Internship Report 2026 - Symbolic Bideduction

Zacharie Moughanim
supervised by David Baelde and Stéphanie Delaune

1. Introduction & Background

We study the bideduction problem in the symbolic model of cryptography. In this model, messages are terms over a signature Σ of functions and their arity, variables $\mathcal{X}$ and *names* $\mathcal{N}$ representing the randoms.

In this context, a term is either:
- a variable;
- a name;
- if $f \in \Sigma$ is a function of arity a and $T_0, ..., T_{a-1}$ are terms, then $f(T_0, ..., T_{a-1})$ is a term.

Example. With $\Sigma_{\text{enc}} = \{\mathsf{enc}/2, \mathsf{dec}/2\}$, $\mathsf{enc}(\mathsf{dec}(n, k_0), k_1)$, x, $\mathsf{dec}(y, z)$ are terms.

A **context** is a term with a hole. Holes are written $\Box$.

The semantics of functions is embedded in an *equational theory* generated by a set of equations between terms X. We write $T \underset{E}{=} U$ "T and U are equal up to the equational theory E". The generated theory corresponds to the closure by application of context and by substitution of names and variables i.e.
- If $T = U \in X$ and C is a context, then $C[T] \underset{E}{=} C[U]$;
- If $T = U \in X$, $S_0, ..., S_{k-1}$ are terms and $x_0, ..., x_{k-1}$ are variables, then $T[S_0/x_0, ..., S_{k-1}/x_{k-1}] \underset{E}{=} U[S_0/x_0, ..., S_{k-1}/x_{k-1}]$.

We often give equations in an oriented manner, in which case we call them *rewriting rules* and enhance their orientations by using an $\rightarrow$ symbol instead of an $=$.

Example. For decryption/encryption, we use the rules:
- $\mathsf{dec}(\mathsf{enc}(x, y), y) \rightarrow x$.

For projections/pairs we use the rules:
- $\mathsf{fst}(\langle x,y \rangle) \rightarrow x$;
- $\mathsf{snd}(\langle x,y \rangle) \rightarrow y$.

In the setting of oriented equations, it is convenient to have a single representant for a class of term up to the theory. If the rewriting system is confluent and terminates, we call *the value* V of a term T the only term such that $T \underset{E}{=} V$ and such that no rewriting rule can be applied to V. We write $T{\downarrow}$ the value of T, and $T \Downarrow V$ to say that T *evaluates* to V i.e. its value is V.

A **frame** $\phi = (M_0, ..., M_{n-1})$ is a sequence of terms, representing the sequences of messages available to the attacker.

A **recipe** R is a context (a term with holes) with multiple hole, that we'll denote by $\Box_0, ..., \Box_i, ...$ that cannot contain any name.

We denote by $R[\phi]$ the frame ϕ applied to R i.e. the term where each hole $\Box_i$ of R has been replaced by the i^{th} term of ϕ.

There are two main predicates in this model to study cryptography:
- deduction, informally, whether an attacker can compute a term given available terms, formally, given a sequence ϕ and a term T we say that T is deducible from ϕ, written $\phi \vdash T$, if there is a recipe R such that $R[\phi] \underset{E}{=} T$;

- static equivalence, informally, whether an attacker can distinguish two frames. Formally, given two frames ϕ, ϕ', we say that ϕ and ϕ' are *statically equivalent*, written $\phi \approx_s \phi'$, if for all recipes R_0, R_1, $R_0[\phi] = R_1[\phi]$ iff $R_0[\phi'] = R_1[\phi']$.

During the internship, we studied a different problem in the same setting, and we observe how its decidability relates to the aforementioned problems.

We say that T, T' are bideducible from ϕ, ϕ', written $\phi, \phi' \triangleright T, T'$ if there is a recipe R such that $R[\phi] = T$ and $R[\phi'] = T'$. In the context of bideduction in general, a pair of term is called a *biterm*; in the context of deciding whether $\phi, \phi' \triangleright T, T'$, the pair T, T' is *the target biterm.*

We weren't able to adress the problem in a general manner as proving its decidability for a general class of theory as has already been done for static equivalence and deduction [1], we thus approached the problem from different angle:

- In Section 2, we restrict the semantics given on terms, using a more restrictive setting than equational theory, and prove that bideduction is decidable if the rewriting rules verify some conditions.

2. Destructors, one rule by destructor

We first consider a variant of theories, where some particular symbol called *destructors* cannot appear in terms but only in recipes, and are thus meaningful only if they reduce.

Definition 1 : Let $\Sigma = \Sigma_{\text{constr}} \sqcup \Sigma_{\text{destr}}$ be a signature, E an equational theory generated by destructor rules of the form $d\big(U_0, ..., U_{\mathsf{ar}(d)-1}\big) \to T$ where :

- $d \in \Sigma_{\text{destr}}$ (with at most one rule by destructor);
- there is a $i_d, 0 \leq i_d < a$, a function symbol f_d and some terms (that may contain variables) $S_0, ..., S_{\mathsf{ar}(f_d)-1}$ such that $U_{i_d} = f_d\big(S_0, ..., S_{\mathsf{ar}(f_d)-1}\big)$;
- $\mathsf{vars}(T) \subseteq \mathsf{vars}(S_0) \cup ... \cup \mathsf{vars}\big(S_{\mathsf{ar}(f_d)-1}\big)$;
- for all $j, 0 \leq j < \mathsf{ar}(d), j \neq i_d$ we have $\mathsf{vars}\big(U_j\big) \subseteq \mathsf{vars}\big(U_{i_d}\big)$.

Let ϕ, ϕ' sequences of terms on that signature. We define $\mathcal{I}^{\text{destr},\Sigma,E}_{\phi,\phi'}$ be the proof system containing the following inference rules :

- input rules $(T, T') \in \mathsf{zip}(\phi, \phi') \dfrac{}{T, T'}$ (input)
- constructor rules for each function in Σ_{constr} of arity a, $\dfrac{T_0, T'_0 \quad ... \quad T_{\mathsf{ar}(f)-1}, T'_{\mathsf{ar}(f)-1}}{f\big(T_0,...,T_{\mathsf{ar}(f)-1}\big), f\big(T'_0,...,T'_{\mathsf{ar}(f)-1}\big)}$ (constr f).
- destructor rules for each rule $d\big(U_0, ..., U_{\mathsf{ar}(d)-1}\big) \to T$,

  $\mathrm{dom}(\sigma) = \mathsf{vars}\big(U_0,...,U_{\mathsf{ar}(d)-1},T\big) \dfrac{U_0\sigma, U_0\sigma' \quad ... \quad U_{\mathsf{ar}(d)-1}\sigma, U_{\mathsf{ar}(d)-1}\sigma'}{T\sigma, T\sigma'}$ (destr d)

  we call the premise containing the U_i of which T is a subterm the *main premise.*

The fact that there is at most one rule by destructor ensures the synchronisation in the reduction within a recipe, which leads in the proof system to a single inference rule by destructor.

The first restriction on destructor rules ensures that the theory is subterm; the second says that a destructor *destructs* a constructor f and reuses its argument in a new context R_d; the fourth ensures that if we know a term T_{main} matching U_{i_d}, we can determine the whole $d\big(U_0, ..., U_{\mathsf{ar}(d)-1}\big)$, without letting any variable free.

Example. $\mathsf{dec}(\mathsf{enc}(x, y), y) \to x$: the result is a subterm of $\mathsf{enc}(x, y)$, and by knowing $\mathsf{enc}(T, T')$ we can determine that the key needed to obtain T from $\mathsf{dec}(\mathsf{enc}(T, T'), \sqcup)$ must be T'.

Definition 2 : Let $\Sigma, E, \phi = (T_0, ..., T_{n-1}), \phi' = (T'_0, ..., T'_{n-1})$. We define by induction on the proofs (resp. on the recipes) the corresponding recipe (resp. proof).

