Symbolic Probabilistic Analysis of Off-line Guessing

Bruno Conchinha[1], David Basin[2], and Carlos Caleiro[3]

[1] Information Security Group, ETH Zürich, Zürich, Switzerland[1,2]
brunoco@inf.ethz.ch[1], basin@inf.ethz.ch[2]
[2] SQIG - Instituto de Telecomunicações, Department of Mathematics,
IST, TU Lisbon, Portugal
ccal@math.ist.utl.pt[3]

Abstract. We introduce a probabilistic framework for the automated analysis of security protocols. Our framework provides a general method for expressing properties of cryptographic primitives, modeling an attacker who is more powerful than conventional Dolev-Yao attackers. Within our framework, we can model equational properties of cryptographic primitives as well as property statements about their weaknesses, e.g. primitives leaking partial information about messages or the use of weak algorithms for random number generation. Moreover, we can use these properties to find attacks and estimate their success probability. Existing symbolic methods can neither model such properties nor find such attacks.
We show that the probability estimates we obtain are negligibly different from those yielded by a generalized random oracle model based on sampling (the random variables associated to symbolic) terms into bitstrings, while respecting the stipulated properties of cryptographic primitives.
As case studies, we use a prototype implementation of our framework to model non-trivial properties of RSA encryption and automatically estimate the probability of off-line guessing attacks on the EKE protocol.

Keywords: Probability, Off-line Guessing, Equational Theories, Random Oracle Model

1 Introduction

Cryptographic protocols play an important role in securing distributed computation and it is crucial that they work correctly. Symbolic verification approaches are usually based on the *Dolev-Yao model*: messages are represented by terms in a term algebra, cryptography is assumed to be perfect, and properties of cryptographic operators are formalized equationally [1]. This strong abstraction eases analysis and numerous successful verification tools rely on it [2, 3]. However, it may not accurately represent an attacker's capabilities. As a consequence, broad classes of attacks that rely on cryptanalysis or weaknesses of cryptographic primitives fall outside the scope of such methods.

Proving security by reasoning directly about bitstrings, as in *computational approaches* [4–6], yields stronger security guarantees. However, it requires long, error-prone, hand-written proofs to establish the security of given protocols using specific cryptographic primitives.

Much research has been devoted to bridging the gap between these two approaches. Broadly speaking, this line of work aims to automatically obtain strong protocol security guarantees under sufficiently strong assumptions on the security of cryptographic primitives. Good surveys of this work include [7, 8]. We discuss existing approaches and their limitations in greater detail later in this section.

Related work. Much work has been devoted to bridging the gap between symbolic and computational models. There are two main lines of research in this direction: (1) obtaining computational soundness results for symbolic methods, and (2) developing techniques that reason directly with computational models.

The first line of research, developing computational soundness results, was initiated with Abadi and Rogaway's seminal paper [9]. They investigated assumptions under which symbolic security implies computational security, whereby protocols that are secure against a Dolev-Yao attacker are also secure against the much more powerful adversary of the computational world. Such results now exist for more general protocols and stronger attackers [10, 11], observational equivalence [12], length revealing and same-key revealing encryption systems [13], encryption with composed keys [14] and hash functions [15], to name but a few. The limitations of these results include the very strong assumptions on the security of cryptographic primitives, the requirement that messages are tagged so that their structure is known to any observer, and the difficulty of extending the results to new cryptographic primitives, which usually involves re-doing most of the work.

The second line of research aims to obtain, automatically where possible, a sequence of property-preserving transformations between game-theoretic problem formulations, as is done in computational security proofs. The original ideas in this direction of [16–18] have been developed into tools like CryptoVerif [19], CertiCrypt [20] and EasyCrypt [21]. Moreover, [22] introduces a logic for reasoning about cryptographic primitives in such models. These tools can prove protocols correct in the computational world and, when successful, provide upper bounds on the probability of an attack.

More recently, [23] proposes yet another approach: a symbolic framework in which it is possible to express security properties of cryptographic primitives and use them to prove computational protocol security. However, failure to obtain a security proof does not necessarily yield a feasible attack, and so far no automated method exists to prove protocol security in this setting.

Our applications in this paper focus on off-line guessing attacks. Off-line guessing attacks have been widely studied in symbolic settings by using static equivalence [24, 25] and closely related notions [26]. In [27] a computational soundness result is given, albeit in a rather restricted scenario. However, off-line guessing attacks remain a real threat to protocol security. Password-cracking

software is freely available on the internet, and is remarkably successful [28]. Furthermore, such attacks often rely on weaknesses of cryptographic primitives that cannot be modeled by existing automated methods [29, 30].

Contributions. We present a fundamentally new approach to strengthening the security guarantees provided by automated methods. Our approach is in a sense dual to the current main lines of research that aims to bridge the gap between symbolic and the computational models: Rather than assuming strong security properties of cryptographic primitives and using them to prove security, we explicitly describe weaknesses of cryptographic primitives and random number generation algorithms and use them to find attacks.

We propose a probabilistic framework for security protocol analysis in which properties of cryptographic primitives can be specified. Besides equational properties, our framework allows us to express security relevant properties of random number generation algorithms and relations between the input and the output of cryptographic primitives. For instance, it can model a random number generation algorithm that generates bitstrings representing primes of a certain length, a hash function that leaks partial information about the original message, or a cryptosystem whose valid public keys have some redundancy. The specified properties can then be used to find attacks and to estimate their success probability. Such properties cannot be modeled by existing symbolic methods and yet often lead to attacks on real-world implementations.

We model cryptographic functions using a generalized random oracle model. Given a concrete specification of the cryptographic primitives used and their properties, symbolic terms are sampled to bitstrings in a way that ensures that the properties of the specification are always satisfied, but otherwise functions behave as random oracles. Under reasonable assumptions on the specification, we can define such generalized random oracles and prove that they yield a valid probabilistic model. Moreover, we show that probabilities in this model can be effectively computed, and we provide a prototype implementation that calculates these probabilities.

We believe that this model is interesting in its own right. It is a non-trivial generalization of the standard model of random oracle for hash functions, and it captures the intuitive idea that cryptographic primitives satisfy stated properties, which can be explored by an attacker, but otherwise behave ideally.

We illustrate the usefulness of our framework by using it to analyze the security of protocols against off-line guessing attacks. Given the pervasive use of weak human-picked passwords, off-line guessing attacks are a major concern in security protocol analysis and have been the subject of much research, using symbolic [24–26] and computational approaches [31], and relating the two via computational soundness results [27, 32]. We show that our framework can be used to straightforwardly represent non-trivial properties of cryptographic primitives like the redundancy of RSA keys. We will use these properties to model off-line guessing attacks on the EKE protocol [29] and estimate their success probabilities using our implementation. Although these attacks are well-known, their analysis was previously outside the scope of symbolic methods. Further applica-

tions of our approach (not described here) include reasoning about differential cryptanalysis or side-channel attacks [33] as well as short-string authentication and distance-bounding protocols.

Outline. In Section 2 we describe our framework's syntax and semantics. In Section 3 we introduce our generalized random oracle model and show that it yields a computable probability measure. In Section 4 we show how our framework can be used to find realistic off-line guessing attacks that rely on non-trivial probabilistic properties of the cryptographic primitives. In Section 5 we draw conclusions and discuss future work. Full proofs of all results are given in Appendix A. Appendix B explains how to compute probabilities in our probabilistic model.

2 Definitions

Symbolic terms. A *signature* $\Sigma = \biguplus_{n\in\mathbb{N}} \Sigma_n$ is a set of function symbols, where Σ_i contains the symbols of arity i. Given a set G of generators, we define $T_\Sigma(G)$ as the smallest set such that $G \subseteq T_\Sigma(G)$, and if $f \in \Sigma_n$ and $t_1, \ldots, t_n \in T_\Sigma(G)$, then $f(t_1, \ldots, t_n) \in T_\Sigma(G)$. For simplicity, we write c instead of $c()$ if $c \in \Sigma_0$. Below, unless otherwise stated, we will consider $G = \emptyset$ and write T_Σ instead of $T_\Sigma(\emptyset)$.

We define the *head* of a term $t = f(t_1, \ldots, t_n)$ by $\mathit{head}(t) = f$. We define the set $\mathit{sub}(t)$ of *subterms* of a term $t = f(t_1, \ldots, t_n)$ inductively by $\mathit{sub}(t) = \{t\} \cup (\bigcup_{i=1}^{n} \mathit{sub}(t_i))$, as usual, and the set $\mathit{psub}(t)$ of *proper subterms* of t by $\mathit{psub}(t) = \mathit{sub}(t) \setminus \{t\}$. If $f\colon A \to B$ and $A' \subseteq A$, we write $f[A']$ for the image $\{f(a) \mid a \in A'\}$ of A' under f.

Equational theories. Algebraic properties of symbolic terms are captured by equational theories. Given a signature Σ, an *equational theory* $\approx$ is a congruence relation on T_Σ. As usual, we write $t \approx t'$ instead of $(t, t') \in \approx$. We will consider an equational theory $\approx_R$ obtained from a subterm convergent rewriting system R, as in [34].

Example 1. The signature $\Sigma^{\mathcal{DY}}$ is used to represent the cryptographic primitives present in simple Dolev-Yao models containing a hash function, symmetric encryption and decryption, pairing and projections. It is given by $\Sigma^{\mathcal{DY}} = \Sigma_1^{\mathcal{DY}} \cup \Sigma_2^{\mathcal{DY}}$, where $\Sigma_1^{\mathcal{DY}} = \{\mathsf{h}, \pi_1, \pi_2\}$ and $\Sigma_2^{\mathcal{DY}} = \left\{ \{\!|\cdot|\!\}_\cdot, \{\!|\cdot|\!\}_\cdot^{-1}, \langle\cdot,\cdot\rangle \right\}$.

Standard equational properties of these primitives are represented by the rewriting system $R_{\mathcal{DY}}$ containing the rules $\pi_1(\langle x, y\rangle) \to x$, $\pi_2(\langle x, y\rangle) \to y$, and $\left\{ \left| \{\!|x|\!\}_y \right| \right\}_y^{-1} \to x$. It is simple to check that this rewriting system is convergent.

Property statements. Property statements represent properties of function symbols by expressing relations between their inputs and outputs.

Let $\mathcal{T}$ be a set of *types*. Given a signature Σ, a *property statement* is a tuple $(f, T_1, \ldots, T_n, T)$, written $f[T_1, \ldots, T_n] \subseteq T$, where $f \in \Sigma_n$ and $T_1, \ldots, T_n, T \in$

$\mathcal{T}$. If $ps = (f[T_1, \ldots, T_n] \subseteq T)$, we define the *head symbol* of ps by $head(ps) = f$, $dom(ps) = T_1 \times \ldots \times T_n$ and $ran(ps) = T$.

Given a set PS of property statements and $f \in \Sigma$, we denote by PS_f the set of property statements in PS whose head symbol is f. Note that, in general, we may have more than one property statement associated to each function symbol. We write $f[T_1, \ldots, T_n] \subseteq_{PS} T$ instead of $(f[T_1, \ldots, T_n] \subseteq T) \in PS$.

Syntax. The syntax of our setup is defined by a four-tuple $\langle \Sigma, \approx_R, \mathcal{T}, PS, [\![\cdot]\!] \rangle$,

where Σ is a signature, $\approx_R$ is an equational theory on T_Σ defined by a convergent rewriting system R, $\mathcal{T}$ is a set of types, and PS is a set of property statements.

We require that the signature contains an infinite number of constants, that is, Σ_0 is infinite, and that $\Sigma \setminus \Sigma_0$ is finite. Constant symbols represent either cryptographically relevant constants (such as the constant bitstring 0 for XOR) or random data generated by agents or the attacker.

Interpretation functions. Type interpretation functions associate to each type a set of bitstrings, thereby providing a meaning to types. We write $\mathcal{B}$ for $\{0, 1\}$. Given a set $\mathcal{T}$ of types, a *type interpretation function* is a function $[\![\cdot]\!] : \mathcal{T} \to \mathcal{P}(\mathcal{B}^*)$ that associates types to sets of bitstrings such that $[\![T]\!]$ is finite and non-empty for every $T \in \mathcal{T}$. We extend this interpretation function to products, and define $[\![T_1 \times \ldots \times T_n]\!] = [\![T_1]\!] \times \ldots \times [\![T_n]\!]$.

A *setup specification* is a pair $\mathcal{S} = \langle S, [\![\cdot]\!] \rangle$, where $S = \langle \Sigma, \approx_R, \mathcal{T}, PS, [\![\cdot]\!] \rangle$ is a four-tuple defining the syntax of the setup as in the above paragraph and $[\![\cdot]\!]$ is an interpretation function, which consistently defines the behavior of all function symbols: Concretely, we require that $PS_f \neq \emptyset$ for all $f \in \Sigma$, and if $ps_1, ps_2 \in PS_f$ then $[\![dom(ps_1)]\!] \cap [\![dom(ps_2)]\!] = \emptyset$. For $c \in \Sigma_0$, these conditions imply that there is a single $T \in \mathcal{T}$ such that $c \subseteq_{PS} T$. We denote this unique T by $type(c)$.

We assume that the function represented by a function symbol is undefined unless otherwise stated by a property statement concerning that function symbol: that is, we assume that, if $f \in \Sigma_n$ and there is no $ps \in PS_f$ such that $(b_1, \ldots, b_n) \in [\![dom(ps)]\!]$, then the function represented by the symbol f is undefined on the input $(b_1, \ldots, b_n)$. Under these conditions, we set the *domain of definability* of f to be $dom_{\mathcal{S}}(f) = \biguplus_{ps \in PS_f} [\![dom(ps)]\!]$. Note that $\emptyset \subsetneq dom_{\mathcal{S}}(f) \subseteq (\mathcal{B}^*)^n$ for all $f \in \Sigma$.

Example 2. We specify a simple yet realistic setup extending the typical Dolev-Yao model, by modeling a hash function h that maps any bitstring to a bitstring of length 256, a pairing function that given any pair of bitstrings returns their labeled concatenation, and a symmetric encryption scheme that uses a block cipher together with some reversible padding technique.

We first enrich the signature $\Sigma^{\mathcal{DY}}$ with constants c_i for $i \in \mathbb{N}$. The rewriting system $R_{\mathcal{DY}}$ remains unchanged. The types we will consider and their interpretations under $[\![\cdot]\!]$ are as follows:

– pw represents weak (e.g., human-chosen) passwords. We model these passwords as being encoded by 256-bit bitstrings, but sampled from a relatively small set: thus, $[\![\mathsf{pw}]\!] \subset \mathcal{B}^{256}$ and $|[\![\mathsf{pw}]\!]| = 2^{24}$;
– $\mathsf{sym\_key}$ represents symmetric keys, with $[\![\mathsf{sym\_key}]\!] = \mathcal{B}^{256}$;
– text represents one block of plaintext, with $[\![\mathsf{sym\_key}]\!] = \mathcal{B}^{256}$;
– $T_{\mathcal{B}^n}$ with $[\![T_{\mathcal{B}^n}]\!] = \mathcal{B}^n$ for each $n \in \mathbb{N}$;
– $T_{\mathcal{B}^{(n,m)}}$ with $[\![T_{\mathcal{B}^{(n,m)}}]\!] = \mathcal{B}^{(n,m)} = \bigcup_{i=n}^{m} \mathcal{B}^i$ for each $n, m \in \mathbb{N}$; and
– $T_{\mathcal{B}^{n\#m}}$ represents the set of bitstrings that are labeled concatenations of two bitstrings of size n and m, with $[\![T_{\mathcal{B}^{n\#m}}]\!] = \mathcal{B}^{n\#m} \subseteq \mathcal{B}^{n+m+\lceil \log(n+m) \rceil}$, for each $n, m \in \mathbb{N}$.

We define the set PS by

$$\begin{aligned} PS = \{\ & h[T_{\mathcal{B}^n}] \subseteq T_{\mathcal{B}^{256}}, \pi_1[T_{\mathcal{B}^{n\#m}}] \subseteq T_{\mathcal{B}^n}, \pi_2[T_{\mathcal{B}^{n\#m}}] \subseteq T_{\mathcal{B}^m}, \\ & \langle T_{\mathcal{B}^n}, T_{\mathcal{B}^m} \rangle \subseteq T_{\mathcal{B}^{n\#m}}, \\ & \{\!|T_{\mathcal{B}^{(256n+1,256(n+1)}}|\!\}_{T_{\mathcal{B}^{256}}} \subseteq T_{\mathcal{B}^{256(n+1)}}, \\ & \{\!|T_{\mathcal{B}^{256(n+1)}}|\!\}^{-1}_{T_{\mathcal{B}^{256}}} \subseteq T_{\mathcal{B}^{(256n+1,256(n+1))}} \mid n, m \in \mathbb{N}.\}. \end{aligned}$$

Note that all functions are modeled as undefined on all arguments that fall outside the domains of these property statements. For example, symmetric encryption of any term is undefined unless the key is a 256-bit bitstring.

Example 3. We use our framework to formalize RSA encryption, taking into account properties of the key generation algorithm. An RSA public key is a pair (n, e), where $n = p \cdot q$ and p, q are large primes (typically of around 512 bits), and the exponent e is relatively coprime to $\varphi(n) = (p-1)(q-1)$. The private key d is the multiplicative inverse of e modulo $\varphi(n)$.

We extend the setup specification of Example 2. We add to the signature the following five primitives: the unary functions mod, expn, and inv, representing the extraction of the modulus, the exponent, and the exponent's multiplicative inverse, respectively, from a randomly generated RSA public-private key pair; a binary function $\{\cdot\}^{-1}_{\cdot} \in \Sigma_2^{\mathcal{DY}}$, representing the RSA decryption function; and a ternary function $\{\cdot\}_{\cdot,\cdot}$, representing RSA encryption.

The only rewriting rule that we must add to model RSA encryption is $\left\{\{m\}_{\mathsf{mod}(k),\mathsf{expn}(k)}\right\}^{-1}_{\mathsf{inv}(k)} \to m$, where m, k are variables.

The additional types that we will use to formalize relevant properties of these functions and their interpretations are as follows: random represents the random values used to generate an RSA public-private key pair, including two 512-bit prime numbers and the 1024-bit exponent, with $[\![\mathsf{random}]\!] \subseteq \mathcal{B}^{2048}$; $\mathsf{prodprime}$ represents the product of two 512-bit prime numbers, so that $[\![\mathsf{prodprime}]\!] \subseteq \mathcal{B}^{1024}$ and $|[\![\mathsf{prodprime}]\!]| \approx 2^{1008}$ (by the prime number theorem); odd represents 1024-bit odd numbers, with $[\![\mathsf{odd}]\!] \subseteq \mathcal{B}^{1024}$ and $|[\![\mathsf{odd}]\!]| = 2^{1023}$.

We add the following property statements:

– $\mathsf{mod}[\mathsf{random}] \subseteq \mathsf{prodprime}$, because the modulo of an RSA public key is the product of two primes.

– $\mathsf{expn}[\mathsf{random}] \subseteq \mathsf{odd}$, because the exponent of an RSA public key is always odd.
– $\mathsf{inv}[\mathsf{random}] \subseteq T_{\mathcal{B}^{1024}}$, because an RSA private key is a 1024-bit bitstring. Note that we do not allow extracting modulos, exponents, or inverses from anything other than a valid value for generating an RSA key pair.
– $\{T_{\mathcal{B}^{1024}}\}_{\mathsf{prodprime,odd}} \subseteq T_{\mathcal{B}^{1024}}$. This property states that encrypting any 1024-bit plaintext with a valid RSA public key yields a 1024-bit bitstring. Note that the encryption is undefined if the plaintext is not a 1024-bit bitstring, the modulus is not the product of two primes, or the exponent is even.
– $\{T_{\mathcal{B}^{1024}}\}^{-1}_{T_{\mathcal{B}^{1024}}} \subseteq T_{\mathcal{B}^{1024}}$. RSA decryption takes a ciphertext and a private key which are both 1024-bit bitstrings, and outputs a 1024-bit plaintext.

For simplicity, we merely require here that the public-key exponent is odd, rather than requiring it to be coprime with the modulus.

2.1 Semantics

Let us fix a setup specification $\mathcal{S} = \langle\langle \Sigma, \approx_R, \mathcal{T}, PS\rangle, [\![\cdot]\!]\rangle$.

Term assignments. Term assignments associate a value to each symbolic term. Given $\mathcal{S}$, recall that the domain of definability of a function symbol $f \in \Sigma_n$, $dom_{\mathcal{S}}(f)$, may be a proper subset of $(\mathcal{B}^*)^n$. Therefore, an error/undefined value, represented by $\bot$, is also considered. We will write $\mathcal{B}^*_\bot$ for the set $\mathcal{B}^* \cup \{\bot\}$. Note that $\bot$ never occurs in the domain of definability of any function; thus, applying a function to an undefined argument always yields an undefined result, as expected.

Let $\mathbf{\Omega}$ be the set of all functions $\omega\colon T_\Sigma \to \mathcal{B}^*_\bot$. We say that $\omega \in \mathbf{\Omega}$ *satisfies* the equational theory $\approx_R$, and we write $\omega \models \approx_R$, if, whenever $t \approx_R t'$, either $\omega(t) = \omega(t')$, or $\omega(t) = \bot$, or $\omega(t') = \bot$. We say that ω satisfies a property statement ps (under $[\![\cdot]\!]$), and write $\omega \models_{[\![\cdot]\!]} ps$ if, whenever $(\omega(t_1), \ldots, \omega(t_n)) \in [\![dom(ps)]\!]$, then $\omega(f(t_1, \ldots, t_n)) \in [\![ran(ps)]\!]$, and whenever $(\omega(t_1), \ldots, \omega(t_n)) \notin [\![dom(ps)]\!]$ for all $ps \in PS_f$, then $\omega(f(t_1, \ldots, t_n)) = \bot$. We say that ω satisfies PS (under $[\![\cdot]\!]$), and write $\omega \models_{[\![\cdot]\!]} PS$, if $\omega \models_{[\![\cdot]\!]} ps$ for all $ps \in PS$.

