Behavioural Equivalences for Dynamic Web Data

(Working Draft)

Sergio Maffeis Philippa Gardner
Department of Computing, Imperial College London
{maffeis,pg}@doc.ic.ac.uk

March, 2004.

Abstract

We study behavioural equivalences for dynamic web data in X$d\pi$, a model for reasoning about behaviour found in (for example) dynamic web page programming, applet interaction, and web-service orchestration. X$d\pi$ is based on an idealised model of semistructured data, and an extension of the π-calculus with locations and operations for interacting with data. The equivalences are non-standard due to the integration of data and processes, and the presence of locations.

Contents

List of Tables

1 Introduction

Web data, such as XML, plays a fundamental rôle in the exchange of information between globally distributed applications. Applications naturally fall into some sort of mediator approach: systems are divided into peers, with mechanisms based on XML for interaction between peers. The development of analysis techniques, languages and tools for web data is by no means straightforward. In particular, although web services allow for interaction between processes and data, direct interaction between processes is not well-supported.ciao!

Peer-to-peer data management systems are decentralised distributed systems where each component offers the same set of basic functionalities and acts both as a producer and as a consumer of information. We model systems where each peer consists of an XML data repository and a working space where processes are allowed to run. Our processes can be regarded as agents with a simple set of functionalities; they communicate with each other, query and update the local repository, and migrate to other peers to continue execution. Process definitions can be included in documents[1], and can be selected for execution by other processes. These functionalities are enough to express most of the dynamic behaviour found in web data, such as web services, distributed (and replicated) documents [1], distributed query patterns [21], hyperlinks, forms, and scripting.

The X$d\pi$-calculus [7] provides a formal description of such systems. It is based on a network of locations (peers) containing a (semi-structured) data model, and π-like processes [18, 22, 11] for modelling process interaction, process migration, and interaction with data. The data model consists of unordered labelled trees, with embedded processes for querying and updating data, and explicit pointers for referring to other parts of the network: for example, a document with a hyperlink referring to another site, and a light-weight trusted process for retrieving information associated with the link. A behavioural understanding of dynamic web data can serve as a starting point for the use of formal techniques. Moreover, the combination of web services and scripted processes provides the data engineer with many alternative patterns for exchanging information on the web [2, 19], and equational reasoning becomes useful to show, for example, that some complex data-exchange protocol conforms to its specification.

We study behavioural equivalences in Core X$d\pi$, which is a slight simplification of X$d\pi$, where both the data and the process component of a network are explicitly located, and therefore easier to analyse independently. We identify two notions of contextual equivalence for *open* networks, based on the observation of the data structure at each location, or of the capabilities of process to access data. We derive the corresponding process equivalences so that when two equivalent pieces of code are put in the same position in a network, the resulting networks cannot be distinguished by an observer. Process equivalences appear to be sensitive to the set of locations composing the network. This feature, together with having scripted processes as values, requires non trivial techniques for defining a labelled-bisimulation-based proof method.

[1] We regard process definitions in documents as an atomic piece of data, and we do not consider queries which modify such definitions.

1.1 Related Work

Our model is related to the Active XML approach to data integration developed independently by Abiteboul et al. [2]. Several distributed query languages, such as [19, 15, 4], extend traditional query languages with facilities for distribution awareness. Our approach is closest to the `ubQL` query language of [21], partly motivated by ideas from the π-calculus [20]. Process calculi have also been used for example to study security properties of web services [9], and to program XML-based Home Area Networks devices [3].

In [7] we have defined a first notion of barbed equivalence, and we have sketched a proof method based on higher-order bisimulation. In this paper we study in detail behavioural equivalences, improving and extending significantly the previous results. Core X$d\pi$ uses ideas from [5], and the contextual equivalences are based on the reduction-closed framework of [13]. Our labelled transition system and bisimulation exploit a translation technique from higher-order to first-order actions proposed in [14], and based on [23]. Ours is the first attempt to study behavioural equivalences of web-based (higher-order) data-sharing applications, and is characterised by its emphasis on dynamic data.

2 Core X$d\pi$

In X$d\pi$, a peer-to-peer network is represented as a set of locations each containing a data-tree and some processes. In order to reason modularly on data and processes, we model a network in Core X$d\pi$ as a pair (D, P), where D is a set of located trees, each one representing the data component of a location[2], and P is a multiset of located-process, representing both the services provided by each peer and the agents in execution on behalf of other peers.

2.1 Trees

Our data model extends the unordered labelled rooted trees of [6], with leaves which can either be scripted processes or pointers to data. We use the following constructs: *edge labels* denoted by $\mathtt{a}, \mathtt{b}, \mathtt{c} \in \mathcal{A}$, *path expressions* denoted by $p, q \in \mathcal{E}$ used to identify specific subtrees, and *location names* of the form $\circlearrowleft, l, m \in \mathcal{L}$, where the 'self' location $\circlearrowleft$ refers to the enclosing location. The set of data trees, denoted $\mathcal{T}$, is given by

$$T ::= 0 \mid T \,|\, T \mid \mathtt{a}[\,T\,] \mid \mathtt{a}[\,\Box P\,] \mid \mathtt{a}[\,@l{:}p\,]$$

Tree 0 denotes a rooted tree with no content. Tree $T_1 \,|\, T_2$ denotes the composition of T_1 and T_2, which simply joins the roots. A tree of the form $\mathtt{a}[\,...\,]$ denotes a tree with a single branch labelled $\mathtt{a}$ which can have three types of content: a subtree T; a *scripted process* $\Box P$, which is a static process awaiting a command to run; a *pointer* $@l{:}p$, which denotes a pointer to a set of subtrees identified by path expression p in the tree at location l. The structural congruence for trees states that trees are unordered, and scripted processes are identified up to the structural congruence for processes (see Table 1).

[2] From now on we consider *location* and *peer* as synonyms.

Structural congruence is the least congruence satisfying alpha-conversion, the commutative monoidal laws for $(0, |)$ on trees and processes, and the axioms reported below:

$$
\begin{array}{rll}
(\textsc{Trees}) & U \equiv U' \Longrightarrow \mathsf{a}[\,U\,] \equiv \mathsf{a}[\,U'\,] & \\
(\textsc{Values}) & v' \equiv w' \wedge \tilde{v} \equiv \tilde{w} \Longrightarrow v', \tilde{v} \equiv w', \tilde{w} & P \equiv Q \Longrightarrow \Box P \equiv \Box Q \\
(\textsc{Processes}) & (\nu c)(\nu c')P \equiv (\nu c')(\nu c)P & (\nu c)0 \equiv 0 \\
& c \notin fn(P) \Longrightarrow P \,|\, (\nu c)Q \equiv (\nu c)(P \,|\, Q) & \\
& V \equiv V' \wedge P \equiv Q \Longrightarrow l{\cdot}\mathsf{update}_p(\chi, V).P \equiv l{\cdot}\mathsf{update}_p(\chi, V').Q & \\
(\textsc{Stores}) & \forall\, l.\, D(l) \equiv B(l) \Longrightarrow D \equiv B & \\
(\textsc{Networks}) & D \equiv B \wedge P \equiv Q \Longrightarrow (D, P) \equiv (B, Q) &
\end{array}
$$

Table 1: Structural congruence.

We regard a path p as a function from trees to sets of nodes (up to structural congruence): $p(T)$ denotes the tree T where the nodes identified by p are selected. For simplicity we do not show node identifiers explicitly, but we underline the selected nodes. We describe paths using a subset of XPath [16], where "a" denotes a step along an edge labelled a, "/" denotes path composition, ".." a step back, "//" any node, and ".", which can appear only inside trees, denotes the path from the root of the tree to the current node. For example, in $\underline{\mathsf{a}[\,\underline{\mathsf{a}[\,S\,]}\,|\,\mathsf{b}[\,S'\,]\,|\,\mathsf{c}[\,T'\,]\,]}$ we have underlined the nodes selected by path $//\mathsf{a}$.

2.2 Located Processes

Our processes are based on asynchronous π_2-processes [5] extended with an operation for manipulating the tree structure (update) and one for selecting a script for execution (run). Generic variables are x, y, z, channel names or channel variables are a, b, c, (where necessary, the meaning will be clear from the context) and values are

$$u, v, w ::= T \mid c \mid l \mid p \mid \Box P$$

We use the notation $\tilde{z}$ for vectors of variables, and $\tilde{v}$ for vectors of values and variables. Identifiers U, V range over scripted processes, pointers and trees. The set of processes, denoted by $\mathcal{P}$, is given by

$$
\begin{array}{rl}
P, Q, R ::= & 0 \mid P \,|\, P \mid \overline{l{\cdot}b}\langle\tilde{v}\rangle \mid l{\cdot}b(\tilde{z}).P \mid !l{\cdot}b(\tilde{z}).P \mid (\nu c)P \\
& \mid l{\cdot}\mathsf{update}_p(\chi, V).P \mid l{\cdot}\mathsf{run}_p
\end{array}
$$

The processes in the first line of the grammar are constructs arising from the π_2-calculus: the *output* process $\overline{l{\cdot}b}\langle\tilde{v}\rangle$ which denotes a vector of values $\tilde{v}$ waiting to be sent via channel b at location l, the *input* process $l{\cdot}b(\tilde{z}).P$ which is waiting to receive values from an output process via channel b at l, and the standard *nil*, *composition*, *restriction* and *replicated input*. We assume a simple sorting discipline, to ensure that the number (and sort) of values sent along a channel matches the number (and sort) of variables expected to receive those values. Channel names $\mathcal{C}$ are partitioned into *public* and *session* channels, denoted $\mathcal{C}_p$ and $\mathcal{C}_s$ respectively. Public channels denote those channels that are intended to have the same meaning at each location, such as "finger", and cannot be restricted. Session channels are used for process interaction, and can be restricted We assume on session channels the usual notions of *free* and *bound* names (fn, bn). Scripted processes, being just code, cannot have free session names.

Command $l{\cdot}\mathsf{run}_p$ activates the scripted processes denoted by p from the tree of l. Command $l{\cdot}\mathsf{update}_p(\chi, U).P$ is used to interact with the data tree at l. *Patterns* χ, ξ have the form

$$\chi ::= X \mid @x{:}y \mid \Box X,$$

where X denotes a tree or process variable. In an update, U can contain variables and must have the same sort as χ. The variables free in χ are bound in U and P. The update command finds all the values U_i given by the path p, pattern-matches these values with χ to obtain the substitution σ_i when it exists. For each successful pattern-matching, it replaces the U_i with $U\sigma_i$ and evolves to $P\sigma_i$. Below we give some commands derived from update:

$$\begin{array}{rcll}
l{\cdot}\mathsf{copy}_p(X).P & \triangleq & l{\cdot}\mathsf{update}_p(X, X).P & \text{copy the tree at } p \text{ and use it in } P \\
l{\cdot}\mathsf{cut}_p(X).P & \triangleq & l{\cdot}\mathsf{update}_p(X, 0).P & \text{cut the tree at } p \text{ and use it in } P \\
l{\cdot}\mathsf{paste}_p\langle T\rangle.P & \triangleq & l{\cdot}\mathsf{update}_p(X, X \mid T).P & \left\{ \begin{array}{l} \text{where } X \text{ is not free in } T \text{ or } P, \\ \text{paste tree } T \text{ at } p \text{ and evolve to } P \end{array} \right.
\end{array}$$

2.3 Networks and Stores

A network is represented by a pair (D, P) where the first component (the *store*) is a finite partial function from location names to trees, and the second component is a process. Interaction between processes and data is always local, as will be shown by rules (Update) and (Run) in Table 2, and consequently the store can be considered as a *distributed* function. We write $dom(D)$ to denote the domain of store D. We write $D_1 \uplus D_2$ for the union of stores D_1 and D_2 with disjoint domains. The network (D, P) is well-formed if D and P contain no free variables, and all the scripted processes have no free session names.

Our reduction semantics on networks will be closed with respect to *network contexts* $(C_{\mathcal{S}}, C_{\mathcal{P}})$, where *store contexts* $C_{\mathcal{S}}$ are defined by $C_{\mathcal{S}} ::= - \mid C_{\mathcal{S}} \uplus D$ and *process contexts* $C_{\mathcal{P}}$ are defined by $C_{\mathcal{P}} ::= - \mid \ C_{\mathcal{P}} \mid P \ \mid (\nu c)\, C_{\mathcal{P}}$. We will omit the subscripts from contexts when no ambiguity can arise. Given a network (D, P) and a context $C = (C_1, C_2)$, we write $C\{(D, P)\}$ for their composition: for example, if $C_1 = - \uplus B$, $C_2 = (\nu c)-$ then $C\{(D, P)\} \ = \ (D \uplus B, (\nu c)P)$. Composition of stores and networks is defined only for stores with disjoint domains.

2.4 Reduction Semantics

The reduction relation $\rightarrow$, relying on an updating function $\rightsquigarrow$, describes processes interaction, the interaction between processes and data, and (implicitly) the movement of processes across locations (Table 2).

First we describe the reduction relation. Rules (Com) and (!Com) are basically the standard communication rules for the π-calculus, except that processes only communicate if they are at the same location l, and l is in the store. Rule (Update) provides

(Reduction Axioms) The reduction relation on processes is the smallest relation closed with respect to reduction contexts, structural congruence and the axioms:

(Com) $(\{l \mapsto T\}, \overline{l{\cdot}c}\langle\tilde{v}\rangle \mid l{\cdot}c(\tilde{x}).P) \rightarrow (\{l \mapsto T\}, P\{\tilde{v}/\tilde{x}\})$

(!Com) $(\{l \mapsto T\}, \overline{l{\cdot}c}\langle\tilde{v}\rangle \mid !l{\cdot}c(\tilde{x}).P) \rightarrow (\{l \mapsto T\}, !l{\cdot}c(\tilde{x}).P|P\{\tilde{v}/\tilde{x}\})$

(Update) $$\frac{p(T) \rightsquigarrow_{p,l,\chi,V} T', \{\sigma_1, \cdots, \sigma_n\}}{(\{l \mapsto T\}, l{\cdot}\mathsf{update}_p(\chi, V).P) \rightarrow (\{l \mapsto T'\}, P\sigma_1 \mid \cdots \mid P\sigma_n)}$$

(Run) $$\frac{p(T) \rightsquigarrow_{p,l,\Box X,\Box X} T, \{\{\Box P_1/\Box X\}, \cdots, \{\Box P_n/\Box X\}\}}{(\{l \mapsto T\}, l{\cdot}\mathsf{run}_p) \rightarrow (\{l \mapsto T\}, P_1 \mid \cdots \mid P_n)}$$

(Updating Function) Below Σ is a multiset of substitutions, and $\oplus$ is multiset union, and $\Theta = p, l, \chi, V$ are the parameters of an update or run command.

(Zero) $0 \rightsquigarrow_\Theta 0, \emptyset$ (Link) $\mathsf{a}[\,@m{:}q\,] \rightsquigarrow_\Theta \mathsf{a}[\,@m{:}q\,], \emptyset$

(Script) $\mathsf{a}[\,\Box Q\,] \rightsquigarrow_\Theta \mathsf{a}[\,\Box Q\,], \emptyset$ (Node) $\dfrac{T \rightsquigarrow_\Theta T', \Sigma}{\mathsf{a}[\,T\,] \rightsquigarrow_\Theta \mathsf{a}[\,T'\,], \Sigma}$

(Par) $$\frac{T \rightsquigarrow_\Theta T', \Sigma_1 \qquad S \rightsquigarrow_\Theta S', \Sigma_2}{T \mid S \rightsquigarrow_\Theta T' \mid S', \Sigma_1 \oplus \Sigma_2}$$

(Up) $$\frac{\mathsf{match}(U, \chi) = \sigma \qquad V\sigma \rightsquigarrow_\Theta V', \Sigma \qquad \Theta = p, l, \chi, V}{\mathsf{a}\underline{[\,U\,]} \rightsquigarrow_\Theta \mathsf{a}[\,V'\,], \{\sigma\{l/\circlearrowleft, p/.\}\} \oplus \Sigma}$$

Table 2: Reduction axioms and updating function.

interaction between processes and data. Given the command $l{\cdot}\mathsf{update}_p(\chi, V).P$ and the tree T at l in the store, the updating function $\rightsquigarrow$ takes $p(T)$ as an argument, matches each identifier U_i in $p(T)$ with the pattern χ to obtain the substitution σ_i, replaces each U_i with $V\sigma_i$ in T, and returns the continuation process $P_i\sigma_i$. Rule (Run) is a special case of update, where the tree is not modified, and the scripted processes $\Box P_i$ identified by $p(T)$ are activated in parallel to yield the continuation P_i.

We now describe the updating function $\rightsquigarrow$, which is parameterised by p, l, χ, V, the arguments of an update or run command. The first five rules define simply a traversal of the tree collecting the set of substitutions Σ, whereas Rule (Up) is responsible for the actual update. It applies to the identified nodes (underlined), matching U with χ, to obtain substitution σ. When σ exists, the process continues recursively updating $V\sigma$, until some subtree V' with a set of substitutions denoted by Σ is returned. At this point U is replaced with V', and Σ is returned, together with $\sigma\{l/\circlearrowleft, p/.\}$ (where any reference to the current location $\circlearrowleft$ and path "." is substituted by the actual values l and p).

As an example for the cut operation, consider $T = \mathsf{c}[\,\mathsf{a}[\,T_1\,] \mid \mathsf{a}[\,T_2\,] \mid \mathsf{b}[\,S\,]\,]$ and $T' = \mathsf{c}[\,\mathsf{a}[\,0\,] \mid \mathsf{a}[\,0\,] \mid \mathsf{b}[\,S\,]\,]$. We have that

$$(\{l \mapsto T\}, l{\cdot}\mathsf{cut}_{\mathsf{c/a}}(X).P) \rightarrow (\{l \mapsto T'\}, P\{T_1/X\} \mid P\{T_2/X\})$$

where the subtrees T_1 and T_2, identified by $\mathsf{c/a}$, are removed from the store, and each is passed to a copy of P. We give now an example of run and of the substitution of local references. If $S = \mathsf{a}[\,\mathsf{b}[\,\Box\overline{\circlearrowleft\cdot a}\langle v\rangle\,]\,|\,\mathsf{b}[\,\Box m\cdot\mathsf{run}_{./../\mathtt{c}}\,]\,]$, then

$$(\{l \mapsto S\}, \mathsf{run}_{\mathtt{a/b}}) \rightarrow (\{l \mapsto S\}, \overline{l\cdot a}\langle v\rangle \,|\, m\cdot\mathsf{run}_{\mathtt{a/b/../c}})$$

The store S is unaffected by the `run` operation, which spawns the two processes identified by `a/b`, where the local path ./../`c` is replaced by `a/b/../c`, and $\circlearrowleft$ is replaced by l. Note that $m\cdot\mathsf{run}_{\mathtt{a/b/../c}}$ is located at m, which is not in the domain of the store. There is no reduction rule for such a process, which represents mobile code "lost" due to network partitioning, or to an invalid network address[3].

We conclude the section with an example. Consider, at location m, a web service `get` for downloading data which, given a path expression p, returns a stream of messages containing the subtrees denoted by p at m. The service is described by process $m\cdot\mathtt{get} = !m\cdot get(x,y,z).m\cdot\mathsf{copy}_x(Y).\overline{y\cdot z}\langle Y\rangle$, where channel get inputs a path x, a location y, and a channel z, and returns its results at y on z. The corresponding service invocation from l is $l\cdot\mathtt{call}(m, get, p) = (\nu c)(\overline{m\cdot get}\langle p, l, c\rangle \,|\, !l\cdot c(Y).R)$, where R is some code handling each result. We will see in Section 4 that invoking $m\cdot\mathtt{get}$ with $l\cdot\mathtt{call}(m, get, p)$ is equivalent to running (from l) the specification $l\cdot\mathsf{spec} = m\cdot\mathsf{copy}_x(Y).R$.

See [7] for other motivating examples (web services, XLink, e-forms).

3 Contextual Equivalences

In this section we study equivalences for networks and processes. In particular, we define when two processes are equivalent in such a way that when the processes are put in the same position in the network, the resulting networks are equivalent. In Section 4, we introduce a proof method for showing process equivalence.

