Stateful Applied Pi Calculus: Observational Equivalence and Labelled Bisimilarity

Arapinis, Myrto; Liu, Jia; Ritter, Eike; Ryan, Mark

DOI:
10.1016/j.jlamp.2017.03.001

Stateful Applied Pi Calculus: Observational Equivalence and Labelled Bisimilarity

Myrto Arapinis[a], Jia Liu[b,*], Eike Ritter[c], Mark Ryan[c]

[a]School of Informatics, University of Edinburgh, UK
[b]Institute for Communication Systems (ICS), 5G Innovation Centre (5GIC), University of Surrey, UK
[c]School of Computer Science, University of Birmingham, UK

Abstract

We extend Abadi-Fournet's applied pi calculus with state cells, which are used to reason about protocols that store persistent information. Examples are protocols involving databases or hardware modules with internal state. We distinguish between private state cells, which are not available to the attacker, and public state cells, which arise when a private state cell is compromised by the attacker. For processes involving only private state cells we define observational equivalence and labelled bisimilarity in the same way as in the original applied pi calculus, and show that they coincide. Our result implies Abadi-Fournet's theorem – the coincidence of observational equivalence and labelled bisimilarity – in a revised version of the applied pi calculus. For processes involving public state cells, we can essentially keep the definition of observational equivalence, but need to strengthen the definition of labelled bisimulation in order to show that observational equivalence and labelled bisimilarity coincide in this case as well.

1. Introduction

Security protocols are small distributed programs that use cryptography in order to achieve multiple security goals like confidentiality, authentication. The complexity that arises from their distributed nature motivates formal analysis in order to prove logical properties of their behaviour; fortunately, they are often small enough to make this kind of analysis feasible. Various logical methods have been used to model security protocols; process calculi have been particularly successful [3, 5, 34]. For example, the TLS protocol used by billions of users every day was analysed using ProVerif [12].

More recently, protocol analysis methods have been applied to stateful protocols – that is, protocols which involve persistent state information that can affect and be changed by protocol runs. Hardware devices that have some internal memory can be described by such protocols. For example, Yubikey is a USB device which generates one-time passwords based on encryptions of a secret ID, a running counter and some random values using a unique AES-128 key contained in the device. The trusted

*Corresponding author. E-mail: jia.kitoun.liu@gmail.com

platform module (TPM) is another hardware chip that has a variety of registers which represent its state, and protocols for updating them. Radio-frequency identification (RFID) is a wireless technology for automatic identification and is currently deployed in electronic passports, tags for consumer goods, livestock and pets tracking, etc. An RFID-tag has a small area for storing secrets, which may be modified.

A process calculus can be made to work with such stateful protocols either by extension or by encoding. Extension means adding to the calculus explicit constructs for working with the stateful aspects, while encoding means using combinations of the primitives that already exist. Encodings have the advantage that they keep the calculus simple and elegant, but (as argued in [3]) there may not be encodings for all the aspects we want, and in cases that encodings exist they may not be suitable for the analysis of security properties.

In this paper we choose to extend the applied pi calculus rather than use the encoding for two reasons. Firstly, state and channels are conceptually different: states store information whereas channels are used for communication. There is also a well-established way of adding state to programming languages which will apply to add state to the applied pi calculus. Secondly, automated protocol verification tools based on the applied pi calculus like ProVerif often fail to prove security properties when using the encoding via restricted channels. ProVerif also provides some built-in features, such as tables and phases, which provide only limited ways for modelling states. In particular, tables are defined as predicates which allow processes to store data by extending a predicate for the data. Hence there is no notion of the "current" state, and values cannot be deleted from tables. Phases are used to model the protocols with several stages. But there can be only finitely many phases, which can only be run in sequence, whereas a state may have infinitely many arbitrary values. StatVerif [8] extends ProVerif with explicit states, thereby implementing the extension of the applied pi calculus presented in this paper. It has been successfully used in cases where ProVerif fails.

Our Contributions. We present an extension of the applied pi calculus by adding state cells, which are used to reason about protocols that store persistent information. We distinguish between private state cells, which are not available to the attacker, and public state cells, which arise when a private state cell is compromised by the attacker. In our stateful language, a private state cell is guarded by the scope restriction; its access is limited to some designated processes. When a private state cell gets compromised, the cell becomes public and this scenario is modelled by removing the scope restriction of that cell. We first define observational equivalence and labelled bisimilarity for processes having only private state cells, and we prove that two notions coincide as expected. By encoding the private state cells with restricted channels while keeping observational equivalence, our coincidence result can be seen to imply Abadi-Fournet's theorem [3, Theorem 1], in a revised version of applied pi calculus. As far as we can see, the only available proof for this theorem is [31] which is an unpublished manuscript. Despite having no published proof, this theorem has been widely used in many publications, for example [21, 9, 4, 20, 22].

We also discuss an extension of our language with public state cells. The obvious notion of labelled bisimilarity does not capture observational equivalence on public state cells. Designing a labelled bisimilarity on public state cells turns out to be unexpectedly difficult. Public state cells introduce many special language features which are significantly different from private state cells. Moreover, the addition of public state cells increases the capabilities of the attacker significantly. Hence we strengthen the definition of labelled bisimilarity to show that observational equivalence and labelled bisimulation coincide.

As an illustration, we analyse the OSK protocol [28] for RFID tags. We model its untraceability by private state cells and model its forward privacy by public state cells.

This paper is an extension of the conference version [7] with the complete proofs.

Related Work. StatVerif [8] is an extension of ProVerif process language [14] with private state cells. The main contribution there is to extend the ProVerif compiler to a compiler for StatVerif. So far, StatVerif can only handle secrecy properties, which are modelled as reachability properties of traces. SAPIC [29] is a similar tool as StatVerif except that SAPIC is based on Tamarin verifier [36] rather than ProVerif. Both StatVerif and SAPIC are the compilers which translate a stateful language into a low-level language that is directly supported by a tool, i.e., horn-clauses supported by ProVerif, multiset rewriting rules (in which antecedents of applied rules are withdrawn from the knowledge set in order to represent state changes) supported by Tamarin. However, none of the existing works study the language feature of the stateful language and the notions of process equivalences are never defined for a stateful language. Process equivalences are important concepts which can be used to model the indistinguishability properties in security protocols [5, 3].

This paper describes the process calculus on which StatVerif is based. More precisely, the focus in this paper is to build a stateful language based on applied pi calculus, explore its language features and discuss indistinguishability, which is modelled by observational equivalence and analysed by labelled bisimilarity. This paper provides therefore the basis to extend StatVerif to handle bisimilarity.

There are other languages that have been used to model protocols involving persistent state, but they are lower-level languages that are further away than our process language from the protocol design. Strand spaces have been generalised to work with the global state required by a trusted party charged with enforcing fair exchange [27].

Tamarin has been used to analyse stateful protocols directly without going through the stateful language of SAPIC, e.g. for the analysis of hardware password tokens [30]. Multi-set rewriting is also used in [33], where state changes are important to represent revocation of cryptographic keys. Horn clauses rather than multiset rewriting are used in [24], in order to represent state changes made to registers of the TPM hardware module.

Reasoning about programming languages involving states has been extensively studied (e.g. [37, 25]). There are very strong interactions between programming language features and state, hence the reasoning principles are very specific to the precise combination of features. In this work we build on the work on reasoning principles for process calculi using bisimulation and show how to extend these principles to handle global state.

Outline. The next section defines syntax and semantics for the stateful applied pi calculus. Section 3 discusses the process equivalences and encoding for private state cells, and derives Abadi-Fournet's theorem. Section 6 extends our stateful language with public state cells. The paper concludes in Section 7.

2. Stateful Applied Pi Calculus

In this section, we extend the applied pi calculus [3] with constructs for states, and define its operational semantics. In fact, we do not directly build the stateful language on top of applied pi calculus, because we want to avoid working with the *structural equivalence* relation. More precisely, reasoning about the equivalent classes induced by structural equivalence turns out to be difficult and normally results in long tedious proofs [23, 20, 32, 19]. Our language inherits constructs for scope restriction, communication and active substitutions from applied pi calculus while having multisets of processes and active substitutions makes it possible to specify an operational semantics which does not involve any structural equivalence.

2.1. Syntax

We assume two disjoint, infinite sets $\mathcal{N}$ and $\mathcal{V}$ of *names* and *variables*, respectively. We rely on a sort system including a universal base sort, a cell sort and a channel sort. The sort system splits $\mathcal{N}$ into channel names $\mathcal{N}_{ch}$, base names $\mathcal{N}_b$ and cell names $\mathcal{N}_s$; similarly, $\mathcal{V}$ is split into channel variables $\mathcal{V}_{ch}$ and base variables $\mathcal{V}_b$. Unless otherwise stated, we use a, b, c as channel names, s, t as cell names, and x, y, z as variables. Meta variables u, v, w are used to range over both names and variables.

A signature Σ consists of a finite set of function symbols, each with an arity. A function symbol with arity 0 is a constant. Function symbols are required to take arguments and produce results of the base sort only. *Terms*, ranged over by M, N, are built up from variables and names by function application:

M, N	$::=$	terms
	a, b, c, k, m, n, s	names
	x, y, z	variables
	$f(M_1, \ldots, M_\ell)$	function application

We write $var(M)$ and $name(M)$ for the variables and names in M, respectively. Tuples such as $u_1 \cdots u_\ell$ and $M_1 \cdots M_\ell$ will be denoted by $\widetilde{u}$ and $\widetilde{M}$, respectively. Terms are equipped with an equational theory $=_\Sigma$ that is an equivalence relation closed under substitutions of terms for variables, one-to-one renamings and function applications.

The grammar for the *plain process* is given below. The operators for nil process 0, parallel composition $|$, replication !, scope restriction νn, conditional `if - then - else` , input $u(x)$ and output $\overline{u}\langle M\rangle$ are the same as the ones in applied pi calculus [3]. The process $[s \mapsto M]$ represents that the current value stored in a cell s is M. The process `lock` $s.P$ locks the cell s for the subsequent process P. When the cell s is locked, another process that intends to access the cell has to wait until the cell is unlocked by a primitive `unlock` s. The process `read` s `as` $x.P$ reads the value in the cell and stores it in x. The process $s := M.P$ assigns the value M to the cell and continues as P.

$P, Q, R ::=$	plain process
0	nil process
$P \mid Q$	parallel composition
$!\,P$	replication
$\nu n.P$	name restriction
$\texttt{if}\ M = N\ \texttt{then}\ P\ \texttt{else}\ Q$	conditional
$u(x).P$	input
$\overline{u}\langle M\rangle.P$	output
$[s \mapsto M]$	cell s, containing term M
$s := M.\,P$	writing a cell
$\texttt{read}\ s\ \texttt{as}\ x.P$	reading a cell
$\texttt{lock}\ s.P$	locking a cell
$\texttt{unlock}\ s.P$	unlocking a cell

subject to the following requirements:

- x, M, N are not of cell sort; $u \in \mathcal{N}_{ch} \cup \mathcal{V}_{ch}$ and $s \in \mathcal{N}_s$; additionally, M is of base sort in both $[s \mapsto M]$ and $s := M.\,P$;
- for every $\texttt{lock}\ s.\,P$, the part P of the process must not include parallel or replication unless it is after an $\texttt{unlock}\ s$.
- for a given cell name s, the replication operator ! must not occur between νs and $[s \mapsto M]$.

These side conditions rule out some nonsense processes, such as $\texttt{lock}\ s.\,!P$, $\texttt{lock}\ s.\,(P \mid Q)$, $\nu s.![s \mapsto M]$ and $\nu s.([s \mapsto M] \mid [s \mapsto N])$, while keep some reasonable processes, such as $\texttt{lock}\ s.\texttt{unlock}\ s.\,!P$, $\texttt{lock}\ s.\texttt{unlock}\ s.\,(P \mid Q)$ and $!\nu s.[s \mapsto M]$.

An *extended process*, ranged over by A, B, C, is an expression of the form

$$\nu\widetilde{n}.\,(\sigma, S, \mathcal{P})$$

where

- $\nu\widetilde{n}$ is a set of name restrictions;
- σ is a substitution $\{M_1/x_1, \ldots, M_n/x_n\}$ which replaces variables of base sort with terms of base sort; we define $dom(\sigma) := \{x_1, \ldots, x_n\}$ and $dom(\nu\widetilde{n}.(\sigma, S, \mathcal{P})) := dom(\sigma)$; we require that $dom(\sigma) \cap fv(M_1, \ldots, M_n, \mathcal{P}, S) = \emptyset$;
- $S = \{s_1 \mapsto M_1, \ldots, s_m \mapsto M_m\}$ is a set of state cells such that $s_1, \ldots, s_m$ are pairwise-distinct cell names and terms $M_1, \ldots, M_m$ are of base sort; we write $dom(S)$ for $\{s_1, \ldots, s_m\}$ and $S(s_i)$ for M_i ($1 \leq i \leq m$);
- $[s \mapsto M]$ can only occur at most once for a given cell name s, and if a cell name s is not restricted by any νs, a state cell $s \mapsto M$ can only occur in S;

- $\mathcal{P} = \{(P_1, L_1), \ldots, (P_k, L_k)\}$ is a multiset of pairs where P_i is a plain process and L_i is a set of cell names; $L_i \cap L_j = \emptyset$ for any $1 \leq i, j \leq k$ and $i \neq j$; for each $s \in L_i$, the part of the process P_i must not include parallel or replication unless it is after a `unlock` s; we write $locks(\mathcal{P})$ for the set $L_1 \cup \cdots \cup L_k$, namely the locked cells in $\mathcal{P}$.

In an extended process $\nu\widetilde{n}.(\sigma, S, \mathcal{P})$, the substitution σ is similar to the active substitutions in applied pi calculus [3] which denote the static knowledge that the process exposes to the environment. A minor difference with [3] is that substitutions here are only defined on terms of base sort which will be explained later. State cells are mutable and the value of a cell may be changed during the running of processes. If a process P locks a cell s, then this status information will be kept as $(P, \{s\} \cup L)$ in $\mathcal{P}$. At any time, the cell s can be locked at most once in $\mathcal{P}$.

The variable x in "$u(x)$" and "`read` s `as` x" are bound, as well as the name n in νn. This leads to the usual notions of bound and free names and variables. We shall use $fn(A)$ for free names, use $fs(A)$ for free cell names, use $fv(A)$ for free variables, use $bn(A)$ for bound names, and use $bv(A)$ for bound variables of A. Let $fnv(A) = fn(A) \cup fv(A)$ and $bnv(A) = bn(A) \cup bv(A)$. Following the conventions in [35], we shall identify processes which are α-convertible. We write "=" for both syntactical equality and equivalence under α-conversion. Captures of bound names and bound variables are avoided by implicit α-conversion.

An extended process $\nu\widetilde{n}.(\sigma, S, \mathcal{P})$ is called *closed* if the following conditions all hold: 1) each variable is either defined by σ or bound; 2) each cell name s is defined by exactly one "$s \mapsto M$" (either in S or in $\mathcal{P}$); 3) $locks(\mathcal{P}) \subseteq dom(S)$. Note that a variable defined in σ will not occur in S or $\mathcal{P}$ because of the condition $dom(\sigma) \cap fv(M_1, \ldots, M_n, \mathcal{P}, S) = \emptyset$ in the above definition of extended processes.

We may write $(\sigma, S, \mathcal{P})$ for $\nu\emptyset.(\sigma, S, \mathcal{P})$, and write $\nu\widetilde{n}, \widetilde{m}.(\sigma, S, \mathcal{P})$ for $\nu(\widetilde{n} \cup \widetilde{m}).(\sigma, S, \mathcal{P})$.

When we write $\sigma = \sigma_1 \cup \sigma_2$ for some substitution σ or $S = S_1 \cup S_2$ for some state cells S, we assume that $dom(\sigma_1) \cap dom(\sigma_2) = \emptyset$ as well as $dom(S_1) \cap dom(S_2) = \emptyset$. For variables $\widetilde{x}$, we define $\sigma_{\backslash\widetilde{x}}$ to be the substitution $\{\, z\sigma/z \mid z \in dom(\sigma) \text{ and } z \notin \widetilde{x}\,\}$. If $A = \nu\widetilde{n}.(\sigma, S, \mathcal{P})$, we write $A_{\backslash\widetilde{x}}$ for $\nu\widetilde{n}.(\sigma_{\backslash\widetilde{x}}, S, \mathcal{P})$.

An *evaluation context* $\nu\widetilde{n}.(\sigma\text{-}, S\text{-}, \mathcal{P}\text{-})$ is an extended process with holes "-" for substitution, state cells and plain processes. Let $\mathcal{C} = \nu\widetilde{n}.(\sigma\text{-}, S\text{-}, \mathcal{P}\text{-})$ be an evaluation context and $A = \nu\widetilde{m}.(\sigma_a, S_a, \mathcal{P}_a)$ be a closed extended process with $\widetilde{m} \cap (\widetilde{n} \cup fn(\sigma, S, \mathcal{P})) = dom(\sigma) \cap dom(\sigma_a) = dom(S) \cap dom(S_a) = \emptyset$. The result of applying $\mathcal{C}$ to A is an extended process defined by:

$$\mathcal{C}[A] = \nu\widetilde{n}, \widetilde{m}.(\sigma\sigma_a \cup \sigma_a, S\sigma_a \cup S_a, \mathcal{P}\sigma_a \cup \mathcal{P}_a)$$

An evaluation context $\mathcal{C}$ *closes* A when $\mathcal{C}[A]$ is a closed extended process.

The main differences between our language and the language in StatVerif [8] are: 1) In our language, terms are divided into three types: base type, channel type and cell type, while StatVerif only has one universal type of terms. The active substitutions in our language are only defined on the terms of base type and terms can only be input and output on the terms of channel type, which is to fix the flaw of the coincidence result between labelled bisimilarity and observational equivalence [3] and will be further discussed in Section 5. Moreover, we don't allow input and output terms of cell type which will be explained in the following Section 6.1. 2) Our language uses the α-conversion to automatically change the bound names to avoid name collisions, while StatVerif uses a fixed set to record the bound names.

2.2. Operational Semantics

The *transition* relation $A \xrightarrow{\alpha} A'$ is the smallest relation on extended processes defined by the rules in Figure 1. The action α is either an internal action τ, an input $a(x)$, an output of channel name $\overline{a}\langle c\rangle$, an output of bound channel name $\nu c.\overline{a}\langle c\rangle$, or an output of terms of base sort $\nu x.\overline{a}\langle x\rangle$. The transitions for conditional branch, communication, sending and receiving channel names and complex messages are typical and essentially the same as the ones in applied pi calculus. In particular, the output $\nu x.\overline{a}\langle x\rangle$ for term M generates an "alias" x for M which is kept in the substitution part of the extended process. As mentioned before, state cells are used to model the hardware or the database to which the access is usually mutually-exclusive. When a state cell is locked, the other process that intends to access the cell must wait until the cell is released.

2.3. Case study

In this section, we demonstrate the intelligibility of our stateful applied pi calculus by comparing the formalisation of Trusted Platform Module (TPM) in applied pi calculus and its formalisation in stateful applied pi. Intelligibility of the translation from English specification of security protocols to formal model is important since the design of security protocols are error-prone and usually complicated.

State cells can be encoded by private channels which will be studied in the following Section 4. The exclusive access to the cell is modelled by the unique features of private channels. For example, in process $\nu c.\,(\overline{c} \mid c.a_1.a_2\overline{c} \mid c.b.\overline{c})$ [1], the actions a_1 and a_2 cannot be interrupted by b. However, encoding state cells with private channels is pretty incomprehensible and not intuitive. For example. an input $c(x)$ on a private channel could be an input action, could also be an encoding for a lock or read primitive, or encoding for something else. We cannot be sure unless we analyse the semantics of the whole process. In comparison, when using stateful primitives (such as lock, read), the meaning can be interpreted immediately from their syntax, and it reminds the reader here is an operation on state cells rather than an ordinary sending or receiving a message on a channel. We illustrate this point by a case study on modelling the Trusted Platform Module (TPM).

Overview of Trusted Platform Module (TPM). TPM is a hardware chip designed to enable commodity computers to achieve greater levels of security. TPMs are manufactured by chip producers, including Atmel, Broadcom, Infineon, Sinosun, STMicroelectronics, and Winbond. It is specified by the Trusted Computing Group (TCG) industry consortium. The TPM offers an application program interface (API) providing operations related to:

[1] We omit the objects in input $u(x)$ and output $\overline{u}\langle M\rangle$ and write u and $\overline{u}$ instead when the objects do not matter.

$$
\begin{array}{rcl}
\nu\widetilde{n}.\,(\sigma, S, \mathcal{P} \cup \{(\,!\,P, \emptyset)\}) & \xrightarrow{\tau} & \nu\widetilde{n}.\,(\sigma, S, \mathcal{P} \cup \{(\,!\,P, \emptyset), (P, \emptyset)\}) \\
\nu\widetilde{n}.(\sigma, S, \mathcal{P} \cup \{(P \mid Q, \emptyset)\}) & \xrightarrow{\tau} & \nu\widetilde{n}.(\sigma, S, \mathcal{P} \cup \{(P, \emptyset), (Q, \emptyset)\}) \\
\nu\widetilde{n}.(\sigma, S, \mathcal{P} \cup \{(\nu m.P, L)\}) & \xrightarrow{\tau} & \nu\widetilde{n}, m.(\sigma, S, \mathcal{P} \cup \{(P, L)\}) \\
 & & \textbf{if } m \notin \mathit{fn}(\widetilde{n}, \sigma, S, \mathcal{P}, L) \\
\nu\widetilde{n}.(\sigma, S, \mathcal{P} \cup \{([s \mapsto M], \emptyset)\}) & \xrightarrow{\tau} & \nu\widetilde{n}.(\sigma, S \cup \{s \mapsto M\}, \mathcal{P}) \\
 & & \textbf{if } s \in \widetilde{n} \text{ and } s \notin \mathit{dom}(S) \\
\nu\widetilde{n}.(\sigma, S, \mathcal{P} \cup \{(a(x).P, L_1)\} \cup \{(\overline{a}\langle M\rangle.Q, L_2)\}) & \xrightarrow{\tau} & \nu\widetilde{n}.(\sigma, S, \mathcal{P} \cup \{(P\{M/x\}, L_1), (Q, L_2)\})) \\
\nu\widetilde{n}.(\sigma, S, \mathcal{P} \cup \{(\texttt{if } M = N \texttt{ then } P \texttt{ else } Q, L)\}) & \xrightarrow{\tau} & \nu\widetilde{n}.(\sigma, S, \mathcal{P} \cup \{(P, L)\}) \\
 & & \textbf{if } M =_\Sigma N \\
\nu\widetilde{n}.(\sigma, S, \mathcal{P} \cup \{(\texttt{if } M = N \texttt{ then } P \texttt{ else } Q, L)\}) & \xrightarrow{\tau} & \nu\widetilde{n}.(\sigma, S, \mathcal{P} \cup \{(Q, L)\}) \\
 & & \textbf{if } M \neq_\Sigma N \text{ and } \mathit{var}(M, N) = \emptyset \\
\nu\widetilde{n}.(\sigma, S \cup \{s \mapsto M\}, \mathcal{P} \cup \{(\texttt{read } s \texttt{ as } x.P, L)\}) & \xrightarrow{\tau} & \nu\widetilde{n}.(\sigma, S \cup \{s \mapsto M\}, \mathcal{P} \cup \{(P\{M/x\}, L)\}) \\
 & & \textbf{if } s \in \widetilde{n} \cup L \text{ and } s \notin \mathit{locks}(\mathcal{P}) \\
\nu\widetilde{n}.(\sigma, S \cup \{s \mapsto M\}, \mathcal{P} \cup \{(s := N.P, L)\}) & \xrightarrow{\tau} & \nu\widetilde{n}.(\sigma, S \cup \{s \mapsto N\}, \mathcal{P} \cup \{(P, L)\}) \\
 & & \textbf{if } s \in \widetilde{n} \cup L \text{ and } s \notin \mathit{locks}(\mathcal{P}) \\
\nu\widetilde{n}.(\sigma, S \cup \{s \mapsto M\}, \mathcal{P} \cup \{(\texttt{lock } s.P, L)\}) & \xrightarrow{\tau} & \nu\widetilde{n}.(\sigma, S \cup \{s \mapsto M\}, \mathcal{P} \cup \{(P, L \cup \{s\})\}) \\
 & & \textbf{if } s \in \widetilde{n} \text{ and } s \notin L \cup \mathit{locks}(\mathcal{P}) \\
\nu\widetilde{n}.(\sigma, S \cup \{s \mapsto M\}, \mathcal{P} \cup \{(\texttt{unlock } s.P, L)\}) & \xrightarrow{\tau} & \nu\widetilde{n}.(\sigma, S \cup \{s \mapsto M\}, \mathcal{P} \cup \{(P, L \setminus \{s\})\}) \\
 & & \textbf{if } s \in \widetilde{n} \cap L \\
\nu\widetilde{n}.(\sigma, S, \mathcal{P} \cup \{(a(x).P, L)\}) & \xrightarrow{a(M)} & \nu\widetilde{n}.(\sigma, S, \mathcal{P} \cup \{(P\{M\sigma/x\}, L)\}) \\
 & & \textbf{if } \mathit{name}(a, M) \cap \widetilde{n} = \emptyset \\
\nu\widetilde{n}.(\sigma, S, \mathcal{P} \cup \{(\overline{a}\langle c\rangle.P, L)\}) & \xrightarrow{\overline{a}\langle c\rangle} & \nu\widetilde{n}.(\sigma, S, \mathcal{P} \cup \{(P, L)\}) \\
 & & \textbf{if } a, c \notin \widetilde{n} \\
\nu\widetilde{n}, c.(\sigma, S, \mathcal{P} \cup \{(\overline{a}\langle c\rangle.P, L)\}) & \xrightarrow{\nu c.\overline{a}\langle c\rangle} & \nu\widetilde{n}.(\sigma, S, \mathcal{P} \cup \{(P, L)\}) \\
 & & \textbf{if } a, c \notin \widetilde{n} \text{ and } a \neq c \\
\nu\widetilde{n}.(\sigma, S, \mathcal{P} \cup \{(\overline{a}\langle M\rangle.P, L)\}) & \xrightarrow{\nu x.\overline{a}\langle x\rangle} & \nu\widetilde{n}.(\sigma \cup \{M/x\}, S, \mathcal{P} \cup \{(P, L)\}) \\
 & & \textbf{if } a \notin \widetilde{n} \text{ and } M \text{ is of base sort and } x \text{ is fresh}
\end{array}
$$

Figure 1: Operational Semantics

- Platform configuration registers (PCRs): the TPM contains at least 16 PCRs in its shielded memory which store platform configuration measurements. The only operation for changing the value u of a PCR is to extend it by a value v, resulting in the PCR value $\mathsf{hash}(u, v)$. A PCR can be either static or dynamic. For simplicity, in our formal model, we assume TPM only has two PCRs: a static PCR and a dynamic PCR. A system reboot will reset the value in the static PCRs and dynamic PCRs to -1. Only an SKINIT instruction (described below) can reset a dynamic PCR to 0. This enables a remote verifier to distinguish between a reboot and a dynamic reset.

- Secure key management and storage: the TPM can generate new keys, and impose restrictions on their use. TPMs provide sealed storage, whereby data can be encrypted using a 2048-bit RSA key whose private component never leaves the TPM in unencrypted form. For simplicity, in our formalisation, we assume the data is encrypted directly under the storage root key (SRK). The SRK is a pair of RSA keys that is used to encrypt other keys stored outside the TPM. SRK is embedded in the TPM. The sealed data can be bound to a particular software state, as defined by the contents of various PCRs. The TPM will only unseal (decrypt) the data when the PCRs contain the values specified by the seal command.

  Seal function encrypts a secret m with a public key $\mathsf{pk}(k)$, a specific PCR value v and a secret $tpmpf$ as $\mathsf{aenc}\left(\mathsf{pk}(k), tpmpf, v, m\right)$. The decryption is modelled as an equation $\mathsf{adec}\left(x, \mathsf{aenc}\left(\mathsf{pk}(x), y, z, u\right)\right) = \langle y, z, u \rangle$.

- The SKINIT instruction creates an isolated execution environment in which security-sensitive code can be protected from all other software and devices. The code to be executed within this protected environment is called Secure Loader Block (SLB). The SKINIT instruction takes the physical memory address of SLB as its only argument. The SKINIT instruction resets the value of a dynamic PCR to 0, transmits a copy of the SLB to the system's TPM and extends the value of PCR with the measurement of the SLB, and then begins to execute the SLB.

Each command (e.g., reboot, extend) on TPM is executed atomically without interruption. To formalise TPM, we introduce three state cells: state cell tpm for access control, state cell $spcr$ for static PCR, and state cell $dpcr$ for dynamic PCR. The operations on TPM include *reboot, extend, skinit* and we assume these commands are sent on the corresponding public channels *reboot, extend, skinit*. These operations are modelled as processes REBOOT, EXTEND, SKINIT correspondingly. The formalisation of TPM in applied pi calculus is given in Figure 2 and the formalisation of TPM in stateful applied pi calculus is given in Figure 3.

Syntax Sugar. We write "$\texttt{let}\ \langle x, y \rangle = M\ \texttt{in}\ P$" for "$P\left\{\mathsf{fst}\left(M\right)/x, \mathsf{snd}\left(M\right)/y\right\}$", and similarly "$\texttt{let}\ \langle x, y, z \rangle = M\ \texttt{in}\ P$" for "$P\left\{\mathsf{fst}\left(M\right)/x, \mathsf{snd}\left(M\right)/y, \mathsf{trd}\left(M\right)/z\right\}$".

3. Process Equivalences for Private State Cells

In this section, we discuss the language features of stateful applied pi with only private state cells, that is, each cell name s occurring in the processes is within the scope

$$
\begin{aligned}
TPM := \ & \nu\ tpm, spcr, dpcr, srk, tpmpf, secret. \\
& \Big(\overline{c}\langle \mathsf{pk}(srk)\rangle \mid \overline{c}\langle \mathsf{aenc}\,(\mathsf{pk}\,(srk)\,, tpmpf, \mathsf{hash}(\mathbf{0}, \mathbf{slb}), secret)\rangle \mid \\
& \overline{tpm}\langle \mathbf{init}\rangle \mid \overline{spcr}\langle -\mathbf{1}\rangle \mid \overline{dpcr}\langle -\mathbf{1}\rangle \mid SKINIT \mid REBOOT \mid EXTEND\Big)
\end{aligned}
$$

$$
\begin{aligned}
SKINIT := \ & !\ \ tpm\,(z)\,.\ skinit\,(xArgs)\,.\ dpcr\,(x)\,.\ (\overline{dpcr}\langle \mathbf{0}\rangle \mid \\
& \quad \mathtt{let}\ \langle xSlb, xCom, xSBlob\rangle = xArgs\ \mathtt{in} \\
& \quad dpcr\,(y)\,.(\overline{dpcr}\langle \mathsf{hash}(\mathbf{0}, xSlb)\rangle \mid \\
& \qquad \mathtt{if}\ xCom = \mathbf{unseal}\ \mathtt{then} \\
& \qquad\quad \mathtt{let}\ \langle yProof, yPcr, ySecret\rangle = \mathsf{adec}(srk, xSBlob)\ \mathtt{in} \\
& \qquad\quad dpcr\,(xPcr)\,.(\overline{dpcr}\langle xPcr\rangle \mid \\
& \qquad\quad \mathtt{if}\ \langle tpmpf, xPcr\rangle = \langle yProof, yPcr\rangle \\
& \qquad\qquad \mathtt{then}\ \overline{c}\langle ySecret\rangle.\ dpcr\,(xPcr)\,.\ \big(\overline{dpcr}\langle \mathsf{hash}(xPcr, \mathbf{end})\rangle \mid \overline{tpm}\langle z\rangle\big) \\
& \qquad\qquad \mathtt{else}\ dpcr\,(xPcr)\,.\ \big(\overline{dpcr}\langle \mathsf{hash}(xPcr, \mathbf{end})\rangle \mid \overline{tpm}\langle z\rangle\big) \\
& \qquad\quad) \\
& \qquad \mathtt{else}\ \overline{tpm}\langle z\rangle \\
& \quad) \\
&)
\end{aligned}
$$

$$
REBOOT := \ !\ \ tpm\,(z)\,.\ reboot\,(xArgs)\,.\ spcr\,(x)\,.\ \overline{spcr}\langle -\mathbf{1}\rangle.\ dpcr\,(y)\,.\ \overline{dpcr}\langle -\mathbf{1}\rangle.\ \overline{tpm}\langle z\rangle
$$

$$
\begin{aligned}
EXTEND := \ & !\ \ tpm\,(z)\,. \\
& extend\,(xArgs)\,. \\
& \mathtt{let}\ \langle xCom, xHash\rangle = xArgs\ \mathtt{in} \\
& \mathtt{if}\ xCom = \mathbf{static} \\
& \quad \mathtt{then}\ spcr\,(xPcr)\,.\ \big(\overline{spcr}\langle \mathsf{hash}(xPcr, xHash)\rangle \mid \overline{tpm}\langle z\rangle\big) \\
& \quad \mathtt{else\ if}\ xCom = \mathbf{dynamic} \\
& \qquad\quad \mathtt{then}\ \ dpcr\,(yPcr)\,.\ \big(\overline{dpcr}\langle \mathsf{hash}(yPcr, xHash)\rangle \mid \overline{tpm}\langle z\rangle\big) \\
& \qquad\quad \mathtt{else}\ \ \overline{tpm}\langle z\rangle
\end{aligned}
$$

Figure 2: Modelling TPM in applied pi calculus

$$TPM := \nu\ tpm, spcr, dpcr, srk, tpmpf, secret.$$

$$\left(\emptyset, \left\{\begin{array}{c} tpm \mapsto \mathbf{init} \\ spcr \mapsto -\mathbf{1} \\ dpcr \mapsto -\mathbf{1} \end{array}\right\}, \left\{\begin{array}{l} (\bar{c}\langle \mathsf{pk}(srk)\rangle, \emptyset) \\ (\bar{c}\langle \mathsf{aenc}\left(\mathsf{pk}\left(srk\right), tpmpf, \mathsf{hash}(\mathbf{0}, \mathbf{slb}), secret\right)\rangle, \emptyset) \\ (SKINIT \mid REBOOT \mid EXTEND, \emptyset) \end{array}\right\}\right)$$

$$\begin{array}{ll} SKINIT := & !\ \mathtt{lock}\ tpm \\ & skinit\,(xArgs) \\ & dpcr := \mathbf{0} \\ & \mathtt{let}\ \langle xSlb, xCom, xSBlob\rangle = xArgs\ \mathtt{in} \\ & dpcr := \mathsf{hash}(\mathbf{0}, xSlb) \\ & \mathtt{if}\ xCom = \mathbf{unseal}\ \mathtt{then} \\ & \quad \mathtt{let}\ \langle yProof, yPcr, ySecret\rangle = \mathsf{adec}(srk, xSBlob)\ in \\ & \quad \mathtt{read}\ dpcr\ \mathtt{as}\ xPcr \\ & \quad \mathtt{if}\ \langle tpmpf, xPcr\rangle = \langle yProof, yPcr\rangle \\ & \qquad \mathtt{then}\ \bar{c}\langle ySecret\rangle.\ dpcr := \mathsf{hash}(xPcr, \mathbf{end}).\ \mathtt{unlock}\ tpm \\ & \qquad \mathtt{else}\ dpcr := \mathsf{hash}(xPcr, \mathbf{end}).\ \mathtt{unlock}\ tpm \\ & \mathtt{else}\ \mathtt{unlock}\ tpm \end{array}$$

$$REBOOT := !\ \mathtt{lock}\ tpm.\ reboot\,(xArgs)\,.\ spcr := -\mathbf{1}.\ dpcr := -\mathbf{1}.\ \mathtt{unlock}\ tpm$$

$$\begin{array}{ll} EXTEND := & !\ \mathtt{lock}\ tpm \\ & extend\,(xArgs) \\ & \mathtt{let}\ \langle xCom, xHash\rangle = xArgs\ \mathtt{in} \\ & \mathtt{if}\ xCom = \mathbf{static} \\ & \quad \mathtt{then}\ \mathtt{read}\ spcr\ \mathtt{as}\ xPcr.\ spcr := \mathsf{hash}(xPcr, xHash).\ \mathtt{unlock}\ tpm \\ & \quad \mathtt{else}\ \mathtt{if}\ xCom = \mathbf{dynamic} \\ & \qquad\quad \mathtt{then}\ \mathtt{read}\ dpcr\ \mathtt{as}\ yPcr.\ dpcr := \mathsf{hash}(yPcr, xHash).\ \mathtt{unlock}\ tpm \\ & \qquad\quad \mathtt{else}\ \mathtt{unlock}\ tpm \end{array}$$

Figure 3: Modelling TPM in stateful applied pi calculus

of a restriction νs. We first present the coincidence between observational equivalence and labelled bisimilarity on the extended processes with only private state cells. Then we propose an encoding of private cells by using restricted channel names.

3.1. Observational Equivalence

For private state cells, we define observational equivalence in a similar way as in [3]. Observational equivalence [3] has been widely used to model properties of security protocols. It captures the intuition of indistinguishability from the attacker's point of view. Security properties such as anonymity [4], privacy [22, 6] and strong secrecy [13] are usually formalised by observational equivalence.

We write $\Longrightarrow$ for the reflexive and transitive closure of $\xrightarrow{\tau}$; we define $\xRightarrow{\alpha}$ to be $\Longrightarrow\xrightarrow{\alpha}\Longrightarrow$; we define $\xRightarrow{\widehat{\alpha}}$ to be $\xRightarrow{\alpha}$ if α is not τ and $\Longrightarrow$ if $\alpha = \tau$. We write $A \Downarrow_a$ when $A \Longrightarrow \nu\widetilde{n}.(\sigma, S, \mathcal{P} \cup \{(\overline{a}\langle M\rangle.P, L)\})$ with $a \notin \widetilde{n}$.

Definition 1. *Observational equivalence ($\approx$) is the largest symmetric relation $\mathcal{R}$ on pairs of closed extended processes with only private state cells, such that $A\ \mathcal{R}\ B$ implies*

(i) $dom(A) = dom(B)$*;*

(ii) if $A \Downarrow_a$ then $B \Downarrow_a$;

(iii) if $A \Longrightarrow A'$ then $B \Longrightarrow B'$ and $A'\ \mathcal{R}\ B'$ for some B';

(iv) for all closing evaluation contexts $\mathcal{C}$ with only private cells, $\mathcal{C}[A]\ \mathcal{R}\ \mathcal{C}[B]$.

Observational equivalence is a contextual equivalence where the contexts model the active attackers who can intercept and forge messages. In the following examples, we illustrate the use of observational equivalence in the stateful language by analysing the untraceability of the RFID tags.

Example 2. *We start by analysing a naive protocol for RFID tag identification. The tag simply reads its id and sends it to the reader. We assume the attacker can eavesdrop on the radio frequency signals between the tag and the reader. In other words, all the communications between the tag and the reader are visible to the attacker. The operations on the tag can be modelled by: $P(s) = \texttt{read}\ s\ \texttt{as}\ x.\overline{a}\langle x\rangle$. One security concern for RFID tags is to avoid third-party attacker tracking. The attacker is not supposed to trace the tag according to its outputs. Using the definition in [6], the untraceability can be modelled by observational equivalence:*

$$(\emptyset, \emptyset, \{(\,!\,\nu s, id.([s \mapsto id] \mid P(s)), \emptyset)\}) \approx (\emptyset, \emptyset, \{(\,!\,\nu s, id.([s \mapsto id] \mid !\,P(s)), \emptyset)\})$$

In the left process, each tag s can be used at most once. In the right process, each tag s can be used an unbounded number of times. The above equivalence does not hold, which means this protocol is traceable. By eavesdropping on channel a of the right process, the attacker can get a data sequence: "$id, id, id \cdots$", while a particular id can occur at most once in the first process.

Example 3. *The OSK protocol [28] is a simple identification protocol for RFID tags which aims to satisfy third-party untraceability. The tag can perform two independent one-way functions g and h. An initial secret is stored in the tag and is known to the back-end database. On each run of the protocol, the tag computes the hash g of its current value and sends the result to the reader. The reader forwards the message to the back-end database for identification. The tag then updates its value with the hash h of its current value. The operations related to a tag s can be modelled by:*

$$T(s) = \texttt{lock}\ s.\ \texttt{read}\ s\ \texttt{as}\ x.\ \overline{a}\langle g(x)\rangle.\ s := h(x).\ \texttt{unlock}\ s$$

Let RD be process modelling the reader and back-end database. Similar to Example 2, the untraceability can be represented by

$$\begin{aligned} &(\emptyset, \emptyset, \{(\,!\,\nu s, k.([s \mapsto k] \mid T(s) \mid RD), \emptyset)\}) \\ \approx\, &(\emptyset, \emptyset, \{(\,!\,\nu s, k.([s \mapsto k] \mid !\,T(s) \mid RD), \emptyset)\}) \end{aligned}$$

In the second process, for a particular tag s which contains value k, the data sequence observed by the attacker on channel a is "$g(k), g(h(k)), g(h(h(k)))\cdots$". Without knowing the secret k, these appear just random data to the attacker and so the attacker cannot link these data to the same tag. The observational equivalence between these two processes means the attacker cannot identify the multiple runnings of a particular tag. The "$\texttt{lock}\ s \cdots \texttt{unlock}\ s$" ensures exclusive access to the tag. After the reader reads the tag, the tag must be renewed before the next access to the tag; otherwise the tag would be traceable.

3.2. Labelled Bisimilarity

The universal quantifier over the contexts makes it difficult to prove observational equivalence. Hence labelled bisimilarity is introduced in [3] to capture observational equivalence. Labelled bisimilarity consists of static equivalence and behavioural equivalence.

Definition 4. *Two processes A and B are* statically equivalent*, written as $A \approx_s B$, if $dom(A) = dom(B)$, and for any terms M and N with $var(M, N) \subseteq dom(A)$, $M\sigma_1 =_\Sigma N\sigma_1$ iff $M\sigma_2 =_\Sigma N\sigma_2$ where $A = \nu\widetilde{n}_1.(\sigma_1, S_1, \mathcal{P}_1)$ and $B = \nu\widetilde{n}_2.(\sigma_2, S_2, \mathcal{P}_2)$ for some $\widetilde{n}_1, \widetilde{n}_2$ such that $(\widetilde{n}_1 \cup \widetilde{n}_2) \cap name(M, N) = \emptyset$.*

Our definition of static equivalence is essentially the same as the one in [3], as the definition in [3] is invariant under structural equivalence already. Although static equivalence is in general undecidable, there are well established ways, including tools, for verifying static equivalence [2, 17, 18, 10, 16]. Static equivalence defines the indistinguishability between the environmental knowledge exposed by two processes. The environmental knowledge is modelled by the substitutions in the extended processes. For example, let $A = \nu k, m.(\{k/x, m/y\}, \emptyset, \emptyset)$ and $B = \nu k.(\{k/x, h(k)/y\}, \emptyset, \emptyset)$. The test $h(x) = y$ fails under the application of A's substitution $\{k/x, m/y\}$, while succeeds under the application of B's substitution $\{k/x, h(k)/y\}$. Hence $A \not\approx_s B$.

Definition 5. *Labelled bisimilarity ($\approx_l$) is the largest symmetric relation $\mathcal{R}$ between pairs of closed extended processes with only private state cells such that $A\ \mathcal{R}\ B$ implies*

1. $A \approx_s B$*;*

2. if $A \xrightarrow{\alpha} A'$ *and* $fv(\alpha) \subseteq dom(A)$ *and* $bn(\alpha) \cap fn(B) = \emptyset$*, then* $B \stackrel{\widehat{\alpha}}{\Longrightarrow} B'$ *such that* $A' \mathcal{R} B'$ *for some* B'*.*

Instead of using arbitrary contexts, labelled bisimilarity relies on the direct comparison of the transitions.

3.3. Soundness and Completeness

In this section, we show that when there is only private state cells in the language, labelled bisimilarity can fully capture observational equivalence. For an evaluation context $\mathcal{C}$, we write $\mathcal{C}[A]_{\setminus\widetilde{x}}$ for the process $(\mathcal{C}[A])_{\setminus\widetilde{x}}$. We write $\prod_{i\in I} P_i$ for the parallel composition $P_1 \mid P_2 \mid \cdots \mid P_{|I|}$.

The following Lemma 6 states that the labelled bisimilarity is closed under the application of contexts:

Lemma 6. *Let* A *be a closed extended process with only private state cells and* $\mathcal{C} = \nu\widetilde{n}.(\sigma\text{-}, S\text{-}, \mathcal{P}\text{-})$ *be a closing evaluation context with only private state cells and* $\widetilde{x} \subseteq dom(A)$*.*

1. If $A \xrightarrow{c(M\sigma)} B$ *with* $name(c, M) \cap \widetilde{n} = \emptyset$ *and* $var(M) \subseteq dom(\mathcal{C}[A]_{\setminus\widetilde{x}})$*, then* $\mathcal{C}[A]_{\setminus\widetilde{x}} \xrightarrow{c(M)} \mathcal{C}[B]_{\setminus\widetilde{x}}$*;*

2. If $A \xrightarrow{\alpha} B$ *with* $name(\alpha) \cap \widetilde{n} = \emptyset$ *and* $var(\alpha) \cap \widetilde{x} = \emptyset$*, then* $\mathcal{C}[A]_{\setminus\widetilde{x}} \xrightarrow{\alpha} \mathcal{C}[B]_{\setminus\widetilde{x}}$ *when* α *is not an input.*

Proof. Proof can be found in Appendix A.

Using Lemma 6 several times, we can obtain the following corollary:

Corollary 7. *Let* A *be a closed extended process with only private state cells and* $\mathcal{C} = \nu\widetilde{n}.(\sigma\text{-}, S\text{-}, \mathcal{P}\text{-})$ *be a closing evaluation context with only private state cells and* $\widetilde{x} \subseteq dom(A)$*.*

1. If $A \stackrel{c(M\sigma)}{\Longrightarrow} B$ *with* $name(c, M) \cap \widetilde{n} = \emptyset$ *and* $var(M) \subseteq dom(\mathcal{C}[A]_{\setminus\widetilde{x}})$*, then* $\mathcal{C}[A]_{\setminus\widetilde{x}} \stackrel{c(M)}{\Longrightarrow} \mathcal{C}[B]_{\setminus\widetilde{x}}$*;*

2. If $A \stackrel{\alpha}{\Longrightarrow} B$ *with* $name(\alpha) \cap \widetilde{n} = \emptyset$ *and* $var(\alpha) \cap \widetilde{x} = \emptyset$*, then* $\mathcal{C}[A]_{\setminus\widetilde{x}} \stackrel{\alpha}{\Longrightarrow} \mathcal{C}[B]_{\setminus\widetilde{x}}$ *when* α *is not an input.*

We first prove that labelled bisimilarity is sound w.r.t. observational equivalence. That is to say the labelled bisimilarity is closed under the application of arbitrary context:

Proposition 8 (Soundness). *On closed extended processes with only private state cells, the labelled bisimilarity* $\approx_l$ *is a congruence.*

Proof. We prove that $\approx_l$ is a congruence by constructing the following set

$$\mathcal{R} = \{ (\mathcal{C}[A_1], \mathcal{C}[A_2]) \mid A_1 \approx_l A_2,\ \mathcal{C} \text{ is a closing evaluation context with only private state cells} \}$$

and prove that $\mathcal{R} \subseteq \approx_l$.

Assume $(\mathcal{C}[A_1], \mathcal{C}[A_2]) \in \mathcal{R}$ because of $A_1 \approx_l A_2$ where $A_i = \nu\widetilde{n}_i.(\sigma_i, S_i, \mathcal{P}_i)$ with $i = 1, 2$ and $\mathcal{C} = \nu\widetilde{n}.(\sigma\text{-}, S\text{-}, \mathcal{P}\text{-})$. Then $\mathcal{C}[A_i] = \nu\widetilde{n}, \widetilde{n}_i.(\sigma\sigma_i \cup \sigma_i, S\sigma_i \cup S_i, \mathcal{P}\sigma_i \cup \mathcal{P}_i)$. To prove $\mathcal{R} \subseteq \approx_l$, we need to show $\mathcal{C}[A_1] \approx_s \mathcal{C}[A_2]$, and if $\mathcal{C}[A_1] \xrightarrow{\alpha} B_1$ for some B_1 then there exists B_2 such that $\mathcal{C}[A_2] \xRightarrow{\widehat{\alpha}} B_2$ and $(B_1, B_2) \in \mathcal{R}$.

First we check the static equivalence $\mathcal{C}[A_1] \approx_s \mathcal{C}[A_2]$. Let $\varphi_i = \sigma\sigma_i \cup \sigma_i$ with $i = 1, 2$. From $dom(\sigma_1) = dom(\sigma_2)$, we have $dom(\varphi_1) = dom(\varphi_2)$. Note that for any term M with $var(M) \subseteq dom(\varphi_i)$, we have $M\varphi_i = (M\sigma)\sigma_i$ and $var(M\sigma) \subseteq dom(\sigma_i)$ since $dom(\sigma) \cap dom(\sigma_i) = \emptyset$ and $\mathcal{C}$ is a closing evaluation context to A_i. Assume terms M, N with $var(M, N) \subseteq dom(\varphi_i)$ and $M\varphi_1 =_\Sigma N\varphi_1$. We shall prove that $M\varphi_2 =_\Sigma N\varphi_2$. From the above analysis, we have $(M\sigma)\sigma_1 = M\varphi_1$, $(N\sigma)\sigma_1 = N\varphi_1$, $(M\sigma)\sigma_1 =_\Sigma (N\sigma)\sigma_1$ and $var(M\sigma, N\sigma) \subseteq dom(\sigma_i)$. Since $A_1 \approx_s A_2$, we have $(M\sigma)\sigma_2 =_\Sigma (N\sigma)\sigma_2$. From $(M\sigma)\sigma_2 = M\varphi_2$ and $(N\sigma)\sigma_2 = N\varphi_2$, we have $M\varphi_2 =_\Sigma N\varphi_2$. Hence we have $\mathcal{C}[A_1] \approx_s \mathcal{C}[A_2]$.

For the behavioural equivalence, we discuss the different cases of α. For each transition $\mathcal{C}[A_1] \xrightarrow{\alpha} B_1$, we need to find some matched transitions $\mathcal{C}[A_2] \xRightarrow{\widehat{\alpha}} B_2$ such that $(B_1, B_2) \in \mathcal{R}$.

1. Assume a transition is about reading a cell s and

$$\mathcal{C}[A_1] = \nu\widetilde{n}, \widetilde{n}_1.(\sigma\sigma_1 \cup \sigma_1, S\sigma_1 \cup S_1, \mathcal{P}\sigma_1 \cup \mathcal{P}_1) \xrightarrow{\tau} B_1$$

The "$\mathtt{read}\ s\ \mathtt{as}\ z$" comes either from the context $\mathcal{C}$ or from the process A_1.

(a) Assume $\mathtt{read}\ s\ \mathtt{as}\ z$ is from the context $\mathcal{C}$. Since A_1, A_2 only contain private state cells, the context $\mathcal{C}$ cannot access any private state cells in A_1, A_2. Thus s can only be a cell defined in S in context $\mathcal{C}$. Assume $\mathcal{C} = \nu\widetilde{n}.(\sigma\text{-}, S' \cup \{s \mapsto M\}\text{-}, \mathcal{P}' \cup \{(\mathtt{read}\ s\ \mathtt{as}\ z.P\sigma_1, L)\}\text{-})$, then

$$\begin{aligned}
&\mathcal{C}[A_1] = \\
&\nu\widetilde{n}, \widetilde{n}_1.\, (\sigma\sigma_1 \cup \sigma_1, S'\sigma_1 \cup \{s \mapsto M\sigma_1\} \cup S_1, \mathcal{P}'\sigma_1 \cup \{(\mathtt{read}\ \boldsymbol{s}\ \mathtt{as}\ \boldsymbol{z.P\sigma_1}, L)\} \cup \mathcal{P}_1) \\
&\xrightarrow{\tau} B_1 = \\
&\nu\widetilde{n}, \widetilde{n}_1.\, (\sigma\sigma_1 \cup \sigma_1, S'\sigma_1 \cup \{s \mapsto M\sigma_1\} \cup S_1, \mathcal{P}'\sigma_1 \cup \{(\boldsymbol{(P\sigma_1)\,\{M\sigma_1/z\}}, L)\} \cup \mathcal{P}_1) \\
&= \nu\widetilde{n}, \widetilde{n}_1.\, (\sigma\sigma_1 \cup \sigma_1, S'\sigma_1 \cup \{s \mapsto M\sigma_1\} \cup S_1, \mathcal{P}'\sigma_1 \cup \{(\boldsymbol{(P\,\{M/z\})\sigma_1}, L)\} \cup \mathcal{P}_1)
\end{aligned}$$

From the structure of $\mathcal{C}$, we can have the following transitions from $\mathcal{C}[A_2]$:

$$\begin{aligned}
&\mathcal{C}[A_2] = \nu\widetilde{n}, \widetilde{n}_2.(\sigma\sigma_2 \cup \sigma_2, S'\sigma_2 \cup \{s \mapsto M\sigma_2\} \cup S_2, \boldsymbol{\mathcal{P}\sigma_2} \cup \mathcal{P}_2) \\
&= \nu\widetilde{n}, \widetilde{n}_2.\, \big(\sigma\sigma_2 \cup \sigma_2, S'\sigma_2 \cup \{s \mapsto M\sigma_2\} \cup S_2, \boldsymbol{\mathcal{P}'\sigma_2} \cup \{(\mathtt{read}\ \boldsymbol{s}\ \mathtt{as}\ \boldsymbol{z.P\sigma_2}, L)\} \cup \mathcal{P}_2\big) \\
&\xrightarrow{\tau} B_2 = \\
&\nu\widetilde{n}, \widetilde{n}_2.\, (\sigma\sigma_2 \cup \sigma_2, S'\sigma_2 \cup \{s \mapsto M\sigma_2\} \cup S_2, \mathcal{P}'\sigma_2 \cup \{(\boldsymbol{(P\sigma_2)\,\{M\sigma_2/z\}}, L)\} \cup \mathcal{P}_2) \\
&= \nu\widetilde{n}, \widetilde{n}_2.\, (\sigma\sigma_2 \cup \sigma_2, S'\sigma_2 \cup \{s \mapsto M\sigma_2\} \cup S_2, \mathcal{P}'\sigma_2 \cup \{(\boldsymbol{(P\,\{M/z\})\sigma_2}, L)\} \cup \mathcal{P}_2)
\end{aligned}$$

Let $\mathcal{C}' = \nu\widetilde{n}.(\sigma\text{-}, S\text{-}, \mathcal{P}'_2 \cup \{(P\{M/z\}, L)\}\text{-})$. Then we can verify that $\mathcal{C}'[A_i] = B_i$ for $i = 1, 2$. Since $A_1 \approx_l A_2$, we have $(B_1, B_2) \in \mathcal{R}$.

