Undecidability of Static Equivalence in Leaf Permutative Theories

Serdar Erbatur[1], Andrew M. Marshall[2], Paliath Narendran[3], and Christophe Ringeissen[4]

[1] University of Texas at Dallas, Richardson, TX, USA
[2] University of Mary Washington, Fredericksburg, VA, USA
[3] University at Albany, SUNY, Albany, NY, USA
[4] Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France

1 Introduction

We are interested in knowledge problems that occur in the analysis of security protocols, namely deduction, distinguishability, and static equivalence. In this context, the capabilities of an intruder are specified by an equational theory, possibly expressed by a term rewrite system. For example, a number of decision procedures for these knowledge problems have been developed for the particular case of subterm convergent rewrite systems [1]. Non-orientable axioms are also useful in practice and the prominent case of Associativity-Commutativity (AC for short) has been successfully investigated and a number of decision procedures have been developed, for example see [1, 4]. The AC equational theory is not the only example of non-orientable axioms that deserve to be investigated. The simpler case of shallow permutative axioms, like Commutativity, and decision procedures for the above knowledge problems have been developed in [9]. In this paper, we consider permutative theories in general. For the deduction problem, decidability for the class of permutative theories can be easily proven. The main contribution of the paper is to prove the undecidability of the static equivalence problem in the more restricted class of leaf permutative theories which is a subclass of permutative theories. To retrieve decidability of static equivalence, one can restrict to some particular variable-permuting theories [8].

2 Preliminaries

We use the standard notation of equational unification [6] and term rewriting systems [5].

Notions of Knowledge. The applied pi calculus and frames are used to model attacker knowledge [2]. In this model, the set of messages or terms which the attacker knows, and which could have been obtained from observing one or more protocol sessions, are the set of terms in $Ran(\sigma)$ of the frame $\phi = \nu\tilde{n}.\sigma$, where σ is a substitution ranging over ground terms. We also need to model cryptographic concepts such as nonces, keys, and publicly known values. We do this by using names, which are essentially free constants. Here also, we need to track the names which the attacker knows, such as public values, and the names which the attacker does not know a priori, such as freshly generated nonces. $\tilde{n}$ consists of a finite set of restricted names, these names represent freshly generated names which remain secret from the attacker. The set of names occurring in a term t is denoted by $fn(t)$. For any frame $\phi = \nu\tilde{n}.\sigma$, let $fn(\phi)$ be the set of names $fn(\sigma)\backslash\tilde{n}$ where $fn(\sigma) = \bigcup_{t\in Ran(\sigma)} fn(t)$; and for any term t, let $t\phi$ denote by a slight abuse of notation the term $t\sigma$. We say that a term t *satisfies the name restriction (of ϕ)* if $fn(t) \cap \tilde{n} = \emptyset$.

Let us now define the knowledge problems we consider in this paper.

Definition 1 (Deduction)**.** *Let $\phi = \nu\tilde{n}.\sigma$ be a frame, and t a ground term. We say that t is* deduced from ϕ modulo E*, denoted by $\phi \vdash_E t$, if there exists a term ζ such that $\zeta\sigma =_E t$ and $fn(\zeta) \cap \tilde{n} = \emptyset$. The term ζ is called a* recipe *of t in ϕ modulo E.*

Definition 2 (Static Equivalence)**.** *Two terms s and t are* equal *in a frame $\phi = \nu\tilde{n}.\sigma$ modulo an equational theory E, denoted $(s =_E t)\phi$, if $s\sigma =_E t\sigma$, and $\tilde{n} \cap (fn(s) \cup fn(t)) = \emptyset$. The set of all equalities $s = t$ such that $(s =_E t)\phi$ is denoted by $Eq(\phi)$. Given a set of equalities Eq, the fact that $(s =_E t)\phi$ for all $s = t \in Eq$ is denoted by $\phi \models Eq$. Two frames $\phi = \nu\tilde{n}.\sigma$ and $\psi = \nu\tilde{n}.\tau$ are* statically equivalent modulo E*, denoted as $\phi \approx_E \psi$, if $Dom(\sigma) = Dom(\tau)$, $\phi \models Eq(\psi)$ and $\psi \models Eq(\phi)$.*

Alternatively, one can consider the negation of this problem, finding a term pair that shows two frames are not static equivalent. That is, it distinguishes the two frames.