First, we describe the transformation $\mathscr{R}$ of proof into recipes:

- $\mathscr{R}\left(\dfrac{}{T_i, T'_i}\,(\text{input})\right) = \square_i$;
- $\mathscr{R}\left(\dfrac{\Pi_0 \quad ... \quad \Pi_{\mathsf{ar}(f)-1}}{f\left(T_0,...,T_{\mathsf{ar}(f)-1}\right), f\left(T'_0,...,T'_{\mathsf{ar}(f)-1}\right)}\,(\text{constr } f)\right) = f\Big(\mathscr{R}(\Pi_0), ..., \mathscr{R}\left(\Pi_{\mathsf{ar}(f)-1}\right)\Big)$;
- $\mathscr{R}\left(\dfrac{\Pi_0 \quad ... \quad \Pi_{\mathsf{ar}(d)-1}}{T\sigma, T\sigma'}\,(\text{destr } d)\right) = d\Big(\mathscr{R}(\Pi_0), ..., \mathscr{R}\left(\Pi_{\mathsf{ar}(d)-1}\right)\Big)$.

Now, we describe the transformation $\mathscr{P}_{\Pi_0,...,\Pi_{k-1}}$ of recipes into proofs:

- $\mathscr{P}_{\Pi}(\square_i) = \Pi_i$;
- $\mathscr{P}_{\Pi}\Big(f\left(R_0, ..., R_{\mathsf{ar}(f)-1}\right)\Big) = \dfrac{\mathscr{P}(R_0) \quad ... \quad \mathscr{P}\left(R_{\mathsf{ar}(f)-1}\right)}{f\left(T_0,...,T_{\mathsf{ar}(f)-1}\right), f\left(T'_0,...,T'_{\mathsf{ar}(f)-1}\right)}\,(\text{constr } f)$;
- $\mathscr{P}_{\Pi}\Big(d\left(R_0, ..., R_{\mathsf{ar}(d)-1}\right)\Big) = \dfrac{\mathscr{P}(R_0) \quad ... \quad \mathscr{P}\left(R_{\mathsf{ar}(d)-1}\right)}{T\sigma, T\sigma'}\,(\text{destr } d)$.

σ in the destructor case and the T_is in the constructor case are deduced from the conclusion of the subproofs.

Notice how $\mathscr{P}$ does not depend on ϕ, ϕ', in opposition to $\mathscr{R}$. This will later allow to prove properties by induction more smoothly.

Lemma 3 : Let $\phi = (T_0, ..., T_{n-1})$, $\phi' = (T'_0, ..., T'_{n-1})$ be sequences of terms and Π a proof of T, T'.

- $\mathscr{R}(\Pi)$ is a recipe such that $\mathscr{R}(\Pi)[\phi] = T, \mathscr{R}(\Pi)[\phi'] = T'$; and
- $|\mathscr{R}(\Pi)| = |\Pi|$.

Proof. Both are proven by straightforward induction.

Lemma 4 : Let $\Pi_0, ..., \Pi_{k-1}$ be proofs of respectively $(T_0, T'_0), ..., (T_{k-1}, T'_{k-1})$ and R a recipe.

- $\mathscr{P}_{\Pi_0,...,\Pi_{k-1}}(R)$ is a proof of $R[T_0, ..., T_{k-1}], R[T'_0, ..., T'_{k-1}]$; and
- $|\mathscr{P}_{\Pi_0,...,\Pi_{k-1}}(R)| = |R| + \sum_{\square_i \text{ occurence appearing in } R} |\Pi_i| - 1$.

Proof. Both are proven by straightforward induction.

Proposition 5 : $\mathcal{I}^{\text{destr},\Sigma,E}_{\phi,\phi'}$ is a sound and complete proof system for bideduction.

Proof. Let's first show soundness.

- Let T, T' be a biterm provable in $\mathcal{I}^{\text{destr},\Sigma,E}_{\phi,\phi'}$. There is a proof tree Π in this system proving T, T'. Then we have $\mathscr{R}(\Pi)[\phi] = T$ and $\mathscr{R}(\Pi)[\phi] = T'$ by Lemma 3, thus, $\phi, \phi' \rhd T, T'$.

Now completeness.

- Let T, T' be a biterm bideducible from ϕ, ϕ'. We have a recipe R such that $R[\phi] = T$ and $R[\phi'] = T'$. Therefore, with $\Pi_0 = \dfrac{}{\phi_0, \phi'_0}\,(\text{input})$, ..., $\Pi_{n-1} = \dfrac{}{\phi_{n-1}, \phi'_{n-1}}\,(\text{input})$ we have $\mathscr{P}_{\Pi_0,...,\Pi_{n-1}}(R)$ deriving the biterm $(R[\phi_0, ..., \phi_{n-1}], R[\phi'_0, ..., \phi'_{n-1}]) = (R[\phi], R[\phi']) = (T, T')$.

Definition 6 : A destructor/constructor equational theory E is *strictly loopless* if for each destructor rule $d\Big(U_0, ..., f_d\Big(S_0, ..., S_{\mathsf{ar}(f_d)-1}\Big), ..., U_{\mathsf{ar}(d)-1}\Big) \to T$ there is a recipe R_d with $\mathsf{ar}(d) - 1 + \mathsf{ar}(f)$ holes $\Box_0, ..., \Box_{i_d-1}, \Box_{i_d^0}, ..., \Box_{i_d}^{\mathsf{ar}(f)-1}, \Box_{i_d+1}, ..., \Box_{\mathsf{ar}(d)-1}$ such that:
(1) $R_d\Big[U_0, ..., U_{i_d-1}, S_0, ..., S_{\mathsf{ar}(f_d)-1}, U_{i_d+1}, ..., U_{\mathsf{ar}(d)-1}\Big] \underset{E}{=} T$;
(2) R_d is either:
- of the form $d'\left(R_d^0, ..., R_d^{\mathsf{ar}(d')-1}\right)$ such that:
  (i) each R_d^i contains only constructor and holes; or
  (ii) $R_d^{i_{d'}} = \Box_{\sqcup}$ i.e. the main subterm does not contain a constructor;
  (iii) each $\Box_i$ appears at most once in R_d;
  (iv) $|R_d| \leq 1 + \mathsf{ar}(d) + \mathsf{ar}(f)$;

  or
- directly a hole.

Notice that in the second case of (2), the recipe still satisfies the subbullets (iii). and (iv).

We know look into a theory inspired from blind signature, used for electronic voting, which is an example where one destructor rule does not strictly extract a subterm.

Definition 7 : Let $E_{\text{blind}}^{\text{destr}}$ be the equational theory over $\Sigma = \Sigma_{\text{constr}} \sqcup \Sigma_{\text{destr}}$ where $\Sigma_{\text{constr}} = \{\mathsf{commit}, \mathsf{host}, \mathsf{pk}, \mathsf{blind}, \mathsf{sign}\}$, $\Sigma_{\text{destr}} = \{\mathsf{open}, \mathsf{getpk}, \mathsf{checksign}, \mathsf{unblind}_1, \mathsf{unblind}_2\}$ generated by the rules :
- $\mathsf{open}(\mathsf{commit}(x, y), y) = x$;
- $\mathsf{getpk}(\mathsf{host}(x)) = x$;
- $\mathsf{checksign}(\mathsf{sign}(x, y), \mathsf{pk}(y)) = x$;
- $\mathsf{unblind}_1(\mathsf{blind}(x, y), y) = x$;
- $\mathsf{unblind}_2(\mathsf{sign}(\mathsf{blind}(x, y), z), y) = \mathsf{sign}(x, z)$.

Example. $E_{\text{pairenc}}^{\text{destr}}$ is strictly loopless, with $R_{\mathsf{dec}} = \Box_0^0$, $R_{\mathsf{fst}} = \Box_0^0$ and $T_{\mathsf{snd}} = \Box_0^1$.

$E_{\text{blind}}^{\text{destr}}$ is also strictly loopless. The only rule for which it is not trivial is the rule for $\mathsf{unblind}_2$. In this case, $R_{\mathsf{unblind}_2} = \mathsf{unblind}_1(\Box_0^0, \Box_1)$ satisfies the conditions.