We say that ω satisfies the setup presentation $\mathcal{S}$, and write $\omega \models \mathcal{S}$, when $\omega \models \approx_R$ and $\omega \models_{[\![\cdot]\!]} PS$. We denote by $\Omega_{\mathcal{S}}$ the set of all $\omega \in \mathbf{\Omega}$ that satisfy $\mathcal{S}$.

Example 4. Functions ω that satisfy our equational theory may be such that $\omega(t) = \bot$ and $\omega(t') \neq \bot$ for terms t and t' such that $t \approx_R t'$. To see why this is allowed, recall from Example 2 that $\{\!|\cdot|\!\}^{-1}_{\cdot}$ represents a symmetric encryption algorithm in which valid keys always have 256 bits. Let $t, k \in \Sigma_0$, with $type(t) = \mathsf{text}$, and $t' = \{\!|\{\!|t|\!\}_k|\!\}^{-1}_k$. We have $t \approx_R t'$. Now, if ω represents a possible real-world assignment (of terms to bitstrings), we have $\omega(t) \neq \bot$ (since t represents a bitstring freshly sampled from $\mathcal{B}^{256}$). Moreover, if $\omega(k)$ is not a 256-bit bitstring, then $\omega(t') = \bot$ since our encryption and decryption functions are only defined for 256-bit keys. Therefore, $\omega(\{\!|\{\!|t|\!\}_k|\!\}^{-1}_k) = \bot$.

Finitely-generated events. Valid protocol execution traces are finite and, therefore, contain only finitely many terms. We are therefore interested in events that depend on only finitely many terms. For each finite set of terms $K \subseteq T_\Sigma$, let Λ_K be the set of functions $\lambda : K \to \mathcal{P}(\mathcal{B}^*_\perp)$ and, for each $\lambda \in \Lambda_K$, let

$$\Omega_\lambda = \{\omega \in \mathbf{\Omega} \mid \omega(t) \in \lambda(t) \text{ for each } t \in K\}.$$

Let $\Lambda = \bigcup_{K \in \mathcal{P}_{fin}(T_\Sigma)} \Lambda_K$ and $\Omega_\Lambda = \{\Omega_\lambda \mid \lambda \in \Lambda\}$, where $\mathcal{P}_{fin}(X)$ is the set of *finite* subsets of X. Note that Ω_Λ is the set of subsets of $\mathbf{\Omega}$ whose specification depends on only the instantiation of finitely many terms. Thus, we want our probability measure to be defined in the σ-algebra generated by Ω_Λ. Let $\mathcal{F}$ be this σ-algebra; we say that $\mathcal{F}$ is the σ-algebra of *finitely generated events*.

Probabilistic models. Given a setup specification $\mathcal{S}$, we consider probability spaces $(\mathbf{\Omega}, \mathcal{F}, \mu)$, where $\mathbf{\Omega}$ and $\mathcal{F}$ are as defined above and $\mu : \mathcal{F} \to [0, 1]$ is a probability measure. Note that $\mathbf{\Omega}$ and $\mathcal{F}$ are fixed for a given $\mathcal{S}$; it is the probability measure μ that we are interested in studying.

If $t \in T_\Sigma$, we write $\widehat{t} : \mathbf{\Omega} \to \mathcal{B}^*_\perp$ to denote the random variable on $\mathbf{\Omega}$ defined by $\widehat{t}(\omega) = \omega(t)$. We adopt standard (abuses of) notation from probability theory. If $C(b_1, \ldots, b_n)$ is a condition whose satisfaction depends on the bitstring values $b_1, \ldots, b_n$, we use the notational convention that

$$P_\mu[C(\widehat{t_1}, \ldots, \widehat{t_n})] = \mu(\{\omega \in \mathbf{\Omega} \mid C(\widehat{t_1}(\omega), \ldots, \widehat{t_n}(\omega))\}),$$

provided that $\{\omega \in \mathbf{\Omega} \mid C(\widehat{t_1}(\omega), \ldots, \widehat{t_n}(\omega))\} \in \mathcal{F}$. If $\Omega \in \mathcal{F}$, we will also write $P_\mu[\Omega]$ instead of $\mu(\Omega)$. We use standard notation for conditional probability. Namely, if $P_\mu[B] > 0$, then $P_\mu[A \mid B] = P_\mu[A, B]/P_\mu[B]$ is the *conditional probability of A given B*.

We say μ *satisfies* the equational theory $\approx_R$ if $\mu(\{\omega \mid \omega \models \approx_R\}) = 1$, and we write $\mu \models \approx_R$ to denote this fact. Analogously, we define the satisfaction of the set of property statements PS (under $[\![\cdot]\!]$) by μ, $\mu \models_{[\![\cdot]\!]} PS$, by $\mu(\{\omega \mid \omega \models_{[\![\cdot]\!]} PS\}) = 1$. We say that μ *satisfies*, or is a *model* of, the setup specification $\mathcal{S}$, written $\mu \models \mathcal{S}$, if $\mu \models \approx_R$ and $\mu \models_{[\![\cdot]\!]} PS$. Note that μ is a model of $\mathcal{S}$ if and only if $\mu(\Omega_\mathcal{S}) = 1$.

3 A Generalized Random Oracle Model

In this section we propose an algorithm for sampling the random variables associated with symbolic terms. Our sampling algorithm interprets functions as random oracles subject to satisfying our setup specification $\mathcal{S} = \langle \Sigma, \approx_R, \mathcal{T}, PS, [\![\cdot]\!] \rangle$.

3.1 Tentative term sampling in the ROM

Term sampling. Suppose that $K \subset T_\Sigma$ is a finite set of terms and P is a partition of K. We define $\approx_P$ to be the smallest congruence relation on T_Σ such that $\approx_R \subseteq \approx_P$ and $t \approx_P t'$ whenever there is $p \in P$ such that $t, t' \in p$.

The sampling algorithm below builds a function ψ_{ROM} mapping a finite set of terms to $\mathcal{B}_\perp^*$. We denote by $P(\psi_{ROM})$ the partition of $dom(\psi_{ROM})$ given by $P(\psi_{ROM}) = \{\psi_{ROM}^{-1}(b) \mid b \in ran(\psi_{ROM})\}$. The algorithm is probabilistic: at various steps, it samples a random bitstring from a finite subset of $\mathcal{B}_\perp^*$. We assume that this sampling is always done with uniform probability distribution. We also assume fixed some total order $\prec$ on the set of terms such that, if $t \in psub(t')$, then $t \prec t'$. We say that such an order is *subterm-compatible*.

Algorithm 1 (Tentative Term Sampling Algorithm)
Input: *a finite set of terms* $K \subseteq T_\Sigma$.
Output: *a function* $\psi_{ROM}\colon sub[K] \to \mathcal{B}_\perp^*$.

1: $\psi_{ROM} \leftarrow \emptyset$
2: *let* $t_1, \ldots, t_k$ *be such that* $t_1 \prec \ldots \prec t_k$ *and* $sub[K] = \{t_1, \ldots, t_k\}$
3: **for** i **from** 1 **to** k
4: *let* $t_i = f(t'_1, \ldots, t'_n)$
5: **if** $(\psi_{ROM}(t'_1), \ldots \psi_{ROM}(t'_n)) \notin dom_{\mathcal{S}}(f)$
6: $\psi_{ROM}(t_i) \leftarrow \perp$
7: **continue**
8: *let* ps *be the unique* $ps \in PS_f$ *s.t.* $(\psi_{ROM}(t'_1), \ldots \psi_{ROM}(t'_n)) \in [\![dom(ps)]\!]$
9: **if** $\exists t' \in dom(\psi_{ROM})$. $t \approx_{P(\psi_{ROM})} t'$ ***and*** $\psi_{ROM}(t') \neq \perp$
10: $\psi_{ROM}(t_i) \leftarrow \psi_{ROM}(t')$
11: **continue**
12: *randomly sample* b *from* $[\![ran(ps)]\!]$
13: $\psi_{ROM}(t_i) \leftarrow b$
14: **return** ψ_{ROM}

Algorithm 1 samples terms in order (lines 2–3), by interpreting each function symbol as a random oracle with uniform probability distribution (lines 12–13), and respecting the equational theory in case an equal term has already been sampled (lines 9–10), as long as all its argument values (previously sampled) are defined and form a tuple in its domain of definability (lines 5–6).

Problems with the tentative term sampling algorithm. We show that Algorithm 1 does not necessarily yield a probability measure over $\mathcal{F}$ as desired.

Given a finite set $K \subseteq T_\Sigma$ and a subterm-compatible order $\prec$, Algorithm 1 is a probabilistic algorithm, and thus outputs a function $\psi\colon sub[K] \to \mathcal{B}_\perp^*$ with some probability distribution. We would therefore like to define a model μ of $\mathcal{S}$ by defining $\mu(\Omega_\lambda)$ for each generator Ω_λ of $\mathcal{F}$ as the probability that executing Algorithm 1 on input $dom(\lambda)$ yields as output a function ψ_{ROM} such that, for each $t \in dom(\lambda)$, $\psi_{ROM}(t) \in \lambda(t)$.

Unfortunately, the next example shows that this is not well-defined in general. More precisely, we consider two terms, t and a, and show that the algorithm samples t and a to the same bitstring with a probability that depends on the input set K and the order relation $\prec$. Thus, letting $\lambda_b = \{t \mapsto b, a \mapsto b\}$ for each $b \in \mathcal{B}_\perp^*$, we have that the probability of the (measurable) set $\bigcup_{b \in \mathcal{B}_\perp^*} \Omega_{\lambda_b}$ depends on the input set K and the order relation $\prec$ considered.

Example 5. Suppose that $a, b, k \in \Sigma_0$ are such that $type(a) = T_{\mathcal{B}^{1024}}$, $type(b) = T_{\mathcal{B}^{1024}}$ and $type(k) = \mathsf{random}$. Consider executing Algorithm 1 on the set $\{t\}$, with $t = \left\{ \{a\}_{\mathsf{mod}(k),\mathsf{expn}(k)} \right\}_b^{-1}$. Algorithm 1 outputs a function $\psi : sub(t) \to \mathcal{B}^*_\perp$. Let us consider the probability that $\psi(t) = \psi(a)$. It is simple to check that both $\psi(t)$ and $\psi(a)$ are sampled by Algorithm 1 with uniform probability distribution from $\mathcal{B}^{1024}$. Therefore, the probability that $\psi(t) = \psi(a)$ is 2^{-1024}.

Now, consider executing Algorithm 1 on the set $\{t, \mathsf{inv}(k)\}$. If $t \prec \mathsf{inv}(k)$, then the execution of Algorithm 1 will be exactly the same until $\psi(s)$ is sampled for all terms $s \in sub(t)$, and $\psi(\mathsf{inv}(k))$ is only sampled afterwards. Therefore, $\psi(s)$ is sampled according to the same probability distribution for all $s \in sub(t)$, and the probability that $\psi(t) = \psi(a)$ is still 2^{-1024}. However, if $\mathsf{inv}(k) \prec b$, we have a probability of 2^{-1024} that $\psi(b) = \psi(\mathsf{inv}(k))$. If $\psi(b) = \psi(\mathsf{inv}(k))$, then we have $\psi(t) = \psi(a)$ with probability 1. Otherwise, $\psi(t)$ and $\psi(a)$ will still be sampled from $\mathcal{B}^{1024}$ with uniform probability distribution, and the probability that they are sampled to the same value is again 2^{-1024}. In this case, we conclude that

$$P[\psi(a) = \psi(t)] = 2^{-1024} \cdot (2 - 2^{-1024}) \neq 2^{-1024}.$$

Thus, the probability that $\psi(t) = \psi(a)$ depends on both the set of terms K input to the algorithm and the order $\prec$.

Nevertheless, the following result shows that, given a fixed a finite set of terms K and a subterm-compatible order $\prec$, Algorithm 1 does yield a probability distribution on the σ-algebra $\mathcal{F}_K$ generated by the set $\{\Omega_\lambda \mid \lambda \in \Lambda_K\}$. We remark that $\mathcal{F}_K$ is the set of σ-algebra of events that depend only on the instantiation of terms in the set K.

Theorem 2. *There is a unique probability distribution $\mu^{K,\prec} : \mathcal{F}_{sub[K]} \to [0, 1]$ such that, for each $\lambda \in \Lambda_K$, $\mu^{K,\prec}(\Omega_\lambda)$ is the probability that executing Algorithm 1 on input K and using the order $\prec$ yields a function ψ_{ROM} such that, for each $t \in K$, $\psi_{ROM}(t) \in \lambda(t)$.*

3.2 Revised term sampling in the ROM

To avoid problems like the one illustrated by Example 5 we need two additional hypotheses on the setup specification $\mathcal{S}$. We will explicitly distinguish a set of weak function symbols and consider a revised algorithm that uses this distinction. This revised algorithm is equivalent to Algorithm 1 when all functions are treated as weak. We show that, under these hypotheses, we can define a probability measure from this new sampling algorithm, while also simplifying the calculation of probabilities.

Weak terms. We assume fixed a set $\Sigma^W \subseteq \Sigma$ of *weak function symbols*. We say that a term $t \in T_\Sigma$ is *weak* if $head(t) \in \Sigma^W$, and denote by T^W the set of weak terms. Intuitively, weak function symbols are those that represent functions whose outputs are sampled from "small" sets, and a probabilistic model

must therefore take into account the possibility of collisions between them. By contrast, non-weak function symbols are those that represent functions whose outputs are sampled from large enough sets, so that ignoring the possibility of collisions changes our probability estimates only negligibly. Theorem 6, stated below, formalizes this idea.

Example 6. In our running example, we consider the set of weak function symbols $\Sigma^W = \{\mathsf{h}\} \cup \{a \in \Sigma_0 \mid a \subseteq_{PS} \mathsf{pw}\}$. That is, a term is weak if it is an hash or if it is derived from a humanly-chosen password.

Term sampling revisited. If K and K' are sets of terms and P is a partition of K, we let $P\mid_{K'} = \{p \cap K' \mid p \in P\}$. Note that $P\mid_{K'}$ is a partition of $K \cap K'$. We denote by $W(\psi_{ROM})$ the partition $P(\psi_{ROM})\mid_{T^W}$.

Our revised term sampling algorithm, targeted at solving the anomaly described in Example 5, is the same as Algorithm 1 with the exception that we replace the condition $t \approx_{P(\psi_{ROM})} t'$ by $t \approx_{W(\psi_{ROM})} t'$ in line 9.

Algorithm 3 (Term Sampling Algorithm)
Input: *a finite set of terms* $K \subseteq T_\Sigma$
Output: *a function* $\psi_{ROM} \colon sub[K] \to \mathcal{B}^*_\perp$ *Pseudocode identical to Algorithm 1, but with line 9 replaced by:*

9: **if** $\exists t' \in dom(\psi_{ROM}).\ t \approx_{W(\psi_{ROM})} t'$ ***and*** $\psi_{ROM}(t') \neq \perp$

This revised algorithm yields a probability distribution on $\mathcal{F}$ provided that the setup specifiction $\mathcal{S}$ satisfies two reasonable conditions, as follows.

Disjointness. The first condition we require on the specification $\mathcal{S}$ is that weak function symbols do not occur in the rewriting system R.

Intuitively, this disjointness condition implies that the equality of terms depends only on the equalities between their weak subterms. Therefore, sampling terms in a different order does not affect any equalities because terms are sampled only after all their subterms have been sampled. This is condition excludes cases like the one described in Example 5: because $\mathsf{inv} \notin \Sigma^W$, even if $\psi_{ROM}(b) = \psi_{ROM}(\mathsf{inv}(k))$, we never have $\left\{\{a\}_{\mathsf{mod}(k),\mathsf{expn}(k)}\right\}_b^{-1} \approx_{W(\psi_{ROM})} a$. The key idea is that equalities between non-weak terms may be disregarded, as the terms are equal only with negligible probability. We remark that ignoring equalities between non-weak terms, besides allowing us to consistently define a probability measure, also simplifies the calculation of probabilities. In Appendix A we present a simple algorithm for deciding $\approx_P$ (that is, given terms t and t', decide whether $t \approx_P t'$), and thus to perform the test in Line 9 of Algorithms 1 and its revised version.

Compatibility. The second condition we require on our setup is *compatibility*. Let K be a finite set of terms and P be a partition of K. Recall the definition of $\approx_P$ given in Section 3. We say that P is $\approx_R$*-closed* if, for all $t, t' \in K$, if $t \approx_P t'$, then there is $p \in P$ such that $t, t' \in p$. We are interested in partitions of weak

terms. Thus, given a finite set K, we denote by $\mathcal{P}_R^W(K)$ the set of $\approx_R$-closed partitions of $sub[K] \cap T^W$.

Example 7. Consider the set of terms $K = \left\{\mathsf{h}(\{\!|\{\!|t|\!\}_k|\!\}_{k'}^{-1}), \mathsf{h}(t)\right\}$, with $k \subseteq_{PS} \mathsf{pw}$, $k' \subseteq_{PS} \mathsf{pw}$, and $t \notin T^W$. Then, $\left\{\{k, k'\}, \left\{\mathsf{h}(\{\!|\{\!|t|\!\}_k|\!\}_{k'}^{-1})\right\}, \{\mathsf{h}(t)\}\right\}$ is a partition of $sub[K] \cap T^W$ that is not $\approx_R$-closed.

A *selection function for* K is a function $\iota\colon sub[K] \to PS \cup \{\bot\}$ such that, for each $t \in sub[K]$, either $\iota(t) = \bot$ or $head(\iota(t)) = head(t)$. Given $\omega \in \mathbf{\Omega}$, we say that ω *satisfies* ι if, for all $t = f(t_1, \ldots, t_n) \in sub[K]$, either $(\omega(t_1), \ldots, \omega(t_n)) \in [\![dom(\iota(t))]\!]$ and $\omega(t) \in [\![ran(\iota(t))]\!]$, or $(\omega(t_1), \ldots, \omega(t_n)) \notin dom_{\mathcal{S}}(f)$ and $\iota(t) = \omega(t) = \bot$. We denote by $I(K)$ the set of selection functions for K, and by $I_{\mathcal{S}}(K) \subseteq I(K)$ the set of selection functions ι for K such that there is $\omega \in \mathbf{\Omega}$ that satisfies ι. In Appendix A we show that, given a finite set of terms K, $I_{\mathcal{S}}(K)$ is a finite and computable set.

A selection function for a finite set K determines which property statement applies to each term in $sub[K]$. Note that, if $\omega \in \mathbf{\Omega}$ is an assignment satisfying PS and K is a finite set of terms, there exists exactly one selection function $\iota \in I(K)$ satisfied by ω: it is the function ι that associates each term $f(t_1, \ldots, t_n)$ to the unique property statement $ps \in PS_f$ such that $(\omega(t_1), \ldots, \omega(t_n)) \in [\![dom(ps)]\!]$, or $\bot$ if no such ps exists.

The compatibility condition is that, if K is a finite set of terms, $t \in sub[K]$, $P \in \mathcal{P}_R^W(K)$, $\iota \in I_{\mathcal{S}}(K)$, and $\iota(t) \neq \bot$, then there is $t' \in sub(t)$ such that $t \approx_{P|_{psub(t)}} t'$ and, whenever $t'' \in sub[K]$ and $t \approx_{P|_{psub(t)}} t''$, either $\iota(t'') = \bot$ or $[\![ran(\iota(t'))]\!] \subseteq [\![ran(\iota(t''))]\!]$.

Intuitively, the compatibility condition requires the equational theory $\approx_R$ and the property statements in PS to be compatible. It is a basic requirement that should be satisfied by any meaningful setup specification. The following example illustrates this.

Example 8 (Incompatibility between $\approx_R$ and PS). Consider a rewriting system R containing the symmetric decryption rewrite rule $\left\{\left|\{\!|x|\!\}_y\right|\right\}_y^{-1} \to x$ and the property statements $\{\!|T_{\mathcal{B}^{256}}|\!\}_{T_{\mathcal{B}^{256}}}^{-1} \subseteq T_{\mathcal{B}^{128}}$, $\{\!|T_{\mathcal{B}^{256}}|\!\}_{T_{\mathcal{B}^{256}}} \subseteq T_{\mathcal{B}^{256}}$. Let $t' = \{\!|\{\!|t|\!\}_k|\!\}_k^{-1}$, where $t, k \in \Sigma_0$ and $type(t) = type(k) = T_{\mathcal{B}^{256}}$. In this case, we have $\iota(t) = T_{\mathcal{B}^{256}}$ and $\iota(t') = T_{\mathcal{B}^{128}}$ for all selection functions $\iota \in I_{\mathcal{S}}(\{t, t'\})$. We have $t \approx_R t'$, $[\![ran(\iota(t))]\!] = T_{\mathcal{B}^{256}}$, and $[\![ran(\iota(t'))]\!] = T_{\mathcal{B}^{128}}$. Because $\mathcal{B}^{128} \cap \mathcal{B}^{256} = \emptyset$, it follows that there is no $\omega \in \mathbf{\Omega}$ that satisfies $\approx_R$ and PS.

