A probabilistic polynomial-time calculus for the analysis of cryptographic protocols *†

J. Mitchell
Stanford University

A. Ramanathan
Stanford University

A. Scedrov ‡
University of Pennsylvania

V. Teague §
Stanford University

April 11, 2002

Abstract

We prove properties of a process calculus that is designed for analyzing security protocols. Our long-term goal is to develop a form of protocol analysis, consistent with standard cryptographic assumptions, that provides a language for expressing probabilistic polynomial-time protocol steps, a specification method based on a compositional form of equivalence, and a logical basis for reasoning about equivalence.

The process calculus is a restricted form of π-calculus, with bounded replication and probabilistic polynomial-time expressions allowed in messages and boolean tests. To avoid inconsistency between security and nondeterminism, messages are scheduled probabilistically instead of nondeterministically. We prove that evaluation of any process expression halts in probabilistic polynomial time and define a form of asymptotic protocol equivalence that allows security properties to be expressed using *observational equivalence*, a standard relation from programming language theory that involves quantifying over possible environments that might interact with the protocol.

A technique based on a form of bisimulation is proved sufficient to establish basic instances of observational equivalences. The method is illustrated by showing that a standard cryptographic definition, the cryptographically strong pseudo-random number generator, is characterized by observational equivalence.

*Partially supported by DoD MURI "Semantic Consistency in Information Exchange," ONR Grant N00014-97-1-0505.

†Additional support by OSD/ONR CIP/SW URI "Software Quality and Infrastructure Protection for Diffuse Computing," ONR Grant N00014-01-1-0795.

‡Additional support from NSF Grants CCR-9800785 and CCR-0098096.

§Partially supported by Stanford Graduate Fellowship.

1 Introduction

There are a variety of methods used in the analysis of security protocols. The main systematic or formal approaches include specialized logics such as BAN logic [BAN89, DMP01], special-purpose tools designed for cryptographic protocol analysis [KMM94], and theorem proving [Pau97b, Pau97a] and model-checking techniques using several general purpose tools described in [Low96, Mea96, MMS97, Ros95, Sch96]. Although these approaches differ in significant ways, all reflect the same basic assumptions about the way an adversary may interact with the protocol or attempt to decrypt encrypted messages. This common model, largely derived from [DY81] and suggestions found in [NS78] (see, *e.g.,* [CDL$^+$99]), allows a protocol adversary to nondeterministically choose among possible actions. This convenient idealization is intended to give the adversary a chance to find an attack if one exists. In the presence of nondeterminism, however, the set of messages an adversary may use to interfere with a protocol must be restricted severely. Although these Dolev-Yao-style assumptions make protocol analysis tractable, they also make it possible to "verify" protocols that are in fact susceptible to simple attacks that lie outside the adversary model. A further limitation of deterministic or nondeterministic settings is the inability to analyze probabilistic protocols.

In this paper we describe some technical properties of a process calculus that was proposed earlier in [LMMS98, LMMS99] as the basis for a form of protocol analysis that is formal, yet close in foundations to the mathematical setting of modern cryptography. The framework relies on a language for defining communicating polynomial-time processes [MMS98]. The reason we restrict processes to probabilistic polynomial time is so that we can reason about the security of protocols by quantifying over all "adversarial" processes definable in the language. In effect, establishing a bound on the running time of an adversary allows us to relax the simplifying assumptions on what the adversary might do. In particular, we can consider adversaries that might send randomly chosen messages, or perform sophisticated (yet probabilistic polynomial-time) computation to derive an attack from messages they overhear on the network. An important aspect of our framework is that we can analyze probabilistic as well as deterministic encryption functions and protocols. Without a probabilistic framework, it would not be possible to analyze an encryption function such as ElGamal [ElG85], for which a single plaintext may have more than one ciphertext.

Some of the basic ideas of this work are outlined in [LMMS98], with the term language presented in [MMS98] and further example protocols considered in [LMMS99]. Much of this paper is based on a preliminary report in [MRST01]. The closest technical precursor is the Abadi and Gordon spi-calculus [AG99, AG98] which uses observational equivalence and channel abstraction but does not involve probability or computational complexity bounds; subsequent related work is cited in [AF01], for example. Prior work on CSP and security protocols, e.g., [Ros95, Sch96], also uses process calculus and security specifications in the form of equivalence or related approximation orderings on processes. An important parallel effort with similar goals, the paradigm of "universally composable security", can be found in [Can00].

In this paper we establish that protocols specified in our language can be evaluated in probabilistic polynomial time and we provide a tool based on bisimulation [Mil89] to obtain some simple observational equivalences. Although our main long-term objective is to base protocol analysis on standard cryptographic assumptions, this framework may also shed new light on basic questions in cryptography. In particular, the characterization of "secure" encryption function, for use in protocols, does not appear to have been completely settled. While the definition of *semantic security* in [GM84] appears to have been accepted, there are stronger notions such as *non-malleability* [DDN91] that are more appropriate to protocol analysis. In a sense, the difference is that semantic security is natural for the single transmission of an encrypted message, while non-malleability accounts for vulnerabilities that may arise in more complex protocols. Our framework provides a setting for working backwards from security properties of a protocol to derive necessary properties of underlying encryption primitives. While we freely admit that much more needs to be done to produce a systematic analysis method, we believe that a foundational setting for protocol analysis that incorporates probability and complexity restrictions has much to offer in the future.

2 Preliminaries

2.1 Probabilistic Functions

Definition 2.1. We define a *probabilistic function* $\mathcal{F}$ from X to Y as a function $\mathcal{F} : X \times Y \to [0,1]$ that satisfies the following two conditions:

1. $\forall x \in X : \sum_{y \in Y} \mathcal{F}(x,y) \leq 1$,
2. For each x in X, the size of the set $\{y |\, y \in Y, \mathcal{F}(x,y) > 0\}$ is finite.

For some $x \in X, y \in Y$, if $\mathcal{F}(x,y) = p$, we say that $\mathcal{F}$ *maps* x *to* y *with probability* p or that $\mathcal{F}(x) = y$ with probability p.

Definition 2.2. Let $\mathcal{F} : X \times Y \to [0,1]$ be a probabilistic function. We refer to X as the *domain of* $\mathcal{F}$, Y as the *codomain of* $\mathcal{F}$, and $\cup_{x \in X} \{y |\, y \in Y, \mathcal{F}(x,y) > 0\}$ as the *range of* $\mathcal{F}$.

2.2 Composition of Probabilistic Functions

Definition 2.3. We define the *composition* $\mathcal{F} : X \times Z \to [0,1]$ of two probabilistic functions $\mathcal{F}_1 : X \times Y \to [0,1]$ and $\mathcal{F}_2 : Y \times Z \to [0,1]$ as a probabilistic function that satisfies the following condition:

$$\forall x \in X . \forall z \in Z : \mathcal{F}(x,z) = \sum_{y \in Y} \mathcal{F}_1(x,y) \cdot \mathcal{F}_2(y,z) \tag{2.1}$$

We write the composition of $\mathcal{F}_1$ and $\mathcal{F}_2$ as $\mathcal{F}_2 \circ \mathcal{F}_1$.

Lemma 2.4. *Given two probabilistic functions $\mathcal{F}_1 : X \times Y \to [0,1]$ and $\mathcal{F}_2 : Y \times Z \to [0,1]$, the composition $\mathcal{F}_2 \circ \mathcal{F}_1 = \mathcal{F} : X \times Z \to [0,1]$ is a probabilistic function, i.e., it satisfies the condition $\forall x \in X : \sum_{z \in Z} \mathcal{F}(x,z) \leq 1$.*

Proof. For any fixed $x \in X$:

$$\begin{aligned}
\sum_{z \in Z} \mathcal{F}(x,z) &= \sum_{z \in Z} \sum_{y \in Y} \mathcal{F}_1(x,y) \cdot \mathcal{F}_2(y,z) && \text{by defn. 2.3} \\
&= \sum_{y \in Y} \Big(\mathcal{F}_1(x,y) \cdot \sum_{z \in Z} \mathcal{F}_2(y,z) \Big) \\
&\leq \sum_{y \in Y} \mathcal{F}_1(x,y) && \text{by defn. 2.1} \\
&\leq 1 && \text{by defn. 2.1}
\end{aligned}$$

Hence, composition is a probabilistic function. □

2.3 Terminology

In what follows, P denotes a process, T denotes a term, $p(x)$ denotes one of an infinity of *bandwidth polynomials* such that $\forall a \in \mathbb{N} : p(a) \geq 0$. $c_{p(x)}$ denotes one of a countable infinity $\mathbf{C}_{q(x)}$ of *channel names* associated with the polynomial $p(x)$ in such a way that no channel name is associated with more than one polynomial.

There is a special *security parameter*, n, that can appear in terms (section 2.3.1), as the parameter of the replication operator (defn. 2.14), as the argument to the bandwidth polynomials, and in contexts (defn. 2.24). The bandwidth polynomial gives, for each value of n, the maximum number of bits that can be transmitted on that channel (explored in more detail in defn. 3.3). The two-place relation $\equiv$ stands for syntactic identity. We denote variables by the letters x, y and so on. Finally, the function $\gamma(x)$ is a polynomial such that $\forall a \in \mathbb{N} : \gamma(a) \geq 0$.

Definition 2.5. [Ata99] An *oracle Turing machine* is a Turing machine with an extra oracle tape and three extra states q_{query}, q_{yes} and q_{no}. When the machine enters state q_{query} control passes to the state q_{yes} if the contents on the oracle tape are in the oracle set; otherwise, control passes to the state q_{no}. Given an oracle Turing machine M, we will write $M(\varphi, \vec{x})$ to specify the behavior of M on input $\vec{x}$ using oracle φ.

We only consider binary oracles *i.e.,* oracles that produce either a '1' or a '0' when queried. Thus a binary oracle φ determines a set $X_\varphi = \{x | \varphi(x)\}$. Querying φ at x then determines whether $x \in X_\varphi$.

Definition 2.6. An oracle Turing machine M runs in *oracle polynomial time* if there exists a polynomial q such that for all oracles φ, $M(\varphi, \vec{x})$ halts in time $q(|\vec{x}|)$ where $\vec{x} = x_1, \ldots, x_m$ and $|\vec{x}| = |x_1|, \ldots, |x_m|$.

In order to query the oracle, M must write the number at which it wants to query the oracle on the oracle tape. If M runs in oracle polynomial time, then $M(\vec{x})$ queries the oracle set with at most $q(|\vec{x}|)$ bits.

Definition 2.7. Let M be a poly-time oracle Turing machine. We can view M as a *probabilistic poly-time Turing machine* if we randomly choose an oracle from the space of oracles that can be queried in the time bound of M. More precisely, let M be an oracle machine running in time bounded by the polynomial q. Since $M(\vec{x})$ can only query the oracle with at most $q(\vec{x})$ bits, we have a finite space of oracles on which M runs in time bounded by q. Call this space $\mathbf{q}$. Then, we can view M as a probabilistic poly-time Turing machine where we say that $M(\vec{x}) = y$ with probability p if, choosing an oracle φ uniformly at random from $\mathbf{q}$, the probability that $M(\vec{x}, \varphi) = y$ is p.

Given a poly-time probabilistic Turing machine M, we will write $M(\vec{x})$ to specify the behavior of M on input $\vec{x}$ using a randomly chosen oracle.

Definition 2.8. A probabilistic poly-time Turing machine M *computes* a probabilistic function $\mathcal{F}$ if for all inputs $\vec{x}$ and for all outputs y we have that $\mathcal{F}(\vec{x}, y) = \mathrm{Prob}\big[M(\vec{x}) = y\big]$.

We note that a probabilistic poly-time Turing machine cannot make more queries than a polynomial in the size of its input, each of which has polynomially-bounded length, before it provides an output. Consequently, a probabilistic poly-time Turing machine can only provide a finite (polynomially-bounded in the size of its input) number of outputs for each given output. As a consequence, every probabilistic poly-time Turing machine must satisfy the second condition of the definition of a probabilistic function (that only a finite number of outputs can be possibly produced for each input) simply because these machines are polynomially-bounded.

Definition 2.9. A probabilistic function $\mathcal{F}$ is *poly-time* if it is computed by a probabilistic poly-time Turing machine.

2.3.1 Terms

A functional term calculus based on [Hof99, Bel92] and a semantics for that calculus are studied in [MMS98]. Though we omit the details here, we do note that terms can contain the security parameter n as well as a function *rand* that returns one random bit.

For each term T with variables $x_1, \ldots, x_k$, there is a probabilistic poly-time Turing machine M_T of $k+1$ inputs and a polynomial $q_T(v_1, \ldots, v_{k+1})$ such that the following two theorems hold:

Theorem 2.10. *[MMS98] Let T be a term with k variables and let M_T be the machine associated with T. Then T, with $a_1, \ldots, a_k$ substituted for its k variables and with n chosen as the value for its security parameter, reduces to a with probability p if and only if $M_T(a_1, \ldots, a_k, n)$ returns a with probability p in time at most $q_T(|a_1|, \ldots, |a_k|, |n|)$.*

Essentially, we will simply consider the probabilistic poly-time Turing machine M_T to define the meaning of the term T.

Theorem 2.11. *[MMS98] For each probabilistic poly-time function f, there exists a term T such that M_T computes f.*

Let T be a term with k variables given by x^k and let M_T be the Turing machine associated with T. Since M_T is a probabilistic polynomial-time Turing machine, M_T must compute a probabilistic function f *i.e.*, $M_T(a^k, |n|)$ evaluates to a with probability p if and only if $f(a^k, |n|)$ evaluates to a with probability p. We will write $T(a^k, |n|)$ to mean that T has values for its variables given by a^k and n as its security parameter. Theorem 2.10 insists $T(a^k, n)$ reduces to a with probability p if and only if $M_T(a^k, |n|)$ evaluates to a with probability p whence $T(a^k, n)$ reduces to a with probability p if and only if $f(a^k, |n|)$ evaluates to a with probability p. Since f is a probabilistic function, T must also be a probabilistic function (with domain $\mathbb{N}^k \times \mathbb{N}$ and codomain $\mathbb{N}$).

Definition 2.12. We write $T \xrightarrow{r}_e a$ on inputs $x_1, \ldots, x_k$ if the probability that $M_T(x_1, \ldots, x_k, n) = a$ is r. We say that T *evaluates* to a with probability r.

We note that the atom a evaluates to a with probability one *i.e.*, $a \xrightarrow{1}_e a$.

Definition 2.13. We define two probabilities here: (1) the probability of two terms evaluating to the same number, and, (2) the probability of two terms evaluating to different numbers.

1. $\mathrm{Prob}_{=}\big[T_1, T_2\big] = \sum_a s_1 s_2$,
   where $a \in \mathbb{N}, T_1 \xrightarrow{s_1}_e a, T_2 \xrightarrow{s_2}_e a$, and
2. $\mathrm{Prob}_{\neq}\big[T_1, T_2\big] = \sum_{\langle a_1, a_2 \rangle} s_1 s_2$,
   where $a_1, a_2 \in \mathbb{N}, a_1 \neq a_2, T_1 \xrightarrow{s_1}_e a_1, T_2 \xrightarrow{s_2}_e a_2$.

2.3.2 Processes

Definition 2.14. The syntax of processes is given by the following grammar:

$$
\begin{array}{lcll}
P & ::= & \mathbf{0} & \text{(termination)} \\
 & & \nu_{c_{p(|n|)}}.(P) & \text{(private channel)} \\
 & & c_{p(|n|)}(x).(P) & \text{(input)} \\
 & & c_{p(|n|)}\langle T \rangle & \text{(output)} \\
 & & [T = T].(P) & \text{(match)} \\
 & & (P \mid P) & \text{(parallel composition)} \\
 & & !_{\gamma(|n|)}.(P) & (\gamma(|n|)\text{-fold replication})
\end{array}
$$

Every input or output on a private channel must appear within the scope of a ν-operator binding that channel. Otherwise the expression is considered unevaluable.

Another word about channel names. The polynomial $p(|n|)$ associated with the channel c is a *bandwidth parameter* on c—for further details see section 3.3. Also, as mentioned in section 2.3, no channel name is associated with more than one polynomial.

For simplicity, after fixing n, when we evaluate a process P, we replace all sub-processes (see defn. 2.25) of P of the form $!_{\gamma(|n|)}.R$ with $\gamma(|n|)$ copies of R in parallel. Furthermore, we define $!_{\gamma(|n|)}.R$ to be left associative. Finally, we assume that all channel names and variable names are α-renamed apart.

Finally, in what follows we will tend to omit parentheses if the parse tree of an expression is unambiguous.

Definition 2.15. A process P may contain the security parameter n (section 2.3). We will write P^m to signify that the parameter n is assigned the natural number m.

Definition 2.16. We define an *input expression* as a process of the form $c_{p(|n|)}(x).P$ and an *output expression* is of the form $c_{p(|n|)}\langle T\rangle$.

Definition 2.17. We call a process expression with no free variables a *closed* process expression.

Definition 2.18. Let P be a process expression and let T be a term. If T is not in the scope of any input operator, we say that T is an *exposed term.* Similarly, let $[T_1 = T_2].R$ be a sub-process (see defn. 2.25) of P. We will say that $[T_1 = T_2].R$ is an *exposed match* if it does not appear in the scope of any input operator.

Definition 2.19. We define the indicator function $\mathrm{Eq?}_{\mathbf{0}}(P)$ as a unary function on processes that has the value 1 when its input is the **0** process and 0 otherwise:

$$\mathrm{Eq?}_{\mathbf{0}}(Q) = \begin{cases} 0 & \text{if } Q \not\equiv \mathbf{0}, \\ 1 & \text{if } Q \equiv \mathbf{0}. \end{cases}$$

Definition 2.20. *outerEval* is a function from pairs of closed process expressions and closed process expressions with no exposed un-evaluated terms and no exposed matches to the interval $[0, 1]$ (meaning that the first expression in the pair outer-evaluates to the second expression in the pair with probability given by the output value of the function):

$$\begin{array}{ll}
\mathit{outerEval}(\mathbf{0}, \mathbf{0}) & = 1 \\
\mathit{outerEval}(\nu_{c_{p(|n|)}}.(P), \nu_{c_{p(|n|)}}.(P')) & = s_1 \\
\mathit{outerEval}(c_{p(|n|)}(x).(Q), c_{p(|n|)}(x).(Q)) & = 1 \\
\mathit{outerEval}(c_{p(|n|)}\langle T\rangle, c_{p(|n|)}\langle a\rangle) & = s_2 \\
\mathit{outerEval}([T_1 = T_2].(Q_1), Q_2) & \\
\quad = s_3 \cdot \mathrm{Prob}_{=}[T_1, T_2] + \mathrm{Eq?}_{\mathbf{0}}(Q_2) \cdot \mathrm{Prob}_{\neq}[T_1, T_2] & \\
\mathit{outerEval}((P_1 \mid P_2), (P_1' \mid P_2')) & = s_4 \cdot s_5 \\
\mathit{outerEval}(P, Q) & = 0 \text{ otherwise.}
\end{array}$$

where $\mathit{outerEval}(P, P') = s_1$,

$a \in \mathbb{N}, T \xrightarrow{s_2}_e a$,

$\mathit{outerEval}(Q_1, Q_2) = s_3$,

$\mathit{outerEval}(P_1, P_1') = s_4$,

$\mathit{outerEval}(P_2, P_2') = s_5$

If $outerEval(P, P') = s$ then we say that P *outer-evaluates* to P' with probability s, or that $P \xrightarrow{s}_o P'$.

The match case in the definition of *outerEval* bears some discussion. Consider the match $[T_1 = T_2].P$. We wish to compute the probability of P outer-evaluating to P'. There are two distinct cases here:

1. $P' \not\equiv \mathbf{0}$, and,
2. $P' \equiv \mathbf{0}$.

In the first case, we will only outer-evaluate P if the match succeeds, *i.e.,* $T_1 = T_2$. In the second case, however, we can arrive at $\mathbf{0}$ either by passing the match and outer-evaluating P to $\mathbf{0}$ or by simply failing the match. Hence, the probability of outer-evaluating to P' is given by the sum of the product of the probability of passing the match and outer-evaluating P to P' and, in the case that we wish to outer-evaluate to a $\mathbf{0}$, failing the match—hence the use of the indicator function in the second term of the sum.

We must be sure, however, that we do not over count in this expression. From the definition of cases 1 and 2 we can see that P' can never satisfy both cases simultaneously—hence cases 1 and 2 are disjoint. We do not over count inside case 2 as the sets over which we define the two summations in case 2 (one from the $\mathrm{Prob}_=\big[T_1, T_2\big]$ term and one from the $\mathrm{Prob}_{\neq}\big[T_1, T_2\big]$ term) are disjoint.

Finally, it should be clear from that definition that $outerEval(P, P) = 1$ only when P cannot be outer-evaluated. Otherwise, $outerEval(P, P) = 0$ since an expression that can be outer-evaluated will be outer-evaluated.

Lemma 2.21. *outerEval is a probabilistic function.*

Proof. Let $\mathbf{P}$ be the set of all closed process expressions. We proceed by an induction on the structure of process expressions. The basis is easy to establish:

1. $outerEval(\mathbf{0}, \mathbf{0}) = 1$. Since $outerEval(\mathbf{0}, P) = 0$ whenever $P \not\equiv \mathbf{0}$ (as $outerEval(P, Q) = 0$ when $outerEval(P, Q)$ does not fall under one of the other cases in the definition of *outerEval*), it follows that $\{P | \, P \in \mathbf{P}, outerEval(\mathbf{0}, P) > 0\}$ is a finite set and that $\sum_{P \in \mathbf{P}} outerEval(\mathbf{0}, P) = 1 \leq 1$.

2. $outerEval(c_{p(|n|)}(x).(Q), c_{p(|n|)}(x).(Q)) = 1$. As in the previous case, it is easy to establish that $\{P | \, P \in \mathbf{P}, outerEval(c_{p(|n|)}(x).(Q), P) > 0\}$ is a finite set and that $\sum_{P \in \mathbf{P}} outerEval(c_{p(|n|)}(x).(Q), P) = 1 \leq 1$.

3. $outerEval(c_{p(|n|)}\langle T\rangle, c_{p(|n|)}\langle a\rangle) = p$ where $p = \mathrm{Prob}\big[T \rightarrow_e a\big]$. We first note that *outerEval* always maps $c_{p(|n|)}\langle T\rangle$ to $P \not\equiv c_{p(|n|}\langle a\rangle$ with probability 0. Then it is easy to see that because T is a probabilistic function, the set of expressions to which *outerEval* maps $c_{p(|n|)}\langle T\rangle$ with non-zero probability is finite and that, furthermore, $\sum_{P \in \mathbf{P}} outerEval(c_{p(|n|)}\langle T\rangle, P) \leq 1$.

We assume as our inductive hypothesis that for all process expressions P of length less than k the following two statements hold:

1. $|\{Q|\, Q \in \mathbf{P}, \mathit{outerEval}(P,Q) > 0\}|$ is finite,
2. $\sum_{Q\in\mathbf{P}} \mathit{outerEval}(P,Q) \leq 1$.

We now proceed to establish the inductive step:

1. $\mathit{outerEval}(\nu_{c_{p(|n|)}}.(P), \nu_{c_{p(|n|)}}.(P')) = \mathit{outerEval}(P,P')$. We know that $\mathit{outerEval}(\nu_{c_{p(|n|)}}.(P), Q \not\equiv \nu_{c_{p(|n|)}}.(R)) = 0$ since $\mathit{outerEval}(P,Q) = 0$ when it does not fit into any of the other cases. The inductive hypothesis allows us to easily complete the proof for this case since the probability that *outerEval* maps $\nu_{c_{p(|n|)}}.(P)$ to $\nu_{c_{p(|n|)}}.(Q)$ is exactly the probability that *outerEval* maps P to Q.

2. $\mathit{outerEval}([T_1 = T_2].(Q_1), Q_2) = \mathit{outerEval}(Q_1,Q_2) \cdot \mathrm{Prob}_=\big[T_1,T_2\big] + \mathrm{Eq?}_{\mathbf{0}}(Q_2) \cdot \mathrm{Prob}_{\neq}\big[T_1,T_2\big]$. Now

   $$\sum_{Q_2\in\mathbf{P}} \mathit{outerEval}(Q_1,Q_2) \cdot \mathrm{Prob}_=\big[T_1,T_2\big] + \mathrm{Eq?}_{\mathbf{0}}(Q_2) \cdot \mathrm{Prob}_{\neq}\big[T_1,T_2\big]$$

   $$= \mathrm{Eq?}_{\mathbf{0}}(Q_2) \cdot \mathrm{Prob}_{\neq}\big[T_1,T_2\big] + \mathrm{Prob}_=\big[T_1,T_2\big] \cdot \sum_{Q_2\in\mathbf{P}} \mathit{outerEval}(Q_1,Q_2)$$

   $$\leq \mathrm{Eq?}_{\mathbf{0}}(Q_2) \cdot \mathrm{Prob}_{\neq}\big[T_1,T_2\big] + \mathrm{Prob}_=\big[T_1,T_2\big]$$

   since, by the inductive hypothesis, $\sum_{Q_2\in\mathbf{P}} \mathit{outerEval}(Q_1,Q_2) \leq 1$. But

   $$\mathrm{Eq?}_{\mathbf{0}}(Q_2) \cdot \mathrm{Prob}_{\neq}\big[T_1,T_2\big] + \mathrm{Prob}_=\big[T_1,T_2\big] \leq \mathrm{Prob}_{\neq}\big[T_1,T_2\big] + \mathrm{Prob}_=\big[T_1,T_2\big]$$

   since $\mathrm{Eq?}_{\mathbf{0}}(Q_2)$ returns either a 1 or a 0. Finally, it is easy to see that $\mathrm{Prob}_{\neq}\big[T_1,T_2\big] + \mathrm{Prob}_=\big[T_1,T_2\big] \leq 1$ which establishes one half of the inductive hypothesis. To obtain the finiteness condition, we observe that $\{P|\, P \in \mathbf{P}, \mathit{outerEval}([T_1 = T_2].(Q_1), P)\}$ is just the set $\{\mathbf{0}\} \cup \{P|\, P \in \mathbf{P}, \mathit{outerEval}(Q_1, P)\}$ in the case that $\mathrm{Prob}_{\neq}\big[T_1,T_2\big] > 0$ and $\{P|\, P \in \mathbf{P}, \mathit{outerEval}(Q_1, P)\}$ otherwise.

3. $\mathit{outerEval}((P_1 \mid P_2), (P_1' \mid P_2')) = \mathit{outerEval}(P_1,P_1') \cdot \mathit{outerEval}(P_2,P_2')$. Now

   $$\sum_{Q_1,Q_2\in\mathbf{P}} \mathit{outerEval}(P_1 \mid P_2, Q_1 \mid Q_2) = \sum_{Q_1,Q_2\in\mathbf{P}} \mathit{outerEval}(P_1,Q_1) \cdot \mathit{outerEval}(P_2,Q_2)$$

   which is just

   $$\sum_{Q_1\in\mathbf{P}} \mathit{outerEval}(P_1,Q_1) \cdot \sum_{Q_2\in\mathbf{P}} \mathit{outerEval}(P_2,Q_2)$$

   which, by the inductive hypothesis, is at most 1. Since *outerEval* always maps $P_1 \mid P_2$ to and expression $Q \not\equiv Q_1 \mid Q_2$ with probability 0, we establish one-half of the inductive hypothesis. To obtain the finiteness condition, we note that $\{Q_1, Q_2 | P \in \mathbf{P}, outerEval(P_1 \mid P_2, Q_1 \mid Q_2)\}$ is just the cross-product $\{Q_1 | P \in \mathbf{P}, outerEval(P_1, Q_1)\} \times \{Q_2 | P \in \mathbf{P}, outerEval(P_2, Q_2)\}$.

We have established that $\forall P \in \mathbf{P} : \sum_{Q \in \mathbf{P}} outerEval(P, Q) \leq 1$ and that for each $P \in \mathbf{P}$ the size of the set $\{Q | Q \in \mathbf{P}, outerEval(P, Q) > 0\}$ is finite. Since the set of all closed process expressions with no exposed un-evaluated terms or no exposed matches is a subset of the set of all closed process expressions, *outerEval* must be a probabilistic function. □

Lemma 2.22. *Let P, Q and R be closed process expressions such that $P \xrightarrow{s_1>0}_o Q$ and $Q \xrightarrow{s_2>0}_o R$. Then, Q has all exposed terms reduced to atoms and no exposed matches, $Q \equiv R$, and $s_2 = 1$.*

Proof. By a straightforward induction on the structure of P. We give a sketch of the proof.

The bases, $P \equiv \mathbf{0}$ and $P \equiv c_{p(|n|)}\langle T \rangle$, are easy to see.

In the case that $P \equiv \nu_{c_{p(|n|)}}.(P')$ we notice that $Q \equiv \nu_{c_{p(|n|)}}.(Q')$ where $P' \xrightarrow{s_1}_o Q'$. By the inductive hypothesis, we have that $Q' \xrightarrow{1}_o Q'$ and the result follows. In the case that $P \equiv c_{p(|n|)}(x).P'$ we notice that $P \xrightarrow{1}_o Q$ and the result follows. In the case that $P \equiv [T_1 = T_2].P'$ we have two cases. If both T_1 and T_2 evaluate to the same number then $Q \equiv Q'$ where $P' \xrightarrow{t_1}_o Q'$. By the inductive hypothesis $Q' \xrightarrow{1}_o Q'$ and the result follows. In the case that T_1 and T_2 evaluate to different numbers we can see that $Q \equiv \mathbf{0}$ and the result follows easily. In the case that $P \equiv P_1 \mid P_2$ the inductive hypothesis gives us that $P_1 \xrightarrow{s_1'}_o Q_1 \xrightarrow{1}_o Q_1$ and $P_2 \xrightarrow{s_2'}_o Q_2 \xrightarrow{1}_o Q_2$ and the result follows. □

So, outer-evaluation is idempotent.

Corollary 2.23. *Let P and Q be processes and let $\rightarrow_{o,m}$ represent the m-fold composition of $\rightarrow_o$. If $P \xrightarrow{s>0}_o Q$, then $Q \xrightarrow{1}_{o,m} Q$.*

Proof. Directly from lemma 2.22. □

Definition 2.24. Let Γ be the set of expressions generated by the following grammar:

$$
\begin{array}{rcl}
C[\ ] & ::= & [\ ]_i \\
 & & \nu_{c_{p(|n|)}}.(C[\ ]) \\
 & & c_{p(|n|)}(x).(C[\ ]) \\
 & & [T = T].(C[\ ]) \\
 & & (C[\ ] \mid C[\ ]) \\
 & & (C[\ ] \mid P) \\
 & & (P \mid C[\ ]) \\
 & & !_{\gamma(|n|)}.(C[\ ])
\end{array}
$$

where $i \in \mathbb{N}$.

A *context* is a process expression that is a member of Γ whose "holes" (indicated by numbered empty square brackets $[\ ]_i$) are numbered uniquely.

Given a context $C[\ ]$ and processes $P_1, \ldots, P_m$, the notation $C[P_1, \ldots, P_m]$ means that we substitute the process P_i for the hole $[\ ]_i$ in $C[\ ]$.

Finally, we will indicate the number of holes is a context by placing an (often un-numbered) empty square bracket for each hole in the context. For example, $C[\ ]$ will be a one-hole context while $D[\ ][\ ][\ ]$ has three holes.

Definition 2.25. We will say that B is a *sub-process* of A if there exists a context $C[\ ]$ such that $C[B] \equiv A$. If $C[\ ] \not\equiv [\ ]$ we will say that B is a *strict* sub-process of A. Clearly, if $A \equiv C[B]$ then B is a sub-process of A.

Definition 2.26. Let P and Q be process expressions and $c_{p(|n|)}$ be a channel. We say that P is *bound by* $c_{p(|n|)}$ *in* Q if $c_{p(|n|)}(x).D[P]$ is a sub-process of Q for some variable x and context $D[\ ]$.

Definition 2.27. Let $C[\ ]$ be a context and P a process. Then, $C[\ ]$ is *minimal for* P if every free variable of P is bound in $C[P]$ and each channel to which P is bound in $C[P]$ binds a free variable of P.

