Solving the insecurity problem for assertions

R. Ramanujam ✉
The Institute of Mathematical Sciences, Chennai (Retd.)
Homi Bhabha National Institute, Mumbai (Retd.)
Azim Premji University, Bengaluru (Visiting)

Vaishnavi Sundararajan ✉
Indian Institute of Technology Delhi, New Delhi, India

S. P. Suresh[1] ✉
Chennai Mathematical Institute, Chennai, India
CNRS UMI 2000 ReLaX

Abstract

In the symbolic verification of cryptographic protocols, a central problem is deciding whether a protocol admits an execution which leaks a designated secret to the malicious intruder. In [38], it is shown that, when considering finitely many sessions, this "insecurity problem" is NP-complete. Central to their proof strategy is the observation that any execution of a protocol can be simulated by one where the intruder only communicates terms of bounded size. However, when we consider models where, in addition to terms, one can also communicate logical statements about terms, the analysis of the insecurity problem becomes tricky when both these inference systems are considered together. In this paper we consider the insecurity problem for protocols with logical statements that include *equality on terms* and *existential quantification*. Witnesses for existential quantifiers may be unbounded, and obtaining small witness terms while maintaining equality proofs complicates the analysis considerably. We extend techniques from [38] to show that this problem is also in NP.

2012 ACM Subject Classification Security and privacy → Logic and verification

Keywords and phrases Security protocols; Assertions; Insecurity problem; Equality and existentials; Small witnesses

1 Introduction

1.1 Symbolic analysis of cryptographic protocols

Symbolic analysis of cryptographic protocols is a long-standing field of study, with the Dolev-Yao model [22] being the standard. In this model, cryptographic operations are abstracted as operators in a term algebra, and the ability to build new messages from old ones is specified by rewrite rules or a proof system. The model includes an intruder who controls the network, and can see, block, inject, redirect, as well as derive terms, but cannot break cryptography. Informally, protocols are specified as a finite sequence of *communications* between *principals/agents*. We illustrate this model with the following example.

▶ **Example 1.** Alice sends to Bob her public key as well as a randomly-chosen value encrypted in Bob's public key. Bob receives it, decrypts it using his private key, encrypts it in Alice's public key, and sends it back to her. We split each communication into a send and a receive. We formalize the protocol as two *roles*: an *initiator* role $\mathsf{init}(A, B)$ (left column) and a *responder* role $\mathsf{resp}(B)$ (right

[1] Partially supported by a grant from the Infosys Foundation.

column). We use $!C$ and $?C$ to denote a send and a receive respectively by an agent $C \in \{A, B\}$. k_A and k_B stand for the private keys of A and B respectively, $pk(k)$ stands for the public key corresponding to a key k, and $\{t\}_k$ stands for the encryption of a message t using a key k.

$$\begin{array}{rl|rl}
A: & \text{Generate fresh } m & ?B: & (x, \{y\}_{pk(k_B)}) \\
!A: & (pk(k_A), \{m\}_{pk(k_B)}) & !B: & \{y\}_x \\
?A: & \{m\}_{pk(k_A)} & &
\end{array}$$

The protocol itself can be thought of as a program running potentially unboundedly-many copies (*sessions*) of init and resp in parallel. Each copy instantiates parameters A and B with agent names, while x and y denote parts of messages received while participating in a session, and will be instantiated accordingly. An *execution* (*run*) of a protocol is an interleaving of a finite set of sessions, such that every sent message can be generated by the sender (based on their current knowledge), and received messages by the intruder I (since every received message comes from the channel, and could have been potentially tampered with by the intruder).

Is there any execution of this protocol at the end of which the intruder can derive m? This property is called *confidentiality*. In fact, the intruder can effect the following man-in-the-middle attack, at the end of which A thinks m is secret between her and B, while B thinks m is secret between him and I. B receives a message where x can be matched with $pk(k_I)$ and y with m, and thus sends out $\{m\}_{pk(k_I)}$.

$$\begin{array}{rlrl}
!A: & (pk(k_A), \{m\}_{pk(k_B)}) & & \\
 & & ?B: & (pk(k_I), \{m\}_{pk(k_B)}) \\
 & & !B: & \{m\}_{pk(k_I)} \\
?A: & \{m\}_{pk(k_A)} & &
\end{array}$$

1.2 Communicating "assertions"

The Dolev-Yao model and its extensions have been studied extensively over the last forty years. People have studied extensions that express richer classes of protocols and security properties [1, 7, 10, 17], and associated decidability and complexity results [2, 8, 9, 12, 14–16, 18, 20, 23, 30, 36, 37]. Various verification tools have also been built based on these formal models [10, 11, 13, 21, 33].

In this paper, we consider an extension introduced in [35], which gives agents the power to communicate terms as well as logical formulas about them. These formulas, called *assertions*, involve equality of terms, existential quantification, conjunction, and disjunction. For instance, we can reveal partial information about some encrypted term $\{m\}_k$ to a recipient who does not know the key k (for instance, that the value of m is either 0 or 1, without revealing which) by sending the assertion $\exists xy \left[\{x\}_y = \{m\}_k \land x \in \{0, 1\}\right]$. So we see that assertions allow us to model protocols that involve some kinds of *certification*. Traditionally, such certification is often modelled using *zero-knowledge proofs*.

The Dolev-Yao model can also be extended with a special class of zero-knowledge terms [6, 7]. But in these extensions, one important component is missing: logical reasoning over certificates. This is especially important in situations where certificates communicate partial information. For example, two partial-information certificates of the form $x \in \{0, 1\}$ and $x \in \{0, 2\}$ can lead to the inference of strictly greater information, namely $x = 0$, potentially violating some security guarantees. This is one of the main features of the model in [35]. Making "assertions", as that paper refers to such logical statements, first-class citizens provides a threefold advantage: a more transparent specification of protocols which captures design intent better, the ability to explicitly reason about certificates and thus analyze protocols more precisely, and the ability to state some security properties more easily. In [35], the authors express examples (the FOO [24] and Helios [3] e-voting protocols) and specify security properties using assertions. We describe the modelling of the FOO protocol in detail in Section 2.3.

In [35], any communicated assertion is "believed" by the recipients. One way to implement this feature is to communicate a zero knowledge proof of the assertion. But formally, we send the assertion itself rather than a term standing for a zero-knowledge proof, which also allows us the possibility of choosing other implementations for the assertion. Another way in which [35] differs from modelling assertions using ZKP terms is that these proofs need not be built ab initio every time. One can compose a new proof by combining existing proofs. These can be implemented using composable ZKPs [26]. These issues have been discussed in [31], which considers a logical language with conjunction and existential quantification and modular construction of ZKPs for these formulas. However, unlike [31], assertions also allow "destructive reasoning" from existing knowledge via elimination rules.

The main focus in this paper is to solve an interesting technical problem in our model with assertions – the *insecurity problem for finitely many sessions*.

1.3 The insecurity problem for finitely many sessions

The attack on Example 1 indicates that even for simple protocols, one needs to consider non-trivial scenarios to detect security violations. A canonical problem of interest is the *insecurity problem*, which asks if a given protocol admits a run that leaks a secret to the intruder. A run is characterized by an interleaving of protocol roles (init and resp in Example 1), with a substitution for the variables in messages received by agents during these roles. There can be infinitely many such substitutions, i.e. a potentially infinite number of executions, and thus, the insecurity problem is undecidable in general [4, 23, 27]. In [38], the authors consider a restricted set of runs, and show that the insecurity problem is in NP when one considers at most K sessions, for some fixed K.

Even with only a finite number of sessions, the intruder can inject arbitrarily large terms in place of variables. Thus, there is no bound on the size of terms encountered in a run. The work in [38] gets around this complication by showing that if there is any attack at all given by an interleaving of roles and a substitution, there is an attack given by the same interleaving and a 'small' substitution. This "new" attack is such that the intruder can derive the same terms at the end, and the size of all messages transmitted is bounded by a polynomial in the size of the protocol specification. Hence the insecurity problem with boundedly many sessions can be solved in NP.

As with terms, one can formulate the insecurity problem for assertions as well. The general problem continues to be undecidable, so we consider the case of finitely many sessions. With existential quantification, we now have two types of variables – those used to identify parts of received messages (instantiated at runtime by the actual message sent by the intruder), and quantified variables that occur in assertions. As earlier, there is no a priori bound on the size of terms assigned to the first kind of variables. But there is another source of unboundedness: to derive a quantified assertion $\exists x. \alpha$, one must derive $\alpha(t)$ for some "witness" t. There is no a priori bound on the size of t either, and proof search is further complicated by any potential interaction between these two sources of unboundedness. When we simulate a substitution for the "intruder" variables with a small one, the witnesses for quantifiers might change too, but we still need to preserve some derivations under these new witnesses.

We extend the techniques of [38], while considering interactions between multiple substitutions and having to preserve more complex derivations, to obtain a somewhat surprising result – the insecurity problem for assertions for finitely many sessions remains in NP.

1.4 Related work

There are many extensions of the basic Dolev-Yao model that aim to capture various cryptographic operators and their properties [2,8,15–17,20,30]. Algebraic properties of operators like XOR, blinding, distributive encryption &c. are studied by means of *equation theories*, which are also referred to as *intruder theories* in the security literature. Equations in these theories are implicitly universally quantified, and the intention is that any term matching one side of the equation may be replaced by the other side. For example, if the theory contains a rule of the form

$$unblind(sign(blind(x, y), k), y) = sign(x, k),$$

it means that any instance of the LHS can be replaced by the corresponding instance of the RHS. Such equations correspond to proof rules in the system for deriving terms in this paper (examples of such systems are given in Section 2.1).

Equality assertions, on the other hand, are to be treated literally, and not as rewrite rules. For instance, given an assertion of the form $\{x\}_k = \{t\}_k$, we cannot replace all terms of the form $\{u\}_k$ by $\{t\}_k$. In fact, these equality assertions are objects that are manipulated by proof rules, rather than being another style of expressing derivations between terms.

Along with studying the derivability problem for such extensions, several of these papers also extend the results of [38] by addressing the active intruder problem for finitely many sessions. For instance, [15,16] obtain NP decision procedures in the case of extending Dolev-Yao with rules for XOR. The current paper, however, extends [38] along a different dimension, to solve both the passive and active intruder problems for assertions, and is thus not subsumed by any of these works on equation theories.

1.5 Organization of the paper

In Section 2, we first introduce the syntax for terms and assertions. We present an example of modelling with assertions via the FOO e-voting protocol, and then present the proof system for assertions. Then we define protocols and runs for this new system. In Section 3, we first present a high-level overview of the various steps involved in solving the insecurity problem, and then we move on to Section 4, where we present the technical results in detail and prove that insecurity for the assertion system is in NP. We present some ideas for future research in Section 5.

$$\frac{X \vdash t_1 \quad \cdots \quad X \vdash t_n}{X \vdash \mathsf{f}(t_1, \dots, t_n)} \qquad \frac{X \vdash \mathsf{f}(t_1, \dots, t_n) \quad X \vdash u_1 \quad \cdots \quad X \vdash u_m}{X \vdash t_i}$$

Figure 1 General form of constructor and destructor rules

2 Modeling security protocols

2.1 Terms: Syntax and Derivation System

In this model, each communicated message is modelled as a term in an algebra, which has operators for pairing, encryption, hashing &c. New terms can be derived from old ones using proof rules, which specify the behaviour of these operators. We begin with a set N of names (atomic terms, with no further structure), and a set of variables V. We denote by $\mathsf{A} \subseteq \mathsf{N}$ the set of agents, with $I \in \mathsf{A}$ being the malicious intruder. We denote by $\mathsf{V}_q \subset \mathsf{V}$ the variables used for quantification, and by V_i the set $\mathsf{V} \setminus \mathsf{V}_q$. The set of terms, denoted by T, is given by

$$t \in \mathsf{T} ::= x \mid m \mid \mathsf{f}(t_1, \dots, t_n)$$

where $x \in \mathsf{V}$, $m \in \mathsf{N}$, $t_1, \dots t_n \in \mathsf{T}$, and f is an n-ary operator. The set of *ground* terms are those without variables. A substitution σ is a partial function with finite support from V_i to T. Its domain is denoted by $\mathsf{dom}(\sigma)$. We assume that $\sigma(x) = x$ for $x \notin \mathsf{dom}(\sigma)$. The set of subterms of t is denoted by $\mathsf{st}(t)$, and defined as usual. The set of variables appearing in t is denoted by $\mathsf{vars}(t)$.

Each f has constructor rules and destructor rules, expressed in terms of sequents of the form $X \vdash t$ (to be read as "t is derived from X"), where $X \cup \{t\}$ is a finite set of terms. Figure 1 gives the general form of a constructor rule (on the left) and a destructor rule (on the right). In a destructor rule, the conclusion t_i is an immediate subterm of the leftmost premise, which is designated as the *major premise* of the rule. The ax rule (which derives $X \vdash t$ when $t \in X$) is also considered a destructor rule for technical purposes. We say $X \vdash_{dy} t$ if there is a proof of $X \vdash t$ using these constructor and destructor rules, and $X \vdash_{dy} S$ to mean that $X \vdash_{dy} t$ for every $t \in S$.

For any proof π of $X \vdash t$, we denote by $\mathsf{axioms}(\pi)$ the set X, by $\mathsf{conc}(\pi)$ the term t, and by $\mathsf{terms}(\pi)$ all terms occurring in π. π is said to be normal if a constructor rule does not yield the major premise of a destructor rule. We only consider proof systems which enjoy the following three properties:

- *Normalization*: Every proof π of $X \vdash t$ can be converted into a normal proof ϖ of the same.
- *Subterm property*: For any normal proof ϖ of $X \vdash t$, $\mathsf{terms}(\varpi) \subseteq \mathsf{st}(X \cup \{t\})$, and if ϖ ends in a destructor rule, $\mathsf{terms}(\varpi) \subseteq \mathsf{st}(X)$.
- *Efficient derivability checks*: There is a PTIME algorithm for checking derivability.

The normalization and subterm properties combined are referred to as *locality* in the security literature. This is a notion identified in [32], and is crucially used in solving the derivability problem for many classes of inference systems, including many intruder theories.

▶ **Example 2.** A term algebra with pairing, symmetric and asymmetric encryption operations, where $m, k \in \mathsf{N}$ and $t, u \in \mathsf{T}$ is given by $t := m \mid pk(k) \mid (t, u) \mid \{t\}_k \mid \{\!|t|\!\}_{pk(k)}$. The proof system for this algebra is shown in Table 1. This system enjoys normalization and the subterm property [38].

$\dfrac{}{X \cup \{t\} \vdash t}\,\mathsf{ax}$	$\dfrac{X \vdash (t_1, t_2)}{X \vdash t_i}\,\mathsf{split}$	$\dfrac{X \vdash \{t\}_k \quad X \vdash k}{X \vdash t}\,\mathsf{sdec}$	$\dfrac{X \vdash \{\!\!\{t\}\!\!\}_{pk(k)} \quad X \vdash k}{X \vdash t}\,\mathsf{adec}$
$\dfrac{X \vdash k}{X \vdash pk(k)}\,\mathsf{pk}$	$\dfrac{X \vdash t \quad X \vdash u}{X \vdash (t, u)}\,\mathsf{pair}$	$\dfrac{X \vdash t \quad X \vdash k}{X \vdash \{t\}_k}\,\mathsf{senc}$	$\dfrac{X \vdash t \quad X \vdash pk(k)}{X \vdash \{\!\!\{t\}\!\!\}_{pk(k)}}\,\mathsf{aenc}$

Table 1 Proof system for the term algebra in Example 2

2.2 Assertions

We consider an assertion syntax which includes equality over terms (to avoid overloading the $=$ operator, we denote equality between t and u by $t \bowtie u$), predicates, conjunction, existentially quantified assertions, list membership, and a *says* connective. Existential quantification allows us to make statements that convey partial information about terms, in particular, allowing us to hide terms or parts thereof. The *says* connective works like a signature over assertions, indicating who endorses the fact conveyed by the assertion. List membership, which we denote by $\twoheadleftarrow$, acts as a restricted form of disjunction. Predicates allow us to express some protocol-specific facts. As we will see over the later sections, this fragment allows us to express example protocols of interest, as well as yields a decidable active intruder problem for boundedly many sessions.

In the following, $t, u \in \mathrm{T}$, P is an m-ary predicate, $u_1, \ldots, u_m, t_0 \in \mathrm{N} \cup \mathrm{V}$, and $t_1, \ldots, t_n \in \mathrm{N}$,[2] $x \in \mathrm{V}_q$, and $pk(k)$ is the public key corresponding to a secret key k.

$$\alpha ::= t \bowtie u \mid \mathrm{P}(u_1, \ldots, u_m) \mid t_0 \twoheadleftarrow [t_1, \ldots, t_n] \mid \alpha_0 \wedge \alpha_1 \mid \exists x.\, \alpha(x) \mid pk(k)\ \mathit{says}\ \alpha$$

By *atomic assertions*, we mean assertions that are not of the form $\alpha \wedge \beta$ or $\exists x \alpha$.

We denote the free (resp. bound) variables occurring in an assertion α by $\mathsf{fv}(\alpha)$ and $\mathsf{bv}(\alpha)$. $\mathsf{vars}(\alpha) = \mathsf{fv}(\alpha) \cup \mathsf{bv}(\alpha)$. The set of subterms (resp. subformulas) of α is given by $\mathsf{st}(\alpha)$ (resp. $\mathsf{sf}(\alpha)$). We can lift these notions to sets of assertions as usual. For a substitution λ, we obtain $\lambda(\alpha)$ by replacing x in α by $\lambda(x)$ for all $x \in \mathsf{fv}(\alpha)$.

We now define the *public terms* of an assertion α. These are essentially the terms that α is "about", which are always communicated along with α. Quantified variables in an assertion stand for "private" terms, so if a term t occurring in α has quantified variables, it cannot itself be public. But it is not reasonable to declare all other subterms to be public terms either. For instance, if an assertion talks about $\mathsf{senc}(v, k)$, the term $\mathsf{senc}(v, k)$ should be public, but probably not v or k itself. Hence we define the public terms of α, denoted $\mathsf{pubs}(\alpha)$, as the set of all *maximal* subterms of α which contain no quantified variables. In other words, $t \in \mathsf{pubs}(\alpha)$ iff $t \in \mathsf{st}(\alpha)$, $\mathsf{vars}(t) \cap \mathrm{V}_q = \emptyset$, and $\forall u \in \mathsf{st}(\alpha) : t \in \mathsf{st}(u) \implies \mathsf{vars}(u) \cap \mathrm{V}_q \neq \emptyset$.

▶ **Example 3.** A (with secret key k) encrypts a vote v in a key r unknown to B and states that it is one of two allowed values.

$$\mathrm{A} \rightarrow \mathrm{B} : \{v\}_r,\ pk(k)\ \mathit{says}\ \left\{\exists xy.\{x\}_y \bowtie \{v\}_r \wedge x \twoheadleftarrow [0, 1]\right\}$$

The set of public terms of this assertion is $\{\{v\}_r, 0, 1\}$.

[2] We could consider arbitrary terms in list membership, but this simple syntax suffices for most examples. Similarly for $\mathrm{P}(u_1, \ldots .u_m)$.

Assertions, like terms, can be involved in sends and receives. However, since assertions are logical formulas, we can also have agents check them for derivability and take some action based on the result of this check, without any send/receive. We call such an action an assert. As part of an assert α action, an agent A checks to see if α is derivable from their current knowledge. If it is, A continues with their role, otherwise A aborts. An assert action allows us to model some minimal branching based on the derivability of assertions from agents' local states.

Note that this does not involve any absolute notion of the "truth" (or lack thereof) of an assertion. An agent can only locally check if an assertion can be "verified", i.e. obtained from what they know about the system at that point in the execution. It might well be the case that while an assert α check passes for an agent A, a different agent B might not have enough information to be able to derive α, and abort. Conversely, if some agent's internal state has been compromised somehow and made inconsistent, they might even be able to assert something like $0 = 1$, which is patently false. We are only concerned with the verifiability of assertions, and not their absolute truth values.

Having introduced this system, we now present the modelling of the well-known FOO e-voting protocol [24]. This is a minor modification of the presentation in [35].

2.3 Example: FOO e-voting Protocol

The FOO e-voting protocol was proposed in 1992 and closely mirrors the way one votes offline. There is a voter V, an authority A who verifies voter identities, and a collector C who computes the final tally.

To model this using only terms [24, 29], *blinding* is used. One can use t and b to make a *blind pair* $\mathsf{blind}(t, b)$, and get $\mathsf{sign}(t, k)$ from $\mathsf{sign}(\mathsf{blind}(t, b), k)$ and b. The voter authenticates themselves to the authority using their signing key sk_V, and uses the blinding operation to have the authority certify it without knowing the actual vote. The authority's signature $\mathsf{sign}(\cdot, sk_A)$ percolates through to the vote when the voter removes the blind, and the voter can then anonymously send (denoted by $\looparrowright$) this signed vote to the collector for inclusion into the final tally. This specification is shown below.

$$
\begin{aligned}
V \rightarrow A &: \mathsf{sign}(\mathsf{blind}(\{v\}_r, b), sk_V)\\
A \rightarrow V &: \mathsf{sign}(\mathsf{blind}(\{v\}_r, b), sk_A)\\
V \looparrowright C &: \mathsf{sign}(\{v\}_r, sk_A)
\end{aligned}
$$

We model the voting phase of FOO as below, following [35]. In fact, the use of assertions allows one to also specify an eligibility check for voters via an assert. If the user is not eligible, the protocol aborts. Further, voters can also state that their vote is for an allowable candidate from the list l. These are left implicit in the terms-only modelling.

$$
\begin{aligned}
V \rightarrow A &: \{v\}_p, V\ \mathit{says}\ \left[\exists xr.\{x\}_r \bowtie \{v\}_p \wedge x \twoheadleftarrow l\right]\\
A &: \mathsf{assert}\ \mathsf{el}(V)\\
A \rightarrow V &: A\ \mathit{says}\ \left\{\mathsf{el}(V) \wedge V\ \mathit{says}\ \left[\exists xr.\{x\}_r \bowtie \{v\}_p \wedge x \twoheadleftarrow l\right]\right\}\\
V \looparrowright C &: \{v\}_q, \exists Uys.\, A\ \mathit{says}\ \left\{\mathsf{el}(U) \wedge U\ \mathit{says}\ \left[\exists xr.\{x\}_r \bowtie \{y\}_s \wedge x \twoheadleftarrow l\right]\right\} \wedge \left\{\exists w.\{y\}_w \bowtie \{v\}_q\right\}
\end{aligned}
$$

V first sends to A their encrypted vote along with an assertion claiming that it is for a candidate from the list l. The authority checks the voter's eligibility via the assert action on the el predicate. If the check passes, the authority issues a certificate stating that the voter is allowed to vote, crucially, without modifying the term containing the vote. V then existentially quantifies out their name from this certificate, and anonymously sends to C a re-encryption of the vote authorized by A along with a certificate to that effect. Here, p and q are freshly-generated ephemeral keys. Thus, the intent behind the various communications is made more transparent than in the model with blind signatures. One can show that this satisfies anonymity [35].