(b) Assume $\texttt{read}\ s\ \texttt{as}\ z$ is from the process and

$$A_1 = \nu\widetilde{n}_1.(\sigma_1, S'_1 \cup \{s \mapsto M\}, \{(\texttt{read}\ s\ \texttt{as}\ z.P, L)\} \cup \mathcal{P}'_1)$$

Then

$$\begin{aligned}&\mathcal{C}[A_1] = \nu\widetilde{n}, \widetilde{n}_1.\,(\sigma\sigma_1 \cup \sigma_1, S\sigma_1 \cup S'_1 \cup \{s \mapsto M\}, \mathcal{P}\sigma_1 \cup \{(\texttt{read}\ \boldsymbol{s}\ \texttt{as}\ \boldsymbol{z.P}, L)\} \cup \mathcal{P}'_1)\\ &\xrightarrow{\tau} B_1 = \nu\widetilde{n}, \widetilde{n}_1.\,((\sigma\sigma_1 \cup \sigma_1, S\sigma_1 \cup S'_1 \cup \{s \mapsto M\}, \mathcal{P}\sigma_1 \cup \{(\boldsymbol{P\{M/z\}}, L)\} \cup \mathcal{P}'_1))\end{aligned}$$

Then A_1 can perform the read action and

$$\begin{aligned}A_1 &= \nu\widetilde{n}_1.(\sigma_1, S_1 \cup S'_1 \cup \{s \mapsto M\}, \mathcal{P}'_1 \cup \{(\texttt{read}\ \boldsymbol{s}\ \texttt{as}\ \boldsymbol{z.P}, L)\})\\ &\xrightarrow{\tau} A'_1 = \nu\widetilde{n}_1.(\sigma_1, S_1 \cup S'_1 \cup \{s \mapsto M\}, \mathcal{P}'_1 \cup \{(\boldsymbol{P\{M/z\}}\,x, L)\})\end{aligned}$$

and $\mathcal{C}[A'_1] = B_1$. From $A_1 \approx_l A_2$, there exists A'_2 such that $A_2 \Longrightarrow A'_2 \approx_l A'_1$. Using Corollary 7 we obtain $\mathcal{C}[A_2] \Longrightarrow \mathcal{C}[A'_2]$. Let $B_2 = \mathcal{C}[A'_2]$. Hence $(B_1, B_2) \in \mathcal{R}$.

2. Assume a transition is about locking a cell s and

$$\mathcal{C}[A_1] = \nu\widetilde{n}, \widetilde{n}_1.(\sigma\sigma_1 \cup \sigma_1, S\sigma_1 \cup S_1, \mathcal{P}\sigma_1 \cup \mathcal{P}_1) \xrightarrow{\tau} B_1$$

and $s \in \widetilde{n} \cup \widetilde{n}_1$ and $s \notin \mathit{locks}(\mathcal{P}_1, \mathcal{P})$. The $\texttt{lock}\ s$ comes either from $\mathcal{P}$ in the context part or from $\mathcal{P}_1$ in the process part.

(a) Assume $\texttt{lock}\ s$ is from the context part and $\mathcal{P} = \mathcal{P}' \cup \{(\texttt{lock}\ s.P, L)\}$.

$$\begin{aligned}&\mathcal{C}[A_1] = \nu\widetilde{n}, \widetilde{n}_1.(\sigma\sigma_1 \cup \sigma_1, S\sigma_1 \cup S_1, \mathcal{P}'\sigma_1 \cup \{(\texttt{lock}\ \boldsymbol{s.P\sigma_1}, L)\} \cup \mathcal{P}_1)\\ &\xrightarrow{\tau} B_1 = \nu\widetilde{n}, \widetilde{n}_1.(\sigma\sigma_1 \cup \sigma_1, S\sigma_1 \cup S_1, \mathcal{P}'\sigma_1 \cup \{\boldsymbol{(P\sigma_1, L \cup \{s\})}\} \cup \mathcal{P}_1)\end{aligned}$$

Since A_1, A_2 only contain private state cells, the context $\mathcal{C}$ cannot access any private state cells in A_1, A_2. Thus s is a state cell from context $\mathcal{C}$. We can have the following transitions from $\mathcal{C}[A_2]$:

$$\begin{aligned}\mathcal{C}[A_2] =\ & \nu\widetilde{n}, \widetilde{n}_2.(\sigma\sigma_2 \cup \sigma_2, S\sigma_2 \cup S_2, \boldsymbol{\mathcal{P}\sigma_2} \cup \mathcal{P}_2)\\ =\ & \nu\widetilde{n}, \widetilde{n}_2.(\sigma\sigma_2 \cup \sigma_2, S\sigma_2 \cup S_2, \mathcal{P}'\sigma_2 \cup \{\boldsymbol{(}\texttt{lock}\ \boldsymbol{s.P\sigma_2, L)}\} \cup \mathcal{P}_2)\\ \xrightarrow{\tau} B_2 =\ & \nu\widetilde{n}, \widetilde{n}_2.(\sigma\sigma_2 \cup \sigma_2, S\sigma_2 \cup S_2, \mathcal{P}'\sigma_2 \cup \{\boldsymbol{(P\sigma_2, L \cup \{s\})}\} \cup \mathcal{P}_2)\end{aligned}$$

Let $\mathcal{C}' = \nu\widetilde{n}.(\sigma\text{-}, S\text{-}, \mathcal{P}' \cup \{(P, L \cup \{s\})\}\text{-})$. Then we can verify that $\mathcal{C}'[A_i] = B_i$ for $i = 1, 2$. Since $A_1 \approx_l A_2$, we have $(B_1, B_2) \in \mathcal{R}$.

(b) Assume $\mathcal{P}_1 = \mathcal{P}'_1 \cup \{(\texttt{lock}\ s.P, L)\}$ and

$$\begin{aligned}&\mathcal{C}[A_1] = \nu\widetilde{n}, \widetilde{n}_1.(\sigma\sigma_1 \cup \sigma_1, S\sigma_1 \cup S_1, \mathcal{P}\sigma_1 \cup \{\boldsymbol{(}\texttt{lock}\ \boldsymbol{s.P, L)}\} \cup \mathcal{P}'_1)\\ &\xrightarrow{\tau} B_1 = \nu\widetilde{n}, \widetilde{n}_1.(\sigma\sigma_1 \cup \sigma_1, S\sigma_1 \cup S_1, \mathcal{P}\sigma_1 \cup \{\boldsymbol{(P, L \cup \{s\})}\} \cup \mathcal{P}'_1)\end{aligned}$$

Then A_1 can perform the lock action and

$$\begin{aligned} A_1 =& \nu\widetilde{n}_1.(\sigma_1, S_1, \mathcal{P}_1' \cup \{(\texttt{lock}\ \boldsymbol{s.P, L})\}) \\ &\xrightarrow{\tau} A_1' = \nu\widetilde{n}_1.(\sigma_1, S_1, \mathcal{P}_1' \cup \{(\boldsymbol{P, L \cup \{s\}})\}) \end{aligned}$$

and $\mathcal{C}[A_1'] = B_1$. From $A_1 \approx_l A_2$, there exists A_2' such that $A_2 \Longrightarrow A_2' \approx_l A_1'$. Using Corollary 7 we obtain $\mathcal{C}[A_2] \Longrightarrow \mathcal{C}[A_2']$. Let $B_2 = \mathcal{C}[A_2']$. We know that $(B_1, B_2) \in \mathcal{R}$.

3. The analysis for cases when the transition is caused by writing or unlocking is similar as above.

4. Assume

$$\mathcal{C}[A_1] = \nu\widetilde{n}, \widetilde{n}_1.(\sigma\sigma_1 \cup \sigma_1, S\sigma_1 \cup S_1, \mathcal{P}\sigma_1 \cup \mathcal{P}_1) \xrightarrow{\overline{a}\langle c\rangle} B_1$$

The output comes either from $\mathcal{P}$ in the context part or from $\mathcal{P}_1$ in the process part.

(a) Assume the output is from the context part and $\mathcal{P} = \mathcal{P}' \cup \{(\overline{a}\langle c\rangle.P, L)\}$.

$$\begin{aligned} &\mathcal{C}[A_1] = \nu\widetilde{n}, \widetilde{n}_1.(\sigma\sigma_1 \cup \sigma_1, S\sigma_1 \cup S_1, \mathcal{P}'\sigma_1 \cup \{(\boldsymbol{\overline{a}\langle c\rangle.P\sigma_1}, L)\} \cup \mathcal{P}_1) \\ \xrightarrow{\overline{a}\langle c\rangle}\ & B_1 = \nu\widetilde{n}, \widetilde{n}_1.(\sigma\sigma_1 \cup \sigma_1, S\sigma_1 \cup S_1, \mathcal{P}'\sigma_1 \cup \{(\boldsymbol{P\sigma_1}, L)\} \cup \mathcal{P}_1) \end{aligned}$$

Since the output comes from context, we can have the following transitions from $\mathcal{C}[A_2]$:

$$\begin{aligned} \mathcal{C}[A_2] =\ & \nu\widetilde{n}, \widetilde{n}_2.(\sigma\sigma_2 \cup \sigma_2, S\sigma_2 \cup S_2, \boldsymbol{\mathcal{P}\sigma_2} \cup \mathcal{P}_2) \\ =\ & \nu\widetilde{n}, \widetilde{n}_2.(\sigma\sigma_2 \cup \sigma_2, S\sigma_2 \cup S_2, \boldsymbol{\mathcal{P}'\sigma_2 \cup \{(\overline{a}\langle c\rangle.P\sigma_2, L)\}} \cup \mathcal{P}_2) \\ \xrightarrow{\overline{a}\langle c\rangle} B_2 =\ & \nu\widetilde{n}, \widetilde{n}_2.(\sigma\sigma_2 \cup \sigma_2, S\sigma_2 \cup S_2, \mathcal{P}'\sigma_2 \cup \{(\boldsymbol{P\sigma_2}, L)\} \cup \mathcal{P}_2) \end{aligned}$$

Let $\mathcal{C}' = \nu\widetilde{n}.(\sigma\text{-}, S\text{-}, \mathcal{P}_2' \cup \{(P, L)\}\text{-})$. Then we can verify that $\mathcal{C}'[A_i] = B_i$ for $i = 1, 2$. Since $A_1 \approx_l A_2$, we have $(B_1, B_2) \in \mathcal{R}$.

(b) Assume $\mathcal{P}_1 = \mathcal{P}_1' \cup \{(\overline{a}\langle c\rangle.P, L)\}$ and

$$\begin{aligned} &\mathcal{C}[A_1] = \nu\widetilde{n}, \widetilde{n}_1.(\sigma\sigma_1 \cup \sigma_1, S\sigma_1 \cup S_1, \mathcal{P}\sigma_1 \cup \{(\boldsymbol{\overline{a}\langle c\rangle.P}, L)\} \cup \mathcal{P}_1') \\ \xrightarrow{\overline{a}\langle c\rangle}\ & B_1 = \nu\widetilde{n}, \widetilde{n}_1.(\sigma\sigma_1 \cup \sigma_1, S\sigma_1 \cup S_1, \mathcal{P}\sigma_1 \cup \{(\boldsymbol{P}, L)\} \cup \mathcal{P}_1') \end{aligned}$$

Then A_1 can perform the output action and

$$A_1 = \nu\widetilde{n}_1.(\sigma_1, S_1, \mathcal{P}_1' \cup \{(\boldsymbol{\overline{a}\langle c\rangle.P}, L)\}) \xrightarrow{\overline{a}\langle c\rangle} A_1' = \nu\widetilde{n}_1.(\sigma_1, S_1, \mathcal{P}_1' \cup \{(\boldsymbol{P}, L)\})$$

and $\mathcal{C}[A_1'] = B_1$. From $A_1 \approx_l A_2$, there exists A_2' such that $A_2 \stackrel{\overline{a}\langle c\rangle}{\Longrightarrow} A_2' \approx_l A_1'$. Using Corollary 7 we obtain $\mathcal{C}[A_2] \stackrel{\overline{a}\langle c\rangle}{\Longrightarrow} \mathcal{C}[A_2']$. Let $B_2 = \mathcal{C}[A_2']$. We know that $(B_1, B_2) \in \mathcal{R}$.

5. Assume

$$\mathcal{C}[A_1] = \nu\widetilde{n}, \widetilde{n}_1.(\sigma\sigma_1 \cup \sigma_1, S\sigma_1 \cup S_1, \mathcal{P}\sigma_1 \cup \mathcal{P}_1) \xrightarrow{a(M)} B_1$$

where $name(a, M) \cap (\widetilde{n} \cup \widetilde{n}_1) = \emptyset$ and $fv(M) \subseteq dom(\sigma, \sigma_1)$.

The input action is defined either in $\mathcal{P}$ in the context part or in $\mathcal{P}_1$ in the process part.

(a) Assume the input action is defined in the context part, i.e., $\mathcal{P} = \mathcal{P}' \cup \{(a(z).P, L)\}$ for some $\mathcal{P}', P, L$ and $z \notin fv(A_1, A_2, \mathcal{C})$.

$$\begin{aligned} &\mathcal{C}[A_1] = \nu\widetilde{n}, \widetilde{n}_1.(\sigma\sigma_1 \cup \sigma_1, S\sigma_1 \cup S_1, \mathcal{P}'\sigma_1 \cup \{(\boldsymbol{a(z).P\sigma_1}, L)\} \cup \mathcal{P}_1) \\ &\xrightarrow{a(M)} B_1 \\ &= \nu\widetilde{n}, \widetilde{n}_1.\,(\sigma\sigma_1 \cup \sigma_1, S\sigma_1 \cup S_1, \mathcal{P}'\sigma_1 \cup \{(\boldsymbol{P\sigma_1 \{M(\sigma\sigma_1 \cup \sigma_1)/z\}}, L)\} \cup \mathcal{P}_1)) \\ &= \nu\widetilde{n}, \widetilde{n}_1.\,(\sigma\sigma_1 \cup \sigma_1, S\sigma_1 \cup S_1, \mathcal{P}'\sigma_1 \cup \{(\boldsymbol{(P\{M\sigma/z\})\sigma_1}, L)\} \cup \mathcal{P}_1) \end{aligned}$$

We construct a new evaluation context $\mathcal{C}' = \nu\widetilde{n}.(\sigma, S, \mathcal{P}' \cup \{(P\{M\sigma/z\}, L)\})$. We can easily verify that $\mathcal{C}'[A_1] = B_1$ and $\mathcal{C}[A_2] \xrightarrow{a(M)} \mathcal{C}'[A_2]$. Since $(A_1, A_2) \in \mathcal{R}$, we have $(\mathcal{C}'[A_1], \mathcal{C}'[A_2]) \in \mathcal{R}$.

(b) Assume the input action is defined in the process part, i.e., $\mathcal{P}_1 = \mathcal{P}_1' \cup \{(a(z).P, L)\}$ for some $\mathcal{P}_1', P, L$

$$\begin{aligned} &\mathcal{C}[A_1] = \nu\widetilde{n}, \widetilde{n}_1.(\sigma\sigma_1 \cup \sigma_1, S\sigma_1 \cup S_1, \mathcal{P}\sigma_1 \cup \{(\boldsymbol{a(z).P}, L)\} \cup \mathcal{P}_1') \\ &\xrightarrow{a(M)} B_1 = \\ &\nu\widetilde{n}, \widetilde{n}_1.\,(\sigma\sigma_1 \cup \sigma_1, S\sigma_1 \cup S_1, \mathcal{P}\sigma_1 \cup \{(\boldsymbol{P\{M(\sigma\sigma_1 \cup \sigma_1)/z\}}, L)\} \cup \mathcal{P}_1')) \end{aligned}$$

Then let A_1 input $M\sigma$ on channel a and we get

$$\begin{aligned} A_1 &= \nu\widetilde{n}_1.(\sigma_1, S_1, \mathcal{P}_1' \cup \{(\boldsymbol{a(z).P}, L)\}) \\ &\xrightarrow{a(M\sigma)} A_1' = \nu\widetilde{n}_1.(\sigma_1, S_1, \mathcal{P}_1' \cup \{(\boldsymbol{P\{(M\sigma)\sigma_1/z\}}, L)\}) \end{aligned}$$

Since $fv(M) \subseteq dom(\sigma, \sigma_1)$ and $dom(\sigma) \cap dom(\sigma_1) = \emptyset$, we have $(M\sigma)\sigma_1 = M(\sigma\sigma_1 \cup \sigma_1)$. We can further verify that $\mathcal{C}[A_1'] = B_1$. From $A_1 \approx_l A_2$, we know that $A_2 \overset{a(M\sigma)}{\Longrightarrow} A_2' \approx_l A_1'$. Using Corollary 7 we obtain $\mathcal{C}[A_2] \overset{a(M)}{\Longrightarrow} \mathcal{C}[A_2']$. Let $B_2 = \mathcal{C}[A_2']$. We know that $(B_1, B_2) \in \mathcal{R}$.

6. Assume

$$\mathcal{C}[A_1] = \nu\widetilde{n}, \widetilde{n}_1.(\sigma\sigma_1 \cup \sigma_1, S\sigma_1 \cup S_1, \mathcal{P}\sigma_1 \cup \mathcal{P}_1) \xrightarrow{\nu z.\overline{a}\langle z\rangle} B_1$$

The output comes either from $\mathcal{P}$ in the context part or from $\mathcal{P}_1$ in the process part.

(a) Assume the output is from the context part and $\mathcal{P} = \mathcal{P}' \cup \{(\overline{a}\langle M\rangle.P, L)\}$.

$$\begin{aligned} &\mathcal{C}[A_1] = \nu\widetilde{n}, \widetilde{n}_1.\,(\sigma\sigma_1 \cup \sigma_1, S\sigma_1 \cup S_1, \mathcal{P}'\sigma_1 \cup \{(\boldsymbol{\overline{a}\langle M\sigma_1\rangle.P\sigma_1}, L)\} \cup \mathcal{P}_1) \\ &\xrightarrow{\nu z.\overline{a}\langle z\rangle} B_1 = \nu\widetilde{n}, \widetilde{n}_1.\,(\sigma\sigma_1 \cup \sigma_1 \cup \boldsymbol{\{M\sigma_1/z\}}, S\sigma_1 \cup S_1, \mathcal{P}'\sigma_1 \cup \{(\boldsymbol{P\sigma_1}, L)\} \cup \mathcal{P}_1) \end{aligned}$$

Since the output comes from context, we can have the following transitions from $\mathcal{C}[A_2]$:

$$\begin{aligned}\mathcal{C}[A_2] = &\ \nu\widetilde{n},\widetilde{n}_2.(\sigma\sigma_2 \cup \sigma_2, S\sigma_2 \cup S_2, \boldsymbol{\mathcal{P}\sigma_2} \cup \mathcal{P}_2)\\ = &\ \nu\widetilde{n},\widetilde{n}_2.(\sigma\sigma_2 \cup \sigma_2, S\sigma_2 \cup S_2, \mathcal{P}'\sigma_2 \cup \{(\boldsymbol{\overline{a}\langle M\sigma_2\rangle.P\sigma_2}, L)\} \cup \mathcal{P}_2))\\ \xrightarrow{\nu z.\overline{a}\langle z\rangle} B_2 = &\ \nu\widetilde{n},\widetilde{n}_2.(\sigma\sigma_2 \cup \sigma_2 \cup \boldsymbol{\{M\sigma_2/z\}}, S\sigma_2 \cup S_2, \mathcal{P}'\sigma_2 \cup \{(\boldsymbol{P\sigma_2}, L)\} \cup \mathcal{P}_2))\end{aligned}$$

Let $\mathcal{C}' = \nu\widetilde{n}.(\sigma \cup \{M/z\}$ -, S-, $\mathcal{P}_2' \cup \{(P, L)\}$ -). Then we can verify that $\mathcal{C}'[A_i] = B_i$ for $i = 1, 2$. Since $A_1 \approx_l A_2$, we have $(B_1, B_2) \in \mathcal{R}$.

(b) Assume $\mathcal{P}_1 = \mathcal{P}_1' \cup \{(\overline{a}\langle M\rangle.P, L)\}$ and

$$\begin{aligned}&\mathcal{C}[A_1] = \nu\widetilde{n},\widetilde{n}_1.(\sigma\sigma_1 \cup \sigma_1, S\sigma_1 \cup S_1, \mathcal{P}\sigma_1 \cup \{(\boldsymbol{\overline{a}\langle M\rangle.P}, L)\} \cup \mathcal{P}_1')\\ \xrightarrow{\nu z.\overline{a}\langle z\rangle}\ &B_1 = \nu\widetilde{n},\widetilde{n}_1.(\sigma\sigma_1 \cup \sigma_1 \cup \boldsymbol{\{M/z\}}, S\sigma_1 \cup S_1, \mathcal{P}\sigma_1 \cup \{(\boldsymbol{P}, L)\} \cup \mathcal{P}_1')\end{aligned}$$

Then A_1 can perform the output action and

$$\begin{aligned}A_1 = &\ \nu\widetilde{n}_1.(\sigma_1, S_1, \mathcal{P}_1' \cup \{(\boldsymbol{\overline{a}\langle M\rangle.P}, L)\})\\ &\xrightarrow{\nu z.\overline{a}\langle z\rangle} A_1' = \nu\widetilde{n}_1.(\sigma_1 \cup \boldsymbol{\{M/z\}}, S_1, \mathcal{P}_1' \cup \{(\boldsymbol{P}, L)\})\end{aligned}$$

and $\mathcal{C}[A_1'] = B_1$. From $A_1 \approx_l A_2$, there exists A_2' such that $A_2 \stackrel{\nu z.\overline{a}\langle z\rangle}{\Longrightarrow} A_2' \approx_l A_1'$. Using Corollary 7 we obtain $\mathcal{C}[A_2] \stackrel{\nu z.\overline{a}\langle z\rangle}{\Longrightarrow} \mathcal{C}[A_2']$. Let $B_2 = \mathcal{C}[A_2']$. Hence $(B_1, B_2) \in \mathcal{R}$.

7. The other cases are similar.

Next, we shall prove the completeness of labelled bisimilarity:

Proposition 9 (Completeness). *On closed extended processes with only private state cells, observational equivalence $\approx$ implies labelled bisimilarity $\approx_l$.*

Proof. To show $\approx \subseteq \approx_l$, we construct the following set $\mathcal{R}$ and prove that $\mathcal{R} \subseteq \approx_l$.

$$\mathcal{R} = \{\ (A_1, A_2) \mid \exists\, \widetilde{a}, \widetilde{b}, \widetilde{c}, \widetilde{y} \text{ s.t. } \mathcal{C}[A_1]_{\backslash\widetilde{y}} \approx \mathcal{C}[A_2]_{\backslash\widetilde{y}}\ \}$$

where $\mathcal{C} = \nu\widetilde{c}.(\text{-}, \text{-}, \{(\overline{a}_i\langle y_i\rangle, \emptyset)\}_{i\in I} \cup \left\{(\overline{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J}\text{-})$ with

- $\widetilde{a}, \widetilde{b}, \widetilde{c}$ are pairwise-distinct channel names;
- $(\widetilde{a} \cup \widetilde{b}) \cap fn(A_1, A_2, \widetilde{c}) = \emptyset$;
- $\widetilde{a} = \{a_i\}_{i\in I}$ and $\widetilde{b} = \{b_j\}_{j\in J}$ and $\widetilde{c} = \{c_j\}_{j\in J}$;
- $\widetilde{y} \subseteq dom(A_1)$ and $\widetilde{y} = \{y_i\}_{i\in I}$.

We will prove $\mathcal{R} \subseteq \approx_l$. Note that this is sufficient for proving $\approx \subseteq \approx_l$, since if $A_1 \approx A_2$ then $(A_1, A_2) \in \mathcal{R}$ (by letting $\widetilde{a} = \widetilde{b} = \widetilde{c} = \widetilde{y} = \emptyset$) and from $\mathcal{R} \subseteq \approx_l$ we know $A_1 \approx_l A_2$. The reason we need to introduce the context $\mathcal{C}$ and remove variables $\widetilde{y}$ is that the labelled transition $\xrightarrow{\nu c.\overline{a}\langle c\rangle}$ makes the bound name c become free and the transition $\xrightarrow{\nu x.\overline{a}\langle x\rangle}$ generates a new substitution for term M. These cannot happen in internal transitions $\xrightarrow{\tau}$ when considering observational equivalence. To simulate outputting a bound name and a term, we store their values by output actions $\overline{a}_i\langle y_i\rangle$ and $\overline{b}_j\langle c_j\rangle$ and remove the corresponding variables y_i from the substitution. The attacker can refer to these values by using a corresponding input action $a_i(x)$ and $b_j(z)$.

To show $\mathcal{R} \subseteq \approx_l$, assume $(A_1, A_2) \in \mathcal{R}$ because of $\mathcal{C}[A_1]_{\setminus\widetilde{y}} \approx \mathcal{C}[A_2]_{\setminus\widetilde{y}}$ where $\mathcal{C}, \widetilde{y}$ are stated as above. We shall prove the static equivalence $A_1 \approx_s A_2$, and if $A_1 \xrightarrow{\alpha} A_1'$ for some A_1' then there exists A_2' such that $A_2 \stackrel{\widehat{\alpha}}{\Longrightarrow} A_2'$ and $(A_1', A_2') \in \mathcal{R}$.

1. **First we prove that A_1 and A_2 are statically equivalent, i.e., $A_1 \approx_s A_2$.** According to the definition of static equivalence, consider two terms N_1, N_2 with $var(N_1, N_2) \subseteq dom(A_1)$ and let $A_k = \nu\widetilde{n}_k.(\sigma_k, S_k, \mathcal{P}_k)$ with $k = 1, 2$ for some $\widetilde{n}_1, \widetilde{n}_2$ which do not occur in N_1, N_2. Assume $N_1\sigma_1 =_\Sigma N_2\sigma_1$, we shall prove that $N_1\sigma_2 =_\Sigma N_2\sigma_2$. The idea of the proof is to construct a context $\mathcal{C}'$ for testing whether $N_1 = N_2$ and then applying this context to $\mathcal{C}[A_1]_{\setminus\widetilde{y}}$ and $\mathcal{C}[A_2]_{\setminus\widetilde{y}}$ to see if they behave in the same way. Although $\widetilde{y}$ are removed in $\mathcal{C}[A_1]_{\setminus\widetilde{y}}$ and $\mathcal{C}[A_2]_{\setminus\widetilde{y}}$, the values of $\widetilde{y}$ are actually stored in $\overline{a}_i\langle y_i\rangle$ for $i \in I$ in the context $\mathcal{C}$. Hence we can get these values by performing input actions on channel a_i with $i \in I$. Selecting a fresh channel name d, we first construct the following plain process P_c:

$$P_c = a_1(x_1).a_2(x_2).\cdots.a_{|I|}(x_{|I|}).\texttt{if } N_1\{x_i/y_i\}_{i\in I} = N_2\{x_i/y_i\}_{i\in I} \texttt{ then } \overline{d}$$

Then we construct an evaluation context $\mathcal{C}' = (\text{-}, \text{-}, \{(P_c, \emptyset)\}\ \text{-})$ and apply it to $\mathcal{C}[A_1]_{\setminus\widetilde{y}}$ and have

$$\begin{aligned}
&\mathcal{C}'[\mathcal{C}[A_1]_{\setminus\widetilde{y}}] = \\
&\nu\widetilde{c},\widetilde{n}_1.\Big(\sigma_{1\setminus\widetilde{y}}, S_1, \mathcal{P}_1 \cup \{(\boldsymbol{\overline{a}_i\langle y_i\sigma_1\rangle}, \emptyset)\}_{i\in I} \cup \big\{(\overline{b}_j\langle c_j\rangle, \emptyset)\big\}_{j\in J} \cup \big\{(\boldsymbol{P_c\sigma_{1\setminus\widetilde{y}}}, \emptyset)\big\})\Big) \\
&\Longrightarrow \nu\widetilde{c},\widetilde{n}_1.\left(\sigma_{1\setminus\widetilde{y}}, S_1, \begin{array}{c}\mathcal{P}_1 \cup \big\{(\overline{b}_j\langle c_j\rangle, \emptyset)\big\}_{j\in J} \\ \cup \left\{\left(\begin{array}{l}\texttt{if } \boldsymbol{(N_1\sigma_{1\setminus\widetilde{y}})\{y_i\sigma_1/y_i\}_{i\in I}} \\ \quad = \boldsymbol{(N_2\sigma_{1\setminus\widetilde{y}})\{y_i\sigma_1/y_i\}_{i\in I}} \texttt{ then } \boldsymbol{\overline{d}}\end{array}, \emptyset\right)\right\}\end{array}\right)
\end{aligned}$$

It is clear that $(N_1\sigma_{1\setminus\widetilde{y}})\{y_i\sigma_1/y_i\}_{i\in I} = N_1\sigma_1 =_\Sigma N_2\sigma_1 = (N_2\sigma_{1\setminus\widetilde{y}})\{y_i\sigma_1/y_i\}_{i\in I}$, thus the conditional branch jumps to `then` and we can see that $\mathcal{C}'[\mathcal{C}[A_1]_{\setminus\widetilde{y}}] \Downarrow_d$. Since $\mathcal{C}[A_1]_{\setminus\widetilde{y}} \approx \mathcal{C}[A_2]_{\setminus\widetilde{y}}$ and the equivalence should be closed under any closing evaluation context, it should hold that $\mathcal{C}'[\mathcal{C}[A_2]_{\setminus\widetilde{y}}] \Downarrow_d$ and that means

$$\begin{aligned}
&\mathcal{C}'[\mathcal{C}[A_2]_{\setminus\widetilde{y}}] = \\
&\quad \nu\widetilde{c},\widetilde{n}_2.(\sigma_{2\setminus\widetilde{y}}, S_2, \mathcal{P}_2 \cup \{(\boldsymbol{\overline{a}_i\langle y_i\sigma_2\rangle}, \emptyset)\}_{i\in I} \cup \big\{(\overline{b}_j\langle c_j\rangle, \emptyset)\big\}_{j\in J} \cup \big\{(\boldsymbol{P_c\sigma_{2\setminus\widetilde{y}}}, \emptyset)\big\})
\end{aligned}$$

$$\Longrightarrow$$

$$\nu\widetilde{c},\widetilde{n}_2,\widetilde{m}'.\left(\sigma_{2\backslash\widetilde{y}}, S_2', \begin{array}{l}\mathcal{P}_2' \cup \left\{(\bar{b}_j\langle c_j\rangle,\emptyset)\right\}_{j\in J} \\ \cup \left\{ \left(\begin{array}{r} \texttt{if } \boldsymbol{(N_1\sigma_{2\backslash\widetilde{y}})\,\{y_i\sigma_2/y_i\}_{i\in I}} \\ \boldsymbol{= (N_2\sigma_{2\backslash\widetilde{y}})\,\{y_i\sigma_2/y_i\}_{i\in I}} \texttt{ then } \boldsymbol{\overline{d}} \end{array}, \emptyset \right) \right\}\end{array}\right)$$

$$\Longrightarrow \nu\widetilde{c},\widetilde{n}_2,\widetilde{m}''.(\sigma_{2\backslash\widetilde{y}}, S_2'', \mathcal{P}_2'' \cup \left\{(\bar{b}_j\langle c_j\rangle,\emptyset)\right\}_{j\in J} \cup \{\boldsymbol{(\overline{d},\emptyset)}\})$$

This requires $(N_1\sigma_{2\backslash\widetilde{y}})\,\{y_i\sigma_2/y_i\}_{i\in I} =_\Sigma (N_2\sigma_{2\backslash\widetilde{y}})\,\{y_i\sigma_2/y_i\}_{i\in I}$. From $N_k\sigma_2 = (N_k\sigma_{2\backslash\widetilde{y}})\,\{y_i\sigma_2/y_i\}_{i\in I}$ for $k = 1, 2$, we have $N_1\sigma_2 =_\Sigma N_2\sigma_2$. Hence $A_1 \approx_s A_2$.

2. **Now we proceed to show the behavioural equivalence between A_1 and A_2.** Assume $A_1 \xrightarrow{\alpha} A_1'$ for some A_1' then there exists A_2' such that $A_2 \overset{\widehat{\alpha}}{\Longrightarrow} A_2'$ and $(A_1', A_2') \in \mathcal{R}$.

(a) Assume $A_1 = \nu\widetilde{n}_1.(\sigma_1, S_1, \mathcal{P}_1) \xrightarrow{\tau} A_1' = \nu\widetilde{n}_1'.(\sigma_1, S_1', \mathcal{P}_1')$ for some $\widetilde{n}_1', S_1', \mathcal{P}_1'$. Using Corollary 7, we have

$$\mathcal{C}[A_1]_{\backslash\widetilde{y}} = \nu\widetilde{c},\widetilde{n}_1.(\sigma_{1\backslash\widetilde{y}}, S_1, \mathcal{P}_1 \cup \{(\overline{a}_i\langle y_i\sigma_1\rangle,\emptyset)\}_{i\in I} \cup \left\{(\bar{b}_j\langle c_j\rangle,\emptyset)\right\}_{j\in J})$$
$$\xrightarrow{\tau} \mathcal{C}[A_1']_{\backslash\widetilde{y}} = \nu\widetilde{c},\widetilde{n}_1'.(\sigma_{1\backslash\widetilde{y}}, S_1', \mathcal{P}_1' \cup \{(\overline{a}_i\langle y_i\sigma_1\rangle,\emptyset)\}_{i\in I} \cup \left\{(\bar{b}_j\langle c_j\rangle,\emptyset)\right\}_{j\in J})$$

Since $\mathcal{C}[A_1]_{\backslash\widetilde{y}} \approx \mathcal{C}[A_2]_{\backslash\widetilde{y}}$, there exists B such that $\mathcal{C}[A_2]_{\backslash\widetilde{y}} \Longrightarrow B \approx \mathcal{C}[A_1']_{\backslash\widetilde{y}}$. Since $\mathcal{C}[A_1']_{\backslash\widetilde{y}} \Downarrow_{a_i,b_j}$, it has to be $B \Downarrow_{a_i,b_j}$. Since a_i, b_j do not occur in A_1, A_2, these outputs $\overline{a}_i\langle y_i\rangle, \bar{b}_j\langle c_j\rangle$ are not involved in the transitions $\mathcal{C}[A_2]_{\backslash\widetilde{y}} \Longrightarrow B$. Thus the only possibility for B is that $B = \nu\widetilde{c},\widetilde{n}_2'.(\sigma_{2\backslash\widetilde{y}}, S_2', \mathcal{P}_2' \cup \{(\overline{a}_i\langle y_i\sigma_2\rangle,\emptyset)\}_{i\in I} \cup \left\{(\bar{b}_j\langle c_j\rangle,\emptyset)\right\}_{j\in J})$ for some $\widetilde{n}_2', S_2', \mathcal{P}_2'$. Let $A_2' = \nu\widetilde{n}_2,\widetilde{n}_2'.(\sigma_2, S_2', \mathcal{P}_2')$, then $A_2 \Longrightarrow A_2'$ and $\mathcal{C}[A_2']_{\backslash\widetilde{y}} = B$. From $\mathcal{C}[A_1']_{\backslash\widetilde{y}} \approx \mathcal{C}[A_2']_{\backslash\widetilde{y}}$, we have $(A_1', A_2') \in \mathcal{R}$.

(b) Assume $A_1 = \nu\widetilde{n}_1.(\sigma_1, S_1, \mathcal{P}_1' \cup \{(\overline{a}\langle e\rangle.P, L)\}) \xrightarrow{\overline{a}\langle e\rangle} A_1' = \nu\widetilde{n}_1.(\sigma_1, S_1, \mathcal{P}_1' \cup \{(P, L)\})$ when $a, e \notin \widetilde{n}$. The proof is divided into four cases, according to whether a, e occur in $\widetilde{c}$. If a, e are free names, they can be used directly. But if a, e are bounded by $\widetilde{c}$, we cannot directly refer to them. But the names in $\widetilde{c}$ are stored in the output actions $\bar{b}_j\langle c_j\rangle$ for $j \in J$. Hence we can get these bound names by using an additional input action on b_j in the context.

i. We start by analysing the simplest case when $a, e \notin \widetilde{c}$. In this case, we can directly use a, e in the context. Let $\mathcal{C}' = (\text{-}, \text{-}, \left\{(\overline{d},\emptyset)\right\} \cup \{(a(x).\mathit{if}\ x = e\ \mathit{then}\ d, \emptyset)\}\ \text{-})$, where d is fresh. Applying $\mathcal{C}'$ to $\mathcal{C}[A_1]_{\backslash\widetilde{y}}$, we can see that

$$\mathcal{C}'[\mathcal{C}[A_1]_{\backslash\widetilde{y}}]$$
$$= \nu\widetilde{c},\widetilde{n}_1.\left(\sigma_1, S_1, \begin{array}{l}\mathcal{P}_1' \cup \{(\overline{a}_i\langle y_i\sigma_1\rangle,\emptyset)\}_{i\in I} \cup \left\{(\bar{b}_j\langle c_j\rangle,\emptyset)\right\}_{j\in J} \\ \cup \{(\boldsymbol{\overline{a}\langle e\rangle.P}, L)\}, (\overline{d},\emptyset), (\boldsymbol{a(x)}.\texttt{if } \boldsymbol{x = e} \texttt{ then } \boldsymbol{d}, \emptyset)\end{array}\right)$$
$$\xrightarrow{\tau}$$
$$\nu\widetilde{c},\widetilde{n}_1.\left(\sigma_1, S_1, \begin{array}{l}\mathcal{P}_1' \cup \{(\overline{a}_i\langle y_i\sigma_1\rangle,\emptyset)\}_{i\in I} \\ \cup \left\{(\bar{b}_j\langle c_j\rangle,\emptyset)\right\}_{j\in J} \cup \left\{(P, L), (\boldsymbol{\overline{d}},\emptyset), (\texttt{if } \boldsymbol{e = e} \texttt{ then } \boldsymbol{d}, \emptyset)\right\}\end{array}\right)$$
$$\Longrightarrow B_1 = \nu\widetilde{c},\widetilde{n}_1.\left(\sigma_1, S_1, \mathcal{P}_1' \cup \{(\overline{a}_i\langle y_i\sigma_1\rangle,\emptyset)\}_{i\in I} \cup \left\{(\bar{b}_j\langle c_j\rangle,\emptyset)\right\}_{j\in J} \cup \{(P, L)\}\right)$$

Since $\mathcal{C}[A_1]_{\backslash\widetilde{y}} \approx \mathcal{C}[A_2]_{\backslash\widetilde{y}}$ and $\approx$ is closed under evaluation contexts, we know that $\mathcal{C}'[\mathcal{C}[A_1]_{\backslash\widetilde{y}}] \approx \mathcal{C}'[\mathcal{C}[A_2]_{\backslash\widetilde{y}}]$. Then there exists B_2 such that

$$\mathcal{C}'[\mathcal{C}[A_2]_{\backslash\widetilde{y}}] \Longrightarrow B_2 \approx B_1$$

For $i \in I, j \in J$, we know that $B_1 \Downarrow_{a_i,b_j}$ and $B_1 \not\Downarrow_d$. Thus it should be $B_2 \Downarrow_{a_i,b_j}$ and $B_2 \not\Downarrow_d$. Since a is different from a_i, b_j and a_i, b_j do not occur in A_1, A_2, the only possibility for the transitions $\mathcal{C}'[\mathcal{C}[A_2]_{\backslash\widetilde{y}}] \Longrightarrow B_2$ is that

$$\begin{aligned}
&\mathcal{C}'[\mathcal{C}[A_2]_{\backslash\widetilde{y}}] = \nu\widetilde{c}, \widetilde{n}_2.\left(\sigma_2, \boldsymbol{S_2}, \begin{array}{c}\boldsymbol{\mathcal{P}_2} \cup \{(\overline{a}_i\langle y_i\sigma_2\rangle, \emptyset)\}_{i\in I} \cup \left\{(\overline{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J} \\ \cup \left\{(\overline{d}, \emptyset), (a(x).\texttt{if } x = e \texttt{ then } d, \emptyset)\right\}\end{array}\right) \\
&\Longrightarrow \nu\widetilde{c}, \widetilde{n}_2, \widetilde{m}.\left(\sigma_2, \boldsymbol{S_2'}, \begin{array}{c}\boldsymbol{\mathcal{P}_2'} \cup \{(\overline{a}_i\langle y_i\sigma_2\rangle, \emptyset)\}_{i\in I} \cup \left\{(\overline{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J} \\ \cup \left\{(\overline{d}, \emptyset), \boldsymbol{(a(x).\texttt{if } x = e \texttt{ then } d}, \emptyset)\right\}\end{array}\right) \\
&\xrightarrow{\tau} \nu\widetilde{c}, \widetilde{n}_2, \widetilde{m}.\left(\sigma_2, \boldsymbol{S_2'}, \begin{array}{c}\boldsymbol{\mathcal{P}_2''} \cup \{(\overline{a}_i\langle y_i\sigma_2\rangle, \emptyset)\}_{i\in I} \cup \left\{(\overline{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J} \\ \cup \left\{(\overline{d}, \emptyset), \boldsymbol{(\texttt{if } e = e \texttt{ then } d}, \emptyset)\right\}\end{array}\right) \\
&\Longrightarrow \nu\widetilde{c}, \widetilde{n}_2, \widetilde{m}'.\left(\sigma_2, \boldsymbol{S_2''}, \begin{array}{c}\boldsymbol{\mathcal{P}_2'''} \cup \{(\overline{a}_i\langle y_i\sigma_2\rangle, \emptyset)\}_{i\in I} \cup \left\{(\overline{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J} \\ \cup \left\{(\overline{d}, \emptyset), (\texttt{if } e = e \texttt{ then } d, \emptyset)\right\}\end{array}\right) \\
&\xrightarrow{\tau} \nu\widetilde{c}, \widetilde{n}_2, \widetilde{m}'.\left(\sigma_2, S_2'', \begin{array}{c}\boldsymbol{\mathcal{P}_2'''} \cup \{(\overline{a}_i\langle y_i\sigma_2\rangle, \emptyset)\}_{i\in I} \cup \left\{(\overline{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J} \\ \cup \left\{(\overline{d}, \emptyset), (\boldsymbol{d}, \emptyset)\right\}\end{array}\right) \\
&\Longrightarrow \nu\widetilde{c}, \widetilde{n}_2, \widetilde{m}''.\left(\sigma_2, S_2''', \begin{array}{c}\boldsymbol{\mathcal{P}_2^{(4)}} \cup \{(\overline{a}_i\langle y_i\sigma_2\rangle, \emptyset)\}_{i\in I} \cup \left\{(\overline{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J} \\ \cup \left\{(\boldsymbol{\overline{d}}, \emptyset), (\boldsymbol{d}, \emptyset)\right\}\end{array}\right) \\
&\xrightarrow{\tau} \nu\widetilde{c}, \widetilde{n}_2, \widetilde{m}''.\left(\sigma_2, \boldsymbol{S_2''', \mathcal{P}_2^{(4)}} \cup \{(\overline{a}_i\langle y_i\sigma_2\rangle, \emptyset)\}_{i\in I} \cup \left\{(\overline{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J}\right) \\
&\Longrightarrow B_2 = \nu\widetilde{c}, \widetilde{n}_2, \widetilde{m}'''.\left(\sigma_2, \boldsymbol{S_2^{(4)}, \mathcal{P}_2^{(5)}} \cup \{(\overline{a}_i\langle y_i\sigma_2\rangle, \emptyset)\}_{i\in I} \cup \left\{(\overline{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J}\right)
\end{aligned}$$

Let $A_2' = \nu\widetilde{n}_2, \widetilde{m}'''.(\sigma_2, S_2^{(4)}, \mathcal{P}_2^{(5)})$. We can easily verify that $\mathcal{C}[A_2']_{\backslash\widetilde{y}} = B_2$. Since the outputs $\overline{a}_i\langle y_i\rangle, \overline{b}_j\langle c_j\rangle$ are not involved in the transitions, we have

$$\begin{aligned}
&A_2 \Longrightarrow \nu\widetilde{n}_2, \widetilde{m}.(\sigma_2, S_2', \mathcal{P}_2') \xrightarrow{\overline{a}\langle e\rangle} \nu\widetilde{n}_2, \widetilde{m}.(\sigma_2, S_2', \mathcal{P}_2'') \Longrightarrow \nu\widetilde{n}_2, \widetilde{m}'.(\sigma_2, S_2'', \mathcal{P}_2^{(3)}) \\
&\quad \Longrightarrow \nu\widetilde{n}_2, \widetilde{m}''.(\sigma_2, S_2^{(3)}, \mathcal{P}_2^{(4)}) \Longrightarrow A_2' = \nu\widetilde{n}_2, \widetilde{m}'''.(\sigma_2, S_2^{(4)}, \mathcal{P}_2^{(5)})
\end{aligned}$$

Hence $A_1 \xrightarrow{\overline{a}\langle e\rangle} A_1'$, $A_2 \stackrel{\overline{a}\langle e\rangle}{\Longrightarrow} A_2'$ and $\mathcal{C}[A_1']_{\backslash\widetilde{y}} \approx \mathcal{C}[A_2']_{\backslash\widetilde{y}}$. Then $(A_1', A_2') \in \mathcal{R}$.

ii. If $a = c_k$ for some $k \in J$ and $e \notin \widetilde{c}$, let

$$\mathcal{C}' = (\text{-}, \text{-}, \left\{(\overline{d}, \emptyset), (b_k(u).u(x).\texttt{if } x = e \texttt{ then } d.\overline{b}_k\langle u\rangle, \emptyset)\right\} \text{-})$$

where d is fresh. Note that each time we consume a $\overline{b}_j\langle u\rangle$, we need to generate a new one since we require each name in $\widetilde{c}$ has an output action.

$$\mathcal{C}'[\mathcal{C}[A_1]_{\backslash\widetilde{y}}] =$$

$$\nu\widetilde{c},\widetilde{n}_1.\left(\sigma_1, S_1, \begin{array}{c}\mathcal{P}'_1 \cup \{(\overline{a}_i\langle y_i\sigma_1\rangle,\emptyset)\}_{i\in I} \cup \left\{(\boldsymbol{\overline{b}_j\langle c_j\rangle},\emptyset)\right\}_{j\in J} \cup \left\{(\overline{a}\langle e\rangle.P, L),(\overline{d},\emptyset)\right\}\\ \cup\left\{(\boldsymbol{b_k(u).u(x).}\texttt{if } \boldsymbol{x=e} \texttt{ then } \boldsymbol{d.\overline{b}_k\langle u\rangle},\emptyset)\right\}\end{array}\right)$$

$$\xrightarrow{\tau}$$

$$\nu\widetilde{c},\widetilde{n}_1.\left(\sigma_1, S_1, \begin{array}{c}\mathcal{P}'_1 \cup \{(\overline{a}_i\langle y_i\sigma_1\rangle,\emptyset)\}_{i\in I} \cup \left\{\boldsymbol{(\overline{b}_j\langle c_j\rangle,\emptyset)}\right\}_{\boldsymbol{j\in J\backslash k}}\\ \cup\left\{(\boldsymbol{\overline{a}\langle e\rangle.P}, L),(\overline{d},\emptyset)\right\} \cup \left\{(\boldsymbol{a(x).}\texttt{if } \boldsymbol{x=e} \texttt{ then } \boldsymbol{d.\overline{b}_k\langle a\rangle},\emptyset)\right\}\end{array}\right)$$

$$\Longrightarrow B_1 = \nu\widetilde{c},\widetilde{n}_1.\left(\sigma_1, S_1, \mathcal{P}'_1 \cup \{(\overline{a}_i\langle y_i\sigma_1\rangle,\emptyset)\}_{i\in I} \cup \left\{\boldsymbol{(\overline{b}_j\langle c_j\rangle,\emptyset)}_{j\in J}\right\} \cup \{(P,L)\}\right)$$

We can easily verify that $B_1 = \mathcal{C}[A'_1]_{\backslash\widetilde{y}}$. Since $\mathcal{C}'[\mathcal{C}[A_1]_{\backslash\widetilde{y}}] \approx \mathcal{C}'[\mathcal{C}[A_2]_{\backslash\widetilde{y}}]$, there exists B_2 such that

$$\mathcal{C}'[\mathcal{C}[A_2]_{\backslash\widetilde{y}}] \Longrightarrow B_2 \approx B_1$$

From $B_1 \Downarrow_{a_i,b_j}$ and $B_1 \not\Downarrow_d$, we should also have $B_2 \Downarrow_{a_i,b_j}$ and $B_2 \not\Downarrow_d$. Thus the only possibility for the transitions $\mathcal{C}'[\mathcal{C}[A_2]_{\backslash\widetilde{y}}] \Longrightarrow B_2$ are:

$$\mathcal{C}'[\mathcal{C}[A_2]_{\backslash\widetilde{y}}]$$

$$= \nu\widetilde{c},\widetilde{n}_2.\left(\sigma_2, \boldsymbol{S_2}, \begin{array}{c}\boldsymbol{P_2} \cup \{(\overline{a}_i\langle y_i\sigma_2\rangle,\emptyset)\}_{i\in I} \cup \left\{(\overline{b}_j\langle c_j\rangle,\emptyset)\right\}_{j\in J} \cup \left\{(\overline{d},\emptyset)\right\}\\ \cup\left\{(b_k(u).u(x).\texttt{if } x=e \texttt{ then } d.\overline{b}_k\langle u\rangle,\emptyset)\right\}\end{array}\right)$$

$$\Longrightarrow \nu\widetilde{c},\widetilde{n}_2,\widetilde{m}.\left(\sigma_2, \boldsymbol{S'_2}, \begin{array}{c}\boldsymbol{\mathcal{P}'_2} \cup \{(\overline{a}_i\langle y_i\sigma_2\rangle,\emptyset)\}_{i\in I} \cup \left\{\boldsymbol{(\overline{b}_j\langle c_j\rangle,\emptyset)}\right\}_{\boldsymbol{j\in J}} \cup \left\{(\overline{d},\emptyset)\right\}\\ \cup\left\{(\boldsymbol{b_k(u).u(x).}\texttt{if } \boldsymbol{x=e} \texttt{ then } \boldsymbol{d.\overline{b}_k\langle u\rangle},\emptyset)\right\}\end{array}\right)$$

$$\xrightarrow{\tau} \nu\widetilde{c},\widetilde{n}_2,\widetilde{m}.\left(\sigma_2, \boldsymbol{S'_2}, \begin{array}{c}\boldsymbol{\mathcal{P}'_2} \cup \{(\overline{a}_i\langle y_i\sigma_2\rangle,\emptyset)\}_{i\in I} \cup \left\{\boldsymbol{(\overline{b}_j\langle c_j\rangle,\emptyset)}\right\}_{\boldsymbol{j\in J\backslash k}}\\ \cup\left\{(\overline{d},\emptyset),(\boldsymbol{a(x).}\texttt{if } \boldsymbol{x=e} \texttt{ then } \boldsymbol{d.\overline{b}_k\langle a\rangle},\emptyset)\right\}\end{array}\right)$$

$$\Longrightarrow \nu\widetilde{c},\widetilde{n}_2,\widetilde{m}'.\left(\sigma_2, \boldsymbol{S''_2}, \begin{array}{c}\boldsymbol{\mathcal{P}''_2} \cup \{(\overline{a}_i\langle y_i\sigma_2\rangle,\emptyset)\}_{i\in I} \cup \left\{(\overline{b}_j\langle c_j\rangle,\emptyset)\right\}_{j\in J\backslash k}\\ \cup\left\{(\overline{d},\emptyset),(a(x).\texttt{if } x=e \texttt{ then } d.\overline{b}_k\langle a\rangle,\emptyset)\right\}\end{array}\right)$$

$$\xrightarrow{\tau} \nu\widetilde{c},\widetilde{n}_2,\widetilde{m}'.\left(\sigma_2, \boldsymbol{S''_2}, \begin{array}{c}\boldsymbol{\mathcal{P}'''_2} \cup \{(\overline{a}_i\langle y_i\sigma_2\rangle,\emptyset)\}_{i\in I} \cup \left\{(\overline{b}_j\langle c_j\rangle,\emptyset)\right\}_{j\in J\backslash k}\\ \cup\left\{(\overline{d},\emptyset),(\texttt{if } \boldsymbol{e=e} \texttt{ then } (\boldsymbol{d.\overline{b}_k\langle a\rangle}),\emptyset)\right\}\end{array}\right)$$

$$\Longrightarrow \nu\widetilde{c},\widetilde{n}_2,\widetilde{m}''.\left(\sigma_2, \boldsymbol{S'''_2}, \begin{array}{c}\boldsymbol{\mathcal{P}^{(4)}_2} \cup \{(\overline{a}_i\langle y_i\sigma_2\rangle,\emptyset)\}_{i\in I} \cup \left\{(\overline{b}_j\langle c_j\rangle,\emptyset)\right\}_{j\in J\backslash k}\\ \cup\left\{(\overline{d},\emptyset),(\texttt{if } \boldsymbol{e=e} \texttt{ then } \boldsymbol{d.\overline{b}_k\langle a\rangle},\emptyset)\right\}\end{array}\right)$$

$$\xrightarrow{\tau} \nu\widetilde{c},\widetilde{n}_2,\widetilde{m}''.\left(\sigma_2, \boldsymbol{S'''_2}, \begin{array}{c}\boldsymbol{\mathcal{P}^{(4)}_2} \cup \{(\overline{a}_i\langle y_i\sigma_2\rangle,\emptyset)\}_{i\in I} \cup \left\{(\overline{b}_j\langle c_j\rangle,\emptyset)\right\}_{j\in J\backslash k}\\ \cup\left\{(\overline{d},\emptyset),(d.\overline{b}_k\langle a\rangle,\emptyset)\right\}\end{array}\right)$$

$$\Longrightarrow \nu\widetilde{c},\widetilde{n}_2,\widetilde{m}'''.\left(\sigma_2, \boldsymbol{S^{(4)}_2}, \begin{array}{c}\boldsymbol{\mathcal{P}^{(5)}_2} \cup \{(\overline{a}_i\langle y_i\sigma_2\rangle,\emptyset)\}_{i\in I} \cup \left\{(\overline{b}_j\langle c_j\rangle,\emptyset)\right\}_{j\in J\backslash k}\\ \cup\left\{(\boldsymbol{\overline{d}},\emptyset),(\boldsymbol{d.\overline{b}_k\langle a\rangle},\emptyset)\right\}\end{array}\right)$$

$$\xrightarrow{\tau} \nu\widetilde{c},\widetilde{n}_2,\widetilde{m}'''.\left(\sigma_2, \boldsymbol{S^{(4)}_2}, \boldsymbol{\mathcal{P}^{(5)}_2} \cup \{(\overline{a}_i\langle y_i\sigma_2\rangle,\emptyset)\}_{i\in I} \cup \left\{\boldsymbol{(\overline{b}_j\langle c_j\rangle,\emptyset)}\right\}_{\boldsymbol{j\in J}}\right)$$

$$\Longrightarrow B_2 = \nu\widetilde{c},\widetilde{n}_2,\widetilde{m}^{(4)}.\left(\sigma_2, \boldsymbol{S^{(5)}_2}, \boldsymbol{\mathcal{P}^{(6)}_2} \cup \{(\overline{a}_i\langle y_i\sigma_2\rangle,\emptyset)\}_{i\in I} \cup \left\{(\overline{b}_j\langle c_j\rangle,\emptyset)\right\}_{j\in J}\right)$$

Let $A_2' = \nu\widetilde{n}_2, \widetilde{m}^{(4)}.(\sigma_2, S_2^{(5)}, \mathcal{P}_2^{(6)})$. We can easily verify that $\mathcal{C}[A_2']_{\backslash\widetilde{y}} = B_2$. And we have the following transitions from A_2 to A_2':

$$\begin{aligned}
A_2 &\Longrightarrow \nu\widetilde{n}_2, \widetilde{m}.(\sigma_2, S_2', \mathcal{P}_2') \Longrightarrow \nu\widetilde{n}_2, \widetilde{m}'.(\sigma_2, S_2'', \mathcal{P}_2'') \\
&\xrightarrow{\overline{a}\langle e\rangle} \nu\widetilde{n}_2, \widetilde{m}'.(\sigma_2, S_2'', \mathcal{P}_2''') \Longrightarrow \nu\widetilde{n}_2, \widetilde{m}''.(\sigma_2, S_2''', \mathcal{P}_2^{(4)}) \\
&\Longrightarrow \nu\widetilde{n}_2, \widetilde{m}'''.(\sigma_2, S_2^{(4)}, \mathcal{P}_2^{(5)}) \Longrightarrow A_2' = \nu\widetilde{n}_2, \widetilde{m}^{(4)}.(\sigma_2, S_2^{(5)}, \mathcal{P}_2^{(6)})
\end{aligned}$$

Hence $A_1 \xrightarrow{\overline{a}\langle e\rangle} A_1'$, $A_2 \overset{\overline{a}\langle e\rangle}{\Longrightarrow} A_2'$ and $\mathcal{C}[A_1']_{\backslash\widetilde{y}} \approx \mathcal{C}[A_2']_{\backslash\widetilde{y}}$. Then $(A_1', A_2') \in \mathcal{R}$.

iii. If $e = c_k$ with $k \in J$ and $a \notin \widetilde{c}$, let

$$\mathcal{C}' = (\text{-}, \text{-}, \left\{(\overline{d}, \emptyset), (b_k(v).a(x).\mathtt{if}\ x = v\ \mathtt{then}\ d.\overline{b}_k\langle v\rangle, \emptyset)\right\} \text{-})$$

where d is fresh. The rest of analysis is similar as above.

iv. If $a = e = c_k$ with $k \in J$, let

$$\mathcal{C}' = (\text{-}, \text{-}, \left\{(\overline{d}, \emptyset), (b_k(u).u(x).\mathtt{if}\ x = u\ \mathtt{then}\ d.\overline{b}_k\langle u\rangle, \emptyset)\right\} \text{-})$$

where d is fresh. The rest of analysis is similar as above.

v. If $a = c_j$ and $e = c_k$ with $j \neq k$ and $j, k \in J$, let

$$\mathcal{C}' = (\text{-}, \text{-}, \left\{(\overline{d}, \emptyset), (b_j(u).b_k(v).u(x).\mathtt{if}\ x = v\ \mathtt{then}\ (d.\overline{b}_j\langle u\rangle \mid \overline{b}_k\langle v\rangle), \emptyset)\right\} \text{-})$$

where d is fresh. The rest of analysis is similar as above.

