Local XOR Unification: Definitions, Algorithms and Application to Cryptography

Hai Lin[1] and Christopher Lynch[2]

[1] Clarkson University, Potsdam, NY, U.S.A.
hlin@clarkson.edu
[2] Clarkson University, Potsdam, NY, U.S.A.
clynch@clarkson.edu

Abstract

Unification techniques have been proven to be useful for formal analysis of cryptographic systems. In this paper, we introduce a new unification problem called local XOR unification, motivated by formal analysis of security of modes of operation. The goal in local XOR unification is to find a substitution making two terms equivalent modulo the theory of exclusive-or, but each variable is only allowed to be mapped to a term from a given set of terms. We present two versions of the local XOR unification problem, and give algorithms to solve them, proving soundness, completeness and termination.

1 Introduction

In logic, unification is an algorithmic process of solving equations between symbolic expressions. The basic form of unification is syntactic unification, where both sides of each equation must be made exactly the same under some substitution. In equational unification, both sides of each equation can be the same under some substitution, modulo some background equational theory.

Equational unification has a number of successful applications in cryptography protocol verification [1, 2, 3]. This work is motivated by symbolic reasoning of cryptographic modes of operation, and it is closely related to [7]. The idea in [7] is that a block of message can be modelled symbolically as a term. In order to break a mode of operation, an adversary needs to solve some unification problem. The unification problem is unification modulo the theory of exclusive-xor, together with some free function symbols. Another feature of the unification problem is that each variable is only allowed to be mapped to a term from a given set of terms. Intuitively, the set of terms associated with a variable (modelling some plain-text block) include everything that an adversary is able to compute when generating that plain-text block. We call this kind of unification problem *local XOR unification*. We present two concrete versions of the local XOR unification problem. They are called *f-rooted local unification*, and *$\oplus$-rooted local unification*. They can be used to analyze different cryptographic modes of operation. For f-rooted local unification, this paper gives a unification algorithm, which can find a subset of all unifiers, called *minimum unifiers*. For $\oplus$-rooted local unification, this paper gives a unification algorithm, which can find all unifiers. We prove the soundness completeness and termination of both of those algorithms.

The rest of this paper is organized as follows. In Section 2, we discuss some background that will be used in this paper. In Section 3, we introduce two concrete versions of the local XOR unification problem and discuss how they are related to security of modes of operation. In Section 4 and Section 5, we describe how we solve these unification problems. We conclude in Section 6.

2 Preliminaries

A *term* can be built up from constants $(0, r_1, r_2, \ldots)$, variables, Boolean values $\oplus$, $\times$, two unary function symbols f and h. We distinguish between two types of variables: *term variables* and *Boolean variables*. A term variable $(x, x_1, x_2, \ldots)$ can be instantiated by a term. A Boolean variable $(b, b_1, b_2, \ldots)$ can only be instantiated by a Boolean value (0, 1). If a term t does not contain any variable, t is a *ground term*. If t does not contain any Boolean variables or Boolean values, we call it a *pure term*. Otherwise, we call it a *mixed term*. To be more precise, we can build up terms in the following way.

$term := constant \mid variable \mid f(term) \mid h(term) \mid term \oplus term \mid bool \times term$
$variable := term\_variable \mid Boolean\_variable$
$bool := Boolean\_variable \mid 0 \mid 1$

We assume that all terms are simplified by applying the following rules in R modulo AC(Associativity and Commutativity).

$$R = \{t \oplus t \to 0, t \oplus 0 \to t, 1 \times t \to t, 0 \times t \to 0\}$$

We use a mixed term to represent a set of terms. For example, suppose that $mt = b_1 \times r \oplus b_2 \times f(r)$, then mt represents a set of terms $T = \{0, r, f(r), r \oplus f(r)\}$. The idea is that under all possible Boolean values for b_1 and b_2, mt is one of the terms in T after simplification. We will use this idea in Section 5.

A term t is an f-rooted term if t is of the form $f(t')$. Similarly, a $\times$-rooted term is of the form $co \times t$, where co is either a Boolean variable or a Boolean value, and t is a term. We call co the *coefficient* of $co \times t$. An $\oplus$-rooted term t is of the form $t_1 \oplus t_2 \oplus \ldots \oplus t_n$, where $t_1, \ldots, t_n$ are called *summands* of t. Let $T = \{t_1, t_2, \ldots, t_n\}$ be a set of terms, we use $\oplus T$ to denote $t_1 \oplus t_2 \oplus \ldots \oplus t_n$.

A *substitution* is a set of bindings for variables. If a substitution σ is of the form $\{x_1 \mapsto t_1, x_2 \mapsto t_2, \ldots, x_n \mapsto t_n\}$. We can write it as $\sigma = \{x_1 \mapsto t_1\} \cup \Gamma$, where $\Gamma = \{x_2 \mapsto t_2, \ldots, x_n \mapsto t_n\}$. A *term substitution* maps term variables to terms. A *Boolean substitution* maps Boolean variables to Boolean values. We use $domain(\sigma)$ to denote the domain of a substitution σ. In this paper, we do not consider any substitutions σ, where $domain(\sigma)$ contains both term variables and Boolean variables. For simplicity, from now on, variables mean term variables, and substitutions mean term substitutions.

We can apply a substitution σ to a term t in the straightforward way. If $x \in domain(\sigma)$, then replace each occurrence of x in t by $x\sigma$. Similarly, if we apply a Boolean substitution τ to a term t, we replace each occurrence of a Boolean variable b in t by $b\tau$. Let $T = \{t_1, t_2, \ldots, t_n\}$ be a set of terms, σ be a substitution, then $T\sigma = \{t_1\sigma, t_2\sigma, \ldots, t_n\sigma\}$. Similarly, $T\tau = \{t_1\tau, t_2\tau, \ldots, t_n\tau\}$, where τ is a Boolean substitution.

We use $\sigma_1\sigma_2$ to denote the composition of two substitutions σ_1 and σ_2. We call $\sigma_1\sigma_2$ an *instance* of σ_1. For any term t, $t(\sigma_1\sigma_2) = (t\sigma_1)\sigma_2$. We use $\sigma \circ \tau$ to denote the composition of a substitution $\sigma = \{x_1 \mapsto t_1, x_2 \mapsto t_2, \ldots, x_n \mapsto t_n\}$ and a Boolean substitution τ. $\sigma \circ \tau = \{x_1 \mapsto t_1\tau, x_2 \mapsto t_2\tau, \ldots, x_n \mapsto t_n\tau\}$. So for any term t, $t(\sigma \circ \tau) = (t\sigma)\tau$. We use $\tau_1 \uplus \tau_2$ to denote the *consistent union* of two Boolean substitutions τ_1 and τ_2. This operation is undefined if there exists a Boolean variable b s.t. $b \in domain(\tau_1) \cap domain(\tau_2)$ and $b\tau_1 \neq b\tau_2$. Otherwise,

$$b(\tau_1 \uplus \tau_2) = \begin{cases} b\tau_1 & x \in domain(\tau_1) \\ b\tau_2 & \text{otherwise} \end{cases}$$

Note that if $domain(\tau_1) \cap domain(\tau_2) = \emptyset$, then $\tau_1 \uplus \tau_2$ is exactly $\tau_1 \cup \tau_2$.

We use $t_1 =_\oplus t_2$ to denote that t_1 and t_2 are the same after simplification using the rules in R. Let eq be an equation of the form $t_1 \overset{?}{=} t_2$, where t_1 and t_2 are pure terms. σ is a unifier of eq if $t_1\sigma =_\oplus t_2\sigma$. Let eqs be a set of equations on pure terms, σ is a unifier of eqs if σ is a unifier of all the equations in eqs. The following definition defines a complete set of unifiers of a set of equations. [6] shows how complete set of unifiers can be computed.

Definition 2.1. We use $CSU_\oplus(eqs)$ to denote the *complete set of unifiers* of eqs. $CSU_\oplus(eqs)$ is a set of substitutions s.t.

- If $\sigma \in CSU_\oplus(eqs)$, then σ is a unifier of eqs.
- If τ is a unifier of eqs if there exist $\sigma \in CSU_\oplus(eqs)$ s.t. $\tau = \sigma\sigma'$ for some σ'.

A *position* p is a sequence of positive integers. $len(p)$ denotes the number of positive integers in p, λ denotes the empty sequence. Let p_1 and p_2 be two positions, p_1 is a *prefix* of p_2 if p_2 is of the form $p_1.i_1.i_2...i_n$, where i_j is a positive integer $(1 \leq j \leq n)$. p_1 is an *immediate prefix* of p_2 if p_2 is of the form $p_1.i$, where i is a positive integer. Let t be a term, and p be a position, we use $t|_p$ to denote the subterm of t at position p. For any term t, $t|_\lambda = t$. If g is a function symbol of arity n, then $g(t_1, t_2, \ldots, t_n)|_{i,i_1,i_2,\ldots,i_m} = t_i|_{i_1,i_2,\ldots,i_m}$. We use $t[s]|_p$ to denote the term obtained by replacing in t the subterm at position p by term s. For example, $r_1 \oplus f(r_2)|_{2.1} = r_2$, $r_1 \oplus f(r_2)[x_1]|_{2.1} = r_1 \oplus f(x_1)$.

If t is a term, p is a position, σ is a substitution, then $(t|_p)\sigma = (t\sigma)|_p$.

3 Local XOR Unification

The goal in local XOR unification is to find a substitution making two terms equivalent modulo the theory of exclusive-or, but each variable is only allowed to be mapped to a term from a given set of terms. In Section 3.1 and 3.2, we introduce two concrete versions of local XOR unification. In Section 3.3, we discuss how they are related to cryptography.

3.1 f-rooted Local Unification

Definition 3.1 defines f-mappings, which map variables to sets of terms. Definition 3.2 defines a relation $\prec_{\mathscr{M}_f}$. Definition 3.3 defines a concrete version of the local XOR unification called f-rooted local unification, since the goal is to unify two f-rooted terms.

Definition 3.1. Let $\mathscr{M}$ be a mapping from variables to sets of pure terms. $\mathscr{M}$ is an *f-mapping* if

- For each variable $x \in domain(\mathscr{M})$, $0 \in \mathscr{M}(x)$.
- For each variable x, if $t \in \mathscr{M}(x)$, then t is either a constant or an f-rooted term.

Definition 3.2. Let $\mathscr{M}$ be an f-mapping.

- If $t \in \mathscr{M}(x)$, then $t \prec_{\mathscr{M}_f} x$.
- If $t \prec_{\mathscr{M}_f} x$, then $h(t) \prec_{\mathscr{M}_f} x$.
- If $t_1 \prec_{\mathscr{M}_f} x$, $t_2 \prec_{\mathscr{M}_f} x$, $\ldots$, $t_n \prec_{\mathscr{M}_f} x$, then $t_1 \oplus t_2 \oplus \ldots \oplus t_n \prec_{\mathscr{M}_f} x$.

$t \not\prec_{\mathscr{M}_f} x$ if $t \prec_{\mathscr{M}_f} x$ does not hold.

σ is an admissible $\mathscr{M}_f$-substitution if, for each variable $x \in \mathit{domain}(\sigma)$, $x\sigma =_{\oplus} t\sigma$, for some $t \prec_{\mathscr{M}_f} x$.

Definition 3.3. Let $\mathscr{M}$ be an f-mapping. t_1 and t_2 are two f-rooted pure terms. t_1 and t_2 are *$\mathscr{M}_f$-unifiable* if and only if there exists some substitution σ such that

- σ is an admissible $\mathscr{M}_f$-substitution.
- $t_1\sigma =_{\oplus} t_2\sigma$.

σ is called a *$\mathscr{M}_f$-unifier* of t_1 and t_2. The problem of checking if t_1 and t_2 are $\mathscr{M}_f$-unifiable is called *f-rooted local unification.*

Appendix A provides a function $\mathscr{M}_f$-ADMISSIBLE for checking if some substitution is an admissible $\mathscr{M}_f$-substitution.

3.2 $\oplus$-rooted Local Unification

Definition 3.4 defines $\oplus$-mappings, which map variables to sets of terms. Definition 3.5 defines a relation $\prec_{\mathscr{M}_{\oplus}}$. Definition 3.6 defines a concrete version of the local XOR unification called $\oplus$-rooted local unification, since the goal is to unify some $\oplus$-rooted term with 0.

Definition 3.4. Let $\mathscr{M}$ be a mapping from variables to sets of pure terms. $\mathscr{M}$ is an *$\oplus$-mapping* if

- For each variable x_i, $0 \in \mathscr{M}(x_i)$.
- For each variable x_i, if $t \in \mathscr{M}(x_i)$, then h does not occur in t, and t does not contain any variable x_j, where $j \geq i$.
- If $i < j$, then $\mathscr{M}(x_i) \subseteq \mathscr{M}(x_j)$.

Definition 3.5. Let $\mathscr{M}$ be an $\oplus$-mapping.

- If $t \in \mathscr{M}(x)$, then $t \prec_{\mathscr{M}_{\oplus}} x$.
- If $t_1 \prec_{\mathscr{M}_{\oplus}} x$, $t_2 \prec_{\mathscr{M}_{\oplus}} x$, ..., $t_n \prec_{\mathscr{M}_{\oplus}} x$, then $t_1 \oplus t_2 \oplus \ldots \oplus t_n \prec_{\mathscr{M}_{\oplus}} x$.

$t \not\prec_{\mathscr{M}_{\oplus}} x$ if $t \prec_{\mathscr{M}_{\oplus}} x$ does not hold.

σ is an admissible $\mathscr{M}_{\oplus}$-substitution if, for each variable x in the domain of σ, $x\sigma =_{\oplus} t\sigma$, for some $t \prec_{\mathscr{M}_{\oplus}} x$.

Definition 3.6. Let $\mathscr{M}$ be an $\oplus$-mapping. $T = \{t_1, t_2, \ldots, t_n\}$, where h does not occur in T. A subset T' of T is *$\mathscr{M}_{\oplus}$-unifiable with 0* if and only if there exists some substitution σ such that

- σ is an admissible $\mathscr{M}_{\oplus}$-substitution.
- $T'\sigma = 0$.

σ is called an *$\mathscr{M}_{\oplus}$-unifier* of T'. The problem of checking if some subset T' of T is $\mathscr{M}_{\oplus}$-unifiable with 0 is called *$\oplus$-rooted local unification.*

3.3 Application to Cryptography

Both unification problems are related to security of modes of operation. The idea in [7] is that a block of message can be modelled symbolically as a term. Variables model plain-text blocks. Non-zero constants model a block of random bits. "0" models a block of message consisting of all 0's. f models a block cipher, h models a publicly computable cryptographic hash function.

