Protocol Insecurity with Assertions

R Ramanujam
The Institute of Mathematical Sciences, Chennai
jam@imsc.res.in

Vaishnavi Sundararajan
University of California Santa Cruz
vasundar@ucsc.edu

S P Suresh
Chennai Mathematical Institute
spsuresh@cmi.ac.in

February 10, 2022

Abstract
In the study of symbolic verification of cryptographic protocols, a central result due to [17] is that the insecurity problem (deciding whether a protocol admits an execution which leaks a designated secret to the intruder) for security protocols with finitely many sessions 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. They prove this by analyzing how variables used in the protocol specification can be used in different contexts by the intruder. However, when we consider protocols where, in addition to terms, some logical statements about the terms (called assertions in [15]) are also communicated, the analysis of the insecurity problem becomes tricky. In this paper we consider the insecurity problem for protocols with a class of assertions that includes equality on terms and existential quantification. The intruder can potentially exploit the fact that witnesses for existential quantifiers may be unbounded, and obtaining small witness terms while maintaining equality proofs complicates the analysis considerably. We use a notion of well typed equality proofs that helps in obtaining the reduction, and show that this problem is in NP.

1 Introduction
A common scenario in cryptographic protocols is one where an agent receives a message whose structure they do not know in full (e.g. the message might be an encryption whose decryption key the agent does not know). Even though the structure is not known, the agent might be expected to send some modification of the message (e.g. sign the message). An intruder can make use of this, and at runtime, can force honest agents to handle arbitrarily large messages. In the symbolic modelling of protocols, such unknown messages are represented by variables, and in order to solve the insecurity problem (i.e. check if the protocol
admits an execution that leaks a secret to the intruder), one has to consider different instantiations of these variables. This, along with the fact that the intruder can force agents to participate in parallel sessions, contributes to the undecidability of the insecurity problem [3, 9, 13]. Even when one only searches for attacks with a bounded number of sessions, the problem is hard. Messages occurring in attacks could have potentially unbounded sizes (compared to terms in the specification), and search can thus become intractable.

[17] solves this problem for a bounded number of sessions in a clever way. The key idea is a small substitution property, which states that if there is any attack under a substitution \( \sigma \) involving arbitrarily large terms, there is one under a ‘small’ substitution \( \tau \). This attack is such that the intruder can derive the same secret and the size of all messages transmitted is bounded by a polynomial in the size of the specification. Hence they show that the insecurity problem with bounded sessions has an NP algorithm.

This symbolic approach to the modelling and analysis of cryptographic protocols is called the Dolev-Yao model [8]. In this approach, messages are modelled as terms in an algebra. The intruder controls the network, and can see, block, inject, and redirect messages, as well as derive new terms from old according to set rules (but cannot break cryptography). Various extensions to the basic Dolev-Yao model have been proposed, and automated tools have been developed based on them [1, 5, 9, 10, 14].

One such extension involves terms that represent zero knowledge proofs (ZKPs), along with message terms [4]. The ZKPs ensure that an honest agent who might not know the structure of a term, is still assured that it satisfies some required property. For instance, an election authority might need an assurance that the message is an encryption of a valid vote, without being able to see the vote. [15] suggests a general approach for such extensions, where assertions are communicated along with terms: assertions are logical formulas involving equality of terms, existential quantification, conjunction, and disjunction. This framework also allows logical reasoning involving these assertions. [15] shows how to express example protocols (the FOO [11] and Helios [2] e-voting protocols) and specify security properties.

For example, consider a toy e-voting protocol, in which a voter tells an authority that their vote is one of two allowed values \( a \) and \( b \), without revealing the vote. This can be specified by communicating an encrypted term \( \text{senc}(v, k) \) where \( v \) is the vote, and \( k \) is a fresh key, along with an assertion of the form \( \exists xy. \{ \text{senc}(v, k) = \text{senc}(x, y) \land (x = a \lor x = b) \} \). This generalization enables interesting reasoning with these logical statements. For instance, if the voter communicates both \( \exists xy. \{ \text{senc}(v, k) = \text{senc}(x, y) \land (x = a \lor x = b) \} \) and \( \exists xy. \{ \text{senc}(v, k) = \text{senc}(x, y) \land (x = a \lor x = c) \} \), where \( b \neq c \), the intruder can deduce that the vote is actually \( a \).

With assertions, the problem of unbounded substitution of the “intruder variables” occurring in the protocol specification remains. But there is another reason why techniques to deal with unbounded substitutions are relevant for assertions: to derive a quantified assertion \( \exists x. \alpha \), one might need to derive \( \alpha(t) \) for a “witness” \( t \). A priori, there is no bound on the size of \( t \), and this complicates proof search. In this paper, we show that the techniques from [17] can be adapted and extended to solve the insecurity problem for a subclass of the assertions in [15] – containing equality, conjunction, and existential quantification. In Appendix D,
we describe how the results can be extended to handle a restricted form of disjunction (list membership) as well. We first present an overview of the technique used by [17].

### 1.1 Attacks involving non-atomic terms

The protocol specification mentions variables that stand for terms instantiated by the intruder in the course of an execution. The intruder’s choices are formalized as a substitution $\sigma$. Consider a classic example as follows.

**Example 1.1.** A sends to B the term $\{A, \{n\}_{pk(B)}\}_{pk(B)}$, and B responds with $\{n\}_{pk(A)}$ (where $\{t\}_{pk(u)}$ represents the encryption of $t$ with the public key of $u$). $n$ is a fresh name generated (and intended to be secret) by A, which will be instantiated by a new random value every time the protocol is executed.

Since the intruder controls the network, there is no guarantee that messages sent out will be received by the intended recipient at all. So the model splits communications into sends and receives. Recipients expect messages to conform to some patterns, with some parts of a message being specific terms ($pk(B)$ being the key in the term expected by $B$, for example), while other parts with unknown structure being modelled by variables ($A$ and $n$ in the term expected by $B$). So the B “role” in Example 1.1 looks as follows: B receives a term of the form $\{x, \{y\}_{pk(B)}\}_{pk(B)}$, and then sends out $\{y\}_{pk(x)}$. This protocol admits an attack on the secrecy of a value that A thinks is secret between her and B.

**Example 1.2.** A sends out a term $\{A, \{m\}_{pk(B)}\}_{pk(B)}$. The intruder $I$ intercepts this term so that $B$ does not receive it, and then sends to $B$ the term $\{I, \{A, \{m\}_{pk(B)}\}_{pk(B)}\}_{pk(B)}$. Observe that $B$ is only expecting a variable encrypted in his public key as the second component of the pair, and the intruder substitutes this variable with the term they have obtained from A’s communication.

$B$ responds by encrypting the second component of the pair using the public key of the first component, i.e. B sends out $\{A, \{m\}_{pk(I)}\}_{pk(I)}$. I can now decrypt this term, since it is encrypted in their public key.

Now, I has access to $\{m\}_{pk(B)}$, but they cannot decrypt this to get at $m$. Since the first action in $B$’s role is a receive, I once again sends out a message to $B$, triggering a new instance of $B$’s role. This message is $\{I, \{m\}_{pk(B)}\}_{pk(B)}$. $B$ follows the protocol and sends back $\{m\}_{pk(I)}$, at which point $I$ has access to $m$ by decryption. While $m$ has already been “leaked”, I could, to finish the protocol with $A$, masquerade as $B$, and send back to $A$ the expected term $\{m\}_{pk(A)}$.

---

1This requires no decryption. Also note that anyone – including $I$ – can encrypt using $B$’s public key and communicate with $B$. 

The attack, therefore, looks like this

\[ A \rightarrow (I) : \{ A, \{ m \}_{pk(B)} \}_{pk(B)} \]
\[ I \rightarrow B : \{ I, \{ A, \{ m \}_{pk(B)} \} _{pk(B)} \}_{pk(B)} \]
\[ B \rightarrow I : \{ A, \{ m \} _{pk(B)} \}_{pk(I)} \]
\[ I \rightarrow B : \{ I, \{ m \}_{pk(B)} \}_{pk(B)} \]
\[ B \rightarrow I : \{ m \}_{pk(I)} \]
\[ (I) \rightarrow A : \{ m \}_{pk(A)} \]

This attack involves two instances (sessions) of the B-role. Rather than think of these as two separate substitutions for the variables \(x\) and \(y\) appearing in B’s role, we can make two copies of the role with variable names \(x_1, y_1\), and \(x_2, y_2\), and consider the single substitution

\[ [x_1 \mapsto I, y_1 \mapsto \{ A, \{ m \}_{pk(B)} \}_{pk(B)}, x_2 \mapsto I, y_2 \mapsto m] \]

Such type flaws are well-studied in the literature [12, 16]. The key takeaway is that attacks might involve the use of non-atomic terms for variables (like for \(x_1\) here).

The important insight of [17] is that we can distinguish variables based on whether their instantiations match non-variable patterns in the specification or not. We call variables of the latter kind “minimal”. For an efficient intruder, every variable is of the former kind, as in Example 1.2. But this need not always be the case, and therefore, the attack space is in general unbounded. [17] shows that terms which do not unify with any non-variable pattern in the specification can be replaced by a constant. This cascades throughout the substitution, yielding a different attack, which admits bounds on the size of the substituted terms.

### 1.2 Contributions of this paper

There are two sources of unboundedness when we consider the insecurity problem for assertions: the substitution for intruder variables, and witnesses for quantified variables. Importantly, these two might interact. When we simulate a large substitution \(\sigma\) (for the intruder variables) with a small one \(\tau\), the witnesses might change as well. Even with these changed witnesses, we need to derive the same equality assertions under both \(\sigma\) and \(\tau\). The following example illustrates the difficulty with preserving equalities.

**Example 1.3.** Suppose we have terms \(t_1, u_1\) and a proof \(\pi\) of an equality assertion \(\text{eq}(t_1, u_1)\) from \(\sigma(T; E)\) where \(T\) is a set of terms, and \(E\) is a set of equality assertions, and we want to derive a “corresponding” equality assertion from \(\tau(T; E)\).

The most straightforward strategy is to follow the structure of \(\pi\). Suppose, however, that the last rule of \(\pi\) is a “projection” that derives \(\text{eq}(t_1, u_1)\) from \(\text{eq}((t_1, t_2), (u_1, u_2))\), where \((t_1, t_2) = \sigma(x)\) for some minimal intruder variable \(x\). Following [17], \(\tau\) would map \(x\) to a constant. If \(\tau\) does not map \((u_1, u_2)\) to the same constant, the equality does not hold under \(\tau\) and cannot be derived anymore.
With this motivation, we introduce a notion of well-typed derivations. In such proofs, we identify which terms and equalities match patterns in the protocol, and which do not. This analysis of proofs allows us to translate proofs under $\sigma$ to proofs under $\tau$, a corresponding “small” substitution.

1.3 Organization of the paper

In Section 2, we introduce the syntax and derivation systems for terms and the restricted subclass of assertions that we consider. Section 3 contains an outline of the derivability problem for assertions, which is an important ingredient in solving the insecurity problem. Section 4 defines various notions required for tackling the insecurity problem, which is solved in Section 5.

2 Syntax and proof systems

2.1 Terms

We assume a countable set of basic terms $\mathcal{B}$, partitioned into two subsets, names $\mathcal{N}$ and keys $\mathcal{K}$. The set of public names (agent names, constants etc) is denoted by $\mathcal{N}_{pub} \subset \mathcal{N}$. We assume a set of agents $\mathcal{A} \subset \mathcal{N}_{pub}$, including the intruder $I$. We also have a countable set of variables $\mathcal{V}$. We assume a set of operators $\mathcal{F}$. The grammar for generating terms is given by

$$t \in \mathcal{T} ::= m \mid f(t_1, \ldots, t_n)$$

where $m \in \mathcal{B} \cup \mathcal{V}$, $t_1, \ldots, t_n \in \mathcal{T}$, and $f \in \mathcal{F}$ is an $n$-ary operator. The set of ground terms (without variables) is denoted by $\mathcal{G} \subseteq \mathcal{T}$.

Each operator $f \in \mathcal{F}$ has constructor rules and destructor rules. Constructor rules have $f(t_1, \ldots, t_n)$ in the conclusion, and $t_1$ through $t_n$ appear among the premises. Destructor rules have $f(t_1, \ldots, t_n)$ as the major premise, and the terms appearing in the minor premises are subterms of $f(t_1, \ldots, t_n)$, with the conclusion in $\{t_1, \ldots, t_n\}$. These rules together make up the derivation system. In Table 1 we show an example derivation system for the term algebra with operators for pairing (constructor: $\text{pair}$; destructors: $\text{fst}$, $\text{snd}$) and symmetric encryption (constructor: $\text{senc}$; destructor: $\text{sdec}$). Other operators and rules can be added to the system as necessary. The proof system mentions sequents of the form $X \vdash t$, to be read as “$X$ derives $t$”. $X \cup \{t\} \subseteq_{\text{fin}} \mathcal{T}$. $X \vdash_{dy} t$ denotes the fact that there is a proof of $X \vdash t$ using the rules in Table 1.

The set of subterms of a term $t$ is denoted by $\text{st}(t)$, and defined as usual. The set of names (resp. variables) appearing in $t$ is denoted by $\text{names}(t)$ (resp. $\text{vars}(t)$).

A normal proof is one where the conclusion of a constructor rule is not the major premise of a destructor rule.

**Theorem 2.1** (Normalization and subterm property for $\vdash_{dy}$). If $X \vdash_{dy} t$, there is a normal proof of $X \vdash t$. If a sequent $X \vdash u$ appears in a normal proof $\pi$ ending in $X \vdash t$, then $u \in \text{st}(X \cup \{t\})$. Further, if $\pi$ ends in a destructor rule, $u \in \text{st}(X)$.  

5
\[ t \in X \quad \frac{\text{ax}}{X \vdash t} \quad X \vdash t_0 \quad X \vdash t_1 \quad X \vdash \text{pair}(t_0, t_1) \quad X \vdash \text{fst} \quad X \vdash t_0 \quad X \vdash \text{snd} \quad X \vdash t_1 \quad X \vdash \text{pair}(t_0, t_1) \quad X \vdash \text{pair}(t_0, t_1) \quad X \vdash \text{pair}(t_0, t_1) \quad X \vdash \text{pair}(t_0, t_1) \quad X \vdash \text{pair}(t_0, t_1) \]

\[ n \in \mathcal{N}_{\text{pub}} \quad \frac{\text{pb}}{X \vdash n} \quad X \vdash t \quad X \vdash k \quad X \vdash \text{senc}(t, k) \quad X \vdash \text{senc}(t, k) \quad X \vdash k \quad X \vdash \text{sdec}(t, k) \quad X \vdash \text{sdec}(t, k) \]

Table 1: Proof system for terms with \text{pair} and \text{senc}. \( X \subseteq \text{fin} \mathcal{T}; \, t, t_0, t_1 \in \mathcal{T}; \, k \in \mathcal{K} \cup \mathcal{V} \).

Using the above, one can provide a PTIME algorithm to check whether \( X \vdash_{dy} t [9, 17] \).

### 2.2 Assertions

Assertions are first-order statements, built using equality, conjunction, and existential quantification. We fix a set of variables \( \mathcal{V}_q \subseteq \mathcal{V} \) which are used for quantification (we distinguish these from the set of intruder variables \( \mathcal{V}_i \subseteq \mathcal{V} \setminus \mathcal{V}_q \)). The syntax is as below, with \( t, t' \in \mathcal{T} \), and \( x \in \mathcal{V}_q \).

\[ \alpha, \beta := \text{eq}(t, t') \mid \alpha \land \beta \mid \exists x. \alpha \]

We denote by \( \text{vars}(\alpha) \) (resp. \( \text{fv}(\alpha), \text{bv}(\alpha) \)) the set of all (resp. free, bound) variables of \( \alpha \), and by \( \text{names}(\alpha) \) the set of all names appearing in (terms in) \( \alpha \). The set of subformulas of \( \alpha \) (denoted \( \text{sf}(\alpha) \)) is defined as the smallest set of assertions \( Z \) such that:

1. \( \alpha \in Z \),
2. if \( (\beta \land \gamma) \in Z \), then \( \{\beta, \gamma\} \subseteq Z \), and
3. if \( \exists x. \beta \in Z \), then \( \beta \in Z \).

A substitution \( \sigma \) is a partial map with finite support from \( \mathcal{V} \) to \( \mathcal{T} \), and can be lifted to \( \mathcal{T} \) inductively. For an assertion \( \alpha \), \( \sigma(\alpha) \) is obtained by replacing all \( x \in \text{fv}(\alpha) \) by \( \sigma(x) \).

**Example 2.2.** Suppose \( A \) sends to \( B \) the terms \( \text{senc}(m, k) \) and \( \text{senc}(m', k') \), where \( B \) does not know \( k \) or \( k' \). She can send \( \exists xyzw. \beta \), where \( \beta \) is given by:

\[ \text{eq}(\text{senc}(m, k), \text{senc}(x, y)) \land \text{eq}(\text{senc}(m', k'), \text{senc}(z, w)) \]

to tell \( B \) that these terms are encryptions. She can reveal further structure, for instance, that the two terms inside the encryptions are the same, by adding \( \text{eq}(x, z) \) to the conjunction, or that the keys are the same, by adding \( \text{eq}(y, w) \).

#### 2.2.1 Structured assertions

In Example 2.2, the assertion contains \( \text{eq}(\text{senc}(m, k), \text{senc}(x, y)) \). However, the send should only reveal \( \text{senc}(m, k) \), but not \( m \) or \( k \). We thus need to identify the terms made public by an assertion. In addition, note that while we intend variables to stand for terms with no known structure, there is nothing in the assertion syntax itself that prevents an agent from equating
a variable with a non-atomic term, thereby revealing structure. While a variable can unify with a non-atomic term under a substitution, we do not want such equalities to appear in protocol specifications. Thus, we introduce a structuring restriction which allows only equalities between two atomic terms (names or variables), and equalities between a pattern \( pat \) and a term \( t \) which has at least as much structure as \( pat \). While this yields a strict subclass of assertions, it suffices to capture many scenarios, including example protocols in [15].

**Definition 2.3** (Structured assertions). A structured assertion is an assertion of the form

\[
\alpha := \exists x_1 \ldots x_n \{ \text{eq}(\text{pat}, t) \land \text{eq}(r_1, s_1) \land \ldots \land \text{eq}(r_k, s_k) \}
\]

which satisfies the following conditions:

1. \( \text{names}(\text{pat}) = \emptyset \),
2. \( \text{vars}(\text{pat}) = \{x_1, \ldots, x_n\} \),
3. \( \text{vars}(t) \cap \text{vars}(\text{pat}) = \emptyset \),
4. all \( r_i \) and \( s_i \) are in \( \text{vars}(\text{pat}) \cup \mathcal{N} \), and
5. \( t \) is an instance of \( \text{pat} \), i.e. there is a \( \mu \) with \( \text{dom}(\mu) = \text{vars}(\text{pat}) \) s.t. \( \mu(\text{pat}) = t \).

The set of public terms of \( \alpha \) is \( \text{pubs}(\alpha) := \{t\} \cup \bigcup_{i \leq k} \text{names}(r_i, s_i) \).

**Example 2.4.** The assertion \( \alpha \) in Example 2.2 is not structured. The structured form for the same is \( \exists xyzw. \beta' \), where \( \beta' = \text{eq}(((\text{senc}(m, k), \text{senc}(m', k'))), (\text{senc}(x, y), \text{senc}(z, w))) \) with \( \text{pubs}(\alpha) = \{(\text{senc}(m, k), \text{senc}(m', k'))\} \).

**Definition 2.5** (Clean and Sanitized). An assertion \( \alpha \) is said to be clean if no variable is bound by distinct quantifiers occurring in \( \alpha \), and \( \text{fv}(\alpha) \cap \text{bv}(\alpha) = \emptyset \).

For \( S \subseteq \mathcal{T} \) and a set \( A \) of assertions, \((S; A)\) is sanitized if

1. every \( \alpha \in A \) is clean and structured,
2. for \( \alpha, \beta \in A \), \( \text{bv}(\alpha) \cap \text{bv}(\beta) = \emptyset \),
3. \( (\text{vars}(S) \cup \text{fv}(A)) \cap \text{bv}(A) = \emptyset \), and
4. for all \( \alpha \in A \) and \( t \in \text{pubs}(\alpha) \), \( S \vdash_{dy} t \).

One can always sanitize any \((S; A)\) by adding to \( S \) the public terms of assertions in \( A \) and renaming bound variables wherever necessary. We will hereafter assume that all assertions occurring in a protocol are structured, and that \((S; A \cup \{\alpha\})\) is sanitized for all \((S; A) \vdash_{a} \alpha \) that we consider.
2.2.2 Abstractability

To be able to substitute a variable $x$ inside a term, an agent $A$ needs to be able to access the position of $x$. $A$ should be able to deconstruct the term down to that position, replace $x$, and construct the whole term back. This depends on other terms $A$ has access to. This notion is key to our rules for equality and exists introduction, and is formalized as abstractability. We will view terms as trees, with $\mathbb{P}(t)$ denoting the set of positions in the tree of the term $t$, and defined to be a subset of $\mathbb{N}^*$, where $\varepsilon$ denotes the empty word in $\mathbb{N}^*$.