(c) α is a base input $a(M)$. Assume $A_1 = \nu\widetilde{n}_1.(\sigma_1, S_1, \mathcal{P}_1' \cup \{(a(x).P, L)\}) \xrightarrow{a(M)} A_1' = \nu\widetilde{n}_1.(\sigma_1, S_1, \mathcal{P}_1' \cup \{(P\{M\sigma_1/x\}, L)\})$ and $fv(M) \subseteq dom(\sigma_1)$.

i. If $a \notin \widetilde{c}$, let $\pi := a_1(x_1).a_2(x_2).\cdots.a_{|I|}(x_{|I|})$ and consider the evaluation context

$$\mathcal{C}' = \left(\text{-}, \text{-}, \left\{ \left(\prod_{i\in I} \overline{d}_i, \emptyset\right), \left(\pi.\overline{a}\langle M\{x_i/y_i\}_{i\in I}\rangle. \left(\prod_{i\in I} d_i.\overline{a}_i\langle x_i\rangle\right), \emptyset\right) \right\} \text{-}\right)$$

where $\{d_i\}_{i\in I}$ are fresh. Note that the use of d_i is to make sure $(\prod_{i\in I} d_i.\overline{a}_i\langle x_i\rangle, \emptyset)$ will be split into $\{(\overline{a}_i\langle x_i\rangle, \emptyset)\}_{i\in I}$. Applying $\mathcal{C}'$ to $\mathcal{C}[A_1]_{\backslash\widetilde{y}}$, we can see that

$$\begin{aligned}
&\mathcal{C}'[\mathcal{C}[A_1]_{\backslash\widetilde{y}}] \Longrightarrow \\
&\qquad B_1 := \nu\widetilde{c}, \widetilde{n}_1. \left(\sigma_1, S_1, \begin{matrix} \mathcal{P}_1' \cup \{(\overline{a}_i\langle y_i\sigma_1\rangle, \emptyset)\}_{i\in I} \cup \left\{(\overline{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J} \\ \cup\, \{(P\{M\sigma_1/x\}, L)\} \end{matrix}\right)
\end{aligned}$$

We can verify that $\mathcal{C}[A_1']_{\backslash\widetilde{y}} = B_1$. Similarly we have $\mathcal{C}'[\mathcal{C}[A_1]_{\backslash\widetilde{y}}] \approx \mathcal{C}'[\mathcal{C}[A_2]_{\backslash\widetilde{y}}]$. Then there exists B_2 such that

$$\mathcal{C}'[\mathcal{C}[A_2]_{\backslash\widetilde{y}}] \Longrightarrow B_2 \approx B_1$$

Since $\mathcal{C}'[\mathcal{C}[A_1]_{\backslash\widetilde{y}}] \Downarrow_{a_i,b_j,d_i}$ and $B_1 \Downarrow_{a_i,b_j}$ but $B_1 \not\Downarrow_{d_i}$, it should be that $B_2 \Downarrow_{a_i,b_j}$ but $B_2 \not\Downarrow_{d_i}$. Hence the only possibility of $\mathcal{C}'[\mathcal{C}[A_2]_{\backslash\widetilde{y}}] \Longrightarrow B_2$ is that

$$\begin{aligned}
&\mathcal{C}'[\mathcal{C}[A_2]_{\backslash\widetilde{y}}] \\
&= \nu\widetilde{c}, \widetilde{n}_2. \left(\sigma_2, S_2, \begin{array}{c} \mathcal{P}_2 \cup \{(\overline{a}_i\langle y_i\sigma_2\rangle, \emptyset)\}_{i\in I} \cup \left\{(\bar{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J} \\ \cup \left\{ (\prod_{i\in I} \overline{d}_i, \emptyset), (\pi.\overline{a}\langle M\,\{x_i/y_i\}_{i\in I}\rangle. \prod_{i\in I} d_i.\overline{a}_i\langle x_i\rangle, \emptyset) \right\} \end{array} \right) \\
&\Longrightarrow B_2 := \nu\widetilde{c}, \widetilde{n}'_2. \left(\sigma_2, S'_2, \mathcal{P}'_2 \cup \{(\overline{a}_i\langle y_i\sigma_2\rangle, \emptyset)\}_{i\in I} \cup \left\{(\bar{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J}\right)
\end{aligned}$$

Let $A'_2 = \nu\widetilde{n}'_2.(\sigma_2, S'_2, \mathcal{P}'_2)$. We can easily verify that $\mathcal{C}[A'_2]_{\backslash\widetilde{y}} = B_2$. Then we have

$$A_2 = \nu\widetilde{n}_2.(\sigma_2, S_2, \mathcal{P}_2) \Longrightarrow A'_2 = \nu\widetilde{n}'_2.(\sigma_2, S'_2, \mathcal{P}'_2)$$

Since $\mathcal{C}[A'_1]_{\backslash\widetilde{y}} \approx \mathcal{C}[A'_2]_{\backslash\widetilde{y}}$, we have $(A'_1, A'_2) \in \mathcal{R}$.

ii. If $a = c_j$ for some $j \in J$, let $\pi := a_1(x_1).a_2(x_2).\cdots.a_{|I|}(x_{|I|})$ and

$$\mathcal{C}' = \left(\text{-}, \text{-}, \left\{ (\prod_{i\in I} \overline{d}_i, \emptyset), (\pi.b_j(u).\overline{u}\langle M\,\{x_i/y_i\}_{i\in I}\rangle.(\bar{b}_j\langle u\rangle \mid \prod_{i\in I} d_i.\overline{a}_i\langle x_i\rangle), \emptyset) \right\} \text{-} \right)$$

where $\{d_i\}_{i\in I}$ are fresh channel names. The analysis is similar as above.

(d) α is an input $a(e)$ of channel name e. We require that $a_i, b_j \notin fn(\widetilde{n}_1, \widetilde{n}_2, \widetilde{c}, A_1, A_2)$. The arbitrary input value e may be one of a_i, b_j and thus may violate this condition in the subsequent processes. In that case, we can choose a fresh name d to replace e in $\mathcal{C}$ and obtain a new equivalence $\mathcal{C}\,\{d/e\}\,[A_1]_{\backslash\widetilde{y}} \approx \mathcal{C}\,\{d/e\}\,[A_2]_{\backslash\widetilde{y}}$. Hence, for simplicity, we can safely assume that no conflict is introduced by e. Note that we treat the input of the channel name in a separate case because the channel names are different from base terms. When the input is a base term M, M can contain variables defined in σ, thus we need to use variables from σ when constructing context C. But when the input is a channel name, we don't need anything from σ. Assume $A_1 = \nu\widetilde{n}_1.(\sigma_1, S_1, \mathcal{P}'_1 \cup \{(a(x).P, L)\}) \xrightarrow{a(e)} A'_1 = \nu\widetilde{n}_1.(\sigma_1, S_1, \mathcal{P}'_1 \cup \{(P\,\{e/x\}\,, L)\})$. Similarly,

i. If $a, e \notin \widetilde{c}$, consider the evaluation context $\mathcal{C}' = (\text{-}, \text{-}, \left\{(\overline{d}, \emptyset), (\overline{a}\langle e\rangle.d, \emptyset)\right\} \text{-})$ where d is fresh. Applying $\mathcal{C}'$ to $\mathcal{C}[A_1]_{\backslash\widetilde{y}}$, we can see that

$$\begin{aligned}
&\mathcal{C}'[\mathcal{C}[A_1]_{\backslash\widetilde{y}}] \Longrightarrow \\
&\qquad B_1 := \nu\widetilde{c}, \widetilde{n}_1. \left(\sigma_1, S_1, \begin{array}{c} \mathcal{P}'_1 \cup \{(\overline{a}_i\langle y_i\sigma_1\rangle, \emptyset)\}_{i\in I} \cup \left\{(\bar{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J} \\ \cup \{(P\,\{e/x\}\,, L)\} \end{array} \right)
\end{aligned}$$

We can verify that $\mathcal{C}[A'_1]_{\backslash\widetilde{y}} = B_1$. Similarly we have $\mathcal{C}'[\mathcal{C}[A_1]_{\backslash\widetilde{y}}] \approx \mathcal{C}'[\mathcal{C}[A_2]_{\backslash\widetilde{y}}]$. Then there exists B_2 such that

$$\mathcal{C}'[\mathcal{C}[A_2]_{\backslash\widetilde{y}}] \Longrightarrow B_2 \approx B_1$$

Since $\mathcal{C}'[\mathcal{C}[A_1]_{\setminus\widetilde{y}}] \Downarrow_{a_i,b_j,d}$ and $B_1 \Downarrow_{a_i,b_j}$ but $B_1 \not\Downarrow_d$, it should be that $B_2 \Downarrow_{a_i,b_j}$ but $B_2 \not\Downarrow_d$. Hence the only possibility of $\mathcal{C}'[\mathcal{C}[A_2]_{\setminus\widetilde{y}}] \Longrightarrow B_2$ is that

$$\mathcal{C}'[\mathcal{C}[A_2]_{\setminus\widetilde{y}}] = \nu\widetilde{c},\widetilde{n}_2.\left(\sigma_2, S_2, \begin{array}{c}\mathcal{P}_2 \cup \{(\overline{a}_i\langle y_i\sigma_2\rangle, \emptyset)\}_{i\in I} \cup \left\{(\overline{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J} \\ \cup \left\{(\overline{d}, \emptyset), (\overline{a}\langle e\rangle.d, \emptyset)\right\}\end{array}\right)$$

$$\Longrightarrow B_2 := \nu\widetilde{c},\widetilde{n}'_2.\left(\sigma_2, S'_2, \mathcal{P}'_2 \cup \{(\overline{a}_i\langle y_i\sigma_2\rangle, \emptyset)\}_{i\in I} \cup \left\{(\overline{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J}\right)$$

Let $A'_2 = \nu\widetilde{n}'_2.(\sigma_2, S'_2, \mathcal{P}'_2)$. We can easily verify that $\mathcal{C}[A'_2]_{\setminus\widetilde{y}} = B_2$. Then we have

$$A_2 = \nu\widetilde{n}_2.(\sigma_2, S_2, \mathcal{P}_2) \Longrightarrow A'_2 = \nu\widetilde{n}'_2.(\sigma_2, S'_2, \mathcal{P}'_2)$$

Since $\mathcal{C}[A'_1]_{\setminus\widetilde{y}} \approx \mathcal{C}[A'_2]_{\setminus\widetilde{y}}$, we have $(A'_1, A'_2) \in \mathcal{R}$.

ii. If $a = c_j$ for some $j \in J$ and $e \notin \widetilde{c}$, consider the evaluation context

$$\mathcal{C}' = (\text{-}, \text{-}, \left\{(\overline{d}, \emptyset), (b_j(u).\overline{u}\langle e\rangle.d.\overline{b}_j\langle u\rangle, \emptyset)\right\} \text{-})$$

where d is fresh. The analysis is similar as above.

iii. If $a = e = c_k$ for some $k \in J$, let $\mathcal{C}' = (\text{-}, \text{-}, \left\{(\overline{d}, \emptyset), (b_k(u).\overline{u}\langle u\rangle.d.\overline{b}_k\langle u\rangle, \emptyset)\right\} \text{-})$ where d is fresh. The analysis is similar as above.

iv. If $a = c_j$ and $e = c_k$ for some $j, k \in J$ with $j \neq k$, let

$$\mathcal{C}' = (\text{-}, \text{-}, \left\{(\overline{d}, \emptyset), (b_j(u).b_k(v).\overline{u}\langle v\rangle.(d.\overline{b}_j\langle u\rangle \mid \overline{b}_k\langle v\rangle), \emptyset)\right\} \text{-})$$

where d is fresh. The analysis is similar as above.

(e) Assume $A_1 = \nu\widetilde{n}'_1, e.(\sigma_1, S_1, \mathcal{P}'_1 \cup \{(\overline{a}\langle e\rangle.P, L)\}) \xrightarrow{\nu e.\overline{a}\langle e\rangle} A'_1 = \nu\widetilde{n}'_1.(\sigma_1, S_1, \mathcal{P}'_1 \cup \{(P, L)\})$ with $e \notin \widetilde{n}'_1$. In observational equivalence, internal transitions can never make the channel name e free. Thus, we need to construct an evaluation context that is able to provide the information for the names that was output previously. For notational convenience, we write *if* $x \in V$ *then 0 else* P, where $V = \{u_1, u_2, \cdots, u_k\}$, for

$$\begin{array}{l} \textit{if } x = u_1 \textit{ then } 0 \\ \quad \textit{else if } x = u_2 \textit{ then } 0 \\ \quad\quad \cdots\cdots \\ \quad\quad \textit{else if } x = u_k \textit{ then } 0 \textit{ else } P \end{array}$$

i. If $a \notin \widetilde{c}$, consider the evaluation context

$$\mathcal{C}' = (\text{-}, \text{-}, \left\{(\overline{d}, \emptyset), (a(x).\texttt{if } x \in \mathit{fn}(A_1, A_2) \texttt{ then } 0 \texttt{ else } d.\overline{b}_l\langle x\rangle, \emptyset)\right\} \text{-})$$

with b_l, d are fresh, then

$$\mathcal{C}'[\mathcal{C}[A_1]_{\setminus\widetilde{y}}] =$$

$$\nu\widetilde{c},\widetilde{n}'_1, e.\left(\sigma_{1\setminus\widetilde{y}}, S_1, \begin{array}{l}\mathcal{P}'_1 \cup \{(\overline{a}_i\langle y_i\sigma_1\rangle, \emptyset)\}_{i\in I} \cup \left\{(\overline{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J} \\ \cup \{(\overline{a}\langle e\rangle.P, L)\} \cup \left\{(\overline{d}, \emptyset)\right\} \\ \cup \left\{(a(x).\texttt{if } x \in \mathit{fn}(A_1, A_2) \texttt{ then } 0 \texttt{ else } d.\overline{b}_l\langle x\rangle, \emptyset)\right\}\end{array}\right)$$

$$\Longrightarrow B_1 =$$

$$\nu\widetilde{c}, \widetilde{n}_1', e. \left(\sigma_{1\setminus\widetilde{y}}, S_1, \begin{array}{c} \mathcal{P}_1' \cup \{(\overline{a}_i\langle y_i\sigma_1\rangle, \emptyset)\}_{i\in I} \cup \left\{(\overline{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J} \\ \cup \left\{(P, L), (\overline{b}_l\langle e\rangle, \emptyset)\right\} \end{array} \right)$$

The output $\overline{b}_l\langle e\rangle$ enables e to be accessed by environment through b_l in future. Similar to the above analysis, we have $\mathcal{C}'[\mathcal{C}[A_2]_{\setminus\widetilde{y}}] \Longrightarrow B_2 = \nu\widetilde{c}, \widetilde{n}_2', e, \widetilde{m}.(\sigma_{2\setminus\widetilde{y}}, S_2', \mathcal{P}_2' \cup \{(\overline{a}_i\langle y_i\sigma_2\rangle, \emptyset)\}_{i\in I} \cup \left\{(\overline{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J} \cup \left\{(\overline{b}_l\langle e\rangle, \emptyset)\right\})$ and $B_1 \approx B_2$. And also $A_2 \overset{\nu e.\overline{a}\langle e\rangle}{\Longrightarrow} A_2' = \nu\widetilde{n}_2', \widetilde{m}.(\sigma_2, S_2', \mathcal{P}_2')$. We construct a new context $\mathcal{C}'' = \nu\widetilde{c}, e.(\text{-}, \text{-}, \{(\overline{a}_i\langle y_i\rangle, \emptyset)\}_{i\in I}$
$\cup \left\{(\overline{b}_l\langle e\rangle, \emptyset)\right\} \cup \left\{(\overline{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J}\text{-})$. Then we can verify that $B_k = \mathcal{C}''[A_k']_{\setminus\widetilde{y}}$ with $k = 1, 2$. Hence we know that $(A_1', A_2') \in \mathcal{R}$.

ii. if $a = c_j$, $j \in J$, let $\mathcal{C}' = (\text{-}, \text{-}, \left\{(\overline{d}, \emptyset), (b_j(u).u(x).(d.\overline{b}_l\langle x\rangle \mid \overline{b}_j\langle u\rangle), \emptyset)\right\}\text{-})$ with b_l, d are fresh. The analysis is similar as above.

(f) Assume $A_1 = \nu\widetilde{n}_1.(\sigma_1, S_1, \mathcal{P}_1' \cup \{(\overline{a}\langle M_1\rangle.P, L)\}) \xrightarrow{\nu x.\overline{a}\langle x\rangle} A_1' = \nu\widetilde{n}_1.(\sigma_1 \cup \{M_1/x\}, S_1, \mathcal{P}_1' \cup \{(P, L)\})$ with $x \notin fv(A_1)$. In observational equivalence, internal transitions can never make term M_1 free or generate an substitution for M_1. Thus, we need to construct an evaluation context that is able to provide the information for the terms that have already been output previously.

i. if $a \notin \widetilde{c}$, consider the evaluation context $\mathcal{C}' = (\text{-}, \text{-}, \left\{(\overline{d}, \emptyset), (a(x).d.\overline{a}_l\langle x\rangle, \emptyset)\right\}\text{-})$ with a_l, d are fresh, then

$$\mathcal{C}'[\mathcal{C}[A_1]_{\setminus\widetilde{y}}] = \nu\widetilde{c}, \widetilde{n}_1. \left(\sigma_{1\setminus\widetilde{y}}, S_1, \begin{array}{c} \mathcal{P}_1' \cup \{(\overline{a}_i\langle y_i\sigma_1\rangle, \emptyset)\}_{i\in I} \cup \left\{(\overline{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J} \\ \cup \left\{(\overline{a}\langle M_1\rangle.P, L), (\overline{d}, \emptyset), (a(x).d.\overline{a}_l\langle x\rangle, \emptyset)\right\} \end{array} \right)$$

$$\Longrightarrow B_1 = \nu\widetilde{c}, \widetilde{n}_1. \left(\sigma_{1\setminus\widetilde{y}}, S_1, \begin{array}{c} \mathcal{P}_1' \cup \{(\overline{a}_i\langle y_i\sigma_1\rangle, \emptyset)\}_{i\in I} \cup \left\{(\overline{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J} \\ \cup \{(P, L), (\overline{a}_l\langle M_1\rangle, \emptyset)\} \end{array} \right)$$

The output $\overline{a}_l\langle M\rangle$ makes M to be accessed by environment through a_l in future. Similar to the above analysis, we have $\mathcal{C}'[\mathcal{C}[A_2]_{\setminus\widetilde{y}}] \Longrightarrow B_2 = \nu\widetilde{c}, \widetilde{n}_2, \widetilde{m}.(\sigma_{2\setminus\widetilde{y}}, S_2', \mathcal{P}_2' \cup \{(\overline{a}_i\langle y_i\sigma_2\rangle, \emptyset)\}_{i\in I} \cup \left\{(\overline{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J} \cup \{(\overline{a}_l\langle M_2\rangle, \emptyset)\})$ and $B_1 \approx B_2$. We can see that $A_2 \overset{\nu x.\overline{a}\langle x\rangle}{\Longrightarrow} A_2' = \nu\widetilde{n}_2, \widetilde{m}.(\sigma_2 \cup \{M_2/x\}, S_2', \mathcal{P}_2')$. Let $\mathcal{C}'' = \nu\widetilde{c}.(\text{-}, \text{-}, \{(\overline{a}_i\langle y_i\rangle, \emptyset)\}_{i\in I} \cup \{(\overline{a}_l\langle x\rangle, \emptyset)\} \cup \left\{(\overline{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J}\text{-})$. Then we can verify that $B_k = \mathcal{C}''[A_k']_{\setminus\widetilde{y}, x}$ with $k = 1, 2$. Hence we know that $(A_1', A_2') \in \mathcal{R}$.

ii. if $a = c_j$, $j \in J$, let $\mathcal{C}' = (\text{-}, \text{-}, \left\{(\overline{d}, \emptyset), (b_j(u).u(x).(d.\overline{a}_l\langle x\rangle \mid \overline{b}_j\langle u\rangle), \emptyset)\right\}\text{-})$ with a_l, d are fresh. The analysis is similar as above.

Theorem 10 (Coincidence). *On closed extended processes with only private state cells, it holds that* $\approx = \approx_l$.

Proof.

- For any $A \approx_l B$, we can easily check $dom(A) = dom(B)$ and $A \Downarrow_a$ then $B \Downarrow_a$. Using Proposition 8, we know $\mathcal{C}[A] \approx_l \mathcal{C}[B]$ for any context $\mathcal{C}$. According to Definition 1, we know $\approx_l \subseteq \approx$.

$$\lfloor 0 \rfloor_S = 0 \qquad \lfloor P \mid Q \rfloor_S = \lfloor P \rfloor_S \mid \lfloor Q \rfloor_S \qquad \lfloor \nu n.P \rfloor_S = \nu n. \lfloor P \rfloor_S \text{ if } n \notin \mathcal{N}_s$$
$$\lfloor ! P \rfloor_S = ! \lfloor P \rfloor_S \qquad \lfloor u(x).P \rfloor_S = u(x). \lfloor P \rfloor_S \qquad \lfloor \overline{u}\langle M \rangle.P \rfloor_S = \overline{u}\langle M \rangle. \lfloor P \rfloor_S$$
$$\lfloor \texttt{if } M = N \texttt{ then } P \texttt{ else } Q \rfloor_S = \texttt{if } M = N \texttt{ then } \lfloor P \rfloor_S \texttt{ else } \lfloor Q \rfloor_S$$
$$\lfloor s \mapsto M \rfloor_S = \overline{c_s}\langle M \rangle \qquad \lfloor \nu s.P \rfloor_S = \nu c_s. \lfloor P \rfloor_S \text{ if } s \in \mathcal{N}_s$$
$$\lfloor \texttt{lock } s.P \rfloor_S = \begin{cases} c_s(x). \lfloor P \rfloor_{S \cup \{s \mapsto x\}} & \text{if } s \notin dom(S) \text{ and } x \text{ is fresh} \\ 0 & \text{otherwise} \end{cases}$$
$$\lfloor \texttt{unlock } s.P \rfloor_S = \begin{cases} \overline{c_s}\langle M \rangle \mid \lfloor P \rfloor_T & \text{if } S = T \cup \{s \mapsto M\} \\ 0 & \text{otherwise} \end{cases}$$
$$\lfloor \texttt{read } s \texttt{ as } x.P \rfloor_S = \begin{cases} \lfloor P\{M/x\} \rfloor_S & \text{if } S = T \cup \{s \mapsto M\} \\ c_s(x).(\overline{c_s}\langle x \rangle \mid \lfloor P \rfloor_S) & \text{otherwise} \end{cases}$$
$$\lfloor s := M.P \rfloor_S = \begin{cases} \lfloor P \rfloor_{T \cup \{s \mapsto M\}} & \text{if } S = T \cup \{s \mapsto N\} \\ c_s(x).(\overline{c_s}\langle M \rangle \mid \lfloor P \rfloor_S) & \text{otherwise select fresh variable } x \end{cases}$$

Figure 4: Encoding private state cells with restricted channels

- The other direction $\approx \subseteq \approx_l$ is shown in Proposition 9.

4. Encoding Private State Cells with Restricted Channels

Private state cells can be encoded by restricted channels. This is an important observation; moreover, we will use this to prove Abadi-Fournet's theorem in the following Section 5. However, when modelling security protocols, the drawback of representing private state cells by restricted channels is that it may introduce false attacks when using the automatic tool ProVerif as argued in [8]. The reason is that some features of restricted channels are abstracted away when ProVerif translates process calculus into Horn clauses [15]. To solve this problem, we introduce the primitives for lock, read, write and unlock which will help us design better translations for stateful protocols in ProVerif. This has been demonstrated by the verification of reachability [8], and will be useful in future for verifying observational equivalence.

4.1. Encoding Private State Cells

We encode the extended processes with only private state cells into a subset of the extended processes which do not contain any cell name. Since the target language of the encoding does not have any cell name, we abbreviate extended processes $\nu\widetilde{n}.(\sigma, \emptyset, \{(P_i, \emptyset)\}_{i\in I})$ with no cell name to $\nu\widetilde{n}.(\sigma, \{P_i\}_{i\in I})$.

First we define encoding $\lfloor P \rfloor_S$ in Figure 4 for the plain process P under a given set of state cells $S = \{s_1 \mapsto M_1, \ldots, s_n \mapsto M_n\}$. For each cell s, we select a fresh channel name c_s. The encoding in Figure 4 only affects the part related to cell names, leaving other parts like input and output unchanged. The state cell $s \mapsto M$ and $\texttt{unlock}\ s$ are both encoded by an output $\overline{c_s}\langle M \rangle$ on the restricted channel c_s. The $\texttt{lock}\ s$ is represented by an input $c_s(x)$ on the same channel c_s. To read the cell $\texttt{read}\ s\ \texttt{as}\ x$, we use the input $c_s(x)$ to get the value from the cell and then put the value back $\overline{c_s}\langle x\rangle$, which enables the other operations on cell s in future. To write a new value into the cell $s := N$, we need to first consume the existing $\overline{c_s}\langle M\rangle$ by an input $c_s(x)$ and then generate a new output $\overline{c_s}\langle N\rangle$. Our encoding ensures that there is only one output $\overline{c_s}\langle M\rangle$ available on a specified restricted channel c_s at each moment. When the cell is locked, namely $\overline{c_s}\langle M\rangle$ is consumed by some $c_s(x)$, the other processes that intend to access the cell have to wait until an output $\overline{c_s}\langle N\rangle$ is available.

Let $A = \nu\widetilde{s}, \widetilde{n}. \Big(\sigma, \{s_i \mapsto M_i\}_{i\in I}, \{(P_j, L_j)\}_{j\in J}\Big)$ be an extended process [2] where $\widetilde{s} \subset \mathcal{N}_s$ and $\widetilde{n} \cap \mathcal{N}_s = \emptyset$. We define the encoding $\lfloor A\rfloor$ as:

$$\lfloor A\rfloor = \nu\widetilde{c_s}, \widetilde{n}. \left(\sigma, \{\overline{c_{s_i}}\langle M_i\rangle\}_{i\in U} \cup \left\{ \lfloor P_j\rfloor_{S_j} \right\}_{j\in J}\right)$$

where $U = \{\, i \mid s_i \notin \bigcup_{j\in J} L_j \text{ and } i \in I \,\}$ and $S_j = \{\, s_i \mapsto M_i \mid s_i \in L_j \text{ and } i \in I \,\}$. Intuitively, U is the set of indices of the unlocked state cells in $\{s_i \mapsto M_i\}_{i\in I}$, and S_j is the set of state cells locked by L_j.

Example 11. *Let $A = \nu s.(\emptyset, \{s \mapsto 0\}, \{(T(s), \emptyset)\})$ where $T(s)$ is defined in Example 3. Then $\lfloor A\rfloor = \nu c_s.(\emptyset, \{\overline{c_s}\langle 0\rangle, \lfloor T(s)\rfloor_\emptyset\})$ with $\lfloor T(s)\rfloor_\emptyset = c_s(z).\overline{a}\langle g(z)\rangle.\overline{c_s}\langle h(z)\rangle$ obtained by:*

$$\begin{aligned}
\lfloor T(s)\rfloor_\emptyset &= \lfloor \texttt{lock}\ s.\texttt{read}\ s\ \texttt{as}\ x.\overline{a}\langle g(x)\rangle.s := h(x).\texttt{unlock}\ s\rfloor_\emptyset \\
&= c_s(z).\, \lfloor \texttt{read}\ s\ \texttt{as}\ x.\overline{a}\langle g(x)\rangle.s := h(x).\texttt{unlock}\ s\rfloor_{\{s\mapsto z\}} \\
&= c_s(z).\, \lfloor \overline{a}\langle g(z)\rangle.s := h(z).\texttt{unlock}\ s\rfloor_{\{s\mapsto z\}} \\
&= c_s(z).\overline{a}\langle g(z)\rangle.\, \lfloor s := h(z).\texttt{unlock}\ s\rfloor_{\{s\mapsto z\}} \\
&= c_s(z).\overline{a}\langle g(z)\rangle.\, \lfloor \texttt{unlock}\ s\rfloor_{\{s\mapsto h(z)\}} \\
&= c_s(z).\overline{a}\langle g(z)\rangle.\overline{c_s}\langle h(z)\rangle
\end{aligned}$$

4.2. Soundness and Completeness of the Encoding

We call the process $\nu\widetilde{n}.(\sigma, \{P_j\}_{j\in J})$ described in Section 4 which does not contain any cell name a *pure extended process*. The operational semantics for pure extended process is still defined by Figure 1. On closed pure extended processes, the labelled bisimilarity are defined exactly the same as in Definition 5, while the observational equivalence $\approx^e$ is defined exactly the same as in Definition 1 except that the evaluation context does not contain any cell name.

We first define another equivalence $\simeq$ on the pure extended process.

Definition 12. *Let $\simeq$ be the smallest equivalence relation on pure extended processes closed under α-conversion such that*

$$\begin{array}{llcl}
I. & \nu\widetilde{n}, m.(\sigma, \mathcal{P}) & \simeq & \nu\widetilde{n}.(\sigma, \mathcal{P}) \quad \textit{if } m \notin fn(\widetilde{n}, \sigma, \mathcal{P}) \\
II. & \nu\widetilde{n}.(\sigma, \mathcal{P} \cup \{\nu m.P\}) & \simeq & \nu\widetilde{n}, m.(\sigma, \mathcal{P} \cup \{P\}) \quad \textit{if } m \notin fn(\widetilde{n}, \sigma, \mathcal{P}) \\
III. & \nu\widetilde{n}.(\sigma, \mathcal{P} \cup \{P \mid Q\}) & \simeq & \nu\widetilde{n}.(\sigma, \mathcal{P} \cup \{P\} \cup \{Q\}) \\
IV. & \nu\widetilde{n}.(\sigma\{M/x\}, \mathcal{P}\{M/x\}) & \simeq & \nu\widetilde{n}.(\sigma\{N/x\}, \mathcal{P}\{N/x\}) \ \textit{ if } M =_\Sigma N
\end{array}$$

[2] We abbreviate the set $\{\, s_i \mapsto M_i \mid i \in I \,\}$ as $\{s_i \mapsto M_i\}_{i\in I}$.

We write $A \simeq^1 B$ when the rewriting is just one step, i.e., by using one of the above four rules. In the following discussion, when we consider the derivation sequence $A \simeq^1 A_1 \simeq^1 A_2 \cdots \simeq^1 A_n \simeq^1 B$ for the closed pure extended processes A and B, we can safely assume that $A_1, A_2, \cdots, A_n$ are all closed pure extended processes. The above rule IV may introduce some redundant variables, for example $(\emptyset, \{\overline{a}\langle m\rangle\}) \simeq (\emptyset, \{\overline{a}\langle \mathsf{dec}(\mathsf{enc}(m, x), x)\rangle\})$ introduces a redundant variable x using a symmetric decryption rule $\mathsf{dec}(\mathsf{enc}(z, x), x) =_\Sigma z$. This kind of variables are meaningless and we can use an injective renaming ϱ to substitute these redundant variables with fresh names and get a new closed derivation sequence $A \simeq^1 \varrho(A_1) \simeq^1 \varrho(A_2) \cdots \simeq^1 \varrho(A_n) \simeq^1 B$. These redundant variables introduced by $\simeq$ are all dummy varialbes which are actually useless.

Lemma 13. *Let A, B be two closed pure extended processes. If $B \simeq^1 A \xrightarrow{\alpha} A'$ with $fv(\alpha) \subseteq dom(A)$ then there exists a closed pure extended process B' such that either $B \xRightarrow{\widehat{\alpha}} A'$ or $B \xrightarrow{\alpha} B' \simeq^1 A'$.*

Proof. See Appendix B.

Corollary 14. *Let A, B be two closed pure extended processes. If $B \simeq A \xrightarrow{\alpha} A'$ with $fv(\alpha) \subseteq dom(A)$ then $B \xRightarrow{\widehat{\alpha}} B' \simeq A'$ for some closed pure extended process B'.*

Proof. Using Lemma 13 several times.

Corollary 15. *Assume two closed pure extended processes A, B and $fv(\alpha) \subseteq dom(A)$. If $B \simeq A \xRightarrow{\alpha} A'$ then $B \xRightarrow{\widehat{\alpha}} B' \simeq A'$ for some closed pure extended process B'.*

Proof. By repeated applications of Corollary 14.

Now we start to prove that encoding preserves observational equivalence. Given a set of cells $S = \{s_1 \mapsto M_1, \cdots, s_n \mapsto M_n\}$ and a set of locks L, we define the projection $S|_L$ of S under L to be the set $\{\, t \mapsto N \mid \{t \mapsto N\} \subseteq S$ and $t \in L\,\}$.

Lemma 16. *Let A be a closed extended process and $fv(\alpha) \subseteq dom(A)$. If $A \xrightarrow{\alpha} B$ then $\lfloor A \rfloor \xRightarrow{\widehat{\alpha}} \lfloor B \rfloor$.*

Proof. See Appendix B.

Corollary 17. *Let A be a closed extended process and $fv(\alpha) \subseteq dom(A)$. If $A \xRightarrow{\alpha} B$ then $\lfloor A \rfloor \xRightarrow{\widehat{\alpha}} \lfloor B \rfloor$.*

Proof. If $A \xrightarrow{\alpha} A'$ and A is closed, we can verify that A' is also closed. This enables us to use Lemma 16 several times and get the conclusion.

Lemma 18. *Let A be a closed extended process and $fv(\alpha) \subseteq dom(A)$. If $\lfloor A \rfloor \xrightarrow{\alpha} B$ then $A \overset{\widehat{\alpha}}{\Longrightarrow} A'$ and $\lfloor A' \rfloor \simeq B$ for some A'.*

Proof. We only detail the proof for the communication on channel c_s which is obtained by encoding the cell name s. The other cases are trivial. Assume $\lfloor A \rfloor = \nu\widetilde{n}.(\sigma, \mathcal{P} \cup \{\overline{c_s}\langle M\rangle, c_s(x).P\}) \xrightarrow{\tau} B = \nu\widetilde{n}.(\sigma, \mathcal{P} \cup \{P\{M/x\}\})$. The input $c_s(x)$ may be encoded from $\texttt{lock}\ s$, $\texttt{read}\ s\ \texttt{as}\ x$ or $s := N$ where s is not locked, and the output may come from $\{s \mapsto M\}$ in plain process or in set of cells. We only detail the proof for the case when $\{s \mapsto M\}$ is already in cells part. The other case is similar.

1. Assume $A = \nu\widetilde{k}.(\sigma, S \cup \{s \mapsto M\}, \mathcal{Q} \cup \{(\texttt{lock}\ s.Q, L)\})$ with $s \notin L$. We have that the encoding of $\widetilde{k}$ is $\widetilde{n}$ while the encoding of $\mathcal{Q}$ and S under locks $locks(\mathcal{Q}) \cup L$ is $\mathcal{P}$. And the encoding $\lfloor \texttt{lock}\ s.Q \rfloor_{S|_L} = c_s(x).\lfloor Q \rfloor_{S|_L \cup \{s \mapsto x\}} = c_s(x).P$. Thus we have $\lfloor Q \rfloor_{S|_L \cup \{s \mapsto x\}} = P$. Substitute x with M, we get $\lfloor Q \rfloor_{S|_L \cup \{s \mapsto M\}} = P\{M/x\}$ since $x \notin fv(Q)$. Consider the transition $A \xrightarrow{\tau} A' := \nu\widetilde{k}.(\sigma, S \cup \{s \mapsto M\}, \mathcal{Q} \cup \{(Q, L \cup \{s\})\})$, then we have $\lfloor A' \rfloor = \nu\widetilde{n}.(\sigma, \mathcal{P} \cup \left\{\lfloor Q \rfloor_{S|_L \cup \{s \mapsto M\}}\right\}) = \nu\widetilde{n}.(\sigma, \mathcal{P} \cup \{P\{M/x\}\}) = B$.

2. Assume $A = \nu\widetilde{k}.(\sigma, S \cup \{s \mapsto M\}, \mathcal{Q} \cup \{(\texttt{read}\ s\ \texttt{as}\ x.Q, L)\})$ with $s \notin L \cup locks(\mathcal{Q})$. We have that the encoding of $\widetilde{k}$ is $\widetilde{n}$ while the encoding of $\mathcal{Q}$ and S under locks $locks(\mathcal{Q}) \cup L$ is $\mathcal{P}$. And the encoding $\lfloor \texttt{read}\ s\ \texttt{as}\ x.Q \rfloor_{S|_L} = c_s(x).(\overline{c_s}\langle x\rangle \mid \lfloor Q \rfloor_{S|_L}) = c_s(x).P$. Thus we get $\overline{c_s}\langle x\rangle \mid \lfloor Q \rfloor_{S|_L} = P$. Consider the transition $A \xrightarrow{\tau} A' = \nu\widetilde{k}.(\sigma, S \cup \{s \mapsto M\}, \mathcal{Q} \cup \{(Q\{M/x\}, L)\})$. Substituting x with M, we get $(\overline{c_s}\langle M\rangle \mid \lfloor Q\{M/x\} \rfloor_{S|_L}) = P\{M/x\}$ since $x \notin fv(S|_L)$. Thus we have $\lfloor A' \rfloor = \nu\widetilde{n}.(\sigma, \mathcal{P} \cup \left\{\overline{c_s}\langle M\rangle, \lfloor Q\{M/x\} \rfloor_{S|_L}\right\}) \simeq \nu\widetilde{n}.(\sigma, \mathcal{P} \cup \left\{\overline{c_s}\langle M\rangle \mid \lfloor Q\{M/x\} \rfloor_{S|_L}\right\}) = B$.

3. Assume $A = \nu\widetilde{k}.(\sigma, S \cup \{s \mapsto M\}, \mathcal{Q} \cup \{(s := N.Q, L)\})$ with $s \notin L \cup locks(\mathcal{Q})$. We have that the encoding of $\widetilde{k}$ is $\widetilde{n}$ while the encoding of $\mathcal{Q}$ and S under locked cells $locks(\mathcal{Q}) \cup L$ is $\mathcal{P}$. And the encoding $\lfloor s := N.Q \rfloor_{S|_L} = c_s(x).(\overline{c_s}\langle N\rangle \mid \lfloor Q \rfloor_{S|_L}) = c_s(x).P$. Thus we get $\overline{c_s}\langle N\rangle \mid \lfloor Q \rfloor_{S|_L} = P$. Consider the transition $A \xrightarrow{\tau} A' = \nu\widetilde{k}.(\sigma, S \cup \{s \mapsto M\}, \mathcal{Q} \cup \{(Q, L)\})$.Thus we have $\lfloor A' \rfloor = \nu\widetilde{n}.(\sigma, \mathcal{P} \cup \left\{\overline{c_s}\langle N\rangle, \lfloor Q \rfloor_{S|_L}\right\})$ $\simeq \nu\widetilde{n}.(\sigma, \mathcal{P} \cup \left\{\overline{c_s}\langle N\rangle \mid \lfloor Q \rfloor_{S|_L}\right\}) = B$.

Corollary 19. *Let A be a closed extended process and $fv(\alpha) \subseteq dom(A)$. If $\lfloor A \rfloor \overset{\alpha}{\Longrightarrow} B$ then $A \overset{\widehat{\alpha}}{\Longrightarrow} A'$ and $\lfloor A' \rfloor \simeq B$ for some A'.*

Proof. Using Lemma 18 and Corollary 15 several times.

The following theorem states that encoding preserves the observational equivalence:

Theorem 20. *For two closed extended processes A, B with only private state cells, we have $A \approx B$ iff $\lfloor A \rfloor \approx^e \lfloor B \rfloor$ where $\approx^e$ is an equivalence defined exactly the same as Definition 1 except the context $\mathcal{C}$ does not contain any cell names.*

Proof.

1. We construct the following set $\mathcal{R}$ on pairs of closed extended processes:

$$\mathcal{R} = \{ (A, B) \mid \lfloor A \rfloor \simeq D_1 \approx^e D_2 \simeq \lfloor B \rfloor \}$$

and prove that $\mathcal{R} \subseteq \approx$.

If $A \Downarrow_c$, by Corollary 17, we have $\lfloor A \rfloor \Downarrow_c$. Using Corollary 15 we have $D_1 \Downarrow_c$. Since $D_1 \approx^e D_2$, we have $D_2 \Downarrow_c$. Using Corollary 15, we have $\lfloor B \rfloor \Downarrow_c$. By Corollary 19 we know that $B \Downarrow_c$.

If $A \Longrightarrow A'$, by Corollary 17, we have $\lfloor A \rfloor \Longrightarrow \lfloor A' \rfloor$. From Corollary 15, there exists D_1' such that $D_1 \Longrightarrow D_1' \simeq \lfloor A' \rfloor$. Since $D_1 \approx^e D_2$, there exists D_2' such that $D_2 \Longrightarrow D_2' \approx^e D_1'$. By Corollary 15, there exists D_2'' such that $\lfloor B \rfloor \Longrightarrow D_2'' \simeq D_2'$. By Corollary 19, there exists B' such that $B \Longrightarrow B'$ and $\lfloor B' \rfloor \simeq D_2'' \simeq D_2'$. From $A \Longrightarrow A'$ and $\lfloor A' \rfloor \simeq D_1' \approx^e D_2'$, we know that $(A', B') \in \mathcal{R}$.

For any evaluation context $\mathcal{C} = \nu\widetilde{n}.(\sigma\text{-}, S\text{-}, \mathcal{P}\text{-})$, we need to prove that $(\mathcal{C}[A], \mathcal{C}[B]) \in \mathcal{R}$. We can use the same encoding to encode away all the cell names in the context $\mathcal{C}$ and get a new evaluation context $\lfloor \mathcal{C} \rfloor = \nu\widetilde{l}.(\sigma\text{-}, \mathcal{Q}\text{-})$. Assume $A = \nu\widetilde{n}_1.(\sigma_1, S_1, \mathcal{P}_1)$ and $\lfloor A \rfloor = \nu\widetilde{m}_1.(\sigma_1, \mathcal{Q}_1)$. Then we can see that $\mathcal{C}[A] = \nu\widetilde{n}, \widetilde{m}.(\sigma\sigma_1 \cup \sigma_1, S\sigma_1 \cup S_1, \mathcal{P}\sigma_1 \cup \mathcal{P}_1)$ and $\lfloor \mathcal{C} \rfloor [\lfloor A \rfloor] = \nu\widetilde{l}, \widetilde{m}_1.(\sigma\sigma_1 \cup \sigma_1, \mathcal{Q}\sigma_1 \cup \mathcal{Q}_1)$. Note that $\mathcal{C}$ and A do not share any cell name. Applying encoding to $\mathcal{C}[A]$ we get $\lfloor \mathcal{C}[A] \rfloor = \nu\widetilde{l}, \widetilde{m}_1.(\sigma\sigma_1 \cup \sigma_1, \mathcal{Q}\sigma_1 \cup \mathcal{Q}_1) = \lfloor \mathcal{C} \rfloor [\lfloor A \rfloor]$. Similarly we have $\lfloor \mathcal{C}[B] \rfloor = \lfloor \mathcal{C} \rfloor [\lfloor B \rfloor]$. From $\lfloor A \rfloor \simeq D_1$ and $D_2 \simeq \lfloor B \rfloor$, we can see that $\lfloor \mathcal{C} \rfloor [\lfloor A \rfloor] \simeq \lfloor \mathcal{C} \rfloor [D_1]$ and $\lfloor \mathcal{C} \rfloor [D_2] \simeq \lfloor \mathcal{C} \rfloor [\lfloor B \rfloor]$. From $D_1 \approx^e D_2$, applying context $\lfloor \mathcal{C} \rfloor$, we can see that $\lfloor \mathcal{C} \rfloor [D_1] \approx^e \lfloor \mathcal{C} \rfloor [D_2]$. In brief, we have $\lfloor \mathcal{C}[A] \rfloor = \lfloor \mathcal{C} \rfloor [\lfloor A \rfloor] \simeq \lfloor \mathcal{C} \rfloor [D_1] \approx^e \lfloor \mathcal{C} \rfloor [D_2] \simeq \lfloor \mathcal{C} \rfloor [\lfloor B \rfloor] = \lfloor \mathcal{C}[B] \rfloor$. Thus we know $(\mathcal{C}[A], \mathcal{C}[B]) \in \mathcal{R}$.

2. We construct the following set $\mathcal{S}$ on pairs of closed extended processes:

$$\mathcal{S} = \{ (D_1, D_2) \mid D_1 \simeq \lfloor A \rfloor, A \approx B, \lfloor B \rfloor \simeq D_2 \}$$

and prove that $\mathcal{S} \subseteq \approx^e$.

If $D_1 \Downarrow_c$, by Corollary 15, we have $\lfloor A \rfloor \Downarrow_c$. Using Corollary 19 we have $A \Downarrow_c$. Since $A \approx B$, we have $B \Downarrow_c$. By Corollary 17 we know that $\lfloor B \rfloor \Downarrow_c$. Using Corollary 15, we have $D_2 \Downarrow_c$.

If $D_1 \Longrightarrow D_1'$, by Corollary 15, we have $\lfloor A \rfloor \Longrightarrow A_1$. From Corollary 19, there exists A' such that $A \Longrightarrow A'$ and $\lfloor A' \rfloor \simeq A_1$. Since $A \approx B$, there exists B' such that $B \Longrightarrow B' \approx A'$. By Corollary 17, we have $\lfloor B \rfloor \Longrightarrow \lfloor B' \rfloor$. By Corollary 15, there exists D_2' such that $D_2 \Longrightarrow D_2'$ and $\lfloor B' \rfloor \simeq D_2'$. From $D_1 \Longrightarrow D_1'$ and $D_1' \simeq \lfloor A' \rfloor$, we know that $(D_1', D_2') \in \mathcal{R}$.

For any pure evaluation context $\mathcal{C}$, we can easily see that $\mathcal{C}[D_1] \simeq \mathcal{C}[\lfloor A \rfloor] = \lfloor \mathcal{C}[A] \rfloor$ and $\mathcal{C}[D_2] \simeq \mathcal{C}[\lfloor B \rfloor] = \lfloor \mathcal{C}[B] \rfloor$ and $\mathcal{C}[A] \approx \mathcal{C}[B]$, thus $(\mathcal{C}[D_1], \mathcal{C}[D_2]) \in \mathcal{S}$.

5. Proof of Abadi-Fournet's Theorem

We shall use our Theorem 10 and Theorem 20 to derive Abadi-Fournet's theorem, namely Theorem 1 in [3]. We revise the original applied pi calculus [3] slightly: *active substitutions are only defined on terms of base sort*; otherwise Theorem 1 in [3] does not hold [11].[3] Since the active substitutions in applied pi calculus float everywhere in the extended processes, in order to prove Abadi-Fournet's theorem, we need to normalise the extended processes first. We can transform the extended processes in the applied pi calculus – denoted by A_r, B_r, C_r to avoid confusion – into the extended processes in stateful applied pi calculus by function $\mathcal{T}$ (assume bound names are pairwise-distinct and different from free names): [4]

$$\mathcal{T}(0) = (\emptyset, \emptyset) \qquad \mathcal{T}(\{M/x\}) = (\{M/x\}, \emptyset) \qquad \mathcal{T}(\nu n.A_r) = \nu n.\mathcal{T}(A_r)$$

$$\mathcal{T}(\nu x.A_r) = \nu \widetilde{n}.(\sigma, \mathcal{P}) \ \textbf{if}\ \mathcal{T}(A_r) = \nu \widetilde{n}.(\sigma \cup \{M/x\}, \mathcal{P})$$

$$\mathcal{T}(A_r^1 \mid A_r^2) = \nu \widetilde{n}_1, \widetilde{n}_2.((\sigma_1 \cup \sigma_2)^*, (\mathcal{P}_1 \cup \mathcal{P}_2)(\sigma_1 \cup \sigma_2)^*) \ \textbf{if}\ \mathcal{T}(A_r^i) = \nu \widetilde{n}_i.(\sigma_i, \mathcal{P}_i) \text{ for } i = 1, 2$$

$$\mathcal{T}(A_r) = (\emptyset, \{A_r\}) \text{ in all other cases of } A_r$$

Intuitively, $\mathcal{T}$ pulls out name restrictions, applies active substitutions and separates them from the plain processes, and eliminates variable restrictions. For instance, $\mathcal{T}(\overline{a}\langle x\rangle.\nu n.\overline{a}\langle n\rangle \mid \nu k.\{k/x\}) = \nu k.(\{k/x\}, \{\overline{a}\langle k\rangle.\nu n.\overline{a}\langle n\rangle\})$. This normalisation $\mathcal{T}$ preserves both observational equivalence and labelled bisimilarity:

Theorem 21 (Soundness and Completeness of Stateful Applied Pi). *For two closed extended processes A_r and B_r in applied pi calculus,*

1. *A_r and B_r are labelled bisimilar in applied pi iff $\mathcal{T}(A_r) \approx_l \mathcal{T}(B_r)$;*
2. *A_r and B_r are observationally equivalent in applied pi iff $\mathcal{T}(A_r) \approx^e \mathcal{T}(B_r)$;*

Proof. See Appendix C.

With all the theorems ready, now we can prove Abadi-Fournet's theorem:

Corollary 22 (Coincidence in Applied Pi). *Observational equivalence coincides with labelled bisimilarity in applied pi calculus.*

[3]Here is a counter example: let $A_r = \nu c.(\overline{c}.\overline{a} \mid \{c/x\})$ and $B_r = \nu c.(0 \mid \{c/x\})$. Obviously A_r and B_r are labelled bisimilar since their frames are the same and both have no transitions. However, they are not observationally equivalent. Consider the context $x(y)$, then $A_r \mid x(y) \Downarrow_a$ but $B_r \mid x(y) \not\Downarrow_a$.

[4]We write σ^* for the result of composing the substitution σ with itself repeatedly until an idempotent substitution is reached.

Proof. This is a direct corollary of Theorem 10, Theorem 20 and Theorem 21:

A_r and B_r are observationally equivalent
  iff $\mathcal{T}(A_r) \approx^e \mathcal{T}(B_r)$ by Theorem 21 (2)
  iff $\mathcal{T}(A_r) \approx \mathcal{T}(B_r)$ by Theorem 20 and $\lfloor \mathcal{T}(A_r) \rfloor = \mathcal{T}(A_r)$ and $\lfloor \mathcal{T}(B_r) \rfloor = \mathcal{T}(B_r)$
  iff $\mathcal{T}(A_r) \approx_l \mathcal{T}(B_r)$ by Theorem 10
  iff A_r and B_r are labelled bisimilar by Theorem 21 (1)

6. Extending the Language with Public State Cells

Hardware modules like TPMs and smart cards are intended to be secure, but an attacker might succeed in finding ways of compromising their tamper-resistant features. Similarly, attackers can potentially hack into databases [1]. We model these attacks by considering that the attacker compromises the private state cells, after which they are public. Protocols may provide some security properties that hold even under such compromises of the hardware or database. A typical example is forward privacy [26] which requires the past events remain secure even if the attacker compromises the device. This will be further discussed in the following Example 28 and Example 29. A cell s not in the scope of νs is public, which enables the attacker to lock the cell, read its contents or overwrite it.

We now give the details of the syntactic additions for public cells and the definition of observational equivalence. To let a private state cell become public, we extend the plain processes in Section 2 with a new primitive `open` $s.P$ Extended processes are defined as before. We extend the transitions in Fig. 1 by a new transition relation $\xrightarrow{\tau(s)}$ defined in Fig. 5 for reasoning about public state cells. These internal transitions specify on which public state cell the operations are performed. The label $\tau(s)$ is necessary when we later define labelled bisimilarity. It is worth pointing out that $\tau(s)$ is defined for the read, write and lock operations on a public cell s (the first three rules in Fig. 5) only when the cell is unlocked. When a public cell is locked, the operations on this cell become invisible to the other processes, thus the operations on a locked public cell are defined by internal transitions τ in Fig. 1. When a public cell s is unlocked, the operations on it are visible, thus are defined by $\tau(s)$ to indicate there is an operation on the cell s.