We consider a setting where an adversary interacts with an encryption oracle. The adversary sends plain-text blocks to the encryption oracle, and the encryption oracle responds by encrypting plain-text blocks using a mode of operation. A mode of operation is IND\$-CPA secure if an adversary has negligible advantage to distinguish between a sequence of blocks returned by the mode of operation and a sequence of randomly generated blocks. We consider multiple schedules for modes of operation. In a block-wise schedule (as discussed in [4]), an encrypted block is sent to the adversary immediate after it is generated by the encryption oracle. In a message-wise schedule, all the encrypted blocks are sent to the adversary after the entire message is encrypted.

In [7], a *symbolic history* refers to the messages exchanged during a process in which an adversary interacts with the encryption oracle to encrypt a message. In order to model the symbolic history of a mode of operation, we use a sequence of MOO-tuples (Mode-of-Operation tuples) of the form (S_P, t, S_C), where S_P is a set of plain-text blocks needed to encrypt the current plain-text block, t is a term that represents the encryption of the current block, S_C is a set of cipher-text blocks that are sent to the adversary after encrypting the current plain-text block. Here are two examples:

Example 3.1. Block-wise Cipher Block Chaining for 3 cipher-text blocks:
$(\emptyset, r, \{r\})$,
$(\{x_1\}, f(r \oplus x_1), \{f(r \oplus x_1)\})$,
$(\{x_2\}, f(f(r \oplus x_1) \oplus x_2), \{f(f(r \oplus x_1) \oplus x_2)\})$

Let $\mathscr{M}(x)$ be the set of terms containing 0 and all cipher-text blocks that are sent to an adversary before the adversary generates x. Intuitively $t \in \mathscr{M}(x)$ means that t is immediately available to the adversary. $t \prec_{\mathscr{M}_f} x$ means that the adversary can compute t when generating x, using terms in $\mathscr{M}(x)$ and some functions that the adversary can compute (e.g. $\oplus$ and h). So in Example 3.1, $\mathscr{M}(x_1) = \{0, r\}$, $\mathscr{M}(x_2) = \{0, r, f(r \oplus x_1)\}$. Then $f(f(r \oplus x_1) \oplus x_2)$ and $f(r \oplus x_1)$ are $\mathscr{M}_f$-unifiable under the substitution $\{x_1 \mapsto r, x_2 \mapsto f(0)\}$. Thus, block-wise Cipher Block Chaining is not IND\$-CPA secure. The idea is that if the adversary can generate some cipher-text blocks s.t. two of them are the same, then the adversary can distinguish between a sequence of cipher-text blocks returned by the encryption oracle and a sequence of randomly generated blocks.

In general, let M be a mode of operation, in which the encryption oracle only returns constants and f-rooted terms (e.g. Cipher Block Chaining, Propagating Cipher Block Chaining, Accumulated Block Cipher [5]). In [7], the author shows that M is IND\$-CPA secure if and only if no symbolic history of M contains two terms returned by the encryption oracle that are $\mathscr{M}_f$-unifiable.

Example 3.2. Message-wise Output Feedback Mode for 3 cipher-text blocks:
$(\emptyset, r, \emptyset)$,
$(\{x_1\}, f(r) \oplus x_1, \emptyset)$,
$(\{x_2\}, f(f(r)) \oplus x_2, \{r, f(r) \oplus x_1, f(f(r)) \oplus x_2\})$

Again, let $\mathscr{M}(x)$ be the set of terms containing 0 and all cipher-text blocks that are sent to an adversary before the adversary generates x. So in Example 3.2, $\mathscr{M}(x_1) = \{0\}, \mathscr{M}(x_2) = \{0\}$.

No subset of $\{r, f(r) \oplus x_1, f(f(r)) \oplus x_2\}$ is $\mathscr{M}_\oplus$-unifiable with 0.

In general, let M be a mode of operation, where no cryptographic hash function is used (e.g. Cipher Feedback Mode, Output Feedback Mode). In [7], the author shows that M is IND\$-CPA secure if and only if no symbolic history of M contains any terms returned by the encryption oracle that are $\mathscr{M}_\oplus$-unifiable with 0.

4 Solving f-rooted Local Unification

In this section, we present an algorithm for solving f-rooted local unification. Given an f-mapping $\mathscr{M}$ and two pure f-rooted terms t_1 and t_2, the goal is to check if t_1 and t_2 are $\mathscr{M}_f$-unifiable. By Definition 3.3 and 2.1, any $\mathscr{M}_f$-unifier of t_1 and t_2 must be an instance of some $\tau \in CSU_\oplus(t_1 \stackrel{?}{=} t_2)$. Therefore, we nondeterministically start from a substitution $\tau \in CSU_\oplus(t_1 \stackrel{?}{=} t_2)$. We keep composing τ with some substitution σ until we get a $\mathscr{M}_f$-unifier of t_1 and t_2. In this section, we only consider pure terms.

First we introduce the following notion of an *open position*.

Definition 4.1. An *open position* is of the form (t, p) where t is a term, p is a position, and for all prefix p' of p, $t|_{p'}$ is not an f-rooted term. (t, p) is an *innermost open position* if (t, p) is an open position, and $t|_p$ is a constant, a variable or an f-rooted term.

An innermost open position (t, p) is *extended* from an open position (t, p') if p' is a prefix of p.

Example 4.1. Let $t = f(r_1) \oplus h(r_2)$.

(1) $(t, 1.1)$ is not an open position, since $t|_1$ is an f-rooted term.
(2) $(t, 2)$ is an open position.
(3) $(t, 2.1)$ is an innermost open position.
(4) $(t, 2.1)$ is extended from $(t, 2)$.

Definition 4.2. Let $\mathscr{M}$ be an f-mapping, τ be a substitution. $(x\tau, p)$ is a *bad open position* in τ if for all $t' \prec_{\mathscr{M}_f} x$, $x\tau|_p \neq_\oplus t'\tau$.

$(x\tau, p)$ is a *bad innermost open position* in τ if $(x\tau, p)$ is both an innermost open position and a bad open position in τ.

The following Algorithm 1 can be used to check if $(x\tau, p)$ is a bad open position in τ. The function COMPUTABLE is from Algorithm 7 in Appendix A.

Algorithm 1 Check if $(x\tau, p)$ is a bad open position in τ

```
1: function BAD-OPEN-POSITION($x, \tau, p, \mathscr{M}$)
2:     return not COMPUTABLE-BY-ADVERSARY($x\tau|_p, \tau, \mathscr{M}, x$)
3: end function
```

The following lemma states that if θ is a $\mathscr{M}_f$-unifier, then there is no bad open position in θ.

Lemma 4.1. *Let $\mathscr{M}$ be an f-mapping, and let t_1 and t_2 be two f-rooted terms. If θ is a $\mathscr{M}_f$-unifier of t_1 and t_2, then for any open position $(x\theta, p)$, $x\theta\,|_p =_\oplus t\theta$, for some $t \prec_{\mathscr{M}_f} x$.*

Proof. Induction on the length of p. If $p = \lambda$, $x\theta|_p =_\oplus x\theta$, this is trivial by Definition 3.2. Suppose that $p' = i_1.i_2...i_{n-1}$, and $x\theta\,|_{p'} =_\oplus t\theta$, for some $t \prec_{\mathscr{M}_f} x$. We consider two cases:

(1) Suppose that $t = h(t')$, where $t' \prec_{\mathscr{M}_f} x$. Let p be $i_1.i_2...i_{n-1}, 1$. Then $x\theta\mid_p =_{\oplus} t'\theta$.

(2) Suppose that $t = t_1 \oplus t_2 \oplus \dots t_m$, where where $t_i \prec_{\mathscr{M}_f} x$ $(1 \leq i \leq m)$. Let p be $i_1.i_2...i_{n-1}, i$ $(1 \leq i \leq m)$, $x\theta\mid_p = t_i\theta$. □

We use Example 4.2 as a running example to explain the ideas behind the inference rules.

Example 4.2. Suppose that we have the following mapping $\mathscr{M}$. We want to check if $f(x_2)$ and $f(h(x_3) \oplus f(0) \oplus f(x_4) \oplus f(f(r_1 \oplus r_2)))$ are $\mathscr{M}_f$-unifiable.

$\mathscr{M}(x_1) = \{0, r_1\}$
$\mathscr{M}(x_2) = \{0, r_1, f(x_1 \oplus r_1)\}$
$\mathscr{M}(x_3) = \{0, r_1, f(x_1 \oplus r_1), f(x_2)\}$
$\mathscr{M}(x_4) = \{0, r_1, f(x_1 \oplus r_1), f(x_2), f(r_1 \oplus r_2)\}$

$CSU_{\oplus}(\{f(x_2) \overset{?}{=} f(h(x_3) \oplus f(0) \oplus f(x_4) \oplus f(f(r_1 \oplus r_2)))\}) = \{\tau\}$, where $\tau = \{x_2 \mapsto h(x_3) \oplus f(0) \oplus f(x_4) \oplus f(f(r_1 \oplus r_2))\}$.

We have the following bad open positions in τ.
$(x_2\tau, \lambda)$: $x_2\tau|_\lambda = h(x_3) \oplus f(0) \oplus f(x_4) \oplus f(f(r_1 \oplus r_2))$
$(x_2\tau, 1)$: $x_2\tau|_1 = h(x_3)$
$(x_2\tau, 1.1)$: $x_2\tau|_{1.1} = x_3$
$(x_2\tau, 2)$: $x_2\tau|_2 = f(0)$
$(x_2\tau, 3)$: $x_2\tau|_3 = f(x_4)$
$(x_2\tau, 4)$: $x_2\tau|_4 = f(f(r_1 \oplus r_2))$

By Lemma 4.1, if τ is a $\mathscr{M}_f$-unifier. Then there must not be any bad open position in τ. The inference rules in $\mathscr{R}_{fix}$ allow us to somehow deal with the bad open positions.

$\mathscr{R}_{fix}$

$$\frac{\{x \mapsto t\} \cup \tau}{(\{x \mapsto t\} \cup \tau)\sigma}\ \mathit{Prev}$$

, where (1) $x \in domain(\mathscr{M})$. (2) $\sigma \in CSU_{\oplus}(t|_p \overset{?}{=} t'\tau)$, where $t' \in \mathscr{M}(x)$, and (t, p) is a bad innermost open position in $\{x \mapsto t\} \cup \tau$.
We use $Prev(\tau, \sigma)$ to indicate the result of applying the $Prev$ rule, where σ is used to instantiate τ. So $Prev(\tau, \sigma) = \tau\sigma$.

$$\frac{\{x \mapsto t\} \cup \tau}{(\{x \mapsto t\} \cup \tau)\sigma}\ \mathit{Cancel}$$

, where (1) $x \in domain(\mathscr{M})$. (2) $\sigma \in CSU_{\oplus}(t|_{p_1} \oplus t|_{p_2} \oplus \dots \oplus t|_{p_n} \overset{?}{=} 0)$, where $\forall i \in \{1, 2, \dots, n\}$, (t, p_i) are open positions that share a common immediate prefix p, and $t|_p$ is an $\oplus$-rooted term. (3) $\exists i \in \{1, 2, \dots, n\}$ s.t. (t, p_i) is a bad open position in $\{x \mapsto t\} \cup \tau$.

Note that we need condition (1) in both inference rules in $\mathscr{R}_{fix}$, since procedures for solving unification modulo exclusive-xor may generate new variables. We do not apply the $Prev$ rule or

Cancel rule for new variables, generated by procedures for solving unification modulo exclusive-xor. We use $Cancel(\tau, \sigma)$ to indicate the result of applying the *Cancel* rule, where σ is used to instantiate τ. So $Cancel(\tau, \sigma) = \tau\sigma$.

The function f-ROOTED-UNIFY$(\mathscr{M}, t_1, t_2)$ in Algorithm 2 checks if two terms t_1 and t_2 are $\mathscr{M}_f$-unifiable. We prove properties (soundness, completeness, termination) in Appendix B. Example 4.3 shows the details of how we use f-ROOTED-UNIFY to solve the unification problem in Example 4.2. We use τ_i to denote τ at the i-th iteration of the loop in function FIX(τ).

Example 4.3. Continuing Example 4.2, $\tau = \{x_2 \mapsto h(x_3) \oplus f(0) \oplus f(x_4) \oplus f(f(r_1 \oplus r_2))\}$. FIX$(\tau)$ is called.

$\tau_1 = \{x_2 \mapsto h(x_3) \oplus f(0) \oplus f(x_4) \oplus f(f(r_1 \oplus r_2))$

$\tau_2 = Prev(\tau_1, \sigma_1) = \{x_2 \mapsto h(x_3) \oplus f(0) \oplus f(x_4) \oplus f(f(r_1 \oplus r_2)), x_1 \mapsto r_1\}$, where $\sigma_1 = \{x_1 \mapsto r_1\}$ is a unifier of $\{f(0) \stackrel{?}{=} f(x_1 \oplus r_1)\}$. That is, σ_1 is a unifier of $x_2\tau_1|_2$ and $f(x_1 \oplus r_1)$, where $f(x_1 \oplus r_1) \in \mathscr{M}(x_2)$.

$\tau_3 = Cancel(\tau_2, \sigma_2) = \{x_2 \mapsto h(x_3) \oplus f(0), x_1 \mapsto r_1, x_4 \mapsto f(r_1 \oplus r_2)\}$, where $\sigma_2 = \{x_4 \mapsto f(r_1 \oplus r_2)\}$ is a unifier of $\{f(x_4) \stackrel{?}{=} f(f(r_1 \oplus r_2))\}$ That is, σ_2 is a unifier of $x_2\tau_2|_3$ and $x_2\tau_2|_4$.

$\tau_4 = Prev(\tau_3, \sigma_3) = \{x_2 \mapsto h(r_1) \oplus f(0), x_1 \mapsto r_1, x_4 \mapsto f(r_1 \oplus r_2), x_3 \mapsto r_1\}$, where $\sigma_3 = \{x_3 \mapsto r_1\}$ is a unifier of $\{x_3 \stackrel{?}{=} r_1\}$ That is, σ_3 is a unifier of $x_2\tau_3|_{1.1}$ and r_1, where $r_1 \in \mathscr{M}(x_2)$.

τ_4 is a $\mathscr{M}_f$-substitution. f-ROOTED-UNIFY returns τ_4.

Algorithm 2 Checking if t_1 and t_2 are $\mathscr{M}_f$-unifiable

```
function f-ROOTED-UNIFY($\mathscr{M}, t_1, t_2$)
        // (1) $\mathscr{M}$ is an $f$-mapping. (2) $t_1$ and $t_2$ are two $f$-rooted terms.

    Nondeterministically choose a unifier $\tau$ from $CSU_\oplus(t_1 \stackrel{?}{=} t_2)$.
    $result \leftarrow FIX(\tau)$
    if $result = None$ then
        return $false$            //$t_1$ and $t_2$ are not $\mathscr{M}_f$-unifiable".
    else
        return $result$            //$result$ is a $\mathscr{M}_f$-unifier of $t_1$ and $t_2$.
    end if
end function

function FIX($\tau$)
    while some rule is applicable to $\tau$ do
        $\tau \leftarrow Prev(\tau, \sigma)$ or $Cancel(\tau, \sigma)$.            //Nodeterministically apply a rule.
    end while
    if $\mathscr{M}_f$-ADMISSIBLE($\mathscr{M}, \tau$) = $true$ then
        return $\tau$.
    else
        return $None$.
    end if
end function
```

5 Solving $\oplus$-rooted Local Unification

In this section, we present an algorithm for solving the $\oplus$-rooted local unification problem, which we introduced in Section 3.2. Given some $\oplus$-mapping $\mathscr{M}$ and a set of pure terms $\mathscr{C} = \{t_1, t_2, \ldots, t_n\}$. The goal is to check if some subset $\mathscr{C}'$ of $\mathscr{C}$ is *$\mathscr{M}_\oplus$-unifiable* with 0. We will use the following Example 5.1 as a running example:

Example 5.1. $\mathscr{M}(x_1) = \{r\}$.