We now apply an argument that is already used to decide deduction [2], we restrict the search of proof tree to a particular form called decomposition, and we'll show that we can search such a proof for a given instance of bideduction in finite time, therefore deciding bideducton.

Definition 8 : A proof tree in some inference system $\mathcal{I}_{\phi,\phi'}^{\text{destr},E,\Sigma}$ is said to be in decomposition form if the last rule is a destructor, and in the main premise of this destructor rule, the last rule used is not a constructor, and all subproofs are in decomposition form.

A proof tree is said to be in decomposition/composition form if for each node whose last rule is a destructor, the associated subtree is in decomposition form.

Definition 9 : We define the destructor height of a proof by induction on proofs as :

$$\mathsf{hDestr}\left(\frac{}{V, V'}\,(\text{input})\right) = 0$$

$$\mathsf{hDestr}\left(\frac{\boxed{\Pi_0} \quad \dots \quad \boxed{\Pi_{\mathsf{ar}(d)-1}}}{V, V'}\,(\text{destr } d)\right) =$$

$$1 + \max\Big(\mathsf{hDestr}(\Pi_0), \dots, \mathsf{hDestr}\Big(\Pi_{\mathsf{ar}(d)-1}\Big)\Big)$$

$$\mathsf{hDestr}\left(\frac{\boxed{\Pi_0} \quad \dots \quad \boxed{\Pi_{\mathsf{ar}(d)-1}}}{f\big(V_0,\dots,V_{\mathsf{ar}(f)-1}\big), f\big(V_0',\dots,V_{\mathsf{ar}(f)-1}'\big)}\,(\text{constr } f)\right) =$$

$$\max\Big(\mathsf{hDestr}(\Pi_0), \dots, \mathsf{hDestr}\Big(\Pi_{\mathsf{ar}(f)-1}\Big)\Big)$$

Definition 10 : Let $\Sigma = \Sigma_{\text{constr}} \sqcup \Sigma_{\text{destr}}$ be a signature, E an equational theory generated by destructor rules, and ϕ, ϕ' sequences of terms.

Σ, E is said to satisfy decomposition/composition if for all ϕ, ϕ', V, V' such that V, V' is derivable in $\mathcal{I}^{\text{destr},\Sigma,E}_{\phi,\phi'}$ by a proof Π, there is a proof Π' of V, V' in decomposition/composition form and $\mathsf{hDestr}(\Pi') \leq \mathsf{hDestr}(\Pi)$.

Lemma 11 : Let Σ be a signature, E a strictly loopless equational theory. Then E satisfy decomposition/composition.

Proof. Let ϕ, ϕ' be sequences of terms, T, T' derivable in the system $\mathcal{I}^{\text{destr},\Sigma,E}_{\phi,\phi'}$. We show by induction on $k \in \mathbb{N}$ that if there is Π a proof of size k of T, T',then there is a proof Π' of T, T' in decomposition/composition form such that $\mathsf{hDestr}(\Pi') \leq \mathsf{hDestr}(\Pi)$.

- $|\Pi| = 1$: In this case Π must end with an (input) rule and is already of the required form.
- $|\Pi| \geq 1$ we examine further subcases depending on the last rule of Π.
  - If Π ends with a (constr) rule, then it's straightforward by applying the I.H. on the subproofs ✓.
  - If $\Pi = \dfrac{\boxed{\Pi_0} \quad \dots \quad \boxed{\Pi_{\text{main}}} \quad \dots \quad \boxed{\Pi_{\mathsf{ar}(d)-1}}}{U_0\sigma, U_0\sigma' \quad \dots \quad U_{i_d}\sigma, U_{i_d}\sigma' \quad \dots \quad U_{\mathsf{ar}(d)-1}\sigma, U_{\mathsf{ar}(d)-1}\sigma'}\,(\text{destr } d)$, there are two further subcases: either
    - the last rule in Π_{main} is a (destr) or (input) rule, in which case it's straightforward by applying the I.H. on each subproofs; or
    - the last rule is a (constr) rule, in which case it must be (constr f_d). Then, we have $\Pi = \dfrac{\dots \quad \dfrac{\boxed{\Pi^0_{\text{main}}}}{S_0\sigma, S_0\sigma'} \quad \dots \quad \dfrac{\boxed{\Pi^{\mathsf{ar}(f)-1}_{\text{main}}}}{S_{\mathsf{ar}(f)-1}\sigma, S_{\mathsf{ar}(f)-1}\sigma'} \quad \dots}{f_d\big(S_0\sigma, \dots, S_{\mathsf{ar}(f_d)-1}\sigma\big), f_d\big(S_0\sigma', \dots, S_{\mathsf{ar}(f_d)-1}\sigma'\big)}\,(\text{destr } d)}{T\sigma, T\sigma'}$. Consider now $\mathscr{P}_{\Pi^0_{\text{main}},\dots,\Pi^{\mathsf{ar}(f_d)-1}_{\text{main}}}(R_d)$, which is a proof of $R_d\big[S_0\sigma, \dots, S_{\mathsf{ar}(f_d)-1}\sigma\big], R_d\big[S_0\sigma', \dots, S_{\mathsf{ar}(f_d)-1}\sigma'\big]$ (Lemma 4) which is $T\sigma, T\sigma'$ up to E. We want to apply the I.H. to this proof, so we must ensure that its size is less than the size of Π.

      Let $\mathcal{H}_U = \{i \in \{0, \dots, i_d - 1, i_d + 1, \dots, \mathsf{ar}(d) - 1\} \mid \Box_i \text{ appears in } R_d\}$ and $\mathcal{H}_S = \Big\{j \in \{0, \dots, \mathsf{ar}(f) - 1\} \mid \Box^j_{i_d} \text{ appears in } R_d\Big\}$.

      Let $\overline{\Pi} = \Big(\Pi_0, \dots, \Pi_{i_d-1}, \Pi^0_{\text{main}}, \dots, \Pi^{\mathsf{ar}(f)-1}_{\text{main}}, \Pi_{i_d+1}, \dots, \Pi_{\mathsf{ar}(d)-1}\Big)$. Since each hole appears at most once in R_d (Lemma 4 + (5) of strict loopless):

$$
\begin{aligned}
|\mathscr{P}_{\Pi}(R_d)| &\leq |R_d| + \sum_{i \in \mathcal{H}_U} |\Pi_i| - 1 + \sum_{j \in \mathcal{H}_S} |\Pi^j_{\text{main}}| - 1 \\
&\leq 1 + \mathsf{ar}(d) + \mathsf{ar}(f) + \sum_{i \in \mathcal{H}_U} |\Pi_i| - 1 + \sum_{j \in \mathcal{H}_S} |\Pi^j_{\text{main}}| - 1 \text{ ((2.iv) of strict looplessness)} \\
&= 1 + \sum_{\substack{i=0 \\ i \neq i_d}}^{\mathsf{ar}(d)-1} 1 + \sum_{i \in \mathcal{H}_U} |\Pi_i| - 1 + \sum_{i=0}^{\mathsf{ar}(f)-1} 1 + \sum_{j \in \mathcal{H}_S} |\Pi^j_{\text{main}}| - 1 \\
&= 1 + \sum_{\substack{i=0 \\ i \neq i_d \wedge i \notin \mathcal{H}_U}}^{\mathsf{ar}(d)-1} 1 + \sum_{i \in \mathcal{H}_U} |\Pi_i| + \sum_{\substack{i=0 \\ i \notin \mathcal{H}_S}}^{\mathsf{ar}(f)-1} 1 + \sum_{j \in \mathcal{H}_S} |\Pi^j_{\text{main}}| \\
&\leq 1 + \sum_{\substack{i=0 \\ i \neq i_d}}^{\mathsf{ar}(d)-1} |\Pi_i| + \sum_{i=0}^{\mathsf{ar}(f)-1} |\Pi^j_{\text{main}}| \\
&< 1 + 1 + \sum_{\substack{i=0 \\ i \neq i_d}}^{\mathsf{ar}(d)-1} |\Pi_i| + \sum_{i=0}^{\mathsf{ar}(f)-1} |\Pi^j_{\text{main}}| \\
&= 1 + \sum_{\substack{i=0 \\ i \neq i_d}}^{\mathsf{ar}(d)-1} |\Pi_i| + \left(1 + \sum_{i=0}^{\mathsf{ar}(f)-1} |\Pi^j_{\text{main}}|\right) \\
&= 1 + \sum_{\substack{i=0 \\ i \neq i_d}}^{\mathsf{ar}(d)-1} |\Pi_i| + |\Pi_{i_d}| = 1 + \sum_{i=0}^{\mathsf{ar}(d)-1} |\Pi_i| = |\Pi|
\end{aligned}
$$

Therefore we can apply the I.H. to $\mathscr{P}_{\Pi^0_{\text{main}},\ldots,\Pi^{\mathsf{ar}(f_d)-1}_{\text{main}}}(R_d)$ which is a strictly smaller proof of $T\sigma, T\sigma'$. This concludes this case ✓.