One can also specify security properties in a more natural manner (as compared to the terms-only model). For instance, one can say that *vote secrecy* is ensured in the above protocol if there is no run where the intruder can derive the assertion $\exists xy : [\{v\}_p = \{x\}_y \land x = v]$. Note that this means that while anyone can derive the value of v, which is public, they should not be able to identify the value inside the encrypted vote $\{v\}_p$ as being a particular public name. To express this in the terms-only formulation, one has to check whether two runs that only differ in the vote v can be distinguished by the intruder [19]. It can be seen from [35] that proving such properties might involve considering multiple runs simultaneously, but their specification itself does not refer to a notion of equivalence.

▶ **Example 4.** Consider a protocol where V sends to A the vote encrypted in a fresh key k, and an assertion that the vote belongs to an allowable list l of candidates. This looks as follows. $V \rightarrow A : \{v\}_k, \exists xr. \{\{x\}_r \bowtie \{v\}_k \land x \twoheadleftarrow l\}$.

Suppose this same protocol is used for two elections that V participates in simultaneously, where the first election has candidates 0 and 1 (so $l_1 = [0, 1]$) and the second has candidates 0 and 2 (so $l_2 = [0, 2]$).

V wants to vote for 0 in both elections. Since the vote is for the same candidate, V (unwisely) decides to reuse the same term, instead of re-encrypting in a fresh key. So we have a run where V sends both $\exists xr. \{\{x\}_r \bowtie \{v\}_k \land x \twoheadleftarrow [0, 1]\}$ and $\exists ys. \big\{\{y\}_s \bowtie \{v\}_k \land y \twoheadleftarrow [0, 2]\big\}$. Now, since the same term $\{v\}_k$ is involved in both assertions, an observer ought to be able to deduce that the vote is actually for 0. This would allow them access to both the identity of a voter as well as their vote, falsifying anonymity. The assertion system formally captures such inference via a proof system.

2.4 Abstractability and Proof System

Before we present the proof system, we need to fix under what conditions one can derive a new assertion from existing ones. In a security context, it becomes important to distinguish when a term is accessible inside an assertion versus when it is not. To substitute a term u (with, say, v) inside a term t, an agent A essentially needs to break the term down to that position, replace u with v, and construct the whole term back. This depends on other terms A has access to. We formalize this notion as "abstractability", which requires us to first define the set of term positions of an assertion.

We will view terms as trees, with $\mathbb{P}(t) \subseteq \mathbb{N}^*$ denoting the set of positions of the term t, and ε the empty word in $\mathbb{N}^*$. We will also view assertions as trees, with any operator forming the root of its subtree, and its operands standing for its children. We will only be interested in the position where terms occur in assertions, not those of the various operators. We define these as follows.

▶ **Definition 5** (Term positions of an assertion). *We define the term positions of an assertion α, denoted $\mathbb{P}(\alpha)$, as follows:*

- $\mathbb{P}(t \bowtie t') = \{0 \cdot p \mid p \in \mathbb{P}(t)\} \cup \{1 \cdot p \mid p \in \mathbb{P}(t')\}$
- $\mathbb{P}(P(u_0, \dots, u_m)) = \{0, \dots, m\}$
- $\mathbb{P}(t \twoheadleftarrow [t_1, \dots, t_n]) = \{0, 1, \dots, n\}$
- $\mathbb{P}(\alpha \wedge \beta) = \{0 \cdot p \mid p \in \mathbb{P}(\alpha)\} \cup \{1 \cdot p \mid p \in \mathbb{P}(\beta)\}$
- $\mathbb{P}(\exists x.\alpha) = \{0 \cdot p \mid p \in \mathbb{P}(\alpha)\}$
- $\mathbb{P}(pk(k) \; says \; \alpha) = \{0, 00\} \cup \{1 \cdot p \mid p \in \mathbb{P}(\alpha)\}$

For $t, r \in \mathsf{T}$, and $p \in \mathbb{P}(t)$, $t|_p$ is the subterm of t rooted at p. The set of positions of r in t is $\mathbb{P}_r(t) \coloneqq \{p \in \mathbb{P}(t) \mid t|_p = r\}$. For $P \subseteq \mathbb{P}(t)$, $t[r]_P$ is obtained by replacing the subterm of t occurring at each $p \in P$ with r. We will use analogous notation for assertions.

▶ **Definition 6** (Abstractable positions of a term). *Let $S \cup \{t\} \subseteq \mathsf{T}$. The set of abstractable positions of t w.r.t. S, denoted $\mathbb{A}(S, t)$, is defined as follows. For $p \in \mathbb{P}(t)$, let $\mathbb{Q}_p = \{\varepsilon\} \cup \{qi \in \mathbb{P}(t) \mid q$ is a proper prefix of $p\}$. Then $\mathbb{A}(S, t) \coloneqq \{p \in \mathbb{P}(t) \mid S \vdash_{dy} t|_q$ for all $q \in \mathbb{Q}_p\}$.*

For $S = \{\{m\}_k\}_{k'}, (n_1, n_2)$ and $t = (\{\{m\}_k\}_{k'}, (n_1, n_2))$, $\mathbb{P}(t) = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001\}$ and $\mathbb{A}(S, t) = \{\varepsilon, 0, 1, 10, 11\}$. The abstractable positions are shown in bold in Figure 2.

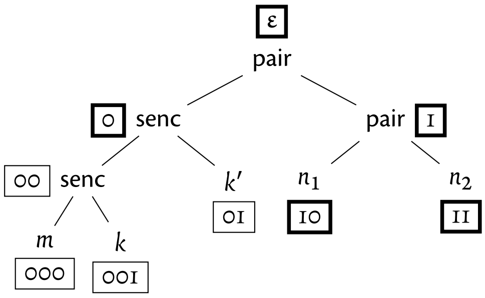


Figure 2 Abstractable positions w.r.t. $S = \{\{m\}_k\}_{k'}, (n_1, n_2)\}$

Now, an inductive definition seems like it might suffice to lift the notion of abstractable positions for assertions. However, a problem arises when we consider an assertion of the form $\exists x.\alpha$. Let $\alpha = \exists b.\{\{m\}_b \bowtie \{m\}_k\}$. Suppose we want to get $\exists ab.\{\{a\}_b \bowtie \{m\}_k\}$ from α in the presence of the set $S = \{m, k\}$. That position of m in α must be abstractable w.r.t S, i.e. we require that $S \vdash_{dy} \{m\}_b$, but S does not even contain the quantified variable b. We must therefore consider derivability from $S \cup \{b\}$ in this case, not S.

▶ **Definition 7** (Abstractable positions of an assertion). *The set of* abstractable positions of α w.r.t. S, *denoted by $\mathbb{A}(S, \alpha)$, is:*

- $\mathbb{A}(S, t_0 \bowtie t_1) = \{i \cdot p \mid i \in \{0, 1\}, \; p \in \mathbb{A}(S, t_i)\}$
- $\mathbb{A}(S, P(u_1, \dots, u_m)) = \{i \mid 1 \leqslant i \leqslant m, S \vdash_{dy} u_i\}$
- $\mathbb{A}(S, t \twoheadleftarrow [t_1, \dots, t_n]) = \{0\}$
- $\mathbb{A}(S, \alpha_0 \wedge \alpha_1) = \{i \cdot p \mid i \in \{0, 1\}, \; p \in \mathbb{A}(S, \alpha_i)\}$
- $\mathbb{A}(S, \exists x.\alpha) = \{0 \cdot p \mid p \in \mathbb{A}(S \cup \{x\}, \alpha)\}$
- $\mathbb{A}(S, pk(k) \; says \; \alpha) = \{0\} \cup \{1 \cdot p \mid p \in \mathbb{A}(S, \alpha)\}$

We now state a fundamental property of abstractability, which will be used in some of the more technical proofs later.

▶ **Lemma 8.** *Let $S \cup \{t, r\} \subseteq \mathrm{T}$ s.t. $S \vdash_{dy} r$. If $x \notin \mathsf{vars}(S)$ and $P = \mathbb{P}_x(t) \subseteq \mathbb{A}(S \cup \{x\}, t)$, then $\mathbb{A}(S, t[r]_P) \cap \mathbb{P}(t) = \mathbb{A}(S \cup \{x\}, t)$. Similarly, if $x \notin \mathsf{vars}(S)$ and $P = \mathbb{P}_x(t) \subseteq \mathbb{A}(S \cup \{x\}, \beta)$, then $\mathbb{A}(S, \beta[r]_P) \cap \mathbb{P}(\beta) = \mathbb{A}(S \cup \{x\}, \beta)$.*

Proof. We only prove the statement for terms. The statement for assertions follows from it. For any term a and any set $Q \subseteq \mathbb{P}(a)$, we let $a|_Q$ denote $\{a|_q \mid q \in Q\}$. We now observe some general properties of abstractability.

For any T, a and $q \in \mathbb{A}(\mathrm{T}, a)$ s.t. $a|_q = \mathsf{f}(a_0, \ldots, a_n)$, either $\{q0, \ldots, qn\} \subseteq \mathbb{A}(\mathrm{T}, a)$ and $a|_{\{q0,\ldots,qn\}} \vdash_{dy} a|_q$ via a constructor rule, or q is a maximal position in $\mathbb{A}(\mathrm{T}, a)$ (it is not the prefix of any other position in the set). We have the following two properties.

1. Let $M = \{q \in \mathbb{P}(a) \mid q$ is a maximal position in $\mathbb{A}(\mathrm{T}, a)\}$. Then for every $p \in \mathbb{A}(\mathrm{T}, a)$, $a|_M \vdash_{dy} a|_p$ via a proof consisting only of constructor rules.
2. Suppose $Q \subseteq \mathbb{P}(a)$ is prefix-closed (if $q \in Q$ and p is a prefix of q, then $p \in Q$) and sibling-closed (if $qi \in Q$ and $qj \in \mathbb{P}(a)$, then $qj \in Q$). If $\mathrm{T} \vdash_{dy} a|_q$ for every maximal $q \in Q$, then $Q \subseteq \mathbb{A}(\mathrm{T}, a)$.

We now prove the statement of the lemma. Let $u = t[r]_P$, and let A and B denote $\mathbb{A}(S \cup \{x\}, t)$ and $\mathbb{A}(S, u) \cap \mathbb{P}(t)$ respectively. Note that A and B are both prefix-closed and sibling-closed. Let M (resp. N) be the set of maximal positions in A (resp. B).

Since $P \subseteq A$ is the set of x-positions in t, $P \subseteq M$ and no $q \in M$ is a prefix of a position in P. Thus, for every $q \in M$, either $t|_q = x$, or $x \notin \mathsf{vars}(t|_q)$. If $t|_q = x$, $u|_q = r$, and $S \vdash_{dy} u|_q$ (since $S \vdash_{dy} r$). If $x \notin \mathsf{vars}(t|_q)$, then $u|_q = t|_q$ and $S \vdash_{dy} u|_q$. This is because $q \in \mathbb{A}(S \cup \{x\}, t)$, so $S \cup \{x\} \vdash_{dy} t|_q$, but x does not occur in the conclusion. Thus we have $S \vdash_{dy} u|_q$ for every $q \in M$. Since A is prefix-closed and sibling-closed, by 2, we get $A \subseteq \mathbb{A}(S, u)$. Since $A \subseteq \mathbb{P}(t)$ as well, we get $A \subseteq B$.

By similar reasoning as above, we can see that $S \cup \{x\} \vdash_{dy} t|_q$ for each $q \in N$. (For some of these positions q, x does not occur at all in the subterm at that position, and $t|_q = u|_q$ is derivable from S. For other positions q, $t|_q = x$ and is derivable from $S \cup \{x\}$.) Therefore $B \subseteq A$. ⊣

The assertion proof system is shown in Table 2. We say $S; A \vdash_a \alpha$ if α can be derived from $S; A$ using these rules. We say $S; A \vdash_a \Gamma$ if $S; A \vdash_a \gamma$ for every $\gamma \in \Gamma$.

We say that $S; A \vdash_{eq} \alpha$ if α can be derived from $S; A$ by a proof which does not use any of the rules from $\{\wedge\mathsf{i}, \wedge\mathsf{e}, \exists\mathsf{i}, \exists\mathsf{e}, \mathsf{say}\}$. Recall that an atomic assertion is one that is not of the form $\alpha \wedge \beta$ or $\exists x.\alpha$. The $\vdash_{eq}$ system is used typically when $A \cup \{\alpha\}$ consists only of atomic assertions, and we want to ensure that there is no use of the rules for $\wedge$ and $\exists$ in these proofs. To ensure this, we also need to avoid the say rule. Otherwise, we might allow a derivation of $pk(k)$ *says* $(\alpha \wedge \beta)$ using $\alpha \wedge \beta$, which itself can be derived only using $\wedge$i (since the LHS contains only atomic assertions).

The proofs in Section 4 crucially appeal to some properties of $\vdash_{eq}$ proofs, which we detail below.

▶ **Definition 9.** *Suppose $\mathrm{E} \cup \{\alpha\}$ consists only of atomic formulas and π is a proof of $(\mathrm{T}; \mathrm{E}) \vdash_{eq} \alpha$. We use "$\mathrm{r}_1$ precedes r_2 in π" to mean that the conclusion of some application of r_1 is a premise of an application of r_2 in π.*

We say that π is normal *if the following hold.*

1. *All $\vdash_{dy}$ subproofs are normal.*
2. sym *is only preceded by* ax *or* prom.

$\dfrac{}{S; A \cup \{\alpha\} \vdash \alpha}\,\mathsf{ax}$	$\dfrac{S \vdash_{dy} t}{S; A \vdash t \bowtie t}\,\mathsf{eq}$	$\dfrac{S; A \vdash t_1 \bowtie u_1 \cdots S; A \vdash t_r \bowtie u_r}{S; A \vdash \mathsf{f}(t_1, \dots, t_r) \bowtie \mathsf{f}(u_1, \dots, u_r)}\,\mathsf{cons}$
$\dfrac{S; A \vdash t \bowtie u}{S; A \vdash u \bowtie t}\,\mathsf{sym}$	$\dfrac{S; A \vdash t_1 \bowtie t_2 \cdots S; A \vdash t_k \bowtie t_{k+1}}{S; A \vdash t_1 \bowtie t_{k+1}}\,\mathsf{trans}$	$\dfrac{S; A \vdash \mathsf{f}(t_1, .., t_r) \bowtie \mathsf{f}(u_1, .., u_r)}{S; A \vdash t_i \bowtie u_i}\,\mathsf{proj}_i^{\dagger}$
$\dfrac{S; A \vdash \alpha_0 \quad S; A \vdash \alpha_1}{S; A \vdash \alpha_0 \wedge \alpha_1}\,\wedge\mathsf{i}$	$\dfrac{S; A \vdash \alpha_0 \wedge \alpha_1}{S; A \vdash \alpha_i}\,\wedge\mathsf{e}_i$	$\dfrac{S; A \vdash t \twoheadleftarrow l \quad S; A \vdash t \bowtie u}{S; A \vdash u \twoheadleftarrow l}\,\mathsf{subst}$
$\dfrac{S; A \vdash \alpha[t]_P \quad S \vdash_{dy} t}{S; A \vdash \exists x.\alpha}\,\exists\mathsf{i}^{\ddagger}$	$\dfrac{S; A \vdash \exists x.\alpha \quad S \cup \{y\}; A \cup \{\alpha[y]_P\} \vdash \gamma}{S; A \vdash \gamma}\,\exists\mathsf{e}^{\S}$	$\dfrac{S; A \vdash \alpha \quad S \vdash_{dy} k}{S; A \vdash pk(k)\ \mathit{says}\ \alpha}\,\mathsf{say}$
$\dfrac{S; A \vdash t \twoheadleftarrow [n]}{S; A \vdash t \bowtie n}\,\mathsf{prom}$	$\dfrac{S; A \vdash t \twoheadleftarrow l_1 \dots S; A \vdash t \twoheadleftarrow l_m}{S; A \vdash t \twoheadleftarrow (l_1 \cap \dots \cap l_m)}\,\mathsf{int}$	$\dfrac{S; A \vdash t \bowtie n_i \quad S \vdash_{dy} n_1 \cdots S \vdash_{dy} n_k}{S; A \vdash t \twoheadleftarrow [n_1, \dots, n_k]}\,\mathsf{wk}$

Table 2 Derivation system $\vdash_a$ for assertions.

†: $\{0i, 1i \mid i \leqslant r\} \subseteq \mathbb{A}(S, \mathsf{f}(t_1, \dots, t_r) \bowtie \mathsf{f}(u_1, \dots, u_r))$. ‡: $P = \mathbb{P}_x(\alpha) \subseteq \mathbb{A}(S \cup \{x\}, \alpha)$.

§: $y \notin \mathsf{fv}(S) \cup \mathsf{fv}(A) \cup \mathsf{fv}(\gamma)$ and $P = \mathbb{P}_x(\alpha)$.

3. eq *is only preceded by a destructor rule.*
4. *No premise of a* trans *is of the form* $a \bowtie a$, *or the conclusion of a* trans.
5. *Adjacent premises of a* trans *are not conclusions of* cons.
6. int *is not preceded by* int *or* wk.
7. *No subproof ending in* proj *contains* cons.

We would like to prove a *normalization theorem* – whenever $(T; E) \vdash_{eq} \alpha$ then there is a normal proof of $(T; E) \vdash \alpha$ in the $\vdash_{eq}$ system. But this is unfortunately not possible for arbitrary sets of equalities E. Consider the following derivation of $(T; E) \vdash a \bowtie t$, where $T = \{a, b, c, d, r, s, t, u\}$ and $E = \{\mathsf{f}(a, b) \bowtie \mathsf{g}(c, d), c \bowtie r, d \bowtie s, \mathsf{g}(r, s) \bowtie \mathsf{f}(t, u)\}$. (We omit the LHS in the proofs, for the sake of readability.)

$$\dfrac{\dfrac{}{\mathsf{f}(a,b) \bowtie \mathsf{g}(c,d)}\,\mathsf{ax} \qquad \dfrac{\dfrac{}{c \bowtie r}\,\mathsf{ax} \quad \dfrac{}{d \bowtie s}\,\mathsf{ax}}{\mathsf{g}(c,d) \bowtie \mathsf{g}(r,s)}\,\mathsf{cons} \qquad \dfrac{}{\mathsf{g}(r,s) \bowtie \mathsf{f}(t,u)}\,\mathsf{ax}}{\dfrac{\mathsf{f}(a,b) \bowtie \mathsf{f}(t,u)}{a \bowtie t}\,\mathsf{proj}}\,\mathsf{trans}$$

The above proof is not normal (it ends in proj but contains an application of cons), and there is no way to modify it to a normal proof. If E had contained the equalities $\mathsf{f}(a, b) \bowtie \mathsf{f}(c, d)$ and $\mathsf{f}(r, s) \bowtie \mathsf{f}(t, u)$ instead of the ones involving g, then we would have the following normal proof of $a \bowtie t$.

$$\dfrac{\dfrac{\dfrac{}{\mathsf{f}(a,b) \bowtie \mathsf{f}(c,d)}\,\mathsf{ax}}{a \bowtie c}\,\mathsf{proj} \qquad \dfrac{}{c \bowtie r}\,\mathsf{ax} \qquad \dfrac{\dfrac{}{\mathsf{f}(r,s) \bowtie \mathsf{f}(t,u)}\,\mathsf{ax}}{r \bowtie t}\,\mathsf{proj}}{a \bowtie t}\,\mathsf{trans}$$

The difference between the two cases is the presence of the equations of the form $\mathsf{f}(\cdots) \bowtie \mathsf{g}(\cdots)$ in E. We can capture this more abstractly by the notion of *consistency*. We say that $(T; E)$ is consistent if there is a substitution λ such that $\lambda(a) = \lambda(b)$ whenever $(T; E) \vdash_{eq} a \bowtie b$ and $\lambda(t) \in \{t_1, \dots, t_n\}$ whenever $(T; E) \vdash_{eq} t \twoheadleftarrow [t_1, \dots, t_n]$. We will assume throughout the rest of the paper that we are only dealing with consistent sets.[3]

We now state the normalization theorem and subterm property for $\vdash_{eq}$ proofs. First, we define the following notions.

- $\mathsf{terms}(\pi) \coloneqq \{t \mid$ a subproof of π derives α and t is a maximal subterm of $\alpha\}$.
- $\mathsf{lists}(E) \coloneqq \{l \mid \exists t : t \twoheadleftarrow l$ is in $E\}$.
- $\mathsf{lists}(\pi) \coloneqq \{l \mid$ a subproof of π derives $t \twoheadleftarrow l\}$.

▶ **Theorem 10** (Normalization & Subterm Property for $\vdash_{eq}$).

1. *If $(T; E) \vdash_{eq} \alpha$ then there is a normal proof of $(T; E) \vdash \alpha$ in the $\vdash_{eq}$ system.*
2. *For any normal proof π of $T; E \vdash_{eq} \alpha$, letting $Y = \mathsf{st}(T) \cup \mathsf{st}(E \cup \{\alpha\})$, we have:*
   - $\mathsf{terms}(\pi) \subseteq Y$.
   - $\mathsf{lists}(\pi) \subseteq \mathsf{lists}(E \cup \{\alpha\}) \cup \{[n] \mid n \in Y\}$.