**Definition 2.6.** We define the term positions of an assertion $\alpha$, denoted $\mathbb{P}(\alpha)$, as follows:

- $\mathbb{P}(\text{eq}(t, t')) = \{0 \cdot p \mid p \in \mathbb{P}(t)\} \cup \{1 \cdot p \mid p \in \mathbb{P}(t')\}$
- $\mathbb{P}(\alpha \land \beta) = \{0 \cdot p \mid p \in \mathbb{P}(\alpha)\} \cup \{1 \cdot p \mid p \in \mathbb{P}(\beta)\}$
- $\mathbb{P}(\exists x. \alpha) = \{0 \cdot p \mid p \in \mathbb{P}(\alpha)\}$

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

**Definition 2.7.** Let $S \cup \{t\} \subseteq \mathcal{T}$ and $p \in \mathbb{P}(t)$. Define $\mathbb{Q}_p = \{\varepsilon\} \cup \{q \cdot i \in \mathbb{P}(t) \mid q \text{ is a proper prefix of } p\}$. We say that $p$ is an abstractable position of $t$ w.r.t. $S$ if $S \vdash_{dy} t|_q$ for all $q \in \mathbb{Q}_p$.

The set of abstractable positions of $t$ w.r.t. $S$ is denoted $\mathbb{A}(S, t)$.

**Example 2.8.** Consider a term $t = f(g(h(a, b, c), t_0), t_1, t_2)$. The set $\mathbb{Q}_{001} = \{\varepsilon, 0, 1, 2, 00, 01, 000, 001, 002\}$, and $001 \in \mathbb{A}(S, t)$ iff $S$ derives all the subterms of $t$ at positions in $\mathbb{Q}_{001}$.

**Definition 2.9.** For a set $S$ of terms and an assertion $\alpha$, the abstractable positions of $\alpha$ w.r.t. $S$ (denoted $\mathbb{A}(S, \alpha)$) are as follows.

- $\mathbb{A}(S, \text{eq}(t, u)) = \{0 \cdot p \mid p \in \mathbb{A}(S, t)\} \cup \{1 \cdot p \mid p \in \mathbb{A}(S, u)\}$
- $\mathbb{A}(S, \alpha \land \beta) = \{0 \cdot p \mid p \in \mathbb{A}(S, \alpha)\} \cup \{1 \cdot p \mid p \in \mathbb{A}(S, \beta)\}$
- $\mathbb{A}(S, \exists x. \alpha) = \{0 \cdot p \mid p \in \mathbb{A}(S \cup \{x\}, \alpha)\}$

**Example 2.10.** Let $\alpha = \exists b. \text{eq}(\text{senc}(m, b), \text{senc}(m, k))$ and $S = \{m\}$. To derive the assertion $\beta = \exists ab. \text{eq}(\text{senc}(a, b), \text{senc}(m, k))$ we need to use exists introduction on $\alpha$. Note that we want to quantify out $m$ at position 000 from $\alpha$, and so we need $S$ to derive $\text{senc}(m, b)$. Therefore, in Definition 2.9, we add the quantified variable to $S$ when we consider derivabilities for an existential assertion, and $\mathbb{A}(S, \alpha) = \{00, 000, 001\}$.

**Observation 2.11.** Let $\alpha$ be an assertion and $S \cup \{t\} \subseteq \mathcal{T}$.

1. Suppose $x \neq y$, $\{x, y\} \subseteq \text{fv}(\alpha)$, $\{x, y\} \cap \text{vars}(S \cup \{t\}) = \emptyset$, and $S \vdash_{dy} t$. Let $P = \mathbb{P}_x(\alpha)$ and $\beta = \alpha[t|_P]$. If $P \subseteq \mathbb{A}(S \cup \{x\}, \alpha)$, then $\mathbb{P}_y(\alpha) \subseteq \mathbb{A}(S \cup \{x\}, \alpha)$ iff $\mathbb{P}_y(\beta) \subseteq \mathbb{A}(S, \beta)$.

2. For $x \in \text{fv}(\alpha)$, there exists $T_x \subseteq \text{st}(\alpha)$ s.t. $\mathbb{P}_x(\alpha) \subseteq \mathbb{A}(S, \alpha)$ iff $S \vdash_{dy} t$ for all $t \in T_x$. 

8
We denote by $S$ any sanitized sets, and the quantified variable proofs (NIZKPs) can be backed by an operational model where assertions are non-interactive zero-knowledge. An agent can always forward an NIZKP received from some other agent, but they cannot gain access to the term the NIZKP talks about, or build upon the NIZKP without knowing the underlying term. Note that the restriction on witnesses does not preclude the renaming of bound variables, as shown below. Suppose $P \subseteq \mathbb{P}_x(\text{eq}(r, s)) \cap \mathbb{A}(S \cup \{x\}, \text{eq}(r, s))$. 

2.2.3 Derivation system

We present the proof system for assertions in Table 2. The sequents are of the form $S; A \vdash \alpha$, where $S \subseteq_{\text{fin}} \mathbb{T}$, and $A \cup \{\alpha\}$ is a finite set of assertions, such that $(S; A \cup \{\alpha\})$ is sanitized. We denote by $S; A \vdash_{\alpha} \alpha$ the fact that there is a derivation of $S; A \vdash \alpha$ using this system.

Table 2: Derivation system $\vdash_{\alpha}$ for assertions. $\text{eq}$ stands for $\{0, 1\} \subseteq \mathbb{A}(S, \text{eq}(f(t_1, \ldots, t_r), f(u_1, \ldots, u_r)))$. $\star$ states that $P \subseteq \mathbb{P}_x(\text{eq}(r, s)) \cap \mathbb{A}(S \cup \{x\}, \text{eq}(r, s))$. $\dagger$ states that $P \subseteq \mathbb{P}_x(r) \cap \mathbb{A}(S \cup \{x\}, r)$ and either $t \in S \cap \mathcal{V}_q$ or $st(t) \cap \mathcal{V}_q = \emptyset$. $\dagger$ stands for $x \notin \text{fv}(S) \cup \text{fv}(A) \cup \text{fv}(\gamma)$.

| Rule | Premise | Conclusion |
|------|---------|------------|
| $S; A \vdash \alpha \{\alpha\} \vdash \alpha$ | $S \vdash_{\text{eq}} t$ | $S; A \vdash_{\text{eq}}(f(t_1, \ldots, t_r), f(u_1, \ldots, u_r))$ |
| $S; A \vdash \alpha \{\alpha\} \vdash \alpha$ | $S; A \vdash_{\text{eq}}(t, t)$ | $S; A \vdash_{\text{eq}}(t, u)$ |
| $S; A \vdash \alpha \{\alpha\} \vdash \alpha$ | $S; A \vdash_{\text{eq}}(r, s)[t]_P$ | $S; A \vdash_{\text{eq}}(r, s)[u]_P$ |
| $S; A \vdash \alpha \{\alpha\} \vdash \alpha$ | $S; A \vdash_{\text{eq}}(r, s)[t]_P$ | $S; A \vdash_{\text{eq}}(r, s)[u]_P$ |
| $S; A \vdash \alpha \{\alpha\} \vdash \alpha$ | $S; A \vdash_{\text{eq}}(r, s)[t]_P$ | $S; A \vdash_{\text{eq}}(r, s)[u]_P$ |
| $S; A \vdash \exists x. a$ | $S \cup \{x\}; A \cup \{\alpha\} \vdash \gamma$ | $S; A \vdash \gamma$ |

In $\exists e$, we use $x$ itself as the eigenvariable, instead of some fresh $z$, since we only consider sanitized sets, and the quantified variable $x$ does not appear on the LHS.
3 The derivability problem for assertions

In this section, we analyze the derivability problem for assertions in terms of a simpler proof system. All assertions we consider are structured.

Definition 3.1 (Derivability problem for $\vdash_a$). Given a set $S$ of terms, and a set $A \cup \{\alpha\}$ of assertions such that $(S; A \cup \{\alpha\})$ is sanitized, check whether $S; A \vdash_a \alpha$.

3.1 Down-closures and moving to $\vdash_{\text{core}}$

The $\vdash_a$ system enjoys the following “left” properties which follow from the elimination rules for $\land$ and $\exists$. They allow us to create “down closures”, i.e. an LHS composed of only equality formulas. A normal proof from such an LHS will not involve any elimination rules.

Lemma 3.2. $S; A \cup \{\alpha \land \beta\} \vdash_a \gamma$ iff $S; A \cup \{\alpha, \beta\} \vdash_a \gamma$.

Lemma 3.3. Let $S, A, \exists x.\alpha$ and $\gamma$ be such that $x \in \mathcal{F}_S(A(S))$ and $\mathbb{P}_x(\alpha) \subseteq A(S \cup \{x\}, \alpha)$. Then $S; A \cup \{\exists x.\alpha\} \vdash_a \gamma$ iff $S \cup \{x\}; A \cup \{\alpha\} \vdash_a \gamma$.

Definition 3.4. $(T; B)$ is down-closed if:

1. if $\text{eq}(t, u) \in B$ then $\text{eq}(u, t) \in B$,
2. if $\beta \land \gamma \in B$ then $\{\beta, \gamma\} \subseteq B$, and
3. if $\exists x.\alpha \in B$ then $x \in T$ and $\alpha \in B$.

For any set $B$ of assertions, $\text{trim}(B)$ is the set of all equality assertions in $B$. If $(T; B)$ is down-closed, then $T; \text{trim}(B) \vdash \beta$ for all $\beta \in B$ via a proof using only $\text{ax}, \land$, and $\exists$. The down-closure of $(S; A)$ is $\text{dc}(S; A) = (T; \text{trim}(B))$, where $(T; B)$ is the smallest down-closed set such that $S \subseteq T$ and $A \subseteq B$.

Definition 3.5 (Eq-pure). For $T \subseteq \mathcal{T}$ and $E$ a set of equalities, $(T; E)$ is eq-pure if for every $\text{eq}(t, u) \in E$, $T \vdash_{dy} t$, $T \vdash_{dy} u$ and $\text{eq}(u, t) \in E$. For sanitized $(S; A)$, $\text{dc}(S; A)$ is eq-pure.

Lemmas 3.2 and 3.3 allow us to move to the subsystem $\vdash_{\text{core}}$, which consists of the rules $\text{ax}$, $\text{eq}$, $\text{subst}$, $\text{proj}$, $\land$, and $\exists$. $\vdash_{\text{core}}$ can itself be reduced to the simpler system $\vdash_{\text{eq}}$.

Lemma 3.6. For any $\gamma$, $(S; A) \vdash_a \gamma$ iff $\text{dc}(S; A) \vdash_{\text{core}} \gamma$.

Definition 3.7 (Proof system $\vdash_{\text{eq}}$). For $T \cup \{t, u\} \subseteq \mathcal{T}$ and $E$ a set of equalities, we say that $(T; E) \vdash_{\text{eq}} \text{eq}(t, u)$ iff there is a proof of $(T; E) \vdash \text{eq}(t, u)$ using $\text{ax}$, $\text{eq}$, and $\text{proj}$ as in Table 2, and the rules $\text{cons}$ and $\text{trans}$, as below, with $f \in \mathcal{F}$.

\[
\begin{array}{c}
T; E \vdash \text{eq}(t_1, u_1) \quad \ldots \quad T; E \vdash \text{eq}(t_r, u_r) \\
T; E \vdash \text{eq}(f(t_1, \ldots, t_r), f(u_1, \ldots, u_r)) \quad \text{cons}
\end{array}
\]

\[
\begin{array}{c}
T; E \vdash \text{eq}(t_1, t_2) \quad T; E \vdash \text{eq}(t_2, t_3) \quad \ldots \quad T; E \vdash \text{eq}(t_{r-1}, t_r) \\
T; E \vdash \text{eq}(t_1, t_r) \quad \text{trans}
\end{array}
\]

10
We can show by induction on the structure of ⊢_eq proofs that if an eq-pure \((T; E)\) derives eq\((t, u)\), then \(T \vdash_{dy} t\) and \(T \vdash_{dy} u\). We will often use this property.

**Definition 3.8** (Consistent). A set of equalities \(E\) is consistent if there is a \(\lambda\) s.t. \(\lambda(t) = \lambda(u)\) for each \(\text{eq}(t, u) \in E\).

**Lemma 3.9.** If \((T; E)\) is eq-pure and \(E\) is consistent, then for any terms \(t, u\), \((T; E) \vdash_{core} \text{eq}(t, u)\) iff \((T; E) \vdash_{eq} \text{eq}(t, u)\).

For the forward direction, we can perform an induction on the size of the set of abstractable positions where subst is applied. For the reverse direction, we show that the rules trans and cons can be captured in the ⊢_core system. The detailed proof is presented in Appendix A.

### 3.2 Linking ⊢_core and ⊢_eq

Let \(\alpha = \exists x_1 \ldots x_k\). \(\beta\) be a structured assertion, with \(\beta = \gamma_1 \land \cdots \land \gamma_n\). Let \((T; E)\) be eq-pure and \(\text{vars}(T; E) \cap \text{bv} (\alpha) = \emptyset\). We aim to reduce \((T; E) \vdash_{core} \alpha\) to a set of simpler conditions, involving \(\vdash_{dy}\) and \(\vdash_{eq}\), which involves an analysis of normal proofs of \((T; E) \vdash_{core} \alpha\). Any such proof ends in a series of applications of \(\exists i\). The quantifiers are introduced in the order \(x_k, \ldots, x_1\). So it is convenient to define \(\alpha_j = \exists x_{j+1} \ldots x_k \cdot \beta\), for each \(j \in \{0, \ldots, k\}\). We write \(\alpha_j(s_1, \ldots, s_j)\) to denote \(\alpha\) where each \(x_i\) is replaced by \(s_l\), for \(l \leq j\).

We also define \(P_j = \mathbb{P}_{x_j}(\alpha_j)\). If \(s_1, \ldots, s_{j-1}\) are terms which do not have an occurrence of \(x_j\), then \(\mathbb{P}_{x_j}(\alpha_j(s_1, \ldots, s_{j-1}, x_j)) = P_j\). Also note that for any term \(s_j\), \(\alpha_j(s_1, \ldots, s_j) = \alpha_j(s_1, \ldots, s_{j-1}, x_j)[s_j]_{P_j}\).

Suppose \(\pi\) is a normal proof of \((T; E) \vdash_{core} \alpha\). In \(\pi\), we first derive the equality assertions, combine them using \(\land i\), and then introduce quantifiers. So,

- there exists a \(k\)-tuple \(\bar{t} = t_1, \ldots, t_k\), s.t. \((T; E) \vdash_{core} \gamma_i(\bar{t})\) (i.e. \((T; E) \vdash_{eq} \gamma_i(\bar{t})\), by Lemma 3.9); and

- for \(j \in \{1, \ldots, k\}\), there is an application of \(\exists i\) as below:

\[
\frac{(T; E) \vdash \alpha_j(t_1, \ldots, t_j) \quad T \vdash_{dy} t_j}{(T; E) \vdash \alpha_{j-1}(t_1, \ldots, t_{j-1})} \exists i
\]

with the condition that either \(\text{vars}(t_j) \cap \forall q = \emptyset\) or \(t_j \in T \cap \forall q\) and that \(P_j \subseteq A(T \cup \{x_j\}, \alpha_j(t_1, \ldots, t_{j-1}, x_j))\).

If we are given \(\bar{t}\) satisfying the above conditions, then we can first construct a proof of \(\beta(\bar{t}) = \alpha_k(\bar{t})\) by a series of applications of \(\land i\) on the ⊢_eq-proofs, and then apply the \(\exists i\) instances in order to get a proof of \(\alpha_0 = \alpha\). Thus we have a necessary and sufficient condition

---

2We only consider \((T; E) = dc(S; A)\) for sanitized \((S; A \cup \{\alpha\})\), which guarantees this.

3Note that \(\alpha_0 = \alpha\) and \(\alpha_k = \beta\).
for \((T;E) \vdash_{\text{core}} \alpha\). Further, expanding the definition of \(\mathbb{A}\) for existential formulas, \(P_j \subseteq \mathbb{A}(T \cup \{x_j\}, \alpha_j(t_1, \ldots, t_{j-1}, x_j))\) iff \(P_j \subseteq \mathbb{A}(T \cup \{x_1, \ldots, x_k\}, \alpha_k(t_1, \ldots, t_{j-1}, x_j, \ldots, x_k))\). By applying Observation 2.11 (1) repeatedly, this is equivalent to \(P_j \subseteq \mathbb{A}(T \cup \{x_1, \ldots, x_k\}, \alpha_k)\). Finally, by Observation 2.11 (2), the above is true iff there exists a designated \(S_j \subseteq \text{st}(\alpha_k)\) such that \(\forall r \in S_j, T \cup \{x_1, \ldots, x_k\} \vdash_{\text{dy}} r\).

Letting \(\mu = \{x_j \mapsto t_j \mid j \leq k\}\) and \(S = \bigcup_{j \leq k} S_j\), we get:

**Theorem 3.10.** For eq-pure \((T; E)\) and a clean structured assertion \(\alpha = \exists x_1 \ldots x_k \cdot \beta\) such that \(\text{vars}(T; E) \cap \text{bv}(\alpha) = \emptyset\), there is a set \(S \subseteq \text{st}(\alpha)\) such that \((T; E) \vdash_{\text{core}} \alpha\) iff there is a substitution \(\mu\) with \(\text{dom}(\mu) = \text{bv}(\alpha)\) such that:

- for all \(x \in \text{dom}(\mu)\), \(T \vdash_{\text{dy}} \mu(x)\), and either \(\mu(x) \in T \cap \mathcal{V}_q \backslash \text{bv}(\alpha)\) or \(\text{vars}(\mu(x)) \cap \mathcal{V}_q = \emptyset\);
- \((T; E) \vdash_{\text{eq}} \mu(\gamma)\) for each atomic subformula \(\gamma\) of \(\alpha\); and
- for each \(s \in S\), \(T \cup \{x_1, \ldots, x_k\} \vdash_{\text{dy}} s\).

### 3.3 Normalization for \(\vdash_{\text{eq}}\)

To analyze the \(\vdash_{\text{eq}}\) system, we now introduce a notion of normal proofs.

**Definition 3.11** (Normality for \(\vdash_{\text{eq}}\)). For \(T \subseteq \mathcal{T}\) and a set \(E \cup \{\text{eq}(t, u)\}\) of equalities, we say that a proof \(\pi\) of \(T; E \vdash_{\text{eq}} \text{eq}(t, u)\) is normal if it satisfies:

1. the premise of an eq is only the conclusion of a destructor rule or pb,
2. no premise of a trans is of the form \(\text{eq}(a, a)\), or the conclusion of a trans,
3. adjacent premises of a trans are not conclusions of \(\text{cons}\), and
4. no subproof ending in proj contains \(\text{cons}\).

**Theorem 3.12** (Normalization for \(\vdash_{\text{eq}}\)-proofs). Let \((T; E)\) be eq-pure with consistent \(E\). If \((T; E) \vdash_{\text{eq}} \text{eq}(t, u)\), then there is a normal \(\vdash_{\text{eq}}\)-proof of \((T; E) \vdash \text{eq}(t, u)\).

As we will see in Section 5, \(E\) is consistent for all the sets \((T; E)\) that we consider in the derivability constraints. So we can appeal to Theorem 3.12 and assume that every \(\vdash_{\text{eq}}\)-proof has a corresponding normal proof.

**Theorem 3.13** (Subterm property for \(\vdash_{\text{eq}}\)). Let \(\pi\) be a normal proof of \(T; E \vdash_{\text{eq}} \text{eq}(t, u)\), with normal \(\vdash_{\text{dy}}\) subproofs. Let \(\xi\) be a subproof of \(\pi\) ending in \(T; E \vdash \text{eq}(r, s)\). Then, \(r, s \in \text{st}(T \cup \{t, u\}) \cup \text{st}(E)\). Further, if \(\text{cons}\) does not occur in \(\pi\), then \(r, s \in \text{st}(T) \cup \text{st}(E) \cup \mathcal{N}_{\text{pub}}\).

The proofs for Theorems 3.12 and 3.13 are given in Appendix E. We now state a proposition about normal proofs that will be useful later. The proof is simple and omitted.
Proposition 3.14. Suppose \( \pi \) is a normal proof ending in \text{trans} with \( \pi' \) an immediate subproof. Then, \( \pi' \) does not end in \text{eq}, and if \text{cons} appears in \( \pi' \), then it is the last rule of \( \pi' \).

To decide whether \((T; E) \vdash_{eq} \text{eq}(t, u)\), for eq-pure \((T; E)\), we give a PTIME saturation-based procedure (shown in Appendix C) that utilizes Theorem 3.13 and computes the set

\[
\text{Conseq}(T, E) = \{ \text{eq}(r, s) \mid r, s \in \text{st}(T \cup \{t, u\}) \cup \text{st}(E),
(T; E) \vdash_{eq} \text{eq}(r, s) \}
\]

and then checks whether \( \text{eq}(t, u) \in \text{Conseq}(T, E) \).

