Trace Equivalence and Epistemic Logic to Express Security Properties

Kiraku Minami

Abstract

In process algebra, we can express security properties using an equivalence on processes. However, it is not clear which equivalence is the most suitable for the purpose. Indeed, several definitions of some properties are proposed. For example, the definition of privacy is not unique. This situation means that we are not certain how to express an intuitive security notion. Namely, there is a gap between an intuitive security notion and the formulation. Proper formalization is essential for verification, and our purpose is to bridge this gap.

In the case of the applied pi calculus, an outputted message is not explicitly expressed. This feature suggests that trace equivalence appropriately expresses indistinguishability for attackers in the applied pi calculus. By chasing interchanging bound names and scope extrusions, we prove that trace equivalence is a congruence. Therefore, a security property expressed using trace equivalence is preserved by an application of a context.

Moreover, we construct an epistemic logic for the applied pi calculus. We show that its logical equivalence agrees with trace equivalence. It means that trace equivalence is suitable in the presence of a non-adaptive attacker. Besides, we define several security properties using our epistemic logic.

Acknowledgments

I thank Professor Masahito Hasegawa for his invaluable support and guidance through the doctoral course. I also thank Professor Nobuko Yoshida for discussions.

I would also like to show my appreciation to my family for moral support.

Declaration

I declare that this thesis has been written by myself. The results in Chapters 3 and 4 are based on my published paper [28], and this thesis expands it to encompass proofs and details.

Contents

Chapter 1

Introduction

1.1 Background and Motivation

Information technology is indispensable to our daily lives in modern society, and many communication protocols are developed to transmit data securely. Verification of security properties of each protocol is essential, but it is not easy.

In the first place, how to formalize security notions is unclear. Various definitions of the same security property have been proposed. We will show an example later. One of our goals is to give a foundation for formalization. Besides, how to model communication and concurrency is also unclear; many such models have also been developed. In this work, we focus on process algebra because it allows us to handle parallel composition naturally.

In process algebra, information hiding such as secrecy is represented using an equivalence on processes. Various equivalences exist (cf. [35]), but which is the most suitable for expressing confidentiality is unclear. We show privacy as an example. Delaune et al. [12] defined privacy in electronic voting regarding the applied pi calculus [2], which extends the π-calculus [26, 27], as follows.

Definition 1 ([12, Definition 9]). *A voting protocol respects vote-privacy (or just privacy) if*

$$S[V_A\{^a/_v\}|V_B\{^b/_v\}] \approx_l S[V_A\{^b/_v\}|V_B\{^a/_v\}]$$

for all possible votes a and b.

V_A and V_B denote the voters containing the free variable v to express a vote. $\{^a/_v\}$ and $\{^b/_v\}$ are substitutions. S is an evaluation context and denotes other voters and authorities. Intuitively, when the protocol respects privacy, this definition states that an attacker cannot distinguish two situations where votes are swapped. Note that indistinguishability is expressed using labeled bisimilarity $\approx_l$ in this definition. Is it the most suitable? This question is nontrivial. Indeed, Chadha et al. [7] gave another definition and claimed that trace equivalence is more suitable regarding privacy than bisimilarity. We also claim that trace equivalence is more suitable to express security properties in the presence of a non-adaptive attacker. Similar arguments are not abundant in previous work.

In the applied pi calculus, a process can send not only names but also terms, but we do not explicitly express sent messages. We indirectly represent them using variables to refer messages. This feature enables us to handle cryptographic protocols naturally and suggests that trace equivalence means indistinguishability for attackers. This is because attackers whom we consider can observe only labeled transitions. We recall the syntax and semantics of the applied pi calculus in §2.3.

Both bisimilarity and trace equivalence on labeled transition systems (LTSs) are well studied. Bisimilarity for the applied pi calculus is also well studied. However, trace equivalence for the applied pi calculus (and other variants of the π-calculus) has not drawn as much attention as bisimilarity. This is perhaps because trace equivalence is much coarser than bisimilarity. However, security properties sometimes require that different processes are regarded as the same. For example, consider secrecy. We want to make two processes that send different messages indistinguishable. In this case, trace equivalence is enough, but bisimilarity is not always optimal because it is too strong. Bisimilarity requires that possible actions for each process are the same. However, a non-adaptive attacker cannot detect a difference in feasibility. Here, "non-adaptive" means that the attacker cannot control participants. Thereby, a fine equivalence such as bisimilarity is not always adequate. Bisimilarity is probably suitable for more powerful attackers.

In addition, bisimilarity is a congruence. That is, an application of a context preserve bisimilarity. This property enables us to handle processes algebraically, so this is also why bisimilarity draws attention.

Epistemic logic is often used to express confidentiality directly (e.g. [7, 24, 33]). For example, when a message M sent by an agent a is anonymous, we might say that an adversary cannot know who sent M. In epistemic logic, we can express it with a formula such as $\neg K\mathrm{Send}(a, M)$. This logical formulation is close to our intuition. Nevertheless, research into an epistemic logic for the applied pi calculus is not abundant.

In this thesis, we assume that attackers can observe only labeled transitions. Remarkably, they cannot observe what action participants can do. This assumption is natural because attackers in this thesis are non-adaptive. We also assume that an attacker can send messages to participants. In other words, we suppose that an attacker dominates the network.

This thesis is an extended version of [28].

1.2 Contributions

We show that trace equivalence for the applied pi calculus is a congruence in chapter 3. We introduce concurrent normal forms for proving it. Second, we give an epistemic logic that characterizes trace equivalence in chapter 4. Besides, we define security properties such as role interchangeability, secrecy [24, 33], and openness, which generalizes identity, using our epistemic logic. Moreover, we show that parallel composition does not generally preserve secrecy and openness.

On the other hand, trace equivalence characterizes total secrecy, so an application of contexts preserves it.

Our results suggest that trace equivalence is more suitable to express (non-probabilistic) security notions than bisimilarity.

Chapter 2

Process Algebra

2.1 CCS

Process algebras are a family of frameworks to model concurrent systems. CCS (Calculus of Communicating Systems) [25] is a typical example.

CCS processes are abstractions of concurrent systems. They are made of actions and co-actions. Formally, they are defined by Backus–Naur form.

$$P, Q ::= 0 \mid a.P \mid \overline{a}.P \mid \tau.P \mid P|Q \mid P + Q \mid P \setminus L \mid P[f] \mid \mathbf{fix}_j(\{X_i = P_i : i \in I\})$$

0 is a null process. It acts nothing. a is an action, and $\overline{a}$ is the co-action of a. a and $\overline{a}$ are complementary such as receiving and sending a message. τ is a silent action. An environment cannot observe a τ action.

$P|Q$ is a parallel composition of P and Q. This composition does not force them to synchronize. $P + Q$ is a nondeterministic branch.

$\setminus L$ is a restriction operator, where L is a set of labels. If a is in L, $P \setminus L$ cannot act a nor $\overline{a}$. $[f]$ is relabeling operator.

$\mathbf{fix}_j$ is a fixed-point operator. $\mathbf{fix}_j(\{X_i = P_i : i \in I\})$ is j-th component of a fixed-point of $\{X_i = P_i : i \in I\}$.

Their behavior is formally defined by structural operational semantics. We show representative rules.

$$\frac{}{\alpha.P \xrightarrow{\alpha} P} \qquad \frac{P \xrightarrow{a} P' \quad Q \xrightarrow{\overline{a}} Q'}{P|Q \xrightarrow{\tau} P'|Q'}$$

$\alpha.P$ acts α at first and behaves like P. Complementary actions a and $\overline{a}$ can be simultaneously done, and then an environment cannot observe it.

For example, a vending machine is represented as a below process VM.

$$VM = \mathbf{fix}(X = \ coin.button.\overline{juice}.X)$$

This vending machine sells only juice. The action *coin* expresses receiving a coin. When the button is pushed, the process does *button*. $\overline{juice}$ is providing a cup of juice.

On the other hand, a customer is also represented as a process.

$$C = \overline{coin}.\overline{button}.juice.0$$

The action $\overline{coin}$ means that the customer inserts a coin. $coin$ and $\overline{coin}$ are complementary. Processes can do complementary actions simultaneously.

$$\begin{aligned}
VM \mid C &= coin.button.\overline{juice}.VM \mid \overline{coin}.\overline{button}.juice.0 \\
&\longrightarrow button.\overline{juice}.VM \mid \overline{button}.juice.0 \\
&\longrightarrow \overline{juice}.VM \mid juice.0 \\
&\longrightarrow VM \mid 0
\end{aligned}$$

2.2 The π-calculus

The π-calculus [26, 27] is a CCS-style process algebra. π-calculus processes can interchange channels as messages, so we can express mobility using the π-calculus. Processes can use channels received from other processes to communicate. That is, an interaction between processes can change communication topology.

π-calculus processes are defined as below:

$$P, Q ::= 0 \mid x(y).P \mid \overline{x}y.P \mid \tau.P \mid P|Q \mid P + Q \mid (x)P \mid [x = y]P \mid A(x_1, ..., x_n)$$

$x(y).P$ receives a channel on x and substitutes it for y. $\overline{x}y.P$ sends a channel y on x. (x) is a restriction operator. $(x)P$ cannot use the channel x to communicate with an environment but can use it in P. We can consider that $(x)P$ creates a new channel x and behaves like P. The environment does not know x, so it cannot use x. $[x = y]$ is a match operator. $[x = y]P$ behaves like P when $x = y$ holds. Otherwise, it behaves like 0. A is an agent identifier. The defining equation $A(x_1, ..., x_n) \overset{\text{def}}{=} P$ decides behavior of A.

We show representative rules.

$$\frac{}{\overline{x}y.P \xrightarrow{\overline{x}y} P} \qquad \frac{z \notin \mathrm{fn}((y)P)}{x(y).P \xrightarrow{x(z)} P\{z/y\}} \qquad \frac{P \xrightarrow{\overline{x}y} P' \quad Q \xrightarrow{x(z)} Q'}{P|Q \xrightarrow{\tau} P'|Q'\{y/z\}}$$

$$\frac{P \xrightarrow{\overline{x}y} P' \quad y \neq x \quad w \notin \mathrm{fn}((y)P)}{(y)P \xrightarrow{\overline{x}(w)} P'\{w/y\}} \qquad \frac{P \xrightarrow{\overline{x}(w)} P' \quad Q \xrightarrow{x(w)} Q'}{P|Q \xrightarrow{\tau} (w)(P'|Q')\}}$$

The first rule expresses sending a channel y. The second rule expresses receiving a channel. Notice that a scheme of late instantiation is adopted. We regard z as a placeholder in the second rule. It is instantiated only when internal communication happens. In fact, Q receives a concrete channel y from P in the third rule.

In the fourth rule, $(y)P$ sends the bound name y to an environment. This rule allows α-conversion of y to avoid a collision with a free name. The fifth rule happens a scope extrusion. The scope of the bound name w grows through the transition. The fourth rule is named OPEN, and the fifth rule is named CLOSE.

We illustrate a change of communication topology.

$$\begin{aligned} \overline{x}y.0 \mid x(z).\overline{z}w.0 \mid y(z).0 &\xrightarrow{\tau} 0 \mid \overline{y}w.0 \mid y(z).0 \\ &\xrightarrow{\tau} 0 \mid 0 \mid 0 \end{aligned}$$

At first, the first process sends a channel y to the second process. Then, the second process can use y to send a channel w to the third process. The second process could not communicate with the third process at the start of these transitions.

These processes can make other transitions.

$$\begin{aligned} \overline{x}y.0 \mid x(z).\overline{z}w.0 \mid y(z).0 &\xrightarrow{\tau} 0 \mid \overline{y}w.0 \mid y(z).0 \\ &\xrightarrow{\overline{y}w} 0 \mid 0 \mid y(z).0 \end{aligned}$$

The second process sends w to an environment to use the channel y received from the first process.

We also illustrate a scope extrusion.

$$\begin{aligned} (y)(\overline{x}y.y(v).0) \mid x(z).\overline{z}w.0 \mid \overline{y}w.0 &\xrightarrow{\tau} (y)(y(v).0 \mid \overline{y}w.0) \mid \overline{y}w.0 \\ &\xrightarrow{\tau} 0 \mid 0 \mid \overline{y}w.0 \end{aligned}$$

The first process sends a bound name y to the second process. Then, a scope extrusion happens, and the second process uses the bound name y. Although the third process also has the name y, it cannot communicate with the first process. This is because the scope of the bound name y does not cover the third process.

2.3 The Applied Pi Calculus

The applied pi calculus [2] is an extension of the π-calculus. Functions and equations are added to the π-calculus. In addition, we distinguish between names and variables. Thus we can handle cryptographic protocols naturally using the applied pi calculus to assume that perfect encryption exists.

2.3.1 Syntax

Let Σ be a signature equipped with an equational theory. Terms are made from names, variables, and function symbols. We say that a term is ground when it contains no variables.

For example, symmetric-key encryption $enc(x, y)$, decryption $dec(x, y)$, and hash function $h(x)$ are often used. enc and dec are function symbols with arity 2, and h is a function symbol with arity 1. $enc(x, y)$ expresses a ciphertext of x

obtained to use a key y, and $dec(x, y)$ expresses decryption of x using a key y. enc and dec have an equation $dec(enc(x, y), y) = x$. On the other hand, h has no equations.

We recall the syntax of the applied pi calculus. Here, M, N... range over terms, while n on names and x on variables.

$$P, Q ::= 0 \mid \overline{M}\langle N\rangle.P \mid M(x).P \mid \nu n.P \mid \\ \text{if } M = N \text{ then } P \text{ else } Q \mid P + Q \mid P|Q \mid !P$$

$$A, B ::= P \mid \nu n.A \mid \nu x.A \mid A|B \mid \{^M/_x\}$$

$P, Q, ...$ are *plain processes*. ν is a binding operator. Bound names are often interpreted as nonces, secret keys, and so on. In other words, νn is regarded as creating a new value n. $\nu n.A$ does not require that n occurs in A, but $\nu x.A$ requires that x occurs in A. νx hides x from an environment. $|$ is a parallel composition operator. Parallel composition is asynchronous. $+$ is a nondeterministic choice operator. ! is a replication operator. That is, $!P$ is unbounded copies of P. We often omit .0 at the end of an expression. For example, we abbreviate $\overline{a}\langle b\rangle.0|c(x).0$ to $\overline{a}\langle b\rangle|c(x)$.

Plain processes are similar to π-calculus processes, but they are not the same. A π-calculus process can send only a name. On the other hand, an applied pi calculus process can send a term. Besides, a channel consists of a term. An object of an input prefix is a variable, so names do not change while the process runs. This invariance is also a difference from π-calculus.

$A, B, ...$ are *extended processes*. We call $\{^M/_x\}$ an active substitution. This notion is peculiar to the applied pi calculus. An active substitution $\{^M/_x\}$ substitutes M for x in a neighbor process. $\nu u.\nu v.A$ is often abbreviated to $\nu uv.A$. We say that A is finite if A contains no replications.

fn(A) and bn(A) denote the sets of free names and bound names of a process A, respectively. fv and bv are similar to them. If $\text{fn}(A) \cap \text{bn}(A) = \emptyset$ and no bound names are restricted twice, we say that A is name-distinct. Variable-distinctness is defined similarly. n(M) denotes the set of names that appear in a term M. v(M) is similar to it.

The *domain* dom(A) of an extended process A is inductively defined below. If variables in neighbor concurrently running processes are in dom(A), the process A affects those variables. If $\text{fv}(A) = \text{dom}(A)$, we say that A is closed.

$$\text{dom}(P) = \emptyset,\ \text{dom}(\nu n.A) = \text{dom}(A),\ \text{dom}(\nu x.A) = \text{dom}(A) \setminus \{x\}, \\ \text{dom}(A|B) = \text{dom}(A) \cup \text{dom}(B),\ \text{dom}(\{^M/_x\}) = \{x\}$$

The domain of A is the set of references to messages that A sends to an environment.

2.3.2 Semantics

A *context* is an expression containing one hole. Contexts are given by the syntax below:

$$\begin{aligned}C[\_] ::= \;& \_ \mid \overline{M}\langle N\rangle.C[\_] \mid M(x).C[\_] \mid \nu n.C[\_]\\ & \mid \text{ if } M = N \text{ then } C[\_] \text{ else } Q \mid \text{ if } M = N \text{ then } P \text{ else } C[\_]\\ & \mid C[\_] + Q \mid P + C[\_] \mid C[\_]|B \mid A|C[\_] \mid !C[\_] \mid \nu x.C[\_]\end{aligned}$$

$C[A]$ denotes the result of replacing the hole with the process A if it is syntactically correct. For example, $\overline{M}\langle N\rangle.C[\{^{a}/_{x}\}]$ is not correct.

An *evaluation context* is a context whose hole is neither under a replication, a conditional branch, a nondeterministic branch, nor an action prefix. That is, they are given by the syntax below:

$$E[\_] ::= \_ \mid \nu n.E[\_] \mid E[\_]|B \mid A|E[\_] \mid \nu x.E[\_]$$

For example, $P \mid \_$ and $\nu n.\_$ are evaluation contexts, but $\overline{M}\langle N\rangle.\_$ is not an evaluation context.

Structural equivalence $\equiv$ is the smallest equivalence relation on extended processes that is closed under applications of evaluation contexts and α-conversion, such that:

$$\begin{aligned} A|0 &\equiv A\\ (A|B)|C &\equiv A|(B|C)\\ A|B &\equiv B|A\\ (\nu u.A)|B &\equiv \nu u.(A|B) \text{ if } u \notin \mathrm{fn}(B) \cup \mathrm{fv}(B)\\ \nu u.\nu v.A &\equiv \nu v.\nu u.A\\ !P &\equiv P|!P\\ (P+Q)+R &\equiv P+(Q+R)\\ P+Q &\equiv Q+P\\ \nu x.\{M/x\} &\equiv 0\\ A|\{M/x\} &\equiv A\{M/x\}|\{M/x\}\\ \{M/x\} &\equiv \{N/x\} \text{ if } \Sigma \vdash M = N\end{aligned}$$

where $[^{M}/_{x}]$ is a capture-avoiding substitution. That is, $A[^{M}/_{x}]$ is the result of replacing x in A with M. Note that structurally equivalent processes do not always have the same free names.

The second from the last represents how an active substitution $\{^{M}/_{x}\}$ acts.

In the original applied pi calculus, $\nu n.0 \equiv 0$ is also required, but we removed it. This is because we want to keep the number of bound names.

Definition 2. *Internal reduction $\rightarrow$ is the smallest relation on extended processes closed under structural equivalence and applications of evaluation contexts, such* *that:*

$$\begin{aligned}
\text{if } M = N \text{ then } P \text{ else } Q \to P &\quad \text{when } \Sigma \vdash M = N\\
\text{if } M = N \text{ then } P \text{ else } Q \to Q &\quad \text{when } \Sigma \not\vdash M = N\\
P + Q \to P &\\
\overline{M}\langle N\rangle.P \mid M(x).Q \to P \mid Q[{}^{N}/_{x}], &
\end{aligned}$$

where terms M and N in the second rule are ground.

Internal reductions are silent. In other words, an environment cannot notice that a process makes an internal reduction.

Note that the original version of the applied pi calculus does not contain a nondeterministic choice operator $+$. We adopt the definition in [12]. This operator does not increase the expressive power of the original version of the applied pi calculus because $P + Q$ and $\nu c.(\overline{c}\langle s\rangle|c(x).P|c(x).Q)$ are bisimilar (where c and x do not appear in P nor Q). Bisimilarity is the most strong behavioral equivalence. We will give the definition later.

The last line represents synchronous communication on a channel M. We emphasize that an environment cannot observe what is interchanged.

Next, we recall labeled semantics and requisite equivalence relations.

$$\frac{}{M(x).P \xrightarrow{M(N)} P[{}^{N}/_{x}]} \qquad \frac{x \notin \mathrm{fv}(\overline{M}\langle N\rangle.P)}{\overline{M}\langle N\rangle.P \xrightarrow{\nu x.\overline{M}\langle x\rangle} P|\{{}^{N}/_{x}\}}$$

$$\frac{A \xrightarrow{\alpha} A' \quad u \text{ does not appear in } \alpha.}{\nu u.A \xrightarrow{\alpha} \nu u.A'}$$

$$\frac{A \xrightarrow{\alpha} A' \quad \mathrm{bv}(\alpha) \cap \mathrm{fv}(B) = \emptyset}{A|B \xrightarrow{\alpha} A'|B} \qquad \frac{A \equiv A' \quad A' \xrightarrow{\alpha} B' \quad B' \equiv B}{A \xrightarrow{\alpha} B}$$

The first rule expresses an input. Note that a scheme of early instantiation is adopted. This scheme is different from the π-calculus.

In the π-calculus, an environment is regarded as other processes. On the other hand, we regard an environment as an attacker in the applied pi calculus. The attacker can send arbitrary possible messages, so early instantiation is suitable.

The second rule represents an output. Note that an active substitution $\{{}^{N}/_{x}\}$ is generated, and the term N does not appear in the action label $\nu x.\overline{M}\langle x\rangle$. That is, the attacker cannot know the structure of the message N in general.

No rules correspond to OPEN and CLOSE, but scope extrusions can happen by internal reductions. For example,

$$\begin{aligned}
\nu b.(\overline{a}\langle b\rangle.b(x)) \mid a(y).\overline{y}\langle c\rangle \mid \overline{b}\langle d\rangle &\longrightarrow \nu b.(b(x) \mid \overline{b}\langle c\rangle) \mid \overline{b}\langle d\rangle\\
&\longrightarrow 0 \mid 0 \mid \overline{b}\langle d\rangle
\end{aligned}$$

A *frame* is an extended process generated from 0 and active substitutions using restriction and parallel composition. fr(A) denotes a process obtained by replacing plain processes in A with 0, and we call it a frame of A. We can consider that $\mathrm{fr}(A)$ is a list of messages exported by A.

μ denotes an action. We define $\Longrightarrow$ as the transitive reflexive closure of $\longrightarrow$, and $\overset{\alpha}{\Longrightarrow}$ as $\Longrightarrow\xrightarrow{\alpha}\Longrightarrow$. $\overset{\mu}{\Longrightarrow}$ is the former when μ is silent and is the latter otherwise.

Definition 3. *$(M = N)\varphi \overset{\mathrm{def}}{\Leftrightarrow} \mathrm{v}(M) \cup \mathrm{v}(N) \subseteq \mathrm{dom}(\varphi)$ and $M\sigma = N\sigma$ where $\varphi \equiv \nu\widetilde{n}.\sigma$ and $\widetilde{n} \cap \mathrm{n}(M, N) = \emptyset$ for some names $\widetilde{n}$ and active substitutions σ.*

$(M = N)\varphi$ means that an attacker cannot distinguish M and N using φ.

Definition 4. *The static equivalence on closed frames is given by*

$$\varphi \approx_s \psi \overset{\mathrm{def}}{\Leftrightarrow} \mathrm{dom}(\varphi) = \mathrm{dom}(\psi) \text{ and } \forall M, N; (M = N)\varphi \Leftrightarrow (M = N)\psi$$

for closed frames φ and ψ. The static equivalence on closed processes is given by

$$A \approx_s B \overset{\mathrm{def}}{\Leftrightarrow} \mathrm{fr}(A) \approx_s \mathrm{fr}(B)$$

for closed processes A and B.

Static equivalence means that an attacker has the same information about which terms are equal.

Definition 5. *A trace* **tr** *is a finite derivation*

$$\mathbf{tr} = A_0 \overset{\mu_1}{\Longrightarrow} \dots \overset{\mu_n}{\Longrightarrow} A_n$$

such that every A_i is closed and $\mathrm{fv}(\mu_i) \subseteq \mathrm{dom}(A_{i-1})$ for all i. If A_n can perform no actions, the trace **tr** *is said to be complete or maximal.*

Given a trace **tr***, let $\mathbf{tr}[i]$ be its i-th process A_i, and $\mathbf{tr}[i,j]$ be the trace*

$$A_i \overset{\mu_{i+1}}{\Longrightarrow} \dots \overset{\mu_j}{\Longrightarrow} A_j$$

where $0 \leq i \leq j \leq n$. The length of the trace **tr** *is denoted by $|\mathbf{tr}| = n$.*

If each $\overset{\mu_i}{\Longrightarrow}$ accord with $\xrightarrow{\mu_i}$, we say that **tr** *is full.*

An attacker can observe traces but cannot observe the structure of the process.

Definition 6. *Let* **tr** *be a trace $A_0 \overset{\mu_1}{\Longrightarrow} \dots \overset{\mu_n}{\Longrightarrow} A_n$ and $\mathbf{tr}'$ be a trace $B_0 \overset{\mu'_1}{\Longrightarrow} \dots \overset{\mu'_m}{\Longrightarrow} B_m$. Static equivalence between* **tr** *and $\mathbf{tr}'$ is defined below:*

$$\mathbf{tr} \sim_t \mathbf{tr}' \overset{\mathrm{def}}{\Leftrightarrow} n = m \text{ and } \mu_i = \mu'_i \text{ and } A_i \approx_s B_i \text{ for all } i.$$

An attacker cannot distinguish statically equivalent traces.

$\mathrm{tr}(A)$ denotes a set of traces of A, and $\mathrm{tr}_{\max}(A)$ denotes a set of maximal traces of A.

Definition 7. *Let A and B be closed processes.*

$$A \subseteq_t B \overset{\text{def}}{\Leftrightarrow} \forall \mathbf{tr} \in \text{tr}(A)\ \exists \mathbf{tr'} \in \text{tr}(B) \text{ s.t. } \mathbf{tr} \sim_t \mathbf{tr'},$$
$$A \approx_t B \overset{\text{def}}{\Leftrightarrow} A \subseteq_t B \textit{ and } B \subseteq_t A.$$

Let A and B be (not necessarily closed) two processes. Let σ be a map that maps a variables in $(\text{fv}(A)\backslash\text{dom}(A))\cup(\text{fv}(B)\backslash\text{dom}(B))$ to ground terms. When $A\sigma \subseteq_t B\sigma$ holds for every such σ, we also denote it as $A \subseteq_t B$. $A \approx_t B$ is similarly defined.

$A \subseteq_t B$ means that each trace of A is imitated by some trace of B.

We later show that non-adaptive active attackers cannot distinguish trace equivalent processes.

Problem 1.
Input: *Closed extended processes A, B, and a trace $\mathbf{tr} \in \text{tr}(B)$.*
Question: *Does there exist a trace $\mathbf{tr'} \in \text{tr}(A)$ such that $\mathbf{tr} \sim_t \mathbf{tr'}$?*

Proposition 1. *There are signatures for which Problem 1 is undecidable, even though the input processes are restricted to plain processes.*

Proof. It is proved in [3, Proposition 5] that static equivalence is undecidable in general. We reduce the decision problem for static equivalence to Problem 1.

Let φ and ψ be closed frames. We assume that $\text{dom}(\varphi) = \text{dom}(\psi)$.

Let $\varphi = \nu\widetilde{n}.\{{}^{M_1}/_{x_1}, ..., {}^{M_l}/_{x_l}\}, \psi = \nu\widetilde{m}.\{{}^{N_1}/_{x_1}, ..., {}^{N_l}/_{x_l}\}$.

Let $P = \nu\widetilde{n}.\overline{a}\langle M_1\rangle...\overline{a}\langle M_l\rangle, Q = \nu\widetilde{m}.\overline{a}\langle N_1\rangle...\overline{a}\langle N_l\rangle$, where $a \notin \widetilde{n} \cup \widetilde{m}$.

Let

$$\mathbf{tr} = P \overset{\nu x_1.\overline{a}\langle x_1\rangle}{\longrightarrow} \nu\widetilde{n}.(\overline{a}\langle M_2\rangle...\overline{a}\langle M_l\rangle|\{{}^{M_1}/_{x_1}\}) \overset{\nu x_2.\overline{a}\langle x_2\rangle}{\longrightarrow} ... \overset{\nu x_l.\overline{a}\langle x_l\rangle}{\longrightarrow} \nu\widetilde{n}.\{{}^{M_1}/_{x_1}, ..., {}^{M_l}/_{x_l}\}.$$

We prove that $\varphi \approx_s \psi$ iff there exists a trace $\mathbf{tr'} \in \text{tr}(A)$ such that $\mathbf{tr} \sim_t \mathbf{tr'}$.

