)\right) \\ ready(a_w, c_w, bid(\overline{v})) & if\ \left(l = send\left(a_w, Update(pair\ c_w\ \lambda x. ready\ bid(\overline{v}))\right)\right) \\ l & else \end{cases} \qquad \text{Eq. 79}$$

$$ls2p(l) = lp2s^{-1}(l) = \begin{cases} l = deliver\left(a_r, Update(pair\ c_r\ \lambda x. ready\ bid(\overline{v}))\right) & if\ \left(update(a_r, c_r, bid(\overline{v}))\right) \\ l = send\left(a_w, Update(pair\ c_w\ \lambda x. ready\ bid(\overline{v}))\right) & if\ \left(ready(a_w, c_w, bid(\overline{v}))\right) \\ l & else \end{cases} \qquad \text{Eq. 80}$$

It is obvious that
$lp2s(l, l') \wedge ls2p(l', l'') \Rightarrow l = l''$ Eq. 81
$ls2p(l, l') \wedge lp2s(l', l'') \Rightarrow l = l''$ Eq. 82

For the forward direction, we add the following to the invariant between the two configurations ${}^{s}K$ and ${}^{p}K$:
$\forall l \in Enabled({}^{p}K) \Rightarrow lp2s(l) \in Enabled({}^{s}K)$ Eq. 83
$\forall l \in Fired({}^{s}K) \Rightarrow ls2p(l) \in Fired({}^{p}K)$ Eq. 84
For the reverse direction, we add the following to the invariant between the two configurations ${}^{p}K$ and ${}^{s}K$:
$\forall l \in Enabled({}^{s}K) \Rightarrow ls2p(l) \in Enabled({}^{p}K)$ Eq. 85
$\forall l \in Fired({}^{p}K) \Rightarrow lp2s(l) \in Fired({}^{s}K)$ Eq. 86

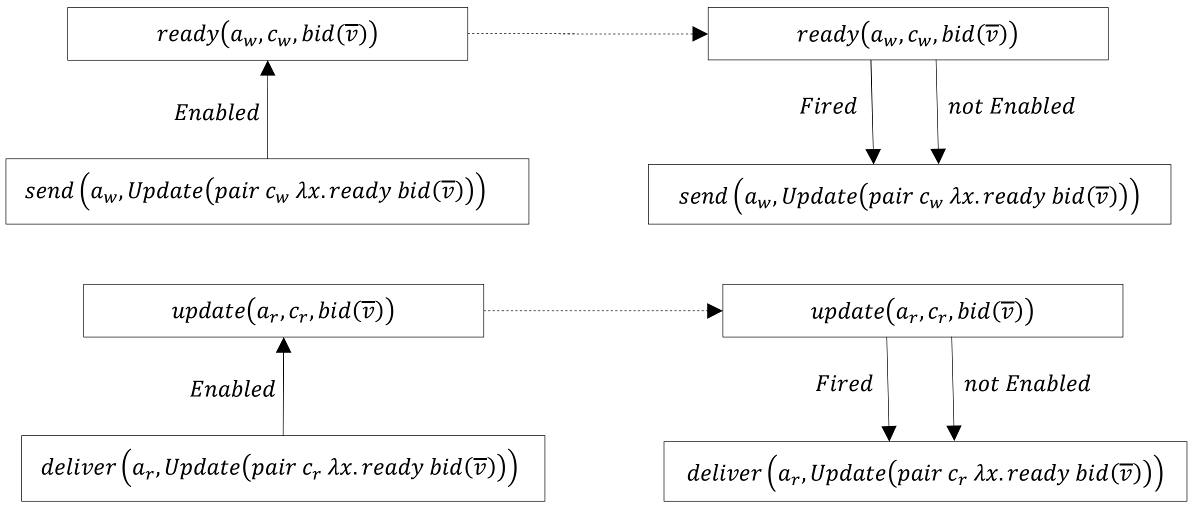
$ready(a_w, c_w, bid(\overline{v}))$
Enabled
$send\left(a_w, Update(pair\ c_w\ \lambda x.ready\ bid(\overline{v}))\right)$
$ready(a_w, c_w, bid(\overline{v}))$
Fired
not Enabled
$send\left(a_w, Update(pair\ c_w\ \lambda x.ready\ bid(\overline{v}))\right)$
$update(a_r, c_r, bid(\overline{v}))$
Enabled
$deliver\left(a_r, Update(pair\ c_r\ \lambda x.ready\ bid(\overline{v}))\right)$
$update(a_r, c_r, bid(\overline{v}))$
Fired
not Enabled
$deliver\left(a_r, Update(pair\ c_r\ \lambda x.ready\ bid(\overline{v}))\right)$