$\mathscr{M}(x_2) = \{r, f(r \oplus x_1) \oplus x_1\}$.

$\mathscr{C} = \{r, f(r \oplus x_1) \oplus x_1, f(f(r \oplus x_1) \oplus x_2)\}$.

$\mathscr{C}$ is $\mathscr{M}_\oplus$-unifiable with 0, under $\{x_1 \mapsto r, x_2 \mapsto f(0)\}$. We will show how we solve this unification problem. In Section 5.1, we present a helper function REMOVE-TERM-VARIABLES($\mathscr{M}, \mathscr{C}$), which computes a special term substitution $\gamma^{\mathscr{M}}$. We use $\gamma^{\mathscr{M}}$ to instantiate terms in $\mathscr{C}$, thereby remove term variables in $\mathscr{C}$. So $\mathscr{C}_{mix} = \mathscr{C}\gamma^{\mathscr{M}}$. In Section 5.2, we present another helper function ANNOTATE-AND-SATURATE, which tries to figure out if the XOR of some subset of $\mathscr{C}_{mix}$ is 0 under some Boolean substitution τ. In Section 5.3, we put things together and present a function XOR-ROOTED-UNIFY, which solves the $\oplus$-rooted local unification problem.

In this section, variables mean term variables, substitutions mean term substitutions. We use $t, t_1, t_2, \ldots$ to denote pure terms, and use $mt, mt_1, mt_2, \ldots$ to denote mixed terms. $\sigma, \sigma_1, \sigma_2, \ldots$ denote term substitutions, $\tau, \tau_1, \tau_2, \ldots$ denote Boolean substitutions.

5.1 Handling Variables

In order to illustrate the idea, let us consider Example 5.1. According to Definition 3.5, x_1 can only be instantiated by 0 or r. We denote this as $x_1 \mapsto b_1 \times r$, where b_1 is a Boolean variable. Similarly, $x_2 \mapsto b_2 \times r \oplus b_3 \times (f(r \oplus x_1) \oplus x_1)$, or equivalently $x_2 \mapsto b_2 \times r \oplus b_4 \times f(r \oplus x_1)$. (Under all possible values for b_2, b_3, b_4, x_2 maps to the same set of terms.) We then instantiate x_1 and x_2 in $\mathscr{C}$ and get a set of mixed terms $\mathscr{C}_{mix} = \{r, f(r \oplus (b_1 \times r)) \oplus (b_1 \times r), f(f(r \oplus (b_1 \times r)) \oplus b_2 \times r \oplus b_4 \times f(r \oplus (b_1 \times r)))\}$. Notice that there are no variables in $\mathscr{C}_{mix}$.

To formalize the above idea. We have the following definitions. Suppose that $t = t_1 \oplus t_2 \oplus \ldots \oplus t_m \oplus x_1 \oplus x_2 \oplus \ldots x_n$, where $t_i (1 \leq i \leq m)$ is not a variable. We define $NonVar(t) = t_1 \oplus t_2 \oplus \ldots \oplus t_m$, and $Var(t) = x_1 \oplus x_2 \oplus \ldots \oplus x_n$. Let $T = \{t_1, t_2, \ldots t_n\}$ be a set of mixed terms, we define $NonVar(T) = \{NonVar(t_1), NonVar(t_2), \ldots, NonVar(t_n)\}$. Let $\vec{B} = \langle b_1, b_2, \ldots, b_n \rangle$ be a vector of Boolean variables, $T \bullet \vec{B}$ is defined to be $b_1 \times t_1 \oplus b_2 \times t_2 \oplus \ldots b_n \times t_n$. $T \bullet \vec{B}$ is called a *linear combination* of $t_1, t_2, \ldots, t_n$.

Given an $\oplus$-mapping $\mathscr{M}$ and a set of terms $\mathscr{C}$, our initial goal is to find a subset of terms in $\mathscr{C}$ s.t. their XOR is 0 under some admissible $\mathscr{M}_\oplus$-substitution. REMOVE-TERM-VARIABLES($\mathscr{M}, \mathscr{C}$) in Algorithm 3 returns $(\gamma^{\mathscr{M}}, \mathscr{C}_{mix})$, where $\mathscr{C}_{mix} = \mathscr{C}\gamma^{\mathscr{M}}$. Our new goal is to find a subset of terms in $\mathscr{C}_{mix}$ s.t. their XOR is 0 under some Boolean substitution. In Appendix C, we show that for any Boolean substitution τ, $\gamma^{\mathscr{M}} \circ \tau$ is always an admissible $\mathscr{M}_\oplus$-substitution.

Algorithm 3 remove term variables

```
function REMOVE-TERM-VARIABLES($\mathscr{M}, \mathscr{C}$)
    // Assume (1) $domain(\mathscr{M}) = \{x_1, x_2, \ldots, x_m\}$. (2) $\mathscr{C} = \{t_1, t_2, \ldots, t_n\}$.

    $\gamma^{\mathscr{M}} \leftarrow \{x_1 \mapsto t'_1, x_2 \mapsto t'_2, \ldots, x_m \mapsto t'_m\}$, where $t'_i = NonVar(\mathscr{M}(x_i)) \bullet \vec{B}_i$, and $\vec{B}_i$ is a vector of fresh Boolean variables.

    $\mathscr{C}_{mix} \leftarrow \mathscr{C}\gamma^{\mathscr{M}}$.

    return $(\gamma^{\mathscr{M}}, \mathscr{C}_{mix})$
end function

```

5.2 Applying Inference Rules

Consider Example 5.1. REMOVE-TERM-VARIABLES($\mathscr{M}, \mathscr{C}$) returns $(\gamma^{\mathscr{M}}, \mathscr{C}_{mix})$, where

$\gamma^{\mathscr{M}} = \{x_1 \mapsto b_1 \times r, x_2 \mapsto b_2 \times r \oplus b_4 \times f(r \oplus x_1)\}$.

$\mathscr{C}_{mix} = \{r, f(r \oplus (b_1 \times r)) \oplus (b_1 \times r), f(f(r \oplus (b_1 \times r)) \oplus b_2 \times r \oplus b_4 \times f(r \oplus (b_1 \times r)))\}$.

It turns out that the XOR of all three terms in $\mathscr{C}_{mix}$ is 0 under $\{b_1 \mapsto 1, b_2 \mapsto 0, b_4 \mapsto 1\}$. We use two procedures to figure that out.

In Section 5.2.1, we present a procedure for checking if two mixed terms are unifiable under some Boolean substitution. For example, we can unify r with $b_1 \times r$ under $\{b_1 \mapsto 1\}$. $f(r \oplus (b_1 \times r))$ and $f(f(r \oplus (b_1 \times r)) \oplus b_2 \times r \oplus b_4 \times f(r \oplus (b_1 \times r)))$ are also unifiable under $\{b_1 \mapsto 1, b_2 \mapsto 0, b_4 \mapsto 1\}$.

In Section 5.2.2, we present another procedure that can take the XOR of two mixed terms in $\mathscr{C}_{mix}$, calls the first procedure to unify two summands occurring in those mixed terms, thereby cancel the two summands. For example, the procedure can take $t_1 = f(r \oplus (b_1 \times r)) \oplus (b_1 \times r)$ and $t_2 = f(f(r \oplus (b_1 \times r)) \oplus b_2 \times r \oplus b_4 \times f(r \oplus (b_1 \times r)))$, unify $f(r \oplus (b_1 \times r))$ and t_2 under $\tau = \{b_1 \mapsto 1, b_2 \mapsto 0, b_4 \mapsto 1\}$. Therefore, $(t_1 \oplus t_2)\tau = r$.

5.2.1 Unifying Two Mixed Terms

Definition 5.1. Let *eq* be an equation $t_1 \overset{?}{=} t_2$ on mixed terms without term variables. *eq* is *Boolean unifiable* under some substitution τ if $t_1\tau =_{\oplus} t_2\tau$. We call τ a *Boolean unifier* of *eq*. Let *eqs* be a set of equations on mixed terms without term variables. *eqs* is *Boolean unifiable* under τ if each equation in *eqs* is Boolean unifiable under τ. We call τ a *Boolean unifier* of *eqs*.

The function BOOLEAN-UNIFY(t_1, t_2) from Algorithm 4 checks if t_1 and t_2 are Boolean unifiable by nondeterministically applying the inference rules in $\mathscr{R}_{bool-unif}$ to *states*. A *state* *st* is of the form $eqs|\tau$, where *eqs* is a set of equations on mixed terms and τ is a Boolean substitution. To check if two linear mixed terms t_1 and t_2 are Boolean unifiable, we start from the initial state: $\{t_1 \oplus t_2 \overset{?}{=} 0\}|\epsilon$, and we keep applying the inference rules in $\mathscr{R}_{bool-unif}$ nondeterministically. Eventually we reach either a final state or a stuck state. A final state is a state of the form $\emptyset|\tau$. A stuck state is a state of the form $eqs|\tau$, where $eqs \neq \emptyset$, and no rule applies. We call $st_1, st_2, \ldots$ a trace of t_1 and t_2 if st_1 is $\{t_1 \oplus t_2 \overset{?}{=} 0\}|\epsilon$, and $\forall i > 1$, st_i is the result of applying one of the inference rules to st_{i-1}. In Appendix D, we prove properties about BOOLEAN-UNIFY (Soundness, Completeness and Termination).

In order to cut down the search space, we define the notion of $dep_f(t)$, where t is a mixed term without any term variables. If t is a ground term, $dep_f(t)$ is the maximum number of nested f symbols in t. Or equivalently, $dep_f(t)$ is the maximum number of f symbols among all root-to-leaf paths of the syntax tree of t. If t contains Boolean variables, we define $dep_f(t)$ as $\{dep_f(t\tau) \mid \tau \text{ is a Boolean substitution}, t\tau \text{ is gound}\}$. We write $dep_f(t) = [l, h]$ to denote that $dep_f(t)$ can be at least l and at most h. Note that two terms cannot possibly be Boolean unifiable unless their dep_f intervals overlap. Given a term t without any term variables, $dep_f(t)$ can be computed in the following way.

- $dep_f(r) = [0, 0]$, where r is a constant.
- $dep_f(b \times t) = [0, h]$, where $dep_f(t) = [l, h]$.
- $dep_f(f(t)) = [l + 1, h + 1]$, where $dep_f(t) = [l, h]$.
- $dep_f(t_1 \oplus t_2) = [0, max(h_1, h_2)]$, where $dep_f(t_1) = [l_1, h_1]$, $dep_f(t_2) = [l_2, h_2]$ and $dep_f(t_1)$ and $dep_f(t_2)$ overlap.
- $dep_f(t_1 \oplus t_2) = [max(l_1, l_2), max(h_1, h_2)]$, where $dep_f(t_1) = [l_1, h_1]$, $dep_f(t_2) = [l_2, h_2]$ and $dep_f(t_1)$ and $dep_f(t_2)$ do not overlap.

Example 5.2. $dep_f(b_1 \times r \oplus f(r)) = [1, 1]$ $\qquad$ $dep_f(r \oplus b_1 \times f(r)) = [0, 1]$.

$\mathscr{R}_{bool-unif}$

$$\frac{\{0 \overset{?}{=} 0\} \cup \Gamma \mid \tau}{\Gamma \mid \tau} \; \textit{Remove}$$

$$\frac{\{b \times t_1 \oplus t_2 \overset{?}{=} 0\} \cup \Gamma \mid \tau}{(\{t_2 \overset{?}{=} 0\} \cup \Gamma)(\{b \mapsto 0\}) \mid \tau \cup \{b \mapsto 0\}} \; \textit{Disappear}$$

$$\frac{\{b_1 \times r_1 \oplus b_2 \times r_2 \oplus t \overset{?}{=} 0\} \cup \Gamma \mid \tau}{(\{t \overset{?}{=} 0\} \cup \Gamma)\tau' \mid \tau \cup \tau'} \; \textit{Duplicate}_r$$

where $\tau' = \{b_1 \mapsto 1, b_2 \mapsto 1\}$, $dep_f(r_1)$ and $dep_f(r_2)$ overlap.

$$\frac{\{b_1 \times f(t_1) \oplus b_2 \times f(t_2) \oplus t_3 \overset{?}{=} 0\} \cup \Gamma \mid \tau}{(\{t_3 \overset{?}{=} 0, t_1 \oplus t_2 \overset{?}{=} 0\}\Gamma)\tau' \mid \tau \cup \tau'} \; \textit{Duplicate}_f$$

where $\tau' = \{b_1 \mapsto 1, b_2 \mapsto 1\}$, $dep(t_1)_f$ and $dep_f(t_2)$ overlap.