A trace $\mathbf{tr'} \in \text{tr}(Q)$ whose actions correspond to $\mathbf{tr}$ is the only below:

$$Q \overset{\nu x_1.\overline{a}\langle x_1\rangle}{\longrightarrow} ... \overset{\nu x_l.\overline{a}\langle x_l\rangle}{\longrightarrow} \nu\widetilde{m}.\{{}^{N_1}/_{x_1}, ..., {}^{N_l}/_{x_l}\}.$$

We assume that $\mathbf{tr} \sim_t \mathbf{tr'}$. Then, $\nu\widetilde{n}.\{{}^{M_1}/_{x_1}, ..., {}^{M_l}/_{x_l}\} \approx_s \nu\widetilde{m}.\{{}^{N_1}/_{x_1}, ..., {}^{N_l}/_{x_l}\}$, so $\Leftarrow$ holds.

Next, we assume that $\varphi \approx_s \psi$. Then,

$$\text{fr}(\mathbf{tr}[i]) \equiv \nu x_{i+1}...x_l.\varphi$$
$$\text{fr}(\mathbf{tr'}[i]) \equiv \nu x_{i+1}...x_l.\psi.$$

Therefore, it follows that $\mathbf{tr} \sim_t \mathbf{tr'}$. Namely, $\Rightarrow$ also holds. □

Proposition 2. *If the static equivalence on a signature Σ is decidable, Problem 1 is decidable for finite A.*

Proof. The number of traces in $\mathrm{tr}(A)$ whose actions correspond to ones in $\mathbf{tr}$ is finite because every action yields finitely many processes.

We only have to check whether each process is statically equivalent to the correspondent process in $\mathbf{tr}$ for such traces. $\square$

In general, trace equivalence is undecidable. However, if processes contain no replications, and the equational theory on Σ is a subterm convergent destructor rewriting system, then trace equivalence is coNEXP complete [8].

Labeled bisimilarity is also usually used. If two processes are bisimilar, they can imitate each other.

Definition 8. *A binary relation $\mathcal{R}$ between closed extended processes is called a labeled simulation if and only if whenever $A\mathcal{R}B$,*

1. $A \approx_s B$
2. $$[A \xrightarrow{\mu} A' \textit{ and } A' : \text{closed } \textit{and } \mathrm{fv}(\mu) \subseteq \mathrm{dom}(A)]$$ $$\Rightarrow \exists B' \text{ s.t. } B \overset{\mu}{\Longrightarrow} B' \textit{ and } A\mathcal{R}B$$

If both $\mathcal{R}$ and $\mathcal{R}^{-1}$ are labeled simulations, we say that $\mathcal{R}$ is a labeled bisimulation. We call the largest labeled bisimulation a labeled bisimilarity.

It is proved in [1, Theorem 4.1] that the labeled bisimilarity is closed by applying an evaluation context.

Labeled bisimilarity requires two processes to have the same feasibility. However, the condition may be too strong if an attacker cannot control participants directly. We show that trace equivalence is more suitable than labeled bisimilarity in Chapter 4.

Chapter 3

Congruency of Trace Equivalence

Congruence is a crucial notion for concurrent systems. We say that an equivalence relation $\approx$ between systems is a congruence if $A \approx B$ implies $C[A] \approx C[B]$ for any system A, B, and any context $C[\_]$. This condition may look natural, but whether it is satisfied is not evident in many cases.

Theorem 1, which states that trace equivalence for the applied pi calculus is a congruence, is our main result. Even though trace equivalence for the π-calculus is not a congruence, Theorem 1 holds. This is ascribed to the difference between the π-calculus and the applied pi calculus, namely, to the fact that names and variables are distinguished in the applied pi calculus. This is why adding an input prefix does not break trace equivalence. Besides, a scheme of late instantiation for an input transition is used in the π-calculus, so parallel composition may break trace equivalence. On the other hand, a scheme of early instantiation is used in the applied pi calculus. This scheme enables us to decompose a trace of a parallel composed process into traces of component processes.

Example 1. *We consider the π-calculus and put*

$$P = z(z') \mid \overline{y}y'.\overline{w}w' \text{ and } Q = z(z').\overline{y}y'.\overline{w}w' + \overline{y}y'.z(z').\overline{w}w' + \overline{y}y'.\overline{w}w'.z(z').$$

Then, $x(z).P$ and $x(z).Q$ are trace equivalent because y cannot be substituted into z, but $\overline{x}y \mid x(z).P$ and $\overline{x}y \mid x(z).Q$ are not trace equivalent. The former process can make the transition below:

$$\overline{x}y \mid x(z).P \xrightarrow{\tau} y(z') \mid \overline{y}y'.\overline{w}w' \xrightarrow{\tau} \overline{w}w' \xrightarrow{\overline{w}w'} 0$$

However, $\overline{x}y \mid x(z).Q$ cannot do $\overline{w}w'$ before doing $\overline{y}y'$.

On the other hand, $x(z).P$ and $x(z).Q$ are not trace equivalent in the applied pi calculus (when we regard x, y, y', w, and w' as names). This is because instantiation is early. $x(z).P$ can make the transition below:

$$x(z).P \xrightarrow{x(y)} y(z') \mid \overline{y}y'.\overline{w}w' \to \overline{w}w' \xrightarrow{\nu v.\overline{w}\langle v\rangle} \{^{w'}/_{v}\}$$

However, $x(z).Q$ cannot do $\nu v.\overline{w}\langle v\rangle$ before doing $\nu u.\overline{y}\langle u\rangle$ after doing $x(y)$.

Abadi et al. [2] defined partial normal forms to prove that labeled bisimilarity is closed by applications of closing evaluation contexts. They gave operational semantics on partial normal forms and classified transitions between ordinal processes into six cases using partial normal forms.

To prove the following theorem, we use partial normal forms and define *concurrent normal forms* of traces. Moreover, transitions in a concurrent normal trace have to be particular forms.

Abadi et al. [2] studied decomposition and composition of reductions on partial normal forms. We study decomposition and composition of concurrent normal traces.

Theorem 1. $\approx_t$ *is a congruence.*

The proof is very long and complicated, so we only present an outline of our proof for the proposition below. Other cases are easy. The proof is given in Appendix A.

Proposition 3. $A \approx_t B \Rightarrow A|C \approx_t B|C$.

Proof outline. First, we define concurrent normal forms. A concurrent normal form is a particular form of a trace of a parallel composed process. A concurrent normal trace captures changes of scopes of bound names. Each process in a concurrent normal trace is of the form $\nu\widetilde{r}\widetilde{s}.(\nu\widetilde{x}.(\sigma|P)\rho \mid \nu\widetilde{y}.(\rho|Q)\sigma)$, where σ and ρ are (active) substitutions. Terms sent by the left process are accumulated in σ, and bound names sent by the left process are accumulated in $\widetilde{s}$. Variables to refer messages sent by synchronous communication are bound. Symmetric cases are similar.

Second, for any trace t of $A|C$, we prove that there exists a concurrent normal trace t' of $A|C$ such that $t \sim_t t'$. Thus, we have to consider only concurrent normal traces.

Third, given a concurrent normal trace $\mathbf{tr}$ of $A|C$, we prove that we can construct traces of A and C which each process in them is of the form $\nu\widetilde{s}.(\sigma|P)\rho$ or $\nu\widetilde{r}.(\rho|Q)\sigma$.

Finally, we take a trace $\mathbf{tr}'$ of B which is statically equivalent to the extracted trace of A as the above, combine it with $\mathbf{tr}'$, and prove that the result is statically equivalent to the given trace $\mathbf{tr}$. □

Example 2. *Let h be a unary function symbol which has no equations.*
$\nu m.a(x).\overline{x}\langle m\rangle \approx_t \nu m.a(x).\overline{x}\langle h(m)\rangle$ *holds. Then,*

$$\nu m.a(x).\overline{x}\langle m\rangle \mid \nu n.\overline{a}\langle n\rangle.n(y).\overline{b}\langle y\rangle \approx_t \nu m.a(x).\overline{x}\langle h(m)\rangle \mid \nu n.\overline{a}\langle n\rangle.n(y).\overline{b}\langle y\rangle$$

is shown as follows.

We arbitrarily take a trace $\mathbf{tr}$ *of the left-hand side. We consider*

$$\begin{aligned}
&\mathbf{tr} : \nu m.a(x).\overline{x}\langle m\rangle \mid \nu n.\overline{a}\langle n\rangle.n(y).\overline{b}\langle y\rangle\\
&\xrightarrow{\nu z.\overline{a}\langle z\rangle} \nu m.a(x).\overline{x}\langle m\rangle \mid \nu n.(n(y).\overline{b}\langle y\rangle \mid \{^{n}/_{z}\})\\
&\xrightarrow{a(z)} \nu mn.(\overline{n}\langle m\rangle \mid n(y).\overline{b}\langle y\rangle \mid \{^{n}/_{z}\})\\
&\longrightarrow \nu mn.(\overline{b}\langle m\rangle \mid \{^{n}/_{z}\})\\
&\xrightarrow{\nu w.\overline{b}\langle w\rangle} \nu mn.\{^{n}/_{z}, ^{m}/_{w}\}
\end{aligned}$$

as an example. We transform it into a concurrent normal form.

$$\begin{aligned}
&\mathbf{tr'} : \nu m.a(x).\overline{x}\langle m\rangle \mid \nu n.\overline{a}\langle n\rangle.n(y).\overline{b}\langle y\rangle\\
&\xrightarrow{\nu z.\overline{a}\langle z\rangle} \nu n.((\nu m.a(x).\overline{x}\langle m\rangle)[{}^{n}/_{z}] \mid n(y).\overline{b}\langle y\rangle \mid \{{}^{n}/_{z}\})\\
&\xrightarrow{a(z)} \nu n.((\nu m.\overline{z}\langle m\rangle)[{}^{n}/_{z}] \mid n(y).\overline{b}\langle y\rangle \mid \{{}^{n}/_{z}\})\\
&\longrightarrow \nu nm.((\nu v.\{{}^{m}/_{v}\})[{}^{n}/_{z}] \mid (\overline{b}\langle v\rangle \mid \{{}^{n}/_{z}\})[{}^{m}/_{v}])\\
&\xrightarrow{\nu w.\overline{b}\langle w\rangle} \nu nm.((\nu v.\{{}^{m}/_{v}\})[{}^{n}/_{z}, {}^{v}/_{w}] \mid \{{}^{n}/_{z}, {}^{v}/_{w}\}[{}^{m}/_{v}])
\end{aligned}$$

Next, we decompose it into traces of component processes.

$$\begin{aligned}
&\mathbf{tr_1} : \nu m.a(x).\overline{x}\langle m\rangle\\
&\xrightarrow{a(n)} (\nu m.\overline{z}\langle m\rangle)[{}^{n}/_{z}]\\
&\xrightarrow{\nu v.\overline{n}\langle v\rangle} (\nu m.\{{}^{m}/_{v}\})[{}^{n}/_{z}]
\end{aligned}
\qquad
\begin{aligned}
&\mathbf{tr_2} : \nu n.\overline{a}\langle n\rangle.n(y).\overline{b}\langle y\rangle\\
&\xrightarrow{\nu z.\overline{a}\langle z\rangle} \nu n.(n(y).\overline{b}\langle y\rangle \mid \{{}^{n}/_{z}\})\\
&\xrightarrow{z(m)} \nu n.(\overline{b}\langle v\rangle \mid \{{}^{n}/_{z}\})[{}^{m}/_{v}]\\
&\xrightarrow{\nu w.\overline{b}\langle w\rangle} \nu n.\{{}^{n}/_{z}, {}^{v}/_{w}\}[{}^{m}/_{v}]
\end{aligned}$$

We can take a trace of $\nu m.a(x).\overline{x}\langle h(m)\rangle$ which is statically equivalent to the former because $\nu m.a(x).\overline{x}\langle m\rangle \approx_t \nu m.a(x).\overline{x}\langle h(m)\rangle$ holds.

$$\begin{aligned}
&\mathbf{tr_3} : \nu m.a(x).\overline{x}\langle h(m)\rangle\\
&\xrightarrow{a(n)} (\nu m.\overline{z}\langle h(m)\rangle)[{}^{n}/_{z}]\\
&\xrightarrow{\nu v.\overline{n}\langle v\rangle} (\nu m.\{{}^{h(m)}/_{v}\})[{}^{n}/_{z}]
\end{aligned}$$

Finally, we compose $\mathbf{tr_2}$ and $\mathbf{tr_3}$ and obtain a desired trace $\mathbf{tr}_4$.

$$\begin{aligned}
&\mathbf{tr_4} : \nu m.a(x).\overline{x}\langle h(m)\rangle \mid \nu n.\overline{a}\langle n\rangle.n(y).\overline{b}\langle y\rangle\\
&\xrightarrow{\nu z.\overline{a}\langle z\rangle} \nu n.((\nu m.a(x).\overline{x}\langle h(m)\rangle)[{}^{n}/_{z}] \mid n(y).\overline{b}\langle y\rangle \mid \{{}^{n}/_{z}\})\\
&\xrightarrow{a(z)} \nu n.((\nu m.\overline{z}\langle h(m)\rangle)[{}^{n}/_{z}] \mid n(y).\overline{b}\langle y\rangle \mid \{{}^{n}/_{z}\})\\
&\longrightarrow \nu nm.((\nu v.\{{}^{h(m)}/_{v}\})[{}^{n}/_{z}] \mid (\overline{b}\langle v\rangle \mid \{{}^{n}/_{z}\})[{}^{h(m)}/_{v}])\\
&\xrightarrow{\nu w.\overline{b}\langle w\rangle} \nu nm.((\nu v.\{{}^{h(m)}/_{v}\})[{}^{n}/_{z}, {}^{v}/_{w}] \mid \{{}^{n}/_{z}, {}^{v}/_{w}\}[{}^{h(m)}/_{v}])
\end{aligned}$$

□

We briefly compare trace equivalence and may-testing equivalence [9] before moving on to the next chapter.

Definition 9.

$$A \Downarrow a \text{ if and only if } \exists E, M, P \text{ s.t. } A \Rightarrow E[\overline{a}\langle M\rangle.P]$$

$A \Downarrow a$ means that A can use the channel a to send a term after several reductions.

Definition 10. *A test (C, c) is a pair of an evaluation context C and a channel c. Let A be a closed extended process. We assume that A does not contain c. When $C[A] \Downarrow c$, we say that A passes the test.*

Definition 11. *Two closed extended processes A and B are may-testing equivalent if and only if they pass the same tests.*

Cheval et al. [9] proved that trace equivalence is equivalent to may-testing equivalence for image-finite processes. Here, image-finite means that each sequence of actions generates finitely many frames up to static equivalence.

May-testing equivalence is a congruence, so trace equivalence is also a congruence. On the other hand, taking all processes into account, may-testing equivalence does not imply trace equivalence. Indeed, they gave a concrete counterexample. Thus, we cannot use the same technique.

We can regard contexts as *concrete* attackers. Note that how to attack is decided before the target process operates. This is why may-testing equivalence is coarser than trace equivalence.

Chapter 4

An Epistemic Logic for the Applied Pi Calculus

We omit several proofs in this chapter. They are given in Appendix B.

4.1 Syntax

We propose an epistemic logic for the applied pi calculus. Our logic is inspired by [7], but it is a bit different. We give the syntax of formulas.

$$\begin{aligned}
\delta ::=& \top \mid M_1 = M_2 \mid M \in \mathrm{dom} \mid \delta_1 \vee \delta_2 \mid \neg\delta \\
\varphi ::=& \delta \mid \varphi_1 \vee \varphi_2 \mid \neg\varphi \mid \langle\mu\rangle_{-}\varphi \mid F\varphi \mid K\varphi
\end{aligned}$$

where M_1, M_2, and M are terms, and μ is an action. We call δ a static formula and φ a modal formula. A static formula δ mentions equality of terms. On the other hand, a modal formula φ mentions traces.

$\langle\mu\rangle_{-}\varphi$ states that the previous action is μ, and φ holds just before observing μ. $F\varphi$ states that φ holds some time or other. The operator K expresses an attacker's knowledge, i.e., $K\varphi$ means the attacker knows that φ holds.

We often use abbreviations. Notably, P expresses $\neg K \neg$, and G expresses $\neg F \neg$. $P\varphi$ means that an attacker does not know φ does not hold. In other words, the attacker thinks that the possibility that φ holds remains. $G\varphi$ means that φ is always true.

4.2 Semantics

Our logic is an LTL-like logic with an epistemic operator. Let A be a name-variable-distinct process. We define $\mathrm{sv}(A) = \mathrm{fv}(A) \setminus \mathrm{dom}(A)$. Let ρ be an assignment which maps $\mathrm{sv}(A)$ to ground terms, $\mathbf{tr} \in \mathrm{tr}(A\rho)$, $0 \leq i \leq |\mathbf{tr}|$, and M_1 and M_2 be terms. Remember that $\mathrm{fr}(A)$ is a frame of A.

We suppose that δ and φ contain no variables other than $\mathrm{sv}(A) \cup \mathrm{dom}(\mathbf{tr}[i])$.

We omit the semantics of logical operators because they are defined as expected.

$$A, \rho, \mathbf{tr}, i \models M_1 = M_2 \text{ iff } (M_1\rho = M_2\rho)\mathrm{fr}(\mathbf{tr}[i])$$
$$A, \rho, \mathbf{tr}, i \models M \in \mathrm{dom} \text{ iff } M \text{ is a variable } x, \text{ and } x \in \mathrm{dom}(\mathbf{tr}[i])$$
$$A, \rho, \mathbf{tr}, i \models \langle \mu \rangle_{-}\varphi \text{ iff } \mathbf{tr}[i-1] \overset{\mu}{\Longrightarrow} \mathbf{tr}[i] \text{ in } \mathbf{tr} \text{ and } A, \rho, \mathbf{tr}, i-1 \models \varphi$$
$$A, \rho, \mathbf{tr}, i \models F\varphi \text{ iff } \exists j \geq i \text{ s.t. } A, \rho, \mathbf{tr}, j \models \varphi$$
$$A, \rho, \mathbf{tr}, i \models K\varphi \text{ iff } \forall \rho' \forall \mathbf{tr}' \in \mathrm{tr}(A\rho'); \mathbf{tr}[0,i] \sim_t \mathbf{tr}'[0,i] \Rightarrow A, \rho', \mathbf{tr}', i \models \varphi$$

We suppose that an attacker does not know what terms are assigned to free variables before the process runs. In other words, secret information is expressed as the assignment ρ. Hence, the definition of K contains a quantifier over assignments $\forall \rho'$. Recall that an attacker can observe only labeled transitions, so accessibility is defined based on static equivalence on traces.

We also define the satisfiability of formulas containing free variables. We put $\widetilde{y} = \mathrm{dom}(\mathbf{tr}[i])$. Let $\widetilde{x} = \mathrm{sv}(A)$. We suppose that φ contains no variables other than $\widetilde{x}, \widetilde{y}$, and $\widetilde{z}$.

$$A, \rho, \mathbf{tr}, i \models \varphi(\widetilde{x}, \widetilde{y}, \widetilde{z}) \text{ iff } \forall \widetilde{M}; A, \rho, \mathbf{tr}, i \models \varphi(\widetilde{x}, \widetilde{y}, \widetilde{M}),$$

where $\widetilde{M}$ is a sequence of ground terms.

From now, we suppose that all processes are name-variable-distinct. We often omit restriction of a domain of definition. $D(\rho)$ is a domain of definition of ρ.

When a formula φ is satisfied over all possible runs of a process A at the start of the run, we say that A satisfies φ.

Definition 12. $A \models \varphi \overset{\mathrm{def}}{\Leftrightarrow} \forall \rho\ \forall \mathbf{tr} \in \mathrm{tr}(A\rho); A, \rho, \mathbf{tr}, 0 \models \varphi$

Note that $A \not\models \varphi$ is not equivalent to $A \models \neg\varphi$.

Definition 13. $A \sqsubseteq_s B \overset{\mathrm{def}}{\Leftrightarrow} \forall \delta\ \forall \rho; A, \rho\restriction_{\mathrm{sv}(A)}, A\rho, 0 \models \delta \Rightarrow B, \rho\restriction_{\mathrm{sv}(B)}, B\rho, 0 \models \delta.$

$A \equiv_s B \overset{\mathrm{def}}{\Leftrightarrow} A \sqsubseteq_s B$ *and* $B \sqsubseteq_s A$.

Lemma 1. *If* $\mathrm{sv}(A) = \mathrm{sv}(B)$*, then*

$$[\forall \rho; A\rho \approx_s B\rho] \Leftrightarrow A \equiv_s B$$

Definition 14. $A \sqsubseteq_L B \overset{\mathrm{def}}{\Leftrightarrow} \forall \rho\ \forall \mathbf{tr} \in \mathrm{tr}(A\rho)\ \exists \mathbf{tr}' \in \mathrm{tr}(B\rho)$
s.t. $\forall i\ \forall \varphi; [A, \rho\restriction_{\mathrm{sv}(A)}, \mathbf{tr}, i \models \varphi \Leftrightarrow B, \rho\restriction_{\mathrm{sv}(B)}, \mathbf{tr}', i \models \varphi]$

$A \equiv_L B \overset{\mathrm{def}}{\Leftrightarrow} A \sqsubseteq_L B$ *and* $B \sqsubseteq_L A$.

Proposition 4. $A \sqsubseteq_L B \Rightarrow [\forall \varphi; B \models \varphi \Rightarrow A \models \varphi]$.

Proof. Assuming that $B \models \varphi$, we arbitrarily take ρ and $\mathbf{tr} \in \mathrm{tr}(A\rho)$.

By assumption, there exists $\mathbf{tr}' \in \mathrm{tr}(B\rho)$ such that

$$\forall i \forall \varphi'; [A, \rho\restriction_{\mathrm{sv}(A)}, \mathbf{tr}, i \models \varphi' \Leftrightarrow B, \rho\restriction_{\mathrm{sv}(B)}, \mathbf{tr}', i \models \varphi']. \tag{4.1}$$

Then, $B, \rho\restriction_{\mathrm{sv}(B)}, \mathbf{tr}', 0 \models \varphi$ because of $B \models \varphi$.
By (4.1), $A, \rho\restriction_{\mathrm{sv}(A)}, \mathbf{tr}, 0 \models \varphi$.
By arbitrariness of ρ and $\mathbf{tr}$, it holds that $A \models \varphi$.

□

May-testing equivalence does not imply trace equivalence. The counterexample below is given in [9], but we only modify notation.

$$
\begin{aligned}
A &= \nu b c_1.(\overline{c_1}\langle token\rangle | c_1(x).\bar{c}\langle b\rangle | c_1(x).B) \\
B &= \nu c_2.(\overline{c_2}\langle h(a)\rangle | c_2(x).\bar{c}\langle x\rangle | !c_2(x).\overline{c_2}\langle h(x)\rangle),
\end{aligned}
$$

where h is a function symbol which has no equations.

A and B are may-testing equivalent, but they are not trace equivalent. In addition, A and B satisfy the same formulas. 0 denotes the null substitution.

Lemma 2. *For any φ and the above A and B, we consider two traces.*

$$
\begin{aligned}
\mathbf{tr}:\ & A \Longrightarrow A_1 \Longrightarrow \dots \Longrightarrow A_k \overset{\nu y.\bar{c}\langle y\rangle}{\Longrightarrow} A'_1 \Longrightarrow \dots \Longrightarrow A'_m \\
\mathbf{tr}':\ & B \Longrightarrow B_1 \Longrightarrow \dots \Longrightarrow B_k \overset{\nu y.\bar{c}\langle y\rangle}{\Longrightarrow} B'_1 \Longrightarrow \dots \Longrightarrow B'_m,
\end{aligned}
$$

where $\mathrm{fr}(A'_m) \equiv \nu b c_1 c_2.\{h^n(a)/y\}$ *and* $\mathrm{fr}(B'_m) \equiv \nu c_2.\{h^n(a)/y\}$. *Then,*

$$A, 0, \mathbf{tr}, i \models \varphi \Leftrightarrow B, 0, \mathbf{tr}', i \models \varphi$$

for any i.

Lemma 3. *For any φ and the above A and B, we consider two traces.*

$$
\begin{aligned}
\mathbf{tr}:\ & A \Longrightarrow A_1 \Longrightarrow \dots \Longrightarrow A_k \overset{\nu y.\bar{c}\langle y\rangle}{\Longrightarrow} A'_1 \Longrightarrow \dots \Longrightarrow A'_m \\
\mathbf{tr}':\ & B \Longrightarrow B_1 \Longrightarrow \dots \Longrightarrow B_k \overset{\nu y.\bar{c}\langle y\rangle}{\Longrightarrow} B'_1 \Longrightarrow \dots \Longrightarrow B'_m,
\end{aligned}
$$

where $\mathrm{fr}(A'_m) \equiv \nu b c_1 c_2.\{b/y\}$ *and* $\mathrm{fr}(B'_m) \equiv \nu c_2.\{h^{n+1}(a)/y\}$. *$n$ is the number of h occurring in φ. Then,*

$$A, 0, \mathbf{tr}, i \models \varphi \Leftrightarrow B, 0, \mathbf{tr}', i \models \varphi$$

for any i.

Lemma 4. *For any φ and the above A and B, we consider two traces.*

$$
\begin{aligned}
\mathbf{tr}:\ & A \Longrightarrow A_1 \Longrightarrow \dots \Longrightarrow A_k \\
\mathbf{tr}':\ & B \Longrightarrow B_1 \Longrightarrow \dots \Longrightarrow B_k
\end{aligned}
$$

Then,

$$A, 0, \mathbf{tr}, i \models \varphi \Leftrightarrow B, 0, \mathbf{tr}', i \models \varphi$$

for any i.

Proof. We simultaneously prove Lemma 2, 3, and 4 by induction on φ.

1. $\top$ or $M \in \mathrm{dom}$.

   These cases are trivial.

2. $M_1 = M_2$

   Lemma 2 and 4 trivially hold.

   For Lemma 3, $(M_1 = M_2)\mathrm{fr}(A'_m) \Leftrightarrow (M_1 = M_2)\mathrm{fr}(B'_m)$ because $h^{n+1}(a)$ behaves like a fresh name in $M_1 = M_2$.

3. $\varphi_1 \vee \varphi_2, \neg\varphi, \langle\mu\rangle_{-}\varphi$ or $F\varphi$

   These cases follow from induction hypothesis.

4. $K\varphi$

   We assume that $A, 0, \mathbf{tr}, i \models K\varphi$.

   For and $\mathbf{tr}'' \in \mathrm{tr}(A)$ such as $\mathbf{tr}[0,i] \sim_t \mathbf{tr}''[0,i]$, it holds that $A, 0, \mathbf{tr}'', i \models \varphi$.

   We arbitarily take $\mathbf{tr}''' \in \mathrm{tr}(B)$ such as $\mathbf{tr}'[0,i] \sim_t \mathbf{tr}'''[0,i]$.

   Then, we can choose $\mathbf{tr}'' \in \mathrm{tr}(A)$ such as $\mathbf{tr}[0,i] \sim_t \mathbf{tr}''[0,i]$ to apply Lemma 2, 3, or 4.

□

Proposition 5. *For the above A and B,*

$$\forall\varphi; [A \models \varphi \Leftrightarrow B \models \varphi]$$

Proof. It immediately follows from Lemma 2, 3, and 4.

□

We consider the situation below:

The unknown process X exists. The attacker knows that X equals A or B and wants to attack a target using X. Then, he starts to attack, presuming that X is equal to A. If X is really equal to A, there is no problem. If X is equal to B, he eventually notices that his premise is wrong and can do it over.

In any case, he can attack based on correct information about X. Hence, we consider that A should be distinguished from B.