Let $A = \nu\widetilde{n}.(\sigma, S, \mathcal{P})$ and we write $locks(A)$ for the set $locks(\mathcal{P}) \setminus \widetilde{n}$. We write $unlocks(A)$ for the set $fs(A) \setminus locks(A)$, namely the unlocked public state cells. We write $\stackrel{\epsilon}{\Longrightarrow}$ for the reflexive and transitive closure of $\xrightarrow{\tau}$ and $\xrightarrow{\tau(s)}$ for any cell s. We write $A \Downarrow_a$ when $A \stackrel{\epsilon}{\Longrightarrow} \nu\widetilde{n}.(\sigma, S, \mathcal{P} \cup \{(\overline{a}\langle M\rangle.P, L)\})$ with $a \notin \widetilde{n}$.

6.1. Observational Equivalence

We first define observational equivalence for our stateful language in the presence of public state cells. In principle, we stick to the original definition of observational equivalence [3] as much as possible in order to capture the intuition of indistinguishability from the attacker's point of view.

$$\begin{array}{rcl}
\nu\widetilde{n}.(\sigma, S \cup \{s \mapsto M\}, \mathcal{P} \cup \{(\texttt{read}\ s\ \texttt{as}\ x.P, L)\}) & \xrightarrow{\tau(s)} & \nu\widetilde{n}.(\sigma, S \cup \{s \mapsto M\}, \mathcal{P} \cup \{(P\{M/x\}, L)\}) \\
& & \textbf{if } s \notin \widetilde{n} \cup L \cup \mathit{locks}(\mathcal{P}) \\
\nu\widetilde{n}.(\sigma, S \cup \{s \mapsto M\}, \mathcal{P} \cup \{(s := N.P, L)\}) & \xrightarrow{\tau(s)} & \nu\widetilde{n}.(\sigma, S \cup \{s \mapsto N\}, \mathcal{P} \cup \{(P, L)\}) \\
& & \textbf{if } s \notin \widetilde{n} \cup L \cup \mathit{locks}(\mathcal{P}) \\
\nu\widetilde{n}.(\sigma, S \cup \{s \mapsto M\}, \mathcal{P} \cup \{(\texttt{lock}\ s.P, L)\}) & \xrightarrow{\tau(s)} & \nu\widetilde{n}.(\sigma, S \cup \{s \mapsto M\}, \mathcal{P} \cup \{(P, L \cup \{s\})\}) \\
& & \textbf{if } s \notin \widetilde{n} \cup L \cup \mathit{locks}(\mathcal{P}) \\
\nu\widetilde{n}.(\sigma, S \cup \{s \mapsto M\}, \mathcal{P} \cup \{(\texttt{unlock}\ s.P, L)\}) & \xrightarrow{\tau(s)} & \nu\widetilde{n}.(\sigma, S \cup \{s \mapsto M\}, \mathcal{P} \cup \{(P, L \setminus \{s\})\}) \\
& & \textbf{if } s \notin \widetilde{n} \cup \mathit{locks}(\mathcal{P}) \text{ and } s \in L \\
\nu\widetilde{n}, s.(\sigma, S \cup \{s \mapsto M\}, \mathcal{P} \cup \{(\texttt{open}\ s.P, L)\}) & \xrightarrow{\tau(s)} & \nu\widetilde{n}.(\sigma, S \cup \{s \mapsto M\}, \mathcal{P} \cup \{(P, L)\}) \\
& & \textbf{if } s \notin \widetilde{n}
\end{array}$$

Figure 5: Internal transitions for public state cells.

Definition 23. *Observational equivalence ($\approx$) is the largest symmetric relation $\mathcal{R}$ on pairs of closed extended processes (which may contain public state cells) such that $A \mathrel{\mathcal{R}} B$ implies*

(i) $\mathit{locks}(A) = \mathit{locks}(B)$, $\mathit{fs}(A) = \mathit{fs}(B)$ and $\mathit{dom}(A) = \mathit{dom}(B)$;

(ii) if $A \Downarrow_a$ then $B \Downarrow_a$;

(iii) if $A \stackrel{\epsilon}{\Longrightarrow} A'$ then $B \stackrel{\epsilon}{\Longrightarrow} B'$ and $A' \mathrel{\mathcal{R}} B'$ for some B';

(iv) for all closing evaluation contexts $\mathcal{C}$, $\mathcal{C}[A] \mathrel{\mathcal{R}} \mathcal{C}[B]$.

The definition of observational equivalence on public state cells is similar to the one for private state cells, but the language features of public state cells are significantly different from private state cells. The addition of public state cells increases the power of the attacker significantly, as without the name restriction νs for a state cell s, when s is unlocked, the attacker can lock the cell, read its content and overwrite it. To illustrate this point, we start by analysing several examples.

Example 24. *The attacker can lock the unlocked public state cells. Assume*

$$A = (\emptyset, \{s \mapsto 0\}, \{(\overline{c}\langle b\rangle, \emptyset)\}) \qquad B = (\emptyset, \{s \mapsto 0\}, \{(\texttt{read}\ s\ \texttt{as}\ x.\ \overline{c}\langle b\rangle, \emptyset)\})$$

A and B are not observationally equivalent. Let $\mathcal{C} =$ (-, -, $\{(0, \{s\})\}$ -). The context $\mathcal{C}$ does nothing but holds the lock on cell s and it will never release the lock. So we have $\mathcal{C}[A] \Downarrow_c$ but $\mathcal{C}[B] \not\Downarrow_c$ because reading cell s in B is blocked forever by context $\mathcal{C}$.

In comparison, the following extended processes A, B are observationally equivalent:

$$\begin{aligned}
A &= (\emptyset, \{s \mapsto 0\}, \{(\texttt{read}\ s\ \texttt{as}\ x.\ \overline{c}\langle b\rangle, \emptyset)\}) \\
B &= (\emptyset, \{s \mapsto 0\}, \{(\texttt{read}\ s\ \texttt{as}\ x.\ \texttt{read}\ s\ \texttt{as}\ y.\ \overline{c}\langle b\rangle, \emptyset)\})
\end{aligned}$$

When A performs the reading, B can match it by performing its two reading together. When B performs one reading, A can match it by doing nothing.

Example 25. *The attacker can read an unlocked public state cell. Assume*

$$A = (\emptyset, \{s \mapsto 0\}, \{(\,!\,s := 0, \emptyset), (\,!\,s := 1, \emptyset)\})$$
$$B = (\emptyset, \{s \mapsto 1\}, \{(\,!\,s := 0, \emptyset), (\,!\,s := 1, \emptyset)\})$$

Cell s is unlocked in both A and B. Both A and B can write 0 or 1 to the cell s arbitrary number of times. The only difference between A and B is the initial values in cell s. A and B are not observationally equivalent because the context

$$\mathcal{C} = (\text{-}, \text{-}, \{(\mathtt{read}\ s\ \mathtt{as}\ x.\ \mathtt{if}\ x = 0\ \mathtt{then}\ \overline{c}\langle b\rangle, \{s\})\}\ \text{-})$$

can distinguish them. The context $\mathcal{C}$ holds the lock of cell s, thus no one can change the value in s when $\mathcal{C}$ reads the value. We have $\mathcal{C}[A] \Downarrow_c$ but $\mathcal{C}[B] \not\Downarrow_c$.

In comparison, the following processes are observationally equivalent:

$$A' = (\emptyset, \{s \mapsto 0\}, \{(\,!\,s := 0, \emptyset), (\,!\,s := 1, \emptyset), (\mathtt{unlock}\ s, \{s\})\})$$
$$B' = (\emptyset, \{s \mapsto 1\}, \{(\,!\,s := 0, \emptyset), (\,!\,s := 1, \emptyset), (\mathtt{unlock}\ s, \{s\})\})$$

Cell s is locked in both A' and B'. When a cell is locked, the attacker cannot see its value until it is unlocked. Both A' and B' can adjust the value of cell s after `unlock` *s. Assume*

$$A' \xrightarrow{\tau(s)} (\emptyset, \{s \mapsto 0\}, \{(\,!\,s := 0, \emptyset), (\,!\,s := 1, \emptyset), (0, \emptyset)\})$$

Then B' can match this transition by first unlocking the cell s and then doing a writing $s := 0$ and evolving to exactly the same process:

$$B' \xrightarrow{\tau(s)} (\emptyset, \{s \mapsto 1\}, \{(\,!\,s := 0, \emptyset), (\,!\,s := 1, \emptyset), (0, \emptyset)\})$$
$$\xRightarrow{\tau(s)} (\emptyset, \{s \mapsto 0\}, \{(\,!\,s := 0, \emptyset), (\,!\,s := 1, \emptyset), (0, \emptyset)\})$$

Intuitively, the locked or unlocked status of a public state cell is observable by the environment. Therefore, we require $\mathit{locks}(A) = \mathit{locks}(B)$ and $\mathit{fs}(A) = \mathit{fs}(B)$ in the definition of observational equivalence. Furthermore, without this condition, this definition would not yield an equivalence relation, as transitivity does not hold in general. For example, consider the following extended processes,

$$A = (\emptyset, \{s \mapsto 0\}, \{(\,!\,s := 0, \emptyset), (\,!\,s := 1, \emptyset), (\,!\,\mathtt{lock}\ s.\mathtt{unlock}\ s, \emptyset)\})$$
$$B = (\emptyset, \{s \mapsto 1\}, \{(\,!\,s := 0, \emptyset), (\,!\,s := 1, \emptyset), (\,!\,\mathtt{lock}\ s.\mathtt{unlock}\ s, \emptyset), (\mathtt{unlock}\ s, \{s\})\})$$
$$C = (\emptyset, \{s \mapsto 1\}, \{(\,!\,s := 0, \emptyset), (\,!\,s := 1, \emptyset), (\,!\,\mathtt{lock}\ s.\mathtt{unlock}\ s, \emptyset)\})$$

Without the condition, then A and B would be equivalent, as well as B and C, because the value in s can always be adjusted to be exactly the same after `unlock` s. But A and C are not equivalent as analysed in Example 25.

Example 26. *The value in an unlocked public state cell is a part of the attacker's knowledge. Assume*

$$A = \nu k.(\emptyset, \{s \mapsto k\}, \{(s := 0.a(x).\mathtt{if}\ x = k\ \mathtt{then}\ \overline{c}\langle b\rangle, \emptyset)\})$$
$$B = \nu k.(\emptyset, \{s \mapsto k\}, \{(s := 0.a(x), \emptyset)\})$$

A and B are not observationally equivalent. Let $\mathcal{C} = (\text{-}, \text{-}, \{(\texttt{read}\ s\ \texttt{as}\ y.\ \overline{a}\langle y\rangle, \emptyset)\}\ \text{-})$*. Then* $\mathcal{C}[A] \Downarrow_c$ *but* $\mathcal{C}[B] \not\Downarrow_c$ *because*

$$\begin{aligned}\mathcal{C}[A] &\xrightarrow{\tau(s)} \nu k.\,(\emptyset, \{s \mapsto k\}, \{(\overline{a}\langle k\rangle, \emptyset), (s := 0.a(x).\texttt{if}\ x = k\ \texttt{then}\ \overline{c}\langle b\rangle, \emptyset)\}) \\ &\xrightarrow{\tau(s)} \nu k.\,(\emptyset, \{s \mapsto 0\}, \{(\overline{a}\langle k\rangle, \emptyset), (a(x).\texttt{if}\ x = k\ \texttt{then}\ \overline{c}\langle b\rangle, \emptyset)\}) \\ &\Longrightarrow\ \ \nu k.\,(\emptyset, \{s \mapsto 0\}, \{(\overline{c}\langle b\rangle, \emptyset)\})\end{aligned}$$

But there is no output on channel c in $C[B]$*. Hence* $A \not\approx B$*.*

Example 27. *The attacker can write an arbitrary value into an unlocked public cell. Assume two extended processes*

$$A = (\emptyset, \{s \mapsto 0\}, \{(s := 0.\ s := 0, \emptyset)\}) \qquad B = (\emptyset, \{s \mapsto 0\}, \{(s := 0, \emptyset)\})$$

A and B are not observationally equivalent. Applying $\mathcal{C} = (\text{-}, \text{-}, \{(s := 1.s := 1, \emptyset)\}\ \text{-})$ *to both* A *and* B*, the interleaving of* $s := 0$ *and* $s := 1$ *can generate a sequence of values* $0, 1, 0, 1, 0$ *in cell s in* $\mathcal{C}[A]$*, while the closest sequence generated by* $\mathcal{C}[B]$ *should be* $0, 1, 0, 1, 1$*. So when the attacker keeps on reading the value in cell* s*, he would be able to notice the difference.*

Instead of using the primitive $\texttt{open}\ s$, an alternative way for making a private state cell become public is to send cell name s on a free channel $\overline{c}\langle s\rangle.P$. The reason we choose the primitive $\texttt{open}\ s.P$ here is because sending and receiving cell names through channels is too powerful, and will lead to soundness problems when we define labelled bisimilarity later. For example, let

$$A = (\emptyset, \emptyset, \{(c(x).\texttt{read}\ x\ \texttt{as}\ z.\overline{a}\langle z\rangle, \emptyset)\}) \qquad B = (\emptyset, \emptyset, \{(c(x), \emptyset)\})$$

In the presence of input and output for cell names, A and B are not observationally equivalent. Let $\mathcal{C} = (\text{-}, \{t \mapsto 0\}\ \text{-}, \{(\overline{c}\langle t\rangle, \emptyset)\}\ \text{-})$. The context $\mathcal{C}$ brings his own state cell $t \mapsto 0$ and we have $\mathcal{C}[A] \Downarrow_a$ but $\mathcal{C}[B] \not\Downarrow_a$. That is to say, in order to define a sound labelled bisimilarity, we have to allow a process like $(\emptyset, \emptyset, \{(\texttt{read}\ t\ \texttt{as}\ z.\ \overline{a}\langle z\rangle, \emptyset)\})$ to perform the reading even without a state cell $t \mapsto 0$. This requires a rather complex definition of labelled bisimilarity, while what we want is to simply free a cell which can be achieved by $\texttt{open}\ s.P$.

Now we give examples of the use of public state cells for modelling protocols and security properties. Another security concern for RFID tags is forward privacy [28]. In the following Example 28 and Example 29, we shall illustrate how to model forward privacy by public state cells. Forward privacy requires that even the attacker breaks the tag, the past events should still be untraceable. Public state cells enable us to model the compromised tags.

Example 28. *We consider an improved version of the naive protocol in Example 2. Instead of simply outputting the tag's id, the tag generates a random number* r*, hashes its id concatenated with* r *and then sends both* r *and* $h(id, r)$ *to the reader for identification. This can be modelled by:*

$$Q(s) = \texttt{read}\ s\ \texttt{as}\ x.\,\nu r.\,\overline{a}\langle(r, h(x, r))\rangle$$

Upon receiving the value, the reader identifies the tag by performing a brute-force search of its known ids. By observing on channel a, the attacker can get the data pairs from a particular tag s: $(r_1, h(id, r_1)), (r_2, h(id, r_2)), (r_3, h(id, r_3)) \cdots$. Since the hash function is not invertible, without knowing the value of id, these data appear as just random data to the attacker. Hence this improved version satisfies the untraceability defined in Example 2. But it does not have the forward privacy. Let RD be process modelling the reader and back-end database. The forward privacy can be characterised by the observational equivalence

$$(\emptyset, \emptyset, \{(\,!\,\nu s, id.([s \mapsto id] \mid Q(s) \mid \texttt{open}\ s.\,!\,Q(s) \mid RD), \emptyset)\}) \\ \approx (\emptyset, \emptyset, \{(\,!\,\nu s, id.([s \mapsto id] \mid !\,Q(s) \mid \texttt{open}\ s \mid RD), \emptyset)\})$$

The primitive open s *makes the private state cell s become public. Before the cell s is broken, the attacker cannot decide how the system runs. In other words, whether the tag s is used for only once, namely $Q(s)$, or is used for arbitrary number of times, namely $!\,Q(s)$, it is out of the control of the attacker. But after the tag is broken, the attacker fully controls the tag, so he knows when and where the tag is used. Despite knowing the events that happen after the tag is broken, the attacker should still not be able to trace the past events. Therefore, in the first process, we add $!\,Q(s)$ after* open s *to model this scenario. Intuitively, only the events before the tag is broken may be different while the events after the tag is broken are exactly the same. Hence the above observational equivalence can capture forward privacy.*

However the above equivalence does not hold which means there is no forward privacy in this protocol. The attacker can obtain the id from the broken tag and then verify whether the previously gathered data $(r_1, h(id, r_1))$ and $(r_2, h(id, r_2))$ refer to the same tag id by hashing id with r_1 (or r_2) and then comparing the result with $h(id, r_1)$ (or $h(id, r_2)$).

Example 29. *Continuing with the OSK protocol in Example 3, we model the forward privacy by the observational equivalence:*

$$(\emptyset, \emptyset, \{(\,!\,\nu s, k.([s \mapsto k] \mid T(s) \mid \texttt{open}\ s.\,!\,T(s) \mid RD), \emptyset)\}) \\ \approx (\emptyset, \emptyset, \{(\,!\,\nu s, k.([s \mapsto k] \mid !\,T(s) \mid \texttt{open}\ s \mid RD), \emptyset)\})$$

Before the tag is broken, the attacker can get a sequence $g(k), g(h(k)), g(h(h(k))), \cdots$ by eavesdropping on channel a. Right after each reading, the value in the tag will be updated to the hash of previous value: $h(k), h(h(k)), h(h(h(k))) \cdots$. When the tag is broken, the attacker will get from the tag a value $h^i(k)$ for some integer i. This value is not helpful for the attacker to infer whether the data $g(k), g(h(k)), \cdots, g(h^{i-1}(k))$ are from the same tag. Hence the OSK protocol can ensure the forward privacy.

6.2. Labelled Bisimilarity

In order to ease the verification of observational equivalence which is defined using the universal quantifier over contexts, we shall define labelled bisimilarity which replaces quantification over contexts by suitably labelled transitions. The traditional definition for labelled bisimilarity is neither sound nor complete w.r.t. observational equivalence in the presence of public state cells. We propose a novel definition for labelled bisimilarity and show how it solves all the problems caused by public state cells.

For a given cell s, we define $\xRightarrow{\tau(s)}$ to be the reflexive and transitive closure of $\xrightarrow{\tau}$ and $\xrightarrow{\tau(s)}$. We still use α to range over $\tau, a(M), \overline{a}\langle c\rangle, \nu c.\overline{a}\langle c\rangle$ and $\nu x.\overline{a}\langle x\rangle$, and use $\Longrightarrow$ for the reflexive and transitive closure of $\xrightarrow{\tau}$, and use $\xRightarrow{\widehat{\alpha}}$ for $\xRightarrow{\alpha}$ if α is not τ and for $\Longrightarrow$ if $\alpha = \tau$. Note that α cannot be $\tau(s)$.

To define labelled bisimilarity, we need an auxiliary transition relation $\xrightarrow{s:=N}$ for setting the values of public state cells:

$$\begin{array}{rcl} \nu\widetilde{n}.(\sigma, S \cup \{s \mapsto M\}, \mathcal{P}) & \xrightarrow{s:=N} & \nu\widetilde{n}.(\sigma, S \cup \{s \mapsto N\sigma\}, \mathcal{P}) \\ & & \textbf{if } s \notin \widetilde{n} \cup \mathit{locks}(\mathcal{P}) \text{ and } name(N) \cap \widetilde{n} = \emptyset \\ \nu\widetilde{n}.(\sigma, S, \mathcal{P}) & \xrightarrow{s:=N} & \nu\widetilde{n}.(\sigma, S, \mathcal{P}) \textbf{ if } s \in \widetilde{n} \cup \mathit{locks}(\mathcal{P}) \end{array}$$

The first rule of $\xrightarrow{s:=N}$ represents the attacker's ability to overwrite the public state cells. The second rule does not change the value of the cell s and is just for compatibility with `unlock` s and `open` s in Definition 31. We write $A \xrightarrow{s:=N}\xRightarrow{\tau(s)} A'$ for the combination of transitions $A \xrightarrow{s:=N} B$ and $B \xRightarrow{\tau(s)} A'$ for some B.

Definition 30. *Given two extended processes $A_i = \nu\widetilde{n}_i.(\sigma_i, S_i, \mathcal{P}_i)(i = 1, 2)$ such that $dom(\sigma_1) = dom(\sigma_2)$ and $fs(A_1) = fs(A_2)$ and $locks(A_1) = locks(A_2)$. We define* extensible state cells $\mathtt{esc}(A_1, A_2)$ *of A_1 and A_2 as*

$$\begin{array}{l} \mathtt{esc}(A_1, A_2) := \\ \quad \{\, s \mid s \in fs(A_1) \setminus locks(A_1), \nexists x \in dom(\sigma_1) \text{ s.t. } S_1(s) = x\sigma_1 \text{ and } S_2(s) = x\sigma_2 \,\} \end{array}$$

Intuitively, $\mathtt{esc}(A_1, A_2)$ is a chosen subset of unlocked public state cells of A_1, A_2 such that the values of those cells haven't been extended into the substitutions of A_1, A_2.

Definition 31. *Labelled bisimilarity ($\approx_l$) is the largest symmetric relation $\mathcal{R}$ between pairs of closed extended processes $A_i = \nu\widetilde{n}_i.(\sigma_i, S_i, \mathcal{P}_i)$ with $i = 1, 2$ such that $A_1 \,\mathcal{R}\, A_2$ implies*

1. *$locks(A_1) = locks(A_2)$, $fs(A_1) = fs(A_2)$ and $dom(A_1) = dom(A_2)$;*
2. *Select a fresh base variable x_s for each $s \in \mathtt{esc}(A_1, A_2)$. Let*

$$A_i^e = \nu\widetilde{n}_i.(\sigma_i \cup \{S_i(s)/x_s\}_{s \in \mathtt{esc}(A_1,A_2)}, S_i, \mathcal{P}_i) \text{ for } i = 1, 2$$

Then

(a) $A_1^e \approx_s A_2^e$;

(b) if $A_1^e \xrightarrow{s:=N}\xrightarrow{\tau(s)} B_1$ with $var(N) \subseteq dom(A_1^e)$, then there exists B_2 such that $A_2^e \xrightarrow{s:=N}\xRightarrow{\tau(s)} B_2$ and $B_1 \,\mathcal{R}\, B_2$;

(c) if $A_1^e \xrightarrow{\alpha} B_1$ and $fv(\alpha) \subseteq dom(A_1^e)$ and $bnv(\alpha) \cap fnv(A_2^e) = \emptyset$, then there exists B_2 such that $A_2^e \stackrel{\widehat{\alpha}}{\Longrightarrow} B_2$ and $B_1 \mathcal{R} B_2$.

The static equivalence $A_1^e \approx_s A_2^e$ in Definition 31 is exactly the same as the one defined in Definition 4. Before we compare the static equivalence and the transitions in labelled bisimilarity, we extend A_i to A_i^e with values from unlocked public state cells. This is to reflect the fact that attacker's ability to read values from these cells.

Example 32. *Consider the extended processes A and B in Example 25. As we have already shown, A and B are not observationally equivalent. Hence they are not supposed to be labelled bisimilar. We first extend A and B to A^e and B^e respectively:*

$$A^e = (\{0/z\}, \{s \mapsto 0\}, \{(\,!\, s := 0, \emptyset), (\,!\, s := 1, \emptyset)\})$$
$$B^e = (\{1/z\}, \{s \mapsto 1\}, \{(\,!\, s := 0, \emptyset), (\,!\, s := 1, \emptyset)\})$$

Clearly the static equivalence between A^e and B^e does not hold, namely $A^e \not\approx_s B^e$, because the test $z = 0$ can distinguish them. Thus we have $A \not\approx_l B$.

The extension is not only for comparing the static equivalence, but also for comparing the transitions. In labelled bisimilarity, we compare the transitions starting from the extensions A^e and B^e, rather than the original processes A and B. The reason is that we need to keep a copy of the cell values, otherwise we would lose the values when someone overwrites the cells.

Example 33. *Consider the extended processes A and B in Example 26. The extension A^e of A can perform the following transition:*

$$\begin{aligned} A^e = \ & \nu k.(\{k/z\}, \{s \mapsto k\}, \{(s := 0.a(x).\mathtt{if}\ x = k\ \mathtt{then}\ \overline{c}\langle b\rangle, \emptyset)\}) \\ \xrightarrow{\tau(s)} \ & \nu k.(\{k/z\}, \{s \mapsto 0\}, \{(a(x).\mathtt{if}\ x = k\ \mathtt{then}\ \overline{c}\langle b\rangle, \emptyset)\}) \\ \xRightarrow{a(z)} \ & \nu k.(\{k/z\}, \{s \mapsto 0\}, \{(\overline{c}\langle b\rangle, \emptyset)\}) \\ \xrightarrow{\overline{c}\langle b\rangle} \ & \nu k.(\{k/z\}, \{s \mapsto 0\}, \{(0, \emptyset)\}) \end{aligned}$$

But it is impossible for B's extension $B^e = \nu k.(\{k/z\}, \{s \mapsto k\}, \{(s := 0.\ a(x), \emptyset)\})$ to perform an output on channel c. Hence $A \not\approx_l B$.

We use $\xrightarrow{s:=N}\xrightarrow{\tau(s)}$ rather than $\xrightarrow{\tau(s)}$ in labelled bisimilarity because the attacker can set any unlocked public state cell to an arbitrary value. We shall illustrate this point by the following two examples.

Example 34. *Assume*

$$A = (\{0/y, 1/z\}, \{s \mapsto 0\}, \{(\mathtt{read}\ s\ \mathtt{as}\ x.\ \mathtt{if}\ x = 1\ \mathtt{then}\ \overline{c}\langle 0\rangle, \emptyset)\})$$
$$B = (\{0/y, 1/z\}, \{s \mapsto 0\}, \emptyset)$$

A and B are not observationally equivalent. Applying context $\mathcal{C} = (\emptyset, \emptyset, \{(s := 1, \emptyset)\})$ to A and B, we can see that $\mathcal{C}[A] \Downarrow_c$ but $\mathcal{C}[B] \not\Downarrow_c$.

Now we shall distinguish them in labelled bisimilarity. Since the current value in cell s is 0 which has already been stored in variable y, we don't need to extend A and B. Then A can perform the following transition

$$\begin{aligned} A &\xrightarrow{s:=1}\xrightarrow{\tau(s)} (\{0/y, 1/z\}, \{s \mapsto 1\}, \{(\texttt{if } 1 = 1 \texttt{ then } \overline{c}\langle a\rangle, \emptyset)\}) \\ &\xrightarrow{\overline{c}\langle a\rangle} (\{0/y, 1/z\}, \{s \mapsto 1\}, \{0, \emptyset\}) \end{aligned}$$

But there is no way for B to perform an output action. Hence $A \not\approx_l B$.

Example 35. *As illustrated in Example 27, A and B are not observationally equivalent. In labelled bisimilarity, we first extend A and B to A_1^e and B_1^e:*

$$\begin{aligned} A_1^e &= (\{0/x\}, \{s \mapsto 0\}, \{(s := 0.s := 0, \emptyset)\}) \\ B_1^e &= (\{0/x\}, \{s \mapsto 0\}, \{(s := 0, \emptyset)\}) \end{aligned}$$

Then let A_1^e perform actions $\xrightarrow{s:=1}\xrightarrow{\tau(s)}$,

$$A_1^e \xrightarrow{s:=1}\xrightarrow{\tau(s)} A_2^e = (\{0/x\}, \{s \mapsto 0\}, \{(s := 0, \emptyset)\})$$

Note that action $\xrightarrow{s:=1}$ sets the value of cell s to 1. Hence, B_1^e can only match the above transition by resetting the value of cell s to 0:

$$B_1^e \xrightarrow{s:=1}\xrightarrow{\tau(s)} B_2^e = (\{0/x\}, \{s \mapsto 0\}, \{(0, \emptyset)\})$$

Since the values of cell s in A_2^e and B_2^e are still 0 which have already been stored in variable x, we don't need to extend them again. Then let A_2^e perform the actions $\xrightarrow{s:=1}\xrightarrow{\tau(s)}$:

$$A_2^e \xrightarrow{s:=1}\xrightarrow{\tau(s)} A_3^e = (\{0/x\}, \{s \mapsto 0\}, \{(0, \emptyset)\})$$

But now what B_2^e can do is just

$$B_2^e \xrightarrow{s:=1}\Longrightarrow B_3^e = (\{0/x\}, \{s \mapsto 1\}, \{(0, \emptyset)\})$$

Extending A_2^e and B_2^e to the following A' and B':

$$\begin{aligned} A' &= (\{0/x, 0/z\}, \{s \mapsto 0\}, \{(0, \emptyset)\}) \\ B' &= (\{0/x, 1/z\}, \{s \mapsto 1\}, \{(0, \emptyset)\}) \end{aligned}$$

We can see that $A' \not\approx_s B'$ because the test $z = 0$ can distinguish them. Thus A and B are not labelled bisimilar, i.e. $A \not\approx_l B$.

Note that the transition $\xrightarrow{s:=N}$ is not included in $\xrightarrow{\alpha}$. We only need to use $\xrightarrow{s:=N}$ to change the value of the unlocked public state cell s when the processes perform some actions related to s. Comparing the combination of two transitions together ($\xrightarrow{s:=N}\xrightarrow{\tau(s)}$) in Definition 31 optimises the definition to be better suited as an assisted tool for analysing observational equivalence. Otherwise, if we follow the traditional way to define labelled bisimilarity, i.e. comparing $A_1^e \xrightarrow{s:=N} B_1^e$ and $A_1^e \xrightarrow{\tau(s)} B_1^e$ separately, the action $\xrightarrow{s:=N}$ would generate infinitely many unnecessary branches. For example, let $A = (\emptyset, \{s \mapsto 0\}, \emptyset)$. Even there is no action, A could keep on performing $\xrightarrow{s:=N}$ and would never stop: $A \xrightarrow{s:=1} (\emptyset, \{s \mapsto 1\}, \emptyset) \xrightarrow{s:=2} (\emptyset, \{s \mapsto 2\}, \emptyset) \xrightarrow{s:=3} (\emptyset, \{s \mapsto 3\}, \emptyset) \cdots$

We require $A_1^e \xrightarrow{s:=N}\xrightarrow{\tau(s)} B_1$ to be matched by $A_2^e \xrightarrow{s:=N}\overset{\tau(s)}{\Longrightarrow} B_2$ with the same s in the action in labelled bisimilarity. In other words, A_2^e can only match the transition of A_1^e by at most operating on the same cell s. This is equal to say the attacker holds the locks of all the unlocked public cell except cell s in A_1^e. If A_1^e does not do act on cell s, then A_2^e are not allowed to match A_1^e by operating on s.

Example 36. *Extend A and B in Example 24 to $A^e = (\{0/z\}, \{s \mapsto 0\}, \{(\overline{c}\langle b\rangle, \emptyset)\})$ and $B^e = (\{0/z\}, \{s \mapsto 0\}, \{(\texttt{read}\ s\ \texttt{as}\ x.\ \overline{c}\langle b\rangle, \emptyset)\})$. We can see that $A^e \xrightarrow{\overline{c}\langle b\rangle} (\emptyset, \{s \mapsto 0\}, \{(0, \emptyset)\})$, but there is no way for B^e to do the same output action $\overline{c}\langle b\rangle$ without going through the reading on cell s. Hence $A \not\approx_l B$.*

6.3. Soundness and Completeness

In this section, we will show our labelled bisimilarity given in Definition 31 can fully capture the observational equivalence given in Definition 23.

The following lemma states that labelled bisimilarity is closed when adding substitutions for terms stored in extensible state cells:

Lemma 37. *Assume $A_1 \approx_l A_2$ where $A_i = \nu\widetilde{n}_i.(\sigma_i, S_i, \mathcal{P}_i)$ for $i = 1, 2$. Assume $\texttt{esc}(A_1, A_2) = \{s_k\}_{k\in I}$ and $\{s_k \mapsto M_k^i\}_{k\in I} \subseteq S_i$ for some terms M_k^i. Select fresh variables $\{z_k\}_{k\in I}$, then*

$$\nu\widetilde{n}_1.(\sigma_1 \cup \{M_k^1/z_k\}_{k\in I}, S_1, \mathcal{P}_1) \approx_l \nu\widetilde{n}_2.(\sigma_2 \cup \{M_k^2/z_k\}_{k\in I}, S_2, \mathcal{P}_2)$$

Proof. We construct the following set $\mathcal{R}$:

$$\begin{aligned}&\mathcal{R} :=\\ &\{\, (\nu\widetilde{n}_1.(\sigma_1 \cup \{M_k^1/z_k\}_{k\in I}, S_1, \mathcal{P}_1),\ \ \nu\widetilde{n}_2.(\sigma_2 \cup \{M_k^2/z_k\}_{k\in I}, S_2, \mathcal{P}_2)) \mid\\ &\quad A_1 \approx_l A_2 \text{ where } A_i = \nu\widetilde{n}_i.(\sigma_i, S_i, \mathcal{P}_i) \text{ for } i = 1, 2, \{s_k\}_{k\in I} = \texttt{esc}(A_1, A_2),\\ &\qquad\qquad \{s_k \mapsto M_k^i\}_{k\in I} \subseteq S_i \text{ for } i = 1, 2, \{z_k\}_{k\in I} \text{ are fresh variables}\,\} \ \bigcup \approx_l\end{aligned}$$

We shall prove $\mathcal{R} \subseteq \approx_l$. Let $B_i = \nu\widetilde{n}_i.(\sigma_i \cup \{M_k^i/z_k\}_{k\in I}, S_i, \mathcal{P}_i)$ for $i = 1, 2$. According to the definition of extensible state cells, we can easily see that $\texttt{esc}(B_1, B_2) = \emptyset$. Hence we do not need to extend B_1, B_2 when comparing them for labelled bisimilarity. In other words, B_1, B_2 are both extensions of A_1, A_2 and B_1, B_2. Since $A_1 \approx_l A_2$, we have $B_1 \approx_s B_2$ by Definition 31.

Now we proceed to check the behaviour equivalence between B_1 and B_2.

1. Assume $B_1 \xrightarrow{s:=N} \xrightarrow{\tau(s)} B'_1$ with $var(N) \subseteq dom(B_1)$ and s public and unlocked. Since $A_1 \approx_l A_2$ and their extensions are B_1, B_2, we know there exists B'_2 such that $B_2 \xrightarrow{s:=N} \xRightarrow{\tau(s)} B'_2 \approx_l B'_1$. By the construction of $\mathcal{R}$, we know $(B'_1, B'_2) \in \mathcal{R}$.

2. Assume $B_1 \xrightarrow{\alpha} B'_1$ with $fv(\alpha) \subseteq dom(A_1)$ and $bnv(\alpha) \cap fnv(B_2) = \emptyset$. Since $A_1 \approx_l A_2$ and their extensions are B_1, B_2, we know there exists B'_2 such that $B_2 \xRightarrow{\widehat{\alpha}} B'_2 \approx_l B'_1$. According to the construction of $\mathcal{R}$, we know $(B'_1, B'_2) \in \mathcal{R}$.

Now we proceed to prove the soundness of our labelled bisimilarity for public state cells:

Proposition 38 (Soundness). *If $A \approx_l B$ then $A \approx B$.*

Proof. It is sufficient to prove that $\approx_l$ is a congruence. We construct the following set:

$$\mathcal{R} = \{\, (\mathcal{C}[A_1]_{\backslash \widetilde{x}}, \mathcal{C}[A_2]_{\backslash \widetilde{x}}) \mid A_1 \approx_l A_2, \text{ a closing evaluation context } \mathcal{C},\ \widetilde{x} \subseteq dom(A_1) \,\}$$

and prove that $\mathcal{R} \subseteq \approx_l$. Note that this is sufficient for proving $\approx_l \subseteq \approx$. For any $A\mathcal{R}B$, because $\mathcal{R} \subseteq \approx_l$, we have $A \approx_l B$. Then we can easily check the conditions (i), (ii), (iii) in Definition 23 hold. For the condition (iv), since $A \approx_l B$, by the construction of $\mathcal{R}$, we can see $(\mathcal{C}[A], \mathcal{C}[B]) \in \mathcal{R}$ by letting $\widetilde{x} = \emptyset$. Therefore $\mathcal{R} \subseteq \approx$.

Assume $(\mathcal{C}[A_1]_{\backslash \widetilde{x}}, \mathcal{C}[A_2]_{\backslash \widetilde{x}}) \in \mathcal{R}$ because of $A_1 \approx_l A_2$ where $\mathcal{C} = \nu\widetilde{n}.(\sigma\text{-}, S\text{-}, \mathcal{P}\text{-})$ and $A_i = \nu\widetilde{n}_i.(\sigma_i, S_i, \mathcal{P}_i)$ with $i = 1, 2$. By Definition 31, we will first extend $\mathcal{C}[A_1]_{\backslash \widetilde{x}}, \mathcal{C}[A_2]_{\backslash \widetilde{x}}$ with substitutions for their extensible state cells, and then show the static equivalence and behavior equivalence between the extensions.

Assume the extensible state cells

$$\begin{aligned} \texttt{esc}(\mathcal{C}[A_1]_{\backslash \widetilde{x}}, \mathcal{C}[A_2]_{\backslash \widetilde{x}}) &= \{r_k\}_{k\in I_r} \cup \{s_k\}_{k\in I_s} \cup \{\delta_k\}_{k\in\Delta} \\ \texttt{esc}(A_1, A_2) &= \{s_k\}_{k\in I_s} \cup \{t_k\}_{k\in I_t} \end{aligned}$$

where $\{r_k\}_{k\in I_r} \subseteq dom(S)$, $\{\delta_k\}_{k\in\Delta} \subseteq dom(S_i)$ and $\{s_k\}_{k\in I_s} \subseteq dom(S_i)$ for $i = 1, 2$. Intuitively, $\{t_k\}_{k\in I_t}$ are the extensible state cells for A_1, A_2 but become inextensible because of the application of context $\mathcal{C}$ (for example, the context $\mathcal{C}$ may have a restriction νs which makes an extensible public cell s private, or $\mathcal{C}$ may introduce a substitution which has the value of the cell s). $\{\delta_k\}_{k\in\Delta}$ are the public cells from $dom(S_i)$, and are not extensible in A_i because of the substitutions on $\widetilde{x}$, but they become extensible in $\mathcal{C}[A_i]_{\backslash \widetilde{x}}$ because the substitutions on $\widetilde{x}$ are removed. By Definition 30 of extensible cells, there exists $\{x_{j_k}\}_{k\in\Delta}$ with $x_{j_k} \in \widetilde{x}$ and $S_i(\delta_k) = x_{j_k}\sigma_i$ for $k \in \Delta$ and $i = 1, 2$. Select pairwise-distinct fresh variables $\{z_{r_k}\}_{k\in I_r}, \{z_{s_k}\}_{k\in I_s}, \{z_{t_k}\}_{k\in I_t}, \{z_{\delta_k}\}_{k\in\Delta}$ and let $\sigma_r = \{S(r_k)/z_{r_k}\}_{k\in I_r}$ and $\sigma_s^i = \{S_i(s_k)/z_{s_k}\}_{k\in I_s}$ and $\sigma_t^i = \{S_i(t_k)/z_{t_k}\}_{k\in I_t}$ and $\widehat{\sigma} = \{x_{j_k}/z_{\delta_k}\}_{k\in\Delta}$. Let

$$\varphi_i = \sigma_i \cup \sigma_s^i \cup \sigma_t^i \qquad\qquad \varphi_i^e = \sigma\sigma_i \cup \sigma_r\sigma_i \cup \widehat{\sigma}\sigma_i \cup \sigma_{i\backslash\widetilde{x}} \cup \sigma_s^i$$

Then we extend process A_i by adding substitutions for extensible state cells, i.e., σ_s^i and σ_t^i, with $i = 1, 2$:

$$B_i := \nu\widetilde{n}_i.(\varphi_i, S_i, \mathcal{P}_i)$$

Since $A_1 \approx_l A_2$, using Lemma 37, we get $B_1 \approx_l B_2$. Also we extend process $\mathcal{C}[A_i]_{\backslash \widetilde{x}}$ by adding substitutions for extensible state cells, i.e., $\sigma_r\sigma_i \cup \widehat{\sigma}\sigma_i \cup \sigma_s^i$, for $i = 1, 2$:

$$D_i := \nu\widetilde{n}, \widetilde{n}_i.(\varphi_i^e, S\sigma_i \cup S_i, \mathcal{P}\sigma_i \cup \mathcal{P}_i)$$

We first prove the static equivalence $D_1 \approx_s D_2$. Assume terms N_1, N_2 with $var(N_1, N_2) \subseteq dom(\varphi_1^e)$ and $N_1\varphi_1^e =_\Sigma N_2\varphi_1^e$, we will show that $N_1\varphi_2^e =_\Sigma N_2\varphi_2^e$. We can see that $N_k\varphi_i^e = N_k(\sigma\sigma_i \cup \sigma_r\sigma_i \cup \widehat{\sigma}\sigma_i \cup \sigma_{i\backslash\widetilde{x}} \cup \sigma_s^i) = (N_k(\sigma \cup \sigma_r \cup \widehat{\sigma}))(\sigma_i \cup \sigma_s^i)$ for $k = 1, 2$ and $i = 1, 2$. Since $\mathcal{C}$ closes A_i, we can see that $var(N_k(\sigma \cup \sigma_r \cup \widehat{\sigma})) \subseteq dom(\varphi_i)$ for $k = 1, 2$. Thus we have $N_k\varphi_i^e = (N_k(\sigma \cup \sigma_r \cup \widehat{\sigma}))\varphi_i$. From the hypothesis $N_1\varphi_1^e =_\Sigma N_2\varphi_1^e$, we know $(N_1(\sigma \cup \sigma_r \cup \widehat{\sigma}))\varphi_1 =_\Sigma (N_2(\sigma \cup \sigma_r \cup \widehat{\sigma}))\varphi_1$. From $B_1 \approx_s B_2$, we know that $(N_1(\sigma \cup \sigma_r \cup \widehat{\sigma}))\varphi_2 =_\Sigma (N_2(\sigma \cup \sigma_r \cup \widehat{\sigma}))\varphi_2$. From $N_k\varphi_i^e = (N_k(\sigma \cup \sigma_r \cup \widehat{\sigma}))\varphi_i$, we know that $N_1\varphi_2^e =_\Sigma N_2\varphi_2^e$. Hence $D_1 \approx_s D_2$.

Now we proceed to prove the behavioural equivalence between D_1 and D_2.

1. Assume $D_1 \xrightarrow{s:=N}\xrightarrow{\tau(s)} D_1'$ with $var(N) \subseteq dom(D_1)$. We only detail the proof for the case that s is an unlocked public cell in D_1. The analysis for the case when s is locked or bound is similar. Cell name s comes either from context, i.e. $s \in dom(S)$, or from process A_1, i.e. $s \in dom(S_1)$.

(a) Assume s comes from the context, i.e., $S = S' \cup \{s \mapsto M\}$.Then

$$\begin{aligned} D_1 &= \nu\widetilde{n}, \widetilde{n}_1.(\varphi_1^e, S'\sigma_1 \cup \{s \mapsto M\sigma_1\} \cup S_1, \mathcal{P}\sigma_1 \cup \mathcal{P}_1) \\ &\xrightarrow{s:=N} \nu\widetilde{n}, \widetilde{n}_1.(\varphi_1^e, S'\sigma_1 \cup \{s \mapsto N\varphi_1^e\} \cup S_1, \mathcal{P}\sigma_1 \cup \mathcal{P}_1) \xrightarrow{\tau(s)} D_1' \end{aligned}$$

We shall discuss the different cases of $\tau(s)$. Because s is a unlocked public cell, $\tau(s)$ can be locking the cell s, or reading the cell s, or writing a term to the cell s. Since s is from the context, these actions should also come from the processes in the context, i.e., from $\mathcal{P}$.

i. if $\mathcal{P} = \mathcal{P}' \cup \{(\texttt{lock}\ s.P, L)\}$, then $D_1' = \nu\widetilde{n}, \widetilde{n}_1.(\varphi_1^e, S'\sigma_1 \cup \{s \mapsto N\varphi_1^e\} \cup S_1, \mathcal{P}'\sigma_1 \cup \{(P\sigma_1, L \cup \{s\})\} \cup \mathcal{P}_1)$. We construct a new evaluation context $\mathcal{C}' = \nu\widetilde{n}.(\sigma \cup \sigma_r \cup \widehat{\sigma}\text{-}, S' \cup \{s \mapsto N(\sigma \cup \sigma_r \cup \widehat{\sigma})\}\text{-}, \mathcal{P}' \cup \{(P, L \cup \{s\})\}\text{-})$. Since $var(N) \subseteq dom(\varphi_1^e)$, we have $var(N(\sigma \cup \sigma_r \cup \widehat{\sigma})) \subseteq dom(\sigma_i, \sigma_s^i)$. We can see that $N\varphi_i^e = (N(\sigma \cup \sigma_r \cup \widehat{\sigma}))(\sigma_i \cup \sigma_s^i)$ for $i = 1, 2$. We can verify that $D_1' = \mathcal{C}'[B_1]_{\backslash \widetilde{x}, \widetilde{z}_t}$ and

$$\begin{aligned} &D_2 = \nu\widetilde{n}, \widetilde{n}_2.(\varphi_2^e, S'\sigma_2 \cup \{s \mapsto M\sigma_2\} \cup S_2, \mathcal{P}\sigma_2 \cup \mathcal{P}_2) \\ &\xrightarrow{s:=N} \nu\widetilde{n}, \widetilde{n}_2.(\varphi_2^e, S'\sigma_2 \cup \{s \mapsto N\varphi_2^e\} \cup S_2, \mathcal{P}\sigma_2 \cup \mathcal{P}_2) \xrightarrow{\tau(s)} D_2' = \mathcal{C}'[B_2]_{\backslash \widetilde{x}, \widetilde{z}_t} \end{aligned}$$

From $B_1 \approx_l B_2$ and the construction of $\mathcal{R}$, we have $(D_1', D_2') \in \mathcal{R}$.

ii. if $\mathcal{P} = \mathcal{P}' \cup \{(\texttt{read}\ s\ \texttt{as}\ y.P, L)\}$, then $D_1' = \nu\widetilde{n}, \widetilde{n}_1.(\varphi_1^e, S'\sigma_1 \cup \{s \mapsto N\varphi_1^e\} \cup S_1, \mathcal{P}'\sigma_1 \cup \{((P\sigma_1)\{N\varphi_1^e/y\}, L)\} \cup \mathcal{P}_1)$. We construct a context $\mathcal{C}' = \nu\widetilde{n}.(\sigma \cup \sigma_r \cup \widehat{\sigma}\text{-}, S' \cup \{s \mapsto N(\sigma \cup \sigma_r \cup \widehat{\sigma})\}\text{-}, \mathcal{P}' \cup \{(P\{N(\sigma \cup \sigma_r \cup \widehat{\sigma})/y\}, L)\}\text{-})$. The rest of analysis is similar to case i.

iii. if $\mathcal{P} = \mathcal{P}' \cup \{(s := N'.P, L)\}$, then $D_1' = \nu\widetilde{n}, \widetilde{n}_1.(\varphi_1^e, S'\sigma_1 \cup \{s \mapsto N'\sigma_1\} \cup S_1, \mathcal{P}'\sigma_1 \cup \{(P\sigma_1, L)\} \cup \mathcal{P}_1)$. Let $\mathcal{C}' = \nu\widetilde{n}.(\sigma \cup \sigma_r \cup \widehat{\sigma}\text{-}, S' \cup \{s \mapsto N'\}\text{-}, \mathcal{P}' \cup \{(P, L)\}\text{-})$. The rest of analysis is similar to case i.

(b) Assume s comes from A_i and $S_i = S_i' \cup \{s \mapsto M_i\}$ with $i = 1, 2$. Then

$$\begin{aligned} D_1 &= \nu\widetilde{n}, \widetilde{n}_1.(\varphi_1^e, S\sigma_1 \cup S_1' \cup \{\boldsymbol{s \mapsto M_1}\}, \mathcal{P}\sigma_1 \cup \mathcal{P}_1) \\ &\xrightarrow{s:=N} D_1' = \nu\widetilde{n}, \widetilde{n}_1.(\varphi_1^e, S\sigma_1 \cup S_1' \cup \{\boldsymbol{s \mapsto N\varphi_1^e}\}, \mathcal{P}\sigma_1 \cup \boldsymbol{\mathcal{P}_1}) \\ &\xrightarrow{\tau(s)} D_1'' = \nu\widetilde{n}, \widetilde{n}_1.(\varphi_1^e, S\sigma_1 \cup S_1' \cup \{\boldsymbol{s \mapsto N_1}\}, \mathcal{P}\sigma_1 \cup \boldsymbol{\mathcal{P}_1'}) \end{aligned}$$

The transition $D_1' \xrightarrow{\tau(s)} D_1''$ operates on the cell s which has nothing to do with the context part. So we can have that

$$\begin{aligned} B_1 &= \nu\widetilde{n}_1.(\varphi_1, S_1' \cup \{\boldsymbol{s \mapsto M_1}\}, \mathcal{P}_1) \\ &\xrightarrow{s:=N(\sigma\cup\sigma_r\cup\widehat{\sigma})} C_1' = \nu\widetilde{n}_1.(\varphi_1, S_1' \cup \{\boldsymbol{s \mapsto N\varphi_1^e}\}, \boldsymbol{\mathcal{P}_1}) \\ &\qquad \text{since } (N(\sigma \cup \sigma_r \cup \widehat{\sigma}))\varphi_1 = N\varphi_1^e \\ &\xrightarrow{\tau(s)} C_1'' = \nu\widetilde{n}_1.(\varphi_1, S_1' \cup \{\boldsymbol{s \mapsto M_1'}\}, \boldsymbol{\mathcal{P}_1'}) \end{aligned}$$

Let $\mathcal{C}' = \nu\widetilde{n}.(\sigma \cup \sigma_r \cup \widehat{\sigma}\text{-}, S\text{-}, \mathcal{P}\text{-})$. We can verify that $D_1'' = \mathcal{C}'[C_1'']_{\setminus\widetilde{x},\widetilde{z}_t}$. Since $A_1 \approx_l A_2$, there exists C_2'' such that

$$\begin{aligned} B_2 &= \nu\widetilde{n}_2.(\varphi_2, S_2' \cup \{\boldsymbol{s \mapsto M_2}\}, \mathcal{P}_2) \\ &\xrightarrow{s:=N(\sigma\cup\sigma_r\cup\widehat{\sigma})} C_2' = \boldsymbol{\nu\widetilde{n}_2}.(\varphi_2, \boldsymbol{S_2'} \cup \{\boldsymbol{s \mapsto N\varphi_2^e}\}, \boldsymbol{\mathcal{P}_2}) \\ &\qquad \text{since } (N(\sigma \cup \sigma_r \cup \widehat{\sigma}))\varphi_2 = N\varphi_2^e \\ &\overset{\tau(s)}{\Longrightarrow} C_2'' = \boldsymbol{\nu\widetilde{n}'}.(\varphi_2, \boldsymbol{S_2''}, \boldsymbol{\mathcal{P}_2'}) \end{aligned}$$

and $C_1'' \approx_l C_2''$. Applying the context $\mathcal{C}'$ and removing variables $\widetilde{x}, \widetilde{z}_t$,

$$\begin{aligned} D_2 = \mathcal{C}'[B_2]_{\setminus\widetilde{x},\widetilde{z}_t} &= \nu\widetilde{n}, \widetilde{n}_2.(\varphi_2^e, S\sigma_2 \cup S_2' \cup \{\boldsymbol{s \mapsto M_2}\}, \mathcal{P}\sigma_2 \cup \mathcal{P}_2) \\ \xrightarrow{s:=N} \mathcal{C}'[C_2']_{\setminus\widetilde{x},\widetilde{z}_t} &= \nu\widetilde{n}, \boldsymbol{\widetilde{n}_2}.(\varphi_2^e, S\sigma_2 \cup \boldsymbol{S_2'} \cup \{\boldsymbol{s \mapsto N\varphi_2^e}\}, \mathcal{P}\sigma_2 \cup \boldsymbol{\mathcal{P}_2}) \\ \overset{\tau(s)}{\Longrightarrow} D_2'' = \mathcal{C}'[C_2'']_{\setminus\widetilde{x},\widetilde{z}_t} &= \nu\widetilde{n}, \boldsymbol{\nu\widetilde{n}'}.(\varphi_2^e, S\sigma_2 \cup \boldsymbol{S_2''}, \mathcal{P}\sigma_2 \cup \boldsymbol{\mathcal{P}_2'}) \end{aligned}$$

Since $C_1'' \approx_l C_2''$, $D_1'' = \mathcal{C}'[C_1'']_{\setminus\widetilde{x},\widetilde{z}_t}$ and $D_2'' = \mathcal{C}'[C_2'']_{\setminus\widetilde{x},\widetilde{z}_t}$, by the construction of $\mathcal{R}$, we have $(D_1'', D_2'') \in \mathcal{R}$.

2. Assume $D_1 \xrightarrow{a(N)} D_1'$ with $var(N) \subseteq dom(D_1)$. The input action comes either from context part or from the process part.

(a) Assume the input action is from the context part, i.e., $\mathcal{P} = \mathcal{P}' \cup \{(a(x).P, L)\}$.

$$\begin{aligned} D_1 &= \nu\widetilde{n}, \widetilde{n}_1.(\varphi_1^e, S\sigma_1 \cup S_1, \mathcal{P}'\sigma_1 \cup \{(\boldsymbol{a(x).P\sigma_1}, L)\} \cup \mathcal{P}_1) \\ &\xrightarrow{a(N)} D_1' = \nu\widetilde{n}, \widetilde{n}_1.(\varphi_1^e, S\sigma_1 \cup S_1, \mathcal{P}'\sigma_1 \cup \{(\boldsymbol{P\sigma_1 \{N\varphi_1^e/x\}}, L)\} \cup \mathcal{P}_1) \end{aligned}$$

We construct a new evaluation context

$$\mathcal{C}' = \nu\widetilde{n}.(\sigma \cup \sigma_r \cup \widehat{\sigma}\text{-}, S\text{-}, \mathcal{P}' \cup \{(P\{N(\sigma \cup \sigma_r \cup \widehat{\sigma})/x\}, L)\}\text{-})$$

We can verify that $\mathcal{C}'[B_1]_{\setminus\widetilde{x},\widetilde{z}_t} = D_1'$ and $D_2 \xrightarrow{a(N)} D_2' = \mathcal{C}'[B_2]_{\setminus\widetilde{x},\widetilde{z}_t}$. Thus we have $(D_1', D_2') \in \mathcal{R}$.