Definition 2.28. The set of *schedulable processes* $S(P)$ of an outer-evaluated process P is defined inductively as:

$$\begin{array}{lcl} S(\mathbf{0}) & = & \emptyset \\ S(\nu_{c_{p(|n|)}}.(Q)) & = & S(Q) \\ S(c_{p(|n|)}(x).(Q)) & = & \{c_{p(|n|)}(x).(Q)\} \\ S(c_{p(|n|)}\langle T\rangle) & = & \{c_{p(|n|)}\langle T\rangle\} \\ S((Q_1 \mid Q_2)) & = & S(Q_1) \cup S(Q_2) \end{array}$$

Note that every process in $S(P)$ is either waiting for input or ready to output.

Definition 2.29. The set of *communication triples* $C(P)$ is defined as:

$$\begin{array}{rl} \{\langle P_1, P_2, D_{P_1,P_2}[\ ][\ ]\rangle \mid & P_1 \equiv c_{p(|n|)}\langle a\rangle, P_2 \equiv c_{p(|n|)}(x).R, \\ & P_i \in S(P), \\ & P \equiv D_{P_1,P_2}[P_1, P_2]\} \end{array}$$

Given a process P and a context $D_{P_1,P_2}[\ ][\ ]$, P_1 and P_2 are uniquely determined.

Example 2.30. Consider the process expression:

$$P \equiv c_{p(|n|)}\langle 0\rangle \mid c_{p(|n|)}\langle 0\rangle \mid c_{p(|n|)}(x).d_{q(|n|)}\langle 1\rangle$$

The process expression $c_{p(|n|)}(x).d_{q(|n|)}\langle 1\rangle$ could receive its input from either one of the two output expressions. While both communications look the same, their outputs come from distinct locations in P. We distinguish between the two communications by associating $[\ ]_1 \mid c_{p(|n|)}\langle 0\rangle \mid [\ ]_2$ with the first communication (the one where the output 'came' from the first process in the parallel composition) and associating $c_{p(|n|)}\langle 0\rangle \mid [\ ]_1 \mid [\ ]_2$ with the second communication (the pair where the output 'came' from the second process in the parallel composition). So, $C(P)$ is the set:

$$\Big\{\langle c_{p(|n|)}\langle 0\rangle, c_{p(|n|)}(x).d_{q(|n|)}\langle 1\rangle, [\ ]_1 \mid c_{p(|n|)}\langle 0\rangle \mid [\ ]_2\rangle\ ,$$
$$\langle c_{p(|n|)}\langle 0\rangle, c_{p(|n|)}(x).d_{q(|n|)}\langle 1\rangle, c_{p(|n|)}\langle 0\rangle \mid [\ ]_1 \mid [\ ]_2\rangle\Big\}$$

Definition 2.31. The set of *eligible processes* $E(P)$ is defined as:

$$E(P) = \begin{cases} C(P) & \text{iff every channel in } C(P) \text{ is public,} \\ C(P)|_{\text{private channels}} & \text{otherwise.} \end{cases}$$

Note that restricting $E(P)$ to just private channels anytime $C(P)$ contains a private channel ensures that all private communications occurs prior to public communication. This allows us to "hide" private communications from a prying observer by making private communications privileged.

3 Evaluation

3.1 Scheduler Reduction

Definition 3.1. We write $[a/x]\,P$ to mean "substitute a for all free occurrences of x in P." We will define the *substitution operation* as follows:

$$\begin{array}{lcl}
[a/x]\,\mathbf{0} & = & \mathbf{0} \\
[a/x]\,\nu_{c_{p(|n|)}}.(Q) & = & \nu_{c_{p(|n|)}}.([a/x]\,Q) \\
[a/x]\,c_{p(|n|)}(x).(Q) & = & c_{p(|n|)}(x).(Q) \\
[a/x]\,c_{p(|n|)}(y).(Q) & = & c_{p(|n|)}(y).([a/x]\,Q) \\
[a/x]\,c_{p(|n|)}\langle T\rangle & = & c_{p(|n|)}\langle(\lambda x.T)\ a\rangle \\
[a/x]\,[T_1 = T_2].(Q) & = & [(\lambda x.T_1)\ a = (\lambda x.T_2)\ a].([a/x]\,Q) \\
[a/x]\,(P_1 \mid P_2) & = & ([a/x]\,P_1 \mid [a/x]\,P_2) \\
[a/x]!_{\gamma(|n|)}.(Q) & = & !_{\gamma(|n|)}.([a/x]\,Q)
\end{array}$$

We can generalize the substitution operation to context families, *i.e.,* to substituting a for all free occurrences of x in $C[\ ]$.

Definition 3.2. We define a scheduler S to be a probabilistic poly-time computable function that, given a finite set of communication triples, chooses an element of the given set of triples such that the following two properties hold:

1. For every set of communication triples E, if $S(E,e) > 0$ then $e \in E$; and,
2. $\sum_{e\in E} S(E,e) = 1$.

If $S(E,e) = r$ we will say that the scheduler S picks the communication triple e from the set E with probability r.

The second condition ensures that the scheduler makes *progress* by forcing the scheduler to select a communication triple to evaluate whenever the scheduler is given, as input, a non-empty set of communication triples.

A scheduler runs in time polynomial in the size of its input. While the runtime of a scheduler may not be easily characterized at this stage we will only use the scheduler on sets of communication triples obtained by computing the eligible process set of a process expression. This set, the reader will discover, has size in the square of the length of the process expression which, in turn, is a polynomial in the size of n (the security parameter). Thus, even though we cannot provide a characterization of the run-time of a scheduler (other than that it is polynomial in the size of the finite set of triples given as input), for our purposes, every scheduler will run in time bounded by a polynomial in the size of n.

Definition 3.3. Let a be a natural number, n the value of the security parameter, and S a scheduler. We will say that P reduces to Q in *one communication step with respect to S with probability* r if and only if:

$$\begin{aligned}\exists e \equiv \langle P_1, P_2, D_{P_1,P_2}[\ \ ][\ \ ]\rangle \in E(P) : \; & S(E(P), e) = r, \\ & P_1 \equiv c_{p(|n|)}\langle a\rangle, \\ & P_2 \equiv c_{p(|n|)}(x).P_2', \\ & Q \equiv D_{P_1,P_2}\left[\mathbf{0}, [a'/x]P_2'\right], \\ & a' = a \bmod 2^{p(|n|)} - 1.\end{aligned}$$

We write this as $P \xrightarrow{r}_S Q$.

We reduce a modulo $2^{p(|n|)}-1$ in order to ensure that at most only the $p(|n|)$ least significant bits of a are transmitted.

By stipulating that at most $p(|n|)$ bits of a message can be transmitted we ensure that we cannot create exponential-time processes out of polynomial-time process expressions—something that we can do if we allowed arbitrary length messages to be transmitted. Consider the process expression:

$$P \equiv !_{\gamma(|n|)}.c_{p(|n|)}(x).c_{p(|n|)}\langle x^2\rangle \mid c_{p(|n|)}\langle 2\rangle$$

Let the polynomial $\gamma(|x|)$ return x always. While more complex polynomials will make the results more dramatic, this choice of γ simplifies matters. It is clear that P will square its input (here initialized to 2) n times. We will call the output of this process o.

The table below shows outputs for several different values of n:

n	o	$\lvert o\rvert$
1	$4 = 2^2$	$2 = 2^1$
2	$16 = 2^4$	$4 = 2^2$
3	$256 = 2^8$	$8 = 2^3$
4	$65,536 = 2^{16}$	$16 = 2^4$
5	$4,294,967,296 = 2^{32}$	$32 = 2^5$
$\vdots$	$\vdots$	$\vdots$

Clearly, P outputs values of length exponential in n. Now if the output of P is used as the input to some poly-time process expression Q, then we will obtain an exponential-time process since Q must run on exponentially long values. However, if we truncate the messages transmitted then no message can ever get too long, *i.e.,* exponentially long.

3.2 Evaluation

3.2.1 Macro-Step Reduction

Intuitively, we define the *macro-step reduction* of P to P' based on the following three stage procedure:

1. We outer-evaluate P, yielding R.
2. The scheduler picks a process triple E from $E(R)$ and performs the selected communication. This results in the process expression R'.
3. Finally, we outer-evaluate R', yielding P'.

Thus, macro-step reduction is merely shorthand for a procedure whereby we select a communication to perform and evaluate the result until it becomes necessary for the next communication to be selected. The first outer-evaluation simply ensures that the process expression on which we perform the communication step satisfies the condition on $S(P)$ specified in defn. 2.28. Barring the first outer-evaluation, then, the evaluation of process expressions becomes an alternating series of process evaluations (the outer-evaluations) and communication steps. We will define macro-step reduction so that it is just a communication step followed by a process evaluation—with a preceding process evaluation for the reason mentioned above.

Definition 3.4. More precisely, *macro-step reduction with respect to scheduler S* is the probabilistic function defined by the composition:

$$\rightarrow_o \circ (\rightarrow_S \circ \rightarrow_o) \tag{3.1a}$$

The probability r of the macro-step reduction of P to P' is obtained directly from definition 2.3:

$$r = \sum_{Q,Q' \mid P \xrightarrow{a}_o Q \xrightarrow{b}_S Q' \xrightarrow{c}_o P'} abc \tag{3.1b}$$

We will write the macro-step reduction of P to P' with respect to the scheduler S with probability r as $P \xrightarrow{r}_{1,S} P'$.

3.2.2 m-Step Reduction

Definition 3.5. Let P and P' be two process expressions. Define the set $\mathrm{Paths}_{P,P'}$ as:

$$\{\{Q_1,\ldots,Q_{m-1}\} \mid P \xrightarrow{p_1}_{1,S} Q_1 \xrightarrow{p_2}_{1,S} \cdots \xrightarrow{p_{m-1}}_{1,S} Q_{m-1} \xrightarrow{p_m}_{1,S} P'\}$$

We call $\mathrm{Paths}_{P,P'}$ *the set of paths from P to P'*.

Definition 3.6. The probability that P *reduces to P' in m steps with respect to the scheduler S* is given by:

$$\sum_{\substack{\{Q_1,\cdots,Q_{m-1}\}\in \mathrm{Paths}_{P,P'} \\ P\xrightarrow{p_1}_{1,S}Q_1\xrightarrow{p_2}_{1,S}\cdots\xrightarrow{p_{m-1}}_{1,S}Q_{m-1}\xrightarrow{p_m}_{1,S}P'}} p_1p_2\cdots p_m$$

We write the m-step reduction of P to P' with probability r as $P \xrightarrow{r}_{m,S} P'$.

3.2.3 Evaluation

Definition 3.7. We say that P *evaluates to Q with respect to the scheduler S* if, using the scheduler S, P reduces to Q via some number of macro-steps.

We write the evaluation of P to P' with respect to the scheduler S as $P \hookrightarrow_S P'$.

4 The Complexity of Evaluating Process Expressions

Our bound on evaluation will be derived as a certain combination of three bounds: a bound on the length of any evaluation sequence, a bound on any outer-evaluation step, and a bound on any communication step. We do have a pedagogical consideration to point out, though. In this section we introduce the lemmas on which the main theorem depends after the main theorem has been proved. Since the proof of the lemmas depend on the particular constructions adopted during the proof of the main theorem, we felt that this particular choice would aid both in conciseness and clarity.

Definition 4.1. We have a natural number $\mathcal{L}_T$, the *length* of T, associated with each term T.

If T is a term, $\mathcal{L}_T$ is the product of a constant and the syntactic length of T, *i.e.,* $\mathcal{L}_T = c \cdot \mathit{length}(T)$ where c is a constant. More details about $\mathcal{L}_T$ can be found in the proof of lemma 4.15.

Definition 4.2. Let P be a process expression and n be the value for the security parameter. We define a polynomial $\mathit{normlength}(P)$, the *(normal) length* *of* P, inductively as follows:

$$
\begin{array}{lcl}
\mathit{normlength}(\mathbf{0}) & = & 1 \\
\mathit{normlength}(\nu_{c_{p(|n|)}}.(Q)) & = & c + \mathit{normlength}(Q) \\
\mathit{normlength}(c_{p(|n|)}(x).(Q)) & = & c + \mathit{normlength}(Q) \\
\mathit{normlength}(c_{p(|n|)}\langle T\rangle) & = & c + \mathcal{L}_T \\
\mathit{normlength}([T_1 = T_2].(Q)) & = & c + \mathcal{L}_{T_1} + \mathcal{L}_{T_2} + \mathit{normlength}(Q) \\
\mathit{normlength}((Q_1 \mid Q_2)) & = & c + \mathit{normlength}(Q_1) + \mathit{normlength}(Q_2) \\
\mathit{normlength}(!_{\gamma(|n|)}.(Q)) & = & \gamma(|n|) \cdot c \cdot \mathit{normlength}(Q)
\end{array}
$$

where c is a constant and $\mathcal{L}_T$ is the length of the term T.

Since the length of a term does not depend on n (*i.e.,* it is a constant for our purposes), it is clear from the definition that $\mathit{normlength}(P)$ is polynomial in $|n|$. Furthermore, $\mathit{normlength}(P)$ is positive for all n. This measure of length is an indicator of the intuitive (actual) length of a process expression P.

However, our proof of the time-complexity of process evaluation is made more tractable if we employ another definition of length—the notion of the padded length of P has the useful property that no macro-step will cause the padded length of P (or, indeed, the padded length of any of P's sub-processes) to increase. This property, as the reader will see, is extremely useful as it allows us to perform induction on the padded length of a process expression.

Definition 4.3. Let P be some process. Let $\mathbf{p}$ be the set of bandwidth polynomials that appear in P. Note that $\mathbf{p}$ is a finite (polynomial in $|n|$ big) set. For each $i \in \mathbb{N}$, let $\mathbf{a}$ be the set of the coefficients of the x_i terms in each polynomial in $\mathbf{p}$. Notice that $\mathbf{a}$ is finite since $\mathbf{p}$ is finite. For each $i \in \mathbb{N}$ we define the term a_i as the maximum term in $\mathbf{a}$. Since $\mathbf{a}$ is finite, the greatest coefficient is well-defined. We then define the polynomial $\chi(x)$ as $\sum_{i=0}^{\infty} a_i x^i$. Notice that $\forall b \in \mathbb{N}.\forall p(x) \in \mathbf{p} : \chi(b) \geq p(b)$.

Definition 4.4. Let P be a process expression and n be the value for the security parameter. We define a polynomial $\mathit{length}(P)$, the *padded length of* P, inductively as follows:

$$
\begin{array}{lcl}
\mathit{length}(\mathbf{0}) & = & 1 \\
\mathit{length}(\nu_{c_{p(|n|)}}.(Q)) & = & c + p(|n|) + \mathit{length}(Q) \\
\mathit{length}(c_{p(|n|)}(x).(Q)) & = & c + p(|n|) + \chi(|n|) + \mathit{length}(Q) \\
\mathit{length}(c_{p(|n|)}\langle T\rangle) & = & c_1 + p(|n|) + \mathcal{L}_T + (\mathit{inputs}(P) + 1) \cdot c_2 \cdot \chi(|n|) \\
\mathit{length}([T_1 = T_2].(Q)) & = & c + (d_1 + \mathcal{L}_{T_1} + (\mathit{inputs}(P) + 1) \cdot d_2 \cdot \chi(|n|)) \\
 & & \quad + (e_1 + \mathcal{L}_{T_2} + (\mathit{inputs}(P) + 1) \cdot e_2 \cdot \chi(|n|)) + \\
 & & \quad + \mathit{length}(Q) \\
\mathit{length}((Q_1 \mid Q_2)) & = & c + \mathit{length}(Q_1) + \mathit{length}(Q_2) \\
\mathit{length}(!_{\gamma(|n|)}.(Q)) & = & \gamma(|n|) \cdot c \cdot \mathit{length}(Q)
\end{array}
$$

where c, c_1, c_2, d_1, d_2, e_1, e_2 are constants and $\mathcal{L}_T$ is the length of the term T.

Since the length of a term does not depend on n (*i.e.,* it is a constant for our purposes), it is clear from the definition that $\mathit{length}(P)$ is polynomial in $|n|$. Furthermore, $\mathit{length}(P)$ is positive for all n.

The rationale behind the construction of $length(P)$ should be clear given the discussion of padding on page 19. Some brief notes are in order though. For technical reasons, we assume that a variable has length $\chi(|n|)$ (see defn. 4.3 and lemma 4.15 for details). This allows us to substitute values for variables without pushing symbols around since the variables leave enough space for any conceivable value (as $\chi(|n|)$ exceeds the size of any bandwidth polynomial found in the process expression). Additionally, each term leaves enough blank space for the λ-abstractions created by communication steps (recall that at most $inputs(P)$ such communication steps can occur during evaluation). Finally in adding a $p(|n|)$ cost every time the bandwidth polynomial $p(|n|)$ appears, we make sure to leave enough space to write down a particular value for the bandwidth once a particular value for the security parameter has been chosen.

In what follows, we shall mean by the phrase "length of P" the padded length of P and not the unpadded (normal) length of P unless otherwise explicitly stated.

Recall that P^i signifies that the security parameter has value i in P (defn. 2.15).

Definition 4.5. Let P be a process expression. Consider the reduction of P^n to the normal form Q^n using the scheduler S. Let $\vec{o}$ be a sequence of communication triples on public channels such that the communication triple $e \in \vec{o}$ if and only if e was selected by S during the reduction of P^n. We will say that P^n *produced* $\vec{o}$ *via the reduction to* Q^n.

Definition 4.6. Let M be an oracle Turing machine. We say that M *computes the reduction of* P just when the following two conditions hold:

1. If $M(\vec{x}, \vec{b})$ halts with Q^n and a sequence $\vec{o}$ of communication triples on public channels on its output tape then P^n reduces to normal form Q^n in m steps using the random choices given by $\vec{b}$ (the oracle tape) such that P^n produced $\vec{o}$ during the reduction to Q^n.

2. If P^n reduces to normal form Q^n in m steps with random choices given by a sequence $\vec{r}$ and producing the public communications $\vec{o}$ then $M(\vec{x}, \vec{r})$ halts with Q^n and $\vec{o}$ on its output tape.

Theorem 4.7. *Let P be a process expression, S a scheduler. There exist a poly-time oracle Turing machine $M_{P,S}$ and a polynomial $r(x)$ such that $M_{P,S}$ computes the reduction of P and $M_{P,S}(n, \vec{b})$ halts in time $r(|n|)$ for all $n \in \mathbb{N}$ and $\vec{b} \in \{0,1\}^*$.*

The Turing machine we construct given a process and a scheduler will behave in accordance with the operational semantics we have given for process expressions. That is to say, in order to evaluate a process expression it will first substitute a value for n into the security parameter (which appears in the bandwidth polynomial, replication polynomials and terms), then it will replace all replication operators (expressions of the form $!_{\gamma(|n|)}.Q$) with the equivalent parallel composition (that is $\gamma(|n|)$ copies of Q in parallel). Following this step, it will 'pad' out the expression—this is a technical necessity that will help us establish our time-bound. At this point it will start the actual evaluation. Evaluation proceeds as a series of macro-steps. To perform a macro-step the Turing machine begins by performing an outer-evaluation step (specifically we evaluate all evaluable terms and matches). Whenever a random choice needs to be made (*i.e.*, whenever a coin needs to be flipped) during outer-evaluation, a single bit is read off the oracle tape (which contains the input bit sequence $\vec{b}$). It then computes the set of eligible communication triples (which contain pairs of input expression and output expressions that can communicate with each other). If the set of eligible communication triples is empty it means that evaluation can no longer proceed—we have reached a normal form—and so the machine halts. Otherwise, the machine invokes the scheduler (which uses the oracle tape whenever it needs to make a random choice) to select a communication step after which it performs the indicated communication step. Finally a second outer-evaluation step is performed and the next macro-step is started.

Proof. The machine $M_{P,S}$ is a five-tape Turing machine that starts with a value for n on its input tape, a value for $\vec{b}$ on its oracle tape, the process expression P on its first output tape, an empty second output tape (this tape will record all communication triples on public channels that are scheduled to occur by the scheduler S), and an empty scratch tape. $M_{P,S}$ starts with a preprocessing step, followed by a sequence of macro-steps.

Preprocessing

$M_{P,S}$ continues by substituting the value for n given as an input in the bandwidth polynomials, replication polynomials and terms that appear in P. It does this by copying P onto the scratch tape but substituting the value $q(|\bar{n}|)$ whenever it encounters the polynomial $q(|n|)$ and by creating the λ-substitution instance $(\lambda n.T)\ \bar{n}$ of the term T where $\bar{n}$ is the value to be substituted for the security parameter n. This yields a process P that is ready to evaluate since we now have a value for n. It copies this process back onto the output tape. We note that this procedure is deterministic and takes time polynomial in the size of n—the original expression simply had a place-holder for n; consequently its length was some constant that did not depend on n; after substituting a value for n we derive a process expression whose length does depend on n (recall that definition 4.4 shows that the length of a process expression is, in fact, a polynomial in the size of n). Substituting a value for n takes time polynomial in the size of n (necessitated by the fact that the machine has to write down $c \cdot |n|$ bits—c is a constant that accounts for the λ-substitutions that are created—each time it substitutes a value for n) and copying the expression back onto the output tape takes time polynomial in the size of n.

The machine then needs to rewrite each replication operator as a parallel composition. It does this by copying the process expression P from the output tape onto the scratch tape but replacing each expression $!_{\gamma(|n|).(Q)}$ with $\gamma(|n|)$ copies of Q in parallel. Again this procedure is not probabilistic. Furthermore it takes time polynomial in the size of n—an examination of the definition of the padded length of a process expression shows that the replication operator $!_{\gamma(|n|)}.(Q)$ has length $\gamma(|n|) \cdot c \cdot \mathit{length}(Q)$ which accounts for the expansion in length caused by rewriting the replication operator as a parallel composition. Hence, the length of the process expression obtained be rewriting all replication operators is still a polynomial in the size of n—in fact, its (padded) length remains the same as before the replications were rewritten.

The next step is to "pad" out the length of the process expression. By doing this we hope to leave enough blank space around every term and variable so that performing communication steps will not increase the (padded) length of the process expression.

Define $\chi(x)$ as per defn. 4.3. $M_{P,S}$ starts padding by initially rewriting each variable x in P so that it takes up $\chi(|n|)$ spaces on the input tape. It does so by inserting $\chi(|n|) - 1$ blank symbols before the variable so that x becomes the string $\triangle \cdots \triangle\ x$. Since the length of P is a polynomial in the size of n there are at most a polynomial in the size of n variables each of which has a polynomial in the size of n number of blank symbols inserted before it. Thus, it is easy to see that padding out the variables of P can be done by the $M_{P,S}$ in at most a polynomial in the size of n number of steps. What this padding does is to ensure that substituting a value for x can be done without increasing the length of a padded process P (since $\chi(|n|)$ exceeds the size of any bandwidth polynomial and since we only need to write down at most a bandwidth polynomial number of bits when substituting a value for a variable, we will always have plenty of space).

During the evaluation of P there can be at most $\mathit{inputs}(P)$ communication steps (lemma 4.12). Let T be an arbitrary term in P. Each communication step during the evaluation of P can hide T behind one level of λ-abstraction (as pg. 21 will show, when evaluating a communication step if $M_{P,S}$ has to substitute a value a for a variable x into the term T, $M_{P,S}$ creates the λ-substitution instance $(\lambda x.T)\ a$). So, at the end of evaluation, in the worst case, T will be hidden behind $\mathit{inputs}(P) + 1$ λ-abstractions (the additive term accounts for the extra λ-abstraction created in order to provide a value for n to the term). The increase in the length of the term T caused by a single λ-abstraction is a polynomial in the size of n—T becomes $(\lambda x.T)\ a$. The symbols added to T are 'λ', 'x', '.', '(', ')' and 'a' (recall that x and a are considered to be $\chi(|n|)$ bits long). Thus, if we add $(\mathit{inputs}(P) + 1) \cdot c \cdot \chi(|n|)$ blank spaces before each term in P, we will have enough space around each term to ensure that we can account for the maximal increase in the size of terms incurred during evaluation due to the creation of λ-abstractions during communication steps. Again, this padding can be done in a polynomial in the size of n amount of time since there are at most a polynomial in the size of n number of terms and $\mathit{inputs}(P)$ is a polynomial in the size of n.

Hence, padding P can be done in a polynomial in the size of n amount of time by a Turing machine. Furthermore, the length of P after padding is precisely that given by the definition of $\mathit{length}(P)$—the padded length of P.

At this point, in time polynomial in the size of n, $M_{P,S}$ has performed a deterministic procedure that has substituted a value for the security parameter, rewritten replication operators as parallel compositions and padded out the process in anticipation for evaluation.

$M_{P,S}$ evaluates P by performing a series of macro-steps. It is here that the probabilistic nature of process evaluation comes to the fore. In order to perform a macro-step $M_{P,S}$ has to perform an outer-evaluation, and then a communication step, and finally another macro-step. Let us assume that we are trying to perform the ith macro-step and let us assume that the expression R is on the output tape.

First outer-evaluation step

To perform the initial outer-evaluation, $M_{P,S}$ performs two passes. In the first pass $M_{P,S}$ evaluates as many exposed output terms as it can (detecting an exposed term is easy—when $M_{P,S}$ reaches an input expression, it ignores every syntactic element of Q until it reaches the end of the scope of the input expression; otherwise, every term is exposed). Each term it hits will be an expression of the form $(\lambda x_1 \ldots (\lambda x_m.(\lambda n.T)\ a_{m+1}) \ldots a_1)$—the λ binding the variable n provides a value to the security parameter. $M_{P,S}$ then invokes the machine M_T (defined with respect to the term T in theorem 2.10) that evaluates the term T with the arguments $a_1, \ldots, a_{m+1}$. Since $M_{P,S}$ was built based on P, if P has terms T_1 through T_k, we can hardwire into $M_{P,S}$ the k oracle machines that evaluate the k terms that appear in P—whenever the oracle machine that evaluates the term T requires a random bit, the machine $M_{P,S}$ provides that random bit by reading off the oracle tape (which contains the bit sequence $\vec{b}$). Now, when the value to which T reduces is transmitted, it will be truncated according to the bandwidth polynomial. So $M_{P,S}$ writes down only the $q(|n|)$ least significant bits of the value returned by M_T where $q(|n|)$ is the bandwidth polynomial associated with the channel on which T is transmitted. We note that $M_T(a_1, \ldots, a_{m+1})$ reduces to the atom a using coin-flips given by $\vec{r}$ exactly when T reduces to a using coin-flips given by $\vec{r}$ since, by definition, M_T is the denotation of T (page 5).

The second pass evaluates all exposed matches. Whenever $M_{P,S}$ hits an exposed match of the form:

$$[(\lambda x_1 \ldots (\lambda x_l.(\lambda n.T_1)\ a_{l+1}) \ldots a_1) = (\lambda y_1 \ldots (\lambda y_m.(\lambda n.T)\ b_{m+1}) \ldots b_1)].S$$

it evaluates T_1 and T_2 using the machines M_{T_1} and M_{T_2} in exactly the same way that terms found in outputs were evaluated. This takes time polynomial in the size of n. It then writes down the values obtained in full on the scratch tape. Let us call the value that M_{T_1} produces a_1 (this is the value to which T_1 reduces) and let us call the value that M_{T_2} produces a_2 (this is the value to which T_2 reduces). We note that since M_{T_1} and M_{T_2} halt in time polynomial in the size of n, the lengths of a_1 and a_2 must, in the worst case, be polynomial in the size of n.

By lemma 4.19 we know that each outer-evaluation done during the evaluation of P takes time at most a polynomial in the size of n and, more importantly, we know that there exists a polynomial $\kappa_o(x)$ such that each outer-evaluation that occurs during the evaluation of P takes time at most $\kappa_o(|n|)$. $M_{P,S}$ thus obtains an output expression R'.

Communication step

$M_{P,S}$ will then perform a communication step on R'. To do so, it starts by computing the set of schedulable processes $S(R')$. It does this by making a single pass over R' and writing down on the scratch tape every input or output expression that is ready to communicate—see defn. 2.28 for details. It then computes the set of eligible processes (defn. 2.31) by creating from the set of schedulable processes a set of communication triples (defn. 2.29), each of which identifies an input/output pair that is ready to communicate. This (deterministic) procedure takes time $|S(R')|^2$; we note that $|S(R')|$ is a polynomial in the size of n since the length of R is polynomial in the size of n. If the set of eligible communication triples is empty, $M_{P,S}$ halts with R' on the output tape.

Otherwise, $M_{P,S}$ selects a communication π to perform using the scheduler S—since $M_{P,S}$ was built knowing S we can ensure that $M_{P,S}$ contains the polynomial time oracle Turing machine M_S that computes the scheduler S (M_S must exist since, by definition, S is a probabilistic, poly-time computable function). Furthermore, we know that a communication π will be selected (*i.e.*, the scheduler will not 'freeze' when deciding which communication triple to perform) since schedulers, by definition, must with probability one select some communication to perform at each stage (this is the progress condition on schedulers). Again, whenever the oracle machine M_S requires a random bit, $M_{P,S}$ provides that bit by reading a value off the oracle tape. Finally it performs the communication step indicated by S thereby obtaining the expression R''. By lemma 4.14 we know that $length(R'') \leq length(P)$. Thus, selecting the communication to perform at the ith macro-step can be bounded by $c \cdot length(P) + |d \cdot length(P)|^2$—the term $c \cdot length(P)$ accounts for the multiple passes required to construct the set of eligible communication triples and the $|d \cdot length(P)|^2$ term accounts for the time needed for the scheduler to select a communication triple.

$M_{P,S}$ then checks if the selected communication is a private or public communication. This can be done by making a single pass over the process expression and checking if the channel on which the communication occurs is bound to any ν operator. Consequently, this takes time polynomial in the size of n (since the padded length of the process expression is polynomial in the size of n). If the communication is public, it records the appropriate communication triple on the second output tape. This takes time polynomial in the size of n since i, the value communicated by the communication triple, must be polynomial in the size of n (i is truncated by the bandwidth of c, the channel on which the communication occurs). Hence there exists a single polynomial $\kappa_s(|n|)$ such that the time required to select the communication to perform in the ith macro-step and record the associated communication triple on the second output tape is bounded by $\kappa_s(|n|)$.

In order to perform the communication step, $M_{P,S}$ makes one pass to remove the input and output specified in π. Specifically, it rewrites the sub-process

$c_{p(|n|)}(x).Q$ specified as the input in π with $\triangle \cdots \triangle Q$ (where $\triangle$ is a special character that $M_{P,S}$ ignores—this allows $M_{P,S}$ to delete sub-processes of P without having to push symbols around) and the sub-process $c_{p(|n|)}\langle a\rangle$ specified as the output in π with $\triangle \cdots \triangle\mathbf{0}$.

$M_{P,S}$ then makes another pass over the process replacing all instances of x with a. Since we assume that all channels are α-renamed apart and that both a and x are at most $\chi(|n|)$ bits long, $M_{P,S}$ can do this substitution without any real effort except when $M_{P,S}$ creates a λ-substitution instance out of a term T. We create λ-substitution instances out of terms in order to delay the actual substitution of values for variables into terms until the moment of evaluation; since the length of terms does not depend on $|n|$ we cannot bound the expansion caused by substituting values for variables unless we delay substitution through the mechanism of λ-abstractions.

So when $M_{P,S}$ creates a λ-substitution instance of a term, it needs to insert symbols. However, each communication step adds one level of λ-abstraction; so, when $M_{P,S}$ pads out the process expression it leaves sufficient room around each term T, *i.e.,* $M_{P,S}$ adds some fixed amount of blank space after each term. Since the notation for a single λ-abstraction is constant sized and the value to apply to that λ-abstraction is at most $\chi(|n|)$ bits long, the same amount of space per term into which a value for x is to be substituted will suffice. $M_{P,S}$ added this extra space to each term during the initial pass that padded out P. Essentially, $M_{P,S}$ is adding a polynomial in $|n|$ amount of padding to the process—the length of the expression is a polynomial in $|n|$ and an upper-bound on the number of terms in the process.

Then, we can easily perform the rewriting necessitated by the communication step without moving symbols around. Thus, two passes over the process should suffice to perform the communication step thereby deriving the expression R''.