If infinite logical disjunction is allowed, we can discern A and B using our epistemic logic. Indeed,

$$A \not\models (\langle \nu x.\bar{c}\langle x\rangle\rangle_{-}\top) \to \bigvee_n x = h^n(a)$$

$$B \models (\langle \nu x.\bar{c}\langle x\rangle\rangle_{-}\top) \to \bigvee_n x = h^n(a)$$

Thus, $A \not\equiv_L B$. Our epistemic logic is LTL-like, so Definition 14 is reasonable as logical equivalence. We prove that trace equivalence corresponds with logical equivalence $\equiv_L$ in the next section. Following the results, we consider that trace equivalence is more suitable than may-testing equivalence.

4.3 Correspondence with Trace Equivalence

We prove that trace equivalent processes satisfy the same formulas.

Theorem 2. *If* $\mathrm{sv}(A) = \mathrm{sv}(B)$*, then*
1. $A \approx_t B \Rightarrow A \sqsubseteq_L B$*; 2.* $A \sqsubseteq_L B \Rightarrow A \subseteq_t B$

Proof outline.
1) We prove

$$\forall\rho\ \forall\mathbf{tr} \in \mathrm{tr}(A\rho), \mathbf{tr}' \in \mathrm{tr}(B\rho);$$
$$\mathbf{tr} \sim_t \mathbf{tr}' \Rightarrow \forall i\ \forall\varphi; [A, \rho, \mathbf{tr}, i \models \varphi \Leftrightarrow B, \rho, \mathbf{tr}', i \models \varphi],$$

by induction on the syntax of formulas.
2) We arbitrarily take an assignment ρ and $\mathbf{tr} \in \mathrm{tr}(A\rho)$.
By $A \sqsubseteq_L B, \exists\mathbf{tr}' \in \mathrm{tr}(B\rho)$ s.t.$\forall i\ \forall\varphi; [A, \rho, \mathbf{tr}, i \models \varphi \Leftrightarrow B, \rho, \mathbf{tr}', i \models \varphi]$.
Then, we can prove $\mathbf{tr} \sim_t \mathbf{tr}'$.
By arbitrariness of $\mathbf{tr}$, it immediately follows that $A \sqsubseteq_L B \Rightarrow A \subseteq_t B$. □

Why is the antecedent of 1 trace equivalence? In other words, does the converse of 2 hold? The answer is "no." The converse of 2 does not hold. The cause is the existence of the epistemic operator K. We give a counterexample.

Example 3.

$$A = \overline{x}\langle s\rangle$$
$$B = \overline{x}\langle s\rangle + \overline{a}\langle s\rangle$$

$A \subseteq_t B$ *obviously holds. Nevertheless,*

$$A \not\models G(\langle\nu y.\overline{a}\langle y\rangle\rangle_{-}\top \rightarrow P(x \neq a))$$
$$B \models G(\langle\nu y.\overline{a}\langle y\rangle\rangle_{-}\top \rightarrow P(x \neq a))$$

The contraposition of Proposition 4 implies $A \not\sqsubseteq_L B$*.*

Moreover, $A \subseteq_t B$ does not imply that $[\forall\varphi; A \models \varphi \Rightarrow B \models \varphi]$. The cause is the existence of the temporal operator F. We give a counterexample.

Example 4.

$$A = \overline{a}\langle s\rangle$$
$$B = \overline{a}\langle s\rangle.\overline{b}\langle s\rangle$$

$A \subseteq_t B$ *obviously holds. Nevertheless,*

$$A \models G(\neg\langle\nu y.\overline{b}\langle y\rangle\rangle_{-}\top)$$
$$B \not\models G(\neg\langle\nu y.\overline{b}\langle y\rangle\rangle_{-}\top)$$

We can immediately conclude that the following theorem holds.

Theorem 3. $A \approx_t B \Leftrightarrow A \equiv_L B$*.*

This theorem suggests that trace equivalence is suitable to define security properties. We give Proposition 6 as an example in the following section.

4.4 Application

We define *minimal secrecy.* We regard it as generalized minimal anonymity [18]. Anonymity means that who takes the action, such as casting a ballot, is hidden from an attacker. If an action a performed by an agent i is minimally anonymous, an observer cannot concern that the agent i have performed the action a. On the other hand, if a variable x satisfying a property δ is minimally secret, an attacker cannot concern that the variable x satisfies the property δ. Recall that δ denotes a static formula.

Definition 15. *A variable x is minimally secret with respect to δ in A iff*

$$A \models G(P(\neg\delta(x))).$$

This definition means that attackers cannot be sure that $\delta(x)$ is true.

This property is not robust. For instance, although x is minimally secret with respect to a nontrivial formula δ, x is not always minimally secret with respect to $\neg\delta$. Hereafter, we often omit objects of outputs.

Example 5. *We put $\delta(z) : z \neq a \wedge z \neq b$.*
We consider a process

$$\text{if } x = a \text{ then } \overline{c} \text{ else } \overline{d}.$$

Then x is minimally secret with respect to δ, but not secret with respect to $\neg\delta$.

Moreover, $\vee$ does not preserve minimally secret. However, $\wedge$ preserves it.

Although x is minimally secret in A, x is not always secret in $A|A$. Besides, restriction does not always preserve minimal secrecy.

Example 6. *We put $\delta(z) : z = a$. We put*

$$P = \text{if } x = a \text{ then } (\overline{a}\langle s\rangle + \overline{b}\langle s\rangle) \text{ else } \overline{a}\langle s\rangle, Q = \text{if } x = b \text{ then } \overline{b}\langle s\rangle \text{ else } \overline{c}\langle s\rangle.$$

Then x is minimally secret with respect to δ in $P + Q$, but not secret in $(P + Q)|(P + Q)$.

Example 7. *We put $\delta(z) : z = a$. Then, x is minimally secret with respect to δ in $\overline{x} + \overline{a}$, but not secret in $\nu a.(\overline{x} + \overline{a})$.*

We define *total secrecy.* We can also regard it as generalized total anonymity [18]. If an action a performed by an agent i is totally anonymous, an observer thinks that it is possible that any agent has performed the action. That is, the observer can obtain no information about the performer. On the other hand, if a variable x is totally secret, an attacker cannot concern that the variable x satisfies any property δ. Total secrecy states that attackers can obtain no information about x.

Definition 16. *x is totally secret in $A(x, \widetilde{y})$ iff*

$$\forall \delta(z, \widetilde{z}, \widetilde{w}); A(x, \widetilde{y}) \models G(P(\neg\delta(x, \widetilde{y}, \widetilde{w})))$$

where δ contains no variables other than ones in $\{z\} \cup \widetilde{z} \cup \widetilde{w}$ and satisfies that

$$\forall \widetilde{N} \forall \psi \exists M : \textit{ground} \text{ s.t. } \psi \models \neg\delta(M, \widetilde{N}, \widetilde{w}).$$

Besides, $|\widetilde{y}| = |\widetilde{z}|$ and $\widetilde{w} \cap (\{x\} \cup \widetilde{y}) = \emptyset$.

The latter condition means that δ is nontrivial in any case.

Proposition 6. *x is totally secret in $A(x, \widetilde{y})$ iff $A(x, \widetilde{y}) \approx_t A(x', \widetilde{y})$.*

Proof. $\Rightarrow$) We suppose for the sake of contradiction that $A(x, \widetilde{y}) \not\approx_t A(x', \widetilde{y})$.

There exist M_1, M_2, and $\widetilde{N}$ that are ground such that $A(M_1, \widetilde{N}) \not\approx_t A(M_2, \widetilde{N})$.

We suppose that $A(M_1, \widetilde{N}) \not\sqsubseteq_t A(M_2, \widetilde{N})$. Then, there exists $\mathbf{tr} \in \mathrm{tr}(A(M_1, \widetilde{N}))$ such that any trace of $A(M_2, \widetilde{N})$ is not statically equivalent to $\mathbf{tr}$.

We put $\delta(z, \widetilde{z}) : z \neq M_2 \vee \widetilde{z} \neq \widetilde{N}$. Then

$$A(x, \widetilde{y}), (x \mapsto M_1, \widetilde{y} \mapsto \widetilde{N}), \mathbf{tr}, |\mathbf{tr}| \models K\delta(x, \widetilde{y}).$$

This contradicts total secrecy.

$\Leftarrow$) We arbitrarily take $\delta, \rho, \mathbf{tr} \in \mathrm{tr}(A(\rho(x), \rho(\widetilde{y})))$ and i, where δ meets the demand of Definition 16.

We take M such that $\mathrm{fr}(\mathbf{tr}[i]) \models \neg\delta(M, \rho(\widetilde{y}), \widetilde{w})$. Let ρ' be

$$\rho'(y) = \begin{cases} M & (y = x) \\ \rho(y) & (\textit{otherwise}). \end{cases}$$

By assumption, $A(\rho(x), \rho(\widetilde{y})) \approx_t A(M, \rho(\widetilde{y}))$.

Hence, there exists $\mathbf{tr'} \in \mathrm{tr}(A(M, \rho(\widetilde{y})))$ such that $\mathbf{tr} \sim_t \mathbf{tr'}$.

Then, $A(x, \widetilde{y}), \rho', \mathbf{tr'}, i \models \neg\delta(M, \rho(\widetilde{y}), \widetilde{w})$.

Therefore, $A(x, \widetilde{y}), \rho, \mathbf{tr}, i \models P(\neg\delta(x, \rho(\widetilde{y}), \widetilde{w}))$.

Then, $A(x, \widetilde{y}) \models G(P(\neg\delta(x, \rho(\widetilde{y}), \widetilde{w})))$. □

Theorem 4. *If x is totally secret in $A(x, \widetilde{y})$, then x is also totally secret in $E[A(x, \widetilde{y})]$ for every context $E[\_]$ which does not contain x.*

Proof. It immediately follows from Theorem 1 and Proposition 6. □

If the context $E[\_]$ contains x, Theorem 4 does not hold. Indeed,

$$E[\_] = \text{ if } x = a \text{ then } \_ \text{ else } 0$$

is a counterexample.

Our framework can generalize role interchangeability [24]. When x_i satisfies a property δ_k, and x_l satisfies a property δ_j, an attacker thinks that it is possible that x_l satisfies a property δ_k and x_i satisfies a property δ_j.

Definition 17. *We put* $\mathrm{sv}(A) = \{x_1, ..., x_p\}$*,* $J = \{1, ..., q\}$*, and* $I = \{1, ..., p\}$*.* (x_i, δ_k) *is role interchangeable regarding* $\{\delta_j(z_j, \widetilde{y_j})\}_{j\in J}$ *in* A *iff*

$$A(x_1, ..., x_p) \models G(\delta_k(x_i, \widetilde{y_k}) \to \bigwedge_{l\in I}\bigwedge_{j\in J}(\delta_j(x_l, \widetilde{y_j}) \to P(\delta_k(x_l, \widetilde{y_k}) \wedge \delta_j(x_i, \widetilde{y_j}))))$$

where $\widetilde{y_j} \cap \{x_1, ..., x_p\} = \emptyset$ *for all* $j \in J$*.*

Proposition 7.
$\forall\widetilde{M}\ \forall i\ \forall\mathbf{tr} \in \mathrm{tr}(A(M_1, ..., M_p))$
$\exists\widetilde{N}\ \exists\mathbf{tr'} \in \mathrm{tr}(A(M_i, N_2, ..., N_{i-1}, M_1, N_{i+1}, ..., N_p))$ s.t. $\mathbf{tr} \sim_t \mathbf{tr'}$
$\Leftrightarrow (x_1, \delta_k)$ *is role interchangeable with respect to* $\{\delta_j\}$ *in* A *for all* $\{\delta_j\}$ *and* k*.*

Corollary 1. $\forall l \in I \setminus \{i\}; A(x_1, ..., x_i, ..., x_l, ..., x_p) \approx_t A(x_1, ..., x_l, ..., x_i, ..., x_p)$
$\Rightarrow (x_i, \delta_k)$ *is role interchangeable with respect to* $\{\delta_j\}$ *in* A *for all* $\{\delta_j\}$ *and* k*.*

The converse holds only when $p = 2$.

Proposition 8. $A(x_1, x_2) \approx_t A(x_2, x_1)$
$\Leftrightarrow (x_1, \delta_k)$ *is role interchangeable with respect to* $\{\delta_j\}_{j\in J}$ *in* A *for all* $\{\delta_j\}_{j\in J}$ *and* k*.*

We give a counterexample for $p = 3$.

Example 8. *We put*

$$A(x, y, z) = \text{if } x = y \text{ then } \overline{x} + \overline{z} \text{ else if } x = z \text{ then } \overline{x} + \overline{y} \text{ else } \overline{y} + \overline{z}$$

Then, (x, δ_k) *is role interchangeable regarding* $\{\delta_j\}$ *in* A *for all* $\{\delta_j\}_{j\in J}$ *and* k *by Proposition 7, but* $A(a, b, a) \not\approx_t A(b, a, a)$*. Thus,* $A(x, y, z) \not\approx_t A(y, x, z)$*.*

We can also consider role permutativity. Mano [23] showed that it is strictly stronger than role interchangeability. Role permutativity states that even if p values are swapped, an attacker cannot notice it. Here, $\mathfrak{S}_p$ denotes the symmetric group on $\{1, ..., p\}$.

Definition 18. *We put* $\mathrm{sv}(A) = \{x_1, ..., x_p\}$*,* $J = \{1, ..., q\}$*, and* $I = \{1, ..., p\}$*.* $\{\delta_j\}_{j\in J}$ *is role permutable in* A *iff*

$$\forall n \le p\ \forall\psi \in \mathfrak{S}_p; A(x_1, ..., x_p) \models G(\bigwedge_{k\le n}\delta_{i_k}(x_{i_k}, \widetilde{y_k}) \to P(\bigwedge_{k\le n}\delta_{i_k}(x_{i_{\psi(k)}}, \widetilde{y_k})))$$

where $\widetilde{y_j} \cap \{x_1, ..., x_p\} = \emptyset$ *for all* j *and each* i_k *differs.*

Proposition 9. $\forall\psi \in \mathfrak{S}_p; A(x_1, ..., x_p) \approx_t A(x_{\psi(1)}, ..., x_{\psi(p)})$
$\Leftrightarrow \{\delta_j\}_{j\in J}$ *is role permutable in* A *for all* $\{\delta_j\}_{j\in J}$*.*

The proof is similar to Proposition 7.

We define openness. We regard it as generalized identity [33]. Identity is the property of revealing what a specific agent i performed. If x is open, the value of x is disclosed. Parallel composition does not preserve openness of a specific variable.

Definition 19. *x is open in A under $\Delta(x)$ iff*

$$\forall \rho\ \forall \mathbf{tr} \in \mathrm{tr}_{\max}(A\rho); A, \rho, \mathbf{tr}, |\mathbf{tr}| \models \Delta(x) \to K(\Delta(x) \to (x = x\rho)).$$

Example 9. *We put $\Delta(z) : z = r \vee z = s$,*

$$P = \text{if } x = r \text{ then } \overline{a}\langle n\rangle \text{ else } \overline{b}\langle n\rangle \textit{ and } Q = \text{if } x = r \text{ then } \overline{b}\langle n\rangle \text{ else } \overline{a}\langle n\rangle.$$

Then x is open in P and Q under $\Delta(x)$, but x is not open in $P|Q$ under $\Delta(x)$.

$$P|Q, [x \mapsto r], \overline{a}\overline{b}, 2 \not\models \Delta(x) \to K(\Delta(x) \to (x = r)),$$

where $\overline{a}\overline{b}$ is

$$P|Q \xrightarrow{\nu y.\overline{a}\langle y\rangle} Q|\{n/y\} \xrightarrow{\nu z.\overline{b}\langle z\rangle} \{n/y, n/z\}.$$

Note that it is proved in [33] that identity is preserved by parallel composition. However, the definition of parallel composition is much different from our definition. The authors considered the case that a specific agent performs two different actions.

Problem 2.
Input: *An extended process A, an assignment ρ, a trace $\mathbf{tr} \in \mathrm{tr}(A\rho)$, an integer $0 \le i \le |\mathbf{tr}|$, and a formula φ.*
Question: *Does $A, \rho, \mathbf{tr}, i \models \varphi$ hold?*

Proposition 10. *Even if the word problem in Σ is decidable, Problem 2 can be undecidable.*

Abadi and Cortier [3] proved that static equivalence can be undecidable even if the word problem in Σ is decidable. Proposition 10 follows from it.

We a bit change semantics of $K\varphi$. We repeat it.

$$A, \rho, \mathbf{tr}, i \models K\varphi \text{ iff } \forall \rho'\ \forall \mathbf{tr}' \in \mathrm{tr}(A\rho'); \mathbf{tr}[0, i] \sim_t \mathbf{tr}'[0, i] \Rightarrow A, \rho', \mathbf{tr}', i \models \varphi,$$

Now, we restrict ρ' to be an assignment to names.

We also change the definition of satisfaction. We repeat it.

$$A \models \varphi \text{ iff } \forall \rho\ \forall \mathbf{tr} \in \mathrm{tr}(A\rho); A, \rho, \mathbf{tr}, 0 \models \varphi$$

Now, we restrict ρ to be an assignment which maps free variables to only names and restrict inputted messages in $\mathbf{tr}$ to be only variables. That is, we assume that an attacker cannot tamper with a message. In other words, the attacker can only transfer messages without any change.

Problem 3.
Input: *An extended process A and a formula φ.*
Question: *Does $A \models \varphi$ hold?*

A *convergent subterm theory* is an equational theory defined by finite equations whose each right-hand side is a proper subterm of the left-hand side.

Proposition 11. *If the equational theory on Σ is a convergent subterm theory and the extended process A is finite, Problem 3 is decidable.*

4.5 Comparison with the Work of Chadha et al.

4.5.1 Epistemic Logic

Chadha et al. [7] proposed epistemic logic for the applied pi calculus. We compare our logic and their logic in this subsection.

First, they added *events* to the applied pi calculus as follows. e is an event, and $\widetilde{M}$ is a sequence of terms. They are parameters of e.

$$P, Q ::= ... \mid \mathrm{e}(\widetilde{M}).P$$

$$\mathrm{e}(\widetilde{M}).P \longrightarrow P \mid [\mathrm{e}(\widetilde{M})]$$

$[\mathrm{e}(\widetilde{M})]$ is an event store. When an event $e(\widetilde{M})$ happens, an event store $[\mathrm{e}(\widetilde{M})]$ is generated.

Event stores cannot do any internal reduction and labeled transition. An environment cannot observe any event and event store, so this enrichment does not affect behavioral equivalences.

They defined static formulas **Has** and $\widehat{\mathbf{evt}}$. **Has** directly represents attackers' knowledge. $\mathbf{Has}(\widehat{T})$ means that an attacker can obtain a term T from a frame. $\widehat{\mathbf{evt}}(\widehat{T}_1, .., \widehat{T}_r)$ means that the event **evt** parametrized by $T_1, ..., T_r$ had occurred. In addition, their logic contains a quantifier over terms.

Their logic is α-sensitive. As a matter of fact, α-equivalent processes do not always satisfy the same formulas in their framework because secret values are expressed as bound names or through events.

We show an example. $\nu n.0$ satisfies $\mathbf{Has}(\widehat{m})$, but $\nu m.0$ does not satisfy it. In addition, statically equivalent processes do not always satisfy the same closed formulas.

On the other hand, we express secret values as assignments to free variables, so our logic is sensitive to changing variables.

Information about a term handled in their logic is its value and its structure. For example, $\exists z.\mathbf{Has}(\mathbf{hash}(z))$ means that an attacker has the hash of something.

Our logic cannot express information about structures. Information about terms is expressed as a relationship between terms. That is, our logic can express partial information about secret values.

Three kinds of modal operators, $\Box, \boxminus$ and $\mathbf{K}$ are introduced in their paper. $\Box\varphi$ means that φ is always true in the future. $\boxminus$ is a backward temporal operator, and $\mathbf{K}$ is an epistemic operator. $\mathbf{K}\varphi$ means that an attacker knows that φ is true.

Alternatively, our logic also has an epistemic operator K and a forward temporal operator F, but F is like the dual of $\Box$. $F\varphi$ means that φ holds some time or other. Furthermore, $\langle\mu\rangle_{-}$ is introduced to express an attacker's knowledge about the dynamic behavior of the target process. In contrast, their logic cannot describe information about transitions. Indeed, $\overline{a}\langle n\rangle.0$ and $\overline{b}\langle n\rangle.0$ are not even trace equivalent, but they satisfy the same formulas in their logic.

Chadha et al. directly expressed an agreement property of authentication using their logic. Our logic cannot directly express it, but it is difficult to express the agreement property in the original applied pi calculus anyway.

4.5.2 Privacy

Chadha et al. developed the definition of privacy in e-voting as follows. They considered protocol instances in which two voters Alice and Bob participate, and voting options are **0** and **1**.

Chadha et al. called

$$\mathcal{V} = \Sigma_{v_a, v_b \in \{\mathbf{0},\mathbf{1}\}} \mathrm{votes}(v_a, v_b).V(v_a, v_b)$$

a voting process. $\mathrm{votes}(v_a, v_b)$ is an event, which means that Alice voted v_a and Bob voted v_b.

On the other hand, votes are expressed as values of free variables in our framework.

Definition 20 ([7, Definition 9]). *The voting process $\mathcal{V}$ respects* privacy *if $\mathcal{V} \models$ **Aprivacy** $\wedge$ **Bprivacy** where*

- **Aprivacy** $\overset{\text{def}}{=} \wedge_{\mathrm{v} \in \{\mathbf{0},\mathbf{1}\}} \Box(\mathbf{K}(\mathbf{Avote}(\mathrm{v})) \to \mathbf{Bvote}(\mathrm{v}))$, *and*

- **Bprivacy** $\overset{\text{def}}{=} \wedge_{\mathrm{v} \in \{\mathbf{0},\mathbf{1}\}} \Box(\mathbf{K}(\mathbf{Bvote}(\mathrm{v})) \to \mathbf{Avote}(\mathrm{v}))$.

Avote(v) means that Alice voted v, and **Bvote**(v) is similar.

Minimal secrecy of a vote never holds because an attacker trivially knows votes when all votes agree. We consider protocol instances in which m voters participate and n voting options exist. Let v_i be a vote of i. We consider the property below:

$$\vee_{j,k} v_j \neq v_k \to \wedge_i \wedge_v G(K(v_i = v) \to v_1 = v \wedge \ldots \wedge v_{i-1} = v \wedge v_{i+1} = v \wedge \ldots \wedge v_m = v)$$

The consequence in $G(\ldots)$ implies that $v_i \neq v$ due to the antecedent condition, so we can rewrite the property.

$$\vee_{j,k} v_j \neq v_k \to \wedge_i \wedge_v G(K(v_i = v) \to v_i \neq v)$$

Moreover, we take the contraposition in G.

$$\vee_{j,k} v_j \neq v_k \to \wedge_i \wedge_v G(v_i = v \to P(v_i \neq v))$$

That is,

$$\vee_{j,k} v_j \neq v_k \to \wedge_i \wedge_v G(P(v_i \neq v))$$

This consequence is exactly minimal secrecy. Besides, minimal secrecy of voting implies privacy, so privacy and minimal secrecy of voting agree under the disagreement condition $\vee_{j,k} v_j \neq v_k$.

It was shown that $V(\mathbf{0}, \mathbf{1}) \approx_t V(\mathbf{1}, \mathbf{0})$ implies that $\mathcal{V}$ respects privacy, and the partial converse was given in [7]. We give several similar properties of minimal secrecy.

Chadha et al. defined publishing traces and abort traces. We slightly extend their definition.

Definition 21. *We arbitrarily take votes* $\mathbf{v_1}, ..., \mathbf{v_m}$.

A maximal trace $\mathbf{tr} \in \mathrm{tr}_{\max}(V(\mathbf{v_1}, ..., \mathbf{v_m}))$ *is a publishing trace if for any* $\mathbf{v'_1}, ..., \mathbf{v'_m}$ *such that* $\{\mathbf{v_1}, ..., \mathbf{v_m}\} \neq \{\mathbf{v'_1}, ..., \mathbf{v'_m}\}$ *as multisets, there is no* $\mathbf{tr'} \in \mathrm{tr}(V(\mathbf{v'_1}, ..., \mathbf{v'_m}))$ *such that* $\mathbf{tr'}$ *is statically equivalent to* $\mathbf{tr}$.

We say that $\mathbf{tr}$ *is an abort trace if it is not a publishing trace.*

Intuitively, a maximal trace $\mathbf{tr}$ is publishing if a poll is declared in the trace. On the other hand, an environment cannot know the voting result in an abort trace.

Definition 22. *We arbitrarily take votes* $\mathbf{v_1}, ..., \mathbf{v_m}$ *and an abort trace* $\mathbf{tr} \in \mathrm{tr}_{\max}(V(\mathbf{v_1}, ..., \mathbf{v_m}))$. *We say that a voting process* $\mathcal{V}$ *is equivalent for aborts if for any votes* $\mathbf{v'_1}, ..., \mathbf{v'_m}$ *there exists a* $\mathbf{tr'} \in \mathrm{tr}_{\max}(V(\mathbf{v'_1}, ..., \mathbf{v'_m}))$ *such that* $\mathbf{tr'}$ *is statically equivalent to* $\mathbf{tr}$.

If a voting process $\mathcal{V}$ is equivalent for aborts, every abort trace does not leak information about votes.

Proposition 12. *We assume that a voting process* $\mathcal{V}$ *is equivalent for aborts, and minimal secrecy of each vote in* $V(v_1, ..., v_m)$ *holds under the disagreement condition* $\vee_{j,k} v_j \neq v_k$.

1. $m = 2 \Rightarrow V(v_1, v_2) \approx_t V(v_2, v_1)$.

2. $m = 3$ *and* $n = 2 \Rightarrow V(v_1, v_2, v_3) \approx_t V(v_2, v_1, v_3)$.

3. *Otherwise,* $V(v_1, ..., v_i, ..., v_j, ..., v_m) \approx_t V(v_1, ..., v_j, ..., v_i, ..., v_m)$ *does not always hold.*

Proof. 1: Given a and b are different votes.

For the sake of contradiction, we suppose that $V(a, b) \not\sqsubseteq_t V(b, a)$.

There exists a maximal trace $\mathbf{tr} \in \mathrm{tr}_{\max}(V(a, b))$ such that any trace of $V(b, a)$ is not statically equivalent to $\mathbf{tr}$.

$\mathcal{V}$ is equivalent for aborts, so $\mathbf{tr}$ is publishing.

Hence any trace of $V(p, q)$ is not statically equivalent to $\mathbf{tr}$ if $\{p, q\} \neq \{a, b\}$.

Thus, an attacker knows that voter 1 voted a, and it contradicts minimal secrecy.

2: For the sake of contradiction, we suppose that $V(\mathbf{0}, \mathbf{1}, \mathbf{0}) \not\sqsubseteq_t V(\mathbf{1}, \mathbf{0}, \mathbf{0})$.

There exists a maximal trace $\mathbf{tr} \in \mathrm{tr}_{\max}(V(\mathbf{0}, \mathbf{1}, \mathbf{0}))$ such that any trace of $V(\mathbf{1}, \mathbf{0}, \mathbf{0})$ is not statically equivalent to $\mathbf{tr}$.

$\mathcal{V}$ is equivalent for aborts, so $\mathbf{tr}$ is publishing.

Hence any trace of $V(p, q, r)$ is not statically equivalent to $\mathbf{tr}$ if $\{p, q, r\} \neq \{\mathbf{0}, \mathbf{1}, \mathbf{0}\}$ as multisets.

Thus, an attacker knows that voter 1 voted $\mathbf{0}$, and it contradicts minimal secrecy.

3: If $m = 4$ and $n = 2$, there exists a counterexample. We consider a process $V(v_1, v_2, v_3, v_4)$ as below. Here, all signals are the same.

- If the number of $\mathbf{1}$ is one, V sends a signal on a channel a.