Algorithm 12 :

$\textsc{BideduceConstrFrom}_{\Sigma_{\text{constr}}}(\Phi, T, T')$:
If $(T, T') \in \Phi$:
  `true`
Else
  Match T, T' :
    $f\left(T_0, ..., T_{\mathsf{ar}(f)-1}\right), f\left(T'_0, ..., T'_{\mathsf{ar}(f)-1}\right) \to \&\&_{0 \leq j < \mathsf{ar}(f)}\ \textsc{BideduceConstrFrom}\left(\Phi, T_j, T_j\right)$
    $\_, \_ \to$ `false`

$\textsc{Destruct}_{\Sigma_{\text{constr}}, \Sigma_{\text{destr}}}(\phi, \phi')$:
$\Phi_{\text{old}} := \varnothing$
$\Phi := \mathsf{zip}(\phi, \phi')$
Until $\Phi = \Phi_{\text{old}}$:
  $\Phi_{\text{old}} := \mathsf{copy}(\Phi)$
  For each destructor $d \in E_{\text{destr}}$ whose rule is $d\left(U_0, ..., U_{\mathsf{ar}(d)-1}\right) \to T$:
    For each $(U_{\text{main}}, U'_{\text{main}}) \in \Phi_{\text{old}}$:
      For all σ, σ' s.t. $U_{i_d}\sigma = U_{\text{main}}, U_{i_d}\sigma' = U'_{\text{main}}$:
        If $\&\&_{\substack{0 \leq j < \mathsf{ar}(d) \\ j \neq i_d}}\ \textsc{BideduceConstrFrom}\left(\Phi_{\text{old}}, U_j\sigma, U_j\sigma'\right)$:
          $\Phi := \Phi \cup \{(T\sigma, T\sigma')\}$
Φ

$\textsc{BiDeduce}_{\Sigma_{\text{constr}}, \Sigma_{\text{destr}}}(\phi, \phi', T, T')$:
$\Pi_{\text{constr}} := \textsc{Construct}(T, T')$
bideducibleDestr := $\textsc{Destruct}(\phi, \phi')$
$\textsc{BideduceConstrFrom}_{\Sigma_{\text{constr}}}(\text{bideducibleDestr}, T, T')$

Definition 13 : Let Σ, E be signature and theory according to the destructor/constructor semantics. Let the set of term provable by a decomposition proof, written $\mathsf{bidecomp}_{\Sigma,\ E}(\phi, \phi')$, be :

$$\left\{(V, V') \;\middle|\; \exists \Pi, \Pi \text{ proves } (V, V') \text{ in } \mathcal{I}^{\text{destr}, E, \Sigma_{\text{constr}} \sqcup \Sigma_{\text{destr}}}_{\phi, \phi'} \text{ and } \Pi \text{ is in decomposition form}\right\}$$

Definition 14 : For all $k \in \mathbb{N}$, we define $\mathsf{bidecomp}^k_{\Sigma,\ E}(\phi, \phi') =$

$$\left\{(V, V') \;\middle|\; \exists \Pi, \Pi \text{ proves } (V, V') \text{ in } \mathcal{I}^{\text{destr}, E, \Sigma}_{\phi, \phi'}, \Pi \text{ is in decomposition form and } \mathsf{hDestr}(\Pi) \leq k\right\}$$

Proposition 15 : It should be clear that $\mathsf{bidecomp}_{E,\ \Sigma}(\phi, \phi') = \bigcup_{k \in \mathbb{N}} \mathsf{bidecomp}^k_{E,\ \Sigma}(\phi, \phi')$.

Plus, if $\mathsf{bidecomp}^{k_0}_{E,\ \Sigma}(\phi, \phi') = \mathsf{bidecomp}^{k_0+1}_{E,\ \Sigma}(\phi, \phi')$, then for all $l \geq k_0, \mathsf{bidecomp}^{k_0}_{E,\ \Sigma}(\phi, \phi') = \mathsf{bidecomp}^{l}_{E,\ \Sigma}(\phi, \phi')$. Indeed, if that's the case then there is no decomposition proof Π such that $\mathsf{hDestr}(\Pi) = k_0 + 1$. However, considering a proof Π' s.t. $\mathsf{hDestr}(\Pi') = k_0 + 2$, one of its immediate subproof must be of destructor height exactly $k_0 + 1$, we can extend this argument to any $l \geq k_0$ by a straightforward induction.

Consequently, we have that if $\mathsf{bidecomp}^k_{E,\ \Sigma}(\phi, \phi') = \mathsf{bidecomp}^{k+1}_{E,\ \Sigma}(\phi, \phi')$, then $\mathsf{bidecomp}_{E,\ \Sigma}(\phi, \phi') = \bigcup_{k \in \mathbb{N}} \mathsf{bidecomp}^k_{E,\ \Sigma}(\phi, \phi') = \mathsf{bidecomp}^{k_0}_{E,\ \Sigma}(\phi, \phi')$.

Lemma 16 : Let Σ, E be constructor/destructor. For all set of biterm Φ and T, T' a biterm, $\textsc{BideduceConstrFrom}(\Phi, T, T') = \mathsf{true}$ iff there is a proof Π of (T, T') containing only constructor rules, and whose leafs are all in Φ.

Proof.

- If $\textsc{BideduceConstrFrom}(\Phi, T, T') = \mathsf{true}$ then we can show by a straightforward induction on the pair (T, T') that we can construct such a proof.
- If there is such a proof Π, then by a straightforward induction on Π, the syntactic constraints on the conclusion of each constructor rules implies that we'll reach l.5 and therefore that $\textsc{BideduceConstrFrom}(\Phi, T, T') = \mathsf{true}$.

Lemma 17 : Let $\Phi \subseteq \mathsf{bidecomp}^k_{\Sigma,\ E}(\phi, \phi')$ and T, T' such that $\textsc{BideduceConstrFrom}(\Phi, T, T') = \mathsf{true}$. We have $(T, T') \in \mathsf{bidecomp}^k_{\Sigma,\ E}(\phi, \phi')$. $\mathsf{hDestr}(\Pi)$

Proof. By a straightforward induction on the pair (T, T'), at each recursive call we apply a constructor rule that does not increase the destructor height.

Lemma 18 : Let $\Sigma = \Sigma_{\text{constr}} \sqcup \Sigma_{\text{destr}}$, E be constructor/destructor. If Σ, E satisfy decomposition/composition, then for all ϕ, ϕ', $\textsc{Destruct}_{\Sigma_{\text{constr}}, \Sigma_{\text{destr}}}(\phi, \phi')$ terminates and its result is $\mathsf{bidecomp}_{\Sigma,\ E}(\phi, \phi')$.

Proof. Let's show that the invariant for the **Until** loop : "Φ in $\textsc{Destruct}_{\Sigma_{\text{constr}}, \Sigma_{\text{destr}}}(\phi, \phi')$ is precisely $\mathsf{bidecomp}^k_{\Sigma,\ E}(\phi, \phi')$ after k iteration of the loop".

If we can prove this invariant, we have — thanks to the Proposition 15 — that the final set is indeed $\mathsf{bidecomp}^{\Sigma}_{E}(\phi, \phi')$.