Note, however, that having $\{\!|T_{\mathcal{B}^{256}}|\!\}_{T_{\mathcal{B}^{256}}}^{-1} \subseteq T_{\mathcal{B}^{256}}$, instead of $\{\!|T_{\mathcal{B}^{256}}|\!\}_{T_{\mathcal{B}^{256}}}^{-1} \subseteq T_{\mathcal{B}^{128}}$, we could have $type(t) = B$ for any non-empty set $B \subseteq \mathcal{B}^{256}$ without violating our compatibility condition.

Example 9. With the choice of Σ^W described in Example 6, our running example (described in Examples 1, 2, 3) and 4 satisfies the disjointness and compatibility conditions.

Probability measure. We show that, under the disjointness and compatibility hypotheses, the revised sampling algorithm yields a probability measure μ_{ROM} of $\mathcal{S}$. For each total subterm-compatible order $\prec$, each $\lambda \in \Lambda$, and each finite set of terms K such that $dom(\lambda) \subseteq K$, let $\mu^{K,\prec}(\lambda)$ be the probability that executing the sampling the revised version of Algorithm 1 on input $dom(\lambda)$ using the order $\prec$ yields a function $\psi_{ROM}\colon sub[K] \to \mathcal{B}^*_\perp$ such that $\psi_{ROM}(t) \in \lambda(t)$ for all $t \in K$.

Theorem 4. *Suppose that the disjointness and compatibility conditions are satisfied for the subterm-compatible orders $\prec$ and $\prec'$. Then, if $\lambda, \lambda' \in \Lambda$ are such that $\Omega_\lambda = \Omega_{\lambda'}$, we have $\mu^{\prec}(\lambda) = \mu^{\prec'}(\lambda')$.*

In light of Theorem 4, we define the function $\mu_{ROM}\colon \Omega_\Lambda \to [0,1]$ for each $\lambda \in \Lambda$ as $\mu_{ROM}(\Omega_\lambda) = \mu^{\prec}(\lambda)$ for any subterm-compatible order $\prec$.

Theorem 5. *There exists a unique extension of μ_{ROM} to $\mathcal{F}$ that is a probability measure. Using the same symbol μ_{ROM} to refer to this extension, we have $\mu_{ROM}(\Omega_\mathcal{S}) = 1$. Hence, μ_{ROM} is a model of $\mathcal{S}$.*

We adopt the abuse of notation used in the above theorem and use the same symbol μ_{ROM} to refer to the unique extension of μ_{ROM} to $\mathcal{F}$ that is a probability measure. In Appendix B we show how to compute probabilities using the probability measure μ_{ROM}.

3.3 Comparing the two probability measures

We describe the relationship between the probability measures $\mu^{K,\prec}$ described in Theorem 2 and the probability measure μ_{ROM} described in Theorem 5.

For each $f \in \Sigma$, let $L_f = \min_{ps \in PS_f} [\![ran(ps)]\!]$ and $L = \min_{f \in \Sigma \setminus \Sigma^w} L_f$. Note that, if we assume that non-weak terms are always sampled from "large" sets of bitstrings whenever they are defined, then L is large as well. Intuitively, Theorem 6 shows that, if this is the case, the different probability measures we have described coincide except on a set whose probability is "small". More precisely, the two probability measures coincide except on a set whose probability is a polynomial function of $1/L$.

Theorem 6. *For any finite set of terms K, there exists a set $\Omega(K)$ such that, for any subterm-compatible order $\prec$:*

(1) for any $\lambda \in \Lambda_K$, $\mu_\eta^{K,\prec}(\Omega_\lambda \cap \Omega(K)) = \mu_{ROM,\eta}(\Omega_\lambda \cap \Omega(K))$;
(2) there exists a constant $c(K)$ such that

$$\mu_\eta^{K,\prec}(\mathbf{\Omega} \setminus \Omega(K)) = \mu_{ROM,\eta}(\mathbf{\Omega} \setminus \Omega(K)) \leq c(K) \cdot |I_\mathcal{S}(K)| \cdot (1/L).$$

Note that the statement of Theorem 6 is stronger than merely bounding the difference in the probability of sets in Ω_Λ. For example, Theorem 6 implies that the probability of two terms being sampled to the same bitstring as measured by the two different probability measures is also bound by $c(K) \cdot |I_\mathcal{S}(K)| \cdot (1/L)$.

Asymptotic interpretation. Suppose that, for each $\eta \in \mathbb{N}$, $[\![\cdot]\!]_\eta$ is a type interpretation function and $\mathcal{S}_\eta = \langle \Sigma, \approx_R, \mathcal{T}, PS, [\![\cdot]\!]_\eta \rangle$ is a setup specification which satisfies the disjointness and compatibility conditions. Assume further that $1/L_\eta$ is negligible as a function of η, where $L_{f,\eta} = \min_{ps \in PS_f} [\![ran(ps)]\!]_\eta$ and $L_\eta = \min_{f \in \Sigma \setminus \Sigma^W} L_{f,\eta}$ for each $\eta \in \mathbb{N}$. Note that this condition is equivalent to requiring, for each function symbol $f \in \Sigma \setminus \Sigma^W$ and each $ps \in PS_f$, that $1/\left|[\![ran(ps)]\!]_\eta\right|$ is negligible as a function of η. Intuitively, this condition requires that non-weak terms, when defined, are always mapped to bitstrings sampled from large enough sets. Specifically, the sizes of the sets from which outputs of f are sampled should grow faster than any polynomial as a function of the parameter η.

Let $\mu_\eta^{K,\prec}$ (respectively, $\mu_{ROM,\eta}$) be the probability measure given by Theorem 2 (respectively, Theorem 5) when Algorithm 1 (respectively, the revised version of algorithm 1) is executed using the interpretation function $[\![\cdot]\!]_\eta$. Then, the following is a corollary of Theorem 6.

Corollary 1. *Let K be a finite set of terms, and suppose that $\left|I_{\mathcal{S}_\eta}(K)\right|$ grows polynomially as a function of η. For any finite set of terms K, there exists a set $\Omega(K)$ such that, for any subterm-compatible order $\prec$:*

(1) for any $\lambda \in \Lambda_K$, $\mu_\eta^{K,\prec}(\Omega_\lambda \cap \Omega(K)) = \mu_{ROM,\eta}(\Omega_\lambda \cap \Omega(K))$;
(2) $\mu_\eta^{K,\prec}(\mathbf{\Omega} \setminus \Omega(K)) = \mu_{ROM,\eta}(\mathbf{\Omega} \setminus \Omega(K))$, and both quantities are negligible as functions of η.

3.4 Computing probabilities

In Appendix B we show that the probability measure μ_{ROM} can be equivalently defined algebraically. This algebraic definition reduces the problem of computing probabilities of the form

$$P_{\mu_{ROM}}[t_1 \in B_1, \ldots, t_n \in B_n, t'_1 = t''_1, \ldots, t'_{n'} = t''_{n''}] \tag{1}$$

(where $B_1, \ldots, B_n$ are sets of bitstrings) to computing the sizes of intersections of sets in $\{B_1, \ldots, B_n\} \cup \mathcal{T}$. A full specification of the interpretations of types is not necessary.

Our prototype implementation computes probabilities of the form (1) for the cryptographic primitives and respective properties considered in our running example. The user must, however, specify the sizes of intersections of the sets of bitstrings $B_1, \ldots, B_n$ with the specified property types. Let $T = \{t_1, \ldots, t_n, t'_1, t''_1, \ldots, t'_{n'}, t'_{n''}\}$. Because we must consider $\approx_R$-closed partitions of the set $T_\Sigma^W \cap sub[T]$ of weak subterms of T, the complexity of the computation is exponential in the number of such weak subterms. However, for the setup specification considered in our running example, if T contains no subterms of the form $\pi_i(t)$ for some $i \in \{1, 2\}$ and some term t such that $head(t) \neq \langle \cdot, \cdot \rangle$, the complexity is linear in the number of non-weak subterms of T.

4 Off-line guessing

In this section we describe how properties of cryptographic primitives described in our setup specification $\mathcal{S}$ can be used to find and estimate the success probability of non-trivial off-line guessing attacks.

Suppose that a protocol should keep a term s secret, and that there is a small set $B \subset \mathcal{B}^*$ of bitstrings such that s represents a bitstring in this set. This may be the case, for instance, if s is computed from a password chosen by a human, or if s is the hash of a secret term. If an attacker can feasibly enumerate all bitstrings $b \in B$, he may use his knowledge to try to rule out the posibility that s represents each bitstring b. Ultimately, the attacker's goal is to learn s by excluding all but one bitstring from the set B. In this way an attacker can learn the term s even if he can not directly deduce it by constructing terms and reasoning equationally. This strategy is called *off-line guessing*, as the attacker need not interact with agents to verify his guess.

4.1 Attacker model

We will assume fixed an infinite set $\mathcal{N} \subseteq \Sigma_0$, such that $\Sigma_0 \setminus \mathcal{N}$ is finite. Intuitively, $\mathcal{N}$ contains the symbols representing random data generated by the agents, whereas $\Sigma_0 \setminus \mathcal{N}$ contains the symbols in the signature that represent cryptographically relevant constants (for instance, the bitstring 0 when modeling XOR).

A *substitution* is a partial function $\sigma\colon \mathcal{V} \nrightarrow T_\Sigma$. As usual, we abuse notation by using the same symbol σ for a substitution and its homomorphic extension to $T_\Sigma(X)$, and write $t\sigma$ instead of $\sigma(t)$.

We represent an attacker's knowledge by a frame [35], which is a pair $(\tilde{n}, \sigma)$, written $\nu\tilde{n}.\sigma$, where $\tilde{n} \subseteq \mathcal{N}$ is a finite set of names and $\sigma\colon \mathcal{V} \nrightarrow T_\Sigma$ is a substitution. Intuitively, names in $\tilde{n}$ represent fresh data randomly generated by honest agents and unknown to the attacker, and terms in the range of σ represent messages learned by the attacker, for instance by eavesdropping on the network. Given a frame $\phi = \nu\tilde{n}.\sigma$, we define $T_\phi = T_{\Sigma\setminus\tilde{n}}(dom(\sigma))$. We say that terms in T_ϕ are *ϕ-recipes*. Such terms represent the ways that an attacker can build terms using his knowledge. A term t can be *constructed* from ϕ if there is a ϕ-recipe ζ such that $\zeta\sigma = t$. The set of *terms constructible from* ϕ is $\sigma[T_\phi]$.

Suppose that an attacker whose knowledge is represented by a frame $\phi = \nu\tilde{n}.\sigma$ tries to mount an off-line guessing attack of a secret term s. We require that the set of bitstrings tried by the attacker is $[\![type(w)]\!]$ for some $w \in \mathcal{N}$ that does not occur in either $\tilde{n}$ or σ, and we model the attacker's guess by w. Letting $x \notin dom(\sigma)$ be a fresh variable, we consider the frames $\phi_s = \nu\tilde{n}_x.\sigma_s$ and $\phi_w = \nu\tilde{n}_x.\sigma_w$, where $\tilde{n}_x = \tilde{n} \cup \{x\}$, $\sigma_s = \sigma \cup \{x \mapsto s\}$, and $\sigma_w = \sigma \cup \{x \mapsto w\}$. Here, ϕ_s represents the attacker's knowledge using the right guess, while ϕ_w represents his knowledge when his guess is wrong.

Guess verifiers. We consider two ways in which an attacker can verify whether his guess w is correct. First, he can use his guess to construct a pair of terms

(t, t') that are equal under $\approx_R$ if $w = s$, but different if $w \neq s$. This is the usual definition used in symbolic methods and has been studied using the standard notion of static equivalence [25,27,35]. Second, he can use his guess to construct a term t whose corresponding bitstring satisfies some given property if $w = s$, and not necessarily otherwise.

To reason about the first of these strategies, we use the standard notion of static equivalence. We say that the pairs of terms used by the attacker in such tests are *equational verifiers*. To define this notion precisely, we use the notion of *subterm at position p*: Given a term t and $p \in \mathbb{N}^*$, we denote the *subterm of t at position p* by $t|_p$, where $t|_\epsilon = t$ and, for $t = f(t_1, \ldots, t_n)$, $t|_{i.p} = t_i|_p$ for $i \in \{1, \ldots, n\}$, where $i.p$ denotes the sequence of integers obtained by prepending i to the sequence p.

Definition 1. *The set $eqv(\phi, t)$ of* equational verifiers *of a term t (under ϕ) is the set of pairs (t, t') such that $t, t' \in T_{\phi_s}$, $t\sigma_s \approx_R t'\sigma_s$, $t\sigma_w \not\approx_R t'\sigma_w$, and there is no $p \in \mathbb{N}^* \setminus \{\epsilon\}$ such that these conditions hold for the pair $(t|_p, t'|_p)$.*

To model the second attacking strategy, we will consider a set $\mathcal{TT}$ of *test types*, which model the attacker's ability to test whether a bitstring is in a given set. Thus, to model a realistic attacker it is important to choose test types such that, whenever $T \in \mathcal{TT}$, it is computationally feasible to check whether a given bitstring is in $[\![T]\!]$.

Example 10. We will consider the following test types

- odd, with $[\![\mathsf{odd}]\!]$ corresponding to the set of 1024-bit bitstrings that represent an odd number, so that $|[\![\mathsf{odd}]\!]| = 2^{1023}$;
- nspf, with $[\![\mathsf{nspf}]\!]$ corresponding to the set of 1024-bit bitstrings representing numbers with no prime factors smaller than 10^6. We have $|[\![\mathsf{nspf}]\!]| \approx 2^{1024}/24$.

These test types are used to model off-line guessing attacks in Section 4.2.

Definition 2. *The set $tv(\phi, t)$ of* type verifiers *of t (under ϕ) is the set of pairs (t, TT) such that $t \in T_{\phi_s}$, $T \in \mathcal{TT}$, $P_{\mu_{ROM}}[\widehat{t\sigma_s} \in [\![TT]\!]] = 1$, $P_{\mu_{ROM}}[\widehat{t\sigma_w} \in [\![TT]\!]] \neq 1$, and there are no $p \in \mathbb{N}^* \setminus \{\epsilon\}$, $TT' \in \mathcal{TT}$ such that these conditions hold for $(t|_p, TT')$.*

Our requirements on the sub-positions of verifiers prevent us from having infinite sets of spurious verifiers. For instance, let $\mathsf{h}^0(t) = t$ and $\mathsf{h}^{n+1}(t) = \mathsf{h}(\mathsf{h}^n(t))$ for each $n \in \mathbb{N}$, and suppose that (t, t') is an equational verifier. Then, without our requirement on the sub-positions, so would be all pairs of the form $(\mathsf{h}^i(t), \mathsf{h}^i(t'))$ for $i \in \mathbb{N}$. However, if an attacker tests whether $t\sigma_w$ and $t'\sigma_w$ correspond to the same bitstring, there is no more information to be gained by testing whether the same holds for $h^i(t)\sigma_w$ and $h^i(t')\sigma_w$.

Computing equational verifiers is closely related to deciding static equivalence. Indeed, the algorithm for deciding static equivalence presented in [36] effectively computes equational verifiers as part of the decision procedure. Computing type verifiers is a less direct extension of existing algorithms. Nevertheless,

it is simple to check that, for any type verifier (t, T), some rewrite rule must apply to $t\sigma_s$ and not to $t\sigma_w$, since otherwise the normal form of $t\sigma_s$ is simply the normal form of $t\sigma_w$ with all occurrences of w replaced by s, and therefore the two terms have the same types. Computing recipes t that have this property (and such that none of their proper subterms have this property) is also part of the ordinary execution of algorithms for static equivalence.

4.2 Off-line guessing examples

We now present several examples of off-line guessing attacks. These examples illustrate that such attacks can result from implementation details that, while often trivial, are outside the scope of traditional symbolic methods. We show how such details can be modeled in our framework and used to estimate the probability of attacks. All probability calculations in this section rely on the setup specification described in our running example and are performed automatically by our implementation.

Example 11 (Attack on a stored password hash). This simple example considers an authentication server that stores password hashes instead of the users' passwords themselves. Let $s \in \Sigma_0$ be a weak password (*i.e.*, $type(s) = \mathsf{pw}$). Suppose that an attacker obtains its hash $h(s)$ and wants to use it to off-line guess s. The attacker's knowledge is represented by $\phi = \nu\tilde{n}.\sigma = \nu\{s\}.\{x_1 \mapsto h(s)\}$.

To analyze off-line guessing in our framework, consider the frames $\phi_s = \nu\{s, w\}.\{x_1 \mapsto h(s), x \mapsto s\}$ and $\phi_w = \nu\{s, w\}.\{x_1 \mapsto h(s), x \mapsto w\}$. Here, the set of type verifiers is empty ($tv(\phi, s) = \emptyset$), and the set of equational verifiers is $eqv(\phi, s) = \{(x_1, h(x)\}$.

Recall that $[\![\mathsf{pw}]\!] \subseteq \mathcal{B}^{256}$ and $(h[\mathcal{B}^{256}] \subseteq \mathcal{B}^{256}) \in PS$. Thus, we expect that each wrong guess satisfies the equation $h(s) = h(w)$ with probability 2^{-256}. Since $|[\![\mathsf{pw}]\!]| \approx 2^{24}$, there are $2^{24} - 1$ wrong guesses to consider. Hence, the expected number of guesses w satisfying $h(w) = h(s)$ is $1 + \frac{2^{24}-1}{2^{256}}$, and we obtain an estimated probability of success of $\frac{1}{1+\frac{2^{24}-1}{2^{256}}} \approx \frac{1}{1+\frac{1}{2^{232}}}$.

Example 12. The EKE (Encrypted Key Exchange) protocol is designed to allow two parties to exchange authenticated information using a weak symmetric key without allowing off-line guessing attacks. It is known that the redundancy of RSA public keys can be exploited to mount off-line guessing attacks on this protocol [29]. We show now how our methods can be used to estimate the success probability of this off-line guessing attack.

For representing these attacks, it is sufficient to consider the first step of the protocol. Let A and B be agents sharing a weak password $s \in \Sigma_0$, with $type(s) = \mathsf{pw}$. For the first message, A randomly samples a bitstring from $[\![\mathsf{random}]\!]$, represented by a term $r \in \Sigma_0$ such that $type(r) = \mathsf{random}$. Afterwards, she uses it to compute an RSA public key $\langle \mathsf{mod}(r), \mathsf{expn}(r) \rangle$. Then, A (symmetrically) encrypts this public key with the shared password s and sends the encryption to B. To keep our analysis simple, we assume that the participants encrypt the modulus and the exponent separately and send them over the

network as a pair of encryptions (instead of the encryption of the pair). Thus, this first message is represented by the term $\langle \{|\mathsf{mod}(r)|\}_s, \{|\mathsf{expn}(r)|\}_s \rangle$. See [29] for a full description of the protocol.

After observing this message in the network, the attacker's knowledge is described by the frame $\phi = \nu\tilde{n}.\sigma$, where $\sigma = \{x_1 \mapsto \langle \{|\mathsf{mod}(r)|\}_s, \{|\mathsf{expn}(r)|\}_s \rangle\}$ and $\tilde{n} = \{r\}$. The relevant frames for the analysis of off-line guessing attacks are $\phi_s = \nu\tilde{n}_w.\sigma_s$ and $\phi_w = \nu\tilde{n}_w.\sigma_w$, where $\tilde{n}_w = \tilde{n} \cup \{w\}$, $\sigma_s = \sigma \cup \{x_2 \mapsto s\}$, and $\sigma_w = \sigma \cup \{x_2 \mapsto w\}$.

In this case there are no equational verifiers: $eqv(\phi, s) = \emptyset$. However, while it may be infeasible to check whether the modulus is indeed the product of two large prime factors, an attacker can nevertheless use his guess w to decrypt the pair sent by A and test whether the resulting modulus has small prime factors and whether the exponent e is odd. Thus,

$$tv(\phi, s) = \left\{ (\{|\pi_1(x_1)|\}_{x_2}^{-1}, \mathsf{nspf}), (\{|\pi_2(x_1)|\}_{x_2}^{-1}, \mathsf{odd}) \right\}.$$

Each wrong guess satisfies the two type verifiers with probability

$$P_\mu \left[\widehat{\{|\pi_1(x_1)|\}_{x_2}^{-1}} \in [\![\mathsf{nspf}]\!], \widehat{\{|\pi_2(x_1)|\}_{x_2}^{-1}} \in [\![\mathsf{odd}]\!] \right] \approx \frac{1}{48}.$$

Since there are $2^{24} - 1$ wrong guesses, we estimate the probability of success of this off-line guessing attack as described above to be $\frac{1}{1+(2^{24}-1)/48} \approx 2^{-18.5}$.

Example 13. Consider now the same setup as in Example 12, except that only the exponent of the RSA public key is encrypted in the first message. The authors of EKE note [29] that the protocol is still vulnerable to off-line guessing attacks: Since the exponent of an RSA key is always odd, one can decrypt each encryption of a public key with each guess. For the right guess, decrypting each encryption will yield an odd exponent. The probability that a wrong guess achieves this decreases exponentially with the number of encryptions available to the attacker.

To formalize this in our setting, we let $\phi = \nu\tilde{n}.\sigma$ be the frame representing the attacker's knowledge, where $\sigma = \{x_i \mapsto \langle \mathsf{mod}(r_i), \{|\mathsf{expn}(r_i)|\}_s \rangle \mid i \in \{1, \ldots, n\}\}$ and $\tilde{n} = \{r_1, \ldots, r_n, s\}$. The frames ϕ_s and ϕ_w used are as expected: $\phi_s = \nu\tilde{n}_w.\sigma_s$ and $\phi_w = \nu\tilde{n}_w.\sigma_w$, where $\tilde{n}_w = \tilde{n} \cup \{w\}$, $\sigma_s = \sigma \cup \{x_{n+1} \mapsto s\}$, and $\sigma_w = \sigma \cup \{x_{n+1} \mapsto w\}$.