4 Protocols, runs and the insecurity problem

There are many ways of presenting security protocols [1, 9, 15, 17]. We present here an adaptation of the model in [17]. A protocol is given by a finite set of roles, each role consisting of a finite sequence of alternating receives and sends (each send is triggered by a receive). These are the actions of honest agents. The Dolev-Yao intruder controls the network, so every message sent is added to the intruder’s knowledge base. Every message received can be assumed to have been either generated or forwarded by the intruder, so every message received must be derived by the intruder. Only structured assertions are communicated – a term \( t \) can be modelled via the assertion \( \alpha_t = \exists w(\text{eq}(w, t)) \) (note that \( \text{pubs}(\alpha_t) = \{t\} \)).

Definition 4.1 (Protocols). A protocol \( Pr \) is a finite set of roles, with each role a finite sequence of receive-send pairs \( \beta_1 \Rightarrow \alpha_1, \ldots, \beta_n \Rightarrow \alpha_n \) corresponding to an agent \( u \in \mathcal{A} \setminus \{I\} \). The \( \beta_i \)s are assertions received by \( u \), and the \( \alpha_i \)s are sent by \( u \).

The \( \alpha_i \)s and \( \beta_i \)s can have bound variables (from \( \mathcal{V}_q \)) as well as free variables (from \( \mathcal{V}_i \)). Instantiating the free variables with appropriate ground terms yields a session. A protocol execution (or run) is obtained by interleaving a finite number of sessions that satisfy some derivability conditions.

Each role/session consists of the receives and sends of a particular honest agent. Some free variables in a role occur first in an \( \alpha_i \). These are controlled by the agent. In a session, they would be instantiated with a name generated by the agent. The free variables which occur first in a \( \beta_j \) are controlled by the intruder, who supplies an instantiation for them, which the agent follows thereafter.

For executions with a bounded number of sessions, we can follow [17], make multiple copies of roles (with distinct variables), and view all sessions as obtained by one substitution \( \sigma \). Further, we substitute fresh names for the variables controlled by honest agents and make them part of the role. Thus, \( \sigma \) only instantiates the “intruder variables”.

We now set up some machinery for handling the derivability constraints on the communicated messages.
Definition 4.2 (Knowledge states, knowledge functions). A knowledge state is a pair $(X; \Phi)$ where $X$ is a finite set of terms and $\Phi$ is a finite set of assertions. A knowledge function is a function $k$ with $\text{dom}(k) = \mathcal{A}$ such that for each $a \in \mathcal{A}$, $k(a)$ is a knowledge state.

Given a knowledge state $(X; \Phi)$ and an assertion $\alpha$, we define update$((X; \Phi), \alpha)$ to be the pair $(X \cup \text{pubs}(\alpha), \Phi \cup \{\alpha\})$.

Definition 4.3 (Protocol runs). A run of a protocol $Pr$ (with one session) is a pair $(\xi, \sigma)$ where:

- $\xi = u_1 : \beta_1 \Rightarrow \alpha_1, \ldots, u_n : \beta_n \Rightarrow \alpha_n$ with each $u_i \in \mathcal{A} \setminus \{I\}$
- $\beta_1 \Rightarrow \alpha_1, \ldots, \beta_n \Rightarrow \alpha_n$ is an interleaving of the roles of $Pr$.
- $\sigma$ is a ground substitution with $\text{dom}(\sigma) = \text{fv}(\xi)$.
- For all $i \leq n$ and $x \in \text{fv}(\alpha_i)$, there is $j \leq i$ s.t. $x \in \text{fv}(\beta_j)$.
- There is a sequence $k_0 \ldots k_n$ of knowledge functions s.t.:
  - $k_0(a) = (X_a; \emptyset)$, where $X_a$ is the set of initial terms known to the agent $a$ (secret keys, shared keys etc).
  - For all $i < n$,
    $$k_{i+1}(a) = \begin{cases} k_i(a) & \text{if } a \neq u_i, a \neq I \\ \text{update}(k_i(a), \beta_i) & \text{if } a = u_i \\ \text{update}(k_i(a), \alpha_i) & \text{if } a = I \end{cases}$$
  - For $i \leq n$, $k_i(u_i) \vdash_a \alpha_i$ and $\sigma(k_{i-1}(I)) \vdash_a \sigma(\beta_i)$.

Note that honest agent derivations $(k_i(u_i) \vdash_a \alpha_i)$ do not depend on accidental unifications with intruder variables under $\sigma$; rather, they hold in the “abstract”. Hereafter, for readability, we denote $\text{dom}(\sigma)$ by $\mathcal{V}_\sigma$.

Definition 4.4 (Secrecy property and attacks). A secrecy property is specified by an assertion $\gamma$, e.g. the encoding of a designated term that the intruder should not access, or an assertion representing more information than the intruder is allowed to know. An attack which violates the secrecy property $\gamma$ is a run of the protocol where $\sigma(k_n(I)) \vdash_a \sigma(\gamma)$.

Definition 4.5 (Insecurity problem). Given a protocol $Pr$ and a designated secret $\gamma$, check whether there exists an attack on $Pr$ violating $\gamma$.

Let $(\xi, \sigma)$ be a run of a protocol $Pr$, with $\xi$, $\sigma$ and $k_0 \ldots k_n$ as in Definition 4.3. We define some notions and state some facts about this run that will be of interest later. Let $i \in \{1, \ldots, n\}$.

(R1) $HE_i$ (resp. $IE_i$) is the set of equalities that occur in $\alpha_i$ (resp. $\beta_i$). $HE_0 := \emptyset$ and $IE_0 := \emptyset$.  

(R2) $HV_i := \text{bv}(\alpha_i)$ and $IV_i := \text{bv}(\beta_i)$. Since we can rename bound variables, we can assume that \{${HV_i, IV_i \mid 1 \leq i \leq n}$\} is a mutually disjoint collection of sets of variables. $HV_0 := \emptyset$ and $IV_0 := \emptyset$.

(R3) There is an $\text{eq}(pat, t)$ in $HE_i$ such that $\text{vars}(pat) = HV_i$, $\text{names}(pat) = \emptyset$, $\text{vars}(t) \cap HV_i = \emptyset$ and $t$ is an instance of $pat$. Similarly for each $\beta_i$, $IE_i$ and $IV_i$.

(R4) $HT_i := \text{pubs}(\alpha_i)$ and $IT_i := \text{pubs}(\beta_i)$. Observe that $\text{vars}(HT_i) \cap HV_i = \emptyset$ and $\text{vars}(IT_i) \cap IV_i = \emptyset$. $HT_0 := \emptyset$ and $IT_0 := \emptyset$. From Definition 4.3, if $x \in \text{vars}(HT_i)$, there is $j \leq i$ s.t. $x \in \text{vars}(IT_j)$.

(R5) $(S_i; D_i) := \text{dc}(k_i(u_i))$, and $(T_i; E_i) := \text{dc}(k_i(I))$. Further $(T_0; E_0) := \text{dc}(k_0(I))$. Note that $S_i \subseteq \bigcup_{1 \leq j \leq i} (IT_j \cup IV_j)$, and $T_i = \bigcup_{1 \leq j \leq i} (HT_j \cup HV_i)$. Hence, for every $t \in S_i \cup T_i$, $\text{vars}(t) \subseteq \text{dom}(\sigma)$ or $t \in \mathcal{Y}_q$.

(R6) $k_i(u_i) \vdash_a \alpha_i$. Thus, $\sigma(k_i(u_i)) \vdash_a \sigma(\alpha_i)$. By Theorem 3.10, there is a $\theta_i$ with $\text{dom}(\theta_i) = HV_i$ s.t. for every $x \in HV_i$, $\sigma(S_i) \vdash_{dy} \theta_i(x)$ and $\theta_i(x) \in \mathcal{Y}_q$ or $\text{vars}(\theta_i(x)) \cap \mathcal{Y}_q = \emptyset$.

(R7) $\sigma(k_{i-1}(I)) \vdash_a \sigma(\beta_i)$. By Theorem 3.10, there is a $\mu_i$ with $\text{dom}(\mu_i) = IV_i$ s.t. for each $x \in IV_i$, $\sigma(T_{i-1}) \vdash_{dy} \mu_i(x)$ and $\mu_i(x) \in \mathcal{Y}_q$ or $\text{vars}(\mu_i(x)) \cap \mathcal{Y}_q = \emptyset$.

(R8) By Theorem 3.10, there exist $U_i \subseteq \text{st}(\alpha_i), V_i \subseteq \text{st}(\beta_i)$ (corresponding to abstractability conditions) s.t.

$$
\begin{align*}
\sigma(S_i) \vdash_{dy} (\sigma \cup \theta_i)(r) & \quad \forall r \in HT_i \\
\sigma(S_i; D_i) \vdash_{eq} (\sigma \cup \theta_i)(\text{eq}(r, s)) & \quad \forall \text{eq}(r, s) \in HE_i \\
S_i \cup \text{bv}(\alpha_i) \vdash_{dy} r & \quad \forall r \in U_i \\
\sigma(T_{i-1}) \vdash_{dy} (\sigma \cup \mu_i)(r) & \quad \forall r \in IT_i \\
\sigma(T_{i-1}; E_{i-1}) \vdash_{eq} (\sigma \cup \mu_i)(\text{eq}(r, s)) & \quad \forall \text{eq}(r, s) \in IE_i \\
T_{i-1} \cup \text{bv}(\beta_i) \vdash_{dy} r & \quad \forall r \in V_i
\end{align*}
$$

(R9) Let $\text{st} = \bigcup_{i \leq n} (\text{st}(S_i) \cup \text{st}(D_i) \cup \text{st}(T_i) \cup \text{st}(E_i))$ and $\text{st}^- = \text{st} \setminus \text{dom}(\sigma)$ and $\text{Substs} = \{\sigma, \mu_i, \theta_i \mid 1 \leq i \leq n\}$.

We obtain the following strengthening of (R6) and (R7) by noting that $\sigma(S_i)$ only has ground terms or variables from $\mathcal{Y}_q$, and thus cannot derive $t$ containing variables from $\mathcal{Y}_i$.

**Proposition 4.6.** For all $x \in \text{dom}(\theta_i)$, $\theta_i(x)$ is ground or $\theta_i(x) \in \mathcal{Y}_q$. For all $x \in \text{dom}(\mu_i)$, $\mu_i(x)$ is ground or in $\mathcal{Y}_q$.

By Prop. 4.6, for all $\lambda \in \text{Substs}$, $\text{vars}(\lambda(v)) \cap \text{dom}(\sigma) = \emptyset$ for all $v \in IV_i$. Also, for all $\lambda, \lambda' \in \text{Substs}$, $\text{dom}(\lambda) \cap \text{dom}(\lambda') = \emptyset$. So $\sigma \cup \lambda = \sigma \circ \lambda$, and we denote it by $\sigma \lambda$.

A priori, there is no bound on the size of $\lambda(x)$ for any $\lambda \in \text{Substs}$ and $x \in \text{dom}(\lambda)$. We will now establish the existence of “small” substitutions $\tau, \theta_1, \ldots, \theta_n, \nu_1, \ldots, \nu_n$ (every term in the range of these substitutions is either atomic or the image of some term from $\text{st}^-$) and which satisfy the conditions for a run (with the same sets $S_i, D_i, T_i, E_i$ etc.). The
only difference is in (R8) in the list above. For these “small” substitutions, we only need to ensure that the following two conditions hold:

\[
\tau(T_{i-1}) \vdash_{eq} \tau\nu_i(r) \quad \text{for all } r \in IT_i \\
\tau(T_{i-1}; E_{i-1}) \vdash_{eq} \tau\nu_i(eq(r, s)) \quad \text{for all } eq(r, s) \in IE_i
\]

This is because the honest agent derivation of each \(\alpha_i\), as well as the derivations for the abstractability conditions, go through even without applying \(\sigma\), and so the same “abstract” derivations are preserved under the new substitutions.

5 Insecurity is in NP

We first show that for every \(i \leq n\), \(D_i\) and \(E_i\) are consistent (according to Definition 3.8) by defining a substitution \(\tilde{\mu}\) which makes the equalities in these sets true. Intuitively, \(\tilde{\mu}(x)\) is the ground term that corresponds to the variable \(x\). Note that the terms in the ranges of any \(\theta_i\) or \(\mu_i\) may contain variables from \(Y_q\). \(\tilde{\mu}(x)\) is the ground term obtained by following the appropriate chain of substitutions.

**Definition 5.1** (\(\tilde{\mu}\)). Define \(\tilde{\mu}\) as follows:

- for all \(x \in dom(\sigma)\), \(\tilde{\mu}(x) := \sigma(x)\);
- for all \(x \in dom(\theta_i)\), assuming that \(\tilde{\mu}(y)\) is already defined for all \(y \in dom(\sigma) \cup dom(\theta_j) \cup dom(\mu_l)\) for \(j < i\) and \(l \leq i\), \(\tilde{\mu}(x) := \tilde{\mu}(\theta_i(x))\);
- for all \(x \in dom(\mu_i)\), assuming that \(\tilde{\mu}(y)\) is already defined for all \(y \in dom(\sigma) \cup dom(\theta_j) \cup dom(\mu_j)\) for \(j < i\), \(\tilde{\mu}(x) := \tilde{\mu}(\mu_i(x))\).

The next two lemmas show that each \(D_i\) and \(E_i\) is consistent and that consistency is preserved by derivations. The proofs are presented in Appendix B.

**Lemma 5.2.** Suppose \(\lambda(r) = \lambda(s)\) for each \(eq(r, s) \in E\), and \(T; E \vdash_{eq} eq(t, u)\). Then \(\lambda(t) = \lambda(u)\).

**Lemma 5.3.** Let \(t, u \in \mathcal{T}\). For any \(i \in \{1, \ldots, n\}\) and any \(l \in \{0, \ldots, n\}\),

1. if \(\gamma \in D_i \cup E_i\), then \(\tilde{\mu}(t) = \tilde{\mu}(u)\).
2. if \(\sigma(T_{i-1}; E_{i-1}) \vdash_{eq} \sigma \mu_i(eq(t, u))\), then \(\tilde{\mu}(t) = \tilde{\mu}(u)\).
3. if \(\sigma(S_i; D_i) \vdash_{eq} \sigma \theta_i(eq(t, u))\), then \(\tilde{\mu}(t) = \tilde{\mu}(u)\).

Compared to [17], we have an extra challenge here. We have to consider multiple substitutions – a \(\sigma\) for the intruder variables, and one \(\mu_i\) or \(\theta_i\) for each quantified assertion. It would be convenient if we could relate the \(\mu_i\)s and \(\theta_i\)s to \(\sigma\), so that we only have one substitution to manipulate into a “small” one. The following proposition provides this link via \(\tilde{\mu}\), crucially by appealing to structured assertions. Recall that \(dom(\sigma) = Y_i^\sigma\).
Proposition 5.4. For any \( t \in \text{st} \), there is \( r \in \text{st} \) such that \( \text{vars}(r) \subseteq \mathcal{V}_i^\sigma \), \( r \) is an instance of \( t \), and \( \tilde{\mu}(t) = \sigma(r) \).

Proof. For any \( t \in \text{st} \), if \( \text{vars}(t) \subseteq \mathcal{V}_i^\sigma \) then \( r := t \). Otherwise, \( t \in \text{st}(\text{HE}_i \cup \text{IE}_i) \) for some \( i \). There are \( \text{pat}, s \in \text{st} \) such that \( \text{vars}(\text{pat}) \cap \mathcal{V}_i^\sigma = \emptyset \), \( \text{vars}(s) \subseteq \mathcal{V}_i^\sigma \), \( s \) is an instance of \( \text{pat} \), \( t \) is a subterm of \( \text{pat} \), and \( \text{eq}(\text{pat}, s) \in \text{HE}_i \cup \text{IE}_i \). Suppose \( t = \text{pat}|_p \). Then, \( s|_p \) is an instance of \( t \), so \( r := s|_p \). Since \( \text{eq}(\text{pat}, s) \in D_{i+1} \cup E_{i+1} \), by Lemma 5.3, \( \tilde{\mu}(\text{pat}) = \tilde{\mu}(s) \), and thus \( \tilde{\mu}(t) = \tilde{\mu}(r) \). But since \( \text{vars}(r) \subseteq \mathcal{V}_i^\sigma \), \( \tilde{\mu}(t) = \sigma(r) \). \( \square \)

We now define a notion of “minimal variables”, which are variables that do not unify with any non-atomic term mentioned in the protocol. The idea is that they can be freely “zapped” to an atomic constant.

Definition 5.5 (Minimal variables). \( x \in \mathcal{V}_i^\sigma \) is minimal if there is no \( t \in \text{st}^- \) such that \( \text{vars}(t) \subseteq \mathcal{V}_i^\sigma \) and \( \sigma(x) = \sigma(t) \).

Suppose \( x \) is minimal and \( \sigma(x) \) is a subterm of some term learnt by the intruder during a run. This means that \( \sigma(x) \) was part of an earlier send by an honest agent. But honest agents do not generate unconstrained messages on their own. Thus, \( \sigma(x) \) must have been part of a message sent by the intruder earlier. One can also see that if the intruder can derive \( \text{eq}(x, r) \) for minimal \( x \) and \( r \in \text{st} \), then \( r \notin \text{st}^- \). If it were, we could find (by Prop. 5.4) \( s \in \text{st}^- \) with \( \text{vars}(s) \subseteq \mathcal{V}_i^\sigma \) such that \( \sigma(x) = \sigma(s) \), which contradicts the minimality of \( x \). Hence either \( r \) is minimal, or \( r \in \mathcal{V} \setminus \mathcal{V}_i^\sigma \). We state these two properties below. The proofs are given in Appendix B.

Lemma 5.6. If \( x \) is minimal and \( \sigma(x) \in \text{st}(\sigma(T_i)) \), then there is an \( r < p \) such that \( \sigma(x) \in \text{st}(\sigma(I_T_r)) \).

Proposition 5.7. Suppose \( i \leq n \), \( x \) is minimal, and \( s \in \text{st} \). Let \( \lambda \in \{ \sigma, \sigma\mu_i \} \). If \( \sigma(T_{i-1}; E_{i-1}) \vdash_{\text{eq}} \lambda(\text{eq}(x, s)) \), then either \( s \) is minimal, or \( s \in \mathcal{V} \setminus \mathcal{V}_i^\sigma \).

There is a potential problem with our proposed strategy of zapping minimal variables to a constant. Proofs might “dip below” a non-atomic \( \sigma(x) \), for minimal \( x \). “Dipping below” means that we apply a destructor rule on \( \sigma(x) \) to obtain a term that is not of the form \( \sigma(r) \) for any \( r \in \text{st} \). If we zap \( \sigma(x) \), we can no longer apply this destructor rule. To address this problem, we define well-typed proofs, in which there is no dipping below. We then show that any normal proof can be converted to a well-typed one. The intuition is that for a minimal \( x \), \( \sigma(x) \) can always be constructed from its constituents, so any dipping below can be replaced by a proof of the appropriate constituent.

Definition 5.8 (Well-typed proof). For \( X \cup \{ r \} \subseteq \mathcal{T} \), a proof \( \pi \) of \( X \vdash_{\text{dy}} r \) is well-typed if for each subproof \( \pi' \), either it ends in a constructor rule, or \( \pi' \) derives \( s \in \sigma(\text{st}^-) \).

An important technical lemma proved in [17] is that if \( x \) is minimal and \( \sigma(T_i) \vdash_{\text{dy}} \sigma(x) \), then there is a proof of \( \sigma(T_i) \vdash \sigma(x) \) ending in a constructor rule. The following lemma is a generalization of this result.

17
Lemma 5.9. For all \( t \) and all \( i \in \{0,\ldots,n\} \), if \( \sigma(T_i) \vdash_{dy} t \), there is a well-typed normal proof of \( \sigma(T_i) \vdash_{dy} t \).

Proof. The proof proceeds by induction on \( i \). Suppose there is a normal proof \( \hat{\rho} \) of \( \sigma(T_i) \vdash_{dy} t \). Let \( \rho \), with conclusion \( \sigma(T_i) \vdash_{dy} r \), be a subproof of \( \hat{\rho} \) such that every immediate subproof of \( \rho \) is well-typed but \( \rho \) is not. So \( \rho \) does not end in a constructor, and \( r \notin \sigma(st^-) \).

- If \( \rho \) ends in \( \text{ax} \), then \( r = \sigma(u) \) for some \( u \in T_i \). Thus \( r \in \sigma(st) \) but \( r \notin \sigma(st^-) \), i.e. \( u \notin st^- \), and hence \( u \) is a variable \( x \in \mathcal{V}_{i}^{\sigma} \). If \( x \) is not minimal, then \( r = \sigma(x) = \sigma(s) \) for some \( s \in st^- \), which contradicts \( r \notin \sigma(st^-) \).