- $k = 0$: before the first execution of the loop, Φ contains only biterms from ϕ, ϕ' directly i.e. biterms that are derivable by a single application of an (input) rule. Notice that decomposition proof with a destructor height of 0 contains no destructor rule, therefore are exactly of the form of a single application of (input). This concludes this case ✓.
- $k \geq 0$: Assume the result holds for k, we extend it to $k + 1$.
  - Let's first show that throughout the loop, $\Phi \subseteq \mathsf{bidecomp}^{k+1}_{E,\ \Sigma}(\phi, \phi')$. We only add biterm to the set when there is a rule $d\big(U_0, ..., U_{\mathsf{ar}(d)-1}\big) \to T$ with:
    - $U_{\text{main}}, U'_{\text{main}} \in \Phi_{\text{old}} = \mathsf{bidecomp}^k_{\Sigma,\ E}(\phi, \phi')$ (by I.H.), both matching U_{i_p} via σ, σ';
    - for all $j \neq i_d$, the corresponding $U_j\sigma, U_j\sigma'$ is derivable by $\textsc{BideduceConstrFrom}\big(\Phi_{\text{old}}, U_j\sigma, U_j\sigma'\big) = \textsc{BideduceConstrFrom}\Big(\mathsf{bidecomp}^k_{\Sigma,\ E}(\phi, \phi'), U_j\sigma, U_j\sigma'\Big)$.

    Therefore, we have that there is a decomposition proof Π_{main} proving $U_{\text{main}}, U'_{\text{main}}$ s.t. $\mathsf{hDestr}(\Pi_{\text{main}}) \leq k$. Plus, by Lemma 17, we also have proofs Π_j for $0 \leq j < \mathsf{ar}(d)$ and $j \neq i_d$ proving $U_j\sigma, U_j\sigma'$. Therefore, the tree $\dfrac{\boxed{\Pi_0} \quad ... \quad \boxed{\Pi_{\text{main}}} \quad ... \quad \boxed{\Pi_{\mathsf{ar}(d)-1}}}{T\sigma, T\sigma'}$ (destr d) if of destructor height $\leq k + 1$, and we have $(T\sigma, T\sigma') \in \mathsf{bidecomp}^{k+1}_{\Sigma,\ E}(\phi, \phi')$ ✓.
  - Let's now show the converse i.e. that $\mathsf{bidecomp}^{k+1}_{E,\ \Sigma}(\phi, \phi') \subseteq \Phi$.

    Let Π be a decomposition proof of some pair $(V, V') \in \mathsf{bidecomp}^{k+1}_{E,\ \Sigma}(\phi, \phi')$ of minimum destructor height. First, if $\mathsf{hDestr}(\Pi) \leq k$, $(V, V') \in \mathsf{bidecomp}^E(\Sigma, k + 1) \in \Phi$ by I.H.

    Assume $\mathsf{hDestr}(\Pi) = k + 1$. Π is of the form

    $$\dfrac{\dfrac{\boxed{\Pi_0}}{U_0\sigma, U_0\sigma'} \quad ... \quad \dfrac{\boxed{\Pi_{\text{main}}}}{U_{i_d}\sigma, U_{i_d}\sigma'} \quad ... \quad \dfrac{\boxed{\Pi_{i_d}}}{U_{\mathsf{ar}(d)-1}\sigma, U_{\mathsf{ar}(d)-1}\sigma'}\ (\text{destr } d)}{T\sigma, T\sigma'}$$

    and all subproofs are of destructor height $\leq k$. Since Π_{pincipal} is in decomposition form, we directly have that $\big(U_{i_d}\sigma, U_{i_d}\sigma'\big) \in \mathsf{bidecomp}^k_{\Sigma,\ E}(\phi, \phi') = \Phi_{\text{old}}$, therefore, line 8 is reached with σ, σ' and $U_{i_d}\sigma, U_{i_d}\sigma'$.

We now want to prove that for each j s.t. $0 \leq j < \mathsf{ar}(d), j \neq i_d$ we have $\textsc{BideduceConstrFrom}\big(\Phi_{\text{old}}, U_j\sigma, U_j\sigma'\big) = \mathtt{true}$. Let j be any such index.

Since E, Σ satisfy decomposition/composition, there is a Π'_j in decomposition form proving $U_j\sigma, U_j\sigma'$ such that $\mathsf{hDestr}\big(\Pi'_j\big) \leq \mathsf{hDestr}\big(\Pi_j\big) \leq k$. Consider any node of Π'_j such that the last rule used to prove this node is a destructor. The corresponding subproof is therefore destructor height $\leq k$, and thus, the label of this node is in $\mathsf{bidecomp}^{k}_{\Sigma,\ E}(\phi, \phi') = \Phi_{\text{old}}$. Hence, by Lemma 16, $\textsc{BideduceConstrFrom}\big(\Phi_{\text{old}}, U_j\sigma, U_j\sigma'\big) = \mathtt{true}$.

Finally, since this is true for each such j, we enter the then branch of the conditional statement, and add (V, V') to Φ, thus $(V, V') \in \Phi$ after this execution of the loop ✓.

This concludes the proof of the invariant and of the lemma.

Lemma 19 : Let Σ, E be constructor/destructor. If Σ, E satisfy decomposition/composition, then bideduction is decidable.

Proof. Let's show that in this case, $\textsc{BiDeduce}_{\Sigma_{\text{constr}}, \Sigma_{\text{destr}}}(\phi, \phi', T, T') = \mathtt{true}$ iff $\phi, \phi' \rhd T, T'$.

First, BiDeduce terminates since the only call to Destruct terminates by assumption, and BideduceConstrFrom may recursively call itself at most $\min(|T|, |T'|)$ times.

Let's first prove the algorithm is correct i.e. that if $\textsc{BiDeduce}_{\Sigma_{\text{constr}}, \Sigma_{\text{destr}}}(\phi, \phi', T, T') = \mathtt{true}$ then $\phi, \phi' \rhd T, T'$. We'll then prove the converse.

- By Lemma 18, we already have that bideducibleDestr l.2 of BiDeduce is a set of derivable pair in the proof system. Plus, notice that the match case in BideduceConstrFrom corresponds to the application of a constructor rule on the target terms, we thus get the correction.
- Assume $\phi, \phi' \rhd T, T'$. In this case, by Definition 10, we have a proof Π of the form described in the definition whose root is T, T'.

  Let Π^{constr} be the greatest cut of Π containing only constructor rules i.e. for each leaf in Π^{constr}, the last rule applied on this leaf in Π is a destructor rule or (input).

  Let's show by induction on Π^{constr} that $\textsc{BideduceConstrFrom}(\textsc{Destruct}(\phi, \phi'), T, T') = \mathtt{true}$.
  - If Π^{constr} is a leaf, then Π is in decomposition form, therefore $(T, T') \in \textsc{Destruct}(\phi, \phi')$ by Lemma 18, and therfore $\textsc{BideduceConstrFrom}(\textsc{Destruct}(\phi, \phi'), T, T') = \mathtt{true}$ ✓.
  - Otherwise, there is $f \in \Sigma_{\text{constr}}$ such that $\Pi^{\text{constr}} = \dfrac{\overset{\Pi_0^{\text{constr}}}{\triangle} \quad \cdots \quad \overset{\Pi_{\mathsf{ar}(f)-1}^{\text{constr}}}{\triangle}}{f\big(T_0,\ldots,T_{\mathsf{ar}(f)-1}\big), f\big(T'_0,\ldots,T'_{\mathsf{ar}(f)-1}\big)}\,(\text{constr } f)$. We have two cases, either
    - $(T, T') \in \textsc{Destruct}(\phi, \phi')$ and $\textsc{BideduceConstrFrom}(\textsc{Destruct}(\phi, \phi'), T, T') = \mathtt{true}$ ✓ or
    - the first condition fails and we go in the else branch. By I.H. applied to the recursive call we have $\textsc{BideduceConstrFrom}(\textsc{Destruct}(\phi, \phi'), T_i, T'_i) = \mathtt{true}$, and therefore $\textsc{BideduceConstrFrom}(\textsc{Destruct}(\phi, \phi'), T, T') = \mathtt{true}$ ✓.