3.1 Network Equivalences

We base our network equivalences on the reduction-closed framework of Honda and Yoshida [13]. The equivalences depend on the choice of observables, and we study two cases: data trees, and potential interactions between processes and trees. We begin with standard generic definitions, based on some observation relation $N \downarrow_\beta$ which states that network N exhibits the observable β. We then study specific observation relations.

Definition 3.1 *The* weak observation relation *induced by $\downarrow_\beta$, denoted by $\Downarrow_\beta$, is defined by $N \Downarrow_\beta \;\triangleq\; \exists N'.\, N \rightarrow N' \wedge N' \downarrow_\beta$. The* reduction congruence induced by $\downarrow_\beta$, *denoted by $\simeq$, is the largest symmetric relation $\dot{\simeq}$ on networks such that $N \dot{\simeq} M$ implies*

- *N and M have the same observables: $N \downarrow_\beta \Rightarrow M \Downarrow_\beta$;*

[3] In our model it is not possible for a process to create a new location. In fact, processes represent either scripts or web services, none of which could realistically create new peers. Hence the domain of a network is invariant under reduction. Nonetheless we consider open systems, since we admit network composition.

Our approach differs from the one of e.g. [10], where process migration can have the effect of creating a new location. We will see in Section 3 how our choice requires new techniques for studying behavioural equivalences.

- $\dot{\simeq}$ *is reduction-closed:* $N \to N' \Rightarrow (\exists M'.M \to^* M' \ \wedge \ N' \dot{\simeq} M')$;
- $\dot{\simeq}$ *is closed under network contexts:* $\forall C.C[N] \dot{\simeq} C[M]$.

In the setting of dynamic web data, a natural criterion to decide when two systems are equivalent is to compare the structure of the data tree at each location. In fact processes can be seen as working in the background, and hence should not be directly observable. Comparing trees up to structural congruence would be overly restrictive because of scripted processes, which can be semantically equivalent without being structurally congruent. Instead we consider a weaker notion of equivalence on trees which does not look at scripts (and pointers). These can be analysed indirectly by suitable contexts.

Definition 3.2 *We define* observation congruence*, denoted by $\equiv^t$, as the structural congruence of Table 1 with the additional axioms $\Box P \equiv^t \Box Q$ and $@l{:}p \equiv^t @m{:}q$.*

For example, if $T = \mathtt{a}[\,\mathtt{b}[\,\Box P\,] \,|\, \mathtt{b}[T'\,] \,|\, \mathtt{c}[\,@l{:}p\,]\,]$ and $S = \mathtt{a}[\,\mathtt{c}[\,@m{:}q\,] \,|\, \mathtt{b}[S'\,] \,|\, \mathtt{b}[\,\Box Q\,]\,]$ with $T' \equiv^t S'$, then we have $T \equiv^t S$.

Definition 3.3 *A* tree observable *has the form $l{\cdot}T$, where l is a location name and T is a tree. We define the observation relation $N \downarrow_{l\cdot T}$ on networks and tree observables by $N \downarrow_{l\cdot T} \triangleq \exists C,S.\ N = C\{(\{l \mapsto S\},0)\} \wedge S \equiv^t T$, that is, N contains a location l with an S tree-congruent to T.* Tree congruence *($\simeq^t$) is the reduction congruence induced by tree observables.*

For example, let us consider $\mathtt{swap} = !l{\cdot}c(X).l{\cdot}\mathsf{update}_{/}(Y,X).l{\cdot}\mathsf{update}_{/}(Z,Y).\overline{l{\cdot}c}\langle Z\rangle$, and $\mathtt{alt}(T,S) = (\{l \mapsto S\},(\nu c)(\overline{l{\cdot}c}\langle T\rangle \,|\, \mathtt{swap}))$. We have that $\mathtt{alt}(T,S) \simeq^t \mathtt{alt}(S,T)$ for any T and S, since each process can mimic the other and swap the trees. As a negative example, consider $\mathtt{net}(P) = (\{l \mapsto \mathtt{a}[\,\Box P\,]\},0)$, $P_1 = \circlearrowleft \cdot\mathsf{cut}_a(X)$, and $P_2 = \circlearrowleft \cdot\mathsf{paste}_{/}\langle \mathtt{b}[\,0\,]\rangle$. We have that $\mathtt{net}(P_1) \not\simeq^t \mathtt{net}(P_2)$, since $\mathtt{test} = (-,-\,|\, l{\cdot}\mathsf{run}_a)$ is a context separating $\mathtt{net}(P_1)$ from $\mathtt{net}(P_2)$: $\mathtt{test}\{\mathtt{net}(P_1)\} \Downarrow_{l\cdot 0}$ but $\mathtt{test}\{\mathtt{net}(P_2)\} \not\Downarrow_{l\cdot 0}$.

We now consider two alternative definitions of observables and compare the induced reduction congruences with tree congruence.

Definition 3.4 *(i) We define the* minimal *tree observation relation by $N \downarrow_l \triangleq \exists C.\ N \equiv^t C\{(\{l \mapsto 0\},0)\}$, that is, N contains a location l where the tree is empty.* 0-congruence *($\simeq^0$) is the reduction congruence induced by minimal tree observables. (ii) We define the* output *observation relation by $N \downarrow_{l\cdot a} \triangleq \exists\, D,T,\tilde{v},\tilde{b},P.\ N \equiv (D \uplus \{l \mapsto T\},(\nu\tilde{b})(\overline{l{\cdot}a}\langle\tilde{v}\rangle \,|\, P))$ where $a \notin \tilde{b}$, that is, N contains a location l with an output process ready to communicate on a.* Output congruence *($\simeq^{\langle\rangle}$) is the reduction congruence induced by output observables.*

Theorem 3.1 *The relations $\simeq^t$, $\simeq^0$ and $\simeq^{\langle\rangle}$ coincide.*

Another natural choice for observables, motivated by security concerns, is to consider the capabilities of a processes to access data. We define *barbs* revealing where in a located tree a process can potentially read or write some data.

Definition 3.5 *A* barb *has the form $l{\cdot}p$, where l is a location name and p is a path expression. We define the observation relation $N \downarrow_{l\cdot p}$ on networks and barbs by $N \downarrow_{l\cdot p} \triangleq \exists\, C,T,\chi,U,P.\ N \equiv C\{(\{l \mapsto T\}, l{\cdot}\mathsf{update}_p(\chi,U).P)\}$, that is, N contains a location l with an update_p command.* Barbed congruence *($\simeq^b$) is the reduction congruence induced by barbs.*

For example, if $\mathtt{xch}(T_1,T_2) = (\nu c)(\overline{l\cdot c}\langle\rangle \mid !l\cdot c().l\cdot\mathsf{update}_p(X,T_1).l\cdot\mathsf{update}_p(X,T_2).\overline{l\cdot c}\langle\rangle)$ then we have $(\{l \mapsto S\}, \mathtt{xch}(T_1,T_2)) \simeq^b (\{l \mapsto S\}, \mathtt{xch}(T_2,T_1))$, for all T_1, T_2 and S. In fact, the processes have the same barbs, and if S contains a subtree at p, they can simulate each other.

Notice that a barb $l\cdot p$ merely records the location and the path at which some update command could take place, giving no information on *how* the data could be modified, and ignoring run commands. Again, this information can be observed indirectly using some context.

Theorem 3.2 *Barbed congruence strictly implies tree congruence:* $\simeq^b \subsetneq \simeq^t$.

The inclusion is strict: for all D, $(D,0) \simeq^t (D, l\cdot\mathsf{copy}_p(x))$, since the stores are equal and $l\cdot\mathsf{copy}_p(x)$ has no effect, but $(D,0) \not\simeq^b (D, l\cdot\mathsf{copy}_p(x))$ because $l\cdot\mathsf{copy}_p(x) \downarrow_{l\cdot p}$. Structural congruence for networks is included in $\simeq^b$, and therefore in $\simeq^t$.

3.2 Process Equivalences

We now analyse process behaviour, which is influenced by the locations present in the network (*network connectivity*). Consider replacing the definition of a service at location l, which uses only local data, with an equivalent one depending on data from another location m. If we can assume that m is always connected, then the behaviour of the services is the same. On the other hand, if location m should fail, the behaviour of the new one is affected. With network equivalences, the "reliable" locations are those in the domain of the store. With process equivalences, it is necessary to state explicitly the minimum set of reliable locations. For example, consider $\mathtt{oldS} = l\cdot\mathsf{cut}_/(X)$ and $\mathtt{newS} = (\nu c)(\overline{m\cdot c}\langle/\rangle \mid m\cdot c(x).l\cdot\mathsf{cut}_x(X))$. The two processes are equivalent if m is reliable, otherwise they are not: in the context $(\{l \mapsto T\}, -)$ the first process can delete T, but the second one cannot move.

Process equivalence requires considering two processes equivalent if they are equivalent in all possible network contexts, starting from a given domain.

Definition 3.6 *Given a network equivalence $\simeq$ and a set of location names Λ,* domain process equivalence *is* $\sim_\Lambda = \{(P,Q) | \forall D\,.\,\Lambda \subseteq dom(D) \Longrightarrow (D,P) \simeq (D,Q)\}$. Domain tree equivalence, *($\sim^t_\Lambda$), is the domain process equivalence induced by $\simeq^t$, and* domain barbed equivalence *($\sim^b_\Lambda$), is the one induced by $\simeq^b$.*

For example, for any Λ, $\mathtt{xch}(T_1,T_2) \sim^b_\Lambda \mathtt{xch}(T_2,T_1)$. As in the case for network equivalences (Theorem 3.2), we have $\sim^b_\Lambda \subsetneq \sim^t_\Lambda$, with the same counterexample.

In order to be able to replace a process sub-term by an equivalent one, we extend process equivalences to *open* terms (terms with free variables).

Definition 3.7 Full process contexts *are defined by*

$$C ::= - \mid C \mid P \mid (\nu c)\ C \mid l\cdot a(\tilde{x}).C \mid !l\cdot a(\tilde{x}).C \mid l\cdot\mathsf{update}_p(\chi, V).C$$

Definition 3.8 *A substitution σ is a* closing substitution *for P iff $P\sigma$ is closed. Given an equivalence $\sim$ for closed processes, and two open processes P and Q, we say that $P \sim Q$ iff $P\sigma \sim Q\sigma$ for all closing substitutions σ.*

Theorem 3.3 *For all Λ, (i) if $\Lambda \subset \Lambda'$ then $\sim^t_\Lambda \subset \sim^t_{\Lambda'}$ and $\sim^b_\Lambda \subset \sim^b_{\Lambda'}$; (ii) $\sim^t_\Lambda$ and $\sim^b_\Lambda$ (both on open and closed processes) are congruences over full contexts.*

As an example for the strict inclusion of (i), considering the processes `oldS` and `newS` given above, we have `oldS` $\sim^b_{l,m}$ `newS` but `oldS` $\not\sim^b_l$ `newS`.

Remark 3.1 *Core X$d\pi$ is an extension of the asynchronous π-calculus, and accordingly the* asynchrony law*, stating that the presence of a communication buffer cannot be observed, holds also in our setting:* $!l{\cdot}a(x).\overline{l{\cdot}a}\langle x\rangle \sim^b_\Lambda 0$*. On the other hand, the law for* equators *does not hold: let* $l{\cdot}\mathtt{E}(a,b)\ =!l{\cdot}a(x).\overline{l{\cdot}b}\langle x\rangle \,|\, !l{\cdot}b(x).\overline{l{\cdot}a}\langle x\rangle$*, then* $l{\cdot}\mathtt{E}(a,b)\,|\,\overline{l{\cdot}c}\langle a\rangle \not\sim^b_\Lambda l{\cdot}\mathtt{E}(a,b)\,|\,\overline{l{\cdot}c}\langle b\rangle$*, since a context can read b from c at l, and use it at some fresh location m where no equator is defined.*

We can see a network equivalence as a relation on the cartesian product of stores and processes. Consequently, a process equivalence, in order to agree with network equivalence, must be contained in the projection on the process component of network equivalence. We show that process tree and barbed equivalences are in fact the largest congruences contained in such projections. For simplicity, we restrict our attention in this paragraph to the relations $\sim^t_\emptyset$ and $\sim^b_\emptyset$, since the results can be easily generalised to an arbitrary Λ. We omit the subscript $\emptyset$ from the relations for readability.

Definition 3.9 *(i) Given an arbitrary network equivalence $\simeq$, we define its process-projection $\Pi_2(\simeq)$ as $\{(P,Q)|\exists D,B.dom(D)=dom(B)\wedge(D,P)\simeq(B,Q)\}$, and if $(P,Q)\in\Pi_2(\simeq)$ we write $P\sim_{\exists 2} Q$. (ii) We define the corresponding congruence relation $\Pi^c_2(\simeq)$ by $\{(P,Q)\mid\forall C.C\{P\}\sim_{\exists 2} C\{Q\}\}$, and if $(P,Q)\in\Pi^c_2(\simeq)$ we write $P\sim^c_{\exists 2} Q$.*

By definition $\Pi^c_2(\simeq)$ is the largest congruence contained in $\Pi_2(\simeq)$, and therefore the largest congruence on processes which is compatible with the network equivalence $\simeq$. Note that this definition is very weak, since it requires, for two processes to be equivalent, that there exists two stores such that the corresponding networks are equivalent.

Theorem 3.4 *(i) $\Pi^c_2(\simeq^t)$ coincides with $\sim^t$. (ii) $\Pi^c_2(\simeq^b)$ coincides with $\sim^b$.*

4 A Proof Method for Process Equivalence

Process equivalence, as defined in Section 3, is hard to use in practice, because it requires closure under all store and process contexts. In this section we provide a coinductive equivalence which does not quantify over contexts.

The main difficulties involved in defining a bisimulation for Core X$d\pi$ are caused by having scripted processes among values, and by barbed equivalence being sensitive to the presence of locations. We solve the first problem by translating messages containing scripts into ones where each script is replaced by a first-order value (a *trigger*), and placing in parallel to the process being analysed some *definitions* associating to each trigger the code of the scripted process. In this way it is possible to take into consideration the interaction between scripts and their contexts. For a discussion of this technique see [14], where it is used on the higher-order π-calculus. We solve the second problem using a generalisation of the concept of bisimulation to families of relations indexed by sets of locations, which we call *domain-dependent bisimilarity*. Communication is asynchronous, hence we borrow techniques from the asynchronous π-calculus.

4.1 Labelled Transition System

Let $\mathcal{K}$, ranged over by i, j, k, be the set of *trigger* names, disjoint from $\mathcal{C}$. We introduce a construct $\langle k \Leftarrow \Box P\rangle$, called a *definition*, which associates a scripted process to the trigger name k. There is no reduction rule for definitions, which are analysed only in the lts. Parallel compositions of processes and definitions are called *configurations* K, L, and together with contexts C are given by

$$u ::= \ldots \mid k \quad K, L ::= K \mid K \mid (\nu c)K \mid P \mid \langle k \Leftarrow \Box P\rangle \quad C ::= - \mid K \mid C \mid (\nu c)C$$

The new set of values is given by the values considered so far together with all the values where one ore more scripted processes have been replaced by triggers. We let underlined letters $\underline{u}, \underline{v}$ range over *first-order* values (values not containing scripted processes), and we will omit the underlining when there is no ambiguity.

Structural congruence is extended to configurations in the obvious way. A configuration K is well-formed if its processes are well-formed, there is at most one $\langle k \Leftarrow \Box P\rangle$ for each k, and processes in definitions do not contain triggers. When an output or an update action takes place, we use a relation F to replace scripted processes in values with fresh triggers, and to place in parallel the definitions recording the scripted code associated to each trigger. In general F translates a potentially higher-order term $\tilde{v}$ into a first-order term $\underline{\tilde{u}}$, and it collects a parallel composition Θ of definitions corresponding to the scripts extracted from the values, for some vector of fresh, distinct triggers $\tilde{k}$ (as in $F(\tilde{v}) = (\tilde{u}, \Theta, \tilde{k})$). Relation F is defined as a homomorphism (preserving the freshness of triggers) on all terms, and is defined as $F(\Box P) = (k; \langle k \Leftarrow \Box P\rangle; k)$ for scripts (see Table 3 in the Appendix).

Transition labels α_l are indexed with the location at which actions take place, and are defined below, where $d \in \mathcal{C}_s \cup \mathcal{K}$:

$$\alpha_l ::= (\tilde{d})\overline{l{\cdot}c}\langle\underline{\tilde{v}}\rangle \mid l{\cdot}c(\underline{\tilde{v}}) \mid l{\cdot}\tau \mid (\tilde{k})l{\cdot}\mathsf{update}_p(\underline{\tilde{U}}, (\chi)\underline{\tilde{V}}) \mid l{\cdot}\mathsf{run}_p \mid l{\cdot}k(p)$$

Label $l{\cdot}\tau$ denotes communication at l. The label for update contains a vector of results $\tilde{U}$ (corresponding to the range of Σ in the updating function) and treats $(\chi)\tilde{V}$ as an abstraction on the pattern variables (which therefore are subject to alpha-conversion). The vector $(\tilde{k})$ is used by the side conditions of the lts to enforce freshness of triggers, and binds the triggers $\tilde{k}$. Finally, label $l{\cdot}k(p)$ signals that the script defined by k is selected for execution, with implicit parameters l and p. We explain now the rules for the lts, leaving the formal definition to Table 4 in the Appendix. Labelled transitions are defined for well-formed configurations. We have standard contextual rules, with the side conditions adapted to avoid clashes of trigger names. The rule for communication is

$$(\textsc{Com}) \quad \overline{l{\cdot}c}\langle\tilde{v}\rangle \mid l{\cdot}c(\tilde{x}).P \xrightarrow{l\cdot\tau} P\{\tilde{v}/\tilde{x}\}$$

and correspond closely to the reduction rule. The rule with replicated input is analogous. The rule for input and output, in the style of [12], are

$$(\textsc{In}) \quad 0 \xrightarrow{l\cdot c(\underline{\tilde{v}})} \overline{l{\cdot}c}\langle\underline{\tilde{v}}\rangle \qquad (\textsc{Out}) \quad \overline{l{\cdot}c}\langle\tilde{v}\rangle \xrightarrow{(\tilde{k})\overline{l\cdot c}\langle\tilde{u}\rangle} \Theta$$

where $F(\tilde{v}) = (\tilde{u}; \Theta; \tilde{k})$. Any scripted process in v is replaced by a trigger in $\tilde{u}$, and Θ is the parallel composition of all the definitions associated to $\tilde{k}$. In an input transition values must be necessarily first-order. The rule for updates is

$$(\textsc{Update}) \quad l{\cdot}\mathsf{update}_p(\chi, V).P \xrightarrow{(\tilde{k})l\cdot\mathsf{update}_p(\underline{\tilde{U}},(\chi)\tilde{V'})} R \mid \Theta$$

for any first-order $\tilde{U} = \{U_1, \cdots, U_n\}$, $F(V_1, \ldots, V_n) = (\tilde{V'}; \Theta; \tilde{k})$ where all $V_i = V$, $\tilde{k}$ is fresh, and $R = P\{U_1/\chi\} \mid \cdots \mid P\{U_n/\chi\}$. The conditions are determined by the idea that $\tilde{U}$ can be considered as (first-order) parameters received in input, and $\tilde{V}$ as parameters of as many output actions as there are input matches. We conclude with the rules for running a script and analysing its definition:

$$(\textsc{Run}) \quad l{\cdot}\mathsf{run}_p \xrightarrow{l\cdot\mathsf{run}_p} 0 \qquad (\textsc{Trigger}) \quad \langle k \Leftarrow \Box P\rangle \xrightarrow{l\cdot k(p)} \langle k \Leftarrow \Box P\rangle \mid P\{l/\circlearrowleft\}\{p/.\}$$

The first rule simply records the location and path from which we run a script; the second one effectively executes a copy of a script, initialised at l and p.