By lemma 4.23 we know that there exists a polynomial $\kappa_c(x)$ such that any communication step that occurs during the evaluation of $\mathcal{P}$ can be performed in time bounded by $\kappa_c(|n|)$.

Second outer-evaluation step

Then $M_{P,S}$ performs a second outer-evaluation step which, the reader will recall, takes at most $\kappa_o(|n|)$ time and writes the process expression R''' on the output tape. $M_{P,S}$ then performs the next macro-step.

So when $M_{P,S}$ finally halts it will do so with a (padded) normal form Q^n on its output tape and a sequence of communication triples $\vec{\sigma}$ on the second output tape such that $\vec{\sigma}$ consists of exactly those public communications generated during the evaluation of P using random choices given by $\vec{b}$.

We note that $M_{P,S}(n, \vec{b})$ halts with normal form Q^n on its output tape making random choices indicated by $\vec{b}$ if and only if P^n reduces, by making random choices indicated by $\vec{b}$, to normal form Q^n—this follows from the fact that term-evaluation and scheduler selection are both probabilistic poly-time computable. Similarly, we note that the sequence of communication triples $\vec{\sigma}$ is generated by $M_{P,S}(n, \vec{b})$ using random choices $\vec{b}$ if and only if the reduction of

P^n to Q^n using random choices $\vec{b}$ produces $\vec{o}$.

How long does this take? In the worst case $M_{P,S}$ will run for t macro-steps. Each step takes time $2.\kappa_o(|n|) + \kappa_s(|n|) + \kappa_c(|n|)$ for the two outer-evaluations, the selection of a communication step and the actual substitution indicated by the communication step. Additionally, there is a one-time cost of $c \cdot length(P)$ for the initial preprocessing (substituting a value for n and padding). Thus, process evaluation takes time $c \cdot length(P) + t(2.\kappa_o(|n|) + \kappa_s(|n|) + \kappa_c(|n|))$. Now, by lemma 4.12, we know that in the worst case $inputs(P)$ macro-steps will occur. Since $length(P)$ and $inputs(P)$ are both polynomials in the size of n we can bound evaluation time by a polynomial in the size of n. □

Corollary 4.8. *Let P be an evaluable process expression and S a scheduler. Define a probabilistic function $evals_{P,S}$ such that $evals_{P,S}(n, Q) = p$ if and only if P^n reduces to normal form Q^n with probability p. Then $evals_{P,S}$ is probabilistic poly-time (in the size of the security parameter n) computable.*

Proof. Since we have, by theorem 4.7, a probabilistic poly-time Turing machine that can reduce P to a normal form, the result follows. □

So, loosely speaking, process evaluation can be done in probabilistic polynomial (in the size of the security parameter n) time. We speak loosely here because process evaluation is non-uniform (since term evaluation is non-uniform).

Remark 4.9. Process evaluation is non-uniform (meaning that we cannot construct one machine that can evaluate any process expression) because term evaluation in non-uniform. Since the outer-evaluation step requires that terms be evaluated, the machine we construct to evaluate a particular process expression depends on the terms that appear in that expression; consequently, different process expressions will require different machines to evaluate depending on the terms that appear in the process expression being evaluated. We could generalize theorem 4.7 so that the machine constructed depends not on the process expression but on a set of terms—given terms $T_1, \ldots, T_m$ and a scheduler S we could construct a machine $M_{T_1,\ldots,T_m,S}$ that would evaluate any process expression whose terms were members of the set $\{T_1, \ldots, T_m\}$; consequently a particular machine $M_{T_1,\ldots,T_m,S}$ could evaluate many different process expression. However, as we shall see, the machine constructed in theorem 4.7 can be extended naturally to the definition of process families (defn. 5.3) and so we leave the details of the alternate construction to the interested reader.

4.1 A Bound on the Length of any Evaluation Sequence

Definition 4.10. Let P be a process expression. We define $\mathit{inputs}(P)$, the *number of inputs in* P, inductively as follows:

$$\begin{array}{lcl} \mathit{inputs}(\mathbf{0}) & = & 0 \\ \mathit{inputs}(\nu_{c_{p(|n|)}}.(Q)) & = & \mathit{inputs}(Q) \\ \mathit{inputs}(c_{p(|n|)}(x).(Q)) & = & 1 + \mathit{inputs}(Q) \\ \mathit{inputs}(c_{p(|n|)}\langle T \rangle) & = & 0 \\ \mathit{inputs}([T_1 = T_2].(Q)) & = & \mathit{inputs}(Q) \\ \mathit{inputs}((Q_1 \mid Q_2)) & = & \mathit{inputs}(Q_1) + \mathit{inputs}(Q_2) \\ \mathit{inputs}(!_{\gamma(|n|)}.(Q)) & = & \gamma(|n|) \cdot \mathit{inputs}(Q) \end{array}$$

It is clear from the definition that $\mathit{inputs}(P)$ is a polynomial in $|n|$ that is non-negative for all input values.

Definition 4.11. Let P be a process expression. We define $\mathit{outputs}(P)$, the *number of outputs in* P, inductively as follows:

$$\begin{array}{lcl} \mathit{outputs}(\mathbf{0}) & = & 0 \\ \mathit{outputs}(\nu_{c_{p(|n|)}}.(Q)) & = & \mathit{outputs}(Q) \\ \mathit{outputs}(c_{p(|n|)}(x).(Q)) & = & \mathit{outputs}(Q) \\ \mathit{outputs}(c_{p(|n|)}\langle T \rangle) & = & 1 \\ \mathit{outputs}([T_1 = T_2].(Q)) & = & \mathit{outputs}(Q) \\ \mathit{outputs}((Q_1 \mid Q_2)) & = & \mathit{outputs}(Q_1) + \mathit{outputs}(Q_2) \\ \mathit{outputs}(!_{\gamma(|n|)}.(Q)) & = & \gamma(|n|) \cdot \mathit{outputs}(Q) \end{array}$$

It is clear from the definition that $\mathit{outputs}(P)$ is a polynomial in $|n|$ that is non-negative for all input values.

The following lemma is straightforward.

Lemma 4.12. *Let P be a closed process. Then, for all n and schedulers S, during any evaluation of P, at most $\mathit{inputs}(P)$ communication steps occur.*

Proof. Consider any macro-step reduction during the evaluation of P to Q. A macro-step reduction (defn. 3.4) is made up of an $\mathit{outerEval}$, followed by a communication step, followed by another $\mathit{outerEval}$. From the definition of $\mathit{outerEval}$ (defn. 2.20) we know that the $\mathit{outerEval}$ steps can only reduce the number of inputs or outputs in the expression being evaluated. The communication step (defn. 3.3) syntactically eliminates a send/receive pair from the expression being evaluated. This strictly reduces the number of inputs. Communication stops when (or before) the number of inputs is zero. Therefore at most $\mathit{inputs}(P)$ communication steps occur during an evaluation of P. □

Corollary 4.13. *Let P be a closed process. Then, for all n and schedulers S, during any evaluation of P, at most $\mathit{inputs}(P)$ macro-steps occur.*

Proof. From lemma 4.12 and the fact that each macro-step reduction contains exactly one communication step (defn. 3.4). □

Let us observe that the length does not increase as a result of macro-steps.

Lemma 4.14. *Let P be a closed process and R a process such that R is the result of performing some number of macro-steps on P. Then for all n, $length(R) \leq length(P)$.*

Proof. Each macro-step consists of two outer-evaluation steps and a communication step. The outer-evaluation step replaces terms in outputs with atoms and matches with either **0**s or the expressions bound to the matches. However we note that only $\chi(|n|)$ bits of the value to which a term evaluate are written down in an output (a space cost accounted for by padding out the process expression). Additionally, outer-evaluations of matches replace sub-expressions with shorter sub-expressions—$M_{P,S}$ never keeps the values to which the terms in a match evaluate; rather, $M_{P,S}$ simply checks whether the values of the two terms agree or disagree.

Each communication step removes an input from P and replaces an output with a **0**. Furthermore, the contribution to the length of P by a particular term T in P accounts for the maximum size to which T can grow by the creation of successive λ-substitution instances of T due to a succession of communication steps.

Hence, $length(R)$ cannot exceed $length(P)$. □

4.2 A Bound on Outer-Evaluation Time

We now show that any outer-evaluation step may be carried out on a polynomial-time oracle Turing machine. We rely on the assignment of polynomial-time oracle Turing machines M_T to terms T discussed in section 2.3.1.

Lemma 4.15. *Let $\{T_1, \ldots, T_m\}$ be a multiset of m terms and let $M_{T_1}, \ldots, M_{T_m}$ be their associated oracle polynomial-time Turing machines. Let P be a closed process containing only terms of the form $(\lambda x_1 \cdots \lambda x_r.\lambda n.T)\ a_1 \cdots a_{r+1}$, where T is a term in the multiset $\{T_1, \ldots, T_m\}$. Then, the oracle machine $M_{P,S}$ outer-evaluates P in time polynomial in $|n|$.*

Proof. By insisting that $\{T_1, \ldots, T_m\}$ be a multiset we can simplify our analysis by considering every term in P to be distinct even if two are syntactically identical (*i.e.*, the same term appears twice in P).

From theorem 4.7, we know how outer-evaluation is done. Outer-evaluation consists of two passes: during the first pass we evaluate each exposed term of the form $(\lambda x_1 \ldots \lambda x_m.\lambda n.T)\ a_{m+1} \ldots a_1$ that is in an output whose bandwidth is $q(|n|)$ by evaluating $M_T(a_1, \ldots, a_{m+1})$ and writing down the $q(|n|)$ least significant bits; in the second pass we evaluate each exposed match by evaluating the terms $(\lambda y_1 \ldots \lambda y_l.\lambda n.T_1)\ a_{l+1} \ldots a_1$ and $(\lambda y_1 \ldots \lambda y_m.\lambda n.T_2)\ a_{m+1} \ldots a_1$ in the match as in the previous pass but writing down the values obtained by evaluating T_1 and T_2 in full on the scratch tape and then replacing the match with a **0** or the expression bound to the match depending on whether the two values obtained correspond or not.

How much does this cost in the sense of time complexity? $M_{P,S}$ "touches" each syntactic element of the process expression P. Whenever it hits a term $(\lambda x_1 \cdots \lambda x_i \lambda n.T)\ a_{i+1} \cdots a_1$, it evaluates T by evaluating the algorithm M_T at $a_1, \ldots, a_{i+1}$. This takes a polynomial $q_T(|a_1|, \ldots, |a_i|, |a_{i+1}|)$ time. So, the cost of outer-evaluating P is a function of both the length of P and the costs of evaluating each exposed term that the $M_{P,S}$ encounters. However, when evaluating matches, in addition to the cost incurred by evaluating the two terms in the match there is an additional cost incurred for comparing the two resultant atoms. Assume that evaluating the first term in the match took time $q_1(|a_1|, \ldots, |a_l|, |n|)$ and the second term took time $q_2(|b_1|, \ldots, |b_m|, |n|)$. Then the length of the atom obtained by evaluating the first term is at most $q_1(|a_1|, \ldots, |a_l|, |n|)$ and the length of the atom obtained by evaluating the second term is at most $q_2(|b_1|, \ldots, |b_m|, |n|)$ whence we can account for the extra time needed to evaluate the match by adding a multiplicative constant to the cost incurred for just evaluating the terms.

The length of P is given inductively in defn. 4.4 as $\mathit{length}(P)$. Assuming that there are terms $U_1, \ldots, U_s$ in P out of which the first i are exposed, the cost of outer-evaluating P is:

$$t(P) = c \cdot \mathit{length}(P) + \sum_{j=1}^{i} \left(\mathcal{L}_{U_j} + c_j \cdot q_{T_j}(|a_1|, \ldots, |a_{\alpha(U_j)}|, |n|) \right) \qquad (4.1a)$$

where $U_x \equiv (\lambda x_1 \cdots \lambda x_b . \lambda n.T_x)\ a_{b+1} \cdots a_1$, $\alpha(U_x) = b$ and c_j is a multiplicative constant used to account for the cost of evaluating a match. The $\mathcal{L}_{U_x}$ term accounts for the cost of actually determining the $\alpha(U_x)$ arguments to T_x. Finally, the term $c \cdot \mathit{length}(P)$ accounts for the cost of parsing a padded P to determine whether a term or an expression is exposed or not (note that the expansion in length caused by padding P is accounted for in the definition of $\mathit{length}(P)$).

Now q_{T_j} is a polynomial of degree d in $\alpha(U_j)+1$ variables (the $\alpha(U_j)$ inputs to the term and the security parameter) *i.e.,*

$$q_{T_j} = c_1 {a_1}^{d_1^1} \cdots {a_{\alpha(U_j)+1}}^{d_1^{\alpha(U_j)+1}} + \cdots + c_k {a_1}^{d_k^1} \cdots {a_{\alpha(U_j)+1}}^{d_k^{\alpha(U_j)+1}}$$

where $d_i^j \in \{0, \ldots, d\}$ and $c_i \in \mathbb{Z}$. Clearly

$$q_{T_j} \leq q'_{T_j} = |c_1| {a_1}^{d_1^1} \cdots {a_{\alpha(U_j)+1}}^{d_1^{\alpha(U_j)+1}} + \cdots + |c_k| {a_1}^{d_k^1} \cdots {a_{\alpha(U_j)+1}}^{d_k^{\alpha(U_j)+1}}$$

since the values substituted for the $\alpha(U_j)+1$ variables are all in $\mathbb{N}$ and, hence, are all non-negative. We observe that any argument to T_j is at most $\chi(|n|)$ bits long since the value for the security parameter is, by definition, of length $|n|$ and every other variable gets a value via the evaluation of a communication step. So,

$$q_{T_j}(|a_1|, \ldots, |a_{\alpha(U_j)}|, |n|) \leq q'_{T_j}(\overbrace{\chi(|n|), \ldots, \chi(|n|)}^{\alpha(U_j) \text{ times}}, |n|)$$

But, q'_{T_j} then becomes just a function of $\chi(|n|)$ and $|n|$. Since $\mathcal{L}_{U_j}$ does not depend on n, we obtain that $t(P) \leq \tau(P)$ where:

$$\tau(P) = c \cdot length(P) + \sum_{j=1}^{i} \Big(q'_{U_j}(\chi(|n|), |n|) \Big) \tag{4.1b}$$

where $q'_{U_j}(\chi(|n|), |n|) = \mathcal{L}_{U_j} + c_j \cdot q'_{T_j}(\overbrace{\chi(|n|), \ldots, \chi(|n|)}^{\alpha(U_j) \text{ times}}, |n|)$.

Thus, clearly $\tau(P)$ is a polynomial in $|n|$ (as $length(P)$ is polynomial in $|n|$) that bounds the amount of time the pTM takes to outer-evaluate P. $\square$

Since $M_{P,S}$ is an oracle Turing machine that outer-evaluates P for all values of n and was constructed from the oracle Turing machines $M_{T_1}, \ldots, M_{T_m}$, we will say that $M_{P,S}$ is *uniformly constructed* from the pTMs $M_{T_1}, \ldots, M_{T_m}$.

Definition 4.16. We define $\kappa(P)$, the cost of outer-evaluating a closed process expression P, as:

$$\kappa(P) = c \cdot length(P) + \sum_{j=1}^{s} \Big(q'_{T_j}(\chi(|n|), |n|) \Big)$$

where the process P has the terms $T_1, \ldots, T_s$ and q'_{T_j} is defined as for equation (4.1b) and c is a constant that accounts for the cost of parsing a padded P to determine exposed expressions and terms.

Whereas $\tau(P)$ is the time needed to outer-evaluate just the exposed terms and matches in P, $\kappa(P)$ is the cost of outer-evaluating every term and match in P. The following lemma is immediate.

Lemma 4.17. *Let P be a closed process. Then, $\tau(P) \leq \kappa(P)$.*

Proof. Directly from the fact that $\kappa(P)$ bounds the cost of evaluating every term in P and $\tau(P)$ bounds the cost of evaluating just the exposed terms in P. $\square$

It may also be shown that macro-steps do not increase the cost of outer-evaluation.

Lemma 4.18. *Let P be a closed process expression and S a scheduler. Let R be a process such that R is obtained by performing a finite number of macro-steps on P. Then for all n and k, $\kappa(R) \leq \kappa(P)$.*

Proof. A macro-step reduction from P to R is an outer-evaluation followed by a communication step followed by an outer-evaluation. The outer-evaluation steps only remove sub-expressions of P. Specifically, exposed matches in P are removed (thereby eliminating additive polynomials) and exposed terms in P are reduced to atoms (which replace polynomial costs with unit time evaluation costs).

The communication steps replace sub-processes of the form $c_{p(|n|)}\langle a\rangle$ with $\mathbf{0}$. However, we pad the $\mathbf{0}$ with blank symbols so that it takes up as much space on the tape as the $c_{p(|n|)}\langle a\rangle$ it replaced—this allows us to avoid paying costs for contracting the entire process expression when we replace the (comparatively) lengthy expression $c_{p(|n|)}\langle a\rangle$ with the short expression $\mathbf{0}$. Thus, we replace a polynomial cost sub-process ($c_{p(|n|)}\langle a\rangle$) with a sub-process ($\mathbf{0}$) that costs at most as much as $c_{p(|n|)}\langle a\rangle$ did (due to the padding with blank symbols). We also replace sub-processes of the form $c_{p(|n|)}(x).Q$ with $[a/x]\,Q$ (we replace the fragment $c_{p(|n|)}(x).$ with blank symbols thereby ensuring that the process length does not increase and so ensuring that the cost remains at most the same—recall that we account for the expansion in Q by adding padding to the variable names so that sufficient space is left for writing down a everywhere x occurs in Q). Additionally, communication steps replace terms T with $T' \equiv (\lambda x.T)\ a$. Consequently, each term T in R may, in the worst case, be of the form $(\lambda x_1 \cdots \lambda x_{inputs(P)}.\lambda n.T')\ a_{inputs(P)+1} \cdots a_1$ where T' is a term in P, *i.e.,* terms in R are λ-substitution instances of terms in P. Hence the cost of evaluating T' is the cost of evaluating a substitution instance of T'—the $inputs(P) + 1$ λ-abstractions cause an increase in the length of terms that we account for by padding P. Thus, the contribution of T to $\kappa(R)$ in no greater than the contribution of T' to $\kappa(P)$. $\square$

Thus we also obtain

Lemma 4.19. *Let P be a closed process. Let R be a process such that P evaluates to R in at most m macro-steps. Then $\tau(R) \leq \kappa(P)$.*

Proof. Directly from lemmas 4.18 and 4.17. $\square$

4.3 A Bound on the Communication Step

Lemma 4.20. *Let P be a closed process and π a communication triple. Then if $M_{P,S}$ performs the communication step indicated by π on P, it does so in time polynomial in $|n|$.*

Proof. We know that P is padded. This allows us to assume, as we did in the proof of lemma 4.15, that variables are $\chi(|n|)$ bits long. This means that we can easily substitute values for variables by simply overwriting the variable.

In order to perform the communication step, we make one pass to remove the input and output specified in π. We then make another pass over the process replacing all instances of x with a except in the case of terms where we create a λ-substitution instance of the term. Since we assume that all channels are α-renamed apart and that both a and x are at most $\chi(|n|)$ bits long, we can do this substitution without any real effort.

So, the time needed to perform the communication step is just $\sigma(P) = c_0 \cdot length(P)$ where c_0 is a constant that accounts for the multiple passes (one to eliminate the input and output expressions and another to perform the substitutions indicated by the communication step) and $length(P)$ accounts for the increased length caused by padding out P. Clearly, $\sigma(P)$ is a polynomial in $|n|$ since $\mathit{length}(P)$ is such a polynomial. $\square$

Definition 4.21. Let P be a process expression. We notice that if there are $\mathit{inputs}(P)$ inputs and $\mathit{outputs}(P)$ outputs in P, then there are at most $\mathit{inputs}(P) \cdot \mathit{outputs}(P)$ communication steps possible. Let c_i be the constant factor that accounts for the multiple passes incurred during the evaluation of the ith communication step (meaning, the ith communication out of the set of $\mathit{inputs}(P) \cdot \mathit{outputs}(P)$ communication steps; not the communication taken at the ith evaluation step).

We define $\varphi(P)$, the cost of performing a substitution given by a communication triple π on the closed process expression P, as:

$$\varphi(P) = c_0 \cdot \mathit{length}(P) + c_1 \cdot \mathit{length}(P) + \cdots + c_k \cdot \mathit{length}(P)$$

where $k = \mathit{inputs}(P) \cdot \mathit{outputs}(P)$.

$\sigma(P)$ is the time needed to perform the substitution specified by π. However, $\varphi(P)$ is meant to bound the cost of any substitution that might possibly occur in the course of evaluating P. A process has $\mathit{inputs}(P)$ inputs and $\mathit{outputs}(P)$ outputs. Since an arbitrary substitution consists of one input expression and one output expression, there are at most $\mathit{inputs}(P) \cdot \mathit{outputs}(P)$ possible communications that can occur. At any communication step (by lemma 4.13, there are at most $\mathit{inputs}(P)$ such steps), only one potential communication occurs. Hence φ is an upper bound on the time needed to perform any communication that might occur in the evaluation of P.

It is clear that

Lemma 4.22. *Let P be a closed process. Then, $\sigma(P) \leq \varphi(P)$.*

Proof. Directly from the definitions of $\sigma(P)$ and $\varphi(P)$. $\square$

It follows from lemmas 4.14 and 4.22 that

Lemma 4.23. *Let P be a closed process expression and S a scheduler. Let R be a process such that R is obtained by a finite number of macro-steps from P. Then for all n, $\sigma(R) \leq \varphi(P)$.*

Proof. It is easy to see that $\varphi(R) \leq \varphi(P)$. During the evaluation of P, each macro-step reduction removes one input and one output from the process being reduced. Hence, $\mathit{inputs}(R) \leq \mathit{inputs}(P)$ and $\mathit{outputs}(R) \leq \mathit{outputs}(P)$. Furthermore, the length of P is defined so that macro-steps cannot increase the length of P (lemma 4.14). We use lemma 4.22 to complete the proof. $\square$

5 Equivalence

Definition 5.1. We define an *observation* to be a test on a particular public channel for a particular natural number. More precisely, we will define *Obs* to be the set of all possible observations, *i.e.,* the set of pairs $\langle i, c_{p(|n|)}\rangle$ where $i \in [0..2^{p(|n|)}-1]$ is a natural number and $c_{p(|n|)}$ is a public channel. If, during an evaluation of process expression P using the scheduler S, the scheduler selects the communication triple $\langle c_{p(|n|)}\langle i\rangle, c_{p(|n|)}(x).P', C[\ ][\ ]\rangle$, with $c_{p(|n|)}$ a public channel, we will say that the *observable* $\langle i, c_{p(|n|)}\rangle \in \mathit{Obs}$ was observed and write $P \leadsto_S \langle i, c_{p(|n|)}\rangle$.

Definition 5.2. Let Δ be the set of expressions generated by the following sub-grammar of the one which produces contexts (see defn. 2.24):

$$\begin{array}{rcl} C[\ ] & ::= & [\ ]_i \\ & & c_{p(|n|)}(x).(C[\ ]) \\ & & [T = T].(C[\ ]) \\ & & (C[\ ] \mid P) \\ & & (P \mid C[\ ]) \\ & & !_{\gamma(|n|)}.(C[\ ]) \end{array}$$

where $i \in \mathbb{N}$.

An *adversarial context* is a process expression that is a member of Δ—Δ is just the subset of the set of contexts (Γ) that are single-hole contexts where the hole doesn't appear in the scope of a ν-operator. As for a standard context, the holes are numbered uniquely (an easy enough task since there is only one hole).

We wish to be able to say that two process expressions are equivalent if and only if they behave in the same way in the presence of any adversary. Formally speaking, we will identify the space of adversaries with a subset of the space of contexts; namely, the adversarial contexts. Adversarial contexts are not allowed to bind the component plugged into the "hole" to a ν-operator. One might argue that preventing adversaries from having this this ability constitutes a (perhaps unreasonable) weakening of our threat model. The issue at stake here is how precisely to model in our calculus the intuitive distinction between remote and local computation. In particular, we wish to consider an adversary in light of the notion of network security. Consequently, giving the adversary the power to alter the flow of protocol participants' private computations seems unreasonable—yet this is precisely what an adversary can do if allowed to bind public channels of a protocol to ν-operators. Equivalently, in network security, an attacker who sabotages your server by introducing a virus into your OS is not considered an appropriate threat. Our choice of threat model depends on certain complex modelling issues that are beyond the scope of this paper

Definition 5.3. A *process family* $\mathcal{P}$ is the set $\langle P^i \mid i \in \mathbb{N}\rangle$ (recall that P^i signifies that the security parameter n is assigned the value i in the expression P). Since contexts may contain the process parameter n, we can define the *context family* $\mathcal{C}[\ ]$ and the *adversarial context family* $\mathcal{C}[\ ]$ analogously.

We evaluate a process family by picking a value for n and then evaluating the indicated member of the process family. In this manner we can extend all the concepts regarding process expressions and contexts to process and context families. Furthermore, it should be easy to see that the machine $M_{P,S}$ (constructed in theorem 4.7) that evaluates the process expression P using scheduler S is in fact a machine that evaluates the process family $\mathcal{P}$ using scheduler S.

Definition 5.4. We define $\mathbf{P}$ to be the set of all process families, $\mathbf{C}$ to be the set of all context families and $\mathbf{A}$ to be the set of all adversarial context families.

5.1 Observational Equivalence

Definition 5.5. Let $\mathcal{P}$ be a process family. We will say that the channel name $c_{q(|n|)}$ *appears in* $\mathcal{P}$ just when there are inputs and outputs on $c_{q(|n|)}$ that are sub-processes of $\mathcal{P}$.

Definition 5.6. Let $\mathcal{P}$ and $\mathcal{Q}$ be two process families. Let $\mathbf{q}$ be the set of polynomials $q(x)$ such that $\forall y : q(y) > 0$. We will say that $\mathcal{P}$ and $\mathcal{Q}$ are *observationally equivalent with respect to the scheduler* S if:

$$\forall q(x) \in \mathbf{q}.\forall \mathcal{C}[\ \ ] \in \mathbf{A}.\forall o \in Obs.\exists n_o.\forall n > n_o :$$
$$\Big|\ \mathrm{Prob}\big[C^n[P^n] \leadsto_S o\big] - \mathrm{Prob}\big[C^n[Q^n] \leadsto_S o\big]\Big| \leq \frac{1}{q(n)}$$

If this is so, we will write that $\mathcal{P} \cong_S \mathcal{Q}$.

Definition 5.7. Let $\mathbf{S}$ be a set of schedulers. If $\forall S \in \mathbf{S} : \mathcal{P} \cong_S \mathcal{Q}$ then we will write $\mathcal{P} \cong_{\mathbf{S}} \mathcal{Q}$ and we will say that $\mathcal{P}$ and $\mathcal{Q}$ are *observationally equivalent with respect to the set of schedulers* $\mathbf{S}$.

Definition 5.8. Let $\mathbf{S}$ be the set of all schedulers. If $\mathcal{P} \cong_{\mathbf{S}} \mathcal{Q}$ then we will write $\mathcal{P} \cong \mathcal{Q}$ and we will write that $\mathcal{P}$ and $\mathcal{Q}$ are *observationally equivalent.*

It is our goal to define adversaries to a protocol as adversarial contexts. Then, we will be able to prove security properties by stating observational equivalences.

The idea here is that if two process families are the same, then no adversarial context (*i.e.,* attacker) can distinguish between the two with significant probability once the security parameter gets large enough—the probability of one process family producing an observable is indistinguishable (to within an arbitrary polynomial factor) from the probability of the other process family producing that same observable.

However, if we forced observational equivalence to hold under all contexts, then we are essentially saying that adversaries can attack a protocol by changing the way a protocol executes (by specifying the scope of a private channel and, thereby, being able to read values transmitted on private channels). However, such attacks do not seem appropriate given our model of adversary capabilities. Hence, we discount such contexts from our definition of observational equivalence.

We can only such a property in the case that n is large enough as most any security system can be defeated by brute-force searches if the search-space is small enough. So, even though a context (*i.e.,* a brute force search over, say, keys) may distinguish two process families (that is break one protocol and not the other), once we increase the security parameter sufficiently, that context will get defeated.

The astute reader might argue that it might be possible for two process families $\mathcal{P}$ and $\mathcal{Q}$, when evaluated in the presence of a context family $\mathcal{C}[\ \ ]$, to produce the same distribution on observables but different distributions on sequences of observables. For example, though the process families $\mathcal{P}$ and $\mathcal{Q}$ might, both, produce the observables o_1 and o_2 with probability one, they are distinguishable by an observer because $\mathcal{P}$ always produces the observable sequence o_1, o_2 and $\mathcal{Q}$ always produces the observable sequence o_2, o_1. We believe that, given a particular scheduler and a particular pair of process families, it is possible to construct a context $\mathcal{C}[\ \ ]$ that distinguishes between the two on the basis of the joint distributions on observables. The guiding intuition is as follows. Let $\mathcal{A}$ be the distinguishing algorithm that an observer applies to the sequence of observables generated by the two process families. We construct $\mathcal{C}[\ \ ]$ to mount a so-called "man-in-the-middle" attack on the expression plugged into $\mathcal{C}[\ \ ]$'s hole. This is implemented as a sequence of nested inputs that intercept the desired observables and substitute them into an output term corresponding to $\mathcal{A}$. In order to ensure that the process expression plugged into the hole proceeds normally, the intercepted values are retransmitted. Naturally, such a construction affects the probabilistic behavior of the process family plugged into the hole. We freely admit that work needs to be done in order to provide a general characterization of the nature of this impact. However, in this paper we wished to outline some important properties of our analytic framework and we leave this problem to a future effort.

Lemma 5.9. *Let S be a scheduler. $\cong_S$ is an equivalence relation.*

Proof. Let $\mathbf{q}$ be the set of polynomials $q(x)$ such that $\forall y : q(y) > 0$. Reflexivity and symmetry follow from the definition of $\cong_S$. For transitivity, we let $\mathcal{P} \cong_S \mathcal{Q}$ and $\mathcal{Q} \cong_S \mathcal{R}$. Then:

$$\forall q(x) \in \mathbf{q}.\forall \mathcal{C}[\ \ ] \in \mathbf{A}.\forall o \in \mathit{Obs}.\exists n_1.\forall n > n_1 :$$
$$\Big|\ \mathrm{Prob}\big[C^n[P^n] \rightsquigarrow_S o\big] - \mathrm{Prob}\big[C^n[Q^n] \rightsquigarrow_S o\big]\Big| \leq \frac{1}{2q(n)}$$

and:

$$\forall q(x) \in \mathbf{q}.\forall \mathcal{C}[\ \ ] \in \mathbf{A}.\forall o \in \mathit{Obs}.\exists n_2.\forall n > n_2 :$$
$$\Big|\ \mathrm{Prob}\big[C^n[Q^n] \rightsquigarrow_S o\big] - \mathrm{Prob}\big[C^n[R^n] \rightsquigarrow_S o\big]\Big| \leq \frac{1}{2q(n)}$$

We let $n_3 = \max(n_1, n_2)$. Then transitivity follows directly:

$$\forall q(x) \in \mathbf{q}.\forall \mathcal{C}[\ \ ] \in \mathbf{A}.\forall o \in \mathit{Obs}.\forall n > n_3 :$$
$$\Big|\ \mathrm{Prob}\big[C^n[P^n] \rightsquigarrow_S o\big] - \mathrm{Prob}\big[C^n[R^n] \rightsquigarrow_S o\big]\Big|$$

$$
\begin{aligned}
&= \big|\, \mathrm{Prob}\big[C^n[P^n] \rightsquigarrow_S o\big] - \mathrm{Prob}\big[C^n[Q^n] \rightsquigarrow_S o\big] \\
&\qquad + \mathrm{Prob}\big[C^n[Q^n] \rightsquigarrow_S o\big] - \mathrm{Prob}\big[C^n[R^n] \rightsquigarrow_S o\big]\big| \\
&\leq \big|\, \mathrm{Prob}\big[C^n[P^n] \rightsquigarrow_S o\big] - \mathrm{Prob}\big[C^n[Q^n] \rightsquigarrow_S o\big]\big| \\
&\qquad + \big|\, \mathrm{Prob}\big[C^n[Q^n] \rightsquigarrow_S o\big] - \mathrm{Prob}\big[C^n[R^n] \rightsquigarrow_S o\big]\big| \\
&\leq \tfrac{1}{2q(n)} + \tfrac{1}{2q(n)} \ \leq\ \tfrac{1}{q(n)}
\end{aligned}
$$

So, if $\mathcal{P} \cong_S \mathcal{Q}$ and $\mathcal{Q} \cong_S \mathcal{R}$, then $\mathcal{P} \cong_S \mathcal{R}$. □

Lemma 5.10. *Let* $\mathbf{S}$ *be a set of schedulers. Then* $\cong_{\mathbf{S}}$ *and* $\cong$ *are equivalence relations.*

Proof. Let S be a scheduler. Since $\cong_S$ is an equivalence relation, it follows that $\cong_{\mathbf{S}}$ and $\cong$ are equivalence relations. □

We now show that $\cong_S$ is a congruence and derive some useful consequences.