- If \( \rho \) ends in a destructor rule, let \( \rho_1 \) be the subproof deriving the major premise \( r_1 \). By normality, \( \rho_1 \) ends in a destructor, and since \( \rho_1 \) is well-typed, \( r_1 = \sigma(u_1) \) for some \( u_1 \in st^- \). So \( u_1 \notin \mathcal{V}_{i}^{\sigma} \). If \( u_1 \in \mathcal{V} \setminus \mathcal{V}_{i}^{\sigma} \), \( r_1 = \sigma(u_1) = u_1 \) would be a variable, and a destructor could not be applied to \( r_1 \). Hence, \( u_1 \) has the same outermost structure as \( r_1 \), i.e. there is an immediate subterm \( u \) of \( u_1 \) such that \( r = \sigma(u) \). \( u_1 \in st \), so \( u \in st \) also. But \( \rho \) is not well-typed, so \( u \notin st^- \). Thus, \( u \in \mathcal{V}_{i}^{\sigma} \). Arguing as in the previous case, we get that \( u \) is minimal.

Thus, \( r = \sigma(x) \) for a minimal \( x \) and \( \rho \) ends in \( \text{ax} \) or a destructor rule. By Theorem 2.1, \( \sigma(x) \in st(\sigma(T_i)) \). By Lemma 5.6, there is \( j < i \) such that \( \sigma(x) \in st(\sigma(\mu_j(IT_j))) \). Let \( p \) be the earliest index such that \( \sigma(x) \in st(\sigma(\mu_p(IT_p))) \). Let \( s \in IT_p \) be such that \( \sigma(x) \in st(\sigma(\mu_p(s))) \).

By (R8), \( \sigma(T_{p-1}) \vdash_{dy} \sigma\mu_p(a) \) for every \( a \in IT_p \), so we have a normal proof \( \pi \) of \( \sigma(T_{p-1}) \vdash_{dy} \sigma\mu_p(s) \).

So we have \( \hat{\rho} = \sigma\mu_p(s) \) and a normal proof \( \pi \) of \( \sigma(T_{p-1}) \vdash \hat{\rho} \) such that \( \sigma(x) \in st(\hat{\rho}) \). Consider a minimal subproof \( \varpi \) of \( \pi \) such that \( \sigma(x) \) is a subterm of its conclusion \( \hat{\rho} \). If \( \varpi \) ends in \( \text{ax} \) or a destructor, then by Theorem 2.1, \( \hat{\varpi} \in st(\sigma(T_{p-1})) \), and hence \( \sigma(x) \in st(\sigma(T_{p-1})) \). But by Lemma 5.6, there is \( r < p - 1 \) such that \( \sigma(x) \in st(\sigma(\mu_r(IT_r))) \), contradicting the fact that \( p \) is the earliest such index. So \( \varpi \) ends in a constructor. If \( \sigma(x) \neq \hat{\varpi} \), then \( \sigma(x) \) is a subterm of one of the premises \( \hat{\varpi} \). But then there is a proper subproof of \( \varpi \) such that \( \sigma(x) \) is a subterm of its conclusion, which contradicts the minimality of \( \varpi \). Thus, \( \sigma(x) = \hat{\varpi} \) and \( \varpi \) proves \( \sigma(T_{p-1}) \vdash_{dy} \sigma(x) \).

Since \( p - 1 < i \), we know by IH that there is a well-typed normal proof of \( \sigma(T_{p-1}) \vdash_{dy} \sigma(x) \). By adding terms on the LHS to get \( \sigma(T_i) \), we get a well-typed normal proof \( \pi_x \) of \( \sigma(T_i) \vdash_{dy} \sigma(x) \). Now we can replace \( \rho \) in \( \hat{\rho} \) by \( \pi_x \). By thus repeatedly replacing minimal non-well-typed proofs, we can rid \( \hat{\rho} \) of all subproofs which are not well-typed, and get a well-typed normal proof of \( \sigma(T_i) \vdash_{dy} t \). \( \square \)

Definition 5.10 (Well-typed equality proofs). A proof \( \pi \) of \( S; A \vdash_{eq} \text{eq}(r, s) \) is well-typed if for every subproof \( \pi' \) with conclusion \( S; A \vdash_{eq} \text{eq}(t, u) \),

1. \( \pi' \) contains an occurrence of the \text{cons} rule, or
2. both \( t \) and \( u \) are in \( \sigma(st) \), or
3. \( t = u \).
Lemma 5.11. For all $i \leq n$, every normal $\vdash_{eq}$ proof from $\sigma(T_i; E_i)$ is a well-typed equality proof.

Proof. Let $\pi$ be a normal proof of $\sigma(T_i; E_i) \vdash_{eq} \text{eq}(t, u)$ ending in $r$. Assume all proper subproofs of $\pi$ are well-typed.

$r = \text{ax}$: $\text{eq}(t, u) \in \sigma(E_i)$. So $t, u \in \sigma(\text{st})$ and $\pi$ is well-typed.

$r = \text{eq}$: $t = u$, and so $\pi$ is well-typed.

$r = \text{trans}$: Let $\pi_1, \ldots, \pi_k$ be the immediate (well-typed) subproofs of $\pi$, with $\pi_i$ ending in $\text{eq}(t_i, t_{i+1})$, where $t = t_1$ and $u = t_{k+1}$. The following two cases arise.

- If $\text{cons}$ occurs in a $\pi_i$, then $\text{cons}$ occurs in $\pi$.
- If $\text{cons}$ does not occur in any of the $\pi_i$s, by normality, for all $i \leq k$, $t_i \neq t_{i+1}$.
  Specifically, $t_1 \neq t_2$ and $t_k \neq t_{k+1}$. $\pi_1$ and $\pi_k$ are well-typed, so $t_1, t_2, t_k, t_{k+1} \in \sigma(\text{st})$. In particular, $t_1, t_{k+1} \in \sigma(\text{st})$.

Thus, in both cases, $\pi$ is well-typed.

$r = \text{proj}$: Let $\pi'$ be the immediate (well-typed) subproof of $\pi$, with conclusion $\text{eq}(\hat{\tau}, \hat{u})$. Since $\pi$ is normal, neither $\pi$ nor $\pi'$ contains $\text{cons}$. The following two cases arise.

$\hat{t} = \hat{u}$: $t$ and $u$ are immediate subterms of $\hat{t}$ and $\hat{u}$, so $t = u$.

$\hat{t} \neq \hat{u}$ and $\hat{t}, \hat{u} \in \sigma(\text{st})$: Suppose there exist $\hat{r}, \hat{s} \in \text{st}^-$ such that $\sigma(\hat{r}) = \hat{t}$ and $\sigma(\hat{s}) = \hat{u}$. Then, $\hat{r}$ and $\hat{s}$ are terms whose outermost structure matches that of $\hat{t}$ and $\hat{u}$ respectively. Now $t$ and $u$ are immediate subterms of $\hat{t}$ and $\hat{u}$, so pick the corresponding immediate subterms $r$ of $\hat{r}$ and $s$ of $\hat{s}$. Thus $\sigma(r) = t$ and $\sigma(s) = u$, and $r, s \in \text{st}$, so $t, u \in \sigma(\text{st})$.

Suppose $\hat{t} \notin \sigma(\text{st}^-)$. Then, $\hat{t} = \sigma(x)$ for a minimal $x$, and thus, $\sigma(T_i; \sigma(E_i)) \vdash_{eq} \sigma(\text{eq}(x, a))$ for some $a \in \text{st}$. By Prop. 5.7, either $a$ itself is minimal, or $a \in \mathcal{V} \setminus \mathcal{V}_i^\sigma$.

But if $a \in \mathcal{V} \setminus \mathcal{V}_i^\sigma$, then $\hat{u} = \sigma(a) = a$ would be an atomic term, and we could not have applied the $\text{proj}$ rule on $\pi'$. So $a$ is minimal. But by Lemma 5.3, $\sigma(x) = \tilde{\mu}(x) = \tilde{\mu}(a) = \sigma(a)$, and hence $\hat{t} = \hat{u}$, which contradicts our assumption. The case when $\hat{t} \notin \sigma(\text{st}^-)$ is handled similarly.

$r = \text{cons}$: All proper subproofs are well-typed by IH and $\pi$ has an occurrence of $\text{cons}$, so $\pi$ is well-typed.

The next series of definitions formalizes the “small substitutions” $\tau, \vartheta_i, \nu_i$.

Definition 5.12 (Zappable). A term $t$ is zapppable if there is a minimal $x$ such that $t = \sigma(x)$.

Observe that if $t \in \sigma(\text{st}^-)$, then it is not zapppable.
Definition 5.13 ($\bar{T}$). Fix a name $m \in N_{pub} \setminus st(rng(\bar{\mu}))$. For any term $t$, $\bar{T}$ is defined inductively as follows:

$$
\bar{T} = \begin{cases} 
t & \text{if } t \in N \cup V \\
m & \text{if } t \notin N \cup V, t \text{ zappable} \\
f(t_1, \ldots, t_k) & \text{if } t = f(t_1, \ldots, t_k), t \text{ not zappable} 
\end{cases}
$$

Definition 5.14 (Small substitutions $\tau, \theta_i, \nu_i$).

- For all $x \in \text{dom}(\sigma)$, $\tau(x) := \sigma(x)$.
- For all $x \in \text{dom}(\theta_i)$, $\psi_i(x) := \theta_i(x)$.
- For all $x \in \text{dom}(\mu_i)$, $\nu_i(x) := \mu_i(x)$.

$\sigma(t) = t = \tau(t)$, for a $t$ where $\text{vars}(t) = \emptyset$. However, if $t = \sigma(x)$ for some minimal $x$, $\sigma(t) = \bar{T} = m \neq \tau(t)$. Thus, in general, it is not true that $\tau(t) = \sigma(t)$ or that $\tau\nu_i(t) = \sigma\mu_i(t)$. The next lemma, however, shows that this holds for all $t \in \text{st}$.

Lemma 5.15. For any $t \in \text{st}$, and any $i \leq n$, $\tau\nu_i(t) = \sigma\mu_i(t)$ and $\tau\theta_i(t) = \sigma\theta_i(t)$.

Proof. We prove the claim for $\sigma\mu_i$ by induction on $t$. The proof for $\sigma\theta_i$ is similar.

$t = x \in V_i^\sigma$: $\sigma\mu_i(x) = \sigma(x)$ and $\tau\nu_i(x) = \tau(x)$. By definition, $\tau(x) = \sigma(x)$.

$t = x \in \text{dom}(\mu_i)$: $\sigma\mu_i(x) = \mu_i(x)$ and $\tau\nu_i(x) = \nu_i(x)$. By definition, $\nu_i(x) = \mu_i(x)$.

$t \in N$ or $t \in V \setminus \text{dom}(\sigma\mu_i)$: We have $\bar{T} = t$ and $\sigma\mu_i(t) = t = \tau\nu_i(t)$. Therefore $\tau\nu_i(t) = t = \bar{T} = \sigma\mu_i(t)$.

$t = f(t_1, \ldots, t_k)$: For $j \leq k$, since $t_j \in \text{st}$, we apply IH to get $\tau\nu_i(t_j) = \bar{\mu}_i(t_j)$. Suppose $\sigma\mu_i(t) = \sigma(x)$ for some minimal $x$. Then, $\sigma\mu_i(t)$ is ground, so $\sigma\mu_i(t) = \bar{\mu}(t) = \bar{\mu}(x) = \sigma(x)$. But by Prop. 5.4, there is $r \in \text{st}$ which is an instance of $t$ (and hence in $\text{st}^-$), with $\text{vars}(r) \subseteq V_i^\sigma$ and $\bar{\mu}(t) = \sigma(r)$. So we have that $\sigma(x) = \sigma(r)$ for some $r \in \text{st}^-$, contradicting $x$ being minimal. Thus, $\sigma\mu_i(t)$ is not zappable, and

$$
\bar{\sigma}\mu_i(t) = f(\sigma\mu_i(t_1), \ldots, \sigma\mu_i(t_k)) = f(\sigma\mu_i(t_1), \ldots, \sigma\mu_i(t_k))
$$

Let $(T; E)$ denote $(T_{i-1}; E_{i-1})$. We now aim to show that if $\sigma(T; E) \vdash_{eq} \sigma\mu_i(eq(t, u))$ then $\tau(T; E) \vdash_{eq} \tau\nu_i(eq(t, u))$, and similarly for $\vdash_{dy}$ proofs. It is convenient to prove this for arbitrary terms $r$, not just those expressed as $\sigma\mu_i(t)$ — i.e., if $\sigma(T; E) \vdash_{eq} eq(r, s)$, then $\tau(T; E) \vdash_{eq} eq(\bar{\tau}, s)$. But here we face a problem: proofs can “grow into” a non-atomic $\sigma(x)$ for minimal $x$. We illustrate this in the following example.
Example 5.16. Let \( t = f(t_1, t_2) \), and \( u = f(u_1, u_2) = \sigma(x) \) for minimal \( x \). Suppose we can
derive \( \text{eq}(t_1, u_1) \) and \( \text{eq}(t_2, u_2) \) from \( \sigma(T; E) \). Then, we can apply \text{cons} to obtain a proof of
\( \text{eq}(t, \sigma(x)) \) from \( \sigma(T; E) \). But now, assume \( \forall(t_1, t_2) \cap \forall_q \neq \emptyset \) and \( f(t_1, t_2) \neq \sigma \mu_l(r) \) for any \( r \). Since \( t \) is not zappable, \( T \) is of the form \( f(\cdot, \cdot) \), while \( \overline{\mu} = m \). However, \( \lambda(T) = m \) is not true for any \( \lambda \), and \( \tau(T; E) \vdash_{eq} \text{eq}(\overline{t}, \overline{u}) \) is not provable.

Thus when a proof derives terms which “grow into” \( \sigma(x) \) as a result of a constructor rule
or the \text{cons} rule, we cannot simply follow the inductive definition of \( T \). We deal with such
situations by zapping any term that is not of the form \( \sigma \lambda(r) \) for \( \lambda \in \text{Substs} \) and \( r \in \text{st} \). So in
Example 5.16, we zap both sides of the equality to \( m \). This is formalized by the translation function \( \llbracket \cdot \rrbracket \).

Definition 5.17 (Translation function \( \llbracket \cdot \rrbracket \)).

\[
[tt] = \begin{cases} 
\overline{T} & \text{if } t \in \text{sst} \\
\overline{m} & \text{otherwise} 
\end{cases}
\]

\[
[\text{eq}(t, u)] = \begin{cases} 
\text{eq}(\overline{T}, \overline{u}) & \text{if } t, u \in \text{sst} \\
\text{eq}(\overline{m}, \overline{m}) & \text{otherwise} 
\end{cases}
\]

where \( \text{sst} = \{ \sigma \lambda(t) \mid t \in \text{st}, \lambda \in \text{Substs} \} \).

Note that if either \( t \) or \( u \) is not in \( \text{sst} \), then \( [\text{eq}(t, u)] \) is a dummy equality. This reflects
the fact that in well-typed proofs, once one goes beyond \( \text{sst} \), one cannot recover any “usable”
information that matches a pattern in the protocol.

Even in the situations where \( [t] \) is identical to \( \overline{T} \), \( [t_i] \) for some subterm \( t_i \) need not be
the same as \( \overline{T_i} \). The next proposition ensures a limited correspondence for such cases. If
\( t = f(t_1, \ldots, t_n) \in \text{sst} \), then by definition, \( [t] = \overline{T} \). If in addition \( t \) is not zappable, then
for each \( i \leq n, [t_i] = \overline{T_i} \). The proof (presented in Appendix B) is a case analysis on the
structure of the term \( t \in \text{st} \) such that \( t = \sigma \lambda(r) \), and invokes Prop. 5.4.

Proposition 5.18. If \( t = f(t_1, \ldots, t_n) \in \text{sst} \) and \( t \) is not zappable, then each \( t_i \in \text{sst} \).

We now show that \( \llbracket \cdot \rrbracket \) preserves \( \vdash_{dy} \) and \( \vdash_{eq} \) proofs.

Lemma 5.19. For any \( i \leq n \) and any term \( t \),

\[
\sigma(T_i) \vdash_{dy} t \implies \tau(T_i) \vdash_{dy} [t].
\]

Proof. Let \( X = \sigma(T_i) \) and \( Y = \tau(T_i) \). Observe that \( Y = \{ \tau(r) \mid r \in T_i \} = \{ \overline{\sigma(r)} \mid r \in T_i \} = 
\{ \overline{s} \mid s \in X \} \).

Suppose there is a proof of \( X \vdash_{dy} t \). Then there is a well-typed normal proof \( \pi \) of the
same, ending in \( r \). We show by induction on structure of proofs that \( Y \vdash_{dy} [t] \).

\footnote{Note that \( \sigma(\text{st}) \subseteq \text{sst} \).}
\( \mathbf{r = pb:} \) \( [t] = \overline{t} = t \in \mathcal{N}_{pb} \) or \( [t] = m \in \mathcal{N}_{pb} \). In both cases, there is a proof of \( Y \vdash_{dy} [t] \) which ends in \( \mathbf{pb} \).

\( \mathbf{r = ax:} \) \( t \in X \subseteq \sigma(\mathbf{st}) \subseteq \mathbf{sst} \) and hence \( [t] = \overline{t} \in Y \). Thus there is a proof of \( Y \vdash_{dy} [t] \) which ends in \( \mathbf{ax} \).

\( \mathbf{r} \) is a destructor rule: Let the immediate subproofs of \( \pi \) be \( \pi_1, \ldots, \pi_k \), deriving \( t_1, \ldots, t_k \) respectively, with \( t_1 \) being the major premise, and \( t \) an immediate subterm of \( t_1 \). Since \( \pi \) is well-typed and normal, \( \pi_1 \) ends in a destructor rule. Since \( \pi_1 \) is also well-typed, \( t_1 = \sigma(u_1) \) for some \( u_1 \in \mathbf{st}^- \), with the same outermost operator as \( t_1 \). Thus, \( t = \sigma(u) \) for an immediate subterm \( u \) of \( u_1 \). Since \( t_1 \in \sigma(\mathbf{st}^-) \), it is not zappable. Also, \( t_1, t \in \sigma(\mathbf{st}) \subseteq \mathbf{sst} \), and hence \( [t] = \overline{t_1} \) and \( [t] = \overline{t} \). By IH, there is a proof \( \omega_i \) of \( Y \vdash_{dy} [t_i] \) for each \( i \leq k \), and applying \( \mathbf{r} \) on the \( \omega_i \)s, we get a proof of \( Y \vdash_{dy} [t] \).

\( \mathbf{r} \) is a constructor rule: Let \( t = f(t_1, \ldots, t_k) \) and let the immediate subproofs of \( \pi \) be \( \pi_1, \ldots, \pi_k \), with conclusions \( t_1, \ldots, t_k \) respectively. By IH, there is a proof \( \omega_i \) of \( Y \vdash_{dy} [t_i] \) for each \( i \leq k \). Suppose \( t \in \mathbf{sst} \). Then \( [t] = \overline{t} \). If \( t \) is zappable, then \( \overline{t} = m \), and we have a proof of \( Y \vdash_{dy} [t] \) ending in \( \mathbf{pb} \). If \( t \) is not zappable, then by Prop. 5.18, each \( t_i \in \mathbf{sst} \). Thus, \( [t_i] = \overline{t_i} \) for each \( i \leq n \), and \( [t] = f([t_1], \ldots, [t_n]) = f([\overline{t_1}], \ldots, [\overline{t_n}]) \), and we can apply \( \mathbf{r} \) on the \( \omega_i \)s to get a proof of \( Y \vdash_{dy} [t] \). If \( t \notin \mathbf{sst} \), then \( [t] = m \), and again we have a proof of \( Y \vdash_{dy} [t] \) ending in \( \mathbf{pb} \).

Lemma 5.20. For all \( i \leq n \) and all terms \( t, u \),

\[
\sigma(T_i; E_i) \vdash_{eq} \mathbf{eq}(t, u) \implies \tau(T_i; E_i) \vdash_{eq} [\mathbf{eq}(t, u)].
\]

Proof. For ease of notation, we let \( (X; A) = \sigma(T_i; E_i) \) and \( (Y; B) = \tau(T_i; E_i) \). As earlier, \( Y = \{ \overline{s} \mid s \in X \} \). Similarly, \( B = \{ \mathbf{eq}(\sigma(r), \sigma(r')) \mid \mathbf{eq}(r, r') \in E_i \} = \{ \mathbf{eq}(\sigma(r), \sigma(r')) \mid \mathbf{eq}(r, r') \in E_i \} \).

Suppose there is a well-typed normal proof of \( X; A \vdash_{eq} \mathbf{eq}(t, u) \) ending in \( \mathbf{r} \). We show by induction on structure of proofs that \( Y; B \vdash_{eq} [\mathbf{eq}(t, u)] \).

* \( \mathbf{r = ax:} \) \( \mathbf{eq}(t, u) \in A \), and so \( t, u \in \sigma(T_i) \subseteq \sigma(\mathbf{st}) \subseteq \mathbf{sst} \). Hence, \( [\mathbf{eq}(t, u)] = \mathbf{eq}(\overline{t}, \overline{u}) \in B \), and there is a proof of \( Y; B \vdash_{eq} [\mathbf{eq}(t, u)] \) ending in \( \mathbf{ax} \).