4.2 Domain Bisimilarity

We introduce now our bisimulation equivalence. The intuition is that when two processes are running in a domain Λ, we need to check that, if a process makes an action α_l with $l \in \Lambda$, then the other one can mimic it, possibly relying on the existence of other locations in Λ. If $l \notin \Lambda$ we need not worry about matching actions. But since the domain can be extended by composing networks, we need to make sure that actions not in Λ are also matched, this time in a different relation parameterised by $\Lambda \cup \{l\}$.

We use the notation $K \xrightarrow{\tau}_\Lambda K'$ if $K \xrightarrow{l\cdot\tau} K'$ for some $l \in \Lambda$, and $\overset{\alpha_l}{\twoheadrightarrow}_\Lambda \triangleq {\xrightarrow{\tau}_\Lambda}^* \circ \xrightarrow{\alpha_l} \circ {\xrightarrow{\tau}_\Lambda}^*$ if $l \in \Lambda, \alpha_l \neq l{\cdot}\tau$, and $\overset{\tau}{\twoheadrightarrow}_\Lambda \triangleq {\xrightarrow{\tau}_\Lambda}^*$. The function $bn(-)$ extends to triggers in the obvious way. We say that an action α_l is relevant to a configuration K, abbreviated by $rel(\alpha_l, K)$, if $bn(\alpha_l) \cap fn(K) = \emptyset$.

Definition 4.1 *A family of symmetric relations on configurations (indexed with sets of locations)* $\dot{\approx} = \{\dot{\approx}_\Lambda | \Lambda \subseteq \mathcal{L}\}$ *is a* domain bisimulation *if* $K \dot{\approx}_\Lambda L$ *and* $K \xrightarrow{\alpha_l} K'$ *implies:*

1. *if* $l \in \Lambda$ *with* $rel(\alpha_l, L)$ *then* $L \overset{\alpha'_l}{\twoheadrightarrow}_\Lambda L'$ *where* $\alpha'_l \equiv \alpha_l$ *and* $K' \dot{\approx}_\Lambda L'$*;*
2. *if* $l \notin \Lambda$ *then* $K \dot{\approx}_{\Lambda\cup\{l\}} L$*.*

Domain bisimilarity ($\approx$) is the (pointwise) largest domain bisimulation. Two open processes P, Q are Λ-bisimilar iff for all closing substitutions σ, $P\sigma \approx_\Lambda Q\sigma$.

In Appendix E we show that domain bisimilarity can be defined as the largest fixpoint of a monotonic operator on families of relations.

The proof technique needed to show that $K \approx_\Lambda L$, consists of exhibiting a domain bisimulation $\dot{\approx} = \{\dot{\approx}_\Delta | \Lambda \subseteq \Delta \subseteq \mathcal{L}\}$ such that $K \dot{\approx}_\Lambda L$. It is less burdensome than it may seem: the family is monotonic, and therefore starting from the pairs in $\dot{\approx}_\Lambda$, we can build each $\dot{\approx}_{\Delta\cup\{l\}}$ from $\dot{\approx}_\Delta$ adding only the pairs where the first component makes a move at l.

Theorem 4.1 *For all Λ, (i) if $\Lambda \subset \Lambda'$ then $\approx_\Lambda \subset \approx_{\Lambda'}$; (ii) $\approx_\Lambda$ is a congruence on configurations, and the restriction of $\approx_\Lambda$ to processes is a congruence on processes.*

This theorem corresponds to Theorem 3.3, but point (ii) here is much harder to prove since $\approx_\Lambda$ is not contextual *a priori*. The congruence property of $\approx_\Lambda$ plays a fundamental role in the theorem below, justifying the use of domain bisimilarity as a proof method for our process equivalences.

Theorem 4.2 *Process bisimilarity is a sound approximation of process barbed congruence. For all Λ, if $P \approx_\Lambda Q$ then $P \sim^b_\Lambda Q$.*

The converse implication does not hold, as can be seen from $\mathtt{xch}(T,S) \sim^b_\Lambda \mathtt{xch}(S,T)$ and point (1) below.

4.3 Examples

We start with an example of the proof method. We call *distributed equator* the process $\mathtt{dE}(l\cdot a, m\cdot b) = !l\cdot a(\tilde{x}).\overline{m\cdot b}\langle\tilde{x}\rangle \,|\, !m\cdot b(\tilde{x}).\overline{l\cdot a}\langle\tilde{x}\rangle$. It has the effect of making the use of channel a at l undistinguishable from the use of channel m at b, a key property to define optimisations for web services. Let $E_1 = \mathtt{dE}(l\cdot a, m\cdot b) \,|\, \overline{l\cdot a}\langle\tilde{v}\rangle$ and $E_2 = \mathtt{dE}(l\cdot a, m\cdot b) \,|\, \overline{m\cdot b}\langle\tilde{v}\rangle$. We show that $E_1 \approx_{\{l,m\}} E_2$. We need to give a domain bisimulation $\mathcal{R} = \{\mathcal{R}_\Delta\}_{\{l,m\}\subseteq\Delta}$ such that $\mathcal{R}_{\{l,m\}}$ contains the two processes. In this case, it suffices to take the family where $\mathcal{R}_\Delta = \{(E_1,E_2),(E_2,E_1)\} \cup I$ for all Δ, where I is the identity relation. In fact, if $E_1 \xrightarrow{\alpha_l} E_1'$ then $E_2 \xrightarrow{m\cdot\tau}\xrightarrow{\alpha_l} E_1'$, and similarly for m, or $n \notin \{l,m\}$.

Using domain bisimilarity we can show the (non-) equivalences given below (discussed in Section 2 and Section 3):

1. for any Λ, if $S \not\equiv^t T$ then $\mathtt{xch}(T,S) \not\approx_\Lambda \mathtt{xch}(S,T)$;
2. $\mathtt{oldS} \approx_\Lambda \mathtt{newS}$ if and only if $m \in \Lambda$;
3. for any Λ, $!l\cdot a(\tilde{x}).\overline{l\cdot a}\langle\tilde{x}\rangle \approx_\Lambda 0$ and $l\cdot\mathtt{E}(a,b) \,|\, \overline{l\cdot c}\langle a\rangle \not\approx_\Lambda l\cdot\mathtt{E}(a,b) \,|\, \overline{l\cdot c}\langle b\rangle$;
4. $(\nu\, get)(m\cdot\mathtt{get} \,|\, l\cdot\mathtt{call}(m, get, p)) \approx_\Lambda (\nu\, get)(m\cdot\mathtt{get} \,|\, l\cdot\mathtt{spec})$ if and only if $m \in \Lambda$.

We conclude with an example on replication of web services. Consider the two services, meant to be interchangeable, defined as $\mathtt{s}_1 = !m\cdot b(\tilde{x},y,z).(\overline{n\cdot a}\langle\tilde{x},y,z\rangle \oplus_m S)$ and $\mathtt{s}_2 = !n\cdot a(\tilde{x},y,z).(\overline{m\cdot b}\langle\tilde{x},y,z\rangle \oplus_n S)$, where $P \oplus_l Q = (\nu c)(\overline{l\cdot c}\langle\rangle \,|\, l\cdot c().P \,|\, l\cdot c().Q)$. Both offer the same functionality S, but an internal choice determines whether the service will be provided locally, or delegated to the other location. We can now embed in the data indifferently a service call to $\mathtt{s_1}$ or one to $\mathtt{s_2}$, as justified by the equation given below

$$\mathtt{s}_1 \,|\, \mathtt{s}_2 \,|\, l\cdot\mathsf{paste}_p\langle\mathtt{sc}[\,\Box l\cdot\mathtt{call}(m,b,\tilde{v})\,]\rangle \approx_{\{m,n\}} \mathtt{s}_1 \,|\, \mathtt{s}_2 \,|\, l\cdot\mathsf{paste}_p\langle\mathtt{sc}[\,\Box l\cdot\mathtt{call}(n,a,\tilde{v})\,]\rangle$$

5 Conclusions

Our work shows how a behavioural understanding of systems for the exchange of dynamic data on the web can be grounded on the existing research on process calculi, but requires also the development of new techniques.

We leave to future work the study of complete characterisations of the contextual equivalences, which we believe could be based on weak bisimulations able to abstract away (partly) from update actions.

5.1 Acknowledgments

We thank Alex Ahern, Martin Berger, Cristiano Calcagno, Jonathan Hayman, Andrew Phillips, Iain Phillips, Maria Grazia Vigliotti, Nobuko Yoshida and Uri Zarfaty for useful comments and suggestions.

References

[1] Abiteboul, S., A. Bonifati, G. Cobena, I. Manolescu and T. Milo, *Dynamic XML documents with distribution and replication*, in: *Proceedings of SIGMOD'03*, 2003.

[2] Abiteboul, S. et al., *Active XML primer*, INRIA Futurs, GEMO Report number 275 (2003).

[3] Bierman, G. and P. Sewell, *Iota: a concurrent XML scripting language with application to Home Area Networks*, University of Cambridge Technical Report 557 (2003).

[4] Braumandl, R., M. Keidl, A. Kemper, D. Kossmann, A. Kreutz, S. Seltzsam and K. Stocker, *Objectglobe: Ubiquitous query processing on the internet*, To appear in the VLDB Journal: Special Issue on E-Services (2002).

[5] Carbone, M. and S. Maffeis, *On the expressive power of polyadic synchronisation in π-calculus*, Nordic Journal of Computing **10** (2003), pp. 70–98.

[6] Cardelli, L. and G. Ghelli, *A query language based on the ambient logic*, in: *Proceedings of ESOP'01*, LNCS **2028** (2001), pp. 1–22.

[7] Gardner, P. and S. Maffeis, *Modeling dynamic Web data*, in: G. Lausen and D. Suciu, editors, *Proc. of DBPL'03* (2003).

[8] Gardner, P. and S. Maffeis, *Modelling dynamic Web data*, Imperial College London Technical Report (2003).

[9] Gordon, A. and R. Pucella, *Validating a web service security abstraction by typing*, in: *Proceedings of the 2002 ACM Workshop on XML Security*, 2002, pp. 18–29.

[10] Hennessy, M. and J. Riely, *Resource access control in systems of mobile agents*, in: *Proceedings of HLCL '98*, ENTCS **16.3** (1998), pp. 3–17.

[11] Honda, K. and M. Tokoro, *An object calculus for asynchronous communication*, in: *Proceedings of ECOOP*, LNCS **512** (1991), pp. 133–147.

[12] Honda, K. and M. Tokoro, *On asynchronous communication semantics*, LNCS **612** (1992), pp. 21–51.

[13] Honda, K. and N. Yoshida, *On reduction-based process semantics*, Theoretical Computer Science **151** (1995), pp. 437–486.

[14] Jeffrey, A. and J. Rathke, *Contextual equivalence for higher-order pi-calculus revisited*, Computer Science Report 04/2002, University of Sussex (2002).

[15] Kemper, A. and C. Wiesner, *Hyperqueries: Dynamic distributed query processing on the internet*, in: *Proceedings of VLDB'01*, 2001, pp. 551–560.

[16] World Wide Web Consortium, *XML Path Language (XPath) Version 1.0*, available at `http://w3.org/TR/xpath`.

[17] Milner, R., "Communication and concurrency," Prentice-Hall, Inc., 1989.

[18] Milner, R., J. Parrow and J. Walker, *A calculus of mobile processes, I and II*, Information and Computation **100** (1992), pp. 1–40,41–77.

[19] Sahuguet, A., "ubQL: A Distributed Query Language to Program Distributed Query Systems," Ph.D. thesis, University of Pennsylvania (2002).

[20] Sahuguet, A., B. Pierce and V. Tannen, *Distributed Query Optimization: Can Mobile Agents Help?*, Unpublished draft.

[21] Sahuguet, A. and V. Tannen, *Resource Sharing Through Query Process Migration*, University of Pennsylvania Technical Report MS-CIS-01-10 (2001).

[22] Sangiorgi, D. and D. Walker, "The π-calculus: a Theory of Mobile Processes," Cambridge University Press, 2001.

[23] Sangirogi, D., *Expressing mobility in process algebras: First-order and higher-order paradigms*, PhD thesis, University of Edinburgh (1992).

A Tables

$$\frac{\begin{array}{l} F(v) = (v'; \Theta; \tilde{k}) \\ F(\tilde{v}) = (\tilde{v'}; \Theta'; \tilde{k'}) \\ \tilde{k} \cap (\tilde{k'} \cup fn(v) \cup fn(\tilde{v})) = \emptyset \\ \tilde{k'} \cap (\tilde{k} \cup fn(v) \cup fn(\tilde{v})) = \emptyset \end{array}}{F(v, \tilde{v}) = (v', \tilde{v'}; \Theta \mid \Theta'; \tilde{k}, \tilde{k'})} \qquad \frac{\begin{array}{l} F(U) = (U'; \Theta; \tilde{k}) \\ F(\tilde{U}) = (\tilde{U}'; \Theta'; \tilde{k'}) \\ \tilde{k} \cap (\tilde{k'} \cup fn(U) \cup fn(\tilde{U})) = \emptyset \\ \tilde{k'} \cap (\tilde{k} \cup fn(U) \cup fn(\tilde{U})) = \emptyset \end{array}}{F(U, \tilde{U}) = (U', \tilde{U}'; \Theta \mid \Theta'; \tilde{k}, \tilde{k'})}$$

$$F(c) = (c; 0; ()) \qquad F(k) = (k; 0; ())$$

$$F(l) = (l; 0; ()) \qquad F(p) = (p; 0; ())$$

$$F(@l{:}p) = (@l{:}p; 0; ()) \qquad F(0) = (0; 0; ())$$

$$F(\Box P) = (k; \langle k \Leftarrow \Box P \rangle; k) \qquad \frac{\begin{array}{l} F(T_1) = (T_1'; \Theta_1; \tilde{k_1}) \\ F(T_2) = (T_2'; \Theta_2; \tilde{k_2}) \\ \tilde{k_1} \cap (\tilde{k_2} \cup fn(T_1) \cup fn(T_2)) = \emptyset \\ \tilde{k_2} \cap (\tilde{k_1} \cup fn(T_1) \cup fn(T_2)) = \emptyset \end{array}}{F(T_1 \mid T_2) = (T_1' \mid T_2'; \Theta_1 \mid \Theta_2; \tilde{k_1}, \tilde{k_2})}$$

$$\frac{F(T) = (T'; \Theta; \tilde{k})}{F(\mathsf{a}[\,T\,]) = (\mathsf{a}[\,T'\,]; \Theta; \tilde{k})}$$

Table 3: Relation F

(Res) $$\frac{K \xrightarrow{\alpha_l} K'}{(\nu c)K \xrightarrow{\alpha_l} (\nu c)K'}\ c \notin n(\alpha_l)$$

(Par) $$\frac{K \xrightarrow{\alpha_l} K'}{K \mid L \xrightarrow{\alpha_l} K' \mid L}\ bn(\alpha_l) \cap fn(L) = \emptyset$$

(Struct) $$\frac{K \equiv L \xrightarrow{\alpha_l} L' \equiv K'}{K \xrightarrow{\alpha_l} K'}$$

(Open) $$\frac{K \xrightarrow{(\tilde{d},\tilde{k})\overline{l\cdot c}\langle\tilde{v}\rangle} K'}{(\nu b)K \xrightarrow{(b,\tilde{d},\tilde{k})\overline{l\cdot c}\langle\tilde{v}\rangle} K'}\ b \neq c, b \in fn(\tilde{v})$$

(Com) $$\overline{l\cdot c}\langle\tilde{v}\rangle \mid l\cdot c(\tilde{x}).P \xrightarrow{l\cdot\tau} P\{\tilde{v}/\tilde{x}\}$$

(Com!) $$\overline{l\cdot c}\langle\tilde{v}\rangle \mid !l\cdot c(\tilde{x}).P \xrightarrow{l\cdot\tau} !l\cdot c(\tilde{x}).P \mid P\{\tilde{v}/\tilde{x}\}$$

(Update) $$l\cdot\mathsf{update}_p(\chi, V).P \xrightarrow{(\tilde{k})l\cdot\mathsf{update}_p(\underline{\tilde{U}},(\chi)\tilde{V}')} P\{U_1/\chi\} \mid \cdots \mid P\{U_n/\chi\} \mid \Theta$$

for any $\tilde{U} = \{U_1, \ldots, U_n\}$ with $\tilde{k}$ fresh, and $F(V_1, \ldots, V_n) = (\tilde{V}'; \Theta; \tilde{k})$, where $V_i = V$

(Out) $$\overline{l\cdot c}\langle\tilde{v}\rangle \xrightarrow{(\tilde{k})\overline{l\cdot c}\langle\tilde{v'}\rangle} \Theta \text{ where } F(\tilde{v}) = (\tilde{v'}; \Theta; \tilde{k})$$

(Run) $$l\cdot\mathsf{run}_p \xrightarrow{l\cdot\mathsf{run}_p} 0$$

(Trigger) $$\langle k \Leftarrow \Box P\rangle \xrightarrow{l\cdot k(p)} \langle k \Leftarrow \Box P\rangle \mid P\{l/\circlearrowleft\}\{p/.\}$$

(In) $$0 \xrightarrow{l\cdot c(\underline{\tilde{v}})} \overline{l\cdot c}\langle\underline{\tilde{v}}\rangle$$

Table 4: Labelled Transition System

B Proofs of Section 3

B.1 Robustness of $\simeq^t$

Lemma B.1 *For all networks N, there exists a context $C = \mathtt{check}(l{\cdot}T, n, ok, ko)$, where n, ok, ko are location names not appearing in N, such that*

1. *$N \downarrow_{l\cdot T}$ implies exists N' such that $C\{N\} \to^* N'$ and $N' \downarrow_{ok}$ and $N' \not\Downarrow_{ko}$;*

2. *$C\{N\} \to^* N'$ and $N' \Downarrow_{ok}$ and $N' \not\Downarrow_{ko}$ implies $N \Downarrow_{l\cdot T}$.*

Proof. (Sketch). Consider the process

$$P = l{\cdot}\mathsf{copy}_{/}(X).n{\cdot}\mathsf{update}_{/}(Y, X).n{\cdot}\mathtt{chk}(T, ko).ok{\cdot}\mathsf{cut}_{/}(X)$$

where $n{\cdot}\mathtt{chk}(T, ko).-$ is a prefix which analyses at location n the structure of T. If the tree has the expected structure, the prefix will allow the continuation to execute, otherwise it will not produce an error signal (an observation at ko). Context $\mathtt{check}(l{\cdot}T, n, m)$ is given by

$$(\{n \mapsto 0\} \uplus \{ok \mapsto \mathtt{a}[\,0\,]\} \uplus \{ko \mapsto \mathtt{a}[\,0\,]\} \uplus -, P \,|\, -)$$

Since n is fresh, there can be no interference from processes in N on the working of $n{\cdot}\mathtt{chk}(T, ko)$, and since ok is fresh, we can observe ok only if the checking has been successful. For each T, a process $n{\cdot}\mathtt{chk}(T, ko)$ can be programmed by recursively analysing the top level of the tree, using restricted channels in a way to get stuck if some expected node is not found, and to produce an observation at ko if there are more nodes than expected, for each sort of nodes. The construction depends heavily on the paths and update semantics, but is possible (following *ad-hoc* strategies) for a large set of querying and update languages. □

Statement of Theorem 3.1. *The relations $\simeq^t$, $\simeq^0$ and $\simeq^{\langle\rangle}$ coincide.*

Proof.

- ($\simeq^t \subseteq \simeq^0$): follows directly from the definitions, noticing that $N \downarrow_{l\cdot 0}$ coincides with $N \downarrow_l$.

- ($\simeq^0 \subseteq \simeq^t$): given $N \simeq^0 M$ we want to show that $N \simeq^t M$. Since both relations are reduction closed and contextual, all we need to show is that $N \simeq^0 M$ implies $N \downarrow_{l\cdot T} \Longrightarrow M \Downarrow_{l\cdot T}$. Suppose $N \downarrow_{l\cdot T}$, and consider the context $C = \mathtt{check}(l{\cdot}T, n, ok, ko)$ of Lemma B.1, for some fresh locations n, ok, ko. By point (1) of the lemma, there exists N' such that $C\{N\} \to^* N'$, $N' \downarrow_{ok}$, and $N' \not\Downarrow_{ko}$.Since $\simeq^0$ is contextual, $C\{N\} \simeq^0 C\{M\}$, and since it is reduction closed, there exists M' such that $C\{M\} \to^* M'$ and $N' \simeq^0 M'$. But then it must be the case that $M' \Downarrow_{ok}$ and $M' \not\Downarrow_{ko}$, and by point (2) of the lemma we conclude with $M \Downarrow_{l\cdot T}$.