Equivalence 5.11 (`SUBST-EQ`)**.** *Let S be a scheduler. Then* $\mathcal{P} \cong_S \mathcal{Q} \iff \forall \mathcal{C}[\ ] \in \mathbf{A} : \mathcal{C}[\mathcal{P}] \cong_S \mathcal{C}[\mathcal{Q}]$.

Proof. First, the reverse direction. Since $\forall \mathcal{C}[\ ] \in \mathbf{A} : \mathcal{C}[\mathcal{P}] \cong_S \mathcal{C}[\mathcal{Q}]$, we have that $[\mathcal{P}] \cong_S [\mathcal{Q}]$ and the desired result is trivially obtained.

Now for the forward direction. Assume, by way of constructing a contradiction, that for some adversarial context family $\mathcal{D}[\ ]$ and some polynomial $q(x)$ we have that $\big|\, \mathrm{Prob}\big[D^{\bar{n}}[C^{\bar{n}}[P^{\bar{n}}]] \rightsquigarrow_S o\big] - \mathrm{Prob}\big[D^{\bar{n}}[C^{\bar{n}}[Q^{\bar{n}}]] \rightsquigarrow_S o\big]\big| > \frac{1}{q(\bar{n})}$. But then the context $D^{\bar{n}}[C^{\bar{n}}[\ ]]$ distinguishes between $\mathcal{P}$ and $\mathcal{Q}$. Since $\mathcal{D}[\mathcal{C}[\ ]]$ is an adversarial context, we obtain a contradiction as $\mathcal{P} \cong_S \mathcal{Q}$ implies the observational equivalence with respect to S of $\mathcal{P}$ and $\mathcal{Q}$ under all adversarial contexts. □

We shall refer to `SUBST-EQ` as the rule of *substitutive equivalence*. A series of properties follow immediately from `SUBST-EQ`.

Equivalence 5.12 (`REPL-EQ`)**.** $\mathcal{P} \cong_S \mathcal{Q} \implies !_{r(|n|)}.\mathcal{P} \cong_S\ !_{r(|n|)}.\mathcal{Q}$ *where r is a polynomial such that* $\forall a \in \mathbb{N} : r(a) \geq 0$.

Equivalence 5.13 (`OUTPUTS-EQ`)**.** *If* $\mathcal{P} \cong_S \mathcal{Q}$ *then:*

$$c_{1p(|n|)}\langle m_1\rangle \mid \cdots \mid c_{ip(|n|)}\langle m_i\rangle \mid \mathcal{P} \cong_S c_{1p(|n|)}\langle m_1\rangle \mid \cdots \mid c_{ip(|n|)}\langle m_i\rangle \mid \mathcal{Q}$$

Equivalence 5.14 (`MATCH-EQ`)**.** $\mathcal{P} \cong_S \mathcal{Q} \implies [T_1 = T_2].\mathcal{P} \cong_S [T_1 = T_2].\mathcal{Q}$.

Equivalence 5.15 (`PARALLEL-COMP`)**.** *If* $\mathcal{P}_1 \cong_S \mathcal{P}_2$ *and* $\mathcal{Q}_1 \cong_S \mathcal{Q}_2$*, then* $\mathcal{P}_1 \mid \mathcal{Q}_1 \cong_S \mathcal{P}_2 \mid \mathcal{Q}_2$.

Proof. From $\mathcal{P}_1 \cong_S \mathcal{P}_2$ and substitutive equivalence, it follows that $\mathcal{P}_1 \mid \mathcal{Q}_1 \cong_S \mathcal{P}_2 \mid \mathcal{Q}_1$. Furthermore, from $\mathcal{Q}_1 \cong_S \mathcal{Q}_2$ and substitutive equivalence we get that $\mathcal{P}_2 \mid \mathcal{Q}_1 \cong_S \mathcal{P}_2 \mid \mathcal{Q}_2$. But then, from transitivity of observational equivalence we get that $\mathcal{P}_1 \mid \mathcal{Q}_1 \cong_S \mathcal{P}_2 \mid \mathcal{Q}_2$. □

5.2 Probabilistic Bisimulation

In this section we we define probabilistic bisimulation for the process families in our probabilistic process calculus. Probabilistic bisimulation will be very useful in proving a variety of properties of observational equivalence. The ideas here are based on bisimulation as presented in [Mil89, LS91, vGSS95].

Given process families $\mathcal{P}$ and $\mathcal{Q}$, the existence of a probabilistic bisimulation between $\mathcal{P}$ and $\mathcal{Q}$, which we write as $\mathcal{P} \sim \mathcal{Q}$, is a structural property that essentially stipulates that the iterated probabilistic evaluations starting from the process expression $\mathcal{P}$ may be simulated, in the sense of execution graphs, by the iterated probabilistic evaluations starting from the process expression $\mathcal{Q}$, and *vice versa*.

The main goal of this section is to define precisely the notion of probabilistic bisimulation (definition 5.54) and to prove that $\mathcal{P} \sim \mathcal{Q}$ implies $\mathcal{P} \cong_{\mathbf{W}} \mathcal{Q}$ (theorem 5.74) where $\mathbf{W}$ is the set of all weakly uniform schedulers (defn. 5.71) The latter property will be used in section 5.3 to establish basic properties of observational equivalence. The current section is of a rather technical nature and it may be skipped in the first reading.

In what follows it will be help the reader to recall the definitions of $S(\mathcal{P})$ (the set of schedulable processes—defn. 2.28) and $E(\mathcal{P})$ (the set of eligible communication triples—defn. 2.31).

5.2.1 Transitions

Here we will define transitions. Using transitions we will be able to model any macro-step that occurs during the evaluation of a process family in an arbitrary adversarial context.

Definition 5.16. Let $\mathcal{P}$ be a process family. We define the set $O(\mathcal{P})$ of *output transitions* as follows:

$$\begin{aligned}\{\langle s \equiv c_{p(|n|)}\langle a\rangle, \mathcal{C}_s[\ \ ]\rangle \,|\, & s \in S(\mathcal{P}), a \in [0, 2^{p(|n|)} - 1],\\ & \mathcal{P} \equiv \mathcal{C}_s[s], \mathcal{C}_s[\ \ ] \in \mathbf{C},\\ & c_{p(|n|)} \text{ not a private channel}\}\end{aligned}$$

Definition 5.17. Let $\mathcal{P}$ be a process family. We define the set $I(\mathcal{P})$ of *input transitions* as follows:

$$\begin{aligned}\{\langle s \equiv c_{p(|n|)}(x).\mathcal{R}, a, \mathcal{C}_s[\ \ ]\rangle \,|\, & s \in S(\mathcal{P}),\\ & a \in [0, 2^{p(|n|)} - 1],\\ & \mathcal{P} \equiv \mathcal{C}_s[s], \mathcal{C}_s[\ \ ] \in \mathbf{C},\\ & c_{p(|n|)} \text{ not private}\}\end{aligned}$$

Definition 5.18. Let $\mathcal{P}$ be a process family. The set $A(\mathcal{P})$ of *actual transitions* is just the set $E(\mathcal{P})$ of eligible communications (defn. 2.31).

Definition 5.19. If $\mathcal{P}$ has free variables (say, $v_1, \ldots, v_m$), we will create the set $F(\mathcal{P}) = \{\langle v_i, a\rangle \,|\, v_i \in \{v_1, \ldots, v_m\}, a \in \mathbb{N}\}$ of *free variable transitions.*

Definition 5.20. We say that a term T or a process expression P is *outermost* if it does not appear in the scope of an input expression or a match expression.

We note that every outermost term is an exposed term (a term that does not appear in the scope of an input expression).

Definition 5.21. Let $\mathcal{P}$ be a process family with un-outer-evaluated exposed terms (defn. 2.18). We define the set $EOT(\mathcal{P})$ of *exposed output-term transitions* as:

$$\{\langle \mathcal{C}[\ \ ], c_{p(|n|)}\langle T\rangle, a\rangle \,|\, \mathcal{P} \equiv \mathcal{C}[c_{p(|n|)}\langle T\rangle], a \in [0, 2^{p(|n|)} - 1]\}$$

where T is an outermost term.

Definition 5.22. Let $\mathcal{P}$ be a process family with un-outer-evaluated exposed terms (defn. 2.18). We define the set $EMT(\mathcal{P})$ of *exposed match-term transitions* as:

$$\{\langle \mathcal{C}[\ \ ], [T = U].\mathcal{Q}, 1, a\rangle \,|\, \mathcal{P} \equiv \mathcal{C}[[T = U].\mathcal{Q}], a \in \mathbb{N}\} \cup \{\langle \mathcal{C}[\ \ ], [T = U].\mathcal{Q}, 2, a\rangle \,|\, \mathcal{P} \equiv \mathcal{C}[[T = U].\mathcal{Q}], a \in \mathbb{N}\}$$

where T is an outermost term.

The '1's and '2's in the exposed match-term transitions indicate whether the first or second term in the match is the term that is being evaluated by the transition.

Definition 5.23. Let $\mathcal{P}$ be a process family with un-outer-evaluated exposed terms (defn. 2.18). We define the set $ET(\mathcal{P})$ of *exposed term transitions* as $EOT(\mathcal{P})$ if $EMT(\mathcal{P}) = \emptyset$ and $EMT(\mathcal{P})$ otherwise.

Thus, we ensure that we evaluate terms in matches before we evaluate other terms.

Definition 5.24. Let $\mathcal{P}$ be a process family with exposed matches (defn. 2.18). We define the set $EM(\mathcal{P})$ of *exposed match transitions* as:

$$\{\langle \mathcal{C}[\ \ ], [a_1 = a_2].\mathcal{Q}, \text{fail}\rangle, \langle \mathcal{C}[\ \ ], [a_1 = a_2].\mathcal{Q}, \text{pass}\rangle\}$$

where $\mathcal{P} \equiv \mathcal{C}[[a_1 = a_2].\mathcal{Q}]$, $a_1, a_2 \in \mathbb{N}$ and the match $[a_1 = a_2].\mathcal{Q}$ is outermost.

Insisting that the syntactic element being evaluated by exposed output-term transitions, exposed match-term transitions and exposed match transitions (defined in defns. 5.21, 5.22 and 5.24 respectively) be outermost prevents any attempt to outer-evaluate a term (or process expression in the cased of exposed match transitions) that appears in the scope of an input operator or another match operator *i.e.*, is not outermost.

Definition 5.25. We define the set $OE(\mathcal{P})$ of *outer-evaluation transition steps* as $ET(\mathcal{P})$ if $EM(\mathcal{P}) = \emptyset$ and $EM(\mathcal{P})$ otherwise.

Let $\mathcal{P}$ be an un-outer-evaluated process family. Then $OE(\mathcal{P}) \neq \emptyset$. Now let us reduce $\mathcal{P}$ using solely outer-evaluation transition steps until we arrive at a process family $\mathcal{Q}$ such that $OE(\mathcal{Q}) = \emptyset$. Then it is easy to see that $\mathcal{Q}$ is a process family obtained by outer-evaluating $\mathcal{P}$.

Definition 5.26. We define the set $\Pi(\mathcal{P})$ as follows:

$$\Pi(\mathcal{P}) = \begin{cases} OE(\mathcal{P}) & \mathcal{P} \text{ is not outer-evaluated,} \\ I(\mathcal{P}) \cup O(\mathcal{P}) \cup A(\mathcal{P}) & \text{if } E(\mathcal{P}) \text{ has no private communications,} \\ A(\mathcal{P}) & \text{if } E(\mathcal{P}) \text{ has private communications,} \\ \emptyset & \text{otherwise.} \end{cases}$$

Definition 5.27. $T(\mathcal{P})$, the set of *transitions* of $\mathcal{P}$, is defined as follows:

$$T(\mathcal{P}) = \begin{cases} F(\mathcal{P}) & \text{if } FV(\mathcal{P}) \neq \emptyset, \\ \Pi(\mathcal{P}) & \text{if } FV(\mathcal{P}) = \emptyset. \end{cases}$$

where $FV(\mathcal{P})$ is the set of free variables of $\mathcal{P}$.

Definition 5.28. If the transition t is in $A(\mathcal{P})$ we shall call it an *actual transition*. If t is in $I(\mathcal{P}) \cup O(\mathcal{P}) \cup F(\mathcal{P})$, we shall call t a *possible transition*.

We will say that a transition $t \in A(\mathcal{P})$ is a *private transition* if t performs a communication step that occurs on a private channel. Similarly, we will say that $t \in A(\mathcal{P}) \cup I(\mathcal{P}) \cup O(\mathcal{P})$ is a *public transition* if t occurs on a public channel.

Transitions in $OE(\mathcal{P})$ are *empty transitions* since such transitions do not produce observables (as they do not place natural numbers on public channels). We shall use the symbol ϵ to mean any type of empty transition. Finally, if t is a transition, we shall say that $e \in t$ just when the e is an element of the tuple t.

Definition 5.29. We introduce a special symbol $\bar{\mathbf{0}}$ that, while functionally identical to $\mathbf{0}$, can only be produced by failing a match. Thus, we can distinguish between $\mathbf{0}$s produced in the conventional manner (say by evaluating an output expression, or by hard-coding a $\mathbf{0}$ into the process) and $\mathbf{0}$s produced by failing matches. This will simplify our life, somewhat, in future sections (the reader is referred to section 5.2.4).

We now provide a semantics for each transition type.

Definition 5.30. Evaluating $\mathcal{P}$ via the output transition $\langle c_{p(|n|)}\langle a\rangle, \mathcal{C}[\ ]\rangle$ causes $\mathcal{P}$ to reduce to $\mathcal{C}[\mathbf{0}]$.

Evaluating $\mathcal{P}$ via the input transition $\langle c_{p(|n|)}(x).\mathcal{R}, a, \mathcal{C}[\ ]\rangle$ causes $\mathcal{P}$ to reduce to $\mathcal{C}[[a/x]\,\mathcal{R}]$.

Evaluating $\mathcal{P}$ via the actual transition $\langle c_{p(|n|)}(x).\mathcal{R}, c_{p(|n|)}\langle a\rangle, \mathcal{C}[\ ][\ ]\rangle$ causes $\mathcal{P}$ to reduce to $\mathcal{C}[\mathbf{0}][[a/x]\,\mathcal{R}]$.

Evaluating $\mathcal{P}$ via the free variable transition $\langle x, a\rangle$ causes $\mathcal{P}$ to reduce to $[a/x]\,\mathcal{P}$.

If $\mathcal{P} \equiv \mathcal{C}[c_{p(|n|)}\langle T\rangle]$ is reduced via the transition $\langle \mathcal{C}[\ \ ], c_{p(|n|)}\langle T\rangle, a\rangle$, we will obtain $\mathcal{C}[c_{p(|n|)}\langle a\rangle]$. If $\mathcal{P} \equiv \mathcal{C}[[T = U].\mathcal{Q}]$ is reduced via the transition $\langle \mathcal{C}[\ \ ], [T = U].\mathcal{Q}, 1, a\rangle$, we will obtain $\mathcal{C}[[a = U].\mathcal{Q}]$. And similarly, if $\mathcal{P} \equiv \mathcal{C}[[T = U].\mathcal{Q}]$ is reduced via the transition $\langle \mathcal{C}[\ \ ], [T = U].\mathcal{Q}, 2, a\rangle$, we will obtain $\mathcal{C}[[T = a].\mathcal{Q}]$.

Evaluating $\mathcal{C}[[a_1 = a_2].\mathcal{Q}]$ via the transition $\langle \mathcal{C}[\ \ ], [a_1 = a_2].\mathcal{Q}, \mathrm{fail}\rangle$ causes $\mathcal{C}[[a_1 = a_2].\mathcal{Q}]$ to reduce to $\mathcal{C}[\bar{\mathbf{0}}]$. On the other hand, evaluating $\mathcal{C}[[a_1 = a_2].\mathcal{Q}]$ via the transition $\langle \mathcal{C}[\ \ ], [a_1 = a_2].\mathcal{Q}, \mathrm{pass}\rangle$ causes $\mathcal{C}[[a_1 = a_2].\mathcal{Q}]$ to reduce to $\mathcal{C}[\mathcal{Q}]$.

Transitions are intended to capture the notion of a evaluation steps, *i.e.,* macro-steps (defn. 3.4). Assume that the process family $\mathcal{P}$ is ready to perform the macro-step indicated by an input/output pair π that is ready to communicate. This macro-step consists of an outer-evaluation followed by the communication step indicated by π (which is just substituting a value for the appropriate variable in $\mathcal{P}$) followed by another outer-evaluation. We can capture this macro-step using a series of empty transitions to represent the initial outer-evaluation, an actual transition to represent the communication step indicated by π, and another series of empty transitions to represent the final outer-evaluation.

Now, $\mathcal{P}$ could be evaluated in some adversarial context. Consequently there could be additional input/output pairs that are ready to communicate where either the input or the output is supplied by the adversarial context in which we are evaluating $\mathcal{P}$. We will deal with the consequences of evaluating $\mathcal{P}$ in such an adversarial context by supposing that $\mathcal{P}$ is evaluated in a "maximally benevolent adversarial context", *i.e.,* an adversarial context that can provide any value to any public input in $\mathcal{P}$ and read any value sent by any public output in $\mathcal{P}$. It is easy to see that the effects of such hypothetical macro-steps can be represented using our system of transitions—rather than using an actual transition to model the communication step, we use a possible transition to model the communication step.

Additionally, if $\mathcal{P}$ contains free variables, then there exists an adversarial context that can bind these free variables to channels and supply values to them (*i.e.,* a sort of evaluation context). To capture the effects of such a context we can use free variable transitions to supply values to the free variables and then empty transitions to model the subsequent outer-evaluations.

Finally, a note about the order in which these transitions occur. Given a process family $\mathcal{P}$ we have constructed our set of possible transitions, $T(\mathcal{P})$, in such a way that all free variable transitions occur first (if they exist), followed by any empty (*i.e.,* exposed term and exposed match) transitions (again, only if they exist). Only then will we perform our "communication" transitions (either actual communications or hypothetical communications involving the benevolent adversarial context).

Hence, it is easy to see that any macro-step that occurs during the evaluation of $\mathcal{P}$ in some arbitrary adversarial context can be modelled with transitions.

We now add probabilities to our notion of a transition.

Definition 5.31. Let $\mathcal{P}$ be a process family. Let t be a transition in $T(\mathcal{P})$.

We will write $\mathcal{P} \xrightarrow{p}_t \mathcal{Q}$ to indicate that $\mathcal{P}$ reduces to $\mathcal{Q}$ with probability p upon performing the transition t.

We will write $\mathcal{P} \rightarrow_t \mathcal{Q}$ to indicate that $\mathcal{P}$ reduces to $\mathcal{Q}$ with non-zero probability upon performing the transition t.

We will write $\mathrm{Prob}\big[\mathcal{P} \rightarrow_t \mathcal{Q}\big]$ to mean the probability that the transition t reduces $\mathcal{P}$ to $\mathcal{Q}$.

Definition 5.32. The precise probabilities associated with various types of transitions are given by the table below:

Transition type	Probability
$t \in I(\mathcal{P})$	1
$t \in O(\mathcal{P})$	1
$t \in A(\mathcal{P})$	1
$t \in F(\mathcal{P})$	1
$\langle \mathcal{C}[\ \ ], c_{p(\lvert n \rvert)}\langle T \rangle, a\rangle$	$\mathrm{Prob}\big[T \rightarrow_e a\big]$
$\langle \mathcal{C}[\ \ ], [T = U].\mathcal{Q}, 1, a\rangle$	$\mathrm{Prob}\big[T \rightarrow_e a\big]$
$\langle \mathcal{C}[\ \ ], [T = U].\mathcal{Q}, 2, a\rangle$	$\mathrm{Prob}\big[U \rightarrow_e a\big]$
$t \in EM(\mathcal{P})$	1

Notice that the probability associated with a particular transition does not include the probability of selecting that transition out of the set of possible transitions ($T(\mathcal{P})$). When actually evaluating $\mathcal{P}$ to $\mathcal{Q}$ we define the probability of taking a particular macro-step as the product of the probability of a scheduler selecting the communication indicated by the macro-step (say it substitutes a for x in $\mathcal{P}$) and the probability of outer-evaluating $[a/x]\,\mathcal{P}$ to $\mathcal{Q}$. Since we wish to generalize over schedulers and we wish transitions to capture even hypothetical macro-steps (*i.e.,* macro-steps that will not occur in evaluating $\mathcal{P}$ but might occur in evaluating $\mathcal{P}$ in some context), we cannot factor in the selection probability for transitions. Hence, the probabilities provided in the table are in fact conditional on the assumption that the transition was in fact selected.

Definition 5.33. Let t be a transition such that $\mathcal{P} \rightarrow_t \mathcal{Q}$. Then, we define the value of the function $\sigma(t)$ as $\mathcal{P}$ and the value of the function $\tau(t)$ as $\mathcal{Q}$—σ and τ respectively stand for the *source* and *target* of the transition t.

Definition 5.34. Consider a process family $\mathcal{P}$. Let $S \subseteq T(\mathcal{P})$ be a set of transitions and let $\mathcal{D}[\ \ ]$ be a context family. We will write $\mathcal{D}[S]$ to mean the set of transitions obtained by replacing the locating context $\mathcal{C}[\ \ ]$ found in each transition $t \in S$ with the context $\mathcal{D}[\mathcal{C}[\ \ ]]$. Similarly, we will write $\mathcal{D}[t]$ to mean the transition obtained by replacing the locating context $\mathcal{C}[\ \ ]$ found in the transition t with the context $\mathcal{D}[\mathcal{C}[\ \ ]]$.

Definition 5.35. Let I be a set of input transitions and O be a set of output transitions. Then the *product of I and O*, written $I \cdot O$, is the set:

$$\{\langle c_{p(\lvert n \rvert)}\langle a \rangle, c_{p(\lvert n \rvert)}(x).\mathcal{P}, \mathcal{C}_1[\ \ ] \mid \mathcal{C}_2[\ \ ]\rangle \mid \langle c_{p(\lvert n \rvert)}(x).\mathcal{P}, a, \mathcal{C}_1[\ \ ]\rangle \in I, \\ \langle c_{p(\lvert n \rvert)}\langle a \rangle, \mathcal{C}_2[\ \ ]\rangle \in O\}$$

Similarly, we will write $i \cdot o$ to mean the actual transition obtained by combining the input transition i and the output transition o where i and o occur on the same public channel.

Definition 5.36. Let $\mathbf{T}$ be the set of all transitions. We define a function $\rho : \mathbf{T} \to [1..6]$ as follows:

$$\rho(t) = \begin{cases} 1 & \text{if } t \text{ is a public transition,} \\ 2 & \text{if } t \text{ is a private transition,} \\ 3 & \text{if } t \text{ is an exposed output-term transition,} \\ 4 & \text{if } t \text{ is an exposed match-term transition,} \\ 5 & \text{if } t \text{ is an exposed match transition,} \\ 6 & \text{if } t \text{ is a free variable transition.} \end{cases}$$

$\rho(t)$ returns the priority of the transition t.

It will be useful for our purposes to be able to characterize the behavior of $\mathcal{P} \mid \mathcal{Q}$ in terms of the individual behaviors of $\mathcal{P}$ and $\mathcal{Q}$. The following definition and two lemmas proceed to do just this. We start by defining a set $\widehat{T}(\mathcal{P} \mid \mathcal{Q})$ of transitions. Lemma 5.38 proves that $\widehat{T}(\mathcal{P} \mid \mathcal{Q})$ and $T(\mathcal{P} \mid \mathcal{Q})$ (the set of transitions that $\mathcal{P} \mid \mathcal{Q}$ can take) are the same set. Lemma 5.40 then shows, roughly speaking, that $\mathcal{P} \mid \mathcal{Q} \xrightarrow{p_1 p_2}_t \mathcal{P}' \mid \mathcal{Q}$ if and only if $\mathcal{P} \xrightarrow{p_1}_{u_1} \mathcal{P}'$ and $\mathcal{Q} \xrightarrow{p_2}_{u_2} \mathcal{Q}'$.

Definition 5.37. Let $\mathcal{P}$ and $\mathcal{Q}$ be process families and let $\mathcal{C}_p[\ \ ]$ and $\mathcal{C}_q[\ \ ]$ be the context families $\mathcal{P} \mid [\ \ ]$ and $[\ \ ] \mid \mathcal{Q}$ respectively. We proceed to define a set of transitions $\widehat{T}(\mathcal{P} \mid \mathcal{Q})$ by case analysis on $T(\mathcal{P} \mid \mathcal{Q})$.

1. Suppose $T(\mathcal{P} \mid \mathcal{Q})$ contains free variable transitions. $\widehat{T}(\mathcal{P} \mid \mathcal{Q})$ is the set $F(\mathcal{P}) \cup F(\mathcal{Q})$.

2. Suppose $T(\mathcal{P} \mid \mathcal{Q})$ contains exposed match transitions. $\widehat{T}(\mathcal{P} \mid \mathcal{Q})$ is the set $\mathcal{C}_q[EM(\mathcal{P})] \cup \mathcal{C}_p[EM(\mathcal{Q})]$.

3. Suppose $T(\mathcal{P} \mid \mathcal{Q})$ contains exposed match-term transitions. $\widehat{T}(\mathcal{P} \mid \mathcal{Q})$ is the set $\mathcal{C}_q[EMT(\mathcal{P})] \cup \mathcal{C}_p[EMT(\mathcal{Q})]$.

4. Suppose $T(\mathcal{P} \mid \mathcal{Q})$ contains exposed output-term transitions. $\widehat{T}(\mathcal{P} \mid \mathcal{Q})$ is the set $\mathcal{C}_q[EOT(\mathcal{P})] \cup \mathcal{C}_p[EOT(\mathcal{Q})]$.

5. Suppose $T(\mathcal{P} \mid \mathcal{Q})$ contains private transitions. Then $\widehat{T}(\mathcal{P} \mid \mathcal{Q})$ is the set $\mathcal{C}_q[A(\mathcal{P})] \cup \mathcal{C}_p[A(\mathcal{Q})]$.

6. Suppose $T(\mathcal{P} \mid \mathcal{Q})$ contains public transitions. Then $\widehat{T}(\mathcal{P} \mid \mathcal{Q})$ is the set $\bigcup\{\mathcal{C}_q[I(\mathcal{P})], \mathcal{C}_q[O(\mathcal{P})], \mathcal{C}_q[A(\mathcal{P})]\} \cup \bigcup\{\mathcal{C}_p[I(\mathcal{Q})], \mathcal{C}_p[O(\mathcal{Q})], \mathcal{C}_p[A(\mathcal{Q})]\} \cup I(\mathcal{P}) \cdot O(\mathcal{Q}) \cup I(\mathcal{Q}) \cdot O(\mathcal{P})$.

Lemma 5.38. *Let $\mathcal{P}$ and $\mathcal{Q}$ be process families. Define $\widehat{T}(\mathcal{P} \mid \mathcal{Q})$ as per defn. 5.37. Then $T(\mathcal{P} \mid \mathcal{Q}) = \widehat{T}(\mathcal{P} \mid \mathcal{Q})$.*

Proof. There are several cases to consider:

1. Free variables. Assume that $T(\mathcal{P} \mid \mathcal{Q})$ contains a free variable transition. Then $T(\mathcal{P} \mid \mathcal{Q})$ can only contain free variable transitions as free variable transitions out-prioritize all other transition types. Suppose $\langle x, a\rangle \in T(\mathcal{P} \mid \mathcal{Q})$. Then x appears free in $\mathcal{P} \mid \mathcal{Q}$ whence x appears free in either $\mathcal{P}$ or $\mathcal{Q}$. So $\langle x, a\rangle$ is a member of either $F(\mathcal{P})$ or $F(\mathcal{Q})$. So $T(\mathcal{P} \mid \mathcal{Q}) \subseteq \widehat{T}(\mathcal{P} \mid \mathcal{Q})$.

   Assume that $\widehat{T}(\mathcal{P} \mid \mathcal{Q}) = F(\mathcal{P}) \cup F(\mathcal{Q}) \neq \emptyset$. Then we have three cases to consider:

   (a) $F(\mathcal{P}) \neq \emptyset$ and $F(\mathcal{Q}) \neq \emptyset$. Then both $\mathcal{P}$ and $\mathcal{Q}$ have free variables. Now $F(\mathcal{P} \mid \mathcal{Q})$ is defined to be $F(\mathcal{P}) \cup F(\mathcal{Q})$. Since free variable transitions out-prioritize any other sort of transition, $T(\mathcal{P} \mid \mathcal{Q})$ must be $F(\mathcal{P}) \cup F(\mathcal{Q})$ whence $\widehat{T}(\mathcal{P} \mid \mathcal{Q}) \subseteq T(\mathcal{P} \mid \mathcal{Q})$.

   (b) $F(\mathcal{P}) = \emptyset$. Then $\mathcal{P}$ has no free variables. However, since by assumption, $F(\mathcal{P}) \cup F(\mathcal{Q})$ is non-empty, it must be that $\mathcal{Q}$ has free variables. But then $T(\mathcal{P} \mid \mathcal{Q}) = F(\mathcal{Q})$ whence $\widehat{T}(\mathcal{P} \mid \mathcal{Q}) \subseteq T(\mathcal{P} \mid \mathcal{Q})$.

   (c) $F(\mathcal{Q}) = \emptyset$. Symmetric to the previous case.

   So $\widehat{T}(\mathcal{P} \mid \mathcal{Q}) = T(\mathcal{P} \mid \mathcal{Q})$.

2. Exposed matches. The argument here is very similar to that employed in the previous case. However, we need to account for the locating contexts found in all transitions save free variable transitions.

   Assume that $T(\mathcal{P} \mid \mathcal{Q})$ contains an exposed match transition t. Then $T(\mathcal{P} \mid \mathcal{Q})$ cannot contain any free variable transitions. Otherwise, the free variable transitions would out-prioritize the exposed match transitions and, consequently, $T(\mathcal{P} \mid \mathcal{Q})$ would not contain any exposed match transitions. Furthermore, $T(\mathcal{P} \mid \mathcal{Q})$ cannot contain any transition u of lower priority since t would out-prioritize u. Since $t \in T(\mathcal{P} \mid \mathcal{Q})$ is an exposed match transition, either $\mathcal{P}$ or $\mathcal{Q}$ must contain, as a sub-process, the match that t evaluates. In the former case, we have a transition $t' \in T(\mathcal{P})$ that evaluates the match as t does (*i.e.*, if u fails the match if and only if t does so). But then the transition $[t'] \mid \mathcal{Q}$ must evaluate the match exactly as u does (and, hence, as t does). The case where the match appears as a sub-process of $\mathcal{Q}$ is symmetric to the case where the match appears as a sub-process of $\mathcal{P}$. So $T(\mathcal{P} \mid \mathcal{Q}) \subseteq \widehat{T}(\mathcal{P} \mid \mathcal{Q})$.