As before, there are no equational verifiers: $eqv(\phi, s) = \emptyset$. The set of type verifiers is given by $tv(\phi, s) = \left\{ (\{|\pi_2(x_i)|\}_{x_{n+1}}^{-1}, \mathsf{odd}) \mid i \in \{1, \ldots, n\} \right\}$. As in Example 12, we obtain $\frac{1}{1+\frac{2^{24}-1}{2^n}} = \frac{2^n}{2^n+2^{24}-1}$ as an estimate for the probability of success of this off-line guessing attack.

5 Conclusion

We presented a symbolic, automatable probabilistic framework for security protocol analysis. Our framework allows one to express properties of cryptographic

primitives beyond standard equational properties, thereby modeling a stronger attacker than in the standard Dolev-Yao model. We illustrated the usefulness of this approach by modeling non-trivial properties of RSA encryption and using them to analyze off-line guessing attacks on the EKE protocol, currently outside the scope of other symbolic methods.

We have proposed a probability distribution based on interpreting functions as random oracles subject to satisfying the properties of cryptographic primitives described in our setup. This is a non-trivial generalization of the random oracle model. By using this probability distribution, we can (automatically) reason about an attack's success probability. We provide a prototype implementation of our methods, which computes probabilities in our formalization of a Dolev-Yao attacker using RSA asymmetric encryption. Our implementation is available at [37].

More generally, our approach can be used to analyze a broad range of attacks and weaknesses of cryptographic primitives that could not previously be analyzed by symbolic models. These include some forms of cryptanalysis (such as differential cryptanalysis to AES, DES or hash functions, as in [38]) and side-channel attacks [33]. Short-string authentication, used in device pairing protocols [39], and distance-bounding protocols relying on rapid-bit exchange, such as [40], Both are ill-suited for analysis with existing symbolic methods as their analysis is intrinsically probabilistic.

As future work, we plan to integrate this approach with a symbolic protocol model-checker capable of generating protocol execution traces and the probabilities relevant for deciding whether a trace allows an attack. In the case of off-line guessing, this amounts to computing the sets of equational and type verifiers, a task closely related to that of deciding static equivalence. Since our probabilistic analysis can be performed automatically (as illustrated by our prototype), this allows our analysis to be fully automated. We expect that such an approach will allow us to find numerous new protocol attacks which depend on properties of the cryptographic primitives used.

References

1. V. Cortier, S. Delaune, and P. Lafourcade, "A survey of algebraic properties used in cryptographic protocols," *J. Comput. Secur.*, vol. 14, pp. 1–43, January 2006.
2. B. Blanchet, "An efficient cryptographic protocol verifier based on prolog rules," in *Proceedings of the 14th IEEE workshop on Computer Security Foundations*, CSFW '01, (Washington, DC, USA), pp. 82–96, IEEE Computer Society, 2001.
3. A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuellar, P. Hankes Drielsma, P.-C. Heám, J. Mantovani, S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Viganò, and L. Vigneron, "The AVISPA Tool for the Automated Validation of Internet Security Protocols and Applications," in *Proceedings of the 17th International Conference on Computer Aided Verification (CAV'05)* (K. Etessami and S. K. Rajamani, eds.), vol. 3576 of *LNCS*, Springer, 2005.
4. S. Goldwasser and S. Micali, "Probabilistic encryption," *J. Comput. Syst. Sci.*, vol. 28, no. 2, pp. 270–299, 1984.

5. M. Bellare and P. Rogaway, "Entity authentication and key distribution," in *CRYPTO* (D. R. Stinson, ed.), vol. 773 of *Lecture Notes in Computer Science*, pp. 232–249, Springer, 1993.
6. B. Warinschi, "A computational analysis of the Needham-Schrder-(Lwe) protocol," *Journal of Computer Security*, vol. 13, no. 3, pp. 565–591, 2005.
7. V. Cortier, S. Kremer, and B. Warinschi, "A survey of symbolic methods in computational analysis of cryptographic systems," *J. Autom. Reasoning*, vol. 46, no. 3-4, pp. 225–259, 2011.
8. B. Blanchet, "Security protocol verification: Symbolic and computational models," in Degano and Guttman [41], pp. 3–29.
9. M. Abadi and P. Rogaway, "Reconciling two views of cryptography (the computational soundness of formal encryption)," *J. Cryptology*, vol. 20, no. 3, p. 395, 2007.
10. M. Backes, A. Malik, and D. Unruh, "Computational soundness without protocol restrictions," in *ACM Conference on Computer and Communications Security* (T. Yu, G. Danezis, and V. D. Gligor, eds.), pp. 699–711, ACM, 2012.
11. M. Backes, B. Pfitzmann, and M. Waidner, "A composable cryptographic library with nested operations," in *ACM Conference on Computer and Communications Security* (S. Jajodia, V. Atluri, and T. Jaeger, eds.), pp. 220–230, ACM, 2003.
12. H. Comon-Lundh and V. Cortier, "Computational soundness of observational equivalence," in *ACM Conference on Computer and Communications Security* (P. Ning, P. F. Syverson, and S. Jha, eds.), pp. 109–118, ACM, 2008.
13. P. Adão, G. Bana, J. Herzog, and A. Scedrov, "Soundness and completeness of formal encryption: The cases of key cycles and partial information leakage," *Journal of Computer Security*, vol. 17, no. 5, pp. 737–797, 2009.
14. P. Laud and R. Corin, "Sound computational interpretation of formal encryption with composed keys," in *ICISC* (J. I. Lim and D. H. Lee, eds.), vol. 2971 of *Lecture Notes in Computer Science*, pp. 55–66, Springer, 2003.
15. V. Cortier, S. Kremer, R. Küsters, and B. Warinschi, "Computationally sound symbolic secrecy in the presence of hash functions," *IACR Cryptology ePrint Archive*, vol. 2006, p. 218, 2006.
16. S. Halevi, "A plausible approach to computer-aided cryptographic proofs," *IACR Cryptology ePrint Archive*, vol. 2005, p. 181, 2005.
17. M. Backes and P. Laud, "Computationally sound secrecy proofs by mechanized flow analysis," in *ACM Conference on Computer and Communications Security* (A. Juels, R. N. Wright, and S. D. C. di Vimercati, eds.), pp. 370–379, ACM, 2006.
18. B. Blanchet and D. Pointcheval, "Automated security proofs with sequences of games," in *CRYPTO* (C. Dwork, ed.), vol. 4117 of *Lecture Notes in Computer Science*, pp. 537–554, Springer, 2006.
19. B. Blanchet, "A computationally sound mechanized prover for security protocols," *IEEE Trans. Dependable Sec. Comput.*, vol. 5, no. 4, pp. 193–207, 2008.
20. G. Barthe, B. Grégoire, and S. Z. Béguelin, "Formal certification of code-based cryptographic proofs," in *POPL* (Z. Shao and B. C. Pierce, eds.), pp. 90–101, ACM, 2009.
21. G. Barthe, J. M. Crespo, B. Grégoire, C. Kunz, and S. Zanella Béguelin, "Computer-aided cryptographic proofs," in *3rd International Conference on Interactive Theorem Proving, ITP 2012*, pp. 12–27, Springer, 2012.
22. G. Barthe, M. Daubignard, B. M. Kapron, and Y. Lakhnech, "Computational indistinguishability logic," in *ACM Conference on Computer and Communications*

Security (E. Al-Shaer, A. D. Keromytis, and V. Shmatikov, eds.), pp. 375–386, ACM, 2010.
23. G. Bana and H. Comon-Lundh, "Towards unconditional soundness: Computationally complete symbolic attacker," in Degano and Guttman [41], pp. 189–208.
24. R. Corin, J. Doumen, and S. Etalle, "Analysing password protocol security against off-line dictionary attacks," *Electron. Notes Theor. Comput. Sci.*, vol. 121, pp. 47–63, February 2005.
25. M. Baudet, "Deciding security of protocols against off-line guessing attacks," in *Proceedings of the 12th ACM conference on Computer and communications security*, CCS '05, (New York, NY, USA), pp. 16–25, ACM, 2005.
26. Z. Li and W. Wang, "Rethinking about guessing attacks," in *ASIACCS* (B. S. N. Cheung, L. C. K. Hui, R. S. Sandhu, and D. S. Wong, eds.), pp. 316–325, ACM, 2011.
27. M. Abadi, M. Baudet, and B. Warinschi, "Guessing attacks and the computational soundness of static equivalence," *Journal of Computer Security*, pp. 909–968, December 2010.
28. "Sectools.org: Top 125 network security tools," Jan. 2013.
29. S. M. Bellovin and M. Merritt, "Encrypted Key Exchange: Password-based protocols secure against dictionary attacks," in *IEEE SYMPOSIUM ON RESEARCH IN SECURITY AND PRIVACY*, pp. 72–84, 1992.
30. J. Munilla and A. Peinado, "Off-line password-guessing attack to peyravian-jeffries's remote user authentication protocol," *Computer Communications*, vol. 30, no. 1, pp. 52–54, 2006.
31. S. Halevi and H. Krawczyk, "Public-key cryptography and password protocols," *ACM Trans. Inf. Syst. Secur.*, vol. 2, no. 3, pp. 230–268, 1999.
32. M. Abadi and B. Warinschi, "Password-based encryption analyzed," in *ICALP* (L. Caires, G. F. Italiano, L. Monteiro, C. Palamidessi, and M. Yung, eds.), vol. 3580 of *Lecture Notes in Computer Science*, pp. 664–676, Springer, 2005.
33. B. Köpf and D. A. Basin, "An information-theoretic model for adaptive side-channel attacks," in *ACM Conference on Computer and Communications Security* (P. Ning, S. D. C. di Vimercati, and P. F. Syverson, eds.), pp. 286–296, ACM, 2007.
34. M. Abadi and V. Cortier, "Deciding knowledge in security protocols under equational theories," *Theor. Comput. Sci.*, vol. 367, pp. 2–32, November 2006.
35. M. Abadi and C. Fournet, "Mobile values, new names, and secure communication," in *Proc. of the 28th ACM Symp. on Principles of Programming Languages*, POPL '01, (New York, NY, USA), pp. 104–115, ACM, 2001.
36. B. Conchinha, D. A. Basin, and C. Caleiro, "FAST: An efficient decision procedure for deduction and static equivalence," in *RTA* (M. Schmidt-Schauß, ed.), vol. 10 of *LIPIcs*, pp. 11–20, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011.
37. "`http://www.infsec.ethz.ch/people/brunoco`," 2013.
38. B. Montalto and C. Caleiro, "Modeling and reasoning about an attacker with cryptanalytical capabilities," *Electr. Notes Theor. Comput. Sci.*, vol. 253, no. 3, pp. 143–165, 2009.
39. S. Laur and K. Nyberg, "Efficient mutual data authentication using manually authenticated strings," in *CANS* (D. Pointcheval, Y. Mu, and K. Chen, eds.), vol. 4301 of *Lecture Notes in Computer Science*, pp. 90–107, Springer, 2006.
40. J. Munilla and A. Peinado, "Distance bounding protocols for RFID enhanced by using void-challenges and analysis in noisy channels," *Wireless Communications and Mobile Computing*, vol. 8, no. 9, pp. 1227–1232, 2008.

41. P. Degano and J. D. Guttman, eds., *Principles of Security and Trust - First International Conference, POST 2012, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2012, Tallinn, Estonia, March 24 - April 1, 2012, Proceedings*, vol. 7215 of *Lecture Notes in Computer Science*, Springer, 2012.

Appendix A

For each finite set of terms K, let Ψ_K be the set of functions $\psi\colon K \to \mathcal{B}^*_\perp$. Let $\Psi = \bigcup_{K \in \mathcal{P}_{fin}(T_\Sigma)} \Psi_K$, where $\mathcal{P}_{fin}(X)$ represents the set of finite subsets of X. For each $\psi \in \Psi$, let $\Omega_\psi = \{\omega \in \mathbf{\Omega} \mid t \in dom(\psi) \Rightarrow \omega(t) = \psi(t)\}$. Let $\Omega_\Psi = \{\Omega_\psi \mid \psi \in \Psi\}$ and, for each K, $\Omega_{\Psi_K} = \{\Omega_\psi \mid \psi \in \Psi_K\}$. Furthermore, for each $\lambda \in \Lambda_K$, define $\Psi(\lambda) = \{\psi \in \Psi_K \mid t \in K \Rightarrow \psi(t) \in \lambda(t)\}$. We remark that $\Omega_\lambda = \biguplus_{\psi \in \Psi(\lambda)} \Omega_\psi$, and that $\Psi(\lambda)$ is the only subset of Ψ_K such that the above equality holds.

Proof (Theorem 2). We first note that if $\lambda, \lambda' \in \Lambda_{sub[K]}$ are distinct, then $\Omega_\lambda \neq \Omega_{\lambda'}$. Therefore, $\mu^{K,\prec}$ is well-defined.

Letting $\lambda_{\mathcal{B}^*_\perp} = \{t \mapsto \mathcal{B}^*_\perp \mid t \in sub[K]\}$, we have $\Omega_{\lambda_{\mathcal{B}^*_\perp}} \in \mathcal{F}_K$ and $\mathbf{\Omega} = \Omega_{\lambda_{\mathcal{B}^*_\perp}}$. It is clear that $\mu^{K,\prec}(\Omega_{\lambda_{\mathcal{B}^*_\perp}}) = 1$.

Now, $\Omega \in \mathcal{F}_K$ can be written as $\Omega = \biguplus_{\psi \in \Psi} \Omega_\psi$ for some countable subset Ψ of Ψ_K. Sets of the form $\biguplus_{\psi \in \Psi'} \Omega_\psi$ for some countable subset Ψ' of Ψ_K are closed under complementation and countable unions. Conversely, for each $\psi \in \Psi_{sub[K]}$, let $\lambda_\psi\colon sub[K] \to \mathcal{P}(\mathcal{B}^*_\perp)$ be defined by $\lambda_\psi(t) = \{\psi(t)\}$ for each $t \in K$. It is clear that $\Omega_{\lambda_\psi} = \Omega_\psi$, and thus $\Omega_\psi \in \Omega_\Lambda \subseteq \mathcal{F}$ for all $\psi \in \Psi_K$.

We conclude that, for all $\Omega \in \mathbf{\Omega}$, $\Omega \in \mathcal{F}_K$ if and only if there exists a sequence of *distinct* $\psi_1, \ldots, \psi_k \in \Psi_{sub[K]}$ (with $k \in \mathbb{N} \cup \{\infty\}$) such that $\Omega = \biguplus_{i=1}^{k} \Omega_{\psi_i}$. Thus, for any sequence $\{\Omega_i\}_{i \in \mathbb{N}}$ such that $\Omega_i \in \mathcal{F}_K$ for all $i \in \mathbb{N}$ and $\Omega_i \cap \Omega_j = \emptyset$ whenever $i \neq j$, there exist $k_i \in \mathbb{N} \cup \{\infty\}$ and $\psi_{i,j} \in \Psi_{sub[K]}$ (with $j \in \{1, \ldots, k_i\}$) such that $\Omega_i = \biguplus_{j=1}^{k_i} \Omega_{\psi_{i,j}}$ and $\biguplus_{i \in \mathbb{N}} \Omega_i = \biguplus_{i \in \mathbb{N}} \biguplus_{j=1}^{k_i} \Omega_{\psi_{i,j}}$. The definition of $\mu^{K,\prec}$ implies that

$$\mu^{K,\prec}\left(\biguplus_{i \in \mathbb{N}} \Omega_i\right) = \mu^{K,\prec}\left(\biguplus_{i \in \mathbb{N}} \biguplus_{j=1}^{k_i} \Omega_{\psi_{i,j}}\right) = \sum_{i \in \mathbb{N}} \sum_{j=1}^{k_i} \mu(\Omega_{\psi_{i,j}}),$$

and, for each $i \in \mathbb{N}$,

$$\mu^{K,\prec}(\Omega_i) = \mu^{K,\prec}\left(\biguplus_{j=1}^{k_i} \Omega_{\psi_{i,j}}\right) = \sum_{j=1}^{k_i} \mu(\Omega_{\psi_{i,j}}).$$

Thus, $\mu^{K,\prec}$ is σ-additive:

$$\mu^{K,\prec}\left(\biguplus_{i \in \mathbb{N}} \Omega_i\right) = \sum_{i \in \mathbb{N}} \sum_{j=1}^{k_i} \mu(\Omega_{\psi_{i,j}}) = \sum_{i \in \mathbb{N}} \mu(\Omega_i),$$

concluding the proof. □

In the following, we assume that K is a finite set of terms, $P \in \mathcal{P}^W_R(K)$, $\prec$ is a subterm-compatible order, and $\mathcal{N}_p$ is an infinite set of names (disjoint from Σ). We say that a P-renaming is an injective function $\tau\colon P \rightarrowtail \mathcal{N}_p$ mapping P-equivalence classes to names in $\mathcal{N}_p$. We will denote by P^* be the partition of

$(T_\Sigma^W \cap sub[K]) \cup \tau[P]$ given by $P^* = \{p \cup \{\tau(p)\} \mid p \in P\}$. Moreover, we define $\kappa_P^\tau : T_\Sigma(\mathcal{N}_p) \to T_\Sigma(\mathcal{N}_p)$ inductively by $\kappa_P^\tau(t) = \tau(p)$ if there is $p \in P^*$ and $t' \in p$ such that $t' \approx_{P^*} t$ and $\kappa_P^\tau(f(t_1, \ldots, t_n)) = f(\kappa(t_1), \ldots, \kappa(t_n))\downarrow$ otherwise. We note that, for all $t \in T_\Sigma(\mathcal{N}_p)$, $\kappa_P^\tau(\kappa_P^\tau(t)) = \kappa_P^\tau(t)$ and $\kappa_P^\tau(t) \approx_{P^*} t$. We can drop both τ and P when they are clear from context.

Lemma 1. *For all $t, t' \in T_\Sigma$, all finite sets of terms K, all $P \in \mathcal{P}_R^W(K)$, and all P-renamings τ, we have $t \approx_P t'$ if and only if $\kappa_P^\tau(t) = \kappa_P^\tau(t')$.*

Proof. Let $\sim$ be the relation on terms given by $t \sim t'$ if and only if $\kappa(t) = \kappa(t')$. Since $t, t' \in T_\Sigma$, we have $t \approx_P t'$ if and only if $t \approx_{P^*} t'$; therefore, it is sufficient to prove that $\sim = \approx_{P^*}$. Because $t \approx_{P^*} \kappa(t)$, it is clear that $\sim \subseteq \approx_{P^*}$.

It remains to prove that $\approx_{P^*} \subseteq \sim$. It is sufficient to show:

(1) $\sim$ is a congruence relation;
(2) $\approx_R \subseteq \sim$;
(3) if $p \in P$ and $t, t' \in p$, then $t \sim t'$.

(1) It is clear that $\sim$ is reflexive, symmetric and transitive. Suppose that $t_1 \sim t_1'$, $\ldots$, $t_n \sim t_n'$ and $f \in \Sigma_n$. Let $t = f(t_1, \ldots, t_n)$ and $t' = f(t_1', \ldots, t_n')$. If there is $p \in P$ and $t'' \in p$ such that $t \approx_P t''$, then, because $\sim \subseteq \approx_{P^*}$, we have $t' \approx_{P^*} t \approx_{P^*} t''$; and because $t, t', t'' \in T_\Sigma$, we have $t' \approx_P t''$. Thus, $\kappa(t) = \tau(p) = \kappa(t')$, and $t \sim t'$. Otherwise, we have

$$\kappa(t) = f(\kappa(t_1), \ldots, \kappa(t_n)) = f(\kappa(t_1'), \ldots, \kappa(t_n')) = \kappa(t').$$

Thus, $\sim^*$ is a congruence relation.

(2) In light of (1), it is sufficient to show that $t \sim t\downarrow$ for all t. We note that, for every term $t = f(t_1, \ldots, t_n)$, either $\kappa(t) = f(\kappa(t_1), \ldots, \kappa(t_n))\downarrow$ or $\kappa(t) = \tau(p)$ for some $p \in P$. Therefore, $\kappa(t)$ is in normal form for all t.

We prove the result by induction on t. If $t = c()$ for some $c \in \Sigma_0$, then $t = t\downarrow$, and the result is clear. Otherwise, let $t = f(t_1, \ldots, t_n)$, with $n > 0$.

If there is $p \in P$ and $t' \in p$ such that $t \approx_P t'$, we also have $t\downarrow \approx_P t'$. Therefore, $\kappa(t) = \kappa(t') = \tau(p)$, and thus $t \sim t\downarrow$.