- If $(v_1, v_2, v_3, v_4) = (\mathbf{0}, \mathbf{0}, \mathbf{1}, \mathbf{1}), (\mathbf{0}, \mathbf{1}, \mathbf{0}, \mathbf{1}), (\mathbf{1}, \mathbf{0}, \mathbf{1}, \mathbf{0})$, V sends a signal on a channel b.
- If $(v_1, v_2, v_3, v_4) = (\mathbf{0}, \mathbf{1}, \mathbf{1}, \mathbf{0}), (\mathbf{1}, \mathbf{0}, \mathbf{0}, \mathbf{1}), (\mathbf{1}, \mathbf{1}, \mathbf{0}, \mathbf{0})$, V sends a signal on a channel c.
- If the number of **1** is three, V sends a signal on a channel d.
- If all votes are **0**, V sends a signal on a channel e.
- If all votes are **1**, V sends a signal on a channel f.

$\mathcal{V}$ is equivalent for aborts, but $V(\mathbf{0}, \mathbf{0}, \mathbf{1}, \mathbf{1}) \not\approx_t V(\mathbf{1}, \mathbf{0}, \mathbf{0}, \mathbf{1})$.

If $m = 3$ and $n = 3$, there also exists a counterexample. We consider a process $V(v_1, v_2, v_3)$ as below. Here, all signals are the same.

- If $(v_1, v_2, v_3) = (\mathbf{0}, \mathbf{1}, \mathbf{2}), (\mathbf{2}, \mathbf{1}, \mathbf{0}), (\mathbf{1}, \mathbf{2}, \mathbf{0})$, V sends a signal on a channel a.
- If $(v_1, v_2, v_3) = (\mathbf{0}, \mathbf{2}, \mathbf{1}), (\mathbf{1}, \mathbf{0}, \mathbf{2}), (\mathbf{2}, \mathbf{0}, \mathbf{1})$, V sends a signal on a channel b.
- If the number of **0** is two and the number of **1** is one, V sends a signal on a channel c.
- If the number of **0** is two and the number of **2** is one, V sends a signal on a channel d.
- If the number of **0** is one and the number of **1** is two, V sends a signal on a channel e.
- If the number of **0** is one and the number of **2** is two, V sends a signal on a channel f.
- If the number of **1** is two and the number of **2** is one, V sends a signal on a channel g.
- If the number of **1** is one and the number of **2** is two, V sends a signal on a channel h.

$\mathcal{V}$ is equivalent for aborts, but $V(\mathbf{0}, \mathbf{1}, \mathbf{2}) \not\approx_t V(\mathbf{1}, \mathbf{0}, \mathbf{2})$. □

Chapter 5

Related Work

5.1 Process Algebras

Logics about behavior of labeled transition systems originate from Hennessy-Milner logic [19] that is a modal logic characterizing observational congruence. That is, observational equivalent systems satisfy the same modal formulas when these systems are image-finite.

Process algebra is a special LTS. The spi calculus [4] is an extension of the π-calculus [26, 27]. It enables us to handle symmetric-key encryption based on the Dolev-Yao model [13]. In the spi calculus, two ciphertexts obtained by encrypting two different plaintexts are indistinguishable unless an observer gets a secret key. Abadi and Gordon formalized security properties using testing equivalence. This testing equivalence is a bit different from may-testing equivalence for the applied pi calculus. The former considers only parallel compositions as tests.

We focused on the applied pi calculus [2] because it is more expressive than the spi calculus. That is, we intend to handle more various security notions. In this calculus, a process can send not only names but also terms via alias variables. Due to this feature, we can handle not only secrecy but also stricter properties. The authors proved that observational equivalence and labeled bisimilarity correspond.

Chadha et al. [7] already developed an epistemic logic for the applied pi calculus. They defined formulas **Has** and $\widehat{\mathbf{evt}}$. **Has** directly represents attackers' knowledge, and $\widehat{\mathbf{evt}}$ means that a particular event had occurred. Temporal modalities were also used, but they do neither mention the just previous nor next action. The epistemic operator **K** was defined based on static equivalence on traces. The authors suggested that trace equivalence is more suitable than labeled bisimilarity when we consider privacy. However, a correspondent relation between logic and behavior of processes was not provided.

Horne [20] introduced quasi-open bisimilarity, and he proved that it coincides with open bisimilarity. Moreover, intuitionistic modal logic $\mathcal{FM}$ characterizes quasi-open bisimilarity. The law of excluded middle does not hold in the logic because processes containing a free variable are also considered.

Parrow et al. [29] defined nominal transition systems and developed modal logic characterizing bisimilarity for a nominal transition system. Process algebra is one of nominal transition systems. A nominal transition system has a nominal set of states and a nominal set of actions. A nominal set [30] is a set with a permutation action, and each element has a finite support. If a and b are not in the support of an element X in a nominal set, the transposition (a, b) does not affect X. A support of a process corresponds with a set of free names of the process.

Fiore and Abadi [15] developed symbolic models of processes. They gave a procedure to decide whether an environment can derive a message M. Their technique can be used for verification. However, equivalences on processes were not studied in the paper.

Goubault-Larrecq et al. [17] proposed probabilistic applied pi calculus. In this calculus, our Theorem 1 no longer holds. It is known that trace distribution preorder [31] is not a congruence. On the other hand, it is shown in [5] that probabilistic trace equivalence for nondeterministic and probabilistic LTS is a congruence concerning parallel composition. Probabilistic trace equivalence is coarser than trace distribution equivalence. The former considers traces weighted by probability, but the latter considers the probabilistic distribution of traces. Which is better is not clear.

Knight et al. [22] developed spatial and epistemic process calculus. Their study is for concurrent constraint programming, so their processes can add constraints. They proved that observational equivalence is a congruence. Their processes do not have labeled actions, so observational equivalence states that equivalent processes provide the same results. On the other hand, in the applied pi calculus, trace equivalent processes provide equivalent traces and indistinguishable information.

5.2 Logics for Concurrent Systems

Knight et al. [21] defined an epistemic logic for an LTS. This framework is based on Hennessy-Milner logic, and it handles multiple agents' knowledge. They also proved weak completeness. However, compositionality was not discussed.

Toninho and Caires [32] proposed a dynamic spatial epistemic logic, which reasons what information a process can obtain. The epistemic operator means not only an attacker's knowledge but also a participant's knowledge, so, for example, the logic can reason a correspondence assertion.

Tsukada et al. [33] studied sequential and parallel compositionality of security notions using an epistemic logic for a multiagent system. They proved that neither anonymity nor privacy is generally preserved by composition and gave a sufficient condition for preservation. However, this word "parallel" merely means that the same agent acts two actions in the paper. That is, concurrency was not considered.

Clarkson and Schneider [11] generalized trace properties to hyperproperties, and Clarkson et al. [10] developed hyperLTL and hyperCTL* for hyperproperties. A trace property is a set of traces. On the other hand, a hyperproperty is a set of sets of traces. Hyperproperties can express security properties which trace properties cannot express. The authors regarded systems as sets of traces, so hyperproperties are properties about systems. Our security properties are also proper hyperproperties. The advantage of our work over these works is relating trace equivalence to attackers' knowledge. In previous work, the relation between equivalence and knowledge is not clear.

5.3 Other Approaches

Canetti et al. [6] defined implementation for task-PIOAs (Probabilistic I/O Automata). According to their definition, $\mathcal{T}_1$ implements $\mathcal{T}_2$ iff the set of behaviors of $\mathcal{T}_1$ composed with $\mathcal{E}$ is included in the set of behaviors of $\mathcal{T}_2$ composed with $\mathcal{E}$ for every environment $\mathcal{E}$. Here, behavior is the set of trace distributions. The implementation relation is preserved by parallel composition.

Giro and D'Argenio [16] pointed out that ordinary schedulers may give rise to unnatural behavior. For example, an attacker can guess secret information to be subject to a particular scheduler. To solve this problem, they provided several reasonable subclasses of schedulers.

Eisentraut et al. [14] also studied subclasses of schedulers for probabilistic automata. They defined late distribution bisimulation and proved that late distribution bisimulation concerning distributed schedulers is compositional. We may need to specify subclasses of schedulers to state a probabilistic variant of Theorem 1.

In this paper, we characterized trace equivalence in terms of our epistemic logic. That is, we showed that a non-adaptive active intruder cannot distinguish trace equivalent processes. We also focused on how composition of systems affects security properties. We proved that any composition preserves total secrecy and role permutativity. This is because trace equivalence is a congruence.

Chapter 6

Conclusions

6.1 Summary

In this paper, we proved that an application of a context preserves trace equivalence for the applied pi calculus. That is, trace equivalence is a congruence. This theorem is our first main result. Although trace equivalence for the π-calculus is not a congruence, trace equivalence for the applied pi calculus is a congruence. This contrast is ascribed to the following two differences. First, names and variables are distinguished in the applied pi calculus, but they are not distinguished in the π-calculus. Second, late instantiation is adopted in the π-calculus, but early instantiation is adopted in the applied pi calculus. We introduced concurrent normal traces to prove it. A concurrent normal trace explicitly indicates scope extrusions and terms each component process sent.

In addition, we provided an epistemic logic for the applied pi calculus. This logic is an LTL-like logic, so we can describe several security notions. We formulated minimal secrecy, total secrecy, role interchangeability, role permutativity, and openness. We regard them as generalized security properties regarding multiagent systems.

Minimal secrecy of x with respect to δ means that an attacker cannot concern that the variable x satisfies the property δ. Total secrecy of x means that an attacker can obtain no information about x. We associated trace equivalence with total secrecy. Role interchangeability (resp. permutativity) means that properties of two (resp. several) variables can be swapped. If x is open, an environment can know the value of x.

Minimal secrecy is not robust. Indeed, an application of a context does not preserve minimal secrecy. However, total secrecy is preserved because trace equivalence is a congruence. We also give necessary and sufficient conditions for role interchangeability and role permutativity, respectively using trace equivalence. Openness is also not robust.

We conclude that trace equivalence is suitable to express non-probabilistic indistinguishability in the view of security in the presence of a non-adaptive active adversary.

6.2 Future Work

First, our epistemic logic states an adversary's knowledge. We intend to construct a logic for a process's knowledge. It will bridge the gap between multiagent systems and process calculi. Ufferman et al. [34] proposed a dynamic epistemic logic whose knowledge updates are expressed as π-calculus processes. However, the relation between process equivalence and logic is unclear. Bisimilar processes do not always happen the same update. We would like to construct a logic such that a certain process equivalence implies that every agent can obtain the same information.

Second, formalizations of other security properties such as non-malleability are also the next topics.

Finally, what logic is suitable for security in the presence of an adaptive attacker is still open.

References

[1] Martín Abadi, Bruno Blanchet, and Cédric Fournet. The applied pi calculus: Mobile values, new names, and secure communication. *arXiv:1609.03003*, 2016.

[2] Martín Abadi, Bruno Blanchet, and Cédric Fournet. The applied pi calculus: Mobile values, new names, and secure communication. *J. ACM*, 65(1):1–41, 2017.

[3] Martín Abadi and Véronique Cortier. Deciding knowledge in security protocols under equational theories. *Theoretical Computer Science*, 367(1-2):2–32, 2006.

[4] Martín Abadi and Andrew D Gordon. A calculus for cryptographic protocols: The spi calculus. *Information and computation*, 148(1):1–70, 1999.

[5] Marco Bernardo, Rocco De Nicola, and Michele Loreti. Revisiting Trace and Testing Equivalences for Nondeterministic and Probabilistic Processes. *Logical Methods in Computer Science*, 10(1), 2014.

[6] Ran Canetti, Ling Cheung, Dilsun Kaynar, Moses Liskov, Nancy Lynch, Olivier Pereira, and Roberto Segala. Task-structured probabilistic I/O automata. *Journal of Computer and System Sciences*, 94:63 – 97, 2018.

[7] Rohit Chadha, Stéphanie Delaune, and Steve Kremer. Epistemic logic for the applied pi calculus. In David Lee, Antónia Lopes, and Arnd Poetzsch-Heffter, editors, *FMOODS/FORTE 2009*, volume 5522 of *LNCS*, pages 182–197, Berlin, Heidelberg, 2009. Springer.

[8] V. Cheval, S. Kremer, and I. Rakotonirina. Deepsec: Deciding equivalence properties in security protocols theory and practice. In *SP 2018*, pages 529–546, 2018.

[9] Vincent Cheval, Véronique Cortier, and Stéphanie Delaune. Deciding equivalence-based properties using constraint solving. *Theoretical Computer Science*, 492:1 – 39, 2013.

[10] Michael R. Clarkson, Bernd Finkbeiner, Masoud Koleini, Kristopher K. Micinski, Markus N. Rabe, and César Sánchez. Temporal logics for hyperproperties. In Martín Abadi and Steve Kremer, editors, *POST 2014*, volume 8414 of *LNCS*, pages 265–284, Berlin, Heidelberg, 2014. Springer.

[11] Michael R Clarkson and Fred B Schneider. Hyperproperties. *Journal of Computer Security*, 18(6):1157–1210, 2010.

[12] Stéphanie Delaune, Steve Kremer, and Mark Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 17(4):435–487, 2009.

[13] Danny Dolev and Andrew C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.

[14] Christian Eisentraut, Jens Chr. Godskesen, Holger Hermanns, Lei Song, and Lijun Zhang. Probabilistic bisimulation for realistic schedulers. In Nikolaj Bjørner and Frank de Boer, editors, *FM 2015*, volume 9109 of *LNCS*, pages 248–264, Cham, 2015. Springer.

[15] M. Fiore and M. Abadi. Computing symbolic models for verifying cryptographic protocols. In *CSFW-14*, pages 160–173, 2001.

[16] Sergio Giro and P.R. D'Argenio. On the expressive power of schedulers in distributed probabilistic systems. *Electronic Notes in Theoretical Computer Science*, 253(3):45 – 71, 2009.

[17] Jean Goubault-Larrecq, Catuscia Palamidessi, and Angelo Troina. A probabilistic applied pi–calculus. In Zhong Shao, editor, *APLAS 2007*, volume 4807 of *LNCS*, pages 175–190, Berlin, Heidelberg, 2007. Springer.

[18] J.Y. Halpern and K.R. O'Neill. Anonymity and information hiding in multiagent systems. In *16th IEEE Computer Security Foundations Workshop, 2003. Proceedings.*, pages 75–88, 2003.

[19] Matthew Hennessy and Robin Milner. On observing nondeterminism and concurrency. In Jaco de Bakker and Jan van Leeuwen, editors, *Automata, Languages and Programming*, volume 85 of *LNCS*, pages 299–309, Berlin, Heidelberg, 1980. Springer.

[20] Ross Horne. A bisimilarity congruence for the applied pi-calculus sufficiently coarse to verify privacy properties. *arXiv:1811.02536*, 2018.

[21] Sophia Knight, Radu Mardare, and Prakash Panangaden. Combining epistemic logic and hennessy-milner logic. In Robert L. Constable and Alexandra Silva, editors, *Logic and Program Semantics*, volume 7230 of *LNCS*, pages 219–243. Springer, Berlin, Heidelberg, 2012.

[22] Sophia Knight, Catuscia Palamidessi, Prakash Panangaden, and Frank D. Valencia. Spatial and epistemic modalities in constraint-based process calculi. In Maciej Koutny and Irek Ulidowski, editors, *CONCUR 2012*, volume 7454 of *LNCS*, pages 317–332, Berlin, Heidelberg, 2012. Springer.

[23] Ken Mano. *Formal Specification and Verification of Anonymity and Privacy.* PhD thesis, Nagoya University, 2013.

[24] Ken Mano, Yoshinobu Kawabe, Hideki Sakurada, and Yasuyuki Tsukada. Role interchange for anonymity and privacy of voting. *Journal of Logic and Computation*, 20(6):1251–1288, 2010.

[25] R. Milner. *Communication and Concurrency.* Prentice-Hall, Inc., USA, 1989.

[26] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, i. *Inf. Comput.*, 100(1):1–40, 1992.

[27] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, ii. *Inf. Comput.*, 100(1):41–77, 1992.

[28] Kiraku Minami. Trace equivalence and epistemic logic to express security properties. In Alexey Gotsman and Ana Sokolova, editors, *Formal Techniques for Distributed Objects, Components, and Systems*, pages 115–132, Cham, 2020. Springer International Publishing.

[29] Joachim Parrow, Johannes Borgström, Lars-Henrik Eriksson, Ramunas Gutkovas, and Tjark Weber. Modal Logics for Nominal Transition Systems. In *CONCUR 2015*, volume 42 of *LIPIcs*, pages 198–211. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2015.

[30] Andrew M. Pitts. Nominal logic, a first order theory of names and binding. *Information and Computation*, 186(2):165–193, 2003. Theoretical Aspects of Computer Software (TACS 2001).

[31] Roberto Segala. A compositional trace-based semantics for probabilistic automata. In Insup Lee and Scott A. Smolka, editors, *CONCUR 1995*, volume 962 of *LNCS*, pages 234–248, Berlin, Heidelberg, 1995. Springer.

[32] Bernardo Toninho and Luís Caires. A spatial-epistemic logic for reasoning about security protocols. In *SecCo 2010*, volume 51 of *EPTCS*, pages 1–15. Open Publishing Association, 2011.

[33] Yasuyuki Tsukada, Hideki Sakurada, Ken Mano, and Yoshifumi Manabe. On compositional reasoning about anonymity and privacy in epistemic logic. *Annals of Mathematics and Artificial Intelligence*, 78(2):101–129, 2016.

[34] Eric Ufferman, Pedro Arturo Góngora, and Francisco Hernández Quiroz. A complete proof system for a dynamic epistemic logic based upon finite π-calculus processes. In *Advances in Modal Logic*, pages 470–482, 2010.

[35] R. J. van Glabbeek. The linear time - branching time spectrum. In J. C. M. Baeten and J. W. Klop, editors, *CONCUR 1990*, volume 458 of *LNCS*, pages 278–297, Berlin, Heidelberg, 1990. Springer.

Appendix A

Proof of Theorem 1

This chapter proves the theorem below by case analysis, and this is our main result.

Theorem 1. $\approx_t$ *is a congruence.*

When A and B are name-variable-distinct and $\mathrm{fn}(A)\cap\mathrm{bn}(B) = \mathrm{bn}(A)\cap\mathrm{fn}(B) = \mathrm{bn}(A) \cap \mathrm{bn}(B) = \emptyset$, we say that A and B are bind-exclusive.

In this appendix, we assume that every process is bind-exclusive. Otherwise, we α-convert processes so that it is satisfied. This appendix uses many results about partial normal forms in [1].

A.1 Partial Normal Forms

Partial normal forms make considering internal reductions easy. The partial normal form of A is written as $\mathrm{pnf}(A)$. We call a process which is in the partial normal form a normal process.

Definition 23. *Let σ and ρ be active substitutions.*

We assume that

$$\sigma|\rho \equiv \{M_1/x_1, ..., M_n/x_n\}$$

where each M_i does not contain $x_1, ..., x_i$ by reordering. We define $\sigma_0 = 0$ and $\sigma_{i+1} = \sigma_i\{M_{i+1}/x_{i+1}\}|\{M_{i+1}/x_{i+1}\}$. Then, we also define $\sigma \uplus \rho = \sigma_n$.

Definition 24. *We suppose that* $\mathrm{pnf}(A) = \nu\widetilde{n}.(\sigma|P)$ *and* $\mathrm{pnf}(B) = \nu\widetilde{m}.(\rho|Q)$*.*

$$\begin{aligned}
\mathrm{pnf}(P) =& 0|P \\
\mathrm{pnf}(\{M/x\}) =& \{M/x\}|0 \\
\mathrm{pnf}(\nu n.A) =& \nu n\widetilde{n}.(\sigma|P) \\
\mathrm{pnf}(\nu x.A) =& \nu\widetilde{n}.(\sigma_{\mathrm{dom}(\sigma)\setminus\{x\}}|P) \\
\mathrm{pnf}(A|B) =& \nu\widetilde{n}\widetilde{m}.(\sigma \uplus \rho|(P|Q)(\sigma \uplus \rho)).
\end{aligned}$$

Note that $A \equiv \mathrm{pnf}(A)$.

A.2 Lemmas for the Proof

First, we summarize lemmas for the proof. Hereafter, $P, Q, ...$ are plain processes, and $A, B, ...$ are extended processes.

Lemma 5. $[P \xrightarrow{\alpha\sigma} A] \Rightarrow [\sigma|P \xrightarrow{\alpha} \sigma|A]$.

Proof. We prove this lemma by case analysis.

1. $\alpha = N(M)$

   By [1, Lemma B.10], $P \equiv \nu\widetilde{n}.(N\sigma(x).P'|P_2)$ and $A \equiv \nu\widetilde{n}.(P'\{M\sigma/x\}|P_2)$ for some $\widetilde{n}, P', P_2$.

   Hence, $\sigma|P \equiv \sigma|\nu\widetilde{n}.(N(x).P'|P_2)$ and $\sigma|A \equiv \sigma|\nu\widetilde{n}.(P'\{M/x\}|P_2)$.

   Therefore, $\sigma|P \xrightarrow{\alpha} \sigma|A$.

2. $\alpha = \nu x.\overline{N}\langle x\rangle$

   By [1, Lemma B.10], $P \equiv \nu\widetilde{n}.(\overline{N\sigma}\langle M\sigma\rangle.P'|P_2)$ and $A \equiv \nu\widetilde{n}.(P'|\{M\sigma/x\}|P_2)$ for some $\widetilde{n}, P', P_2$.

   Hence, $\sigma|P \equiv \sigma|\nu\widetilde{n}.(\overline{N}\langle M\rangle.P'|P_2)$ and $\sigma|A \equiv \sigma|\nu\widetilde{n}.(P'|\{M/x\}|P_2)$.

   Therefore, $\sigma|P \xrightarrow{\alpha} \sigma|A$.

□

Lemma 6.

$$\nu u.A \xrightarrow{\mu} B \textit{ and } A : \text{closed} \textit{ and } \mathrm{fv}(\mu) \subseteq \mathrm{dom}(\nu u.A) \textit{ and } \mathrm{n}(\mu) \cap \mathrm{bn}(\nu u.A) = \emptyset$$
$$\Rightarrow \exists B' \textit{ s.t. } A \xrightarrow{\mu} B' \textit{ and } B \equiv \nu u.B'$$

Proof. Let $\mathrm{pnf}(A) = \nu\widetilde{n}.(\sigma|P)$. We prove this lemma by case analysis.

1. u is a name n, and μ is silent.

   $\mathrm{pnf}(\nu n.A) = \nu n\widetilde{n}.(\sigma|P)$.

   By [1, Lemma B.23], $P \longrightarrow P'$ and $B \equiv \nu n\widetilde{n}.(\sigma|P')$ for some closed P'.

   Therefore, $A \equiv \nu\widetilde{n}.(\sigma|P) \longrightarrow \nu\widetilde{n}.(\sigma|P')$ because internal reductions are closed by an application of an evaluation context.

   $B' = \nu\widetilde{n}.(\sigma|P')$ satisfies this lemma.

2. u is a name n, and μ is a labeled action α.

   $\mathrm{pnf}(\nu n.A) = \nu n\widetilde{n}.(\sigma|P)$.

   By [1, Lemma B.19], $P \xrightarrow{\alpha\sigma} C$ and $B \equiv \nu n\widetilde{n}.(\sigma|C)$ for some C.

   By Lemma 5, $\sigma|P \xrightarrow{\alpha} \sigma|C$.

   Therefore, $A \equiv \nu\widetilde{n}.(\sigma|P) \xrightarrow{\alpha} \nu\widetilde{n}.(\sigma|C)$.

   $B' = \nu\widetilde{n}.(\sigma|C)$ satisfies this lemma.

3. u is a variable x, and μ is silent.

   $\text{pnf}(\nu x.A) = \nu\widetilde{n}.(\sigma'|P)$ where $\sigma' = \sigma_{|dom(\sigma)\setminus\{x\}}$. Note that $\sigma' \equiv \nu x.\sigma$.

   By [1, Lemma B.23], $P \longrightarrow P'$ and $B \equiv \nu\widetilde{n}.(\sigma'|P')$ for some closed P'.

   Therefore, $A \equiv \nu\widetilde{n}.(\sigma|P) \longrightarrow \nu\widetilde{n}.(\sigma|P')$ because internal reductions are closed by an application of an evaluation context.

   $B' = \nu\widetilde{n}.(\sigma|P')$ satisfies this lemma.

4. u is a variable x, and μ is a labelled action α.

   $\text{pnf}(\nu x.A) = \nu\widetilde{n}.(\sigma'|P)$ where $\sigma' = \sigma_{|dom(\sigma)\setminus\{x\}}$.

   By [1, Lemma B.19], $P \xrightarrow{\alpha\sigma'} C$ and $B \equiv \nu\widetilde{n}.(\sigma'|C)$ for some C.

   By Lemma 5, $\sigma'|P \xrightarrow{\alpha} \sigma'|C$.

   Therefore, $A \equiv \nu\widetilde{n}.(\sigma|P) \xrightarrow{\alpha} \nu\widetilde{n}.(\sigma|C)$.

   $B' = \nu\widetilde{n}.(\sigma|C)$ satisfies this lemma.

□

Lemma 7. *Let $\nu\widetilde{n}.(\sigma|P)$ be a closed normal process.*
$\nu\widetilde{n}.(\sigma|P) \xrightarrow{\alpha} A$ *and* $\alpha\sigma = \beta\sigma$ *and* $\text{fv}(\alpha) \subseteq \text{dom}(\sigma)$ *and* $\widetilde{n} \cap (\text{n}(\alpha) \cup \text{n}(\beta)) = \emptyset$
$\Rightarrow \nu\widetilde{n}.(\sigma|P) \xrightarrow{\beta} A$.

Proof. Let $\nu\widetilde{n}.(\sigma|P)$ be a closed normal process.
By [1, Lemma B.19], $P \xrightarrow{\alpha\sigma} B \wedge A \equiv \nu\widetilde{n}.(\sigma|B)$.
By the assumption, $P \xrightarrow{\beta\sigma} B$.
By Lemma 5, $\sigma|P \xrightarrow{\beta} \sigma|B$, so $\nu\widetilde{n}.(\sigma|P) \xrightarrow{\beta} \nu\widetilde{n}.(\sigma|B)$.
Therefore, $\nu\widetilde{n}.(\sigma|P) \xrightarrow{\beta} A$. □

Lemma 8.
$\sigma|A \xrightarrow{\mu} \sigma|B$ *and* $\text{dom}(\sigma) \cap \text{fv}(\mu) = \emptyset$ *and* $[x \in \text{dom}(\sigma) \Rightarrow x\sigma$: *closed]*
$\Rightarrow A\sigma \xrightarrow{\mu} B\sigma$.

Proof. Let $\widetilde{x} = \text{dom}(\sigma)$.
$\nu\widetilde{x}.(\sigma|A) \xrightarrow{\mu} \nu\widetilde{x}.(\sigma|B)$ because $\text{dom}(\sigma) \cap \text{fv}(\mu) = \emptyset$.
That is, $A\sigma \xrightarrow{\mu} B\sigma$. □

Lemma 9. $\sigma|P \xrightarrow{\mu} B$ *and* $\sigma|P$: *closed normal and* $\text{fv}(\mu) \subseteq \text{dom}(\sigma)$
$\Rightarrow \exists B'$ s.t. $P\sigma \xrightarrow{\mu\sigma} B'$ *and* $B \equiv \sigma|B'$.

Proof.

1. μ is silent.

   By [1, Lemma B.23], $P \longrightarrow Q$ and $B \equiv \sigma|Q$.

   By [1, Lemma B.3], $P\sigma \longrightarrow Q\sigma$.

   $\sigma|Q\sigma \equiv \sigma|Q \equiv B$, so $B' \equiv Q\sigma$ satisfies this lemma.

2. μ is a labeled action α.

   By [1, Lemma B.19], $P \xrightarrow{\alpha\sigma} C \wedge B \equiv \sigma | C$ for some C.

   By [1, Lemma B.11], $P\sigma \xrightarrow{\alpha\sigma} C\sigma$.

   $\sigma | C\sigma \equiv \sigma | C \equiv B$, so $B' \equiv C\sigma$ satisfies this lemma.

□

A.3 The Case of Applying a Context without Parallel Composition

We start to prove Theorem 1. First of all, we consider the application of a context without parallel composition. This case is straightforward, and the proof is simple, but some propositions rely on Proposition 3.

First, we define unfolding and safe traces.