* \( \mathbf{r = eq:} \) \( t = u \) and \( X \vdash_{dy} t \). If \( t \notin \mathbf{sst} \), \( [\mathbf{eq}(t, u)] = \mathbf{eq}(m, m) \). Otherwise, \( [\mathbf{eq}(t, u)] = \mathbf{eq}([t], [t]) \), and by Lemma 5.19, \( Y \vdash_{dy} [t] \). In both cases, we have a proof of \( Y; B \vdash_{eq} [\mathbf{eq}(t, u)] \) which ends in \( \mathbf{eq} \).

* \( \mathbf{r = trans:} \) Let \( \pi_1, \ldots, \pi_k \) be the immediate subproofs of \( \pi \), with each \( \pi_i \) deriving \( X; A \vdash_{eq} (t_i, t_{i+1}) \), with \( t = t_1 \) and \( u = t_{k+1} \). If either \( t_1 \notin \mathbf{sst} \) or \( t_k \notin \mathbf{sst} \), then \( [\mathbf{eq}(t, u)] \) is \( \mathbf{eq}(m, m) \), which is derivable by a proof ending in \( \mathbf{eq} \).

Suppose \( t_1, t_k \in \mathbf{sst} \). Then \( [\mathbf{eq}(t, u)] = [\mathbf{eq}(t_1, t_{k+1})] = \mathbf{eq}([\overline{t_1}, \overline{t_{k+1}}]). \) By normality, none of the \( \pi_i \)s proves equality between identical terms. By well-typedness, for every \( \pi_i \) that does not contain \( \mathbf{cons} \), both \( t_i \) and \( t_{i+1} \) are in \( \sigma(\mathbf{st}) \). But \( \pi \) is normal, so no two adjacent \( \pi_i \)s
end in \textbf{cons}, so every alternate \( \pi_i \) ends in a rule other than \textbf{cons}. Therefore, by Prop. \ref{prop:alt}, for \( i \leq k \), either \( \pi_i \) or \( \pi_{i+1} \) does not contain \textbf{cons}. Thus, \( t_2, \ldots, t_k \in \sigma(st) \subseteq \text{sst} \). By assumption, \( t_i, t_k \in \text{sst} \), so \( t_i \in \text{sst} \) for all \( i \leq k + 1 \), and thus \( \text{eq}(t_i, t_{i+1}) = \text{eq}(t_i, t_{i+1}) \) for all \( i \leq k + 1 \). By IH, there are \( \varpi_1, \ldots, \varpi_k \), with each \( \varpi_i \) deriving \( \varpi_i \). Apply \text{trans} to get a proof of \( Y; B \vdash_{eq} \text{eq}(t_i, t_{i+1}) \).

- **\( r = \text{proj} \):** Let the immediate subproof of \( \pi \) be \( \pi' \), and let the conclusion of \( \pi' \) be \( X; A \vdash \text{eq}(f(t_1, \ldots, t_k), f(u_1, \ldots, u_k)) \). Let \( t = t_i \) and \( u = u_i \). By the abstractability condition, \( X \vdash_{dy} t_i \) and \( X \vdash_{dy} u_i \) for all \( i \leq k \). For readability, let \( a = f(t_1, \ldots, t_k) \) and \( b = f(u_1, \ldots, u_k) \). Since \( \pi \) (and therefore \( \pi' \)) is well-formed and normal, there is no occurrence of \textbf{cons} in \( \pi' \). So two cases arise: either \( a = b \) or \( a, b \in \sigma(st) \). In the former case, \( t = u \) as well, and we get a proof of \( Y; B \vdash_{eq} \text{eq}(t, u) \) ending in \text{eq}. In the latter case, \( a \neq b \), and \( a, b \in \sigma(st) \). Suppose there are \( c, d \in \text{st}^{-} \) such that \( \sigma(c) = a \) and \( \sigma(d) = b \). Then neither \( a \) nor \( b \) is zappable, so both \( \overline{a} \) and \( \overline{b} \) are non-atomic and have the same outermost structure. By IH, there is a proof \( \varpi \) of \( Y; B \vdash_{eq} \text{eq}(a, b) \). But \( \text{eq}(a, b) = \text{eq}(\overline{a}, \overline{b}) \) where \( \overline{a} = f(t_1, \ldots, t_k) \) (and similarly for \( \overline{b} \)). Since \( Y \vdash_{dy} t_i \) and \( Y \vdash_{dy} \overline{t}_i \) for all \( i \leq k \) (by Lemma \ref{lemma:dy}), the abstractability side conditions are fulfilled, and we can apply \text{proj} on \( \varpi \) to get a proof of \( Y; B \vdash_{eq} \text{eq}(\overline{t}_i, \overline{u}_i) \), i.e. \( Y; B \vdash_{eq} \text{eq}(t, u) \).

Suppose there is no such \( c \in \text{st}^{-} \), i.e. \( a \notin \sigma(\text{st}^{-}) \). Then \( a = \sigma(x) \) for some minimal \( x \). But then, \( X; A \vdash \sigma(\text{eq}(x, s)) \) for some \( s \in \text{st} \). By Prop. \ref{prop:min}, either \( s \in \mathcal{V}_i^\sigma \) is minimal, or \( s \in \mathcal{V} \setminus \mathcal{V}_i^\sigma \). But if \( s \in \mathcal{V} \setminus \mathcal{V}_i^\sigma \), then \( b = \sigma(s) = s \) would be atomic, and we could not have applied \text{proj}. So \( s \) is a minimal \( x \), and by Lemma \ref{lemma:min}, \( \sigma(x) = \sigma(y) \). Hence \( a = b \), which contradicts our assumption. The case when \( b = \sigma(x) \) for some minimal \( x \) is similar.

- **\( r = \text{cons} \):** Let \( t = f(t_1, \ldots, t_k) \) and \( u = f(u_1, \ldots, u_k) \). Let the immediate subproofs of \( \pi \) be \( \pi_1, \ldots, \pi_k \), with each \( \pi_i \) proving \( X; A \vdash \text{eq}(t_i, u_i) \). The following cases arise.

\begin{itemize}
  \item \( \{t, u\} \not\subseteq \text{sst} \): Then \( \text{eq}(t, u) = \text{eq}(m, m) \), and we have \( Y; B \vdash_{eq} \text{eq}(t, u) \) via a proof ending in \text{eq}.

  \item \( t = u \): Because \( X; A \vdash_{eq} \text{eq}(t, u) \) and \( (X; A) \) is eq-pure, we have \( X \vdash_{dy} t \). By Lemma \ref{lemma:dy}, we have \( Y \vdash_{dy} [t] \). Thus we have a proof of \( Y; B \vdash_{eq} \text{eq}(t, u) \) ending in \text{eq}.

  \item \( t, u \in \text{sst} \) with \( t \neq u \): Here, \( \text{eq}(t, u) = \text{eq}(t, u) \). Suppose there are \( r, s \in \text{st}^{-} \) such that \( \sigma(r) = t \) and \( \sigma(s) = u \). Then \( t \) and \( u \) are not zappable. By Prop. \ref{prop:zapp}, \( t_i, u_i \in \text{sst} \). So, for \( i \leq k \), \( \text{eq}(t_i, u_i) = \text{eq}(t_i, u_i) \) and \( \text{eq}(t, u) = \text{eq}(f(t_1, \ldots, t_k), f(u_i, \ldots, u_k)) \). By IH, we have for each \( i \leq k \) a proof \( \varpi_i \) of \( Y; B \vdash_{eq} \text{eq}(t_i, u_i) \). Apply \text{cons} on \( \varpi_i \)s to get a proof of \( Y; B \vdash_{eq} \text{eq}(t, u) \).

  Suppose on the other hand that \( t \notin \sigma(\text{st}^{-}) \). Then \( t = \sigma(x) \) for some minimal \( x \). But then, \( X; A \vdash_{eq} \sigma(\text{eq}(x, s)) \) for some \( s \in \text{st} \). By Prop. \ref{prop:min}, either \( s \) is a minimal variable, or \( s \in \mathcal{V} \setminus \mathcal{V}_i^\sigma \). But if \( s \in \mathcal{V} \setminus \mathcal{V}_i^\sigma \), then \( u = \sigma(s) = s \) would be atomic, which could not be yielded by \textbf{cons}. So \( s \) is a minimal variable \( y \). But then, by Lemma \ref{lemma:min}, \( \sigma(x) = \sigma(y) \), and so \( t = u \), but we assumed \( t \neq u \). \( \square \)
For any \( t, u \in \text{st} \), we have that \( \sigma \mu_i(t), \sigma \mu_i(u) \in \text{sst} \). Thus \( [\sigma \mu_i(t)] = \overline{\sigma \mu_i(t)} = \tau \nu_i(t) \). Likewise, \( [\sigma \mu_i(\text{eq}(t, u))] = \tau \nu_i(\text{eq}(t, u)) \). Thus we get our main theorem, by applying Lemmas 5.19 and 5.20.

**Theorem 5.21.** Let \( t, u \in \text{st} \) and \( i \leq n \).

- If \( \sigma(T_{i-1}) \vdash_{d_y} \sigma \mu_i(t) \) then \( \tau(T_{i-1}) \vdash_{d_y} \tau \nu_i(t) \).
- If \( \sigma(T_{i-1}; E_{i-1}) \vdash_{eq} \sigma \mu_i(\text{eq}(t, u)) \) then \( \tau(T_{i-1}; E_{i-1}) \vdash_{eq} \tau \nu_i(\text{eq}(t, u)) \).

### 5.1 Smallness of \( \tau, \vartheta_i, \nu_i \)

**Definition 5.22** (Small substitutions). A substitution \( \varrho \) is said to be small if for all \( x \in \text{dom}(\varrho) \), \( |\text{st}(\varrho(x))| \leq |\text{st}| + 1 \), i.e. the number of distinct subterms in \( \varrho(x) \) is bounded by the number of terms in \( \text{st} \cup \{m\} \).

**Theorem 5.23.** Every \( \varrho \in \{\tau, \vartheta_i, \nu_i \mid i \leq n\} \) is small.

**Proof.** For all \( x \in \text{dom}(\tau) \), \( \tau(x) = \overline{\sigma(x)} \). So \( |\text{st}(\tau(x))| = |\text{st}(\overline{\sigma(x)})| \). For \( x \in \text{dom}(\nu_i) \), \( \nu_i(x) = \overline{\mu_i(x)} \). By Prop. 4.6, \( \mu_i(x) \) is a variable \( y \in \mathcal{V}_q \), or ground. In the former case, \( \nu_i(x) = y \), and \( |\text{st}(\nu_i(x))| = 1 \). In the latter case, by Prop. 5.4, \( \mu_i(x) = \tilde{\mu}(x) = \sigma(t) \) for a \( t \in \text{st} \) with \( \text{vars}(t) \subseteq \mathcal{V}_{\sigma} \). Thus \( \nu_i(x) = \overline{\sigma(t)} \) for some \( t \in \text{st} \), and hence \( |\text{st}(\nu_i(x))| = |\text{st}(\overline{\sigma(t)})| \) for a \( t \in \text{st} \). Similarly for \( x \in \text{dom}(\vartheta_i) \). Thus for all \( \varrho \in \{\tau, \vartheta_i, \nu_i \mid i \leq n\} \) and all \( x \in \text{dom}(\varrho) \), we have \( |\text{st}(\varrho(x))| = |\text{st}(\overline{\sigma(t)})| \) for a \( t \in \text{st} \cup \{m\} \) with \( \text{vars}(t) \subseteq \mathcal{V}_{\sigma} \).

We now prove that for all such \( t \), \( |\text{st}(\overline{\sigma(t)})| \leq |\text{st} \cup \{m\}| \), which would imply the statement of the theorem. The technique is similar to that of [17]. We proceed by constructing a finite sequence of pairs \((E_0, V_0), \ldots, (E_N, V_N)\), such that:

- (p1) each \( E_i \subseteq \text{st} \cup \{m\} \),
- (p2) \( V_N = \emptyset \), and
- (p3) for all \( p \leq N \): \( |\text{st}(\overline{\sigma(t)})| \leq |\text{st}(E_p)| + \sum_{z \in V_p} |\text{st}(\overline{\sigma(z)})| \).

Applying (p3) on \( N \) and using (p2), \( |\text{st}(\overline{\sigma(t)})| \leq |\text{st}(E_N)| \). From (p1), \( \text{st}(E_N) \subseteq \text{st}(\text{st} \cup \{m\}) \), so \( |\text{st}(\overline{\sigma(t)})| \leq |\text{st}| + 1 \).

We define \( \text{es}(E) = |\text{st}(E)|, \text{vs}(V) = \sum_{z \in V} |\text{st}(\overline{\sigma(z)})| \) and \( \text{sz}(E, V) = \text{es}(E) + \text{vs}(V) \).

We satisfy (p2) and (p3) by choosing pairs \((E_p, V_p)\) s.t.:

- (q1) \( |\text{st}(\overline{\sigma(t)})| \leq \text{sz}(E_0, V_0) \),
- (q2) \( \text{sz}(E_p, V_p) \leq \text{sz}(E_{p+1}, V_{p+1}) \), and
- (q3) \( \text{vs}(V_{p+1}) < \text{vs}(V_p) \).
(p3) follows from (q1) and (q2), while the existence of \( N \) satisfying (p2) follows from (q3) (if we choose \( V_0 \) to be finite).

We define \( E_0 = \{ t \} \) and \( V_0 = \text{vars}(t) \). (q1) is satisfied since for every \( s \in \text{st}(\sigma(t)) \), either \( s = \sigma(t') \) for some \( t' \in \text{st}(t) \) or \( s \in \text{st}(\sigma(z)) \) for some \( z \in \text{vars}(t) \). Also, \( V_0 \) is finite.

Suppose we have \( (E_p, V_p) \) with some \( y \in V_p \). If \( y \) is minimal, \( \sigma(y) = m \). Take \( E_{p+1} = E_p \cup \{ m \} \) and \( V_{p+1} = V_p \setminus \{ y \} \), satisfying (q2) and (q3).

Otherwise \( \sigma(y) = \sigma(r) \) for an \( r \in st^- \) with \( \text{vars}(r) \subseteq \text{dom}(\sigma) \). Let \( \text{vars}(r) = \{ z_1, \ldots, z_k \} \).

As earlier, for any \( s \in \text{st}(\sigma(r)) \), either \( s = \sigma(r') \) for some \( r' \in st(t) \) or \( s \in \text{st}(\sigma(z)) \) for a \( z \in \text{vars}(r) \). So, \( |\text{st}(\sigma(r))| \leq |\text{st}(r)| + \sum_{i \leq k} |\text{st}(\sigma(z_i))| \).

Let \( (E_{p+1}, V_{p+1}) = (E_p \cup \{ \sigma(r) \}, V_p \setminus \{ y \} \cup \{ z_1, \ldots, z_k \}) \). Let \( V' = V_p \setminus \{ y \} \). Then, \( \text{sz}(E_p, V_p) = \text{es}(E_p) + |\text{st}(\sigma(y))| + \text{vs}(V') = \text{es}(E_p) + |\text{st}(\sigma(r))| + \text{vs}(V') \leq \text{es}(E_p) + |\text{st}(\sigma(r))| + \text{vs}(V') + \sum_{i \leq k} |\text{st}(\sigma(z_i))| = \text{sz}(E_{p+1}, V_{p+1}) \).

Note that \( \sum_{i \leq k} |\text{st}(\sigma(z_i))| < |\text{st}(\sigma(r))| = |\text{st}(\sigma(y))| \). From this and the definition of \( V_{p+1} \), (q3) immediately follows.

We thus have a sequence of pairs \( (E_p, V_p) \) satisfying (q1)–(q3), and therefore (p1)–(p3), and the proof is complete. \( \square \)

### 5.2 An NP procedure for insecurity

As we saw in Section 4, a protocol \( Pr \) has an attack with one session if there is a substitution \( \sigma \), a sequence \( \xi = u_1 : \beta_1 \Rightarrow \alpha_1, \ldots, u_n : \beta_n \Rightarrow \alpha_n \) with each \( u_i \in \mathcal{A} \setminus \{ I \} \), a sequence of knowledge functions \( k_0, \ldots, k_n \), and a sequence of substitutions \( \mu_1, \ldots, \mu_n \) such that:

- for each \( i \leq n \), \( k_i(u_i) \vdash _{eq} \alpha_i \).
- for each \( i \leq n \), letting \( (X; E) = dc(k_{i-1}(I)) \):
  - for \( r \in \text{pubs}(\beta_i) \cup \text{dom}(\mu_i) \), \( \sigma(X) \vdash _{dy} \sigma(t) \).
  - for \( \text{eq}(r, s) \in \text{sf}(\beta_i) \), \( \sigma(X; E) \vdash _{eq} \sigma(t) \).

Also, for each \( i \leq n \), there are sets \( U_i \subseteq \text{st}(\alpha_i) \) and \( V_i \subseteq \text{st}(\beta_i) \) s.t. for all \( r \in U_i \), \( S_i \cup \text{bv}(\alpha_i) \vdash _{dy} r \), and for all \( r \in V_i \), \( T_{i-1} \cup \text{bv}(\beta_i) \vdash _{dy} r \). We have proved the existence of substitutions \( \tau \) and \( \nu_1, \ldots, \nu_n \) such that we can replace \( \sigma \) and \( \mu_i \) by \( \tau \) and \( \nu_i \) in the above derivabilities. In Section 5.1, we showed that \( \tau \) and the \( \nu_i \)s are small, i.e. that the number of subterms of each \( \tau(x) \) and \( \nu_i(x) \) is bounded by \( |\text{st}| + 1 \), which itself is polynomial in the size of the protocol specification.

For the honest agent derivations (of the \( \alpha_i \)s), there is no \( \sigma \) involved. So we can assume that we are working with the identity substitution on the intruder variables. Proposition 5.4 would translate in this case to saying that for all \( t \in \text{st} \), there is some \( r \in \text{st} \) such that \( \tilde{\mu}(t) = r \in \text{st} \). So the witness substitution for each \( i \) (call it \( \kappa_i \)) is always small.

Thus our algorithm makes polynomially many guesses of objects of polynomial size – assertions, the substitutions for witnesses (\( \nu_i \)s and \( \kappa_i \)s), and the intruder substitution (\( \tau \)). It just needs to check that the various equality derivations and term derivations hold, and this can be done in polynomial time. Thus the insecurity problem is in NP.
6 Conclusion

[17] solves the insecurity problem for finitely many sessions. This is an important technical component in the symbolic analysis of security protocols. It guarantees termination for verification procedures that search for attacks on protocols involving a bounded number of sessions. We have shown how to adapt and extend the techniques of [17] to handle the insecurity problem for a subclass of the assertions presented in [15], consisting of equality, conjunction, and existential quantification. In Appendix D, we also describe how these results can be extended to list membership (which can be viewed as a restricted form of disjunction).

While the subclass we consider suffices to cover the kind of example protocols presented in [15], it is a restriction nonetheless. It is an interesting challenge to see if our results scale to the full system in [15], especially to handle full disjunction. The starting point of our analysis of $(S; A) \vdash_a \alpha$ was to consider proofs without logical elimination rules, but from the unique down-closure of $(S; A)$. When we consider the standard rule for disjunction elimination, we have multiple down-closures for a particular $(S; A)$, and we need to check that each one of them derives $\alpha$. Some of these down-closures might not represent the “truth” – i.e. there might not be any substitution $\lambda$ under which all equality assertions are true. This means that the techniques in this paper do not directly extend to handle disjunction. We can reason about list membership using an “intersection” rule in place of disjunction elimination, which helps ensure that we have a unique down-closure for each $(S; A)$.

Equality assertions are closely related to tests of the kind we see in the notion of static equivalence in the applied pi calculus [1, 7]. We believe that incorporating communicable assertions would constitute an interesting extension of applied pi, and also help us understand the role and power of assertions better.

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, Oct 2017.

[2] Ben Adida. Helios: Web-based open-audit voting. In Proceedings of the 17th Conference on Security Symposium, SS’08, pages 335–348, USA, 2008. USENIX Association.

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

[4] Michael Backes, Cătălin Hritcu, and Matteo Maffei. Type-checking zero-knowledge. In Proceedings of the 15th ACM Conference on Computer and Communications Security, CCS ’08, pages 357–370, 2008.
[5] Bruno Blanchet. Modeling and verifying security protocols with the applied pi calculus and proverif. *Foundations and Trends in Privacy and Security*, 1:1–135, 10 2016.

[6] Manuel Blum, Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Noninteractive zero-knowledge. *SIAM Journal on Computing*, 20(6):1084–1118, 1991.

[7] Véronique Cortier and Steve Kremer. Formal models and techniques for analyzing security protocols: A tutorial. *Found. Trends Program. Lang.*, 1(3):151–267, 2014.

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

[9] Nancy Durgin, Patrick Lincoln, John Mitchell, and Andre Scedrov. Multiset rewriting and the complexity of bounded security protocols. *Journal of Computer Security*, 12:247–311, 02 2004.