Definition 3 (Frame Distinguishability)**.** *Given frames $\phi = \nu\tilde{n}.\sigma$ and $\psi = \nu\tilde{n}.\tau$, we say that ϕ is distinguishable from ψ in theory E, denoted $\phi \not\approx_E \psi$, if there exists two terms, t and s (with $\tilde{n} \cap (fn(s) \cup fn(t)) = \emptyset$), such that $t\sigma =_E s\sigma$ and $t\tau \neq_E s\tau$.*

Classes of Permutative Theories. We also need to define the (sub)classes of permutative theories we consider in this paper. This is important not only for properly defining the results proven here but also because there are some previous definitions of leaf permutative theories which don't match the definition given here. For example, the definition of leaf permutative given in this paper differs from the one given in [7] (See Definition 6 below). In [7](Definition 6) the permutation is restricted to just the variables and is only applicable to linear terms. Thus, the definition in [7] would perhaps be better named as a linear variable permutative, while the one given here is just a permutation if the leaf-nodes of the term, see Definition 5.

Definition 4 (Permutative Theory)**.** *An equational theory E is* permutative *if for each axiom $l = r$ in E, l and r contain the same symbols with the same number of occurrences.*

One can easily check that $A = \{f(x, f(y, z)) = f(f(x, y), z)\}$ (Associativity), $C = \{f(x, y) = f(y, x)\}$ (Commutativity) and $AC = \{f(x, f(y, z)) = f(f(x, y), z),\ f(x, y) = f(y, x)\}$ (Associativity-Commutativity) are permutative.

Two important subclasses of permutative theories are given by considering the cases where the permutations only occur on leafs or on variables.

Definition 5 (Leaf Permutative Theory)**.** *An equational theory E is* Leaf permutative *if for each axiom $l = r$ in E, r is a leaf permutation of l, i.e., $r = l\sigma$, where σ is a permutation of the leaf nodes of l (variables and constants of l).*

For example, C is leaf permutative, but A is not.

Definition 6 (Variable-Permuting Theory)**.** *An axiom $l = r$ is said to be* variable-permuting *[12] if all the following conditions are satisfied:*

1. *the set of occurrences of l is identical to the set of occurrences of r,*
2. *for any non-variable occurrence p of l, $l(p) = r(p)$,*
3. *for any $x \in Var(l) \cup Var(r)$, the number of occurrences of x in l is identical to the number of occurrences of x in r.*

Both deduction and distinguishability are known to be decidable in subterm convergent rewrite systems [1]. It has already been shown in [8] that deduction is decidable in permutative theories.

Theorem 1 ([8])**.** *Deduction is decidable in any permutative theory.*

This is due to the fact that you can put a bound on the number of terms you need to consider since permutative theories are non-size reducing. However, when considering distinguishability, the problem becomes more difficult as we show in the next section.

3 Distinguishability is Undecidable for Leaf Permutative Theories

In this section we prove that the frame distinguishability problem, and thus static equivalence, is undecidable for leaf permutative theories. The results proceeds as follows, we first state an undecidable result for Linear Bounded Automata (LBA) which we will use in the reduction. LBA are Turing Machines with a tape that is bounded by the size of the input string (plus two tape end-caps) [10, 11]. Next, we show how to create a leaf permutative TRS from an LBA. Finally, we use the TRS and the frame distinguishability problem to solve the undecidable problem for LBA.

Lemma 1. *Given a deterministic LBA, M, with input alphabet Σ, it's undecidable if there exists a string, $w \in \Sigma^*$, such that M accepts w.*

Proof. Easy reduction from the LBA empty language problem proved undecidable in [3]. There it is shown that it is undecidable if for an arbitrary deterministic LBA M, $L(M) = \emptyset$. □

Lemma 2. *Given a deterministic LBA, M, one can construct a leaf permutative TRS R such that if M accepts a string w then there exists a term t which encodes the initial configuration of M on input w and a term s that encodes the final accepting configuration of M on w such that $t{\downarrow_R} = s$.*

Proof. Here we modify the encoding from [14] to obtain a conversion from an LBA to a leaf permutative TRS.