   Assume that $\widehat{T}(\mathcal{P} \mid \mathcal{Q}) = \mathcal{C}_q[EM(\mathcal{P})] \cup \mathcal{C}_p[EM(\mathcal{Q})] \neq \emptyset$. Then we have three cases to consider:

   (a) $EM(\mathcal{P}) \neq \emptyset$ and $EM(\mathcal{Q}) \neq \emptyset$. Suppose $t \in \widehat{T}(\mathcal{P} \mid \mathcal{Q})$ is an exposed match transition that evaluates the match $[a_1 = a_2].\mathcal{R}$. Then either there must exist a transition $u \in EM(\mathcal{P})$ such that $t = [u] \mid \mathcal{Q}$, or there must exist a transition $v \in EM(\mathcal{Q})$ such that $t = \mathcal{P} \mid [v]$. But $[a_1 = a_2].\mathcal{R}$ is also an exposed match of $\mathcal{P} \mid \mathcal{Q}$ that is ready to be evaluated. The transition that evaluates it must be of the form $[u] \mid \mathcal{Q}$ or $\mathcal{P} \mid [v]$ depending on whether $[a_1 = a_2].\mathcal{R}$ is a sub-process of $\mathcal{P}$ or $\mathcal{Q}$ whence $\widehat{T}(\mathcal{P} \mid \mathcal{Q}) \subseteq T(\mathcal{P} \mid \mathcal{Q})$.

(b) $EM(\mathcal{P}) = \emptyset$. Then $\mathcal{P}$ has no exposed matches that are ready to evaluate. Furthermore, $T(\mathcal{P})$ cannot have any transition of higher priority than an exposed match transition. Otherwise, $T(\mathcal{P} \mid \mathcal{Q})$ would have transitions of higher priority than exposed match transitions whence $T(\mathcal{P} \mid \mathcal{Q})$ would not contain exposed match transitions. Since by assumption, $\mathcal{C}_p[EM(\mathcal{P})] \cup \mathcal{C}_p[EM(\mathcal{Q})]$ is non-empty, it must be that $\mathcal{Q}$ has exposed matches that are ready to be evaluated. But then $T(\mathcal{P} \mid \mathcal{Q}) = \mathcal{C}_p[EM(\mathcal{Q})]$ whence $\widehat{T}(\mathcal{P} \mid \mathcal{Q}) \subseteq T(\mathcal{P} \mid \mathcal{Q})$.

(c) $EM(\mathcal{Q}) = \emptyset$. Symmetric to the previous case.

So $\widehat{T}(\mathcal{P} \mid \mathcal{Q}) = T(\mathcal{P} \mid \mathcal{Q})$.

3. Exposed match-terms. We proceed as per case 2 remembering the priority relationships between different transition types.

4. Exposed output-terms. We proceed as per case 2.

5. Private transitions. We proceed as per case 2.

6. Public transitions. Assume that $T(\mathcal{P} \mid \mathcal{Q})$ contains a public transition t. Then $T(\mathcal{P} \mid \mathcal{Q})$ cannot contain any transitions of higher priority. Otherwise, those transitions would out-prioritize the public transitions and, consequently, $T(\mathcal{P} \mid \mathcal{Q})$ would not contain any public transitions. Furthermore, $T(\mathcal{P} \mid \mathcal{Q})$ cannot contain any transition u of lower priority since such transitions do not exist! There are two kinds of transitions in $T(\mathcal{P} \mid \mathcal{Q})$: 1) transitions with a locating context of the form $\mathcal{P} \mid \mathcal{C}[\ \ ]$ or $\mathcal{C}[\ \ ] \mid \mathcal{Q}$ (basic transitions); and, 2) transitions with a locating context of the form $\mathcal{C}[\ \ ] \mid \mathcal{D}[\ \ ]$ (synthetic transitions). Suppose t is a basic transition. Then, arguments similar to those used in case 2 can be used to show that t must also be a member of $\widehat{T}(\mathcal{P} \mid \mathcal{Q})$. If t is a synthetic transition then t must be a communication between $\mathcal{P}$ and $\mathcal{Q}$ whence either $\mathcal{P}$ has the input and $\mathcal{Q}$ the output, or $\mathcal{Q}$ has the input and $\mathcal{P}$ the output. In the former case, $t \in I(\mathcal{P}) \cdot O(\mathcal{Q})$ while in the latter case $t \in I(\mathcal{Q}) \cdot O(\mathcal{P})$. So $t \in \widehat{T}(\mathcal{P} \mid \mathcal{Q})$ whence $T(\mathcal{P} \mid \mathcal{Q}) \subseteq \widehat{T}(\mathcal{P} \mid \mathcal{Q})$.

Assume that $t \in \widehat{T}(\mathcal{P} \mid \mathcal{Q})$. There are three cases to consider:

(a) $\bigcup\{\mathcal{C}_q[I(\mathcal{P})], \mathcal{C}_q[O(\mathcal{P})], \mathcal{C}_q[A(\mathcal{P})]\}$ and $\bigcup\{\mathcal{C}_p[I(\mathcal{Q})], \mathcal{C}_p[O(\mathcal{Q})], \mathcal{C}_p[A(\mathcal{Q})]\}$ are both non-empty. If t is an element of

$$\bigcup\{\mathcal{C}_q[I(\mathcal{P})], \mathcal{C}_q[O(\mathcal{P})], \mathcal{C}_q[A(\mathcal{P})]\} \cup \bigcup\{\mathcal{C}_p[I(\mathcal{Q})], \mathcal{C}_p[O(\mathcal{Q})], \mathcal{C}_p[A(\mathcal{Q})]\}$$

the arguments brought to bear in case 2 can be used to show that $t \in T(\mathcal{P} \mid \mathcal{Q})$. So suppose that $t \in I(\mathcal{P}) \cdot O(\mathcal{Q})$. Then, we have an input expression that is a sub-process of $\mathcal{P}$ and an output expression that is a sub-process of $\mathcal{Q}$ such that both the input and output are on the same public channel. Furthermore, both the input and output are ready to communicate and $T(\mathcal{P} \mid \mathcal{Q})$ will, therefore, contain t. The case where $t \in I(\mathcal{Q}) \cdot O(\mathcal{P})$ is symmetric to the case where $t \in I(\mathcal{P}) \cdot O(\mathcal{Q})$. So, $\widehat{T}(\mathcal{P} \mid \mathcal{Q}) \subseteq T(\mathcal{P} \mid \mathcal{Q})$.

(b) $\bigcup\{\mathcal{C}_q[I(\mathcal{P})], \mathcal{C}_q[O(\mathcal{P})], \mathcal{C}_q[A(\mathcal{P})]\} = \emptyset$. So, $I(\mathcal{Q}) \cdot O(\mathcal{P}) = I(\mathcal{P}) \cdot O(\mathcal{Q}) = \emptyset$. Since $\widehat{T}(\mathcal{P} \mid \mathcal{Q})$ is non-empty, we can conclude that $t \in \bigcup\{\mathcal{C}_p[I(\mathcal{Q})], \mathcal{C}_p[O(\mathcal{Q})], \mathcal{C}_p[A(\mathcal{Q})]\}$. Whence the arguments brought to bear in case 2 can be used to show that $t \in T(\mathcal{P} \mid \mathcal{Q})$. So $\widehat{T}(\mathcal{P} \mid \mathcal{Q}) \subseteq T(\mathcal{P} \mid \mathcal{Q})$.

(c) $\bigcup\{\mathcal{C}_p[I(\mathcal{Q})], \mathcal{C}_p[O(\mathcal{Q})], \mathcal{C}_p[A(\mathcal{Q})]\} = \emptyset$. Symmetric to the previous case.

So $\widehat{T}(\mathcal{P} \mid \mathcal{Q}) = T(\mathcal{P} \mid \mathcal{Q})$.

Since the cases are exhaustive, $T(\mathcal{P} \mid \mathcal{Q}) = \widehat{T}(\mathcal{P} \mid \mathcal{Q})$. □

Definition 5.39. We define a two-place predicate $samePublicChannel(x, y)$ that returns true if and only if x and y are two transitions on the same public channel and one is an input transition and the other an output transition.

Lemma 5.40. *The following propositions are true:*

1. $\mathcal{P} \mid \mathcal{Q} \xrightarrow{1}_{\langle x,a\rangle} [a/x]\,\mathcal{P} \mid \mathcal{Q} \implies \mathcal{P} \xrightarrow{1}_{\langle x,a\rangle} [a/x]\,\mathcal{P}$ *where* $\mathcal{P}$ *contains the free variable* x,

2. $\mathcal{P} \xrightarrow{1}_{\langle x,a\rangle} [a/x]\,\mathcal{P} \implies \mathcal{P} \mid \mathcal{Q} \xrightarrow{1}_{\langle x,a\rangle} ([a/x]\,\mathcal{P}) \mid ([a/x]\,\mathcal{Q})$ *where* $\mathcal{P}$ *contains the free variable* x *and* $\mathcal{Q}$ *may or may not contain the free variable* x,

3. $\mathcal{P} \mid \mathcal{Q} \xrightarrow{1}_{\langle x,a\rangle} \mathcal{P} \mid [a/x]\,\mathcal{Q} \implies \mathcal{Q} \xrightarrow{1}_{\langle x,a\rangle} [a/x]\,\mathcal{Q}$ *where* $\mathcal{Q}$ *contains the free variable* x,

4. $\mathcal{Q} \xrightarrow{1}_{\langle x,a\rangle} [a/x]\,\mathcal{Q} \implies \mathcal{P} \mid \mathcal{Q} \xrightarrow{1}_{\langle x,a\rangle} ([a/x]\,\mathcal{P}) \mid ([a/x]\,\mathcal{Q})$ *where* $\mathcal{Q}$ *contains the free variable* x *and* $\mathcal{P}$ *may or may not contain the free variable* x,

5. $\mathcal{P} \mid \mathcal{Q} \xrightarrow{1}_{\langle x,a\rangle} [a/x]\,\mathcal{P} \mid [a/x]\,\mathcal{Q} \iff (\mathcal{P} \xrightarrow{1}_{\langle x,a\rangle} [a/x]\,\mathcal{P}) \wedge (\mathcal{Q} \xrightarrow{1}_{\langle x,a\rangle} [a/x]\,\mathcal{Q})$ *where both* $\mathcal{P}$ *and* $\mathcal{Q}$ *contain the free variable* x,

6. $\mathcal{P} \mid \mathcal{Q} \xrightarrow{p}_{t \notin F(\mathcal{P} \mid \mathcal{Q})} \mathcal{P}' \mid \mathcal{Q} \iff \big(\exists u \in T(\mathcal{P}) : \mathcal{P} \xrightarrow{p}_{u} \mathcal{P}') \wedge (\forall v \in T(\mathcal{Q}) : \rho(v) \leq \rho(u))\big)$,

7. $\mathcal{P} \mid \mathcal{Q} \xrightarrow{p}_{t \notin F(\mathcal{P} \mid \mathcal{Q})} \mathcal{P} \mid \mathcal{Q}' \iff \big(\exists u \in T(\mathcal{Q}) : (\mathcal{Q} \xrightarrow{p}_{u} \mathcal{Q}') \wedge (\forall v \in T(\mathcal{P}) : \rho(v) \leq \rho(u))\big)$,

8. $\mathcal{P} \mid \mathcal{Q} \xrightarrow{1}_{t \in A(\mathcal{P} \mid \mathcal{Q})} \mathcal{P}' \mid \mathcal{Q}' \iff (\exists u_1 \in T(\mathcal{P}) : \mathcal{P} \xrightarrow{1}_{u_1} \mathcal{P}') \wedge (\exists u_2 \in T(\mathcal{Q}) : \mathcal{Q} \xrightarrow{1}_{u_2} \mathcal{Q}') \wedge (samePublicChannel(u_1, u_2))$.

Proof. We handle each case in turn:

1. $\mathcal{P} \mid \mathcal{Q} \xrightarrow{1}_{\langle x,a \rangle} [a/x]\,\mathcal{P} \mid \mathcal{Q} \implies \mathcal{P} \xrightarrow{1}_{\langle x,a \rangle} [a/x]\,\mathcal{P}$ where $\mathcal{P}$ contains the free variable x. Since $\mathcal{P} \mid \mathcal{Q}$ reduces upon the free variable transition $\langle x, a \rangle$ to $[a/x]\,\mathcal{P} \mid \mathcal{Q}$, we know that x appears free in $\mathcal{P}$ (but not in $\mathcal{Q}$). Consequently, $T(\mathcal{P})$ must contain the free variable transition $\langle x, a \rangle$ whence the result follows.

2. $\mathcal{P} \xrightarrow{1}_{\langle x,a \rangle} [a/x]\,\mathcal{P} \implies \mathcal{P} \mid \mathcal{Q} \xrightarrow{1}_{\langle x,a \rangle} ([a/x]\,\mathcal{P}) \mid ([a/x]\,\mathcal{Q})$ where $\mathcal{P}$ contains the free variable x and $\mathcal{Q}$ may or may not contain the free variable x. Since $\mathcal{P}$ has the free variable x, we know that $\mathcal{P} \mid \mathcal{Q}$ has the free variable x. We are now confronted with two choices: 1) $\mathcal{Q}$ does not have the free variable x whence $\mathcal{P} \mid \mathcal{Q}$ reduces to $([a/x]\,\mathcal{P}) \mid \mathcal{Q}$ via the free variable transition $\langle x, a \rangle$ (since $[x/a]\,\mathcal{R} \equiv \mathcal{R}$ when x does not appear free in $\mathcal{R}$); or, 2) $\mathcal{Q}$ has the free variable x whence $\mathcal{P} \mid \mathcal{Q}$ reduces to $[a/x]\,\mathcal{P} \mid [a/x]\,\mathcal{Q}$ via the free variable transition $\langle x, a \rangle$.

3. $\mathcal{P} \mid \mathcal{Q} \xrightarrow{1}_{\langle x,a \rangle} \mathcal{P} \mid [a/x]\,\mathcal{Q} \implies \mathcal{Q} \xrightarrow{1}_{\langle x,a \rangle} [a/x]\,\mathcal{Q}$ where $\mathcal{Q}$ contains the free variable x. Analogous to case 1.

4. $\mathcal{Q} \xrightarrow{1}_{\langle x,a \rangle} [a/x]\,\mathcal{Q} \implies \mathcal{P} \mid \mathcal{Q} \xrightarrow{1}_{\langle x,a \rangle} ([a/x]\,\mathcal{P}) \mid ([a/x]\,\mathcal{Q})$ where $\mathcal{Q}$ contains the free variable x and $\mathcal{P}$ may or may not contain the free variable x. Analogous to case 2.

5. $\mathcal{P} \mid \mathcal{Q} \xrightarrow{1}_{\langle x,a \rangle} [a/x]\,\mathcal{P} \mid [a/x]\,\mathcal{Q} \iff (\mathcal{P} \xrightarrow{1}_{\langle x,a \rangle} [a/x]\,\mathcal{P}) \wedge (\mathcal{Q} \xrightarrow{1}_{\langle x,a \rangle} [a/x]\,\mathcal{Q})$ where both $\mathcal{P}$ and $\mathcal{Q}$ contain the free variable x. First the forward direction. Assume that $\mathcal{P} \mid \mathcal{Q} \xrightarrow{1}_{\langle x,a \rangle} [a/x]\,\mathcal{P} \mid [a/x]\,\mathcal{Q}$. Then x must appear free in both $\mathcal{P}$ and $\mathcal{Q}$ whence we easily obtain the forward direction.

   Now for the reverse direction. Assume that $\mathcal{P} \xrightarrow{1}_{\langle x,a \rangle} [a/x]\,\mathcal{P}$ and $\mathcal{Q} \xrightarrow{1}_{\langle x,a \rangle} [a/x]\,\mathcal{Q}$. Then x must appear free in $\mathcal{P} \mid \mathcal{Q}$ and, furthermore, it appears free in both $\mathcal{P}$ and $\mathcal{Q}$. Whence we easily obtain the reverse direction.

6. $\mathcal{P} \mid \mathcal{Q} \xrightarrow{p}_{t \notin F(\mathcal{P} \mid \mathcal{Q})} \mathcal{P}' \mid \mathcal{Q} \iff \big(\exists u \in T(\mathcal{P}) : \mathcal{P} \xrightarrow{p}_{u} \mathcal{P}') \wedge (\forall v \in T(\mathcal{Q}) : \rho(v) \leq \rho(u))\big)$. We will start with $t \in EM(\mathcal{P} \mid \mathcal{Q})$.

   First the forward direction. Assume that $\mathcal{P} \mid \mathcal{Q} \xrightarrow{1}_{t \in EM(\mathcal{P} \mid \mathcal{Q})} \mathcal{P}' \mid \mathcal{Q}$. Further assume that t evaluates the expression $[a_1 = a_2].\mathcal{R}$. Then $[a_1 = a_2].\mathcal{R}$ must be a sub-process of $\mathcal{P}$. Since $\mathcal{P}$ cannot contain any free variables, there must exist a transition $u \in T(\mathcal{P})$ that evaluates $[a_1 = a_2].\mathcal{R}$ such that $t = [u] \mid \mathcal{Q}$. So $\mathcal{P} \xrightarrow{1}_{u} \mathcal{P}'$. Furthermore, if $T(\mathcal{Q})$ contained a transition of higher priority than an exposed match transition (in this case only free variable transitions have higher priority), then $T(\mathcal{P} \mid \mathcal{Q})$ would have to contain a transition of higher priority than an exposed match transition whence $T(\mathcal{P} \mid \mathcal{Q})$ could not contain any exposed match transitions. Consequently, $T(\mathcal{Q})$ cannot contain transitions of higher priority than exposed match transitions.

Now for the reverse direction. We assume that $\mathcal{P} \xrightarrow{1}_{u \in EM(\mathcal{P})} \mathcal{P}'$ and assume that $T(\mathcal{Q})$ does not contain any transitions that could out-prioritize u. Then, $T(\mathcal{P} \mid \mathcal{Q})$ must contain the exposed match transition $t = [u] \mid \mathcal{Q}$. But then $\mathcal{P} \mid \mathcal{Q} \xrightarrow{1}_t \mathcal{P}' \mid \mathcal{Q}$ and the result is obtained.

In order to establish the desired result for each transition-type, we proceed similarly in the case of exposed match-term transitions, exposed output-term transitions, private transitions and public transitions, making use of the priority relationships established between the various types of transitions and taking care to note that the probability of a term evaluating to an atom depends only on the term and nothing else.

7. $\mathcal{P} \mid \mathcal{Q} \xrightarrow{p}_{t \notin F(\mathcal{P} \mid \mathcal{Q})} \mathcal{P} \mid \mathcal{Q}' \iff \big(\exists u \in T(\mathcal{Q}) : (\mathcal{Q} \xrightarrow{p}_u \mathcal{Q}') \wedge (\forall v \in T(\mathcal{P}) : \rho(v) \leq \rho(u))\big)$. Analogous to case 6.

8. $\mathcal{P} \mid \mathcal{Q} \xrightarrow{1}_{t \in A(\mathcal{P} \mid \mathcal{Q})} \mathcal{P}' \mid \mathcal{Q}' \iff (\exists u_1 \in T(\mathcal{P}) : \mathcal{P} \xrightarrow{1}_{u_1} \mathcal{P}') \wedge (\exists u_2 \in T(\mathcal{Q}) : \mathcal{Q} \xrightarrow{1}_{u_2} \mathcal{Q}') \wedge (\mathit{samePublicChannel}(u_1, u_2))$. First the forward direction. Assume that $\mathcal{P} \mid \mathcal{Q} \xrightarrow{1}_{t \in A(\mathcal{P} \mid \mathcal{Q})} \mathcal{P}' \mid \mathcal{Q}'$. In this case, there must be a communication between $\mathcal{P}$ and $\mathcal{Q}$. This communication cannot be on a private channel, since the scope of private channel communications in $\mathcal{P}$ and $\mathcal{Q}$ is restricted to $\mathcal{P}$ and $\mathcal{Q}$ respectively. Hence, we have a public communication between $\mathcal{P}$ and $\mathcal{Q}$. Furthermore, the input must be in one process and the output in the other. Also the input and output must be on the same public channel—otherwise communication between the two would have been impossible. Furthermore, neither $\mathcal{P}$ nor $\mathcal{Q}$ can have free variables or un-outer-evaluated sub-processes. Consequently, either $\mathcal{P} \xrightarrow{1}_{u_1 \in I(\mathcal{P})} \mathcal{P}'$ and $\mathcal{Q} \xrightarrow{1}_{u_2 \in O(\mathcal{Q})} \mathcal{Q}'$; or, $\mathcal{P} \xrightarrow{1}_{u_1 \in O(\mathcal{P})} \mathcal{P}'$ and $\mathcal{Q} \xrightarrow{1}_{u_2 \in I(\mathcal{Q})} \mathcal{Q}'$ depending on where the input and output expressions are located.

   Now for the reverse direction. We now have two cases:

   (a) There are only two transition types that can simultaneously affect both elements in a parallel composition (*i.e.*, affect $\mathcal{P}$ and $\mathcal{Q}$ when evaluating $\mathcal{P} \mid \mathcal{Q}$): free variable transitions and actual transitions. In order to obtain the reverse direction we, therefore, assume that $\mathcal{P} \xrightarrow{1}_{u_1 \in I(\mathcal{P})} \mathcal{P}'$ and $\mathcal{Q} \xrightarrow{1}_{u_2 \in O(\mathcal{Q})} \mathcal{Q}'$ with u_1 and u_2 occurring on the same public channel (we will handle the case where u_1 is an output transition and u_2 is an input transition separately). In fact, u_1 and u_2 *must* both be communications on public channels (for any process family $\mathcal{R}$, $I(\mathcal{R})$ and $O(\mathcal{R})$ contain, respectively, only inputs and outputs on public channels). Furthermore, since $I(\mathcal{P}) \subseteq T(\mathcal{P})$, we know that there are no private actual communications in $\mathcal{P}$ (recall that private communications have priority over public communications). Similarly, $O(\mathcal{Q}) \subseteq T(\mathcal{Q})$ implies that there are no private actual communications in $\mathcal{Q}$. Consequently, using lemma 5.38, we have that the transition $u_1 \cdot u_2 \in T(\mathcal{P} \mid \mathcal{Q})$ whence we obtain the desired result.

(b) $\mathcal{P} \xrightarrow{1}_{u_1 \in O(\mathcal{P})} \mathcal{P}'$ and $\mathcal{Q} \xrightarrow{1}_{u_2 \in I(\mathcal{Q})} \mathcal{Q}'$ with u_1 and u_2 occurring on the same public channel. Analogous to case 8a.

And so we establish the eight propositions in the statement of the lemma. $\square$

Lemmas 5.38 and 5.40 mean that we can characterize the behavior of $\mathcal{P} \mid \mathcal{Q}$ entirely in terms of the individual behaviors of $\mathcal{P}$ and $\mathcal{Q}$.

5.2.2 Execution Graphs

Here we define a execution graph. This is just a directed acyclic graph that uses transitions to capture the behavior of a process family during evaluation under arbitrary adversarial contexts. Since transitions can be used to model even hypothetical macro-steps (*i.e.*, those macro-steps caused by communication with a suitable adversarial evaluation context), they are a natural foundation on which to build our notion of an execution graph.

Definition 5.41. Let $\mathcal{P}$ be a process family. We define an *execution graph $G_\mathcal{P}$ for a process family $\mathcal{P}$* as a graph $\langle \mathcal{V}, \mathcal{E} \rangle$. We label vertices $v \in \mathcal{V}$ with process families and edges $e \in \mathcal{E}$ with a pair consisting of a transition and its associated probability. We define a function l that given edges $e \in \mathcal{E}$ produces the transition-probability pair labelling e, and given vertices $v \in \mathcal{V}$ produces the process family labelling v. We then construct the execution graph for $\mathcal{P}$ as follows:

1. Create a vertex v and label it with the process family $\mathcal{P}$.
2. For each vertex v and for each transition $t \in T(l(v))$ and for each $\mathcal{R}$, if $l(v) \to_t \mathcal{R}$, then if $\mathcal{R}$ does not already label a vertex in the execution graph add a vertex u labelled by $\mathcal{R}$ and add an edge labelled by $\langle t, \mathrm{Prob}\big[l(v) \to_t \mathcal{R}\big]\rangle$ from v to u, otherwise add an edge labelled by $\langle t, \mathrm{Prob}\big[l(v) \to_t \mathcal{R}\big]\rangle$ from v to the node in the execution graph already labelled by $\mathcal{R}$.
3. Close the transition system under step 2.

Henceforth we shall identify nodes and process families since a process family labels at most one node in a given execution graph.

Lemma 5.42. *Let $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ be the execution graph for any process family $\mathcal{P}$. Then $\mathcal{G}$ is a directed acyclic graph.*

Proof. We begin by augmenting the definition of the length of a process family to account for free variables. We define a polynomial *length*$'(\mathcal{P})$ inductively as follows:

$$\begin{array}{lcl} length'(\mathbf{0}) & = & 0 \\ length'(\nu_{c_{p(|n|)}}.(\mathcal{Q})) & = & length'(\mathcal{Q}) \\ length'(c_{p(|n|)}(x).(\mathcal{Q})) & = & 1 + length'(\mathcal{Q}) \\ length'(c_{p(|n|)}\langle T\rangle) & = & f_T \\ length'([T_1 = T_2].(\mathcal{Q})) & = & f_{T_1} + f_{T_2} + length'(\mathcal{Q}) \\ length'((Q_1 \mid Q_2)) & = & length'(Q_1) + length'(Q_2) \\ length'(!_{\gamma(|n|)}.(Q)) & = & \gamma(|n|) \cdot length'(Q) \end{array}$$

where f_T is a function that takes on the value 1 if T is an atom, 2 if T is not an atom but contains no free variables and $2 + i$ if T contains i free variables.

There are three broad types of transitions: (1) communication transitions *viz.,* transitions in $I(\mathcal{P})$, $O(\mathcal{P})$ and $E(\mathcal{P})$; (2) outer-evaluation transitions *viz.,* transitions in $EM(\mathcal{P})$ and $ET(\mathcal{P})$; and, (3) free-variable transitions found in $F(\mathcal{P})$. We note that each transition type strictly decreases $length'(\mathcal{P})$:

1. If t is a free variable transition, then evaluating it has one of two effects. Either a term T in $\mathcal{P}$ is replaced by an atom (t was just a single variable) or a term T has one less free variable. In either case, it is easy to see that $length'(\mathcal{P})$ strictly reduces.

2. If t is an exposed match-term transition, then evaluating it replaces a non-atomic term with an atom. So $length'(\mathcal{P})$ must strictly reduce.

3. If t is an exposed output-term transition, then evaluating it replaces a non-atomic term with an atom. So $length'(\mathcal{P})$ must strictly reduce.

4. If t is an exposed match transition, then evaluating it causes one of two things to happen. If the match is passed then we remove two atomic terms and are left with the expression bound to the match. In the case that the match failed, we eliminate two atoms and the expression bound to the match. In either case, $length'(\mathcal{P})$ must strictly reduce.

5. If t is in $I(\mathcal{P})$ (an input transition), then evaluating it causes an input expression to be replaced with the expression bound to the input. Hence the length reduces by at least one (the cost of the $c_{p(|n|)}(x)$ prefix). The length may reduce further if the substitution necessitated by evaluating t causes some term to become an atom or cause a reduction in the number of free variables of some term. So $length'(\mathcal{P})$ must strictly reduce.

6. If t is in $O(\mathcal{P})$ (an output transition), then evaluating it causes some output expression to be replaced with $\mathbf{0}$ (which has cost 0). So $length'(\mathcal{P})$ must strictly reduce.

7. If t is in $E(\mathcal{P})$ then we simultaneously replace some output expression with a $\mathbf{0}$ (as in the case that t is an output transition) and replace an input expression with the expression $\mathcal{Q}$ where $\mathcal{Q}$ is the expression obtained by performing the substitution necessitated by t on the expression bound to

the input (as in the case that t is an input transition). Hence $length'(\mathcal{P})$ must strictly reduce.

Since each transition strictly reduces $length'$, $\mathcal{G}$ must be an acyclic graph. Since, edges are labelled by transitions and transitions are directed (from source to target), $\mathcal{G}$ must also be directed. $\square$

Definition 5.43. We shall write $G_{\mathcal{P}}$ to mean the *execution graph for* $\mathcal{P}$.

Definition 5.44. We will refer to the node $\mathcal{P}$ in $G_{\mathcal{P}}$ as the *root node* and we shall refer to a node of a execution graph that has out-degree zero as a *terminal node.*

Note that the root node of an execution graph has in-degree zero.

Definition 5.45. We say that $\mathcal{P}$ is *done* if $T(\mathcal{P}) = \emptyset$.

Lemma 5.46. *Terminal nodes are labelled with process families that are done.*

Proof. By contradiction. Assume that a terminal node u (*i.e.,* a node with out-degree zero) is labelled by a process $\mathcal{P}$ that is not done. But then $T(\mathcal{P})$ is non-empty whence there must be at least one edge leaving u, thereby contradicting the assumption. $\square$

Definition 5.47. Let $G_{\mathcal{P}} = \langle \mathcal{V}, \mathcal{E} \rangle$ be a labelled transition system for the process family $\mathcal{P}$. We indicate that an edge goes from the vertex u to the vertex v by writing (u, v).

If $e \in \mathcal{E}$, we define $\text{Prob}\big[e\big]$, the probability associated with the edge e, as $\pi_2(l(e))$ where π_2 is the second projection function. That is to say $\text{Prob}\big[e\big]$ returns the probability associated with the transition that generated e.

We will write $u : \mathcal{P}$ to mean a vertex labelled by $\mathcal{P}$ and $e : t$ to mean an edge labelled by the pair $\langle t, \text{Prob}\big[e\big] \rangle$.

Finally, if $\mathcal{Q}$ labels a vertex in $\mathcal{V}$ we will write $\mathcal{Q} \in G_{\mathcal{P}}$. Similarly, if $\langle t, \text{Prob}\big[e\big] \rangle$ labels an edge e in $\mathcal{E}$ we will write $t \in G_{\mathcal{P}}$.

5.2.3 The Probabilistic Bisimulation Relation

Definition 5.48. We say that two transitions t and u are *analogous* or *analogues* if the structure of t and u are given by the following table:

t	u
$\langle c_{q(\lvert n \rvert)}(x).\mathcal{P}, c_{q(\lvert n \rvert)}\langle a \rangle, \mathcal{C}[\ ][\ ] \rangle$	$\langle c_{q(\lvert n \rvert)}(x).\mathcal{Q}, c_{q(\lvert n \rvert)}\langle a \rangle, \mathcal{D}[\ ][\ ] \rangle$
$\langle c_{q(\lvert n \rvert)}(x).\mathcal{P}, a, \mathcal{C}[\ ] \rangle$	$\langle c_{q(\lvert n \rvert)}(x).\mathcal{Q}, a, \mathcal{D}[\ ] \rangle$
$\langle c_{q(\lvert n \rvert)}\langle a \rangle, \mathcal{C}[\ ] \rangle$	$\langle c_{q(\lvert n \rvert)}\langle a \rangle, \mathcal{D}[\ ] \rangle$
$\langle \mathcal{C}[\ ], c_{q(\lvert n \rvert)}\langle T \rangle, a \rangle$	$\langle \mathcal{D}[\ ], c_{q(\lvert n \rvert)}\langle T \rangle, a \rangle$
$\langle \mathcal{C}[\ ], [T = U].\mathcal{P}, i, a \rangle$	$\langle \mathcal{D}[\ ], [T = U].\mathcal{Q}, i, a \rangle$
$\langle x, a \rangle$	$\langle x, a \rangle$
$\langle \mathcal{C}[\ ], [a_1 = a_2].\mathcal{P}, \text{fail} \rangle$	$\langle \mathcal{D}[\ ], [b_1 = b_2].\mathcal{Q}, \text{fail} \rangle$
$\langle \mathcal{C}[\ ], [a_1 = a_2].\mathcal{P}, \text{pass} \rangle$	$\langle \mathcal{D}[\ ], [b_1 = b_2].\mathcal{Q}, \text{pass} \rangle$

Definition 5.49. Let $\mathcal{P}$ and $\mathcal{Q}$ be process families. We say that the relation $\gamma \subseteq T(\mathcal{P}) \times T(\mathcal{Q})$ is a *correspondence between $\mathcal{P}$ and $\mathcal{Q}$* if:

1. $\langle u, v\rangle \in \gamma$ implies that u and v are analogous,
2. $\langle u, v_1\rangle \in \gamma$ and $\langle u, v_2\rangle \in \gamma$ implies that $v_1 = v_2$,
3. $\langle u_1, v\rangle \in \gamma$ and $\langle u_2, v\rangle \in \gamma$ implies that $u_1 = u_2$.