Definition 25. *Let* $\mathbf{tr}$ *be a trace* $A_0 \overset{\mu_1}{\Longrightarrow} ... \overset{\mu_n}{\Longrightarrow} A_n$. *We arbitrarily take processes* $A_{01}, ..., A_{0m_0}, ..., A_{n-1,1}, ..., A_{n-1,m_{n-1}}$ *such that*

$$A_0 \longrightarrow A_{01} \longrightarrow ... \longrightarrow A_{0l_0} \xrightarrow{\mu_1} A_{0,l_0+1} \longrightarrow ... \longrightarrow A_{0m_0} = A_1 \longrightarrow ... \longrightarrow A_n$$

and each transition is derived without α*-conversion. If* μ_i *is silent,* m_i *can be 0. The above derivation is denoted by* $\text{unfold}(\mathbf{tr})$.

Every internal reduction is explicitly displayed in $\text{unfold}(\mathbf{tr})$. This form is convenient for analyzing traces. Note that $\text{unfold}(\mathbf{tr})$ is not always unique.

Definition 26. *A trace* $\mathbf{tr}$ *is safe with respect to* A. $\overset{\text{def}}{\Leftrightarrow}$ *Every action in* $\mathbf{tr}$ *contains no elements in* $\text{bn}(A) \cup \text{bv}(A)$, *and each transition is derived without* α*-conversion.*

If $\mathbf{tr}$ *is a trace of* A, *we merely say that* $\mathbf{tr}$ *is safe or* $\mathbf{tr}$ *is a safe trace.*

Example 10.

$$\nu m.a(x).\bar{b}\langle f(x,m)\rangle \xrightarrow{a(m)} \nu n.\bar{b}\langle f(m,n)\rangle \xrightarrow{\nu y.\bar{b}\langle y\rangle} \nu n.\{f(m,n)/y\}$$

This trace is not safe because α*-conversion happened in the first transition.*

We give another definition of trace equivalence.

Definition 27. *Let* A *and* B *be two closed name-variable-distinct extended processes.* $A \sqsubseteq_t' B$ *if and only if we can always complete the procedure below.*

1. *We* α*-convert* A *and* B *such that* A *and* B *are bind-exclusive.*

2. *We arbitrarily choose a trace* $\mathbf{tr}$ *of* A *which is safe with respect to* A *and* B.

3. *We take a safe trace of* B *which is statically equivalent to* $\mathbf{tr}$.

When $A \subseteq'_t B$ and $B \subseteq'_t A$, we denote by $A \approx'_t B$.

Let A and B be two name-variable-distinct extended processes. We α-convert A and B such that A and B are bind-exclusive. Let σ be a map that maps a variable in $\mathrm{sv}(A) \cup \mathrm{sv}(B)$ to a ground term. When $A\sigma \subseteq'_t B\sigma$ for any σ and capture-avoiding, we also say that they are trace equivalent and denote as $A \subseteq'_t B$. $A \approx'_t B$ is similarly defined.

Note that σ causes capture-avoiding substitution.

This definition may look strange, but this is equivalent to the previous definition.

Proposition 13.

Let A and B be two closed name-variable-distinct extended processes.

$$A \subseteq'_t B \Leftrightarrow A \subseteq_t B$$

Proof. $\Rightarrow$) We arbitrarily take a trace $\mathbf{tr}$ of A.

First, we α-convert A and B to A' and B' respectively such that A' and B' are bind-exclusive and they bind no names in actions that are in $\mathbf{tr}$. We replace A with A' in $\mathbf{tr}$.

Secondly, we convert processes in $\mathbf{tr}$ to structurally equivalent processes such that

$$C \xRightarrow{\mu} D \text{ in } \mathbf{tr} \ \Rightarrow \mathrm{fn}(D) \subseteq \mathrm{fn}(C) \cup \mathrm{n}(\mu).$$

Thirdly, we α-convert processes in $\mathbf{tr}$ such that every transition is derived without α-conversion.

Now, we got a safe trace $\mathbf{tr}''$ of A', so we can obtain a safe trace $\mathbf{tr}'$ of B' such that $\mathbf{tr}'' \sim_t \mathbf{tr}'$ because of $A \subseteq'_t B$.

Then, we α-convert B' in $\mathbf{tr}''$ to B and obtain a trace of B that is statically equivalent to $\mathbf{tr}$.

$\Leftarrow$) We α-convert A and B to A' and B' respectively such that A' and B' are bind-exclusive.

Next, we arbitrarily choose a trace $\mathbf{tr}$ of A' which is safe with respect to A' and B'.

We α-convert A' in $\mathbf{tr}$ to A.

By assumption, we obtain a trace $\mathbf{tr}'$ of B that is statically equivalent to $\mathbf{tr}$.

We replace B with B' in $\mathbf{tr}'$.

Moreover, we convert processes in $\mathbf{tr}'$ to structural equivalent processes such that

$$C \xRightarrow{\mu} D \text{ in } \mathbf{tr}' \ \Rightarrow \mathrm{fn}(D) \subseteq \mathrm{fn}(C) \cup \mathrm{n}(\mu).$$

In addition, we α-convert processes in $\mathbf{tr}'$ such that every transition is derived without α-conversion.

Now, we got a safe trace of B' that is statically equivalent to $\mathbf{tr}$. □

Hereafter, we adopt Definition 27 as trace equivalence. That is, $\approx_t$ means $\approx'_t$ in Definition 27.

Proposition 14. $P \approx_t Q \Rightarrow \overline{M}\langle N\rangle.P \approx_t \overline{M}\langle N\rangle.Q$

Proof. Assume that $P \approx_t Q$.

It is sufficient to prove that $\overline{M\sigma}\langle N\sigma\rangle.P\sigma \approx_t \overline{M\sigma}\langle N\sigma\rangle.Q\sigma$ for all assignments σ. Thus, we can suppose that P, Q, M, and N are closed without loss of generality. We arbitrarily take a safe trace **tr** of $\overline{M}\langle N\rangle.P$. Here, unfold(**tr**) is

$$\overline{M}\langle N\rangle.P \overset{\nu x.\overline{M}\langle x\rangle}{\longrightarrow} A \xrightarrow{\mu} ...$$

and $A \equiv P|\{N/x\}$ for some x.

By Proposition 3, $P|\{N/x\} \approx_t Q|\{N/x\}$, so there exists a trace of $Q|\{N/x\}$ which is statically equivalent to the part $A \xrightarrow{\mu} ...$ of unfold(**tr**). We add $\overline{M}\langle N\rangle.Q \overset{\nu x.\overline{M}\langle x\rangle}{\longrightarrow}$ to it, and we get a trace of $\overline{M}\langle N\rangle.Q$ that is statically equivalent to unfold(**tr**). We omit some silent actions and get the desired trace. Hence, $\overline{M}\langle N\rangle.P \subseteq_t \overline{M}\langle N\rangle.Q$ and vice versa. □

Proposition 15. $P \approx_t Q \Rightarrow M(x).P \approx_t M(x).Q$

Proof. Assume that $P \approx_t Q$.

For all assignments σ, it is sufficient to prove that $M\sigma(x).P\sigma \approx_t M\sigma(x).Q\sigma$. Thus, we can suppose that M is closed and $\mathrm{fv}(P) \cup \mathrm{fv}(Q) \subseteq \{x\}$ without loss of generality. Note that x is not mapped by σ because it is not free in $M(x).P$ and $M(x).Q$.

We arbitrarily take a safe trace **tr** of $M(x).P$. Here, unfold(**tr**) is

$$M(x).P \xrightarrow{M(N)} P' \xrightarrow{\mu} ...$$

and $P' \equiv P\{{}^{N}/_{x}\}$ for some ground N.

By $P\{{}^{N}/_{x}\} \approx_s Q\{{}^{N}/_{x}\}$, there exists a trace of $Q\{{}^{N}/_{x}\}$ which is statically equivalent to the part $P' \xrightarrow{\mu} ...$ of unfold(**tr**). We add $M(x).Q \xrightarrow{M(N)}$ to it, and we get the desired trace of $M(x).Q$ that is statically equivalent to unfold(**tr**). We omit some silent actions and get the desired trace. Hence, $M(x).P \subseteq_t M(x).Q$ and vice versa. □

Proposition 16. $A \approx_t B \Rightarrow \nu u.A \approx_t \nu u.B$ *(When u is a variable, $u \in \mathrm{dom}(A)$.)*

Proof. Assume that $A \approx_t B$.

For all assignments σ, it is sufficient to prove that $(\nu u.A)\sigma \approx_t (\nu u.B)\sigma$. Thus, we can suppose that A and B are closed without loss of generality. We arbitrarily take a safe trace **tr** of $\nu u.A$. Here, unfold(**tr**) is

$$\nu u.A \xrightarrow{\mu_1} C_1 \xrightarrow{\mu_2} ...$$

By Lemma 6, there exists C_1' such that $A \xrightarrow{\mu_1} C_1'$ and $C_1 \equiv \nu u.C_1'$. Moreover, we use Lemma 6 repeatedly, and we get

$$\begin{array}{l} \mathrm{unfold}(\mathbf{tr}) \sim_t \nu u.A \xrightarrow{\mu_1} \nu u.C_1' \xrightarrow{\mu_2} \nu u.C_2' \xrightarrow{\mu_3} ... \\ \qquad\qquad\qquad\quad A \xrightarrow{\mu_1} \qquad C_1' \xrightarrow{\mu_2} \qquad C_2' \xrightarrow{\mu_3} ... \end{array}$$

We take a trace of B which is statically equivalent to the second line and add $\nu u.$ to each process. This processing is possible because **tr** is safe. Recall that every action in a safe trace contains no bound names and bound variables in the first process. Hence, μ_i does not contain u.

$$\begin{array}{c}
B \xrightarrow{\mu_1} \quad D_1 \xrightarrow{\mu_2} \quad D_2 \xrightarrow{\mu_3} ... \\
\nu u.B \xrightarrow{\mu_1} \nu u.D_1 \xrightarrow{\mu_2} \nu u.D_2 \xrightarrow{\mu_3} ...
\end{array}$$

The last line is statically equivalent to unfold(**tr**). Note that restriction preserves static equivalence. We omit some internal reductions and get the desired trace. Hence, $\nu u.A \subseteq_t \nu u.B$ and vice versa. □

Proposition 17. $P \approx_t Q \Rightarrow$ if $M = N$ then P else $R \approx_t$ if $M = N$ then Q else R.

Proof. Assume that $P \approx_t Q$. For all assignments σ, it is sufficient to prove that

$$\text{if } M\sigma = N\sigma \text{ then } P\sigma \text{ else } R\sigma \approx_t \text{if } M\sigma = N\sigma \text{ then } Q\sigma \text{ else } R\sigma.$$

Thus, we can suppose that P, Q, R, M, and N are closed without loss of generality. We arbitrarily take a safe trace **tr** of if $M = N$ then P else R. Here, unfold(**tr**) is

$$\text{if } M = N \text{ then } P \text{ else } R \longrightarrow A \xrightarrow{\mu} ...$$

We know that $A \equiv P$ or $A \equiv R$, so we can regard $A \xrightarrow{\mu} ...$ as a trace of P or R. In the former case, there exists a trace of Q which is statically equivalent to it. We add if $M = N$ then Q else $R \longrightarrow$ to it and get a trace of if $M = N$ then Q else R which is statically equivalent to unfold(**tr**).

In the latter case, we add if $M = N$ then Q else $R \longrightarrow$ to $A \xrightarrow{\mu} ...$ and get a trace of if $M = N$ then Q else R which is statically equivalent to unfold(**tr**). We omit some internal reductions and get the desired trace. Hence,

$$\text{if } M = N \text{ then } P \text{ else } R \subseteq_t \text{if } M = N \text{ then } Q \text{ else } R$$

and vice versa. □

Proposition 18. $P \approx_t Q \Rightarrow P + R \approx_t Q + R$.

Proof. Assume that $P \approx_t Q$. For all assignments σ, it is sufficient to prove that

$$P\sigma + R\sigma \approx_t Q\sigma + R\sigma.$$

Thus, we can suppose that P, Q, and R are closed without loss of generality.

We arbitrarily take a safe trace **tr** of $P + R$. Here, unfold(**tr**) is $P + R \longrightarrow A \xrightarrow{\mu} ....$ We know that $A \equiv P$ or $A \equiv R$, so we can regard $A \xrightarrow{\mu} ...$ as a trace of P or R. In the former case, there exists a trace of Q which is statically equivalent to it. We add $Q+R \longrightarrow$ to it and get a trace of $Q+R$ which is statically equivalent to unfold(**tr**).

In the latter case, we add $Q + R \longrightarrow$ to $A \xrightarrow{\mu} ...$ and get a trace of $Q + R$ which is statically equivalent to unfold(**tr**). We omit some internal reductions and get the desired trace. Hence, $P + R \subseteq_t Q + R$ and vice versa. □

The case of replication needs several lemmas.

Lemma 10. *If A and P are closed and $A|!P \longrightarrow B$, it holds one of the following:*

1. $\exists A'$ s.t. $A \longrightarrow A'$ *and* $B \equiv A'|!P$.
2. $\exists P''$ s.t. $P|P \longrightarrow P''$ *and* $B \equiv A|P''|!P$.
3. $\exists E$ s.t. $A|P \longrightarrow E$ *and* $B \equiv E|!P$.

Proof. Let $\mathrm{pnf}(A) = \nu\widetilde{n}.(\sigma|Q)$.
Then, $\mathrm{pnf}(A|!P) = \nu\widetilde{n}.(\sigma|Q|!P)$. Note that $\mathrm{n}(P) \cap \widetilde{n} = \emptyset$.
By [1, Lemma B.23], $Q|!P \longrightarrow R$ and $B \equiv \nu\widetilde{n}.(\sigma|R)$ for some closed R.
By [1, Lemma B.24], we consider the following four cases:

1. $Q \longrightarrow Q'$ and $R \equiv Q'|!P$ for some closed Q'.
   $A \equiv \nu\widetilde{n}.(\sigma|Q) \longrightarrow \nu\widetilde{n}.(\sigma|Q')$ and $\nu\widetilde{n}.(\sigma|Q')|!P \equiv \nu\widetilde{n}.(\sigma|Q'|!P) \equiv B$.
   Then, $A' = \nu\widetilde{n}.(\sigma|Q')$ satisfies case 1 of this lemma.

2. $!P \longrightarrow P'$ and $R \equiv Q|P'$ for some closed P'.
   By [1, Lemma B.24], $P|P \longrightarrow P''$ and $P' \equiv P''|!P$ for some closed P''.
   $B \equiv \nu\widetilde{n}.(\sigma|R) \equiv \nu\widetilde{n}.(\sigma|Q)|P''|!P \equiv A|P''|!P$, so case 1 of this lemma is satisfied.

3. $Q \xrightarrow{N(x)} B'$ and $!P \xrightarrow{\nu x.\overline{N}\langle x\rangle} C$ and $R \equiv \nu x.(B'|C)$ for some B', C, x, and ground N.
   By [1, Lemma B.18], $P \xrightarrow{\nu x.\overline{N}\langle x\rangle} D$ and $C \equiv D|!P$ for some D.
   Then,
$$\begin{aligned}A|!P \longrightarrow B \equiv &\nu\widetilde{n}.(\sigma|\nu x.(B'|D|!P))\\ \equiv &\nu\widetilde{n}.(\sigma|\nu x.(B'|D)|!P)\\ \equiv &\nu\widetilde{n}.(\sigma|\nu x.(B'|D))|!P\end{aligned}$$
   Thus, $E = \nu\widetilde{n}.(\sigma|\nu x.(B'|D))$ satisfies case 3 of this lemma.

4. $!P \xrightarrow{N(x)} B'$ and $Q \xrightarrow{\nu x.\overline{N}\langle x\rangle} C$ and $R \equiv \nu x.(B'|C)$ for some B', C, x, and ground N.
   By [1, Lemma B.18], $P \xrightarrow{N(x)} D$ and $B' \equiv D|!P$ for some D.
   Then,
$$\begin{aligned}A|!P \longrightarrow B \equiv &\nu\widetilde{n}.(\sigma|\nu x.(D|!P|C))\\ \equiv &\nu\widetilde{n}.(\sigma|\nu x.(D|C)|!P)\\ \equiv &\nu\widetilde{n}.(\sigma|\nu x.(D|C))|!P\end{aligned}$$
   Thus, $E = \nu\widetilde{n}.(\sigma|\nu x.(D|C))$ satisfies case 3 of this lemma.

□

Corollary 2. *If A and P are closed and $A|!P \longrightarrow B$, then $A|P|P \longrightarrow C$ and $B \equiv C|!P$ for some C.*

Lemma 11. *If A and P are closed and $A|!P \xrightarrow{\alpha} B$ and $\mathrm{fv}(\alpha) \subseteq \mathrm{dom}(A)$ and $\mathrm{n}(\alpha) \cap \mathrm{bn}(A|!P) = \emptyset$, it holds one of the following:*

1. $\exists E$ s.t. $A \xrightarrow{\alpha} E$ *and* $B \equiv E|!P$.
2. $\exists F$ s.t. $A|P \xrightarrow{\alpha} F$ *and* $B \equiv F|!P$.

Proof. Let $\mathrm{pnf}(A) = \nu\widetilde{n}.(\sigma|Q)$.
Then, $\mathrm{pnf}(A|!P) = \nu\widetilde{n}.(\sigma|Q|!P)$.
By [1, Lemma B.19], $Q|!P \xrightarrow{\alpha\sigma} C$ and $B \equiv \nu\widetilde{n}.(\sigma|C)$ for some C.
By [1, Lemma B.18], we consider the following two cases:

1. $Q \xrightarrow{\alpha\sigma} D$ and $C \equiv D|!P$ for some D.

   By Lemma 5, $\sigma|Q \xrightarrow{\alpha} \sigma|D$.

   Thus, $A \equiv \nu\widetilde{n}.(\sigma|Q) \xrightarrow{\alpha} \nu\widetilde{n}.(\sigma|D)$.

   In addition, $\nu\widetilde{n}.(\sigma|D)|!P \equiv \nu\widetilde{n}.(\sigma|D|!P) \equiv B$, so $E = \nu\widetilde{n}.(\sigma|D)$ satisfies case 1 of this lemma.

2. $!P \xrightarrow{\alpha\sigma} D$ and $C \equiv Q|D$ for some D.

   By [1, Lemma B.18], $P \xrightarrow{\alpha\sigma} E$ and $D \equiv E|!P$ for some E.

   By Lemma 5, $\sigma|P \xrightarrow{\alpha} \sigma|E$.

   Thus, $A|P \equiv \nu\widetilde{n}.(\sigma|Q)|P \xrightarrow{\alpha} \nu\widetilde{n}.(\sigma|Q|E)$.

   In addition, $\nu\widetilde{n}.(\sigma|Q|E)|!P \equiv \nu\widetilde{n}.(\sigma|Q|E|!P) \equiv B$, so $F = \nu\widetilde{n}.(\sigma|Q|E)$ satisfies case 2 of this lemma.

□

Corollary 3. *If A and P are closed and $A|!P \xrightarrow{\alpha} B$ and $\mathrm{fv}(\alpha) \subseteq \mathrm{dom}(A)$ and $\mathrm{n}(\alpha) \cap \mathrm{bn}(A|!P) = \emptyset$, then $A|P \xrightarrow{\alpha} C$ and $B \equiv C|!P$ for some C.*

Proposition 19. *$P \approx_t Q \Rightarrow !P \ \approx_t \ !Q$.*

Proof. Assume that $P \approx_t Q$.

For all assignments σ, it is sufficient to prove that $!P\sigma \approx_t !Q\sigma$. Thus, we can suppose that P and Q are closed without loss of generality.

We arbitrarily take a safe trace **tr** of $!P$. By corollary 2 and 3, we obtain a trace **tr**$'$ which is of the form

$$P^n|!P \xrightarrow{\mu_1} A_1|!P \xrightarrow{\mu_2} A_2|!P \xrightarrow{\mu_3} ...$$

for some n and statically equivalent to unfold(**tr**). We also obtain

$$P^n \xrightarrow{\mu_1} A_1 \xrightarrow{\mu_2} A_2 \xrightarrow{\mu_3} ...$$

to remove $!P$ from each process in $\mathbf{tr}'$. Here, n depends on $\mathbf{tr}$, and P^n is n concurrent processes. Strictly speaking, processes in P^n are not the same when $\mathrm{bn}(P) \neq \emptyset$. In this case, we α-convert P and make processes in P^n bind-exclusive.

By Proposition 3, $P^n \approx_t Q^n$, so there exists a trace of Q^n which is statically equivalent to

$$P^n \xrightarrow{\mu_1} A_1 \xrightarrow{\mu_2} A_2 \xrightarrow{\mu_3} ...$$

We add $|!Q$ to each process in the trace, omit some internal reductions and get the desired trace. Hence, $!P \subseteq_t !Q$ and vice versa. □

A.4 The Case of Parallel Composition

A.4.1 A Definition of a Concurrent Normal Trace

Proposition 3. *$A \approx_t B \Rightarrow A|C \approx_t B|C$.*

In this section, we always assume that structural equivalent processes are constructed without α-conversion.

We define action between active substitutions and also define concurrent normal forms of traces.

Definition 28.

$$[\sigma\rho] = (\sigma \uplus \rho)_{|\mathrm{dom}(\sigma)}$$

Lemma 12. *$\sigma\rho \equiv [\sigma\rho]$.*

Proof. Let $\widetilde{y} = \mathrm{dom}(\rho)$. Then, $\sigma\rho \equiv \nu\widetilde{y}.(\sigma|\rho) \equiv \nu\widetilde{y}.(\sigma \uplus \rho) \equiv [\sigma\rho]$. □

Lemma 13. *$\rho\sigma \equiv \rho[\sigma\rho]$.*

Proof. Let $\widetilde{x} = \mathrm{dom}(\sigma)$. Then,

$$\begin{aligned}\rho\sigma &\equiv \nu\widetilde{x}.(\rho|\sigma) \\ &\equiv \nu\widetilde{x}.(\rho|\sigma\rho) \\ &\equiv \nu\widetilde{x}.(\rho|[\sigma\rho]) \quad \text{By Lemma 12} \\ &\equiv \rho[\sigma\rho].\end{aligned}$$

□

Lemma 14. *$(\sigma|P)\rho \equiv [\sigma\rho]|P[\rho\sigma]$.*

Proof. Let $\widetilde{y} = \mathrm{dom}(\rho)$. Then,

$$\begin{aligned}(\sigma|P)\rho &= \sigma\rho|P\rho \\ &\equiv [\sigma\rho]|P\rho \\ &\equiv [\sigma\rho]|P\rho[\sigma\rho] \\ &\equiv [\sigma\rho]|\nu\widetilde{y}.(P|\rho[\sigma\rho]) \\ &\equiv [\sigma\rho]|\nu\widetilde{y}.(P|\rho\sigma) \quad \text{By Lemma 13} \\ &\equiv [\sigma\rho]|P\rho\sigma.\end{aligned}$$

□

Concurrent normal forms are designed to record communication completely.

Definition 29. *A concurrent normal trace* **tr** *of $A|C$ is a trace which satisfies the following conditions.*

1. **tr** *is full and safe.*
2. *Each process in* **tr** *is of the form $\nu\widetilde{r_m}\widetilde{s_m}.(\nu\widetilde{x_m}.A_m\rho_m|\nu\widetilde{y_m}.C_m\sigma_m)$.*

   In addition, The conditions below are satisfied.

   - $\mathrm{fv}(A_m) \setminus \mathrm{dom}(A_m) \subseteq \mathrm{dom}(\rho_m)$ *and* $\mathrm{fv}(C_m) \setminus \mathrm{dom}(C_m) \subseteq \mathrm{dom}(\sigma_m)$.
   - $\mathrm{n}(\rho_m) \cap \widetilde{s_m} = \emptyset$ *and* $\mathrm{n}(\sigma_m) \cap \widetilde{r_m} = \emptyset$.
   - $A_m = \sigma_m|P_m$ *and* $C_m = \rho_m|Q_m$ *and they are normal for some* P_m *and* Q_m*, and* $\mathrm{n}(Q_m) \cap \widetilde{s_m} = \emptyset$ *and* $\mathrm{n}(P_m) \cap \widetilde{r_m} = \emptyset$.

   Let $D_m = \nu\widetilde{r_m}\widetilde{s_m}.(\nu\widetilde{x_m}.A_m\rho_m|\nu\widetilde{y_m}.C_m\sigma_m)$.
3. *For every* $D_m \longrightarrow D_{m+1}$ *in* **tr**, *it holds one of the following:*

   (a) $A_m\rho_m \longrightarrow A_{m+1}\rho_m$ *and elements in* D_{m+1} *except for* A_{m+1} *are same as counterparts in* D_m.

   (b) $C_m\sigma_m \longrightarrow C_{m+1}\sigma_m$ *and elements in* D_{m+1} *except for* C_{m+1} *are same as counterparts in* D_m.

   (c)
   - $A_m\rho_m \xrightarrow{N'\rho_m(x)} A_{m+1}\rho_m$ *and* $C_m\sigma_m \xrightarrow{\nu x.\overline{N'\sigma_m}\langle x\rangle} \nu\widetilde{m}.C_{m+1}\sigma_m$,
   - $\widetilde{r_{m+1}} = \widetilde{r_m}\widetilde{m}$,
   - $\rho_{m+1} = \rho_m \uplus \{M/x\}$,
   - $\widetilde{m} = \mathrm{n}(M) \cap \mathrm{bn}(C_m)$,
   - $(N'\rho_m)[\sigma_m\rho_m] = (N'\sigma_m)[\rho_m\sigma_m]$*: ground,*
   - $\widetilde{r_m} \cap \mathrm{n}(N') = \emptyset$ *and* $\widetilde{s_m} \cap \mathrm{n}(N') = \emptyset$,
   - $M[\sigma_m\rho_m]$*: ground,*
   - $\widetilde{y_{m+1}} = \widetilde{y_m}x$,
   - $\mathrm{n}(M) \cap \widetilde{s_m} = \emptyset$,

   for some N', M*, and* x*, and the other parts of* D_{m+1} *are the same as counterparts in* D_m.

   (d)
   - $C_m\sigma_m \xrightarrow{N'\sigma_m(x)} C_{m+1}\sigma_m$,
   - $A_m\rho_m \xrightarrow{\nu x.\overline{N'\rho_m}\langle x\rangle} \nu\widetilde{m}.A_{m+1}\rho_m$,
   - $\widetilde{s_{m+1}} = \widetilde{s_m}\widetilde{m}$,
   - $\sigma_{m+1} = \sigma_m \uplus \{M/x\}$,
   - $\widetilde{m} = \mathrm{n}(M) \cap \mathrm{bn}(A_m)$,
   - $(N'\rho_m)[\sigma_m\rho_m] = (N'\sigma_m)[\rho_m\sigma_m]$*: ground,*
   - $\widetilde{s_m} \cap \mathrm{n}(N') = \emptyset$,
   - $\widetilde{r_m} \cap \mathrm{n}(N') = \emptyset.$,

- $M[\rho_m\sigma_m]$: *ground,*
- $\widetilde{x_{m+1}} = \widetilde{x_m}x$,
- $\mathrm{n}(M) \cap \widetilde{r_m} = \emptyset$,

for some N', M, *and* x, *and the other parts of* D_{m+1} *are the same as counterparts in* D_m.

4. *For every* $D_m \overset{N(M)}{\longrightarrow} D_{m+1}$ *in* $\mathbf{tr}$*, it holds one of the following:*

   (a) $A_m\rho_m \overset{N(M)[\rho_m\sigma_m]}{\longrightarrow} A_{m+1}\rho_m$ *and elements in* D_{m+1} *except for* A_{m+1} *are the same as counterparts in* D_m.

   (b) $C_m\sigma_m \overset{N(M)[\sigma_m\rho_m]}{\longrightarrow} C_{m+1}\sigma_m$ *and elements in* D_{m+1} *except for* C_{m+1} *are the same as counterparts in* D_m.