- ($\simeq^t \subseteq \simeq^{\langle\rangle}$): similar to the previous case. Suppose $N \downarrow_{l\cdot a}$ and consider the context $C = (\{m \mapsto 0\} \uplus -, l{\cdot}a(\tilde{x}).m{\cdot}\mathsf{paste}_{/}\langle \mathtt{ok}[\,0\,]\rangle \,|\, -)$, where m is a fresh location name and $\tilde{x}$ has the same arity as the object of the output causing $\downarrow_{l\cdot a}$. We have that $N \downarrow_{l\cdot a}$ implies $C\{N\} \Downarrow_{m\cdot \mathtt{ok}[\,0\,]}$, which by contextuality of $\simeq^t$ implies $C\{M\} \Downarrow_{m\cdot \mathtt{ok}[\,0\,]}$, which can happen only if $M \Downarrow_{l\cdot a}$, since m is fresh.

- ($\simeq^{\langle\rangle} \subseteq \simeq^t$): analogous to ($\simeq^0 \subseteq \simeq^t$), adapting the context `check` of Lemma B.1 in order to use $\Downarrow_{l\cdot ok}$ for success and $\Downarrow_{l\cdot ko}$ for failure, for some fresh channel names *ok* and *ko*.

□

B.2 Soundness of $\simeq^b$

Statement of Theorem 3.2. *Barbed congruence strictly implies tree congruence:* $\simeq^b \subsetneq \simeq^t$.

Proof. (Sketch). ($\simeq^b \subset \simeq^t$): by reasoning as in the proof of Theorem 3.1, adapting Lemma B.1 to show success with a barb at some fresh location called *ok* and failure with a barb at *ko*. ($\simeq^b \neq \simeq^t$): the networks $N = (\{l \mapsto T\}, 0)$ and $M = (\{l \mapsto T\}, l\cdot\mathsf{copy}_p(X).0)$ constitute a counterexample. In fact $N \simeq^t M$ because the stores are the same and $l\cdot\mathsf{copy}_p(X).0$ is not able to affect either the store or other computations, but $N \not\simeq^b M$, since the barbs are different. □

B.3 Monotonicity and congruence of $\sim^t_\Lambda$ and $\sim^b_\Lambda$

Statement of Theorem 3.3. *For all* Λ, *(i) if* $\Lambda \subset \Lambda'$ *then* $\sim^t_\Lambda \subset \sim^t_{\Lambda'}$ *and* $\sim^b_\Lambda \subset \sim^b_{\Lambda'}$; *(ii)* $\sim^t_\Lambda$ *and* $\sim^b_\Lambda$ *(both on open and closed processes) are congruences over full contexts.*

Proof. (i) In both cases the inclusion follows by Definition 3.6, and the strictness by deriving a suitable counterexample from processes `oldS` and `newS` (see Lemma C.3). (ii) We start with the case for open processes. By Definition 3.6 we have that $\sim^t_\Lambda$ and $\sim^b_\Lambda$ are closed under reduction contexts for processes. We need to show that they are also closed under all prefixes.

We show the case for $\sim^t_\Lambda$, the case for $\sim^b_\Lambda$ follows a similar structure. Let $K ::= l\cdot a(\tilde{x}).- \mid\ !l\cdot a(\tilde{x}).- \mid l\cdot\mathsf{update}_p(\chi, V).-$, and let C, C' be generic reduction contexts. We show that for an arbitrary D with $\Lambda \subseteq dom(D)$, the relation

$$\mathcal{S} = \{((D, C\{K\{P\}\}), (D, C\{K\{Q\}\}))|P \sim^t_\Lambda Q\} \cup \simeq^t$$

is included in $\simeq^t$. By construction $\mathcal{S}$ is closed under reduction contexts and respects observables. We show that it is reduction closed by induction on the structure of C, and we limit our analysis to the most representative cases. The base case is when we have the pair $(D, K\{P\}), ((D, K\{Q\})$, where reduction is possible only for $K = l\cdot\mathsf{update}_p(\chi, V).-$. If the update reduction can take place then we get $(D', P\sigma_1 \mid \ldots \mid P\sigma_n)$ for the first component, and we can do the same move with the second obtaining $(D', Q\sigma_1 \mid \ldots \mid Q\sigma_n)$, which by $P \sim^t_\Lambda Q$, transitivity of $\sim^t_\Lambda$, and Definition 3.8, belongs to $\simeq^t$ (notice that by definition $\simeq^t \subseteq \mathcal{S}$, and $\mathcal{S}$ is transitive).

Another interesting case is when $C = \overline{l\cdot a}\langle\tilde{v}\rangle|-$ and $K =!l\cdot a(\tilde{x}).-$. Then we have $(D, K\{P\}) \rightarrow (D, K\{P\} \mid P\{\tilde{v}/\tilde{x}\})$ and also $(D, K\{Q\}) \rightarrow (D, K\{Q\} \mid Q\{\tilde{v}/\tilde{x}\})$. By $P \sim^t_\Lambda Q$ and Definition 3.8, we have $(D, P\{\tilde{v}/\tilde{x}\}) \simeq^t (D, Q\{\tilde{v}/\tilde{x}\})$ and by contextuality of $\simeq^t$ we get $(D, K\{P\} \mid P\{\tilde{v}/\tilde{x}\}) \simeq^t (D, K\{P\} \mid Q\{\tilde{v}/\tilde{x}\})$, and by inductive hypothesis $(D, K\{P\}) \simeq^t (D, K\{Q\})$, which implies $(D, K\{P\} \mid Q\{\tilde{v}/\tilde{x}\}) \simeq^t (D, K\{Q\} \mid Q\{\tilde{v}/\tilde{x}\})$, and by transitivity $(D, K\{P\} \mid P\{\tilde{v}/\tilde{x}\}) \simeq^t (D, K\{Q\} \mid Q\{\tilde{v}/\tilde{x}\})$. The case for closed processes is easier, because it does not need Definition 3.8. □

B.4 Maximality of $\sim^t$ and $\sim^b$

Statement of Theorem 3.4. *(i)* $\Pi_2^c(\simeq^t)$ *coincides with* $\sim^t$*. (ii)* $\Pi_2^c(\simeq^b)$ *coincides with* $\sim^b$*.*

Proof. Both $\sim^t \subseteq \Pi_2^c(\simeq^t)$ and $\sim^b \subseteq \Pi_2^c(\simeq^b)$ follow from point (ii) of Theorem 3.3, and by definition of the relations (Definition 3.6 and Definition 3.9).

We show the converse inclusion for $\sim^b$ first. We give the case for stores composed by a single location, which can be easily generalised. Assume that for some l, S, T, we have $(\{l \mapsto S\}, P) \simeq^b (\{l \mapsto T\}, Q)$. Let $K = l{\cdot}\mathsf{update}_{/}(X, R).l{\cdot}a().-$, for arbitrary R, a. Since $\Pi_2^c(\simeq^t)$ is a congruence, we must have that $N = (\{l \mapsto S\}, K\{P\}) \simeq^b (\{l \mapsto T\}, K\{Q\}) = M$. Since $N \to (\{l \mapsto R\}, l{\cdot}a(x).P) = N'$, and $N' \not\Downarrow_{m\cdot p}$ for any m, p, then by reduction closure it must be the case that $M \to (\{l \mapsto R\}, l{\cdot}a(x).Q) = M' \simeq^b N'$, otherwise a barb would distinguish the two processes. We can now apply closure under reduction contexts obtaining, for $C = (\emptyset, (\nu a)(\overline{l{\cdot}a}\langle\rangle \mid -))$, $C\{N'\} \simeq^t C\{M'\}$. But since $C\{N'\} \simeq^t (\{l \mapsto R\}, P)$ and $C\{M'\} \simeq^t (\{l \mapsto R\}, Q)$, by transitivity of $\simeq^t$ we have $(\{l \mapsto R\}, P) \simeq^t (\{l \mapsto R\}, Q)$. Since R is arbitrary, we have $\forall R.(\{l \mapsto R\}, P) \simeq^t (\{l \mapsto R\}, Q)$, and generalising this construction for any location in a store, we get $\Pi_2^c(\simeq^b) \subseteq \sim^b$.

The proof for $\sim^t$ is similar but relies on Theorem 3.1: it is in fact convenient to use $\simeq^{\langle\rangle}$ as a starting relation, in order to force the step from M to M' reasoning on channel observables. □

C Monotonicity and congruence of $\approx_\Lambda$

The proof that $\approx_\Lambda$ is a congruence adapts the techniques for the asynchronous π-calculus of Honda and Yoshida [13] to the framework of Sangiorgi [23] and Jeffrey and Rathke [14]. The techniques can be adapted to domain bisimilarity fairly easily, but our congruence proof is complicated by the incompleteness of $\approx_\Lambda$. In fact, closure under parallel composition of $\approx_\Lambda$ turns out to be difficult to prove directly. For example, Jeffrey and Rathke are able to prove soundness of their bisimilarity by showing that an auxiliary relation, only reduction closed, is closed under parallel composition, and then, by showing completeness, they derive that bisimilarity is closed under parallel composition.

It is important to keep in mind, during the proofs, the distinction between session names and public names, and that a scripted processes is well-formed only if it has no free session names. Moreover, as previously stated, configurations are well-formed if for any trigger name there is at most one definition, and the scripted processes in the definitions do not contain trigger names.

In all the proofs we reason modulo structural congruence of the lts actions.

C.1 Basic lemmas

Many proofs rely on the technical lemma given below, relating the transitions in the lts with the syntactic structure of configurations, up to structural congruence. The lemma which will be used in the proofs below without stating it explicitly.

Lemma C.1

1. $K \xrightarrow{(\tilde{b},\tilde{k})\overline{l\cdot c}\langle\tilde{u}\rangle} K'$ *if and only if* $K \equiv (\nu\tilde{b})(L \,|\, \overline{l\cdot c}\langle\tilde{v}\rangle)$ *where* $c \notin \{\tilde{b}\}$ *(and* $\{\tilde{b}\} \subseteq \{\tilde{v}\}$*), and* $K' \equiv L \,|\, \Theta$ *where* $F(\tilde{v}) = (\tilde{u}; \Theta; \tilde{k})$.

2. $K \xrightarrow{l\cdot c(\tilde{v})} K'$ *if and only if* $K' \equiv K \,|\, \overline{l\cdot c}\langle\tilde{v}\rangle$.

3. $K \xrightarrow{l\cdot\tau} K'$ *if and only if*
   - $K \equiv (\nu\tilde{b})(L \,|\, l\cdot c(\tilde{x}).Q \,|\, \overline{l\cdot c}\langle\tilde{v}\rangle)$ *and* $K' \equiv (\nu\tilde{b})(L \,|\, Q\{\tilde{v}/\tilde{x}\})$*, or*
   - $K \equiv (\nu\tilde{b})(L \,|\, !l\cdot c(\tilde{x}).Q \,|\, \overline{l\cdot c}\langle\tilde{v}\rangle)$ *and* $K' \equiv (\nu\tilde{b})(L \,|\, !l\cdot c(\tilde{x}).Q \,|\, Q\{\tilde{v}/\tilde{x}\})$.

4. $K \xrightarrow{(\tilde{k})l\cdot\mathsf{update}_p(\tilde{U},(\chi)\tilde{V'})} K'$ *if and only if* $K \equiv (\nu\tilde{b})(L \,|\, l\cdot\mathsf{update}_p(\chi, V).Q)$ *and* $K' \equiv (\nu\tilde{b})(L \,|\, Q\{U_1/\chi\} \,|\, \cdots \,|\, Q\{U_n/\chi\} \,|\, \Theta)$*, for some* $\tilde{U} = \{U_1, \cdots, U_n\}$ *with* $F(V_1, \ldots, V_n) = (\tilde{V'}; \Theta; \tilde{k})$*,* $\tilde{k}$ *fresh, and each* $V_i = V$.

5. $K \xrightarrow{l\cdot\mathsf{run}_p} K'$ *if and only if* $K \equiv K' \,|\, l\cdot\mathsf{run}_p$.

6. $K \xrightarrow{l\cdot k(p)} K'$ *if and only if* $K \equiv L \,|\, \langle k \Leftarrow \Box P\rangle$ *and* $K' \equiv K \,|\, P\{l/\circlearrowleft\}\{p/.\}$.

Proof. The *if* part follows immediately by definition of lts.

The *only if* part follows by induction on the number of derivation steps needed to obtain the labelled transition. We give the case for bound output (1) as an example. Suppose $K \xrightarrow{(\tilde{b},\tilde{k})\overline{l\cdot c}\langle\tilde{u}\rangle} K'$ is derived in a single step (i.e. using the axiom (Out)). The only possibility is for $K = \overline{l\cdot c}\langle\tilde{v}\rangle$ and for $F(\tilde{v}) = (\tilde{u}; \Theta; \tilde{k})$ we conclude with $\{\tilde{b}\} = \emptyset$ and $K' = \Theta$. For the inductive case, assume that $K_1 \xrightarrow{(\tilde{b_1},\tilde{k})\overline{l\cdot c}\langle\tilde{u}\rangle} K_1'$, with $K_1 \equiv (\nu\tilde{b_1})(L_1 \,|\, \overline{l\cdot c}\langle\tilde{v}\rangle)$ where $c \notin \{\tilde{b_1}\}$ (and $\{\tilde{b_1}\} \subseteq \{\tilde{v}\}$), and $K_1' \equiv L_1 \,|\, \Theta$ with $F(\tilde{v}) = (\tilde{u}; \Theta; \tilde{k})$, is derived in n steps. Then a derivation of $n+1$ steps can only be obtained by applying (Res), (Struct), (Par) or (Open). The case for (Struct) is trivial. If rule (Res) is applied then we must have $(\nu d)K_1 \xrightarrow{(\tilde{b_1},\tilde{k})\overline{l\cdot c}\langle\tilde{u}\rangle} (\nu d)K_1'$ with $d \notin n((\tilde{b_1},\tilde{k})\overline{l\cdot c}\langle\tilde{u}\rangle)$. Since F does not affect session channels, we conclude using structural congruence to bring the outermost restriction around L_1: $(\nu d)K_1 \equiv (\nu\tilde{b_1})((\nu d)(L_1) \,|\, \overline{l\cdot c}\langle\tilde{v}\rangle)$, and $(\nu d)K_1' \equiv (\nu\tilde{b_1})((\nu d)(L_1) \,|\, \Theta)$. The cases for (Par) and (Open) are similar. □

Bisimulation up-to structural congruence is used extensively, and is easy to justify in our setting due to the lts (Struct) rule.

Lemma C.2 (Bisimulation up-to structural congruence) *If* $K \equiv M \approx_\Lambda N \equiv L$ *then* $K \approx_\Lambda L$.

Proof. The relation $\mathcal{R} = \big\{(K, L) \;\big|\; K \equiv M \approx_\Delta N \equiv L\big\}_{\Lambda\subseteq\Delta}$ is a domain bisimulation. Follows easily using rule (Struct) of the lts. □

C.2 Monotonicity of $\approx_\Lambda$

Lemma C.3 *Consider the two processes* $\texttt{oldS}(l) = l{\cdot}\texttt{cut}_/(X)$ *and* $\texttt{newS}(l,m) = (\nu c)(\overline{m{\cdot}c}\langle/\rangle \mid m{\cdot}c(x).l{\cdot}\texttt{cut}_x(X))$*, where* $l \neq m$*. Then*

1. *if* $m \in \Lambda$ *then* $\texttt{oldS}(l) \approx_\Lambda \texttt{newS}(l,m)$*;*
2. *if* $m \notin \Lambda$ *then* $\texttt{oldS}(l) \not\approx_\Lambda \texttt{newS}(l,m)$*.*

Proof.

1. We show that the family R where

$$\mathcal{R}_\Delta = \{(\texttt{oldS}(l), \texttt{newS}(l,m)), (\texttt{newS}(l,m), \texttt{oldS}(l))\} \cup I$$

for any $\Delta \supseteq \Lambda$ is a domain bisimulation. Suppose $\texttt{oldS}(l) \xrightarrow{\alpha_l} K$. If $l \in \Lambda$ then we have that $\texttt{newS}(l,m) \xrightarrow{m\cdot\tau} \texttt{oldS}(l)$, and therefore $\texttt{newS}(l,m) \overset{\alpha_l}{\twoheadrightarrow}_\Lambda K$ since $m \in \Lambda$, and we conclude because $(K,K) \in I$ implies $(K,K) \in \mathcal{R}_\Lambda$. If $l \notin \Lambda$ the same argument applies to $\mathcal{R}_{\Lambda\cup\{l\}}$.

2. We have two cases. If $l \in \Lambda$ then $\texttt{oldS}(l)$ can make an update transition at l, but $\texttt{newS}(l,m)$ cannot, since the only move it can make is a τ at m, which is not allowed because $m \notin \Lambda$. If $l \notin \Lambda$ the same argument can be applied to the relation $\approx_{\Lambda\cup\{l\}}$.

$\square$

Statement of point (i) of Theorem 4.1. *For all* Λ*, if* $\Lambda \subset \Lambda'$ *then* $\approx_\Lambda \subset \approx_{\Lambda'}$*.*

Proof. The inclusion follows by Definition 4.1, noticing that with rule (In) in the lts it is always possible to make an input action at location l, for any l also not in Λ. The inclusion is strict: if $\Lambda \subset \Lambda'$ then there exists an m such that $m \in \Lambda' \setminus \Lambda$, and by Lemma C.3 we have, for any $l \neq m$, $\texttt{oldS}(l) \approx_{\Lambda'} \texttt{newS}(l,m)$ but $\texttt{oldS}(l) \not\approx_\Lambda \texttt{newS}(l,m)$. $\square$

C.3 Variant Lemma

The first step towards the proof is to generalise the variant lemma of [13] to channel and trigger names. Moreover, we extend it to bijective substitutions (here called *switchings*), obtaining a simpler, purely co-inductive proof. Below we let a, b, c range over channel or trigger names, and we consider only well-sorted substitutions (i.e. replacing channels for channels and triggers for triggers).