Armed with these notions, we present a saturation-based procedure in Algorithm 1 for deciding whether $T; E \vdash_{eq} \alpha$, where $E \cup \{\alpha\}$ consists only of atomic assertions. The procedure computes the set

$$\mathsf{E}^{\alpha}_{T,E} \coloneqq \left\{\beta \mid \beta \text{ is atomic}, \beta \in Z, (T; E) \vdash_{eq} \beta\right\}$$

where Z is as defined in Algorithm 1, and checks if $\alpha \in \mathsf{E}^{\alpha}_{T,E}$.

Algorithm 1 Algorithm to compute $\mathsf{E}^{\alpha}_{T,E}$, given $(T; E), \alpha$

```
$Y \leftarrow \mathsf{st}(S) \cup \mathsf{st}(E \cup \{\alpha\})$;
$Z \leftarrow \left\{\beta \mid \beta \text{ is atomic}, \mathsf{st}(\beta) \in Y, \mathsf{lists}(\beta) \subseteq \mathsf{lists}(E) \cup \{[n] \mid n \in Y\}\right\}$;
$B \leftarrow \emptyset$;
$C \leftarrow E$;
while $(B \neq C)$ do
    $B \leftarrow C$;
    $C \leftarrow B \cup \left\{\beta \in Z \mid \beta \text{ can be obtained from } B \text{ using one application of any rule in } \vdash_{a}\right\}$;
end while
return $B$.
```

Letting $M = |\mathsf{st}(T) \cup \mathsf{st}(E \cup \{\alpha\})|$ and $N = |\mathsf{lists}(E)|$, it can be seen that the algorithm runs in time polynomial in $M + N$. There at most $(M + N)^2$ atomic formulas that can be added in C, and hence the **while** loop runs for at most $(M + N)^2$ iterations. In each iteration, the amount of work to be done is polynomial in $M + N$. (Recall that $\vdash_{dy}$ can be decided in PTIME.) Thus the algorithm works in time polynomial in $M + N$, and hence polynomial in the size of $(T; E \cup \{\alpha\})$.

[3] The $(T; E)$ we will encounter while solving the insecurity problem will be consistent, as we shall see later.

2.5 Protocols and runs

Following [10, 38], a *protocol* is given by a finite set of roles, each role consisting of a finite sequence of alternating receives and sends (each send triggered by a receive).[4] These are the actions of *honest agents*. Every sent message is added to the Dolev-Yao intruder's knowledge base. Each received message is assumed to have come from the intruder, so it must be derivable by the intruder. We assume that only assertions are communicated – a term t can be modelled via the assertion $t \bowtie t$, whose only public term is t.

A *protocol* Pr is a finite set of *roles*, each of the form $(\beta_1, \alpha_1) \dots (\beta_m, \alpha_m)$, where the α_is and β_is are assertions. An $x \in \mathsf{fv}(Pr)$ is said to be an *agent variable* if it occurs first in an α_i; otherwise it is an *intruder variable*. Each role is a sequence of actions by an agent, receiving the β_is and sending the α_is in response. The α_is and β_is can have bound variables from V_q as well as free variables from V_i. Instantiating the free variables with appropriately-typed ground terms yields a *session*. A *run* is obtained by interleaving a finite number of sessions that satisfy the required derivability conditions. It is convenient to instantiate the free variables of a role in two stages. Agent variables are instantiated with names before starting a session, but intruder variables can be mapped to terms only at runtime.

A *session* of a protocol Pr is a sequence of the form $u : \beta_1 \Rightarrow \alpha_1 \ \cdots \ u : \beta_l \Rightarrow \alpha_l$ where $u \in \mathsf{A}$ and $(\beta_1, \alpha_1) \cdots (\beta_l, \alpha_l)$ is a prefix of a role of Pr with all the agent variables instantiated by values from N. A set of sessions S of Pr is *coherent* if $\mathsf{fv}(\xi) \cap \mathsf{fv}(\xi') = \emptyset$ for distinct $\xi, \xi' \in S$. One can always achieve coherence by renaming intruder variables as necessary.

A run is an interleaving of sessions where each message sent by an agent should be constructible from their knowledge. A *knowledge state* is a pair $(X; \Phi)$ where X is a finite set of terms and Φ is a finite set of assertions. A *knowledge function* k is such that $\mathsf{dom}(\mathsf{k}) = \mathsf{A}$ and for each $a \in \mathsf{A}$, $\mathsf{k}(a)$ is a knowledge state.

For a knowledge state $(X; \Phi)$ and an assertion α, $\mathit{update}((X; \Phi), \alpha) \coloneqq (X \cup \mathsf{pubs}(\alpha), \Phi \cup \{\alpha\})$.

▶ **Definition 11.** *A* run *of a protocol Pr is a pair (ξ, σ) where:*

- $\xi \coloneqq u_1 : \beta_1 \Rightarrow \alpha_1, \dots, u_n : \beta_n \Rightarrow \alpha_n$ *is an interleaving of a finite, coherent set of sessions of Pr.*
- σ *is a ground substitution with* $\mathsf{dom}(\sigma) = \mathsf{fv}(\xi)$.
- *There is a sequence $\mathsf{k}_0 \dots \mathsf{k}_n$ of knowledge functions s.t.:*
  - $\mathsf{k}_0(a) = (X_a; \emptyset)$, *where X_a is a finite set of initial terms known to a (a's secret key, public keys, public names etc).*
  - *For all $i < n$,*

$$\mathsf{k}_{i+1}(a) = \begin{cases} \mathsf{k}_i(a) & \text{if } a \neq u_i, a \neq I \\ \mathit{update}(\mathsf{k}_i(a), \beta_i) & \text{if } a = u_i \\ \mathit{update}(\mathsf{k}_i(a), \alpha_i) & \text{if } a = I \end{cases}$$

  - *For $i \leqslant n$, $\mathsf{k}_i(u_i) \vdash_a \alpha_i$ and $\sigma(\mathsf{k}_{i-1}(I)) \vdash_a \sigma(\beta_i)$.*

[4] Apart from send and receive actions, we can also consider actions of the form assert α to model rudimentary branching in protocols, which we used for specifying the FOO protocol. But we omit these in the formal model, for ease of presentation. We discuss handling such branching in Section 5.4.

Note that honest agent derivations of the form $\mathsf{k}_i(u_i) \vdash_a \alpha_i$ do not depend on accidental unification with intruder variables under σ; rather, they hold even in the "abstract".

We can write an A-session and a B-session for the Example 1 protocol as $A : \beta_1 \Rightarrow \alpha_1, A : \beta_3 \Rightarrow \alpha_3$ and $B : \beta_2 \Rightarrow \alpha_2$. (To save space, we denote by p_A and p_B the keys $pk(k_A)$ and $pk(k_B)$.) We assume that A starts a session by receiving a dummy name s, and ends the session by sending s out, and code up each communicated term t from Example 1 as the assertion $t \bowtie t$. Note that A, B, p_A, m, and p_B are names used to instantiate agent variables in these sessions. The set of these two sessions is coherent.

$$
\begin{array}{lcl@{\qquad\qquad}lcl}
\beta_1 & = & s \bowtie s & \alpha_1 & = & \{(p_A, \{m\}_{p_B}) \bowtie (p_A, \{m\}_{p_B})\} \\
\beta_2 & = & \{(x, \{y\}_{p_B}) \bowtie (x, \{y\}_{p_B})\} & \alpha_2 & = & \{\{y\}_x \bowtie \{y\}_x\} \\
\beta_3 & = & \{\{m\}_{p_A} \bowtie \{m\}_{p_A}\} & \alpha_3 & = & s \bowtie s
\end{array}
$$

Consider the substitution $\sigma = [x \mapsto p_A, y \mapsto m]$ applied to $\xi = A : \beta_1 \Rightarrow \alpha_1, B : \beta_2 \Rightarrow \alpha_2, A : \beta_3 \Rightarrow \alpha_3$. This would be a run (ξ, σ) where the intruder just observes traffic on the network, but does not interfere otherwise.

Let $X_B = \{A, B, p_A, p_B, k_B\}$. $\mathsf{k}_0(B) = (X_B; \emptyset)$. Note that $\mathsf{k}_1(B) = \mathsf{k}_0(B)$. There is an update to B's knowledge state only upon receipt of β_2. So, $\mathsf{k}_2(B) = update(\mathsf{k}_1(B), \beta_2)$ is given by $(X'; \Phi)$ where $X' = X \cup \{(p_A, \{m\}_{p_B})\}$ and $\Phi = \{\{(p_A, \{m\}_{p_B}) \bowtie (p_A, \{m\}_{p_B})\}$.

We can also consider a run with the same ξ under a substitution $\sigma = [x \mapsto pk(k_I), y \mapsto m]$, which represents the man-in-the-middle attack shown earlier.

A *secrecy property* is given by an assertion γ that the intruder should not know. A K*-bounded attack* which violates the secrecy of γ is a run of the protocol with at most K sessions where $\sigma(\mathsf{k}_n(I)) \vdash_a \sigma(\gamma)$.

▶ **Definition 12** (K-bounded insecurity problem). *Given a protocol Pr and a designated assertion γ, check whether there exists a K-bounded attack on Pr violating the secrecy of γ.*

We will use "insecurity problem" to mean the K-bounded insecurity problem for some K.

3 Proof strategy for the insecurity problem

In subsequent sections, we will show that the K-bounded insecurity problem for assertions is in NP. But first, we provide an overview of the proof strategy we will employ.

Given a protocol Pr, a secrecy property specified by an assertion γ and a bound K (in unary), one way to check if there is a K-bounded attack works as follows: Guess a coherent set of sessions of size K, an interleaving $\xi = u_1 : \beta_1 \Rightarrow \alpha_1, \ldots, u_n : \beta_n \Rightarrow \alpha_n$, and a substitution σ with $\mathsf{dom}(\sigma) = \mathsf{fv}(\xi)$, and check that (ξ, σ) satisfies the conditions in Definition 11. For this, we need an effective check for derivabilities of the form $\sigma(\mathsf{k}_{i-1}(I)) \vdash_a \sigma(\beta_i)$.

As with terms, this needs us to bound the size of terms assigned to variables by σ. However, we also have quantified variables in our proofs, for which witnesses need to be assigned. To check whether a formula of the form $\exists x. \alpha$ is derivable, one would in general have to check if $\alpha(t)$ is derivable for some t, which might be unboundedly large. To get an effective algorithm, we have to show that if there is a witness at all, there is a witness of small size.

One way to represent these witnesses is via a substitution μ which maps each quantified variable x to the appropriate witness. To obtain small witnesses, we adapt the techniques of [38]. For this, it is helpful to first simplify the LHS to contain only atomic formulas. Any normal proof of α from such an LHS will not involve $\wedge$e or $\exists$e. We further show, via Theorem 16, that these proofs can be decomposed into multiple proofs, one for each atomic subformula of α (with witnesses instantiated by μ), and then applying $\wedge$i and $\exists$i.

Applying Theorem 16 to each derivability check $\sigma(\mathsf{k}_{i-1}(I)) \vdash_a \sigma(\beta_i)$ for $1 \leqslant i \leqslant n$, we get a set of witness substitutions $\{\mu_1, \dots, \mu_n\}$. We would like to ensure that all of these, along with σ, can be chosen to be “small”.

In order to obtain these small substitutions, we follow the techniques of [38]. This involves identifying and mapping “zappable” variables to atomic terms – these are variables that do not map to any term that “corresponds” to one in the protocol specification. However, unlike [38], we need to do this simultaneously for multiple substitutions – σ (which instantiates intruder variables) and μ_i (which instantiates quantified variables). The various μ_is might be influenced by σ, so preserving derivabilities when moving to small substitutions becomes a challenge. In order to do this, we employ a notion of “typed proofs”, both for the $\vdash_{dy}$ and $\vdash_{eq}$ systems. We show that any proof can be converted to a typed equivalent, and typed proofs make it easier for us to replace the substitutions therein with small ones while preserving derivations.

We will now present the solution in detail.

4 Solving the insecurity problem for $\vdash_a$

We fix a protocol Pr and a run (ξ, σ) of Pr. By renaming variables if necessary, we can ensure that $\mathsf{fv}(\xi) \cap \mathsf{V}_q = \emptyset$. Thus, in all proof sequents that we consider, no variable has both free and bound occurrences. We can also ensure that no variable is quantified by distinct quantifiers. Furthermore, whenever we use $(S; A)$, we mean that S is a set of terms, A is a set of assertions, and S derives the public terms of all assertions in A.

We also use $\mathsf{vars}(S; A)$ to mean $\mathsf{vars}(S) \cup \mathsf{vars}(A)$ and $\mathsf{fv}(S; A)$ to mean $\mathsf{vars}(S) \cup \mathsf{fv}(A)$.

As a first step, we move to an LHS consisting solely of atomic formulas. For this, we will employ the following two “left” properties enjoyed by the $\vdash_a$ system.

▶ **Lemma 13.**

1. *$(S; A \cup \{\alpha \wedge \beta\}) \vdash_a \gamma$ iff $(S; A \cup \{\alpha, \beta\}) \vdash_a \gamma$.*
2. *Let $S, A, \exists x.\alpha$ and γ be such that $x \notin \mathsf{vars}(S) \cup \mathsf{vars}(A \cup \{\gamma\})$ and $\mathbb{P}_x(\alpha) \subseteq \mathbb{A}(S \cup \{x\}, \alpha)$. Then $(S; A \cup \{\exists x.\alpha\}) \vdash_a \gamma$ iff $(S \cup \{x\}; A \cup \{\alpha\}) \vdash_a \gamma$.*

Proof. 1. To save space, we use A, φ to mean $A \cup \{\varphi\}$ in the proof to follow.

For the left to right direction, let π be a proof of $S; A, \alpha \wedge \beta \vdash \gamma$. The following is a proof of $S; A, \alpha, \beta \vdash \gamma$.

$$\dfrac{\dfrac{\dfrac{}{S; A, \alpha, \beta \vdash \alpha}\,\mathsf{ax} \qquad \dfrac{}{S; A, \alpha, \beta \vdash \beta}\,\mathsf{ax}}{S; A, \alpha, \beta \vdash \alpha \wedge \beta}\,\wedge\mathsf{i} \qquad \begin{array}{c}\pi\\ \vdots\\ S; A, \alpha \wedge \beta \vdash \gamma\end{array}}{S; A, \alpha, \beta \vdash \gamma}$$

For the other direction, let π be a proof of $S; A, \alpha, \beta \vdash \gamma$. We obtain a proof of $S; A, \alpha \wedge \beta \vdash \gamma$

below. We omit the $S; A$ part of the LHS to conserve space.

$$\dfrac{\dfrac{\dfrac{}{\alpha\wedge\beta\vdash\alpha\wedge\beta}\,\text{ax}}{\alpha\wedge\beta\vdash\beta}\,\wedge\text{e} \qquad \dfrac{\dfrac{\dfrac{}{\alpha\wedge\beta\vdash\alpha\wedge\beta}\,\text{ax}}{\alpha\wedge\beta\vdash\alpha}\,\wedge\text{e} \qquad \begin{matrix}\pi\\ \vdots\\ \alpha,\beta\vdash\gamma\end{matrix}}{\alpha\wedge\beta,\beta\vdash\gamma}}{\alpha\wedge\beta\vdash\gamma}$$

We have freely used the *cut rule*, which is admissible in our system.

$$\dfrac{S; A \vdash \varphi \qquad S; B, \varphi \vdash \psi}{S; A \cup B \vdash \psi}$$

If π_0 and π_1 are derivations of the left and right premises as above, then we can replace each axiom rule occurring in π_1 and deriving φ, with the proof π_0, thus yielding a proof of $S; A \cup B \vdash \psi$.

2. For the left to right direction, let π be a proof of $S; A, \exists x.\alpha \vdash \gamma$. Note that we have a proof π_1 of $\exists x.\alpha$ from $(S, x; A, \alpha)$, where the $\exists$i rule is justified because the abstractability side condition $\mathbb{P}_x(\alpha) \subseteq \mathbb{A}(S \cup \{x\}, \alpha)$ is assumed. We can then use the cut rule (which is admissible in $\vdash_a$) on this proof along with the proof π to get $(S, x; A, \alpha) \vdash_a \gamma$.

$$\dfrac{\dfrac{\dfrac{}{S, x; A, \alpha \vdash \alpha}\,\text{ax}}{S, x; A, \alpha \vdash \exists x.\alpha}\,\exists\text{i} \qquad \begin{matrix}\pi\\ \vdots\\ S; A, \exists x.\alpha \vdash \gamma\end{matrix}}{S, x; A, \alpha \vdash \gamma}\,\text{cut}$$

For the other direction, let π be a proof of $S, x; A, \alpha \vdash \gamma$. We obtain a proof of $S; A, \exists x.\alpha \vdash \gamma$ as follows.

$$\dfrac{\dfrac{}{S; A, \exists x.\alpha \vdash \exists x.\alpha}\,\text{ax} \qquad \begin{matrix}\pi\\ \vdots\\ S, x; A, \alpha \vdash \gamma\end{matrix}}{S; A, \exists x.\alpha \vdash \gamma}\,\exists\text{e}$$

⊣

This leads us to a notion of *kernel*.

▶ **Definition 14.** *The* atoms *of an assertion α, denoted $\mathsf{at}(\alpha)$, is the set of all maximal atomic subformulas of α. The* kernel *of $(S; A)$, denoted $\mathit{ker}(S; A)$, is given by $(T; E)$ where $T = S \cup \mathsf{bv}(A)$ and $E = \{\beta \in \mathsf{at}(\alpha) \mid \alpha \in A\}$.*

Any $x \in \mathsf{bv}(A)$ which is added to T can be thought of as an "eigenvariable" which witnesses an existential assertion in A. If we derive some γ from $(T \cup \{x\}; \beta)$, since we only consider γ such that $\mathsf{vars}(\gamma) \cap \mathsf{bv}(A) = \emptyset$, we can also derive it from $(T; \exists x.\beta)$. Lemma 13 can thus always be applied, and it can be shown that kernels preserve derivability, i.e. $(S; A) \vdash_a \gamma$ iff $\mathit{ker}(S; A) \vdash_a \gamma$ for any γ.

Here is another basic property of kernels, which is crucially used in many proofs later.

▶ **Lemma 15.** *Suppose $(T; E) = \mathit{ker}(S; A)$ for some $(S; A)$. If $(T; E) \vdash_a \alpha$ and $a \in \mathsf{pubs}(\alpha)$, then $T \vdash_{dy} a$. If $(T; E) \vdash_{eq} t \bowtie u$ then $T \vdash_{dy} t$ and $T \vdash_{dy} u$.*

Proof. Recall that we only consider $(S; A)$ such that $\mathsf{fv}(S; A) \cap \mathsf{V}_q = \emptyset$, and $S \vdash_{dy} \mathsf{pubs}(\beta)$ for all $\beta \in A$. Since $(T; E) = ker(S; A)$, we have $T = S \cup \mathsf{bv}(A)$ and $E = \{\gamma \in \mathsf{at}(\beta) \mid \beta \in A\}$. Thus $T \vdash_{dy} \mathsf{pubs}(\gamma)$ for every $\gamma \in E$, and $\mathsf{vars}(E) \cap \mathsf{V}_q \subseteq T$.

Let π be a proof of $(T; E) \vdash_a \alpha$. Note that π has no occurrence of $\exists\mathsf{e}$ or $\wedge\mathsf{e}$. We assume that all premises of eq are normal $\vdash_{dy}$ proofs ending in a destructor (by repeatedly turning all constructor + eq patterns into eq + cons). We show by induction that $T \vdash_{dy} \mathsf{pubs}(\alpha)$. Let r denote the last rule of π.

- r = ax: $\alpha \in E$. So $T \vdash_{dy} \mathsf{pubs}(\alpha)$.
- r = eq: α is $t \bowtie t$ with $T \vdash_{dy} t$ via a proof ending in destructor. Since any term in T is either in V_q or contains no variables from V_q, and since $t \in \mathsf{st}(T)$, we see that $\mathsf{pubs}(\alpha)$ is $\{t\}$ or $\emptyset$, and $T \vdash_{dy} \mathsf{pubs}(\alpha)$ in both cases.
- r $\in$ {sym, trans, prom, int, subst, $\wedge$i}: Any $t \in \mathsf{pubs}(\alpha)$ is in $\mathsf{pubs}(\beta)$ for one of the premises β, and the result follows.
- r = cons: α is of the form $t \bowtie u$, where $t = \mathsf{f}(t_1, \dots, t_k)$ and $u = \mathsf{f}(u_1, \dots, u_k)$, and the immediate subproofs of π derive $t_1 \bowtie u_1, \dots, t_k \bowtie u_k$. Now, any term in $\mathsf{pubs}(\alpha)$ is a public term of one of the premises (and we can apply IH), unless it is t or u. Say it is t. Then, t is a maximal subterm of α which avoid V_q, and thus it must be that $t_1, \dots, t_k$ are also public terms of the premises. Thus $T \vdash_{dy} \{t_1, \dots, t_k\}$ by IH, and hence $T \vdash_{dy} t$. Similarly for u.
- r = proj: α is $t \bowtie u$, and any public term of α is a public term of the premise (and we can apply IH), unless it is t or u. But by abstractability, $T \vdash_{dy} \{t, u\}$, and we are done.
- r = wk: α is $t \twoheadleftarrow [n_0, \dots, n_k]$, where t and all the n_i's are variables or names. The premise is $t \bowtie n_i$ for some i, and we also require that $S \vdash_{dy} n_i$ for all i. Combining this with the IH, we see that $S \vdash_{dy} \mathsf{pubs}(\alpha)$.
- r = say: α is of the form $pk(k)$ *says* β, and β is proved by the immediate subproof. We also have that $S \vdash k$ and hence $S \vdash pk(k)$. Any other public term occurring in α occurs in β, so by IH we have that $S \vdash_{dy} \mathsf{pubs}(\alpha)$.
- r = $\exists$i: α is of the form $\exists x.\beta$, with premise $\gamma = \beta[r]_{\mathrm{P}}$, where $\mathrm{P} = \mathbb{P}_x(\beta)$. We also have, by the other requirements for the rule, $T \vdash_{dy} r$ and $\mathrm{P} \subseteq \mathbb{A}(T \cup \{x\}, \beta)$. By Lemma 8, $\mathrm{P} \subseteq \mathbb{A}(T, \gamma)$. Consider any $a = \alpha|_q \in \mathsf{pubs}(\alpha)$. If $a \in \mathsf{pubs}(\gamma)$, then we can apply IH. Otherwise, q has to be a sibling of some position in $p \in \mathrm{P}$. In other words, a is public in α because its sibling is x, but in γ, the x is replaced by r (and $\mathsf{vars}(r) \cap \mathsf{V}_q = \emptyset$), so a is no longer a *maximal* subterm avoiding V_q. Since the set of abstractable positions is sibling-closed, $q \in \mathbb{A}(T, \alpha)$, and since subterms at abstractable positions are derivable, $T \vdash_{dy} a$.