If this is not the case, then $\kappa(t) = f(\kappa(t_1), \ldots, \kappa(t_n))\downarrow$. By the induction hypothesis, $\kappa(t_1) = \kappa(t_1\downarrow)$, $\ldots$, $\kappa(t_n) = \kappa(t_n\downarrow)$. If $f(t_1\downarrow, \ldots, t_n\downarrow)$ is in normal form, then $t\downarrow = f(t_1\downarrow, \ldots, t_n\downarrow)$, and

$$\kappa(t) = f(\kappa(t_1), \ldots, \kappa(t_n))\downarrow = f(\kappa(t_1\downarrow), \ldots, \kappa(t_n\downarrow))\downarrow = \kappa(f(t_1\downarrow, \ldots, t_n\downarrow)) = \kappa(t\downarrow).$$

Suppose then that $f(t_1\downarrow, \ldots, t_n\downarrow)$ is not in normal form. Then, there is $(l \to r) \in R$ and a substitution σ such that $f(t_1\downarrow, \ldots, t_n\downarrow) = l\sigma$. Let $\sigma_\kappa : dom(\sigma) \to T_\Sigma(\mathcal{N}_p)$ be the substitution such that, for each $x \in dom(\sigma)$, $x\sigma_\kappa$ is obtained from $x\sigma$ by taking each outermost subterm s of $x\sigma$ for which there is $p \in P$ and $t' \in p$ such that $s \approx_P t'$, and replacing s by $\tau(p)$. Each proper subterm of $f(t_1\downarrow, \ldots, t_n\downarrow)$ is in normal form, and thus, recalling that weak function symbols do not occur in R, we have $s \in T_\Sigma^W$ for all such subterms s. Therefore, $x\sigma_\kappa$ is

obtained from $x\sigma$ by replacing subterms of $x\sigma$ whose head symbols are in Σ^W by the corresponding names in $\mathcal{N}_p$. By a similar reasoning, for each $i \in \{1, \ldots, n\}$, we can obtain $\kappa(t_i{\downarrow})$ by taking the outermost subterms s of $t_i{\downarrow}$ for which there is $p \in P$ and $t' \in p$ such that $s \approx_P t'$, replacing them with their corresponding name $\tau(p)$, and taking the normal form of the resulting term. Thus, $\kappa(t_i{\downarrow}) = (t_i\sigma_\kappa){\downarrow}$. Noting again that such subterms s must have head symbols in Σ^W which do not occur in R and using the fact that $\rightarrow_R$ is convergent, we conclude that

$$\kappa(t) = f(\kappa(t_1{\downarrow}), \ldots, \kappa(t_n{\downarrow})){\downarrow} = f((t_1\sigma_\kappa){\downarrow}, \ldots, (t_n\sigma_\kappa){\downarrow}){\downarrow} = l\sigma_\kappa{\downarrow}\ .$$

Similarly, $r\sigma_\kappa{\downarrow} = \kappa(r\sigma)$. Furthermore, $r\sigma$ is a proper subset of $l\sigma$, and thus $r\sigma = (r\sigma){\downarrow} = t{\downarrow}$. We conclude that

$$\begin{aligned}\kappa(t) = f(\kappa(t_1), \ldots, \kappa(t_n)){\downarrow} &= f(\kappa(t_1{\downarrow}), \ldots, \kappa(t_n{\downarrow})){\downarrow} \\ &= (l\sigma_\kappa){\downarrow} = (r\sigma_\kappa){\downarrow} = \kappa(r\sigma) = \kappa(t{\downarrow}),\end{aligned}$$

concluding the proof.

(3) Whenever $p \in P$ and $t \in p$, we have $\kappa(t) = \tau(p)$. Property (3) follows. □

If K is a finite set of terms and $P \in \mathcal{P}_R^W(K)$, a *P-renaming* is an injective function $\tau\colon P \to \mathcal{N}^+$.

Algorithm 7 (P, R-renaming)
Input: *$K, P \in \mathcal{P}_R^W(K)$, a subterm-compatible order $\prec$, and a P-renaming τ*
Output: *functions $\tau^+, \tau^*\colon sub[K] \to T_\Sigma(\mathcal{N}_p)$*

1: $\tau^+, \tau^* \leftarrow \emptyset$
2: *let $t_1, \ldots, t_k$ be such that $t_1 \prec \ldots \prec t_k$ and $sub[K] = \{t_1, \ldots, t_k\}$*
3: **for** i **from** 1 **to** k
4: *let $t_i = f(t'_1, \ldots, t'_n)$*
5: $\tau^+(t_i) \leftarrow f(\tau^*(t'_1), \ldots, \tau^*(t'_n)){\downarrow}$
6: **if** *there is $j \leq i$ s.t. $\tau^+(t_j) = \tau^+(t_i)$ and $t_j \in p$ for some $p \in P$*
7: $\tau^*(t_i) \leftarrow \tau(p)$
8: **else** $\tau^*(t_i) \leftarrow \tau^+(t_i)$
9: **return** τ^+, τ^*

Lemma 2. *Consider the function τ^* output by Algorithm 7 on input $(K, P, \prec, \tau)$. For all $t \in sub[K]$, we have $\tau^*(t) = \kappa_P^\tau(t)$.*

Proof. The proof is by induction on $|sub[K]|$. The result is clear if $|sub[K]| = 0$. Suppose then that $|sub[K]| = n + 1$, and that $\tau^*(t_i) = \kappa(t_i)$ for all $i \in \{1, \ldots, n\}$, where $sub[K] = \{t_1, \ldots, t_{n+1}\}$ and $t_1 \prec \ldots \prec t_{n+1}$. Let $t_{n+1} = f_{n+1}(t_{n+1,1}, \ldots, t_{n+1,k_{n+1}})$.

If there is $j \leq n + 1$ and $p \in P$ such that $t_j \in p$ and $\tau^+(t_j) = \tau^+(t_{n+1})$, then $t_j \approx_P t_{n+1}$. This is because, letting $t_j = f_j(t_{j,1}, \ldots, t_{j,k_j})$ and using the induction hypothesis, we have

$$\tau^+(t_j) = f_j(\tau^*(t_{j,1}), \ldots, \tau^*(t_{j,k_j})){\downarrow} \approx_{P^*} t_j,$$

and, similarly,

$$\tau^+(t_{n+1}) = f_{n+1}(\tau^*(t_{n+1,1}), \ldots, \tau^*(t_{n+1,k_{n+1}}))\downarrow \approx_{P^*} t_{n+1}.$$

This implies that $t_j \approx_{P^*} t_{n+1}$, and because $t_j, t_{n+1} \in T_\Sigma$, we have $t_j \approx_P t_{n+1}$. Thus, we have $\kappa(t_{n+1}) = \tau^*(t_{n+1}) = \tau(p)$.

Suppose then that there is no $p \in P$ and $t \in p$ such that $t_{n+1} \approx_P t$. The reasoning above implies that there is no $j \leq n+1$ and $p \in P$ such that $t_j \in p$ and $\tau^+(t_j) = \tau^+(t_{n+1})$. Thus,

$$\begin{aligned}\tau^*(t_{n+1}) = \tau^+(t_{n+1}) &= f_{n+1}(\tau^*(t_{n+1,1}), \ldots, \tau^*(t_{n+1,k_{n+1}}))\downarrow \\ &= f_{n+1}(\kappa(t_{n+1,1}), \ldots, \kappa(t_{n+1,k_{n+1}}))\downarrow = \kappa(t_{n+1}),\end{aligned}$$

using the induction hypothesis.

It remains to consider the case that there is $p \in P$ and $t \in p$ such that $t_{n+1} \approx_P t$. In this case, we have $\kappa(t_{n+1}) = \tau(p)$; therefore, we have to prove that $\tau^*(t_{n+1}) = \tau(p)$ as well. Because $t \in sub[K]$, we have $t = t_j$ for some $j \leq n+1$. Letting $t_{n+1} = f_{n+1}(t_{n+1,1}, \ldots, t_{n+1,k_{n+1}})$, the induction hypothesis yields

$$\begin{aligned}\tau^+(t_{n+1}) &= f_{n+1}(\tau^*(t_{n+1,1}), \ldots, \tau^*(t_{n+1,k_{n+1}}))\downarrow \\ &= f_{n+1}(\kappa(t_{n+1,1}), \ldots, \kappa(t_{n+1,k_{n+1}}))\downarrow .\end{aligned}$$

Letting $t_j = f_j(t_{j,1}, \ldots, t_{j,k_j})$ and using the induction hypothesis again, we have

$$\begin{aligned}\tau^+(t_j) &= f_j(\tau^*(t_{j,1}), \ldots, \tau^*(t_{j,k_j}))\downarrow \\ &= f_j(\kappa(t_{j,1}), \ldots, \kappa(t_{j,k_j}))\downarrow \\ &= f_j(\kappa(t_{j,1}), \ldots, \kappa(t_{j,k_j})),\end{aligned}$$

because $\kappa(t)$ is in normal form for all t and $f_j \in \Sigma^W$ does not occur in R. Combining the fact that $\tau^+(t) \approx_{P^*} t$ for all t with the two equalities above, it follows that

$$f_{n+1}(\kappa(t_{n+1,1}), \ldots, \kappa(t_{n+1,k_{n+1}}))\downarrow \approx_{P^*} f_j(\kappa(t_{j,1}), \ldots, \kappa(t_{j,k_j})).$$

Furthermore, $\kappa(t)$ is always in normal form: therefore, we have either

$$f_{n+1}(\kappa(t_{n+1,1}), \ldots, \kappa(t_{n+1,k_{n+1}}))\downarrow = f_{n+1}(\kappa(t_{n+1,1}), \ldots, \kappa(t_{n+1,k_{n+1}}))$$

or

$$f_{n+1}(\kappa(t_{n+1,1}), \ldots, \kappa(t_{n+1,k_{n+1}}))\downarrow \in sub(f_{n+1}(\kappa(t_{n+1,1}), \ldots, \kappa(t_{n+1,k_{n+1}}))).$$

In the latter case, we note that, whenever $s \in sub(\kappa(t))$, we have $\kappa(s) = s$: Therefore, we have

$$f_{n+1}(\kappa(t_{n+1,1}), \ldots, \kappa(t_{n+1,k_{n+1}}))\downarrow = \kappa(f_{n+1}(\kappa(t_{n+1,1}), \ldots, \kappa(t_{n+1,k_{n+1}}))\downarrow),$$

and because $t_{n+1} \approx_P t_j$ and $t_j \in p$, we have

$$\tau^+(t_{n+1}) = \kappa(f_{n+1}(\kappa(t_{n+1,1}), \ldots, \kappa(t_{n+1,k_{n+1}}))\downarrow) = \tau(p),$$

as desired.

Let us now consider the case that

$$f_{n+1}(\kappa(t_{n+1,1}),\ldots,\kappa(t_{n+1,k_{n+1}}))\downarrow = f_{n+1}(\kappa(t_{n+1,1}),\ldots,\kappa(t_{n+1,k_{n+1}})).$$

If $f_{n+1} \notin \Sigma^W$, then $f_{n+1}(\kappa(t_{n+1,1}),\ldots,\kappa(t_{n+1,k_{n+1}})) \not\approx_{P^*} t_j$, since $head(t_j) \in \Sigma^W$; it follows that $t_{n+1} \not\approx_{P^*} t_j$, a contradiction. Therefore, we must have $f_{n+1} \in \Sigma^W$. In this case, $t_{n+1} \in T^W_\Sigma$, and because P is $\approx_R$-closed, we have $t_{n+1} \in p$ and $\tau^*(t_{n+1}) = \tau(p)$. □

If K is a finite set of terms and $\iota \in I(K)$, we will write $\omega \models \iota$ to denote that ω satisfies ι. If $\psi \in \Psi_{sub[K]}$ for some finite set of terms K, we say that ψ satisfies ι, and write $\psi \models \iota$, if, for all $t = f(t_1,\ldots,t_n) \in K$, $(\psi(t_1),\ldots,\psi(t_n)) \in [\![dom(\iota(t))]\!]$ and $\psi(t) \in [\![ran(\iota(t))]\!]$.

Lemma 3. *Let K be a finite set of terms and $\psi \in \Psi_{sub[K]}$. Then, there exists at most one $\iota \in I(K)$ such that $\psi \models \iota$.*

Proof. For each $t \in sub[K]$, letting $t = f(t_1,\ldots,t_n)$, there is at most one $ps \in PS_f$ such that $(\psi(t_1),\ldots,\psi(t_n)) \in [\![dom(ps)]\!]$. If $\psi \models \iota$, we must have $\iota(t) = ps$ in this case, and $\iota(t) = \bot$ if no such ps exists; thus, there is at most one ι that is satisfied by ψ. □

In light of Lemma 3, if $K = sub[K]$ and $\psi \in \Psi_K$, we will write $\iota(\psi)$ for the only $\iota \in I(K)$ such that $\psi \models \iota$, and set $\iota(\psi) = \bot$ if no such ι exists. Moreover, if $\psi\colon sub[K] \to \mathcal{B}^*_\bot$, we say that ψ satisfies $\approx_{W(\psi)}$ if, whenever $t, t' \in sub[K]$ are such that $t \approx_{W(\psi)} t'$, we have either $\psi(t) = \psi(t')$, $\psi(t) = \bot$, or $\psi(t') = \bot$.

Lemma 4. *At any point of any execution of Algorithm 3, there is a finite set K' such that $dom(\psi_{ROM}) = sub[K']$. Furthermore, ψ_{ROM} satisfies $\approx_{W(\psi_{ROM})}$ and exactly one selection function $\iota \in I_{\mathcal{S}}(K')$.*

Proof. Consider an execution of Algorithm 3 for input K and using a subterm-compatible order $\prec$. Let $t_1,\ldots,t_n$ be such that $sub[K] = \{t_1,\ldots,t_n\}$ and $t_1 \prec \ldots \prec t_n$. Algorithm 3 samples $\psi_{ROM}(t_1),\ldots,\psi_{ROM}(t_n)$, in this order. We prove the result on the execution of Algorithm 3. For all $i \in \{0,\ldots,n\}$, let $K_i = \{t_1,\ldots,t_i\}$ and let $\psi_{ROM,i}$ be the function ψ_{ROM} used in the algorithm after the i-th execution of the cycle in lines 3-13, so that $dom(\psi_{ROM,i}) = K_i$.

In the beginning of the execution, we have $K_0 = dom(\psi_{ROM}) = \emptyset$ and $I_{\mathcal{S}}(K_0) = \{\emptyset\}$. Thus, the result is clear. Suppose then that the result holds for $j \in \{1,\ldots,i\}$, and consider the $i+1$-th execution of the cycle.

To prove that $\psi_{ROM,i+1}$ satisfies $\approx_{W(\psi_{ROM,i+1})}$, let k, k' be indexes such that $t_k \approx_{W(\psi_{ROM,i+1})} t_{k'}$. Note that, if $i+1 \in \{k, k'\}$, then lines 5 and 9 of the algorithm description imply the result. There are two cases: either (1) $t_k \approx_{W,\psi_{ROM,i}} t_{k'}$, or (2) not. In case (2), lemmas 1 and 2 imply that $i+1 \in \{k, k'\}$, by noting that $\tau^*(t_k)$ and $\tau^*(t_{k'})$ do not depend on t_{i+1}. Therefore, the result holds. Otherwise, either $i+1 \in \{k, k'\}$, and the result holds as before, or

$i+1 \notin \{k, k'\}$, and the result is implied by the induction hypothesis (since $\psi_{ROM,i+1}(t_m) = \psi_{ROM,i}(t_m)$ whenever $1 \leq m \leq i$).

Furthermore, we have $psub(t_{i+1}) \subseteq K_i$ (because $\prec$ is a subterm-compatible order), and the induction hypothesis yields

$$sub[K_{i+1}] = sub[K_i] \cup \{t_{i+1}\} = K_i \cup \{t_{i+1}\} = K_{i+1}.$$

It remains to prove that ψ_{ROM} satisfies $\approx_{W(\psi_{ROM})}$. Lemma 3 implies that, for any j, there is at most one $\iota \in I_{\mathcal{S}}(K_j)$ such that $\psi_{ROM,j}$ satisfies ι. Thus, it is sufficient to show that $\psi_{ROM,i+1}$ satisfies some $\iota \in I_{\mathcal{S}}(K_{i+1})$. By the induction hypothesis, there exists $\iota \in I_{\mathcal{S}}(K_i)$ such that ψ_{ROM} satisfies ι. Let $t_{i+1} = f_{i+1}(t'_{i+1,1}, \ldots, t'_{i+1,k})$. If $(\psi_{ROM,i}(t'_{i+1,1}), \ldots, \psi_{ROM,i}(t'_{i+1,k})) \notin dom_{\mathcal{S}}(f_{i+1})$, then $\psi_{ROM,i+1}(t_{i+1}) = \bot$ will be sampled to $\bot$. Then, letting $\iota' = \iota \cup \{t_{i+1} \mapsto \bot\}$, it is clear that $\psi_{ROM,i+1}$ satisfies $\iota' \in I_{\mathcal{S}}(K_{i+1})$.

Otherwise, there is $ps \in PS_{f_{i+1}}$ such that

$$(\psi_{ROM,i}(t'_{i+1,1}), \ldots, \psi_{ROM,i}(t'_{i+1,k})) \in [\![dom(ps)]\!].$$

Let $\iota' = \iota \cup \{t_{i+1} \mapsto ps\}$; we have $\iota' \in I_{\mathcal{S}}(K_{i+1})$, and it is sufficient to show that $\psi_{ROM,i+1}$ satisfies ι'. Let $t' = f'(t'_1, \ldots, t'_{k'})$ be the minimal (with respect to $\prec$) term in K_{i+1} such that $t' \approx_{P(\psi_{ROM,i})} t$ and

$$(\psi_{ROM,i}(t'_1), \ldots, \psi_{ROM,i}(t'_{k'})) \in dom_{\mathcal{S}}(f')$$

(note that we may have $t' = t_{i+1}$). Then, we have $\psi_{ROM,i+1}(t_{i+1}) = \psi_{ROM,i}(t')$, and $\psi_{ROM,i}(t')$ is sampled from $[\![ran(\iota(t'))]\!]$. The compatibility condition implies that, for all $\iota'' \in I_{\mathcal{S}}(K_{i+1})$ and all $t'' \in K_{i+1}$, $[\![ran(\iota''(t'))]\!] \subseteq [\![ran(\iota''(t''))]\!]$. It follows that $[\![ran(\iota'(t'))]\!] \subseteq [\![ran(\iota'(t_{i+1})]\!]$; therefore,

$$\psi_{ROM,i+1}(t_{i+1}) = \psi_{ROM,i}(t') \in [\![ran(\iota'(t'))]\!] \subseteq [\![ran(\iota'(t_{i+1}))]\!],$$

and thus $\psi_{ROM,i+1}$ satisfies ι'. □

In the following, if $t \in sub[K]$ and $\prec$ is a subterm-compatible order, we denote by $\widehat{t}^{K,\prec}$ the random variable representing $\psi_{ROM}(t)$, where ψ_{ROM} is the output of Algorithm 3 when executed on input K using the order $\prec$.

Lemma 5. *Let K be a finite set of terms, $\prec$ be a subterm-compatible order, and $\psi\colon sub[K] \to \mathcal{B}^*_\bot$. Suppose that ψ satisfies $\approx_{W(\psi)}$ and there is $\iota \in I_{\mathcal{S}}(K)$ such that ψ satisfies ι. Let $t' = f(t'_1, \ldots, t'_n)$ be a term such that $t' \notin K$, $psub(t') \subseteq sub[K]$, and $t \prec t'$ for all $t \in K$. Let $b \in \mathcal{B}^*_\bot$, $K' = K \cup \{t'\}$ and $\psi' = \psi \cup \{t' \mapsto b\}$. Define*

$$P[K, \psi', t'] = P[\widehat{t'}^{K',\prec} = b \mid \widehat{t}^{K',\prec} = \psi(t) \text{ for all } t \in sub[K]].$$

Consider the function τ^+ output by Algorithm 7 for the input (K', $W(\psi')$, $\prec$, τ) for some $W(\psi')$-renaming τ.

If there is no $ps \in PS$ such that $head(ps) = head(t')$ and $(\psi'(t'_1), \ldots, \psi'(t'_n)) \in dom(ps)$, then $P[K, \psi', t'] = 1$ if $b = \bot$ and $P[K, \psi', t'] = 0$ otherwise.

Otherwise, letting ps be the only property statement satisfying the condition above, we have:

– *if there is* $t \in sub[K]$ *such that either (1)* $\tau^+(t) = \tau^+(t')$ *or (2)* $\tau^+(t') \in \mathcal{N}^+$ *and* $\tau^*(t) = \tau^+(t')$*, then:*
  - *if* $\psi(t) = b$*, then* $P[K, \psi', t'] = 1$*;*
  - *if* $\psi(t) \neq b$*, then* $P[K, \psi', t'] = 0$*;*
– *if such a* t *does not exist, then* $P[K, \psi', t'] = 1/\left|[\![ran(ps)]\!]\right|$ *if* $b \in [\![ran(ps)]\!]$ *and* $P[K, \psi', t'] = 0$ *otherwise.*

Proof. Consider an execution of Algorithm 3 on input $K \cup \{t'\}$ using the order $\prec$. Then, $P[K, \psi', t']$ corresponds to the probability that such an execution outputs a function $\psi_{ROM} = \psi'$ given that, before the last step of the execution (when $\psi_{ROM}(t')$ is sampled), we have $\psi_{ROM} = \psi$ (corresponding to the sampling of the terms in $sub[K]$). It is clear from the definition of the algorithm that, if there is no $ps \in PS$ such that $head(ps) = head(t')$ and $(\psi'(t'_1), \ldots, \psi'(t'_n)) \in dom(ps)$, then $\psi_{ROM}(t')$ is sampled to $\perp$, and therefore $P[K, \psi', t'] = 1$ if $b = \perp$ and $P[K, \psi', t'] = 0$ otherwise.

If there is $ps \in PS$ such that the above condition is satisfied and there is $t \in sub[K]$ such that either (1) $\tau^+(t) = \tau^+(t')$ or (2) $\tau^+(t') \in \mathcal{N}^+$ and $\tau^*(t) = \tau^+(t')$, we have $t \approx_{W(\psi)} t'$. Therefore, $\psi_{ROM}(t')$ is sampled as $\psi_{ROM}(t') = \psi_{ROM}(t)$, and the result follows.