Definition C.1 (Switching) *We use the notation* $K^{a\frown b}$ *(switching) for the bijective substitution* $K\{c/a\}\{a/b\}\{b/c\}$*, with* $c \notin fn(K)$*.*

Observation C.4 *From the definition of switching and substitution we have that switching is self-dual* $(K^{a\frown b})^{a\frown b} = K$ *and symmetric* $K^{a\frown b} = K^{b\frown a}$*.*

Lemma C.5 *If* $K \xrightarrow{\alpha_l} K'$ *then* $K^{a\frown b} \xrightarrow{\alpha_l^{a\frown b}} K'^{a\frown b}$*.*

Proof. We show the case where $K = P$ and $K' = P'$ are processes. The extension to configurations is trivial. The proof goes by cases on α_l, using Lemma C.1. We show the case for the bound output which is the most interesting, the other cases for α_l are simpler. Let $\rho = ^{a \frown b}$, and suppose, without loss of generality, that $P \xrightarrow{\alpha_l} P'$ with $\alpha_l = (b, \tilde{b})\overline{l \cdot a}\langle \tilde{v} \rangle$. We have that $P \equiv (\nu b, \tilde{b})(P' \,|\, \overline{l \cdot a}\langle \tilde{v} \rangle) \equiv (\nu c, \tilde{c})(P'\{c, \tilde{c}/b, \tilde{b}\} \,|\, \overline{l \cdot a}\langle \tilde{v}\{c, \tilde{c}/b, \tilde{b}\}\rangle)\rho\rho$, where $c, \tilde{c}$ are fresh, and applying the inner switching we obtain

$$(\nu c, \tilde{c})(P'\{c, \tilde{c}/b, \tilde{b}\} \,|\, \overline{l \cdot a}\langle \tilde{v}\{c, \tilde{c}/b, \tilde{b}\}\rangle)\rho\rho = (\nu c, \tilde{c})(P'\{c, \tilde{c}/b, \tilde{b}\}\rho \,|\, \overline{l \cdot b}\langle \tilde{v}\{c, \tilde{c}/b, \tilde{b}\}\rho\rangle)\rho$$
$$= (\nu c, \tilde{c})(P'\{c, \tilde{c}/b, \tilde{b}\}\{b/a\} \,|\, \overline{l \cdot b}\langle \tilde{v}\{c, \tilde{c}/b, \tilde{b}\}\{b/a\}\rangle)\rho$$

where in the second equation we have used the fact that b has been replaced by c. Since a does not appear free anymore, we can alpha-convert c with a and $\tilde{c}$ with $\tilde{b}$ in the last term, which becomes

$$(\nu a, \tilde{c})(P'\{c, \tilde{c}/b, \tilde{b}\}\{b/a\}\{a, \tilde{b}/c, \tilde{c}\} \,|\, \overline{l \cdot b}\langle \tilde{v}\{a, \tilde{c}/b, \tilde{b}\}\{b/a\}\{a, \tilde{b}/c, \tilde{c}\}\rangle)\rho$$

and which by definition of switching is $(\nu a, \tilde{b})(P'\rho \,|\, \overline{l \cdot b}\langle \tilde{v}\rho\rangle)\rho$. We have shown that $P \equiv (\nu a, \tilde{b})(P'\rho \,|\, \overline{l \cdot b}\langle \tilde{v}\rho\rangle)\rho$, from which $P\rho = (\nu a, \tilde{b})(P'\rho \,|\, \overline{l \cdot b}\langle \tilde{v}\rangle\rho) \xrightarrow{\alpha_l \rho} P'\rho$. □

Lemma C.6 (Variant lemma) *If $K \approx_\Lambda L$ then $K^{a \frown b} \approx_\Lambda L^{a \frown b}$. In particular, if $b \notin fn(K, L)$ then $K \approx_\Lambda L$ implies $K\{b/a\} \approx_\Lambda L\{b/a\}$.*

Proof. Let $\rho = ^{a \frown b}$. We show that the family $\mathcal{R} = \big\{(K\rho, L\rho) \;\big|\; K \approx_\Delta L\big\}_{\Lambda \subseteq \Delta}$ is a domain bisimulation. Suppose that $K \approx_\Delta L$ for some Δ. If $K\rho \xrightarrow{\alpha_l} K'$ and $l \notin \Delta$ then by $K \approx_\Delta L$ we get $K \approx_{\Delta \cup \{l\}} L$ and therefore $(K\rho, L\rho) \in \mathcal{R}_{\Delta \cup \{l\}}$. If $l \in \Delta$ and $rel(\alpha_l, L\rho)$ then by Lemma C.5, $K \xrightarrow{\alpha_l \rho} K'\rho$ with $rel(\alpha_l \rho, L)$, by bisimilarity $L \xrightarrow{\alpha'_l \rho}\!\!\!\!\rightarrow_\Delta L'$ with $K'\rho \approx_\Delta L'$, and again by Lemma C.5 $L\rho \xrightarrow{\alpha_l}\!\!\!\!\rightarrow_\Delta L'\rho$, therefore $(K'\rho, L'\rho) \in \mathcal{R}_\Delta$. □

C.4 Congruence of $\approx_\Lambda$

Following Jeffrey and Rathke [14], we use a *merge* operator to reconstruct processes from configurations[4]. The merge operator will play a substantial rôle in showing that $\approx_\Lambda$ is closed under parallel composition.

Definition C.2 (Merge) *The* merge *operator $\langle\!\langle - \rangle\!\rangle$ on well-formed configurations is defined as:*

$$\langle\!\langle P \rangle\!\rangle = P \quad \langle\!\langle (\nu \tilde{c})K \rangle\!\rangle = (\nu \tilde{c})\langle\!\langle K \rangle\!\rangle \quad \langle\!\langle K \,|\, \langle k \Leftarrow \Box P \rangle \rangle\!\rangle = \langle\!\langle K\{\Box P/k\} \rangle\!\rangle$$

Notation 1

- *We denote with $\Theta^{\tilde{k}}$ a configuration $\langle k_1 \Leftarrow \Box P_1 \rangle \,|\, \ldots \,|\, \langle k_n \Leftarrow \Box P_n \rangle$ for some (possibly null) vector $\tilde{k} = k_1, \ldots, k_n$ of distinct triggers, and some set of processes $P_1, \ldots P_n$.*

[4] Since scripted processes in definitions cannot contain triggers, our merge operator is total.

- *Given a configuration* $\Theta^{\tilde{k}} = \langle k_1 \Leftarrow \Box P_1\rangle \,|\, \ldots \,|\, \langle k_n \Leftarrow \Box P_n\rangle$*, we denote by the superscript* $-^{\Theta^{\tilde{k}}}$ *the substitution* $-\{\Box P_1, \ldots, \Box P_n / k_1, \ldots, k_n\}$.
- *Given a configuration* $\Theta^{\tilde{k}}$ *and a vector of distinct triggers* $\tilde{j}$*, such that* $\tilde{k} = k_1, \ldots, k_n$ *and* $\tilde{j} = j_1, \ldots, j_n$*, we denote* $\Theta^{\tilde{k}}\{j_1, \ldots, j_n / k_1, \ldots, k_n\}$ *by* $\Theta^{\tilde{k} \frown \tilde{j}}$.

Note that two groups of definitions $\Theta^{\tilde{k}}$ and $\Theta^{\tilde{j}}$ identified by the same name but by different vectors of triggers can in principle be arbitrarily different. We will also let $\Omega^{\tilde{k}}$ and $\Phi^{\tilde{k}}$ range on groups of definitions.

Observation C.7 *From the syntax of configurations and the well-formedness condition on scripts, for any well-formed configuration* K *there exist a process* P *and a configuration* $\Theta^{\tilde{k}}$ *such that* $K \equiv P \,|\, \Theta^{\tilde{k}}$ *and* $\langle\!\langle K \rangle\!\rangle \equiv P^{\Theta^{\tilde{k}}}$*. Moreover for any vector* $\tilde{j}$, $P^{\Theta^{\tilde{k}}} = P\{\tilde{j}/\tilde{k}\}^{\Theta^{\tilde{k} \frown \tilde{j}}}$ *(if the latter term is defined).*

From now on we assume always to have well-formed configurations, and where not otherwise stated, we assume that the triggers in vectors $\tilde{k}, \tilde{i}, \tilde{j}, \ldots$ are all distinct.

Lemma C.8 *Given* $\Theta^{\tilde{k}}$*, if* $P \xrightarrow{\alpha_l} P'$ *and* $\tilde{k} \cap n(\alpha_l) = \emptyset$ *then* $\langle\!\langle P \,|\, \Theta^{\tilde{k}} \rangle\!\rangle \xrightarrow{\alpha_l} \langle\!\langle P' \,|\, \Theta^{\tilde{k}} \rangle\!\rangle$.

Proof. Follows by syntactical reasoning using Lemma C.1. $\Box$

Lemma C.9 *Let all the configurations below be well-formed.*

1. *If* $K \approx_\Lambda L$ *then* $K \equiv P \,|\, \Theta^{\tilde{k}}$ *and* $L \equiv Q \,|\, \Omega^{\tilde{k}}$.
2. *If* $K \,|\, \Theta^{\tilde{k}} \approx_\Lambda L \,|\, \Omega^{\tilde{k}}$ *then* $K \approx_\Lambda L$.
3. *If* $K \,|\, \Theta^{\tilde{k}} \approx_\Lambda L \,|\, \Omega^{\tilde{k}}$ *then* $K \,|\, \Theta^{\tilde{k}} \,|\, \Theta^{\tilde{k} \frown \tilde{j}} \approx_\Lambda L \,|\, \Omega^{\tilde{k}} \,|\, \Omega^{\tilde{k} \frown \tilde{j}}$.
4. *If* $K \,|\, \Theta^{\tilde{k}} \approx_\Lambda L \,|\, \Omega^{\tilde{k}}$ *then* $K \,|\, \Theta^{\tilde{k} \frown \tilde{j}} \approx_\Lambda L \,|\, \Omega^{\tilde{k} \frown \tilde{j}}$.

Proof.

1. Follows easily from Observation C.7 noticing that if one configuration contains a definition involving a trigger not present in the other configuration they cannot be bisimilar.
2. It is easy to verify that the family of relations
$$\mathcal{R} = \Big\{(K, L) \;\Big|\; K \,|\, \Theta^{\tilde{k}} \approx_\Delta L \,|\, \Omega^{\tilde{k}}\Big\}_{\Lambda \subseteq \Delta}$$
is a domain bisimulation.
3. It is easy to verify that the family of relations
$$\mathcal{R} = \Big\{\Big(K \,|\, \Theta^{\tilde{k}} \,|\, \Theta^{\tilde{k} \frown \tilde{j}}, L \,|\, \Omega^{\tilde{k}} \,|\, \Omega^{\tilde{k} \frown \tilde{j}}\Big) \;\Big|\; K \,|\, \Theta^{\tilde{k}} \approx_\Delta L \,|\, \Omega^{\tilde{k}}\Big\}_{\Lambda \subseteq \Delta}$$
is a domain bisimulation.

4. Follows directly by point (3) and point (2).

$\square$

Lemma C.10 *Suppose $F(\tilde{v}) = (\tilde{v'}; \Theta^{\tilde{k}}; \tilde{k})$. Then*

1. *if $k \in fn(\tilde{v})$ then $k \in fn(\tilde{v'})$;*
2. *if $k \in (fn(\tilde{v'}) \setminus \tilde{k})$ then $k \in fn(\tilde{v})$;*
3. *$\tilde{v} = \tilde{v'}^{\Theta^{\tilde{k}}}$.*

Proof. By induction on the structure of $\tilde{v}$. $\square$

Lemma C.11 *If $P \,|\, \Theta^{\tilde{k}} \,|\, \Theta^{\tilde{j}} \approx_\Lambda P \,|\, \Omega^{\tilde{k}} \,|\, \Omega^{\tilde{j}}$ then $\langle\!\langle P \,|\, \Theta^{\tilde{k}} \rangle\!\rangle \,|\, \Theta^{\tilde{j}} \approx_\Lambda \langle\!\langle Q \,|\, \Omega^{\tilde{k}} \rangle\!\rangle \,|\, \Omega^{\tilde{j}}$.*

Proof. The family of relations

$$\mathcal{R} = \left\{ \left(\langle\!\langle P \,|\, \Theta^{\tilde{k}} \rangle\!\rangle \,|\, \Theta^{\tilde{j}}, \langle\!\langle Q \,|\, \Omega^{\tilde{k}} \rangle\!\rangle \,|\, \Omega^{\tilde{j}} \right) \;\middle|\; P \,|\, \Theta^{\tilde{k}} \,|\, \Theta^{\tilde{j}} \approx_\Delta P \,|\, \Omega^{\tilde{k}} \,|\, \Omega^{\tilde{j}} \right\}_{\Lambda \subseteq \Delta}$$

is a domain bisimulation. Follows by analysing the transitions of $\langle\!\langle P \,|\, \Theta^{\tilde{k}} \rangle\!\rangle \,|\, \Theta^{\tilde{j}}$, using Observation C.7, Lemma C.8 and Lemma C.6. $\square$

Statement of point (ii) of Theorem 4.1. *For all Λ, $\approx_\Lambda$ is a congruence on configurations, and the restriction of $\approx_\Lambda$ to processes is a congruence on processes.*

Proof. We start showing the congruence of $\approx_\Lambda$ on configurations. In particular, for an arbitrary Λ, we have to show the following points:

1. $\approx_\Lambda$ is an equivalence relation;
2. $K \approx_\Lambda L \Longrightarrow K\{\tilde{v}/\tilde{x}\} \approx_\Lambda L\{\tilde{v}/\tilde{x}\}$, for all $\tilde{v}$ and $\tilde{x}$;
3. $K \approx_\Lambda L \Longrightarrow (\nu\tilde{c})K \approx_\Lambda (\nu\tilde{c})L$.
4. $K \approx_\Lambda L \Longrightarrow K \,|\, M \approx_\Lambda L \,|\, M$;

1. Reflexivity and symmetry are immediate. We show transitivity: the family $\mathcal{R} = \big\{(K, L) \;\big|\; K \approx_\Delta M \wedge M \approx_\Delta L\big\}_{\Lambda \subseteq \Delta}$ is a domain bisimulation. let Δ be arbitrary and suppose $K \xrightarrow{\alpha_l} K'$ with $l \notin \Delta$. Then $(K, L) \in \mathcal{R}_{\Delta \cup \{l\}}$ because, by Definition 4.1, $K \approx_{\Delta \cup \{l\}} M$ and $M \approx_{\Delta \cup \{l\}} L$. If $l \in \Delta$ and $rel(\alpha_l, L)$ there are two cases, based on the relevance of α_l to M. If $rel(\alpha_l, M)$ the proof is straightforward. If α_l is not relevant to M then the action must necessarily have some bound names $\tilde{c}$ such that $\{\tilde{c}\} \subseteq fn(M)$. Now choose a $\tilde{c'}$ with the same length as $\tilde{c}$, in such a way that $\{\tilde{c'}\} \cap fn(K, L, M) = \emptyset$. By the side conditions on the lts rule (Par) used to derive the bound transition, $\{\tilde{c}\} \cap fn(K) = \emptyset$ and by Lemma C.6, $K = K\{\tilde{c'}/\tilde{c}\} \approx_\Delta M\{\tilde{c'}/\tilde{c}\} = M'$. Similarly we get $M' \approx_\Delta L$. But now $rel(\alpha_l, M')$, and reasoning as in the previous case, we can conclude.
2. Follows immediately from Definition 4.1.

3. We show that $\mathcal{R} = \left\{((\nu\tilde{c})K, (\nu\tilde{c})L) \;\middle|\; K \approx_\Delta L\right\}_{\Lambda\subseteq\Delta}$ is a domain bisimulation. The input transition requires some attention, as the other ones are easy. We consider only the case where $l \in \Delta$, the other is standard. We have that $(\nu\tilde{c})K \xrightarrow{l\cdot a(\tilde{v})} \overline{l\cdot a}\langle\tilde{v}\rangle \,|\, (\nu\tilde{c})K$. If $\tilde{c} = \tilde{b}, \tilde{c'}$ and $\tilde{b} \subseteq \tilde{v}$ then using alpha-conversion and structural congruence we get $(\nu\tilde{b'}, \tilde{c'})K\{\tilde{b'}/\tilde{b}\} \xrightarrow{l\cdot a(\tilde{v})} (\nu\tilde{b'}, \tilde{c'})(\overline{l\cdot a}\langle\tilde{v}\rangle \,|\, K\{\tilde{b'}/\tilde{b}\})$ where $\tilde{b'}$ are fresh session names. Using the variant lemma (Lemma C.6) on the hypothesis we get $K\{\tilde{b'}/\tilde{b}\} \approx_\Delta L\{\tilde{b'}/\tilde{b}\}$ and since $K\{\tilde{b'}/\tilde{b}\} \xrightarrow{l\cdot a(\tilde{v})} \overline{l\cdot a}\langle\tilde{v}\rangle \,|\, K\{\tilde{b'}/\tilde{b}\}$, it must be the case that $L\{\tilde{b'}/\tilde{b}\} \xrightarrow{l\cdot a(\tilde{v})}\!\!\!\!\twoheadrightarrow_\Delta L' \approx_\Delta \overline{l\cdot a}\langle\tilde{v}\rangle \,|\, K\{\tilde{b'}/\tilde{b}\}$, and we conclude because since $\tilde{v} \cap \tilde{b'}, \tilde{c'} = \emptyset$, we can derive also $(\nu\tilde{b'}, \tilde{c'})(L\{\tilde{b'}/\tilde{b}\}) \xrightarrow{l\cdot a(\tilde{v})}\!\!\!\!\twoheadrightarrow_\Delta (\nu\tilde{b'}, \tilde{c'})L'$.

4. We show that the family of relations $\mathcal{R}$ composed by the pairs

$$(\nu\tilde{c})(\Theta^{\tilde{k}} \,|\, \langle\!\langle P \,|\, \Phi^{\tilde{h}} \rangle\!\rangle \,|\, \langle\!\langle \Theta^{\tilde{j}} \,|\, R \rangle\!\rangle \,|\, \Phi^{\tilde{i}}), (\nu\tilde{c})(\Omega^{\tilde{k}} \,|\, \langle\!\langle Q \,|\, \Phi^{\tilde{h}} \rangle\!\rangle \,|\, \langle\!\langle \Omega^{\tilde{j}} \,|\, R \rangle\!\rangle \,|\, \Phi^{\tilde{i}})$$

such that
$$\Theta^{\tilde{k}} \,|\, \Theta^{\tilde{j}} \,|\, P \approx_\Delta \Omega^{\tilde{k}} \,|\, \Omega^{\tilde{j}} \,|\, Q$$

for $\Lambda \subseteq \Delta$ is a domain bisimulation. (Note that when $\Phi^{\tilde{h}} = \Theta^{\tilde{j}} = 0$ and $\tilde{c}$ is empty, taking $K = \Theta^{\tilde{j}} \,|\, P$, $L = \Omega^{\tilde{j}} \,|\, Q$, and $M = R \,|\, \Phi^{\tilde{i}}$, each $\mathcal{R}_\Delta$ becomes exactly $\{(K \,|\, M, L \,|\, M) \mid K \approx_\Delta L\}$.)

In the proof below we will only consider the transitions for actions α_l with $l \in \Delta$ as the other ones can be dealt with by definition of domain bisimulation, and we will omit the subscript Δ on weak actions for readability. Moreover we will use the abbreviation reported below:

$$\begin{aligned}
K &= (\nu\tilde{c})(\Theta^{\tilde{k}} \,|\, \langle\!\langle P \,|\, \Phi^{\tilde{h}} \rangle\!\rangle \,|\, \langle\!\langle \Theta^{\tilde{j}} \,|\, R \rangle\!\rangle \,|\, \Phi^{\tilde{i}})\\
L &= (\nu\tilde{c})(\Omega^{\tilde{k}} \,|\, \langle\!\langle Q \,|\, \Phi^{\tilde{h}} \rangle\!\rangle \,|\, \langle\!\langle \Omega^{\tilde{j}} \,|\, R \rangle\!\rangle \,|\, \Phi^{\tilde{i}})\\
K_1 &= \Theta^{\tilde{k}} \,|\, \Theta^{\tilde{j}} \,|\, P\\
L_1 &= \Omega^{\tilde{k}} \,|\, \Omega^{\tilde{j}} \,|\, Q
\end{aligned}$$

We use implicitly Lemma C.1 and pattern matching between the syntax of the terms above and the one of the terms in the lemma to analyse each possible transition.