5. *For every* $D_m \overset{\nu x.\overline{N}\langle x\rangle}{\longrightarrow} D_{m+1}$ *in* $\mathbf{tr}$*, it holds one of the following:*

   (a)
   - $A_m\rho_m \overset{\nu x.\overline{N}\langle x\rangle[\rho_m\sigma_m]}{\longrightarrow} \nu\widetilde{m}.A_{m+1}\rho_m$,
   - $\sigma_{m+1} = \sigma_m \uplus \{M/x\}$,
   - $\widetilde{s_{m+1}} = \widetilde{s_m}\widetilde{m}$,
   - $\widetilde{m} \subseteq \mathrm{n}(M)$
   - $M[\rho_m\sigma_m]$: *ground,*
   - $\mathrm{n}(M) \cap \widetilde{r_m} = \emptyset$,

   and the other parts of D_{m+1} *are the same as counterparts in* D_m.

   (b)
   - $C_m\sigma_m \overset{\nu x.\overline{N}\langle x\rangle[\sigma_m\rho_m]}{\longrightarrow} \nu\widetilde{m}.C_{m+1}\sigma_m$,
   - $\rho_{m+1} = \rho_m \uplus \{M/x\}$,
   - $\widetilde{r_{m+1}} = \widetilde{r_m}\widetilde{m}$,
   - $\widetilde{m} \subseteq \mathrm{n}(M)$,
   - $M[\sigma_m\rho_m]$: *ground,*
   - $\mathrm{n}(M) \cap \widetilde{s_m} = \emptyset$,

   and the other parts of D_{m+1} *are the same as counterparts in* D_m.

6. (a) *At 3a, let* A'_m *be a process obtained by applying* ρ_m *to only for the part related to the transition from* A_m*. Then,* $A'_m \longrightarrow A_{m+1}$*, and 3b is similar.*

   (b) *At 3c, let* A'_m *be a process obtained by substituting only for the part related to the transition from* A_m*. Then,* $A'_m \overset{(N'\rho_m)[\sigma_m\rho_m](x)}{\longrightarrow} A_{m+1}$*.* C_m *is similar, and 3d is similar.*

   (c) *At 4a, let* A'_m *be a process obtained by substituting only for the part related to the transition from* A_m*. Then,* $A'_m \overset{N(\rho_m\uplus\sigma_m)(M)}{\longrightarrow} A_{m+1}$*, and 4b is similar.*

(d) At 5a, let A'_m be a process obtained by substituting only for the part related to the transition from A_m. Then, $A'_m \xrightarrow{\nu x.\overline{N(\rho_m \uplus \sigma_m)}\langle x\rangle} A_{m+1}$, and 5b is similar.

What does the form $\nu\widetilde{r_m}\widetilde{s_m}.(\nu\widetilde{x_m}.A_m\rho_m|\nu\widetilde{y_m}.C_m\sigma_m)$ express? First, $\widetilde{r_m}$ is a sequence of bound names which is sent to the left process or the environment by the right process. Similarly, $\widetilde{s_m}$ is a sequence of bound names sent by the left process.

Usually, synchronous communication does not have to generate an active substitution. Nevertheless, concurrent normal forms require synchronous communication to generate an active substitution, and the alias variable is restricted. As a result, σ_m displays terms the left process sent to the right process or the environment. Moreover, names appearing in σ_m are not in $\widetilde{r_m}$ because σ_m is defined from the point of view of the left process.

For instance,

$$\nu rs.(\nu x.(\{^s/_x\}|\overline{f(y,s)}\langle m\rangle)\{^r/_y\}|\nu y.(\{^r/_y\}|f(r,x)(z))\{^s/_x\})$$

satisfies the condition 2. The left process sent s, and the right process sent r. They used synchronous communication, so x and y are restricted respectively, and each scope is spreaded.

A.4.2 A Transformation into a Concurrent Normal Form

We can transform all safe traces of a parallel composed process into a concurrent normal form.

Lemma 15. *For any full safe trace of $A|C$, there exists a concurrent normal trace of $A|C$ such that they are statically equivalent.*

Proof. We transform the given trace in order from the top. Assuming that we transformed the first m-th process.

We suppose that $\nu\widetilde{r}\widetilde{s}.(\nu\widetilde{x}.A_m\rho|\nu\widetilde{y}.C_m\sigma) \xrightarrow{\mu} D$.

By Lemma 6, $\nu\widetilde{x}.A_m\rho|\nu\widetilde{y}.C_m\sigma \xrightarrow{\mu} D'$ and $D \equiv \nu\widetilde{r}\widetilde{s}.D'$.

Let $A_m = \sigma|P$ and $C_m = \rho|Q$. Note that $A_m\rho \equiv [\sigma\rho]|P[\rho\sigma]$ and $C_m\sigma \equiv [\rho\sigma]|Q[\sigma\rho]$ by Lemma 14.

Then, $\nu\widetilde{x}\widetilde{y}.\sigma \uplus \rho|P[\rho\sigma]|Q[\sigma\rho] \xrightarrow{\mu} D'$.

That is, $(\sigma \uplus \rho)_{|dom(\sigma\uplus\rho)\setminus\widetilde{x}\widetilde{y}}|P[\rho\sigma]|Q[\sigma\rho] \xrightarrow{\mu} D'$.

First, we consider when μ is silent.

By [1, Lemma B.23], $P[\rho\sigma]|Q[\sigma\rho] \longrightarrow R$ and $D' \equiv (\sigma \uplus \rho)_{|dom(\sigma\uplus\rho)\setminus\widetilde{x}\widetilde{y}}|R$ for some closed R.

We consider four cases.

1. $P[\rho\sigma] \longrightarrow P'[\rho\sigma]$ and $R \equiv P'[\rho\sigma]|Q[\sigma\rho]$ for some P' where $\mathrm{fv}(P') \subseteq \mathrm{dom}(\rho)$. Furthermore, let P'' be a process obtained by applying $[\rho\sigma]$ to only for the part related to the transition from P. Then, $P'' \longrightarrow P'$. This reduction is possible if we suitably choose P'.

$A_m\rho \longrightarrow [\sigma\rho]|P'[\rho\sigma] \equiv (\sigma|P')\rho.$

Hence,

$$\begin{aligned}\nu\widetilde{r}\widetilde{s}.(\nu\widetilde{x}.A_m\rho|\nu\widetilde{y}.C_m\sigma) \longrightarrow D &\equiv \nu\widetilde{r}\widetilde{s}.((\sigma \uplus \rho)_{|dom(\sigma\uplus\rho)\setminus\widetilde{x}\widetilde{y}}|P'[\rho\sigma]|Q[\sigma\rho]) \\ &\equiv \nu\widetilde{r}\widetilde{s}.(\nu\widetilde{x}.(\sigma|P')\rho|\nu\widetilde{y}.C_m\sigma).\end{aligned}$$

This is the desired form.

2. $Q[\sigma\rho] \longrightarrow Q'[\sigma\rho]$ and $R \equiv P[\rho\sigma]|Q'[\sigma\rho]$ for some Q' where $\mathrm{fv}(Q') \subseteq \mathrm{dom}(\sigma)$. Furthermore, let Q'' be a process obtained by applying $[\sigma\rho]$ to only for the part related to the transition from Q. Then, $Q'' \longrightarrow Q'$. This reduction is possible if we suitably choose Q'.

   This case is similar to case 1.

3. $P[\rho\sigma] \xrightarrow{N(x)} E$ and $Q[\sigma\rho] \xrightarrow{\nu x.\overline{N}\langle x\rangle} F$ and $R \equiv \nu x.(E|F)$ for some E, F, x, and ground N.

   In this case, $A_m\rho \xrightarrow{N'\rho(x)} [\sigma\rho]|E$ and $C_m\sigma \xrightarrow{\nu x.\overline{N'\sigma}\langle x\rangle} [\rho\sigma]|F$ for some N' where $(N'\rho)[\sigma\rho] = N$ and $\widetilde{r} \cap \mathrm{n}(N') = \emptyset$ and $(N'\sigma)[\rho\sigma] = N$ and $\widetilde{s} \cap \mathrm{n}(N') = \emptyset$.

   By [1, Lemma B.10], there exist $\widetilde{l}, P'$ and P_2 such that

$$P[\rho\sigma] \equiv (\nu\widetilde{l}.(N'\sigma(x).P'|P_2))[\rho\sigma] \text{ and } E \equiv (\nu\widetilde{l}.(P'|P_2))[\rho\sigma].$$

   Hence $A_m\rho \xrightarrow{N'\rho(x)} (\sigma|\nu\widetilde{l}.(P'|P_2))\rho$.

   We use again [1, Lemma B.10].

   There exists $\widetilde{m}, \widetilde{t}, M, Q'$, and Q_2 such that

$$\begin{aligned}Q[\sigma\rho] &\equiv (\nu\widetilde{m}\widetilde{t}.(\overline{N'\rho}\langle M\rangle.Q'|Q_2))[\sigma\rho] \\ F &\equiv (\nu\widetilde{m}.(\{^{M}/_{x}\}|\nu\widetilde{t}.(Q'|Q_2)))[\rho\sigma]\end{aligned}$$

   where $\widetilde{m} \subseteq \mathrm{n}(M)$ and $\widetilde{t} \cap \mathrm{n}(M) = \emptyset$ and $\mathrm{n}(M) \cap \widetilde{s} = \emptyset$ and $M[\sigma\rho]$ is ground.

   Hence,

$$C_m\sigma \xrightarrow{\nu x.\overline{N'\sigma}\langle x\rangle} (\rho|\nu\widetilde{m}.(\{^{M}/_{x}\}|\nu\widetilde{t}.(Q'|Q_2)))\sigma \equiv \nu\widetilde{m}.(\rho \uplus \{^{M}/_{x}\}|\nu\widetilde{t}.(Q'|Q_2))\sigma.$$

   Therefore,

$$\begin{aligned}&\nu\widetilde{r}\widetilde{s}.(\nu\widetilde{x}.A_m\rho|\nu\widetilde{y}.C_m\sigma) \\ \rightarrow&\nu\widetilde{r}\widetilde{s}.((\sigma \uplus \rho)_{|dom(\sigma\uplus\rho)\setminus\widetilde{x}\widetilde{y}}|\nu x.((\nu\widetilde{l}.(P'|P_2))[\rho\sigma]|\nu\widetilde{m}.(\{^{M}/_{x}\}|\nu\widetilde{t}.(Q'|Q_2))[\rho\sigma])) \\ \equiv&\nu\widetilde{r}\widetilde{s}.(\nu\widetilde{x}.(\sigma|\nu\widetilde{l}.(P'|P_2))(\rho \uplus \{^{M}/_{x}\})|\nu\widetilde{y}x.(\rho \uplus \{^{M}/_{x}\}|\nu\widetilde{t}.(Q'|Q_2))\sigma)\end{aligned}$$

   This is a desired form.

4. $P[\rho\sigma] \xrightarrow{\nu x.\overline{N}\langle x\rangle} E$ and $Q[\sigma\rho] \xrightarrow{N(x)} F$ and $R \equiv \nu x.(E|F)$ for some E, F, x, and ground N.

   This case is similar to case 3.

Second, we consider when μ is a labeled action α. Note that $\widetilde{r}\widetilde{s} \cap \mathrm{n}(\alpha) = \emptyset$ and $\widetilde{x}\widetilde{y} \cap \mathrm{v}(\alpha) = \emptyset$.

By [1, Lemma B.19], $P[\rho\sigma]|Q[\sigma\rho] \xrightarrow{\alpha(\sigma\uplus\rho)} E$ and $D' \equiv (\sigma \uplus \rho)_{|dom(\sigma\uplus\rho)\setminus\widetilde{x}\widetilde{y}}|E$ for some E.

We consider two cases.

1. $P[\rho\sigma] \xrightarrow{\alpha(\sigma\uplus\rho)} F$ and $E \equiv F|Q[\sigma\rho]$ for some F.

   Then, $A_m\rho \xrightarrow{\alpha[\rho\sigma]} [\sigma\rho]|F$ because of $\sigma \uplus \rho = [\sigma\rho]|[\rho\sigma]$ and Lemma 5.

   In addition, we consider two cases.

   (a) $\alpha = N(M)$

   There exists $\widetilde{l}, P'$ and P_2 such that $P[\rho\sigma] \equiv (\nu\widetilde{l}.(N[\sigma\rho](x).P'|P_2))[\rho\sigma]$ and $F \equiv (\nu\widetilde{l}.(P'\{{}^{M[\sigma\rho]}/_{x}\}|P_2))[\rho\sigma]$.

   Hence, $A_m\rho \xrightarrow{N(M)[\rho\sigma]} (\sigma|\nu\widetilde{l}.(P'\{{}^{M[\sigma\rho]}/_{x}\}|P_2))\rho$.

   $\nu\widetilde{x}.A_m\rho \xrightarrow{N(M)[\rho\sigma]} \nu\widetilde{x}.(\sigma|\nu\widetilde{l}.(P'\{{}^{M[\sigma\rho]}/_{x}\}|P_2))\rho$.

   By Lemma 5, $\nu\widetilde{x}.A_m\rho|[\rho\sigma] \xrightarrow{N(M)} \nu\widetilde{x}.(\sigma|\nu\widetilde{l}.(P'\{{}^{M[\sigma\rho]}/_{x}\}|P_2))\rho|[\rho\sigma]$.

   $\nu\widetilde{x}.A_m\rho|[\rho\sigma]|Q[\sigma\rho] \xrightarrow{N(M)} \nu\widetilde{x}.(\sigma|\nu\widetilde{l}.(P'\{{}^{M[\sigma\rho]}/_{x}\}|P_2))\rho|[\rho\sigma]|Q[\sigma\rho]$.

   By Lemma 14,

   $\nu\widetilde{x}.A_m\rho|\nu\widetilde{y}.(\rho|Q)\sigma \xrightarrow{N(M)} \nu\widetilde{x}.(\sigma|\nu\widetilde{l}.(P'\{{}^{M[\sigma\rho]}/_{x}\}|P_2))\rho|\nu\widetilde{y}.(\rho|Q)\sigma$.

   Therefore,

$$
\begin{aligned}
&\nu\widetilde{r}\widetilde{s}.(\nu\widetilde{x}.A_m\rho|\nu\widetilde{y}.C_m\sigma)\\
&\qquad\qquad\xrightarrow{N(M)} \nu\widetilde{r}\widetilde{s}.(\nu\widetilde{x}.(\sigma|\nu\widetilde{l}.(P'\{{}^{M[\sigma\rho]}/_{x}\}|P_2))\rho|\nu\widetilde{y}.C_m\sigma)
\end{aligned}
$$

   This is the desired form.

   (b) $\alpha = \nu x.\overline{N}\langle x\rangle$

   There exists $\widetilde{m}, \widetilde{t}, M, P'$ and P_2 such that

$$
\begin{aligned}
P[\rho\sigma] &\equiv (\nu\widetilde{m}\widetilde{t}.(\overline{N[\sigma\rho]}\langle x\rangle).P'|P_2))[\rho\sigma]\\
F &\equiv (\nu\widetilde{m}.(\{M/x\}|\nu\widetilde{t}.(P'|P_2)))[\rho\sigma],
\end{aligned}
$$

   where $\widetilde{m} \subseteq \mathrm{n}(M)$ and $\widetilde{t} \cap \mathrm{n}(M) = \emptyset$ and $M[\rho\sigma]$ is ground.

   $[\sigma\rho]|F \equiv (\sigma|(\nu\widetilde{m}.(\{M/x\}|\nu\widetilde{t}.(P'|P_2)))\rho \equiv \nu\widetilde{m}.(\sigma \uplus \{M/x\}|\nu\widetilde{t}.(P'|P_2))\rho$.

   Hence, $A_m\rho \xrightarrow{(\nu x.\overline{N}\langle x\rangle)[\rho\sigma]} \nu\widetilde{m}.(\sigma \uplus \{M/x\}|\nu\widetilde{t}.(P'|P_2))\rho$.

   $\nu\widetilde{x}.A_m\rho \xrightarrow{(\nu x.\overline{N}\langle x\rangle)[\rho\sigma]} \nu\widetilde{x}.\nu\widetilde{m}.(\sigma \uplus \{M/x\}|\nu\widetilde{t}.(P'|P_2))\rho$.

By Lemma 5, $\nu\widetilde{x}.A_m\rho|[\rho\sigma] \overset{(\nu x.\overline{N}\langle x\rangle)}{\longrightarrow} \nu\widetilde{x}.\nu\widetilde{m}.(\sigma\uplus\{M/x\}|\nu\widetilde{t}.(P'|P_2))\rho|[\rho\sigma]$.
$\nu\widetilde{x}.A_m\rho|[\rho\sigma]|Q[\sigma\rho] \overset{(\nu x.\overline{N}\langle x\rangle)}{\longrightarrow} \nu\widetilde{x}.\nu\widetilde{m}.(\sigma \uplus \{M/x\}|\nu\widetilde{t}.(P'|P_2))\rho|[\rho\sigma]|Q[\sigma\rho]$.
By Lemma 14,
$\nu\widetilde{x}.A_m\rho|\nu\widetilde{y}.(\rho|Q)\sigma \overset{(\nu x.\overline{N}\langle x\rangle)}{\longrightarrow} \nu\widetilde{x}.\nu\widetilde{m}.(\sigma \uplus \{M/x\}|\nu\widetilde{t}.(P'|P_2))\rho|\nu\widetilde{y}.(\rho|Q)\sigma$.
Therefore,

$$\begin{aligned}&\nu\widetilde{r}\widetilde{s}.(\nu\widetilde{x}.A_m\rho|\nu\widetilde{y}.C_m\sigma)\\ &\qquad \overset{(\nu x.\overline{N}\langle x\rangle)}{\longrightarrow} \nu\widetilde{r}\widetilde{s}.(\nu\widetilde{x}.\nu\widetilde{m}.(\sigma \uplus \{M/x\}|\nu\widetilde{t}.(P'|P_2))\rho|\nu\widetilde{y}.C_m\sigma)\end{aligned}$$

This is the desired form.

2. $Q[\sigma\rho] \overset{\alpha(\sigma\uplus\rho)}{\longrightarrow} F$ and $E \equiv P[\sigma\rho]|F$ for some F.
   This case is similar to case 1.

□

A.4.3 Extracting a Trace of a Component Process from Concurrent Normal Form

Given a concurrent normal trace of $A|C$, we construct traces of A and C which are each process in them is of the form $\nu\widetilde{s_m}.A_m\rho_m$ or $\nu\widetilde{r_m}.C_m\sigma_m$. Assuming that we transformed the first m-th process.

We omit many subscripts and symmetric cases.

In case 3a in Definition 29, we get $\boxed{\nu\widetilde{s_m}.A_m\rho_m \longrightarrow \nu\widetilde{s_{m+1}}.A_{m+1}\rho_{m+1}}$.

In case 3c,

$$A_m\rho \overset{N'\rho(x)}{\longrightarrow} A_{m+1}\rho.$$
$$A_m\rho \overset{N'\rho(M\rho)}{\longrightarrow} A_{m+1}\{M/x\}\rho$$
$$A_m\rho \overset{N'\rho(M\rho)}{\longrightarrow} A_{m+1}(\rho \uplus \{M/x\})$$

We get $\boxed{\nu\widetilde{s_m}.A_m\rho_m \overset{N'\rho_m(M\rho_m)}{\longrightarrow} \nu\widetilde{s_{m+1}}.A_{m+1}\rho_{m+1}}$.
$C_m\sigma \overset{\nu x.\overline{N'\sigma}\langle x\rangle}{\longrightarrow} \nu\widetilde{m}.C_{m+1}\sigma$.
We get $\boxed{\nu\widetilde{r_m}.C_m\sigma_m \overset{\nu x.\overline{N'\sigma_m}\langle x\rangle}{\longrightarrow} \nu\widetilde{r_{m+1}}.C_{m+1}\sigma_{m+1}}$.
In case 4a, $A_m\rho \overset{N(M)[\rho\sigma]}{\longrightarrow} A_{m+1}\rho$.
$[\sigma\rho]|P[\rho\sigma] \overset{N(M)[\rho\sigma]}{\longrightarrow} A_{m+1}\rho$.
By Lemma 7, $[\sigma\rho]|P[\rho\sigma] \overset{N(M)\rho}{\longrightarrow} A_{m+1}\rho$.
Note that $(N(M)[\rho\sigma])[\sigma\rho] = (N(M)\rho)[\sigma\rho]$.
$A_m\rho \overset{N(M)\rho}{\longrightarrow} A_{m+1}\rho$.
We get $\boxed{\nu\widetilde{s_m}.A_m\rho_m \overset{N(M)\rho_m}{\longrightarrow} \nu\widetilde{s_{m+1}}.A_{m+1}\rho_{m+1}}$.

Case 5a is similar to case 4a. We get $\boxed{\nu\widetilde{s_m}.A_m\rho_m \overset{\nu x.\overline{N}\langle x\rangle\rho_m}{\longrightarrow} \nu\widetilde{s_{m+1}}.A_{m+1}\rho_{m+1}}$.

A.4.4 Composition of Traces

Given trace-equivalent processes A, B, an arbitrary process C, and a trace of $A|C$, we construct a statically equivalent trace of $B|C$. By subsection A.4.2, we can suppose that a given trace $\mathbf{tr}$ of $A|C$ is a concurrent normal form without loss of generality. We extract traces $\mathbf{tr}'$ and $\mathbf{tr}'''$ of A and C as mentioned in subsection A.4.3. There exists a safe trace $\mathbf{tr}''$ of B such that it satisfies the conditions below:

Each transition $\nu\widetilde{s_m}.A_m\rho_m \xrightarrow{\mu} \nu\widetilde{s_{m+1}}.A_{m+1}\rho_{m+1}$ in $\mathbf{tr}'$ corresponds to the transition $\nu\widetilde{s'_m}.B_m \overset{\mu}{\Longrightarrow} \nu\widetilde{s'_{m+1}}.B_{m+1}$ in $\mathbf{tr}''$. Moreover, B_m is of the form $\zeta_m|S_m$, it is normal and $\mathrm{dom}(\zeta_m) = \mathrm{dom}(\sigma_m)$. Especially, S_m is closed.

Each process in a constructed trace is of the form $\nu\widetilde{r_m}\widetilde{s'_m}.(\nu\widetilde{x_m}.B_m|\nu\widetilde{y_m}.C_m\zeta_m)$.

We again omit many subscripts. Note that an internal reduction and a receive action do not change a frame. In addition, $\widetilde{s'_m}$ contains no names in other processes because of bind-exclusiveness. Let $\widetilde{z} = \mathrm{dom}(\sigma)$.

1. Case 3a in Definition 29.

$$\nu\widetilde{r}\widetilde{s}.(\nu\widetilde{x}.A_m\rho|\nu\widetilde{y}.C_m\sigma) \longrightarrow \nu\widetilde{r}\widetilde{s}.(\nu\widetilde{x}.A_{m+1}\rho|\nu\widetilde{y}.C_m\sigma). \tag{1}$$

$$\nu\widetilde{s}.A_m\rho \longrightarrow \nu\widetilde{s}.A_{m+1}\rho. \tag{2}$$

$$\nu\widetilde{s'}.B_m \Longrightarrow \nu\widetilde{s'}.B_{m+1}. \tag{3}$$

(2) is in $\mathbf{tr}'$ and corrsponds to (1). (3) is in $\mathbf{tr}''$ and corresponds to (2).

By Lemma 6, we can suppose that $B_m \Longrightarrow B_{m+1}$. We get a desired transition

$$\boxed{\nu\widetilde{r}\widetilde{s'}.(\nu\widetilde{x}.B_m|\nu\widetilde{y}.C_m\zeta) \Longrightarrow \nu\widetilde{r}\widetilde{s'}.(\nu\widetilde{x}.B_{m+1}|\nu\widetilde{y}.C_m\zeta).}$$

2. Case 3b.

$$\nu\widetilde{r}\widetilde{s}.(\nu\widetilde{x}.A_m\rho|\nu\widetilde{y}.C_m\sigma) \longrightarrow \nu\widetilde{r}\widetilde{s}.(\nu\widetilde{x}.A_m\rho|\nu\widetilde{y}.C_{m+1}\sigma). \tag{4}$$

$$\nu\widetilde{r}.C_m\sigma \longrightarrow \nu\widetilde{r}.C_{m+1}\sigma \tag{5}$$

(5) is in $\mathbf{tr}'''$ and corresponds to (4).

By Lemma 6, we can suppose that $C_m\sigma \longrightarrow C_{m+1}\sigma$.

$C_m\sigma \equiv \nu\widetilde{z}.(C_m|\sigma) \equiv \nu\widetilde{z}.(\rho|Q|\sigma) \equiv \nu\widetilde{z}.(\rho \uplus \sigma|Q) \equiv \nu\widetilde{z}.(\rho \uplus \sigma|Q')$.

Q' is a process obtained from Q to substitute only for the part related to the transition.

$\rho|Q' \longrightarrow C_{m+1}$.

Similarly, $C_m\zeta \equiv \nu\widetilde{z}.(\rho \uplus \zeta|Q'')$.

Q'' is a process obtained from Q to substitute similarly using $\rho \uplus \zeta$.

$\nu\widetilde{s}.[\sigma\rho] \approx_s \nu\widetilde{s'}.\zeta$. Recall that $\nu\widetilde{s}.A_m\rho \approx_s \nu\widetilde{s'}.B_m$

$\nu\widetilde{s}.\rho \uplus \sigma \approx_s \nu\widetilde{s'}.\rho \uplus \zeta$. Note that $\mathrm{n}(\rho) \cap \widetilde{s} = \emptyset$.

Thus, two terms affected by $\rho \uplus \sigma$ at $C_m\sigma \longrightarrow C_{m+1}\sigma$ are also equal in Q''.

$\rho|Q'' \longrightarrow C_{m+1}$.

$C_m\zeta \longrightarrow C_{m+1}\zeta$.

We get a desired transition

$$\boxed{\nu\widetilde{r}\widetilde{s'}.(\nu\widetilde{x}.B_m|\nu\widetilde{y}.C_m\zeta) \longrightarrow \nu\widetilde{r}\widetilde{s'}.(\nu\widetilde{x}.B_m|\nu\widetilde{y}.C_{m+1}\zeta).}$$

3. Case 3c.

$$\nu\widetilde{r}\widetilde{s}.(\nu\widetilde{x}.A_m\rho|\nu\widetilde{y}.C_m\sigma) \longrightarrow \nu\widetilde{r}\widetilde{m}\widetilde{s}.(\nu\widetilde{x}.A_{m+1}(\rho \uplus \{{}^{M}/_{x}\})|\nu\widetilde{y}x.C_{m+1}\sigma). \quad (6)$$

$$\nu\widetilde{s}.A_m\rho \overset{N'\rho(M\rho)}{\longrightarrow} \nu\widetilde{s}.A_{m+1}(\rho \uplus \{{}^{M}/_{x}\}) \quad (7)$$

$$\nu\widetilde{r}.C_m\sigma \overset{\nu x.\overline{N'\sigma}\langle x\rangle}{\longrightarrow} \nu\widetilde{r}\widetilde{m}.C_{m+1}\sigma \quad (8)$$

$$\nu\widetilde{s'}.B_m \overset{N'\rho(M\rho)}{\Longrightarrow} \nu\widetilde{s'}.B_{m+1} \quad (9)$$

(7) is in **tr**$'$. (8) is in **tr**$'''$. Both correspond to (6). (9) is in **tr**$''$ and corresponds to (7).

By Lemma 6, we can suppose that $C_m\sigma \overset{\nu x.\overline{N'\sigma}\langle x\rangle}{\longrightarrow} \nu\widetilde{m}.C_{m+1}\sigma$.

$C_m\sigma \equiv \nu\widetilde{z}.(\rho \uplus \sigma|Q')$.

Q' is a process obtained from Q to substitute only for the part related to transition.

$\rho|Q' \overset{\nu x.\overline{N}\langle x\rangle}{\longrightarrow} \nu\widetilde{m}.C_{m+1}$, where $N = N'\rho[\sigma\rho] = N'\sigma[\rho\sigma]$.

Similarly, $C_m\zeta \equiv \nu\widetilde{z}.(\rho \uplus \zeta|Q'')$.