Now we prove that, for any t, if $t \approx_{W(\psi)} t'$, then either (1) or (2) holds. If $\tau^+(t') \in T^W_\Sigma$, then there exists a subterm $s \in sub[K]$ of t' such that $s \in T^W_\Sigma$, $t' \approx_{W(\psi)} s$, and $\tau^*(s) = \tau^*(t')$, and (1) holds. On the other hand, if $\tau^+(t') \notin T^W_\Sigma$, then $\tau^*(t') = \tau^+(t')$ (since we have $head(t'{\downarrow}) = head(t)$ whenever $t \approx_P t'$ and $t \in T^W_\Sigma$). By Lemma 1, if $t \approx_{W(\psi)} t'$, we have $\tau^*(t) = \tau^*(t')$. It follows that $\tau^*(t) \notin T^W_\Sigma$, and thus $\tau^+(t) = \tau^*(t) = \tau^*(t') = \tau^+(t')$. Therefore, (2) holds.

We conclude that if neither (1) nor (2) hold for t, then $t \not\approx_{W(\psi)} t'$, and if neither condition hold for any $t \in sub[K]$, then there is no $t \in dom(\psi)$ such that $t \approx_{W(\psi)} t'$. In this case, $\psi_{ROM}(t')$ is sampled with uniform probability distribution from $[\![ran(ps)]\!]$, according to the algorithm description, and it follows that $P[K, \psi', t'] = 1/\left|[\![ran(ps)]\!]\right|$ if $b \in [\![ran(ps)]\!]$ and $P[K, \psi', t'] = 0$ otherwise, concluding the proof of the lemma. □

In the following, if ι is any selection function, we adopt the convention that $\iota(\perp) = \perp$, and $ran(\perp) = \perp$, $[\![\perp]\!] = \{\perp\}$. Thus, $[\![ran(\iota(\perp))]\!] = \{\perp\}$.

Corollary 2. *Let* K *be a finite set of terms,* $\prec$ *be a subterm-compatible order, and* $\psi : sub[K] \to \mathcal{B}^*_\perp$ *be a function satisfying* $\approx_{W(\psi)}$ *and some* $\iota \in I_{\mathcal{S}}(K)$*. Consider the function* τ^+ *output by Algorithm 7 for the input* $(K, W(\psi), \prec, \tau)$ *for some* $W(\psi)$*-renaming* τ*. Let* $\tau^K : \tau^+[sub[K]] \to sub[K]$ *be such that, for each* $t^+ \in \tau^+[sub[K]]$*,* $\tau^K(t^+)$ *is the least* $t \in sub[K]$ *such that* $\tau^+(t) = t^+$ *and* $\iota(t) \neq \perp$ *if such a* t *exists, and* $\perp$ *otherwise. Let* ψ_{ROM} *be the function output by Algorithm 3 on input* K *and using the subterm-compatible order* $\prec$*. Then,*

$$P[\psi_{ROM} = \psi] = \prod_{t^+ \in \tau^+[sub[K]] \setminus \mathcal{N}^+} \frac{1}{\left|[\![ran(\iota(\tau^K(t^+)))]\!]\right|}.$$

Proof. Let $t_1, \ldots, t_n$ be such that $dom(\psi) = \{t_1, \ldots, t_n\}$ and $t_1 \prec \ldots \prec t_n$. We have

$$\begin{aligned} & P[\psi_{ROM} = \psi] \\ = \; & P[\psi_{ROM}(t_1) = \psi(t_1), \ldots, \psi_{ROM}(t_n) = \psi(t_n)] \\ = \; & \textstyle\prod_{j=1}^{n} P[\widehat{t_j}^{K,\prec} = \psi(t_j) \mid \widehat{t_1}^{K,\prec} = \psi(t_1), \ldots, \widehat{t_{j-1}}^{K,\prec} = \psi(t_{j-1})] \end{aligned} \tag{2}$$

By Lemma 5, if ψ does not satisfy $\approx_R$ or there is no $\iota \in I_{\mathcal{S}}(K)$ such that ψ satisfies ι, then $P[\psi_{ROM} = \psi] = 0$ (because one of the factors in the product is 0). Therefore, it is sufficient to consider the case that ψ satisfies $\approx_R$ and there is $\iota \in I_{\mathcal{S}}(K)$ such that ψ satisfies ι. Then, for each $j \in \{1, \ldots, n\}$, either (1) there is $k < j$ such that either (a) $\tau^+(t_j) = \tau^+(t_k)$ or (b) $\tau^+(t_j) \in \mathcal{N}^+$ and $\tau^*(t_k) = \tau^+(t_j)$, or (2) there is no such k. In case (1), $\psi_{ROM}(t_j)$ is sampled as $\psi_{ROM}(t_j)$, and

$$P[\widehat{t_j}^{K} = \psi(t_j) \mid \widehat{t_1}^{K} = \psi_{ROM}(t_1), \ldots, \widehat{t_{j-1}}^{K} = \psi_{ROM}(t_{j-1})] = 1.$$

In case (2), we have

$$P[\widehat{t_j}^{K} = \psi(t_j) \mid \widehat{t_1}^{K} = \psi_{ROM}(t_1), \ldots, \widehat{t_{j-1}}^{K} = \psi_{ROM}(t_{j-1})] = \frac{1}{\left|\left[\!\left[ran(\iota(t_{i(\prec,j)}))\right]\!\right]\right|}.$$

The result then follows by combining equation (2) and Lemma 5. □

Lemma 6. *Let K be a finite set of terms, $\prec$, $\prec'$ be subterm-compatible orders, and $\psi\colon sub[K] \to \mathcal{B}^*_\perp$. Suppose that ψ satisfies $\approx_{W(\psi)}$ and there is $\iota \in I_{\mathcal{S}}(K)$ such that ψ satisfies ι. Consider the function τ^+ output by Algorithm 7 on the inputs $K, W(\psi)$. Define the functions $\tau_K, \tau'_K\colon \tau^+[sub[K]] \to T_\Sigma(\mathcal{N}^+)$ such that, for each $t^+ \in \tau^+[sub[K]]$, $\tau_K[t^+]$ (respectively, $\tau'_K[t^+]$) is the least term t with respect to $\prec$ (respectively, $\prec'$) such that $\tau^+(t) = t^+$ and $\iota(t) \neq \perp$ if such a t exists, and $\perp$ otherwise. Then, for all $t^+ \in \tau^+[sub[K]] \setminus \mathcal{N}^+$,*

$$\left[\!\left[ran(\iota(\tau^K(t^+)))\right]\!\right] = \left[\!\left[ran(\iota(\tau^{K'}(t^+)))\right]\!\right].$$

Proof. Let $t^+ \in \tau^+[sub[K]] \setminus \mathcal{N}^+$. If $\iota(t) = \perp$ for all t such that $\tau^+(t) = t^+$, the result holds since

$$\left[\!\left[ran(\iota(\tau^K(t^+)))\right]\!\right] = \left[\!\left[ran(\iota(\tau^{K'}(t^+)))\right]\!\right] = \{\perp\}.$$

Suppose then that this is not the case. By the compatibility condition, there are $t^K, t^{K'} \in sub(t^+)$ such that $t^K \approx_{P|_{psub(t^+)}} t^+$, $t^{K'} \approx_{P|_{psub(t^+)}} t^+$, and, for all t' such that $t' \approx_{P|_{psub(t^+)}} t^+$, we have

$$\left[\!\left[ran(\iota(t^K))\right]\!\right] \subseteq \left[\!\left[ran(\iota(t'))\right]\!\right] \quad \text{and} \quad \left[\!\left[ran(\iota(t^{K'}))\right]\!\right] \subseteq \left[\!\left[ran(\iota(t'))\right]\!\right].$$

Because $t^K \approx_{P|_{psub(t^+)}} \tau^K(t^+)$ and $t^+ \notin \mathcal{N}^+$, we have $\tau^+(t^K) = t^+$, and thus $t^K = \tau^K(t^+)$ (since $\prec, \prec'$ are subterm-compatible orders). Analogously, we have $t^{K'} = \tau^{K'}(t^+)$. We conclude that

$$\left[\!\left[ran(\iota(\tau^{K'}(t^+)))\right]\!\right] \subseteq \left[\!\left[ran(\iota(\tau^K(t^+)))\right]\!\right]$$

and

$$[\![ran(\iota(\tau^K(t^+)))]\!] \subseteq \left[\!\!\left[ran(\iota(\tau^{K'}(t^+)))\right]\!\!\right],$$

and the result follows. □

Proof (Theorem 4). Let $K = dom(\lambda)$, $K' = dom(\lambda')$. Let $t_1, \ldots, t_n$ be such that $sub[K \cap K'] = \{t_1, \ldots, t_n\}$. Because $\Omega_\lambda = \Omega_{\lambda'}$, we must have $\lambda(t) = \lambda'(t)$ for all $t \in K \cap K'$, $\lambda(t) = \mathcal{B}^*_\perp$ for all $t \in K \setminus K'$, and $\lambda'(t) = \mathcal{B}^*_\perp$ for all $t \in K' \setminus K$.

For simplicity, we will write $\widehat{t}^K$ (respectively $\widehat{t}^{K'}$) for the random variable $\widehat{t}^{K,\prec}$ (respectively $\widehat{t}^{K',\prec'}$). The probability that Algorithm 3 on input K and $\prec$ outputs a function ψ_{ROM} such that $\psi_{ROM}(t) \in \lambda(t)$ for all $t \in K$ depends only on the values of $\psi_{ROM}(t')$ for $t' \in sub[K \cap K']$, and the same statement is valid for executing Algorithm 3 on input K' and $\prec'$. Therefore, it is sufficient to prove that, whenever $b_1, \ldots, b_n \in \mathcal{B}^*_\perp$,

$$P[\widehat{t_1}^K = b_1, \ldots, \widehat{t_n}^K = b_n] = P[\widehat{t_1}^{K'} = b_1, \ldots, \widehat{t_n}^{K'} = b_n].$$

Let $\tau^K : \tau^+[sub[K]] \to sub[K]$ be such that, for each $t^+ \in \tau^+[sub[K]]$, $\tau^K(t^+)$ is the least $t \in sub[K]$ such that $\tau^+(t) = t^+$, and define $\tau^{K'}$ analogously.

Corollary 2 implies that

$$P[\widehat{t_1}^K = b_1, \ldots, \widehat{t_n}^K = b_n] = \prod_{t^+ \in \tau^+[sub[K]] \setminus \mathcal{N}^+} \frac{1}{|[\![ran(\iota(\tau^K(t^+)))]\!]|}$$

and

$$P[\widehat{t_1}^{K'} = b_1, \ldots, \widehat{t_n}^{K'} = b_n] = \prod_{t^+ \in \tau^+[sub[K]] \setminus \mathcal{N}^+} \frac{1}{|[\![ran(\iota(\tau^{K'}(t^+)))]\!]|}.$$

Lemma 6 then implies the result.

□

Lemma 7. *Let K be a finite set of terms and $\prec$ be a subterm-compatible order. For each $t \in sub[K]$, there exists a finite set $supp_{PS}(t)$ such that, if ψ_{ROM} is a possible output of Algorithm 3) on input K and using the order $\prec$, then $\psi_{ROM}(t) \in supp_{PS}(t)$.*

Proof. Let $t_1, \ldots, t_n$ be such that $sub[K] = \{t_1, \ldots, t_n\}$ and $t_1 \prec \ldots \prec t_n$. We prove the result for all t_i such that $i \in \{1, \ldots, n\}$ by induction on i. For $i = 0$, we have $t_i \in \Sigma_0$ and, at the point of the execution of the Algorithm in which $\psi_{ROM}(t_0)$ is sampled, we have $dom(\psi_{ROM}) = \emptyset$. Thus, t_0 is either sampled as $\perp$ or sampled from $[\![ran(ps)]\!]$, which is finite by our definition of property statement and interpretation function. Now, suppose that terms $t_1, \ldots, t_k$ have been sampled, so that $dom(\psi_{ROM}) = \{t_1, \ldots, t_k\}$, and consider the point of the execution of the algorithm in which $\psi_{ROM}(t_{k+1})$ is sampled. Let $f \in \Sigma_{n'}$ and $t'_1, \ldots, t'_{n'}$ be such that $t_{k+1} = f(t'_1, \ldots, t'_{n'})$. By the induction hypothesis,

for each $i \in \{1, \ldots, k\}$, there exists a finite set $supp_{PS}(t_i)$ such that, for any execution of the algorithm, $\psi_{ROM}(t_i) \in supp_{PS}(t_i)$. We have

$$(\psi_{ROM}(t'_1), \ldots, \psi_{ROM}(t'_{n'})) \in supp_{PS}(t'_1), \ldots, supp_{PS}(t'_{n'}));$$

therefore, there exists a finite set $PS_{k+1} \subseteq PS_f$ of property statements such that, for each possible ψ_{ROM}, there is exactly one $ps \in PS_{k+1}$ such that $(\psi_{ROM}(t'_1), \ldots, \psi_{ROM}(t'_{n'})) \in [\![dom(ps)]\!]$. Then, t_i is either sampled to $\perp$, or to $\psi_{ROM}(\psi_{ROM}(t'_i))$ for some $i \in \{1, \ldots, k\}$, or to some element of $\bigcup_{ps \in PS_{k+1}} [\![ran(ps)]\!]$. This is a union of finitely many finite sets; thus, we can choose $supp_{PS}(t_{k+1})$ to be this set. □

Lemma 8. *The set Ω_Λ is a semi-ring of sets.*

Proof. If t_1 is any term and $\lambda = \{t_1 \mapsto \emptyset\}$, then $\Omega_\lambda = \emptyset$. Thus, $\emptyset \in \Omega_\Lambda$.

Let $\lambda = \{t_i \mapsto B_i \mid i \in \{1, \ldots, n\}\}$ and $\lambda' = \{t'_i \mapsto B'_i \mid i \in \{1, \ldots, n'\}\}$. Let

$$dom(\lambda) \cup dom(\lambda') = \{t''_1, \ldots, t''_{n''}\}.$$

There are sets $C_1, \ldots, C_{n''}, C'_1, \ldots, C'_{n''}$ such that, choosing

$$\lambda_* = \{t''_i \mapsto C_i \mid i \in \{1, \ldots, n''\}\} \quad \text{and} \quad \lambda'_* = \{t''_i \mapsto C'_i \mid i \in \{1, \ldots, n''\}\},$$

we have $\Omega_{\lambda_*} = \Omega_{\lambda'_*}$. These sets can be obtained by simply choosing $C_i = \lambda(t_i)$ if $t_i \in dom(\lambda)$ and $C_i = \mathcal{B}^*_\perp$ otherwise and, analogously, $C'_i = \lambda'(t_i)$ if $t_i \in dom(\lambda')$ and $C'_i = \mathcal{B}^*_\perp$ otherwise.

We have that

$$\Omega_\lambda \cap \Omega_{\lambda'} = \Omega_{\lambda_*} \cap \Omega_{\lambda'_*} = \Omega_{\lambda''_*},$$

where $\lambda'' = \{t_i \mapsto C_i \cap C'_i \mid i \in \{1, \ldots, n''\}\}$. Thus, Ω_Λ is closed for intersections.

For each $i \in \{1, \ldots, n''\}$, let $C^0_i = C_i \cap C'_i$ and $C^1_i = C_i \setminus C'_i$, and consider the set $\Lambda(\lambda, \lambda')$ of functions $\lambda'' : \{t''_1, \ldots, t''_{n''}\} \to \mathcal{B}^*_\perp$ such that, for each $i \in \{1, \ldots, n''\}$, $\lambda''(t''_i)$ is either C^0_i or C^1_i. Let λ''_0 be the element of $\Lambda(\lambda, \lambda')$ such that, for each $i \in \{1, \ldots, n''\}$, $\lambda''_0(t''_i) = C^0_i$.

We have $\Omega_{\lambda''} \in \Omega_\Lambda$ for all $\lambda'' \in \Lambda(\lambda, \lambda')$, $\Omega_\lambda = \biguplus_{\lambda'' \in \Lambda(\lambda, \lambda')} \Omega_{\lambda''}$, and $\Omega_\lambda \cap \Omega_{\lambda'} = \Omega_{\lambda''_0}$.

We conclude that

$$\Omega_\lambda \setminus \Omega_{\lambda'} = \biguplus_{\lambda'' \in \Lambda(\lambda, \lambda') \setminus \{\lambda''_0\}} \Omega_{\lambda''}$$

is a finite, disjoint union of elements in Ω_Λ. Thus, Ω_Λ is a semi-ring of sets. □

Lemma 9. *Suppose that $\mu(\emptyset) = 0$, $\mu(\mathbf{\Omega}) = 1$ and, whenenver $\lambda_1, \ldots, \lambda_n \in \Lambda$ are such that $\Omega_{\lambda_1}, \ldots, \Omega_{\lambda_n}$ are disjoint sets such that $\biguplus_{i=1}^n \Omega_{\lambda_i} = \Omega_\lambda$ for some $\lambda \in \Lambda$, we have*

$$\mu(\Omega_\lambda) = \sum_{i=1}^{n} \mu(\Omega_{\lambda_i}).$$

Then, there is a unique extension of μ to $\mathcal{F}$ that is a probability measure.

Proof. By Lemma 8 and Caratheodory's extension theorem, it is sufficient to show that, if $\lambda_i \in \Lambda$ for each $i \in \mathbb{N}$ are such that $\Omega_{\lambda_i} \cap \Omega_{\lambda_j} = \emptyset$ whenever $i \neq j$ and there is $\lambda \in \Lambda$ such that $\biguplus_{i \in \mathbb{N}} \Omega_{\lambda_i} = \Omega_\lambda$, then there are finitely many indexes $i_1, \ldots, i_n \in \mathbb{N}$ such that

$$\Omega_\lambda = \biguplus_{j=1}^{n} \Omega_{\lambda_{i_j}}.$$

For each $i \in \mathbb{N}$, let us consider the number k_i and, for each $j \in \{1, \ldots, k_i\}$, the terms t_i^j and the sets of bitstrings B_i^j such that

$$\lambda_i = \left\{ t_i^j \mapsto B_i^j \mid j \in \{1, \ldots, k_i\} \right\}.$$

For each $i \in \mathbb{N}$ and each $j \in \{1, \ldots, k_i\}$, let $\lambda_i^{PS} = \left\{ t_i^j \mapsto B_i^j \cap \mathit{supp}_{PS}(t_i^j) \right\}$, where supp_{PS} is as in Lemma 7. Analogously, let $k \in \mathbb{N}$, $t^1, \ldots, t^k \in T_\Sigma$, and $B^1, \ldots, B^k$ be such that $\lambda = \left\{ t^j \mapsto B^j \mid j \in \{1, \ldots, k\} \right\}$, and define $\lambda^{PS} = \left\{ t^j \mapsto B^j \cap \mathit{supp}_{PS}(t^j) \right\}$.

Lemma 7 implies that $B_i^j \cap \mathit{supp}_{PS}(t_i^j)$ is finite for all i and all j, and (together with the definition of μ),

$$\mu(\Omega_{\lambda_i}) = \mu(\Omega_{\lambda_i^{PS}})$$

and

$$\mu(\Omega_\lambda) = \mu(\Omega_{\lambda^{PS}}).$$

Thus, we may assume without loss of generality that all the sets B_i^j are such that

$$B_i^j \subseteq \mathit{supp}_{PS}(t_i^j).$$

For each $t \in T_\Sigma$, consider the topological space $\mathit{supp}_{PS}(t)$ where all subsets are open. This space is finite and, therefore, it is trivially compact. Now, consider the topological space

$$F = \left\{ \omega : T_\Sigma \to \mathcal{B}_\perp^* \mid \omega(t) \in \mathit{supp}_{PS}(t) \text{ for all } t \right\}.$$

F is the cartesian product of the topological spaces associated to each term t. The open sets in this topological space with the product topology are precisely the sets Ω_λ for functions $\lambda \in \Omega_\Lambda$ such that, for each $t \in \mathit{dom}(\lambda)$, $\lambda(t) \subseteq \mathit{supp}_{PS}(t)$. By Tychonoff's theorem, F with the product topology is also a compact space.

Because the open sets of F form a semi-ring (by an argument entirely analogous to the one we used in the proof of Lemma 8), we know that $F \setminus r$ is a finite union of open sets, and thus is also open. We conclude that r is closed. Because F is compact, it follows that r is also compact. Since $\{\Omega_{\lambda_i} \mid i \in \mathbb{N}\}$ is a open cover of Ω_λ, there must be a finite sub-cover — that is, there must be indexes $i_1, \ldots, i_m$ such that

$$\Omega_\lambda = \biguplus_{k=1}^{m} \Omega_{\lambda_{i_k}}.$$

And because the Ω_{λ_i} are disjoint, we conclude that $\Omega_{\lambda_i} = \emptyset$ for all $i \notin \{i_1, \ldots, i_m\}$. The result then follows from

$$\mu(\Omega_\lambda) = \mu(\biguplus_{i\in\mathbb{N}} \Omega_{\lambda_i}) = \mu(\biguplus_{k=1}^{m} \Omega_{\lambda_{i_k}}) = \sum_{k=1}^{m} \mu(\Omega_{\lambda_{j_k}}) = \sum_{i\in\mathbb{N}} \mu(\Omega_{\lambda_j}).$$

□

Proof (Theorem 5). It is trivial to check that $\mu(\emptyset) = 0$ and $\mu(\mathbf{\Omega}) = 1$. By Lemma 9, it is sufficient to prove that, whenever $\Omega_1, \ldots, \Omega_n \in \Omega_\Lambda$ are pairwise disjoint and there is $\lambda \in \Lambda$ such that $\Omega_\lambda = \biguplus_{i=1}^n \Omega_i$, then $\mu(\Omega_\lambda) = \sum_{i=1}^n \mu(\Omega_{\lambda_i})$.