Note that when we apply the $\textit{Duplicate}_r$ rule or the $\textit{Duplicate}_f$ rule, if a term t is not a $\times$-rooted term, we consider it to be a $\times$-rooted term with coefficient 1. For example, we can make the following inference.

$$\frac{\{r \oplus b \times r \overset{?}{=} 0\} \mid \epsilon}{\{0 \overset{?}{=} 0\} \mid \{b \mapsto 1\}} \; \textit{Duplicate}_r$$

Algorithm 4 Check if two terms are Boolean unifiable

```
1: function BOOLEAN-UNIFY($t_1, t_2$)
2:     /*Assume that $t_1$ and $t_2$ are mixed terms containing no term variables.*/
3:     $st \leftarrow \{t_1 \overset{?}{=} t_2\}|\epsilon$
4:     while $st$ is not a final state or stuck state do
5:         Nondeterministically apply a rule in $\mathscr{R}_{bool-unif}$ to $st$
6:     end while
7:     if $st$ is a final state of the form $\emptyset|\tau$ then
8:         return $\tau$
9:     else
10:        return $false$
11:    end if
12: end function
```

Example 5.3. Check if $t_1 = f(r \oplus (b_1 \times r))$ and $t_2 = f(f(r \oplus (b_1 \times r)) \oplus b_2 \times r \oplus b_4 \times f(r \oplus (b_1 \times r)))$ are Boolean unifiable.

$$\{f(r \oplus (b_1 \times r)) \oplus f(f(r \oplus (b_1 \times r)) \oplus b_2 \times r \oplus b_4 \times f(r \oplus (b_1 \times r))) \overset{?}{=} 0\}|\epsilon$$
$$\xrightarrow{Duplicate_f} \{r \oplus (b_1 \times r) \oplus f(r \oplus (b_1 \times r)) \oplus b_2 \times r \oplus b_4 \times f(r \oplus (b_1 \times r)) \overset{?}{=} 0\}|\epsilon$$
$$\xrightarrow{Duplicate_r} \{f(0) \oplus b_2 \times r \oplus b_4 \times f(0) \overset{?}{=} 0\}|\{b_1 \mapsto 1\}$$
$$\xrightarrow{Disappear} \{f(0) \oplus b_4 \times f(0) \overset{?}{=} 0\}|\{b_1 \mapsto 1, b_2 \mapsto 0\}$$
$$\xrightarrow{Duplicate_f} \{0 \overset{?}{=} 0, 0 \overset{?}{=} 0\}|\{b_1 \mapsto 1, b_2 \mapsto 0, b_4 \mapsto 1\}$$
$$\xrightarrow{Remove} \{0 \overset{?}{=} 0\}|\{b_1 \mapsto 1, b_2 \mapsto 0, b_4 \mapsto 1\}$$
$$\xrightarrow{Remove} \emptyset|\{b_1 \mapsto 1, b_2 \mapsto 0, b_4 \mapsto 1\}$$

So t_1 and t_2 are Boolean unifiable under $\{b_1 \mapsto 1, b_2 \mapsto 0, b_4 \mapsto 1\}$.

5.2.2 Applying High-Level Inference Rules

Let $\mathscr{C}_{mix} = \{mt_1, mt_2, \ldots, mt_n\}$ be a set of mixed terms. Our goal is to take the XOR of a subset of terms in $\mathscr{C}_{mix}$, apply some Boolean substitution and get 0. Roughly speaking, we saturate $\mathscr{C}_{mix}$ using two rules: the *Combine* rule and the *Cancel* rule, which we will introduce later. The *Combine* rule allows us to take the XOR of two mixed terms mt_i, mt_j, and apply some Boolean substitution τ, provided that one summand of mt_i and another summand of mt_j are Boolean unifiable under τ. The *Cancel* rule allows us to take one mixed term mt_i and apply some Boolean substitution τ, provided that two summands of mt_i are Boolean unifiable under τ.

We keep track of some additional information along the way. We annotate each mixed term with a set of indices and a Boolean substitution. More formally, an *annotated mixed term amt* is of the form $mt[I; \tau]$, where mt is a mixed term, I is a set of indices and τ is a Boolean substitution. We need I and τ for the following reasons.

First, we maintain a crucial invariant during the saturation process, which is, if $mt[I; \tau]$ is generated, then $\oplus\{mt_i | i \in I\}\tau = mt$. So I and τ tells us how we can get mt.

Second, If we have $mt_1[I_1; \tau_1]$ and $mt_2[I_2; \tau_2]$, where $I_1 \cap I_2 \neq \emptyset$, we do not take the XOR of mt_1 and mt_2.

Third, If we have $mt_1[I_1; \tau_1]$ and $mt_2[I_2; \tau_2]$, where $\tau_1 \uplus \tau_2$ is undefined, we do not take the XOR of mt_1 and mt_2.

Let $MT = \{mt_1, mt_2, \ldots, mt_n\}$ be a set of mixed terms. We define *annotate*(MT) to be $\{mt_1[\{1\};\epsilon], mt_2[\{2\};\epsilon], \ldots, mt_n[\{n\};\epsilon]\}$. In Algorithm 5, ANNOTATE-AND-SATURATE(C_{mix}) first obtains $\mathscr{C}^1_{mix}$ by annotating $\mathscr{C}_{mix}$. It then saturates $\mathscr{C}^1_{mix}$ using the inference rules in $\mathscr{R}_{saturate}$, and obtains a sequence $\mathscr{C}^1_{mix}, \mathscr{C}^2_{mix}, \mathscr{C}^3_{mix}, \cdots$. $\forall i \geq 1$, $\mathscr{C}^{i+1}_{mix}$ is obtained from $\mathscr{C}^i_{mix}$ by nondeterministically making an inference and adding the result to $\mathscr{C}^i_{mix}$, if it is not already in $\mathscr{C}^i_{mix}$. $\mathscr{C}^n_{mix}$ is a saturation of $\mathscr{C}^1_{mix}$ if nothing new can be added to $\mathscr{C}^n_{mix}$. In Appendix E, we prove properties of ANNOTATE-AND-SATURATE.

We define an ordering on terms. Let mt_1 and mt_2 be two mixed terms. $mt_1 \prec mt_2$ if $dep_f(mt_1) = [l_1, h_1], dep_f(mt_2) = [l_2, h_2]$ and $h_1 < l_2$. Let mt be a mixed term of the form $b_1 \times t_1 \oplus b_2 \times t_2 \oplus \ldots \oplus b_n \times t_n$. $b_i \times t_i$ is a maximal term in mt if there does not exist any $b_j \times t_j$ s.t. $b_i \times t_i \prec b_j \times t_j$. To improve efficiency of the saturation process, we require that both the *Combine* rule and the *Cancel* rule must be applied on maximal terms. We show that this requirement does not affect completeness.

$\mathscr{R}_{saturate}$

$$\frac{b_1 \times t_1 \oplus \Gamma_1[I_1;\tau_1] \quad b_2 \times t_2 \oplus \Gamma_2[I_2;\tau_2]}{(\Gamma_1 \oplus \Gamma_2)\tau_{123}[I_1 \cup I_2;\tau_{123}]}\ \textit{Combine}$$

where
(1) $b_1 \times t_1$ is a maximal term in $b_1 \times t_1 \oplus \Gamma_1$, $b_2 \times t_2$ is a maximal term in $b_2 \times t_2 \oplus \Gamma_2$.
(2) $dep_f(b_1 \times t_1)$ and $dep_f(b_2 \times t_2)$ overlap.
(3) $I_1 \cap I_2 = \emptyset$.
(4) τ_3 is a Boolean unifier of t_1 and t_2, and $\tau_{123} = \tau_1 \uplus \tau_2 \uplus \tau_3 \uplus \{b_1 \mapsto 1, b_2 \mapsto 1\}$.

$$\frac{b_1 \times t_1 \oplus b_2 \times t_2 \oplus \Gamma[I;\tau_1]}{\Gamma\tau_{12}[I;\tau_{12}]}\ \textit{Cancel}$$

where
(1) $b_1 \times t_1$ and $b_2 \times t_2$ are maximal terms in $t_1 \oplus t_2 \oplus \Gamma_2$
(2) τ_2 is a Boolean unifier of t_1 and t_2. $\tau_{12} = \tau_1 \uplus \tau_2 \uplus \{b_1 \mapsto 1, b_2 \mapsto 1\}$.

Note that when we apply the *Combine* rule or the *Cancel* rule, if a term t is not a $\times$-rooted term, we consider it to be a $\times$-rooted term with coefficient 1. For example, we can make the following inference.

$$\frac{b_1 \times r_1 \oplus r_1[\{1\};\epsilon]}{0[\{1\};\{b_1 \mapsto 1\}]}$$

Example 5.4. Consider Example 5.1, where. $\mathscr{M}(x_1) = \{r\}$.
$\mathscr{M}(x_2) = \{r, f(r \oplus x_1) \oplus x_1\}$.
$\mathscr{C} = \{r, f(r \oplus x_1) \oplus x_1, f(f(r \oplus x_1) \oplus x_2)\}$.

REMOVE-TERM-VARIABLES$(\mathscr{M}, \mathscr{C})$ returns:
$\gamma^{\mathscr{M}} = \{x_1 \mapsto b_1 \times r, x_2 \mapsto b_2 \times r \oplus b_4 \times f(r \oplus x_1)\}$.
$\mathscr{C}_{mix} = \{r, f(r \oplus (b_1 \times r)) \oplus (b_1 \times r), f(f(r \oplus (b_1 \times r)) \oplus b_2 \times r \oplus b_4 \times f(r \oplus (b_1 \times r)))\}$.

Here is what ANNOTATE-AND-SATURATE$(\mathscr{C}_{mix})$ does.
$\mathscr{C}^1_{mix} = annotate(\mathscr{C}_{mix}) = \{r[\{1\};\epsilon], f(r \oplus (b_1 \times r)) \oplus (b_1 \times r)[\{2\};\epsilon], f(f(r \oplus (b_1 \times r)) \oplus b_2 \times r \oplus b_4 \times f(r \oplus (b_1 \times r)))[\{3\};\epsilon]\}$.

As we showed in Example 5.3, $f(r\oplus(b_1\times r))$ and $f(f(r\oplus(b_1\times r))\oplus b_2\times r\oplus b_4\times f(r\oplus(b_1\times r)))$ are unifiable under $\{b_1\mapsto 1, b_2\mapsto 0, b_4\mapsto 1\}$. We can apply the *Combine* rule, and get:

$\mathscr{C}^2_{mix} = \{r[\{1\};\epsilon], f(r\oplus(b_1\times r))\oplus(b_1\times r)[\{2\};\epsilon], f(f(r\oplus(b_1\times r))\oplus b_2\times r\oplus b_4\times f(r\oplus(b_1\times r)))[\{3\};\epsilon], r[\{2,3\},\{b_1\mapsto 1, b_2\mapsto 0, b_4\mapsto 1\}]\}$.

Obviously, r and r are unifiable. We can apply the *Combine* rule again, and get:

$\mathscr{C}^3_{mix} = \{r[\{1\};\epsilon], f(r\oplus(b_1\times r))\oplus(b_1\times r)[\{2\};\epsilon], f(f(r\oplus(b_1\times r))\oplus b_2\times r\oplus b_4\times f(r\oplus(b_1\times r)))[\{3\};\epsilon], r[\{2,3\},\{b_1\mapsto 1, b_2\mapsto 0, b_4\mapsto 1\}], 0[\{1,2,3\},\{b_1\mapsto 1, b_2\mapsto 0, b_4\mapsto 1\}]\}$.

Algorithm 5 annotate and saturate

```
1: function ANNOTATE-AND-SATURATE($\mathscr{C}_{mix}$)
2:     //$\mathscr{C}_{mix} = \{mt_1, mt_2, \dots, mt_n\}$
3:
4:     $\mathscr{C}^1_{mix} \leftarrow annotate(\mathscr{C}_{mix})$
5:     Saturate $\mathscr{C}^1_{mix}$ using the Combine rule and Cancel rule.
6:
7:     return $\mathscr{C}^n_{mix}$, which is the result of the saturation
8: end function
9:
```

5.3 Putting Things Together

Example 5.5. Continuing with Example 5.4, we showed that $0[\{1,2,3\};\{b_1\mapsto 1, b_2\mapsto 0, b_4\mapsto 1\}]\in\mathscr{C}^3_{mix}$. According to Lemma E.2, the XOR of all three terms in $\mathscr{C}_{mix}$ is 0 under $\{b_1\mapsto 1, b_2\mapsto 0, b_4\mapsto 1\}$.

We can compose $\gamma^{\mathscr{M}} = \{x_1\mapsto b_1\times r, x_2\mapsto b_2\times r\oplus b_4\times f(r\oplus x_1)\}$ with $\{b_1\mapsto 1, b_2\mapsto 0, b_4\mapsto 1\}$ and get an admissible $\mathscr{M}_\oplus$-substitution: $\sigma = \{x_1\mapsto r, x_2\mapsto f(0)\}$ s.t. $\oplus\{r, f(r\oplus x_1)\oplus x_1, f(f(r\oplus x_1)\oplus x_2)\}\sigma = 0$.

Algorithm 6 describes a function XOR-ROOTED-UNIFY for solving $\oplus$-rooted local unification. Appendix F proves properties (Soundness, Completeness and Termination) of XOR-ROOTED-UNIFY.

Algorithm 6 Solving $\oplus$-rooted local unification

```
function XOR-ROOTED-UNIFY($\mathscr{M}, \mathscr{C}$)
    //$\mathscr{M}$ must be an $\oplus$-mapping.
    //Assume: $\mathscr{C} = \{t_1, t_2, \ldots, t_n\}$.

    $(\gamma^{\mathscr{M}}, \mathscr{C}_{mix}) \leftarrow$ REMOVE-TERM-VARIABLES($\mathscr{M}, \mathscr{C}$)
    $\mathscr{C}^n_{mix} \leftarrow$ ANNOTATE-AND-SATURATE($\mathscr{C}_{mix}$)
    $result \leftarrow \emptyset$
    for all annotated mixed term $amt$ in $\mathscr{C}^n_{mix}$ do
        if $amt$ is of the form $0[I; \tau] \in \mathscr{C}^n_{mix}$ then
            $result \leftarrow result \cup (I, \gamma^{\mathscr{M}} \circ \tau)$   //$\{t_i | i \in I\}$ is $\mathscr{M}_{\oplus}$-unifiable with 0 under $\gamma^{\mathscr{M}} \circ \tau$
        end if
    end for
    if $result = \emptyset$ then
        return not $\mathscr{M}_{\oplus}$-unifiable
    else
        return ($\mathscr{M}_{\oplus}$-unifiable, $result$)
    end if
end function
```

6 Discussion

In this paper, we introduce a new unification problem, called local XOR unification. And we present two concrete versions of the local XOR unification problem. f-rooted local unification can be used to analyze modes of operation including Cipher Block Chaining, Propagating Cipher Block Chaining, Accumulated Block Cipher, etc. $\oplus$-rooted local unification can be used to analyze modes of operation including Cipher Feedback Mode, Output Feedback Mode, etc.

In Section 5, we discuss a method for solving the $\oplus$-rooted local unification problem. The notion of $dep_f(t)$, which we introduced in Section 5.2.1, is crucial for efficiency. We require that the *Combine* rule and the *Cancel* rule must be applied on maximal terms, which overlap in depth. For this reason, if we apply our method to Output Feedback Mode (or Cipher Feedback Mode), no inference is allowed. The saturation process terminates immediately. This matches the intuition why Output Feedback Mode is secure. In Output Feedback Mode, we generate a block of random bits r, compute $f(r), f(f(r)), f(f(f(r)))\ldots$, and use them to mask $x_1, x_2, x_3, \ldots$ just like one-time pad. The adversary cannot get rid of $f(r), f(f(r)), f(f(f(r))), \ldots$. Therefore Output Feedback Mode is secure. Using our method, any inference must be applied on $f(r), f(f(r)), f(f(f(r))), \ldots$, since they are the maximum summands in the $\oplus$-rooted terms where they occur. Since no two of them can possibly unify and cancel with each other, there is no way to make any progress towards obtaining 0.

The unification problems that we consider in this paper are related to security of modes of operation in the bounded setting, where a bounded number of cipher-text blocks are considered. As ongoing work, we are considering security of modes of operation in the unbounded setting, where we consider an unbounded number of cipher-text blocks. We come up with sufficient conditions for security of modes of operation. We also propose a method for checking those conditions. The idea is based on the notion of dep_f of a term.