Q'' is a process obtained from Q to substitute similarly using $\rho \uplus \zeta$.

$\nu\widetilde{s}.[\sigma\rho] \approx_s \nu\widetilde{s'}.\zeta$.

$\nu\widetilde{s}.\rho \uplus \sigma \approx_s \nu\widetilde{s'}.\rho \uplus \zeta$.

Some channel in Q becomes equal to $N = N'\sigma[\rho\sigma] = N'(\rho \uplus \sigma)$ by $\rho \uplus \sigma$, so this channel becomes equal to $N'(\rho \uplus \zeta)$ by $\rho \uplus \zeta$. Note that $\widetilde{s} \cap \mathrm{n}(N') = \emptyset$.

Thus, $\rho|Q'' \overset{\nu x.\overline{N'(\rho\uplus\zeta)}\langle x\rangle}{\longrightarrow} \nu\widetilde{m}.C_{m+1}$.

$$\nu\widetilde{r}\widetilde{s'}.(\nu\widetilde{x}.B_m|\nu\widetilde{y}.C_m\zeta) \equiv \nu\widetilde{r}\widetilde{s'}.(\nu\widetilde{x}\widetilde{y}.(\zeta|\rho\zeta|S|Q\zeta))$$

$\nu\widetilde{s'}.B_m \Longrightarrow \nu\widetilde{s'}.B' \overset{N'\rho(M\rho)}{\longrightarrow} \nu\widetilde{s'}.B'' \Longrightarrow \nu\widetilde{s'}.B_{m+1}$ for some B' and B'' because $\nu\widetilde{s'}.B_m \overset{N'\rho(M\rho)}{\Longrightarrow} \nu\widetilde{s'}.B_{m+1}$.

We suppose that

- $B' \equiv \zeta|\hat{S}$,
- $\hat{S} \equiv \nu\widetilde{p}.(N'(\rho \uplus \zeta)(x).S'|S_2)$,
- $S \Longrightarrow \hat{S} \overset{N'(\rho\uplus\zeta)(M(\rho\uplus\zeta))}{\longrightarrow} D' \Longrightarrow D$,
- $B_{m+1} = \zeta|D$,
- $Q\zeta \overset{\nu x.\overline{N'(\rho\uplus\zeta)}\langle x\rangle}{\longrightarrow} E$, and
- $\nu\widetilde{m}.C_{m+1}\zeta \equiv \rho\zeta|E$.

Let $Q\zeta \equiv (\nu\widetilde{m}\widetilde{q}.(\overline{N'\rho}\langle M\rangle.Q'|Q_2))\zeta$.

This is possible because of $Q\zeta \overset{\nu x.\overline{N'(\rho\uplus\zeta)}\langle x\rangle}{\longrightarrow} E$.

By $\nu\widetilde{s'}.B_m \Longrightarrow \nu\widetilde{s'}.B'$ and $\hat{S} \overset{N'(\rho\uplus\zeta)(M(\rho\uplus\zeta))}{\Longrightarrow} D$,

$$\begin{aligned}
&\nu\widetilde{r}\widetilde{s'}.(\nu\widetilde{x}.B_m|\nu\widetilde{y}.C_m\zeta)\\
\Longrightarrow&\nu\widetilde{r}\widetilde{s'}.(\nu\widetilde{x}\widetilde{y}.(\zeta|\rho\zeta|\nu\widetilde{p}.(N'\rho(x).S'|S_2)|(\nu\widetilde{m}\widetilde{q}.(\overline{N'\rho}\langle M\rangle.Q'|Q_2))\zeta)\\
\Longrightarrow&\nu\widetilde{r}\widetilde{s'}.(\nu\widetilde{x}\widetilde{y}.(\zeta|\rho\zeta|\nu x.(D|(\nu\widetilde{m}\widetilde{q}.(Q'|\{M/x\}|Q_2))\zeta)))\\
\equiv&\nu\widetilde{r}\widetilde{m}\widetilde{s'}.(\nu\widetilde{x}.(\zeta|D)|\nu\widetilde{y}x.(\rho\uplus\{M/x\}|\nu\widetilde{q}.(Q'|Q_2)))\zeta).
\end{aligned}$$

We get the desired transition

$$\boxed{\nu\widetilde{r}\widetilde{s'}.(\nu\widetilde{x}.B_m|\nu\widetilde{y}.C_m\zeta) \Longrightarrow \nu\widetilde{r}\widetilde{m}\widetilde{s'}.(\nu\widetilde{x}.B_{m+1}|\nu\widetilde{y}x.C_{m+1}\zeta).}$$

4. Case 3d.

$$\nu\widetilde{r}\widetilde{s}.(\nu\widetilde{x}.A_m\rho|\nu\widetilde{y}.C_m\sigma) \longrightarrow \nu\widetilde{r}\widetilde{s}\widetilde{m}.(\nu\widetilde{x}x.A_{m+1}\rho|\nu\widetilde{y}.C_{m+1}(\sigma\uplus\{M/x\})) \tag{10}$$

$$\nu\widetilde{s}.A_m\rho \overset{\nu x.\overline{N'\rho}\langle x\rangle}{\longrightarrow} \nu\widetilde{s}\widetilde{m}.A_{m+1}\rho \tag{11}$$

$$\nu\widetilde{r}.C_m\sigma \overset{N'\sigma(M\sigma)}{\longrightarrow} \nu\widetilde{r}.C_{m+1}(\sigma\uplus\{M/x\}) \tag{12}$$

$$\nu\widetilde{s'}.B_m \overset{\nu x.\overline{N'\rho}\langle x\rangle}{\Longrightarrow} \nu\widetilde{s'}\widetilde{m'}.B_{m+1} \tag{13}$$

(11) is in **tr**$'$. (12) is in **tr**$'''$. Both correspond to (10). (13) is in **tr**$''$ and corresponds to (11). We suppose that $\zeta_{m+1} = \zeta_m \uplus \{M'/x\}$.

$C_m\sigma \overset{N'\sigma(x)}{\longrightarrow} C_{m+1}\sigma$.

$C_m\sigma \equiv \nu\widetilde{z}.(\rho\uplus\sigma|Q')$, where Q' is a process obtained from Q to substitute only for the input channel.

$\rho|Q' \overset{N(x)}{\longrightarrow} C_{m+1}$, where $N = N'\rho[\sigma\rho] = N'\sigma[\rho\sigma]$.

Similarly, $C_m\zeta \equiv \nu\widetilde{z}.(\rho\uplus\zeta|Q'')$.

Q'' is a process obtained from Q to substitute similarly using $\rho\uplus\zeta$.

$\nu\widetilde{s}.(\rho\uplus\sigma) \approx_s \nu\widetilde{s'}.(\rho\uplus\zeta)$.

Some channel in Q is equal to $N = N'\sigma[\rho\sigma] = N'(\rho\uplus\sigma)$ by $\rho\uplus\sigma$, so this channel becomes equal to $N'(\rho\uplus\zeta)$ by $\rho\uplus\zeta$. Note that $\widetilde{s}\cap \mathrm{n}(N') = \emptyset$.

Thus, $\rho|Q'' \overset{N'(\rho\uplus\zeta)(x)}{\longrightarrow} C_{m+1}$.

$\nu\widetilde{s'}.B_m \Longrightarrow \nu\widetilde{s'}.B' \overset{\nu x.\overline{N'\rho}\langle x\rangle}{\longrightarrow} \nu\widetilde{s''}.B'' \Longrightarrow \nu\widetilde{s''}.B_{m+1}$ for some B' and B''.

We suppose that

- $B' \equiv \zeta|\hat{S}$,
- $\hat{S} \equiv \nu\widetilde{m'}\widetilde{p}.(\overline{N'(\rho\uplus\zeta)}\langle M'\rangle.S'|S_2)$,

- $S \Longrightarrow \hat{S} \overset{\nu x.\overline{N'(\rho \uplus \zeta)}\langle x \rangle}{\longrightarrow} D' \Longrightarrow D$,
- $B_{m+1} = \zeta | D$,
- $Q\zeta \overset{N'(\rho \uplus \zeta)(x)}{\longrightarrow} E$, and
- $C_{m+1}\zeta \equiv \rho\zeta | E$.

Let $Q\zeta \equiv (\nu\widetilde{q}.(N'\rho(x).Q'|Q_2))\zeta$.

By $S \Longrightarrow \hat{S}$,

$$\begin{aligned}
&\nu\widetilde{r}\widetilde{s'}.(\nu\widetilde{x}.B_m|\nu\widetilde{y}.C_m\zeta)\\
\equiv&\nu\widetilde{r}\widetilde{s'}.(\nu\widetilde{x}\widetilde{y}.(\zeta|\rho\zeta|S|Q\zeta))\\
\Longrightarrow&\nu\widetilde{r}\widetilde{s'}.(\nu\widetilde{x}\widetilde{y}.(\zeta|\rho\zeta|\nu\widetilde{m'}\widetilde{p}.(\overline{N'\rho}\langle M'\rangle.S'|S_2)|\nu\widetilde{q}.(N'\rho(x).Q'|Q_2)\zeta))\\
\longrightarrow&\nu\widetilde{r}\widetilde{s'}.(\nu\widetilde{x}\widetilde{y}.(\zeta|\rho\zeta|\nu x.\nu\widetilde{m'}.(\{M'/x\}|\nu\widetilde{p}.(S'|S_2)|(\nu\widetilde{q}.Q'|Q_2))\zeta))\\
\equiv&\nu\widetilde{r}\widetilde{s'}\widetilde{m'}.(\nu\widetilde{x}x.(\zeta \uplus \{M'/x\}|\nu\widetilde{p}.(S'|S_2))|\nu\widetilde{y}.(\rho|\nu\widetilde{q}.(Q'|Q_2))(\zeta \uplus \{M'/x\}))\\
\Longrightarrow&\nu\widetilde{r}\widetilde{s'}\widetilde{m'}.(\nu\widetilde{x}x.B_{m+1}|\nu\widetilde{y}.C_{m+1}(\zeta \uplus \{M'/x\}))
\end{aligned}$$

We get a desired transition

$$\boxed{\nu\widetilde{r}\widetilde{s'}.(\nu\widetilde{x}.B_m|\nu\widetilde{y}.C_m\zeta) \Longrightarrow \nu\widetilde{r}\widetilde{s'}\widetilde{m'}.(\nu\widetilde{x}x.B_{m+1}|\nu\widetilde{y}.C_{m+1}(\zeta \uplus \{M'/x\})).}$$

5. Case 4a.

$$\nu\widetilde{r}\widetilde{s}.(\nu\widetilde{x}.A_m\rho|\nu\widetilde{y}.C_m\sigma) \overset{N(M)}{\longrightarrow} \nu\widetilde{r}\widetilde{s}.(\nu\widetilde{x}.A_{m+1}\rho|\nu\widetilde{y}.C_m\sigma) \tag{14}$$

$$\nu\widetilde{s}.A_m\rho \overset{N(M)\rho}{\longrightarrow} \nu\widetilde{s}.A_{m+1}\rho \tag{15}$$

$$\nu\widetilde{s'}.B_m \overset{N(M)\rho}{\Longrightarrow} \nu\widetilde{s'}.B_{m+1} \tag{16}$$

(15) is in **tr′** and corresponds to (14). Recall subsection A.4.3. (16) is in **tr′** and corresponds to (15).

By Lemma 6, we can suppose that $B_m \overset{N(M)\rho}{\Longrightarrow} B_{m+1}$.

By Lemma 5, $\rho|B_m \overset{N(M)}{\Longrightarrow} \rho|B_{m+1}$.

We get a desired transition

$$\boxed{\nu\widetilde{r}\widetilde{s'}.(\nu\widetilde{x}.B_m|\nu\widetilde{y}.C_m\zeta) \overset{N(M)}{\Longrightarrow} \nu\widetilde{r}\widetilde{s'}.(\nu\widetilde{x}.B_{m+1}|\nu\widetilde{y}.C_m\zeta).}$$

6. Case 4b.

$$\nu\widetilde{r}\widetilde{s}.(\nu\widetilde{x}.A_m\rho|\nu\widetilde{y}.C_m\sigma) \overset{N(M)}{\longrightarrow} \nu\widetilde{r}\widetilde{s}.(\nu\widetilde{x}.A_m\rho|\nu\widetilde{y}.C_{m+1}\sigma) \tag{17}$$

$$\nu\widetilde{r}.C_m\sigma \overset{N(M)\sigma}{\longrightarrow} \nu\widetilde{r}.C_{m+1}\sigma \tag{18}$$

(18) is in **tr‴** and corresponds to (17).

$C_m\sigma \xrightarrow{N(M)\sigma} C_{m+1}\sigma$.

$C_m\sigma \equiv \nu\widetilde{z}.(\rho \uplus \sigma|Q')$ where Q' is a process obtained from Q to substitute only for the input channel.

$\rho|Q' \xrightarrow{N(\rho\uplus\sigma)(M)} C_{m+1}$.

Similarly, $C_m\zeta \equiv \nu\widetilde{z}.(\rho \uplus \zeta|Q'')$.

Q'' is a process obtained from Q to substitute similarly using $\rho \uplus \zeta$.

Some channel in Q is equal to $N(\rho\uplus\zeta)$ by $\rho\uplus\zeta$ because $\nu\widetilde{s}.(\rho\uplus\sigma) \approx_s \nu\widetilde{s'}.(\rho\uplus\zeta)$. Note that $\widetilde{s} \cap \mathrm{n}(N) = \emptyset$.

Thus, $\rho|Q'' \xrightarrow{N(\rho\uplus\zeta)(M)} C_{m+1}$.

$\rho \uplus \zeta|Q'' \xrightarrow{N(\rho\uplus\zeta)(M)} \zeta|C_{m+1}$.

By Lemma 7, $\rho \uplus \zeta|Q'' \xrightarrow{N(M)\zeta} \zeta|C_{m+1}$.

$C_m\zeta \xrightarrow{N(M)\zeta} C_{m+1}\zeta$.

By Lemma 5, $\zeta|C_m\zeta \xrightarrow{N(M)} \zeta|C_{m+1}\zeta$. We get a desired transition

$$\boxed{\nu\widetilde{r}\widetilde{s'}.(\nu\widetilde{x}.B_m|\nu\widetilde{y}.C_m\zeta) \xrightarrow{N(M)} \nu\widetilde{r}\widetilde{s'}.(\nu\widetilde{x}.B_m|\nu\widetilde{y}.C_{m+1}\zeta).}$$

7. Case 5a.

$$\nu\widetilde{r}\widetilde{s}.(\nu\widetilde{x}.A_m\rho|\nu\widetilde{y}.C_m\sigma) \xrightarrow{\nu x.\overline{N}\langle x\rangle} \nu\widetilde{r}\widetilde{s}\widetilde{m}.(\nu\widetilde{x}.A_{m+1}\rho|\nu\widetilde{y}.C_m(\sigma \uplus \{M/x\})) \quad (19)$$

$$\nu\widetilde{s}.A_m\rho \xrightarrow{\nu x.\overline{N\rho}\langle x\rangle} \nu\widetilde{s}\widetilde{m}.A_{m+1}\rho \quad (20)$$

$$\nu\widetilde{s'}.B_m \overset{\nu x.\overline{N\rho}\langle x\rangle}{\Longrightarrow} \nu\widetilde{s'}\widetilde{m'}.B_{m+1} \quad (21)$$

$$\nu\widetilde{r}\widetilde{s'}.(\nu\widetilde{x}.B_m|\nu\widetilde{y}.C_m\zeta) \overset{\nu x.\overline{N\rho}\langle x\rangle}{\Longrightarrow} \nu\widetilde{r}\widetilde{s'}\widetilde{m'}.(\nu\widetilde{x}.B_{m+1}|\nu\widetilde{y}.C_m(\zeta \uplus \{M'/x\})) \quad (22)$$

(20) is in $\mathbf{tr}'$ and corresponds to (19). Recall subsection A.4.3. (21) is in $\mathbf{tr}''$ and corresponds to (20). (22) is deriverd from (21). Note that x does not appear in C_m.

(22) contains an output, so it changes a frame. We have to check that static equivalence is preserved.

$\nu\widetilde{s}.[\sigma\rho] \approx_s \nu\widetilde{s'}.\zeta$ because $\nu\widetilde{s}.A_m\rho \approx_s \nu\widetilde{s'}.B_m$.

$\nu\widetilde{s}\widetilde{m}.[(\sigma\uplus\{M/x\})\rho] \approx_s \nu\widetilde{s'}\widetilde{m'}.(\zeta\uplus\{M'/x\})$ since $\nu\widetilde{s}\widetilde{m}.A_{m+1}\rho \approx_s \nu\widetilde{s'}\widetilde{m'}.B_{m+1}$.

In addition, $\nu\widetilde{r}\widetilde{s}.(\nu\widetilde{x}.[\sigma\rho]|\nu\widetilde{y}.[\rho\sigma]) \approx_s \nu\widetilde{r}\widetilde{s'}.(\nu\widetilde{x}.\zeta|\nu\widetilde{y}.\rho\zeta)$.

This is because $\nu\widetilde{r}\widetilde{s}.(\nu\widetilde{x}.A_m\rho|\nu\widetilde{y}.C_m\sigma) \approx_s \nu\widetilde{r}\widetilde{s'}.(\nu\widetilde{x}.B_m|\nu\widetilde{y}.C_m\zeta)$.

$P[\rho\sigma] \xrightarrow{\nu x.\overline{(N\rho)[\sigma\rho]}\langle x\rangle} D$ and $\nu\widetilde{m}.A_{m+1}\rho \equiv [\sigma\rho]|D$ for some D by Lemma 9. Recall Definition 29.

Let

$$P[\rho\sigma] \equiv (\nu\widetilde{m}\widetilde{l}.(\overline{N}\langle M\rangle.P'|P_2))[\rho\sigma]$$
$$D \equiv (\nu\widetilde{m}.(\{M/x\}|\nu\widetilde{l}.(P'|P_2)))[\rho\sigma]$$

Then, $A_{m+1}\rho \equiv (\sigma \uplus \{M/x\}|\nu\widetilde{l}.(P'|P_2))\rho$.

$$\begin{aligned}&\nu\widetilde{r}\widetilde{s}\widetilde{m}.(\nu\widetilde{x}.[(\sigma \uplus \{M/x\})\rho]|\nu\widetilde{y}.[\rho(\sigma \uplus \{M/x\})])\\ \equiv&\nu\widetilde{r}.(\nu\widetilde{x}.\nu\widetilde{s}\widetilde{m}.[(\sigma \uplus \{M/x\})\rho]|\nu\widetilde{y}.\rho)\\ \approx_s&\nu\widetilde{r}.(\nu\widetilde{x}.\nu\widetilde{s'}\widetilde{m'}.(\zeta \uplus \{M'/x\})|\nu\widetilde{y}.\rho)\\ \equiv&\nu\widetilde{r}\widetilde{s'}\widetilde{m'}.(\nu\widetilde{x}.(\zeta \uplus \{M'/x\})|\nu\widetilde{y}.\rho(\zeta \uplus \{M'/x\}))\end{aligned}$$

Note that $\nu\widetilde{s}\widetilde{m}.[(\sigma \uplus \{M/x\})\rho] \approx_s \nu\widetilde{s'}\widetilde{m'}.(\zeta \uplus \{M'/x\})$.
Therefore,

$$\begin{aligned}&\nu\widetilde{r}\widetilde{s}\widetilde{m}.(\nu\widetilde{x}.A_{m+1}\rho|\nu\widetilde{y}.C_m(\sigma \uplus \{M/x\}))\\ \approx_s\ &\nu\widetilde{r}\widetilde{s'}\widetilde{m'}.(\nu\widetilde{x}.B_{m+1}\rho|\nu\widetilde{y}.C_m(\zeta \uplus \{M'/x\}))\end{aligned}$$

Hence, the transition below is a desired one.

$$\boxed{\nu\widetilde{r}\widetilde{s'}.(\nu\widetilde{x}.B_m|\nu\widetilde{y}.C_m\zeta) \overset{\nu x.\overline{N}\langle x\rangle}{\Longrightarrow} \nu\widetilde{r}\widetilde{s'}\widetilde{m'}.(\nu\widetilde{x}.B_{m+1}\rho|\nu\widetilde{y}.C_m(\zeta \uplus \{M/x\})).}$$

8. Case 5b.

$$\nu\widetilde{r}\widetilde{s}.(\nu\widetilde{x}.A_m\rho|\nu\widetilde{y}.C_m\sigma) \xrightarrow{\nu x.\overline{N}\langle x\rangle} \nu\widetilde{r}\widetilde{m}\widetilde{s}.(\nu\widetilde{x}.A_m(\rho \uplus \{M/x\})|\nu\widetilde{y}.C_{m+1}\sigma) \quad (23)$$
$$\nu\widetilde{r}.C_m\sigma \xrightarrow{\nu x.\overline{N\sigma}\langle x\rangle} \nu\widetilde{r}\widetilde{m}.C_{m+1}\sigma \quad (24)$$

(24) is in $\mathbf{tr}'''$ and corresponds to (23).

In the same way with Case 4b, $C_m\zeta \xrightarrow{\nu x.\overline{N\zeta}\langle x\rangle} \nu\widetilde{m}.C_{m+1}\zeta$.

$\nu\widetilde{r}\widetilde{s'}.(\nu\widetilde{x}.B_m|\nu\widetilde{y}.C_m\zeta) \xrightarrow{\nu x.\overline{N}\langle x\rangle} \nu\widetilde{r}\widetilde{m}\widetilde{s'}.(\nu\widetilde{x}.B_m|\nu\widetilde{y}.C_{m+1}\zeta)$.

We have to check that static equivalence is preserved.

Now, $\nu\widetilde{r}\widetilde{s}.(\nu\widetilde{x}.[\sigma\rho]|\nu\widetilde{y}.[\rho\sigma]) \approx_s \nu\widetilde{r}\widetilde{s'}.(\nu\widetilde{x}.\zeta|\nu\widetilde{y}.\rho\zeta)$.

This is because $\nu\widetilde{r}\widetilde{s}.(\nu\widetilde{x}.A_m\rho|\nu\widetilde{y}.C_m\sigma) \approx_s \nu\widetilde{r}\widetilde{s'}.(\nu\widetilde{x}.B_m|\nu\widetilde{y}.C_m\zeta)$.

$\nu\widetilde{s}.[\sigma\rho] \approx_s \nu\widetilde{s'}.\zeta$ because $\nu\widetilde{s}.A_m\rho \approx_s \nu\widetilde{s'}.B_m$.

$\nu\widetilde{s}.[\sigma\rho]|\nu\widetilde{y}.(\rho \uplus \{M/x\}) \approx_s \nu\widetilde{s'}.\zeta|\nu\widetilde{y}.(\rho \uplus \{M/x\})$.

Thus, $\nu\widetilde{r}\widetilde{m}\widetilde{s}.(\nu\widetilde{x}.[\sigma\rho]|\nu\widetilde{y}.(\rho \uplus \{M/x\})) \approx_s \nu\widetilde{r}\widetilde{m}\widetilde{s'}.(\nu\widetilde{x}.\zeta|\nu\widetilde{y}.(\sigma \uplus \{M/x\}))$.

Hence, the transition below is a desired one.

$$\boxed{\nu\widetilde{r}\widetilde{s'}.(\nu\widetilde{x}.B_m|\nu\widetilde{y}.C_m\zeta) \xrightarrow{\nu x.\overline{N}\langle x\rangle} \nu\widetilde{r}\widetilde{m}\widetilde{s'}.(\nu\widetilde{x}.B_m|\nu\widetilde{y}.C_{m+1}\zeta).}$$

Thus we have finished the proof of Proposition 3 and Theorem 1.

Appendix B

Proofs for Chapter 4

B.1 Properties of Security Notions

Lemma 1. *If* $\mathrm{sv}(A) = \mathrm{sv}(B)$*, then*

$$[\forall \rho; A\rho \approx_s B\rho] \Leftrightarrow A \equiv_s B$$

Proof. $\Rightarrow$) We prove $A \equiv_s B$ by induction on the syntax of static formulas.
We arbitrarily take an assignment ρ . We suppose $A, \rho, A\rho, 0 \models \delta$.

1. $\top$.

   Trivially, $B, \rho, B\rho, 0 \models \top$.

2. $M_1 = M_2$.

   By assumption, $(M_1\rho\rho'_{\restriction \mathrm{fv}(M_1)\setminus \mathrm{dom}(A)} = M_2\rho\rho'_{\restriction \mathrm{fv}(M_2)\setminus \mathrm{dom}(A)})\mathrm{fr}(A\rho)$ for any ρ'.

   By $A\rho \approx_s B\rho$, $(M_1\rho\rho'_{\restriction \mathrm{fv}(M_1)\setminus \mathrm{dom}(B)} = M_2\rho\rho'_{\restriction \mathrm{fv}(M_2)\setminus \mathrm{dom}(B)})\mathrm{fr}(B\rho)$.

   This means that $B, \rho, B\rho, 0 \models M_1 = M_2$.

3. $M \in \mathrm{dom}$.

   M must be a variable x.

   By definition, $x \in \mathrm{dom}(A\rho)$.

   By $A\rho \approx_s B\rho$, $x \in \mathrm{dom}(B\rho)$.

   This means that $B, \rho, B\rho, 0 \models x \in \mathrm{dom}$.

4. $\delta_1 \vee \delta_2$

   By assumption, $A, \rho, A\rho, 0 \models \delta_1 \vee \delta_2$.

   By definition, $A, \rho, A\rho, 0 \models \delta_1$ or $A, \rho, A\rho, 0 \models \delta_2$.

   By induction hypothesis, $B, \rho, B\rho, 0 \models \delta_1$ or $B, \rho, B\rho, 0 \models \delta_2$.

   This means that $B, \rho, B\rho, 0 \models \delta_1 \vee \delta_2$.

5. $\neg\delta$.

   By assumption, $A, \rho, A\rho, 0 \not\models \delta$.

   By induction hypothesis, $B, \rho, B\rho, 0 \not\models \delta$.

   This means that $B, \rho, B\rho, 0 \models \neg\delta$.

The converses of these cases are similar. Thus, $A \sqsubseteq_s B$.

$\Leftarrow$) We arbitrarily take an assignment ρ .
We arbitrarily take $x \in \mathrm{dom}(A\rho)$. Then, $A, \rho, A\rho, 0 \models x \in \mathrm{dom}$.
By assumption, $B, \rho, B\rho, 0 \models x \in \mathrm{dom}$. That is, $x \in \mathrm{dom}(B\rho)$.
Thus, $\mathrm{dom}(A\rho) \subseteq \mathrm{dom}(B\rho)$. The converse is similar.
We suppose that $(M = N)\mathrm{fr}(A\rho)$. This means that $A, \rho, A\rho, 0 \models M = N$. By assumption, $B, \rho, B\rho, 0 \models M = N$. In other words, $(M = N)\mathrm{fr}(B\rho)$. Similarly, $(M = N)\mathrm{fr}(B\rho) \Rightarrow (M = N)\mathrm{fr}(A\rho)$. Therefore, $A\rho \approx_s B\rho$. □

Theorem 2. *If* $\mathrm{sv}(A) = \mathrm{sv}(B)$*, then*
1. $A \approx_t B \Rightarrow A \sqsubseteq_L B$*; 2.* $A \sqsubseteq_L B \Rightarrow A \subseteq_t B$

Proof. 1) We prove

$$\forall\rho\forall\mathbf{tr} \in \mathrm{tr}(A\rho)\forall\mathbf{tr}' \in \mathrm{tr}(B\rho); \mathbf{tr} \sim_t \mathbf{tr}' \Rightarrow \forall i\forall\varphi; [A, \rho, \mathbf{tr}, i \models \varphi \Leftrightarrow B, \rho, \mathbf{tr}', i \models \varphi]. \tag{1}$$

We take an assignment ρ arbitrarily. We also arbitrarily take traces $\mathbf{tr}$ and $\mathbf{tr}'$ of $A\rho$ and $B\rho$ respectively such as $\mathbf{tr} \sim_t \mathbf{tr}'$. We arbitrarily take i.

1. δ.

   We suppose that $A, \rho, \mathbf{tr}, i \models \delta$.