Definition 5.50. Let γ be a correspondence between $\mathcal{P}$ and $\mathcal{Q}$. If $\langle u, v\rangle \in \gamma$ we will say that u and v *correspond*.

Definition 5.51. Given binary relations R, R_1 and R_2, the *composition, R_1R_2,* of two binary relations and the *inverse, R^{-1},* of a relation are given by:

$$R_1R_2 \quad = \quad \{\langle x, z\rangle |\, \exists y : \langle x, y\rangle \in R_1 \text{ and } \langle y, z\rangle \in R_2\}$$

$$R^{-1} \quad = \quad \{\langle y, x\rangle |\, \langle x, y\rangle \in R\}$$

Lemma 5.52. *Let γ be a correspondence between $\mathcal{P}$ and $\mathcal{Q}$ and let δ be a correspondence between $\mathcal{Q}$ and $\mathcal{R}$. Then:*

(1) $id = \{\langle t, t\rangle |\, t \in T(\mathcal{P})\}$ *is a correspondence between $\mathcal{P}$ and $\mathcal{P}$*
(2) $\gamma\delta$ *is a correspondence between $\mathcal{P}$ and $\mathcal{R}$*
(3) γ^{-1} *is a correspondence between $\mathcal{Q}$ and $\mathcal{P}$*
(4) $\emptyset$ *is a correspondence between any two process families*

Proof. (1), (3) and (4) are very easy to see. We will show (2). Clearly, if t and u are analogues and u and v are analogues, then t and v are analogues. Now assume that $\langle t, v_1\rangle \in \gamma\delta$ and $\langle t, v_2\rangle \in \gamma\delta$ with $v_1 \neq v_2$. Then, either we have u such that $\langle u, v_1\rangle \in \delta$ and $\langle u, v_2\rangle \in \delta$ or we have $u_1 \neq u_2$ such that $\langle u_1, v_1\rangle \in \delta$ and $\langle u_2, v_2\rangle \in \delta$. The first case contradicts the assumption that δ is a correspondence between $\mathcal{Q}$ and $\mathcal{R}$. The second case implies that $\langle t, u_1\rangle \in \gamma$ and $\langle t, u_2\rangle \in \gamma$ which contradicts the assumption that γ is a correspondence between $\mathcal{P}$ and $\mathcal{Q}$. Hence $\langle t, v_1\rangle \in \gamma\delta$ and $\langle t, v_2\rangle \in \gamma\delta$ implies that $v_1 = v_2$. We argue similarly to show that $\langle t_1, v\rangle \in \gamma\delta$ and $\langle t_2, v\rangle \in \gamma\delta$ implies that $t_1 = t_2$. Hence $\gamma\delta$ is a correspondence between $\mathcal{P}$ and $\mathcal{R}$. $\square$

Definition 5.53. We say that the correspondence-indexed pair $\langle \mathcal{P}, \mathcal{Q}\rangle_\gamma$ is a *correspondence pair* just when γ is a correspondence between $\mathcal{P}$ and $\mathcal{Q}$.

Definition 5.54. We will say that a set B of correspondence pairs is a *probabilistic bisimulation* if $\langle \mathcal{P}, \mathcal{Q}\rangle_\gamma \in B$ implies that:

1. For all $t \in T(\mathcal{P})$, if $\mathcal{P} \xrightarrow{p}_t \mathcal{P}'$ then there exists $\mathcal{Q}'$, u and γ' such that $\mathcal{Q} \xrightarrow{p}_u \mathcal{Q}'$, $\langle t, u\rangle \in \gamma$ and $\langle \mathcal{P}', \mathcal{Q}'\rangle_{\gamma'} \in B$, and,
2. For all $t \in T(\mathcal{Q})$, if $\mathcal{Q} \xrightarrow{p}_u \mathcal{Q}'$ then there exists $\mathcal{P}'$, t and γ' such that $\mathcal{P} \xrightarrow{p}_t \mathcal{P}'$, $\langle t, u\rangle \in \gamma$ and $\langle \mathcal{P}', \mathcal{Q}'\rangle_{\gamma'} \in B$.

Definition 5.55. Given probabilistic bisimulations B, B_1 and B_2, the *composition,* B_1B_2*,* of two probabilistic bisimulations and the *inverse,* B^{-1}*,* of a probabilistic bisimulation are given by:

$$B_1B_2 = \{\langle x,z\rangle_{\gamma_1\gamma_2} | \exists y : \langle x,y\rangle_{\gamma_1} \in B_1 \text{ and } \langle y,z\rangle_{\gamma_2} \in B_2\}$$

$$B^{-1} = \{\langle y,x\rangle_{\gamma^{-1}} | \langle x,y\rangle_{\gamma} \in B\}$$

Lemma 5.56. *Assume that each B_i ($i \in I$) is a probabilistic bisimulation. Then the following are all probabilistic bisimulations:*

(1) $\{\langle \mathcal{P},\mathcal{P}\rangle_{id} | \mathcal{P} \in \mathbf{P}\}$	(2) B_1B_2
(3) B_1^{-1}	(4) $\bigcup_{i\in I} B_i$

Proof. (1) and (3) are very easy to see. We will start by showing (2). Let $\langle \mathcal{P},\mathcal{R}\rangle_{\gamma\delta} \in B_1B_2$. Then, for some Q we have that $\langle \mathcal{P},\mathcal{Q}\rangle_{\gamma} \in B_1$ and $\langle \mathcal{Q},\mathcal{R}\rangle_{\delta} \in B_2$. Let $\mathcal{P} \xrightarrow{p}_t \mathcal{P}'$. Since $\langle \mathcal{P},\mathcal{Q}\rangle_{\gamma} \in B_1$, for some $\mathcal{Q}'$ we have that $\mathcal{Q} \xrightarrow{p}_{u:\langle t,u\rangle\in\gamma} \mathcal{Q}'$ and $\langle \mathcal{P}',\mathcal{Q}'\rangle_{\gamma'} \in B_1$. Also since $\langle \mathcal{Q},\mathcal{R}\rangle_{\delta} \in B_2$ we have, for some $\mathcal{R}'$, that $\mathcal{R} \xrightarrow{p}_{v:\langle u,v\rangle\in\delta} \mathcal{R}'$ and $\langle \mathcal{Q}',\mathcal{R}'\rangle_{\delta'} \in B_2$. Hence $\langle \mathcal{P}',\mathcal{R}'\rangle_{\gamma'\delta'} \in B_1B_2$. By similar reasoning, if $R \xrightarrow{p}_t \mathcal{R}'$, we can find $\mathcal{P}'$ such that $\mathcal{P} \xrightarrow{p}_v \mathcal{P}'$ and $\langle \mathcal{P}',\mathcal{R}'\rangle_{\gamma'\delta'} \in B_1B_2$.

(4) is also straightforward. Let $\langle \mathcal{P},\mathcal{Q}\rangle_{\gamma} \in \bigcup_{i\in I} B_i$. Then, for some B_i, $\langle \mathcal{P},\mathcal{Q}\rangle_{\gamma} \in B_i$. Now let $\mathcal{P} \xrightarrow{p}_t \mathcal{P}'$. Since $\langle \mathcal{P},\mathcal{Q}\rangle \in B_i$, for some $\mathcal{Q}'$ we have that $\mathcal{Q} \xrightarrow{p}_{u:\langle t,u\rangle\in\gamma} \mathcal{Q}'$ and $\langle \mathcal{P}',\mathcal{Q}'\rangle_{\gamma'} \in B_i$. Hence $\langle \mathcal{P}',\mathcal{Q}'\rangle_{\gamma'} \in \bigcup_{i\in I} B_i$. By similar reasoning, if $\mathcal{Q} \xrightarrow{p}_t \mathcal{Q}'$ we can find a $\mathcal{P}'$ such that $\mathcal{P} \xrightarrow{p}_{u:\langle u,t\rangle\in\gamma} \mathcal{P}'$ and $\langle \mathcal{P}',\mathcal{Q}'\rangle_{\gamma'} \in B_i$. □

Definition 5.57. We write that $\mathcal{P}$ is *probabilistically similar to* or *similar to* $\mathcal{Q}$ (written $\mathcal{P} \sim \mathcal{Q}$) if $\langle \mathcal{P},\mathcal{Q}\rangle_{\gamma} \in B$ for some probabilistic bisimulation B.

Hence in order to show that $\mathcal{P} \sim \mathcal{Q}$ it suffices to construct a suitable probabilistic bisimulation.

Definition 5.58. If there exists a probabilistic bisimulation B with $\langle \mathcal{P},\mathcal{Q}\rangle_{\gamma} \in B$, we shall say that B *establishes* $\mathcal{P} \sim \mathcal{Q}$.

Lemma 5.59. *Let $\mathcal{P}$ and $\mathcal{Q}$ be process families such that $\mathcal{P} \sim \mathcal{Q}$. Then $FV(\mathcal{P}) = FV(\mathcal{Q})$.*

Proof. If $\mathcal{P}$ has free variables $x_1,\ldots,x_m$ then $T(\mathcal{P}) = F(\mathcal{P})$ which is just the set of transitions of the form $\langle x_i,a\rangle$ where $a \in \mathbb{N}$ and $1 \le i \le m$. Since $\mathcal{P} \sim \mathcal{Q}$, $T(\mathcal{Q})$ must contain every transition in $F(\mathcal{P})$ whence every free variable of $\mathcal{P}$ is a free variable of $\mathcal{Q}$. Similarly we can conclude that every free variable of $\mathcal{Q}$ is a free variable of $\mathcal{P}$. Whence the free variables of $\mathcal{P}$ are precisely the free variables of $\mathcal{Q}$. □

5.2.4 Compositionality of the Bisimulation Relation

Lemma 5.60. *Let $\mathcal{P} \sim \mathcal{Q}$ be process families such that neither $c_{q(|n|)}(x).\mathcal{P}$ nor $c_{q(|n|)}(x).\mathcal{Q}$ have any free variables. Then $c_{q(|n|)}(x).\mathcal{P} \sim c_{q(|n|)}(x).\mathcal{Q}$.*

Proof. Since $\mathcal{P} \sim \mathcal{Q}$ there must exist a probabilistic bisimulation S such that for some correspondence, δ, between $\mathcal{P}$ and $\mathcal{Q}$, $\langle \mathcal{P}, \mathcal{Q} \rangle_\delta \in S$. Define γ as $\{\langle \langle c_{q(|n|)}(x).\mathcal{P}, a, [\ \ ] \rangle, \langle c_{q(|n|)}(x).\mathcal{Q}, a, [\ \ ] \rangle \rangle \,|\, a \in [0, 2^{q(|n|)} - 1]\}$. It is easy to see that γ is a correspondence between $c_{q(|n|)}(x).\mathcal{P}$ and $c_{q(|n|)}(x).\mathcal{Q}$. We will show that $B = \{\langle c_{p(|n|)}(x).\mathcal{P}, c_{p(|n|)}(x).\mathcal{Q} \rangle_\gamma\} \cup S$ is a suitable probabilistic bisimulation.

Assume that $c_{q(|n|)}(x).\mathcal{P} \xrightarrow{1}_{t = \langle c_{q(|n|)}(x).\mathcal{P}, a, [\ \ ] \rangle} \mathcal{R}$. There are two cases to consider:

1. $\mathcal{P}$ has the free variable x. Then, $\mathcal{R} \equiv [a/x]\,\mathcal{P}$. So, $\mathcal{P} \xrightarrow{1}_{\langle x, a \rangle} [a/x]\,\mathcal{P}$ whence, since $\mathcal{P} \sim \mathcal{Q}$, we have that $\mathcal{Q} \xrightarrow{1}_{\langle x, a \rangle} [a/x]\,\mathcal{Q}$. But there exists $u = \langle c_{q(|n|)}(x).\mathcal{Q}, a, [\ \ ] \rangle$ with $\langle t, u \rangle \in \gamma$. Furthermore $c_{q(|n|)}(x).\mathcal{Q} \xrightarrow{1}_u [a/x]\,\mathcal{Q}$. Since $\mathcal{P} \sim \mathcal{Q}$ we have that $\langle [a/x]\,\mathcal{P}, [a/x]\,\mathcal{Q} \rangle_{\delta'} \in S$.

2. Neither $\mathcal{P}$ nor $\mathcal{Q}$ have any free variables. So $c_{q(|n|)}(x).\mathcal{P} \xrightarrow{1}_t \mathcal{P}$. Now there exists $u = \langle c_{q(|n|)}(x).\mathcal{Q}, a, [\ ] \rangle$ with $\langle t, u \rangle \in \gamma$. Furthermore $c_{q(|n|)}(x).\mathcal{Q} \xrightarrow{1}_u \mathcal{Q}$. Finally, since $\mathcal{P} \sim \mathcal{Q}$, it must be that $\langle \mathcal{P}, \mathcal{Q} \rangle_\delta \in S$.

A symmetric argument allows us to complete the proof. □

Lemma 5.61. *Let $\mathcal{P} \sim \mathcal{Q}$ be process families such that $[T_1 = T_2].\mathcal{P}$ and $[T_1 = T_2].\mathcal{Q}$ have no free variables. Then $[T_1 = T_2].\mathcal{P} \sim [T_1 = T_2].\mathcal{Q}$.*

Proof. Since $\mathcal{P} \sim \mathcal{Q}$ there must exist a probabilistic bisimulation S and a correspondence, δ, between $\mathcal{P}$ and $\mathcal{Q}$ such that $\langle \mathcal{P}, \mathcal{Q} \rangle_\delta \in S$. Define γ_1 as:

$$\{\langle \langle [\ \ ], [a_1 = a_2].\mathcal{P}, \text{pass} \rangle, \langle [\ \ ], [a_1 = a_2].\mathcal{Q}, \text{pass} \rangle \rangle\} \cup \\ \{\langle \langle [\ \ ], [a_1 = a_2].\mathcal{P}, \text{fail} \rangle, \langle [\ \ ], [a_1 = a_2].\mathcal{Q}, \text{fail} \rangle \rangle\}$$

where $a_1, a_2 \in \mathbb{N}$. Define γ_2 as:

$$\{\langle \langle [\ \ ], [T_1 = a].\mathcal{P}, 1, b \rangle, \langle [\ \ ], [T_1 = a].\mathcal{Q}, 1, b \rangle \rangle\}$$

where $a, b \in \mathbb{N}$. Define γ_3 as:

$$\{\langle \langle [\ \ ], [a = T_2].\mathcal{P}, 2, b \rangle, \langle [\ \ ], [a = T_2].\mathcal{Q}, 2, b \rangle \rangle\}$$

where $a, b \in \mathbb{N}$. Define γ_4 as:

$$\{\langle \langle [\ \ ], [T_1 = T_2].\mathcal{P}, 1, b \rangle, \langle , [\ \ ][T_1 = T_2].\mathcal{Q}, 1, b \rangle \rangle\} \cup \\ \{\langle \langle [\ \ ], [T_1 = T_2].\mathcal{P}, 2, b \rangle, \langle [\ \ ], [T_1 = T_2].\mathcal{Q}, 2, b \rangle \rangle\}$$

where $b \in \mathbb{N}$.

It is easy to see that γ_1 is a correspondence between $[a_1 = a_2].\mathcal{P}$ and $[a_1 = a_2].\mathcal{Q}$, γ_2 is a correspondence between $[T_1 = a].\mathcal{P}$ and $[T_1 = a].\mathcal{Q}$, γ_3 is a correspondence between $[a = T_2].\mathcal{P}$ and $[a = T_2].\mathcal{Q}$, and γ_4 is a correspondence between $[T_1 = T_2].\mathcal{P}$ and $[T_1 = T_2].\mathcal{Q}$.

We will show that B given by:

$$\{\langle [T_1 = T_2].\mathcal{P}, [T_1 = T_2].\mathcal{Q}\rangle_{\gamma_4}\} \cup \{\langle [a = T_2].\mathcal{P}, [a = T_2].\mathcal{Q}\rangle_{\gamma_3}\} \cup \{\langle \bar{\mathbf{0}}, \bar{\mathbf{0}}\rangle_{id}\} \cup \{\langle [T_1 = a].\mathcal{P}, [T_1 = a].\mathcal{Q}\rangle_{\gamma_2}\} \cup \{\langle [a_1 = a_2].\mathcal{P}, [a_1 = a_2].\mathcal{Q}\rangle_{\gamma_1}\} \cup S$$

is a suitable probabilistic bisimulation.

Assume that $[T_1 = T_2].\mathcal{P} \xrightarrow{p}_t \mathcal{R}$. There are four cases to consider:

1. T_1 and T_2 are the atoms a_1 and a_2 respectively. Then either $a_1 = a_2$ or $a_1 \neq a_2$. In the former case, $t = \langle [\ \ ], [a_1 = a_2].\mathcal{P}, \text{pass}\rangle$, $p = 1$ and $\mathcal{R} \equiv \mathcal{P}$. But there exists $u = \langle [\ \ ], [a_1 = a_2].\mathcal{Q}, \text{pass}\rangle$ with $\langle t, u\rangle \in \gamma_1$. Furthermore $[a_1 = a_2].\mathcal{Q} \xrightarrow{1}_u \mathcal{Q}$. Finally, the construction of B gives us that $\langle \mathcal{P}, \mathcal{Q}\rangle_\delta \in B$ since, by assumption, $\mathcal{P} \sim \mathcal{Q}$.

   If $a_1 \neq a_2$, then $\mathcal{R} \equiv \bar{\mathbf{0}}$, $p = 1$ and $t = \langle [\ \ ], [a_1 = a_2].\mathcal{Q}, \text{fail}\rangle$. But there exists $u = \langle [\ \ ], [a_1 = a_2].\mathcal{Q}, \text{fail}\rangle$ with $\langle t, u\rangle \in \gamma_1$. Furthermore $[a_1 = a_2].\mathcal{Q} \xrightarrow{1}_u \bar{\mathbf{0}}$. Finally, the construction of B gives us that $\langle \bar{\mathbf{0}}, \bar{\mathbf{0}}\rangle_{id} \in B$.

2. T_1 is the atom a and T_2 is not an atom. Then, $\mathcal{R} \equiv [a = b].\mathcal{P}$, $p = \text{Prob}\big[T_2 \rightarrow_e b\big]$ and $t = \langle [\ \ ], [a = T_2].\mathcal{P}, 2, b\rangle$. But there exists $u = \langle [\ \ ], [a = T_2].\mathcal{Q}, 2, b\rangle$ such that $\langle t, u\rangle \in \gamma_2$. Furthermore, $[a = T_2].\mathcal{Q} \xrightarrow{p'}_u [a = b].\mathcal{Q}$ with $p' = \text{Prob}\big[T_2 \rightarrow_e b\big]$. Now the construction of B gives us that $\langle [a = b].\mathcal{P}, [a = b].\mathcal{Q}\rangle_{\gamma_1} \in B$. In order to complete the proof, we bring to bear the arguments employed in case 1.

3. T_1 is not an atom and T_2 is the atom a. This case is symmetric to case 2.

4. Neither T_1 nor T_2 are atoms. There are two cases to consider. First, assume that $t = \langle [\ \ ], [T_1 = T_2].\mathcal{P}, 2, a\rangle$, $[T_1 = T_2].\mathcal{P} \xrightarrow{p}_t [T_1 = a].\mathcal{P}$ and $p = \text{Prob}\big[T_2 \rightarrow_e a\big]$. But there exists $u = \langle [\ \ ], [T_1 = T_2].\mathcal{Q}, 2, a\rangle$ such that $\langle t, u\rangle \in \gamma_4$. Furthermore, $[T_1 = T_2].\mathcal{Q} \xrightarrow{p'}_u [T_2 = a].\mathcal{Q}$ with $p' = \text{Prob}\big[T_2 \rightarrow_e a\big]$. Additionally, the construction of B gives us that $\langle [T_1 = a].\mathcal{P}, [T_1 = a].\mathcal{Q}\rangle_{\gamma_2} \in B$. To complete the proof we use the arguments employed in case 2.

   Now, assume that $t = \langle [\ \ ], [T_1 = T_2].\mathcal{P}, 1, a\rangle$, $[T_1 = T_2].\mathcal{P} \xrightarrow{p}_t [a = T_2].\mathcal{P}$ and $p = \text{Prob}\big[T_1 \rightarrow_e a\big]$. But there exists $u = \langle [\ \ ], [T_1 = T_2].\mathcal{Q}, 1, a\rangle$ such that $\langle t, u\rangle \in \gamma_4$. Furthermore, $[T_1 = T_2].\mathcal{Q} \xrightarrow{p'}_u [a = T_2].\mathcal{Q}$ with $p' = \text{Prob}\big[T_1 \rightarrow_e a\big]$. Additionally, the construction of B gives us that $\langle [a = T_2].\mathcal{P}, [a = T_2].\mathcal{Q}\rangle_{\gamma_3} \in B$. To complete the proof we use the arguments employed in case 3.

A symmetric argument allows us to complete the proof and hence B is a probabilistic bisimulation. $\square$

Lemma 5.62. *Let $\mathcal{P}$, $\mathcal{Q}$ and $\mathcal{R}$ be process families with no free variables such that $\mathcal{P} \sim \mathcal{Q}$. Then $\mathcal{P} \mid \mathcal{R} \sim \mathcal{Q} \mid \mathcal{R}$.*

Proof. The proof proceeds in two stages. In the first stage we construct a correspondence between transitions of $\mathcal{P} \mid \mathcal{R}$ and $\mathcal{Q} \mid \mathcal{R}$. In the second stage, we use the correspondence obtained in the first stage to construct a suitable probabilistic bisimulation.

Let $\mathcal{P}$, $\mathcal{Q}$ and $\mathcal{R}$ be process families with no free variables. We define $\gamma_{(\mathcal{P} \mid \mathcal{R}, \mathcal{Q} \mid \mathcal{R})}$ where $\mathcal{P} \sim \mathcal{Q}$ as follows. Assume that there exists $t \in T(\mathcal{P} \mid \mathcal{R})$ such that $\mathcal{P} \mid \mathcal{R} \xrightarrow{p}_t \mathcal{P}' \mid \mathcal{R}$. Then, by lemma 5.40, there exists a transition $t' \in T(\mathcal{P})$ such that $\mathcal{P} \xrightarrow{p}_{t'} \mathcal{P}'$. Since $\mathcal{P} \sim \mathcal{Q}$ there exists a corresponding transition $u' \in \mathcal{Q}$ such that $\mathcal{Q} \xrightarrow{p}_{u'} \mathcal{Q}'$. Another application of lemma 5.40 proves that there exists a transition $u \in T(\mathcal{Q} \mid \mathcal{R})$ such that $\mathcal{Q} \mid \mathcal{R} \xrightarrow{p}_u \mathcal{Q}' \mid \mathcal{R}$. We place $\langle t, u \rangle$ in $\gamma_{(\mathcal{P} \mid \mathcal{R}, \mathcal{Q} \mid \mathcal{R})}$.

Assume that there exists $t \in T(\mathcal{P} \mid \mathcal{R})$ such that $\mathcal{P} \mid \mathcal{R} \xrightarrow{p}_t \mathcal{P} \mid \mathcal{R}'$. Then, by lemma 5.40, there exists a transition $t' \in T(\mathcal{R})$ such that $\mathcal{R} \xrightarrow{p}_{t'} \mathcal{R}'$. Again we apply lemma 5.40 and yield that there exists a transition $u \in T(\mathcal{Q} \mid \mathcal{R})$ such that $\mathcal{Q} \mid \mathcal{R} \xrightarrow{p}_u \mathcal{Q} \mid \mathcal{R}'$. We place $\langle t, u \rangle$ in $\gamma_{(\mathcal{P} \mid \mathcal{R}, \mathcal{Q} \mid \mathcal{R})}$.

Assume that there exists $t \in T(\mathcal{P} \mid \mathcal{R})$ such that $\mathcal{P} \mid \mathcal{R} \xrightarrow{p}_t \mathcal{P}' \mid \mathcal{R}'$. Then, by lemma 5.40, there exists a transition $t'_1 \in T(\mathcal{P})$ such that $\mathcal{P} \xrightarrow{p}_{t'_1} \mathcal{P}'$ and a transition $t'_2 \in T(\mathcal{R})$ such that $\mathcal{R} \xrightarrow{p}_{t'_2} \mathcal{R}'$. Since $\mathcal{P} \sim \mathcal{Q}$ there exists a corresponding transition $u'_1 \in \mathcal{Q}$ such that $\mathcal{Q} \xrightarrow{p}_{u'_1} \mathcal{Q}'$. Lemma 5.40 then allows us to conclude that there exists a transition $u \in T(\mathcal{Q} \mid \mathcal{R})$ such that $\mathcal{Q} \mid \mathcal{R} \xrightarrow{p}_u \mathcal{Q}' \mid \mathcal{R}'$. We place $\langle t, u \rangle$ in $\gamma_{(\mathcal{P} \mid \mathcal{R}, \mathcal{Q} \mid \mathcal{R})}$.

It is easy to see that $\gamma_{(\mathcal{P} \mid \mathcal{R}, \mathcal{Q} \mid \mathcal{R})}$ is a correspondence between $\mathcal{P} \mid \mathcal{R}$ and $\mathcal{Q} \mid \mathcal{R}$.

We shall show that $B = \{\langle \mathcal{P} \mid \mathcal{R}, \mathcal{Q} \mid \mathcal{R} \rangle_{\gamma_{(\mathcal{P} \mid \mathcal{R}, \mathcal{Q} \mid \mathcal{R})}}\}$ where $\mathcal{P} \sim \mathcal{Q}$ and neither $\mathcal{P}$ nor $\mathcal{Q}$ nor $\mathcal{R}$ have any free variables is a probabilistic bisimulation. In order to do this, we take an arbitrary correspondence pair π from B and show that all its 'children' (meaning the pairs of process expressions one obtains by evaluating corresponding transitions on the two process families in π) are also correspondence pairs in B. Suppose that $\langle \mathcal{P} \mid \mathcal{R}, \mathcal{Q} \mid \mathcal{R} \rangle_{\gamma_{(\mathcal{P} \mid \mathcal{R}, \mathcal{Q} \mid \mathcal{R})}} \in B$.

There are now three cases to consider:

1. Assume that $\mathcal{P} \mid \mathcal{R} \xrightarrow{p}_t \mathcal{P}' \mid \mathcal{R}$. The correspondence $\gamma_{(\mathcal{P} \mid \mathcal{R}, \mathcal{Q} \mid \mathcal{R})}$ gives us a corresponding transition $u \in T(\mathcal{Q} \mid \mathcal{R})$ such that $\mathcal{Q} \mid \mathcal{R} \xrightarrow{p}_u \mathcal{Q}' \mid \mathcal{R}$. Finally we have that $\langle \mathcal{P}' \mid \mathcal{R}, \mathcal{Q}' \mid \mathcal{R} \rangle_{\gamma_{(\mathcal{P}' \mid \mathcal{R}, \mathcal{Q}' \mid \mathcal{R})}} \in B$.

2. Assume that $\mathcal{P} \mid \mathcal{R} \xrightarrow{p}_t \mathcal{P} \mid \mathcal{R}'$. We proceed as per the previous case.

3. Assume that $\mathcal{P} \mid \mathcal{R} \xrightarrow{p}_t \mathcal{P}' \mid \mathcal{R}'$. The correspondence $\gamma_{(\mathcal{P} \mid \mathcal{R}, \mathcal{Q} \mid \mathcal{R})}$ gives us a corresponding transition $u \in T(\mathcal{Q} \mid \mathcal{R})$ such that $\mathcal{Q} \mid \mathcal{R} \xrightarrow{p}_u \mathcal{Q}' \mid \mathcal{R}'$. Finally, we have that $\langle \mathcal{P}' \mid \mathcal{R}', \mathcal{Q}' \mid \mathcal{R}' \rangle_{\gamma_{(\mathcal{P}' \mid \mathcal{R}', \mathcal{Q}' \mid \mathcal{R}')}} \in B$.

A symmetric argument allows us to complete the proof and show that B is, indeed, a probabilistic bisimulation. $\square$

Lemma 5.63. *Let $\mathcal{P}$, $\mathcal{Q}$ and $\mathcal{R}$ be process families with no free variables such that $\mathcal{P} \sim \mathcal{Q}$. Then $\mathcal{R} \mid \mathcal{P} \sim \mathcal{R} \mid \mathcal{Q}$.*

Proof. As in lemma 5.62. $\square$

Lemma 5.64. *Let $\mathcal{P} \sim \mathcal{Q}$ be process families such that $!_{\eta(|n|)}.\mathcal{P}$ and $!_{\eta(|n|)}.\mathcal{Q}$ have no free variables. Then $!_{\eta(|n|)}.\mathcal{P} \sim !_{\eta(|n|)}.\mathcal{Q}$.*

Proof. Let $\mathcal{C}[\ \ ] \equiv !_{\eta(|n|)}.[\ \ ]$. We proceed via a induction on $\eta(|n|)$. If $\eta(|n|) = 0$ then it is easy to see that $\mathcal{C}[\mathcal{P}] \sim \mathcal{C}[\mathcal{Q}]$. Assume the result for $\eta(|n|) \leq k$. Let $\eta(|n|) = k + 1$. Then $\mathcal{C}[\mathcal{P}] \equiv (!_{\eta(|n|)-1}.\mathcal{P} \mid \mathcal{P})$ and $\mathcal{C}[\mathcal{Q}] \equiv (!_{\eta(|n|)-1}.\mathcal{Q} \mid \mathcal{Q})$ with $!_{\eta(|n|)-1}.\mathcal{P} \sim !_{\eta(|n|)-1}.\mathcal{Q}$ (the inductive hypothesis) and $\mathcal{P} \sim \mathcal{Q}$. Now, lemma 5.62 gives us that a probabilistic bisimulation B_1 exists establishing that $(!_{\eta(|n|)-1}.\mathcal{P} \mid \mathcal{P}) \sim (!_{\eta(|n|)-1}.\mathcal{Q} \mid \mathcal{P})$ and lemma 5.63 gives us that a probabilistic bisimulation B_2 exists establishing that $(!_{\eta(|n|)-1}.\mathcal{Q} \mid \mathcal{P}) \sim (!_{\eta(|n|)-1}.\mathcal{Q} \mid \mathcal{Q})$. Lemma 5.56 gives us that $B_1 B_2$ is a probabilistic bisimulation, thereby allowing us to establish the result. $\square$

Before we continue, let us recall several definitions. If $\mathcal{P}$ is a process family, then $F(\mathcal{P})$ is the set of free variable transitions for $\mathcal{P}$ (defn. 5.19) and $FV(\mathcal{P})$ is the set of free variables of $\mathcal{P}$. It is easy to see the following property.

Definition 5.65. Let $\mathcal{P}$ be a process family with free variables given by $FV(\mathcal{P})$ and let $X = \{x_1, \ldots, x_j\}$ be a tuple of variable names. Let $A = \{a_1, \ldots, a_i\}$ be a tuple set of atoms. We shall write $[A/X]\,\mathcal{P}$ to mean the process family $[a_1, \ldots, a_i/x_1, \ldots, x_j]\,\mathcal{P}$.

If $A = \emptyset$ or $X = \emptyset$ then $[A/X]\,\mathcal{P} = \mathcal{P}$. Furthermore, if $i < j$ then $[A/X]\,\mathcal{P} = [a_1, \ldots, a_i/x_1, \ldots, x_j]\,\mathcal{P} = [a_1, \ldots, a_i/x_1, \ldots, x_i]\,\mathcal{P}$ and if $i > j$ then $[A/X]\,\mathcal{P} = [a_1, \ldots, a_i/x_1, \ldots, x_j]\,\mathcal{P} = [a_1, \ldots, a_j/x_1, \ldots, x_j]\,\mathcal{P}$. Finally, let $X' \subseteq X$ be a tuple of variables that do not appear free in $\mathcal{P}$ and let A' be a tuple such that $a_k \in A' \iff x_k \in X'$. Then $[A/X]\,\mathcal{P} = [A - A'/X - X']\,\mathcal{P}$.