References

[1] Siva Anantharaman, Hai Lin, Christopher Lynch, Paliath Narendran, and Michaël Rusinowitch. Cap unification: application to protocol security modulo homomorphic encryption. In *AsiaCCS 2010*, pages 192–203, 2010.

[2] Serdar Erbatur, Santiago Escobar, Deepak Kapur, Zhiqiang Liu, Christopher Lynch, Catherine A. Meadows, José Meseguer, Paliath Narendran, Sonia Santiago, and Ralf Sasse. Asymmetric unification: A new unification paradigm for cryptographic protocol analysis. In *CADE 13*, pages 231–248, 2013.

[3] Santiago Escobar, Deepak Kapur, Christopher Lynch, Catherine A. Meadows, José Meseguer, Paliath Narendran, and Ralf Sasse. Protocol analysis in maude-npa using unification modulo homomorphic encryption. In *PPDP 2011*, pages 65–76, 2011.

[4] Antoine Joux, Gwenaelle Martinet, and Frederic Valette. Blockwise-adaptive attackers revisiting the (in)security of some provably secure encryption modes: CBC, GEM, IACBC. In *22nd Annual International Cryptology Conference*, pages 17–30, 2002.

[5] L. R. Knudsen. Block chaining modes of operation. Reports in Informatics, ISSN 0333-3590, Department of Informatics, University of Bergen, Norway., 2000.

[6] Zhiqiang Liu and Christopher Lynch. Efficient general unification for xor with homomorphism. In *CADE*, pages 407–421, 2011.

[7] Catherine Meadows. Symbolic and computational reasoning about cryptographic modes of operation. Cryptology ePrint Archive, Report 2020/142, 2020. https://eprint.iacr.org/2020/794.

A An Algorithm for Checking $\mathscr{M}_f$-substitution

Algorithm 7 Checking if some substitution τ is an admissible $\mathscr{M}_f$-substitution

```
function $\mathscr{M}_f$-ADMISSIBLE($\mathscr{M}, \tau$)
        // Checks if $\tau$ is an admissible $\mathscr{M}_f$-substitution.
    for all $x \in domain(\tau)$ do
        if COMPUTABLE-BY-ADVERSARY(($x\tau, \tau, \mathscr{M}, x$)) = false then
            return $false$
        end if
    end for
    return $true$
end function

function COMPUTABLE-BY-ADVERSARY($t, \tau, \mathscr{M}, x$)
    /* Checks if $t\tau = t'\tau$, for some $t' \prec_{\mathscr{M}_f} x$. Or intuitively $t\tau$ is computable by an adversary
when generating $x$.*/

    if $t$ is an $f$-rooted term or a constant then
        Check if $t\tau \in \mathscr{M}(x)\tau$
    else
        if $t = h(t')$ then
            return COMPUTABLE-BY-ADVERSARY($t', \tau, \mathscr{M}, x$)
        else        //Assume $t = t_1 \oplus \ldots \oplus t_n$
            return COMPUTABLE-BY-ADVERSARY($t_1, \tau, \mathscr{M}, x$) and $\cdots$ and COMPUTABLE-BY-
ADVERSARY($t_n, \tau, \mathscr{M}, x$)
        end if
    end if
end function
```

The function $\mathscr{M}_f$-ADMISSIBLE in Algorithm 7 checks if some substitution is an admissible $\mathscr{M}_f$-substitution. It is based on the following 3 lemmas. Note that these 3 lemmas require that $\mathscr{M}$ is an f-mapping. Otherwise if $\mathscr{M}$ maps each variable to a set of arbitrary terms including $\oplus$-rooted terms, the lemmas do not hold. For example, suppose $\mathscr{M}(x) = \{f(r_1) \oplus f(r_2), f(r_1)\}$. By Definition 3.2, $(f(r_1) \oplus f(r_2)) \oplus f(r_1) \prec_{\mathscr{M}_f} x$. $f(r_2) =_\oplus (f(r_1) \oplus f(r_2)) \oplus f(r_1)$, but for all $t \in \mathscr{M}(x)$, $f(r_2) \neq_\oplus t$.

Lemma A.1. *Let $\mathscr{M}$ be an f-mapping, θ be a substitution. If t is an f-rooted term or a constant, and $t\theta =_\oplus t'\theta$, for some $t' \prec_{\mathscr{M}_f} x$, then $t' \in \mathscr{M}(x)$.*

Proof. Induction on $\prec_{\mathscr{M}_f}$ relation.

Suppose that $t' \in \mathscr{M}(x)$, this is trivial.

Suppose that $t' = h(t'')$, where $t'' \prec_{\mathscr{M}_f} x$. This contradicts with our assumption that t is an f-rooted term or a constant.

Suppose that $t' = t_1 \oplus t_2 \oplus \ldots \oplus t_n$, $t_1 \prec_{\mathscr{M}_f} x$, $t_2 \prec_{\mathscr{M}_f} x$, …, $t_n \prec_{\mathscr{M}_f} x$. We consider three cases. (1) If $t'\theta$ is an $\oplus$-rooted term, then this contradicts with our assumption that t is an f-rooted term or a constant. (2) If $t'\theta = 0$, we know that $0 \in \mathscr{M}(x)$. (3) If $t'\theta$ is some $t_i\theta$, by induction hypothesis, $t_i \in \mathscr{M}(x)$.

□

Lemma A.2. *Let $\mathscr{M}$ be an f-mapping, θ be a substitution. If t is an h-rooted term, and $t\theta =_{\oplus} t'\theta$, for some $t' \prec_{\mathscr{M}_f} x$, then t' must be of the form $h(t'')$, where $t'' \prec_{\mathscr{M}_f} x$.*

Proof. Induction on $\prec_{\mathscr{M}_f}$ relation.

Suppose that $t' \in \mathscr{M}(x)$. Since $\mathscr{M}$ is an f-mapping, t' is either an f-rooted term or a constant, which contradicts with our assumption that t is an h-rooted term.

Suppose that $t' = h(t'')$, where $t'' \prec_{\mathscr{M}_f} x$. This lemma holds trivially.

Suppose that $t' = t_1 \oplus t_2 \oplus \ldots \oplus t_n$, $t_1 \prec_{\mathscr{M}_f} x$, $t_2 \prec_{\mathscr{M}_f} x$, …, $t_n \prec_{\mathscr{M}_f} x$. We consider three cases. (1) If $t'\theta$ is an $\oplus$-rooted term, then this contradicts with our assumption that t is an h-rooted term. (2) If $t'\theta = 0$, again this contradicts with our assumption that t is an h-rooted term. (3) If $t'\theta$ is some $t_i\theta$, by induction hypothesis, t_i must be of the form $h(t'')$, where $t'' \prec_{\mathscr{M}_f} x$.

□

Lemma A.3. *Let $\mathscr{M}$ be an f-mapping, θ be a substitution. If t is an $\oplus$-rooted term, and $t\theta =_{\oplus} t'\theta$, for some $t' \prec_{\mathscr{M}_f} x$, then t' must an $\oplus$-rooted term of the form $t_1 \oplus t_2 \oplus \ldots \oplus t_n$, where $t_i \prec_{\mathscr{M}_f} x$ $(1 \leq i \leq n)$.*

Proof. Induction on $\prec_{\mathscr{M}_f}$ relation.

Suppose that $t' \in \mathscr{M}(x)$. Since $\mathscr{M}$ is an f-mapping, t' is either an f-rooted term or a constant, which contradicts with our assumption that t is an $\oplus$-rooted term.

Suppose that $t' = h(t'')$, $t'' \prec_{\mathscr{M}_f} x$. This contradicts with our assumption that t is an $\oplus$-rooted term.

Suppose that $t' = t_1 \oplus t_2 \oplus \ldots \oplus t_m$, where each t_i is not an $\oplus$-rooted term, and $t_i \prec_{\mathscr{M}_f} x$. We consider three cases. (1) If $t'\theta$ is an $\oplus$-rooted term, then this lemma holds trivially. (2) If $t'\theta = 0$, this contradicts with our assumption that t is an h-rooted term. (3) If $t'\theta$ is some $t_i\theta$, again this contradicts with our assumption that t is an h-rooted term.. □

Lemma A.4. *Given any f-mapping $\mathscr{M}$ and any substitution τ, $\mathscr{M}_f$-ADMISSIBLE$(\mathscr{M}, \tau)$ always terminates, furthermore it returns true if and only if τ is an admissible $\mathscr{M}_f$ substitution.*

Proof. $\mathscr{M}_f$-ADMISSIBLE$(\mathscr{M}, \tau)$ checks if for all $x \in domain(\mathscr{M})$, $x\tau = t\tau$, for some $t \prec_{\mathscr{M}_f} x$. We consider 3 cases:

(1) If $x\tau$ is an f-rooted term or a constant, by Lemma A.1, we only need to check if $x\tau = t'\tau$, for some $t' \in \mathscr{M}(x)$.

(2) If $x\tau = h(t')$, by Lemma A.2, we only need to check if $t'\tau = t''\tau$, for some $t'' \prec_{\mathscr{M}_f} x$.

(3) If $x\tau = t_1 \oplus \ldots \oplus t_n$, by Lemma A.3, we only need to check if $t_i = t'\tau$, for some $t' \prec_{\mathscr{M}_f} x$.

CHECK-TERM$(t, \mathscr{M}, x, \tau)$ is a recursive function. Each time it calls CHECK-TERM$(t', \mathscr{M}, x, \tau)$, t' must be a subterm of t. So it always terminates. Therefore $\mathscr{M}_f$-ADMISSIBLE$(\mathscr{M}, \tau)$ always terminates. □

B Properties of f-rooted-unify

In this section, we show that f-ROOTED-UNIFY is sound, complete and terminating. Therefore it is a decision procedure for f-rooted local unification, which we introduced in Section 3.1.

B.1 Termination

We use $\mathscr{BOP}(\tau)$ to denote the set of *bad open positions* in τ. So

$\mathscr{BOP}(\tau) = \{(x\tau, p) \mid (x\tau, p) \text{ is a bad open position in } \tau\}$

The following theorem states that for any substitution τ, $\mathscr{I}_1(\tau)$ always terminates. The idea that each inference either instantiates a variable in $domain(\mathscr{M})$ or fixes a bad open position in τ.

We define $M_i = (|domain(\mathscr{M}) - domain(\tau_i)|, |\mathscr{BOP}(\tau_i)|)$.

Theorem B.1 (Termination)**.** *Given an f-mapping, two f-rooted terms t_1 and t_2, f-ROOTED-UNIFY$(\mathscr{M}, t_1, t_2)$ always terminates.*

Proof. We only need to show that FIX(τ) terminates, where $\tau \in CSU_{\oplus}(t_1 \overset{?}{=} t_2)$.

For any $i > 0$, $\tau_{i+1} = Prev(\tau_i, \sigma_i)$ or $\tau_{i+1} = Cancel(\tau_i, \sigma_i)$. We will show that $M_1, M_2, M_3, \ldots$ is a decreasing sequence. There are two cases to consider.

Case 1. Suppose that some variable in $domain(\mathscr{M})$ is instantiated by applying either the $Prev$ rule or the $Cancel$ rule at the i-th iteration. $|domain(\mathscr{M}) - domain(\tau_{i+1})| < |domain(\mathscr{M}) - domain(\tau_i)|$, therefore $M_{i+1} < M_i$.

Case 2. Suppose that no variable in $domain(\mathscr{M})$ is instantiated at the i-th iteration. This means that $Domain(\sigma_i)$ contains only new variables that are generated by the procedure for unification modulo XOR. In this case, $|domain(\mathscr{M}) - domain(\tau_{i+1})| = |domain(\mathscr{M}) - domain(\tau_i)|$, but both the $Prev$ rule and the $Cancel$ rule fix at least one bad open position in τ_i, so $\mathscr{BOP}(\tau_{i+1}) < \mathscr{BOP}(\tau_i)$. Therefore $M_{i+1} < M_i$.

Since $M_1, M_2, M_3, \ldots$ is well founded, FIX(τ) eventually terminates. □

B.2 Soundness

Theorem B.2 (Soundness)**.** *Given an f-mapping, two f-rooted terms t_1 and t_2, if f-ROOTED-UNIFY$(\mathscr{M}, t_1, t_2)$ returns a substitution τ, then τ is a $\mathscr{M}_f$-unifier of t_1 and t_2.*

Proof. Since line 16 of Algorithm 2 checks if τ is an admissible $\mathscr{M}_f$-substitution, if f-ROOTED-UNIFY$(\mathscr{M}, t_1, t_2)$ returns τ, then τ must be an admissible $\mathscr{M}_f$-substitution.

We now show that τ is a unifier of t_1 and t_2 by induction. Initially this is true, since $\tau \in CSU_{\oplus}(t_1 =_{\oplus} t_2)$. Assume that τ_i is a unifier of t_1 and t_2. If the $Prev$ rule is used at the i-th iteration, $\tau_{i+1} = Prev(\tau_i, \sigma_i) = \tau_i\sigma_i$. If the $Cancel$ rule is used at the i-th iteration, $\tau_{i+1} = Cancel(\tau_i, \sigma_i) = \tau_i\sigma_i$. In both cases, τ_{i+1} is an instance of τ_i. Since τ_i is a unifier of t_1 and t_2, τ_{i+1} is also a unifier of t_1 and t_2.

Therefore τ is a $\mathscr{M}_f$-unifier of t_1 and t_2. □

B.3 Completeness

f-ROOTED-UNIFY can find any minimum $\mathscr{M}_f$-unifier, which we will define later. To illustrate the idea, let us consider the following example.

Example B.1. In Example 4.2,

$\tau = \{x_2 \mapsto h(r_1) \oplus f(0), x_1 \mapsto r_1, x_4 \mapsto f(r_1 \oplus r_2), x_3 \mapsto r_1\}$ is a $\mathscr{M}_f$-unifier of $f(x_2)$ and $f(h(x_3) \oplus f(0) \oplus f(x_4) \oplus f(f(x_3 \oplus r_2)))$. But there are other $\mathscr{M}_f$-unifiers, e.g.:

$\tau' = \{x_2 \mapsto h(h(r_1)) \oplus f(0), x_1 \mapsto r_1, x_4 \mapsto f(r_1 \oplus r_2), x_3 \mapsto h(r_1)\}$

$\tau'' = \{x_2 \mapsto h(h(h(r_1))) \oplus f(0), x_1 \mapsto r_1, x_4 \mapsto f(r_1 \oplus r_2), x_3 \mapsto h(h(r_1))\}$.

We want to be able to say that τ is a minimum $\mathscr{M}_f$-unifier, but τ' and τ'' are not minimum $\mathscr{M}_f$-unifiers.

First we can transform one substitution τ into another substitution τ^L, which is defined as follows.