(b) Assume the input action is from the process part, i.e., $\mathcal{P}_1 = \mathcal{P}_1' \cup \{(a(x).P_1, L)\}$

$$\begin{aligned} D_1 &= \nu\widetilde{n}, \widetilde{n}_1.(\varphi_1^e, S\sigma_1 \cup S_1, \mathcal{P}\sigma_1 \cup \mathcal{P}_1' \cup \{(\boldsymbol{a(x).P_1}, L)\}) \\ &\xrightarrow{a(N)} D_1' = \nu\widetilde{n}, \widetilde{n}_1.(\varphi_1^e, S\sigma_1 \cup S_1, \mathcal{P}\sigma_1 \cup \mathcal{P}_1' \cup \{(\boldsymbol{P_1\{N\varphi_1^e/x\}}, L)\}) \end{aligned}$$

And we have the input from B_1:

$$\begin{aligned} B_1 &= \nu\widetilde{n}_1.(\varphi_1, S_1, \mathcal{P}_1' \cup \{(\boldsymbol{a(x).P_1}, L_1)\}) \\ &\xrightarrow{a(N(\sigma\cup\sigma_r\cup\widehat{\sigma}))} C_1 = \nu\widetilde{n}_1.(\varphi_1, S_1, \mathcal{P}_1' \cup \{(\boldsymbol{P_1\{N\varphi_1^e/x\}}, L_1)\}) \\ &\qquad \text{since } (N(\sigma \cup \sigma_r \cup \widehat{\sigma})\varphi_1 = N\varphi_1^e \end{aligned}$$

Let $\mathcal{C}' = \nu\widetilde{n}.(\sigma \cup \sigma_r \cup \widehat{\sigma}\text{-}, S\text{-}, \mathcal{P}\text{-})$. We can verify that $D_1 = \mathcal{C}'[B_1]_{\setminus\widetilde{x},\widetilde{z}_t}$ and $D_1' = \mathcal{C}'[C_1]_{\setminus\widetilde{x},\widetilde{z}_t}$. Since $A_1 \approx_l A_2$, we should have the following transitions from A_2's extension B_2

$$\begin{aligned} B_2 &= \nu\widetilde{n}_2.(\varphi_2, \boldsymbol{S_2}, \boldsymbol{\mathcal{P}_2}) \\ &\Longrightarrow C_3 = \nu\widetilde{n}_2'.(\varphi_2, \boldsymbol{S_2'}, \boldsymbol{\mathcal{P}_2'} \cup \{\boldsymbol{a(x).P_2}, L_2)\}) \\ &\xrightarrow{a(N(\sigma\cup\sigma_r\cup\widehat{\sigma}))} C_4 = \boldsymbol{\nu\widetilde{n}_2', \widetilde{m}}.(\varphi_2, \boldsymbol{S_2'}, \boldsymbol{\mathcal{P}_2'} \cup \{(\boldsymbol{P_2^c\{N\varphi_2^e/x\}}, L_2)\}) \\ &\qquad \text{since } (N(\sigma \cup \sigma_r \cup \widehat{\sigma})\varphi_2 = N\varphi_2^e \\ &\Longrightarrow C_2 = \boldsymbol{\nu\widetilde{n}_2''}.(\varphi_2, \boldsymbol{S_2''}, \boldsymbol{\mathcal{P}_2''}) \end{aligned}$$

and $C_1 \approx_l C_2$. Applying $\mathcal{C}'$ to the transitions $B_2 \Longrightarrow C_3$ and $C_4 \Longrightarrow C_2$ and remove the variables $\widetilde{x}, \widetilde{z}_t$, we will get

$$\begin{aligned} D_2 &= \mathcal{C}'[B_2]_{\setminus\widetilde{x},\widetilde{z}_t} = \nu\widetilde{n}, \boldsymbol{\widetilde{n}_2}.(\varphi_2^e, S\sigma_2 \cup \boldsymbol{S_2}, \mathcal{P}\sigma_2 \cup \boldsymbol{\mathcal{P}_2}) \\ &\Longrightarrow \mathcal{C}'[C_3]_{\setminus\widetilde{x},\widetilde{z}_t} = \nu\widetilde{n}, \boldsymbol{\widetilde{n}_2'}.(\varphi_2^e, S\sigma_2 \cup \boldsymbol{S_2'}, \mathcal{P}\sigma_2 \cup \boldsymbol{\mathcal{P}_2'} \cup \{(\boldsymbol{a(x).P_2}, L_2)\}) \\ &\xrightarrow{a(N)} \mathcal{C}'[C_4]_{\setminus\widetilde{x},\widetilde{z}_t} = \nu\widetilde{n}, \boldsymbol{\widetilde{n}_2'}.(\varphi_2^e, S\sigma_2 \cup \boldsymbol{S_2'}, \mathcal{P}\sigma_2 \cup \boldsymbol{\mathcal{P}_2'} \cup \{(\boldsymbol{P_2\{N\varphi_2^e/x\}}, L_2)\}) \\ &\Longrightarrow D_2' = \mathcal{C}'[C_2]_{\setminus\widetilde{x},\widetilde{z}_t} = \nu\widetilde{n}, \boldsymbol{\widetilde{n}_2'', \widetilde{m}}.(\varphi_2^e, S\sigma_2 \cup \boldsymbol{S_2''}, \mathcal{P}\sigma_2 \cup \boldsymbol{\mathcal{P}_2''}) \end{aligned}$$

Since $D_1' = \mathcal{C}'[C_1]_{\setminus\widetilde{x},\widetilde{z}_t}$ and $D_2' = \mathcal{C}'[C_2]_{\setminus\widetilde{x},\widetilde{z}_t}$ and $C_1 \approx_l C_2$, we have $(D_1', D_2') \in \mathcal{R}$.

3. Assume $D_1 \xrightarrow{\nu y.\overline{a}\langle y\rangle} D_1'$. The output action comes either from context part or from the process part.

(a) When the output comes from the context, assume $\mathcal{P} = \mathcal{P}' \cup \{(\overline{a}\langle N\rangle.P, L)\}$.

$$\begin{aligned} D_1 &= \nu\widetilde{n}, \widetilde{n}_1.(\varphi_1^e, S\sigma_1 \cup S_1, \mathcal{P}'\sigma_1 \cup \{(\boldsymbol{\overline{a}\langle N\sigma_1\rangle.P\sigma_1}, L)\} \cup \mathcal{P}_1) \\ &\xrightarrow{\nu y.\overline{a}\langle y\rangle} D_1' = \nu\widetilde{n}, \widetilde{n}_1.(\varphi_1^e \cup \boldsymbol{\{N\sigma_1/y\}}, S\sigma_1 \cup S_1, \mathcal{P}'\sigma_1 \cup \{(\boldsymbol{P\sigma_1}, L)\} \cup \mathcal{P}_1) \end{aligned}$$

We construct a new evaluation context $\mathcal{C}' = \nu\widetilde{n}.(\sigma \cup \sigma_r \cup \widehat{\sigma} \cup \{N/y\}\text{-}, S\text{-}, \mathcal{P}' \cup \{(P, L)\}\text{-})$. We can verify that $\mathcal{C}'[B_1]_{\backslash\widetilde{x},\widetilde{z}_t} = D_1'$ and $D_2 \xrightarrow{\nu y.\overline{a}\langle y\rangle} D_2' = \mathcal{C}'[B_2]_{\backslash\widetilde{x},\widetilde{z}_t}$. Thus we have $(D_1', D_2') \in \mathcal{R}$.

(b) When the output comes from the process, assume $\mathcal{P}_1 = \mathcal{P}_1' \cup \{(\overline{a}\langle N_1\rangle.P_1, L_1)\}$

$$\begin{aligned}D_1 &= \nu\widetilde{n}, \widetilde{n}_1.(\varphi_1^e, S\sigma_1 \cup S_1, \mathcal{P}\sigma_1 \cup \mathcal{P}_1' \cup \{(\boldsymbol{\overline{a}\langle N_1\rangle.P_1}, L)\})\\ &\xrightarrow{\nu y.\overline{a}\langle y\rangle} D_1' = \nu\widetilde{n}, \widetilde{n}_1.(\varphi_1^e \cup \boldsymbol{\{N_1/y\}}, S\sigma_1 \cup S_1, \mathcal{P}\sigma_1 \cup \mathcal{P}_1' \cup \{(\boldsymbol{P_1}, L_1)\})\end{aligned}$$

And we have the output from B_1:

$$\begin{aligned}B_1 &= \nu\widetilde{n}_1.(\varphi_1, S_1, \mathcal{P}_1' \cup \{(\boldsymbol{\overline{a}\langle N_1\rangle.P_1}, L_1)\})\\ &\xrightarrow{\nu y.\overline{a}\langle y\rangle} C_1 = \nu\widetilde{n}_1.(\varphi_1 \cup \boldsymbol{\{N_1/y\}}, S_1, \mathcal{P}_1' \cup \{(\boldsymbol{P_1}, L_1)\})\end{aligned}$$

Let $\mathcal{C}' = \nu\widetilde{n}.(\sigma \cup \sigma_r \cup \widehat{\sigma}\text{-}, S\text{-}, \mathcal{P}\text{-})$. We can verify that $D_1 = \mathcal{C}'[B_1]_{\backslash\widetilde{x},\widetilde{z}_t}$ and $D_1' = \mathcal{C}'[C_1]_{\backslash\widetilde{x},\widetilde{z}_t}$. Since $A_1 \approx_l A_2$, for the extension B_2, we should have

$$\begin{aligned}B_2 &= \boldsymbol{\nu\widetilde{n}_2}.(\varphi_2, \boldsymbol{S_2}, \boldsymbol{\mathcal{P}_2})\\ &\Longrightarrow C_3 = \boldsymbol{\nu\widetilde{n}_2'}.(\varphi_2, \boldsymbol{S_2'}, \boldsymbol{\mathcal{P}_2'} \cup \{(\boldsymbol{\overline{a}\langle N_2\rangle.P_2}, L_2)\})\\ &\xrightarrow{\nu y.\overline{a}\langle y\rangle} C_4 = \boldsymbol{\nu\widetilde{n}_2'}.(\varphi_2 \cup \boldsymbol{\{N_2/y\}}, \boldsymbol{S_2'}, \boldsymbol{\mathcal{P}_2'} \cup \{(\boldsymbol{P_2}, L_2)\})\\ &\Longrightarrow C_2 = \boldsymbol{\nu\widetilde{n}_2''}.(\varphi_2 \cup \{N_2/y\}, \boldsymbol{S_2''}, \boldsymbol{\mathcal{P}_2''})\end{aligned}$$

Applying context $\mathcal{C}'$ to the transitions $B_2 \Longrightarrow C_3$ and $C_4 \Longrightarrow C_2$ and remove the variables $\widetilde{x}, \widetilde{z}_t$, we will get

$$\begin{aligned}D_2 = \mathcal{C}'[B_2]_{\backslash\widetilde{x},\widetilde{z}_t} &= \nu\widetilde{n}, \boldsymbol{\widetilde{n}_2}.(\varphi_2^e, S\sigma_2 \cup S_2, \mathcal{P}\sigma_2 \cup \boldsymbol{\mathcal{P}_2})\\ \Longrightarrow \mathcal{C}'[C_3]_{\backslash\widetilde{x},\widetilde{z}_t} &= \nu\widetilde{n}, \boldsymbol{\widetilde{n}_2'}.(\varphi_2^e, S\sigma_2 \cup S_2', \mathcal{P}\sigma_2 \cup \boldsymbol{\mathcal{P}_2'} \cup \{(\boldsymbol{\overline{a}\langle N_2\rangle.P_2}, L_2)\})\\ \xrightarrow{\nu y.\overline{a}\langle y\rangle} \mathcal{C}'[C_4]_{\backslash\widetilde{x},\widetilde{z}_t} &= \nu\widetilde{n}, \boldsymbol{\widetilde{n}_2'}.(\varphi_2^e \cup \boldsymbol{\{N_2/y\}}, S\sigma_2 \cup \boldsymbol{S_2'}, \mathcal{P}\sigma_2 \cup \boldsymbol{\mathcal{P}_2'} \cup \{(\boldsymbol{P_2}, L_2)\})\\ \Longrightarrow D_2' = \mathcal{C}'[C_2]_{\backslash\widetilde{x},\widetilde{z}_t} &= \nu\widetilde{n}, \boldsymbol{\widetilde{n}_2''}.(\varphi_2^e \cup \{N_2/y\}, S\sigma_2 \cup \boldsymbol{S_2''}, \mathcal{P}\sigma_2 \cup \boldsymbol{\mathcal{P}_2''})\end{aligned}$$

Thus we have $(D_1', D_2') \in \mathcal{R}$.

(c) The analysis for the other cases when α is $\overline{a}\langle c\rangle$ or $\nu c.\overline{a}\langle c\rangle$ is similar.

Now we proceed to show the completeness of our labelled bisimilarity. Although $A \Downarrow_a$ is only defined for output action, we can easily test the existence of an input action $b(x)$ by using evaluation context $\mathcal{C} = (\text{-}, \text{-}, \{(\overline{e}, \emptyset), (\overline{b}.e)\}\text{-})$ where e is fresh. It is clear that:

Claim A can perform an input on channel b if and only if there exists B such that $\mathcal{C}[A] \Longrightarrow B$ and $B \not\Downarrow_e$.

Hence in the following proof, for notational convenience, we use the traditional notation $A \Downarrow_{\overline{b}}$ when $A \stackrel{\epsilon}{\Longrightarrow} \nu\widetilde{n}.(\sigma, S, \mathcal{P} \cup (\overline{b}\langle M\rangle.P, L))$ with $b \notin \widetilde{n}$, and use $A \Downarrow_b$ when $A \stackrel{\epsilon}{\Longrightarrow} \nu\widetilde{n}.(\sigma, S, \mathcal{P} \cup (b(x).P, L))$ with $b \notin \widetilde{n}$.

We write $A \not\Downarrow_{\gamma_1 \cdots, \gamma_i, \cdots \gamma_n}$ if $A \not\Downarrow_{\gamma_1} \cdots, A \not\Downarrow_{\gamma_i}, \cdots A \not\Downarrow_{\gamma_n}$ where γ_i is either a_i or $\overline{a}_i$ for some channel name a_i.

Lemma 39. *Assume* $A \xrightarrow{\tau}\xrightarrow{t:=N} A'$ *with* $t \in \mathit{unlocks}(A)$*, then* $A \xrightarrow{t:=N}\xrightarrow{\tau} A'$.

Proof. Since t is an unlocked public state cell in A, we can see that $\xrightarrow{\tau}$ defined in Figure 1 is irrelevant to t. $\xrightarrow{\tau}$ is only related to locked or restricted cells in A. So the conclusion holds obviously.

Corollary 40. *Assume* $A \Longrightarrow\xrightarrow{t:=N} A'$ *with* $t \in \mathit{unlocks}(A)$*, then* $A \xrightarrow{t:=N}\Longrightarrow A'$.

Proof. Recall that $\Longrightarrow$ is a reflexive and transitive closure of $\xrightarrow{\tau}$. We can get this corollary by using Lemma 39 several times.

Proposition 41 (Completeness). *If* $A \approx B$*, then* $A \approx_l B$.

Proof. We define a relation $\mathcal{R}$ as follows:

$$\begin{aligned}\mathcal{R} = \{\, (A_1, A_2) \mid A_i = \nu\widetilde{n}_i.(\sigma_i, S_i, \mathcal{P}_i) \text{ for } i = 1, 2,\\ \text{there exist pairwise-distinct channel names}\\ \widetilde{a}, \widetilde{b}, \widetilde{c}, \widetilde{read}, \widetilde{write}, \widetilde{tag} \text{ such that } \widehat{A}_1 \,\approx\, \widehat{A}_2 \,\}\end{aligned}$$

where

$$\widehat{A}_i := \nu\widetilde{c}, \widetilde{n}_i. \left(\sigma_{i\setminus W}, S_i, \begin{array}{c} \mathcal{P}_i \cup \left\{(\overline{a}_w\langle w\sigma_i\rangle, \emptyset)\right\}_{w\in W} \cup \left\{(\overline{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J} \\ \cup \left\{ \begin{array}{l} (\overline{read}_s\langle S_i(s)\rangle, \emptyset), (\overline{tag}_s, \emptyset), \\ (write_s(x).\, s := x.\, tag_s.\texttt{unlock}\ s, \{s\}) \end{array} \right\}_{s\in U} \end{array} \right) \tag{1}$$

with $i = 1, 2$ and

- $W \subseteq \mathit{dom}(A_1)$ and $U \subseteq \mathit{fs}(A_1) \setminus \mathit{locks}(A_1)$;
- $\widetilde{a}, \widetilde{b}, \widetilde{read}, \widetilde{write}, \widetilde{tag}$ are pairwise-distinct channel names and are different from $\mathit{fn}(A_1, A_2, \widetilde{c}, \widetilde{n}_1, \widetilde{n}_2)$;
- $\widetilde{c} \cap (\widetilde{n}_1 \cup \widetilde{n}_2) = \emptyset$;
- $\widetilde{a} = \{a_w\}_{w\in W}$ and $\widetilde{b} = \{b_j\}_{j\in J}$ and $\widetilde{c} = \{c_j\}_{j\in J}$;
- $\widetilde{read} = \{read_s\}_{s\in U}$ and $\widetilde{write} = \{write_s\}_{s\in U}$ and $\widetilde{tag} = \{tag_s\}_{s\in U}$.

The channel name tag_s is used to mark the moment when the attacker has already changed the value of cell s and before cell s is unlocked. As before, since the object of input $tag_s(x)$ is not important, we omit it and write tag_s for simplicity. Note that $(write_s(x).\, s := x.\, tag_s.\texttt{unlock}\ s, \{s\})$ locks the unlocked public state cells from U. Although the cells in U are locked, the attacker can still read and write these cells via

$\overline{read}_s\langle S_i(s)\rangle$ and $write_s(x)$ without unlocking the cells. As a result, all the operations on these cells become visible when comparing transitions in observational equivalence.

We show that $\mathcal{R}$ satisfies all the conditions of Definition 31, i.e., $\mathcal{R} \subseteq \approx_l$. Note that this is sufficient for proving $\approx \subseteq \approx_l$. Suppose $A_1 \approx A_2$, then we let $W = U = J = \emptyset$ and we have $(A_1, A_2) \in \mathcal{R}$. Therefore $A_1 \approx_l A_2$.

Assume $A_1\ \mathcal{R}\ A_2$ because of $\widehat{A}_1 \approx \widehat{A}_2$ where $A_1, A_2, \widehat{A}_1, \widehat{A}_2$ are defined in above Equation (1). According to Definition 31, first of all, we should extend the extended processes A_1 and A_2. Let

$$\mathtt{esc}(A_1, A_2) = U_1 \cup U_2$$

with $U_1 \subseteq U$ and $U_2 \cap U = \emptyset$. Selecting fresh variables v_s for each $s \in U_1 \cup U_2$, then we shall do the following extensions:

$$B_i = \nu\widetilde{n}_i.(\varphi_i, S_i, \mathcal{P}_i) \qquad \varphi_i = \sigma_i \cup \{S_i(s)/v_s\}_{s\in U_1\cup U_2} \qquad \text{for } i = 1, 2$$

We shall prove that $B_1 \approx_s B_2$, and if $B_1 \xrightarrow{\alpha} B_1'$ (or $B_1 \xrightarrow{s:=N}\xrightarrow{\tau(s)} B_1'$) then there exists B_2' such that $B_2 \overset{\widehat{\alpha}}{\Longrightarrow} B_2'$ (resp. $B_2 \xrightarrow{s:=N}\overset{\tau(s)}{\Longrightarrow} B_2'$) and $(B_1', B_2') \in \mathcal{R}$.

1. **First we need to prove the static equivalence $\boldsymbol{B_1 \approx_s B_2}$.** Assume two terms M, N with $var(M, N) \subseteq dom(B_1)$ and $M\varphi_1 =_\Sigma N\varphi_1$. We shall prove that $M\varphi_2 =_\Sigma N\varphi_2$. We can safely assume that $name(M, N) \cap (\widetilde{n}_1, \widetilde{n}_2) = \emptyset$, otherwise we can use α-equivalence to change $\widetilde{n}_1, \widetilde{n}_2$. Since some part of φ_i ($i = 1, 2$) are stored in the output actions $\overline{a}_w\langle w\sigma_i\rangle, \overline{read}_s\langle S_i(s)\rangle$ in $\widetilde{A}_i$, we need to use corresponding input actions to get these terms. We construct the following evaluation context $\mathcal{C}$:

$$\begin{aligned}
&\mathcal{C} = (\text{-}, \text{-}, \{(\overline{e}, \emptyset), (P, V)\}\ \text{-})\\
&P = a_{w_1}(x_{w_1}) \cdots a_{w_k}(x_{w_k}).\ read_{s_1}(z_{s_1}) \cdots read_{s_n}(z_{s_n}).\\
&\qquad\qquad \mathtt{read}\ s_{n+1}\ \mathtt{as}\ z_{s_{n+1}} \cdots \mathtt{read}\ s_{n+l}\ \mathtt{as}\ z_{s_{n+l}}.\mathtt{if}\ M\rho = N\rho\ \mathtt{then}\ e
\end{aligned}$$

where $\{w_1, \cdots, w_k\} = W$, and $\{s_1, \cdots, s_n\} = U_1$, and $\{s_{n+1}, \cdots, s_{n+l}\} = U_2$, and $V := unlocks(A_1) \setminus U$ and $\rho = \{x_w/w\}_{w\in W} \cup \{z_s/v_s\}_{s\in U_1\cup U_2}$ and e is a fresh channel name.

Apply $\mathcal{C}$ to $\widehat{A}_1$ and then we can do the following transitions:

$$\mathcal{C}[\widehat{A}_1] = \nu\widetilde{c}, \widetilde{n}_1. \left(\begin{array}{ll} & \mathcal{P}_1 \cup \boldsymbol{\{(\overline{a}_w\langle w\sigma_1\rangle, \emptyset)\}_{w\in W}} \cup \left\{(\overline{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J} \\ \sigma_{1\setminus W}, S_1, & \cup \left\{ \begin{array}{l} (\boldsymbol{\overline{read}_s\langle S_1(s)\rangle}, \emptyset), (\overline{tag}_s, \emptyset), \\ (write_s(x).\ s := x.\ tag_s.\mathtt{unlock}\ s, \{s\}) \end{array} \right\}_{s\in U} \\ & \cup \left\{(\overline{e}, \emptyset), (\boldsymbol{P\sigma_{1\setminus W}}, V)\right\} \end{array} \right)$$

$$\Longrightarrow$$

$$\nu\widetilde{c}, \widetilde{n}_1. \left(\begin{array}{ll} & \mathcal{P}_1 \cup \left\{(\overline{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J} \cup \{(\boldsymbol{\overline{e}}, \emptyset)\} \\ \sigma_{1\setminus W}, S_1, & \cup \left\{(\overline{tag}_s, \emptyset), (write_s(x).\ s := x.\ tag_s.\mathtt{unlock}\ s, \{s\})\right\}_{s\in U} \\ & \cup \left\{(\mathtt{if}\ \boldsymbol{((M\rho)\sigma_{1\setminus W})\rho' = ((N\rho)\sigma_{1\setminus W})\rho'}\ \mathtt{then}\ \boldsymbol{e}, V)\right\} \end{array} \right)$$

$$\Longrightarrow D_1 :=$$

$$\nu\widetilde{c},\widetilde{n}_1.\left(\sigma_{1\backslash W}, S_1, \begin{array}{c}\mathcal{P}_1 \cup \left\{(\bar{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J} \cup \{(\mathbf{0}, V)\} \\ \cup \left\{(\overline{tag}_s, \emptyset), (write_s(x).\, s := x.\, tag_s.\mathtt{unlock}\ s, \{s\})\right\}_{s\in U}\end{array}\right)$$

where $\rho' = \{w\sigma_1/x_w\}_{w\in W} \cup \{S_1(s)/z_s\}_{s\in U_1\cup U_2}$. The last step is deduced from the fact that $((M\rho)\sigma_{1\backslash W})\rho' = M\varphi_1$ and $((N\rho)\sigma_{1\backslash W})\rho' = N\varphi_1$ and $M\varphi_1 =_\Sigma N\varphi_1$. It is easy to see that $\mathcal{C}[\widehat{A}_1] \Downarrow_{\overline{a}_w,\bar{b}_j,\overline{read}_s,write_s,\overline{e}}$ with $w \in W, j \in J, s \in U$, while $D_1 \Downarrow_{\bar{b}_j,write_s}$ with $j \in J, s \in U$ but $D_1 \not\Downarrow_{\overline{a}_w,\overline{read}_s,\overline{e}}$ for any $w \in W, s \in U$. Since $\widehat{A}_1 \approx \widehat{A}_2$ and $\approx$ is closed by application of evaluation context, we have $\mathcal{C}[\widehat{A}_1] \approx \mathcal{C}[\widehat{A}_2]$. Hence there should exist D_2 such that $\mathcal{C}[\widehat{A}_2] \overset{\epsilon}{\Longrightarrow} D_2 \approx D_1$ and we should have $D_2 \Downarrow_{\bar{b}_j,write_s}$ with $j \in J, s \in U$ and $D_2 \not\Downarrow_{\overline{a}_w,\overline{read}_s,\overline{e}}$ for any $w \in W, s \in U$. The only possibility for D_2 is that

$$\mathcal{C}[\widehat{A}_2] \Longrightarrow$$

$$\nu\widetilde{c},\widetilde{n}'_2.\left(\sigma_{2\backslash W}, S'_2, \begin{array}{c}\mathcal{P}'_2 \cup \left\{(\bar{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J} \cup \left\{(\overline{tag}_s, \emptyset), (\overline{\boldsymbol{e}}, \emptyset)\right\} \\ \cup \left\{(write_s(x).\, s := x.\, tag_s.\mathtt{unlock}\ s, \{s\})\right\}_{s\in U} \\ \cup \left\{(\mathtt{if}\ \boldsymbol{((M\rho)\sigma_{2\backslash W})\rho'' = ((N\rho)\sigma_{2\backslash W})\rho''}\ \mathtt{then}\ \boldsymbol{e}, V)\right\}\end{array}\right)$$

$$\Longrightarrow D_2 :=$$

$$\nu\widetilde{c},\widetilde{n}''_2.\left(\sigma_{2\backslash W}, S''_2, \begin{array}{c}\mathcal{P}''_2 \cup \left\{(\bar{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J} \cup \left\{(\overline{tag}_s, \emptyset), (\mathbf{0}, V)\right\} \\ \cup \left\{(write_s(x).\, s := x.\, tag_s.\mathtt{unlock}\ s, \{s\})\right\}_{s\in U}\end{array}\right)$$

where $\rho'' = \{w\sigma_2/x_w\}_{w\in W} \cup \{S_2(s)/z_s\}_{s\in U_1\cup U_2}$. We must have $((M\rho)\sigma_{2\backslash W})\rho'' =_\Sigma ((N\rho)\sigma_{2\backslash W})\rho''$, otherwise we wouldn't be able to consume $\overline{e}$. Similarly we know that $((M\rho)\sigma_{2\backslash W})\rho'' = M\varphi_2$ and $((N\rho)\sigma_{2\backslash W})\rho'' = N\varphi_2$. Hence $M\varphi_2 =_\Sigma N\varphi_2$.

2. **Now we proceed to prove the behavioural equivalence of B_1 and B_2.** Without loss of generality, we assume $B_1 \xrightarrow{\alpha} B'_1$ (resp. $B_1 \xrightarrow{s:=N}\xrightarrow{\tau(s)} B'_1$) and prove that there exists B'_2 such that $B_2 \overset{\widehat{\alpha}}{\Longrightarrow} B'_2$ (resp. $B_2 \xrightarrow{s:=N}\overset{\tau(s)}{\Longrightarrow} B'_2$) and $(B'_1, B'_2) \in \mathcal{R}$.

   Before we start to analyse the transitions, we need to preprocess $\widehat{A}_1$ and $\widehat{A}_2$. Recall that B_1 and B_2 are the extensions of A_1 and A_2. When B_1 performs some operations on a public cell s, then B_2 is required to mimic these operations by transitions on the same cell s in the definition of labelled bisimilarity. In other words, B_2 is not allowed to perform any operations on the other public cells which are different from s. Therefore, when using observational equivalence between $\widehat{A}_1$ and $\widehat{A}_2$, we need to make sure the transitions on the cell s from $\widehat{A}_1$ are matched with transitions on the same cell s. To do this, we need to lock and mark these unlocked cells to prevent operations on them. We construct the following context $\mathcal{C}_{ext}$ and apply it to $\widehat{A}_1$ and $\widehat{A}_2$:

$$\mathcal{C}_{ext} :=$$
$$\left(\text{-}, \text{-}, \begin{array}{l} \left\{ (\overline{e}_s, \emptyset), \left(read_s(z). \left(\overline{read}_s\langle z\rangle \mid e_s.\overline{a}_{v_s}\langle z\rangle \right), \emptyset \right) \right\}_{s\in U_1} \\ \cup \left\{ \begin{array}{l} (\overline{e}_s, \emptyset), \left(d_s(z). \left(\overline{read}_s\langle z\rangle \mid e_s.\overline{a}_{v_s}\langle z\rangle \right), \emptyset \right), \left(\overline{tag}_s, \emptyset \right), \\ \left(\texttt{read}\ s\ \texttt{as}\ y.\, \overline{d}_s\langle y\rangle.\, write_s(x).\, s := x.\, tag_s.\texttt{unlock}\ s, \{s\} \right) \end{array} \right\}_{s\in U_2} \\ \cup \left\{ \begin{array}{l} \left(d_s(z).\overline{read}_s\langle z\rangle, \emptyset \right), \left(\overline{tag}_s, \emptyset \right), \\ \left(\texttt{read}\ s\ \texttt{as}\ y.\, \overline{d}_s\langle y\rangle.\, write_s(x).\, s := x.\, tag_s.\texttt{unlock}\ s, \{s\} \right) \end{array} \right\}_{s\in U_3} \end{array} \text{-} \right)$$

where $U_3 := unlocks(A_1) \setminus (U \cup U_2)$, and $\{a_{v_s}\}_{s\in U_1\cup U_2}$ and $\{d_s, tag_s\}_{s\in U_2\cup U_3}$ and $\{e_s\}_{s\in unlocks(A_1)}$ are fresh pairwise-distinct channel names. Since the cells in $s \in U_1$ are already locked and marked in $\widehat{A}_1$ and $\widehat{A}_2$, the context $\mathcal{C}_{ext}$ reads their values and store them in the output $\overline{a}_{v_s}\langle z\rangle$. The cells in $s \in U_2$ are not yet locked and their values are not in the substitutions, so the context $\mathcal{C}_{ext}$ locks these cells and store their values in the output $\overline{a}_{v_s}\langle z\rangle$. The values of cells $s \in U_3$ are already stored in the substitutions, so the context $\mathcal{C}_{ext}$ only locks and marks these cells. The use of $(\overline{e}_s, \emptyset)$ for $s \in U_1 \cup U_2$ is to make sure the parallel composition $\left\{ (\overline{read}_s\langle z\rangle \mid e_s.\overline{a}_{v_s}\langle z\rangle, \emptyset) \right\}$ will be split into $\left\{ (\overline{read}_s\langle z\rangle, \emptyset), (\overline{a}_{v_s}\langle z\rangle, \emptyset) \right\}$ as a result of the communication between $\overline{e}_s$ and e_s.

Since $\approx$ is closed under the application of evaluation contexts, we have $\mathcal{C}_{ext}[\widehat{A}_1] \approx \mathcal{C}_{ext}[\widehat{A}_2]$. Then we can have the following transitions:

$$\mathcal{C}_{ext}[\widehat{A}_1] =$$
$$\nu\widetilde{c}, \widetilde{n}_1.$$
$$\left(\sigma_{1\setminus W}, S_1, \begin{array}{l} \mathcal{P}_1 \cup \{(\overline{a}_w\langle w\sigma_1\rangle, \emptyset)\}_{w\in W} \cup \left\{ (\overline{b}_j\langle c_j\rangle, \emptyset) \right\}_{j\in J} \\ \cup \left\{ \begin{array}{l} (\overline{read}_s\langle S_1(s)\rangle, \emptyset), (\overline{tag}_s, \emptyset), \\ (write_s(x).\, s := x.\, tag_s.\texttt{unlock}\ s, \{s\}) \end{array} \right\}_{\boldsymbol{s\in U}} \\ \cup \left\{ (\overline{e}_s, \emptyset), \left(read_s(z). \left(\overline{read}_s\langle z\rangle \mid e_s.\overline{a}_{v_s}\langle z\rangle \right), \emptyset \right) \right\}_{s\in U_1} \\ \cup \left\{ \begin{array}{l} (\overline{e}_s, \emptyset), \left(d_s(z). \left(\overline{read}_s\langle z\rangle \mid e_s.\overline{a}_{v_s}\langle z\rangle \right), \emptyset \right), \left(\overline{tag}_s, \emptyset \right), \\ \left(\texttt{read}\ s\ \texttt{as}\ y.\overline{d}_s\langle y\rangle.\, write_s(x).s := x.\, tag_s.\texttt{unlock}\ s, \{s\} \right) \end{array} \right\}_{s\in U_2} \\ \cup \left\{ \begin{array}{l} \left(d_s(z).\overline{read}_s\langle z\rangle, \emptyset \right), \left(\overline{tag}_s, \emptyset \right), \\ \left(\texttt{read}\ s\ \texttt{as}\ y.\, \overline{d}_s\langle y\rangle.\, write_s(x).\, s := x.\, tag_s.\texttt{unlock}\ s, \{s\} \right) \end{array} \right\}_{s\in U_3} \end{array} \right)$$

$$
\begin{aligned}
&= \\
&\nu\widetilde{c},\widetilde{n}_1. \\
&\left(\begin{array}{ll} & \mathcal{P}_1 \cup \{(\overline{a}_w\langle w\sigma_1\rangle, \emptyset)\}_{w\in W} \cup \left\{(\bar{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J} \\
& \cup \left\{\begin{array}{l}(\overline{read}_s\langle S_1(s)\rangle, \emptyset), (\overline{tag}_s, \emptyset), \\ (write_s(x).\, s := x.\, tag_s.\mathtt{unlock}\ s, \{s\})\end{array}\right\}_{\boldsymbol{s\in U\setminus U_1}} \\
\sigma_{1\setminus W}, S_1, & \cup \left\{\begin{array}{l}\left(\overline{read}_s\langle S_1(s)\rangle, \emptyset\right), \left(\overline{tag}_s, \emptyset\right), \\ \left(write_s(x).\, s := x.\, tag_s.\mathtt{unlock}\ s, \{s\}\right), \\ \left(\overline{e}_s, \emptyset\right), \left(read_s(z).\left(\overline{read}_s\langle z\rangle \mid e_s.\overline{a}_{v_s}\langle z\rangle\right), \emptyset\right)\end{array}\right\}_{\boldsymbol{s\in U_1}} \\
& \cup \left\{\begin{array}{l}\left(\overline{e}_s, \emptyset\right), \left(d_s(z).\left(\overline{read}_s\langle z\rangle \mid e_s.\overline{a}_{v_s}\langle z\rangle\right), \emptyset\right), \left(\overline{tag}_s, \emptyset\right), \\ \left(\mathtt{read}\ \boldsymbol{s}\ \mathtt{as}\ \boldsymbol{y}.\, \overline{d}_s\langle y\rangle.\, write_s(x).\, s := x.\, tag_s.\mathtt{unlock}\ s, \{s\}\right)\end{array}\right\}_{s\in U_2} \\
& \cup \left\{\begin{array}{l}\left(d_s(z).\overline{read}_s\langle z\rangle, \emptyset\right), \left(\overline{tag}_s, \emptyset\right), \\ \left(\mathtt{read}\ \boldsymbol{s}\ \mathtt{as}\ \boldsymbol{y}.\, \overline{d}_s\langle y\rangle.\, write_s(x).\, s := x.\, tag_s.\mathtt{unlock}\ s, \{s\}\right)\end{array}\right\}_{s\in U_3}
\end{array}\right)
\end{aligned}
$$

$$
\Longrightarrow D_1 :=
$$

$$
\nu\widetilde{c},\widetilde{n}_1.\left(\begin{array}{ll} & \mathcal{P}_1 \cup \{(\overline{a}_w\langle w\sigma_1\rangle, \emptyset)\}_{w\in W} \cup \left\{(\bar{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J} \\
\sigma_{1\setminus W}, S_1, & \cup \{(\overline{a}_{v_s}\langle S_1(s)\rangle, \emptyset)\}_{s\in U_1\cup U_2} \\
& \cup \left\{\begin{array}{l}\left(\boldsymbol{\overline{read}_s\langle S_1(s)\rangle}, \emptyset\right), \left(\overline{tag}_s, \emptyset\right), \\ (write_s(x).s := x.\, tag_s.\mathtt{unlock}\ s, \{s\})\end{array}\right\}_{s\in unlocks(A_1)}
\end{array}\right)
$$

We can see that $D_1 \Downarrow_{\overline{a}_w, \overline{a}_{w_s}, \bar{b}_j, \overline{read}_t, write_t, \overline{tag}_t}$ for $w \in W, j \in J, s \in U_1 \cup U_2, t \in unlocks(A_1)$, while $D_1 \not\Downarrow_{\overline{e}_s, d_t}$ for $s \in U_1 \cup U_2, t \in U_3$. Since $\mathcal{C}_{ext}[\widehat{A}_1] \approx \mathcal{C}_{ext}[\widehat{A}_2]$, there exists D_2 such that $\mathcal{C}_{ext}[\widehat{A}_2] \stackrel{\epsilon}{\Longrightarrow} D_2 \approx D_1$. The only possibility for D_2 is that:

$$
\mathcal{C}_{ext}[\widehat{A}_2] \Longrightarrow D_2 :=
$$

$$
\nu\widetilde{c},\widetilde{n}'_2.\left(\begin{array}{ll} & \mathcal{P}'_2 \cup \{(\overline{a}_w\langle w\sigma_2\rangle, \emptyset)\}_{w\in W} \cup \left\{(\bar{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J} \\
\sigma_{2\setminus W}, S'_2, & \cup \{(\overline{a}_{v_s}\langle S_2(s)\rangle, \emptyset)\}_{s\in U_1\cup U_2} \\
& \cup \left\{\begin{array}{l}\left(\overline{read}_s\langle S_2(s)\rangle, \emptyset\right), \left(\overline{tag}_s, \emptyset\right), \\ (write_s(x).s := x.\, tag_s.\mathtt{unlock}\ s, \{s\})\end{array}\right\}_{s\in unlocks(A_1)}
\end{array}\right)
$$

for some $S'_2, \mathcal{P}'_2$. Since $read_s, write_s, e_s, tag_s$ in $\mathcal{C}_{ext}$ are fresh names, they will not interact with $\widehat{A}_2$. Moreover all the unlocked public state cells in $\widehat{A}_2$ are locked by $\mathcal{C}_{ext}$, hence the values of these cells won't be changed during the transitions. Thus, we can deduce that

$$
B_2 \Longrightarrow E := \nu\widetilde{n}'_2.(\sigma_{2\setminus W}, S'_2, \mathcal{P}'_2)
$$

From $B_1 = \nu\widetilde{n}_1.(\varphi_1, S_1, \mathcal{P}_1)$ and $E = \nu\widetilde{n}'_2.(\varphi_2, S'_2, \mathcal{P}'_2)$ and $D_1 \approx D_2$, we can verify that $(B_1, E) \in \mathcal{R}$.

Now we are ready to analyse each possible transition from B_1.

(a) Assume

$$B_1 = \nu\widetilde{n}_1.\,(\varphi_1, S_1, \mathcal{P}_1) \xrightarrow{\tau} B_1' := \nu\widetilde{n}_1'.\,(\varphi_1, S_1', \mathcal{P}_1')$$

This internal transition can only involve $\widetilde{n}_1, S_1, \mathcal{P}_1$, thus we can get the following transition from D_1:

$$D_1 = \\ \nu\widetilde{c}, \widetilde{n}_1. \left(\sigma_{1\setminus W}, \boldsymbol{S_1}, \begin{array}{c} \boldsymbol{\mathcal{P}_1} \cup \{(\overline{a}_w\langle w\sigma_1\rangle, \emptyset)\}_{w\in W} \cup \left\{(\bar{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J} \\ \cup \{(\overline{a}_{v_s}\langle S_1(s)\rangle, \emptyset)\}_{s\in U_1\cup U_2} \\ \cup \left\{ \begin{array}{l} \left(\overline{read}_s\langle S_1(s)\rangle, \emptyset\right), \left(\overline{tag}_s, \emptyset\right), \\ (write_s(x).s := x.\, tag_s.\texttt{unlock}\ s, \{s\}) \end{array} \right\}_{s\in unlocks(A_1)} \end{array} \right)$$

$$\xrightarrow{\tau} D_1' := \\ \nu\widetilde{c}, \widetilde{n}_1'. \left(\sigma_{1\setminus W}, \boldsymbol{S_1'}, \begin{array}{c} \boldsymbol{\mathcal{P}_1'} \cup \{(\overline{a}_w\langle w\sigma_1\rangle, \emptyset)\}_{w\in W} \cup \left\{(\bar{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J} \\ \cup \{(\overline{a}_{v_s}\langle S_1(s)\rangle, \emptyset)\}_{s\in U_1\cup U_2} \\ \cup \left\{ \begin{array}{l} \left(\overline{read}_s\langle S_1(s)\rangle, \emptyset\right), \left(\overline{tag}_s, \emptyset\right), \\ (write_s(x).s := x.\, tag_s.\texttt{unlock}\ s, \{s\}) \end{array} \right\}_{s\in unlocks(A_1)} \end{array} \right)$$

We can see that $D_1' \Downarrow_{\overline{a}_w, \overline{a}_{v_t}, \bar{b}_j, \overline{read}_s, \overline{tag}_s, write_s}$ for $w \in W, t \in U_1 \cup U_2, j \in J, s \in unlocks(A_1)$. From $D_1 \approx D_2$, there should exist D_2' such that $D_2 \Longrightarrow D_2' \approx D_1'$. The only possibility for D_2' is that

$$D_2 = \\ \nu\widetilde{c}, \widetilde{n}_2'. \left(\sigma_{2\setminus W}, \boldsymbol{S_2'}, \begin{array}{c} \boldsymbol{\mathcal{P}_2'} \cup \{(\overline{a}_w\langle w\sigma_2\rangle, \emptyset)\}_{w\in W} \cup \left\{(\bar{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J} \\ \cup \{(\overline{a}_{v_s}\langle S_2(s)\rangle, \emptyset)\}_{s\in U_1\cup U_2} \\ \cup \left\{ \begin{array}{l} \left(\overline{read}_s\langle S_2(s)\rangle, \emptyset\right), \left(\overline{tag}_s, \emptyset\right), \\ (write_s(x).s := x.\, tag_s.\texttt{unlock}\ s, \{s\}) \end{array} \right\}_{s\in unlocks(A_1)} \end{array} \right)$$

$$\Longrightarrow D_2' := \\ \nu\widetilde{c}, \widetilde{n}_2''. \left(\sigma_{2\setminus W}, \boldsymbol{S_2''}, \begin{array}{c} \boldsymbol{\mathcal{P}_2''} \cup \{(\overline{a}_w\langle w\sigma_2\rangle, \emptyset)\}_{w\in W} \cup \left\{(\bar{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J} \\ \cup \{(\overline{a}_{v_s}\langle S_2(s)\rangle, \emptyset)\}_{s\in U_1\cup U_2} \\ \cup \left\{ \begin{array}{l} \left(\overline{read}_s\langle S_2(s)\rangle, \emptyset\right), \left(\overline{tag}_s, \emptyset\right), \\ (write_s(x).s := x.\, tag_s.\texttt{unlock}\ s, \{s\}) \end{array} \right\}_{s\in unlocks(A_1)} \end{array} \right)$$

The transitions $D_2 \Longrightarrow D_2'$ can only involve $\widetilde{n}_2', S_2', \mathcal{P}_2'$. Thus we can see that

$$E = \nu\widetilde{n}_2'.(\varphi_2, S_2', \mathcal{P}_2') \Longrightarrow B_2' := \nu\widetilde{n}_2''.(\varphi_2, S_2'', \mathcal{P}_2'')$$

Since $B_2 \Longrightarrow E$ and $E \Longrightarrow B_2'$, we have $B_2 \Longrightarrow B_2'$. Comparing the construction of D_1' (resp. D_2') with B_1' (resp. B_2'), we can see that $(B_1', B_2') \in \mathcal{R}$.

(b) Assume

$$B_1 = \nu\widetilde{n}_1.\,(\varphi_1, T_1 \cup \{t \mapsto M_1\}, \mathcal{P}_1) \xrightarrow{t:=N} \nu\widetilde{n}_1.\,(\varphi_1, T_1 \cup \{t \mapsto N\varphi_1\}, \mathcal{P}_1)$$
$$\xrightarrow{\tau(t)} B_1' := \nu\widetilde{n}_1.\,(\varphi_1, T_1 \cup \{t \mapsto M_1'\}, \mathcal{P}_1')$$

where $t \notin \widetilde{n}_1 \cup locks(\mathcal{P}_1)$ and $S_1 = T_1 \cup \{t \mapsto M_1\}$ and $var(N) \subseteq dom(B_1)$.

We need to show that there exists B_2' such that $B_2 \xrightarrow{t:=N}\overset{\tau(t)}{\Longrightarrow} B_2'$ and $(B_1', B_2') \in \mathcal{R}$. The idea is to find a B_2' from E such that $E \xrightarrow{t:=N}\overset{\tau(t)}{\Longrightarrow} B_2'$ and then use Corollary 40 and $B_2 \Longrightarrow E$ to get $B_2 \xrightarrow{t:=N}\Longrightarrow\overset{\tau(t)}{\Longrightarrow} B_2'$.

We construct an evaluation context $\mathcal{C}_t$:

$$\mathcal{C}_t = \left(\_, \_, \left\{ \left(\prod_{i=1}^{n} \overline{e}_i, \emptyset \right), \left(\begin{array}{l} a_{w_1}(x_1). \cdots .a_{w_n}(x_n). \\ \qquad read_t(x).\overline{write}_t\langle N\rho\rangle.\left(\prod_{i=1}^{n} e_i.\overline{a}_{w_i}\langle x_i\rangle\right), \emptyset \end{array} \right) \right\} \quad \_ \right)$$

where $e_1 \cdots e_n$ are pairwise distinct fresh channel names, $\{w_1, \cdots, w_n\} = W \cup \{v_s\}_{s \in U_1 \cup U_2}$ and $\rho = \{x_1/w_1, \cdots, x_n/w_n\}$. Applying $\mathcal{C}_t$ to D_1, we can get the following transitions:

$$\mathcal{C}_t[D_1] =$$
$$\nu\widetilde{c}, \widetilde{n}_1.$$
$$\left(\sigma_{1\setminus W}, T_1 \cup \{t \mapsto \boldsymbol{M_1}\}, \begin{array}{l} \mathcal{P}_1 \cup \{(\overline{a}_w\langle w\sigma_1\rangle, \emptyset)\}_{w \in W} \cup \left\{(\overline{b}_j\langle c_j\rangle, \emptyset)\right\}_{j \in J} \\ \quad \cup \{(\overline{a}_{v_s}\langle S_1(s)\rangle, \emptyset)\}_{s \in U_1 \cup U_2} \\ \quad \cup \left\{ \begin{array}{l} \left(\overline{read}_s\langle S_1(s)\rangle, \emptyset\right), (\overline{tag}_s, \emptyset), \\ (write_s(x).\boldsymbol{s := x}.\ tag_s.\texttt{unlock}\ s, \{s\}) \end{array} \right\}_{s \in unlocks(A_1)} \\ \quad \cup \left\{ \left(\prod_{i=1}^{n} \overline{e}_i, \emptyset \right) \right\} \\ \quad \cup \left\{ \left(\begin{array}{l} a_{w_1}(x_1). \cdots .a_{w_n}(x_n). \\ read_t(x).\overline{write}_t\langle N\rho\rangle.\left(\prod_{i=1}^{n} e_i.\overline{a}_{w_i}\langle x_i\rangle\right), \emptyset \end{array} \right) \right\} \end{array} \right)$$

$$\Longrightarrow D_1' :=$$
$$\nu\widetilde{c}, \widetilde{n}_1.$$
$$\left(\sigma_{1\setminus W}, T_1 \cup \{t \mapsto \boldsymbol{N\varphi_1}\}, \begin{array}{l} \mathcal{P}_1 \cup \{(\overline{a}_w\langle w\sigma_1\rangle, \emptyset)\}_{w \in W} \cup \left\{(\overline{b}_j\langle c_j\rangle, \emptyset)\right\}_{j \in J} \\ \quad \cup \{(\overline{a}_{v_s}\langle S_1(s)\rangle, \emptyset)\}_{s \in U_1 \cup U_2} \\ \quad \cup \left\{ \begin{array}{l} \left(\overline{read}_s\langle S_1(s)\rangle, \emptyset\right), (\overline{tag}_s, \emptyset), \\ (write_s(x).s := x.\ tag_s.\texttt{unlock}\ s, \{s\}) \end{array} \right\}_{s \in unlocks(A_1)}^{s \neq t} \\ \quad \cup \boldsymbol{\{(\texttt{unlock}\ t, \{t\})\}} \end{array} \right)$$

$$\xrightarrow{\tau(t)}$$

$$\nu\widetilde{c},\widetilde{n}_1.\left(\sigma_{1\setminus W}, T_1 \cup \{t \mapsto N\varphi_1\}, \begin{array}{l} \boldsymbol{\mathcal{P}_1} \cup \{(\overline{a}_w\langle w\sigma_1\rangle,\emptyset)\}_{w\in W} \cup \left\{(\overline{b}_j\langle c_j\rangle,\emptyset)\right\}_{j\in J} \\ \cup \{(\overline{a}_{v_s}\langle S_1(s)\rangle,\emptyset)\}_{s\in U_1\cup U_2} \\ \cup \left\{ \begin{array}{l} \left(\overline{read}_s\langle S_1(s)\rangle,\emptyset\right), (\overline{tag}_s,\emptyset), \\ (write_s(x).s := x.\, tag_s.\mathtt{unlock}\ s, \{s\}) \end{array} \right\}^{s\neq t}_{s\in unlocks(A_1)} \end{array}\right)$$

$$\xrightarrow{\tau(t)} D_1'' :=$$

$$\nu\widetilde{c},\widetilde{n}_1.\left(\sigma_{1\setminus W}, T_1 \cup \{t \mapsto M_1'\}, \begin{array}{l} \boldsymbol{\mathcal{P}_1'} \cup \{(\overline{a}_w\langle w\sigma_1\rangle,\emptyset)\}_{w\in W} \cup \left\{(\overline{b}_j\langle c_j\rangle,\emptyset)\right\}_{j\in J} \\ \cup \{(\overline{a}_{v_s}\langle S_1(s)\rangle,\emptyset)\}_{s\in U_1\cup U_2} \\ \cup \left\{ \begin{array}{l} \left(\overline{read}_s\langle S_1(s)\rangle,\emptyset\right), (\overline{tag}_s,\emptyset), \\ (write_s(x).s := x.\, tag_s.\mathtt{unlock}\ s, \{s\}) \end{array} \right\}^{s\neq t}_{s\in unlocks(A_1)} \end{array}\right)$$

In the above transitions, all the public state cells in D_1 are locked. We can see that $D_1' \not\Downarrow_{\overline{e}_1,\cdots,\overline{e}_n,\overline{tag}_t}$. We apply $\mathcal{C}_t$ to D_2. From $\mathcal{C}_t[D_1] \approx \mathcal{C}_t[D_2]$, there should exist D_2' such that $\mathcal{C}_t[D_2] \overset{\epsilon}{\Longrightarrow} D_2' \overset{\epsilon}{\Longrightarrow} D_2''$ and $D_2' \approx D_1'$ and $D_2'' \approx D_1''$. Let $S_2' = T_2 \cup \{t \mapsto M_2\}$. The only possibility for D_2' and D_2'' is that:

$$\mathcal{C}_t[D_2] =$$

$$\nu\widetilde{c},\widetilde{n}_2'.\left(\sigma_{2\setminus W}, \boldsymbol{T_2} \cup \{t \mapsto \boldsymbol{M_2}\}, \begin{array}{l} \boldsymbol{\mathcal{P}_2'} \cup \{(\overline{a}_w\langle w\sigma_2\rangle,\emptyset)\}_{w\in W} \cup \left\{(\overline{b}_j\langle c_j\rangle,\emptyset)\right\}_{j\in J} \\ \cup \{(\overline{a}_{v_s}\langle S_2(s)\rangle,\emptyset)\}_{s\in U_1\cup U_2} \\ \cup \left\{ \begin{array}{l} \left(\overline{read}_s\langle S_2(s)\rangle,\emptyset\right), (\overline{tag}_s,\emptyset), \\ (write_s(x).\boldsymbol{s := x}.\, tag_s.\mathtt{unlock}\ s, \{s\}) \end{array} \right\}_{s\in unlocks(A_1)} \\ \cup \left\{ \left(\prod_{i=1}^{n} \overline{e}_i, \emptyset \right) \right\} \\ \cup \left\{ \left(\begin{array}{l} a_{w_1}(x_1).\cdots.a_{w_n}(x_n). \\ read_t(x).\overline{write}_t\langle N\rho\rangle.\left(\prod_{i=1}^{n} e_i.\overline{a}_{w_i}\langle x_i\rangle\right), \emptyset \end{array} \right) \right\} \end{array}\right)$$

$\Longrightarrow D_2' :=$

$$\nu\widetilde{c},\widetilde{n}_2''.\left(\begin{array}{ll}\sigma_{2\setminus W}, T_2' \cup \{t \mapsto \boldsymbol{N\varphi_2}\}, & \begin{array}{l}\boldsymbol{\mathcal{P}_2''} \cup \{(\overline{a}_w\langle w\sigma_2\rangle, \emptyset)\}_{w\in W} \cup \left\{(\overline{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J} \\ \cup \{(\overline{a}_{v_s}\langle S_2(s)\rangle, \emptyset)\}_{s\in U_1\cup U_2} \\ \cup \left\{\begin{array}{l}\left(\overline{read}_s\langle S_2(s)\rangle, \emptyset\right), \left(\overline{tag}_s, \emptyset\right), \\ \left(write_s(x).s := x.\, tag_s.\mathtt{unlock}\ s, \{s\}\right)\end{array}\right\}_{s\in unlocks(A_1)}^{s\neq t} \\ \cup \boldsymbol{\{(\mathtt{unlock}\ t, \{t\})\}}\end{array}\end{array}\right)$$

$\xrightarrow{\tau(t)}$

$$\nu\widetilde{c},\widetilde{n}_2''.\left(\begin{array}{ll}\sigma_{2\setminus W}, \boldsymbol{T_2' \cup \{t \mapsto N\varphi_2\}}, & \begin{array}{l}\boldsymbol{\mathcal{P}_2''} \cup \{(\overline{a}_w\langle w\sigma_2\rangle, \emptyset)\}_{w\in W} \cup \left\{(\overline{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J} \\ \cup \{(\overline{a}_{v_s}\langle S_2(s)\rangle, \emptyset)\}_{s\in U_1\cup U_2} \\ \cup \left\{\begin{array}{l}\left(\overline{read}_s\langle S_2(s)\rangle, \emptyset\right), \left(\overline{tag}_s, \emptyset\right), \\ \left(write_s(x).s := x.\, tag_s.\mathtt{unlock}\ s, \{s\}\right)\end{array}\right\}_{s\in unlocks(A_1)}^{s\neq t}\end{array}\end{array}\right)$$

$\stackrel{\tau(t)}{\Longrightarrow} D_2'' :=$

$$\nu\widetilde{c},\widetilde{n}_2'''.\left(\begin{array}{ll}\sigma_{2\setminus W}, \boldsymbol{S_2''}, & \begin{array}{l}\boldsymbol{\mathcal{P}_2'''} \cup \{(\overline{a}_w\langle w\sigma_2\rangle, \emptyset)\}_{w\in W} \cup \left\{(\overline{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J} \\ \cup \{(\overline{a}_{v_s}\langle S_2(s)\rangle, \emptyset)\}_{s\in U_1\cup U_2} \\ \cup \left\{\begin{array}{l}\left(\overline{read}_s\langle S_2(s)\rangle, \emptyset\right), \left(\overline{tag}_s, \emptyset\right), \\ \left(write_s(x).s := x.\, tag_s.\mathtt{unlock}\ s, \{s\}\right)\end{array}\right\}_{s\in unlocks(A_1)}^{s\neq t}\end{array}\end{array}\right)$$

We can see that

$$E \Longrightarrow\xrightarrow{t:=N}\Longrightarrow \nu\widetilde{n}_2''.\left(\varphi_2, T_2' \cup \{t \mapsto N\varphi_2\}, \mathcal{P}_2''\right) \stackrel{\tau(t)}{\Longrightarrow} B_2' := \nu\widetilde{n}_2'''.\left(\varphi_2, S_2'', \mathcal{P}_2'''\right)$$

From $B_2 \Longrightarrow E$, we have $B_2 \Longrightarrow\xrightarrow{t:=N}\Longrightarrow\stackrel{\tau(t)}{\Longrightarrow} B_2'$. Using Corollary 40, we know that $B_2 \xrightarrow{t:=N}\stackrel{\tau(t)}{\Longrightarrow} B_2'$. Comparing the constructions of B_1' (resp. B_2') with D_1'' (resp. D_2''), we know that $(B_1', B_2') \in \mathcal{R}$.