3. A destructor case where the algorithm fails

When we consider a more intricate equational theory, without destructors or even with destructors but several rules by destructor, the inference system derived from the equational theory need not yield a one-to-one correspondence between recipes and proof trees. Since a given destructor might reduce with distinct equations on the left and the right side, the proof tree must account, not only for the syntax of the recipe, but also to the evaluation of the biterm within this recipe.

Therefore, this leads to inference system where terms can be “desynchronized”, the left and right side need not evaluate simultaneously according to the same rules.

Definition 20 : Let $\Sigma_{\text{decfail}} = \{\mathsf{enc}/2, \mathsf{dec}/2, \mathsf{fail}/0\}$ and $E^{\text{destr}}_{\text{decfail}}$ the equational theory generated by the two rules :

$$\mathsf{dec}(\mathsf{enc}(x, y), y) \to x$$
$$\mathsf{dec}(x, y) \to \mathsf{fail}$$

The equational theory, in this case, behaves as follows: if we apply one of these two rules, it must be the most precise i.e. $\mathsf{dec}(\mathsf{enc}(m, k)) \underset{E^{\text{destr}}_{\text{decfail}}}{\neq} \mathsf{fail}$.

Example. Consider $\phi = (m, k, k), \phi' = (m, k, k')$ and the instance $\phi, \phi', m, \mathsf{enc}(\mathsf{fail}, k)$ of the bideduction problem.

Notice we indeed have that $\phi, \phi' \rhd m, \mathsf{enc}(\mathsf{fail}, k)$ with the recipe $R = \mathsf{dec}(\mathsf{enc}(x_0, x_1), x_2)$ altough it is not of the form studied in the Section 2 since there is a sequence of destructor/constructor.

If we were to formalize the theory in an inference system, there would be four rules for decryption, one rule for each possible case of reduction, for instance there would be a rule:

$$m' \underset{E^{\text{destr}}_{\text{decfail}}}{\neq} \mathsf{enc}(\sqcup, k') \quad \frac{\mathsf{enc}(m, k), m' \qquad k, k'}{m, \mathsf{fail}}\ \text{decfailR}$$

And in this case we must allow to use a constructor rule on the main premise $\mathsf{enc}(m, k), m'$ to ensure completeness, henceforth the argument of the Section 2 cannot apply here.

Now that we know the previous argument cannot apply in this setting or in the more general setting of dec/enc without a destructor semantics, we study this latter case with a different approach.

It remains to see if in the destructor semantics signature and verification is decidable. A similar algorithm to the one studied in Section 2 may work since, even though there are two rules for a single destructor, the result of the two rules are constants.

4. Dec/Enc without destructors

4.1. General problem

4.1.1. Dec/Enc as an inference system

Trying to encode recipes as proof trees — similarly as in Section 3 — we will have four rules for decryption.

Definition 21 :

$$\frac{}{T_i, T_i'}\,(\text{input}) \qquad \frac{T, T' \qquad K, K'}{\mathsf{enc}(T, K), \mathsf{enc}(T', K')}\,(\text{enc})$$

$$\frac{\mathsf{enc}(T, K), \mathsf{enc}(T', K') \qquad K, K'}{T, T'}\,(\text{decLR}) \qquad T' \neq \mathsf{enc}(\sqcup, K')\ \frac{\mathsf{enc}(T, K), T' \qquad K, K'}{T, \mathsf{dec}(T', K')}\,(\text{decL})$$

$$T \neq \mathsf{enc}(\sqcup, K)\ \frac{T, \mathsf{enc}(T', K') \qquad K, K'}{\mathsf{dec}(T, K), T'}\,(\text{decR})$$

$$\begin{array}{l} T \neq \mathsf{enc}(\sqcup, K) \\ T' \neq \mathsf{enc}(\sqcup, K') \end{array}\ \frac{T, T' \qquad K, K'}{\mathsf{dec}(T, K), \mathsf{dec}(T', K')}\,(\text{dec})$$

Proposition 22 : $\mathcal{I}_{\phi,\phi'}^{E_{\text{enc}}}$ is sound and complete w.r.t. bideduction in E_{enc}.

Proof. Let's first show soundness.

- We show by a straightforward induction on proof trees that for all Π proof of T, T', we have $\phi, \phi' \rhd T, T'$.
- We show by induction on recipes that for all R such that $R[\phi] = T$ and $R[\phi'] = T'$ there is a proof Π of T, T' in $\mathcal{I}_{\phi,\phi'}^{E_{\text{enc}}}$.
  - cases $R = \square_i$ and $R = \mathsf{enc}(R_0, R_1)$ are straightforward by application of (input) and (enc).
  - If $R = \mathsf{dec}(R_0, R_1)$. We first apply the I.H. to R_0 and R_1. We therefore have Π_0 and Π_1 proving respectively $R_0[\phi]$ and $R_1[\phi]$. We discuss further subcases depending on the values of $R_0[\phi]$ and $R_1[\phi]$.
    - $R_0[\phi] \Downarrow \mathsf{enc}(M, K), R_1[\phi] \Downarrow K$ and $R_0[\phi'] \Downarrow \mathsf{enc}(M', K'), R_1[\phi'] \Downarrow K'$. We thus have $R[\phi] \Downarrow M$ and $R[\phi] \Downarrow M'$ and the following proof proves $R[\phi], R[\phi']$:

      $$\frac{\Pi_0 \qquad \Pi_1}{M, M'}\,(\text{decLR})$$

    - $R_0[\phi] \Downarrow \mathsf{enc}(M, K), R_1[\phi] \Downarrow K$ and $R_0[\phi'] \Downarrow U' \neq \mathsf{enc}(\sqcup, K'), R_1[\phi'] \Downarrow K'$. We thus have $R[\phi] \Downarrow M$ and $R[\phi] \Downarrow U'$ and the following proof proves $R[\phi], R[\phi']$:

      $$\frac{\Pi_0 \qquad \Pi_1}{M, \mathsf{dec}(U', K')}\,(\text{decL})$$

    - $R_0[\phi] \Downarrow U \neq \mathsf{enc}(\sqcup, K), R_1[\phi] \Downarrow K$ and $R_0[\phi'] \Downarrow \mathsf{enc}(M', K'), R_1[\phi'] \Downarrow K'$. We thus have $R[\phi] \Downarrow U$ and $R[\phi] \Downarrow M'$ and the following proof proves $R[\phi], R[\phi']$:

      $$\frac{\Pi_0 \qquad \Pi_1}{\mathsf{dec}(U, K), M'}\,(\text{decR})$$

    - $R_0[\phi] \Downarrow U \neq \mathsf{enc}(\sqcup, K), R_1[\phi] \Downarrow K$ and $R_0[\phi'] \Downarrow U' \neq \mathsf{enc}(\sqcup, K'), R_1[\phi'] \Downarrow K'$. We thus have $R[\phi] \Downarrow U$ and $R[\phi] \Downarrow U'$ and the following proof proves $R[\phi], R[\phi']$:

      $$\frac{\Pi_0 \qquad \Pi_1}{\mathsf{dec}(U, K), \mathsf{dec}(U', K')}\,(\text{dec})$$

Definition 23 : A proof Π in `inferenceEnc` is a *half-decomposition-composition proof* if the main premise of each occurence of rule (decLR) is proved by either (input) or (decLR).

It's called a "half-decomposition-composition" proof since it can contain a constructor rule proving the main premise of a destructor rule if it reduces only on one side i.e. for (decL) and (decR).

Lemma 24 : If T, T' is derivable in $\mathcal{I}_{\phi,\phi'}^{E_{\text{enc}}}$ then there is a half-decomposition-composition proof Π proving T, T'.

Proof. We proceed by induction on the size of proofs.