Definition B.1. Let τ be a substitution, L be a list of open positions in τ. τ^L is a substitution, and

- $x\tau^L = x\tau[t]|_p$, where $(x\tau, p) \in L$, t is a subterm of $x\tau|_p$ and $t \neq x\tau|_p$.
- $y\tau^L = y\tau$, if $(y\tau, p) \in L$.

We use $\mathscr{R}(\tau)$ to denote the set of all possible substitutions that can be obtained by the transformation described above.

$\mathscr{R}(\tau) = \{\tau^L \mid L \text{ is a list of open positions.}\}$

Definition B.2. Let $\mathscr{M}$ be an f-mapping, t_1 and t_2 be two f-rooted terms. τ is a *minimum $\mathscr{M}_f$-unifier* of t_1 and t_2 if

(1) τ is a $\mathscr{M}_f$-unifier of t_1 and t_2.

(2) $\forall \tau' \in \mathscr{R}(\tau)$, τ' is not a $\mathscr{M}_f$-unifier of t_1 and t_2.

Let us consider Example B.1. By Definition B.2, τ is a minimum $\mathscr{M}_f$-unifier of $f(x_2)$ and $f(h(x_3) \oplus f(0) \oplus f(x_4) \oplus f(f(x_3 \oplus r_2)))$. But τ' is not, since $\tau'^{\{(x_2\tau', 1), (x_3\tau', \lambda)\}} = \tau$, and τ is a $\mathscr{M}_f$-unifier of t_1 and t_2.

In the following definition, we define potential $\mathscr{M}_f$-minimum unifiers of two terms t_1 and t_2. And we will show that if we start from a potential $\mathscr{M}_f$-minimum unifier τ of t_1 and t_2, we can keep composing τ with some other substitution by making inferences, until we get a minimum $\mathscr{M}_f$-unifier of t_1 and t_2.

Definition B.3. Let $\mathscr{M}$ be an f-mapping, t_1 and t_2 be two f-rooted terms. τ is a *potential $\mathscr{M}_f$-unifier* of t_1 and t_2 if there exists a substitution σ s.t. $\tau\sigma$ is a $\mathscr{M}_f$-unifier of t_1 and t_2. τ is a potential minimum $\mathscr{M}_f$-unifier of t_1 and t_2 if there exists a substitution σ s.t. $\tau\sigma$ is a minimum $\mathscr{M}_f$-unifier of t_1 and t_2.

Let $\mathscr{M}$ be an f-mapping, t_1 and t_2 be two f-rooted terms. By Lemma 4.1, if θ is a $\mathscr{M}_f$-unifier of t_1 and t_2, θ should not contain any bad open position. If some substitution τ contains some bad open position $(x\tau, p)$, it can be dealt with in two possible ways. If $x\tau|_p$ is a summand of a term t, it is possible that $x\tau|_p$ unifies with some other summand of t under some substitution σ, therefore gets cancelled. In Example 4.2, this is how we deal with $(x_1\tau, 3)$ and $(x_1\tau, 4)$. The other possibility is that we can extend $(x\tau, p)$, and get some bad innermost open position $(x\tau, p')$. By Lemma A.1, we only need to unify $x\tau|_{p'}$ with some term $t' \in \mathscr{M}(x)$. In Example 4.2, this is how we deal with $(x_2\tau, 1)$. We extend it to $(x_2\tau, 1.1)$, which is a bad innermost open position. We unify $x_2\tau|_{1.1}$ with r_1, which is in $\mathscr{M}(x_2)$.

Lemma B.3. *Let $\mathscr{M}$ be an f-mapping, τ be a substitution. For any bad open position $(x\tau, p)$ in τ, $(x\tau, p)$ can be extended to some bad innermost open position $(x\tau, p')$ in τ.*

Proof. Induction on $len(p') - len(p)$.

If $x\tau|_p$ is a variable, constant, or an f-rooted term, $(x\tau, p)$ is already a bad innermost open position. So $len(p') - len(p) = 0$.

Suppose that $x\tau|_p = h(t_1)$. $(x\tau, p.1)$ must also be bad open position in τ. Assume that $(x\tau, p.1)$ can be extended to a bad innermost open position $(x\tau, p')$ in τ by appending $i_1.i_2...i_m$ to $p.1$. Therefore (t, p) can be extended to the same bad innermost open position (t, p') by appending $1.i_1.i_2...i_m$ to p.

Suppose that $x\tau|_p = t_1 \oplus t_2 \oplus \ldots \oplus t_n$. There must exist some i s.t. $(x\tau, p.i)$ is also a bad open position in τ. Assume that $(x\tau, p.i)$ can be extended to a bad innermost open position $(x\tau, p')$ in τ by appending $i_1.i_2...i_m$ to $p.i$. $(x\tau, p)$ can be extended to the same bad innermost open position $(x\tau, p')$ by appending $i.i_1.i_2...i_m$ to p. □

Lemma B.4. *Let $\mathscr{M}$ be an f-mapping, t_1 and t_2 be two f-rooted terms. If τ is not a $\mathscr{M}_f$-unifier of t_1 and t_2, and $\tau\sigma$ is a minimum $\mathscr{M}_f$-unifier of t_1 and t_2, then for any bad open position $(x\tau, p)$, at least one of the following is true.*

(1) $(x\tau, p)$ can be extended to some bad innermost open position $(x\tau, p')$ s.t. $(x\tau\ |_{p'} =_\oplus t\tau)\sigma$, for some $t \in \mathscr{M}(x)$.

(2) $x\tau\ |_p$ is a summand of an $\oplus$-rooted term t, t has some other summands $t'_1, t'_2, \ldots, t'_m$ s.t. $(x\tau\ |_p \oplus t'_1 \oplus t'_2 \oplus \ldots \oplus t'_m)\sigma = 0$.

Proof. Let $(x\tau, p)$ be a bad open position in τ. By Lemma B.3, $(x\tau, p)$ can be extended to some bad innermost open position $(x\tau, p')$. If (2) does not hold, then $(x(\tau\sigma), p)$ is also an open position. We know that $\tau\sigma$ is a minimum $\mathscr{M}_f$-unifier of t_1 and t_2. By Lemma 4.1, $x(\tau\sigma)|_{p'} =_\oplus t(\tau\sigma)$, for some $t \prec_{\mathscr{M}_f} x$. So $(x\tau\ |_{p'} =_\oplus t\tau)\sigma$. We now show that $t \in \mathscr{M}(x)$. Since $(x\tau, p')$ is an innermost open position, $x\tau\ |_{p'}$ can only be a variable or an f-rooted term, or a constant.

Case 1. $x\tau\ |_{p'}$ is a constant or f-rooted term. By Lemma A.1, $t \in \mathscr{M}(x)$.

Case 2. $x\tau\ |_{p'} = x'$. Then $x' \notin domain(\tau)$ (Otherwise, $x\tau\ |_{p'} = x'\tau$). Since $\tau\sigma$ is a minimum $\mathscr{M}_f$-unifier, $t \in \mathscr{M}(x)$ (In order words, t is not built up from terms in $\mathscr{M}(x)$ using h and $\oplus$). □

The following Lemma B.5 is our key lemma for proving completeness. According to Lemma B.5, a potential minimum $\mathscr{M}_f$-unifier is either good enough in the sense that it is already a minimum $\mathscr{M}_f$-unifier, or we can take an inference step to get another substitution, which remains to be a potential minimum $\mathscr{M}_f$-unifier. According to Theorem B.1, there cannot be infinitely many inference steps, eventually we must get a minimum $\mathscr{M}_f$-unifier.

Lemma B.5. *Let $\mathscr{M}$ be an f-mapping. t_1 and t_2 are two f-rooted terms. If τ is a potential minimum $\mathscr{M}_f$-unifier of t_1 and t_2, then at least one of the following must be true:*

(1) τ is a minimum $\mathscr{M}_f$-unifier of t_1 and t_2.
(2) The Prev rule is applicable. $Prev(\tau, \delta)$ is a potential minimum $\mathscr{M}_f$-unifier of t_1 and t_2.
(3) The Cancel rule is applicable. $Cancel(\tau, \delta)$ is a potential minimum $\mathscr{M}_f$-unifier of t_1 and t_2.

Proof. If (1) does not hold, by Lemma 4.1, there must be some bad open position $(x\tau, p)$. By Lemma B.4, there are two cases to consider.

Case 1: $(x\tau, p)$ can be extended to some innermost open position $(x\tau, p')$ s.t. $(x\tau\ |_{p'} =_\oplus t\tau)\sigma$, for some $t \in \mathscr{M}(x)$. In this case, we use the *Prev* rule and get $Prev(\tau, \delta)$, where $\delta \in CSU_\oplus(x\tau\ |_{p'} \overset{?}{=} t\tau)$. $\sigma = \delta\delta'$, for some δ'. Since $\tau\sigma$ is a minimum $\mathscr{M}_f$-unifier of t_1 and t_2, so is $\tau\delta\delta'$. Therefore $\tau\delta$, which is $Prev(\tau, \delta)$, is a potential minimum $\mathscr{M}_f$-unifier of t_1 and t_2.

Case 2: $x\tau\ |_p$ is a summand of an $\oplus$-rooted term t, t has some other summands $t'_1, t'_2, \ldots, t'_m$ s.t. $(x\tau\ |_p \oplus t'_1 \oplus t'_2 \oplus \ldots \oplus t'_m)\sigma =_\oplus 0$. In this case, we use the *Cancel* rule and get $Cancel(\tau, \delta)$, where $\delta \in CSU_\oplus((x\tau\ |_p \oplus t'_1 \oplus t'_2 \oplus \ldots \oplus t'_m) \overset{?}{=} 0)$. $\sigma = \delta\delta'$, for some δ'. Since $\tau\sigma$ is a minimum $\mathscr{M}_f$-unifier of t_1 and t_2, so is $\tau\delta\delta'$. Therefore $\tau\delta$, which is $Cancel(\tau, \delta)$, is a potential minimum $\mathscr{M}_f$-unifier of t_1 and t_2. □

Theorem B.6 (Completeness)**.** *Let $\mathscr{M}$ be an f-mapping, t_1 and t_2 be two f-rooted terms. If t_1 and t_2 are $\mathscr{M}_f$-unifiable, then* f-ROOTED-UNIFY$(\mathscr{M}, t_1, t_2)$ *outputs a minimum $\mathscr{M}_f$-unifier of t_1 and t_2.*

Proof. If t_1 and t_2 are $\mathscr{M}_f$-unifiable, then there exists a substitution $\tau \in CSU_\oplus(t_1 \overset{?}{=} t_2)$ s.t. τ is a potential minimum $\mathscr{M}_f$-unifier. By Lemma B.5, the loop invariant at Line 13 in Algorithm 2 is that τ is always a potential minimum $\mathscr{M}_f$-unifier of t_1 and t_2. By Theorem B.1, the loop (Line 13 through Line 15) in Algorithm 2 eventually terminates, at which point no rule applies to τ. Therefore τ must be a minimum $\mathscr{M}_f$-unifier of t_1 and t_2.

□

C Properties of remove-term-variables

In this appendix, we show that if REMOVE-TERM-VARIABLES($\mathscr{M}, \mathscr{C}$) (from Section 5.1) returns $(\gamma^{\mathscr{M}}, \mathscr{C}_{mix})$, then $\gamma^{\mathscr{M}} \circ \tau$ is always an admissible $\mathscr{M}_{\oplus}$-substitution, where τ is any Boolean substitution.

In Section 5.1, we define an operator $\bullet$. Let $\vec{B} = \langle b_1, b_2, \ldots, b_n \rangle$ be a vector of Boolean variables, $T \bullet \vec{B}$ is defined to be $b_1 \times t_1 \oplus b_2 \times t_2 \oplus \ldots b_n \times t_n$. $T \bullet \vec{B}$ is called a *linear combination* of $t_1, t_2, \ldots, t_n$. T defines a *space of terms* $SPACE_T$. A term $t \in SPACE_T$ if and only if there exists some Boolean vector $\vec{B}$ such that $t = T \bullet \vec{B}$. Let $\vec{B_1} = \langle b_{11}, b_{12}, \ldots, b_{1n} \rangle$ and $\vec{B_2} = \langle b_{21}, b_{22}, \ldots, b_{2n} \rangle$ be two vectors. We define $\vec{B_1} \oplus \vec{B_2}$ to be $\langle b_{11} \oplus b_{21}, b_{12} \oplus b_{22}, \ldots, b_{1n} \oplus b_{2n} \rangle$.

The following two lemmas state some basic properties about spaces.

Lemma C.1. *Let T be a set of terms. If $t_i \in SPACE_T$ for all $1 \leq i \leq n$, then $t_1 \oplus t_2 \oplus \ldots \oplus t_n \in SPACE_T$.*

Proof. For all i, there exists $\vec{B_i}$ such that $T \bullet \vec{B_i} = t_i$. So $T \bullet (\vec{B_1} \oplus \vec{B_2} \oplus \ldots \oplus \vec{B_n}) = t_1 \oplus t_2 \oplus \ldots \oplus t_n$. Therefore, $t_1 \oplus t_2 \oplus \ldots \oplus t_n \in SPACE_T$. □

Lemma C.2. *Suppose that $T_1 \subseteq T_2$, then $SPACE_{T_1} \subseteq SPACE_{T_2}$.*

Proof. Suppose that $T_1 = \{t_1, t_2, \ldots, t_m\}$, $T_2 = \{t_1, t_2, \ldots, t_m, t_{m+1}, \ldots, t_n\}$.

Take any $t \in SPACE_{T_1}$. $t = T_1 \bullet \langle b_1, b_2, \ldots, b_m \rangle$. Then $t = T_2 \bullet \langle b_1, b_2, \ldots, b_m, 0, \ldots, 0 \rangle$. Therefore $t \in SPACE_{T_2}$. □

Given any $\oplus$-mapping $\mathscr{M}$, we require that if $i < j$, then $\mathscr{M}(x_i) \subseteq \mathscr{M}(x_j)$. By Lemma C.2, if $i < j$, then $SPACE_{\mathscr{M}(x_i)} \subseteq SPACE_{\mathscr{M}(x_j)}$.

We define γ^* to be $\{x_1 \mapsto t'_1, x_2 \mapsto t'_2, \ldots, x_m \mapsto t'_m\}$, where $t'_i = \mathscr{M}(x_1) \bullet \vec{B_i}$, and $\vec{B_i}$ is a vector of fresh Boolean variables. Let $\tau^{\mathscr{M}}$ be the set of all Boolean substitutions for all the Boolean variables occurring in $\vec{B_i}$ ($1 \leq i \leq m$). We define $\sigma^{\mathscr{M}}$ to be $\{\gamma^* \circ \tau | \tau \in \tau^{\mathscr{M}}\}$. We define $\hat{\sigma}$ to be $\{\gamma^{\mathscr{M}} \circ \tau | \tau \in \tau_{\mathscr{M}}\}$. We show that $\hat{\sigma}$ is the set of all admissible $\mathscr{M}_{\oplus}$-substitution. We do this in two steps. We first show that $\sigma^{\mathscr{M}}$ is the set of all admissible $\mathscr{M}_{\oplus}$-substitutions in Lemma C.3. We then show that $\sigma^{\mathscr{M}} = \hat{\sigma}$ in Lemma C.6. Lemma C.4 and Lemma C.5 are used to prove Lemma C.6.