Now consider an $\vdash_{eq}$ proof of $(T; E) \vdash t \bowtie u$. It has been shown above that $T \vdash_{dy} \mathsf{pubs}(t \bowtie u)$. Consider t. Either $t \in \mathsf{pubs}(t \bowtie u)$, in which case we are done. Otherwise, every maximal subterm of t which avoids V_q is derivable from T, and every $x \in \mathsf{vars}(t) \cap \mathsf{V}_q$ is in T. From these, we can "build up" t using constructor rules only, thereby proving that $T \vdash_{dy} t$. Similarly we can show that $T \vdash_{dy} u$. ⊣

As mentioned earlier, by proof normalization, we decompose a proof π of $(S; A) \vdash \alpha$ into several proofs of atomic subformulas of α (equalities, predicates, list membership, and *says* assertions), and a proof π_0 which uses these atoms as axioms, and applies $\wedge$i and $\exists$i, all with the kernel as LHS.

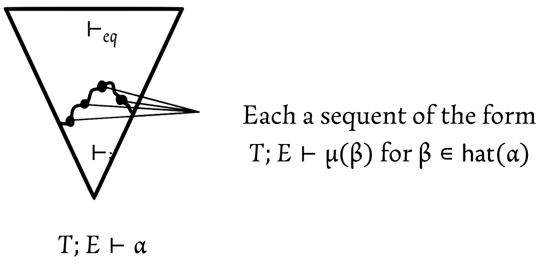
For each of these atomic subformulas, we would like to operate in a proof system which does not involve conjunction or existential quantification. This is easy to do for equalities, predicates, and lists, because the only way to derive such assertions is by deriving other equalities, predicates, and lists.

However, consider subformulas of the form $pk(k)\ says\ \beta$. We can derive those in two ways – either by using ax (if the formula is already in the LHS) or by using the say rule on β and k. In the latter case, β might contain logical operators! Thus, we need to break down β as well.

We thus formalize the *hereditary atoms* of a formula as:

$$\mathsf{hat}(\gamma) = \begin{cases} \mathsf{hat}(\alpha) \cup \mathsf{hat}(\beta) & \text{if } \gamma = \alpha \wedge \beta \\ \mathsf{hat}(\alpha) & \text{if } \gamma = \exists x.\alpha \\ \{pk(k)\ says\ \alpha\} \cup \mathsf{hat}(\alpha) & \text{if } \gamma = pk(k)\ says\ \alpha \\ \{\gamma\} & \text{otherwise} \end{cases}$$

We now reduce any proof of $S; A \vdash_a \alpha$ to one with a very particular structure, as depicted in Figure 3. This new proof has as its LHS the kernel $(T; E)$ of $(S; A)$, and derives α. This proof first involves multiple proofs, each of which is an $\vdash_{eq}$ proof[5] of some hereditary atom of α, with witnesses appropriately assigned to bound variables by a substitution μ. These proofs are then followed by applications of the ax, $\wedge$i, $\exists$i and say rules (represented by $\vdash_i$ in the Figure 3) to get α.



Each a sequent of the form $T; E \vdash \mu(\beta)$ for $\beta \in \mathsf{hat}(\alpha)$

Figure 3 Structure of the new proof guaranteed by Theorem 16

Consider the set X of all hereditary atoms of α which feature in the above reduction. Suppose $\beta \in X$ is of the form $pk(k)\ says\ (\exists x.\delta)$, but $\exists x.\delta \notin X$. Then β can only be derived from the LHS by the ax rule, since there is no other rule in the $\vdash_{eq}$ system that derives a *says* assertion. Thus we do not obtain $\exists x.\delta$ using the $\exists$i rule, and so we do not need to provide a witness for such an x. This is precisely formulated in the next theorem.

▶ **Theorem 16.** *For a formula α s.t. $\mathsf{bv}(\alpha) \cap \mathsf{vars}(S; A) = \emptyset$, and $(T; E) = ker(S; A)$, $(S; A) \vdash_a \alpha$ iff there is $X \subseteq \mathsf{hat}(\alpha)$ and μ with $\mathsf{dom}(\mu) = \mathsf{bv}(\alpha) \setminus \mathsf{bv}(X)$ s.t.:*

[a] $\forall x \in \mathsf{dom}(\mu) : T \vdash_{dy} \mu(x)$.
[b] *For all* $\beta \in X$, $(T; E) \vdash_{eq} \mu(\beta)$.
[c] $(T; \mu(X)) \vdash_a \alpha$ *via a proof using rules from* $\{\mathsf{ax}, \wedge\mathsf{i}, \exists\mathsf{i}, \mathsf{say}\}$.
[d] $\forall x \in \mathsf{dom}(\mu), t \in \mathsf{st}(\alpha)$: $\mathbb{P}_x(t) \subseteq \mathbb{A}(T \cup \mathsf{dom}(\mu), t)$.

In the statement of the theorem, [a] ensures that all witnesses are derivable, [b] ensures that all the atoms in X have a proof (with witnesses instantiated appropriately), and [c] ensures that the

[5] Recall that $\vdash_{eq}$ is the subsystem that does not use any rules from $\{\wedge\mathsf{i}, \wedge\mathsf{e}, \exists\mathsf{i}, \exists\mathsf{e}, \mathsf{say}\}$.

final intros-only proof exists. Finally, [d] ensures that the proper abstractability conditions for applications of ∃i are satisfied. For any set of assertions, we denote the set $\{x \in \mathsf{bv}(\beta) \mid \beta \in X\}$ by $\mathsf{bv}(X)$.

Proof of Theorem 16.

(⇒) Suppose $(S; A) \vdash_a \alpha$. Then, since kernels preserve derivability, $(T; E) \vdash_a \alpha$. Let π be a normal proof of $(T; E) \vdash \alpha$. Since E only has atomic assertions, it is easy to see that there is no occurrence of the ∧e and ∃e rules. Recall that we only consider α such that no x is quantified by quantifiers occurring in two distinct positions in α, and that no variable occurs both free and bound in α. For each $x \in \mathsf{bv}(\alpha)$ introduced in π via an ∃i application, let t_x be the witness used by the ∃i rule introducing the quantifier $\exists x$ in α. Define $\mu(x) \coloneqq t_x$ for each such x. The side conditions for the ∃i occurrences guarantee that $T \vdash_{dy} \mu(x)$ for each $x \in \mathsf{dom}(\mu)$, thus satisfying [a].

Let $X \subseteq \mathsf{hat}(\alpha)$ be all the hereditary atoms of α appearing on the RHS in various subproofs of π. By normalization, one can always place the logical rules after deriving atomic formulas. Hence, we can decompose π into proofs π_β of $(T; E) \vdash \mu(\beta)$ for each $\beta \in X$, and a proof $\hat{\pi}$ deriving $(T; \mu(X)) \vdash \alpha$ using only the ax, ∧i, ∃i and say rules. This proves [b] and [c].

We now prove [d]. It is clear that each subproof of $\hat{\pi}$ has conclusion $\mu(\beta)$ for some $\beta \in \mathsf{sf}(\alpha)$, with $\hat{\pi}$ itself deriving $\mu(\alpha) = \alpha$. We will now show that for every subproof π_0 of $\hat{\pi}$ with conclusion $\mu(\beta)$ and last rule r, we have (letting $Z_\beta = \mathsf{bv}(\beta) \setminus \mathsf{bv}(X)$):

$$\forall x \in Z_\beta, \forall t \in \mathsf{st}(\mu(\beta)) : \mathbb{P}_x(t) \subseteq \mathbb{A}(T \cup Z_\beta, t). \qquad (1)$$

r = ax: $\mu(\beta) \in \mu(X)$, so $Z_\beta = \emptyset$, and so (1) holds vacuously.

r = ∧i: β of the form $\beta_0 \wedge \beta_1$, and $\mathsf{bv}(\beta_0)$ and $\mathsf{bv}(\beta_1)$ are disjoint, and no variable has both free and bound occurrences. So no variable in $\mathsf{bv}(\beta_i)$ occurs in β_{1-i}. So if $x \in \mathsf{bv}(\beta_i)$, and any $t \in \mathsf{st}(\mu(\beta_{1-i}))$, then $\mathbb{P}_x(t) = \emptyset$. So (1) for π_0 follows by IH (applied on the immediate subproofs).

r = say: β is of the form $pk(k)$ *says* β' and every bound variable of β is also bound in β', so we get (1) from IH.

r = ∃i: $\beta = \exists z.\gamma$, and $\mu(\beta) = \exists z.\mu'(\gamma)$, where $\mu' = \mu \restriction (Z_\gamma)$. The immediate subproof of π_0 has conclusion $\mu(\gamma)$.

Now for any $r \in \mathsf{st}(\mu(\beta))$, letting $P = \mathbb{P}_z(r)$, $t = r[\mu(z)]_P \in \mathsf{st}(\mu(\gamma))$. For any $x \in \mathsf{V}$, we have $\mathbb{P}_x(r) = \mathbb{P}_x(t) \cap \mathbb{P}(r)$ and $\mathbb{A}(T \cup Z_\beta, r) = \mathbb{A}(T \cup Z_\gamma, t) \cap \mathbb{P}(r)$ (by Lemma 8).

By IH, for all $x \in Z_\gamma$ and $t \in \mathsf{st}(\mu(\gamma))$, $\mathbb{P}_x(t) \subseteq \mathbb{A}(T \cup Z_\gamma, t) \subseteq \mathbb{A}(T \cup Z_\beta, t)$. So for all $x \in Z_\beta \setminus \{z\}$ and $r \in \mathsf{st}(\mu(\beta))$, $\mathbb{P}_x(r) \subseteq \mathbb{A}(T \cup Z_\beta, r)$.

For z, the abstractability side condition for ∃i implies that for all $r \in \mathsf{st}(\mu(\beta))$, $\mathbb{P}_z(r) \subseteq \mathbb{A}(T \cup Z_\beta, r)$. Thus, equation (1) follows for π_0.

Applying (1) to $\hat{\pi}$, we get [d].

(⇐) This is the easier direction. We just compose all the $\vdash_{eq}$ proofs and the intros-only proof to obtain an $\vdash_a$ proof π of $(T; E) \vdash \alpha$. It can be shown that the ∃i is always enabled in π. To illustrate this, suppose α is $\exists xyz\beta$ and we wish to apply the ∃i rule on $(\exists yz\beta)[r]_P$ to get α, where $P = \mathbb{P}_x(\exists yz\beta)$. The abstractability condition for this rule would be $P \subseteq \mathbb{A}(T \cup \{x\}, \exists yz\beta)$. But expanding the definition of $\mathbb{A}$ for quantified assertions, this translates to $P \subseteq \mathbb{A}(T \cup \{x, y, z\}, \beta)$, which is guaranteed by [d].

Thus, $(T; E) \vdash_a \alpha$ iff $(S; A) \vdash_a \alpha$, and so we are done. $\dashv$

For the rest of the paper, we use the following notation. $(T_i; E_i) := \mathit{ker}(\mathsf{k}_i(I))$ and $(U_i; F_i) := \mathit{ker}(\mathsf{k}_i(u_i))$ for $1 \leqslant i \leqslant n$. Note that $T_i \subseteq T_{i+1}$ and $E_i \subseteq E_{i+1}$ for every i.

Since $\mathsf{dom}(\sigma) = \mathsf{fv}(\xi)$, we have $\sigma(x) = x$ for all $x \in \mathcal{V}_q$. It follows that $\sigma(\mathit{ker}(S; A)) = \mathit{ker}(\sigma(S; A))$, for any $(S; A)$.

Applying Theorem 16 to the $\sigma(\mathsf{k}_{i-1}(I)) \vdash_a \sigma(\beta_i)$ derivations in Definition 11, for every $i \leqslant n$ we get $X_i \subseteq \mathsf{hat}(\beta_i)$ and a substitution μ_i with domain $\mathsf{bv}(\beta_i) \setminus \mathsf{bv}(X_i)$ s.t.:

- for every $x \in \mathsf{dom}(\mu_i)$, $\sigma(T_{i-1}) \vdash_{dy} \mu_i(x)$, and
- $\sigma(T_{i-1}; E_{i-1}) \vdash_{eq} \sigma\mu_i(\gamma)$ for $\gamma \in X_i$.

For every $i \leqslant n$, Definition 11 also states $\mathsf{k}_i(u_i) \vdash_a \alpha_i$, and thus, $\sigma(\mathsf{k}_i(u_i)) \vdash_a \sigma(\alpha_i)$. So Theorem 16 guarantees $Y_i \subseteq \mathsf{hat}(\alpha_i)$ and a substitution θ_i with domain $\mathsf{bv}(\alpha_i) \setminus \mathsf{bv}(Y_i)$ s.t.:

- for every $x \in \mathsf{dom}(\theta_i)$, $\sigma(U_i) \vdash_{dy} \theta_i(x)$, and
- $\sigma(U_i; F_i) \vdash_{eq} \sigma\theta_i(\gamma)$, where $\gamma \in Y_i$.

For any $\gamma \in X_i \cup Y_i$, three possibilities arise.

- γ is of the form $t \bowtie u$.
- γ is of the form $pk(k)$ *says* δ. Such a formula can only be derived using ax, as no other rule in the $\vdash_{eq}$ system generates it. Hence such assertions can be ignored for the rest of this section, which is about preserving non-trivial $\vdash_{eq}$ proofs even after changing some substitutions.
- γ is of the form $P(u_0, \ldots, u_m)$ or $t \twoheadleftarrow l$. Such formulas only mention variables or names, so $\lambda(x)$ can be assumed to be a name for $\lambda \in \{\sigma, \theta_i, \mu_i \mid i \leqslant n\}$ and any variable x occurring in γ. Hence we can ignore such formulas too for the rest of the section, since these formulas do not undergo any change.

Hence we simplify the presentation for the rest of this section by only considering equality assertions γ.

We now have, for every $i \leqslant n$, substitutions μ_i and θ_i, each with domain $\mathsf{bv}(\beta_i)$ and $\mathsf{bv}(\alpha_i)$. However, these substitutions do not necessarily map variables to ground terms. It is possible that $\theta_j(\alpha_j)$ has as a subterm a variable from the domain of some "earlier" μ_i, i.e. one where $i < j$.

If $(T; E) \vdash x \bowtie y$, then x and y ought to actually stand for the same ground term. To capture this, we need a "compound" substitution that maps each variable in the domain of each μ and each θ to a ground term. We now present a motivating example which is followed by the formal definition of this ground substitution.

▶ **Example 17.** Suppose $y \in \mathsf{bv}(\beta_1)$, and $x \in \mathsf{bv}(\alpha_2)$. Consider a situation where $\theta_2(x) = \{y\}_k$ and $\mu_1(y) = (m_0, m_1)$, where $m_0, m_1 \in \mathcal{N}$. Also suppose $(T_2; E_2) \vdash x \bowtie z$ for some $z \in \mathsf{dom}(\sigma)$. We need a λ which maps x and z to the same ground term, i.e. λ needs to be s.t. $\lambda(x) = \lambda(z)$. We can take λ to be $\sigma\mu_1\theta_2$. We see that $\lambda(x) = \sigma(\mu_1(\theta_2(x))) = \sigma(\mu_1(\{y\}_k)) = \sigma(\{(m_0, m_1)\}_k) = \{(m_0, m_1)\}_k$. Observe that $\mathsf{dom}(\lambda) = \mathsf{dom}(\sigma) \cup \mathsf{dom}(\mu_1) \cup \mathsf{dom}(\theta_2)$, and since $z \notin \mathsf{dom}(\mu_1) \cup \mathsf{dom}(\theta_2)$, $\lambda(z) = \sigma(z)$.

▶ **Definition 18.** *The compound substitution which maps any variable in* $\mathsf{dom}(\sigma) \cup \{\mathsf{dom}(\mu_i) \cup \mathsf{dom}(\theta_i) \mid 1 \leqslant i \leqslant n\}$ *to a ground term is given by* $\omega := \sigma\mu_1\theta_1 \ldots \mu_n\theta_n$.

Note that for $\lambda \in \{\sigma, \theta_i, \mu_i \mid i \leqslant n\}$, $\omega(\lambda(x)) = \omega(x)$.

▶ **Lemma 19.** *Suppose* λ *is such that* $\lambda(r) = \lambda(s)$ *for each* $r \bowtie s \in \mathrm{E}$, *and* $\mathrm{T}; \mathrm{E} \vdash_{eq} t \bowtie u$. *Then* $\lambda(t) = \lambda(u)$.

Proof. Suppose $\mathrm{T}; \mathrm{E} \vdash t \bowtie u$ via a proof π with last rule r. The proof is by induction on the structure of π. The following cases arise.

- r = ax: In this case, $t \bowtie u \in \mathrm{E}$, so by assumption, $\lambda(t) = \lambda(u)$.
- r = eq: In this case $t = u$, so $\lambda(t) = \lambda(u)$ as well.
- r = trans: Suppose $t_0 \bowtie t_1, \ldots, t_{n-1} \bowtie t_n$ are the premises of r, with $t = t_0$ and $u = t_n$. By IH, $\lambda(t_{i-1}) = \lambda(t_i)$ for all $i \leqslant n$. It follows that $\lambda(t) = \lambda(u)$.
- r = cons: Let $t = \mathsf{f}(t_1, \ldots, t_n)$ and $u = \mathsf{f}(u_1, \ldots, u_n)$ and let $t_1 \bowtie u_1, \ldots, t_n \bowtie u_n$ be the premises of r. By IH, $\lambda(t_i) = \lambda(u_i)$ for all $i \leqslant n$. Thus we have the following:

$$\lambda(t) = \lambda(\mathsf{f}(t_1, \ldots, t_n)) = \mathsf{f}(\lambda(t_1), \ldots, \lambda(t_n)) = \mathsf{f}(\lambda(u_1), \ldots, \lambda(u_n)) = \lambda(\mathsf{f}(t_1, \ldots, t_n)) = \lambda(u).$$

- r = proj: Let $\mathsf{f}(t_1, \ldots, t_n) \bowtie \mathsf{f}(u_1, \ldots, u_n)$ be the premise of the last rule with $t = t_i$ and $u = u_i$ respectively. By IH, $\lambda(\mathsf{f}(t_1, \ldots, t_n)) = \lambda(\mathsf{f}(u_1, \ldots, u_n))$. So, $\lambda(t) = \lambda(u)$. ⊣

▶ **Lemma 20.** *For any* $i \in \{1, \ldots, n\}$,

1. *if* $t \bowtie u \in \mathrm{E}_i \cup \mathrm{F}_i$, *then* $\omega(t) = \omega(u)$.
2. *if* $\sigma(\mathrm{T}_{i-1}; \mathrm{E}_{i-1}) \vdash_{eq} \sigma\mu_i(t \bowtie u)$, *then* $\omega(t) = \omega(u)$.
3. *if* $\sigma(\mathrm{U}_i; \mathrm{F}_i) \vdash_{eq} \sigma\theta_i(t \bowtie u)$, *then* $\omega(t) = \omega(u)$.

Proof. In addition to $\mathrm{E}_i, \mathrm{F}_i$ for $0 < i \leqslant n$, we also use $\mathrm{E}_0 = \emptyset$, for which claim 1 is vacuously true. We prove the claims simultaneously by induction on $i > 0$. Assume that they hold for all $j < i$ via IH1, IH2, and IH3.

1. Suppose $t \bowtie u \in \mathrm{E}_i$. Then, $\exists j < i : t \bowtie u \in \mathsf{sf}(\alpha_j)$, and $\sigma(\mathrm{U}_j; \mathrm{F}_j) \vdash_{eq} \sigma\theta_j(t \bowtie u)$. By IH3, $\omega(t) = \omega(u)$. If $t \bowtie u \in \mathrm{F}_i$, then $\exists j \leqslant i : t \bowtie u \in \mathsf{sf}(\beta_j)$, and $\sigma(\mathrm{T}_{j-1}; \mathrm{E}_{j-1}) \vdash_{eq} \sigma\mu_j(t \bowtie u)$. If $j < i$, by IH2, $\omega(t) = \omega(u)$. If $j = i$, by IH1, $\omega(r) = \omega(s)$ for every $r \bowtie s \in \mathrm{E}_{i-1}$. Any $a \bowtie b \in \sigma(\mathrm{E}_{i-1})$ is of the form $\sigma(r \bowtie s)$ for some $r \bowtie s \in \mathrm{E}_{i-1}$. Thus, $\omega(a) = \omega(\sigma(r)) = \omega(r) = \omega(s) = \omega(\sigma(s)) = \omega(b)$. By Lemma 19, $\omega(\sigma\mu_j(t)) = \omega(\sigma\mu_j(u))$, i.e. $\omega(t) = \omega(u)$.
2. Suppose $\sigma(\mathrm{T}_{i-1}; \mathrm{E}_{i-1}) \vdash_{eq} \sigma\mu_i(t \bowtie u)$. As above, for each $a \bowtie b \in \sigma(\mathrm{E}_{i-1})$, $\omega(a) = \omega(b)$. By appealing to Lemma 19, we get $\omega(\sigma\mu_i(t)) = \omega(\sigma\mu_i(u))$, i.e. $\omega(t) = \omega(u)$.
3. The proof is similar to the above. ⊣

We developed this preliminary setup for both honest agent derivations as well as intruder derivations in order to demonstrate the interplay between θ and μ, as evidenced in the definition of ω. However, the insecurity problem itself is concerned only with intruder derivability, and therefore, in the next few sections we will focus only on β_i, $(\mathrm{T}_i; \mathrm{E}_i)$, and μ_i. We will discuss honest agent derivations later.