- $(K \xrightarrow{l\cdot k(p)} K')$
  We have two possible cases, depending in whether $k \in \tilde{k}$ or $k \in \tilde{i}$.
  - $\Theta^{\tilde{k}} \xrightarrow{l\cdot k(p)} \Theta^{\tilde{k}} \,|\, P_k$: then by well-formedness of $\Theta^{\tilde{k}}$ we know that P_k contains no trigger names, and therefore we take

    $$K' \equiv (\nu\tilde{c})(\Theta^{\tilde{k}} \,|\, \langle\!\langle P_k \,|\, P \,|\, \Phi^{\tilde{h}} \rangle\!\rangle \,|\, | \,\langle\!\langle \Theta^{\tilde{j}} \,|\, R \rangle\!\rangle \,|\, \Phi^{\tilde{i}})$$

    By $K_1 \approx_\Delta L_1$ and $K_1 \xrightarrow{l\cdot k(p)} K_1 \,|\, P_k = K_1'$ we get $L_1 \xrightarrow{\tau}\!\!\!\!\twoheadrightarrow \xrightarrow{l\cdot k(p)} \xrightarrow{\tau}\!\!\!\!\twoheadrightarrow \Omega^{\tilde{k}} \,|\, \Omega^{\tilde{j}} \,|\, Q' = L_1'$ with $L_1' \approx_\Delta K_1'$. Since the action $\xrightarrow{l\cdot k(p)}$ is necessarily originated by $\Omega^{\tilde{k}}$, we can reorder the reduction obtaining $L_1 \xrightarrow{l \cdot k(p)} L_1 \mid Q_k \xrightarrow{\tau}\!\!\!\!\rightarrow \xrightarrow{\tau}\!\!\!\!\rightarrow L_1'$, and consequently we have

$$L \xrightarrow{l \cdot k(p)} (\nu\tilde{c})(\Omega^{\tilde{k}} \mid \langle\!\langle Q_k \mid Q \mid \Phi^{\tilde{h}} \rangle\!\rangle \mid \langle\!\langle \Omega^{\tilde{j}} \mid R \rangle\!\rangle \mid \Phi^{\tilde{i}}) = L''$$

Using Lemma C.8, we derive

$$L'' \xrightarrow{\tau}\!\!\!\!\rightarrow (\nu\tilde{c})(\Omega^{\tilde{k}} \mid \langle\!\langle Q' \mid \Phi^{\tilde{h}} \rangle\!\rangle \mid \langle\!\langle \Omega^{\tilde{j}} \mid R \rangle\!\rangle \mid \Phi^{\tilde{i}}) = L'$$

and we conclude because $(K', L') \in \mathcal{R}_\Delta$.

– $\Phi^{\tilde{i}} \xrightarrow{l \cdot k(p)} \Phi^{\tilde{i}} \mid R_k$: this case is similar to the previous one but simpler, because instead of using the hypothesis $K_1 \approx_\Delta L_1$, it suffices to use syntactical reasoning.

- $(K \xrightarrow{l \cdot \mathsf{run}_p} K')$
Similar to the previous case but reasoning on P or R, and exploiting the fact that the process term run_p does not contain trigger names, and therefore can be moved out of the merged term.

- $(K \xrightarrow{l \cdot a(\tilde{v})} K')$
We need to avoid the capture of free session names in $\tilde{v}$ by $\tilde{c}$, and the capture of trigger names in order to bring the output process generated by the transition inside the rightmost merge operator in K. By Observation C.7 we have that for a fresh $\tilde{j}'$, $\langle\!\langle \Theta^{\tilde{j}} \mid R \rangle\!\rangle = \langle\!\langle \Theta^{\tilde{j} \frown \tilde{j}'} \mid R\{\tilde{j}'/\tilde{j}\} \rangle\!\rangle$, and therefore, for $\sigma = \{\tilde{c'}/\tilde{c}\}$, we have

$$K' \equiv (\nu\tilde{c'})(\Theta^{\tilde{k}} \mid \langle\!\langle P\sigma \mid \Phi^{\tilde{h}} \rangle\!\rangle \mid \langle\!\langle \Theta^{\tilde{j} \frown \tilde{j}'} \mid R\sigma\{\tilde{j}'/\tilde{j}\} \mid \overline{l \cdot a}\langle \tilde{v} \rangle \rangle\!\rangle \mid \Phi^{\tilde{i}})$$

where we have alpha-converted $\tilde{c}$ to $\tilde{c'}$ to avoid clashes with free session names in the input action. We can derive in the same way a transition for L:

$$L \xrightarrow{l \cdot a(\tilde{v})} (\nu\tilde{c'})(\Omega^{\tilde{k}} \mid \langle\!\langle Q\sigma \mid \Phi^{\tilde{h}} \rangle\!\rangle \mid \langle\!\langle \Omega^{\tilde{j} \frown \tilde{j}'} \mid R\sigma\{\tilde{j}'/\tilde{j}\} \mid \overline{l \cdot a}\langle \tilde{v} \rangle \rangle\!\rangle \mid \Phi^{\tilde{i}}) = L'$$

By $K_1 \approx_\Delta L_1$, using the variant lemma (Lemma C.6) we obtain $\Theta^{\tilde{k}} \mid \Theta^{\tilde{j}} \mid P\sigma \approx_\Delta \Omega^{\tilde{k}} \mid \Omega^{\tilde{j}} \mid Q\sigma$, and using point (4) of Lemma C.9 we get

$$\Theta^{\tilde{k}} \mid \Theta^{\tilde{j} \frown \tilde{j}'} \mid P\sigma \approx_\Delta \Omega^{\tilde{k}} \mid \Omega^{\tilde{j} \frown \tilde{j}'} \mid Q\sigma$$

and we conclude with $(K', L') \in \mathcal{R}_\Delta$.

- $(K \xrightarrow{(\tilde{b}, \tilde{b'}, \tilde{k}', \tilde{i}')\overline{l \cdot a}\langle \tilde{v} \rangle} K')$
We need to distinguish two cases, depending on whether the output transition is originated by R or P.

– $\langle\!\langle \Theta^{\tilde{j}} \mid R \rangle\!\rangle \xrightarrow{(\tilde{b'}, \tilde{k}', \tilde{i}')\overline{l \cdot a}\langle \tilde{v} \rangle} K_0$, where $\tilde{c} = \tilde{c'}, \tilde{b}$. We have that $\langle\!\langle \Theta^{\tilde{j}} \mid R \rangle\!\rangle = R^{\Theta^{\tilde{j}}} \equiv (\nu\tilde{b'})(\overline{l \cdot a}\langle \tilde{v'}^{\Theta^{\tilde{j}}} \rangle \mid R_1^{\Theta^{\tilde{j}}})$, where $\tilde{b'} \subseteq \tilde{v}$ for $F(\tilde{v'}^{\Theta^{\tilde{j}}}) = (\tilde{v}; \Theta^{\tilde{k}'} \mid \Phi^{\tilde{i}'}; \tilde{k}', \tilde{i}')$, where for each $\langle k' \Leftarrow \Box P_0 \rangle$ in $\Theta^{\tilde{k}'}$ there is $\langle j \Leftarrow \Box P_0 \rangle$ in $\Theta^{\tilde{j}}$, and $K_0 \equiv R_1^{\Theta^{\tilde{j}}} \mid \Theta^{\tilde{k}'} \mid \Phi^{\tilde{i}'}$. We define

$$K' \equiv (\nu\tilde{c'})(\Theta^{\tilde{k}} \mid \Theta^{\tilde{k}'} \mid \langle\!\langle P \mid \Phi^{\tilde{h}} \rangle\!\rangle \mid \langle\!\langle \Theta^{\tilde{j}} \mid R_1 \rangle\!\rangle \mid \Phi^{\tilde{i}} \mid \Phi^{\tilde{i}'})$$

Reasoning symmetrically on L, we obtain

$$\langle\!\langle \Omega^{\tilde{j}} \,|\, R \rangle\!\rangle = R^{\Omega^{\tilde{j}}} \equiv (\nu\tilde{b'})(\overline{l{\cdot}a}\langle \tilde{v'}^{\Omega^{\tilde{j}}}\rangle \,|\, R_1^{\Omega^{\tilde{j}}}) \xrightarrow{(\tilde{b'},\tilde{k'},\tilde{i'})\overline{l\cdot a}\langle\tilde{v}\rangle} R_1^{\Omega^{\tilde{j}}} \,|\, \Omega^{\tilde{k'}} \,|\, \Phi^{\tilde{i'}}$$

for $F(\tilde{v'}^{\Omega^{\tilde{j}}}) = (\tilde{v}; \Omega^{\tilde{k'}} \,|\, \Phi^{\tilde{i'}}; \tilde{k'}, \tilde{i'})$, where for each $\langle k' \Leftarrow \Box Q_0\rangle$ in $\Omega^{\tilde{k'}}$ there is $\langle j \Leftarrow \Box Q_0\rangle$ in $\Omega^{\tilde{j}}$, and we define

$$L' \equiv (\nu\tilde{c'})(\Omega^{\tilde{k}} \,|\, \Omega^{\tilde{k'}} \,|\, \langle\!\langle Q \,|\, \Phi^{\tilde{h}}\rangle\!\rangle \,|\, \langle\!\langle \Omega^{\tilde{j}} \,|\, R_1\rangle\!\rangle \,|\, \Phi^{\tilde{i}} \,|\, \Phi^{\tilde{i'}})$$

From $K_1 \approx_\Delta L_1$, using Lemma C.9, we get

$$\Theta^{\tilde{k}} \,|\, \Theta^{\tilde{k'}} \,|\, \Theta^{\tilde{j}} \,|\, P \approx_\Delta \Omega^{\tilde{k}} \,|\, \Omega^{\tilde{k'}} \,|\, \Omega^{\tilde{j}} \,|\, Q$$

and we conclude with $(K', L') \in \mathcal{R}_\Delta$.

– $\langle\!\langle P \,|\, \Phi^{\tilde{h}}\rangle\!\rangle \xrightarrow{(\tilde{b'},\tilde{k'},\tilde{i'})\overline{l\cdot a}\langle\tilde{v}\rangle} K_0$, where $\tilde{c} = \tilde{c'}, \tilde{b}$. We have that $\langle\!\langle P \,|\, \Phi^{\tilde{h}}\rangle\!\rangle = P^{\Phi^{\tilde{h}}} \equiv (\nu\tilde{b'})(\overline{l{\cdot}a}\langle \tilde{v_1}^{\Phi^{\tilde{h}}}\rangle \,|\, P_1^{\Phi^{\tilde{h}}})$, where $\tilde{b'} \subseteq \tilde{v}$. Moreover if $F(\tilde{v_1}) = (\tilde{v'}; \Theta^{\tilde{k'}}; \tilde{k'})$ then by Lemma C.10, $F(\tilde{v_1}^{\Phi^{\tilde{h}}}) = (\tilde{v}; \Theta^{\tilde{k'}} \,|\, \Phi^{\tilde{i'}}; \tilde{k'}, \tilde{i'})$, and for each $\langle i' \Leftarrow \Box R_0\rangle$ in $\Phi^{\tilde{i'}}$ there is $\langle h \Leftarrow \Box R_0\rangle$ in $\Phi^{\tilde{h}}$. We also have $K_0 \equiv P_1^{\Phi^{\tilde{h}}} \,|\, \Theta^{\tilde{k'}} \,|\, \Phi^{\tilde{i'}}$. We define

$$K' \equiv (\nu\tilde{c'})(\Theta^{\tilde{k}} \,|\, \Theta^{\tilde{k'}} \,|\, \langle\!\langle P_1 \,|\, \Phi^{\tilde{h}}\rangle\!\rangle \,|\, \langle\!\langle \Theta^{\tilde{j}} \,|\, R\rangle\!\rangle \,|\, \Phi^{\tilde{i}} \,|\, \Phi^{\tilde{i'}})$$

By the hypothesis $K_1 \approx_\Delta L_1$ and by

$$K_1 \xrightarrow{(\tilde{b'},\tilde{k'})\overline{l\cdot a}\langle\tilde{v'}\rangle} P_1 \,|\, \Theta^{\tilde{k}} \,|\, \Theta^{\tilde{k'}} \,|\, \Theta^{\tilde{j}} = K_1'$$

where $\tilde{b'} \subseteq \tilde{v'}$ and, as previously stated, $F(\tilde{v_1}) = (\tilde{v'}; \Theta^{\tilde{k'}}; \tilde{k'})$, we get

$$L_1 \xrightarrow{(\tilde{b'},\tilde{k'})\overline{l\cdot a}\langle\tilde{v'}\rangle}\!\!\!\!\rightarrow Q_1 \,|\, \Omega^{\tilde{k}} \,|\, \Omega^{\tilde{k'}} \,|\, \Omega^{\tilde{j}} = L_1'$$

and $K_1' \approx_\Delta L_1'$. Since none of the transitions in the trace above can be generated by a definition, we can deduce

$$Q \xrightarrow{\tau}\!\!\!\!\rightarrow (\nu\tilde{b'})(\overline{l{\cdot}a}\langle\tilde{v_2}\rangle \,|\, Q_2) = Q_3, \quad Q_3 \xrightarrow{(\tilde{b'},\tilde{k'})\overline{l\cdot a}\langle\tilde{v'}\rangle} Q_2 \,|\, \Omega^{\tilde{k'}} \xrightarrow{\tau}\!\!\!\!\rightarrow Q_1 \,|\, \Omega^{\tilde{\tilde{k'}}}$$

where $F(\tilde{v_2}) = (\tilde{v'}; \Omega^{\tilde{k'}}; \tilde{k'})$. Note that by Lemma C.10, $\tilde{v_2}$ has exactly the same occurrences of trigger names in $\tilde{h}$ as $\tilde{v'}$, and therefore $\tilde{v_1}$. By Lemma C.8, $\langle\!\langle Q \,|\, \Phi^{\tilde{h}}\rangle\!\rangle \xrightarrow{\tau}\!\!\!\!\rightarrow \langle\!\langle Q_3 \,|\, \Phi^{\tilde{h}}\rangle\!\rangle$, and by syntactical reasoning

$$\langle\!\langle Q_3 \,|\, \Phi^{\tilde{h}}\rangle\!\rangle \xrightarrow{(\tilde{b'},\tilde{k'},\tilde{i'})\overline{l\cdot a}\langle\tilde{v}\rangle} \langle\!\langle Q_2 \,|\, \Phi^{\tilde{h}}\rangle\!\rangle \,|\, \Omega^{\tilde{k'}} \,|\, \Phi^{\tilde{i'}}$$

for $F(\tilde{v_2}^{\Phi^{\tilde{h}}}) = (\tilde{v}; \Omega^{\tilde{k'}} \,|\, \Phi^{\tilde{i'}}; \tilde{k'}, \tilde{i'})$. Again by Lemma C.8, we have

$$\langle\!\langle Q_2 \,|\, \Phi^{\tilde{h}}\rangle\!\rangle \,|\, \Omega^{\tilde{k'}} \,|\, \Phi^{\tilde{i'}} \xrightarrow{\tau}\!\!\!\!\rightarrow \langle\!\langle Q_1 \,|\, \Phi^{\tilde{h}}\rangle\!\rangle \,|\, \Omega^{\tilde{k'}} \,|\, \Phi^{\tilde{i'}}$$

and defining

$$L' \equiv (\nu\tilde{c'})(\Omega^{\tilde{k}} \,|\, \Omega^{\tilde{k'}} \,|\, \langle\!\langle Q_1 \,|\, \Phi^{\tilde{h}}\rangle\!\rangle \,|\, \langle\!\langle \Omega^{\tilde{j}} \,|\, R\rangle\!\rangle \,|\, \Phi^{\tilde{i}} \,|\, \Phi^{\tilde{i'}})$$

we conclude with $(K', L') \in \mathcal{R}_\Delta$.

- $(K \xrightarrow{(\tilde{k})l\cdot\mathsf{update}_p(\tilde{U},(\chi)\tilde{V})} K')$
  Follows combining the argument for input and the one for output.
- $(K \xrightarrow{l\cdot\tau} K')$
  We have three possible cases, depending on wether the transition is determined by R, P, or both R and P.
  - $\langle\!\langle \Theta^{\tilde{j}} \,|\, R \rangle\!\rangle \xrightarrow{\tau} K_0$: we necessarily have $R \xrightarrow{\tau} R'$, and by Lemma C.8 we conclude, since $K_0 \equiv \langle\!\langle \Theta^{\tilde{j}} \,|\, R' \rangle\!\rangle$ and we can do the same transition with L.
  - $\langle\!\langle P \,|\, \Phi^{\tilde{h}} \rangle\!\rangle \xrightarrow{\tau} K_0$: similar to the previous point, using the hypothesis $K_1 \approx_\Delta L_1$ to justify the (weak) transition for L.
  - $\langle\!\langle P \,|\, \Phi^{\tilde{h}} \rangle\!\rangle \,|\, \langle\!\langle \Theta^{\tilde{j}} \,|\, R \rangle\!\rangle \xrightarrow{\tau} K_0$ We must distinguish four cases depending on whether R or P receive the value, and if the value is received on a replicated input.
    * Replicated input by R: $R^{\Theta^{\tilde{j}}} \equiv (\nu\tilde{b})(R_1^{\Theta^{\tilde{j}}} \,|\, !l\cdot a(\tilde{x}).R_2^{\Theta^{\tilde{j}}})$, where $\tilde{b}$ is fresh, and $P^{\Phi^{\tilde{h}}} \equiv (\nu\tilde{c'})(\overline{l\cdot a}\langle \tilde{v}^{\Phi^{\tilde{h}}} \rangle \,|\, P_1^{\Phi^{\tilde{h}}})$, where $\tilde{c'}$ is fresh and $\tilde{c'} \subseteq \tilde{v}^{\Phi^{\tilde{h}}}$, and $K_0 \equiv (\nu\tilde{c'})(P_1^{\Phi^{\tilde{h}}} \,|\, (\nu\tilde{b})(R_1^{\Theta^{\tilde{j}}} \,|\, !l\cdot a(\tilde{x}).R_2^{\Theta^{\tilde{j}}} \,|\, R_2^{\Theta^{\tilde{j}}}\{\tilde{v}^{\Phi^{\tilde{h}}}/\tilde{x}\}))$. Suppose $F(\tilde{v}) = (\tilde{v'}; \Theta^{\tilde{k'}}; \tilde{k'})$ for some fresh $\tilde{k'}$. By Lemma C.10 we have $\tilde{v} = \tilde{v'}^{\Theta^{\tilde{k'}}}$, and we also know that $\tilde{v}$ and $\tilde{v'}$ have the same occurrences of triggers in $\tilde{h}$. We can therefore rewrite $R_2^{\Theta^{\tilde{j}}}\{\tilde{v}^{\Phi^{\tilde{h}}}/\tilde{x}\} = (R_2\{\tilde{v'}^{\Phi^{\tilde{h}}}/\tilde{x}\})^{\Theta^{\tilde{j}},\Theta^{\tilde{k'}}}$, and define

$$K' \equiv (\nu\tilde{c},\tilde{c'})(\Theta^{\tilde{k}} \,|\, \langle\!\langle P_1 \,|\, \Phi^{\tilde{h}} \rangle\!\rangle \,|\, \langle\!\langle \Theta^{\tilde{j}} \,|\, \Theta^{\tilde{k'}} \,|\, (\nu\tilde{b})(R_1 \,|\, !l\cdot a(\tilde{x}).R_2 \,|\, R_2\{\tilde{v'}^{\Phi^{\tilde{h}}}/\tilde{x}\}) \rangle\!\rangle \,|\, \Phi^{\tilde{i}})$$

By the hypothesis $K_1 \approx_\Lambda L_1$ and by

$$K_1 \xrightarrow{(\tilde{c'},\tilde{k'})\overline{l\cdot a}\langle\tilde{v'}\rangle} P_1 \,|\, \Theta^{\tilde{j}} \,|\, \Theta^{\tilde{k}} \,|\, \Theta^{\tilde{k'}} = K_1'$$

where, as stated above, $F(\tilde{v}) = (\tilde{v'}; \Theta^{\tilde{k'}}; \tilde{k'})$, we get

$$L_1 \overset{\tau}{\twoheadrightarrow} (\tilde{c'})(Q_1 \,|\, \overline{l\cdot a}\langle\tilde{v_2}\rangle) \,|\, \Omega^{\tilde{j}} \,|\, \Omega^{\tilde{k}} = L_2, \quad L_2 \xrightarrow{(\tilde{c'},\tilde{k'})\overline{l\cdot a}\langle\tilde{v'}\rangle} Q_1 \,|\, \Omega^{\tilde{j}} \,|\, \Omega^{\tilde{k}} \,|\, \Omega^{\tilde{k'}} = L_3$$

where $F(\tilde{v_2}) = (\tilde{v'}; \Omega^{\tilde{k'}}; \tilde{k'})$, and

$$L_3 \overset{\tau}{\twoheadrightarrow} Q' \,|\, \Omega^{\tilde{j}} \,|\, \Omega^{\tilde{k}} \,|\, \Omega^{\tilde{k'}} = L_1', \quad K_1' \approx_\Lambda L_1'$$

By Lemma C.8,

$$\langle\!\langle Q \,|\, \Phi^{\tilde{h}} \rangle\!\rangle \overset{\tau}{\twoheadrightarrow} \langle\!\langle (\tilde{c'})(Q_1 \,|\, \overline{l\cdot a}\langle\tilde{v_2}\rangle) \,|\, \Phi^{\tilde{h}} \rangle\!\rangle = (\tilde{c'})(Q_1^{\Phi^{\tilde{h}}} \,|\, \overline{l\cdot a}\langle\tilde{v_2}^{\Phi^{\tilde{h}}}\rangle)$$

and therefore, using syntactical reasoning $L \overset{\tau}{\twoheadrightarrow} \xrightarrow{\tau} L''$, where

$$L'' = (\nu\tilde{c},\tilde{c'})(\Omega^{\tilde{k}} \,|\, \langle\!\langle Q_1 \,|\, \Phi^{\tilde{h}} \rangle\!\rangle \,|\, \langle\!\langle \Omega^{\tilde{j}} \,|\, \Omega^{\tilde{k'}} \,|\, (\nu\tilde{b})(R_1 \,|\, !l\cdot a(\tilde{x}).R_2 \,|\, R_2\{\tilde{v'}^{\Phi^{\tilde{h}}}/\tilde{x}\}) \rangle\!\rangle \,|\, \Phi^{\tilde{i}})$$

and, again by Lemma C.8, $L'' \overset{\tau}{\twoheadrightarrow} L'$, for

$$L' \equiv (\nu\tilde{c},\tilde{c'})(\Omega^{\tilde{k}} \,|\, \langle\!\langle Q' \,|\, \Phi^{\tilde{h}} \rangle\!\rangle \,|\, \langle\!\langle \Omega^{\tilde{j}} \,|\, \Omega^{\tilde{k'}} \,|\, (\nu\tilde{b})(R_1 \,|\, !l\cdot a(\tilde{x}).R_2 \,|\, R_2\{\tilde{v'}^{\Phi^{\tilde{h}}}/\tilde{x}\}) \rangle\!\rangle \,|\, \Phi^{\tilde{i}})$$

and we conclude with $(K', L') \in \mathcal{R}_\Delta$.