Let $M = (Q, \Sigma, \Gamma, q_0, q_a, q_r, \delta)$. Assume $\Sigma = \{a_1, a_2, \ldots, a_n\}$ and $\Gamma = \{<, >, \sqcup\} \cup \Sigma$, where $<, >$ are the left and right end caps respectively, and $\sqcup$ is the blank symbol.

We now need to construct the terms that will represent the tape of the LBA. We introduce three new non-constant function symbols, f, g, h and three new constants, a, b, and P. We use each as follows:

- h has arity $|Q| + 1$ and is used to represent the state of the LBA. To represent state q_i, a constant b is placed at the i^{th} position with the remaining positions containing a constants. For example, if $|Q| = 2$ then q_0 is represented as $h(b, a, a)$. The final configuration, with constant b in the final position, is used to represent a non-state or "dummy state", which the use of is described below. We denote this dummy state as q_d. **We use the notation $h(q_i)$ to abbreviate the encoding of the state q_i using h.**

- g has arity $|\Gamma|$ and is used to encode the alphabet characters. We place a constant b at position i in g with constants a placed at all other positions to encode a_i. Positions $n+1, \ldots, n+3$ are used to encode $\{<, >, \sqcup\}$ in the same way. **We use the notation $h(a_i)$ to abbreviate the encoding of the character 'a_i' using h.**

- f is used to form terms which consist of an encoding of a state, an encoding of a single character, and an f-rooted subterm or the constant P. The subterm is used to encode the rest of the string. The constant P is used to stop the encoding. For example, the dummy state and the character a_i can be encoded as the term $f(h(q_d), g(a_i), P)$.

We now form terms representing the state of the LBA and its tape as follows:

- For any string $w \in \Gamma^+$ we represent w as a layered f-term, one layer per alphabet character. The last layer is ended using the constant P. The dummy state is used by default for each of the state positions in the f-terms. **We use the notation $f(\overline{w})$ to abbreviate the encoding of the string w using f-terms**.

We now need to construct the leaf permutative TRS. Let's consider the moves of the transition function, δ, and construct from them a TRS R:

- For each right move, $\delta(q_i, a_i) = (q_j, a_j, R)$, we create a rule for each possible character $a_k \in \Gamma$ below, of the form:

$$f(h(q_i), g(a_i), f(h(q_d), g(a_k), x)) \to f(h(q_d), g(a_j), f(h(q_j), g(a_k), x))$$

- For each left move, $\delta(q_i, a_i) = (q_j, a_j, L)$, we create a rule for each possible character $a_k \in \Gamma$ below, of the form:

$$f(h(q_d), g(a_k), f(h(q_i), g(a_i), x)) \to f(h(q_j), g(a_k), f(h(q_d), g(a_j), x))$$

Finally, we need to describe the initial configurations for the LBA. The LBA will start in the configuration $q_o\langle w\rangle$ for input w. We encode this as an f-term, where all the states are initially $h(q_d)$ except the first (leftmost) h. That is, we encode using the term $t = f(h(q_0), g(<), f(\overline{w>}))$.

Notice that each of the rules in R are leaf permutative. In addition, the LBA accepts the string w iff $f(h(q_0), g(<), f(\overline{w>})) \to_R^* s$ such that s is a term that includes just one encoded $h(q_a)$. □

Lemma 3. *Let M be a deterministic LBA. Then the leaf permutative TRS, R, constructed from M is locally confluent. Furthermore, if M is both deterministic and terminating then R is convergent.*

Next, it's easy to show that from an LBA M_1 one we can construct the following LBA.

Lemma 4. *Let M_1 be a deterministic LBA that before accepting and halting it replaces the tape (except the end caps) with blank symbols and stops (enters the accept or reject state) with the tape head over the left end-cap. One can construct an LBA M_2 from M_1 such that $L(M_2) = \emptyset$ and every transition M_1 and M_2 are the same except the accepting transitions from M_1 are now rejecting in M_2.*

Notice that for each LBA M_1 and M_2 there is a corresponding leaf permutative TRS, R_1 and R_2 respectively. Next we can easily combine these two TRSs into a single TRS.