4.1 Typed proofs for $\vdash_{dy}$ and $\vdash_{eq}$

In order to obtain "small" versions of the various substitutions σ, θ_i, and μ_i while preserving their interaction, we consider a universe of "anchor terms". These are abstract terms that appear in the protocol specification, and for which we have a bound on size. We call these anchors "types". We would eventually like to be able to convert any proof into one that only involves typed terms, i.e. terms that correspond to one of these types under ω.

▶ **Definition 21** (Types and typed terms). *We use the sets* C *(consisting of the terms occurring in* ξ *before applying any substitution) and* D *(the same set, but without variables) to type the terms appearing in any proof.*

$$\mathsf{C} := \bigcup_{i \leqslant n} \{(\mathrm{st}(T_i \cup U_i) \cup \mathrm{st}(E_i \cup F_i))\} \qquad \mathsf{D} := \mathsf{C} \setminus \mathsf{V}$$

A term t *is typed if* $t \in \sigma(\mathsf{D}) \cup \omega(\mathsf{C}) \cup \mathsf{V}_q$.

Note that we must consider $\sigma(\mathsf{D})$ separately from $\omega(\mathsf{C})$. Consider a term of the form $(m, x) \in \mathsf{D}$, where $x \notin \mathrm{dom}(\sigma)$. $\sigma((m, x)) = (m, x)$, but this cannot be in $\omega(\mathsf{C})$, since $\omega(\mathsf{C})$ only contains ground terms. Thus, $\sigma(\mathsf{D}) \not\subseteq \omega(\mathsf{C})$. Here is a useful observation about typed terms.

▶ **Observation 22.** *Every term in* $\sigma(\mathsf{C})$ *is typed.*

Proof. For any $a \in \sigma(\mathsf{C})$, one of the following three cases holds.

$a \in \sigma(\mathsf{D})$: Then a is typed, by definition.
$a = \sigma(x) = x$ for some $x \notin \mathrm{dom}(\sigma)$: Then $a \in \mathsf{V}_q$ and hence typed.
$a = \sigma(x)$ for $x \in \mathrm{dom}(\sigma)$: Then it is also the case that $a = \omega(x) \in \omega(\mathsf{C})$, so a is typed. ⊣

We now define a notion of "zappable terms", which are terms that do not correspond to any type in C. The idea is these terms can be freely "zapped".[6]

▶ **Definition 23** (Zappable terms). *A term* t *is zappable if there is an* $x \in \mathrm{dom}(\omega)$ *such that* $\omega(t) = \omega(x)$, *but there is no* $u \in \mathsf{D}$ *such that* $\omega(x) = \omega(u)$. *We refer to such an* x *as a minimal variable.*

Here are a couple of easy observations that relate to zappable terms.

▶ **Observation 24.**

- *If a term* t *is zappable, then* $t \notin \mathsf{D}$.
- *If a term* $t \in \omega(\mathsf{C})$ *is not zappable, then* $t \in \omega(\mathsf{D})$.
- *For* t, u *s.t.* $\omega(t) = \omega(u)$, t *is zappable iff* u *is zappable.*

▶ **Lemma 25.** *Suppose* $t = \mathsf{f}(t_1, \dots, t_n)$ *and* $u = \mathsf{f}(u_1, \dots, u_n)$ *are typed, and* $\omega(t) = \omega(u)$. *One of the following is true:*

- t *and* u *are not zappable, and* $t_1, \dots, t_n, u_1, \dots, u_n$ *are typed, or*
- t *and* u *are zappable, and* $t = u$.

Proof. Since t and u are non-atomic, $t, u \notin \mathsf{V}_q$. But they are typed, so $t, u \in \sigma(\mathsf{D}) \cup \omega(\mathsf{C})$. We consider two cases:

- **Neither t nor u is zappable:** Consider t. If $t \in \sigma(\mathsf{D})$, each $t_i \in \sigma(\mathsf{C})$, and hence typed (by Observation 22). If $t \in \omega(\mathsf{C})$, then since t is not zappable, $t = \omega(a)$ for some $a \in \mathsf{D}$. Then a has to be of the form $\mathsf{f}(a_1, \dots, a_n)$, with each $a_i \in \mathsf{C}$ and $t_i = \omega(a_i) \in \omega(\mathsf{C})$. Thus each t_i is typed. Reasoning about u in a similar manner, we see that each u_i is typed as well.

[6] In order to motivate the key ideas behind typing, we will often use the word "zap" to mean replacing terms by an atomic name. However, we will formally define this zapping operation in the next subsection.

- **One of t and u is zappable:** Say t is zappable. Then, since $\omega(t) = \omega(u)$, u is zappable as well. Therefore $t, u \notin \sigma(\mathsf{D})$, which implies that $t, u \in \omega(\mathsf{C})$. Therefore both t and u are ground terms, so $t = \omega(t) = \omega(u) = u$. ⊣

We now devise notions of "typed proofs" for the $\vdash_{dy}$ as well as the $\vdash_{eq}$ system, which will help us obtain bounds on the sizes of terms appearing in the ranges of various substitutions. Then, we show that every proof in these systems can be converted into a typed proof.

Consider a proof π witnessing $\sigma(\mathsf{T}_i) \vdash_{dy} t$ for some t. Any term in T_i, since T_i is part of a kernel, is either a bound variable outside the domain of σ (i.e. in V_q) or a public term of some assertion. Note that any variables in public terms of assertions must not be quantified, hence they fall into the domain of σ. Thus, any such t derived from $\sigma(\mathsf{T}_i)$ is either in V_q, or a ground term of the form $\sigma(v)$ for some v.

Now, it is possible that π mentions some term $u \notin \sigma(\mathsf{D})$, even if $t \in \sigma(\mathsf{D})$. If a destructor rule is applied to u in order to obtain a proof of t, we cannot "zap" u into an atomic name while still preserving derivability. This leads us to the following definition of a typed proof in the $\vdash_{dy}$ system, which preserves derivability even after zapping variables as necessary.

▶ **Definition 26.** *[Typed $\vdash_{dy}$ proof] A $\vdash_{dy}$ proof π is typed if for each subproof π', either π' ends in a constructor rule, or $\mathsf{conc}(\pi') \in \sigma(\mathsf{D}) \cup \mathsf{V}_q$, where $\mathsf{conc}(\pi')$ denotes the conclusion derived using π'.*

Armed with this definition of a typed $\vdash_{dy}$ proof, we can show that any proof $\sigma(\mathsf{T}_i) \vdash_{dy} t$ can be transformed into a typed normal equivalent witnessing the same. This transformation crucially uses the following fact about how non-typed terms are generated: any non-typed term u occurring in a $\vdash_{dy}$ proof from $\sigma(\mathsf{T}_i)$ obeys the following:

- appears first as part of a received assertion $\sigma(\beta)$, and
- is generated by the intruder by putting information together, i.e. via a normal proof ending in a constructor.

The intuition behind this is easy to see – honest agents follow the protocol, and will only communicate terms that follow the protocol specification, modulo any insertions by the intruder. Terms that correspond to ones in the protocol specification are always typed, so any non-typed term must have been initially sent out by the intruder, i.e. in a β received by an honest agent. In particular, such a term must have been constructed by the intruder by putting information together, since up till that point, the intruder's knowledge state would have only consisted of typed terms, and destructor rules would preserve "typability". Thus, for any non-typed term t such that $t \in \mathsf{st}(\sigma(\mathsf{T}_i))$, we can always "chase back" to an index $j < i$ at which it was *not* in the subterms of $\sigma(\mathsf{T}_j)$, but still derivable, i.e. $\sigma(\mathsf{T}_j) \vdash_{dy} t$ via a normal proof ending in a constructor rule. This reasoning closely follows the ideas in [38], and is formalized below.

We define $IT_i \coloneqq \mathsf{pubs}(\beta_i)$ and $HT_i \coloneqq \mathsf{pubs}(\alpha_i)$.[7]

▶ **Lemma 27.** *Suppose $t \notin \sigma(\mathsf{D}) \cup \mathsf{V}_q$. If $t \in \mathsf{st}(\sigma(\mathsf{T}_i))$ for some $i \leqslant n$, there is a $k < i$ such that $t \in \mathsf{st}(\sigma(IT_k))$.*

[7] These stand for intruder terms and honest agent terms respectively.

Proof. Suppose $t \in \mathrm{st}(\sigma(u)) \setminus (\sigma(D) \cup V_q)$ for some $u \in T_i$. Then, $t \in \mathrm{st}(\sigma(y))$ for some $y \in \mathrm{vars}(u)$. Since $u \in T_i$, there is a $j < i$ such that $u \in HT_j \cup V_q$. If $u \in V_q$, then $u = y = \sigma(y)$ and $t = y$, but we know that $t \notin V_q$. Thus $u \notin V_q$ and $u \in HT_j$, i.e. $y \in \mathrm{vars}(HT_j)$. Now ξ is an interleaving of sessions of Pr, and $y \in \mathrm{vars}(u)$ where u occurs in an honest agent send in a session. Thus there is an earlier intruder send in the same session in which y occurs. This send occurs before α_j in ξ. Thus there is a $k \leqslant j$ such that $y \in \mathrm{vars}(\mathrm{pubs}(\beta_k)) = \mathrm{vars}(IT_k)$. Thus, $t \in \mathrm{st}(\sigma(IT_k))$. ⊣

▶ **Lemma 28.** *Suppose $i \leqslant n$, $t \notin \sigma(D) \cup V_q$ and $\sigma(T_i) \vdash_{dy} t$ via a normal proof π ending in a destructor rule. Then there is an $l < i$ such that $\sigma(T_l) \vdash_{dy} t$.*

Proof. Since π ends in a destructor rule, $t \in \mathrm{st}(\sigma(T_i))$. By Lemma 27, there is an $i' < i$ such that $t \in \mathrm{st}(\sigma(IT_{i'}))$. Let j be the earliest such index, and let $a \in IT_j$ such that $t \in \mathrm{st}(\sigma(a))$. Since $\sigma(T_{j-1}; E_{j-1}) \vdash_a \sigma\mu_j(\beta_j)$, and $a \in IT_j = \mathrm{pubs}(\beta_j)$, it follows by Lemma 15 that $\sigma(T_{j-1}) \vdash_{dy} \sigma\mu_j(a)$. But $\mathrm{vars}(a) \cap \mathrm{dom}(\mu_j) = \emptyset$, so $\sigma(T_{j-1}) \vdash_{dy} \sigma(a)$, via a normal proof ρ. Consider a minimal subproof χ of ρ such that $t \in \mathrm{st}(\mathrm{conc}(\chi))$. (There is at least one such subproof, namely ρ.) If χ ends in a destructor, then $\mathrm{conc}(\chi) \in \mathrm{st}(\sigma(T_{j-1}))$, and hence $t \in \mathrm{st}(\sigma(T_{j-1}))$. But by Lemma 27, there must be a $k < j - 1$ such that $t \in \mathrm{st}(\sigma(IT_k))$, contradicting the fact that j is the earliest such index. So χ ends in a constructor rule. If $t \neq \mathrm{conc}(\chi)$, then $t \in \mathrm{st}(\mathrm{conc}(\chi'))$, for some proper subproof of χ. But this cannot be, since χ is a minimal proof with this property. Thus, $t = \mathrm{conc}(\chi)$ and χ is a proof of $\sigma(T_{j-1}) \vdash t$ (and we choose our l to be $j - 1$). ⊣

▶ **Theorem 29.** *For all t and all $i \in \{0, \ldots, n\}$, if $\sigma(T_i) \vdash_{dy} t$, then there is a typed normal proof π^* of the same.*

Proof. Assume the theorem holds for all $j < i$. We show how to transform any proof π of $\sigma(T_i) \vdash t$ ending in rule r into a typed normal proof π^* of the same by induction on the structure of π.

r is ax: $t \in \sigma(T_i) \subseteq \sigma(C)$. If $t \in \sigma(D) \cup V_q$, we take π^* to be π itself. Otherwise, there is a $j < i$ such that $\sigma(T_j) \vdash_{dy} t$. We can get a typed normal proof π^* of $\sigma(T_j) \vdash t$ and obtain the required result by weakening the LHS.

r is a constructor: We can find typed normal equivalents for all immediate subproofs, and apply the same constructor rule to get the desired π^*.

r is a destructor: Let $\pi_1, \ldots, \pi_n$ be immediate subproofs of π, with $\mathrm{conc}(\pi_1) = s$ being the major premise (and t being an immediate subterm of s, as a consequence). We can find typed normal equivalents $\pi_1^*, \ldots, \pi_n^*$. If π_1^* ends in a constructor, then we choose π^* to be the immediate subproof of π_1^* s.t. $\mathrm{conc}(\pi^*) = t$.

If π_1^* does not end in a constructor, $s \in \sigma(D) \cup V_q$. Since a destructor rule r was applied on s, $s \notin V_q$. So $s \in \sigma(D)$, and hence $t \in \sigma(C)$. If $t \in \sigma(D) \cup V_q$, we obtain a typed normal π^* by applying r on the π_i^*s. Otherwise, as with ax, we get a typed and normal proof π^* of $\sigma(T_j) \vdash t$ for some $j < i$ and apply weakening. ⊣

Having shown that we can always obtain a typed $\vdash_{dy}$ proof, we now consider $\vdash_{eq}$. We present below an example which will motivate our choices for the definition of a *typed $\vdash_{eq}$ proof*.

Suppose $\sigma(x) = (t_1, t_2)$ for some minimal x, and $\sigma(u) = (u_1, u_2)$ for some term u. Suppose we also have a proof of $t_1 \bowtie u_1$ obtained by applying proj_1 to a proof of $\sigma(x) \bowtie \sigma(u)$, and we want a "corresponding" proof, even after zapping. However, x would be zapped to a name, and we cannot apply proj to an atomic value. We would prefer a proof which allows us to preserve its structure even after zapping. To this end, we define a typed $\vdash_{eq}$ proof as follows.

▶ **Definition 30.** *[Typed $\vdash_{eq}$ proof] A proof π of $X; A \vdash r \bowtie s$ is typed if for every subproof π' with conclusion $X; A \vdash t \bowtie u$,*

- *π' contains an occurrence of the* cons *rule, or*
- *$t = u$, or*
- *t and u are typed terms.*

Intuitively, this definition disallows "asymmetric" zapping of the above kind, and allows us to prove the equivalent of Theorem 29 for $\vdash_{eq}$ proofs.

▶ **Theorem 31.** *For $i \leqslant n$ and $a, b \in \mathsf{T}$, if $\sigma(\mathsf{T}_i; \mathsf{E}_i) \vdash_{eq} a \bowtie b$, there is a typed normal proof of $\sigma(\mathsf{T}_i; \mathsf{E}_i) \vdash a \bowtie b$.*

Proof. By Theorem 10, we know that every $\vdash_{eq}$ proof can be converted to an equivalent normal proof. We can show that every normal $\vdash_{eq}$ proof is typed. The only non-trivial case is when the last rule is proj. Consider a normal proof π of $\sigma(\mathsf{T}_i; \mathsf{E}_i) \vdash a \bowtie b$, whose last rule is proj, and whose immediate (typed normal, by IH) subproof is π' deriving $\mathsf{f}(a_1, \ldots, a \ldots, a_n) \bowtie \mathsf{f}(b_1, \ldots, b, \ldots, b_n)$. Since π is a normal proof ending in proj, the cons rule does not occur in π or π'. Two cases arise:

- $\mathsf{f}(a_1, \ldots, a, \ldots, a_n) = \mathsf{f}(b_1, \ldots, b \ldots, b_n)$, in which case $a = b$ and π is typed.
- $\mathsf{f}(a_1, \ldots, a, \ldots, a_n)$ and $\mathsf{f}(b_1, \ldots, b, \ldots, b_n)$ are both typed terms. By Lemma 25, we see that either $\mathsf{f}(a_1, \ldots, a, \ldots, a_n) = \mathsf{f}(b_1, \ldots, b, \ldots, b_n)$ (whence $a = b$), or a and b are typed, and thus π is typed. ⊣

4.2 Small substitutions σ^*, ω^*, and μ_i^*

Assume that there is an $\mathsf{m} \in \mathsf{T}_0 \cap \mathsf{N}$ s.t. $\mathsf{m} \notin \mathsf{st}(\{\alpha_i, \beta_i\}) \cup \mathsf{st}(\mathsf{rng}(\theta_i) \cup \mathsf{rng}(\mu_i))$ for all i. This can be thought of as a fixed "spare name" that does not appear in the run. We will use this name to formally define a zap operation, as below.

▶ **Definition 32.** *For any term t, we inductively define the zap of t, denoted $\overline{t}$, as follows:*

$$\overline{x} \coloneqq x$$

$$\overline{n} \coloneqq \begin{cases} \mathsf{m} & \text{if } n \text{ is zappable} \\ n & \text{otherwise} \end{cases}$$

$$\overline{\mathsf{f}(t_1, \ldots, t_n)} \coloneqq \begin{cases} \mathsf{m} & \text{if } \mathsf{f}(t_1, \ldots, t_n) \text{ is zappable} \\ \mathsf{f}(\overline{t_1}, \ldots, \overline{t_n}) & \text{otherwise} \end{cases}$$

For a set of terms X, $\overline{X} \coloneqq \{\overline{t} \mid t \in X\}$. For a set of equalities E, $\overline{\mathsf{E}} \coloneqq \{\overline{t} \bowtie \overline{u} \mid t \bowtie u \in \mathsf{E}\}$.

▶ **Definition 33.** *For $\lambda \in \{\sigma, \omega, \mu_i \mid i \leqslant n\}$, the small substitution λ^* is the substitution with $\mathsf{dom}(\lambda^*) = \mathsf{dom}(\lambda)$ and $\lambda^*(x) \coloneqq \overline{\lambda(x)}$ for all $x \in \mathsf{dom}(\lambda)$.*

Here are a few examples that illustrate the above definition, for different choices of λ and C.

▶ **Example 34.**

1. Suppose $\mathsf{C} = \mathsf{st}(\{\mathsf{m}, y, (y_1, \{y_2\}_k)\})$, where y_1, y_2 are minimal, and $\mu_2(y) = (y_1, \{y_2\}_k)$. Then $\mu_2^*(y) = (y_1, \{y_2\}_k)$ and $\omega^*(y) = (\mathsf{m}, \{\mathsf{m}\}_k)$.

2. Suppose $C = \mathrm{st}(\{m, y, y_2, (y_1, x)\})$ and μ_2 is the same as above, with x minimal and $\sigma(x) = \mu_2(\{y_2\}_k)$. Then $\mu_2^*(y) = (y_1, m)$ and $\omega^*(y) = (m, m)$.

Following Definition 33, we can see that $\sigma^*\mu_i^*(x) = \overline{\sigma\mu_i(x)}$ for any $i \leqslant n$ and $x \in V$, but this equality need not lift to bigger terms in general. Consider a minimal $x \in \mathrm{dom}(\sigma)$ with $\sigma(x) = t$. So t is ground, and hence $\mathrm{vars}(t) = \emptyset$. So $\sigma^*\mu_i^*(t) = t$. However, $\overline{\sigma\mu_i(t)} = \overline{t} = m$, since t is zappable. Thus, it is not true that $\sigma^*\mu_i^*(t) = \overline{\sigma\mu_i(t)}$ for all possible terms t. However, we can show that this holds for all $t \in C$.

This requires an analysis of $\sigma\mu_i$. It is clear that $\mathrm{dom}(\sigma) \cap \mathrm{dom}(\mu_i) = \emptyset$. Further, for any $x \in \mathrm{dom}(\mu_i)$, we have $\sigma(T_{i-1}) \vdash_{dy} \mu_i(x)$. Hence $\mathrm{vars}(\mu_i(x)) \subseteq \mathrm{vars}(\sigma(T_{i-1})) \subseteq V_q$, and so we have that $\mathrm{vars}(\mathrm{rng}(\mu_i(x))) \cap \mathrm{dom}(\sigma) = \emptyset$. Similarly $\mathrm{vars}(\mathrm{rng}(\sigma(x))) \cap \mathrm{dom}(\mu_i) = \emptyset$ (since $\sigma(x)$ is ground for any $x \in \mathrm{dom}(\sigma)$). Thus $\sigma\mu_i = \sigma \cup \mu_i$. In other words, for any $x \in \mathrm{dom}(\sigma)$, $\sigma\mu_i(x) = \sigma(x)$ and for any $x \in \mathrm{dom}(\mu_i)$, $\sigma\mu_i(x) = \mu_i(x)$. This property continues to hold for σ^* and μ_i^*.

▶ **Lemma 35.** *For $i \leqslant n$ and $t \in C$, $\sigma^*\mu_i^*(t) = \overline{\sigma\mu_i(t)}$.*

Proof. Consider $\lambda = \mu_i$ for some i. The following cases arise.

- $t = x \in \mathrm{dom}(\sigma\lambda)$: By Definition 33 and the remarks preceding this lemma, if $x \in \mathrm{dom}(\sigma)$, then $\sigma^*\lambda^*(x) = \sigma^*(x) = \overline{\sigma(x)} = \overline{\sigma\lambda(x)}$, and a similar reasoning applies when $x \in \mathrm{dom}(\lambda)$.
- $t = x \notin \mathrm{dom}(\sigma\lambda)$: $\sigma^*\lambda^*(x) = x = \overline{x} = \overline{\sigma\lambda(x)}$.
- $t \in N$: Since $t \in C$, $t \in D$ and hence is not zappable. Thus $\sigma\lambda(t) = t = \sigma^*\lambda^*(t)$. Since $\overline{t} = t$, $\sigma^*\lambda^*(t) = \overline{\sigma\lambda(t)}$.
- $t = f(t_1, \dots, t_n)$: $t \in D$, so $t_1, \dots, t_n \in C$, and for $j \leqslant n$ we get $\sigma^*\lambda^*(t_j) = \overline{\sigma\lambda(t_j)}$ by IH. We claim that $u = \sigma\lambda(t)$ is not zappable, since for any x such that $\omega(u) = \omega(x)$, x is not minimal (since $\omega(x) = \omega(t)$ as well, and $t \in D$). Therefore, we have

$$\overline{\sigma\lambda(t)} = \overline{f(\sigma\lambda(t_1), \dots, \sigma\lambda(t_n))} = f(\overline{\sigma\lambda(t_1)}, \dots, \overline{\sigma\lambda(t_n)}) = f(\sigma^*\lambda^*(t_1), \dots, \sigma^*\lambda^*(t_n)) = \sigma^*\lambda^*(t). \dashv$$

We now show, via Lemmas 36 and 37, that small substitutions preserve derivabilities of both terms and equalities.