* Input by R: analogous to the previous case.
* Input by P: $R^{\Theta^{\tilde{j}}} \equiv (\nu\tilde{c'})(R_1^{\Theta^{\tilde{j}}} \,|\, \overline{l \cdot a}\langle \tilde{v}^{\Theta^{\tilde{j}}}\rangle)$, where $\tilde{c'}$ is fresh and $\tilde{c'} \subseteq \tilde{v}^{\Theta^{\tilde{j}}}$, and $P^{\Phi^{\tilde{h}}} \equiv (\nu\tilde{b})(P_1^{\Phi^{\tilde{h}}} \,|\, l\cdot a(\tilde{x}).P_2^{\Phi^{\tilde{h}}})$, where $\tilde{b}$ is fresh, and $K_0 \equiv (\nu\tilde{c'})((\nu\tilde{b})(P_1^{\Phi^{\tilde{h}}} \,|\, P_2^{\Phi^{\tilde{h}}}\{\tilde{v}^{\Theta^{\tilde{j}}}/\tilde{x}\}) \,|\, R_1^{\Theta^{\tilde{j}}})$. For some vector of fresh triggers $\tilde{j'}$, let $\tilde{v'} = \tilde{v}\{\tilde{j'}/\tilde{j}\}$. We have that $\tilde{v}^{\Theta^{\tilde{j}}} = \tilde{v'}^{\Theta^{\tilde{j} \frown \tilde{j'}}}$. Suppose $F(\tilde{v'}) = (\tilde{v_1}; \Phi^{\tilde{h'}}; \tilde{h'})$ for some fresh $\tilde{h'}$. By Lemma C.10 and the freshness of $\tilde{j'}, \tilde{h'}$ we have $\tilde{v'} = \tilde{v_1}^{\Phi^{\tilde{h'}}}$ and $P_2^{\Phi^{\tilde{h}}}\{\tilde{v}^{\Theta^{\tilde{j}}}/\tilde{x}\} = \langle\!\langle P_2\{\tilde{v'}/\tilde{x}\}\rangle\!\rangle \,|\, \Theta^{\tilde{j} \frown \tilde{j'}}{}^{\Phi^{\tilde{h'}}, \Phi^{\tilde{h}}}$, and we can define

$$K' \equiv (\nu\tilde{c}, \tilde{c'})(\Theta^{\tilde{k}} \,|\, \langle\!\langle\, \langle\!\langle (\nu\tilde{b})(P_1 \,|\, P_2\{\tilde{v_1}/\tilde{x}\} \,|\, \Theta^{\tilde{j} \frown \tilde{j'}})\rangle\!\rangle \,|\, \Phi^{\tilde{h}} \,|\, \Phi^{\tilde{h'}}\rangle\!\rangle \,|\, \langle\!\langle \Theta^{\tilde{j}} \,|\, R_1\rangle\!\rangle \,|\, \Phi^{\tilde{i}})$$

By the hypothesis $K_1 \approx_\Lambda L_1$, and by

$$K_1 \xrightarrow{l \cdot a(\tilde{v_1})}\xrightarrow{\tau} (\nu\tilde{b})(P_1 \,|\, P_2\{\tilde{v_1}/\tilde{x}\}) \,|\, \Theta^{\tilde{j}} \,|\, \Theta^{\tilde{k}} = K_1'$$

we get, composing two weak actions of L_1,

$$L_1 \xrightarrow{l \cdot a(\tilde{v_1})}\!\!\!\!\rightarrow Q' \,|\, \Omega^{\tilde{j}} \,|\, \Omega^{\tilde{k}} = L_1', \quad K_1' \approx_\Lambda L_1'$$

From the transitions above, by syntactical reasoning, we infer that

$$Q \xrightarrow{\tau}\!\!\!\!\rightarrow (\nu\tilde{d})(Q_1 \,|\, l\cdot a(x).Q_2) \xrightarrow{l \cdot a(\tilde{v_1})} (\nu\tilde{d})(Q_1 \,|\, l\cdot a(x).Q_2 \,|\, \overline{l \cdot a}\langle\tilde{v_1}\rangle) \xrightarrow{\tau}\!\!\!\!\rightarrow Q'$$

for some vector of fresh session names $\tilde{d}$. By Lemma C.8 and by syntactical reasoning we get

$$L \xrightarrow{\tau}\!\!\!\!\rightarrow (\nu\tilde{c})(\Omega^{\tilde{k}} \,|\, \langle\!\langle (\nu\tilde{d})(Q_1 \,|\, l\cdot a(x).Q_2) \,|\, \Phi^{\tilde{h}}\rangle\!\rangle \,|\, \langle\!\langle \Theta^{\tilde{j}} \,|\, R\rangle\!\rangle \,|\, \Phi^{\tilde{i}}) = L''$$

Now we use structural congruence, and the equations $\tilde{v}^{\Omega^{\tilde{j}}} = \tilde{v'}^{\Omega^{\tilde{j} \frown \tilde{j'}}}$ and $\tilde{v'} = \tilde{v_1}^{\Phi^{\tilde{h'}}}$, to bring the output of R near to the input.

$$L'' \equiv (\nu\tilde{c}, \tilde{c'})(\Omega^{\tilde{k}} \,|\, \langle\!\langle\, \langle\!\langle (\nu\tilde{d})(Q_1 \,|\, l\cdot a(x).Q_2 \,|\, \overline{l \cdot a}\langle\tilde{v_1}\rangle) \,|\, \Omega^{\tilde{j'}}\rangle\!\rangle \,|\, \Phi^{\tilde{h}} \,|\, \Phi^{\tilde{h'}}\rangle\!\rangle \,|\, \langle\!\langle \Omega^{\tilde{j}} \,|\, R_1\rangle\!\rangle \,|\, \Phi^{\tilde{i}})$$

By another application of Lemma C.8 we deduce

$$L'' \xrightarrow{\tau}\!\!\!\!\rightarrow (\nu\tilde{c}, \tilde{c'})(\Omega^{\tilde{k}} \,|\, \langle\!\langle\, \langle\!\langle Q' \,|\, \Omega^{\tilde{j'}}\rangle\!\rangle \,|\, \Phi^{\tilde{h}} \,|\, \Phi^{\tilde{h'}}\rangle\!\rangle \,|\, \langle\!\langle \Omega^{\tilde{j}} \,|\, R_1\rangle\!\rangle \,|\, \Phi^{\tilde{i}}) = L'$$

Applying point (4) of Lemma C.9 on $K_1' \approx_\Lambda L_1'$ we get $K_1' \,|\, \Theta^{\tilde{j} \frown \tilde{j'}} \approx_\Lambda L_1' \,|\, \Omega^{\tilde{j} \frown \tilde{j'}}$, and by Lemma C.11 we get

$$\langle\!\langle (\nu\tilde{b})(P_1 \,|\, P_2\{\tilde{v_1}/\tilde{x}\}) \,|\, \Theta^{\tilde{j} \frown \tilde{j'}}\rangle\!\rangle \,|\, \Theta^{\tilde{j}} \,|\, \Theta^{\tilde{k}} \approx_\Lambda \langle\!\langle Q' \,|\, \Omega^{\tilde{j} \frown \tilde{j'}}\rangle\!\rangle \,|\, \Omega^{\tilde{k}} \,|\, \Omega^{\tilde{j}}$$

and we conclude because $(K', L') \in \mathcal{R}_\Delta$.
* Replicated input by P: analogous to the previous case.

We now show that the restriction of $\approx_\Lambda$ to processes is a congruence with respect to generic contexts. We need to show the remaining points below:

1. $P \approx_\Lambda Q \Longrightarrow l{\cdot}c(\tilde{x}).P \approx_\Lambda l{\cdot}c(\tilde{x}).Q$;
2. $P \approx_\Lambda Q \Longrightarrow !l{\cdot}c(\tilde{x}).P \approx_\Lambda !l{\cdot}c(\tilde{x}).Q$;
3. $P \approx_\Lambda Q \Longrightarrow l{\cdot}\mathsf{update}_p(\chi, V).P \approx_\Lambda l{\cdot}\mathsf{update}_p(\chi, V).Q$.

Point (1) follows by showing that $\mathcal{R} = \big\{(l{\cdot}c(\tilde{x}).P \,|\, M, l{\cdot}c(\tilde{x}).Q \,|\, M) \;\big|\; P \approx_\Delta Q\big\}_{\Lambda \subseteq \Delta}$, where M is an arbitrary parallel composition of first-order outputs, is a domain bisimulation. Points (2) and (3) are similar, using also transitivity and closure under parallel composition of $\approx_\Lambda$. $\Box$

D Soundness of $\approx_\Lambda$

The main task in proving the soundness of $\approx_\Lambda$ consists in showing that an auxiliary relation $\asymp$ on networks whose processes, in parallel with the ones extracted from the data, are Λ-bisimilar, is included in barbed equivalence.

We start with some auxiliary lemmas.

D.1 Auxiliary lemmas

Lemma D.1 *(i) If $K \xrightarrow{l\cdot\tau} K'$ then, for all D such that $l \in dom(D)$, $(D, \langle\!\langle K \rangle\!\rangle) \rightarrow (D, \langle\!\langle K' \rangle\!\rangle)$. (ii) If $K \xrightarrow{\tau}\!\!\!\rightarrow_\Lambda K'$ then, for all D such that $\Lambda \subseteq dom(D)$, $(D, \langle\!\langle K \rangle\!\rangle) \rightarrow^* (D, \langle\!\langle K' \rangle\!\rangle)$.*

Proof. Point (i) follows easily from Lemma C.8, point (ii) follows directly from (i). $\Box$

Lemma D.2 *Suppose $F(D) = (D'; \Theta^{\tilde{k}}; \tilde{k})$ and $D \equiv \{l \mapsto T\} \uplus E$.*
If $p(T) \rightsquigarrow_{p,l,\Box X.\Box X} T, \Sigma$, where $range(\Sigma) = \{\Box P_1, \dots, \Box P_n\}$, then:

1. *$\Theta^{\tilde{k}} \equiv \Theta^{\tilde{h}} \,|\, \langle j_1 \Leftarrow \Box R_1\rangle \,|\, \dots \,|\, \langle j_n \Leftarrow \Box R_n\rangle$ and $R_i\{l/\circlearrowleft\}\{p/.\} = P_i$.*
2. *if also $F(B) = (B'; \Omega^{\tilde{k}}; \tilde{k})$ with $B' \equiv D'$ then $B \equiv \{l \mapsto S\} \uplus E'$, and $p(S) \rightsquigarrow_{p,l,\Box X,\Box X} S, \Sigma'$ where $range(\Sigma') = \{\Box Q_1, \dots, \Box Q_n\}$*

Proof. By well-founded induction on the depth of nesting of selected nodes in $p(T)$, following a structure similar to the proof of Lemma A.13 of [8], and by definition of F. $\Box$

Lemma D.3 *Suppose $F(D) = (D'; \Theta^{\tilde{k}}; \tilde{k})$ where $D \equiv \{l \mapsto T\} \uplus E$, and $F(B) = (B'; \Omega^{\tilde{k}}; \tilde{k})$ with $B' \equiv D'$. If $p(T) \rightsquigarrow_{p,l,\chi,V} T_1, \Sigma$, where $range(\Sigma) = \tilde{U}$, then:*

1. *$\Theta^{\tilde{k}} \equiv \Theta^{\tilde{h}} \,|\, \Theta^{\tilde{i}}$ where $F(\tilde{U}) = (\tilde{U}'; \Theta^{\tilde{i}}; \tilde{i})$ and $F(V_1, \dots V_n) = (\tilde{V}'; \Theta^{\tilde{j}}; \tilde{j})$ with each $V_i = V$;*
2. *$F(D_1) = (D_1'; \Theta^{\tilde{h}} \,|\, \Theta^{\tilde{j}}; \tilde{h}, \tilde{j})$ where $D_1 \equiv \{l \mapsto T_1\} \uplus E$;*
3. *$B \equiv \{l \mapsto S\} \uplus E'$;*

4. *for any W such that $F(W_1, \ldots W_n) = (\tilde{V}'; \Omega^{\tilde{j}}; \tilde{j})$ with each $W_i = W$, there are S_1, Σ' such that $p(S) \rightsquigarrow_{p,l,\chi,W} S_1, \Sigma'$ where $range(\Sigma') = \{W'_1, \ldots, W'_n\}$ and $F(\tilde{W}') = (\tilde{U}'; \Omega^{\tilde{i}}; \tilde{i})$;*
5. *$\Omega^{\tilde{k}} \equiv \Omega^{\tilde{h}} \mid \Omega^{\tilde{i}}$;*
6. *$F(B_1) = (B'_1; \Omega^{\tilde{h}} \mid \Omega^{\tilde{j}}; \tilde{h}, \tilde{j})$ where $B_1 \equiv \{l \mapsto S_1\} \uplus E'$;*
7. *$D'_1 \equiv B'_1$.*

Proof. By well-founded induction on the depth of nesting of selected nodes in $p(T)$, following a structure similar to the proof of Lemma A.13 of [8], and by definition of F. $\square$

Lemma D.4 *If $(D, P) \to (D_1, P_1)$ then one of the following holds:*

1. *$P \xrightarrow{l \cdot \tau} P_1$, $D \equiv \{l \mapsto T\} \uplus E$, and $D = D_1$.*
2. *$P \xrightarrow{l \cdot \mathsf{run}_p} P_2$, $D \equiv \{l \mapsto T\} \uplus E$, and $D = D_1$, where $p(T) \rightsquigarrow_{p,l,\Box X,\Box X} T, \Sigma$ with $range(\Sigma) = \{\Box P'_1, \cdots, \Box P'_n\}$, and $P_1 \equiv P_2 \mid P'_1 \mid \cdots \mid P'_n$.*
3. *$P \xrightarrow{(\tilde{j}) l \cdot \mathsf{update}_p(\tilde{U}', (\chi)\tilde{V}')} P_2 \mid \Theta^{\tilde{j}}$, $D \equiv \{l \mapsto T\} \uplus E$, and $D_1 \equiv \{l \mapsto T_1\} \uplus E$, where $p(T) \rightsquigarrow_{p,l,\chi,V} T_1, \Sigma$ for $range(\Sigma) = \tilde{U}$ and $F(\tilde{U}) = (\tilde{U}'; \Theta^{\tilde{i}}; \tilde{i})$, $F(V_1, \ldots, V_n) = (\tilde{V}'; \Theta^{\tilde{j}}; \tilde{j})$ where each $V_i = V$, and $P_1 \equiv \langle\!\langle P_2 \mid \Theta^{\tilde{i}} \rangle\!\rangle$.*

Proof. By induction on the derivation of $(D, P) \to (D_1, P_1)$, and by definition of lts. $\square$

D.2 The candidate relation $\asymp$

Definition D.1 *The relation $\asymp$ is composed by the pairs $((D, P), (B, Q))$ such that*

$$F(D) = (D'; \Theta^{\tilde{k}}; \tilde{k}) \wedge F(B) = (B'; \Omega^{\tilde{k}}; \tilde{k}) \wedge D' \equiv B' \wedge P \mid \Theta^{\tilde{k}} \approx_\Lambda Q \mid \Omega^{\tilde{k}}$$

where $\Lambda = dom(B) = dom(D)$.

Lemma D.5 *The relation $\asymp$ is included in $\simeq^b$.*

Proof. We show that $\asymp$ is (1) barb preserving, (2) contextual, and (3) reduction-closed.

1. Follows from the definition of barbs, the hypothesis $P \mid \Theta^{\tilde{k}} \approx_\Lambda Q \mid \Omega^{\tilde{k}}$ and Lemma D.1, noticing that $\Theta^{\tilde{k}}$ and $\Omega^{\tilde{k}}$ cannot perform tau or update transitions.
2. Consider a generic reduction context $(E \uplus -, C\{-\})$: we have, by definition of F, that $F(E \uplus D) = (E' \uplus D'; \Phi^{\tilde{j}} \mid \Theta^{\tilde{k}}; \tilde{j}, \tilde{k})$ and $F(E \uplus B) = (E' \uplus B'; \Phi^{\tilde{j}} \mid \Omega^{\tilde{k}}; \tilde{j}, \tilde{k})$, where $E' \uplus D' \equiv E' \uplus B'$. We must show that $P \mid \Theta^{\tilde{k}} \approx_\Lambda Q \mid \Omega^{\tilde{k}}$ implies $C\{P\} \mid \Phi^{\tilde{j}} \mid \Theta^{\tilde{k}} \approx_\Lambda C\{Q\} \mid \Phi^{\tilde{j}} \mid \Omega^{\tilde{k}}$, which follows by Theorem 4.1, noticing that $C\{P\} \mid \Phi^{\tilde{j}} \mid \Theta^{\tilde{k}} \equiv C\{P \mid \Theta^{\tilde{k}} \mid \Phi^{\tilde{j}}\}$ and $C\{Q\} \mid \Phi^{\tilde{j}} \mid \Omega^{\tilde{k}} \equiv C\{Q \mid \Omega^{\tilde{k}} \mid \Phi^{\tilde{j}}\}$, since scripted processes have no free session names.