(c) Assume $B_1 = \nu\widetilde{n}_1, r.(\varphi_1, T_1 \cup \{r \mapsto M\}, \mathcal{Q}_1 \cup \{(\mathtt{open}\ r.P, L)\}) \xrightarrow{\tau(r)} B_1' := \nu\widetilde{n}_1.(\sigma, S \cup \{r \mapsto M\}, \mathcal{P} \cup \{(P, L)\})$ if $r \notin \widetilde{n}_1$.

We can get the following transition from D_1:

$$D_1 =$$
$$\nu\widetilde{c}, \widetilde{n}_1, \boldsymbol{r}. \left(\sigma_{1\backslash W}, T_1 \cup \{r \mapsto M\}, \quad \begin{array}{l} \mathcal{Q}_1 \cup \{(\texttt{open}\ \boldsymbol{r}.\boldsymbol{P}, \boldsymbol{L})\} \\ \quad \cup \{(\overline{a}_w\langle w\sigma_1\rangle, \emptyset)\}_{w\in W} \cup \left\{(\bar{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J} \\ \quad \cup \{(\overline{a}_{v_s}\langle S_1(s)\rangle, \emptyset)\}_{s\in U_1\cup U_2} \\ \quad \cup \left\{ \begin{array}{l} \left(\overline{read}_s\langle S_1(s)\rangle, \emptyset\right), \left(\overline{tag}_s, \emptyset\right), \\ (write_s(x).s := x.\ tag_s.\texttt{unlock}\ s, \{s\}) \end{array} \right\}_{s\in unlocks(A_1)} \end{array} \right)$$

$$\xrightarrow{\tau(r)} D_1' :=$$
$$\nu\widetilde{c}, \widetilde{n}_1. \left(\sigma_{1\backslash W}, T_1 \cup \{r \mapsto M\}, \quad \begin{array}{l} \mathcal{Q}_1 \cup \{(P, L)\} \\ \quad \cup \{(\overline{a}_w\langle w\sigma_1\rangle, \emptyset)\}_{w\in W} \cup \left\{(\bar{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J} \\ \quad \cup \{(\overline{a}_{v_s}\langle S_1(s)\rangle, \emptyset)\}_{s\in U_1\cup U_2} \\ \quad \cup \left\{ \begin{array}{l} \left(\overline{read}_s\langle S_1(s)\rangle, \emptyset\right), \left(\overline{tag}_s, \emptyset\right), \\ (write_s(x).s := x.\ tag_s.\texttt{unlock}\ s, \{s\}) \end{array} \right\}_{s\in unlocks(A_1)} \end{array} \right)$$

We can see that $D_1' \Downarrow_{\overline{a}_w, \overline{a}_{v_t}, \bar{b}_j, \overline{read}_s, \overline{tag}_s, write_s}$ for $w \in W, t \in U_1 \cup U_2, j \in J, s \in unlocks(A_1)$. We can also see that $fs(D_1') = fs(D_1) \cup \{r\}$. From $D_1 \approx D_2$, there should exist D_2' such that $D_2 \Longrightarrow D_2' \approx D_1'$ which requires $fs(D_2') = fs(D_2) \cup \{r\}$. The only possibility for D_2' is that

$$D_2 =$$
$$\nu\widetilde{c}, \boldsymbol{\widetilde{n}_2'}. \left(\sigma_{2\backslash W}, \boldsymbol{S_2'}, \quad \begin{array}{l} \boldsymbol{\mathcal{P}_2'} \cup \{(\overline{a}_w\langle w\sigma_2\rangle, \emptyset)\}_{w\in W} \cup \left\{(\bar{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J} \\ \quad \cup \{(\overline{a}_{v_s}\langle S_2(s)\rangle, \emptyset)\}_{s\in U_1\cup U_2} \\ \quad \cup \left\{ \begin{array}{l} \left(\overline{read}_s\langle S_2(s)\rangle, \emptyset\right), \left(\overline{tag}_s, \emptyset\right), \\ (write_s(x).s := x.\ tag_s.\texttt{unlock}\ s, \{s\}) \end{array} \right\}_{s\in unlocks(A_1)} \end{array} \right)$$

$$\stackrel{\tau(r)}{\Longrightarrow} D_2' :=$$
$$\nu\widetilde{c}, \boldsymbol{\widetilde{n}_2''}. \left(\sigma_{2\backslash W}, \boldsymbol{S_2''}, \quad \begin{array}{l} \boldsymbol{\mathcal{P}_2''} \cup \{(\overline{a}_w\langle w\sigma_2\rangle, \emptyset)\}_{w\in W} \cup \left\{(\bar{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J} \\ \quad \cup \{(\overline{a}_{v_s}\langle S_2(s)\rangle, \emptyset)\}_{s\in U_1\cup U_2} \\ \quad \cup \left\{ \begin{array}{l} \left(\overline{read}_s\langle S_2(s)\rangle, \emptyset\right), \left(\overline{tag}_s, \emptyset\right), \\ (write_s(x).s := x.\ tag_s.\texttt{unlock}\ s, \{s\}) \end{array} \right\}_{s\in unlocks(A_1)} \end{array} \right)$$

The transitions $D_2 \Longrightarrow D_2'$ can only involve $\widetilde{n}_2', S_2', \mathcal{P}_2'$. Thus we can see that

$$E = \nu\widetilde{n}_2'.(\varphi_2, S_2', \mathcal{P}_2') \stackrel{\tau(r)}{\Longrightarrow} B_2' := \nu\widetilde{n}_2''.(\varphi_2, S_2'', \mathcal{P}_2'')$$

Since $\texttt{lock}\ (D'_1) = \texttt{lock}\ (D'_2)$ and $fs(D'_2) = fs(D_2) \cup \{r\}$ and all the unlocked public state cells in A_1 are locked in both D_2 and D'_2, we can see that $\texttt{lock}\ (B'_1) = \texttt{lock}\ (B'_2)$ and $fs(B'_2) = fs(E) \cup \{r\} = fs(B'_1)$. Since $B_2 \Longrightarrow E$ and $E \Longrightarrow B'_2$, we have $B_2 \Longrightarrow B'_2$. Comparing the construction of D'_1 (resp. D'_2) with B'_1 (resp. B'_2), we can see that $(B'_1, B'_2) \in \mathcal{R}$.

(d) Assume $B_1 = \nu\widetilde{n}_1.(\varphi_1, T_1 \cup \{r \mapsto M\}, \mathcal{Q}_1 \cup \{(\texttt{unlock}\ r.P, L \cup \{r\})\}) \xrightarrow{\tau(r)} B'_1 := \nu\widetilde{n}_1.(\sigma, S \cup \{r \mapsto M\}, \mathcal{P} \cup \{(P, L)\})$ if $r \notin \widetilde{n}_1 \cup \texttt{lock}\ (\mathcal{Q}_1) \cup L$. The analysis is similar as above case.

(e) Assume $B_1 = \nu\widetilde{n}_1.(\varphi_1, S_1, \mathcal{Q}_1 \cup \{(a(x).P_1, L_1)\}) \xrightarrow{a(M)} B'_1 := \nu\widetilde{n}_1.(\varphi_1, S_1, \mathcal{Q}_1 \cup \{(P_1\{M\varphi_1/x\}, L_1)\})$ with $name(a, M) \cap \widetilde{n}_1 = \emptyset$ and $var(M) \subseteq dom(\varphi_1)$ and $\mathcal{P}_1 = \mathcal{Q}_1 \cup \{(a(x).P_1, L_1)\}$.

i. when $a \notin \widetilde{c}$, we construct an evaluation context $\mathcal{C}$:

$$\mathcal{C} = \left(\_, \_, \left\{ \left(\prod_{i=1}^{n} \overline{e}_i, \emptyset \right), \left(\begin{array}{l} a_{w_1}(x_1). \cdots .a_{w_n}(x_n). \\ \quad \overline{a}\langle M\rho\rangle. \left(\prod_{i=1}^{n} e_i.\overline{a}_{w_i}\langle x_i\rangle \right) \end{array}, \emptyset \right) \right\} \quad \_ \right)$$

where $e_1 \cdots e_n$ are pairwise distinct fresh channel names, $\{w_1, \cdots, w_n\} = W \cup \{v_s\}_{s \in U_1 \cup U_2}$ and $\rho = \{x_1/w_1, \cdots, x_n/w_n\}$. Applying $\mathcal{C}$ to D_1, we can get the following transitions:

$$\mathcal{C}[D_1] =$$

$$\nu\widetilde{c}, \widetilde{n}_1. \left(\sigma_{1\backslash W}, S_1, \begin{array}{l} \mathcal{Q}_1 \cup \{(a(x).P_1, L_1)\} \cup \left\{ (\overline{b}_j\langle c_j\rangle, \emptyset) \right\}_{j \in J} \\ \quad \cup \{(\overline{a}_w\langle w\sigma_1\rangle, \emptyset)\}_{w \in W} \cup \{(\overline{a}_{v_s}\langle S_1(s)\rangle, \emptyset)\}_{s \in U_1 \cup U_2} \\ \quad \cup \left\{ \begin{array}{l} \left(\overline{read}_s\langle S_1(s)\rangle, \emptyset\right), (\overline{tag}_s, \emptyset), \\ (write_s(x).s := x.\ tag_s.\texttt{unlock}\ s, \{s\}) \end{array} \right\}_{s \in unlocks(A_1)} \\ \quad \cup \left\{ \left(\prod_{i=1}^{n} \overline{e}_i, \emptyset \right) \right\} \\ \quad \cup \left\{ \left(a_{w_1}(x_1). \cdots .a_{w_n}(x_n).\overline{a}\langle M\rho\rangle. \left(\prod_{i=1}^{n} e_i.\overline{a}_{w_i}\langle x_i\rangle \right), \emptyset \right) \right\} \end{array} \right)$$

$$\Longrightarrow$$

$$\nu\widetilde{c}, \widetilde{n}_1. \left(\sigma_{1\backslash W}, S_1, \begin{array}{l} \mathcal{Q}_1 \cup \{(a(x).P_1, L_1)\} \cup \left\{ (\overline{b}_j\langle c_j\rangle, \emptyset) \right\}_{j \in J} \\ \quad \cup \left\{ \begin{array}{l} \left(\overline{read}_s\langle S_1(s)\rangle, \emptyset\right), (\overline{tag}_s, \emptyset), \\ (write_s(x).s := x.\ tag_s.\texttt{unlock}\ s, \{s\}) \end{array} \right\}_{s \in unlocks(A_1)} \\ \quad \cup \left\{ \left(\prod_{i=1}^{n} \overline{e}_i, \emptyset \right) \right\} \\ \quad \cup \left\{ \left(\overline{a}\langle M\varphi_1\rangle. \left(\prod_{i=1}^{n} e_i.\overline{a}_{w_i}\langle w_i\varphi_1\rangle \right), \emptyset \right) \right\} \end{array} \right)$$

$\Longrightarrow D_1' :=$

$$\nu\widetilde{c},\widetilde{n}_1.\left(\sigma_{1\backslash W}, S_1, \begin{array}{c} \mathcal{Q}_1 \cup \{(P_1\{M\varphi_1/x\}, L_1)\} \cup \left\{(\overline{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J} \\ \cup \{(\overline{a}_w\langle w\sigma_1\rangle, \emptyset)\}_{w\in W} \cup \{(\overline{a}_{v_s}\langle S_1(s)\rangle, \emptyset)\}_{s\in U_1\cup U_2} \\ \cup \left\{\begin{array}{l} \left(\overline{read}_s\langle S_1(s)\rangle, \emptyset\right), (\overline{tag}_s, \emptyset), \\ (write_s(x).s := x.\, tag_s.\mathtt{unlock}\ s, \{s\}) \end{array}\right\}_{s\in unlocks(A_1)} \end{array}\right)$$

Then we apply $\mathcal{C}$ to D_2, and from $\mathcal{C}[D_1] \approx \mathcal{C}[D_2]$. There should exist D_2' such that $\mathcal{C}[D_2] \stackrel{\epsilon}{\Longrightarrow} D_2'$ and $D_2' \approx D_1'$. Since $D_1' \Downarrow_{\overline{a}_w, \overline{a}_{v_s}, \overline{b}_j, \overline{read}_t, \overline{tag}_t, write_t}$ for $w \in W, s \in U_1 \cup U_2, j \in J, t \in unlocks(A_1)$ and $D_1' \not\Downarrow_{\overline{e}_i}$ for $i = 1, \cdots, n$, the only possibility for D_2' is that

$\mathcal{C}[D_2] =$

$$\nu\widetilde{c},\boldsymbol{\widetilde{n}_2'}.\left(\sigma_{2\backslash W}, \boldsymbol{S_2'}, \begin{array}{l} \boldsymbol{\mathcal{P}_2'} \cup \{(\overline{a}_w\langle w\sigma_2\rangle, \emptyset)\}_{w\in W} \cup \{(\overline{a}_{v_s}\langle S_2(s)\rangle, \emptyset)\}_{s\in U_1\cup U_2} \\ \cup \left\{(\overline{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J} \\ \cup \left\{\begin{array}{l} \left(\overline{read}_s\langle S_2(s)\rangle, \emptyset\right), (\overline{tag}_s, \emptyset), \\ (write_s(x).s := x.\, tag_s.\mathtt{unlock}\ s, \{s\}) \end{array}\right\}_{s\in unlocks(A_1)} \\ \cup \left\{\left(\prod_{i=1}^{n} \overline{e}_i, \emptyset\right)\right\} \\ \cup \left\{\left(a_{w_1}(x_1).\cdots.a_{w_n}(x_n).\overline{a}\langle M\rho\rangle.\left(\prod_{i=1}^{n} e_i.\overline{a}_{w_i}\langle x_i\rangle\right), \emptyset\right)\right\} \end{array}\right)$$

$\Longrightarrow$

$$\nu\widetilde{c},\boldsymbol{\widetilde{n}_2''}.\left(\sigma_{2\backslash W}, \boldsymbol{S_2''}, \begin{array}{l} \boldsymbol{\mathcal{P}_2''} \cup \left\{(\overline{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J} \\ \cup \left\{\begin{array}{l} \left(\overline{read}_s\langle S_2(s)\rangle, \emptyset\right), (\overline{tag}_s, \emptyset), \\ (write_s(x).s := x.\, tag_s.\mathtt{unlock}\ s, \{s\}) \end{array}\right\}_{s\in unlocks(A_1)} \\ \cup \left\{\left(\prod_{i=1}^{n} \overline{e}_i, \emptyset\right), \left(\boldsymbol{\overline{a}\langle M\varphi_2\rangle}.\left(\prod_{i=1}^{n} e_i.\overline{a}_{w_i}\langle w_i\varphi_2\rangle\right), \emptyset\right)\right\} \end{array}\right)$$

$\Longrightarrow D_2' :=$

$$\nu\widetilde{c},\boldsymbol{\widetilde{n}_2'''}.\left(\sigma_{2\backslash W}, \boldsymbol{S_2'''}, \begin{array}{l} \boldsymbol{\mathcal{P}_2'''} \cup \{(\overline{a}_w\langle w\sigma_2\rangle, \emptyset)\}_{w\in W} \cup \left\{(\overline{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J} \\ \cup \{(\overline{a}_{v_s}\langle S_2(s)\rangle, \emptyset)\}_{s\in U_1\cup U_2} \\ \cup \left\{\begin{array}{l} \left(\overline{read}_s\langle S_2(s)\rangle, \emptyset\right), (\overline{tag}_s, \emptyset), \\ (write_s(x).s := x.\, tag_s.\mathtt{unlock}\ s, \{s\}) \end{array}\right\}_{s\in unlocks(A_1)} \end{array}\right)$$

In the transitions $\mathcal{C}[D_2] \Longrightarrow D_2'$, there is no operation on public state cells in $unlocks(A_1)$ because these cells are all locked. So we can deduce that

$$E = \nu\widetilde{n}_2'.(\varphi_2, S_2', \mathcal{P}_2') \Longrightarrow \nu\widetilde{n}_2''.(\varphi_2, S_2'', \mathcal{P}_2'') \stackrel{a(M)}{\Longrightarrow} B_2' := \nu\widetilde{n}_2'''.(\varphi_2, S_2''', \mathcal{P}_2''')$$

From $D_1'' \approx D_2''$, we have that $(B_1', B_2') \in \mathcal{R}$.

ii. when $a = c_k$ for some $k \in J$, we construct an evaluation context $\mathcal{C}$:

$$\mathcal{C} = \left(\text{-}, \text{-}, \left\{ \left(\prod_{i=1}^{n} \overline{e}_i, \emptyset \right), \left(\begin{array}{l} a_{w_1}(x_1). \cdots .a_{w_n}(x_n).b_k(u). \\ \qquad \overline{u}\langle M\rho \rangle. \left(\overline{b}_k\langle u \rangle \mid \prod_{i=1}^{n} e_i.\overline{a}_{w_i}\langle x_i \rangle \right) \end{array}, \emptyset \right) \right\} \text{ -} \right)$$

where $e_1 \cdots e_n$ are pairwise distinct fresh channel names, $\{w_1, \cdots, w_n\} = W \cup \{v_s\}_{s \in U_1 \cup U_2}$ and $\rho = \{x_1/w_1, \cdots, x_n/w_n\}$.

(f) Assume

$$B_1 = \nu\widetilde{n}_1.(\varphi_1, S_1, \mathcal{Q}_1 \cup \{(a(x).P_1, L_1)\}) \xrightarrow{a(d)} B_1' := \nu\widetilde{n}_1.(\varphi_1, S_1, \mathcal{Q}_1 \cup \{(P_1\{d/x\}, L_1)\})$$

with $a, d \notin \widetilde{n}_1 = \emptyset$ and $\mathcal{P}_1 = \mathcal{Q}_1 \cup \{(a(x).P_1, L_1)\}$.

i. when $a, d \notin \widetilde{c}$, we construct an evaluation context $\mathcal{C}$:

$$\mathcal{C} = (\text{-}, \text{-}, \{(\overline{e}, \emptyset), (\overline{a}\langle d \rangle.e, \emptyset)\} \text{ -})$$

where e is a fresh channel name. Applying $\mathcal{C}$ to D_1, we can get the following transitions:

$$\mathcal{C}[D_1] =$$

$$\nu\widetilde{c}, \widetilde{n}_1. \left(\sigma_{1\setminus W}, S_1, \begin{array}{l} \mathcal{Q}_1 \cup \{(a(x).P_1, L_1)\} \\ \quad \cup \{(\overline{a}_w\langle w\sigma_1 \rangle, \emptyset)\}_{w \in W} \cup \left\{(\overline{b}_j\langle c_j \rangle, \emptyset)\right\}_{j \in J} \\ \quad \cup \{(\overline{a}_{v_s}\langle S_1(s) \rangle, \emptyset)\}_{s \in U_1 \cup U_2} \\ \quad \cup \left\{ \begin{array}{l} \left(\overline{read}_s\langle S_1(s) \rangle, \emptyset\right), (\overline{tag}_s, \emptyset), \\ (write_s(x).s := x.\, tag_s.\texttt{unlock}\ s, \{s\}) \end{array} \right\}_{s \in unlocks(A_1)} \\ \quad \cup \{(\overline{e}, \emptyset), (\overline{a}\langle d \rangle.e, \emptyset)\} \end{array} \right)$$

$$\Longrightarrow D_1' :=$$

$$\nu\widetilde{c}, \widetilde{n}_1. \left(\sigma_{1\setminus W}, S_1, \begin{array}{l} \mathcal{Q}_1 \cup \{(P_1\{d/x\}, L_1)\} \\ \quad \cup \{(\overline{a}_w\langle w\sigma_1 \rangle, \emptyset)\}_{w \in W} \cup \left\{(\overline{b}_j\langle c_j \rangle, \emptyset)\right\}_{j \in J} \\ \quad \cup \{(\overline{a}_{v_s}\langle S_1(s) \rangle, \emptyset)\}_{s \in U_1 \cup U_2} \\ \quad \cup \left\{ \begin{array}{l} \left(\overline{read}_s\langle S_1(s) \rangle, \emptyset\right), (\overline{tag}_s, \emptyset), \\ (write_s(x).s := x.\, tag_s.\texttt{unlock}\ s, \{s\}) \end{array} \right\}_{s \in unlocks(A_1)} \end{array} \right)$$

Then we apply $\mathcal{C}$ to D_2, and from $\mathcal{C}[D_1] \approx \mathcal{C}[D_2]$. There should exist D_2' such that $\mathcal{C}[D_2] \stackrel{\epsilon}{\Longrightarrow} D_2'$ and $D_2' \approx D_1'$. Since $D_1' \Downarrow_{\overline{a}_w, \overline{a}_{v_s}, \overline{b}_j, \overline{read}_t, \overline{tag}_t, write_t}$ for $w \in W, s \in U_1 \cup U_2, j \in J, t \in unlocks(A_1)$ and $D_1' \not\Downarrow_{\overline{e}}$, the only possibility for D_2'

is that

$$\mathcal{C}[D_2] =$$

$$\nu\widetilde{c}, \widetilde{\boldsymbol{n}}'_{\boldsymbol{2}}. \left(\sigma_{2\setminus W}, \boldsymbol{S}'_{\boldsymbol{2}}, \begin{array}{c} \boldsymbol{\mathcal{P}}'_{\boldsymbol{2}} \cup \{(\overline{a}_w\langle w\sigma_1\rangle, \emptyset)\}_{w\in W} \cup \left\{(\overline{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J} \\ \cup \{(\overline{a}_{v_s}\langle S_2(s)\rangle, \emptyset)\}_{s\in U_1\cup U_2} \\ \cup \left\{ \begin{array}{l} \left(\overline{read}_s\langle S_2(s)\rangle, \emptyset\right), (\overline{tag}_s, \emptyset), \\ (write_s(x).s := x.\ tag_s.\texttt{unlock}\ s, \{s\}) \end{array} \right\}_{s\in unlocks(A_1)} \\ \cup \{(\overline{e}, \emptyset), (\overline{\boldsymbol{a}}\langle\boldsymbol{d}\rangle.e, \emptyset)\} \end{array} \right)$$

$$\implies D'_2 :=$$

$$\nu\widetilde{c}, \widetilde{\boldsymbol{n}}''_{\boldsymbol{2}}. \left(\sigma_{2\setminus W}, \boldsymbol{S}''_{\boldsymbol{2}}, \begin{array}{c} \boldsymbol{\mathcal{P}}''_{\boldsymbol{2}} \cup \{(\overline{a}_w\langle w\sigma_1\rangle, \emptyset)\}_{w\in W} \cup \left\{(\overline{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J} \\ \cup \{(\overline{a}_{v_s}\langle S_1(s)\rangle, \emptyset)\}_{s\in U_1\cup U_2} \\ \cup \left\{ \begin{array}{l} \left(\overline{read}_s\langle S_2(s)\rangle, \emptyset\right), (\overline{tag}_s, \emptyset), \\ (write_s(x).s := x.\ tag_s.\texttt{unlock}\ s, \{s\}) \end{array} \right\}_{s\in unlocks(A_1)} \end{array} \right)$$

In the transitions $\mathcal{C}[D_2] \implies D'_2$, there is no operation on public state cells in $unlocks(A_1)$ because these cells are all locked. So we can deduce that

$$E = \nu\widetilde{n}'_2.\,(\varphi_2, S'_2, \mathcal{P}'_2) \stackrel{a(d)}{\Longrightarrow} B'_2 := \nu\widetilde{n}''_2.\,(\varphi_2, S''_2, \mathcal{P}''_2)$$

From $B_2 \Longrightarrow E$, we have $B_2 \stackrel{a(d)}{\Longrightarrow} B'_2$. From $D''_1 \approx D''_2$, we have that $(B'_1, B'_2) \in \mathcal{R}$.

ii. when $a = c_k$ for some $k \in J$ and $d \notin \widetilde{c}$, we construct an evaluation context $\mathcal{C}$:

$$\mathcal{C} = \left(\text{-}, \text{-}, \left\{(\overline{e}, \emptyset), \left(b_k(u).\overline{u}\langle d\rangle.e.\overline{b}_k\langle u\rangle, \emptyset\right)\right\} \text{-}\right)$$

where e is a fresh channel name. The analysis is similar as above.

iii. when $a \notin \widetilde{c}$ and $d = c_k$ for some $k \in J$, we construct an evaluation context $\mathcal{C}$:

$$\mathcal{C} = \left(\text{-}, \text{-}, \left\{(\overline{e}, \emptyset), \left(b_k(u).\overline{a}\langle u\rangle.e.\overline{b}_k\langle u\rangle, \emptyset\right)\right\} \text{-}\right)$$

where e is a fresh channel name. The analysis is similar as above.

iv. when $a = d = c_k$ for some $k \in J$, we construct an evaluation context $\mathcal{C}$:

$$\mathcal{C} = \left(\text{-}, \text{-}, \left\{(\overline{e}, \emptyset), \left(b_k(u).\overline{u}\langle u\rangle.e.\overline{b}_k\langle u\rangle, \emptyset\right)\right\} \text{-}\right)$$

where e is a fresh channel name. The analysis is similar as above.

v. when $a = c_k$ and $d = c_m$ for some $k, m \in J$ and $k \neq m$, we construct an evaluation context $\mathcal{C}$:

$$\mathcal{C} = \left(\text{-}, \text{-}, \left\{(\overline{e}, \emptyset), \left(b_k(u).b_m(v).\overline{u}\langle v\rangle.(e.\overline{b}_k\langle u\rangle \mid \overline{b}_m\langle v\rangle), \emptyset\right)\right\} \text{-}\right)$$

where e is a fresh channel name. The analysis is similar as above.

(g) Assume $B_1 = \nu\widetilde{n}_1.(\varphi_1, S_1, \mathcal{Q}_1 \cup \{(\overline{a}\langle d\rangle.P_1, L_1)\}) \xrightarrow{\overline{a}\langle d\rangle} B_1' := \nu\widetilde{n}_1.(\varphi_1, S_1, \mathcal{Q}_1 \cup \{(P_1, L_1)\})$ with $a, d \notin \widetilde{n}_1$ and $\mathcal{P}_1 = \mathcal{Q}_1 \cup \{(\overline{a}\langle d\rangle.P_1, L_1)\}$.

i. when $a, d \notin \widetilde{c}$, we construct an evaluation context $\mathcal{C}$:

$$\mathcal{C} = (\text{-}, \text{-}, \{(\overline{e}, \emptyset), (a(x).\mathtt{if}\ x = d\ \mathtt{then}\ e, \emptyset)\}\ \text{-})$$

where e is a fresh channel name. Applying $\mathcal{C}$ to D_1, we can get the following transitions:

$$\mathcal{C}[D_1] = \nu\widetilde{c}, \widetilde{n}_1.\left(\sigma_{1\backslash W}, S_1, \begin{array}{l} \mathcal{Q}_1 \cup \{(\overline{a}\langle d\rangle.P_1, L_1)\} \\ \quad \cup \{(\overline{a}_w\langle w\sigma_1\rangle, \emptyset)\}_{w\in W} \cup \left\{(\overline{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J} \\ \quad \cup \{(\overline{a}_{v_s}\langle S_1(s)\rangle, \emptyset)\}_{s\in U_1\cup U_2} \\ \quad \cup \left\{ \begin{array}{l} \left(\overline{read}_s\langle S_1(s)\rangle, \emptyset\right), (\overline{tag}_s, \emptyset), \\ (write_s(x).s := x.\, tag_s.\mathtt{unlock}\ s, \{s\}) \end{array} \right\}_{s\in unlocks(A_1)} \\ \quad \cup \{(\overline{e}, \emptyset), (a(x).\mathtt{if}\ x = d\ \mathtt{then}\ e, \emptyset)\} \end{array}\right)$$

$$\Longrightarrow D_1' := \nu\widetilde{c}, \widetilde{n}_1.\left(\sigma_{1\backslash W}, S_1, \begin{array}{l} \mathcal{Q}_1 \cup \{(P_1, L_1)\} \\ \quad \cup \{(\overline{a}_w\langle w\sigma_1\rangle, \emptyset)\}_{w\in W} \cup \left\{(\overline{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J} \\ \quad \cup \{(\overline{a}_{v_s}\langle S_1(s)\rangle, \emptyset)\}_{s\in U_1\cup U_2} \\ \quad \cup \left\{ \begin{array}{l} \left(\overline{read}_s\langle S_1(s)\rangle, \emptyset\right), (\overline{tag}_s, \emptyset), \\ (write_s(x).s := x.\, tag_s.\mathtt{unlock}\ s, \{s\}) \end{array} \right\}_{s\in unlocks(A_1)} \end{array}\right)$$

Then we apply $\mathcal{C}$ to D_2. Since $\mathcal{C}[D_1] \approx \mathcal{C}[D_2]$, there should exist D_2' such that $\mathcal{C}[D_2] \stackrel{\epsilon}{\Longrightarrow} D_2' \approx D_1'$.
From $D_1' \Downarrow_{\overline{a}_w, \overline{a}_{v_s}, \overline{b}_j, \overline{read}_t, \overline{tag}_t, write_t}$ for $w \in W, s \in U_1 \cup U_2, j \in J, t \in unlocks(A_1)$ and $D_1' \not\Downarrow_{\overline{e}}$, the only possibility of D_2' is that:

$$\mathcal{C}[D_2] = \nu\widetilde{c}, \boldsymbol{\widetilde{n}_2'}.\left(\sigma_{2\backslash W}, \boldsymbol{S_2'}, \begin{array}{l} \boldsymbol{\mathcal{P}_2'} \cup \{(\overline{a}_w\langle w\sigma_2\rangle, \emptyset)\}_{w\in W} \cup \left\{(\overline{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J} \\ \quad \cup \{(\overline{a}_{v_s}\langle S_2(s)\rangle, \emptyset)\}_{s\in U_1\cup U_2} \\ \quad \cup \left\{ \begin{array}{l} \left(\overline{read}_s\langle S_2(s)\rangle, \emptyset\right), (\overline{tag}_s, \emptyset), \\ (write_s(x).s := x.\, tag_s.\mathtt{unlock}\ s, \{s\}) \end{array} \right\}_{s\in unlocks(A_1)} \\ \quad \cup \{(\overline{e}, \emptyset), (\boldsymbol{a(x)}.\mathtt{if}\ \boldsymbol{x = d}\ \mathtt{then}\ e, \emptyset)\} \end{array}\right)$$

$$\Longrightarrow D_2' :=$$

$$\nu\widetilde{c},\widetilde{\boldsymbol{n}}_{\boldsymbol{2}}''.\left(\sigma_{2\backslash W}, \boldsymbol{S_2''}, \begin{array}{c}\boldsymbol{\mathcal{Q}_2''} \cup \{(\overline{a}_w\langle w\sigma_2\rangle, \emptyset)\}_{w\in W} \cup \left\{(\overline{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J} \\ \cup\, \{(\overline{a}_{v_s}\langle S_2(s)\rangle, \emptyset)\}_{s\in U_1\cup U_2} \\ \cup \left\{\begin{array}{l}\left(\overline{read}_s\langle S_2(s)\rangle, \emptyset\right), (\overline{tag}_s, \emptyset), \\ (write_s(x).s := x.\, tag_s.\texttt{unlock}\ s, \{s\})\end{array}\right\}_{s\in unlocks(A_1)}\end{array}\right)$$

In the transitions $\mathcal{C}[D_2] \Longrightarrow D_2'$, there is no operation on public state cells in $unlocks(A_1)$ because these cells are all locked. So we can deduce that

$$E = \nu\widetilde{n}_2'.\,(\varphi_2, S_2', \mathcal{P}_2') \overset{\overline{a}\langle d\rangle}{\Longrightarrow} B_2' := \nu\widetilde{n}_2''.\,(\varphi_2, S_2'', \mathcal{P}_2'')$$

From $B_2 \Longrightarrow E$, we have $B_2 \overset{\overline{a}\langle d\rangle}{\Longrightarrow} B_2'$. From $D_1'' \approx D_2''$, we have that $(B_1', B_2') \in \mathcal{R}$.

ii. when $a = c_k$ for some $k \in J$ and $d \notin \widetilde{c}$, we construct an evaluation context $\mathcal{C}$:

$$\mathcal{C} = \left(\text{-}, \text{-}, \left\{(\overline{e}, \emptyset), \left(b_k(u).u(x).\texttt{if}\ x = d\ \texttt{then}\ e.\overline{b}_k\langle u\rangle, \emptyset\right)\right\} \text{-}\right)$$

where e is a fresh channel name. The analysis is similar as above.

iii. when $a \notin \widetilde{c}$ and $d = c_k$ for some $k \in J$, we construct an evaluation context $\mathcal{C}$:

$$\mathcal{C} = \left(\text{-}, \text{-}, \left\{(\overline{e}, \emptyset), \left(b_k(u).a(x).\texttt{if}\ x = u\ \texttt{then}\ e.\overline{b}_k\langle u\rangle, \emptyset\right)\right\} \text{-}\right)$$

where e is a fresh channel name. The analysis is similar as above.

iv. when $a = d = c_k$ for some $k \in J$, we construct an evaluation context $\mathcal{C}$:

$$\mathcal{C} = \left(\text{-}, \text{-}, \left\{(\overline{e}, \emptyset), \left(b_k(u).u(x).\texttt{if}\ x = u\ \texttt{then}\ e.\overline{b}_k\langle u\rangle, \emptyset\right)\right\} \text{-}\right)$$

where e is a fresh channel name. The analysis is similar as above.

v. when $a = c_k$ and $d = c_m$ for some $k, m \in J$ and $k \neq m$, we construct an evaluation context $\mathcal{C}$:

$$\mathcal{C} = \left(\text{-}, \text{-}, \left\{(\overline{e}, \emptyset), \left(b_k(u).b_m(v).u(x).\texttt{if}\ x = v\ \texttt{then}\ \left(\overline{b}_k\langle u\rangle \mid e.\overline{b}_m\langle v\rangle\right), \emptyset\right)\right\} \text{-}\right)$$

where e is a fresh channel name. The analysis is similar as above.

(h) Assume $B_1 = \nu\widetilde{n}_1', d.(\varphi_1, S_1, \mathcal{Q}_1 \cup \{(\overline{a}\langle d\rangle.P_1, L_1)\}) \xrightarrow{\nu d.\overline{a}\langle d\rangle} B_1' := \nu\widetilde{n}_1'.(\varphi_1, S_1, \mathcal{Q}_1 \cup \{(P_1, L_1)\})$ with $a, d \notin \widetilde{n}_1'$ and $\mathcal{P}_1 = \mathcal{Q}_1 \cup \{(\overline{a}\langle d\rangle.P_1, L_1)\}$.

i. when $a \notin \widetilde{c}$, we construct an evaluation context $\mathcal{C}$:

$$\mathcal{C} = \left(\text{-}, \text{-}, \left\{(\overline{e}, \emptyset), \left(a(x).\texttt{if}\ x \in fn(B_1, B_2)\ \texttt{then}\ 0\ \texttt{else}\ e.\overline{b}_m\langle x\rangle, \emptyset\right)\right\} \text{-}\right)$$

where e, b_m are different fresh channel names. Applying $\mathcal{C}$ to D_1, we can get the following transitions:

$$\mathcal{C}[D_1] = \nu\widetilde{c}, \widetilde{n}'_1, d. \left(\sigma_{1\setminus W}, S_1, \begin{array}{l} \mathcal{Q}_1 \cup \{(\overline{a}\langle d\rangle.P_1, L_1), (\overline{e}, \emptyset)\} \\ \quad \cup \left\{ \left(a(x).\mathtt{if}\ \boldsymbol{x \in fn(B_1, B_2)}\ \mathtt{then}\ 0\ \mathtt{else}\ e.\bar{b}_m\langle x\rangle, \emptyset\right) \right\} \\ \quad \cup \{(\overline{a}_w\langle w\sigma_1\rangle, \emptyset)\}_{w\in W} \cup \left\{(\bar{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J} \\ \quad \cup \{(\overline{a}_{v_s}\langle S_1(s)\rangle, \emptyset)\}_{s\in U_1\cup U_2} \\ \quad \cup \left\{ \begin{array}{l} \left(\overline{read}_s\langle S_1(s)\rangle, \emptyset\right), (\overline{tag}_s, \emptyset), \\ (write_s(x).s := x.\, tag_s.\mathtt{unlock}\ s, \{s\}) \end{array} \right\}_{s\in unlocks(A_1)} \end{array} \right)$$

$$\Longrightarrow D'_1 :=$$

$$\nu\widetilde{c}, \widetilde{n}'_1, d. \left(\sigma_{1\setminus W}, S_1, \begin{array}{l} \mathcal{Q}_1 \cup \{(P_1, L_1)\} \cup \{(\overline{a}_w\langle w\sigma_1\rangle, \emptyset)\}_{w\in W} \\ \quad \cup \{(\overline{a}_{v_s}\langle S_1(s)\rangle, \emptyset)\}_{s\in U_1\cup U_2} \\ \quad \cup \left\{(\bar{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J} \cup \left\{(\bar{b}_m\langle d\rangle, \emptyset)\right\} \\ \quad \cup \left\{ \begin{array}{l} \left(\overline{read}_s\langle S_1(s)\rangle, \emptyset\right), (\overline{tag}_s, \emptyset), \\ (write_s(x).s := x.\, tag_s.\mathtt{unlock}\ s, \{s\}) \end{array} \right\}_{s\in unlocks(A_1)} \end{array} \right)$$

Then we apply $\mathcal{C}$ to D_2. Since $\mathcal{C}[D_1] \approx \mathcal{C}[D_2]$, there should exist D'_2 such that $\mathcal{C}[D_2] \stackrel{\epsilon}{\Longrightarrow} D'_2 \approx D'_1$.
From $D'_1 \Downarrow_{\overline{a}_w, \overline{a}_{v_s}, \bar{b}_j, \bar{b}_m, \overline{read}_t, \overline{tag}_t, write_t}$ for $w \in W, s \in U_1 \cup U_2, j \in J, t \in unlocks(A_1)$ and $D'_1 \not\Downarrow_{\overline{e}}$, the only possibility of D'_2 is that:

$$\mathcal{C}[D_2] = \nu\widetilde{c}, \widetilde{n}'_2. \left(\sigma_{2\setminus W}, S'_2, \begin{array}{l} \mathcal{P}'_2 \cup \{(\overline{a}_w\langle w\sigma_2\rangle, \emptyset)\}_{w\in W} \cup \left\{(\bar{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J} \\ \quad \cup \{(\overline{a}_{v_s}\langle S_2(s)\rangle, \emptyset)\}_{s\in U_1\cup U_2} \\ \quad \cup \left\{ \begin{array}{l} \left(\overline{read}_s\langle S_2(s)\rangle, \emptyset\right), (\overline{tag}_s, \emptyset), \\ (write_s(x).s := x.\, tag_s.\mathtt{unlock}\ s, \{s\}) \end{array} \right\}_{s\in unlocks(A_1)} \\ \quad \cup \left\{(\overline{e}, \emptyset), \left(a(x).\mathtt{if}\ x \in fn(B_1, B_2)\ \mathtt{then}\ 0\ \mathtt{else}\ e.\bar{b}_m\langle x\rangle, \emptyset\right)\right\} \end{array} \right)$$

$$\Longrightarrow D'_2 :=$$

$$\nu\widetilde{c}, \widetilde{n}''_2, d. \left(\sigma_{2\setminus W}, S''_2, \begin{array}{l} \mathcal{P}''_2 \cup \{(\overline{a}_w\langle w\sigma_2\rangle, \emptyset)\}_{w\in W} \cup \left\{(\bar{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J} \\ \quad \cup \{(\overline{a}_{v_s}\langle S_2(s)\rangle, \emptyset)\}_{s\in U_1\cup U_2} \\ \quad \cup \left\{ \begin{array}{l} \left(\overline{read}_s\langle S_2(s)\rangle, \emptyset\right), (\overline{tag}_s, \emptyset), \\ (write_s(x).s := x.\, tag_s.\mathtt{unlock}\ s, \{s\}) \end{array} \right\}_{s\in unlocks(A_1)} \\ \quad \cup \left\{\left(\bar{b}_m\langle d\rangle, \emptyset\right)\right\} \end{array} \right)$$

In the transitions $\mathcal{C}[D_2] \Longrightarrow D'_2$, there is no operation on public state cells in $unlocks(A_1)$ because these cells are all locked. So we can deduce that

$$E = \nu\widetilde{n}'_2.\, (\varphi_2, S'_2, \mathcal{P}'_2) \stackrel{\nu d.\overline{a}\langle d\rangle}{\Longrightarrow} B'_2 := \nu\widetilde{n}''_2.\, (\varphi_2, S''_2, \mathcal{P}''_2)$$

From $B_2 \Longrightarrow E$, we have $B_2 \stackrel{\nu d.\overline{a}\langle d\rangle}{\Longrightarrow} B_2'$. From $D_1'' \approx D_2''$, we have that $(B_1', B_2') \in \mathcal{R}$.

ii. when $a = c_k$ for some $k \in J$ and $d \notin \widetilde{c}$, we construct an evaluation context $\mathcal{C}$:

$$\mathcal{C} = \left(\text{-}, \text{-}, \left\{(\overline{e}, \emptyset), \left(b_k(u).u(x).\texttt{if } x \in fn(B_1, B_2) \texttt{ then } 0 \texttt{ else } (e.\overline{b}_k\langle u\rangle \mid \overline{b}_m\langle x\rangle), \emptyset\right)\right\} \text{-}\right)$$

where e, b_m are different fresh channel names. The analysis is similar as above.

(i) Assume $B_1 = \nu\widetilde{n}_1.(\varphi_1, S_1, \mathcal{Q}_1 \cup \{(\overline{a}\langle M_1\rangle.P_1, L_1)\}) \xrightarrow{\nu x.\overline{a}\langle x\rangle} \nu\widetilde{n}_1.(\varphi_1 \cup \{M_1/x\}, S_1, \mathcal{P} \cup \{(P_1, L_1)\})$ with $a \notin \widetilde{n}_1$ and M_1 is of the base sort and x is fresh.

i. when $a \notin \widetilde{c}$, selecting a fresh channel name a_x, we construct an evaluation context $\mathcal{C}$:

$$\mathcal{C} = (\text{-}, \text{-}, \{(\overline{e}, \emptyset), (a(z).e.\overline{a}_x\langle z\rangle, \emptyset)\} \text{-})$$

where e is a fresh channel name. Applying $\mathcal{C}$ to D_1, we can get the following transitions:

$$\mathcal{C}[D_1] =$$

$$\nu\widetilde{c}, \widetilde{n}_1. \left(\sigma_{1\backslash W}, S_1, \begin{array}{l} \mathcal{Q}_1 \cup \{(\overline{a}\langle M_1\rangle.P_1, L_1)\} \\ \quad \cup \{(\overline{a}_w\langle w\sigma_1\rangle, \emptyset)\}_{w\in W} \cup \left\{(\overline{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J} \\ \quad \cup \{(\overline{a}_{v_s}\langle S_1(s)\rangle, \emptyset)\}_{s\in U_1\cup U_2} \\ \quad \cup \left\{ \begin{array}{l} \left(\overline{read}_s\langle S_1(s)\rangle, \emptyset\right), (\overline{tag}_s, \emptyset), \\ (write_s(x).s := x.\, tag_s.\texttt{unlock } s, \{s\}) \end{array} \right\}_{s\in unlocks(A_1)} \\ \quad \cup \{(\overline{e}, \emptyset), (a(z).e.\overline{a}_x\langle z\rangle, \emptyset)\} \end{array} \right)$$

$$\Longrightarrow D_1' :=$$

$$\nu\widetilde{c}, \widetilde{n}_1. \left(\sigma_{1\backslash W}, S_1, \begin{array}{l} \mathcal{Q}_1 \cup \{(P_1, L_1)\} \\ \quad \cup \{(\overline{a}_w\langle w\sigma_1\rangle, \emptyset)\}_{w\in W} \cup \left\{(\overline{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J} \\ \quad \cup \{(\overline{a}_{v_s}\langle S_1(s)\rangle, \emptyset)\}_{s\in U_1\cup U_2} \\ \quad \cup \left\{ \begin{array}{l} \left(\overline{read}_s\langle S_1(s)\rangle, \emptyset\right), (\overline{tag}_s, \emptyset), \\ (write_s(x).s := x.\, tag_s.\texttt{unlock } s, \{s\}) \end{array} \right\}_{s\in unlocks(A_1)} \\ \quad \cup \{(\overline{a}_x\langle M_1\rangle, \emptyset)\} \end{array} \right)$$

Then we apply $\mathcal{C}$ to D_2. Since $\mathcal{C}[D_1] \approx \mathcal{C}[D_2]$, there should exist D_2' such that $\mathcal{C}[D_2] \stackrel{\epsilon}{\Longrightarrow} D_2' \approx D_1'$.

From $D_1' \Downarrow_{\overline{a}_w, \overline{a}_{v_s}, \overline{a}_x, \overline{b}_j, \overline{b}_m, \overline{read}_t, \overline{tag}_t, write_t}$ for $w \in W, s \in U_1 \cup U_2, j \in J, t \in$

$unlocks(A_1)$ and $D_1' \not\Downarrow_{\overline{e}}$, the only possibility of D_2' is that:

$$\mathcal{C}[D_2] =$$

$$\nu\widetilde{c}, \widetilde{\boldsymbol{n}}_{\boldsymbol{2}}'. \left(\sigma_{2\backslash W}, \boldsymbol{S_2'}, \begin{array}{c} \boldsymbol{\mathcal{P}_2'} \cup \{(\overline{a}_w\langle w\sigma_2\rangle, \emptyset)\}_{w\in W} \cup \left\{(\overline{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J} \\ \cup \{(\overline{a}_{v_s}\langle S_2(s)\rangle, \emptyset)\}_{s\in U_1\cup U_2} \\ \cup \left\{ \begin{array}{l} \left(\overline{read}_s\langle S_2(s)\rangle, \emptyset\right), (\overline{tag}_s, \emptyset), \\ (write_s(x).s := x.\, tag_s.\mathtt{unlock}\ s, \{s\}) \end{array} \right\}_{s\in unlocks(A_1)} \\ \cup \{(\overline{e}, \emptyset), (\boldsymbol{a(z)}.e.\overline{a}_x\langle z\rangle, \emptyset)\} \end{array} \right)$$

$$\Longrightarrow D_2' :=$$

$$\nu\widetilde{c}, \widetilde{\boldsymbol{n}}_{\boldsymbol{2}}''. \left(\sigma_{2\backslash W}, \boldsymbol{S_2''}, \begin{array}{c} \boldsymbol{\mathcal{P}_2''} \cup \{(\overline{a}_w\langle w\sigma_2\rangle, \emptyset)\}_{w\in W} \cup \left\{(\overline{b}_j\langle c_j\rangle, \emptyset)\right\}_{j\in J} \\ \cup \{(\overline{a}_{v_s}\langle S_2(s)\rangle, \emptyset)\}_{s\in U_1\cup U_2} \\ \cup \left\{ \begin{array}{l} \left(\overline{read}_s\langle S_2(s)\rangle, \emptyset\right), (\overline{tag}_s, \emptyset), \\ (write_s(x).s := x.\, tag_s.\mathtt{unlock}\ s, \{s\}) \end{array} \right\}_{s\in unlocks(A_1)} \\ \cup \{(\overline{a}_x\langle M_2\rangle, \emptyset)\} \end{array} \right)$$

In the transitions $\mathcal{C}[D_2] \Longrightarrow D_2'$, there is no operation on public state cells in $unlocks(A_1)$ because these cells are all locked. So we can deduce that

$$E = \nu\widetilde{n}_2'.\,(\varphi_2, S_2', \mathcal{P}_2') \overset{\nu x.\overline{a}\langle x\rangle}{\Longrightarrow} B_2' := \nu\widetilde{n}_2''.\,(\varphi_2 \cup \{M_2/x\}, S_2'', \mathcal{P}_2'')$$

From $B_2 \Longrightarrow E$, we have $B_2 \overset{\nu x.\overline{a}\langle x\rangle}{\Longrightarrow} B_2'$. From $D_1'' \approx D_2''$, we have that $(B_1', B_2') \in \mathcal{R}$.

ii. when $a = c_k$ for some $k \in J$, selecting a fresh channel name a_x, we construct an evaluation context $\mathcal{C}$:

$$\mathcal{C} = (\text{-}, \text{-}, \{(\overline{e}, \emptyset), (b_k(u).u(z).e.\overline{a}_x\langle z\rangle, \emptyset)\}\ \text{-})$$

where e is a fresh channel name. The analysis is similar as above.

In the presence of public state cells, labelled bisimilarity is both sound and complete with respect to observational equivalence.

Theorem 42 (Coincidence). *In the presence of public state cells,* $\approx_l = \approx$.

Proof. Using Proposition 38 and Proposition 41.

7. Conclusion

We present a stateful language which is a general extension of applied pi calculus with state cells. We stick to the original definition of observational equivalence [3] as much as possible to capture the intuition of indistinguishability from the attacker's point of view, while design the labelled bisimilarity to furthest abstract observational equivalence. When all the state cells are private, we prove that observational equivalence coincides with labelled bisimilarity, which implies Abadi-Fournet's theorem in a revised version of applied pi calculus. In the presence of public state cells, we devise a labelled bisimilarity which is proved to coincide with observational equivalence. In future, we plan to develop a compiler for bi-processes with state cells to automatically verify the observational equivalence, extending the techniques of ProVerif.

Acknowledgements. Dr Jia Liu's work has been funded by the UK EPSRC as part of the PETRAS IoT Research Hub - Cybersecurity of the Internet of Things grant no: EP/N02334X/1.

[1] LinkedIn investigates hacking claims. `http://www.guardian.co.uk/technology/2012/jun/06/linkedin-hacking`.

[2] M. Abadi and V. Cortier. Deciding knowledge in security protocols under equational theories. *Theor. Comput. Sci.*, 367(1-2):2–32, 2006.

[3] M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *Proc. 28th Symposium on Principles of Programming Languages (POPL'01)*, pages 104–115. ACM Press, 2001.

[4] M. Abadi and C. Fournet. Private authentication. *Theor. Comput. Sci.*, 322(3):427–476, 2004.

[5] M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: The spi calculus. In *4th ACM Conference on Computer and Communications Security*, pages 36–47. ACM Press, 1997.

[6] M. Arapinis, T. Chothia, E. Ritter, and M. Ryan. Analysing unlinkability and anonymity using the applied pi calculus. In *Proceedings of IEEE 23rd Computer Security Foundations Symposium*, CSF'10, pages 107–121, 2010.

[7] M. Arapinis, J. Liu, E. Ritter, and M. Ryan. Stateful applied pi calculus. In *Principles of Security and Trust - Third International Conference, POST 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5-13, 2014, Proceedings*, pages 22–41, 2014.

[8] M. Arapinis, E. Ritter, and M. D. Ryan. Statverif: Verification of stateful processes. In *Proceedings of IEEE 24th Computer Security Foundations Symposium*, CSF '11, pages 33–47, 2011. `http://markryan.eu/research/statverif/`.

[9] M. Backes, M. Maffei, and D. Unruh. Zero-knowledge in the applied pi-calculus and automated verification of the direct anonymous attestation protocol. In *IEEE Symposium on Security and Privacy*, pages 202–215, 2008.

[10] M. Baudet, V. Cortier, and S. Delaune. YAPA: A generic tool for computing intruder knowledge. *ACM Transactions on Computational Logic*, 14, 2013.

[11] J. Bengtson, M. Johansson, J. Parrow, and B. Victor. Psi-calculi: a framework for mobile processes with nominal data and logic. *Logical Methods in Computer Science*, 7(1), 2011.

[12] K. Bhargavan, C. Fournet, R. Corin, and E. Zalinescu. Cryptographically verified implementations for TLS. In *Proceedings of the 15th ACM conference on Computer and communications security*, CCS '08, pages 459–468. ACM, 2008.

[13] B. Blanchet. Automatic Proof of Strong Secrecy for Security Protocols. In *IEEE Symposium on Security and Privacy*, pages 86–100, 2004.

[14] B. Blanchet. Automatic verification of correspondences for security protocols. *Journal of Computer Security*, 17(4):363–434, 2009.

[15] B. Blanchet. Automatic Verification of Correspondences for Security Protocols. *Journal of Computer Security*, 17(4):363–434, 2009.

[16] R. Chadha, Ş. Ciobâcă, and S. Kremer. Automated verification of equivalence properties of cryptographic protocols. In *Programming Languages and Systems —Proceedings of the 21th European Symposium on Programming (ESOP'12)*, volume 7211, pages 108–127, Tallinn, Estonia, 2012. Springer.