▶ **Lemma 36.** *For $i \leqslant n$ and any term t, if $\sigma(T_i) \vdash_{dy} t$ then $\sigma^*(T_i) \vdash_{dy} \overline{t}$.*

Proof. Let X and Y stand for $\sigma(T_i)$ and $\sigma^*(T_i)$. Since $X \subseteq C$, by Lemma 35, $\overline{X} = Y$. Let π be a typed normal $\vdash_{dy}$ proof of $X \vdash t$ (ensured by Theorem 29). We prove that $Y \vdash_{dy} \overline{t}$. Consider the last rule r of π. The following cases arise.

- r = ax: $t \in X$, and therefore $\overline{t} \in Y$. Thus $Y \vdash_{dy} \overline{t}$ by ax.
- r is a constructor: Let $t = f(t_1, \dots, t_n)$ and let $\pi_1, \dots, \pi_n$, with $\mathrm{conc}(\pi_i) = t_i$, be the immediate subproofs of π. By IH, there is a proof ϖ_i of $Y \vdash \overline{t_i}$ for each $i \leqslant n$. If t is zappable, then $\overline{t} = m \in Y$ ($m \in T_i$ for all i, so $m \in X$ and $m \in Y$), and we have $Y \vdash_{dy} \overline{t}$ using ax. If t is not zappable, then $\overline{t} = \overline{f(t_1, \dots, t_n)} = f(\overline{t_1}, \dots, \overline{t_n})$, and we can apply r on the ϖ_is to get $Y \vdash_{dy} \overline{t}$.
- r is a destructor: Let the immediate subproofs of π be $\pi_1, \dots, \pi_n$, deriving $t_1, \dots, t_n$ respectively, with t_1 being the major premise, and t an immediate subterm of t_1. Since π is typed normal, π_1 is also typed and ends in a destructor, so by Definition 26, $t_1 \in \sigma(D) \cup V_q$. Since we applied a destructor on t_1, it is not in V_q. Thus, there is some $u_1 \in D$, with the same outermost operator as t_1, such that $t_1 = \sigma(u_1)$. Hence, $\omega(t_1) = \omega(u_1)$.

If t_1 were zappable, there would be a minimal x such that $\omega(x) = \omega(t_1) = \omega(u_1) \in \omega(\mathsf{D})$, which contradicts the minimality of x. Thus, t_1 is not zappable, and $\overline{t_1}$ has the same outermost structure as t_1. By IH, there is a proof ϖ_i of $Y \vdash \overline{t_i}$ for each $i \leqslant n$. Since $\overline{t_1}$ is not atomic, we can apply the destructor r on the ϖ_is to get $Y \vdash_{dy} \overline{t}$. ⊣

▶ **Lemma 37.** *For $i \leqslant n$ and terms t, u, if $\sigma(\mathsf{T}_i; \mathsf{E}_i) \vdash_{eq} t \bowtie u$ then $\sigma^*(\mathsf{T}_i; \mathsf{E}_i) \vdash_{eq} \overline{t} \bowtie \overline{u}$.*

Proof. Let $(X; A)$ and $(Y; B)$ denote $\sigma(\mathsf{T}_i; \mathsf{E}_i)$ and $\sigma^*(\mathsf{T}_i; \mathsf{E}_i)$ respectively. As earlier, using Lemma 35, $\overline{X} = Y$ and $\overline{A} = B$. Let π be a typed normal $\vdash_{eq}$ proof of $X; A \vdash t \bowtie u$ (guaranteed by Theorem 31). We prove that $Y; B \vdash_{eq} \overline{t} \bowtie \overline{u}$. Most of the cases are straightforward, so here we only consider the cases when π ends in proj or cons.

- π ends in proj: Let π' be the immediate subproof of π, proving $X; A \vdash a \bowtie b$ with $a = \mathsf{f}(a_1, \dots, a_n)$, $b = \mathsf{f}(b_1, \dots, b_n)$, and $t = a_i$ and $u = b_i$. By IH, there is a proof ϖ' of $Y; B \vdash \overline{a} \bowtie \overline{b}$. For proj, we need $X \vdash_{dy} \{a_1, \dots, a_n, b_1, \dots, b_n\}$. By Lemma 36, $Y \vdash_{dy} \{\overline{a_1}, \dots, \overline{a_n}, \overline{b_1}, \dots, \overline{b_n}\}$. By Lemma 20, $\omega(a) = \omega(b)$. By normality, cons cannot occur in π. π is also typed, so either $a = b$ or a and b are typed. If $a = b$, then $t = u$, and we have a proof of $Y; B \vdash \overline{t} \bowtie \overline{u}$ ending in eq. If a and b are typed, we apply Lemma 25 and the following two cases arise.
  - a and b not zappable: Then $\overline{a}$ and $\overline{b}$ have the same outermost structure as a and b, and $\overline{t} = \overline{a_i}$ and $\overline{u} = \overline{b_i}$. So we can apply proj on ϖ' to get $Y; B \vdash_{eq} \overline{t} \bowtie \overline{u}$.
  - $a = b$: Then $t = u$ as well, and hence $\overline{t} = \overline{u}$. Since $Y \vdash_{dy} \{\overline{t}, \overline{u}\}$, $Y; B \vdash_{eq} \overline{t} \bowtie \overline{u}$ with last rule eq.
- π ends in cons: Let $t = \mathsf{f}(t_1, \dots, t_n)$ and $u = \mathsf{f}(u_1, \dots, u_n)$. Let π have immediate subproofs $\pi_1, \dots, \pi_n$, each π_i proving $X; A \vdash t_i \bowtie u_i$. By IH, there are proofs $\varpi_1, \dots, \varpi_n$, each ϖ_i proving $Y; B \vdash \overline{t_i} \bowtie \overline{u_i}$. By Lemma 25, two cases arise.
  - t and u not zappable: Then $\overline{t} = \mathsf{f}(\overline{t_1}, \dots, \overline{t_n})$ and $\overline{u} = \mathsf{f}(\overline{u_1}, \dots, \overline{u_n})$. So $Y; B \vdash_{eq} \overline{t} \bowtie \overline{u}$ using cons on the ϖ_is.
  - t and u zappable: Then, $\overline{t} = \overline{u} = \mathsf{m} \in Y$, so we have a proof of $Y; B \vdash \overline{t} \bowtie \overline{u}$ ending in eq. ⊣

Putting Lemmas 35, 36 and 37 together, we get:

▶ **Theorem 38.** *Let $t, u \in \mathsf{C}$ and $i \leqslant n$.*

- *If $\sigma(\mathsf{T}_{i-1}) \vdash_{dy} \sigma\mu_i(t)$ then $\sigma^*(\mathsf{T}_{i-1}) \vdash_{dy} \sigma^*\mu_i^*(t)$.*
- *If $\sigma(\mathsf{T}_{i-1}; \mathsf{E}_{i-1}) \vdash_{eq} \sigma\mu_i(t \bowtie u)$ then $\sigma^*(\mathsf{T}_{i-1}; \mathsf{E}_{i-1}) \vdash_{eq} \sigma^*\mu_i^*(t \bowtie u)$.*

Having shown that the λ^*s simulate the λs, we next show that they allow us a bound on the size of terms therein.

▶ **Theorem 39.** *For $\lambda \in \{\sigma, \omega, \mu_i \mid i \leqslant n\}$, and $x \in \mathsf{dom}(\lambda)$, $|\mathsf{st}(\lambda^*(x))| \leqslant |\mathsf{D}|$.*

Proof. For each λ and any x, $\omega^*(\lambda^*(x)) = \omega^*(x) = \overline{\omega(x)}$ (by Definition 33) and thus, $|\mathsf{st}(\lambda^*(x))| \leqslant |\mathsf{st}(\omega^*(x))|$. So it suffices to prove a bound on $|\mathsf{st}(\omega^*(x))|$. We show that for $t \in \mathsf{C}$, $\mathsf{st}(\omega^*(t)) \subseteq \omega^*(\mathsf{D})$. Note that if $t = x$ is non-minimal, there is an $r \in \mathsf{D}$ s.t. $\omega^*(t) = \omega^*(r)$. Thus it suffices to prove the statement for t which is either a minimal variable or in D.

The proof is by induction on $|\omega^*(t)|$.

- $|\omega^*(t)| = 1$: $\omega^*(t) \in \mathsf{N}$. So $t \in \mathsf{N}$ or t is a minimal variable. If $t \in \mathsf{N}$, $\omega^*(t) = t \in \mathsf{N}$. Otherwise, $\omega^*(t) = \mathsf{m}$. In both these cases, $\mathsf{st}(\omega^*(t)) \subseteq \omega^*(\mathsf{D})$.
- $|\omega^*(t)| > 1$: Let $a \in \mathsf{st}(\omega^*(t))$. If $a = \omega^*(u)$ for some $u \in \mathsf{st}(t) \setminus \mathsf{vars}(t)$, then $a \in \omega^*(\mathsf{D})$. If $a = \omega^*(x)$ for some minimal $x \in \mathsf{vars}(t)$, then $a = \mathsf{m} = \omega^*(\mathsf{m}) \in \omega^*(\mathsf{D})$. If $a \in \mathsf{st}(\omega^*(x))$ for non-minimal $x \in \mathsf{vars}(t)$, then $x \neq t$, and there is an $r \in \mathsf{D}$ s.t. $\omega^*(x) = \omega^*(r)$, and $a \in \mathsf{st}(\omega^*(r))$. Since $|\omega^*(r)| < |\omega^*(t)|$, by IH, $\mathsf{st}(\omega^*(r)) \subseteq \omega^*(\mathsf{D})$. Thus $a \in \omega^*(\mathsf{D})$.

Hence, $|\mathsf{st}(\omega^*(t))| \leqslant |\omega^*(\mathsf{D})| \leqslant |\mathsf{D}|$, for $t \in \mathsf{C}$. ⊣

4.3 NP algorithm for Insecurity: Sketch

After guessing a coherent set of sessions and an interleaving of these sessions of length n, we guess a small substitution σ^*, for each intruder send β_i a set $X_i \subseteq \mathsf{hat}(\beta_i)$ and a small substitution μ_i^* whose domain is $\mathsf{bv}(\beta_i) \setminus \mathsf{bv}(X_i)$. We also guess a sequence of knowledge functions such that the relevant atomic assertions and terms (communicated in the $\sigma^*(\beta_i)$s) are derivable from $\sigma^*(\mathit{ker}(\mathsf{k}_{i-1}(I)))$. These derivability checks in the $\vdash_{eq}$ system can be carried out in time polynomial in the size of the protocol description (using the procedure described in Algorithm 1).

For honest agent derivations, we only deal with derivations of the form $\mathsf{k}_i(u_i) \vdash_a \alpha_i$ (without the σ). This is, in fact, a version of the *passive intruder problem* for assertions. Applying Theorem 16, we reduce this to checks of the form $(\mathrm{U}_i; \mathrm{F}_i) \vdash_{eq} \theta_i(r \bowtie s)$. It is much simpler to ensure that we can obtain θ_is of bounded size, because of the absence of σ. We can think of this as a version of the *passive intruder problem* for the system with assertions. The following theorem, the proof of which can be found in the Appendix, will help us obtain small θ_is.

▶ **Theorem 40.** *If there is a μ satisfying Theorem 16, there is a "small" ν satisfying the same conditions, such that $|\mathsf{st}(\nu(x))| \leqslant |\mathsf{st}(S) \cup \mathsf{st}(A \cup \{\alpha\})|$ for all $x \in \mathsf{dom}(\nu)$.*

In order to check whether $\mathsf{k}_i(u_i) \vdash_a \alpha_i$, we need to guess $X \subseteq \mathsf{hat}(\alpha_i)$ and a small substitution θ_i such that the conditions of Theorem 16 are satisfied. (The smallness of θ_i is guaranteed by Theorem 40.) Each of those conditions can be checked in polynomial time because they only involve $\vdash_{dy}$ proofs (checkable in PTIME), $\vdash_{eq}$ proofs (also checkable in PTIME), and proofs involving only $\{\mathsf{ax}, \wedge\mathsf{i}, \exists\mathsf{i}, \mathsf{say}\}$ (also checkable in PTIME). Thus, honest agent derivability checks are in NP.

5 Discussion and Future Work

5.1 Intruder theories for terms

For terms, we assumed that every operator had constructor and destructor rules, as specified in Figure 1. Such systems are called *constructor-destructor theories*. While the initial results for the active intruder problem were proved for simple theories in [38], that work has been extended to much richer theories [2, 9, 13–16, 20]. As mentioned in Section 1.4, the extension with assertions that we consider is not subsumed by any known intruder theories.

Can one generalize the results of this paper to richer intruder theories? We believe that one can, but one needs to modify a few fundamental notions used so far. We list these considerations below.

- In the main text, we used $\mathsf{st}(t)$ to mean the *syntactic subterms* of t. For a general intruder theory, we will need to assume a function S which maps finite sets of terms to finite sets, and satisfies $\mathsf{st}(X) \subseteq \mathsf{S}(X)$ for any set X.
- To handle the general case, we modify the form of constructors and destructors as follows. In a constructor rule, each immediate subterm of the conclusion is a subterm of one of the premises. In a destructor rule, the conclusion is a subterm of one of the premises.
- We can assume that the intruder theory we consider is local w.r.t. S. That is, whenever X derives t, we have a proof π of $X \vdash t$ such that $\mathsf{terms}(\pi) \subseteq \mathsf{S}(X \cup \{t\})$, and further, if π ends in a destructor rule, $\mathsf{terms}(\pi) \subseteq \mathsf{S}(X)$.
- We modify Definition 21 to use S instead of st. Definitions 23, 26, 30, 32, and 33, on which the proofs in Section 4 hinge, will stay unchanged, since they only refer to C and D.
- We need to prove Theorem 29 for the extended theory before moving onto the $\vdash_{eq}$ system. Determining the conditions on the intruder theory which would guarantee this theorem is left for future work.
- Now, for proofs in the $\vdash_{eq}$ system, there is the following subtlety, which we illustrate by considering the $\vdash_{eq}$ theory built on top of the theory for XOR as presented in [15]. In this intruder theory, there are implicit rewrites in the rules for XOR. For instance, from $a \oplus b$ and $b \oplus c$, we can obtain $a \oplus c$. We would need to carry over these rewrites into the equality rules as well, and in the presence of such rewrites, show that normalization and subterm property hold for the new $\vdash_{eq}$ system.

  In particular, for normalization, we need to eliminate subproofs where an instance of cons appears as the premise for proj. For the basic $\vdash_{eq}$ system, one can do this by picking the appropriate subproof of cons. However, in this new system with XOR, consider a proof of the following form.

  $$\dfrac{\dfrac{T; E \vdash x \bowtie a \oplus b \qquad T; E \vdash y \bowtie b \oplus c}{T; E \vdash x \oplus y \bowtie a \oplus c}\,\mathsf{cons}}{T; E \vdash x \bowtie a}\,\mathsf{proj}_1$$

  Such a proof cannot easily be normalized, since none of these subproofs has the same conclusion. But such a proj rule should not be allowed to begin with, since implicit rewrites are not injective.[8] Thus, proving normalization and the subterm property for any modified $\vdash_{eq}$ system built on top of a general intruder theory seems feasible, provided one appropriately tailors the rules – especially proj – to avoid any unsound behaviour. This is left for future work.

Thus, we can see that the main change in lifting this result to richer intruder theories lies in showing that Theorem 29 holds. One might also need to restrict the new rules one might introduce to the $\vdash_{eq}$ system, and hence mildly modify the proofs of the normalization theorem and Theorem 31.

[8] In the constructor-destructor theories as in Figure 1, we can see that such implicit rewrites do not occur, and all fs considered are injective.

5.2 Constraint solving approach

An algorithmic approach to the active intruder problem is *constraint solving* [16, 34]. Rather than merely proving a bound on the substitution size, these papers present the problem as a series of deducibility constraints (involving variables), the solution to which is a substitution under which all the deducibilities actually hold. They also provide rules for constructing such a substitution.

In Section 4, for a run, we defined the sequence of sets $(T_i; E_i)$, and sets of atomic formulas X_i, for $i \leqslant n$. This can be viewed as a generalized constraint system, where we want to find substitutions under which $(T_i; E_i)$ can derive the equality assertions in X_i, and T_i can derive the public terms of X_i. It is a worthwhile exercise to adapt the existing constraint solving approaches to solve such generalized constraint systems. We leave this for future work.

5.3 Full disjunction

An interesting feature of the language in [35] is the use of disjunction. While our syntax here uses list membership to express a limited form of disjunction that seems to suffice for many protocols, it would be worthwhile to explore the utility of full disjunction and its effect on the active intruder problem.

In fact, with disjunction, we know that even the derivability problem becomes more involved. To check if $(S; A) \vdash_a \gamma$, one can no longer work with a single kernel of $(S; A)$. One can define a notion of "down-closure". For each disjunctive formula $\alpha \vee \beta$, one obtains two down-closures – one containing α, and the other β. In general, many disjunctions could occur in A, and there are exponentially many down-closures for any $(S; A)$. Using a left disjunction property similar to those in Lemma 13 ($\alpha \vee \beta$ derives γ iff γ is derivable from α and from β), we check if the kernels of all down-closures of $(S; A)$ derive γ. Thus, the derivability problem is in Π_2. Some of these down-closures might even contain contradictory assertions, and hence our techniques for the insecurity problem do not seem to directly apply. Exploring these issues is an interesting direction of research and is left for future work.

5.4 Adding if-then-else branching to protocols

As mentioned earlier, we can add an $A : \mathsf{assert}\ \alpha$ action that allows the role to proceed only if α can be derived using the information that A has at the time. Similarly, we can add an action of the form $A : \mathsf{deny}\ \alpha$, which lets the role proceed only if α *cannot be derived* using A's current knowledge. To simulate an if-then-else branch (by specifying a condition α to be checked and an agent A who will check it), we create two roles, one containing $A : \mathsf{assert}\ \alpha$ followed by the actions in the then branch, and the other containing $A : \mathsf{deny}\ \alpha$ followed by the actions in the else branch. We can easily extend our results to protocols involving such assert and deny actions where the condition being checked is whether or not a predicate holds about some atomic terms (for example, $\mathsf{el}(V)$ in Section 2.3).

The fact that a predicate P holds about some terms $\vec{t}$ can be modelled as the presence of $\vec{t}$ in a global list. We can also extend the model to allow agents (with appropriate access privileges) to add and delete entries from global lists, as considered in tools like Proverif [11] and in some versions of applied-pi [5, 28]. The technical proofs in our work continue to hold for these extensions.

5.5 Adding assertions to other models and tools

It is also useful to add communicable assertions to the widely-used applied pi calculus [1]. It would be especially interesting to see how this impacts the notion of static equivalence, and then study expressibility and decidability. As mentioned earlier, one can express certain "equivalence" properties in a more natural manner with assertions as compared to the terms-only model. Another promising extension is to study which equivalence properties can be expressed as reachability properties in this manner, like in [25]. These would also help us to extend existing tools [11, 13, 21, 33] with assertions.

References

1 Martín Abadi, Bruno Blanchet, and Cédric Fournet. The applied pi calculus: mobile values, new names, and secure communication. *Journal of the ACM*, 65(1):1:1–1:41, 2017.

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

3 Ben Adida. Helios: web-based open-audit voting. In *17th Conference on Security Symposium*, pages 335–348, 2008.

4 Roberto M. Amadio, Denis Lugiez, and Vincent Vanackére. On the symbolic reduction of processes with cryptographic functions. *Theoretical Computer Science*, 290(1):695–740, 2003.

5 Myrto Arapinis, Jia Liu, Eike Ritter, and Mark Ryan. Stateful applied pi calculus: observational equivalence and labelled bisimilarity. *Journal of Logical and Algebraic Methods in Programming*, 89:95–149, 2017.

6 Michael Backes, Cătălin Hriţcu, and Matteo Maffei. Automated verification of remote electronic voting protocols in the applied pi-calculus. In *21st IEEE Computer Security Foundations Symposium*, pages 195–209, 2008.

7 Michael Backes, Matteo Maffei, and Dominique Unruh. Zero-knowledge in the applied pi-calculus and automated verification of the Direct Anonymous Attestation protocol. In *29th IEEE Symposium on Security and Privacy*, pages 202–215, 2008.

8 A. Baskar, R. Ramanujam, and S. P. Suresh. A DEXPTIME-complete Dolev-Yao theory with distributive encryption. In *35th International Symposium on Mathematical Foundations of Computer Science*, volume 6281 of *Lecture Notes in Computer Science*, pages 102–113, 2010.

9 Mathieu Baudet. Deciding security of protocols against off-line guessing attacks. In *12th ACM Conference on Computer and Communications Security*, pages 16–25, 2005.

10 Bruno Blanchet. An efficient cryptographic protocol verifier based on Prolog rules. In *14th IEEE Computer Security Foundations Workshop*, pages 82–96, 2001.

11 Bruno Blanchet. Modeling and verifying security protocols with the applied pi calculus and ProVerif. *Foundations and Trends in Privacy and Security*, 1(1):1–135, 2016.

12 Bruno Blanchet and Andreas Podelski. Verification of cryptographic protocols: tagging enforces termination. *Theoretical Computer Science*, 333(1–2):67–90, 2005.

13 Vincent Cheval, Steve Kremer, and Itsaka Rakotonirina. The DEEPSEC prover. In *Computer Aided Verification*, volume 10982 of *Lecture Notes in Computer Science*, pages 28–36, 2018.