Lemma 5. *Given deterministic LBAs M_1 and M_2 and their leaf permutative TRSs R_1 and R_2 respectively. Let q_{0_i} be the initial state for M_i. One can construct a leaf permutative TRS $R_{1,2}$ such that for input string w and term $t_i = f(h(q_{0_i}), g(<), f(\overline{w>}))$, that $t_i \to_{R_{1,2}}^* s_i$ iff $t_i \to_{R_i}^* s_i$, $i \in \{1, 2\}$.*

Proof. One just needs to ensure that $\{Q_1 \setminus \{q_a, q_r\}\} \cap \{Q_2 \setminus \{q_a, q_r\}\} = \emptyset$. Then, the rules of R_1 and R_2 are disjoint. Starting a configuration with a start state in Q_i will ensure each of the following configurations, except the final with shared states q_a or q_r, are disjoint. □

Theorem 2. *Frame distinguishability is undecidable in general when E is a leaf permutative theory.*

Proof. We can proceed by reduction using Lemma 1. Assume we are given a deterministic LBA M_1. Without loss of generality assume that before halting and entering q_a or q_r that M_1 erases its tape and halts with the head over the left end-cap. Furthermore, assume that as an initial step M_1 scans the tape from left to right end-cap and then back to the left. Since these steps could be added to any LBA without changing its language, they don't represent a restriction. We proceed as follows.

1. From M_1 construct the always rejecting LBA M_2 as in Lemma 4. We can assume w.l.g., that $\{Q_1 \setminus \{q_a, q_r\}\} \cap \{Q_2 \setminus \{q_a, q_r\}\} = \emptyset$. This can be done by simply creating a marked version of $Q_1 \setminus \{q_a, q_r\}$ and using that for the set of states for M_2, i.e., Q_2. Let q_{0_1} be the start state for M_1 and q_{0_2} for M_2.

2. From M_1 and M_2 construct R_1 and R_2 respectively as in Lemma 2.

3. Construct $R_{1,2}$ from R_1 and R_2 as in Lemma 5.

4. Construct two frames, such that $\tilde{n} = \emptyset$ for each frame:

   a) $\phi_1 = \nu\tilde{n}.\sigma_1 = \{x \mapsto q_{0_1}, y \mapsto q_a\}$ b) $\phi_2 = \nu\tilde{n}.\sigma_2 = \{x \mapsto q_{0_2}, y \mapsto q_a\}$

   Assume that we have an algorithm for the frame distinguishability problem and it returns a term pair (t, s). Then, $t\sigma_1 \rightarrow^*_{R_{1,2}} s\sigma_1$ and $t\sigma_2 \not\rightarrow^*_{R_{1,2}} s\sigma_2$. Notice, it can't be that $s = t$ nor can t be ground, otherwise $t\sigma_2 \rightarrow^*_{R_{1,2}} s\sigma_2$. Thus, $t\sigma_1 \rightarrow^+_{R_1} s\sigma_1$. We now need to show the following:

   (a) $t\sigma_i$ is a well formed term encoding an initial configuration of an LBA M_i for some initial input string w.

   - Notice that the rules of $R_{1,2}$ can only be applied to well-formed subterms of t. Since M_1 (and thus M_2) first scan the tape from left to right the rewrite derivation on $t\sigma_1$ would stop when any malformed subterm was reached and before the final configuration. However, since the steps of M_2, except the step entering the final configuration, are the same as those for M_1, M_2 would stop at the same point in the derivation from $t\sigma_2$. That is, $t\sigma_2 \rightarrow^*_{R_{1,2}} s\sigma_2$.
   - We can assume the initial rewrite step in the derivations $t\sigma_i \rightarrow^*_{R_{1,2}} s\sigma_i$, occur at ϵ since any part of the term above the subterm $f(h(x), g(<), f(\overline{w >}))$ will not change and thus can be discarded.

   (b) $s\sigma_1$ is a final configuration of an LBA M_1 and represents an accepting configuration.

   - $s\sigma_1$ must be a final *accepting* configuration, otherwise $t\sigma_2 \rightarrow^*_{R_{1,2}} s\sigma_2$. Recall that all the transitions of M_2, except the final ones entering a final configuration, are the same. Thus, starting from a well-formed initial configuration, the only way to have $t\sigma_1 \rightarrow^*_{R_{1,2}} s\sigma_1$ and $t\sigma_2 \not\rightarrow^*_{R_{1,2}} s\sigma_2$ is for $s\sigma_1$ to be a final accepting configuration.