Lemma C.3. *$\sigma^{\mathscr{M}}$ is the set of all admissible $\mathscr{M}_{\oplus}$-substitutions.*

Proof. Take any admissible $\mathscr{M}_{\oplus}$-substitution σ_1, σ_1 maps x_i to the XOR of terms in some subset S_i of $\mathscr{M}(x_i)$. Assume that $\mathscr{M}(x_i) = \{s_1, \ldots, s_n\}$.

Take any substitution σ_2 from $\sigma^{\mathscr{M}}$. For any x_i, σ_2 maps x_i to $\mathscr{M}(x_i) \bullet \vec{B_i}$.

We set B_{ij} to 1 if and only if $s_i \in S_i$, where B_{ij} is the j-th component of $\vec{B_i}$. Then $\sigma_1 = \sigma_2$. □

Lemma C.4. *Given any substitution σ from $\sigma^{\mathscr{M}}$, $\forall i$, $SPACE_{\mathscr{M}(x_i)\sigma} \subseteq SPACE_{NonVar(\mathscr{M}(x_i))\sigma}$.*

Proof. Induction on i.

$\mathscr{M}(x_1)$ only contains ground terms, therefore $\mathscr{M}(x_1)\sigma = NonVar(\mathscr{M}(x_1))\sigma$. The lemma holds trivially.

Assume that $SPACE_{\mathscr{M}(x_j)\sigma} \subseteq SPACE_{NonVar(\mathscr{M}(x_j))\sigma}$, for all $j < i$. We show that for all $t \in \mathscr{M}(x_i)\sigma$, $t \in SPACE_{NonVar(\mathscr{M}(x_i))\sigma}$. Therefore any linear combination of terms in $\mathscr{M}(x_i)\sigma$ is also in $SPACE_{NonVar(\mathscr{M}(x_i))\sigma}$ by Lemma C.1. Hence $SPACE_{\mathscr{M}(x_i)\sigma} \subseteq SPACE_{NonVar(\mathscr{M}(x_i))\sigma}$. We consider two cases.

Case 1. If $t \in \mathscr{M}(x_{i-1})\sigma$, then t is in $SPACE_{\mathscr{M}(x_{i-1})\sigma}$, therefore also in $SPACE_{NonVar(\mathscr{M}(x_{i-1}))\sigma}$ by induction hypothesis.

Case 2. Take any $t \in \mathscr{M}(x_i) - \mathscr{M}(x_{i-1})$, $t\sigma = NonVar(t)\sigma \oplus Var(t)\sigma$. For any variable x_j in $Var(t)$, $j < i$. Therefore $x_j\sigma \in SPACE_{\mathscr{M}(x_j)\sigma}$, where $j < i$. By induction hypothesis, $x_j\sigma \in SPACE_{NonVar(\mathscr{M}(x_j))\sigma}$. By Lemma C.2, $x_j\sigma \in SPACE_{NonVar(\mathscr{M}(x_i))\sigma}$. So $Var(t)\sigma \in SPACE_{NonVar(\mathscr{M}(x_i))\sigma}$. We know that $NonVar(t)\sigma \in SPACE_{NonVar(\mathscr{M}(x_i))\sigma}$. By Lemma C.1, $t\sigma \in SPACE_{NonVar(\mathscr{M}(x_i))\sigma}$. □

Lemma C.5. *Given any substitution σ from $\hat{\sigma}$, $\forall i$, $SPACE_{NonVar(\mathscr{M}(x_i))\sigma} \subseteq SPACE_{\mathscr{M}(x_i)\sigma}$.*

Proof. Induction on i.

$\mathscr{M}(x_1)$ only contains ground terms, therefore $NonVar(\mathscr{M}(x_1))\sigma = \mathscr{M}(x_1)\sigma$. The lemma hold trivially.

Assume that $SPACE_{NonVar(\mathscr{M}(x_j))\sigma} \subseteq SPACE_{\mathscr{M}(x_j)\sigma}$, for all $j < i$. We show that for any $t \in NonVar(\mathscr{M}(x_i))\sigma$, $t \in SPACE_{\mathscr{M}(x_i)\sigma}$. Therefore any linear combination of terms in $NonVar(\mathscr{M}(x_i))\sigma$ is also in $SPACE_{\mathscr{M}(x_i)\sigma}$ by Lemma C.1. Hence $SPACE_{NonVar(\mathscr{M}(x_i))\sigma} \subseteq SPACE_{\mathscr{M}(x_i)\sigma}$. We consider two cases.

Case 1. If $t \in NonVar(\mathscr{M}(x_{i-1}))\sigma$, then t is in $SPACE_{NonVar(\mathscr{M}(x_{i-1}))\sigma}$, therefore also in $SPACE_{\mathscr{M}(x_{i-1})\sigma}$ by induction hypothesis.

Case 2. Take any $t \in \mathscr{M}(x_i) - \mathscr{M}(x_{i-1})$, $NonVar(t)\sigma = t\sigma \oplus Var(t)\sigma$. For any variable x_j in $Var(t)$, $j < i$. Therefore $x_j\sigma \in SPACE_{NonVar(\mathscr{M}(x_j))\sigma}$, where $j < i$. By induction hypothesis, $x_j\sigma \in SPACE_{\mathscr{M}(x_j)\sigma}$. By Lemma C.2, $x_j\sigma \in SPACE_{\mathscr{M}(x_i)\sigma}$. So $Var(t)\sigma \in SPACE_{\mathscr{M}(x_i)\sigma}$. We know that $t\sigma \in SPACE_{\mathscr{M}(x_i)\sigma}$. By Lemma C.1, $NonVar(t)\sigma \in SPACE_{\mathscr{M}(x_i)\sigma}$. □

Lemma C.6. $\sigma^{\mathscr{M}} = \hat{\sigma}$.

Proof. Take any $\sigma \in \sigma^{\mathscr{M}}$, for any $x_i \in domain(\sigma)$, $x_i\sigma$ is some term t in $SPACE_{\mathscr{M}(x_i)\sigma}$. By Lemma C.4, t is also in $SPACE_{NonVar(\mathscr{M}(x_i))\sigma}$. Therefore $t \in \hat{\sigma}$.

Take any $\sigma \in \hat{\sigma}$, for any $x_i \in domain(\sigma)$, $x_i\sigma$ is some term t in $SPACE_{NonVar(\mathscr{M}(x_i))\sigma}$. By Lemma C.5, t is also in $SPACE_{\mathscr{M}(x_i)\sigma}$. Therefore $t \in \sigma^{\mathscr{M}}$. □

Lemma C.7. *$\hat{\sigma}$ is the set of all admissible $\mathscr{M}_{\oplus}$-substitutions.*

Proof. By Lemma C.3, $\sigma^{\mathscr{M}}$ is the set of all admissible $\mathscr{M}_{\oplus}$-substitutions. By Lemma C.6, $\sigma^{\mathscr{M}} = \hat{\sigma}$. □

D Properties of boolean-unify

Lemma D.1 (Soundness)**.** *Let t_1 and t_2 be two terms without term variables. If* BOOLEAN-UNIFY*(t_1, t_2) returns τ, then τ is a Boolean unifier of t_1 and t_2.*

Proof. We prove an invariant by induction, and we will use this invariant to prove this lemma. Here is the invariant. For each state st in a trace of t_1 and t_2, if $st = Eqs|\tau$ and Eqs is unifiable, then $\tau \cup \tau'$ is a Boolean unifier of t_1 and t_2, where τ' is any Boolean unifier of S.

If st is the initial state, the invariant holds trivially.

Assume that the invariant holds for $st_i = Eqs_i|\tau_i$. Let τ_i' be any Boolean unifier of Eqs_i, then $\tau_i' \cup \tau_i$ is a Boolean unifier of t_1 and t_2. We apply an inference and get $st_{i+1} = Eqs_{i+1}|\tau_{i+1}$. We want to show that $\tau_{i+1}' \cup \tau_{i+1}$ is a Boolean unifier of t_1 and t_2, where τ_{i+1}' is any Boolean unifier of Eqs_{i+1}. We consider 4 cases:

(1) If the *Remove* rule is applied, $\tau_{i+1} = \tau_i$ and $\tau_{i+1}' = \tau_i'$. Since $\tau_i' \cup \tau_i$ is a Boolean unifier of t_1 and t_2, $\tau_{i+1}' \cup \tau_{i+1}$ is also a Boolean unifier of t_1 and t_2.

(2) If the *Disappear* rule is applied on a term $b \times t$, If for any Boolean unifier τ_i' of t_1 and t_2, $b\tau_i' \neq 0$, S_{i+1} is not unifiable, the invariant holds trivially. Otherwise, $\tau_{i+1} = \tau_i \cup \{b \mapsto 0\}$, $\tau_i' = \tau_{i+1}' \cup \{b \mapsto 0\}$. Since $\tau_i' \cup \tau_i$ is a Boolean unifier of t_1 and t_2, $\tau_{i+1}' \cup \tau_{i+1}$ is also a Boolean unifier of t_1 and t_2.

(3) If the *Duplicate$_r$* rule is applied on $b_1 \times r$ and $b_2 \times r$. If for any Boolean unifier τ_i' of t_1 and t_2, $b_1\tau_i' \neq 1$ or $b_2\tau_i' \neq 1$, S_{i+1} is not unifiable, the invariant holds trivially. Otherwise, $\tau_{i+1} = \tau_i \cup \{b_1 \mapsto 1, b_2 \mapsto 1\}$, $\tau_i' = \tau_{i+1}' \cup \{b_1 \mapsto 1, b_2 \mapsto 1\}$. Since $\tau_i' \cup \tau_i$ is a Boolean unifier of t_1 and t_2, $\tau_{i+1}' \cup \tau_{i+1}$ is also a Boolean unifier of t_1 and t_2.

(4) If the *Duplicate$_f$* rule is applied on $b_1 \times f(t)$ and $b_2 \times f(t')$. If for any Boolean unifier τ_i' of t_1 and t_2, $b_1\tau_i' \neq 1$ or $b_2\tau_i' \neq 1$, S_{i+1} is not unifiable, the invariant holds trivially. Otherwise, $\tau_{i+1} = \tau_i \cup \{b_1 \mapsto 1, b_2 \mapsto 1\}$, $\tau_i' = \tau_{i+1}' \cup \{b_1 \mapsto 1, b_2 \mapsto 1\}$. Since $\tau_i' \cup \tau_i$ is a Boolean unifier of t_1 and t_2, $\tau_{i+1}' \cup \tau_{i+1}$ is also a Boolean unifier of t_1 and t_2.

The invariant holds for all states in the trace. In particular, it holds for the final state $st_n = \emptyset|\tau_n$. Therefore τ_n, which is what BOOLEAN-UNIFY(t_1, t_2) returns, is a Boolean unifier of t_1 and t_2. □

Lemma D.2 (Termination)**.** *Let t_1 and t_2 be two terms without term variables, any trace of t_1 and t_2 is finite.*

Proof. Each inference decrease the total occurrences of f symbol and constants.

If the *Disappear* rule is used, an f symbol or a constant disappears.

If the *Duplicate$_r$* rule is used, two duplicate constants are cancelled, which decreases the total occurrences by at least 2.

If the *Duplicate$_f$* rule is used, the total occurrences decreases by at least 2.

If the *Remove* rule is used, two 0's are removed, the total occurrences decreases by 2. □

Lemma D.3 (Completeness)**.** *Let t_1 and t_2 be two terms without term variables. If τ is a Boolean unifier of t_1 and t_2, then* BOOLEAN-UNIFY*(t_1, t_2) returns τ (for some nondeterministic choices of the inference rules).*

Proof. Consider a trace $st_1, st_2, \ldots, st_n$, where $st_1 = \{t_1 \oplus t_2 \overset{?}{=} 0\}|\epsilon$. We show that for any i, if st_i is of the form $Eqs_i|\tau_i$, where $Eqs_i \neq \emptyset$ and Eqs_i is unifiable under some τ_i', then we can make an inference to get $Eqs_{i+1}|\tau_{i+1}$, where Eqs_{i+1} is unifiable under some τ_{i+1}' and $\tau_i \cup \tau_i' = \tau_{i+1} \cup \tau_{i+1}'$.

Consider the first equation eq in Eqs_i. Suppose that eq is $0 \stackrel{?}{=} 0$, we apply the *Remove* rule. Otherwise suppose that eq is of the form $b_1 \times t \oplus t' \stackrel{?}{=} 0$. We consider 3 cases.

(1) If $b_1\tau'_i = 0$, we apply the *Disappear* rule. $\tau_{i+1} = \tau_i \cup \{b_1 \mapsto 0\}$, $\tau'_i = \tau'_{i+1} \cup \{b_1 \mapsto 0\}$, therefore $\tau_i \cup \tau'_i = \tau_{i+1} \cup \tau'_{i+1}$.

(2) If $b_1\tau'_i = 1$ and t is a constant r, then we apply the *Duplicate*$_r$ rule on $b_1 \times r$ and $b_2 \times r$, which is a summand of t'. $\tau_{i+1} = \tau_i \cup \{b_1 \mapsto 1, b_2 \mapsto 1\}$, $\tau'_i = \tau'_{i+1} \cup \{b_1 \mapsto 1, b_2 \mapsto 1\}$, therefore $\tau_i \cup \tau'_i = \tau_{i+1} \cup \tau'_{i+1}$.

(3) If $b_1\tau'_i = 1$ and t is of the form $f(t'')$, then we apply the *Duplicate*$_r$ rule on $b_1 \times f(t'')$ and $b_2 \times f(t''')$, which is a summand of t'. $\tau_{i+1} = \tau_i \cup \{b_1 \mapsto 1, b_2 \mapsto 1\}$, $\tau'_i = \tau'_{i+1} \cup \{b_1 \mapsto 1, b_2 \mapsto 1\}$, therefore $\tau_i \cup \tau'_i = \tau_{i+1} \cup \tau'_{i+1}$.

So $\tau_1 \cup \tau'_1 = \tau_2 \cup \tau'_2 = \ldots = \tau_n \cup \tau'_n$.

For the initial state, $\tau_1 = \epsilon$, $\tau'_1 = \tau$. For the final state $\tau'_n = \epsilon$. Therefore, $\tau_n = \tau$, which is what BOOLEAN-UNIFY(t_1, t_2) returns. $\square$

E Properties of annotate-and-saturate

In this appendix, we prove properties of ANNOTATE-AND-SATURATE.

Throughout the process of saturating $\mathscr{C}^1_{mix}$, we maintain the following invariant.