3. Assuming that $(D, P) \asymp (B, Q)$ and $(D, P) \to (D_1, P_1)$, we need to show that $(B, Q) \to^* (B_1, Q_1) \asymp (D_1, P_1)$. For clarity, we write down what $(D, P) \asymp (B, Q)$ means:

$$F(D) = (D'; \Theta^{\tilde{k}}; \tilde{k}) \tag{1}$$

$$F(B) = (B'; \Omega^{\tilde{k}}; \tilde{k}) \tag{2}$$

$$D' \equiv B' \tag{3}$$

$$\Lambda = dom(D) = dom(B) \tag{4}$$

$$P \mid \Theta^{\tilde{k}} \approx_\Lambda Q \mid \Omega^{\tilde{k}} \tag{5}$$

By Lemma D.4, there are three cases:

(a) $P \xrightarrow{l\cdot\tau} P_1$ with

$$D \equiv \{l \mapsto T\} \uplus E \tag{6}$$

$$D = D_1 \tag{7}$$

By definition of lts and by (5), we have

$$\begin{array}{ccc} P \mid \Theta^{\tilde{k}} & \xrightarrow{l\cdot\tau} & P_1 \mid \Theta^{\tilde{k}} \\ \approx_\Lambda \big\downarrow & & \big\uparrow \approx_\Lambda \\ Q \mid \Omega^{\tilde{k}} \xrightarrow{\tau}\!\!\!\!\rightarrow_\Lambda & \xrightarrow{l\cdot\tau} & \xrightarrow{\tau}\!\!\!\!\rightarrow_\Lambda Q_1 \mid \Omega^{\tilde{k}} \end{array}$$

By syntactical reasoning we get $Q \xrightarrow{l\cdot\tau}\!\!\!\!\rightarrow_\Lambda Q_1$, by Lemma D.1 we get $(B, Q) \to^* (B, Q_1)$, and by (2), (3) and (7) we conclude with

$$\begin{array}{ccc} (D, P) & \longrightarrow & (D, P_1) \\ \asymp \big\downarrow & & \big\uparrow \asymp \\ (B, Q) & \xrightarrow{*} & (B, Q_1) \end{array}$$

(b) $P \xrightarrow{l\cdot\mathsf{run}_p} P_2$, with (6), (7), and

$$p(T) \rightsquigarrow\!\!\!|_{\,p,l,\Box X,\Box X} T, \Sigma \tag{8}$$

$$range(\Sigma) = \{\Box P_1', \cdots, \Box P_n'\} \tag{9}$$

and $P_1 \equiv P_2 \mid P_1' \mid \cdots \mid P_n'$. By definition of lts and by (5), we have

$$\begin{array}{ccc} P \mid \Theta^{\tilde{k}} & \xrightarrow{l\cdot\mathsf{run}_p} & P_2 \mid \Theta^{\tilde{k}} \\ \approx_\Lambda \big\downarrow & & \big\uparrow \approx_\Lambda \\ Q \mid \Omega^{\tilde{k}} \xrightarrow{\tau}\!\!\!\!\rightarrow_\Lambda & \xrightarrow{l\cdot\mathsf{run}_p} & \xrightarrow{\tau}\!\!\!\!\rightarrow_\Lambda Q_2 \mid \Omega^{\tilde{k}} \end{array}$$

and by syntactical reasoning we get $Q \xrightarrow{\tau}\!\!\!\!\rightarrow_\Lambda Q_3 \mid l\cdot\mathsf{run}_p \xrightarrow{l\cdot\mathsf{run}_p} Q_3 \xrightarrow{\tau}\!\!\!\!\rightarrow_\Lambda Q_2$. By Lemma D.1, we get

$$(B, Q) \to^* (B, Q_3 \mid l\cdot\mathsf{run}_p) \tag{10}$$

By (1), (2), (3), (6),(8),(9), and point (2) of Lemma D.2, we have

$$B \equiv \{l \mapsto S\} \uplus E' \tag{11}$$

$$p(S) \rightsquigarrow_{p,l,\Box X,\Box X} S, \Sigma' \tag{12}$$

$$range(\Sigma') = \{\Box Q'_1, \ldots, \Box Q'_n\} \tag{13}$$

and therefore by definition of reduction we can derive

$$(B, Q_3 \,|\, l{\cdot}\mathsf{run}_p) \rightarrow (B, Q_3 \,|\, Q') \tag{14}$$

for $Q' \equiv Q'_1 \,|\, \ldots \,|\, Q'_n$, and again by Lemma D.1 and definition of reduction we have

$$(B, Q_3 \,|\, Q') \rightarrow^* (B, Q_2 \,|\, Q') \tag{15}$$

Composing (10), (14), and (15) we get

$$(B, Q) \rightarrow^* (B, Q_2 \,|\, Q') \tag{16}$$

We need to show that for some Q_1 such that $(B, Q_2 \,|\, Q') \rightarrow^* (B, Q_1)$, the equation $P_1 \,|\, \Theta^{\tilde{k}} \approx_\Lambda Q_1 \,|\, \Omega^{\tilde{k}}$ holds.

By (1), (6), (8), and point (1) of Lemma D.2, we have $\Theta^{\tilde{k}} \equiv \Theta^{\tilde{h}} \,|\, \langle j_1 \Leftarrow \Box R_1\rangle \,|\, \ldots \,|\, \langle j_n \Leftarrow \Box R_n\rangle$ and $R_i\{l/\circlearrowleft\}\{p/.\} = P'_i$. By (2), (11), (12), and again point (1) of Lemma D.2, we get $\Omega^{\tilde{k}} \equiv \Omega^{\tilde{h}} \,|\, \langle j_1 \Leftarrow \Box R'_1\rangle \,|\, \ldots \,|\, \langle j_n \Leftarrow \Box R'_n\rangle$ and $R'_i\{l/\circlearrowleft\}\{p/.\} = Q'_i$. We can now derive

$$\begin{array}{ccc}
P_2 \,|\, \Theta^{\tilde{k}} & \xrightarrow{l\cdot j_1(p)} & P_2 \,|\, P'_1 \,|\, \Theta^{\tilde{k}} \\
\approx_\Lambda \Big\downarrow & & \Big\uparrow \approx_\Lambda \\
Q_2 \,|\, \Omega^{\tilde{k}} \xrightarrow{\tau}\!\!\!\!\rightarrow_\Lambda & \xrightarrow{l\cdot j_1(p)} & \xrightarrow{\tau}\!\!\!\!\rightarrow_\Lambda Q^{n-1} \,|\, \Omega^{\tilde{k}} \\
 & \vdots & \\
P_2 \,|\, P'_1 \,|\, \ldots \,|\, P'_{n-1} \,|\, \Theta^{\tilde{k}} & \xrightarrow{l\cdot j_n(p)} & P_1 \,|\, \Theta^{\tilde{k}} \\
\approx_\Lambda \Big\downarrow & & \Big\uparrow \approx_\Lambda \\
Q^1 \,|\, \Omega^{\tilde{k}} \xrightarrow{\tau}\!\!\!\!\rightarrow_\Lambda & \xrightarrow{l\cdot j_n(p)} & \xrightarrow{\tau}\!\!\!\!\rightarrow_\Lambda Q_1 \,|\, \Omega^{\tilde{k}}
\end{array}$$

where by syntactic reasoning

$$Q_2 \,|\, Q'_1 \,|\, \ldots \,|\, Q'_n \xrightarrow{\tau}\!\!\!\!\rightarrow_\Lambda Q^{n-1} \,|\, Q'_2 \,|\, \ldots \,|\, Q'_n \xrightarrow{\tau}\!\!\!\!\rightarrow_\Lambda \cdots \xrightarrow{\tau}\!\!\!\!\rightarrow_\Lambda Q^1 \,|\, Q'_n \xrightarrow{\tau}\!\!\!\!\rightarrow_\Lambda Q_1$$

and, by (16) and Lemma D.1, we conclude with

$$\begin{array}{ccc}
(D, P) & \longrightarrow & (D, P_1) \\
\asymp \Big\downarrow & & \Big\uparrow \asymp \\
(B, Q) & \xrightarrow{\;*\;} & (B, Q_1)
\end{array}$$

(c) $P \xrightarrow{(\tilde{j})l\cdot\mathsf{update}_p(\tilde{U'},(\chi)\tilde{V'})} P_2 \,|\, \Theta^{\tilde{j}}$, with (6) and

$$D_1 \equiv \{l \mapsto T_1\} \uplus E \tag{17}$$

$$p(T) \rightsquigarrow_{p,l,\chi,V} T_1, \Sigma,\ range(\Sigma) = \tilde{U} \tag{18}$$

$$F(\tilde{U}) = (\tilde{U'}; \Theta^{\tilde{i}}; \tilde{i}) \tag{19}$$

$$F(V_1, \ldots, V_n) = (\tilde{V'}; \Theta^{\tilde{j}}; \tilde{j}),\ V_i = V \tag{20}$$

$$P_1 \equiv \langle\!\langle P_2 \,|\, \Theta^{\tilde{i}} \rangle\!\rangle \tag{21}$$

By definition of lts and by (5), we have

$$\begin{array}{ccc} P \,|\, \Theta^{\tilde{k}} & \xrightarrow{(\tilde{j})l\cdot\mathsf{update}_p(\tilde{U'},(\chi)\tilde{V'})} & P_2 \,|\, \Theta^{\tilde{j}} \,|\, \Theta^{\tilde{k}} \\ \approx_\Lambda \big\downarrow & & \big\uparrow \approx_\Lambda \\ Q \,|\, \Omega^{\tilde{k}} \xrightarrow{\tau}\!\!\!\!\to_\Lambda & \xrightarrow{(\tilde{j})l\cdot\mathsf{update}_p(\tilde{U'},(\chi)\tilde{V'})} & \xrightarrow{\tau}\!\!\!\!\to_\Lambda Q_2 \,|\, \Omega^{\tilde{j}} \,|\, \Omega^{\tilde{k}} \end{array} \tag{22}$$

and by syntactical reasoning

$$Q \xrightarrow{\tau}\!\!\!\!\to_\Lambda (\nu\tilde{b})(Q_3 \,|\, l\cdot\mathsf{update}_p(\chi, W).Q_4) = Q_M \tag{23}$$

$$Q_M \xrightarrow{(\tilde{j})l\cdot\mathsf{update}_p(\tilde{U'},(\chi)\tilde{V'})} (\nu\tilde{b})(Q_3 \,|\, Q_5) \,|\, \Omega^{\tilde{j}} \tag{24}$$

$$(\nu\tilde{b})(Q_3 \,|\, Q_5) \,|\, \Omega^{\tilde{j}} \xrightarrow{\tau}\!\!\!\!\to_\Lambda Q_2 \,|\, \Omega^{\tilde{j}} \tag{25}$$

for some W is such that

$$F(W_1, \ldots, W_n) = (\tilde{V'}; \Omega^{\tilde{j}}; \tilde{j}),\ W_i = W \tag{26}$$

By (23) and Lemma D.1, we get $(B, Q) \rightarrow^* (B, Q_M)$.
By (1), (2), (3), (6), (18), and Lemma D.3, we obtain

$$\Theta^{\tilde{k}} \equiv \Theta^{\tilde{h}} \,|\, \Theta^{\tilde{i}} \tag{27}$$

$$F(D_1) = (D_1'; \Theta^{\tilde{h}} \,|\, \Theta^{\tilde{j}}; \tilde{h}, \tilde{j}) \text{ where } D_1 \equiv \{l \mapsto T_1\} \uplus E \tag{28}$$

$$B \equiv \{l \mapsto S\} \uplus E' \tag{29}$$

$$p(S) \rightsquigarrow_{p,l,\chi,W} S_1, \Sigma',\ range(\Sigma') = \{W_1', \ldots, W_n'\} \tag{30}$$

$$F(\tilde{W'}) = (\tilde{U'}; \Omega^{\tilde{i}}; \tilde{i}) \tag{31}$$

$$\Omega^{\tilde{k}} \equiv \Omega^{\tilde{h}} \,|\, \Omega^{\tilde{i}} \tag{32}$$

$$F(B_1) = (B_1'; \Omega^{\tilde{h}} \,|\, \Omega^{\tilde{j}}; \tilde{h}, \tilde{j}) \text{ where } B_1 \equiv \{l \mapsto S_1\} \uplus E' \tag{33}$$

$$D_1' \equiv B_1' \tag{34}$$

Using (29) and (30) on (24), by definition of reduction we can derive

$$(B, Q_M) \rightarrow (B_1, (\nu\tilde{b})(Q_3 \,|\, Q_4\{W_1'/\chi\} \,|\, \ldots \,|\, Q_4\{W_n'/\chi\})) \tag{35}$$

and by (26), (31), and point (3) of Lemma D.4 we have that

$$(\nu\tilde{b})(Q_3 \,|\, Q_4\{W_1'/\chi\} \,|\, \ldots \,|\, Q_4\{W_n'/\chi\}) \equiv \langle\!\langle (\nu\tilde{b})(Q_3 \,|\, Q_5) \,|\, \Omega^{\tilde{i}} \rangle\!\rangle$$

By syntactical reasoning on (25) we get $(\nu\tilde{b})(Q_3 \,|\, Q_5) \xrightarrow{\tau}\!\!\!\rightarrow_\Lambda Q_2$, by Lemma C.8 we obtain $\langle\!\langle(\nu\tilde{b})(Q_3 \,|\, Q_5) \,|\, \Omega^{\tilde{i}}\rangle\!\rangle \xrightarrow{\tau}\!\!\!\rightarrow_\Lambda \langle\!\langle Q_2 \,|\, \Omega^{\tilde{i}}\rangle\!\rangle$, and by Lemma D.1 we have

$$(B_1, \langle\!\langle(\nu\tilde{b})(Q_3 \,|\, Q_5) \,|\, \Omega^{\tilde{i}}\rangle\!\rangle) \rightarrow^* (B_1, \langle\!\langle Q_2 \,|\, \Omega^{\tilde{i}}\rangle\!\rangle) \tag{36}$$

By (22), (27), (32) we get $P_2 \,|\, \Theta^{\tilde{h}} \,|\, \Theta^{\tilde{i}} \,|\, \Theta^{\tilde{j}} \approx_\Lambda Q_2 \,|\, \Omega^{\tilde{h}} \,|\, \Omega^{\tilde{i}} \,|\, \Omega^{\tilde{j}}$, and by Lemma C.11 we get $\langle\!\langle P_2 \,|\, \Theta^{\tilde{i}}\rangle\!\rangle \,|\, \Theta^{\tilde{h}} \,|\, \Theta^{\tilde{j}} \approx_\Lambda \langle\!\langle Q_2 \,|\, \Omega^{\tilde{i}}\rangle\!\rangle \,|\, \Omega^{\tilde{h}} \,|\, \Omega^{\tilde{j}}$, and by (28), (33) and (34) we conclude with

$$\begin{array}{ccc}
(D,P) & \longrightarrow & (D_1, \langle\!\langle P_2 \,|\, \Theta^{\tilde{i}}\rangle\!\rangle) \\
\asymp\Big\downarrow & & \Big\uparrow\asymp \\
(B,Q) & \xrightarrow{\;*\;} & (B_1, \langle\!\langle Q_2 \,|\, \Omega^{\tilde{i}}\rangle\!\rangle)
\end{array}$$

□

D.3 Soundness of $\approx_\Lambda$

Statement of Theorem 4.2. *Process bisimilarity is a sound approximation of process barbed congruence. For all Λ, if $P \approx_\Lambda Q$ then $P \sim^b_\Lambda Q$.*

Proof. We show that domain bisimilarity on processes is a sound approximation of barbed congruence. Given $P \approx_\Lambda Q$, we want to show that $P \sim^b_\Lambda Q$. By definition, $P \sim^b_\Lambda Q$ if, for all D such that $\Lambda \subseteq dom(D)$, $(D,P) \simeq^b (D,Q)$. If $P \approx_\Lambda Q$ then, by closure under parallel composition $P \,|\, K \approx_\Lambda Q \,|\, K$ for all configurations K, and therefore we have that $((D,P),(D,Q)) \in \asymp$, according to Definition D.1. Since by Lemma D.5 $\asymp \subseteq \simeq^b$, we conclude.

The converse implication does not hold. Consider

$$\mathtt{P}(\mathtt{a},\mathtt{b}) = (\nu c)(\overline{l{\cdot}c}\langle\rangle \,|\, !l{\cdot}c().l{\cdot}\mathsf{update}_{/}(X, \mathtt{a}[\,0\,]).l{\cdot}\mathsf{update}_{/}(X, \mathtt{b}[\,0\,]).\overline{l{\cdot}c}\langle\rangle)$$

and the counterexample

$$\mathtt{P}(\mathtt{a},\mathtt{b}) \sim^b_\Lambda \mathtt{P}(\mathtt{b},\mathtt{a}) \qquad \mathtt{P}(\mathtt{a},\mathtt{b}) \not\approx_\Lambda \mathtt{P}(\mathtt{b},\mathtt{a})$$

where, for each tree at l, $\mathtt{P}(\mathtt{a},\mathtt{b})$ can reduce to $\mathtt{P}(\mathtt{b},\mathtt{a})$ (and therefore exhibit the same barb) and vice-versa, whereas the labelled transitions cannot be matched. □

E Domain bisimilarity as a fixed-point

This paragraph follows the structure of Section 4.6 of [17].

Definition E.1 *The operator $\mathcal{F} : (2^{\mathcal{L}} \rightarrow 2^{\mathcal{K}\times\mathcal{K}}) \rightarrow (2^{\mathcal{L}} \rightarrow 2^{\mathcal{K}\times\mathcal{K}})$ over relations on configurations indexed by set of locations, is defined as*

$$(K,L) \in \mathcal{F}(\mathcal{R})(\Lambda) \quad \textit{if and only if}$$

- $K \xrightarrow{\alpha_l} K'$ *implies:*

1. *if* $l \in \Lambda$ *with* $rel(\alpha_l, L)$ *then* $L \overset{\alpha'_l}{\twoheadrightarrow}_\Lambda Q'$ *where* $\alpha'_l \equiv \alpha_l$ *and* $(K', L') \in \mathcal{R}(\Lambda)$*;*
2. *if* $l \notin \Lambda$ *then* $(K, L) \in \mathcal{R}(\Lambda \cup \{l\})$*.*

- $L \xrightarrow{\alpha_l} L'$ *implies:*

1. *if* $l \in \Lambda$ *with* $rel(\alpha_l, K)$ *then* $K \overset{\alpha'_l}{\twoheadrightarrow}_\Lambda K'$ *where* $\alpha'_l \equiv \alpha_l$ *and* $(K', L') \in \mathcal{R}(\Lambda)$*;*
2. *if* $l \notin \Lambda$ *then* $(K, L) \in \mathcal{R}(\Lambda \cup \{l\})$*.*

Definition E.2 *Given the set* $2^{\mathcal{L}} \to 2^{\mathcal{K} \times \mathcal{K}}$ *and two elements* $\mathcal{R}, \mathcal{R}'$ *we will denote with* $\sqsubseteq$ *the point-wise induced ordering, and we will write* $\mathcal{R} \sqsubseteq \mathcal{R}'$ *iff* $\forall \Lambda \in 2^{\mathcal{L}}.\mathcal{R}(\Lambda) \subseteq \mathcal{R}'(\Lambda)$*.*

Proposition E.1

1. $\mathcal{F}$ *is monotonic;*
2. $\mathcal{R}$ *is a domain bisimulation iff* $\mathcal{R} \sqsubseteq \mathcal{F}(\mathcal{R})$*.*

Proof. (1) We must show that given $\mathcal{R} \sqsubseteq \mathcal{R}'$ we have $\mathcal{F}(\mathcal{R}) \sqsubseteq \mathcal{F}(\mathcal{R}')$. By Definition E.2 this is equivalent to show that given a generic Λ, $\mathcal{F}(\mathcal{R})(\Lambda) \subseteq \mathcal{F}(\mathcal{R}')(\Lambda)$ knowing that for all Λ, $\mathcal{R}(\Lambda) \subseteq \mathcal{R}'(\Lambda)$, which follows easily by Definition E.1.
(2) Again by Definition E.2 we have to show that for a generic Λ, $\mathcal{R}(\Lambda) \subseteq \mathcal{F}(\mathcal{R})(\Lambda)$, which follows by Definition 4.1. $\square$

Proposition E.2 *Domain bisimilarity is the largest fixed-point of* $\mathcal{F}$*.*

Proof. By point (2) of Proposition E.1, each domain bisimulation is a pre-fixed-point, and vice versa. By Definition 4.1, $\approx$ is a domain bisimulation, and being the largest, it is also the largest pre-fixed-point of $\mathcal{F}$. By monotonicity (point (1) of Proposition E.1) we have that $\mathcal{F}(\approx) \sqsubseteq \mathcal{F}(\mathcal{F}(\approx))$, and therefore also $\mathcal{F}(\approx)$ is a pre-fixed-point, and since $\approx$ is the largest, we have $\mathcal{F}(\approx) \sqsubseteq \approx$, which gives us $\mathcal{F}(\approx) = \approx$. Since each fixed-point is also a pre-fixed-point, $\approx$ is the largest fixed-point. $\square$

Corollary E.3 *Domain bisimulation is well defined.*