[17] V. Cheval, H. Comon-Lundh, and S. Delaune. Automating security analysis: symbolic equivalence of constraint systems. In *Proceedings of the 5th International Joint Conference on Automated Reasoning (IJCAR'10)*, volume 6173, pages 412–426, 2010.

[18] V. Cheval, H. Comon-Lundh, and S. Delaune. Trace equivalence decision: Negative tests and non-determinism. In *Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS'11)*, pages 321–330, 2011.

[19] V. Cheval, V. Cortier, and S. Delaune. Deciding equivalence-based properties using constraint solving. *Theoretical Computer Science*, 492:1–39, 2013.

[20] V. Cortier and S. Delaune. A method for proving observational equivalence. In *CSF'09*, pages 266–276. IEEE Computer Society Press, July 2009.

[21] V. Cortier, M. Rusinowitch, and E. Zalinescu. Relating two standard notions of secrecy. In *Logical Methods in Computer Science*, volume 3, pages 1–29, 2007.

[22] S. Delaune, S. Kremer, and M. D. Ryan. Verifying privacy-type properties of electronic voting protocols. In *Journal of Computer Security*, 2009.

[23] S. Delaune, S. Kremer, and M. D. Ryan. Symbolic bisimulation for the applied pi calculus. *Journal of Computer Security*, 18(2):317–377, 2010.

[24] S. Delaune, S. Kremer, M. D. Ryan, and G. Steel. Formal analysis of protocols based on TPM state registers. In *Proc. of the 24th IEEE Computer Security Foundations Symposium (CSF'11)*. IEEE Computer Society Press, 2011.

[25] D. Dreyer, G. Neis, A. Rossberg, and L. Birkedal. A relational modal logic for higher-order stateful ADTs. In *Proceedings of the 37th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2010*, pages 185–198, 2010.

[26] F. D. Garcia and P. van Rossum. Modeling privacy for off-line RFID systems. In *CARDIS*, pages 194–208, 2010.

[27] J. D. Guttman. Fair exchange in strand spaces. *Journal of Automated Reasoning*, 2012.

[28] M. O. Koutarou, K. Suzuki, and S. Kinoshita. Cryptographic approach to "privacy-friendly" tags. In *In RFID Privacy Workshop*, 2003.

[29] S. Kremer and R. Knnemann. Automated analysis of security protocols with global state. In *2014 IEEE Symposium on Security and Privacy*, pages 163–178, 2014.

[30] R. Künnemann and G. Steel. YubiSecure? formal security analysis results for the Yubikey and YubiHSM. In *Preliminary Proceedings of the 8th Workshop on Security and Trust Management (STM'12)*, 2012.

[31] J. Liu. A proof of coincidence of labeled bisimilarity and observational equivalence in applied pi calculus. `http://mail.ios.ac.cn/~jliu/papers/Proof.pdf`, 2011. Technical Report, ISCAS-SKLCS-11-05.

[32] J. Liu and H. Lin. A complete symbolic bisimulation for full applied pi calculus. *Theoretical Computer Science*, 458:76–112, 2012.

[33] S. Mödersheim. Abstraction by set-membership: verifying security protocols and web services with databases. In *Proc. 17th ACM Conference on Computer and Communications Security (CCS'10)*, pages 351–360. ACM, 2010.

[34] P. Ryan, S. Schneider, M. Goldsmith, G. Lowe, and B. Roscoe. *The Modelling and Analysis of Security Protocols*. Pearson Education, 2001.

[35] D. Sangiorgi and D. Walker. *The π-calculus: A Theory of Mobile Processes.* Cambridge University Press, Cambridge, UK, 2001.

[36] B. Schmidt, S. Meier, C. Cremers, and D. Basin. Automated analysis of diffie-hellman protocols and advanced security properties. In *25th Computer Security Foundations Symposium (CSF)*, pages 78–94. IEEE, 2012.

[37] A. Turon, D. Dreyer, and L. Birkedal. Unifying refinement and hoare-style reasoning in a logic for higher-order concurrency. In *ICFP 2013*, pages 377–390, 2013.

Appendix A. Proofs in Section 3.2

Lemma 6. Let A be a closed extended process with only private state cells and $\mathcal{C} = \nu\widetilde{n}.(\sigma\text{-}, S\text{-}, \mathcal{P}\text{-})$ be a closing evaluation context with only private state cells and $\widetilde{x} \subseteq dom(A)$.

1. If $A \xrightarrow{c(M\sigma)} B$ with $name(c, M) \cap \widetilde{n} = \emptyset$ and $var(M) \subseteq dom(\mathcal{C}[A]_{\setminus\widetilde{x}})$, then $\mathcal{C}[A]_{\setminus\widetilde{x}} \xrightarrow{c(M)} \mathcal{C}[B]_{\setminus\widetilde{x}}$;

2. If $A \xrightarrow{\alpha} B$ with $name(\alpha) \cap \widetilde{n} = \emptyset$ and $var(\alpha) \cap \widetilde{x} = \emptyset$, then $\mathcal{C}[A]_{\setminus\widetilde{x}} \xrightarrow{\alpha} \mathcal{C}[B]_{\setminus\widetilde{x}}$ when α is not an input.

Proof.

1. Assume $A = \nu\widetilde{n}_a.(\sigma_a, S_a, \mathcal{P}_a \cup \{(c(z).P, L)\}) \xrightarrow{c(M\sigma)} B = \nu\widetilde{n}_a.(\sigma_a, S_a, \mathcal{P}_a \cup \{(P\{(M\sigma)\sigma_a/z\}, L)\})$ where $\widetilde{n} \cap \widetilde{n}_a = \emptyset$. Then

$$\begin{aligned}\mathcal{C}[A]_{\setminus\widetilde{x}} =\ & \nu\widetilde{n}, \widetilde{n}_a.(\sigma\sigma_a \cup \sigma_{a\setminus\widetilde{x}}, S\sigma_a \cup S_a, \mathcal{P}\sigma_a \cup \mathcal{P}_a \cup \{(c(z).P, L)\}) \\ \xrightarrow{c(M)}\ & \nu\widetilde{n}, \widetilde{n}_a.(\sigma\sigma_a \cup \sigma_{a\setminus\widetilde{x}}, S\sigma_a \cup S_a, \mathcal{P}\sigma_a \cup \mathcal{P}_a \cup \{(P\{M(\sigma\sigma_a \cup \sigma_{a\setminus\widetilde{x}})/z\}, L)\}) \\ =\ & \nu\widetilde{n}, \widetilde{n}_a.(\sigma\sigma_a \cup \sigma_{a\setminus\widetilde{x}}, S\sigma_a \cup S_a, \mathcal{P}\sigma_a \cup \mathcal{P}_a \cup \{(P\{(M\sigma)\sigma_a/z\}, L)\}) = \mathcal{C}[B]_{\setminus\widetilde{x}} \\ & \text{since } var(M) \subseteq dom(\mathcal{C}[A]_{\setminus\widetilde{x}}) \text{ and } (M\sigma)\sigma_a = M(\sigma\sigma_a \cup \sigma_{a\setminus\widetilde{x}})\end{aligned}$$

2. When α is not an input, we take $\texttt{lock}\ s$ and channel output $\bar{b}\langle c\rangle$ as examples. The other cases are quite similar.

(a) Assume $A = \nu\widetilde{n}_a.(\sigma_a, S_a \cup \{s \mapsto M\}, \mathcal{P}_a \cup \{(\texttt{lock}\ s.P, L)\}) \xrightarrow{\tau} B = \nu\widetilde{n}_a.(\sigma_a, S_a \cup \{s \mapsto M\}, \mathcal{P}_a \cup \{(P, L \cup \{s\})\})$ where $s \in \widetilde{n}_a$, $s \notin L \cup locks(\mathcal{P}_a)$ and $\widetilde{n} \cap \widetilde{n}_a = \emptyset$.

$$\begin{aligned}\mathcal{C}[A]_{\setminus\widetilde{x}} =\ & \nu\widetilde{n}, \widetilde{n}_a.(\sigma\sigma_a \cup \sigma_{a\setminus\widetilde{x}}, S\sigma_a \cup S_a \cup \{s \mapsto M\}, \mathcal{P}\sigma_a \cup \mathcal{P}_a \cup \{(\texttt{lock}\ s.P, L)\}) \\ \xrightarrow{\tau}\ & \nu\widetilde{n}, \widetilde{n}_a.(\sigma\sigma_a \cup \sigma_{a\setminus\widetilde{x}}, S\sigma_a \cup S_a \cup \{s \mapsto M\}, \mathcal{P}\sigma_a \cup \mathcal{P}_a \cup \{(P, L \cup \{s\})\}) = \mathcal{C}[B]_{\setminus\widetilde{x}}\end{aligned}$$

because $s \in \widetilde{n}_a$ and $s \notin locks(\mathcal{P}, \mathcal{P}_a) \cup L$.

(b) Assume $A = \nu\widetilde{n}_a.(\sigma_a, S_a, \mathcal{P}_a \cup \{(\bar{b}\langle c\rangle.P, L)\}) \xrightarrow{\bar{b}\langle c\rangle} B = \nu\widetilde{n}_a.(\sigma_a, S_a, \mathcal{P}_a \cup \{(P, L)\})$ where $b, c \notin \widetilde{n}_a \cup \widetilde{n}$ and $\widetilde{n} \cap \widetilde{n}_a = \emptyset$.

$$\begin{aligned}\mathcal{C}[A]_{\setminus\widetilde{x}} &= \nu\widetilde{n}, \widetilde{n}_a.(\sigma\sigma_a \cup \sigma_{a\setminus\widetilde{x}}, S\sigma_a \cup S_a, \mathcal{P}\sigma_a \cup \mathcal{P}_a \cup \{(\bar{b}\langle c\rangle.P, L)\}) \\ &\xrightarrow{\bar{b}\langle c\rangle} \nu\widetilde{n}, \widetilde{n}_a.(\sigma\sigma_a \cup \sigma_{a\setminus\widetilde{x}}, S\sigma_a \cup S_a, \mathcal{P}\sigma_a \cup \mathcal{P}_a \cup \{(P, L)\}) = \mathcal{C}[B]_{\setminus\widetilde{x}}\end{aligned}$$

Appendix B. Proofs in Section 4.2

Lemma 13. Let A, B be two closed pure extended processes. If $B \simeq^1 A \xrightarrow{\alpha} A'$ with $fv(\alpha) \subseteq dom(A)$ then there exists a closed pure extended process B' such that either $B \overset{\widehat{\alpha}}{\Longrightarrow} A'$ or $B \xrightarrow{\alpha} B' \simeq^1 A'$.

Proof. We discuss the eight different cases for $B \simeq^1 A$.

1. Assume $B = \nu\widetilde{n}, m.(\sigma, \mathcal{P}) \simeq \nu\widetilde{n}.(\sigma, \mathcal{P}) = A$ or $B = \nu\widetilde{n}.(\sigma, \mathcal{P}) \simeq \nu\widetilde{n}, m.(\sigma, \mathcal{P}) = A$ with $m \notin fn(\widetilde{n}, \sigma, \mathcal{P})$. Since m is a redundant name, it will not affect any actions from $\mathcal{P}$. Hence $B \xrightarrow{\alpha} A'$.

2. Assume $B = \nu\widetilde{n}.(\sigma, \mathcal{P} \cup \{\nu m.P\}) \simeq \nu\widetilde{n}, m.(\sigma, \mathcal{P} \cup \{P\}) = A$ with $m \notin fn(\widetilde{n}, \sigma, \mathcal{P})$. Then we have $B \xrightarrow{\tau} A \xrightarrow{\alpha} A'$.

3. Assume $B = \nu\widetilde{n}, m.(\sigma, \mathcal{P} \cup \{P\}) \simeq \nu\widetilde{n}.(\sigma, \mathcal{P} \cup \{\nu m.P\}) = A$ with $m \notin fn(\widetilde{n}, \sigma, \mathcal{P})$. If $A \xrightarrow{\alpha} A'$ is about pulling out name m, then $B = A'$. For the other cases of $A \xrightarrow{\alpha} A'$, we can easily see that A cannot perform any action from $\nu m.P$ and action α is from $\mathcal{P}$, thus there exists B' such that $B \xrightarrow{\alpha} B' \simeq^1 A'$.

4. Assume $B = \nu\widetilde{n}.(\sigma, \mathcal{P} \cup \{P \mid Q\}) \simeq \nu\widetilde{n}.(\sigma, \mathcal{P} \cup \{P\} \cup \{Q\}) = A$. Then we have $B \xrightarrow{\tau} A \xrightarrow{\alpha} A'$.

5. Assume $B = \nu\widetilde{n}.(\sigma, \mathcal{P} \cup \{P\} \cup \{Q\}) \simeq \nu\widetilde{n}.(\sigma, \mathcal{P} \cup \{P \mid Q\}) = A$. If $A \xrightarrow{\alpha} A'$ is about splitting $P \mid Q$, then $B = A'$. For the other cases of $A \xrightarrow{\alpha} A'$, we can easily see that A cannot perform any action from $P \mid Q$ and action α is from $\mathcal{P}$, thus there exists B' such that $B \xrightarrow{\alpha} B' \simeq^1 A'$.

6. When the $B \simeq^1 A$ replaces some terms, we take conditional branch as an example, the other cases are trivial. Assume $B = \nu\widetilde{n}.(\sigma\{M'/z\}, \mathcal{P}\{M'/z\} \cup \{\texttt{if}\ M\{M'/z\} = N\{M'/z\}\ \texttt{then}\ P\{M'/z\}\ \texttt{else}\ Q\{M'/z\}\}) \simeq \nu\widetilde{n}.(\sigma\{N'/z\}, \mathcal{P}\{N'/z\} \cup \{\texttt{if}\ M\{N'/z\} = N\{N'/z\}\ \texttt{then}\ P\{N'/z\}\ \texttt{else}\ Q\{N'/z\}\}) = A$ and $M' =_\Sigma N'$. Since $M\{M'/z\} =_\Sigma M\{N'/z\}$ and $N\{M'/z\} =_\Sigma N\{N'/z\}$, we can see that $M\{M'/z\} =_\Sigma N\{M'/z\}$ iff $M\{N'/z\} =_\Sigma N\{N'/z\}$. That is to say B and A will jump to the same branch. We take `then` branch as an example here. Then $A' = \nu\widetilde{n}.(\sigma\{N'/z\}, \mathcal{P}\{N'/z\} \cup P\{N'/z\})$. Let $B' = \nu\widetilde{n}.(\sigma\{M'/z\}, \mathcal{P}\{M'/z\} \cup P\{M'/z\})$. Clearly we have $B \xrightarrow{\tau} B' \simeq^1 A'$.

Given a set of cells $S = \{s_1 \mapsto M_1, \cdots, s_n \mapsto M_n\}$ and a set of locks L, we define the projection $S|_L$ of S under L to be the set $\{\, t \mapsto N \mid \{t \mapsto N\} \subseteq S \text{ and } t \in L \,\}$.

Lemma 16. Let A be a closed extended process and $fv(\alpha) \subseteq dom(A)$. If $A \xrightarrow{\alpha} B$ then $\lfloor A \rfloor \overset{\widehat{\alpha}}{\Longrightarrow} \lfloor B \rfloor$.

Proof. We only detail the proof for the transitions related to cell name here. The other cases are trivial. The function $\lfloor\rfloor$ only gathers together the name restrictions of the top level.

1. Assume $A = \nu\widetilde{n}.(\sigma, S, \mathcal{P} \cup \{(s \mapsto M, \emptyset)\}) \xrightarrow{\tau} B = \nu\widetilde{n}.(\sigma, S \cup \{s \mapsto M\}, \mathcal{P})$. Since A is closed, we have that $s \notin \mathit{locks}(\mathcal{P})$. We can easily see that $\lfloor A \rfloor = \lfloor B \rfloor$ from the definition of encoding in Section 4.

2. Assume $A = \nu\widetilde{n}.(\sigma, S \cup \{s \mapsto M\}, \mathcal{P} \cup \{(\texttt{read}\ s\ \texttt{as}\ x.P, L)\}) \xrightarrow{\tau} B = \nu\widetilde{n}.(\sigma, S \cup \{s \mapsto M\}, \mathcal{P} \cup \{(P\{M/x\}, L)\})$. Since this transition only affects cell s, we assume the encoding for the unlocked cells in S is $\mathcal{Q}_1$ and the encoding for $\mathcal{P}$ is $\mathcal{Q}_2$. We also assume the encoding for names $\widetilde{n}$ is $\widetilde{n}'$. The encoding for $\{s \mapsto M\}$ and $\texttt{read}\ s\ \texttt{as}\ x.P$ are different regarding s is locked or not.

(a) if $s \in L$, let $T = S|_L \cup \{s \mapsto M\}$, then $\lfloor A \rfloor = \nu\widetilde{n}'.(\sigma, \mathcal{Q}_1 \cup \mathcal{Q}_2 \cup \{\lfloor P\{M/x\}\rfloor_T\})$ and $\lfloor B \rfloor = \nu\widetilde{n}'.(\sigma, \mathcal{Q}_1 \cup \mathcal{Q}_2 \cup \{\lfloor P\{M/x\}\rfloor_T\})$. Thus we have $\lfloor A \rfloor \Longrightarrow \lfloor B \rfloor$.

(b) if $s \notin L$, then we have $\lfloor A \rfloor = \nu\widetilde{n}'.(\sigma, \mathcal{Q}_1 \cup \mathcal{Q}_2 \cup \left\{\overline{c_s}\langle M\rangle,\ c_s(x).(\overline{c_s}\langle x\rangle \mid \lfloor P\rfloor_{S|_L})\right\})$ and $\lfloor B \rfloor = \nu\widetilde{n}'.\left(\sigma, \mathcal{Q}_1 \cup \mathcal{Q}_2 \cup \left\{\overline{c_s}\langle M\rangle, \lfloor P\{M/x\}\rfloor_{S|_L}\right\}\right)$. Thus $\lfloor A \rfloor \xrightarrow{\tau} \nu\widetilde{n}'.(\sigma, \mathcal{Q}_1 \cup \mathcal{Q}_2 \cup \left\{\overline{c_s}\langle M\rangle \mid \lfloor P\{M/x\}\rfloor_{S|_L}\right\}) \xrightarrow{\tau} \lfloor B \rfloor$.

3. Assume $A = \nu\widetilde{n}.(\sigma, S \cup \{s \mapsto M\}, \mathcal{P} \cup \{(s := N.P, L)\}) \xrightarrow{\tau} B = \nu\widetilde{n}.(\sigma, S \cup \{s \mapsto N\}, \mathcal{P} \cup \{(P, L)\})$. Similar to the read case, we assume the encoding for $\widetilde{n}, S, \mathcal{P}$ are $\widetilde{n}', \mathcal{Q}_1, \mathcal{Q}_2$ respectively.

(a) if $s \in L$, then $\lfloor A \rfloor = \nu\widetilde{n}'.(\sigma, \mathcal{Q}_1 \cup \mathcal{Q}_2 \cup \left\{\lfloor P\rfloor_{S|_L \cup \{s \mapsto N\}}\right\})$ and $\lfloor B \rfloor = \nu\widetilde{n}'.(\sigma, \mathcal{Q}_1 \cup \mathcal{Q}_2 \cup \left\{\lfloor P\rfloor_{S|_L \cup \{s \mapsto N\}}\right\})$. This gives us $\lfloor A \rfloor \Longrightarrow \lfloor B \rfloor$.

(b) if $s \notin L$, then $\lfloor A \rfloor = \nu\widetilde{n}'.\left(\sigma, \mathcal{Q}_1 \cup \mathcal{Q}_2 \cup \left\{\overline{c_s}\langle M\rangle,\ c_s(x).(\overline{c_s}\langle N\rangle \mid \lfloor P\rfloor_{S|_L})\right\}\right)$ where x is a fresh base sort variable and $\lfloor B \rfloor = \nu\widetilde{n}'.\left(\sigma, \mathcal{Q}_1 \cup \mathcal{Q}_2 \cup \left\{\overline{c_s}\langle N\rangle, \lfloor P\rfloor_{S|_L}\right\}\right)$. Thus $\lfloor A \rfloor \xrightarrow{\tau} \nu\widetilde{n}'.\left(\sigma, \mathcal{Q}_1 \cup \mathcal{Q}_2 \cup \left\{\overline{c_s}\langle N\rangle \mid \lfloor P\rfloor_{S|_L}\right\}\right) \xrightarrow{\tau} \lfloor B \rfloor$.

4. Assume $A = \nu\widetilde{n}.(\sigma, S \cup \{s \mapsto M\}, \mathcal{P} \cup \{(\texttt{lock}\ s.P, L)\}) \xrightarrow{\tau} \nu\widetilde{n}.(\sigma, S \cup \{s \mapsto M\}, \mathcal{P} \cup \{(P, L \cup \{s\})\})$ and $s \notin L \cup \mathit{locks}(\mathcal{P})$. Similar to the read case, we assume the encoding for unlocked cells in S is $\mathcal{Q}_1$ and encoding for $\widetilde{n}, \mathcal{P}$ are $\widetilde{n}', \mathcal{Q}_2$ respectively. Then $\lfloor A \rfloor = \nu\widetilde{n}'.(\sigma, \mathcal{Q}_1 \cup \mathcal{Q}_2 \cup \left\{\overline{c_s}\langle M\rangle, c_s(x).\lfloor P\rfloor_{S|_L \cup \{s \mapsto x\}}\right\})$ and $\lfloor B \rfloor = \nu\widetilde{n}'.(\sigma, \mathcal{Q}_1 \cup \mathcal{Q}_2 \cup \left\{\lfloor P\rfloor_{S|_L \cup \{s \mapsto M\}}\right\})$. Since $x \notin \mathit{fv}(P)$, $\lfloor P\rfloor_{S|_L \cup \{s \mapsto x\}}\{M/x\} = \lfloor P\rfloor_{S|_L \cup \{s \mapsto M\}}$. Thus we have $\lfloor A \rfloor \xrightarrow{\tau} \lfloor B \rfloor$.

5. Assume $A = \nu\widetilde{n}.(\sigma, S \cup \{s \mapsto M\}, \mathcal{P} \cup \{(\texttt{unlock}\ s.P, L)\}) \xrightarrow{\tau} B = \nu\widetilde{n}.(\sigma, S \cup \{s \mapsto M\}, \mathcal{P} \cup \{(P, L \setminus \{s\})\})$ and $s \in L$. We assume the encoding for $\widetilde{n}, S, \mathcal{P}$ are $\widetilde{n}', \mathcal{Q}_1, \mathcal{Q}_2$ respectively. Then $\lfloor A \rfloor = \nu\widetilde{n}'.\left(\sigma, \mathcal{Q}_1 \cup \mathcal{Q}_2 \cup \left\{\overline{c_s}\langle M\rangle \mid \lfloor P\rfloor_{S|_L}\right\}\right)$ and $\lfloor B \rfloor = \nu\widetilde{n}'.\left(\sigma, \mathcal{Q}_1 \cup \mathcal{Q}_2 \cup \left\{\overline{c_s}\langle M\rangle, \lfloor P\rfloor_{S|_L}\right\}\right)$. Thus $\lfloor A \rfloor \xrightarrow{\tau} \lfloor B \rfloor$.

Appendix C. Proofs of Theorem 21 and Corollary 22 in Section 5

In this section, we discuss the relation between applied pi calculus and stateful applied pi calculus.

To fix the flaw mentioned in Section 3.1, *we revise the original applied pi calculus [3] slightly that the active substitutions are only defined on terms of base sort.* Since the communication rule in [3] relies on the active substitutions, we need to replace it with the new rule COMM $\overline{a}\langle M\rangle.P_r \mid a(x).Q_r \xrightarrow{\tau} P_r \mid Q_r\{M/x\}$ accordingly.

To avoid confusion, we use A_r, B_r, C_r to refer to the extended processes and use $\mathcal{C}_r$ to refer to the evaluation context in applied pi calculus.

Appendix C.1. An Alternative Semantics for Applied Pi Calculus

To ease the proof, we use an alternative semantics in Figure C.6 of the revised applied pi calculus mentioned above. This semantics has been proved in Appendix A in [32] to yield exactly the same set of observational equivalence (resp. labelled bisimilarity) as the one in [3]. For convenience of reading, we copy the proof in [32] here.

The operational semantics of the applied pi calculus relies heavily on structural equivalence. This is because the analysis of complex data and "alias" mechanism introduced in the calculus depends on structural equivalence rules such as SUBST and REWRITE. Unfortunately such a structural equivalence makes the formal reasoning very difficult. Thus, as a first step, we need to preprocess the original semantics in [3] and rewrite it to a more convenient form while preserving the observational equivalence.

Here in Figure C.6 we replace the two-directional rule $!P_r \equiv P_r \mid !P_r$ in structural equivalence in [3] with the one-directional $!P_r \xrightarrow{\tau} P_r \mid !P_r$ in the internal reduction, as well as replacing the THEN in internal reduction in [3] with $\mathit{if}\ M = N\ \mathit{then}\ P_r\ \mathit{else}\ Q_r \xrightarrow{\tau} P_r$ if $M =_\Sigma N$.

We shall show that the notions of the observational equivalence and the labelled bisimilarity generated by the two sets of rules are exactly the same (Theorem 50 andTheorem 51). In other words, it is adequate to handle replications with $!P_r \xrightarrow{\tau} P_r \mid !P_r$ only.

The observational equivalence and labelled bisimilarity in applied pi calculus are defined by:

Definition 43. *Observational equivalence ($\approx$) is the largest symmetric relation $\mathcal{R}$ between closed extended processes with the same domain such that $A_r \,\mathcal{R}\, B_r$ implies:*

1. *if $A_r \Downarrow_a$ then $B_r \Downarrow_a$;*
2. *if $A_r \Longrightarrow A_r'$, then $B_r \Longrightarrow B_r'$ and $A_r' \,\mathcal{R}\, B_r'$ for some B_r';*
3. *$\mathcal{C}_r[A_r] \,\mathcal{R}\, \mathcal{C}_r[B_r]$ for all closing evaluation contexts $\mathcal{C}_r$.*

Definition 44. *Two terms M and N are* equal in the frame ϕ*, written $(M = N)\phi$, iff $\phi \equiv \nu\widetilde{n}.\sigma$, $\{\widetilde{n}\} \cap name(M, N) = \emptyset$, and $M\sigma =_\Sigma N\sigma$, for some names $\widetilde{n}$ and substitution σ.*

$$
\begin{array}{rcl}
A_r & \equiv & A_r \mid 0 \\
A_r \mid B_r & \equiv & B_r \mid A_r \\
A_r \mid (B_r \mid C_r) & \equiv & (A_r \mid B_r) \mid C_r
\end{array}
$$

$$
\begin{array}{rcl}
\nu x. \{M/x\} & \equiv & 0 \\
\{M/x\} & \equiv & \{N/x\} \;\text{ when } M =_\Sigma N \\
\{M/x\} \mid A_r & \equiv & \{M/x\} \mid A_r \{M/x\}
\end{array}
\qquad
\begin{array}{rcl}
\nu n.0 & \equiv & 0 \\
\nu u.\nu v.A_r & \equiv & \nu v.\nu u.A_r \\
A_r \mid \nu u.B_r & \equiv & \nu u.(A_r \mid B_r) \\
& & \text{when } u \notin \mathit{fnv}(A_r)
\end{array}
$$

$$\textsc{Comm} \quad \overline{a}\langle M\rangle.P_r \mid a(x).Q_r \xrightarrow{\tau} P_r \mid Q_r \{M/x\}$$

$$\textsc{Then} \quad \mathit{if}\ M = N\ \mathit{then}\ P_r\ \mathit{else}\ Q_r \xrightarrow{\tau} P_r \quad \text{if } M =_\Sigma N$$

$$\textsc{Else} \quad \mathit{if}\ M = N\ \mathit{then}\ P_r\ \mathit{else}\ Q_r \xrightarrow{\tau} Q_r \quad \text{if } \mathit{var}(M,N) = \emptyset \text{ and } M \neq_\Sigma N$$

$$\textsc{Rep} \quad !P_r \xrightarrow{\tau} P_r \mid !P_r \qquad \textsc{In} \quad a(x).P_r \xrightarrow{a(M)} P_r \{M/x\}$$

$$\textsc{OutCh} \quad \overline{a}\langle c\rangle.P_r \xrightarrow{\overline{a}\langle c\rangle} P_r \qquad \textsc{OutT} \quad \overline{a}\langle M\rangle.P_r \xrightarrow{\nu x.\overline{a}\langle x\rangle} P_r \mid \{M/x\}$$

where $x \in \mathcal{V}_b$ and $x \notin \mathit{fv}(\overline{a}\langle M\rangle.P_r)$

$$\textsc{OpenCh} \quad \frac{A_r \xrightarrow{\overline{a}\langle c\rangle} B_r \quad a \neq c}{\nu c.A_r \xrightarrow{\nu c.\overline{a}\langle c\rangle} B_r} \qquad \textsc{Scope} \quad \frac{A_r \xrightarrow{\alpha} B_r \quad u \text{ does not occur in } \alpha}{\nu u.A_r \xrightarrow{\alpha} \nu u.B_r}$$

$$\textsc{Par} \quad \frac{A_r \xrightarrow{\alpha} A'_r,\ \mathit{bnv}(\alpha) \cap \mathit{fnv}(B_r) = \emptyset}{A_r \mid B_r \xrightarrow{\alpha} A'_r \mid B_r}$$

$$\textsc{Struct} \quad \frac{A_r \equiv C_r \xrightarrow{\alpha} D_r \equiv B_r}{A_r \xrightarrow{\alpha} B_r}$$

Figure C.6: Operational Semantics of Applied Pi

Two closed frames ϕ_1 and ϕ_2 are statically equivalent, written $\phi_1 \approx_s \phi_2$, if $dom(\phi_1) = dom(\phi_2)$, and for all terms M and N such that $var(M, N) \subseteq dom(\phi_1)$ we have $(M = N)\phi_1$ iff $(M = N)\phi_2$. Two closed extended processes A_r and B_r are statically equivalent, written $A_r \approx_s B_r$, if their frames are.

Definition 45. *Labeled bisimilarity ($\approx_l$) is the largest symmetric relation $\mathcal{R}$ on closed extended processes such that $A_r \,\mathcal{R}\, B_r$ implies:*

1. $A_r \approx_s B_r$
2. *if $A_r \xrightarrow{\alpha} A'_r$ and $fv(\alpha) \subseteq dom(A_r)$ and $bn(\alpha) \cap fn(B_r) = \emptyset$ then $B_r \overset{\widehat{\alpha}}{\Longrightarrow} B'_r$ and $A'_r \,\mathcal{R}\, B'_r$ for some B'_r.*

In order to avoid confusion, in the following discussions we shall use $\equiv_o$, $\xrightarrow{\tau}_o$, $\overset{\tau}{\Longrightarrow}_o$, $\Downarrow^o_a$, $\approx_o$ and $\approx_{l,o}$ to refer to original structural equivalence, (strong and weak) transitions, etc defined in [3]; and use $\equiv$, $\xrightarrow{\tau}$, $\Longrightarrow$, $\Downarrow_a$, $\approx$ and $\approx_l$ for the corresponding ones generated here. To prove that $\approx_o$(resp. $\approx_{l,o}$) coincides with $\approx$(resp. $\approx_l$). We need to explore the relationship between $\xrightarrow{\alpha}_o$ and $\xrightarrow{\alpha}$. Their relations are mainly formalised in the following Lemma 47 and Lemma 48.

We write $A_r \succ^1 B_r$ if A_r can be transformed to B_r by applying to a subterm (which is not under a replication, an input, a conditional, or an output) of A_r an axiom of structural equivalence $\equiv_o$, except that $!P_r \equiv_o P_r \;|!P_r$ can only be used from left to right; we write $\succ$ for the reflexive and transitive closure of $\succ^1$. We say a sequence $A^1_r \succ^1 A^2_r \succ^1 \cdots \succ^1 A^\ell_r$ is a *linear proof sequence* of $A^1_r \succ A^\ell_r$.

Since the use of evaluation context before the use of structural equivalence can be swapped. Two applications of structural equivalence as well as evaluation contexts can be condensed to one, we can always obtain a derivation for any transition in which the use of structural equivalence occurs only once and at the last step. We shall call such a derivation a *normalised derivation*.

For $n \geq 1$, an *n-hole evaluation context* $\mathcal{C}_r$ is an extended process with n holes which are not under a replication, an input, an output or a conditional. We write $\mathcal{C}_r[A^1_r, A^2_r, \cdots, A^n_r]$ for the extended process obtained by filling the holes with processes.

Lemma 46. *Assume $A_r \succ B_r$ and $A_r = \mathcal{C}_r[!P_r]$ with $\mathcal{C}_r$ an evaluation context. Then there exist an evaluation context $\mathcal{C}'_r$ and a plain process Q_r such that $B_r = \mathcal{C}'_r[!Q_r]$ and $\mathcal{C}_r[P_r \;|!P_r] \succ \mathcal{C}'_r[Q_r \;|!Q_r]$.*

Proof. By induction on the length of the linear proof sequence for $\succ$. If the length is 0, the result holds immediately. Now assume $A_r \succ^1 A^1_r \succ^1 A^2_r \cdots \succ^1 A^\ell_r \succ^1 A^{\ell+1}_r = B_r$. By the induction hypothesis there exist a plain process R_r and an evaluation context $\mathcal{C}''_r$ such that

$$A^\ell_r = \mathcal{C}''_r[!R_r] \qquad \mathcal{C}_r[P_r \;|!P_r] \succ \mathcal{C}''_r[R_r \;|!R_r] \tag{C.1}$$

We argue by case analysis on the axiom used in deriving $A^\ell_r \succ^1 A^{\ell+1}_r$. We give the details only for two cases when $\succ^1$ is REWRITE and SUBST. The other cases are similar.

1. $A_o^\ell = \mathcal{C}_r'''[\{M/x\}] \succ^1 \mathcal{C}_r'''[\{N/x\}] = A_r^{\ell+1}$ with $M =_\Sigma N$. Since there is no way that active substitution $\{M/x\}$ can occur inside replications, it is easy to see that there exists a two-hole evaluation context D such that $A_r^\ell = D[!R_r, \{M/x\}]$, $D[!R_r, \cdot] = \mathcal{C}_r'''$ and $D[\cdot, \{M/x\}] = \mathcal{C}_r''$. Using the REWRITE axiom, we know that $D[R_r \mid !R_r, \{M/x\}] \succ^1 D[R_r \mid !R_r, \{N/x\}]$. Let $\mathcal{C}_r' = D[\cdot, \{N/x\}]$ and $Q_r = R_r$. Clearly $A_r^{\ell+1} = \mathcal{C}_r'[!Q_r]$. Hence $\mathcal{C}_r[P_r \mid !P_r] \succ \mathcal{C}_r''[R_r \mid !R_r] \succ^1 \mathcal{C}_r'[Q_r \mid !Q_r]$ and the result holds.

2. (a) $A_r^\ell = \mathcal{C}_r'''[E_r \mid \{M/x\}] \succ^1 \mathcal{C}_r'''[E_r \{M/x\} \mid \{M/x\}] = A_r^{\ell+1}$. Since the hole in any evaluation context has no chance to occur under any replication, $!R_r$ in (C.1) should occur in either E_r or $\mathcal{C}_r'''$. The analysis for the latter case is similar as the above case. Now we consider the former case. Here there exists an evaluation context D such that $E_r = D[!R_r]$ and $\mathcal{C}_r'''[D[\cdot] \mid \{M/x\}] = \mathcal{C}_r''$. The substitution $\{M/x\}$ will apply to D and R_r while rewriting A_r^ℓ to $A_r^{\ell+1}$. Let $D' = D\{M/x\}$ and $Q_r = R_r\{M/x\}$. We can easily see that $A_r^{\ell+1} = \mathcal{C}_r'''[D'[!Q_r] \mid \{M/x\}]$ and $\mathcal{C}_r'''[D[R_r \mid !R_r] \mid \{M/x\}] \succ^1 \mathcal{C}_r'''[D'[Q_r \mid !Q_r] \mid \{M/x\}]$. Let $\mathcal{C}_r' = \mathcal{C}_r'''[D'[\cdot] \mid \{M/x\}]$. Then $A_r^{\ell+1} = \mathcal{C}_r'[!Q_r]$ and $\mathcal{C}[P_r \mid !P_r] \succ \mathcal{C}_r''[R_r \mid !R_r] \succ \mathcal{C}_r'[Q_r \mid !Q_r]$.

(b) $A_r^\ell = \mathcal{C}_r'''[E_r\{M/x\} \mid \{M/x\}] \succ^1 \mathcal{C}_r'''[E_r \mid \{M/x\}] = A_r^{\ell+1}$. When $!R_r$ in (C.1) occurs in $E_r\{M/x\}$, clearly there exist an evaluation context D and a plain process Q_r such that $E_r = D[!Q_r]$ and $Q_r\{M/x\} = R_r$. The rest is similar to the above case.

3. $A_r^\ell = \mathcal{C}_r'''[!P_r'] \succ^1 \mathcal{C}_r'''[P_r' \mid !P_r'] = A_r^{\ell+1}$. When $!P_r'$ is $!R_r$ in (C.1), the result holds trivially; otherwise $!R_r$ in (C.1) should occur in $\mathcal{C}_r'''$ and the remaining analysis is similar.

Lemma 47. *Assume $A_r \xrightarrow{\alpha}_o A_r'$ where A_r, A_r' are closed and α is not $\overline{a}\langle x\rangle$ and $fv(\alpha) \subseteq dom(A_r)$. Then there exist closed B_r, B_r' such that $A_r \succ B_r \xrightarrow{\alpha} B_r' \equiv_o A_r'$.*

Proof. Consider the normalized derivation of transition $A_r \xrightarrow{\alpha}_o A_r'$.

1. α is $a(M)$. Then $A_r \equiv_o \mathcal{C}_r[a(x).Q_r] \xrightarrow{a(M)}_o \mathcal{C}_r[Q_r\{M/x\}] \equiv_o A_r'$ with $\mathcal{C}_r$ an evaluation context and $\mathcal{C}_r[a(x).Q_r] \xrightarrow{a(M)}_o \mathcal{C}_r[Q_r\{M/x\}]$ derived by the rules in [3] without using $\equiv_o$.

We may assume $\mathcal{C}_r[a(x).Q_r]$ and $\mathcal{C}_r[Q_r\{M/x\}]$ are both closed; for otherwise we can let $fv(\mathcal{C}_r[a(x).Q_r]) - dom(\mathcal{C}_r[a(x).Q_r]) = \{x_1, \cdots, x_n\}$ and choose n fresh names $c_1, \cdots, c_n$ and let $\sigma = \{c_1/x_1, \cdots, c_n/x_n\}$. From the hypothesis, we know that $M\sigma = M, x \notin var(\sigma)$, and $dom(A_r) = dom(\mathcal{C}_r[a(x).P_r]) = dom(A_r')$. It is easy to see that $A_r = A_r\sigma \equiv_o \mathcal{C}_r\sigma[a(x).Q_r\sigma] \xrightarrow{a(M)}_o \mathcal{C}_r\sigma[(Q_r\sigma)\{M/x\}] = \mathcal{C}_r\sigma[(Q_r\sigma)\{M\sigma/x\}] \equiv_o A_r'\sigma = A_r'$.

Since $\mathcal{C}_r[a(x).Q_r] \xrightarrow{a(M)}_o \mathcal{C}_r[Q_r\{M/x\}]$ can be derived without using $\equiv_o$, $\mathcal{C}_r[a(x).Q_r] \xrightarrow{a(M)} \mathcal{C}_r[Q_r\{M/x\}]$ can also be derived by rules in Fig. C.6 without using $\equiv$. Thus $A_r \equiv_o \mathcal{C}_r[a(x).Q_r] \xrightarrow{a(M)} \mathcal{C}_r[Q_r\{M/x\}] \equiv_o A_r'$. Now we proceed to construct the required B_r and B'_r as stated in the lemma. The rest of the proof goes by induction on the number of applications of $!P_r \equiv_o P_r \mid !P_r$ from right to left in deriving $A_r \equiv_o \mathcal{C}_r[a(x).Q_r]$. If the number is 0, the result is immediate. So suppose the number is nonzero and consider the last application of $!P_r \equiv_o P_r \mid !P_r$ from right to left (we write $\equiv_o^1$ for the application of an axiom of structural equivalence $\equiv_o$):

$$A_r \equiv_o \mathcal{C}'_r[P_r \mid !P_r] \equiv_o^1 \mathcal{C}'_r[!P_r] \succ \mathcal{C}_r[a(x).Q_r]$$

where $\mathcal{C}'_r$ is also an evaluation context. From Lemma A.1, we know there exists D' such that $\mathcal{C}'_r[P_r \mid !P_r] \succ D'[R_r \mid !R_r]$ and $D'[!R_r] = \mathcal{C}_r[a(x).Q_r]$. Then there exists a two hole evaluation context D such that $D[!R_r, \cdot] = \mathcal{C}_r$ since $a(x).Q_r$ cannot occur inside the replication. Moreover $D[R_r \mid !R_r, a(x).Q_r] \xrightarrow{a(M)} D[R_r \mid !R_r, Q_r\{M/x\}]$ can be derived by the rules in Fig. C.6, and

$$\begin{aligned} A_r \equiv_o \mathcal{C}'_r[P_r \mid !P_r] \succ D[R_r \mid !R_r, a(x).Q_r] &\xrightarrow{a(M)} \\ D[R_r \mid !R_r, Q_r\{M/x\}] &\equiv_o \mathcal{C}_r[Q_r\{M/x\}] \equiv_o A'_r \end{aligned}$$

Replacing $!R_r$ with $R_r \mid !R_r$ does not introduce fresh variables. In other words $D[R_r \mid !R_r, a(x).Q_r]$ and $D[R_r \mid !R_r, Q_r\{M/x\}]$ are also closed. By induction hypothesis, there exist closed B_r, B'_r such that $A_r \succ B_r \xrightarrow{a(M)} B'_r \equiv_o A'_r$.

2. α is $\overline{a}\langle c\rangle$. Then $A_r \equiv_o \mathcal{C}_r[\overline{a}\langle c\rangle.Q_r] \xrightarrow{\overline{a}\langle c\rangle}_o \mathcal{C}_r[Q_r] \equiv_o A'_r$ with $\mathcal{C}_r$ an evaluation context. Clearly $\mathcal{C}_r[\overline{a}\langle c\rangle.Q_r] \xrightarrow{\overline{a}\langle c\rangle} \mathcal{C}_r[Q_r]$. The rest of the proof is similar to the above case.

3. α is $\nu c.\overline{a}\langle c\rangle$. Then $A_r \equiv_o \nu c.\mathcal{C}_r[\overline{a}\langle c\rangle.Q_r] \xrightarrow{\nu c.\overline{a}\langle c\rangle}_o \mathcal{C}_r[Q_r] \equiv_o A'_r$ with $\mathcal{C}_r$ an evaluation context. Then we have $\nu c.\mathcal{C}_r[\overline{a}\langle c\rangle.Q_r] \xrightarrow{\nu c.\overline{a}\langle c\rangle} \mathcal{C}_r[Q_r]$. The rest of the proof is similar.

4. α is $\nu x.\overline{a}\langle x\rangle$. Then $A_r \equiv_o \nu x.\mathcal{C}_r[\overline{a}\langle x\rangle.Q_r] \xrightarrow{\nu x.\overline{a}\langle x\rangle}_o \mathcal{C}_r[Q_r] \equiv_o A'_r$ with $\mathcal{C}_r$ an evaluation context. By the side-condition on extended process in Section 2.1, there is exactly one $\{M/x\}$ in $\mathcal{C}_r$ for the restricted variable x. Thus there exists a two-hole evaluation context D such that $\mathcal{C}_r = D[\{M/x\}, \cdot]$. Since the side-condition for rule OUTT in Fig. C.6 requires x be fresh, we choose a fresh variable y and let $\varrho = \{y/x\}$. By α-conversion, and structural equivalence $\equiv$, we can deduce that

$$\begin{aligned} \nu x.\mathcal{C}_r[\overline{a}\langle x\rangle.Q_r] = \nu x.D[\{M/x\}, \overline{a}\langle x\rangle.Q_r] &= \nu y.\varrho(D)[\{M/y\}, \overline{a}\langle y\rangle.\varrho(Q_r)] \\ &\xrightarrow{\nu x.\overline{a}\langle x\rangle} \nu y.\varrho(D)[\{M/y\}, \varrho(Q_r) \mid \{y/x\}] \\ &\equiv \nu y.D[\{M/y\}, Q_r \mid \{y/x\}] \equiv \nu y.D[\{M/y\} \mid \{y/x\}, Q_r] \\ &\equiv \nu y.D[\{M/y\} \mid \{M/x\}, Q_r] \equiv D[\nu y.\{M/y\} \mid \{M/x\}, Q_r] \\ &\equiv D[\{M/x\}, Q_r] = \mathcal{C}_r[Q_r] \equiv_o A'_r \end{aligned}$$

5. α is τ. There are three cases:

(a) $A_r \equiv_o \mathcal{C}_r[\ \mathit{if}\ M = M\ \mathit{then}\ P_r\ \mathit{else}\ Q_r] \xrightarrow{\tau}_o \mathcal{C}_r[P_r] \equiv_o A'_r$ with $\mathcal{C}_r$ an evaluation context.

(b) $A_r \equiv_o \mathcal{C}_r[\ \mathit{if}\ M = N\ \mathit{then}\ P_r\ \mathit{else}\ Q_r] \xrightarrow{\tau}_o \mathcal{C}_r[Q_r] \equiv_o A'_r$ with $M \neq_\Sigma N$, M, N are ground terms and $\mathcal{C}_r$ an evaluation context.

(c) $A_r \equiv_o \mathcal{C}_r[\overline{a}\langle M\rangle.P_r \mid a(x).Q_r] \xrightarrow{\tau}_o \mathcal{C}_r[P_r \mid Q_r\{M/x\}] \equiv_o A'_r$ with $\mathcal{C}_r$ an evaluation context.

The rest of the proof is similar.

Lemma 48. *Assume α is not $\overline{a}\langle x\rangle$ and A_r, A'_r are closed.*

1. If $A_r \xrightarrow{\alpha}_o A'_r$ then there is a closed A''_r such that $A_r \stackrel{\alpha}{\Longrightarrow} A''_r \equiv_o A'_r$.

2. If $A_r \xrightarrow{\alpha} A'_r$ then either $A_r \equiv_o A'_r$(only possible when α is τ) or $A_r \xrightarrow{\alpha}_o A'_r$.

Proof.

1. Assume $A_r \xrightarrow{\alpha}_o A'_r$. By Lemma 47, there exist closed B_r and B'_r such that $A_r \succ B_r \xrightarrow{\alpha} B'_r \equiv_o A'_r$. Replacing every left to right application of the rule $!P_r \equiv_o P_r \mid !P_r$ in $A_r \succ B_r$ with $!P_r \xrightarrow{\tau} P_r \mid !P_r$, we obtain $A_r \Rightarrow B_r \xrightarrow{\alpha} B'_r \equiv_o A'_r$. Letting $A''_r = B'_r$ gives the conclusion.

2. Assume $A_r \xrightarrow{\alpha} A'_r$ and apply transition induction.

(a) α is $a(M)$. Then $A_r \equiv \mathcal{C}_r[a(x).P] \xrightarrow{a(M)} \mathcal{C}_r[P\{M/x\}] \equiv A'_r$ where $\mathcal{C}_r$ is an evaluation context. Clearly we have $A_r \equiv \mathcal{C}_r[a(x).P] \xrightarrow{a(M)}_o \mathcal{C}_r[P\{M/x\}] \equiv A'_r$. Since $\equiv$ is included in $\equiv_o$, we have $A_r \xrightarrow{a(M)}_o A'_r$.

(b) The cases for α is τ are similar. For replications, assume $A_r \equiv \mathcal{C}_r[!P_r] \xrightarrow{\tau} \mathcal{C}_r[P_r \mid !P_r] \equiv A'_r$, then we have $A_r \equiv_o A'_r$.

(c) α is $\nu x.\overline{a}\langle x\rangle$. We have $A_r \equiv \mathcal{C}_r[\overline{a}\langle M\rangle.P] \xrightarrow{\nu x.\overline{a}\langle x\rangle} \mathcal{C}_r[P \mid \{M/x\}] \equiv A'_r$. Then we know that $A_r \equiv \nu x.\mathcal{C}_r[\overline{a}\langle x\rangle.P \mid \{M/x\}] \xrightarrow{\nu x.\overline{a}\langle x\rangle}_o \mathcal{C}_r[P \mid \{M/x\}] \equiv A'_r$.

(d) α is $\overline{a}\langle c\rangle$. We have $A_r \equiv \mathcal{C}_r[\overline{a}\langle c\rangle.P] \xrightarrow{\overline{a}\langle c\rangle} \mathcal{C}_r[P] \equiv A'_r$. Then we know that $A_r \xrightarrow{\overline{a}\langle c\rangle}_o A'_r$.

(e) α is $\nu c.\overline{a}\langle c\rangle$. We have $A_r \equiv \nu c.\mathcal{C}_r[\overline{a}\langle c\rangle.P] \xrightarrow{\nu c.\overline{a}\langle c\rangle} \mathcal{C}_r[P] \equiv A'_r$. Then we know that $A_r \xrightarrow{\nu c.\overline{a}\langle c\rangle}_o A'_r$.

Corollary 49. *Assume α is not $\overline{a}\langle x\rangle$ and A_r, A'_r are closed.*

1. If $A_r \stackrel{\alpha}{\Longrightarrow}_o A'_r$ then there is a closed A''_r such that $A_r \stackrel{\alpha}{\Longrightarrow} A''_r \equiv_o A'_r$.

2. If $A_r \stackrel{\alpha}{\Longrightarrow} A'_r$ then either $A_r \equiv_o A'_r$(only possible when α is τ) or $A_r \stackrel{\alpha}{\Longrightarrow}_o A'_r$.

Proof. Using Lemma 48 several times.

Theorem 50. *$\approx_o$ coincides with $\approx$.*

Proof.

1. ($\Longrightarrow$) We construct a set $\mathbb{S}$ of pairs of closed extended processes such that

$$\mathbb{S} = \{ (A_r, B_r) \mid A_r \equiv_o \approx_o \equiv_o B_r \}$$

and show $\mathbb{S} \subseteq \approx$. Assume $(A_r, B_r) \in \mathbb{S}$ because of $A_r \equiv_o D_{1,r} \approx_o D_{2,r} \equiv_o B_r$ for some closed extended processes $D_{1,r}$ and $D_{2,r}$.

(a) Assume $A_r \Longrightarrow A'_r$. Using Corollary 49, we have $A_r \Longrightarrow_o A'_r$ or $A_r \equiv_o A'_r$. When $A_r \Longrightarrow_o A'_r$, we have $D_{1,r} \Longrightarrow_o A'_r$. By the definition of $\approx_o$, there exists $D'_{2,r}$ such that $D_{2,r} \Longrightarrow_o D'_{2,r} \approx_o A'_r$. Using Corollary 49 again gives a B'_r such that $B_r \Longrightarrow B'_r \equiv_o D'_{2,r}$. Hence $(A'_r, B'_r) \in \mathbb{S}$. When $A_r \equiv_o A'_r$, let $B'_r = B_r$. Then $B_r \Longrightarrow B'_r$ and $A'_r \equiv_o A_r \equiv_o \approx_o \equiv_o B_r = B'_r$. Hence $(A'_r, B'_r) \in \mathbb{S}$.

(b) If $A_r \Downarrow_a$, then by Corollary 49, we have $A_r \Downarrow^o_a$. From $D_{1,r} \equiv_o A_r$, we have $D_{1,r} \Downarrow^o_a$. From $D_{1,r} \approx_o D_{2,r}$, we have $D_{2,r} \Downarrow^o_a$. From $D_{2,r} \equiv_o B_r$, we have $B_r \Downarrow^o_a$. Using Corollary 49 again, we have $B_r \Downarrow_a$.

(c) Since $\equiv_o$ and $\approx_o$ are both closed by evaluation contexts, we have $\mathcal{C}_r[A_r] \equiv_o \mathcal{C}_r[D_{1,r}] \approx_o \mathcal{C}_r[D_{2,r}] \equiv_o \mathcal{C}_r[B_r]$, namely $(\mathcal{C}_r[A_r], \mathcal{C}_r[B_r]) \in \mathbb{S}$ for any evaluation context $\mathcal{C}_r$.

2. ($\Longleftarrow$) We construct a set $\mathbb{R}$ of pairs of closed extended processes such that

$$\mathbb{R} = \{ (A_r, B_r) \mid A_r \equiv_o \approx \equiv_o B_r \}$$

and show that $\mathbb{R} \subseteq \approx_o$. Assume $(A_r, B_r) \in \mathbb{S}$ because of $A_r \equiv_o D_{1,r} \approx D_{2,r} \equiv_o B_r$ for some closed extended processes $D_{1,r}$ and $D_{2,r}$.

(a) Assume $A_r \stackrel{\tau}{\Longrightarrow}_o A'_r$. Then we have $D_{1,r} \Longrightarrow_o A'_r$. Using Corollary 49, there exists $D'_{1,r}$ such that $D_{1,r} \Longrightarrow D'_{1,r} \equiv_o A'_r$. By the definition of $\approx$, there exists $D'_{2,r}$ such that $D_{2,r} \Longrightarrow D'_{2,r} \approx D'_{1,r}$. Using Corollary 49, it gives $D_{2,r} \Longrightarrow_o D'_{2,r}$ or $D_{2,r} \equiv_o D'_{2,r}$. Since $B_r \equiv_o D_{2,r}$, we have $B_r \Longrightarrow_o D'_{2,r}$ or $B_r \equiv_o D'_{2,r}$. In the former case, let $B'_r = D'_{2,r}$ and in the latter case let $B'_r = B_r$. We have $(A'_r, B'_r) \in \mathbb{R}$.

(b) If $A_r \Downarrow^o_a$, then $D_{1,r} \Downarrow^o_a$. Then by Corollary 49, we have $D_{1,r} \Downarrow_a$. From $D_{1,r} \approx D_{2,r}$, we have $D_{2,r} \Downarrow_a$. Using Corollary 49 again, we have $D_{2,r} \Downarrow^o_a$. From $D_{2,r} \equiv_o B_r$, we have $B_r \Downarrow^o_a$.

(c) Since $\equiv_o$ and $\approx$ are both closed by evaluation contexts, we have $\mathcal{C}_r[A_r] \equiv_o \mathcal{C}_r[D_{1,r}] \approx \mathcal{C}_r[D_{2,r}] \equiv_o \mathcal{C}_r[B_r]$, namely $(\mathcal{C}_r[A_r], \mathcal{C}_r[B_r]) \in \mathbb{R}$ for any evaluation context $\mathcal{C}_r$.

Theorem 51. *$\approx_{l,o}$ coincides with $\approx_l$.*

Proof.