Figure 17. Correspondence of Labels for the Forward Direction

For the forward direction, the premise is that ${}^s\pi$ is fair, we show that the constructed ${}^p\pi$ is fair. To show that ${}^p\pi$ is fair, we assume an enabled label l in configuration ${}^pK$ of the path ${}^p\pi$ and show that l is either eventually fired or permanently disabled in ${}^p\pi$. If ${}^pK$ is a configuration in the middle of multi-steps of ${}^p\pi$ (consider multi-step transitions of Figure 15 and Figure 16.), then l is either just fired in the transition following ${}^pK$ or l is still enabled in the last configuration of the multi-step. Case: l is just fired at ${}^pK$. We trivially have the conclusion. Case: l is enabled in the last configuration of the multi-step. Consider Figure 17. As l is enabled in ${}^pK$, by the invariant (EQ. 83), a corresponding label $l' = lp2s(l)$ is enabled in a configuration ${}^sK$ of the path ${}^s\pi$. As ${}^s\pi$ is fair, l' is either eventually fired or is permanently disabled in ${}^s\pi$. Case: l' is eventually fired in configuration ${}^sK'$ of ${}^s\pi$. By the invariant (EQ. 84), the corresponding label $l'' = ls2p(l)$ is eventually fired in a configuration ${}^pK'$ of ${}^p\pi$. From EQ. 81 on $lp2s(l, l')$ and $ls2p(l', l'')$, we have $l = l''$. This means that l is eventually fired in ${}^p\pi$. Thus, we have the first disjunct of the conclusion. Case: l' is permanently disabled in ${}^s\pi$: There is a configuration ${}^sK'$ in ${}^s\pi$ where l' is not enabled in configurations following ${}^sK'$ in ${}^s\pi$. By the invariant (the contra-positive of EQ. 83), there is a configuration ${}^pK'$ in ${}^p\pi$ where $l'' = lp2s^{-1}(l')$ is not enabled in configurations following ${}^pK'$ in ${}^p\pi$. From $l' = lp2s(l)$ and $l'' = lp2s^{-1}(l')$, we have $l = l''$. Thus, there is a configuration ${}^pK'$ in ${}^p\pi$ where l is not enabled in configurations following ${}^pK'$ in ${}^p\pi$. This means that l is permanently disabled in ${}^p\pi$. Thus, we have the second disjunct of the conclusion.

For the reverse direction, we defined two transformations. The first transformation transforms the original path ${}^p\pi$ to the path ${}^p\pi'$ by removing all the $Update$ message delivery and failed guard evaluations from ${}^p\pi$ and delivering each $Update$ message just before its successful guard evaluation in ${}^p\pi$. It is assume that ${}^p\pi$ is fair and we show that the path ${}^p\pi'$ is fair. From the transformation, we have that ${}^p\pi'$ is fair for any label other than $deliver$ labels for $Update$ messages that their selective receive condition is never satisfied in ${}^p\pi$. Consider the recipient actor of these $Update$ messages. This actor should have been stuck; otherwise, the $Update$ message with the next

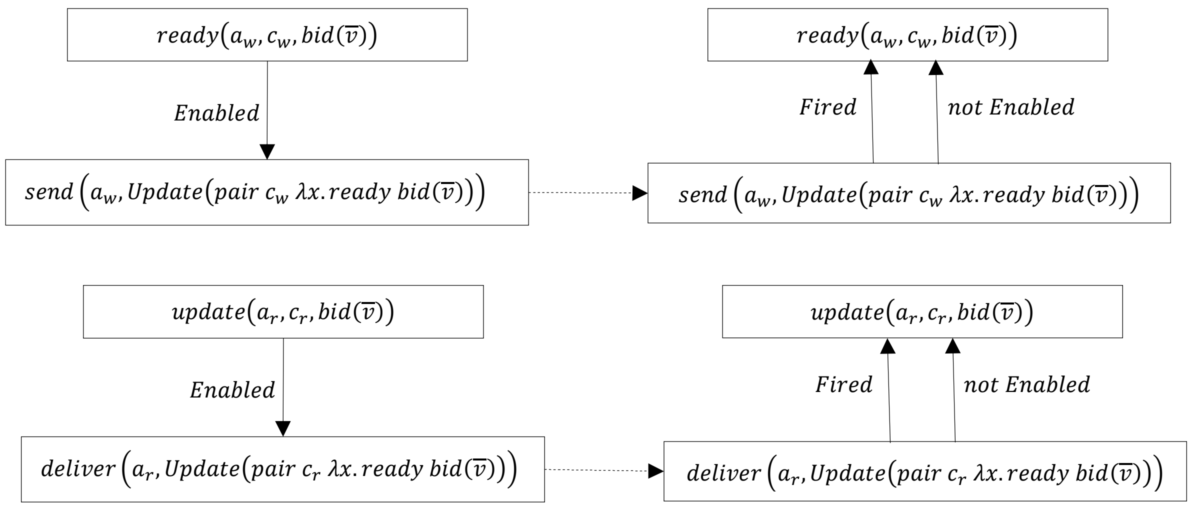
$ready(a_w, c_w, bid(\overline{v}))$
Enabled
$send\left(a_w, Update(pair\ c_w\ \lambda x.ready\ bid(\overline{v}))\right)$
$ready(a_w, c_w, bid(\overline{v}))$
Fired
not Enabled
$send\left(a_w, Update(pair\ c_w\ \lambda x.ready\ bid(\overline{v}))\right)$
$update(a_r, c_r, bid(\overline{v}))$
Enabled
$deliver\left(a_r, Update(pair\ c_r\ \lambda x.ready\ bid(\overline{v}))\right)$
$update(a_r, c_r, bid(\overline{v}))$
Fired
not Enabled
$deliver\left(a_r, Update(pair\ c_r\ \lambda x.ready\ bid(\overline{v}))\right)$