- $|\Pi| = 1$ then Π is simply an application of (input), and is already of the required form.
- $|\Pi| > 1$ we discuss the case of each possible last rule used in Π.
  - (decL), (decR), (dec), (enc): we apply the I.H. on the direct subproofs and immediatly get a proof of the required form.
  - (decLR): in this case, the last rule used to prove the main premise must be (enc) or (input) (by syntactic constraints).
    - If it's (input), after apply the I.H. to the other subproof, it's of the required form.
    - If it's (enc), then $\Pi = \dfrac{\dfrac{\boxed{\Pi_0}}{M, M'} \quad \dfrac{\boxed{\Pi_1}}{K, K'}\,(\text{enc})}{\dfrac{\mathsf{enc}(M, K), \mathsf{enc}(M', K') \qquad K, K'}{M, M'}\,(\text{decLR})}$ And we can apply the I.H. on Π_0 which is a smaller proof of M, M'.

There need not be a "full-decomposition-composition" for each provable biterm.

Example. If we have keys k, k', k'' pairwise distinct, this proof

$$\dfrac{\dfrac{\dfrac{}{m, m'}\,(\text{input}) \quad \dfrac{}{k, k''}\,(\text{input})}{\mathsf{enc}(m, k), \mathsf{enc}(m', k'')}\,(\text{enc}) \qquad \dfrac{}{k, k'}\,(\text{input})}{m, \mathsf{dec}(\mathsf{enc}(m', k''), k')}\,(\text{decL})$$

is not in full-decomposition-composition form, and we can't eliminate the encryption before the decryption rule.

4.1.2. Reduction

Taking account of (decL) and (decR) cause a problem when trying to apply a rule on a potential conclusion. Let's assume we want to use a similar algorithm to the one in the destructor case, since there need not be a proof in full-decomposition-composition form, (decL) and (decR) rules may be entangled with the constructor rule (enc).

A first idea would thus be to integrate (decL) and (decR) to the constructor rules in the algorithm, which searches proof from the conclusions. Now if we want to apply, say (decL) to a conclusion of the form $T, \mathsf{dec}(C', K')$ we have:

$$\dfrac{\mathsf{enc}(T, \boxed{?}), C' \qquad \boxed{?}, K'}{T, \mathsf{dec}(C', K')}\,(\text{decL})$$

Where $\boxed{?}$ might be any term, which we have to guess.

Thus we might think it's a better idea to apply all rules from the premises, but in this case we may apply infinitely many times a rule, making the term grow and because of target biterms of the form $\mathsf{dec}(\ldots(\mathsf{dec}(\mathsf{enc}(\ldots\mathsf{enc}(\ldots)))))$ that may or may not reduce it is hard to put a bound on those terms.

We formalize the need to guess a term in the following lemma, and reduce to a problem which we think is central in the study of dec/enc without destructors.

Lemma 25 : Let $n, m \in \mathcal{N}$, $\psi = T_0, ..., T_{n-1}$ terms that do not contain n or m, $\psi' = T'_0, ..., T'_{n-1}$ terms that do not contain n or m, K_0, K_1 terms that does not contain n or m.

Let $\phi = (n, T_0, ..., T_{n-1})$ and $\phi' = (m, T'_0, ..., T'_{n-1})$.

$\phi, \phi' \rhd \mathsf{dec}(\mathsf{enc}(n, K_0), K_1), m$ iff there is a term L such that $\psi, \psi' \rhd K_0, L$ and $\psi, \psi' \rhd K_1, L$ i.e. there are two recipes R_0, R_1 such that $R_0[\phi] = K_0 \underset{E_{\text{enc}}}{\neq} K_1 = R_1[\phi]$ and $R_0[\phi'] = R_1[\phi']$.

Proof.

- If there is such recipes R_0, R_1, it should be clear that $R = \mathsf{dec}(\mathsf{enc}(\square_0, R_0), R_1)$ proves $\phi, \phi' \rhd \mathsf{dec}(\mathsf{enc}(n, K_0), K_1), m$.
- Conversely let's examine a proof tree of this biterm. Let's first show by induction on recipes that if R is the smallest recipe such that $R[\phi] \Downarrow T == \mathsf{enc}(U, K)$ and n appears in T, then $R = \mathsf{enc}(R_0, R_1)$ for some recipes R_0, R_1.
  - If $R = \square_i$ and since n appears in $R[\phi] = \phi_i\downarrow$, by definition of ϕ, i must be equal to 0 but then $R[\phi] \Downarrow n$ which doesn't start with an encrpytion ✓.
  - If $R = \mathsf{enc}(R_0, R_1)$, it is straightforward.
  - If $R = \mathsf{dec}(R_0, R_1)$, we have two subcases: either
    - $R_0[\phi] \Downarrow \mathsf{enc}(M, K)$ and $R_1[\phi] \Downarrow K$, thus $R[\phi] \Downarrow M$. By I.H. on R_0, $R_0 = \mathsf{enc}(R_0^0, R_0^1)$ and $R_0^0[\phi] \Downarrow M == T$ and is smaller that R ✓.
    - Otherwise, $R[\phi] \Downarrow \mathsf{dec}(R_0[\phi]\downarrow, R_1[\phi]\downarrow) \not\!\!=\!\!= \mathsf{enc}(\sqcup, \sqcup)$ ✓.

  Let's now go back to the main proof. Assume we have R such that $R[\phi] = \mathsf{dec}(\mathsf{enc}(n, K_0), K_1)$ and $R[\phi'] = m$. Notice we must have $R = \mathsf{dec}(R_0, R_1)$, therefore, $R_0[\phi] \Downarrow \mathsf{enc}(n, K_0)$ so it must be of the form $\mathsf{enc}(R_2, R_3)$.

  We thus have $R = \mathsf{dec}(\mathsf{enc}(R_2, R_3), R_1)$.

  We prove that there are R'_0, R'_1 such that $R'_0[\phi] = K_0, R'_1[\phi] = K_1$ and $R'_0[\phi'] \underset{E_{\text{enc}}}{=} R'_0[\phi']$ by induction on the size of R.
  - $R_1[\phi'] \underset{E_{\text{enc}}}{\neq} R_3[\phi']$: in this case $R[\phi] \Downarrow \mathsf{dec}(\mathsf{enc}(R_2[\phi]\downarrow, R_3[\phi]\downarrow), R_1[\phi]\downarrow) \not\!\!=\!\!= m$ ✓.
  - $R_1[\phi] \underset{E_{\text{enc}}}{=} R_3[\phi]$ and $R_1[\phi'] \underset{E_{\text{enc}}}{=} R_3[\phi']$. In this case $R[\phi] \Downarrow R_2[\phi]$ and $R[\phi'] \Downarrow R_2[\phi']$ and we apply the I.H. to R_2 ✓.
  - $R_1[\phi] \underset{E_{\text{enc}}}{\neq} R_3[\phi]$ and $R_1[\phi'] \underset{E_{\text{enc}}}{=} R_3[\phi']$ then we have
    - $R[\phi] \Downarrow \mathsf{dec}(\mathsf{enc}(R_2[\phi]\downarrow, R_3[\phi]\downarrow), R_1[\phi]\downarrow) == \mathsf{dec}(\mathsf{enc}(n, K_0), K_1)$ thus $R_1[\phi] \Downarrow K_1$ and $R_3[\phi] \Downarrow K_0$; and
    - $R_1[\phi] \underset{E_{\text{enc}}}{=} R_3[\phi]$.

    Therefore $R'_0 = R_3$ and $R'_1 = R_1$ satisfy the required conditions ✓.

We now use the Lemma 25 to reduce to a problem similar to static equivalence.

Definition 26 :
Input : ϕ, ϕ' sequences of terms, T_0, T_1 term such that $T_0 \underset{E_{\text{enc}}}{\neq} T_1$.
Output : Is there recipes R_0, R_1 such that $R_0[\phi] \underset{E_{\text{enc}}}{=} T_0, R_1[\phi] \underset{E_{\text{enc}}}{=} T_1$ and $R_0[\phi'] \underset{E_{\text{enc}}}{=} R_1[\phi']$

Notice the resemblance between Definition 26 and the static equivalence problem. Therefore, by Lemma 25, two statically equivalent frames cannot lead to a term of the form $\mathsf{dec}(\mathsf{enc}(\sqcup, \sqcup), \sqcup)$ on the right and $\sqcup$ on the left: it must lead to synchronized recipes for bideduction.