14 Vincent Cheval, Steve Kremer, and Itsaka Rakotonirina. The hitchhiker's guide to decidability and complexity of equivalence properties in security protocols. In *Logic, Language, and Security: Essays Dedicated to Andre Scedrov on the Occasion of his 65th Birthday*, volume 12300 of *Lecture Notes in Computer Science*, pages 127–145, 2020.

15 Yannick Chevalier, Ralf Küsters, Michaël Rusinowitch, and Mathieu Turuani. An NP decision procedure for protocol insecurity with XOR. *Theoretical Computer Science*, 338(1–3):247–274, 2005.

16 Hubert Comon-Lundh and Vitaly Shmatikov. Intruder deductions, constraint solving and insecurity decisions in presence of exclusive or. In *18th IEEE Symposium on Logic in Computer Science*, pages 271–280, 2003.

17 Véronique Cortier, Stéphanie Delaune, and Pascal Lafourcade. A survey of algebraic properties used in cryptographic protocols. *Journal of Computer Security*, 14(1):1–43, 2006.

18 Véronique Cortier, Stéphanie Delaune, and Vaishnavi Sundararajan. A decidable class of security protocols for both reachability and equivalence properties. *Journal of Automated Reasoning*, 65(4):479–520, 2021.

19 Véronique Cortier and Steve Kremer. Formal models and techniques for analyzing security protocols: a tutorial. *Foundations and Trends in Programming Languages*, 1(3):151–267, 2014.

20 Véronique Cortier, Michaël Rusinowitch, and Eugen Zălinescu. A resolution strategy for verifying cryptographic protocols with CBC encryption and blind signatures. In *7th ACM SIGPLAN International Conference on Principles and Practice of Declarative Programming*, pages 12–22, 2005.

21 Cas J. F. Cremers. The Scyther tool: verification, falsification, and analysis of security protocols. In *20th International Conference on Computer Aided Verification*, volume 5123 of *Lecture Notes in Computer Science*, pages 414–418, 2008.

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

23 Nancy Durgin, Patrick Lincoln, John Mitchell, and Andre Scedrov. Multiset rewriting and the complexity of bounded security protocols. *Journal of Computer Security*, 12(2):247–311, 2004.

24 Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. A practical secret voting scheme for large scale elections. In *Advances in Cryptology – AUSCRYPT*, volume 718 of *Lecture Notes in Computer Science*, pages 244–251, 1992.

25 Sébastien Gondron, Sebastian Mödersheim, and Luca Viganò. Privacy as reachability. In *35th IEEE Computer Security Foundations Symposium*, pages 130–146, 2022.

26 Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In *Advances in Cryptology – EUROCRYPT*, volume 4965 of *Lecture Notes in Computer Science*, pages 415–432, 2008.

27 Nevin Heintze and Doug Tygar. A model for secure protocols and their compositions. *IEEE Transactions on Software Engineering*, 22(1):16–30, 1996.

28 Steve Kremer and Robert Künnemann. Automated analysis of security protocols with global state. *Journal of Computer Security*, 24(5):583–616, 2016.

29 Steve Kremer and Mark Ryan. Analysis of an electronic voting protocol in the applied pi calculus. In *Programming Languages and Systems – ESOP 2005*, volume 3444 of *Lecture Notes in Computer Science*, pages 186–200, 2005.

30 Pascal Lafourcade, Denis Lugiez, and Ralf Treinen. Intruder deduction for the equational theory of abelian groups with distributive encryption. *Information and Computation*, 205(4):581–623, 2007.

31 Matteo Maffei, Kim Pecina, and Mathieu Reinert. Security and privacy by declarative design. In *26th IEEE Computer Security Foundations Symposium*, pages 81–96, 2003.

32 David A. McAllester. Automatic recognition of tractability in inference relations. *Journal of the ACM*, 40(2):284–303, 1993.

33 Simon Meier, Benedikt Schmidt, Cas Cremers, and David Basin. The TAMARIN prover for the symbolic analysis of security protocols. In *25th International Conference on Computer Aided Verification*, volume 8044 of *Lecture Notes in Computer Science*, pages 696–701, 2013.

34 Jonathan K. Millen and Vitaly Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *8th ACM Conference on Computer and Communications Security*, pages 166–175, 2001.

35 R. Ramanujam, Vaishnavi Sundararajan, and S. P. Suresh. Existential assertions for voting protocols. In *Financial Cryptography and Data Security*, volume 10323 of *Lecture Notes in Computer Science*, pages 337–352, 2017.

36 R. Ramanujam and S. P. Suresh. Decidability of context-explicit security protocols. *Journal of Computer Security*, 13(1):135–165, 2005.

37 R. Ramanujam and S. P. Suresh. A (restricted) quantifier elimination for security protocols. *Theoretical Computer Science*, 367(1–2):228–256, 2006.

38 Michaël Rusinowitch and Mathieu Turuani. Protocol insecurity with finite number of sessions and composed keys is NP-complete. *Theoretical Computer Science*, 299(1–3):451–475, 2003.

A A substitution rule for equalities

We consider the following subst rule, with the side conditions $P \subseteq \mathbb{P}_x(t \bowtie u) \cap \mathbb{A}(S \cup \{x\}, t \bowtie u)$ and $S \vdash_{dy} \{r, s\}$.

$$\frac{(S; A) \vdash (t \bowtie u)[r]_P \quad (S; A) \vdash r \bowtie s}{(S; A) \vdash (t \bowtie u)[s]_P}\ \mathsf{subst}$$

▶ **Lemma 41.** *The* subst *rule is admissible in* $\vdash_{eq}$.

Proof. We first show that subst can be simulated by a series of applications of subst_1, defined below.

$$\frac{S; A \vdash (t \bowtie u)[r]_p \quad S; A \vdash r \bowtie s}{S; A \vdash (t \bowtie u)[s]_p}\ \mathsf{subst}_1$$

The rule is enabled only if $p \in \mathbb{P}_x(t \bowtie u) \cap \mathbb{A}(S \cup \{x\}, t \bowtie u)$ and $S \vdash_{dy} \{r, s\}$. subst_1 replaces the r *occurring at* p by s.

Let α denote $t \bowtie u$ and consider an instance of subst with $(T; E) \vdash \alpha[r]_P$ and $(T; E) \vdash r \bowtie s$ as premises and $(T; E) \vdash \alpha[s]_P$ as conclusion, with $P = \{p_1, \dots, p_l\} \subseteq \mathbb{P}_x(\alpha) \cap \mathbb{A}(T \cup \{x\}, \alpha)$, and $T \vdash_{dy} \{r, s\}$. The p_is are x-positions, so none of them is a prefix of another. Therefore, even after replacing the xs occurring in the set of positions $P \setminus \{p_i\}$ with some terms, p_i remains an x-position.

For $0 \leqslant i \leqslant l$, we define the following. $P_i := \{p_1, \dots, p_i\}$ and $R_i := P \setminus P_i = \{p_{i+1}, \dots, p_l\}$. We also define $\alpha_i := (\alpha[r]_{R_i})[s]_{P_i}$. Note that $\alpha_0 = \alpha[r]_P$ and $\alpha_l = \alpha[s]_P$. We can see that p_i is an x-position of $\beta_i = (\alpha[r]_{R_i})[s]_{P_{i-1}}$. By Lemma 8 for assertions, $p_i \in \mathbb{A}(T \cup \{x\}, \beta_i)$, since we have $P \subseteq \mathbb{A}(T \cup \{x\}, \alpha)$, We also see that $\alpha_{i-1} = \beta_i[r]_{p_i}$ and $\alpha_i = \beta_i[s]_{p_i}$. Thus we can get from α_{i-1} to α_i using the subst_1 rule, and from α_0 to α_l using a series of subst_1 rules.

Now we show that subst_1 can be simulated in the $\vdash_{eq}$ system. Let π is a proof of $(T; E) \vdash_{eq} r \bowtie s$, and let $T \vdash_{dy} \{r, s\}$. For all p, and for all t, u s.t. $p \in \mathbb{P}_x(t \bowtie u) \cap \mathbb{A}(T \cup \{x\}, t \bowtie u)$, we show that whenever $(T; E) \vdash_{eq} (t \bowtie u)[r]_p$, then $(T; E) \vdash_{eq} (t \bowtie u)[s]_p$. The proof proceeds by induction on the length of p.

- $p = 0$: We have a proof of $r \bowtie u$. By sym, we get a proof of $u \bowtie r$. Combining this with π using trans, we get a proof of $u \bowtie s$, from which we can get a proof of $s \bowtie u$ by applying sym again.
- $p = 1$: We have a proof of $t \bowtie r$. Combining this with π using trans, we get a proof of $t \bowtie s$, as desired.
- $p = 0q$ for some $q \neq \varepsilon$: Note that $(t \bowtie u)[r]_p$ is the same as $t[r]_q \bowtie u$. Suppose $t = f(t_0, \dots, t_n)$ and $q = iq'$ for $i \leqslant n$. Then and $t[r]_q = f(t_0, \dots, t_i[r]_{q'}, \dots, t_n)$. Since $q \in \mathbb{A}(T \cup \{x\}, t)$, we have that $T \vdash_{dy} \{t_0, \dots, t_n\}$, and by eq, $(T; E) \vdash_{eq} t_j \bowtie t_j$ for $j \leqslant n$. From $T \vdash_{dy} r$, and $q' \in \mathbb{A}(T \cup \{x\}, t_i)$, we have that $T \vdash_{dy} t_i[r]_{q'}$. Thus, we have a proof of $(T; E) \vdash_{eq} t_i[r]_{q'} \bowtie t_i[r]_{q'}$ using eq. Applying IH to the position $0q'$, we have that $(T; E) \vdash_{eq} t_i[s]_{q'} \bowtie t_i[r]_{q'}$. By applying cons to this and $(T; E) \vdash_{eq} t_j \bowtie t_j$ for the other $j \leqslant n$, we conclude that $(T; E) \vdash_{eq} t[s]_q \bowtie t[r]_q$. Applying trans to this and $t[r]_q \bowtie u$, we get a proof of $t[s]_q \bowtie u$, i.e. $(t \bowtie u)[s]_p$.
- $p = 1q$ for some $q \neq \varepsilon$: This is similar to the above. ⊣

B Proof of Theorem 40

We want to check if $(S;A) \vdash_a \alpha$, where $\mathsf{bv}(\alpha) \cap \mathsf{vars}(S;A) = \emptyset$. Let $(T;E) = \mathit{ker}(S;A)$. By Theorem 16, this reduces to checking if there is a substitution μ with $\mathsf{dom}(\mu) = \mathsf{bv}(\alpha)$ s.t. and $X \subseteq \mathsf{hat}(\alpha)$ s.t. $\forall x \in \mathsf{bv}(\alpha) : T \vdash_{dy} \mu(x)$ and for all $\beta \in X$, $(T;E) \vdash_a \mu(\beta)$. For formulas in X that are not of the form $t \bowtie u$, all terms occurring in them are variables or names, so μ is atomic on variables occurring in them. It therefore suffices to only consider assertions of the form $t \bowtie u$.

So the problem is as follows. There is a set of terms C closed under subterms, and $(T;E)$ with $\mathsf{st}(T) \cup \mathsf{st}(E) \subseteq \mathsf{C}$. We have a substitution μ with $\mathsf{dom}(\mu) \cap \mathsf{vars}(T;E) = \emptyset$, which satisfies some derivabilities of the form $T \vdash_{dy} \mu(t)$ and $T;E \vdash_{eq} \mu(t \bowtie u)$, where $t, u \in \mathsf{C}$. We seek a small ν that preserves the above derivabilities. To reduce clutter, we use Z to refer to $\mathsf{dom}(\mu)$. Let $\mathsf{D} = \mathsf{C} \setminus Z$. Since $T \vdash_{dy} \mu(x)$, all variables occurring in $\mu(x)$ must also be in $\mathsf{vars}(T)$. But $\mathsf{vars}(T;E) \cap Z = \emptyset$, so $\mathsf{vars}(\mu(x)) \cap Z = \emptyset$.

Define $r \approx s$ iff $T;E \vdash_{eq} \mu(r \bowtie s)$. It is easy to see that $\approx$ is a partial equivalence relation (on the subset of terms r such that $T \vdash_{dy} \mu(r)$).

We say that $x \in Z$ is *minimal* if there is no $t \in \mathsf{D}$ with $x \approx t$. Let V_m denote the set of all minimal variables. Our strategy for finding a small ν is to "zap" minimal variables, and propagate the change to (interpretations of) non-minimal variables. To this end, it is convenient to translate every term to an "equivalent" one with only minimal variables. The notion of equivalence is based on unifiability under μ. The set of all such terms that are equivalent to terms in C is defined as follows.

▶ **Definition 42.** $\widehat{\mathsf{C}} \coloneqq \{t \mid \mathsf{vars}(t) \cap Z \subseteq \mathsf{V}_m$, *and either* $t \in \mathsf{V}_m$ *or* $\exists u \in \mathsf{D} : t \approx u\}$.

▶ **Lemma 43.** *For every* $t \in \mathsf{C}$ *with* $T \vdash_{dy} \mu(t)$*, there is* $t^* \in \widehat{\mathsf{C}}$ *such that:* $T \vdash_{dy} \mu(t^*)$*;* $t \approx t^*$*; and for all* $x \in \mathsf{V}_m$, $\mathbb{P}_x(t^*) \subseteq \mathbb{A}(T \cup Z, t^*)$.

Proof. For $x, y \in Z$, $x \prec y$ iff $\exists r \in \mathsf{D}[x \in \mathsf{st}(r)$ and $r \approx y]$.

We now show that $\prec$ is acyclic. Towards this, we claim that if $x \prec y$ and $y \prec z$, then there is some term a (not necessarily in C) s.t. $\mu(x)$ is a proper subterm of $\mu(a)$ and $a \approx z$. Extending this reasoning, we see that if $x \prec^+ x$, we have some term a such that $\mu(x)$ is a proper subterm of $\mu(a)$ and $(T;E) \vdash_{eq} \mu(a) \bowtie \mu(x)$. But E is consistent, which means that there is some λ s.t. $\lambda(\mu(a)) = \lambda(\mu(x))$. But this is incompatible with $\mu(x)$ being a proper subterm of $\mu(a)$. Thus $\prec$ is acyclic.

We now prove the claim. Suppose $x \prec y$ and $y \prec z$. Then there exists $r, s \in \mathsf{D}$ such that $x \in \mathsf{st}(r)$, $(T;E) \vdash_{eq} \mu(r) \bowtie \mu(y)$, $y \in \mathsf{st}(s)$, and $(T;E) \vdash_{eq} \mu(s) \bowtie \mu(z)$. Let $a = s[r]_{\mathbb{P}_y(s)}$. We see that $\mu(x)$ is a proper subterm of $\mu(a)$. From the abstractability conditions satisfied by μ and the derivability of $\mu(x)$ for all $x \in Z$, we can justify the applications of subst necessary to obtain $(T;E) \vdash_{eq} \mu(a) \bowtie \mu(z)$ and thus $a \approx z$.

Since $\prec$ is acyclic, we can define a notion of *rank* of variables: $\mathit{rank}(x) = \max\{\mathit{rank}(y) \mid y \prec^+ x\} + 1$. For a term $u \in \mathsf{D}$, we define $\mathit{rank}(u) = \max\{\mathit{rank}(x) \mid x \in \mathsf{vars}(u) \cap Z\}$. It is easy to verify that if $u \in \mathsf{D}$ and $x \approx u$, then $\mathit{rank}(x) > \mathit{rank}(u)$. It is also easy to see that every $x \in \mathsf{V}_m \cap Z$ has rank 0.

Coming back to the statement of the lemma, we prove it by induction on $\delta(t) = (\mathit{rank}(t), |t|)$. First fix an ordering on $\widehat{\mathsf{C}}$. For $\delta(t) = (0, 0)$, we have that t is a variable x and $\mathit{rank}(x) = 0$. We have two cases to consider.

- $x \in \mathsf{V}_m$: Choose $x^* = x$.

- $x \notin V_m$: This means that there is some $u \in D$ s.t. $x \approx u$. But since $rank(x) = 0$, $\mathrm{vars}(u) \cap Z = \emptyset$ for each such u. Choose x^* to be the earliest such u (according to an a priori fixed ordering on $\widehat{C}$). Clearly $(T; E) \vdash \mu(x) \bowtie \mu(x^*)$, and by Lemma 15, $T \vdash_{dy} \mu(x^*)$. Finally $\mathrm{vars}(x^*) \cap Z = \emptyset$, so it is vacuously true that $\mathbb{P}_y(x^*) \subseteq \mathbb{A}(T \cup Z, x^*)$ for all $y \in V_m$.

So suppose $\delta(t) > (0, 0)$ and that the theorem is true for all u such that $\delta(u) < \delta(t)$. There are two cases to consider:

- t is a variable, say x: Then $rank(x) > 0$, and there is $u \in D$ s.t. $x \approx u$, whence $rank(u) < rank(x)$. Pick the earliest such $u \in \widehat{C}$. By IH there is u^*, and we define $x^* = u^*$. Since $x \approx u$ and $u \approx u^*$, we have $x \approx x^*$, by transitivity.
- t is not a variable: For each $y \in \mathrm{vars}(t) \cap Z$, there is y^*. We obtain t^* by replacing each y by y^*. Clearly $\mathrm{vars}(t^*) \cap Z \subseteq V_m$. Also since all variables appear in abstractable positions of t, we can justify the relevant applications of subst to justify $t \approx t^*$. Finally, if z appears in an abstractable position in r and y appears in an abstractable position in s, then z appears in an abstractable position in $s[r]_{\mathbb{P}_y(s)}$. Thus the abstractability part of the statement is also fulfilled. ⊣

We now define the substitution ν as follows. Assume that there is some $\mathsf{m} \in T \cap N$ such that $\mathsf{m} \notin \mathrm{st}(E \cup \{\alpha\}) \cup \mathrm{st}(\mathrm{rng}(\mu))$.[9] Let ν_m be the substitution that maps each $x \in V_m$ to m. For all $x \in Z : \nu(x) = \nu_m(x^*)$. Notice that for all $x \in \mathrm{dom}(\nu)$, either $\nu(x) = \mathsf{m}$ or there is $u \in D$ s.t. $\nu(x) = \nu(u)$. Thus we can show that ν is $|C|$-bounded following the proof of Theorem 39. To complete the proof of Theorem 40, we just need to show that ν preserves derivability. This is proved in Theorem 45, the main result of this section. But first we state a useful observation.

▶ **Observation 44.**

1. *For $x \in Z$, if $\mu(x) \in C$ then $x \notin V_m$.*
2. *If $t \in \widehat{C}$ and $\mu(t) \in C$, then $\mathrm{vars}(t) \cap Z = \emptyset$ and $\mu(t) = t$.*

Proof.

1. Let $\mu(x) = t \in C$. Since $\mathrm{vars}(t) \cap Z = \emptyset$, we have that $t \notin Z$ and $\mu(t) = t$. Thus $t \in D$, and $\mu(x) \bowtie \mu(t)$ is derivable using the eq rule, i.e., $x \approx t$. Therefore $x \notin V_m$.
2. For every $x \in \mathrm{vars}(t) \cap Z$, $\mu(x) \in C$. Thus we have $x \notin V_m$, by the previous part. But since $t \in \widehat{C}$, we have that $\mathrm{vars}(t) \cap Z \subseteq V_m$. The only conclusion is that $\mathrm{vars}(t) \cap Z = \emptyset$, and thus $\mu(t) = t$. ⊣

▶ **Theorem 45.**

1. *For any $t \in C$, if $T \vdash_{dy} \mu(t)$ then $T \vdash_{dy} \nu(t)$.*
2. *For any $t, u \in C$, if $T; E \vdash_{eq} \mu(t) \bowtie \mu(u)$ then $T; E \vdash_{eq} \nu(t) \bowtie \nu(u)$.*

Proof. By Lemma 43, it suffices to prove the following. Let $r, s \in \widehat{C}$ s.t. $\forall x \in V_m$, $\mathbb{P}_x((r, s)) \subseteq \mathbb{A}(T \cup Z, (r, s))$. If $T \vdash_{dy} \mu(r)$ then $T \vdash_{dy} \nu_m(r)$; and if $T; E \vdash_{eq} \mu(r) \bowtie \mu(s)$ then $T; E \vdash \nu_m(r) \bowtie \nu_m(s)$.

1. Suppose $T \vdash_{dy} \mu(r)$ for r as above. Since all positions of variables from Z occurring in r are abstractable w.r.t. $T \cup Z$, and since $T \cup \{\mathsf{m}\} \vdash_{dy} \mathsf{m}$, we can easily prove by induction on the size of terms that $T \cup \mathsf{m} \vdash_{dy} \nu_m(r)$.

[9] Thus m is a "spare name" that does not occur in any of the derivations under consideration.