Lemma 5.66. *Let $\mathcal{P}$ be a process family and let $\mathcal{C}[\ \ ]$ be a context family. Then $[A/X]\,\mathcal{C}[\mathcal{P}] \equiv \mathcal{C}'[\mathcal{P}']$ where:*

1. *$\mathcal{C}'[\ \ ] \equiv [A/X]\,\mathcal{C}[\ \ ]$,*
2. *$\mathcal{P}' \equiv [A' \subseteq A/X' \subseteq X]\,\mathcal{P}$ where for each x_i in $X - X'$ there exist contexts $\mathcal{D}_i[\ \ ]$ and $\mathcal{E}_i[\ \ ]$ such that $\mathcal{C}[\ \ ] \equiv \mathcal{D}_i[c_{iq_i(|n|)}(x_i).\mathcal{E}_i[\ \ ]]$ (i.e., we only perform the substitution for those variables of X that are not bound to an input expression of $\mathcal{C}[\ \ ]$ whose scope contains $\mathcal{P}$).*

Proof. A straightforward induction on the structure of contexts, especially if one recalls that all variables are α-renamed apart. $\square$

The main lemma of this section states that $\sim$ is a congruence with respect to adversarial contexts.

Lemma 5.67. *For all process families $\mathcal{P}$ and $\mathcal{Q}$ we have that $\mathcal{P} \sim \mathcal{Q}$ implies $\mathcal{C}[\mathcal{P}] \sim \mathcal{C}[\mathcal{Q}]$ for all adversarial context families $\mathcal{C}[\ \ ]$.*

Proof. We proceed by an induction on the structure of adversarial context families. The basis case occurs when $\mathcal{C}[\ \ ] \equiv [\ \ ]_i$ and it follows trivially that $[\mathcal{P}]_i \sim [\mathcal{Q}]_i$. We now wish to show that $\mathcal{C}[\mathcal{P}] \sim \mathcal{C}[\mathcal{Q}]$ for all adversarial context families $\mathcal{C}[\ \ ]$. So we assume, as our inductive hypothesis, that for all $\mathcal{P}$ and $\mathcal{Q}$ such that $\mathcal{P} \sim \mathcal{Q}$ and for all adversarial context families $\mathcal{D}[\ \ ]$ that are syntactically shorter than $\mathcal{C}[\ \ ]$ we can show $\mathcal{D}[\mathcal{P}] \sim \mathcal{D}[\mathcal{Q}]$. By lemma 5.59 we then have that $FV(\mathcal{D}[\mathcal{P}]) = FV(\mathcal{D}[\mathcal{Q}])$.

We now proceed by induction on $|FV(\mathcal{C}[\mathcal{P}])|$. The basis occurs when $k = 0$. Then we have that $\mathcal{C}[\mathcal{P}]$ and $\mathcal{C}[\mathcal{Q}]$ have no free variables (by an application of lemma 5.59). There are five cases:

1. $\mathcal{C}[\ \ ] \equiv c_{q(|n|)}(x).\mathcal{D}[\ \ ]$. In this case we simply apply lemma 5.60 to obtain the desired result.

2. $\mathcal{C}[\ \ ] \equiv [T_1 = T_2].\mathcal{D}[\ \ ]$. In this case we simply apply lemma 5.61 to obtain the desired result.

3. $\mathcal{C}[\ \ ] \equiv \mathcal{D}[\ \ ] \mid \mathcal{R}$. In this case we simply apply lemma 5.62 to obtain the desired result.

4. $\mathcal{C}[\ \ ] \equiv \mathcal{R} \mid \mathcal{D}[\ \ ]$. In this case we simply apply lemma 5.63 to obtain the desired result.

5. $\mathcal{C}[\ \ ] \equiv !_{q(|n|)}.\mathcal{D}[\ \ ]$. In this case we simply apply lemma 5.64 to obtain the desired result.

So much for the basis. We now assume, as our second inductive hypothesis, that for all $\mathcal{P}$ and $\mathcal{Q}$ if $\mathcal{P} \sim \mathcal{Q}$ we can show $\mathcal{C}[\mathcal{P}] \sim \mathcal{C}[\mathcal{Q}]$ for all adversarial context families $\mathcal{C}[\ \ ]$ where $\mathcal{C}[\mathcal{P}]$ has at most n free variables (and, by lemma 5.59, we also know that $\mathcal{C}[\mathcal{Q}]$ must have at most n free variables).

Now, let $\mathcal{C}[\ \ ]$ be an adversarial context family such that $\mathcal{C}[\mathcal{P}]$ has at most $n+1$ free variables. We now define a set of process families $\mathcal{P}^+$ as $\{[a/x]\,\mathcal{C}[\mathcal{P}] \,|\, x \in FV(\mathcal{C}[\mathcal{P}]), a \in \mathbb{N}\}$ and we define a set of process families $\mathcal{Q}^+$ as $\{[a/x]\,\mathcal{C}[\mathcal{Q}] \,|\, x \in FV(\mathcal{C}[\mathcal{Q}]), a \in \mathbb{N}\}$. Clearly $[a/x]\,\mathcal{C}[\mathcal{P}] \in \mathcal{P}^+ \iff [a/x]\,\mathcal{C}[\mathcal{Q}] \in \mathcal{Q}^+$. Also $[a/x]\,\mathcal{C}[\mathcal{P}] \equiv \mathcal{C}'[\mathcal{P}']$ (by lemma 5.66) and $[a/x]\,\mathcal{C}[\mathcal{Q}] \equiv \mathcal{C}'[\mathcal{Q}']$ (again, by lemma 5.66) where $\mathcal{C}'[\ \ ]$ might be syntactically identical to $\mathcal{C}[\ \ ]$, $\mathcal{P}'$ might be syntactically identical to $\mathcal{P}$, and $\mathcal{Q}'$ might be syntactically identical to $\mathcal{Q}$. Whence, by the second inductive hypothesis, we have that $[a/x]\,\mathcal{C}[\mathcal{P}] \sim [a/x]\,\mathcal{C}[\mathcal{Q}]$. Let $B_{a,x}$ be the probabilistic bisimulation that establishes $[a/x]\,\mathcal{C}[\mathcal{P}] \sim [a/x]\,\mathcal{C}[\mathcal{Q}]$. Define B^* as $\bigcup_{a,x} B_{a,x}$ where $[a/x]\,\mathcal{C}[\mathcal{P}] \in \mathcal{P}^+$ (and, consequently, $[a/x]\,\mathcal{C}[\mathcal{Q}] \in \mathcal{Q}^+$).

We will show that $B = \{\langle \mathcal{C}[\mathcal{P}], \mathcal{C}[\mathcal{Q}] \rangle_{id}\} \cup B^*$ is a probabilistic bisimulation. Clearly id is a correspondence between $T(\mathcal{C}[\mathcal{P}])$ and $T(\mathcal{C}[\mathcal{Q}])$.

Assume that $\mathcal{C}[\mathcal{P}] \xrightarrow{p}_t \mathcal{R}$. Now, t must be of the form $\langle x, a \rangle$ and $p = 1$. Clearly, we also have that $\mathcal{C}[\mathcal{Q}] \xrightarrow{1}_{\langle x,a \rangle} \mathcal{S}$. Furthermore, $\mathcal{R} \equiv [a/x]\,\mathcal{C}[\mathcal{P}] \equiv \mathcal{C}'[\mathcal{P}']$ (by lemma 5.66) and $\mathcal{S} \equiv [a/x]\,\mathcal{C}[\mathcal{Q}] \equiv \mathcal{C}'[\mathcal{Q}']$ (again, by lemma 5.66). So, $\mathcal{R}$ and

$\mathcal{S}$ each have the same n free variables (using lemma 5.59 and the fact that $\mathcal{P}' \sim \mathcal{Q}'$). Hence, by the second inductive hypothesis, there must be a probabilistic bisimulation $B_{a,x}$ such that $B_{a,x}$ establishes that $\mathcal{R} \sim \mathcal{S}$. Hence there must exist a correspondence δ between $T(\mathcal{R})$ and $T(\mathcal{S})$ such that $\langle \mathcal{R}, \mathcal{S} \rangle_\delta \in B$ (the construction of B ensures that $B_{a,x} \subseteq B$). A symmetric argument allows us to complete the inductive step and show that B is a probabilistic bisimulation. $\square$

Furthermore

Lemma 5.68. *Let $\mathcal{P}$ and $\mathcal{Q}$ be process families such that $\mathcal{P} \sim \mathcal{Q}$. Then $|E(\mathcal{P})| = |E(\mathcal{Q})|$.*

Proof. Since $\mathcal{P} \sim \mathcal{Q}$, there exists a correspondence γ between $\mathcal{P}$ and $\mathcal{Q}$ such that every transition in $A(\mathcal{P})$ has a corresponding transition in $A(\mathcal{Q})$. Additionally, $\mathcal{P} \sim \mathcal{Q}$ implies that at most one $t \in A(\mathcal{P})$ maps to each $t' \in A(\mathcal{Q})$. Whence $|E(\mathcal{Q})| \geq |E(\mathcal{P})|$. Similarly, $\mathcal{P} \sim \mathcal{Q}$ allows us to conclude that $|E(\mathcal{P})| \geq |E(\mathcal{Q})|$ whence $|E(\mathcal{P})| = |E(\mathcal{Q})|$. $\square$

We will relate probabilistic bisimulation to our asymptotic notion of observational equivalence stated in definition 5.6. First we need the following two definitions.

Definition 5.69. Let $a \equiv \langle \mathcal{A}_1, \mathcal{A}_2, \mathcal{C}[\ ][\ ] \rangle$ and $b \equiv \langle \mathcal{B}_1, \mathcal{B}_2, \mathcal{D}[\ ][\ ] \rangle$ be macro-steps. We say that a and b are *associated by a probabilistic bisimulation*, written $a \sim b$, just when there exists a probabilistic bisimulation B and a correspondence γ between $\mathcal{C}[\mathcal{A}_1, \mathcal{A}_2]$ and $\mathcal{D}[\mathcal{B}_1, \mathcal{B}_2]$ such that $\langle \mathcal{C}[\mathcal{A}_1, \mathcal{A}_2], \mathcal{D}[\mathcal{B}_1, \mathcal{B}_2] \rangle_\gamma \in B$ and $\langle a, b \rangle \in \gamma$.

Definition 5.70. Let C and D be sets of macro-steps such that $|C| = |D|$ and let S be a scheduler. Let $c \in C$ and $d \in D$ be macro-steps such $c \sim d$. We will say that S is *weakly uniform* if and only if the probability that S picks c from C is the same as the probability that S picks d from D.

Definition 5.71. Let $\mathbf{W}$ be the set of all weakly uniform schedulers.

Definition 5.72. Let $\mathcal{P}$ be a process family. An *eligible path* in $G_{\mathcal{P}}$ (the labelled transition system for $\mathcal{P}$) is a path from the root node to a terminal node that consists only of transitions in $A(\mathcal{P})$ or $OE(\mathcal{P})$ (*i.e.,* only eligible communications and outer-evaluations).

Now we can show

Lemma 5.73. *For all process families $\mathcal{P}$ and $\mathcal{Q}$, if $\mathcal{P} \sim \mathcal{Q}$ then for all weakly uniform schedulers S, for all positive polynomials q we have that:*

$$\forall o \in Obs. \exists n_o. \forall n > n_o : \left| \ Prob\left[P^n \rightsquigarrow_S o\right] - Prob\left[Q^n \rightsquigarrow_S o\right] \right| \leq \frac{1}{q(n)}$$

Proof. Unlike execution graphs, the only way that a process family $\mathcal{P}$ can produce an observable $o = \langle i, c_{p(|n|)} \rangle$ is if there is a sequence of transitions on a path made up of only private communications or actual public communications that produces i on the public channel $c_{p(|n|)}$, *i.e.,* we do not care about evaluation paths that make use of transitions that depend on the benevolent context discussed in section 5.2.1. Hence, every path in the evaluation tree of $\mathcal{P}$ is a path in the execution graph, $G_{\mathcal{P}}$, for $\mathcal{P}$. To factor in the probabilistic effects caused by the scheduler, we need only consider transitions in $A(\mathcal{P})$ (those transitions such that the input and output parts are both sub-expressions of $\mathcal{P}$). These are the only transitions which depend on the scheduler and, consequently, if we show that the scheduler behaves the same on $\mathcal{P}$ and $\mathcal{Q}$, we will be able to establish the desired result.

Let $\mathcal{R}$ be a process family derived from $\mathcal{P}$ via some number (possibly zero) of transition steps. Since $\mathcal{P} \sim \mathcal{Q}$ we have that there exists $\mathcal{S}$ derived from $\mathcal{Q}$ such that $\mathcal{R} \sim \mathcal{S}$. Lemma 5.68 tells us that $|E(\mathcal{R})| = |E(\mathcal{S})|$. Since the scheduler is weakly uniform, the probability of a particular macro-step occurring (*i.e.,* the probability of selecting a particular transition in $A(\mathcal{R})$ during the evaluation of $\mathcal{R}$) is the same as the probability of the corresponding macro-step (by this we mean the macro-step defined by the corresponding transition) occurring during the evaluation of $\mathcal{S}$. Since $\mathcal{R} \sim \mathcal{S}$, the probability associated with any transition u in $T(\mathcal{R})$ is the same as the probability associated with u's corresponding transition. Now a macro-step is a sequence of outer-evaluations (represented by exposed match and exposed term transitions) followed by a communication step (represented by an eligible communication transition) followed by more outer-evaluations. Consequently, the probability of an eligible path in $G_{\mathcal{P}}$ is the same as the probability of the corresponding eligible path in $G_{\mathcal{Q}}$.

Now assume that $\mathcal{S}$ is obtained by evaluating some number of transitions on $\mathcal{Q}$. A similar argument gives us that there exists a $\mathcal{R}$ that is obtained from $\mathcal{P}$ via some number of transitions such that the probability of picking a particular macro-step during the evaluation of $\mathcal{S}$ is the same as the probability of picking the corresponding macro-step during the evaluation of $\mathcal{R}$; and, that the probability of an eligible path in $G_{\mathcal{Q}}$ is the same as the probability of the corresponding eligible path in $G_{\mathcal{P}}$.

Hence, every path in the evaluation of $\mathcal{P}$ that produces o will occur with the same probability in the evaluation of $\mathcal{Q}$ and *vice versa.* □

We can then immediately obtain

Theorem 5.74. $\mathcal{P} \sim \mathcal{Q} \implies \mathcal{P} \cong_{\mathbf{W}} \mathcal{Q}$.

Proof. Direct from lemmas 5.67 and 5.73. □

5.3 Basic properties of observational equivalence

We now relate $\cong$, $\cong_{\mathbf{W}}$ and $\cong_S$.

Lemma 5.75. *Let* $\mathbf{S}$ *be the set of all schedulers. Then the following hold:*

1. $\mathcal{P} \cong \mathcal{Q} \iff \forall S \in \mathbf{S} : \mathcal{P} \cong_S \mathcal{Q}$

2. $\mathcal{P} \cong_W \mathcal{Q} \iff \forall S \in \mathbf{W} : \mathcal{P} \cong_S \mathcal{Q}$

3. $\mathcal{P} \cong \mathcal{Q} \implies \mathcal{P} \cong_{\mathbf{W}} \mathcal{Q}$

4. $\mathcal{P} \cong \mathcal{Q} \implies \mathcal{P} \cong_S \mathcal{Q}$

5. $\mathcal{P} \cong_{\mathbf{W}} \mathcal{Q} \implies \exists S : \mathcal{P} \cong_S \mathcal{Q}$

Proof. Follow directly from the definitions of $\cong_S$, $\cong_{\mathbf{W}}$ and $\cong$. □

The technical groundwork laid in the previous section will now be used to prove several basic properties of observational equivalence, which establish its compositionality.

Basic Equivalence 5.76 (REFLEC-PARALLEL)**.** *$\mathcal{P} \mid \mathcal{Q} \cong_{\mathbf{W}} \mathcal{Q} \mid \mathcal{P}$.*

Proof. We define $\gamma_{(\mathcal{P} \mid \mathcal{Q}, \mathcal{Q} \mid \mathcal{P})}$. Assume that there exists $t \in T(\mathcal{P} \mid \mathcal{Q})$ such that $\mathcal{P} \mid \mathcal{Q} \xrightarrow{p}_t \mathcal{P}' \mid \mathcal{Q}$. Then, by lemma 5.40, there exists a transition $t' \in T(\mathcal{P})$ such that $\mathcal{P} \xrightarrow{p}_{t'} \mathcal{P}'$. Another application of lemma 5.40 allows us to conclude that there exists a transition $u \in T(\mathcal{Q} \mid \mathcal{P})$ such that $\mathcal{Q} \mid \mathcal{P} \xrightarrow{p}_u \mathcal{Q} \mid \mathcal{P}'$. We place $\langle t, u \rangle$ in $\gamma_{(\mathcal{P} \mid \mathcal{Q}, \mathcal{Q} \mid \mathcal{P})}$.

Assume that there exists $t \in T(\mathcal{P} \mid \mathcal{Q})$ such that $\mathcal{P} \mid \mathcal{Q} \xrightarrow{p}_t \mathcal{P} \mid \mathcal{Q}'$. Then, by lemma 5.40, there exists a transition $t' \in T(\mathcal{Q})$ such that $\mathcal{Q} \xrightarrow{p}_{t'} \mathcal{Q}'$. A second application shows that there exists a transition $u \in T(\mathcal{Q} \mid \mathcal{P})$ such that $\mathcal{Q} \mid \mathcal{P} \xrightarrow{p}_u \mathcal{Q}' \mid \mathcal{P}$. We place $\langle t, u \rangle$ in $\gamma_{(\mathcal{P} \mid \mathcal{Q}, \mathcal{Q} \mid \mathcal{P})}$.

Assume that there exists $t \in T(\mathcal{P} \mid \mathcal{Q})$ such that $\mathcal{P} \mid \mathcal{Q} \xrightarrow{p}_t \mathcal{P}' \mid \mathcal{Q}'$. Then, by lemma 5.40, there exists a transition $t'_1 \in T(\mathcal{P})$ such that $\mathcal{P} \xrightarrow{p}_{t'_1} \mathcal{P}'$ and a transition $t'_2 \in T(\mathcal{Q})$ such that $\mathcal{Q} \xrightarrow{p}_{t'_2} \mathcal{Q}'$. Another use of lemma 5.40 gives us that there exists a transition $u \in T(\mathcal{Q} \mid \mathcal{P})$ such that $\mathcal{Q} \mid \mathcal{P} \xrightarrow{p}_u \mathcal{Q}' \mid \mathcal{P}'$. We place $\langle t, u \rangle$ in $\gamma_{(\mathcal{P} \mid \mathcal{Q}, \mathcal{Q} \mid \mathcal{P})}$.

It is easy to see that $\gamma_{(\mathcal{P} \mid \mathcal{Q}, \mathcal{Q} \mid \mathcal{P})}$ is a correspondence between $\mathcal{P} \mid \mathcal{Q}$ and $\mathcal{Q} \mid \mathcal{P}$.

We shall show that $B = \{\langle \mathcal{P} \mid \mathcal{Q}, \mathcal{Q} \mid \mathcal{P} \rangle_{\gamma_{(\mathcal{P} \mid \mathcal{Q}, \mathcal{Q} \mid \mathcal{P})}}\}$ is a probabilistic bisimulation. Suppose that $\langle \mathcal{P} \mid \mathcal{Q}, \mathcal{Q} \mid \mathcal{P} \rangle_{\gamma_{(\mathcal{P} \mid \mathcal{Q}, \mathcal{Q} \mid \mathcal{P})}} \in B$.

There are now three cases to consider:

1. Assume that $\mathcal{P} \mid \mathcal{Q} \xrightarrow{p}_t \mathcal{P}' \mid \mathcal{Q}$. The correspondence $\gamma_{(\mathcal{P} \mid \mathcal{Q}, \mathcal{Q} \mid \mathcal{P})}$ gives us a corresponding transition $u \in T(\mathcal{Q} \mid \mathcal{P})$ such that $\mathcal{Q} \mid \mathcal{P} \xrightarrow{p}_u \mathcal{Q} \mid \mathcal{P}'$. Finally we have that $\langle \mathcal{P}' \mid \mathcal{Q}, \mathcal{Q} \mid \mathcal{P}' \rangle_{\gamma_{(\mathcal{P}' \mid \mathcal{Q}, \mathcal{Q} \mid \mathcal{P}')}} \in B$.

2. Assume that $\mathcal{P} \mid \mathcal{Q} \xrightarrow{p}_t \mathcal{P} \mid \mathcal{Q}'$. We proceed as per the previous case.

3. Assume that $\mathcal{P} \mid \mathcal{Q} \xrightarrow{p}_t \mathcal{P}' \mid \mathcal{Q}'$. The correspondence $\gamma_{(\mathcal{P} \mid \mathcal{Q}, \mathcal{Q} \mid \mathcal{P})}$ gives us a corresponding transition $u \in T(\mathcal{Q} \mid \mathcal{P})$ such that $\mathcal{Q} \mid \mathcal{P} \xrightarrow{p}_u \mathcal{Q}' \mid \mathcal{P}'$. Finally we have that $\langle \mathcal{P}' \mid \mathcal{Q}', \mathcal{Q}' \mid \mathcal{P}' \rangle_{\gamma_{(\mathcal{P}' \mid \mathcal{Q}', \mathcal{Q}' \mid \mathcal{P}')}} \in B$.

A symmetric argument allows us to complete the proof and show that B is, indeed, a probabilistic bisimulation, whence, by lemma 5.74, we have that $\mathcal{P} \mid \mathcal{Q} \cong_{\mathbf{W}} \mathcal{Q} \mid \mathcal{P}$. $\square$

The following two properties are established similarly.

Basic Equivalence 5.77 (`NOOP`). $\mathcal{P} \mid \mathbf{0} \cong_{\mathbf{W}} \mathcal{P} \cong \mathbf{0} \mid \mathcal{P}$.

Proof. We will only show that $\mathcal{P} \mid \mathbf{0} \cong_{\mathbf{W}} \mathcal{P}$. We complete the proof using the transitivity of $\cong_{\mathbf{W}}$ and `REFLEC-PARALLEL`.

We define $\gamma_{(\mathcal{P} \mid \mathbf{0}, \mathcal{P})}$. Assume that there exists $t \in T(\mathcal{P} \mid \mathbf{0})$ such that $\mathcal{P} \mid \mathbf{0} \xrightarrow{p}_t \mathcal{P}' \mid \mathbf{0}$. Then, by lemma 5.40, there exists a transition $u \in T(\mathcal{P})$ such that $\mathcal{P} \xrightarrow{p}_u \mathcal{P}'$. We place $\langle t, u \rangle$ in $\gamma_{(\mathcal{P} \mid \mathbf{0}, \mathcal{P})}$.

Since every transition in $T(\mathcal{P} \mid \mathbf{0})$ affects only $\mathcal{P}$, it is easy to see that $\gamma_{(\mathcal{P} \mid \mathbf{0}, \mathcal{P})}$ is a correspondence between $\mathcal{P} \mid \mathbf{0}$ and $\mathcal{P}$.

We shall show that $B = \{\langle \mathcal{P} \mid \mathbf{0}, \mathcal{P} \rangle_{\gamma_{(\mathcal{P} \mid \mathbf{0}, \mathcal{P})}}\}$ is a probabilistic bisimulation. Suppose that $\langle \mathcal{P} \mid \mathbf{0}, \mathcal{P} \rangle_{\gamma_{(\mathcal{P} \mid \mathbf{0}, \mathcal{P})}} \in B$. Assume that $\mathcal{P} \mid \mathbf{0} \xrightarrow{p}_t \mathcal{P}' \mid \mathbf{0}$. The correspondence $\gamma_{(\mathcal{P} \mid \mathbf{0}, \mathcal{P})}$ gives us a corresponding transition $u \in T(\mathcal{P})$ such that $\mathcal{P} \xrightarrow{p}_u \mathcal{P}'$. Finally we have that $\langle \mathcal{P}' \mid \mathbf{0}, \mathcal{P}' \rangle_{\gamma_{(\mathcal{P}' \mid \mathbf{0}, \mathcal{P}')}} \in B$.

A symmetric argument allows us to complete the proof and show that B is, indeed, a probabilistic bisimulation, whence, by lemma 5.74, we have that $\mathcal{P} \mid \mathbf{0} \cong_{\mathbf{W}} \mathcal{P}$.

Using `REFLEC-PARALLEL` we obtain that $\mathcal{P} \mid \mathbf{0} \cong_{\mathbf{W}} \mathbf{0} \mid \mathcal{P}$. Transitivity gives us the desired result. $\square$

Basic Equivalence 5.78 (`PARALLEL-COMM`). $(\mathcal{P} \mid \mathcal{Q}) \mid \mathcal{R} \cong_{\mathbf{W}} \mathcal{P} \mid (\mathcal{Q} \mid \mathcal{R})$.

Proof. We define $\gamma_{((\mathcal{P} \mid \mathcal{Q}) \mid \mathcal{R}, \mathcal{P} \mid (\mathcal{Q} \mid \mathcal{R}))}$. Let there exist $t \in T((\mathcal{P} \mid \mathcal{Q}) \mid \mathcal{R})$ such that $(\mathcal{P} \mid \mathcal{Q}) \mid \mathcal{R} \xrightarrow{p}_t (\mathcal{P}' \mid \mathcal{Q}) \mid \mathcal{R}$. Then, using lemma 5.40 twice, we can find a transition $u \in T(\mathcal{P} \mid (\mathcal{Q} \mid \mathcal{R}))$ such that $\mathcal{P} \mid (\mathcal{Q} \mid \mathcal{R}) \xrightarrow{p}_u \mathcal{P}' \mid (\mathcal{Q} \mid \mathcal{R})$. We place $\langle t, u \rangle$ in $\gamma_{((\mathcal{P} \mid \mathcal{Q}) \mid \mathcal{R}, \mathcal{P} \mid (\mathcal{Q} \mid \mathcal{R}))}$. The cases when $(\mathcal{P} \mid \mathcal{Q}) \mid \mathcal{R} \xrightarrow{p}_t (\mathcal{P} \mid \mathcal{Q}') \mid \mathcal{R}$ and $(\mathcal{P} \mid \mathcal{Q}) \mid \mathcal{R} \xrightarrow{p}_t (\mathcal{P} \mid \mathcal{Q}) \mid \mathcal{R}'$ are handled in a similar manner.

Assume that there exists $t \in T((\mathcal{P} \mid \mathcal{Q}) \mid \mathcal{R}$ such that $(\mathcal{P} \mid \mathcal{Q}) \mid \mathcal{R} \xrightarrow{p}_t (\mathcal{P}' \mid \mathcal{Q}') \mid \mathcal{R}$. Then, using lemmas 5.40 twice, we can find a transition $u \in T(\mathcal{P} \mid (\mathcal{Q} \mid \mathcal{R}))$ such that $\mathcal{P} \mid (\mathcal{Q} \mid \mathcal{R}) \xrightarrow{p}_u \mathcal{P}' \mid (\mathcal{Q}' \mid \mathcal{R})$. We place $\langle t, u \rangle$ in $\gamma_{((\mathcal{P} \mid \mathcal{Q}) \mid \mathcal{R}, \mathcal{P} \mid (\mathcal{Q} \mid \mathcal{R}))}$. The cases when $(\mathcal{P} \mid \mathcal{Q}) \mid \mathcal{R} \xrightarrow{p}_t (\mathcal{P} \mid \mathcal{Q}') \mid \mathcal{R}'$ and $(\mathcal{P} \mid \mathcal{Q}) \mid \mathcal{R} \xrightarrow{p}_t (\mathcal{P}' \mid \mathcal{Q}) \mid \mathcal{R}'$ are handled in a similar manner.

It is easy to see that $\gamma_{((\mathcal{P} \mid \mathcal{Q}) \mid \mathcal{R}, \mathcal{P} \mid (\mathcal{Q} \mid \mathcal{R}))}$ is a correspondence between $(\mathcal{P} \mid \mathcal{Q}) \mid \mathcal{R}$ and $\mathcal{P} \mid (\mathcal{Q} \mid \mathcal{R})$.

We shall show that $B = \{\langle (\mathcal{P} \mid \mathcal{Q}) \mid \mathcal{R}, \mathcal{P} \mid (\mathcal{Q} \mid \mathcal{R}) \rangle_{\gamma_{((\mathcal{P} \mid \mathcal{Q}) \mid \mathcal{R}, \mathcal{P} \mid (\mathcal{Q} \mid \mathcal{R}))}}\}$ is a probabilistic bisimulation.

Suppose that $\langle (\mathcal{P} \mid \mathcal{Q}) \mid \mathcal{R}, \mathcal{P} \mid (\mathcal{Q} \mid \mathcal{R}) \rangle_{\gamma_{((\mathcal{P} \mid \mathcal{Q}) \mid \mathcal{R}, \mathcal{P} \mid (\mathcal{Q} \mid \mathcal{R}))}} \in B$. There are six cases to consider:

1. Assume $(\mathcal{P} \mid \mathcal{Q}) \mid \mathcal{R} \xrightarrow{p}_t (\mathcal{P}' \mid \mathcal{Q}) \mid \mathcal{R}$. $\gamma_{((\mathcal{P} \mid \mathcal{Q}) \mid \mathcal{R}, \mathcal{P} \mid (\mathcal{Q} \mid \mathcal{R}))}$ gives us a corresponding transition $u \in T(\mathcal{P} \mid (\mathcal{Q} \mid \mathcal{R}))$ such that $\mathcal{P} \mid (\mathcal{Q} \mid \mathcal{R}) \xrightarrow{p}_u \mathcal{P}' \mid (\mathcal{Q} \mid \mathcal{R})$. Finally, we note that:

$$\langle (\mathcal{P}' \mid \mathcal{Q}) \mid \mathcal{R}, \mathcal{P}' \mid (\mathcal{Q} \mid \mathcal{R}) \rangle_{\gamma_{((\mathcal{P}' \mid \mathcal{Q}) \mid \mathcal{R}, \mathcal{P}' \mid (\mathcal{Q} \mid \mathcal{R}))}} \in B$$

2. Assume $(\mathcal{P} \mid \mathcal{Q}) \mid \mathcal{R} \xrightarrow{p}_t (\mathcal{P} \mid \mathcal{Q}') \mid \mathcal{R}$. We argue this as we did the previous case.

3. Assume $(\mathcal{P} \mid \mathcal{Q}) \mid \mathcal{R} \xrightarrow{p}_t (\mathcal{P} \mid \mathcal{Q}) \mid \mathcal{R}'$. We argue this as we did the first case.

4. Assume $(\mathcal{P} \mid \mathcal{Q}) \mid \mathcal{R} \xrightarrow{p}_t (\mathcal{P}' \mid \mathcal{Q}') \mid \mathcal{R}$. $\gamma_{((\mathcal{P} \mid \mathcal{Q}) \mid \mathcal{R}, \mathcal{P} \mid (\mathcal{Q} \mid \mathcal{R}))}$ gives us a corresponding transition $u \in T(\mathcal{P} \mid (\mathcal{Q} \mid \mathcal{R}))$ such that $\mathcal{P} \mid (\mathcal{Q} \mid \mathcal{R}) \xrightarrow{p}_u \mathcal{P}' \mid (\mathcal{Q}' \mid \mathcal{R})$. Finally, we note that:

$$\langle (\mathcal{P}' \mid \mathcal{Q}') \mid \mathcal{R}, \mathcal{P}' \mid (\mathcal{Q}' \mid \mathcal{R}) \rangle_{\gamma_{((\mathcal{P}' \mid \mathcal{Q}') \mid \mathcal{R}, \mathcal{P}' \mid (\mathcal{Q}' \mid \mathcal{R}))}} \in B$$

5. Assume $(\mathcal{P} \mid \mathcal{Q}) \mid \mathcal{R} \xrightarrow{p}_t (\mathcal{P} \mid \mathcal{Q}') \mid \mathcal{R}'$. We argue this as we did the previous case.

6. Assume $(\mathcal{P} \mid \mathcal{Q}) \mid \mathcal{R} \xrightarrow{p}_t (\mathcal{P}' \mid \mathcal{Q}) \mid \mathcal{R}'$. We argue this as we did the fourth case.

A symmetric argument allows us to complete the proof and show that B is, indeed, a probabilistic bisimulation, whence, by lemma 5.74, we have that $(\mathcal{P} \mid \mathcal{Q}) \mid \mathcal{R} \cong_{\mathbf{W}} \mathcal{P} \mid (\mathcal{Q} \mid \mathcal{R})$. □

Another basic property of observational equivalence involves the relationship among the probabilistic polynomial-time computable functions, the terms of our calculus, and the polynomial-time oracle Turing machines, described in section 2.3.1.