   This means that $\mathbf{tr}[i], 0, \mathbf{tr}[i], 0 \models \delta$.

   By $\mathbf{tr} \sim_t \mathbf{tr}'$, $\mathbf{tr}[i] \approx_s \mathbf{tr}'[i]$. By Lemma 1, $\mathbf{tr}[i] \equiv_s \mathbf{tr}'[i]$.

   Thus, $\mathbf{tr}'[i], 0, \mathbf{tr}'[i], 0 \models \delta$. This implies that $B, \rho, \mathbf{tr}', i \models \delta$

   The converse is similar.

2. $\varphi_1 \vee \varphi_2$.

   By induction hypothesis,

   $A, \rho, \mathbf{tr}, i \models \varphi_1 \Leftrightarrow B, \rho, \mathbf{tr}', i \models \varphi_1$, and $A, \rho, \mathbf{tr}, i \models \varphi_2 \Leftrightarrow B, \rho, \mathbf{tr}', i \models \varphi_2$

   Thus, $A, \rho, \mathbf{tr}, i \models \varphi_1 \vee \varphi_2 \Leftrightarrow B, \rho, \mathbf{tr}', i \models \varphi_1 \vee \varphi_2$.

3. $\neg\varphi$.

   By induction hypothesis, $A, \rho, \mathbf{tr}, i \not\models \varphi \Leftrightarrow B, \rho, \mathbf{tr}', i \not\models \varphi$.

   Thus, $A, \rho, \mathbf{tr}, i \models \neg\varphi \Leftrightarrow B, \rho, \mathbf{tr}', i \models \neg\varphi$.

4. $\langle \mu \rangle_{-} \varphi$.

   We suppose that $A, \rho, \mathbf{tr}, i \models \langle \mu \rangle_{-} \varphi$.

   By definition, $\mathbf{tr}[i-1] \overset{\mu}{\Longrightarrow} \mathbf{tr}[i]$ in $\mathbf{tr}$ and $A, \rho, \mathbf{tr}, i-1 \models \varphi$.

   $\mathbf{tr}'[i-1] \overset{\mu}{\Longrightarrow} \mathbf{tr}'[i]$ in $\mathbf{tr}'$ because $\mathbf{tr} \sim_t \mathbf{tr}'$.

   By induction hypothesis, $B, \rho, \mathbf{tr}', i-1 \models \varphi$.

   Thus, $B, \rho, \mathbf{tr}', i \models \langle \mu \rangle_{-} \varphi$.

   The converse is similar.

5. $F\varphi$.

   We suppose that $A, \rho, \mathbf{tr}, i \models F\varphi$.

   By definition, $\exists j \geq i$ s.t. $A, \rho, \mathbf{tr}, j \models \varphi$.

   By induction hypothesis, $B, \rho, \mathbf{tr}', j \models \varphi$.

   Thus, $B, \rho, \mathbf{tr}', i \models F\varphi$.

   The converse is similar.

6. $K\varphi$.

   We suppose that $A, \rho, \mathbf{tr}, i \models K\varphi$.

   By definition, $\forall \rho' \; \forall \mathbf{tr}'' \in \mathrm{tr}(A\rho'); \mathbf{tr}[0,i] \sim_t \mathbf{tr}''[0,i] \Rightarrow A, \rho', \mathbf{tr}'', i \models \varphi$.

   We arbitrarily take ρ' and $\mathbf{tr}''' \in \mathrm{tr}(B\rho')$ such as $\mathbf{tr}'[0,i] \sim_t \mathbf{tr}'''[0,i]$

   By assumption, there exists a trace $\mathbf{tr}''$ of $A\rho'$ such as $\mathbf{tr}''' \sim_t \mathbf{tr}''$.

   Now, $\mathbf{tr}[0,i] \sim_t \mathbf{tr}'[0,i] \sim_t \mathbf{tr}'''[0,i] \sim_t \mathbf{tr}''[0,i]$, so $A, \rho', \mathbf{tr}'', i \models \varphi$.

   By induction hypothesis, $B, \rho', \mathbf{tr}''', i \models \varphi$.

   By arbitrariness of ρ' and $\mathbf{tr}'''$, it follows that $B, \rho, \mathbf{tr}', i \models K\varphi$.

   The converse is similar.

We complete proving (1). It immediately follows that $A \approx_t B \Rightarrow A \sqsubseteq_L B$.

2) We arbitrarily take an assignment ρ. We also arbitrarily take a trace $\mathbf{tr}$ of $A\rho$.

By assumption, $\exists \mathbf{tr}' \in \mathrm{tr}(B\rho)$s.t.$\forall i \forall \varphi; [A, \rho, \mathbf{tr}, i \models \varphi \Leftrightarrow B, \rho, \mathbf{tr}', i \models \varphi]$.

We prove $\mathbf{tr} \sim_t \mathbf{tr}'$.

Let $\mathbf{tr} = A_0 := A\rho \overset{\mu_1}{\Longrightarrow} A_1 \overset{\mu_2}{\Longrightarrow}, ..., \overset{\mu_n}{\Longrightarrow} A_n$. It holds that

$$A, \rho, \mathbf{tr}, 0 \models F\langle \mu_n \rangle_{-} ... \langle \mu_1 \rangle_{-} \top,$$

so it also holds that

$$B, \rho, \mathbf{tr}', 0 \models F\langle \mu_n \rangle_{-} ... \langle \mu_1 \rangle_{-} \top.$$

Moreover,

$$\forall \mu; A, \rho, \mathbf{tr}, 0 \not\models F\langle \mu \rangle_{-} \langle \mu_n \rangle_{-} ... \langle \mu_1 \rangle_{-} \top$$
$$\forall \mu; A, \rho, \mathbf{tr}, 0 \not\models F\langle \mu_n \rangle_{-} ... \langle \mu_1 \rangle_{-} \langle \mu \rangle_{-} \top$$

Thus,

$$\forall \mu; B, \rho, \mathbf{tr'}, 0 \not\models F\langle \mu \rangle_{-} \langle \mu_n \rangle_{-} \ldots \langle \mu_1 \rangle_{-} \top$$
$$\forall \mu; B, \rho, \mathbf{tr'}, 0 \not\models F\langle \mu_n \rangle_{-} \ldots \langle \mu_1 \rangle_{-} \langle \mu \rangle_{-} \top$$

Therefore, $\mathbf{tr'}$ is of the form $B_0 \coloneqq B\rho \overset{\mu_1}{\Longrightarrow} B_1 \overset{\mu_2}{\Longrightarrow}, \ldots, \overset{\mu_n}{\Longrightarrow} B_n$.

We arbitrarily take i. We suppose that $(M = N)\mathrm{fr}(A_i)$.

Now, $A, \rho, \mathbf{tr}, i \models M = N$. Hence, $B, \rho, \mathbf{tr'}, i \models M = N$.

That is, $(M = N)\mathrm{fr}(B_i)$. Thus $(M = N)\mathrm{fr}(A_i) \Rightarrow (M = N)\mathrm{fr}(B_i)$ and the converse similarly holds.

Similarly, $x \in \mathrm{dom}(A_i) \Leftrightarrow x \in \mathrm{dom}(B_i)$.

It follows that $A_i \approx_s B_i$. Therefore, $\mathbf{tr} \sim_t \mathbf{tr'}$.

By arbitrariness of tr, it immediately follows that $A \sqsubseteq_L B \Rightarrow A \subseteq_t B$. □

Proposition 7.
$\forall \widetilde{M}\ \forall i\ \forall \mathbf{tr} \in \mathrm{tr}(A(M_1, \ldots, M_p))$
$\exists \widetilde{N}\ \exists \mathbf{tr'} \in \mathrm{tr}(A(M_i, N_2, \ldots, N_{i-1}, M_1, N_{i+1}, \ldots, N_p))$ s.t. $\mathbf{tr} \sim_t \mathbf{tr'}$
$\Leftrightarrow$ *(x_1, δ_k) is role interchangeable with respect to $\{\delta_j\}$ in A for all $\{\delta_j\}$ and k.*

Proof. $\Rightarrow$) We arbitrarily take $\widetilde{M}$ and i.

Let $\rho = [x_1 \mapsto M_1, \ldots, x_p \mapsto M_p]$.

We arbitrarily take $\mathbf{tr} \in \mathrm{tr}(A\rho)$ and t.

We suppose that $A, \rho, \mathbf{tr}, t \models \delta_k(x_1, \widetilde{y_k}) \wedge \delta_j(x_i, \widetilde{y_j})$.

By assumption, $\exists \widetilde{N} \exists \mathbf{tr'} \in \mathrm{tr}(A(M_i, N_2, \ldots, N_{i-1}, M_1, N_{i+1}, \ldots, N_p))$s.t. $\mathbf{tr} \sim_t \mathbf{tr'}$.

Let

$$\rho' = [x_1 \mapsto M_i, x_2 \mapsto N_2, \ldots, x_{i-1} \mapsto N_{i-1}, x_i \mapsto M_1, x_{i+1} \mapsto M_{i+1}, \ldots, x_P \mapsto N_p].$$

By Lemma 1, $\mathbf{tr}[t] \equiv_s \mathbf{tr'}[t]$ and $\mathbf{tr'}[t], 0, \mathbf{tr'}[t], 0 \models \delta_k(\rho(x_1), \widetilde{y_k}) \wedge \delta_j(\rho(x_i), \widetilde{y_j})$.

Thus, $A, \rho', \mathbf{tr'}, t \models \delta_k(x_i, \widetilde{y_k}) \wedge \delta_j(x_1, \widetilde{y_j})$.

Hence, $A, \rho, \mathbf{tr}, t \models P(\delta_k(x_i, \widetilde{y_k}) \wedge \delta_j(x_1, \widetilde{y_j}))$.

That is, $A \models G(\delta_k(x_1, \widetilde{y_k}) \rightarrow \bigwedge_{i \in I} \bigwedge_{j \in J}(\delta_j(x_i, \widetilde{y_j}) \rightarrow P(\delta_k(x_i, \widetilde{y_k}) \wedge \delta_j(x_1, \widetilde{y_j}))))$.

$\Leftarrow$) We arbitrarily take $\widetilde{M}$ and i.

Let $\rho = [x_1 \mapsto M_1, \ldots, x_p \mapsto M_p]$.

We arbitrarily take $\mathbf{tr} \in \mathrm{tr}(A\rho)$.

Let $\delta_1(z) : z = M_1$ and $\delta_i(z) : z = M_i$.

By assumption, (x_1, δ_1) is role interchangeable with respect to $\{\delta_1, \delta_i\}$.

Hence, $A \models G(\delta_1(x_1) \rightarrow (\delta_i(x_i) \rightarrow P(\delta_1(x_i) \wedge \delta_i(x_1))))$.

Therefore, $A, \rho, \mathbf{tr}, |\mathbf{tr}| \models \delta_1(x_1) \rightarrow (\delta_i(x_i) \rightarrow P(\delta_1(x_i) \wedge \delta_i(x_1)))$.

Because $A, \rho, \mathbf{tr}, |\mathbf{tr}| \models \delta_1(x_1) \wedge \delta_i(x_i)$, it holds that

$$A, \rho, \mathbf{tr}, |\mathbf{tr}| \models P(\delta_1(x_i) \wedge \delta_i(x_1)).$$

That is, $\exists \rho' \exists \mathbf{tr'} \in \mathrm{tr}(A\rho')$ s.t. $\mathbf{tr} \sim_t \mathbf{tr'}$ and $A, \rho', \mathbf{tr'}, |\mathbf{tr'}| \models \delta_1(x_i) \wedge \delta_i(x_1)$.

This means that $\rho'(x_1) = M_i$ and $\rho'(x_i) = M_1$.

Let $N_j = \rho'(x_j)(j \neq 1, i)$.

Then, $\exists \mathbf{tr'} \in \mathrm{tr}(A(M_i, N_2, \ldots, N_{i-1}, M_1, N_{i+1}, \ldots, N_p))$s.t. $\mathbf{tr} \sim_t \mathbf{tr'}$ holds. □

Proposition 8. $A(x_1, x_2) \approx_t A(x_2, x_1)$
$\Leftrightarrow$ (x_1, δ_k) *is role interchangeable with respect to* $\{\delta_j\}_{j \in J}$ *in* A *for all* $\{\delta_j\}_{j \in J}$ *and* k.

Proof. We only have to prove $\Leftarrow$.

For the sake of contradiction, we suppose that $A(x_1, x_2) \not\approx_t A(x_2, x_1)$.

There exist M_1 and M_2 that are closed such that $A(M_1, M_2) \not\approx_t A(M_2, M_1)$.

We suppose that $A(M_1, M_2) \not\sqsubseteq_t A(M_2, M_1)$ without loss of generality.

There exists $\mathbf{tr} \in \mathrm{tr}(A(M_1, M_2))$ such that any trace of $A(M_2, M_1)$ is not statically equivalent to $\mathbf{tr}$.

Let $\delta_1(z) : z = M_1$ and $\delta_2(z) : z = M_2$.

By assumption, $A(x_1, x_2) \models G(\delta_1(x_1) \to (\delta_2(x_2) \to P(\delta_1(x_2) \wedge \delta_2(x_1))))$.

Hence, $A(x_1, x_2), [x_1 \mapsto M_1, x_2 \mapsto M_2], \mathbf{tr}, |\mathbf{tr}| \models P(\delta_1(x_2) \wedge \delta_2(x_1))$.

However, this contradicts the fact that there exists $\mathbf{tr} \in \mathrm{tr}(A(M_1, M_2))$ such that any trace of $A(M_2, M_1)$ is not statically equivalent to $\mathbf{tr}$. □

Proposition 9. $\forall \psi \in \mathfrak{S}_p; A(x_1, ..., x_p) \approx_t A(x_{\psi(1)}, ..., x_{\psi(p)})$
$\Leftrightarrow$ $\{\delta_j\}_{j \in J}$ *is role permutable in* A *for all* $\{\delta_j\}_{j \in J}$.

Proof. $\Rightarrow$) We arbitrarily take $\widetilde{M}, n, \widetilde{i}$ and ψ'. We define ψ such that a permutation on p and $\psi(i_k) = i_{\psi'(k)}$ and $\psi(j) = j (j \notin \{i_1, ..., i_k\})$.

Let $\rho = [x_1 \mapsto M_1, ..., x_p \mapsto M_p]$.

We arbitrarily take $\mathbf{tr} \in \mathrm{tr}(A\rho)$ and t.

We suppose that $A, \rho, \mathbf{tr}, t \models \bigwedge_{k \leq n} \delta_{i_k}(x_{i_k}, \widetilde{y_k})$.

By assumption, $\exists \mathbf{tr'} \in \mathrm{tr}(A(M_{\psi^{-1}(1)}, ..., M_{\psi^{-1}(p)}))$ s.t. $\mathbf{tr} \sim_t \mathbf{tr'}$.

Let $\rho' = [x_1 \mapsto M_{\psi^{-1}(1)}, ..., x_p \mapsto M_{\psi^{-1}(p)}]$.

By Lemma 1, $\mathbf{tr}[t] \equiv_s \mathbf{tr'}[t]$ and $\mathbf{tr'}[t], \rho, \mathbf{tr'}[t], 0 \models \bigwedge_{k \leq n} \delta_{i_k}(\rho(x_{i_k}), \widetilde{y_k})$.

Thus, $A, \rho', \mathbf{tr'}, t \models \bigwedge_{k \leq n} \delta_{i_k}(\rho(x_{i_{\psi'(k)}}), \widetilde{y_k})$.

Hence, $A, \rho, \mathbf{tr}, t \models P(\bigwedge_{k \leq n} \delta_{i_k}(\rho(x_{i_{\psi'(k)}}), \widetilde{y_k}))$.

That is, $A \models G(\bigwedge_{k \leq n} \delta_{i_k}(x_{i_k}, \widetilde{y_k}) \to P(\bigwedge_{k \leq n} \delta_{i_k}(x_{i_{\psi'(k)}}, \widetilde{y_k})))$.

$\Leftarrow$) We arbitrarily take $\widetilde{M}$ and i.

Let $\rho = [x_1 \mapsto M_1, ..., x_p \mapsto M_p]$.

We arbitrarily take $\mathbf{tr} \in \mathrm{tr}(A\rho)$.

Let $\delta_j(z) : z = M_j$ for each $1 \leq j \leq p$. Let $J = \{1, ..., p\}$.

By assumption, $\{\delta_j\}_{j \in J}$ is role permutable in A.

We arbitrarily take $\psi \in \mathfrak{S}_p$.

Hence, $A \models G(\bigwedge_{k \leq p} \delta_k(x_k) \to P(\bigwedge_{k \leq p} \delta_k(x_{\psi(k)})))$.

Therefore, $A, \rho, \mathbf{tr}, |\mathbf{tr}| \models \bigwedge_{k \leq p} \delta_k(x_k) \to P(\bigwedge_{k \leq p} \delta_k(x_{\psi(k)}))$.

Because $A, \rho, \mathbf{tr}, |\mathbf{tr}| \models \bigwedge_{k \leq p} \delta_k(x_k)$, it holds that

$$A, \rho, \mathbf{tr}, |\mathbf{tr}| \models P(\bigwedge_{k \leq p} \delta_k(x_{\psi(k)}))$$

That is, $\exists \rho' \exists \mathbf{tr'} \in \mathrm{tr}(A\rho')$s.t. $\mathbf{tr} \sim_t \mathbf{tr'}$ and $A, \rho', \mathbf{tr'}, |\mathbf{tr'}| \models \bigwedge_{k \leq p} \delta_k(x_{\psi(k)})$.

This means that $\rho'(x_{\psi(k)}) = M_k$ for all k.

Let $\psi'(k) = \psi^{-1}(k)$.

Then, $\exists \mathbf{tr'} \in \mathrm{tr}(A(M_{\psi'(1)}, ..., M_{\psi'(p)}))$s.t. $\mathbf{tr} \sim_t \mathbf{tr'}$ holds. □

Problem 2.

Input: *An extended process A, an assignment ρ, a trace $\mathbf{tr} \in \mathrm{tr}(A)$, an integer $0 \leq i \leq |\mathbf{tr}|$, and a formula φ.*

Question: *Does $A, \rho, \mathbf{tr}, i \models \varphi$ hold?*

Proposition 10. *Even if the word problem in Σ is decidable, Problem 2 can be undecidable.*

Proof. We again reduce the decision problem for static equivalence to Problem 2.

Let φ and ψ be frames. We assume that $\mathrm{dom}(\varphi) = \mathrm{dom}(\psi)$.

Let $\varphi = \nu\widetilde{n}.\{{}^{M_1}/_{x_1}, ..., {}^{M_l}/_{x_l}\}, \psi = \nu\widetilde{m}.\{{}^{N_1}/_{x_1}, ..., {}^{N_l}/_{x_l}\}$.

Let $P = \nu\widetilde{n}.\overline{a}\langle M_1\rangle ... \overline{a}\langle M_l\rangle, Q = \nu\widetilde{m}.\overline{a}\langle N_1\rangle ... \overline{a}\langle N_l\rangle$, where $a \notin \widetilde{n} \cup \widetilde{m}$.

Let $A =$ if $x = b$ then P else Q and $\rho : x \mapsto b$.

Let

$$\mathbf{tr} = A\rho \overset{\nu x_1.\overline{a}\langle x_1\rangle}{\longrightarrow} \nu\widetilde{n}.(\overline{a}\langle M_2\rangle ... \overline{a}\langle M_l\rangle | \{{}^{M_1}/_{x_1}\}) \overset{\nu x_2.\overline{a}\langle x_2\rangle}{\longrightarrow} ... \overset{\nu x_l.\overline{a}\langle x_l\rangle}{\longrightarrow} \nu\widetilde{n}.\{{}^{M_1}/_{x_1}, ..., {}^{M_l}/_{x_l}\}$$

We prove that $\varphi \approx_s \psi \Leftrightarrow A, \rho, \mathbf{tr}, i \models P(x \neq b)$.

$\mathbf{tr}$ is the only trace of $A\rho$ which is statically equivalent to $\mathbf{tr}$.

We arbitrarily take an assignment ρ' which does not map x to b.

A trace $\mathbf{tr'} \in \mathrm{tr}(A\rho')$ whose actions correspond to $\mathbf{tr}$ is the only below:

$$A\rho' \overset{\nu x_1.\overline{a}\langle x_1\rangle}{\Longrightarrow} \nu\widetilde{m}.(\overline{a}\langle N_2\rangle ... \overline{a}\langle N_l\rangle | \{{}^{N_1}/_{x_1}\}) \overset{\nu x_2.\overline{a}\langle x_2\rangle}{\longrightarrow} ... \overset{\nu x_l.\overline{a}\langle x_l\rangle}{\longrightarrow} \nu\widetilde{m}.\{{}^{N_1}/_{x_1}, ..., {}^{N_l}/_{x_l}\}.$$

Because $A, \rho', \mathbf{tr'}, i \models x \neq b$, it holds that $A, \rho, \mathbf{tr}, i \models P(x \neq b) \Rightarrow \varphi \approx_s \psi$.

On the other hand, it holds that $\mathbf{tr} \sim_t \mathbf{tr'}$ if $\varphi \approx_s \psi$. The proof is similar to the proof of Proposition 1.

Hence, it follows that $\varphi \approx_s \psi \Rightarrow A, \rho, \mathbf{tr}, i \models P(x \neq b)$. □

B.2 Proof of Proposition 11

Problem 3.

Input: *An extended process A and a formula φ.*

Question: *Does $A \models \varphi$ hold?*

Proposition 11. *If the equational theory on Σ is a convergent subterm theory and the extended process A contains no replications, Problem 3 is decidable.*

Proof of Proposition 11 needs several lemmas.

Lemma 16. *Let A be an extended process and B a closed extended process.*

Consider a modal formula φ, assignments ρ, ρ' and an arbitrary $\mathbf{tr} \in \mathrm{tr}(B)$, together with a permutation π which does not alter names in A, $\mathbf{tr}[0, i]$ and φ for some $i \leq |\mathbf{tr}|$.

We assume that ρ' is obtained by π from ρ.

Let $X_i = \{\mathbf{tr}' \in \mathrm{tr}(A\rho) | \mathbf{tr}'[0,i] \sim_t \mathbf{tr}[0,i]\}$ and $Y_i = \{\mathbf{tr}' \in \mathrm{tr}(A\rho') | \mathbf{tr}'[0,i] \sim_t \mathbf{tr}[0,i]\}$.

Then there exists a bijection $f_i : X_i \to Y_i$ such that

$$A, \rho, \mathbf{tr}', i \models \varphi \Leftrightarrow A, \rho', f_i(\mathbf{tr}'), i \models \varphi.$$

Proof. We define f_i as $f_i(\mathbf{tr}') = \pi(\mathbf{tr}')$.

By assumption, $\rho' = \pi(\rho)$.

$$A, \rho, \mathbf{tr}', i \models \varphi \Leftrightarrow A, \pi(\rho), \pi(\mathbf{tr}'), i \models \varphi \Leftrightarrow A, \rho', f_i(\mathbf{tr}'), i \models \varphi.$$

□

Lemma 17. *Let T be a finite set of variables. Let S be a finite set of names.*

We define an equivalence relation $\asymp_S$ between assignments which is a map from T to names.

$$\asymp_S = \{(\rho, \rho') | \text{There exists a permutation } \pi \text{ which does not change names in } S \text{ such that } \rho' \text{ is obtained by } \pi \text{ from } \rho.\}$$

Then the quotient space by $\asymp_S$ is finite.

Proof. Let $T = \{x_1, ..., x_l, y_1, ..., y_{m_n}\}$ and

$$\rho = [x_1 \mapsto a_1, ..., x_l \mapsto a_l, \quad y_1 \mapsto b_1, ..., y_{m_1} \mapsto b_1, y_{m_1+1} \mapsto b_2, ..., y_{m_2} \mapsto b_2, ..., y_{m_n} \mapsto b_n],$$

where $a_1, ..., a_l$ are names in S and $b_1, ..., b_n$ are names which is not in S. In addition, $i \neq j \Rightarrow b_i \neq b_j$, but we do not assume that $i \neq j \Rightarrow a_i \neq a_j$.

Then $\rho \asymp_S \rho' \Leftrightarrow [\rho(x_i) = \rho'(x_i)$ and $\rho'(y_i) = \pi(\rho(y_i))$ for some $\pi]$.

Each equivalence class is determined by a division of variables such as described above. The number of such divisions is finite, so the quotient space is finite. □

Proposition 20. *If static equivalence and word problem in Σ are decidable and ρ is restricted to an assignment to names, then Problem 2 is decidable.*

Proof. We prove by induction on φ.

1. $\top$.

   $A, \rho, \mathbf{tr}, i \models \top$ always holds, so this is decidable.

2. $M_1 = M_2$.

   By assumption, this is decidable.

3. $M \in \mathrm{dom}$.

   This relation is trivially decidable.

4. $\delta_1 \vee \delta_2$.

   By induction hypothesis, δ_1 and δ_2 are decidable, so $\delta_1 \vee \delta_2$ is so.

   Moreover, $\neg\delta, \varphi_1 \vee \varphi_2$ and $\neg\varphi$ are similar.

5. $\langle\mu\rangle_{-}\varphi$.

   Whether $\mathbf{tr}[i-1] \overset{\mu}{\Longrightarrow} \mathbf{tr}[i]$ holds in $\mathbf{tr}$ is clearly decidable.

   By induction hypothesis, $A, \rho, \mathbf{tr}, i-1 \models \varphi$ is decidable, so $\langle\mu\rangle_{-}\varphi$ is so.

6. $F\varphi$.

   By induction hypothesis, $A, \rho, \mathbf{tr}, i \models \varphi, ..., A, \rho, \mathbf{tr}, |\mathbf{tr}| \models \varphi$ are decidable, so $F\varphi$ is so.

7. $K\varphi$.

   We consider a quotient space by $\asymp_{\mathrm{n}(\mathbf{tr}[0,i]) \cup \mathrm{n}(A) \cup \mathrm{n}(\varphi)}$. By Lemma 17, this space is finite. Here, $\mathrm{n}(\mathbf{tr}[0,i])$ is a set of names which appear in $\mathbf{tr}[0,i]$.

   A division of variables determines each equivalence class, so this is computable.

   We arbitrarily take a representative ρ' of each equivalence class.

   By Proposition 2, whether there exists $\mathbf{tr'} \in \mathrm{tr}(A\rho')$ such that $\mathbf{tr}[0,i] \sim_t \mathbf{tr'}[0,i]$ is decidable.

   If such traces exist, the number of them is finite.

   We arbitrarily take $\mathbf{tr'} \in \mathrm{tr}(A\rho')$ such as $\mathbf{tr'}[0,i] \sim_t \mathbf{tr}[0,i]$.

   By induction hypothesis, $A, \rho', \mathbf{tr'}, i \models \varphi$ is decidable.

   We arbitrarily take $\rho'' \asymp_{\mathrm{n}(\mathbf{tr}[0,i]) \cup \mathrm{n}(A) \cup \mathrm{n}(\varphi)} \rho'$.

   By Lemma 16,
   $$A, \rho', \mathbf{tr'}, i \models \varphi \Leftrightarrow A, \rho'', f_i(\mathbf{tr'}), i \models \varphi$$
   for some π such that $\rho'' = \pi(\rho')$. Moreover, $\mathbf{tr'}[0,i] \sim_t f_i(\mathbf{tr'})[0,i]$. This is because $f_i(\mathbf{tr'}[0,i]) \sim_t \mathbf{tr}[0,i]$.

   This is why we only have to check whether a representative ρ' of each equivalence class satisfies that $A, \rho', \mathbf{tr'}, i \models \varphi$.

   This procedure can always be completed because of the finiteness of the quotient space.

   In other words, $A, \rho, \mathbf{tr}, i \models K\varphi$ is decidable.

□

It is proved in [3, Theorem 1] that static equivalence on a convergent subterm theory is decidable, so the corollary below immediately follows.

Corollary 4. *If the equational theory on Σ is a convergent subterm theory and ρ is restricted to an assignment to names, then Problem 2 is decidable.*

Now, Proposition 11 immediately follows.