1. ($\Longrightarrow$) We construct the set $\mathbb{S}$ of pairs of closed extended processes such that

$$\mathbb{S} = \{ (A_r, B_r) \mid A_r \equiv_o \approx_{l,o} \equiv_o B_r \}$$

and show $\mathbb{S} \subseteq \approx_l$. Assume $(A_r, B_r) \in \mathbb{S}$ because of $A_r \equiv_o C_r \approx_{l,o} D_r \equiv_o B_r$ for some closed extended processes C_r and D_r. For the static equivalence part, although $\equiv_o$ has the rule REPL while $\equiv$ does not, the rewriting $\mathcal{C}[!P_r] \equiv_o \mathcal{C}[P_r \mid !P_r]$ does not change the frames of processes, i.e. $\phi(\mathcal{C}[!P_r]) = \phi(\mathcal{C}[P_r \mid !P_r])$. Thus $\phi(C_r) \equiv_o \nu\widetilde{n}.\sigma$ implies $\phi(A_r) \equiv \nu\widetilde{n}.\sigma$, and similarly $\phi(D_r) \equiv_o \nu\widetilde{m}.\sigma'$ implies $\phi(B_r) \equiv \nu\widetilde{m}.\sigma'$. Hence $A_r \sim B_r$ holds by the definition of $\sim$.

Now assume $A_r \xrightarrow{\alpha} A'_r$ with $fv(\alpha) \subseteq dom(A_r)$ and $bn(\alpha) \cap fn(B_r) = \emptyset$. By Lemma 48, we have $A_r \xrightarrow{\alpha}_o A'_r$ or $A_r \equiv_o A'_r$.

When $A_r \xrightarrow{\alpha}_o A'_r$, we have $C_r \xrightarrow{\alpha}_o A'_r$. By the definition of $\approx_{l,o}$, there exists D'_r such that $D_r \overset{\widehat{\alpha}}{\Longrightarrow}_o D'_r \approx_{l,o} A'_r$. By Corollary 49, there exists B'_r such that $B_r \overset{\widehat{\alpha}}{\Longrightarrow} B'_r \equiv_o D'_r$. Hence $(A'_r, B'_r) \in \mathbb{S}$.

When $A_r \equiv_o A'_r$, from the proof of Lemma 48, we can know that this could happen only when α is τ. In this case, let $B'_r = B_r$. Then $B_r \Longrightarrow B'_r$ and $A'_r \equiv_o A_r \equiv_o \approx_{l,o} \equiv_o B_r = B'_r$. Hence $(A'_r, B'_r) \in \mathbb{S}$.

2. ($\Longleftarrow$) We construct the set $\mathbb{R}$ of pairs of closed extended processes such that

$$\begin{aligned}\mathbb{R} = \{ (A_r, B_r) \mid \exists \{\widetilde{z}\} \subseteq dom(A_r) : A_r \mid \{\widetilde{z}/\widetilde{y}\} \equiv_o \approx_l \equiv_o B_r \mid \{\widetilde{z}/\widetilde{y}\} \\ \text{for any pairwise-distinct } \widetilde{y} \text{ s.t. } \{\widetilde{y}\} \cap dom(A_r) = \emptyset \text{ and } |\,\widetilde{y}\,| = |\,\widetilde{z}\,| \}\end{aligned}$$

and show that $\mathbb{R} \subseteq \approx_{l,r}$. Note that when $A_r \approx_l B_r$, $\{\widetilde{z}\}$ is chosen to be empty. Assume $(A_r, B_r) \in \mathbb{R}$. Then there exist closed extended processes C_r, D_r and variables $\widetilde{z}$ such that $A_r \mid \{\widetilde{z}/\widetilde{y}\} \equiv_o C_r \approx_l D_r \equiv_o B_r \mid \{\widetilde{z}/\widetilde{y}\}$ for any pairwise-distinct $\widetilde{y}$.

(a) For the static equivalence part, assume $(M = N)\phi(A_r)$ with $var(M,N) \subseteq dom(A_r)$. As argued in 1, $\phi(C_r) \equiv \phi(A_r \mid \{\widetilde{z}/\widetilde{y}\}) = \phi(A_r) \mid \{\widetilde{z}/\widetilde{y}\}$ and $\phi(D_r) \equiv \phi(B_r \mid \{\widetilde{z}/\widetilde{y}\}) = \phi(B_r) \mid \{\widetilde{z}/\widetilde{y}\}$. Since $\{\widetilde{y}\} \cap var(M,N) = \emptyset$, we have $(M = N)\phi(C_r)$. From $\phi(C_r) \sim \phi(D_r)$, we obtain $(M = N)\phi(D_r)$. Now we show $(M = N)\phi(B_r)$. To this end, assume $\phi(D_r) \equiv \nu\widetilde{n}.\sigma$ and $M\sigma =_\Sigma N\sigma$. Then $\phi(B_r) \mid \{\widetilde{z}/\widetilde{y}\} \equiv \nu\widetilde{n}.\sigma \equiv \nu\widetilde{n}.\sigma^*$ and $M\sigma^* =_\Sigma N\sigma^*$($=_\Sigma$ is preserved by application of σ). Let $\sigma' = \sigma^*|_{dom(B_r)}$. Since $\{\widetilde{y}\} \cap fv(B_r) = \emptyset$ and $\{\widetilde{z}\} \subseteq dom(B_r)$, we have $\phi(B_r) \equiv \nu\widetilde{y}.(\phi(B_r) \mid \{\widetilde{z}/\widetilde{y}\}) \equiv \nu\widetilde{y}.\nu\widetilde{n}.\sigma^* \equiv \nu\widetilde{n}.\sigma'$. Furthermore, since $M\sigma' = M\sigma^*$, $N\sigma^* = N\sigma'$ and $M\sigma^* =_\Sigma N\sigma^*$, we have $M\sigma' =_\Sigma N\sigma'$. Thus $(M = N)\phi(B_r)$ holds, hence $A_r \sim_o B_r$.

(b) Assume $A_r \xrightarrow{\alpha}_o A'_r$. We need to show that there exists B'_r such that $B_r \overset{\alpha}{\Longrightarrow}_o B'_r$ and $(A'_r, B'_r) \in \mathbb{R}$. Consider the normalized derivation of transition of $A_r \xrightarrow{\alpha}_o A'_r$. We distinguish two cases depending on whether α is $\overline{a}\langle x\rangle$ or not.

i. α is not $\overline{a}\langle x\rangle$. We can safely assume $\{\widetilde{y}\} \cap bv(\alpha) = \emptyset$ since $\widetilde{y}$ are arbitrary. From $A_r \xrightarrow{\alpha}_o A'_r$, by PAR in [3], we know that $C_r \equiv_o A_r \mid \{\widetilde{z}/\widetilde{y}\} \xrightarrow{\alpha}_o A'_r \mid$

$\{\widetilde{z}/\widetilde{y}\} = C''_r$. Using Corollary 49, there exists C'_r such that $C_r \stackrel{\alpha}{\Longrightarrow} C'_r \equiv_o C''_r$. By hypothesis $C_r \approx_l D_r$, there exists D'_r such that $D_r \stackrel{\widehat{\alpha}}{\Longrightarrow} D'_r$ and $C'_r \approx_l D'_r$. Using Corollary 49, we have $D_r \stackrel{\widehat{\alpha}}{\Longrightarrow}_o D'_r$ or $D_r \equiv_o D'_r$.

We first check the case $D_r \stackrel{\widehat{\alpha}}{\Longrightarrow}_o D'_r$. From $C'_r \equiv_o C''_r$, we have $(\widetilde{z} = \widetilde{y})\phi(C'_r)$, hence also $(\widetilde{z} = \widetilde{y})\phi(D'_r)$. In other words, there exists B'_r such that $D'_r \equiv_o B'_r \mid \{\widetilde{z}/\widetilde{y}\}$ with $\{\widetilde{y}\} \cap fv(B'_r) = \emptyset$ (otherwise we can substitute them with the corresponding variables in $\widetilde{z}$). Adding restrictions $\nu\widetilde{y}$ to $B_r \mid \{\widetilde{z}/\widetilde{y}\} \equiv_o D_r \stackrel{\widehat{\alpha}}{\Longrightarrow}_o D'_r \equiv_o B'_r \mid \{\widetilde{z}/\widetilde{y}\}$, we have $B_r \stackrel{\widehat{\alpha}}{\Longrightarrow}_o B'_r$. From $A'_r \mid \{\widetilde{z}/\widetilde{y}\} \equiv_o C'_r \approx_l D'_r \equiv_o B'_r \mid \{\widetilde{z}/\widetilde{y}\}$, we know that $(A'_r, B'_r) \in \mathbb{R}$.

For the case when $D_r \equiv_o D'_r$, from the proof of Lemma 48, we can know that $D_r \equiv_o D'_r$ could happen only when α is τ. Let $B'_r = B_r$. Then we have $B_r \Longrightarrow_o B'_r$ and $A'_r \mid \{\widetilde{z}/\widetilde{y}\} \equiv_o C'_r \approx_l D'_r \equiv_o D_r \equiv_o B'_r \mid \{\widetilde{z}/\widetilde{y}\}$. Thus $(A'_r, B'_r) \in \mathbb{R}$.

ii. α is $\overline{a}\langle x\rangle$. In this case $A_r \equiv_o \mathcal{C}[\overline{a}\langle x\rangle.P_r] \xrightarrow{\overline{a}\langle x\rangle}_o \mathcal{C}[P_r] \equiv_o A'_r$ with $x \notin bv(\mathcal{C})$. Choose a fresh y', then we have $C_r \equiv_o \nu y'.\mathcal{C}[\overline{a}\langle y'\rangle.P_r \mid \{x/y'\}] \mid \{\widetilde{z}/\widetilde{y}\} \xrightarrow{\nu y'.\overline{a}\langle y'\rangle}_o \mathcal{C}[P_r \mid \{x/y'\}] \mid \{\widetilde{z}/\widetilde{y}\} \equiv_o \mathcal{C}[P_r] \mid \{\widetilde{z}, x/\widetilde{y}, y'\} \equiv_o A'_r \mid \{\widetilde{z}, x/\widetilde{y}, y'\}$ since x is a free variable. From Lemma 48, there exists a closed C'_r such that $C_r \stackrel{\nu y'.\overline{a}\langle y'\rangle}{\Longrightarrow} C'_r \equiv_o A'_r \mid \{\widetilde{z}, x/\widetilde{y}, y'\}$. By $C_r \approx_l D_r$, there exists D'_r such that $D_r \stackrel{\nu y'.\overline{a}\langle y'\rangle}{\Longrightarrow} D'_r \approx_l C'_r$.
Assume $\phi(A_r) \equiv_o \nu\widetilde{m}.\sigma$. Then $\phi(C_r) \equiv_o \nu\widetilde{m}.\sigma \mid \{\widetilde{z}, x/\widetilde{y}, y'\} \equiv_o \nu\widetilde{m}.(\sigma \cup \{\widetilde{z\sigma}, x\sigma/\widetilde{y}, y'\})$. Hence $(\widetilde{z}, x = \widetilde{y}, y')\phi(C'_r)$.[5] Since $\phi(C'_r) \sim \phi(D'_r)$, we obtain $(\widetilde{z}, x = \widetilde{y}, y')\phi(D'_r)$. Thus there exists B'_r such that $D'_r \equiv_o B'_r \mid \{\widetilde{z}, x/\widetilde{y}, y'\}$ with $fv(B'_r) \cap \{\widetilde{y}, y'\} = \emptyset$. Moreover $B_r \equiv_o \nu\widetilde{y}.(B_r \mid \{\widetilde{z}/\widetilde{y}\}) \equiv \nu\widetilde{y}.D_r \stackrel{\nu y'.\overline{a}\langle y'\rangle}{\Longrightarrow} \nu\widetilde{y}.D'_r \equiv_o B'_r \mid \{x/y'\}$. Hence $B_r \Rightarrow \nu y'.\mathcal{C}'[\overline{a}\langle y'\rangle.Q_r] \xrightarrow{\nu y'.\overline{a}\langle y'\rangle} \mathcal{C}'[Q_r] \Rightarrow B'_r \mid \{x/y'\}$ for some evaluation context $\mathcal{C}'$. Since static equivalence is closed under reduction (Lemma 1 in [3]), $\mathcal{C}'[Q_r] \sim B'_r \mid \{x/y'\}$. Moreover, since Q_r is a plain process which does not contain any active substitution, that is to say $\mathcal{C}'$ can rewrite y' with x. Hence we have $\mathcal{C}'[\overline{a}\langle x\rangle.Q_r] \equiv_o \mathcal{C}'[\overline{a}\langle y'\rangle.Q_r]$ which implies $\nu y'.\mathcal{C}'[\overline{a}\langle x\rangle.Q_r] \equiv_o \nu y'.\mathcal{C}'[\overline{a}\langle y'\rangle.Q_r]$. Hence $B_r \equiv_o \nu y'.\mathcal{C}'[\overline{a}\langle x\rangle.Q_r] \stackrel{\overline{a}\langle x\rangle}{\Longrightarrow}_o \nu y'.\mathcal{C}'[Q_r] \Rightarrow \nu y'.(B'_r \mid \{x/y'\}) \equiv_o B'_r$. Since $A'_r \mid \{\widetilde{z}, x/\widetilde{y}, y'\} \equiv_o C'_r \approx_l D'_r \equiv_o B'_r \mid \{\widetilde{z}, x/\widetilde{y}, y'\}$, and $\widetilde{y}$ and y' are arbitrary, we have that $(A'_r, B'_r) \in \mathbb{R}$.

Appendix C.2. Proofs of Theorem 21 and Corollary 22

In the previous Section 5, we define function $\mathcal{T}$ to transform an extended process in applied pi to a pure extended process, namely a extended process with no cell name, in stateful applied pi. In this section, we shall prove that this transformation function $\mathcal{T}$ keeps both observational equivalence and labelled bisimilarity, i.e. Theorem 21 in

[5] $(\widetilde{z} = \widetilde{y})\phi(C'_r)$ abbreviates $(z_1 = y_1)\phi(C_r), \cdots (z_n = y_n)\phi(C'_r)$

Section 5. For the sake of readability, we recall the definition for $\mathcal{T}$ here:

$$\begin{array}{ll}
\mathcal{T}(0) = (\emptyset, \emptyset) & \mathcal{T}(\nu x.A_r) = \nu \widetilde{n}.(\sigma, \mathcal{P}) \\
& \quad \textbf{if } \mathcal{T}(A_r) = \nu \widetilde{n}.(\sigma \cup \{M/x\}, \mathcal{P}) \\
\mathcal{T}(\{M/x\}) = (\{M/x\}, \emptyset) & \mathcal{T}(\nu n.A_r) = \nu n.\mathcal{T}(A_r) \\
\multicolumn{2}{l}{\mathcal{T}(A_r^1 \mid A_r^2) = \nu \widetilde{n}_1, \widetilde{n}_2.((\sigma_1 \cup \sigma_2)^*, (\mathcal{P}_1 \cup \mathcal{P}_2)(\sigma_1 \cup \sigma_2)^*)} \\
& \quad \textbf{if } \mathcal{T}(A_r^i) = \nu \widetilde{n}_i.(\sigma_i, \mathcal{P}_i) \text{ for } i = 1, 2 \\
\multicolumn{2}{l}{\mathcal{T}(A_r) = (\emptyset, \{A_r\}) \text{ in all other cases of } A_r}
\end{array}$$

Lemma 52. *If $A_r \equiv B_r$ then $\mathcal{T}(A_r) \simeq \mathcal{T}(B_r)$.*

Proof. Considering the normalised derivation of $A_r \equiv B_r$. The proof goes by induction on the number of derivation. Assume $A_r \equiv \mathcal{C}[D_r^1] \equiv^1 \mathcal{C}[D_r^2] = B_r$. By induction hypothesis, we have $\mathcal{T}(A_r) \simeq \mathcal{T}(\mathcal{C}[D_r^1])$. We can easily check the structural equivalence $D_r^1 \equiv^1 D_r^2$ defined in Figure C.6 satisfies $\mathcal{T}(D_r^1) \simeq \mathcal{T}(D_r^2)$. Thus we have $\mathcal{T}(\mathcal{C}[D_r^1]) \simeq \mathcal{T}(\mathcal{C}[D_r^2])$. Finally we have $\mathcal{T}(A_r) \simeq \mathcal{T}(B_r)$.

Lemma 53. *Let $\mathcal{C}_r$ be an evaluation context in which bound names and bound variables are pairwise-distinct and different from the free ones in $\mathcal{C}_r$. Let $\widetilde{x}$ be a tuple of pairwise-distinct variables such that the hole is in the scope of an occurrence of νx in $\mathcal{C}_r$. Then $\mathcal{T}(\mathcal{C}_r) = \nu \widetilde{n}.(\sigma_{c \setminus \widetilde{x}}\text{-}, \mathcal{P}_c\text{-})$ for some $\widetilde{n}, \sigma_c, \mathcal{P}_c$.*

For any extended process A_r such that $\mathcal{C}_r[A_r]$ is an extended process, if $\mathcal{T}(A_r) = \nu \widetilde{m}.(\sigma_a, \mathcal{P}_a)$ for some of names $\widetilde{m}$ with $\{\widetilde{m}\} \cap (\widetilde{n} \cup fn(\mathcal{C}_r)) = \emptyset$ and some $\mathcal{P}_a$, then

$$\mathcal{T}(\mathcal{C}_r[A_r]) = \nu \widetilde{n}, \widetilde{m}.((\sigma_c \cup \sigma_a)^*_{\setminus \widetilde{x}}, (\mathcal{P}_c \cup \mathcal{P}_a)(\sigma_c \cup \sigma_a)^*)$$

As a corollary, when A_r is closed, we have $\mathcal{T}(\mathcal{C}_r) = \nu \widetilde{n}.(\sigma_c\text{-}, \mathcal{P}_c\text{-})$ for some $\widetilde{n}, \sigma_c, \mathcal{P}_c$ and.

$$\mathcal{T}(\mathcal{C}_r[A_r]) = \nu \widetilde{n}, \widetilde{m}.(\sigma_c \sigma_a \cup \sigma_{a \setminus \widetilde{x}}, \mathcal{P}_c \sigma_a \cup \mathcal{P}_a).$$

Proof. The proof goes by induction on the structure of $\mathcal{C}_r$.

1. In the base case $\mathcal{C}_r =$ -, we have $\widetilde{n} = \emptyset$, $\sigma_1 = \emptyset$ and $\mathcal{P}_1 = \emptyset$. The conclusion holds trivially.

2. Assume $\mathcal{C}_r = \nu l.\mathcal{C}'_r$, by induction hypothesis, we have

(a) $\mathcal{T}(\mathcal{C}'_r) = \nu \widetilde{n}_1.(\sigma_{1 \setminus \widetilde{x}}\text{-}, \mathcal{P}_1\text{-})$ for some $\widetilde{n}_1, \sigma_1, \mathcal{P}_1$;

(b) for any A_r with $\mathcal{T}(A_r) = \nu \widetilde{m}.(\sigma_a, \mathcal{P}_a)$, we have $\mathcal{T}(\mathcal{C}'_r[A_r]) = \nu \widetilde{n}_1, \widetilde{m}.((\sigma_1 \cup \sigma_a)^*_{\setminus \widetilde{x}}, (\mathcal{P}_1 \cup \mathcal{P}_a)(\sigma_1 \cup \sigma_a)^*)$ where $\widetilde{x}$ is the variables such that the hole in $\mathcal{C}'_r$ is in the scope of νx.

Then we have $\mathcal{T}(\nu l.\mathcal{C}'_r) = \nu l, \widetilde{n}_1.(\sigma_{1 \setminus \widetilde{x}}\text{-}, \mathcal{P}_1\text{-})$ and $\mathcal{T}(\nu l.\mathcal{C}'_r[A_r]) = \nu l, \widetilde{n}_1, \widetilde{m}.((\sigma_1 \cup \sigma_a)^*_{\setminus \widetilde{x}}, (\mathcal{P}_1 \cup \mathcal{P}_a)(\sigma_1 \cup \sigma_a)^*)$.

3. Assume $\mathcal{C}_r = \nu z.\mathcal{C}'_r$, By induction hypothesis, we have

(a) $\mathcal{T}(\mathcal{C}'_r) = \nu\widetilde{n}.(\sigma_{1\setminus\widetilde{x}}\text{-}, \mathcal{P}_1\text{-})$ for some $\widetilde{n}, \sigma_1, \mathcal{P}_1$;

(b) for any A_r with $\mathcal{T}(A_r) = \nu\widetilde{m}.(\sigma_a, \mathcal{P}_a)$, we have $\mathcal{T}(\mathcal{C}'_r[A_r]) = \nu\widetilde{n}, \widetilde{m}.((\sigma_1 \cup \sigma_a)^*_{\setminus\widetilde{x}},$
$(\mathcal{P}_1 \cup \mathcal{P}_a)(\sigma_1 \cup \sigma_a)^*)$ and $\widetilde{x}$ is the variables such that the hole in $\mathcal{C}'_r$ is in the scope of νx.

Then we have $\mathcal{T}(\nu z.\mathcal{C}'_r) = \nu\widetilde{n}.(\sigma_{1\setminus\widetilde{x},z}\text{-}, \mathcal{P}_1\text{-})$ and $\mathcal{T}(\nu z.\mathcal{C}'_r[A_r]) = \nu\widetilde{n}, \widetilde{m}.((\sigma_1 \cup \sigma_a)^*_{\setminus\widetilde{x},z}, (\mathcal{P}_1 \cup \mathcal{P}_a)(\sigma_1 \cup \sigma_a)^*)$.

4. Assume $\mathcal{C}_r = \mathcal{C}'_r \mid B_r$, then $\mathcal{T}(\mathcal{C}_r) = \mathcal{T}(\mathcal{C}'_r \mid B_r)$. By induction hypothesis, we have

(a) $\mathcal{T}(\mathcal{C}'_r) = \nu\widetilde{n}_1.(\sigma_{1\setminus\widetilde{x}}\text{-}, \mathcal{P}_1\text{-})$ for some $\widetilde{n}_1, \sigma_1, \mathcal{P}_1$;

(b) for any A_r with $\mathcal{T}(A_r) = \nu\widetilde{m}.(\sigma_a, \mathcal{P}_a)$, we have $\mathcal{T}(\mathcal{C}'_r[A_r]) = \nu\widetilde{n}_1, \widetilde{m}.((\sigma_1 \cup \sigma_a)^*_{\setminus\widetilde{x}},$
$(\mathcal{P}_1 \cup \mathcal{P}_a)(\sigma_1 \cup \sigma_a)^*)$ where $\widetilde{x}$ is the variables such that the hole in $\mathcal{C}'_r$ is in the scope of νx.

Let $\mathcal{T}(B_r) = \nu\widetilde{n}_2.(\sigma_2, \mathcal{P}_2)$. Then $\mathcal{T}(\mathcal{C}_r \mid B_r) = \nu\widetilde{n}_1, \widetilde{n}_2.((\sigma_{1\setminus\widetilde{x}} \cup \sigma_2)^*\text{-}, (\mathcal{P}_1 \cup \mathcal{P}_2)(\sigma_{1\setminus\widetilde{x}} \cup \sigma_2)^*\text{-})$. And $\mathcal{T}(\mathcal{C}'_r[A_r] \mid B_r) = \nu\widetilde{n}_1, \widetilde{n}_2, \widetilde{m}.(((\sigma_1 \cup \sigma_a)^*_{\setminus\widetilde{x}} \cup \sigma_2)^*, ((\mathcal{P}_1 \cup \mathcal{P}_a)(\sigma_1 \cup \sigma_a)^* \cup \mathcal{P}_2)((\sigma_1 \cup \sigma_a)^*_{\setminus\widetilde{x}} \cup \sigma_2)^*)$. Since the variable restricted by $\nu\widetilde{x}$ cannot occur in B_r and the domains of $\sigma_1, \sigma_2, \sigma_a$ are pairwise disjoint and these substitutions are all cycle-free, we can see that the order of iterating the substitutions $\sigma_1, \sigma_2, \sigma_a$ does not matter and we can derive that $(\sigma_{1\setminus\widetilde{x}} \cup \sigma_2)^* = (\sigma_1 \cup \sigma_2)^*_{\setminus\widetilde{x}}$, $((\sigma_1 \cup \sigma_a)^*_{\setminus\widetilde{x}} \cup \sigma_2)^* = ((\sigma_1 \cup \sigma_a)^* \cup \sigma_2)^*_{\setminus\widetilde{x}}$, and $((\sigma_1 \cup \sigma_a)^* \cup \sigma_2)^* = ((\sigma_1 \cup \sigma_2)^* \cup \sigma_a)^* = (\sigma_1 \cup \sigma_2 \cup \sigma_a)^*$. Since $\widetilde{x}$ do not occur in $\mathcal{P}_1, \mathcal{P}_2$, we have $\mathcal{T}(\mathcal{C}_r \mid B_r) = \nu\widetilde{n}_1, \widetilde{n}_2.((\sigma_1 \cup \sigma_2)^*_{\setminus\widetilde{x}}\text{-}, (\mathcal{P}_1 \cup \mathcal{P}_2)(\sigma_1 \cup \sigma_2)^*\text{-})$. And $\mathcal{T}(\mathcal{C}'_r[A_r] \mid B_r) = \nu\widetilde{n}_1, \widetilde{n}_2, \widetilde{m}.(((\sigma_1 \cup \sigma_a)^* \cup \sigma_2)^*_{\setminus\widetilde{x}}, ((\mathcal{P}_1 \cup \mathcal{P}_a)(\sigma_1 \cup \sigma_a)^* \cup \mathcal{P}_2)((\sigma_1 \cup \sigma_a)^*_{\setminus\widetilde{x}} \cup \sigma_2)^*) = \nu\widetilde{n}_1, \widetilde{n}_2, \widetilde{m}.(((\sigma_1 \cup \sigma_2)^* \cup \sigma_a)^*_{\setminus\widetilde{x}}, ((\mathcal{P}_1 \cup \mathcal{P}_a)(\sigma_1 \cup \sigma_2 \cup \sigma_a)^* \cup \mathcal{P}_2(\sigma_1 \cup \sigma_2 \cup \sigma_a)^*) = \nu\widetilde{n}_1, \widetilde{n}_2, \widetilde{m}.(((\sigma_1 \cup \sigma_2)^* \cup \sigma_a)^*_{\setminus\widetilde{x}}, ((\mathcal{P}_1 \cup \mathcal{P}_2)(\sigma_1 \cup \sigma_2)^* \cup \mathcal{P}_a)((\sigma_1 \cup \sigma_2)^* \cup \sigma_a)^*)$.

When A_r is closed, the active substititons in $\mathcal{C}_r$ will not be applied to A_r, the proof is similar to the above general case.

Lemma 54. *If $A_r \xrightarrow{\alpha} A'_r$ with $fv(A_r) \cap bv(\alpha) = \emptyset$, then $\mathcal{T}(A_r) \stackrel{\alpha}{\Longrightarrow} B \simeq \mathcal{T}(A'_r)$ for some B.*

Proof. Consider the normalized derivation of transition of $A_r \xrightarrow{\alpha} A'_r$. We only take the case when $\alpha = \overline{a}\langle c\rangle$ as an example here and the other cases are similar. Assume $A_r \equiv \mathcal{C}[\overline{a}\langle c\rangle.P_r] \xrightarrow{\overline{a}\langle c\rangle} \mathcal{C}[P_r] \equiv A'_r$ and $\mathcal{T}(\mathcal{C}) = \nu\widetilde{n}.(\sigma_{\setminus\widetilde{x}}\text{-}, \mathcal{P}\text{-})$. By Lemma 52 and Lemma 53, we have that

$$\mathcal{T}(A_r) \simeq \nu\widetilde{n}.(\sigma_{\setminus\widetilde{x}}, \{\overline{a}\langle c\rangle.P_r\sigma\} \cup \mathcal{P}) \xrightarrow{\overline{a}\langle c\rangle} \nu\widetilde{n}.(\sigma_{\setminus\widetilde{x}}, \{P_r\sigma\} \cup \mathcal{P})$$

Let $\mathcal{T}(P_r) = \nu\widetilde{m}.(\emptyset, \mathcal{Q})$ for some $\widetilde{m}, \mathcal{Q}$. From $\mathcal{C}[P_r] \equiv A'_r$, using Lemma 52 and Lemma 53, we have $\mathcal{T}(A'_r) \simeq \mathcal{T}(\mathcal{C}[P_r]) = \nu\widetilde{n}, \widetilde{m}.(\sigma, \mathcal{Q}\sigma \cup \mathcal{P})$. For a plain process P_r, the function $\mathcal{T}$ only pulls the name binders to the top level and split the parallel composition, thus we can see that $\mathcal{T}(A_r) \simeq \nu\widetilde{n}.(\sigma_{\setminus\widetilde{x}}, \{\overline{a}\langle c\rangle.P_r\sigma\} \cup \mathcal{P}) \xrightarrow{\overline{a}\langle c\rangle} \nu\widetilde{n}.(\sigma_{\setminus\widetilde{x}}, \{P_r\sigma\} \cup \mathcal{P}) \Longrightarrow \nu\widetilde{n}, \widetilde{m}.(\sigma_{\setminus\widetilde{x}}, \mathcal{Q}\sigma \cup \mathcal{P}) = \mathcal{T}(\mathcal{C}[P_r]) \simeq \mathcal{T}(A'_r)$. That is to say there exist A and A' such that $\mathcal{T}(A_r) \simeq A \stackrel{\overline{a}\langle c\rangle}{\Longrightarrow} A' \simeq \mathcal{T}(A'_r)$. By Corollary 15, there exists B such that $\mathcal{T}(A_r) \stackrel{\overline{a}\langle c\rangle}{\Longrightarrow} B \simeq A' \simeq \mathcal{T}(A'_r)$. This concludes the proof.

Corollary 55. *If $A_r \stackrel{\alpha}{\Longrightarrow} A'_r$ with $fv(A) \cap bv(\alpha) = \emptyset$, then $\mathcal{T}(A_r) \stackrel{\alpha}{\Longrightarrow} B \simeq \mathcal{T}(A'_r)$ for some B.*

Proof. Using Corollary 15 and Lemma 54 several times.

Lemma 56. *If $\mathcal{T}(A_r) = \nu\widetilde{n}(\sigma, \{P_i\}_i)$ then $A_r \equiv \nu\widetilde{n}.(\sigma \mid \prod_i P_i)$.*

Proof. We proceed induction on the definition of $\mathcal{T}$. The interesting cases are $A_r \mid B_r$ and $\nu x.A_r$ while the other cases are trivial. For parallel composition $A_r \mid B_r$, by induction hypothesis, we know $A_r \equiv \nu\widetilde{n}.(\sigma_1 \mid \prod_i P_i)$ and $B_r \equiv \nu\widetilde{m}.(\sigma_2 \mid \prod_j Q_j)$ where $\mathcal{T}(A_r) = \nu\widetilde{n}.(\sigma_1, \{P_i\}_i)$ and $\mathcal{T}(B_r) = \nu\widetilde{m}.(\sigma_2, \{Q_j\}_j)$. Let $\sigma = (\sigma_1 \cup \sigma_2)^*$. From the definition of $\mathcal{T}$, we have $\mathcal{T}(A_r \mid B_r) = \nu\widetilde{n}, \widetilde{m}.(\sigma, \mathcal{P}_1\sigma \cup \mathcal{P}_2\sigma)$. Note that applying active substitutions until reaching idempotemce keeps structural equivalence. From structural equivalence, we can deduce that $A_r \mid B_r \equiv \nu\widetilde{n}.(\sigma_1 \mid \prod_i P_i) \mid \nu\widetilde{m}.(\sigma_2 \mid \prod_j Q_j) \equiv \nu\widetilde{n}.\nu\widetilde{m}.(\sigma_1 \mid \prod_i P_i \mid \sigma_2 \mid \prod_j Q_j) \equiv \nu\widetilde{n}.\nu\widetilde{m}.(\sigma \mid \prod_i P_i\sigma \mid \prod_j Q_j\sigma)$. The result holds for parallel composition. For the case $\mathcal{T}(\nu x.A_r) = \nu\widetilde{n}.(\sigma, \{P_i\}_i)$ where $\mathcal{T}(A_r) = \nu\widetilde{n}.(\sigma \cup \{M/x\}, \{P_i\}_i)$, by induction hypothesis we have $A_r \equiv \nu\widetilde{n}.(\sigma \mid \{M/x\} \mid \prod_i P_i)$. Since P_i are applied, x will not occur in σ or P_i. Hence we have $\nu x.A_r \equiv \nu x.\nu\widetilde{n}.(\sigma \mid \{M/x\} \mid \prod_i P_i) \equiv \nu\widetilde{n}.(\sigma \mid \prod_i P_i)$ and $\mathcal{T}(\nu x.A_r) = \nu\widetilde{n}.(\sigma, \{P_i\}_i)$.

Lemma 57. *If $\mathcal{T}(A_r) \simeq \nu\widetilde{n}(\sigma, \{P_i\}_i)$ then $A_r \equiv \nu\widetilde{n}.(\sigma \mid \prod_i P_i)$.*

Proof. The proof goes by induction on the number of rewriting steps of $\simeq$. When the number is zero, it is Lemma 56. Assume $\mathcal{T}(A_r) \simeq \nu\widetilde{m}(\sigma', \{P'_i\}_i) \simeq^1 \nu\widetilde{n}(\sigma, \{P_i\}_i)$. By induction hypothesis $A_r \equiv \nu\widetilde{m}.(\sigma' \mid \prod_i P'_i)$. According to Definition 12, we can easily see that $\nu\widetilde{m}.(\sigma' \mid \prod_i P'_i) \equiv \nu\widetilde{n}.(\sigma \mid \prod_i P_i)$. Hence $A_r \equiv \nu\widetilde{n}.(\sigma \mid \prod_i P_i)$.

Lemma 58. *If A_r is closed and $\mathcal{T}(A_r) \xrightarrow{\alpha} A$ with $fv(\alpha) \subseteq dom(\mathcal{T}(A_r))$. Then there exists a closed A'_r such that $A_r \stackrel{\alpha}{\Longrightarrow} A'_r$ and $\mathcal{T}(A'_r) \simeq A$.*

Proof. We take the case for the expansion of replication as the example here. The other cases are similar.

Assume $\mathcal{T}(A_r) = \nu\widetilde{n}.(\sigma, \{Q_i\}_i \cup \{!P_r\}) \xrightarrow{\tau} \nu\widetilde{n}.(\sigma, \{Q_i\}_i \cup \{!P_r, P_r\}) = A$. By Lemma 57, we have $A_r \equiv \nu\widetilde{n}.(\sigma \mid !P_r \mid \prod_i Q_i)$. Hence $A_r \equiv \nu\widetilde{n}.(\sigma \mid !P_r \mid \prod_i Q_i) \xrightarrow{\tau} \nu\widetilde{n}.(\sigma \mid P_r \mid !P_r \mid \prod_i Q_i) = A'_r$. Assume $\mathcal{T}(P_r) = \nu\widetilde{m}.(\emptyset, \mathcal{Q})$ for some $\widetilde{m}, \mathcal{Q}$. Since $\mathcal{T}$ only pulls out name binders and split parallel compositions for P_r, we can see that $\mathcal{T}(A'_r) = \nu\widetilde{n}, \widetilde{m}.(\sigma, \{Q_i\}_i \cup \{!P_r\} \cup \mathcal{Q}) \simeq A$. Since A_r is closed, we know that $\mathcal{T}(A_r)$, A and A'_r are also closed.

Corollary 59. *If A_r is closed and $\mathcal{T}(A_r) \stackrel{\alpha}{\Longrightarrow} A$ with $fv(\alpha) \subseteq dom(\mathcal{T}(A_r))$. Then there exists a closed A'_r such that $A_r \stackrel{\alpha}{\Longrightarrow} A'_r$ and $\mathcal{T}(A'_r) \simeq A$.*

Proof. By repeated applications of Lemma 58 and Corollary 15.

Lemma 60. *Static equivalence $\approx_s$ on pure extended processes is closed under $\simeq$.*

Proof. Since $\approx_s$ is symmetric, it is sufficient to prove $\approx_s \simeq \subseteq \approx_s$. The proof goes by induction on the length of derivation sequence for $\simeq$. When the length is 0, the result holds trivially. For the inductive step, w.l.o.g., we may assume $A \approx_s A' \simeq B \simeq^1 C$. By the induction hypothesis, we have $A \approx_s B$. Now we show $A \approx_s C$. We can easily check $A \approx_s C$ holds for the cases when the rewriting $B \simeq^1 C$ is on restricted names or parallel composition. For the term rewriting case, assume $B = \nu\widetilde{n}.(\sigma\{M/z\}, \mathcal{P}\{M/z\}) \simeq^1 \nu\widetilde{n}.(\sigma\{N/z\}, \mathcal{P}\{N/z\}) = C$ and $M =_\Sigma N$. Then for each $x \in dom(A)$ we have $\sigma\{M/z\}(x) =_\Sigma \sigma\{N/z\}(x)$. Let $A = \nu\widetilde{m}.(\sigma', \mathcal{P}')$. Since $A \approx_s B$, for any N_1, N_2 with $name(N_1, N_2) \cap \{\widetilde{n}, \widetilde{m}\} = \emptyset$, $N_1\sigma' =_\Sigma N_2\sigma'$ iff $N_1\sigma\{M/z\} =_\Sigma N_2\sigma\{M/z\}$. Since $M =_\Sigma N$, $N_1\sigma\{M/z\} =_\Sigma N_1\sigma\{N/z\}$ and $N_2\sigma\{M/z\} =_\Sigma N_2\sigma\{N/z\}$. Thus $N_1\sigma' =_\Sigma N_2\sigma'$ iff $N_1\sigma\{N/z\} =_\Sigma N_2\sigma\{N/z\}$. Therefore $A \approx_s C$.

The transformation function $\approx_s$ preserves static equivalence.

Lemma 61. *Let A_r and B_r be two closed extended processes. Then $A_r \approx_s B_r$ iff $\mathcal{T}(A_r) \approx_s \mathcal{T}(B_r)$.*

Proof. Let $\mathcal{T}(A_r) = \nu\widetilde{n}_1.(\sigma_1, \mathcal{P}_1)$ and $\mathcal{T}(B_r) = \nu\widetilde{n}_2.(\sigma_2, \mathcal{P}_2)$. According to the definition of $\mathcal{T}$, we can see that $\phi(A_r) \equiv \nu\widetilde{n}_1.\sigma_1$. Whenever $\phi(A_r) \equiv \nu\widetilde{m}.\sigma$, we have that $\nu\widetilde{n}_1.\sigma_1 \equiv \nu\widetilde{m}.\sigma$. Using Lemma 52, we have $\nu\widetilde{n}_1.\sigma_1 \simeq \nu\widetilde{m}.\sigma^*$.

1. ($\Longleftarrow$) Let M, N be two arbitrary terms with $var(M, N) \subseteq dom(A_r)$ and $M\sigma =_\Sigma N\sigma$ for some $\phi(A_r) \equiv \nu\widetilde{m}.\sigma$. Since $=_\Sigma$ is closed under the application of substitutions, we have $M\sigma^* =_\Sigma N\sigma^*$. From $\nu\widetilde{n}_2.\sigma_2 \approx_s \nu\widetilde{n}_1.\sigma_1 \simeq \nu\widetilde{m}.\sigma^*$. By Lemma 60, we have $\nu\widetilde{n}_2.\sigma_2 \approx_s \nu\widetilde{m}.\sigma^*$. That is to say $M\sigma_2 =_\Sigma N\sigma_2$. From $\phi(B_r) \equiv \nu\widetilde{n}_2.\sigma_2$, we know $A_r \sim B_r$.

2. ($\Longrightarrow$) Let M, N be two arbitrary terms. Assume $M\sigma_1 =_\Sigma N\sigma_1$. We need to show $M\sigma_2 =_\Sigma N\sigma_2$. Since $\nu\widetilde{n}_1.\sigma_1 \equiv \phi(A_r)$. By the hypothesis $A_r \sim B_r$, there exist $\widetilde{m}, \sigma$ such that $\phi(B_r) \equiv \nu\widetilde{m}.\sigma$ and $M\sigma =_\Sigma N\sigma$. Since $=_\Sigma$ is closed under substitution, it holds that $M\sigma^* =_\Sigma N\sigma^*$. From $\nu\widetilde{m}.\sigma^* \simeq \nu\widetilde{n}_2.\sigma_2$. By Lemma 60 we obtain $\nu\widetilde{m}.\sigma^* \approx_s \nu\widetilde{n}_2.\sigma_2$. Hence $M\sigma_2 =_\Sigma N\sigma_2$. Thus $\mathcal{T}(A_r) \approx_s \mathcal{T}(B_r)$.

The following proposition states that transformation $\mathcal{T}$ keeps labelled bisimilarity.

Proposition 62. *$A_r \approx_l B_r$ if and only if $\mathcal{T}(A_r) \approx_l \mathcal{T}(B_r)$.*

Proof.

1. ($\Longleftarrow$) We construct a set $\mathbb{R}$ on closed extended processes thus

$$\mathbb{R} = \{\, (A_r, B_r) \mid \mathcal{T}(A_r) \simeq \approx_l \simeq \mathcal{T}(B_r) \,\}.$$

We show $\mathbb{R} \subseteq \approx_l$. Suppose $\mathcal{T}(A_r) \simeq C \approx_l D \simeq \mathcal{T}(B_r)$. In combination with Lemma 61 and Lemma 60 we obtain the static equivalence part $A_r \approx_s B_r$ immediately. We are left to show the agreement between transitions. Suppose $A_r \xrightarrow{\alpha} A'_r$ with $fv(\alpha) \subseteq dom(A_r)$. Clearly A_r, A'_r, C, D are all closed. From Lemma 54 and Corollary 14, there exists C' such that $C \stackrel{\alpha}{\Longrightarrow} C' \simeq \mathcal{T}(A'_r)$, where C' is closed because C is closed and $fv(\alpha) \subseteq dom(C) = dom(A_r)$. From $D \approx_l C$, there exists D' such that $D \stackrel{\widehat{\alpha}}{\Longrightarrow} D' \approx_l C'$. By Corollary 15 and Corollary 59 we can deduce that there exists a closed B'_r such that $B_r \stackrel{\widehat{\alpha}}{\Longrightarrow} B'_r$ and $\mathcal{T}(B'_r) \simeq D'$. Hence $(A'_r, B'_r) \in \mathbb{R}$.

2. ($\Longrightarrow$) This direction is proved by constructing a set $\mathbb{S}$ on closed processes thus

$$\mathbb{S} = \{\, (A, B) \mid A \simeq \mathcal{T}(A_r),\ A_r \approx_l B_r,\ \mathcal{T}(B_r) \simeq B \,\}.$$

We show $\mathbb{S} \subseteq \approx_l$. First, $A \approx_s B$ follows from Lemma 61 and Lemma 60. Suppose $A \xrightarrow{\alpha} A'$. By Corollary 14 we have $\mathcal{T}(A_r) \stackrel{\alpha}{\Longrightarrow} A_1 \simeq A'$. By Lemma 58 we have $A_r \xrightarrow{\alpha} A'_r$ and $\mathcal{T}(A'_r) \simeq A_1 \simeq A'$. Since $A_r \approx_l B_r$, there is some B'_r such that $B_r \stackrel{\alpha}{\Longrightarrow} B'_r \approx_l A'_r$. By Corollary 55 and Corollary 15 we have $B \stackrel{\alpha}{\Longrightarrow} B' \simeq \mathcal{T}(B'_r)$. Hence $(A', B') \in \mathbb{S}$.

Now we start to prove that transformation $\mathcal{T}$ keeps observational equivalence. Recall that on closed pure extended processes, the observational equivalence $\approx^e$ is defined exactly the same as in Definition 1 except that the evaluation context is pure, that is, the context does not contain any cell name.

Lemma 63. *Assume two closed pure extended processes A, B. If $A \approx^e B$ then $A_{\setminus \widetilde{z}} \approx^e B_{\setminus \widetilde{z}}$ for any variables $\widetilde{z} \subseteq dom(A)$.*

Proof. We construct a set $\mathcal{R}$ as follows

$$\mathcal{R} = \{\, (A_{\setminus \widetilde{z}}, B_{\setminus \widetilde{z}}) \mid A \approx^e B,\ \widetilde{z} \subseteq dom(A) \,\}$$

and we will prove that $\mathcal{R} \subseteq \approx$. For the part related to $\Downarrow_a$ and $\Longrightarrow$, we can easily see that removing or adding any active substitutions does not affect $\Downarrow_a$ or $\Longrightarrow$. For any evaluation context $\mathcal{C}$, we can safely assume that $fv(\mathcal{C}) \cap \widetilde{z} = \emptyset$. Otherwise we can choose fresh variables $\widetilde{x}$ and let $\varrho = \{\widetilde{x}/\widetilde{z}\}$ and have $A_{\setminus \widetilde{z}} = \varrho(A)_{\setminus \widetilde{x}}, B_{\setminus \widetilde{z}} = \varrho(B)_{\setminus \widetilde{x}}, \varrho(A) \approx^e \varrho(B)$. Thus we have $\mathcal{C}[A_{\setminus \widetilde{z}}] = \mathcal{C}[A]_{\setminus \widetilde{z}}$, $\mathcal{C}[B_{\setminus \widetilde{z}}] = \mathcal{C}[B]_{\setminus \widetilde{z}}$ and $\mathcal{C}[A] \approx^e \mathcal{C}[B]$. Finally $(\mathcal{C}[A_{\setminus \widetilde{z}}], \mathcal{C}[B_{\setminus \widetilde{z}}]) \in \mathcal{R}$.

Lemma 64. *If $A \simeq B$ with A, B are closed pure extended processes. Then $\mathcal{C}[A]_{\setminus\widetilde{z}} \simeq \mathcal{C}[B]_{\setminus\widetilde{z}}$ for any closing pure evaluation context $\mathcal{C}$ and $\widetilde{z} \subseteq dom(A, B)$.*

Proof. The proof goes by induction on the length of proof sequence for $\simeq$. When the length is 0, the result holds trivially. For the inductive step, w.l.o.g., we assume $A \simeq D \simeq^1 B$. As stated before, we can safely assume that D is closed. By the induction hypothesis, we have $\mathcal{C}[A]_{\setminus\widetilde{z}} \simeq \mathcal{C}[D]_{\setminus\widetilde{z}}$. Now we will show $\mathcal{C}[D]_{\setminus\widetilde{z}} \simeq \mathcal{C}[B]_{\setminus\widetilde{z}}$. If the rewriting $D \simeq^1 B$ is about restricted names or parallel composition, the conclusion clearly holds. Assume the rewriting is $D = \nu\widetilde{m}.(\sigma\{M/x\}, \mathcal{P}\{M/x\}) \simeq \nu\widetilde{m}.(\sigma\{N/x\}, \mathcal{P}\{N/x\}) = B$ with $M =_\Sigma N$. Let $\mathcal{C} = \nu\widetilde{n}.(\sigma', \mathcal{P}')$. We can safely assume that x is fresh (otherwise we can use α-conversion). Then $\mathcal{C}[D]_{\setminus\widetilde{z}} = \nu\widetilde{n}.\nu\widetilde{m}.(\sigma'\sigma\{M/x\} \cup \sigma_{\setminus\widetilde{z}}\{M/x\}, \mathcal{P}\{M/x\} \cup \mathcal{P}'\sigma\{M/x\}) \simeq \nu\widetilde{n}.\nu\widetilde{m}.(\sigma'\sigma\{N/x\} \cup \sigma_{\setminus\widetilde{z}}\{N/x\}, \mathcal{P}\{N/x\} \cup \mathcal{P}'\sigma\{N/x\}) = \mathcal{C}[B]_{\setminus\widetilde{z}}$. By transition, we get $\mathcal{C}[A]_{\setminus\widetilde{z}} \simeq \mathcal{C}[B]_{\setminus\widetilde{z}}$.

Proposition 65. *$A_r \approx B_r$ implies $\mathcal{T}(A_r) \approx^e \mathcal{T}(B_r)$.*

Proof.

$$\mathcal{S} = \{ (A, B) \mid A \simeq \mathcal{T}(A_r),\ A_r \approx B_r,\ \mathcal{T}(B_r) \simeq B \}$$

1. First we show that $A \Downarrow_a$ implies $B \Downarrow_a$. By Corollary 15 and Corollary 59, we can see that $A_r \Downarrow_a$. From $A_r \approx B_r$, we have $B_r \Downarrow_a$. Then from Corollary 59 and Corollary 15, we have that $B \Downarrow_a$.

2. Assume $A \Longrightarrow A'$ then we will show that there exists B' such that $B \Longrightarrow B'$ and $(A', B') \in \mathcal{S}$. By Corollary 15 and Corollary 59, we have $A_r \Longrightarrow A'_r$ with $\mathcal{T}(A'_r) \simeq A'$. From $A_r \approx B_r$, there exists B'_r such that $B_r \Longrightarrow B'_r \approx A'_r$. By Corollary 55 and Corollary 15, we know that there exists B' such that $B \Longrightarrow B' \simeq \mathcal{T}(B'_r)$. Hence $(A', B') \in \mathcal{S}$.

3. For any $\mathcal{C}$ we need to show that $(\mathcal{C}[A], \mathcal{C}[B]) \in \mathcal{S}$. Assume $\mathcal{C} = \nu\widetilde{l}.(\sigma, \{P_i\}_i)$. Let $\mathcal{C}_r = \nu\widetilde{l}.(\sigma \mid \prod_i P_i \mid [\cdot])$. Then we can easily see that $\mathcal{T}(\mathcal{C}_r[A_r]) = \mathcal{C}[\mathcal{T}(A_r)]$ and $\mathcal{T}(\mathcal{C}_r[B_r]) = \mathcal{C}[\mathcal{T}(B_r)]$. Since $A \simeq \mathcal{T}(A_r)$ and $B \simeq \mathcal{T}(B_r)$, by Lemma 64, we have $\mathcal{C}[A] \simeq \mathcal{C}[\mathcal{T}(A_r)] = \mathcal{T}(\mathcal{C}_r[A_r])$ and $\mathcal{C}[B] \simeq \mathcal{C}[\mathcal{T}(B_r)] = \mathcal{T}(\mathcal{C}_r[B_r])$. Since $\approx$ is closed by evaluation context, namely $\mathcal{C}_r[A_r] \approx \mathcal{C}_r[B_r]$, we know that $(\mathcal{C}[A], \mathcal{C}[B]) \in \mathcal{S}$.

Proposition 66. *For two closed extended processes A_r and B_r in applied pi calculus [3], $\mathcal{T}(A_r) \approx^e \mathcal{T}(B_r)$ implies $A_r \approx B_r$.*

Proof. We construct the following set

$$\mathcal{R} = \{ (A_r, B_r) \mid \mathcal{T}(A_r) \simeq\approx^e\simeq \mathcal{T}(B_r) \}.$$

and we will show that $\mathcal{R} \subseteq \approx$. Assume $\mathcal{T}(A_r) \simeq A \approx^e B \simeq \mathcal{T}(B_r)$.

1. First we prove that $A_r \Downarrow_a$ implies $B_r \Downarrow_a$. By Corollary 55 and Corollary 15, we know that $A \Downarrow_a$. Since $A \approx^e B$, we have $B \Downarrow_a$. By Corollary 15 and Corollary 59 we have that $B_r \Downarrow_a$.

2. Assume $A_r \Longrightarrow A'_r$, we need to show there exists B'_r such that $B_r \Longrightarrow B'_r$ and $(A'_r, B'_r) \in \mathcal{R}$. By Corollary 55 and Corollary 15, we know $A \Longrightarrow A'$ such that $\mathcal{T}(A'_r) \simeq A'$. Since $A \approx^e B$, we have $B \Longrightarrow B' \approx^e A'$. By Corollary 15 and Corollary 59, there exists B'_r such that $B_r \Longrightarrow B'_r$ and $\mathcal{T}(B'_r) \simeq B'$. Thus $(A'_r, B'_r) \in \mathcal{R}$.

3. For any evaluation context $\mathcal{C}_r$, in case the bound names are not pairwise distinct or different from the free ones, we can use α-conversion to $\mathcal{C}_r[A_r] = \mathcal{C}'_r[\varrho(A_r)], \mathcal{C}_r[B_r] = \mathcal{C}'_r[\varrho(B_r)]$. Then we will have a new sequence $\mathcal{T}(\varrho(A_r)) = \varrho(\mathcal{T}(A_r)) \simeq \varrho(A) \approx^e \varrho(B) \simeq \varrho(\mathcal{T}(B_r)) = \mathcal{T}(\varrho(B_r))$. Hence we assume that the bound names of $\mathcal{C}_r$ are not pairwise distinct or different from the free ones. Assume $\mathcal{T}(A_r) = \nu\widetilde{m}_1.(\sigma_1, \mathcal{P}_1)$ and $\mathcal{T}(B_r) = \nu\widetilde{m}_2.(\sigma_2, \mathcal{P}_2)$. By Lemma 53, we have $\mathcal{T}(\mathcal{C}_r) = \nu\widetilde{l}_1, \widetilde{l}_2.(\sigma, \mathcal{P}), \mathcal{T}(\mathcal{C}_r[A_r]) = \nu\widetilde{l}_1, \widetilde{l}_2, \widetilde{m}_1.(\sigma\sigma_1 \cup \sigma_{1\setminus\widetilde{x}}, \mathcal{P}\sigma_1 \cup \mathcal{P}_1)$ and $\mathcal{T}(\mathcal{C}_r[B_r]) = \nu\widetilde{l}_1, \widetilde{l}_2, \widetilde{m}_2.(\sigma\sigma_2 \cup \sigma_{2\setminus\widetilde{x}}, \mathcal{P}\sigma_1 \cup \mathcal{P}_2)$. Let $\mathcal{C} = \nu\widetilde{l}_1, \widetilde{l}_2.(\sigma, \mathcal{P})$. Hence $\mathcal{T}(\mathcal{C}_r[A_r]) = \mathcal{C}[\mathcal{T}(A_r)]_{\setminus\widetilde{x}}$ and $\mathcal{T}(\mathcal{C}_r[B_r]) = \mathcal{C}[\mathcal{T}(B_r)]_{\setminus\widetilde{x}}$. Since $\mathcal{C}[\mathcal{T}(A_r)] \approx^e \mathcal{C}[\mathcal{T}(B_r)]$, by Lemma 63, we have $\mathcal{C}[\mathcal{T}(A_r)]_{\setminus\widetilde{x}} \approx^e \mathcal{C}[\mathcal{T}(B_r)]_{\setminus\widetilde{x}}$. Hence $(\mathcal{C}_r[A_r], \mathcal{C}_r[B_r]) \in \mathcal{R}$.

Theorem 21. For two closed extended processes A_r and B_r in applied pi calculus [3],

1. A_r and B_r are labelled bisimilar iff $\mathcal{T}(A_r) \approx_l \mathcal{T}(B_r)$.

2. A_r and B_r are observationally equivalent iff $\mathcal{T}(A_r) \approx^e \mathcal{T}(B_r)$;

Proof. This is a direct corollary of Proposition 62, Proposition 65 and Proposition 66.