Equivalence 5.79 (`EQ-DISTRIB`). *Let f and g be probabilistic polynomial-time computable functions such that range(f) = range$(g) = X \subseteq \mathbb{N}$ and both f and g induce the same distribution on X. Let T_f be a non-atomic term such that M_{T_f} computes f and T_g be a non-atomic term such that M_{T_g} computes g. Then for any polynomial q we have $c_{q(|n|)}\langle T_f \rangle \cong_{\mathbf{W}} c_{q(|n|)}\langle T_g \rangle$.*

Proof. As usual, we proceed by constructing a bisimulation between $c_{q(|n|)}\langle T_f \rangle$ and $c_{q(|n|)}\langle T_g \rangle$. Define γ as $\{\langle\langle [\ ], c_{q(|n|)}\langle T_f \rangle, a\rangle, \langle [\ ], c_{q(|n|)}\langle T_g \rangle, a\rangle\rangle \mid T_f \xrightarrow{p>0}_e a\}$. From the definition of T_f and T_g we know that $T_f \xrightarrow{p>0}_e a \iff T_g \xrightarrow{p}_e a$. It is clear that γ is a correspondence between $c_{q(|n|)}\langle T_f \rangle$ and $c_{q(|n|)}\langle T_g \rangle$. Define δ_a as $\{\langle\langle c_{q(|n|)}\langle a \rangle, [\ ]\rangle, \langle c_{q(|n|)}\langle a \rangle, [\ ]\rangle\rangle\}$ where $a \in \mathbb{N}$. It is clear that δ_a is a correspondence (specifically the correspondence id) between $c_{q(|n|)}\langle a \rangle$ and $c_{q(|n|)}\langle a \rangle$. Define a set of correspondence pairs B as:

$$\bigcup_{a \in \mathbb{N}} \{\langle c_{q(|n|)}\langle a \rangle, c_{q(|n|)}\langle a \rangle \rangle_{\delta_a}\} \ \cup\ \{\langle c_{q(|n|)}\langle T_f \rangle, c_{q(|n|)}\langle T_g \rangle\rangle_\gamma\} \ \cup\ \{\langle \mathbf{0}, \mathbf{0} \rangle_\emptyset\}$$

We now show that B is indeed a bisimulation between $c_{q(|n|)}\langle T_f\rangle$ and $c_{q(|n|)}\langle T_g\rangle$. We start by showing that for any $\langle \mathcal{P}, \mathcal{Q}\rangle_\alpha \in B$ if $\mathcal{P} \xrightarrow{p}_t \mathcal{P}'$ then there exists a $\mathcal{Q}'$ such that $\mathcal{Q} \xrightarrow{p}_u \mathcal{Q}'$ with $\langle \mathcal{P}', \mathcal{Q}'\rangle_\beta \in B$.

Consider $\langle c_{q(|n|)}\langle T_f\rangle, c_{q(|n|)}\langle T_g\rangle\rangle_\gamma \in B$. Let $c_{q(|n|)}\langle T_f\rangle \xrightarrow{p}_t c_{q(|n|)}\langle a\rangle$. Then, the correspondence γ gives us a corresponding transition $u \in T(c_{q(|n|)}\langle T_g\rangle)$ such that $c_{q(|n|)}\langle T_g\rangle \xrightarrow{p}_u c_{q(|n|)}\langle a\rangle$. Finally $\langle c_{q(|n|)}\langle a\rangle, c_{q(|n|)}\langle a\rangle\rangle_\delta \in B$ (by construction). A symmetric argument shows that if $c_{q(|n|)}\langle T_g\rangle \xrightarrow{p}_u c_{q(|n|)}\langle a\rangle$, then, using the correspondence γ we can obtain a corresponding transition t such that $c_{q(|n|)}\langle T_f\rangle \xrightarrow{p}_t c_{q(|n|)}\langle a\rangle$ and, furthermore, $\langle c_{q(|n|)}\langle a\rangle, c_{q(|n|)}\langle a\rangle\rangle_{\delta_a} \in B$.

We now consider $\langle c_{q(|n|)}\langle a\rangle, c_{q(|n|)}\langle a\rangle\rangle_{\delta_a} \in B$. Let $c_{q(|n|)}\langle a\rangle \xrightarrow{p}_t \mathbf{0}$. Then, the correspondence δ_a yields a corresponding transition $u \in T(c_{q(|n|)}\langle a\rangle)$ such that $c_{q(|n|)}\langle a\rangle \xrightarrow{p}_u \mathbf{0}$. Finally $\langle \mathbf{0}, \mathbf{0}\rangle_\emptyset \in B$ (by construction).

We complete the proof that B is a probabilistic bisimulation by noting that $T(\mathbf{0}) = \emptyset$. Since $\langle c_{q(|n|)}\langle T_f\rangle, c_{q(|n|)}\langle T_g\rangle\rangle_\gamma \in B$ we can use lemma 5.74 to establish the desired result. □

6 Cryptographic examples

Let us show that our asymptotic notion of observational equivalence between probabilistic polynomial-time processes coincides with the traditional notion of *indistinguishability by polynomial-time statistical tests*, a standard way of characterizing cryptographically strong pseudorandom number generators [Yao82, Gol00, Gol99, Lub96, Gen00].

6.1 Pseudorandom Number Generators

For the remainder of this section, let S refer to an arbitrary scheduler. We begin by recalling several standard notions from cryptographic literature [Yao82, Gol00, Gol99, Lub96, Gen00].

Definition 6.1 (function ensemble). A *function ensemble* f is an indexed family of functions $\{f_n : A_n \to B_n\}_{n\in\mathbb{N}}$.

The reader might wish to review defns. 2.5, 2.6, 2.7, 2.8 and 2.9 for details before proceeding.

Definition 6.2. A function ensemble $f : A_n \to B_n$ is *uniform* if there exists a single Turing machine M that computes f for all values of n, *i.e.,* $M(n, x) = f_n(x)$.

Definition 6.3. A uniform function ensemble $f : A_n \to B_n$ is *poly-time* if there exists a polynomial q and a single Turing machine M such that $M(n, x)$ computes $f_n(x)$ in time at most $q(|n|, |x|)$.

Definition 6.4. A uniform function ensemble $f : A_n \times B_n \to [0, 1]$ is *probabilistic poly-time* if there exists a single probabilistic poly-time Turing machine M such that $M(n)$ computes f_n.

Definition 6.5. A *poly-time statistical test* $\mathcal{A} : \{0,1\}^{m(|n|)} \times \{0,1\} \to [0,1]$ is a $\{0,1\}$-valued probabilistic poly-time function ensemble.

Definition 6.6 (pseudorandom number generator). Let $q(x)$ be a positive polynomial. A *pseudorandom number generator* (PRNG) is a uniform polynomial time function ensemble $f : \{0,1\}^{k(|n|)} \to \{0,1\}^{l(|n|)}$ such that for all poly-time statistical tests $\mathcal{A}$:

$$\forall q(x).\exists n_o.\forall n > n_o : \Big| \operatorname{Prob}_{s \in_R \{0,1\}^{k(|n|)}}\big[\mathcal{A}(f_n(s)) = \text{“1”}\big] - \operatorname{Prob}_{r \in_R \{0,1\}^{l(|n|)}}\big[\mathcal{A}(r) = \text{“1”}\big]\Big| \leq \frac{1}{q(n)}$$

In general, $f : \{0,1\}^{k(|n|)} \to \{0,1\}^{l(|n|)}$ is an interesting PRNG only when $\forall x \in \mathbb{N}.l(x) > k(x)$.

The reader might wish to review the definition of a probabilistic function (defn. 2.1) and composition of probabilistic functions (defn. 2.3) before proceeding.

Definition 6.7. Let $f : \{\epsilon\} \times \{0,1\}^{q(|n|)} \to [0,1]$ be a probabilistic poly-time function ensemble.[1] Let T_f be a term such that M_{T_f} computes f. Then, we say that $c_{q(|n|)}\langle T_f\rangle$ is the *characteristic process family for f with respect to channel $c_{q(|n|)}$.*

Let $f : \{\epsilon\} \times \{0,1\}^{q(|n|)} \to [0,1]$ be a probabilistic poly-time function ensemble and let $\mathcal{P}_f \equiv c_{q(|n|)}\langle T_f\rangle$ be its characteristic process family with respect to channel $c_{q(|n|)}$. Then it is easy to see that $\forall i \in \mathbb{N} : \operatorname{Prob}\big[f(\epsilon) = i\big] = \operatorname{Prob}\big[((c_{q(|n|)}(x).\mathbf{0}) \mid \mathcal{P}_f) \rightsquigarrow_S \langle i, c_{q(|n|)}\rangle\big]$. This is the probability that $\mathcal{P}_f$ is willing to communicate i on channel $c_{q(|n|)}$.

Lemma 6.8. *Let $\mathcal{A} : \{0,1\}^{m(|n|)} \times \{0,1\} \to [0,1]$ be a poly-time statistical test. Let $f : \{\epsilon\} \times \{0,1\}^{m(|n|)} \to [0,1]$ be a probabilistic poly-time function ensemble and let $\mathcal{P}_f$ be its characteristic process family with respect to channel $c_{m(|n|)}$. Then one can construct an adversarial context family $\mathcal{C}_{\mathcal{A}}[\ ]$ using a new channel d_1, such that for any fixed value for n, it is the case that $\forall i \in \mathbb{N}$: $Prob\big[(\mathcal{A} \circ f)(\epsilon) = i\big] = Prob\big[\mathcal{C}_{\mathcal{A}}[\mathcal{P}_f] \rightsquigarrow_S \langle i, d_1\rangle\big]$.*

Proof. If $\mathcal{A}$ is a poly-time statistical test then using theorem 2.11 we can construct the context family $\mathcal{C}_{\mathcal{A}}[\ ] \equiv c_{m(|n|)}(x).d_1\langle T_{\mathcal{A}}\rangle \mid [\ ]$. Note that $\mathcal{C}_{\mathcal{A}}[\ ]$ is an adversarial context family.

By assumption, $\mathcal{P}_f$ is the characteristic process family for f with respect to channel $c_{m(|n|)}$. Furthermore, $T_{\mathcal{A}}$ is produced from $\mathcal{A}$ using theorem 2.11. Hence, the observables $\langle 0, d_1\rangle$ or $\langle 1, d_1\rangle$ are observed during evaluation of $\mathcal{C}_{\mathcal{A}}[\mathcal{P}_f]$. The probability that $\langle 0, d_1\rangle$ is observed must be the same as the probability that $(\mathcal{A} \circ f)(\epsilon) = 0$. Similarly, the probability that $\langle 1, d_1\rangle$ is observed must be the same as the probability that $(\mathcal{A} \circ f)(\epsilon) = 1$. □

[1] ϵ is a dummy symbol. We wish to define a probabilistic function ensemble that takes no input and produces a probabilistic output. However $\emptyset \times A = \emptyset$ where A is any set. Hence, we have function take as input a single dummy value ϵ.

Any context family that implements a poly-time statistical test $\mathcal{A}$ by binding the equivalent term $T_{\mathcal{A}}$ to an input and obtaining values to which the test is applied from an expression plugged into the hole in the context family is, essentially, implementing a poly-time (distinguishing) attack on the expression plugged into the hole. So, we will say that an adversarial context family so constructed (*i.e.,* constructed in the manner of lemma 6.8) is a *poly-time attacker.*

Definition 6.9. Let $\mathcal{P}$ be a process family and o an observable. We will say that $f : \{\epsilon\} \times \{0,1\} \to [0,1]$ is an *indicator for* $\mathcal{P}$ *with respect to* o *and scheduler* S when $\mathrm{Prob}\big[\mathcal{P} \rightsquigarrow_S o\big] = \mathrm{Prob}\big[f(\epsilon) = 1\big]$ and $\mathrm{Prob}\big[\mathcal{P} \not\rightsquigarrow_S o\big] = \mathrm{Prob}\big[f(\epsilon) = 0\big]$.

Lemma 6.10. *Let* $\mathcal{C}[\ \ ]$ *be an adversarial context family and let* o *be an observable. Let* $f : \{\epsilon\} \times \{0,1\}^{m(|n|)} \to [0,1]$ *be any probabilistic poly-time function ensemble and let* $\mathcal{P}_f$ *be its characteristic process family. Then one can specify a poly-time statistical test* t *from the pair* $\langle \mathcal{C}[\ \ ], o\rangle$ *such that* $t \circ f$ *is an indicator for* $\mathcal{C}[\mathcal{P}_f]$ *with respect to the observable* o *and scheduler* S*.*

Proof. We compute $t(a)$ by evaluating $\mathcal{C}[d_{q(|n|)}\langle a\rangle]$ and returning 1 if the observable o was generated and 0 otherwise. To do this, we construct a probabilistic Turing machine $\widehat{M}_{C[d_{q(|n|)}\langle a\rangle],S}$ as follows. Let $M_{C[d_{q(|n|)}\langle a\rangle],S}$ be the oracle Turing machine constructed in the proof of theorem 4.7 and let $q(x)$ be a polynomial that bounds the run-time of $M_{C[d_{q(|n|)}\langle a\rangle],S}$. Then, $\widehat{M}_{C[d_{q(|n|)}\langle a\rangle],S}(n, o)$ first constructs a random bit sequence $\vec{b}$ of length $q(|n|)$ and computes $M_{C[d_{q(|n|)}\langle a\rangle],S}(n, \vec{b})$. Next, $\widehat{M}_{C[d_{q(|n|)}\langle a\rangle],S}$ examines the second output tape of $M_{C[d_{q(|n|)}\langle a\rangle],S}$ (the tape that contains the sequence of observables generated by $M_{C[d_{q(|n|)}\langle a\rangle],S}$) and if the tape contains the observable o given as the second input to $\widehat{M}_{C[d_{q(|n|)}\langle a\rangle],S}$, the machine returns a 1 and a 0 otherwise. (We note that $\widehat{M}_{C[d_{q(|n|)}\langle a\rangle],S}$ takes time polynomial in the size of n.) It is easy to see that $\mathrm{Prob}\big[(t(a))(\epsilon) = 1\big] = \mathrm{Prob}\big[\mathcal{C}[d_{q(|n|)}\langle a\rangle] \rightsquigarrow_S o\big]$ and $\mathrm{Prob}\big[(t(a))(\epsilon) = 0\big] = \mathrm{Prob}\big[\mathcal{C}[d_{q(|n|)}\langle a\rangle] \not\rightsquigarrow_S o\big]$.

Now what happens when we compute $t \circ f$? To evaluate $t \circ f$ we first evaluate $f(\epsilon)$, obtain a value (call it a) and then evaluate $f(a)$. Consider the process expression $\mathcal{C}[\mathcal{P}_f]$ Since $\mathcal{P}_f$ is the characteristic process family for f, $\mathcal{P}_f \equiv \langle T_f\rangle$ where M_{T_f} computes f. To evaluate $\mathcal{C}[d_{q(|n|)}\langle T_f\rangle]$ we first evaluate T_f (obtaining an atom a with the same probability that $f(\epsilon)$ returns a) and then evaluate $\mathcal{C}[d_{q(|n|)}\langle a\rangle]$ (obtaining 1 with the same probability that $\mathcal{C}[d_{q(|n|)}\langle a\rangle]$ generates the observable o and obtaining a 0 with the same probability that $\mathcal{C}[d_{q(|n|)}\langle a\rangle]$ didn't generate the observable o).

Consequently $\mathrm{Prob}\big[(t \circ f)(\epsilon) = 1\big] = \mathrm{Prob}\big[\mathcal{C}[d_{q(|n|)}\langle T_f\rangle] \rightsquigarrow_S o\big]$ and $\mathrm{Prob}\big[(t \circ f)(\epsilon) = 0\big] = \mathrm{Prob}\big[\mathcal{C}[d_{q(|n|)}\langle T_f\rangle] \not\rightsquigarrow_S o\big]$ whence $t \circ f$ is an indicator for $\mathcal{C}[\mathcal{P}_f]$ with respect to the observable o and scheduler S. ☐

Given an adversarial context family, it is easy to see that each potential observable defines a poly-time statistical test. It is important to note that the poly-time statistical test t constructed from the adversarial context family $\mathcal{C}[\ \ ]$ in the previous lemma (lemma 6.10) does not depend on the process family plugged into the hole in $\mathcal{C}[\ \ ]$ or on the function f to whose outputs the test t

is applied. Rather t depends only on $\mathcal{C}[\ ]$ and the choice of observable that we wish the test to detect.

We can now prove that an algorithm taking short strings to long strings is pseudorandom if and only if the process given by the algorithm, when evaluated on a short random input, is observationally equivalent to the process that returns a long random seed.

Definition 6.11. A function ensemble $f : \{\epsilon\} \times \{0,1\}^{l(|n|)} \to [0,1]$ is *random poly-time* if f with respect to n is a poly-time function such that $\forall x \in \{0,1\}^{l(|n|)} : f(\epsilon, x) = \frac{1}{2^{l(|n|)}}$.

Theorem 6.12. *Let $f' : \{0,1\}^{k(|n|)} \to \{0,1\}^{l(|n|)}$ ($\forall x \in \mathbb{N}.l(x) > k(x)$) be a uniform poly-time function ensemble. Let $r : \{\epsilon\} \times \{0,1\}^{l(|n|)} \to [0,1]$ and $s : \{\epsilon\} \times \{0,1\}^{k(|n|)} \to [0,1]$ be uniform poly-time random function ensembles. Define f as $f' \circ s$. Let $\mathcal{F}$ (resp. $\mathcal{R}$) be the characteristic process family for f (resp. r).*

Then, f' is a PRNG if and only if $\mathcal{F} \cong \mathcal{R}$.

Let M_{T_r} compute r and let M_{T_f} compute f. Then, $\mathcal{F}$ is a process family that, essentially, transmits a random seed generated by s to f' (a candidate PRNG—note that the type of $f' \circ s$ is $\{\epsilon\} \times \{0,1\}^{l(|n|)} \to [0,1]$) via function composition, and then transmits the value output by T_f on a public channel. In contrast, $\mathcal{R}$ is a process family that simply transmits the value output by T_r (a function that returns truly random values of the same length as those generated by T_f) on a public channel.

Proof. Assume that $\mathcal{F} \cong_S \mathcal{R}$ and that, by way of producing a contradiction, f' is not a PRNG. Then, we have that there exists a poly-time statistical test $\mathcal{A}$ that distinguishes between the output of f' given a random input value (a seed) and the output of a truly random source. That is to say:

$$\exists q(x).\forall n_o.\exists n > n_o : \Big| \operatorname{Prob}_{x \in_R \{0,1\}^{k(|n|)}}\big[\mathcal{A}(f'(x)) = \text{“1”}\big] \\ - \operatorname{Prob}_{y \in_R \{0,1\}^{l(|n|)}}\big[\mathcal{A}(y) = \text{“1”}\big]\Big| > \frac{1}{q(n)}$$

But we have that $\forall j \in \mathbb{N} : \operatorname{Prob}_{x \in_R \{0,1\}^{k(|n|)}}\big[f'(x) = j\big] = \operatorname{Prob}\big[(f' \circ s)(\epsilon) = j\big]$ and $\forall j \in \mathbb{N} : \operatorname{Prob}_{y \in_R \{0,1\}^{l(|n|)}}\big[y = j\big] = \operatorname{Prob}\big[r(\epsilon) = j\big]$. Hence, by lemma 6.8, we can construct a poly-time attacker $\mathcal{C}[\ ]$ that distinguishes between $\mathcal{F}$ and $\mathcal{R}$ using scheduler S with the same probability that $\mathcal{A}$ distinguishes between f and r. So $\mathcal{C}[\ ]$ will distinguish between $\mathcal{F}$ and $\mathcal{R}$ with probability greater than $\frac{1}{q(n)}$ for some polynomial q (as $\mathcal{A}$ distinguishes between f and r with probability greater than $\frac{1}{q(n)}$), thus producing a contradiction.

Now, assume that f' is a PRNG and that, by way of producing a contradiction, $\mathcal{F} \ncong \mathcal{R}$. Then we have that for some polynomial p there exists an adversarial context family $\mathcal{D}[\ ]$ that distinguishes between the two processes with probability greater than $\frac{1}{p(n)}$ for some scheduler S. Let the distinguishing observation be $\langle i, d_{q(|n|)}\rangle$. We can then use lemma 6.10 to construct a poly-time statistical test that distinguishes between the output of f and a truly random source with precisely the same probability that $\mathcal{D}[\ ]$ distinguishes between $\mathcal{F}$ and $\mathcal{R}$, thereby creating a contradiction.

Since we obtained the result for arbitrary scheduler S, we have that f' is a PRNG if and only if $\mathcal{F} \cong_s \mathcal{R}$ for arbitrary scheduler S whence we obtain that f' is a PRNG if and only if $\mathcal{F} \cong \mathcal{R}$. □

7 Conclusion and Future Directions

The language presented in this paper allows us to define probabilistic processes which communicate over a network that gives an adversary access to those communications.

Our process language uses a separate term language to actually perform computations (the expression $c_{p(|n|)}(x).d_{q(|n|)}\langle x + 2\rangle$, roughly speaking, computes the function $x+2$ but all the computation is done by the term $x+2$). One might argue that such a distinction is not particularly necessary since the presence of polynomially-bounded iteration in our process language should make it expressive enough for probabilistic poly-time functions even without a dedicated term language. This is certainly possible and we leave it to the interested reader to formalize the necessary constructions and establish the properties equivalent to the ones shown here. One major advantage of our approach is that we simplify the proof of the time bound on process evaluation since we do not have to establish that computation be bounded by poly-time; rather we can stipulate that the term language we use have the desired poly-time property. Additionally, we more accurately model the situation from the point of view of an adversary since the sources of computation (terms) are invisible to an adversary; only the information flows can be accessed by an attacker.

Our first significant result is to show that expressions written in our language are all bounded by polynomial time. We then proposed a definition of observational equivalence for probabilistic programs that is based on the view that large differences in probability are easier to observe than small differences. When we distinguish between "large" and "small" using asymptotic behavior, we arrive at a definition of observational equivalence that coincides with a standard concept from cryptography, namely, indistinguishability by polynomial-time statistical tests. While we have not fully explored the consequences of this definition, we believe it may shed new light on other basic concepts in cryptography, such as the distinction between semantically secure and non-malleable cryptosystems.

The steps taken in this paper form the beginning of a larger program that we hope to carry out over the next few years. In part following the program established in the study of spi-calculus [AG99, AG97], we hope to develop methods for reasoning about observational equivalence and use these methods to establish security properties of various protocols. In this paper we also presented a reasoning mechanism inspired by the standard notion of a bisimulation between processes [Mil89]. While this mechanism has proved sufficient to establish some preliminary equivalences, much work needs to be done in developing a range of techniques sufficiently expressive to deal with the non-trivial problems this approach was intended to handle.

Finally, we showed that a fundamental cryptographic notion, that of a pseudo-random number generator, is expressible in our language in a fairly natural way. In future, we hope to demonstrate that a range of other foundational cryptographic primitives, such as pseudo-random function families, are all similarly expressible in our language. By capturing a variety of essential cryptographic notions in our framework, we hope to lay the groundwork for a methodology by which we reason "backwards" from a desired property of a particular protocol to the properties that the various primitives in the protocol must possess in order to establish the desired protocol property.

We expect some interesting foundational questions to arise in the formulation of security properties such as authentication and secrecy. It may also be possible to develop model-checking procedures along the lines of these already explored for probabilistic temporal logics [HJ89, Han94, dA97, HK97]. In fact, we hope to be able to develop automated reasoning procedures for use in a network security setting using techniques developed in our study of the properties of our process calculus.

Acknowledgements: Thanks to M. Abadi, R. Amadio, D. Boneh, R. Canetti, C. Dwork, R. van Glabbeek, M. Hofmann, A. Jeffrey, S. Kannan, B. Kapron, P. Lincoln, A. Martin, R. Milner, I. Mironov, M. Mitchell, M. Naor, P. Panangaden and P. Selinger for helpful discussions and advice on relevant literature.

References

[AF01] M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *28th ACM Symposium on Principles of Programming Languages*, pages 104–115, 2001.

[AG97] M. Abadi and A. Gordon. A calculus for cryptographic protocols: the spi calculus. In *Proc. 4-th ACM Conference on Computer and Communications Security*, pages 36–47, 1997. Revised and expanded versions in *Information and Computation* 148(1999):1-70 and as SRC Research Report 149 (January 1998).

[AG98] M. Abadi and A. Gordon. A bisimulation method for cryptographic protocol. In *Proc. ESOP'98, Springer Lecture Notes in Computer Science*, 1998.

[AG99] M. Abadi and A. Gordon. A calculus for cryptographic protocols: the spi calculus. *Information and Computation*, 143:1–70, 1999. Expanded version available as SRC Research Report 149 (January 1998).

[AJ01] M. Abadi and J. Jürjens. Formal eavesdropping and its computational interpretation. In *Proc. 4-th International Symposium on Theoretical Aspects of Computer Software (TACS2001)*, Tohoku University, Sendai, Japan, 2001. Springer LNCS.

[AR00] M. Abadi and P. Rogaway. Reconciling two views of cryptography (The computational soundness of formal encryption). In *IFIP International Conference on Theoretical Computer Science*, Sendai, Japan, 2000. Full paper to appear in *J. of Cryptology.*

[Ata99] M.J. Atallah, editor. *Algorithms and Theory of Computation Handbook*, pages 19–28. CRC Press LLC, 1999.

[BAN89] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. *Proceedings of the Royal Society, Series A*, 426(1871):233–271, 1989. Also appeared as SRC Research Report 39 and, in a shortened form, in ACM Transactions on Computer Systems 8, 1 (February 1990), 18-36.

[Bel92] S. Bellantoni. *Predicative Recursion and Computational Complexity.* PhD thesis, University of Toronto, 1992.

[Can00] R. Canetti. A unified framework for analyzing security of protocols. Cryptology ePrint Archive: Report 2000/067; see http://eprint.iacr.org/2000/067/, 2000.

[CDL+99] I. Cervesato, N.A. Durgin, P.D. Lincoln, J.C. Mitchell, and A. Scedrov. A meta-notation for protocol analysis. In *12-th IEEE Computer Security Foundations Workshop.* IEEE Computer Society Press, 1999.

[dA97] L. de Alfaro. Temporal logics for the specification of performance and reliability. In *STACS'97*, volume 1200 of *LNCS*, pages 165–176. Springer-Verlag, February 1997.

[DDN91] D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography (extended abstract). In *Proc. 23rd Annual ACM Symposium on the Theory of Computing*, pages 542–552, 1991.

[DMP01] N.A. Durgin, J.C. Mitchell, and D. Pavlovic. A compositional logic for protocol correctness. In *IEEE Computer Security Foundations Workshop*, page (to appear), 2001.

[DY81] D. Dolev and A. Yao. On the security of public-key protocols. In *Proc. 22nd Annual IEEE Symp. Foundations of Computer Science*, pages 350–357, 1981.

[ElG85] T. ElGamal. A public-key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, IT-31:469–472, 1985.

[Gen00] R. Gennaro. An improved pseudo-random generator based on discrete log. In *Proc. CRYPTO 2000*, pages 469–481. Springer LNCS 1880, 2000. Revised version available on www.research.ibm.com/people/r/rosario/.

[GM84] S. Goldwasser and S. Micali. Probabilistic encryption. *J. Computer and System Sciences*, 28:281–308, 1984.

[Gol99] O. Goldreich. *Modern Cryptography, Probabilistic Proofs and Pseudorandomness*. Springer Verlag, 1999.

[Gol00] O. Goldreich. The Foundations of Cryptography — A Book in Preparation. Available on www.wisdom.weizmann.ac.il/˜oded/foc-book.html, 2000.

[Han94] H. Hansson. *Time and Probabilities in Formal Design of Distributed Systems*. Real-Time Safety Critical Systems. Elsevier, 1994.

[HJ89] H. Hansson and B. Jonsson. A framework for reasoning about time and reliability. In *Proc. of Real Time Systems Symposium*, pages 102–111. IEEE, 1989.

[HK97] M. Huth and M. Kwiatkowska. Quantitative analysis and model checking. In *LICS'97*, pages 111–122, 1997.

[Hof99] M. Hofmann. Type systems for polynomial-time computation. Habilitation thesis, Darmstadt, 1999; see www.dcs.ed.ac.uk/home/mxh/papers.html, 1999.

[KMM94] R. Kemmerer, C. Meadows, and J. Millen. Three systems for cryptographic protocol analysis. *J. Cryptology*, 7(2):79–130, 1994.

[LMMS98] P.D. Lincoln, M. Mitchell, J.C. Mitchell, and A. Scedrov. A probabilistic poly-time framework for protocol analysis. In M.K. Reiter, editor, *Proc. 5-th ACM Conference on Computer and Communications Security*, pages 112–121, San Francisco, California, 1998. ACM Press.

[LMMS99] P.D. Lincoln, J.C. Mitchell, M. Mitchell, and A. Scedrov. Probabilistic polynomial-time equivalence and security protocols. In J.M. Wing and J. Woodcock and J. Davies, editor, *Formal Methods World Congress, Vol. I*, pages 776–793, Toulouse, France, 1999. Springer LNCS 1708.

[Low96] G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using CSP and FDR. In *2nd International Workshop on Tools and Algorithms for the Construction and Analysis of Systems*. Springer-Verlag, 1996.

[LS91] K.G. Larsen and A. Skou. Bisimulation through probabilistic testing. *Information and Computation*, 94(1):1–28, 1991.

[Lub96] M. Luby. *Pseudorandomness and Cryptographic Applications*. Princeton Computer Science Notes, Princeton University Press, 1996.

[Mea96] C. Meadows. Analyzing the Needham-Schroeder public-key protocol: a comparison of two approaches. In *Proc. European Symposium On Research In Computer Security*. Springer Verlag, 1996.

[Mil89] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[MMS97] J.C. Mitchell, M. Mitchell, and U. Stern. Automated analysis of cryptographic protocols using Murφ. In *Proc. IEEE Symp. Security and Privacy*, pages 141–151, 1997.

[MMS98] J.C. Mitchell, M. Mitchell, and A. Scedrov. A linguistic characterization of bounded oracle computation and probabilistic polynomial time. In *Proc. 39-th Annual IEEE Symposium on Foundations of Computer Science*, pages 725–733, Palo Alto, California, 1998. IEEE Computer Society Press.

[MRST01] J. Mitchell, A. Ramanathan, A. Scedrov, and V. Teague. A probabilistic polynomial-time calculus for analysis of cryptographic protocols (preliminary report). In *17-th Annual Conference on the Mathematical Foundations of Programming Semantics, Arhus, Denmark, May, 2001*. Electronic Notes in Theoretical Computer Science, Vol. 45, 2001.

[NS78] R. Needham and M. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–9, 1978.

[Pau97a] L.C. Paulson. Mechanized proofs for a recursive authentication protocol. In *10th IEEE Computer Security Foundations Workshop*, pages 84–95, 1997.

[Pau97b] L.C. Paulson. Proving properties of security protocols by induction. In *10th IEEE Computer Security Foundations Workshop*, pages 70–83, 1997.

[PW00] B. Pfitzmann and M. Waidner. Composition and integrity preservation of secure reactive systems. In *7-th ACM Conference on Computer and Communications Security, Athens, November 2000*, pages 245–254. ACM Press, 2000. Preliminary version: IBM Research Report RZ 3234 (# 93280) 06/12/00, IBM Research Division, Zürich, June 2000.

[Ros95] A. W. Roscoe. Modelling and verifying key-exchange protocols using CSP and FDR. In *CSFW VIII*, page 98. IEEE Computer Soc Press, 1995.

[Sch96] S. Schneider. Security properties and CSP. In *IEEE Symp. Security and Privacy*, 1996.

[vGSS95] R.J. van Glabbeek, S.A. Smolka, and B. Steffen. Reactive, generative and stratified models of probabilistic processes. *Information and Computation*, 121(1):59–80, 1995.

[Yao82] A. Yao. Theory and applications of trapdoor functions. In *IEEE Foundations of Computer Science*, pages 80–91, 1982.