Lemma E.1. *Let $\mathscr{C}^1_{mix} = \{mt_1[\{1\};\epsilon], mt_2[\{2\};\epsilon], \ldots, mt_n[\{n\};\epsilon]\}$. For all i, if $t[I;\tau] \in \mathscr{C}^i_{mix}$, then $(\oplus\{mt_i | i \in I\})\tau =_\oplus t$.*

Proof. We prove this lemma by induction on i.

When $i = 1$, this is trivial.

Case 1: Suppose that $b_1 \times t_1 \oplus \Gamma_1[I_1;\tau_1]$ and $b_2 \times t_2 \oplus \Gamma_2[I_2;\tau_2]$ are in $\mathscr{C}^i_{mix}$, we apply the *Combine* rule and get $\Gamma_1 \oplus \Gamma_2[I_1 \cup I_2; \tau_1 \uplus \tau_2 \uplus \tau_3 \uplus \{b_1 \mapsto 1, b_2 \mapsto 1\}]$, where τ_3 is a Boolean unifier of t_1 and t_2.

By induction hypothesis, $(\oplus\{tm_i | i \in I_1\})\tau_1 =_\oplus b_1 \times t_1 \oplus \Gamma_1$, and $(\oplus\{tm_i | i \in I_2\})\tau_2 =_\oplus b_2 \times t_2 \oplus \Gamma_2$. So $(\oplus\{tm_i | i \in I_1 \cup I_2\})\tau =_\oplus \Gamma_1 \oplus \Gamma_2$, where $\tau = \tau_1 \uplus \tau_2 \uplus \tau_3 \uplus \{b_1 \mapsto 1, b_2 \mapsto 1\}$.

Case 2: Suppose that $b_1 \times t_1 \oplus b_2 \times t_2 \oplus \Gamma[I;\tau_1]$ is in $\mathscr{C}^i_{mix}$, we apply the *Cancel* rule and get $\Gamma[I; \tau_1 \uplus \tau_2 \uplus \{b_1 \mapsto 1, b_2 \mapsto 1\}]$, where τ_2 is a Boolean unifier of t_1 and t_2.

By induction hypothesis, $(\oplus\{tm_i | i \in I\})\tau_1 =_\oplus b_1 \times t_1 \oplus b_2 \times t_2 \oplus \Gamma$. So $(\oplus\{tm_i | i \in I\})\tau = \Gamma$, where $\tau = \tau_1 \uplus \tau_2 \uplus \{b_1 \mapsto 1, b_2 \mapsto 1\}$.

□

The following lemma follows directly from Lemma E.1

Lemma E.2 (Soundness)**.** *Let $\mathscr{C}_{mix} = \{mt_1, mt_2, \ldots, mt_n\}$. If $0[I;\tau] \in$ ANNOTATE-AND-SATURATE$(\mathscr{C}_{mix})$, then $(\oplus\{mt_i | i \in I\})\tau = 0$.*

Proof. According to Lemma E.1, since $0[I;\tau] \in \mathscr{C}^n_{mix}$, $(\oplus\{mt_i | i \in I\})\tau = 0$. □

Let $\mathscr{C}_{mix} = \{mt_1, mt_2, \ldots, mt_n\}$. Let $BVar(\mathscr{C}_{mix})$ denotes the set of all Boolean variables in $\mathscr{C}_{mix}$. We define the following set $\mathscr{S}$. Intuitively $\mathscr{S}$ is the set of all possible things that can occur in $\mathscr{C}^n_{mix}$. $\mathscr{S}$ is finite, therefore the saturation process always terminates.

$$\mathscr{S} = \{(\oplus\{mt_i | i \in I\})\tau\ [I;\tau] \mid I \subseteq \{1,2,\ldots,n\}, domain(\tau) \subseteq BVar(\mathscr{C}_{mix})\}.$$

Lemma E.3 (Termination)**.** *For any set of mixed terms $\mathscr{C}_{mix}$, ANNOTATE-AND-SATURATE$(\mathscr{C}_{mix})$ always terminates.*

Proof. After inference step i, $\mathscr{S} - \mathscr{C}^i_{mix}$ becomes smaller than $\mathscr{S} - \mathscr{C}^{i-1}_{mix}$. Since $\mathscr{S}$ is a finite set, the saturation process always terminates. □

Definition E.1. Let $\mathscr{C}^1_{mix} = \{mt_1[\{1\};\epsilon], mt_2[\{2\};\epsilon], \ldots, mt_n[\{n\};\epsilon]\}$. Let I be a set of integers, τ be a Boolean substitution. $\mathscr{C}^i_{mix}$ is *solvable under* (I,τ) if $\mathscr{C}^i_{mix}$ contains $mt_{i_1}[I_{i_1};\tau_{i_1}], mt_{i_2}[I_{i_2};\tau_{i_2}], \ldots mt_{i_m}[I_{i_m};$
s.t.

(i) $I_{i_1} \cup I_{i_2} \cup \ldots \cup I_{i_m} = I$, and $\forall p,q,\ p \neq q \rightarrow I_{i_p} \cap I_{i_q} = \emptyset$.

(ii) There exists some τ' s.t. $(mt_{i_1} \oplus mt_{i_2} \oplus \ldots mt_{i_m})\tau' = 0$ and $\tau' \uplus \tau_{i_1} \uplus \ldots \uplus \tau_{i_m} = \tau$.

The following lemma is used to prove Lemma E.5

Lemma E.4. *For all i, if $\mathscr{C}^i_{mix}$ is solvable under (I,τ), then one of the following must be true*

(1) $0[I;\tau] \in \mathscr{C}^i_{mix}$.

(2) An inference rule is applicable, and $\mathscr{C}^{i+1}_{mix}$ is also solvable under (I,τ).

Proof. Since $\mathscr{C}^i_{mix}$ is solvable under (I, τ), $\mathscr{C}^i_{mix}$ contains $mt_{i_1}[I_{i_1}; \tau_{i_1}], mt_{i_2}[I_{i_2}; \tau_{i_2}], \ldots mt_{i_m}[I_{i_m}; \tau_{i_m}]$ s.t.

(i) $I_{i_1} \cup I_{i_2} \cup \ldots \cup I_{i_m} = I$, and $\forall p, q,\ p \neq q \rightarrow I_{i_p} \cap I_{i_q} = \emptyset$.

(ii) There exists some τ' s.t. $(mt_{i_1} \oplus mt_{i_2} \oplus \ldots mt_{i_m})\tau' = 0$ and $\tau' \uplus \tau_{i_1} \uplus \ldots \uplus \tau_{i_m} = \tau$.

If $0[I; \tau] \notin \mathscr{C}^i_{mix}$, pick a maximum summand $b \times t$ of $t_{i_1} \oplus t_{i_2} \oplus \ldots t_{i_m}$, there must exist some other summand $b' \times t'$ of $t_{i_1} \oplus t_{i_2} \oplus \ldots t_{i_m}$ s.t. $(b \times t)\tau' =_{\oplus} (b' \times t')\tau'$. We consider two cases:

(1) Suppose that both $b \times t$ and $b' \times t'$ occur in some t_{i_j} $(1 \leq j \leq m)$, so $t_{i_j} = (b \times t) \oplus (b' \times t' \oplus t'')$ We apply the *Cancel* rule, and get $t''[I_{i_j}; \tau_{i_1} \uplus \tau'_1]$, where τ'_1 is a Boolean unifier of $b \times t$ and $b' \times t'$. Then $\tau' = \tau'_1 \uplus \tau'_2$, for some τ'_2. Now $\mathscr{C}^{i+1}_{mix}$ contains $t_{i_1}[I_{i_1}; \tau_{i_1}], \ldots, t''[I_{i_j}; \tau_{i_j} \cup \cdot\ \tau'_1], \ldots t_{i_m}[I_{i_m}; \tau_{i_m}]$ s.t.

(i) $I_{i_1} \cup I_{i_2} \cup \ldots \cup I_{i_m} = I$, and $\forall p, q,\ p \neq q \rightarrow I_{i_p} \cap I_{i_q} = \emptyset$.

(ii) $(t_{i_1} \oplus \ldots \oplus t'' \oplus \ldots t_{i_m})\tau'_2 = 0$ and $\tau'_2 \uplus \tau_{i_1} \uplus \ldots \uplus \tau'_1 \uplus \ldots \uplus \tau_{i_m} = \tau$.

Therefore, $\mathscr{C}^{i+1}_{mix}$ is also solvable under (I, τ). $b \times t$ and $b' \times t'$.

(2) Otherwise, suppose that $b \times t$ is a summand of t_{i_x}, and $b' \times t'$ is a summand of t_{i_y} $(x \neq y)$. Suppose that $t_{i_x} = b \times t \oplus t''$ and $t_{i_y} = b' \times t' \oplus t'''$. By applying the *Combine* rule, we get $t'' \oplus t'''[I_{i_x} \uplus I_{i_y}; \tau_{i_x} \uplus \tau_{i_x} \uplus \tau'_1]$, where τ'_1 is a Boolean unifier of $b \times t$ and $b' \times t'$. Then $\tau' = \tau'_1 \uplus \tau'_2$, for some τ'_2. Now $\mathscr{C}^{i+1}_{mix}$ contains $t_{i_1}[I_{i_1}; \tau_{i_1}], \ldots, t'' \oplus t'''[I_{i_x} \cup I_{i_y}; \tau_{i_x} \uplus \tau_{i_y} \uplus \tau'_1], \ldots t_{i_m}[I_{i_m}; \tau_{i_m}]$ s.t.

(i) $I_{i_1} \cup I_{i_2} \cup \ldots \cup I_{i_m} = I$, and $\forall p, q,\ p \neq q \rightarrow I_{i_p} \cap I_{i_q} = \emptyset$.

(ii) $(t_{i_1} \oplus \ldots \oplus t'' \oplus t''' \oplus \ldots t_{i_m})\tau'_2 = 0$ and $\tau'_2 \uplus \tau_{i_1} \uplus \ldots \uplus \tau'_1 \uplus \ldots \uplus \tau_{i_m} = \tau$.

Therefore, $\mathscr{C}^{i+1}_{mix}$ is also solvable under (I, τ). $b \times t$ and $b' \times t'$.

□

Lemma E.5 (Completeness)**.** *Let* $\mathscr{C}_{mix} = \{mt_1, mt_2, \ldots, mt_n\}$*, and* $\mathscr{C}^n_{mix} =$ ANNOTATE-AND-SATURATE*(*$\mathscr{C}_{mix}$*). If* $\oplus\{mt_i \mid i \in I\}\tau = 0$*, then* $0[I; \tau] \in \mathscr{C}^n_{mix}$*.*

Proof. If $\oplus\{mt_i \mid i \in I\}\tau = 0$, then $\mathscr{C}^1_{mix}$ is solvable under (I, τ). According to Lemma E.4, for all i, either $0[I; \tau] \in \mathscr{C}^i_{mix}$ or we can make an inference s.t. $\mathscr{C}^{i+1}_{mix}$ is still solvable under (I, τ). According to Lemma E.3, eventually no inference is possible, therefore $0[I; \tau] \in \mathscr{C}^n_{mix}$. □

F Properties of xor-rooted-unify

We prove the following properties about XOR-ROOTED-UNIFY.

Theorem F.1 (Soundness of XOR-ROOTED-UNIFY)**.** *Given any $\oplus$-mapping $\mathscr{M}$, and any set of pure terms $\mathscr{C} = \{t_1, t_2, \ldots, t_n\}$. If* XOR-ROOTED-UNIFY$(\mathscr{M}, \mathscr{C})$ *returns ($\mathscr{M}_\oplus$-unifiable, result), then for any (I, σ) in result, $\{t_i | i \in I\}$ is $\mathscr{M}_\oplus$-unifiable with 0 under σ.*

Proof. Let $\mathscr{C}_{mix} = \{mt_1, mt_2, \ldots, mt_n\}$. Since $\mathscr{C}_{mix} = \mathscr{C}\gamma^{\mathscr{M}}$, $mt_i = t_i\gamma^{\mathscr{M}}$ $(1 \leq i \leq n)$.

By Lemma E.2, if $0[I; \tau] \in$ ANNOTATE-AND-SATURATE$(\mathscr{C}_{mix})$, then $(\oplus\{mt_i | i \in I\})\tau = 0$. Therefore $(\oplus\{t_i | i \in I\})\gamma_{\mathscr{M}}\tau = 0$, which means that $(\oplus\{t_i | i \in I\})(\gamma_{\mathscr{M}} \circ \tau) = 0$.

$\gamma_{\mathscr{M}} \circ \tau \in \hat{\sigma}$. By Lemma C.7, $\gamma_{\mathscr{M}} \circ \tau$ is an admissible $\mathscr{M}_\oplus$-substitution. □

Theorem F.2 (Termination of XOR-ROOTED-UNIFY)**.** *For any $\oplus$-mapping and any set of pure terms $\mathscr{C}$,* XOR-ROOTED-UNIFY$(\mathscr{M}, \mathscr{C})$ *always terminates.*

Proof. Obviously, REMOVE-TERM-VARIABLES$(\mathscr{M}, \mathscr{C})$ terminates. By Lemma E.3, ANNOTATE-AND-SATURATE$(\mathscr{C}_{mix})$ terminates. Therefore XOR-ROOTED-UNIFY$(\mathscr{M}, \mathscr{C})$ terminates. □

Theorem F.3 (Completeness of XOR-ROOTED-UNIFY)**.** *Given any $\oplus$-mapping $\mathscr{M}$, and any set of pure terms $\mathscr{C} = \{t_1, t_2, \ldots, t_n\}$. If $\{t_i | i \in I\}$ is $\mathscr{M}_\oplus$-unifiable with 0 under σ, then* XOR-ROOTED-UNIFY$(\mathscr{M}, \mathscr{C})$ *returns ($\mathscr{M}_\oplus$-unifiable, result), where $(I, \sigma) \in$ result.*

Proof. $\{t_i | i \in I\}$ is $\mathscr{M}_\oplus$-unifiable with 0 under σ. By Definition 3.6, $\oplus\{t_i | i \in I\}\sigma = 0$, and σ is an admissible $\mathscr{M}_\oplus$-substitution. By Lemma C.7. $\sigma \in \hat{\sigma}$, which means that there exits some Boolean substitution τ s.t. $\gamma^{\mathscr{M}} \circ \tau = \sigma$. Since $\mathscr{C}_{mix} = \mathscr{C}\gamma^{\mathscr{M}}$, $mt_i = t_i\gamma^{\mathscr{M}}$ $(1 \leq i \leq n)$. Then $\oplus\{t_i | i \in I\}(\gamma^{\mathscr{M}} \circ \tau) = 0$, which means that $\oplus\{mt_i | i \in I\}\tau = 0$. By Lemma E.5, $0[I; \tau] \in \mathscr{C}^n_{mix}$. Therefore, XOR-ROOTED-UNIFY$(\mathscr{M}, \mathscr{C})$ returns ($\mathscr{M}_\oplus$-unifiable, *result*), where $(I, \sigma) \in$ *result*. □