[10] F. Javier Thayer Fábrega, Jonathan Herzog, and Joshua Guttmann. Strand spaces: Proving security protocols correct. *Journal of Computer Security*, 7:191–230, 1999.

[11] Atsushi Fujioka, Tatsuaki Okamoto, and Kaazuo Ohta. A practical secret voting scheme for large scale elections. In *ASIACRYPT*, pages 244–251, 1992.

[12] James Heather, Gavin Lowe, and Steve A. Schneider. How to prevent type flaw attacks on security protocols. *J. Comput. Secur.*, 11(2):217–244, 2003.

[13] Nevin Heintze and J. D. Tygar. A model for secure protocols and their compositions. *IEEE Trans. Software Eng.*, 22(1):16–30, 1996.

[14] Simon Meier, Benedikt Schmidt, Cas Cremers, and David Basin. The TAMARIN prover for the symbolic analysis of security protocols. In *Computer Aided Verification*, pages 696–701. Springer Berlin Heidelberg, 2013.

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

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

[17] Michaël Rusinowitch and Mathieu Turuani. Protocol insecurity with a finite number of sessions and composed keys is NP-complete. *Theoretical Computer Science*, 299(1-3):451–475, 2003.
A Proofs for Sections 2 and 3

Observation 2.11. Let $\alpha$ be an assertion and $S \cup \{t\} \subseteq \mathcal{T}$.

1. Suppose $x \neq y$, $\{x, y\} \subseteq \text{fv}(\alpha)$, $\{x, y\} \cap \text{vars}(S \cup \{t\}) = \emptyset$, and $S \vdash_{dy} t$. Let $P = \mathbb{P}_x(\alpha)$ and $\beta = \alpha[t]_P$. If $P \subseteq \mathbb{A}(S \cup \{x\}, \alpha)$, then $\mathbb{P}_y(\alpha) \subseteq \mathbb{A}(S \cup \{x\}, \alpha)$ iff $\mathbb{P}_y(\beta) \subseteq \mathbb{A}(S, \beta)$.

2. For $x \in \text{fv}(\alpha)$, there exists $T_x \subseteq \text{st}(\alpha)$ s.t. $\mathbb{P}_x(\alpha) \subseteq \mathbb{A}(S, \alpha)$ iff $S \vdash_{dy} t$ for all $t \in T_x$.

Proof.

1. Assume the conditions in the statement. Suppose $r$ is a term such that $\{x, y\} \subseteq r$, $P = \mathbb{P}_x(r)$, $s = r[t]_P$. Suppose $P \subseteq \mathbb{A}(S \cup \{x\}, r)$. Since $y$ does not occur in $t$, and only $x$ is replaced by $t$ in $r$, it is clear that $\mathbb{P}_y(r) = \mathbb{P}_y(s)$. Consider any $p \in \mathbb{P}_y(r)$. Now there is a designated set of positions $Q$ s.t. $p \in \mathbb{A}(S \cup \{x\}, r)$ iff $S \cup \{x\} \vdash_{dy} r|_q$ for all $q \in Q$. For any $q \in \mathbb{P}(r)$, it can be seen that $S \cup \{x\} \vdash_{dy} s|_q$. Thus we see that for every $q \in Q$, $S \vdash_{dy} s|_q$. But $Q$ is the same set of positions that guarantees abstractability of $p$, whether we consider $r$ or $s$. Thus we see that $p \in \mathbb{A}(S \cup \{x\}, r)$ iff $p \in \mathbb{A}(S, s)$.

Now the abstractable positions of an assertion $\gamma$ are obtained by prefixing abstractable positions of each maximal subterm of $\gamma$ with an appropriate string (denoting the position of that term inside $\gamma$). Thus the result for assertions follows immediately from the above result for terms.

2. For each $p \in \mathbb{P}(\alpha)$, there is a set of positions $Q$ s.t. $p \in \mathbb{A}(S, \alpha)$ iff $S \vdash_{dy} \alpha|_q$ for all $q \in Q$.

So let $P$ be the union of all the sets of “witnessing positions” for each $p \in \mathbb{P}_x(\alpha)$. Then, taking $T_x = \{\alpha|_p \mid p \in P\}$, we see that $\mathbb{P}_x(\alpha) \subseteq \mathbb{A}(S, \alpha)$ iff $S \vdash_{dy} t$ for all $t \in T_x$. 

Lemma 3.9. If $(T; E)$ is eq-pure and $E$ is consistent, then for any terms $t, u$, $(T; E) \vdash_{\text{core}} \text{eq}(t, u)$ iff $(T; E) \vdash_{eq} \text{eq}(t, u)$.

Proof.

Left-to-right direction: A normal proof of $(T; E) \vdash \text{eq}(t, u)$ in the $\vdash_{\text{core}}$ system does not feature any of the logical introduction rules (since they cannot later be eliminated). Of the remaining rules $\{\text{ax, eq, proj, subst}\}$, all but the $\text{subst}$ rule exist in $\vdash_{eq}$. So we need to show that an occurrence of $\text{subst}$ in the $\vdash_{\text{core}}$ system can be captured by a proof in the $\vdash_{eq}$. For a set of positions $P$, we define $\text{size}(P)$ to be the sum of the lengths of all positions in $P$.

We first claim that for all terms $t, u, r, s$ such that $P = \mathbb{P}_x(\text{eq}(t, u))$: if $(T; E) \vdash_{eq} \text{eq}(t, u)[r]_P$ and $(T; E) \vdash_{eq} \text{eq}(r, s)$, then $(T; E) \vdash_{eq} \text{eq}(t, u)[s]_P$.

The proof is by induction on $\text{size}(P)$. Since $(T; E)$ is eq-pure, $\text{eq}(b, a) \in E$ for any $a, b$ such that $\text{eq}(a, b) \in E$. We can thus show that $(T; E)$ derives $\text{eq}(a, b)$ whenever it derives $\text{eq}(t, u)$. Thus we can freely use symmetry as a rule in our proofs.
Since the subst rule can be applied in $\vdash_{\text{core}}$ on $\text{eq}(t, u)[r]_P$, we have $P \subseteq \mathcal{A}(T \cup \{x\}, \text{eq}(t, u))$. It can be shown that $\mathcal{A}(T \cup \{x\}, \text{eq}(t, u)) \subseteq \mathcal{A}(S, \text{eq}(t, u)[r]_P)$. Throughout this proof, we will consider various subterms of $t$ and $u$, and replacing $x$ inside them by $r$. For each such subterm $w$, for any $Q \subseteq \mathcal{A}(T \cup \{x\}, w)$, we can also show that $Q \subseteq \mathcal{A}(T, w[r]_Q)$.

There are a number of cases to consider now, depending on the set $P$. So we start with proofs $\pi_1$ and $\pi_2$ of $(T; E) \vdash_{eq} \text{eq}(t, u)[r]_P$ and $(T; E) \vdash_{eq} \text{eq}(r, s)$, respectively, and aim to construct a proof of $(T; E) \vdash_{eq} \text{eq}(t, u)[s]_P$. (In the proofs we construct below, we simply write sequents as $\text{eq}(v, w)$ instead of $(T; E) \vdash_{eq} \text{eq}(v, w)$, to increase readability.)

1. $P = \{0\}$: In this case, $\pi_1$ is a proof of $\text{eq}(r, u)$, and we want a proof $\pi$ of $\text{eq}(s, u)$. Now since $\text{eq}(r, s)$ is provable, by symmetry there is a proof $\pi'_2$ of $\text{eq}(s, r)$. We get the desired proof as follows:

   \[
   \begin{array}{ccc}
   \pi'_2 & \cdots & \pi_1 \\
   \vdots & \ddots & \vdots \\
   \text{eq}(s, r) & \cdots & \text{eq}(r, u) \\
   \hline
   \text{eq}(s, u)
   \end{array}
   \]

   \text{trans}

2. $P = \{1\}$: In this case, $\pi_1$ is a proof of $\text{eq}(t, r)$, and we get a proof of $\text{eq}(t, s)$ as given below:

   \[
   \begin{array}{ccc}
   \pi_1 & \cdots & \pi_2 \\
   \vdots & \ddots & \vdots \\
   \text{eq}(t, r) & \cdots & \text{eq}(r, s) \\
   \hline
   \text{eq}(t, s)
   \end{array}
   \]

   \text{trans}

3. $P = \{0, 1\}$: In this case, $\pi_1$ is a proof of $\text{eq}(r, r)$, and we want a proof of $\text{eq}(s, s)$. Since $\pi_2$ proves $\text{eq}(r, s)$ and $(T; E)$ is eq-pure, we have that $T \vdash_{dy} s$. The required proof is:

   \[
   \begin{array}{cc}
   T \vdash_{dy} s \\
   \hline
   \text{eq}(s, s)
   \end{array}
   \]

4. $P = 0Q$ for some non-empty $Q \subseteq \mathcal{A}(S \cup \{x\}) \setminus \{\varepsilon\}$:

   Now, $\text{eq}(t, u)[r]_P$ can be written as $\text{eq}(t[r]_Q, u)$. Since $Q$ contains positions other than $\varepsilon$, it follows that $t$ is not a name or variable, and so is of the form $f(t^0, \ldots, t^{k-1})$. Denoting $P_x(t^i)$ by $Q_i$, we can write $Q$ as $0Q_0 \cup \cdots \cup (k-1)Q_{k-1}$. For $i < k$, letting $t^i[r]_Q = v^i$, we see that

   \[
   t[r]_Q = f(t^0[r]_{Q_0}, \ldots, t^{k-1}[r]_{Q_{k-1}}) = f(v^0, \ldots, v^{k-1}).
   \]

   From the facts $Q \subseteq \mathcal{A}(T \cup \{x\}, t)$ and that at least one $p \neq \varepsilon$ is in $Q$, it follows that $\{0, \ldots, k-1\} \subseteq \mathcal{A}(T, t[r]_Q)$, and it
follows that \( S \vdash_{dy} v^i \) for all \( i < k \). Hence we have \( \vdash_{eq} \) proofs of \( \text{eq}(t^i[r]_Q, v^i) \) for each \( i < k \) (by using the \( \text{eq} \) rule). For each \( i < k \), \( \text{size}(Q_i) < \text{size}(P) \) and hence there is an \( \vdash_{eq} \) proof \( \pi^i \) of \( \text{eq}(t^i[s]_Q, v^i) \), by induction hypothesis. We can now build the following \( \vdash_{eq} \) proof of \( \text{eq}(t[s]_Q, u) \).

\[
\begin{array}{cccc}
\pi^0 & \cdots & \pi^{k-1} \\
\vdash & \cdots & \vdash \\
\text{eq}(t^0[s]_Q, v^0) & \cdots & \text{eq}(t^{k-1}[s]_{Q_{k-1}}, v^{k-1}) \\
\vdash & \cdots & \vdash \\
\text{cons} & \vdash & \text{eq}(t[r]_Q, u) \\
\text{eq}(t[s]_Q, u) & \vdash & \text{eq}(t[s]_Q, u) \\
\end{array}
\]

5. \( P = \{1p \mid p \in R\} \) for some \( R \subseteq \mathbb{A}(S \cup \{x\}, u) \setminus \{\varepsilon\} \): This case is symmetric to the above.

6. \( P = \{0\} \cup \{1p \mid p \in R\} \) for some \( R \subseteq \mathbb{A}(S \cup \{x\}, u) \setminus \{\varepsilon\} \):

In this case, \( \text{eq}(t, u)[r]_P \) is really \( \text{eq}(r, u[r]_R) \), but this means that \( \pi_1 \) is a derivation of \( \text{eq}(r, w) \) for some \( w \), with \( r \) occurring as a strict subterm of \( w \). But that violates consistency of \( E \).

7. \( P = \{0p \mid p \in Q\} \cup \{1\} \) for some \( Q \subseteq \mathbb{A}(T \cup \{x\}, t) \setminus \{\varepsilon\} \): This is symmetric to the above case.

8. \( P = \{0p \mid p \in Q\} \cup \{1p \mid p \in R\} \) for \( Q \subseteq \mathbb{A}(T \cup \{x\}, t) \setminus \{\varepsilon\} \) and \( R \subseteq \mathbb{A}(T \cup \{x\}, u) \setminus \{\varepsilon\} \):

Suppose \( t = f(t^0, \ldots, t^{k-1}) \) and \( u = g(u^0, \ldots, u^{l-1}) \). If \( f \neq g \), then it is again the case that \( \pi_1 \) derives an equality that cannot be made true by any substitution, violating the consistency of \( E \). If \( f = g \) (and hence \( k = l \)), we proceed similarly to case 4 above.

So \( t = f(t^0, \ldots, t^{k-1}) \) and \( u = f(u^0, \ldots, u^{k-1}) \). Then \( \text{eq}(t, u)[r]_P \) can be written as \( \text{eq}(t[r]_Q, u[r]_R) \). As earlier, writing \( Q = \bigcup iQ_i \) and \( R = \bigcup iR_i \), we see that

\[
\begin{align*}
t[r]_Q &= f(t^0[r]_{Q_0}, \ldots, t^{k-1}[r]_{Q_{k-1}}) \\
u[r]_Q &= f(u^0[r]_{R_0}, \ldots, u^{k-1}[r]_{R_{k-1}})
\end{align*}
\]

Recall that \( \pi_1 \) is a proof in the \( \vdash_{eq} \) system of \( \text{eq}(t[r]_Q, u[r]_R) \). Since \( Q \subseteq \mathbb{A}(T \cup \{x\}, t) \), and \( R \subseteq \mathbb{A}(T \cup \{x\}, u) \), and both \( Q \) and \( R \) contain at least one non-root position, it follows that \( \{0, \ldots, k-1\} \subseteq \mathbb{A}(T \cup \{x\}, t) \cap \mathbb{A}(T \cup \{x\}, u) \). Thus, one can apply the \( \text{proj} \) rule \( k \) times on \( \pi_1 \) to get \( \vdash_{eq} \) proofs of \( \text{eq}(t[r]_{Q_i}, u[r]_{R_i}) \), for each \( i < k \). From the fact that \( \text{size}(0Q_i \cup 1R_i) < \text{size}(P) \), we can appeal to induction hypothesis to conclude that there are \( \vdash_{eq} \) proofs \( \pi^0, \ldots, \pi^{k-1} \), with each \( \pi^i \) proving \( \text{eq}(t^i[s]_Q, u[s]_{R_i}) \). We can now build the following \( \vdash_{eq} \) proof of \( \text{eq}(t[s]_Q, u[s]_R) \).
Lemma 5.2. Suppose \( B \) ends in \( \pi \). \( R \)

Proof. \( B \) ends in \( \pi \).

Right-to-left direction: This direction is easier. We just need to show that using \( \text{subst} \), we can simulate \( \text{trans} \) and \( \text{cons} \).

1. Suppose \( \pi \) and \( \pi' \) are proofs of \( (T; E) \vdash \text{core} \ \text{eq}(t, u) \) and \( (T; E) \vdash \text{core} \ \text{eq}(u, v) \), respectively. Taking \( \alpha \) to be \( \text{eq}(t, x) \), we see that \( 1 \in \text{A}(T \cup \{x\}, \alpha) \), and \( \alpha[u]_1 \) is \( \text{eq}(t, u) \), while \( \alpha[v]_1 \) is \( \text{eq}(t, v) \). Below is a proof of \( \text{eq}(t, v) \).

\[
\begin{align*}
\pi & \vdash \text{eq}(t, x), u_1 \\
\pi' & \vdash \text{eq}(t, x), u_1 \\
S & \vdash \text{eq}(u, v) \\
\text{subst} & \vdash \text{eq}(t, v)
\end{align*}
\]

The transitivity rule with multiple premises can be obtained by cascading the binary transitivity rule.

2. Suppose \( \pi^0 \) is a proof of \( (T; E) \vdash \text{core} \ \text{eq}(t^0, u^0) \) and \( \pi^1 \) is a proof of \( (T; E) \vdash \text{core} \ \text{eq}(t^1, u^1) \). Let \( \alpha \) be \( \text{eq}(f(t^0, t^1), f(x, y)) \). We see that \( 10, 11 \in \text{A}(T, \alpha) \). From \( (T; E) \vdash \text{core} \ \text{eq}(t^0, u^0) \) and \( (T; E) \vdash \text{core} \ \text{eq}(t^1, u^1) \), by the previously proved left-to-right direction and the fact that \( (T : E) \) is eq-pure, it follows that \( T \vdash_{dy} t^0 \) and \( T \vdash_{dy} t^1 \). Hence \( T \vdash_{dy} f(t^0, t^1) \), and we have the following proof of \( \text{eq}(f(t^0, t^1), f(u^0, u^1)) \).

\[
\begin{align*}
T & \vdash_{dy} f(t^0, t^1) \\
(T; E) & \vdash \text{eq}(f(t^0, t^1), f(x, y)[t^0, t^1])_{10, 11} \\
\text{eq} & \vdash f(t^0, t^1), f(x, y)[t^0, t^1] \text{eq}(t^0, u^0) \\
\pi^0 & \vdash f(t^0, t^1), f(x, y)[t^0, t^1] \text{eq}(t^0, u^0) \\
\pi^1 & \vdash f(t^0, t^1), f(x, y)[t^0, t^1] \text{eq}(t^0, u^0) \\
\text{subst} & \vdash f(t^0, t^1), f(x, y)[t^0, t^1] \text{eq}(t^0, u^0) \\
S & \vdash \text{eq}(f(t^0, t^1), f(u^0, u^1)) \\
\text{subst} & \vdash \text{eq}(f(t^0, t^1), f(u^0, u^1))
\end{align*}
\]

We can similarly handle functions of multiple arities. \( \square \)

## B Proofs for Section 5

**Lemma 5.2.** Suppose \( \lambda(r) = \lambda(s) \) for each \( \text{eq}(r, s) \in E \), and \( T; E \vdash_{eq} \text{eq}(t, u) \). Then \( \lambda(t) = \lambda(u) \).

**Proof.** Suppose \( T; E \vdash_{eq} \text{eq}(t, u) \) via a proof \( \pi \). The proof is by induction on the structure of \( \pi \). The following cases arise.

**\( \pi \) ends in \( \text{ax} \):** In this case, \( \text{eq}(t, u) \in E \), so by assumption, \( \lambda(t) = \lambda(u) \).
\( \pi \) ends in eq: In this case \( t = u \), so \( \lambda(t) = \lambda(u) \) as well.

\( \pi \) ends in trans: Suppose eq\((t_0, t_1), \ldots, eq(t_{n-1}, t_n)\) are the premises of the last rule of \( \pi \), with \( t = t_0 \) and \( u = t_n \). By IH, \( \lambda(t_{i-1}) = \lambda(t_i) \) for all \( i \leq n \). It follows that \( \lambda(t) = \lambda(u) \).

\( \pi \) ends in cons: Suppose the premises of the last rule are eq\((t_1, u_1), \ldots, eq(t_n, u_n)\), with \( t = f(t_1, \ldots, t_n) \) and \( u = f(u_1, \ldots, u_n) \). By IH, \( \lambda(t_i) = \lambda(u_i) \) for all \( i \leq n \). Thus,

\[
\lambda(t) = \lambda(f(t_1, \ldots, t_n)) = f(\lambda(t_1), \ldots, \lambda(t_n)) \quad \text{and} \quad \lambda(u) = \lambda(f(u_1, \ldots, u_n)) = f(\lambda(u_1), \ldots, \lambda(u_n)).
\]

\( \pi \) ends in proj: Let eq\((f(t_1, \ldots, t_n), f(u_1, \ldots, u_n))\) be the premise of the last rule with \( t = t_i \) and \( u = u_i \) respectively. By IH, \( \lambda(f(t_1, \ldots, t_n)) = \lambda(f(u_1, u_n)) \). So, \( \lambda(t) = \lambda(u) \).

Lemma 5.3. Let \( t, u \in \mathcal{T} \). For any \( i \in \{1, \ldots, n\} \) and any \( l \in \{0, \ldots, n\} \),

1. if \( \gamma \in D_i \cup E_i \), then \( \tilde{\mu}(t) = \tilde{\mu}(u) \).
2. if \( \sigma(T_{i-1}; E_{i-1}) \vdash_{eq} \sigma\mu_i(eq(t, u)) \), then \( \tilde{\mu}(t) = \tilde{\mu}(u) \).
3. if \( \sigma(S_i; D_i) \vdash_{eq} \sigma\theta_i(eq(t, u)) \), then \( \tilde{\mu}(t) = \tilde{\mu}(u) \).

Proof. Since \( E_0 = \emptyset \), claim 1 is vacuously true for \( l = 0 \). We prove the claims simultaneously by induction on \( i \) for \( i > 0 \). Assume that they hold for all \( j < i \) via IH1, IH2, and IH3.