Figure 18. Correspondence of Labels for the Reverse Direction

expected counter value will eventually be processed and satisfy the selective receive condition. As the actor is stuck, the $deliver$ labels for these $Update$ messages are permanently disabled in ${}^{p}\pi$. Every label that is permanently disabled in ${}^{p}\pi$, is also permanently disabled in ${}^{p}\pi'$. Thus, the $deliver$ labels for these $Update$ messages are permanently disabled in ${}^{p}\pi'$. This means that ${}^{p}\pi'$ is fair to these $Update$ messages too.

The big-step transformation transforms the path ${}^{p}\pi$ to the big-step path ${}^{p}\pi^{\dagger}$ by taking some of the internal transitions to the left. It is assumed that ${}^{p}\pi$ is fair and we show that ${}^{p}\pi^{\dagger}$ is fair. As "the enabledness conditions on transitions are preserved by the transformation and that ${}^{p}\pi^{\dagger}$ has exactly the same transitions as ${}^{p}\pi$" [16], ${}^{p}\pi^{\dagger}$ is fair.

Now, we assume that ${}^{p}\pi$ is a fair big-step pure path and prove that the constructed sharing path ${}^{s}\pi$ is fair. To show that ${}^{s}\pi$ is fair, we assume an enabled label l in configuration ${}^{s}K$ of the path ${}^{s}\pi$ and show that l is either eventually fired or permanently disabled in ${}^{s}\pi$. Consider Figure 18. As l is enabled in ${}^{s}K$, by the invariant (EQ. 85), a corresponding label $l' = ls2p(l)$ is enabled in a configuration ${}^{p}K$ of the path ${}^{p}\pi$ (where ${}^{p}K$ is at the end of a multi-step of ${}^{p}\pi$). As ${}^{p}\pi$ is fair, l' is either eventually fired or is permanently disabled in ${}^{p}\pi$. Case: l' is eventually fired in configuration ${}^{p}K'$ of ${}^{p}\pi$. (As ${}^{p}\pi$ is in the big-step form, l' is enabled in ${}^{p}K$ and fired in ${}^{p}K'$ and ${}^{p}K$ is at the end of a multi-step, ${}^{p}K'$ is also at the end of a multi-step.) By the invariant (EQ. 86), the corresponding label $l'' = lp2s(l)$ is eventually fired in a configuration ${}^{s}K'$ of ${}^{s}\pi$. From EQ. 82 on $ls2p(l, l')$ and $lp2s(l', l'')$, we have $l = l''$. This means that l is eventually fired in ${}^{s}\pi$. Thus, we have the first disjunct of the conclusion. Case: l' is permanently disabled in ${}^{p}\pi$: There is a configuration ${}^{p}K'$ in ${}^{p}\pi$ where l' is not enabled in configurations following ${}^{p}K'$ in ${}^{p}\pi$. By the invariant (the contra-positive of EQ. 85), there is a configuration ${}^{s}K'$ in ${}^{s}\pi$ where $l'' = ls2p^{-1}(l')$ is not enabled in configurations following ${}^{s}K'$ in ${}^{s}\pi$. From $l' = ls2p(l)$ and $l'' = ls2p^{-1}(l')$, we have $l = l''$. Thus, there is a configuration ${}^{s}K'$ in ${}^{s}\pi$ where l is not enabled in configurations following ${}^{s}K'$ in ${}^{s}\pi$. This means that l is permanently disabled in ${}^{s}\pi$. Thus, we have the second disjunct of the conclusion.