For each $i \in \{1, \ldots, n\}$, we have $\Omega_{\lambda_i} = \biguplus_{\psi\in\Psi(\lambda_i)} \Omega_\psi$. Whenever $i \neq j$, we have $\Omega_{\lambda_i} \cap \Omega_{\lambda_j} = \emptyset$, and thus also $\Omega_{\psi_i} \neq \Omega_{\psi_j}$ whenever $\psi_i \in \Psi(\lambda_i)$ and $\psi_j \in \Psi(\lambda_j)$. It follows that

$$\biguplus_{i=1}^{n} \biguplus_{\psi\in\Psi(\lambda_i)} \Omega_\psi = \biguplus_{\psi\in\Psi(\lambda)} \Omega_\psi.$$

Now, if λ' is such that $\Omega_{\lambda'} = \biguplus_{\psi\in\Psi(\lambda')} \Omega_\psi$, it is clear that the function ψ_{ROM} output by Algorithm 3 on input $dom(\lambda')$ is such that $\psi_{ROM}(t) \in \lambda'(t)$ for all $t \in K$ if and only if $\psi_{ROM} \in \Psi(\lambda')$. Therefore, it follows that, for all λ,

$$\mu(\Omega_\lambda) = \sum_{\psi\in\Psi(\lambda)} \mu(\Omega_\psi).$$

We obtain

$$\mu(\Omega_\lambda) = \mu(\biguplus_{\psi\in\Psi(\lambda)} \Omega_\psi) = \sum_{\psi\in\Psi(\lambda)} \mu(\Omega_\psi)$$

$$= \sum_{i=1}^{n} \left(\sum_{\psi\in\Psi(\lambda_i)} \mu(\Omega_\psi) \right) = \sum_{i=1}^{n} \mu(\Omega_{\lambda_i}),$$

which proves that μ is a probability measure. It remains to prove $\mu(\Omega_{PS,[\![\cdot]\!],\approx_R}) = 1$. Let $f \in \Sigma$ and $t_{n+1} = f(t_1, \ldots, t_n)$. By Lemma 4, any execution of Algorithm 3 on input $\{t_{n+1}\}$ yields a function ψ_{ROM} which satisfies some $\iota \in I_{\mathcal{S}}(sub(t_{n+1}))$. Thus, if $(\psi_{ROM}(t_1), \ldots, \psi_{ROM}(t_n)) \notin dom_{\mathcal{S}}(f)$, then $\psi_{ROM}(t_{n+1}) = \bot$; otherwise, there is some $ps \in PS_f$ such that $(\psi_{ROM}(t_1), \ldots, \psi_{ROM}(t_n)) \in [\![dom(ps)]\!]$, in which case we must have $\psi_{ROM}(t_{n+1}) \in [\![ran(ps)]\!]$.

If $f \in \Sigma_n$, $t_1, \ldots, t_n$ are terms, and $ps \in PS_f$, let us write $\Omega_{f,t_1,\ldots,t_n,ps}$ for the set of $\omega \in \mathbf{\Omega}$ such that $(\omega(t_1), \ldots, \omega(t_n)) \in [\![dom(ps)]\!]$ and $\omega(t_{n+1}) \notin [\![ran(ps)]\!]$. Analogously, let us write $\Omega_{f,t_1,\ldots,t_n,\bot}$ for the set of $\omega \in \mathbf{\Omega}$ such that $(\omega(t_1), \ldots, \omega(t_n)) \notin dom_{\mathcal{S}}(f)$ and $\omega(t_{n+1}) \neq \bot$. If $\omega \not\models PS$, we must have $\omega \in \Omega_{f,t_1,\ldots,t_n,ps} \cup \Omega_{f,t_1,\ldots,t_n,\bot}$ for some $f \in \Sigma_n$, some terms $t_1, \ldots, t_n$ and some $ps \in PS_f$, and we have seen above that

$$\mu(\Omega_{f,t_1,\ldots,t_n,ps}) = \mu(\Omega_{f,t_1,\ldots,t_n,\bot}) = 0$$

for all such $f, t_1, \ldots, t_n, ps$. Since there are only countably many possible choices for $f, t_1, \ldots, t_n, ps$, it follows that

$$\mu\left(\{\omega \mid \omega \models_{[\![\cdot]\!]} PS\}\right) = 1. \tag{3}$$

On the other hand, suppose that $t_1, t_2 \in T_\Sigma$ are terms such that $t_1 \approx_R t_2$, and assume without loss of generality that $t_1 \prec t_2$. Any execution of Algorithm 3 on input $\{t_1, t_2\}$ will sample $\psi_{ROM}(t_2)$ as $\psi_{ROM}(t_1)$ on line 9. Letting $\psi(b_1, b_2) = \{t_1 \mapsto b_1, t_2 \mapsto b_2\}$ whenever $b_1, b_2 \in \mathcal{B}^*_\perp$, we have that

$$\{\omega \in \mathbf{\Omega} \mid \omega(t_1) \neq \omega(t_2)\} = \bigcup_{b_1 \in \mathcal{B}^*_\perp} \bigcup_{b_2 \in \mathcal{B}^*_\perp \setminus \{b_1\}} \Omega_{\psi(b_1,b_2)}$$

is a set in $\mathcal{F}$, and

$$\mu(\omega \in \mathbf{\Omega} \mid \omega(t_1) \neq \omega(t_2)) = 0.$$

Since there are only countably many choices for t_1 and t_2, it follows that

$$\mu\left(\{\omega \in \mathbf{\Omega} \mid \omega \models \approx_R\}\right) = 1. \tag{4}$$

Combining equations (3) and (4), we conclude that $\mu(\Omega_{\mathcal{S}}) = 1$, as desired. □

Let K be a set of terms and $\psi \in \Psi_K$. As in Algorithm 3, we will denote by $W(\psi)$ the partition on $sub[dom(\psi)] \cap T^W$ induced by ψ: thus, $W(\psi) = P(\psi) \mid_{T^W}$, where $P(\psi) = \left\{\psi^{-1}(b) \mid b \in ran(\psi)\right\}$. We say that ψ is a *colliding instatiation* of K if there exist terms $t_1, t_2 \in sub[K]$ such that t_1 is strong (i.e., $t_1 \notin T^W$), $t_1 \not\approx_{W(\psi)} t_2$, $(\iota(\psi))(t_1) \neq \perp$, and $\psi(t_1) = \psi(t_2)$. We write Ψ^{col}_K for the set of $\psi \in \Psi_K$ that are colliding instantiations of K, and

$$\Omega^{col}(K) = \bigcup_{\psi \in \Psi^{col}_K} \Omega_\psi.$$

Theorem 8. *If K is a finite set of terms and $\prec$ is a subterm-compatible order, then*

(1) for any $\lambda \in \Lambda_K$, $\mu^{K,\prec}_\eta(\Omega_\lambda \cap \overline{\Omega^{col}}) = \mu_{ROM,\eta}(\Omega_\lambda \cap \overline{\Omega^{col}})$;
(2) there exists a constant $c(K)$ such that

$$\mu^{K,\prec}_\eta(\mathbf{\Omega} \setminus \Omega(K)) = \mu_{ROM,\eta}(\mathbf{\Omega} \setminus \Omega(K)) \leq c(K) \cdot |I_{\mathcal{S}}(K)| \cdot (1/L).$$

Proof. Let $t_1, \ldots, t_n$ be such that $sub[K] = \{t_1, \ldots, t_n\}$ and $t_1 \prec \ldots \prec t_n$. Let $\widehat{t}^{K,\prec}$ be the random variable representing the output of Algorithm 1 when executed on input K and using the order $\prec$, and $\widehat{t}$ be the random variable representing the output of Algorithm 3 on input K. It is sufficient to prove the property for the sets Ω_ψ for all $\psi \in \Psi_K$. For all $\psi \in \Psi_K$, we have

$$\mu^{K,\prec}_{ROM,\eta}(\Omega_\psi) = \textstyle\prod_{i=1}^n P[\widehat{t_i}^{K,\prec} = \psi(t_i) \mid j < i \Rightarrow \widehat{t_j}^{K,\prec} = \psi(t_j)] \tag{5}$$

and

$$\mu_{ROM,\eta}(\Omega_\psi) = \prod_{i=1}^{n} P[\widehat{t_i} = \psi(t_i) \mid j < i \Rightarrow \widehat{t_j} = \psi(t_j)]. \tag{6}$$

If ψ is a colliding instantiation of K, then $\Omega_\psi \cap \overline{\Omega^{col}}(K) = \emptyset$, and the result is true since

$$\mu_\eta^{K,\preceq}(\emptyset) = \mu_{ROM,\eta}(\emptyset) = 0.$$

Suppose then that ψ is not a colliding instantiation of K. Let $i \in \{1, \ldots, n\}$, and suppose that $\psi_{ROM}(t_j)$ has been sampled to b_j for all $j < i$. Because ψ is not a colliding instantiation of K, we have $W(\psi \mid \{t_1, \ldots, t_{i-1}\}) = P(\psi \mid \{t_1, \ldots, t_{i-1}\})$. Therefore, the steps executed by the two algorithms when sampling $\psi_{ROM}(t_i)$ are exactly the same. It follows that

$$P[\widehat{t_i}^{K,\prec} = \psi(t_i) \mid j < i \Rightarrow \widehat{t_j}^{K,\prec} = \psi(t_j)] = P[\widehat{t_i} = \psi(t_i) \mid j < i \Rightarrow \widehat{t_j} = \psi(t_j)]$$

for all $i \in \{1, \ldots, n\}$. Thus, equations (5) and (6) imply (1).

To prove (2), for each $\iota \in I_{\mathcal{S}}(K)$, each $j, k \in \{1, \ldots, n\}$ such that $j \neq k$ and $t_k \in T_\Sigma \setminus T_\Sigma^W$, and each $b \in ran(\iota(t_j))$, let $\Psi_{j,k,b}^\iota$ be the set of ψ such that ψ satisfies ι, $t_k \in T_\Sigma \setminus T_\Sigma^W$, $ran(\iota(t_k)) \neq \bot$, $ran(\iota(t_j)) \neq \bot$, and $\psi(t_j) = \psi(t_k)$. We note that, for all $\psi \in \Psi_K^{col}$, there are ι, j, k, b such that $\psi \in \Psi_{j,k,b}^\iota$, and $ran(\iota(t_k)) \geq L$.

Letting $\Psi_{j,k}$ be the set of $\psi \in \Psi_K^{col}$ such that $\psi(t_j) = \psi(t_k)$, we have

$$P[\Psi_{j,k}] \leq \sum_{\iota \in I_{\mathcal{S}}(K)} \sum_{b \in ran(\iota(t_j))} \sum_{\psi \in \Psi_{j,k,b}^\iota} \left(\prod_{i=1}^{n} \frac{1}{|ran(\iota(t_i))|} \right)$$

Now, we note that, for each $\iota \in I_{\mathcal{S}}(K)$, we have

$$\left|\Psi_{j,k,b}^\iota\right| \leq \prod_{i=1, i \neq j,k} |ran(\iota(t_i))|.$$

Therefore, the previous equation implies

$$P[\Psi_{j,k} \leq \sum_{\iota \in I_{\mathcal{S}}(K)} \left(|ran(\iota(t_j))| \cdot \left(\prod_{\substack{i=1 \\ i \neq j,k}}^{n} |ran(\iota(t_i))| \right) \cdot \left(\prod_{i=1}^{n} \frac{1}{|ran(\iota(t_i))|} \right) \right) \leq \frac{1}{L}.$$

It follows that

$$P[\Psi_K^{col}] \leq \sum_{\iota \in I_{\mathcal{S}}(K)} \left(\sum_{j=1}^{n} \left(\sum_{\substack{k=1 \\ k \neq j}}^{n} P[\Psi_{j,k}] \right) \right) \leq |K|^2 \sum_{\iota \in I_{\mathcal{S}}(K)} \frac{1}{L},$$

concluding the proof.

□

Proof (Theorem 6). Follows from Theorem 8, by taking $\Omega(K) = \overline{\Omega^{col}}(K)$.

□

Appendix B: Computing Our Probability Distribution

5.1 Definitions and results

We define additional types error and empty, and extend $[\![\cdot]\!]$ such that $[\![\mathsf{error}]\!] = \{\bot\}$ and $[\![\mathsf{empty}]\!] = \emptyset$. Furthermore, when $X \subseteq \mathcal{B}^*$, we may use a type $T_X \notin \mathcal{T}$, and further extend $[\![\cdot]\!]$ such that $[\![T_X]\!] = X$. Note that we still refer to $[\![\cdot]\!]$ as an interpretation function. Let K be a finite set of terms and $\iota \in I(K)$ be a selection function for K. We define the *supremum ι-support* function $\overline{supp_\iota}\colon K \to \mathcal{T} \cup \{\mathsf{error}\}$ by error if $\iota(t) = \bot$ and $\overline{supp_\iota}(t) = ran(\iota(t))$ otherwise.

$\approx_P^+$ and $\approx_P^*$-equivalence classes. If $t \in sub[K]$, we define

$$[t]_P^* = \left\{(\tau^*)^{-1}(t) \mid t \in \tau^*[sub[K]] \setminus \mathcal{N}^+\right\}$$

to be the τ^*-equivalence class of t. Analogously,

$$[t]_P^+ = \left\{(\tau^+)^{-1}(t) \mid t \in \tau^+[sub[K]]\right\}$$

is the τ^*-equivalence class of t. Similarly, if K is a set of terms, we denote by $[K]_P^*$ (respectively, $[K]_P^+$) the set of τ^*-equivalence classes (respectively, τ^+-equivalence classes) of terms in K. We define $\approx_P^+$ (respectively, $\approx_P^*$) as the equivalence relation such that $t \approx_P^+ t'$ (respectively, $t \approx_P^+$) if $\tau^+(t) = \tau^+(t')$ (respectively, $t \approx_P^* t'$). Since it is clear that $\approx_P \subseteq \approx_P^*$, for each $\approx_P^*$-equivalence class C, there is a $\approx_P^+$-equivalence class C' such that, for all $t \in C$, $[t]_P^* = C'$; for each $C \in [K]_P^+$, we denote by $[C]_P^*$ this unique class C'.

Let $\prec$ be some subterm-compatible order. For each $\lambda \in \Lambda_K$ and each $C \in [sub[K]]_P^+$, we define $\overline{supp_\iota}(C)$ to be $ran(\iota(t))$, with t being least t with respect to $\prec$ such that $t \in C$ and $\iota(t) \neq \bot$. In light of Lemma 6, $\overline{supp_\iota}$ is well-defined and, for each $C \in [sub[K]]_P^+$ and each $t \in C$, either $\overline{supp_\iota}(t) = \bot$ or $\overline{supp_\iota}(C) \subseteq \overline{supp_\iota}(t)$. Note also that, if there is $a \in \Sigma_0$ such that $a \in C$, then $\overline{supp_\iota}(C) \neq \mathsf{error}$.

Infimum support. Given a finite set of terms K, $\lambda \in \Lambda_K$ and $\iota \in I(K)$, we define $\mathcal{P}_{R,\bot}^W(\iota)$ as the set of partitions P of $sub[K] \cap T^W$ for which there is $p_\bot \in P$ such that, for all $t \in sub[K] \cap T^W$, $t \in p_\bot$ if and only if $\iota(t) = \bot$. If $P \in \mathcal{P}_{R,\bot}^W(\iota)$, we define the infimum (ψ, ι, P)-support function for each $t \in sub[K]$ as follows: For each $t \in sub[K]$, $supp_{\lambda,\iota,P}(t)$ is the smallest set such that:

- $\overline{supp_\iota}(t) \subseteq supp_{\lambda,\iota,P}(t)$;
- if $t \in K$, then $T_{\lambda(t)} \in supp_{\lambda,\iota,P}(t)$;
- if $t \approx_P t'$, $T \in supp_{\lambda,\iota,P}(t')$, and $\mathsf{error} \notin \{\overline{supp_\iota}(t), \overline{supp_\iota}(t')\}$, then $T \in supp_{\lambda,\iota,P}(t)$.

For each class $C \in [K]_P^*$ there is a (finite) set $\mathbf{T}$ such that, for each $t \in C$, we have $supp_{\lambda,\iota,P}(t) = \mathbf{T}$ or $\overline{supp_\iota}(t) = \mathsf{error}$. We define $supp_{\lambda,\iota,P}(C)$ for each class $C \in [K]_P^*$ as (1) empty if there is $t \in C$ such that $\overline{supp_\iota}(t) = \mathsf{error}$ and there is $T \in supp_{\lambda,\iota,P}(t)$ such that $\bot \notin [\![T]\!]$, and (2) $supp_{\lambda,\iota,P}(C) = \mathbf{T}$ (where

T is as above) otherwise. Note that $supp_{\lambda,\iota,P}(C) = \mathsf{empty}$ if and only if there is $t \in C$ such that $\overline{supp_\iota}(t) = \mathsf{error}$ and there is $T \in supp_{\lambda,\iota,P}(t)$ such that either $T \in \mathcal{T}$ or $T = T_B$ for some $B \subseteq \mathcal{B}^*$ which does not contain $\bot$. Furthermore, $supp_{\lambda,\iota,P}(C) = \mathsf{error}$ if and only if $supp_{\lambda,\iota,P}(t) = \mathsf{error}$ for all $t \in C \cap K$.

Function sets. Given a finite set of terms K, $\lambda \in \Lambda_K$, $\iota \in I(K)$, and $P \in \mathcal{P}^W_{R,\bot}(K)$, let us denote by $WC(K)$ the set $[sub[K] \cap T^W]^+_P$. We define the following sets:

- $\Psi^D(\lambda, \iota, P)$ is the set of functions $\psi \colon [sub[K]]^+_P \to \mathcal{B}^*_\bot$ such that, for all $C \in [sub[K]]^+_P$, $\psi(C) \in [\![\overline{supp_\iota}(C)]\!]$;
- $\Psi^U(\lambda, \iota, P)$ is the set of functions $\psi \colon [sub[K]]^*_P \to \mathcal{B}^*_\bot$ such that, for all $C \in [sub[K]]^*_P$, $\psi(C) \in supp_{\lambda,\iota,P}(C)$ and, if $C, C' \in WC(sub[K])$ are distinct, then $\psi(C) \neq \psi(C')$.

If $\psi \in \Psi_K$, we will also write $\Psi^D(\psi, \iota, P)$ and $\Psi^U(\psi, \iota, P)$ to denote the sets $\Psi^D(\lambda(\psi), \iota, P)$ and $\Psi^U(\lambda(\psi), \iota, P)$, where $\lambda(\psi) \in \Lambda_K$ is defined by $\lambda(\psi)(t) = \{\psi(t)\}$ for all $t \in K$.

Definition 3. *We define the function $\mu_C \colon \Lambda \to [0,1]$ for each $\lambda \in \Lambda$ by*

$$\mu_C(\lambda) = \sum_{\iota \in I(K)} \left(\sum_{P \in \mathcal{P}^W_{R,\bot}(K)} \frac{|\Psi^U(\lambda, \iota, P)|}{|\Psi^D(\lambda, \iota, P)|} \right),$$

where $K = dom(\lambda)$.

Theorem 9. *If K is a finite set of terms and $\lambda, \lambda' \in \Lambda_K$ are such that $\Omega_\lambda = \Omega_{\lambda'}$, then $\mu_C(\lambda) \in [0,1]$ and $\mu_C(\lambda) = \mu_C(\lambda')$.*

In light of Lemma 9, we use the symbol μ_C for the function $\mu_C \colon \Omega_\Lambda \to [0,1]$ defined, for each $\lambda \in \Lambda$, by $\mu_C(\Omega_\lambda) = \mu_C(\lambda)$.

Theorem 10. *There exists a unique extension of μ_C to $\mathcal{F}$ that is a probability measure. Abusing notation and using the symbol μ_C to refer to this extension, we have $\mu_C = \mu_{ROM}$.*

5.2 Proofs

Lemma 10. *For all valid setups $\mathcal{S}$ and all finite sets of terms K, $I_\mathcal{S}(K)$ is finite.*

Proof. Let $t_1, \ldots, t_n$ be such that $sub[K] = \{t_1, \ldots, t_n\}$ and $t_1 \prec \ldots \prec t_n$. For $i \in \{0, \ldots, n\}$, let $K_i = \{t_1, \ldots, t_i\}$. We prove the result for each K_i by induction on i. We have $I_\mathcal{S}(K_0) = \emptyset$, and thus the result holds.