1. Suppose eq\((t, u) \in E_i \). Then, eq\((t, u) \in HE_j \) for some \( j < i \), and thus \( \sigma(S_j); \sigma(E_j) \vdash_{eq} \sigma\theta_j(eq(t, u)) \). By IH3, \( \tilde{\mu}(t) = \tilde{\mu}(u) \). Now suppose eq\((t, u) \in D_i \). Then eq\((t, u) \in IE_j \) for some \( j \leq i \), and so \( \sigma(T_{j-1}); \sigma(E_{j-1}) \vdash_{eq} \sigma\mu_j(eq(t, u)) \). If \( j < i \), by IH2, \( \tilde{\mu}(t) = \tilde{\mu}(u) \). If \( j = i \), by IH1, \( \tilde{\mu}(r) = \tilde{\mu}(s) \) for every eq\((r, s) \in E_{i-1} \). Any eq\((\hat{r}, \hat{s}) \in \sigma(E_{i-1}) \) is of the form eq\((\sigma(r), \sigma(s)) \) for some eq\((r, s) \in E_{i-1} \). Thus, \( \tilde{\mu}(\hat{r}) = \tilde{\mu}(\sigma(r)) = \tilde{\mu}(r) = \tilde{\mu}(s) = \tilde{\mu}(\sigma(s)) = \tilde{\mu}(\hat{s}) \). By Lemma 5.2, \( \tilde{\mu}(\sigma\mu_j(t)) = \tilde{\mu}(\sigma\mu_j(u)) \), i.e. \( \tilde{\mu}(t) = \tilde{\mu}(u) \).

2. Suppose \( \sigma(T_{i-1}); \sigma(E_{i-1}) \vdash_{eq} \sigma\mu_i(eq(t, u)) \). As earlier, for each eq\((\hat{r}, \hat{s}) \in \sigma(E_{i-1}) \), \( \tilde{\mu}(\hat{r}) = \tilde{\mu}(\hat{s}) \). By Lemma 5.2, \( \tilde{\mu}(\sigma\mu_i(t)) = \tilde{\mu}(\sigma\mu_i(u)) \), i.e. \( \tilde{\mu}(t) = \tilde{\mu}(u) \).

3. The proof follows similarly to that for claim 2.

Lemma 5.6. If \( x \) is minimal and \( \sigma(x) \in st(\sigma(T_p)) \), then there is an \( r < p \) such that \( \sigma(x) \in st(\sigma\mu_r(IT_r)) \).

Proof. Suppose \( \sigma(x) \in st(\sigma(T_p)) \), i.e. \( \sigma(x) \in st(\sigma(u)) \) for some \( u \in T_p \). Since \( x \) is minimal, \( \sigma(x) \neq \sigma(v) \) for any \( v \in st^- \), so \( \sigma(x) \in st(\sigma(y)) \) for \( y \in vars(u) \cap \mathcal{V}_\sigma \). Since \( u \in T_p \), there is a \( q < p \) such that \( u \in HT_q \cup HV_q \). Thus \( y \in vars(HT_q) \cup HV_q \). But since \( y \in \mathcal{V}_\sigma \), we get \( y \notin HV_q \), and so \( y \in vars(HT_q) \). Thus, there is a \( r \leq q \) such that \( y \in vars(IT_r) \). Therefore, \( \sigma(y) \in st(\sigma(IT_r)) \) and \( \sigma(x) \in st(\sigma(IT_r)) \). The only terms in \( st(\sigma(IT_r)) \setminus st(\sigma\mu_r(IT_r)) \) are variables in \( \mathcal{V}_q \). Since \( \sigma(x) \) is ground, it also belongs to \( st(\sigma\mu_r(IT_r)) \).
Proposition 5.7. Suppose $i \leq n$, $x$ is minimal, and $s \in \text{st}$. Let $\lambda \in \{\sigma, \sigma_{\mu_i}\}$. If $\sigma(T_{i-1}; E_{i-1}) \vdash_{eq} \lambda(\text{eq}(x, s))$, then either $s$ is minimal, or $s \in \mathcal{V} \setminus \mathcal{V}_i^\sigma$.

Proof. We give the proof for $\sigma_{\mu_i}$. The proof for $\sigma$ is similar.

Suppose $\sigma(T_{i-1}); \sigma(E_{i-1}) \vdash_{eq} \sigma_{\mu_i}(\text{eq}(x, r))$. By Lemma 5.3, $\tilde{\mu}(x) = \tilde{\mu}(r)$. But $x \in \mathcal{V}_i^\sigma$, so $\tilde{\mu}(x) = \sigma(x)$, and $\sigma(x) = \tilde{\mu}(r)$. If $r$ is not minimal or a variable outside $\mathcal{V}_i^\sigma$, two cases arise:

- **$r \in \mathcal{V}_i^\sigma$ is not minimal:** There is a $t \in \text{st}^-$ with $\text{vars}(t) \subseteq \mathcal{V}_i^\sigma$ such that $\tilde{\mu}(r) = \sigma(r) = \sigma(t)$.

  But then, $\sigma(x) = \tilde{\mu}(r) = \sigma(t)$, which contradicts the minimality of $x$.

- **$r \in \text{st}^-$:** By Prop. 5.4, there is a $t \in \text{st}$ with $\text{vars}(t) \subseteq \mathcal{V}_i^\sigma$ such that $t$ is an instance of $r$ and $\tilde{\mu}(r) = \sigma(t)$. Since $r \in \text{st}^-$ and $t$ is an instance of $r$, $t \in \text{st}^-$ as well. So $\sigma(x) = \tilde{\mu}(r) = \sigma(t)$, contradicting $x$ being minimal.

Thus $r \in \mathcal{V}$ is minimal or $r \notin \mathcal{V}_i^\sigma$.

\[\square\]

Proposition 5.18. If $t = f(t_1, \ldots, t_n) \in \text{sst}$ and $t$ is not zappable, then each $t_i \in \text{sst}$.

Proof. $t \in \text{sst}$ and so there is an $r \in \text{st}$ such that $t = \sigma\lambda(r)$ for $\lambda \in \{\sigma, \theta_i, \mu_i \mid i \leq n\}$. The following cases arise.

- **$r = f(r_1, \ldots, r_n)$:** Then $r_i \in \text{st}$ and $t_i = \sigma\lambda(r_i) \in \text{sst}$.

- **$r = x \in \mathcal{V}_i^\sigma$:** Since $t$ is not zappable, $x$ is not minimal. So there is an $r = f(r_1, \ldots, r_n) \in \text{st}^-$ such that $t = \sigma(r)$, and each $t_i \in \text{sst}$ as above.

- **$r = x \in \text{dom}(\lambda)$, for $\lambda \in \{\theta_i, \mu_i \mid i \leq n\}$:** We know that for all $x \in \text{dom}(\lambda)$, $\lambda(x)$ is either a ground term or a variable. From its structure, $t = \lambda(x)$ is ground. By Proposition 5.4, $\tilde{\mu}(x) = \sigma(s)$ for some $s \in \text{st}$ with $\text{vars}(s) \subseteq \mathcal{V}_i^\sigma$. Then $t = \tilde{\mu}(t) = \tilde{\mu}(\lambda(x)) = \tilde{\mu}(x) = \sigma(s)$, and we can argue as in the previous two cases.

\[\square\]

Theorem 5.21. Let $t, u \in \text{st}$ and $i \leq n$.

- If $\sigma(T_{i-1}) \vdash_{\text{dy}} \sigma_{\mu_i}(t)$ then $\tau(T_{i-1}) \vdash_{\text{dy}} \tau_{\nu_i}(t)$.

- If $\sigma(T_{i-1}; E_{i-1}) \vdash_{eq} \sigma_{\mu_i}(\text{eq}(t, u))$ then $\tau(T_{i-1}; E_{i-1}) \vdash_{eq} \tau_{\nu_i}(\text{eq}(t, u))$.

Proof.

- Suppose there is a proof of $\sigma(T_i) \vdash_{\text{dy}} \sigma_{\mu_i}(t)$. Then there is a proof of $\tau(T_i); \tau(E_i) \vdash_{\text{dy}} \sigma_{\mu_i}(t)$, by Lemma 5.19. Clearly $\sigma_{\mu_i}(t) \in \text{sst}$ and hence $[\sigma_{\mu_i}(t)] = \sigma_{\mu_i}(t) = \tau_{\nu_i}(t)$.

  Thus we have a proof of $\tau(T_i) \vdash_{\text{dy}} \tau_{\nu_i}(t)$.

- Suppose $\sigma(T_i); \sigma(E_i) \vdash_{\text{dy}} \sigma_{\mu_i}(\text{eq}(t, u))$ with $t, u$ as stated. By Lemma 5.20, we have that $\tau(T_i); \tau(E_i) \vdash_{eq} [\sigma_{\mu_i}(\text{eq}(t, u))]$. So, $\sigma_{\mu_i}(t), \sigma_{\mu_i}(u) \in \text{sst}$. Therefore, we get $[\sigma_{\mu_i}(\text{eq}(t, u))] = \text{eq}(\sigma_{\mu_i}(t), \sigma_{\mu_i}(u)) = \text{eq}(\tau_{\nu_i}(t), \tau_{\nu_i}(u)) = \tau_{\nu_i}(\text{eq}(t, u))$. Hence we have a proof of $\tau(T_i); \tau(E_i) \vdash_{eq} \tau_{\nu_i}(\text{eq}(t, u))$.

\[\square\]
C Algorithm to decide $\vdash_{eq}$

We present a saturation-based procedure in Algorithm 1 for deciding whether $T; E \vdash_{eq} eq(t, u)$ for eq-pure and consistent $(T; E)$. The procedure first computes the set

$$\text{Conseq}(T, E) = \{eq(r, s) \mid r, s \in \text{st}(T \cup \{t, u\}) \cup \text{st}(E), T; E \vdash_{eq} eq(r, s)\}$$

and then checks whether $eq(t, u) \in \text{Conseq}(T, E)$.

We start out with a set $C$, which contains all the equalities in $E$, and trivial equalities over all the terms $t$ such that $T \vdash_{dy} t$. We use $r_i$ and $s_i$ to indicate the $i^{th}$ projections of $k$-tuples $r$ and $s$, respectively.

$C$ is initially defined such that it contains all equalities obtained by applying the $\text{ax}$ and $\text{eq}$ rules. $C_1$ corresponds to all equalities obtained by one application of the $\text{trans}$ rule, $C_2$ to all those obtained using $\text{cons}$, and $C_3$ to all those obtained using $\text{proj}$. All these sets are added to $C$ and the procedure iterated till nothing new can be added to $C$.

Letting $\text{st} = \text{st}(T \cup \{t, u\}) \cup \text{st}(E)$, and $M = |\text{st}|$, it can be seen that the algorithm runs in time polynomial in $M$. There are at most $M^2$ equalities that can be added in $C$, and hence the while loop runs for at most $M^2$ iterations. In each iteration, the amount of work to be done is polynomial in $M$. (Recall that $\vdash_{dy}$ can be decided in PTIME.) Thus the algorithm works in time polynomial in $M$.

Algorithm 1 Algorithm to compute $\text{Conseq}(T, E)$, given eq-pure and consistent $(T; E)$

1: $\text{st} \leftarrow \text{st}(S \cup \{t, u\}) \cup \text{st}(E);$  
2: $B \leftarrow \emptyset;$  
3: $C \leftarrow A \cup \{eq(t, t) \mid t \in \text{st}, T \vdash_{dy} t\};$  
4: while $(B \neq C)$ do  
5:   $B \leftarrow C;$  
6:   $C_1 \leftarrow \{eq(r, s) \mid r, s \in \text{st}, \text{and there is a } v \text{ such that } \{eq(r, v), eq(v, s)\} \subseteq B\};$  
7:   $C_2 \leftarrow \{eq(r, s) \mid r, s \in \text{st}, \{eq(r_i, s_i) \mid 1 \leq i \leq k\} \subseteq B\};$  
8:   $C_3 \leftarrow \{eq(r_i, s_i) \mid eq(r, s) \in B, \text{ and for all } j \leq k, S \vdash_{dy} r_j \text{ and } S \vdash_{dy} s_j\};$  
9:   $C \leftarrow B \cup C_1 \cup C_2 \cup C_3;$  
10: end while  
11: return $B.$

D Extending the results to list membership

We now describe how to incorporate list membership assertions into our system, and show that the results can be extended. These assertions are of the form $\text{in}(t, [n_1, \ldots, n_k])$, where $t \in T$ and each $n_i \in N$. The notion of subformula is extended by adding the clause $\text{sf}(\text{in}(t, [n_1, \ldots, n_k])) = \{\text{in}(t, [n_1, \ldots, n_k])\}$. But when we consider subterms, we treat list terms as an indivisible unit. So $\text{st}(\text{in}(t, [n_1, \ldots, n_k])) = \text{st}(t) \cup \{[n_1, \ldots, n_k]\}$ (rather than
A structured assertion is defined to be of the form
\[ \alpha := \exists x_1 \ldots x_n \{ \text{eq}(\text{pat}, t) \land \text{eq}(r_1, s_1) \land \ldots \land \text{eq}(r_k, s_k) \land \text{in}(u_1, l_1) \land \ldots \land \text{in}(u_p, l_p) \} \]

Apart from the other conditions for structured assertions, we demand that each \( u_i \in \text{vars}(\text{pat}) \cup \mathcal{N} \). The public terms of \( \alpha \) are defined as:
\[
\text{pubs}(\alpha) = \{ t \} \cup \bigcup_{i \leq k} \text{names}(r_i, s_i) \cup \bigcup_{j \leq p} \text{names}(u_j, l_j)
\]

We add the following clauses for the definitions of positions, and abstractable positions.

- \( \mathbb{P}(\text{in}(t, [n_1, \ldots, n_k])) = \{ 0 \} \cup \{ 1 \cdot i \mid i \in \{ 1, \ldots, k \} \} \)
- \( \mathcal{A}(S, \text{in}(t, [n_1, \ldots, n_k])) = \{ 0 \cdot p \mid p \in \mathcal{A}(S, t) \} \)

We add the derivation rules given in Table 3 to the \( \vdash \) system, in order to handle membership assertions.

| Rule     | Premise 1            | Premise 2            | Conclusion          |
|----------|----------------------|----------------------|---------------------|
| \( S; A \vdash \text{in}(t, n) \)_prom | \( S; A \vdash \text{in}(t, l_1) \ldots \text{in}(t, l_m) \) | \( S; A \vdash \text{in}(t, \bigcap_{i=1}^m l_i) \) | \( S; A \vdash \text{in}(t, [n_1, \ldots, n_k]) \) |
| \( S; A \vdash \text{eq}(t, n) \) wk | \( S; A \vdash \text{eq}(t, n) \) wk | \( S; A \vdash \text{eq}(t, n) \) wk | \( S; A \vdash \text{eq}(t, n) \) wk |

Table 3: Rules for list membership

We add all these rules to the to the \( \vdash_{eq} \) system as well, along with a \textbf{subst} rule for membership assertions. This rule is of the following form, with \( \text{in}(t, l) \) as the major premise, and \( \text{in}(u, l) \) as the conclusion.

\[
(T; E) \vdash \text{in}(t, l) \quad (T; E) \vdash \text{eq}(t, u) \quad \textbf{subst} \quad (T; E) \vdash \text{in}(u, l)
\]

Note that we do not allow substitutions in the list part of a membership assertion. We do not lose much expressive power due to this, since we only have names inside lists.

We now change the definition of \( \text{trim}(B) \) (in Definition 3.4) to be the set of all atomic formulas of \( B \) (both equality and membership formulas). The definition of \( \text{dc} \) is the same, except that it uses the new definition of \( \text{trim} \).

Suppose \( (T; E) \) is eq-pure. We say that \( E \) is consistent if there is \( \lambda \) s.t. \( \lambda(t) = \lambda(u) \) for each \( \text{eq}(t, u) \in E \), and \( \lambda(t) \in \{ n_1, \ldots, n_k \} \) for each \( \text{in}(t, [n_1, \ldots, n_k]) \in E \).

We add the following clauses to the definition of normal \( \vdash_{eq} \) proofs.

- No premise of \textbf{int} is the conclusion of \textbf{int} or \textbf{wk}
- No premise of \textbf{subst} is the conclusion of \textbf{subst}
Recall that each subterm of a membership assertion is either a list of names or a term. The reason for this is as follows: we want to bound the number of distinct assertions that occur in a normal $\vdash_{eq}$ proof. Since we allow lists, we can form multiple combinations (in fact exponentially many) from any set of names. So it does not suffice to bound the number of non-list terms occurring in the proof – we must bound the number of lists that can occur in a normal proof. The subterm property for normal $\vdash_{eq}$-proofs is defined as follows. Let $\alpha$ be either $\text{eq}(t, u)$ or $\text{in}(t, l)$. Let $T_\alpha$ be $\{ t, u \}$ in the former case, and $\{ t, l \}$ in the latter case. Let $\pi$ be a normal proof of $T; E \vdash_{eq} \alpha$, with normal $\vdash_{dy}$ subproofs. Let $\xi$ be a subproof of $\pi$ with conclusion $T; E \vdash_{eq} \beta$.

- If $\beta = \text{eq}(r, s)$, then $r, s \in \text{st}(T \cup T_\alpha) \cup \text{st}(E)$. Further, if $\text{cons}$ does not occur in $\pi$, then $r, s \in \text{st}(T) \cup \text{st}(E) \cup \mathcal{N}_{\text{pub}}$.
- If $\beta = \text{in}(r, l')$, then $r \in \text{st}(T \cup T_\alpha)$ and either $l' \in \text{st}(T \cup T_\alpha)$ or $l' = [n]$, where $n \in \text{st}(T \cup T_\alpha)$.

We prove the normalization and subterm properties (for $\text{eq}$-pure $(T; E)$ with consistent $E$) in Appendix F.

Finally, consider the changes needed in Section 5 to handle list membership. Most of the reasoning is with equality. We only need to show that proofs of list membership under $\sigma$ can be translated to proofs under $\tau$. So suppose $\pi$ is a proof of $\sigma(T_{i-1}; E_{i-1}) \vdash_{eq} \sigma(\mu_i(t, [n_1, \ldots, n_k]))$. We know that $\mu_i(t) \in \{n_1, \ldots, n_k\}$, so $t \in \mathcal{N} \cup \mathcal{V}$ and each $n_i \in \mathcal{N}$. If $t \in \mathcal{N}$, it has to be one of the $n_i$s, and we can always prove the membership formula under $\tau \nu_i$ as well. Otherwise $t$ is some $x \in \text{dom}(\sigma \mu_i)$ and $\sigma \mu_i(x) \in \{n_1, \ldots, n_k\}$, and we define $\tau(x)$ to be the same as $\sigma(x)$ (or $\nu_i(x) = \mu_i(x)$, in case $x \in \text{dom}(\mu_i)$). Since each $n_i \in \mathcal{N}$, it is easy to see that all the membership assertions can be derived under $\tau$ and $\nu_i$. So we see that we can always find small substitutions to preserve list membership as well. We can extend the PTIME algorithm for $\vdash_{eq}$ (presented in Appendix C) to list membership as well, by utilising the subterm property.

## E Normalization and subterm property for $\vdash_{eq}$

Here we prove the normalization theorem and the subterm property for the $\vdash_{eq}$ system. We first recall the relevant definition of normal derivations.

**Definition 3.11** (Normality for $\vdash_{eq}$). For $T \subseteq \mathcal{T}$ and a set $E \cup \{ \text{eq}(t, u) \}$ of equalities, we say that a proof $\pi$ of $T; E \vdash_{eq} \text{eq}(t, u)$ is normal if it satisfies:

1. the premise of an $\text{eq}$ is only the conclusion of a destructor rule or $\text{pb}$,
2. no premise of a $\text{trans}$ is of the form $\text{eq}(a, a)$, or the conclusion of a $\text{trans}$,
3. adjacent premises of a $\text{trans}$ are not conclusions of $\text{cons}$, and
4. no subproof ending in $\text{proj}$ contains $\text{cons}$. 

36
Theorem 3.12 (Normalization for $\vdash_{eq}$-proofs). Let $(T; E)$ be eq-pure with consistent $E$. If $(T; E) \vdash_{eq} eq(t, u)$, then there is a normal $\vdash_{eq}$-proof of $(T; E) \vdash eq(t, u)$.

Since $(T; E)$ is consistent, there is $\lambda$ s.t. $\lambda(r) = \lambda(s)$ for each $eq(r, s) \in E$. It follows from Lemma 5.2 that $\lambda(t) = \lambda(u)$ whenever $(T; E) \vdash_{eq} eq(t, u)$. Therefore it cannot be the case that $t = f(\cdots)$ and $u = g(\cdots)$, with $f \neq g$.

Proof. Let $\pi$ be any proof of $T; E \vdash eq(\hat{t}, \hat{u})$ such that all DY subproofs of $\pi$ are normal, and let $\rho$ be any subproof of $\pi$. We introduce the following rewrite rules, each of which replaces $\rho$ by a better proof (in case $\rho$ contributes to $\pi$ being non-normal).

In presenting the rewrite rules, we reduce clutter by simply writing $eq(t, u)$ instead of $T; E \vdash eq(t, u)$, since the LHS is the same for all sequents occurring in an $eq$-proof.)