Therefore, a frame distinguishablity algorithm would allow us to decide if for an arbitrary LBA M_1 if there exist some string, w, accepted by M_1, violating Lemma 1. □

If the frame distinguishability problem is undecidable then we also get undecidability of the static equivalence problem.

Corollary 1. *Static equivalence modulo E is undecidable in general when E is a leaf permutative theory.*

4 Conclusion

In the context of security protocols, it is crucial to identify classes of theories with decidable knowledge problems. When a theory is given by a subterm convergent term rewrite system, these knowledge problems are known to be decidable thanks to the seminal work initiated in [1]. Furthermore, decidability of the knowledge problems for shallow permutative theories has been shown in [9]. Then, it is possible to stay on the decidability side when considering a subterm rewrite system which is convergent modulo a shallow permutative theory [9]. In [8] we constructed a restricted subclass of variable-permuting theories, called *SVP* theories, where static equivalence is decidable. These theories restrict the roots of the left and right-hand sides of the rules such that decidability is achieved.

We plan to continue the study of the knowledge problems in equational theories given by rewrite systems modulo permutative axioms. On the one hand, it is possible to consider extensions of subterm rewrite systems, such as contracting rewrite systems [8,13]. On the other hand, we can now envision to go beyond shallow permutative axioms. Due to the undecidability result reported here, it is clear now that we cannot consider any arbitrary set of permutative axioms.

References

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

[2] Martín Abadi and Cédric Fournet. Mobile values, new names, and secure communication. In Chris Hankin and Dave Schmidt, editors, *Conference Record of POPL 2001: The 28th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, London, UK, January 17-19, 2001*, pages 104–115. ACM, 2001.

[3] Zümrüt Akçam, Kimberly A. Cornell, Daniel S. Hono II, Paliath Narendran, and Andrew Pulver. On problems dual to unification: The string-rewriting case, 2023. Available at `https://doi.org/10.48550/arXiv.2103.00386`.

[4] Mauricio Ayala-Rincón, Maribel Fernández, and Daniele Nantes-Sobrinho. Intruder deduction problem for locally stable theories with normal forms and inverses. *Theoretical Computer Science*, 672:64–100, 2017.

[5] Franz Baader and Tobias Nipkow. *Term rewriting and all that.* Cambridge University Press, New York, NY, USA, 1998.

[6] Franz Baader and Wayne Snyder. Unification theory. In John Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*, pages 445–532. Elsevier and MIT Press, 2001.

[7] Thierry Boy De La Tour, Mnacho Echenim, and Paliath Narendran. Unification and matching modulo leaf-permutative equational presentations. In *International Joint Conference on Automated Reasoning*, pages 332–347. Springer, 2008.

[8] Carter Bunch, Saraid Dwyer Satterfield, Serdar Erbatur, Andrew M. Marshall, and Christophe Ringeissen. Knowledge problems in protocol analysis: Extending the notion of subterm convergent, 2024. Available at https://doi.org/10.48550/arXiv.2401.17226.

[9] Serdar Erbatur, Andrew M. Marshall, and Christophe Ringeissen. Computing knowledge in equational extensions of subterm convergent theories. *Math. Struct. Comput. Sci.*, 30(6):683–709, 2020.

[10] John E Hopcroft and Jeffrey D Ullman. Some results on tape-bounded Turing machines. *Journal of the ACM*, 16(1):168–177, 1969.

[11] Sige-Yuki Kuroda. Classes of languages and linear-bounded automata. *Information and control*, 7(2):207–223, 1964.

[12] Paliath Narendran and Friedrich Otto. Single versus simultaneous equational unification and equational unification for variable-permuting theories. *J. Autom. Reason.*, 19(1):87–115, 1997.

[13] Saraid Dwyer Satterfield, Serdar Erbatur, Andrew M. Marshall, and Christophe Ringeissen. Knowledge problems in security protocols: Going beyond subterm convergent theories. In Marco Gaboardi and Femke van Raamsdonk, editors, *8th International Conference on Formal Structures for Computation and Deduction, FSCD 2023, July 3-6, 2023, Rome, Italy*, volume 260 of *LIPIcs*, pages 30:1–30:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

[14] Manfred Schmidt-Schauß. Unification in permutative equational theories is undecidable. *J. Symb. Comput.*, 8(4):415–421, 1989.