Now let $i \in \{1, \ldots, n\}$, and suppose that the result holds for all $j \in \{1, \ldots, i-1\}$. For each $t \in K_{i-1}$, let

$$supp_{i-1}(t) = \bigcup_{\iota \in I_\mathcal{S}(K_{i-1})} [\![ran(\iota(t))]\!] .$$

Suppose that $\iota_i \in I_{\mathcal{S}}(K_i)$. For all $\omega \in \mathbf{\Omega}$ such that $\omega \models \iota_i$, we also have $\omega \models \iota_i \mid_{K_{i-1}}$. Thus, $\iota_i \mid_{K_{i-1}} \in I_{\mathcal{S}}(K_i)$ and, letting $t_i = f_i(t'_1, \ldots, t'_k)$, we have $\omega(t'_j) \in supp_{i-1}(t'_j)$ for all $j \in \{1, \ldots, k\}$. The induction hypothesis implies that $supp_{i-1}(t'_j)$ is finite for all j. Therefore, the set $supp_{i-1}(t'_1) \times \ldots \times supp_{i-1}(t'_k)$ is finite and, if $\omega \models \iota_i$, then

$$(\omega(t'_1), \ldots, \omega(t'_k)) \in supp_{i-1}(t'_1) \times \ldots \times supp_{i-1}(t'_k).$$

Since property statements are assumed to be disjoint, it follows that there exists a finite set $P \subseteq PS_{f_i}$ such that, whenever $\omega \models \iota_i$,

$$(\omega(t'_1), \ldots, \omega(t'_k)) \in [\![dom(ps)]\!]$$

for some $ps \in P$. It follows that

$$I_{\mathcal{S}}(K_i) \subseteq I_{\mathcal{S}}(K_{i-1}) \times (P \cup \{\bot\}),$$

and the induction hypothesis implies that $I_{\mathcal{S}}(K_i)$ is finite. □

Corollary 3. *There exists a function $supp_{PS}\colon T_\Sigma \to \mathcal{P}(\mathcal{B}^*_\bot)$ such that $supp_{PS}(t)$ is finite for all $t \in T_\Sigma$ and, whenever K is a finite set of terms and $\lambda \in \Lambda_K$, we have $\mu_C(\Omega_\lambda) = \mu_C(\Omega_{\lambda_{PS}})$, where $\lambda_{PS} \in \Lambda_K$ is given by $\lambda_{PS}(t) = \lambda(t) \cap supp_{PS}(t)$ for all $t \in K$.*

Proof. For each term t, let

$$supp_{PS}(t) = \bigcup_{\iota \in I(sub(t))} [\![ran(\iota(t))]\!].$$

Lemma 10 implies that there are finitely many $\iota \in I_{\mathcal{S}}(K)$; therefore, $supp_{PS}(t)$ is finite.

It is simple to check that, if $\iota \in I_{\mathcal{S}}(K)$, then $\iota \mid_{sub(t)} \in I_{\mathcal{S}}(sub(t))$. Thus, if $\iota \in I_{\mathcal{S}}(K)$, then $[\![ran(\iota(t))]\!] \subseteq supp_{PS}(t)$. For all $\iota \in I_{\mathcal{S}}(K)$ and all $P \in \mathcal{P}^W_R(K)$, we have

$$[\![supp_{\lambda,\iota,P}(t)]\!] \subseteq [\![\overline{supp_\iota}(t)]\!] = [\![ran(\iota)]\!](t) \subseteq supp_{PS}(t),$$

and thus

$$\begin{aligned} \left[\!\!\left[supp_{\lambda_{PS},\iota,P}(t)\right]\!\!\right] &= [\![supp_{\lambda,\iota,P}(t)]\!] \cap supp_{PS}(t) \\ &= [\![supp_{\lambda,\iota,P}(t)]\!]. \end{aligned}$$

It follows that $\Psi^U(\lambda, \iota, P) = \Psi^U(\lambda_{PS}, \iota, P)$ for all $\iota \in I_{\mathcal{S}}(K)$ and all $P \in \mathcal{P}^W_R(K)$. Thus, we have $\mu_C(\Omega_\lambda) = \mu_C(\Omega_{\lambda_{PS}})$, as desired. □

Let $\lambda \in \Lambda$ and $K = dom(\lambda)$. For each $t \in K$, let

$$supp_K(t) = \bigcup_{\iota \in I_{\mathcal{S}}(K)} [\![ran(\iota(t))]\!].$$

Recall the notion of $\Psi(\lambda)$; We now introduce the related notion $\Psi_{\mathcal{S}}(\lambda)$, defined for each $\lambda \in \Lambda$ such that $dom(\lambda) = K$ as the set of all $\psi\colon K \to \mathcal{B}^*_\bot$ such that, for all $t \in K$, $\psi(t) \in \lambda(t) \cap supp_K(t)$. Note that $supp_K(t)$ is finite (by Lemma 10), and thus so is $\Psi_{\mathcal{S}}(\lambda)$.

Lemma 11. *Let $\lambda \in \Lambda_K$, $\psi \in \Psi_{\mathcal{S}}(\lambda)$, $\iota \in I_{\mathcal{S}}(K)$ and $P \in \mathcal{P}_R^W(K)$. If $\iota = \iota(\psi) \neq \bot$, $P = W(\psi)$, and there exists $\psi^u \in \Psi^U(\lambda, \iota(\psi), P(\psi))$ such that, for each $t \in sub[K]\backslash\mathcal{N}^*$, $\psi(t) = \bot$ if $\overline{supp_\iota}(t) = \mathsf{error}$ and $\psi(t) = \psi^u([t]_P^*)$ otherwise, then*

$$\left|\Psi^U(\psi, \iota, P)\right| = 1.$$

Otherwise,

$$\left|\Psi^U(\psi, \iota, P)\right| = 0.$$

Proof. We prove the following five propositions:

(1) if $\iota(\psi) = \bot$ or $\iota \neq \iota(\psi)$, then $\left|\Psi^U(\psi, \iota, P)\right| = 0$;
(2) if $P \neq W(\psi)$, then $\left|\Psi^U(\psi, \iota, P)\right| = 0$;
(3) if there is $\psi^u \in \Psi^U(\psi, \iota, P)$, then $\psi^u \in \Psi^U(\lambda, \iota, P)$ and, for each $t \in sub[K]$, $\psi(t) = \bot$ if $\overline{supp_\iota}(t) = \mathsf{error}$ and $\psi(t) = \psi^u([t]_P^*)$ otherwise;
(4) there is at most one $\psi^u \in \Psi^U(\psi, \iota, P)$;
(5) if there is $\psi^u \in \Psi^U(\lambda, \iota, P)$ such that, for each $t \in sub[K]$, $\psi(t) = \bot$ if $\overline{supp_\iota}(t) = \mathsf{error}$ and $\psi(t) = \psi^u([t]_P^*)$ otherwise, then $\psi^u \in \Psi^U(\psi, \iota, P)$.

Properties (4) and (5) show that, under the conditions of the theorem, $\left|\Psi^U(\psi, \iota, P)\right| = 1$. Properties (1), (2) and (3) show that $\left|\Psi^U(\psi, \iota, P)\right| = 0$ otherwise.

(1) If $\iota \neq \iota(\psi)$ or $\iota(\psi) = \bot$, then either (1) there is $t \in sub[K]$ such that $\psi(t) \notin [\![ran(\iota(t))]\!]$ or (2) there is $t = f(t_1, \ldots, t_n) \in sub[K]$ such that, letting $\iota(t) = (f[T_1, \ldots, T_n] \subseteq T)$, we have $\psi(t_i) \notin [\![T_i]\!]$ for some $i \in \{1, \ldots, n\}$. In case (1), we have $\{\psi(t)\} \subseteq [\![supp_{\psi,\iota,P}([t]_P^*)]\!]$, and $[\![ran(\iota(t))]\!] \subseteq [\![supp_{\psi,\iota,P}([t]_P^*)]\!]$. We conclude that $\left|[\![supp_{\psi,\iota,P}(t)]\!]\right| = 0$, and thus $\left|\Psi^U(\psi, \iota, P)\right| = 0$. Analogously, in case (2), we have $\{\psi(t_i)\} \subseteq [\![supp_{\psi,\iota,P}([t_i]_P^*)]\!]$, and $[\![T_i]\!] \subseteq [\![supp_{\psi,\iota,P}([t_i]_P^*)]\!]$, and we conclude that $\left|[\![supp_{\psi,\iota,P}(t_i)]\!]\right| = 0$, so that $\left|\Psi^U(\psi, \iota, P)\right| = 0$.

(2) If $P \neq W(\psi)$, we have $P\ |_{\psi^{-1}(\mathcal{B}^*)} \neq P(\psi)\ |_{\psi^{-1}(\mathcal{B}^*)}$, and there are $t, t' \in sub[K] \cap T^W \cap \psi^{-1}(\mathcal{B}^*)$ such that either (1) $t \approx_P t'$ and $\psi(t) \neq \psi(t')$ or (2) $t \not\approx_P t'$ and $\psi(t) = \psi(t')$. Obviously, it is sufficient to prove that $\Psi^U(\psi, \iota, P) = \emptyset$.

In case (1), we have $[\![supp_{\psi,\iota,P}]\!]([t]_P^*) \subseteq \psi(t)$ and $[\![supp_{\psi,\iota,P}]\!]([t']_P^*) \subseteq \psi(t')$; since $[t]_P^* = [t']_P^*$, it follows that $[\![supp_{\psi,\iota,P}([t']_P^*)]\!] = \emptyset$, and thus $\Psi^U(\psi, \iota, P) = \emptyset$.

In case (2), we have $[\![supp_{\psi,\iota,P}]\!]([t]_P^*) \subseteq \psi(t)$ and $[\![supp_{\psi,\iota,P}]\!]([t']_P^*) \subseteq \psi(t')$. Therefore, if $\psi\colon [sub[K]]_P^* \to \mathcal{B}_\bot^*$, we must have $\psi([t]_P^*) = \psi([t']_P^*)$. Since $[t]_P^*, [t']_P^* \in WC(K)$ and $[t]_P^* \neq [t']_P^*$, we again conclude that $\Psi^U(\psi, \iota, P)$.

(3) For each $t \in sub[K]$, we have $supp_{\psi,\iota,P} \subseteq supp_{\lambda,\iota,P}$; therefore, $\Psi^U(\psi, \iota, P) \subseteq \Psi^U(\lambda, \iota, P)$.

– We have $[\![supp_{\psi,\iota,P}(t)]\!] \subseteq [\![\overline{supp_\iota}(t)]\!]$; thus,

$$\overline{supp_\iota}(t) = \mathsf{error} \Rightarrow [\![supp_{\psi,\iota,P}]\!] \subseteq \{\bot\}.$$

We also have $[\![supp_{\psi,\iota,P}(t)]\!] \subseteq \{\psi(t)\}$; thus,

$$\psi(t) \neq \bot \Rightarrow [\![supp_{\psi,\iota,P}(t)]\!] = \emptyset.$$

In this case there can be no function in $\Psi^U(\psi, \iota, P)$, contradicting the hypothesis.

– If $\overline{supp_\iota}(t) \neq \mathsf{error}$, then $[\![supp_{\psi,\iota,P}([t]^*_P)]\!] \subseteq \{\psi(t)\}$. Since $\psi^u \in \Psi^U(\psi, \iota, P)$, we must have $\psi^u([t]^*_P) = \psi_t$.

(4) For each $C \in [sub[K]]^*_P$, either there is $t \in K \cap C$ such that $\overline{supp_\iota}(t) \neq \mathsf{error}$ or not. If there is no such t, then we have $[\![supp_{\psi,\iota,P}(C)]\!] = \{\bot\}$. Otherwise, $[\![supp_{\psi,\iota,P}([t]^*_P)]\!] \subseteq \{\psi(t)\}$. Thus, for all $C \in [sub[K]]^*_P$, $[\![supp_{\psi,\iota,P}(C)]\!]$ is a set with a single element. Since $\psi^u(C) \in [\![supp_{\psi,\iota,P}(C)]\!]$ for all $\psi^u \in \Psi^U(\psi, \iota, P)$ and all $C \in [sub[K]]^*_P$, it follows that $\Psi^U(\psi, \iota, P)$ has at most one element.

if there is $\psi^u \in \Psi^U(\lambda, \iota, P)$ such that, for each $t \in sub[K]$, $\psi(t) = \bot$ if $\overline{supp_\iota}(t) = \mathsf{error}$ and $\psi(t) = \psi^u([t]^*_P)$ otherwise, then $\psi^u \in \Psi^U(\psi, \iota, P)$.

(5) It is sufficient to prove that, for each $t \in sub[K]$, $\psi^u([t]^*_P) \in [\![supp_{\psi,\iota,P}([t]^*_P)]\!]$. Let $t \in sub[K]$ and $C = [t]^*_P$. If there exists $t' \in C$ such that $\overline{supp_\iota}(t') \neq \mathsf{error}$, then $[\![supp_{\psi,\iota,P}(C)]\!] \subseteq [\![supp_{\lambda,\iota,P}(C)]\!] \cap \{\psi(t')\}$; because $\psi^u(C) = \psi(t')$, we have $\psi(t') \in [\![supp_{\lambda,\iota,P}(C)]\!]$. In this case, for all $t' \in C$, we have $\psi^u(C) = \{\psi(t')\}$ and $\{\psi(t')\} = [\![supp_{\lambda,\iota,P}(C)]\!]$. If no such t exists, then we have $[\![supp_{\lambda,\iota,P}(C)]\!] = [\![supp_{\lambda,\iota,P}(C)]\!] = \{\bot\}$, and since $\psi^u \in \Psi^U(\lambda, \iota, P)$, we have $\psi^u(C) = \bot$ and $\psi^u(C) \in [\![supp_{\psi,\iota,P}(C)]\!]$. □

Lemma 12. *Let $\lambda \in \Lambda_K$. We have*

$$\mu_C(\lambda) = \sum_{\psi \in \Psi(\lambda)} \mu_C(\psi).$$

Proof. We have to show that

$$\sum_{\psi \in \Psi(\lambda)} \mu_C(\psi) = \sum_{\iota \in I_S(K)} \left(\sum_{P \in \mathcal{P}^W_R(K)} \frac{|\Psi^U(\lambda, \iota, P)|}{|\Psi^D(\lambda, \iota, P)|} \right) \tag{7}$$

We have

$$\mu_C(\psi) = \sum_{\iota \in I_S(K)} \left(\sum_{P \in \mathcal{P}^W_R(K)} \frac{|\Psi^U(\psi, \iota, P)|}{|\Psi^D(\psi, \iota, P)|} \right).$$

Noting that $\Psi^D(\lambda, \iota, P) = \Psi^D(\psi, \iota, P)$, we conclude

$$\sum_{\psi \in \Psi(\lambda)} \mu_C(\psi) = \sum_{\iota \in I(K)} \left(\sum_{P \in \mathcal{P}^W_R(K)} \left(\sum_{\psi \in \Psi(\lambda)} \frac{|\Psi^U(\psi, \iota, P)|}{|\Psi^D(\psi, \iota, P)|} \right) \right). \tag{8}$$

By Lemma 11, for each $\psi \in \Psi(\lambda)$, each $\iota \in I(K)$, and each $P \in \mathcal{P}_R^W(K)$, $\left|\Psi^U(\psi, \iota, P)\right| = 1$ if $\iota = \iota(\psi)$, $P = P(\psi)$, and there exists $\psi^u \in \Psi^U(\psi, \iota, P)$ such that, for each $t \in K$, $\psi(t) = \perp$ if $\overline{supp_\iota}(t) = \mathsf{error}$ and $\psi(t) = \psi^u([t]_P^*)$ otherwise. Otherwise, $\left|\Psi^U(\psi, \iota, P)\right| = 0$. It is simple to check that, for each $\psi^u \in \Psi^U(\lambda, \iota, P)$ there is one and only one $\psi \in \Psi(\lambda)$ such that this condition is satisfied. Thus, for each $\iota \in I(K)$ and each $P \in \mathcal{P}_R^W(K)$, there are $\left|\Psi^U(\lambda, \iota, P)\right|$ functions $\psi \in \Psi(\lambda)$ such that $\left|\Psi^U(\psi, \iota, P)\right| = 1$, and $\left|\Psi^U(\psi', \iota, P)\right| = 0$ for all other $\psi' \in \Psi(\lambda)$. We obtain

$$\sum_{\psi \in \Psi(\lambda)} \left|\Psi^U(\psi, \iota, P)\right| = \left|\Psi^U(\lambda, \iota, P)\right|.$$

Combining this equality and (8), we obtain (7), concluding the proof. □

Lemma 13. *Let $\lambda_i \in \Lambda$ for each $i \in \{1, \ldots, n\}$. Suppose that the sets $\Omega_{\lambda_1}, \ldots, \Omega_{\lambda_n}$ are pairwise disjoint, and that $\lambda \in \Lambda$ is such that $\Omega_\lambda = \biguplus_{i=1}^n \Omega_{\lambda_i}$. Then,*

$$\mu_C(\Omega_\lambda) = \sum_{i=1}^n \mu_C(\Omega_{\lambda_i}).$$

Proof. Let $K = \bigcup_{i=1}^n dom(\lambda_i)$. For each i, there is $\lambda_i' \in \Lambda_K$ such that $\Omega_{\lambda_i'} = \Omega_{\lambda_i}$. Furthermore, there is $\lambda' \in \Lambda_K$ such that $\Omega_{\lambda'} = \Omega_\lambda$.

Now we note that

$$\Psi(\lambda') = \biguplus_{i=1}^n \Psi(\lambda_i'),$$

and the result follows from Lemmas 9 and 12:

$$\mu_C(\Omega(\lambda)) = \mu_C(\Omega(\lambda')) = \sum_{\psi \in \Psi(\lambda')} \mu_C(\Omega_\psi)$$

$$= \sum_{i=1}^n \left(\sum_{\psi \in \Psi(\lambda_i')} \mu_C(\Omega_\psi) \right) = \sum_{i=1}^n \mu_C(\Omega_{\lambda_i'}) = \sum_{i=1}^n \mu_C(\Omega_{\lambda_i}).$$

□

Lemma 14. *Suppose that $\mu_C(\emptyset) = 0$, $\mu_C(\mathbf{\Omega}) = 1$ and, whenenver $\lambda_1, \ldots, \lambda_n \in \Lambda$ are such that $\Omega_{\lambda_1}, \ldots, \Omega_{\lambda_n}$ are disjoint sets such that $\biguplus_{i=1}^n \Omega_{\lambda_i} = \Omega_\lambda$ for some $\lambda \in \Lambda$, we have*

$$\mu_C(\Omega_\lambda) = \sum_{i=1}^n \mu_C(\Omega_{\lambda_i}).$$

Then, there is a unique extension of μ_C to $\mathcal{F}$ that is a probability measure.

Proof. The proof is entirely similar to the proof of Lemma 9. The only difference is that instead of considering $supp_{PS}$ as in Lemma 7, we consider $supp_{PS}$ as in Corollary 3. □

Proof (Theorem 10). Let $\lambda = \emptyset$. We have $\mathbf{\Omega} = \Omega_\lambda$, and it is trivial to check that $\mu_C(\lambda) = 1$. Moreover, if $\lambda = \{t \mapsto \emptyset\}$ for some term t, we have $\emptyset = \Omega_\lambda$, and $\mu_C(\lambda) = 0$. Therefore, to conclude that μ_C is a probability measure it suffices to combine Lemmas 14 and 13.

Now, μ_C and μ_{ROM} are probability distributions (by theorems 10 and 5), and thus the set $\Omega_\Psi = \{\Omega_\psi \mid \psi \in \Psi\}$ is a generator of $\mathcal{F}$. Therefore, proving that $\mu_C(\Omega_\psi) = \mu_{ROM}(\Omega_\psi)$ for all $\psi \in \Psi$ is sufficient to prove that $\mu_C = \mu_{ROM}$. Let $K = sub[dom(\psi)]$, and let $\psi' = \psi \cup \{t \mapsto \mathcal{B}^*_\perp \mid t \in K \setminus dom(\psi)\}$. We have $\Omega_\psi = \Omega_{\psi'}$, and thus $\mu_C(\Omega_\psi) = \mu_C(\Omega_{\psi'})$ and $\mu_{ROM}(\Omega_\psi) = \mu_{ROM}(\Omega_{\psi'})$, by theorems 9 and 4. Thus, we can assume without loss of generality that $dom(\psi) = sub[dom(\psi)] = K$.

Lemma 4 and Lemma 11 imply that, if ψ does not satisfy P or there is no $\iota \in I_{\mathcal{S}}(K)$ such that $\psi \models \iota$, then $\mu_{ROM}(\psi) = \mu_C(\psi) = 0$ (note that all parcels in the sum that defines μ_C are 0), and the result is proved.

On the other hand, if this is not the case, we have

$$\mu_C(\Omega_\psi) = \frac{\left|\Psi^U(\psi, \iota, P)\right|}{\left|\Psi^D(\psi, \iota, P)\right|}$$

and, by Corollary 2,

$$\mu_{ROM}(\Omega_\psi) = \prod_{t^+ \in \tau^+[sub[K]] \setminus \mathcal{N}^+} \frac{1}{|[\![ran(\iota(\tau^K(t^+)))]\!]|},$$

where $\tau^K : \tau^+[sub[K]] \to sub[K]$ is the function such that, for each $t^+ \in \tau^+[sub[K]]$, $\tau^K(t^+)$ is the least $t \in sub[K]$ with respect to $\prec$ such that $\tau^+(\tau^K(t^+)) = t^+$ and $\iota(t) \neq \perp$ if such a t exists, and $\perp$ otherwise.

By Lemma 11, we have $\left|\Psi^U(\psi, \iota, P)\right| = 1$. On the other hand, the definitions of $\left|\Psi^D(\psi, \iota, P)\right|$ and $\overline{supp_\iota}$ imply that

$$\left|\Psi^D(\psi, \iota, P)\right| = \prod_{C \in [sub[K]]^+_P} |[\![\overline{supp_\iota}(C)]\!]| = \prod_{t^+ \in \tau^+[sub[K]] \setminus \mathcal{N}^+} \left|[\![ran(\iota(\tau^K(t^+)))]\!]\right|.$$

Combining the above equations, we conclude that $\mu_C(\Omega_\psi) = \mu_{ROM}(\Omega_\psi)$, proving the result. □