1. Suppose $\rho$ ends in $eq$ and the immediate subproof ends in a constructor rule other than $pb$. Then we apply the following rewrite.

\[
\begin{array}{c}
\pi_1 \\
\vdots \\
T \vdash_{dy} t_1 \\
\vdots \\
\pi_r \\
\vdots \\
T \vdash_{dy} t_r
\end{array}
\]

\[
\begin{array}{c}
\vdash f
\end{array}
\]

\[
\begin{array}{c}
T \vdash_{dy} f(t_1, \ldots, t_r)
\end{array}
\]

\[
\begin{array}{c}
\vdash eq(f(t_1, \ldots, t_r), f(t_1, \ldots, t_r))
\end{array}
\]

\[
\begin{array}{c}
\downarrow
\end{array}
\]

\[
\begin{array}{c}
\pi_1 \\
\vdots \\
T \vdash_{dy} t_1 \\
\vdots \\
\pi_r \\
\vdots \\
T \vdash_{dy} t_r
\end{array}
\]

\[
\begin{array}{c}
\vdash eq(t_1, t_1) \\
\vdots \\
\vdash eq(t_r, t_r)
\end{array}
\]

\[
\begin{array}{c}
T \vdash_{dy} eq(t_1, t_1) \\
\vdots \\
T \vdash_{dy} eq(t_r, t_r)
\end{array}
\]

\[
\begin{array}{c}
eq (f(t_1, \ldots, t_r), f(t_1, \ldots, t_r))
\end{array}
\]

2. Suppose $\rho$ ends in $\text{trans}$ and one of the immediate premises proves the equality of two
identical terms. Then we apply the following rewrite.

3. Suppose $\rho$ ends in $\text{trans}$ and one of the immediate subproofs ends in $\text{trans}$. Then we apply the following rewrite.

4. Suppose $\rho$ ends in $\text{trans}$ and two adjacent immediate subproofs end in $\text{cons}$. Then we
apply the following rewrite, assuming that $t_e = f(t^1_e, \ldots, t^k_e)$ for $e \in \{i-1, i, i+1\}$.

\[
\begin{array}{cccc}
\pi^1_{i-1} & \pi^k_{i-1} & \pi^1_i & \pi^k_i \\
\vdots & \cdots & \vdots & \cdots \\
\end{array}
\]

apply the following rewrite, assuming that $t_e = f(t^1_e, \ldots, t^k_e)$ for $e \in \{i-1, i, i+1\}$.

5. Suppose $\rho$ ends in $\text{proj}$ and its immediate subproof ends in $\text{cons}$. Then we apply the following rewrite.

\[
\begin{array}{cccc}
\pi^1_{i-1} & \pi^1_i & \pi^k_{i-1} & \pi^k_i \\
\vdots & \vdots & \vdots & \vdots \\
\end{array}
\]

5. Suppose $\rho$ ends in $\text{proj}$ and its immediate subproof ends in $\text{cons}$. Then we apply the following rewrite.

6. Suppose $\rho$ ends in $\text{proj}$ and the immediate subproof ends in $\text{trans}$, and one if its immediate subproofs ends in $\text{cons}$. Then $\rho$ has the following structure.

\[
\begin{array}{cccc}
\pi^1_{i-1} & \pi^k_{i-1} & \pi^1_i & \pi^k_i \\
\vdots & \cdots & \vdots & \cdots \\
\end{array}
\]

6. Suppose $\rho$ ends in $\text{proj}$ and the immediate subproof ends in $\text{trans}$, and one if its immediate subproofs ends in $\text{cons}$. Then $\rho$ has the following structure.

\[
\begin{array}{cccc}
\pi^1_{i-1} & \pi^k_{i-1} & \pi^1_i & \pi^k_i \\
\vdots & \cdots & \vdots & \cdots \\
\end{array}
\]
Since the \textit{proj} rule is applied on \textit{eq}(t_1, t_r)$, there is some constructor $f$ such that $t_e = f(t^1_e, \ldots, t^k_e)$ for $e \in \{1, r\}$. Since \textit{eq}(t_i, t_{i+1})$ is obtained by the \textit{cons} rule, there is some constructor $g$ such that $t_e = g(t^1_e, \ldots, t^r_e)$ for $e \in \{i, i + 1\}$. But \textit{eq}(t_1, t_i)$ is provable from $(T, E)$, which is consistent. Therefore it cannot be the case that $f \neq g$. Thus we see that for all $e \in \{1, i, i + 1, r\}, t_e = f(t^1_e, \ldots, t^r_e)$. We can rewrite $\rho$ as follows.

\[
\begin{array}{c|c|c|c|c}
\pi_1 & \cdots & \pi_{i-1} & \pi_{i+1} & \cdots & \pi_{r-1} \\
\vdots & & \vdots & \vdots & & \vdots \\
\text{eq}(t_1, t_2) & \cdots & \text{eq}(t_{i-1}, t_i) & \text{eq}(t_{i+1}, t_{i+2}) & \cdots & \text{eq}(t_{r-1}, t_r) \\
\text{eq}(t_i, t_i) \quad \text{trans} & & & & & \\
\text{eq}(t^1_i, t^1_i) \quad \text{proj} & & & & & \\
\text{eq}(t^j_i, t^{j+1}_i) \quad \text{eq}(t^j_{i+1}, t^j_{i+1}) \quad \text{trans} & & & & & \\
\text{eq}(t^l_i, t^l_i) & & & & & \\
\end{array}
\]

Note that we can apply \textit{proj} on \textit{eq}(t_1, 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 \textit{proj} rule was applied to $\textit{eq}(t_1, t_r)$ in $\rho$; and for $t_i$ this follows from the fact that $\pi^j_i$ derives $\textit{eq}(t^j_i, t^{j+1}_i)$ for each $j \leq k$, and whenever there is an equality proof of $\textit{eq}(t, u)$, it is the case that $T \vdash_{dy} t$, and so $T \vdash_{dy} t^j_i$ for all $i \leq k$. For a similar reason, we can apply \textit{proj} on \textit{eq}(t_{i+1}, t_r)$.

Suppose we repeatedly apply the above rewrites (in some order) starting with $\pi$ and reach a proof $\xi$ on which we can no longer apply any of the rules. Then $\xi$ satisfies clauses 1 to 4 in the definition of normal proofs (since the rewrite rules 1 to 4 cannot be applied to $\xi$). Clause 5 in the definition of normal proofs is also satisfied by $\xi$, for the following reason. Suppose a subproof $\xi_1$ ends in \textit{proj} and $\xi_2$ is a maximal subproof of $\xi_1$ ending in \textit{cons}. Since $\xi_2$ is maximal, it is clear that $\xi_2$ is not a premise of a proof ending in \textit{cons}. Since rewrite rule 5 cannot be applied anymore, we know that that $\xi_2$ is not the premise of a proof ending in \textit{proj}.

Suppose $\xi_2$ is a premise of a proof $\widehat{\xi}$ ending in \textit{trans}. Since $\xi_2$ is a subproof of $\xi_1$, which ends in \textit{proj}, it has to be that $\widehat{\xi}$ itself is a subproof of $\xi_1$, and thus is a premise of some proof. Since rewrite rules 3 and 6 cannot be applied anymore, $\widehat{\xi}$ itself cannot be a premise of a proof ending in \textit{trans} or \textit{proj}. By maximality of $\xi_2$ again, $\widehat{\xi}$ cannot be a premise of a proof ending in \textit{cons}. Clearly $\widehat{\xi}$ cannot be a premise of a proof ending in \textit{ax} or \textit{eq}. Thus we are forced to conclude that $\xi_2$ is not a premise of a proof ending in \textit{trans}. We have ruled out all possible cases, and thus we are forced to conclude that $\xi_2$ cannot be a subproof of $\xi_1$. Thus we see that \textit{cons} does not occur in any subproof of $\xi$ ending in \textit{proj}. Thus $\xi$ satisfies all the clauses in the definition of normal proofs, and hence $\xi$ is normal.

We next show below that if we apply the rewrite rules repeatedly (in any order whatsoever), the sequence will terminate (and end in a normal proof).

Associate three sizes to an $\vdash_{eq}$-proof $\pi$: $\eta_1(\pi)$ is the sum of the sizes of the $\vdash_{dy}$ subproofs of $\pi$, $\eta_2(\pi)$ is the number of \textit{cons} rules that occur in $\pi$, and $\eta_3(\pi)$ is just the size of the proof $\pi$ (number of nodes in the proof tree). Let $\eta(\pi) = (\eta_1(\pi), \eta_2(\pi), \eta_3(\pi))$.

We now show that if $\pi'$ is obtained from $\pi$ by applying any of the rewrite rules once, $\eta(\pi') < \eta(\pi)$. 40
• If rule 1 is applied, \( \eta_1(\pi') < \eta_1(\pi) \) and so \( \eta(\pi') < \eta(\pi) \).

• If rule 2 or 3 is applied, we have that \( \eta_i(\pi') \leq \eta_i(\pi) \) for \( i \in \{1, 2\} \) and \( \eta_3(\pi') < \eta_3(\pi) \). So \( \eta(\pi') < \eta(\pi) \).

• If rule 4, 5 or 6 is applied, we have \( \eta_1(\pi') \leq \eta_1(\pi) \) and \( \eta_2(\pi') < \eta_2(\pi) \). Therefore, \\

From this, it follows that we cannot have an infinite sequence of rewrites starting from any \( \pi \) (since lexicographic ordering on triples of natural numbers is a well-ordering). Thus every proof \( \pi \) can be transformed to a normal proof \( \xi \) with the same conclusion. \( \square \)

**Theorem 3.13** (Subterm property for \( \vdash_{eq} \)). Let \( \pi \) be a normal proof of \( T; E \vdash_{eq} eq(t, u) \), with normal \( \vdash_{dy} \) subproofs. Let \( \xi \) be a subproof of \( \pi \) ending in \( T; E \vdash eq(r, s) \). Then, \( r, s \in st(T \cup \{ t, u \}) \cup st(E) \). Further, if \textbf{cons} does not occur in \( \pi \), then \( r, s \in st(T) \cup st(E) \cup \mathcal{N}_{pub} \).

In the following proof, we implicitly use Proposition 3.14, which importantly states that in any normal proof ends in \textbf{trans} and an immediate subproof does not end in \textbf{cons}, it does not contain \textbf{cons} at all.

**Proof.** By induction on the structure of \( \pi \), we prove the following stronger induction statement.

• If \( \pi \) ends in \textbf{eq}, then \( r, s \in st(T) \cup \mathcal{N}_{pub} \).

• If \( \pi \) does not contain \textbf{cons} and does not end in \textbf{eq}, then \( r, s \in st(T) \cup st(E) \).

Let \( r \) be the last rule of \( \pi \). We have the following cases.

\textbf{r is ax:} In this case, the only subproof of \( \pi \) is itself, and \( eq(r, s) = eq(t, u) \). Thus \( eq(r, s) \in E \) and hence \( r, s \in st(E) \).

\textbf{r is eq:} In this case \( t = u \) and \( T \vdash_{dy} t \). Since \( \pi \) is a normal proof whose DY subproofs are also normal, \( T \vdash_{dy} t \) is obtained via a proof ending in a destructor rule or \textbf{pb}, and by subterm property for normal DY proofs, it follows that \( t \in st(T) \cup \mathcal{N}_{pub} \). Since the only subproofs of \( \pi \) other than itself are \( \vdash_{dy} \) proofs, \( eq(r, s) = eq(t, u) = eq(t, t) \), and hence \( r = s = t \). Hence \( r, s \in st(T) \cup \mathcal{N}_{pub} \). It is also the case that \( r, s \in st(T \cup \{ t, u \}) \).

\textbf{r is cons:} Let us say \( t = f(t^1, \ldots, t^k) \) and \( u = f(u^1, \ldots, u^k) \), and for each \( i \leq k \), there is a subproof \( \pi_i \) with conclusion \( T; E \vdash eq(t^i, u^i) \). Now any \( eq(r, s) \) occurring in \( \pi \) occurs in one of the \( \pi_i \)'s or is the same as \( eq(t, u) \). In the latter case, clearly \( r, s \in st(T \cup \{ t, u \}) \).

In the former case, suppose \( eq(r, s) \) occurs in \( \pi_i \). By induction hypothesis, \( r, s \in st(T \cup \{ t^i, u^i \}) \cup st(E) \). Since \( t^i \in st(t) \) and \( u^i \in st(u) \), it follows that \( r, s \in st(T \cup \{ t, u \}) \cup st(E) \).

\textbf{r is trans:} Suppose the subproofs of \( \pi \) are \( \pi^1 \) through \( \pi^{r-1} \) with conclusions \( T; E \vdash eq(t_1, t_2) \) through \( T; E \vdash eq(t_{r-1}, t_r) \) respectively. Thus \( t = t_1 \) and \( t_r = u \). Since \( \pi \) is a normal proof, no two adjacent premises of \textbf{r} are obtained by \textbf{cons}, and no premise of \textbf{r} is
obtained by \textbf{trans}. Furthermore, by normality, any immediate subproof of $\pi$ that does not end in \textbf{cons} does not have an occurrence of \textbf{cons}. Now suppose $(r = s)$ occurs in $\pi$. There are several cases to consider:

- \textbf{eq}(r, s)$ is the same as \textbf{eq}(t, u)$. Then $r, s \in st(T \cup \{t, u\})$.
- \textbf{eq}(r, s)$ occurs in $\pi_i$ which does not end in \textbf{cons}. Since $\pi^i$, by Proposition 3.14, we have that \textbf{cons} does not occur in $\pi^i$. By induction hypothesis we know that $r, s \in st(T) \cup st(E)$, and the conclusion follows.
- \textbf{eq}(r, s)$ occurs in $\pi_i$ which ends in \textbf{cons}, and $1 < i < r - 1$. Both $\pi^{i-1}$ and $\pi^{i+1}$ exist, and both end in a rule other than \textbf{trans, cons, or eq}. So \textbf{cons} does not occur in $\pi^{i-1}$ and $\pi^{i+1}$ (by Proposition 3.14), and by induction hypothesis, any term occurring in $\pi^{i-1}$ or $\pi^{i+1}$ is in $st(T) \cup st(E)$. In particular, $t_i$ occurs in $\pi^{i-1}$ and $t_{i+1}$ occurs in $\pi^{i+1}$ and so $t_i, t_{i+1} \in st(T) \cup st(E)$. Since \textbf{eq}(r, s)$ occurs in $\pi^i$, we have by induction hypothesis that $r, s \in st(T \cup \{t_i, t_{i+1}\}) \cup st(E) \subseteq st(T) \cup st(E)$.
- \textbf{eq}(r, s)$ occurs in $\pi_i$ which ends in \textbf{cons}. By normality of $\pi$, we see that $\pi^2$ ends in a rule other than \textbf{trans, cons or eq}. So $\pi^2$ ends in \textbf{ax} or \textbf{proj} and \textbf{cons} does not occur in $\pi^2$. By induction hypothesis, any term occurring in $\pi^2$ (in particular $t_2$) is in $st(T) \cup st(E) \cup \mathcal{M}_{pub}$. Since \textbf{eq}(r, s)$ occurs in $\pi^1$, we have by induction hypothesis that $r, s \in st(T \cup \{t_1, t_2\}) \cup st(E)$. But $t_2 \in st(T) \cup st(E)$ and $t_1 = t$. Thus $r, s \in st(T \cup \{t, u\}) \cup st(E)$.
- \textbf{eq}(r, s)$ occurs in $\pi^{r-1}$ which ends in \textbf{cons}. The proof is similar to the above.

\textbf{r is proj:} Let us say that \textbf{eq}(t, u)$ is got by \textbf{proj} from a proof $\pi'$ with conclusion \textbf{eq}(\hat{t}, \hat{u})$. Since $\pi$ is normal, \textbf{cons} does not occur in $\pi$ (as well as $\pi'$). We consider two cases.

- Suppose $\pi'$ ends in \textbf{eq}. Then $\hat{t} = \hat{u}$, and $t = u$. The only \textbf{eq}(r, s)$ occurring in $\pi$ are \textbf{eq}(\hat{t}, \hat{t})$ and \textbf{eq}(t, t)$. By induction hypothesis applied to $\pi'$, $\hat{t} \in st(T) \cup \mathcal{M}_{pub}$. But since \textbf{proj} was applied on \textbf{eq}(\hat{t}, \hat{t})$, we see that $\hat{t}$ is non-atomic, and hence not in $\mathcal{M}_{pub}$. Thus $\hat{t}$, and hence $t$ (which is a subterm of $\hat{t}$), belong to $st(T)$, and we are done.
- Suppose $\pi'$ does not end in \textbf{eq}. By normality, it does not contain \textbf{cons}. Hence, by induction hypothesis, for any \textbf{eq}(r, s)$ occurring in $\pi'$, $r, s \in st(T) \cup st(E)$. In particular $\hat{t}, \hat{u} \in st(T)\cup st(E)$. Now if \textbf{eq}(r, s)$ occurs in $\pi$, it either occurs in $\pi'$ or is the same as \textbf{eq}(t, u)$. In the former case, we have just seen that $r, s \in st(T) \cup st(E)$. In the latter case, since $t, u \in st(\{\hat{t}, \hat{u}\}) \subseteq st(T) \cup st(E)$, $r, s \in st(T) \cup st(E)$.

\textbf{F Extending normalization to membership assertions}

We follow the strategy in Appendix \textbf{E} to prove normalization in the presence of membership assertions. The proof rewrite rules are the following:
1. If a premise of \textbf{int} is the conclusion of \textbf{int}, then we perform the following rewrite, where we let \( l_{km} = l_k \cap \cdots \cap l_m \) and \( l_{1n} = l_1 \cap \cdots \cap l_n \).

\[ \begin{array}{c}
\pi_1 & \pi_{k-1} & \pi_k & \pi_{m+1} & \pi_m & \pi_n \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
in(t, l_1) & \cdots & \in(t, l_k) & \cdots & \in(t, l_{km}) & \cdots & \in(t, l_m) & \cdots & \in(t, l_{m+1}) & \cdots & \in(t, l_n) \\
\hline
\end{array} \]

\[ \begin{array}{c}
in(t, l_{1n}) \\
\downarrow \\
\begin{array}{c}
\pi_1 & \pi_{k-1} & \pi_k & \pi_{m+1} & \pi_m & \pi_n \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
in(t, l_1) & \cdots & \in(t, l_k) & \cdots & \in(t, l_{m+1}) & \cdots & \in(t, l_n) \\
\hline
\end{array} \]

2. If a premise of \textbf{int} is \textbf{wk}, then we perform the following rewrite. In the following, we let \( l_{1n} = l_1 \cap \cdots \cap l_n = [n_1, \ldots, n_k] \). Because we are considering derivations from a consistent set and \( \pi_i \) proves \textbf{eq}(t, n) \), it has to be that \( \lambda(t) = n \). By consistency again, \( \lambda(t) = n \in \{n_1, \ldots, n_k\} \).

\[ \begin{array}{c}
\pi_1 & \cdots & \pi_i & \cdots & \pi_n \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
in(t, l_1) & \cdots & \textbf{eq}(t, n) & \cdots & \textbf{wk} \\
\hline
\end{array} \]

\[ \begin{array}{c}
in(t, l_{1n}) \\
\downarrow \\
\begin{array}{c}
\pi_1 & \cdots & \pi_i & \cdots & \pi_n \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
in(t, l_1) & \cdots & \textbf{eq}(t, n) & \cdots & \textbf{wk} \\
\hline
\end{array} \]

3. If a premise of \textbf{int} is \textbf{wk}, then we perform the following rewrite. In the following, we let \( l_{1n} = l_1 \cap \cdots \cap l_n = [n_1, \ldots, n_k] \). Because we are considering derivations from a consistent set and \( \pi_i \) proves \textbf{eq}(t, n) \), it has to be that \( \lambda(t) = n \). By consistency again,
\[ \lambda(t) = n \in \{n_1, \ldots, n_k\}. \]

\[
\begin{array}{c}
\pi_{11} \quad \pi_{12} \\
\vdots \\
\text{in}(t, l) \quad \text{eq}(t, u) \\
\hline \\
\pi_2 \\
\vdots \\
\text{in}(u, l) \quad \text{eq}(u, v) \\
\hline \\
\text{in}(v, l) \\
\end{array}
\]

\[
\begin{array}{c}
\pi_{11} \quad \pi_{12} \\
\vdots \\
\text{eq}(t, u) \quad \text{eq}(u, v) \\
\hline \\
\text{eq}(t, l) \quad \text{eq}(t, v) \\
\hline \\
\text{in}(v, l) \\
\end{array}
\]

\[
\begin{array}{c}
\pi_{11} \quad \pi_2 \\
\vdots \\
\text{eq}(t, u) \quad \text{eq}(u, v) \\
\hline \\
\text{eq}(t, l) \quad \text{eq}(t, v) \\
\hline \\
\text{in}(v, l) \\
\end{array}
\]

It is easy to see that both these rewrites reduce the size of the overall proof, so can be incorporated into the normalization proof of the previous section.

As a result of the rules, one can show that every list \( l \) occurring in a proof is part of a premise of \texttt{int} and so occurs in the LHS, or it is survives till it reaches the conclusion, or it is the premise of \texttt{prom}, in which case \( l = [n] \), where \( n \) is a name occurring in the proof. This suffices to prove the stated subterm property.