2. Suppose $T; E \vdash_{eq} \mu(r) \bowtie \mu(s)$ for r, s as above. Let π be a normal proof of $T; E \vdash \mu(r) \bowtie \mu(s)$ with last rule r. We prove the desired statement by induction on the structure of π. There are the following cases to consider.
   - $\mathsf{r} \in \{\mathsf{ax}, \mathsf{eq}, \mathsf{proj}\}$: Three cases arise: $\mu(r) \bowtie \mu(s) \in E$, and thus $\mu(r), \mu(s) \in C$. Or $\mu(r) = \mu(s)$ and $T \vdash_{dy} \mu(r)$ via a proof ending in ax or a destructor rule, and thus $\mu(r), \mu(s) \in \mathsf{st}(T) \subseteq C$. Or by subterm property for normal $\vdash_{eq}$-proofs $\mu(r), \mu(s) \in \mathsf{st}(T; E) \subseteq C$. Thus $\mu(r), \mu(s) \in C$ in all three cases. By Observation 44, $\mathsf{vars}(r, s) \cap Z = \emptyset$. Thus $\nu_m(r) = r = \mu(r)$ and $\nu_m(s) = s = \mu(s)$. Therefore π itself is a proof of $\nu_m(r) \bowtie \nu_m(s)$.
   - $\mathsf{r} = \mathsf{trans}$: Suppose the immediate subproofs are $\pi_1, \ldots, \pi_n$, with each π_i deriving $v_{i-1} \bowtie v_i$. Let $\mu(r) = v_0$ and $\mu(s) = v_n$. Since no π_i ends in trans and no two adjacent π_i's end in cons, each v_i (for $0 < i < n$) appears in at least one proof ending in ax, eq or proj. Thus, by the subterm property, $v_i \in \mathsf{st}(T; E) \subseteq C$ for $0 < i < n$. Since $\mathsf{vars}(T; E) \cap Z = \emptyset$, it follows that $v_i \in \widehat{C}$ and $\mu(v_i) = v_i$. Thus we can view each π_i as deriving $\mu(r_{i-1}) \bowtie \mu(r_i)$, where $r_{i-1}, r_i \in \widehat{C}$ (taking r_0 and r_n to be r and s). By IH, there are proofs $\varpi_1, \ldots, \varpi_n$, with each ϖ_i deriving $\nu_m(r_{i-1}) \bowtie \nu_m(r_i)$. By composing them using trans, we get a proof of $T; E \vdash \nu_m(r) \bowtie \nu_m(s)$, as desired.
   - $\mathsf{r} = \mathsf{cons}$: Suppose $r = \mathsf{f}(r_1, \ldots, r_n)$ and $s = \mathsf{f}(s_1, \ldots, s_n)$. Each $r_i, s_i \in \widehat{C}$, and the immediate subproofs are $\pi_1, \ldots, \pi_n$, with each π_i deriving $\mu(r_i) \bowtie \mu(s_i)$. By IH we have proofs $\varpi_1, \ldots, \varpi_n$, with each ϖ_i proving $\nu_m(r_1) \bowtie \nu_m(s_1)$. We can compose them with the cons rule to get the desired proof of $\nu_m(r) \bowtie \nu_m(s)$.

     Suppose, on the other hand, that r is a variable. Since $r \in \widehat{C}$, $r \in V_m$. Now $s \in \widehat{C}$, so either $s \in V_m$ or there is $a \in D$ with $s \approx a$. But in the second case, $r \approx a$ (by symmetry and transitivity), which cannot happen for a minimal variable r. Therefore $s \in V_m$. And we have $\nu_m(r) = \nu_m(s) = \mathsf{m} \in T$, so there is a proof of $T, E \vdash_{eq} \nu_m(r) \bowtie \nu_m(s)$ ending in eq.

     We have a similar argument in case s is a variable, thereby proving the theorem. ⊣

C Normalization and subterm property for $\vdash_{eq}$

Suppose $E \cup \{\alpha\}$ consists only of atomic formulas and π is a proof of $(T; E) \vdash_{eq} \alpha$. We use "$\mathsf{r}_1$ *precedes* r_2 in π" to mean that the conclusion of some application of r_1 is a premise of an application of r_2 in π.

Recall that π is *normal* if the following hold.

1. All $\vdash_{dy}$ subproofs are normal.
2. sym is only preceded by ax or prom.
3. eq is only preceded by a destructor rule.
4. No premise of a trans is of the form $a \bowtie a$, or the conclusion of a trans.
5. Adjacent premises of a trans are not conclusions of cons.
6. int is not preceded by int or wk.
7. No subproof ending in proj contains cons.

Recall that $(T; E)$ is consistent if there is a substitution λ such that $\lambda(a) = \lambda(b)$ whenever $(T; E) \vdash_{eq} a \bowtie b$ and $\lambda(t) \in \{t_1, \ldots, t_n\}$ whenever $(T; E) \vdash_{eq} t \twoheadleftarrow [t_1, \ldots, t_n]$.

We next prove normalization for $\vdash_{eq}$ proofs (with a consistent LHS). We present proof transformation rules in Table 3. To save space, we use *proof terms* – $\mathsf{r}(\pi_1, \ldots, \pi_n)$ denotes a proof π with last

R1	$\mathsf{eq}(\mathsf{f}(\pi_1, \ldots, \pi_k))$	$\rightsquigarrow$	$\mathsf{cons}_\mathsf{f}(\mathsf{eq}(\pi_1), \ldots, \mathsf{eq}(\pi_k))$
R2	$\mathsf{sym}(\mathsf{eq}(\pi))$	$\rightsquigarrow$	$\mathsf{eq}(\pi)$
R3	$\mathsf{sym}(\mathsf{sym}(\pi))$	$\rightsquigarrow$	π
R4	$\mathsf{sym}(\mathsf{r}(\pi_1, \ldots, \pi_k))$	$\rightsquigarrow$	$\mathsf{r}(\mathsf{sym}(\pi_1), \ldots, \mathsf{sym}(\pi_k))$
R5	$\mathsf{trans}(\pi_1, \ldots, \pi_{i-1}, \varpi, \pi_i, \ldots, \pi_{r-1})$	$\rightsquigarrow$	$\mathsf{trans}(\pi_1, \ldots, \pi_{i-1}, \pi_i, \ldots, \pi_{r-1})$
R6	$\mathsf{trans}(\pi_1, \ldots, \mathsf{trans}(\pi_i^1, \ldots, \pi_i^k), \ldots, \pi_{r-1})$	$\rightsquigarrow$	$\mathsf{trans}(\pi_1, \ldots, \pi_i^1, \ldots, \pi_i^k, \ldots, \pi_{r-1})$
R7	$\mathsf{trans}(\pi_1, \ldots, \mathsf{cons}(\pi_{i-1}^1, \ldots, \pi_{i-1}^k), \mathsf{cons}(\pi_i^1, \ldots, \pi_i^k), \ldots, \pi_{r-1})$	$\rightsquigarrow$	$\mathsf{trans}(\pi_1, \ldots, \mathsf{cons}(\mathsf{trans}(\pi_{i-1}^1, \pi_i^1), \ldots, \mathsf{trans}(\pi_{i-1}^k, \pi_i^k)), \ldots, \pi_{r-1})$
R8	$\mathsf{proj}_m(\mathsf{cons}(\pi_1, \ldots, \pi_k))$	$\rightsquigarrow$	π_m
R9	$\mathsf{proj}_m(\mathsf{trans}(\pi_1, \ldots, \pi_{i-1}, \mathsf{cons}_\mathsf{f}(\pi_i^1, \ldots, \pi_i^k), \pi_{i+1}, \ldots, \pi_{r-1}))$	$\rightsquigarrow$	$\mathsf{trans}(\mathsf{proj}_m(\mathsf{trans}(\pi_1, \ldots, \pi_{i-1})), \pi_i^m, \mathsf{proj}_m(\mathsf{trans}(\pi_{i+1}, \ldots, \pi_{r-1})))$
R10	$\mathsf{int}(\pi_1, \ldots, \pi_{k-1}, \mathsf{int}(\pi_k, \ldots, \pi_m), \pi_{m+1}, \ldots, \pi_n)$	$\rightsquigarrow$	$\mathsf{int}(\pi_1, \ldots, \pi_{k-1}, \pi_k, \ldots, \pi_m, \pi_{m+1}, \ldots, \pi_n)$
R11	$\mathsf{int}(\pi_1, \ldots, \mathsf{wk}(\pi_i), \ldots, \pi_n)$	$\rightsquigarrow$	$\mathsf{wk}(\pi_i)$

Table 3 Proof transformation rules. In R4, $\mathsf{r} \in \{\mathsf{trans}, \mathsf{proj}, \mathsf{cons}\}$. In R5, $\mathsf{conc}(\varpi)$ is assumed to be of the form $a \bowtie a$.

rule r and immediate subproofs $\pi_1, \ldots, \pi_n$. It is assumed that the derivations are from a consistent $(T; E)$. R1 is applicable when f is a constructor rule, and ensures that $\vdash_{dy}$ subproofs do not end in a constructor rule. R2 and R3 eliminate some occurrences of sym, while R4 pushes sym up towards the axioms. R5 and R6 ensure that no premise of trans is the conclusion of eq or trans. R7 ensures that adjacent premises of trans are not the result of cons. R8 simplifies proofs where proj follows cons. We will discuss R9 later. R10 ensures that the conclusion of int is not a premise of int. In R11, π_i proves an equality $v \bowtie n$, and it is weakened to a list membership of the form $v \twoheadleftarrow l'$, but by consistency, even after intersection, the conclusion must be of the form $v \twoheadleftarrow l$ where $\lambda(v)$ is an element of l for some λ. Thus we can directly apply weakening to π_i to get the same conclusion.

R9 requires some explanation. Let π_i be the proof $\mathsf{cons}_\mathsf{f}(\pi_i^1, \ldots, \pi_i^k)$, and let $\mathsf{conc}(\pi_j)$ be $t_j \bowtie t_{j+1}$, for $1 \leqslant j < r$. We see that $\mathsf{conc}(\mathsf{trans}(\pi_1, \ldots, \pi_{r-1}))$ is $t_1 \bowtie t_r$. Since proj is applied on this, there is some constructor g such that $t_e = \mathsf{g}(t_e^1, \ldots, t_e^k)$ for $e \in \{1, r\}$. Since π_i ends in cons_f, we see that $t_e = \mathsf{f}(t_e^1, \ldots, t_e^l)$ for $e \in \{i, i+1\}$. But $t_1 \bowtie t_i$ is provable from $(T; E)$, which is consistent. Therefore it has to be the case that $\mathsf{f} = \mathsf{g}$ (and $k = l$). Thus we see that for all $e \in \{1, i, i+1, r\}$, $t_e = \mathsf{f}(t_e^1, \ldots, t_e^k)$. So we can rewrite the LHS of R9 to the RHS to get a valid proof. Note that we can apply proj on $t_1 \bowtie t_i$ in the transformed proof since all components of t_1 and t_i are abstractable – for t_1 this is true because the proj rule was applied to $t_1 \bowtie t_r$ in the proof on the LHS; and for t_i this follows from the fact that $\pi_i^1, \ldots, \pi_i^k$ derive respectively $t_i^1 \bowtie t_{i+1}^1, \ldots, t_i^k \bowtie t_{i+1}^k$, and so by Lemma 15, $T \vdash_{dy} \{t_i^1, \ldots, t_i^k\}$. For a similar reason, we can apply proj on $t_{i+1} \bowtie t_r$.

▶ **Theorem 46.** *If $(T; E) \vdash_{eq} \alpha$ then there is a normal proof of $(T; E) \vdash \alpha$ in the $\vdash_{eq}$ system.*

Proof. Let π be any proof of $(T; E) \vdash \alpha$ such that all DY subproofs of π are normal. Suppose we repeatedly apply the transformations of Table 3 starting with π and reach a proof ϖ on which we can no longer apply any of the rules. Then ϖ satisfies clauses 1 to 6 in the definition of normal proofs (since none of the rewrite rules, in particular R1–R7 and R10–R11, apply to ϖ).

Clause 7 is also satisfied by ϖ, for the following reason. Suppose a subproof ϖ_1 ends in proj and ϖ_2 is a maximal subproof of ϖ_1 ending in cons. ϖ_2 is a proper subproof of ϖ_1, so there has to be a subproof of ϖ_1 of the form $\rho = \mathsf{r}(\cdots \varpi_2 \cdots)$. Since cons appears as the rule above r, a priori, r can only be one of $\{\mathsf{sym}, \mathsf{trans}, \mathsf{proj}, \mathsf{cons}\}$. But since ϖ_2 is a *maximal subproof* of ϖ_1 ending in cons, $\mathsf{r} \neq \mathsf{cons}$. Since R4 and R8 cannot be applied on ϖ, $\mathsf{r} \notin \{\mathsf{sym}, \mathsf{proj}\}$. But if $\mathsf{r} = \mathsf{trans}$, then ρ is a proper subproof of ϖ_1. In particular, it is the immediate subproof of some $\rho' = \mathsf{r}'(\cdots \rho \cdots)$. Now r′

cannot be subst, since then $\mathsf{conc}(\rho')$ is a list membership assertion, which cannot occur in a proof ending in proj. $\mathsf{r}' \neq$ cons, as that would violate the maximality of ϖ_2. $\mathsf{r}' \notin \{\mathsf{sym}, \mathsf{trans}, \mathsf{proj}\}$, since then one of the rewrite rules R4, R6, R9 would apply to ϖ. We have ruled out all possible cases for r', and thus we are forced to conclude that ϖ_2 cannot be a subproof of ϖ_1. Thus, cons does not occur in any subproof of ϖ ending in proj, and ϖ satisfies all the clauses in the definition of normal proofs.

We next show that we can always reach a stage where no transformation is enabled. To begin with, apply the rules R2–R4 until the premise of each occurrence of sym is the conclusion of an ax or a prom. None of the other rules converts a proof ending in ax or prom to one which does not, so the above property is preserved even if we apply the other rules in any order.

Associate three sizes to an $\vdash_{eq}$-proof π:

- $\delta_1(\pi)$ is the sum of the sizes of the $\vdash_{dy}$ subproofs of π,
- $\delta_2(\pi)$ is the number of cons rules that occur in π, and
- $\delta_3(\pi)$ is the size of the proof π (number of nodes in the proof tree).

We also define $\delta(\pi) \coloneqq (\delta_1(\pi), \delta_2(\pi), \delta_3(\pi))$.

We now show that if π' is obtained from π by one application of any of the transformation rules other than R2–R4, $\delta(\pi') < \delta(\pi)$.

- If R1 is applied, $\delta_1(\pi') < \delta_1(\pi)$ and so $\delta(\pi') < \delta(\pi)$.
- If R7 or R9 is applied, we have $\delta_1(\pi') \leqslant \delta_1(\pi)$ and $\delta_2(\pi') < \delta_2(\pi)$. Therefore, $\delta(\pi') < \delta(\pi)$.
- If R5, R6, R8, R10 or R11 is applied, we have that $\delta_i(\pi') \leqslant \delta_i(\pi)$ for $i \in \{1, 2\}$ and $\delta_3(\pi') < \delta_3(\pi)$. So $\delta(\pi') < \delta(\pi)$.

Thus, once we apply R2–R4 till they can no longer be applied,[10] we cannot have an infinite sequence of transformations starting from any π. Hence, every proof π can be transformed into a normal proof ϖ with the same conclusion. ⊣

We state and prove the subterm property next.

▶ **Theorem 47.** *For any normal proof π of $T; E \vdash_{eq} \alpha$,*
$\mathsf{terms}(\pi) \subseteq \mathsf{st}(T) \cup \mathsf{st}(E \cup \{\alpha\})$, *and*
$\mathsf{lists}(\pi) \subseteq \mathsf{lists}(E \cup \{\alpha\}) \cup \{[n] \mid n \in \mathsf{st}(T) \cup \mathsf{st}(E \cup \{\alpha\})\}$.

If π does not contain cons, *then* $\mathsf{terms}(\pi) \subseteq \mathsf{st}(T) \cup \mathsf{st}(E)$. *Also, if π does not end in* wk *and does not end in* int, *then* $\mathsf{lists}(\pi) \subseteq \mathsf{lists}(E) \cup \{[n] \mid n \in \mathsf{st}(T) \cup \mathsf{st}(E)\}$.

We implicitly use the following easily provable facts.

(F1) If a normal proof π ends in trans and an immediate subproof ϖ does not end in cons, then cons does not occur in ϖ.

(F2) If a normal proof π derives a list membership assertion, cons does not occur in π.

Proof. Let r be the last rule of π. We have the following cases. We mention $\mathsf{lists}(\pi)$ only in cases where the rules involve lists.

[10] This process terminates because the sum of the sizes of subproofs rooted with sym decreases on each application of R2–R4.

- r = ax: $\alpha \in \mathrm{E}$, so $\mathsf{terms}(\pi) \subseteq \mathsf{st}(\mathrm{E})$ and $\mathsf{lists}(\pi) \subseteq \mathsf{lists}(\mathrm{E})$.
- r = eq: α is $t \bowtie t$ and $\mathrm{T} \vdash_{dy} t$. Since π is a normal proof whose $\vdash_{dy}$ subproofs are also normal, $\mathrm{T} \vdash_{dy} t$ via a proof ending in a destructor rule, and by subterm property for $\vdash_{dy}$, it follows that $t \in \mathsf{st}(\mathrm{T})$. Thus $\mathsf{terms}(\pi) = \{t\} \subseteq \mathsf{st}(\mathrm{T})$.
- r = sym: $\mathsf{terms}(\pi) = \mathsf{terms}(\pi')$, where π' is the immediate subproof, and the statement follows by IH.
- r = cons: α is $\mathsf{f}(t_1, \ldots, t_k) \bowtie \mathsf{f}(u_1, \ldots, u_k)$, and for $i \in \{1, \ldots, k\}$, there is a subproof π_i with conclusion $t_i \bowtie u_i$. By IH, $\mathsf{terms}(\pi_i) \subseteq \mathsf{st}(\mathrm{T} \cup \{t_i, u_i\}) \cup \mathsf{st}(\mathrm{E}) \subseteq \mathsf{st}(\mathrm{T}) \cup \mathsf{st}(\mathrm{E} \cup \{\alpha\})$ for $i \in \{1, \ldots, k\}$. Thus $\mathsf{terms}(\pi) \subseteq \mathsf{st}(\mathrm{T}) \cup \mathsf{st}(\mathrm{E} \cup \{\alpha\})$.
- r = trans: Suppose the subproofs of π are π_1 through π_{k-1} with conclusions $t_1 \bowtie t_2$ through $t_{k-1} \bowtie t_k$ respectively, and $\alpha = t_1 \bowtie t_k$. Since π is a normal proof, no two adjacent premises of r are obtained by cons, and no premise of r is obtained by trans. The following cases arise.
  - $r \in \{t_1, t_k\}$. In this case, $r \in \mathsf{st}(\alpha)$.
  - $r \in \mathsf{terms}(\pi_i)$, where π_i does not end in cons. By (F1), cons does not occur in π_i. By IH, $r \in \mathsf{st}(\mathrm{T}) \cup \mathsf{st}(\mathrm{E})$.
  - $r \in \mathsf{terms}(\pi_i)$, where π_i ends in cons, and $1 < i < k-1$. Both π_{i-1} and π_{i+1} end in a rule other than cons, by normality of π. So, by (F1), cons does not occur in π_{i-1} and π_{i+1}, and $t_i, t_{i+1} \in \mathsf{terms}(\pi_{i-1}) \cup \mathsf{terms}(\pi_{i+1}) \subseteq \mathsf{st}(\mathrm{T}) \cup \mathsf{st}(\mathrm{E})$ (by IH on π_{i-1} and π_{i+1}). So, by applying IH on π_i, we get $r \in \mathsf{st}(\mathrm{T}) \cup \mathsf{st}(\mathrm{E} \cup \{t_i \bowtie t_{i+1}\}) \subseteq \mathsf{st}(\mathrm{T}) \cup \mathsf{st}(\mathrm{E})$.
  - $r \in \mathsf{terms}(\pi_1)$, where π_1 ends in cons. By normality of π, we see that π_2 ends in a rule other than cons. So cons does not occur in π_2. By IH on π_2, $t_2 \in \mathsf{terms}(\pi_2) \subseteq \mathsf{st}(\mathrm{T}) \cup \mathsf{st}(\mathrm{E})$. By IH on π_1, $r \in \mathsf{st}(\mathrm{T} \cup \{t_1, t_2\}) \cup \mathsf{st}(\mathrm{E}) \subseteq \mathsf{st}(\mathrm{T}) \cup \mathsf{st}(\mathrm{E} \cup \{\alpha\})$.
  - $r \in \mathsf{terms}(\pi_{k-1})$, where π_{k-1} ends in cons. The proof is similar to the above.
- r = proj: Let $\alpha = t \bowtie u$, got from a proof π' with conclusion $a \bowtie b$. Since π is normal, cons does not occur in π (or in π'). By IH, $a, b \in \mathsf{terms}(\pi') \subseteq \mathsf{st}(\mathrm{T}) \cup \mathsf{st}(\mathrm{E})$. Since $t, u \in \mathsf{st}(\{a, b\})$, we have $\mathsf{terms}(\pi) \subseteq \mathsf{st}(\mathrm{T}) \cup \mathsf{st}(\mathrm{E})$.
- r = prom: α is $t \bowtie u$, and the immediate subproof π' proves $t \twoheadleftarrow [u]$. π' does not contain cons, and so by IH, $\mathsf{terms}(\pi) = \mathsf{terms}(\pi') \subseteq \mathsf{st}(\mathrm{T}) \cup \mathsf{st}(\mathrm{E})$. Note that $\mathsf{lists}(\pi) \subseteq \mathsf{lists}(\pi') \cup \{[u]\}$, so the statement about lists is also true.
- r = wk: Let π' be the immediate subproof. The result follows from IH and the fact that $\mathsf{lists}(\pi) = \mathsf{lists}(\pi') \cup \mathsf{lists}(\alpha)$.
- r = int: All terms in the conclusion appear in some proper subproof, so the statement on terms follows by IH. None of the subproofs ends in int or wk (and does not contain cons). Thus $\mathsf{lists}(\pi') \subseteq \mathsf{lists}(\mathrm{E}) \cup \{[n] \mid n \in \mathsf{st}(\mathrm{T}) \cup \mathsf{st}(\mathrm{E})]$, for every subproof π'. It follows that $\mathsf{lists}(\pi) \subseteq \mathsf{lists}(\mathrm{E} \cup \{\alpha\}) \cup \{[n] \mid n \in \mathsf{st}(\mathrm{T}) \cup \mathsf{st}(\mathrm{E} \cup \{\alpha\})\}$.
- r = subst: Let the major premise be $t \twoheadleftarrow l$ and the minor premise be $t \bowtie u$. Both t, u are from $\mathcal{V} \cup \mathcal{N}$, and thus are in $\mathsf{st}(\mathrm{T}) \cup \mathsf{st}(\mathrm{E})$. The result follows from IH.
- r = say: Let the major premise be β and the minor premise be sk_a. Since $\mathrm{T} \vdash_{dy} sk_a$, $sk_a \in \mathsf{st}(\mathrm{T})$. And $\mathsf{terms}(\pi) \subseteq \mathsf{st}(\mathrm{T}) \cup \mathsf{st}(\mathrm{E}) \cup \mathsf{st}(\beta) \cup \{pk_a\} \subseteq \mathsf{st}(\mathrm{T}) \cup \mathsf{st}(\mathrm{E} \cup \{\alpha\})$. ⊣