Plus, we have that

- if $\phi \approx_s \phi'$, then $\phi, \phi' \not\rhd \mathsf{dec}(\mathsf{enc}(n, K_0), K_1), m$; and
- if $\phi, \phi' \rhd \mathsf{dec}(\mathsf{enc}(n, K_0), K_1), m$ then $\phi \not\approx_s \phi'$.

4.2. Restricting decryption to only atomic keys

We consider a restriction of the semantics of E_{enc} where we enforce keys to be atomic (either names or constants).

Definition 27 : Let $E_{\text{enc}}^{\text{atomic}}$ be the equational theory generated by the rewriting rule:

$$\mathsf{dec}(\mathsf{enc}(T, K), K) \to T \text{ if } K \text{ is a name or a constant}$$

Definition 28 : We denote by $E_{\text{enc}}^{\text{atomic strict}}$ the theory where terms cannot contain nonatomic terms in key position, formally:
- $n \in \mathcal{N}$ is a term;
- if T is a term and $n \in \mathcal{N}$, then $\mathsf{enc}(T, n)$ is a term;
- if T is a term and $n \in \mathcal{N}$, then $\mathsf{dec}(T, n)$ is a term.

Definition 29 : Let $\mathcal{I}_{\phi,\phi'}^{E_{\text{enc}}^{\text{atomic}}}$ be the following inference system.

$$\frac{}{T_i, T_i'}\,(\text{input}) \qquad \frac{T, T' \qquad K, K'}{\mathsf{enc}(T, K), \mathsf{enc}(T', K')}\,(\text{enc})$$

$$\frac{\mathsf{enc}(T, K), \mathsf{enc}(T', K') \qquad K, K'}{T, T'}\,(\text{decLR})$$

$$T' \neq \mathsf{enc}(\sqcup, K'), K' \text{ atomic} \quad \frac{\mathsf{enc}(T, K), T' \qquad K, K'}{T, \mathsf{dec}(T', K')}\,(\text{decL})$$

$$T \neq \mathsf{enc}(\sqcup, K), K \text{ atomic} \quad \frac{T, \mathsf{enc}(T', K') \qquad K, K'}{\mathsf{dec}(T, K), T'}\,(\text{decR})$$

$$\begin{matrix} T \neq \mathsf{enc}(\sqcup, K) \\ T' \neq \mathsf{enc}(\sqcup, K') \end{matrix} \quad \frac{T, T' \qquad K, K'}{\mathsf{dec}(T, K), \mathsf{dec}(T', K')}\,(\text{dec})$$

We now define the inference system corresponding to $E_{\text{enc}}^{\text{atomic strict}}$. Since the atomicity restriction was made on the construction on terms, there will be no more conditions on the inference rules, unlike $\mathcal{I}_{\phi,\phi'}^{E_{\text{enc}}^{\text{atomic}}}$.

Definition 30 : Let $\mathcal{I}_{\phi,\phi'}^{E_{\text{enc}}^{\text{atomic strict}}}$ be the following inference system.

$$\frac{}{T_i, T_i'}\,(\text{input}) \qquad \frac{T, T' \qquad K, K'}{\mathsf{enc}(T,K), \mathsf{enc}(T',K')}\,(\text{enc})$$

$$\frac{\mathsf{enc}(T,K), \mathsf{enc}(T',K') \qquad K, K'}{T, T'}\,(\text{decLR}) \qquad T' \neq \mathsf{enc}(\sqcup, K')\ \frac{\mathsf{enc}(T,K), T' \qquad K, K'}{T, \mathsf{dec}(T',K')}\,(\text{decL})$$

$$T \neq \mathsf{enc}(\sqcup, K)\ \frac{T, \mathsf{enc}(T',K') \qquad K, K'}{\mathsf{dec}(T,K), T'}\,(\text{decR})$$

$$\begin{matrix} T \neq \mathsf{enc}(\sqcup, K) \\ T' \neq \mathsf{enc}(\sqcup, K') \end{matrix}\ \frac{T, T' \qquad K, K'}{\mathsf{dec}(T,K), \mathsf{dec}(T',K')}\,(\text{dec})$$

Proposition 31 : $\mathcal{I}_{\phi,\phi'}^{E_{\text{enc}}^{\text{atomic}}}$ (resp. $\mathcal{I}_{\phi,\phi'}^{E_{\text{enc}}^{\text{atomic strict}}}$) is sound and complete w.r.t. bideduction in $E_{\text{enc}}^{\text{atomic}}$ (resp. $E_{\text{enc}}^{\text{atomic strict}}$).

Proof. Similar to Proposition 22.

Proposition 32 : For $\mathcal{I}_{\phi,\phi'}^{E_{\text{enc}}^{\text{atomic}}}$ and $\mathcal{I}_{\phi,\phi'}^{E_{\text{enc}}^{\text{atomic strict}}}$, proofs can be rewritten in half-decomposition-composition form.

Proof. Same as Lemma 24.

4.2.1. Atomicity

First considering the more general theory $E_{\text{enc}}^{\text{atomic}}$, we may have a similar example as with E_{enc}: we adapt Lemma 25.

Lemma 33 : Let $n, m \in \mathcal{N}$, $\psi = T_0, ..., T_{n-1}$ terms that do not contain n or m, $\psi' = T_0', ..., T_{n-1}'$ terms that do not contain n or m, K_0, K_1 terms that does not contain n or m.

Let $\phi = (n, T_0, ..., T_{n-1})$ and $\phi' = (m, T_0', ..., T_{n-1}')$.

$\phi, \phi' \rhd \mathsf{dec}(\mathsf{enc}(n, K_0), K_1), m$ iff there is a name k' such that $\psi, \psi' \rhd K_0, k'$ and $\psi, \psi' \rhd K_1, k'$ i.e. there are two recipes R_0, R_1 such that $R_0[\phi] = K_0 \underset{E_{\text{enc}}}{\neq} K_1 = R_1[\phi]$, $R_0[\phi'] = R_1[\phi']$ and $R_0[\phi]$ is atomic.

Proof. Similar to Lemma 25.

Even though it's more restrictive than the condition in Lemma 25, it still raises a problem when trying to apply a rule from the conclusions, it does not simplify the problem that much. This variant may be decidable whereas the variant of the general dec/enc is not, but we don't have any results regarding these now.

We therefore focus on strict atomicity, in which this problem will vanish and guessing keys when applying (decL) or (decR) will become more straightforward.

4.2.2. Strict atomicity

Proposition 34 : Let Π be a proof in $\mathcal{I}_{\phi,\phi'}^{E_{\text{enc}}^{\text{atomic strict}}}$ of U, U' two term *that don't contain any* dec symbol. Then Π does not contain rules (decR), (decL) nor (dec).

Proof. We proceed by induction on Π.

- $\dfrac{}{T_i, T_i'}\,(\text{input})$ is already of the required form ✓.
- $\dfrac{\boxed{\Pi_0} \qquad \boxed{\Pi_1}}{\mathsf{enc}(T,K), \mathsf{enc}(T',K')}\,(\text{enc})$ notice the subterms T, K, T', K' don't contain any dec since $U = \mathsf{enc}(T,K)$ and U' $\mathsf{enc}(T',K')$ don't. We apply the I.H. on the direct subproofs, and get a proof of the required form ✓.
- $\dfrac{\dfrac{\boxed{\Pi_0}}{\mathsf{enc}(T,K), \mathsf{enc}(T',K')} \qquad \dfrac{\boxed{\Pi_1}}{K, K'}\,(\text{decLR})}{T, T'}$ similar to previous case.
- $\dfrac{\dfrac{\boxed{\Pi_0}}{\mathsf{enc}(T,K), T'} \qquad \dfrac{\boxed{\Pi_1}}{K, K'}\,(\text{decL})'}{T, \mathsf{dec}(T',K')}$
  $\dfrac{\dfrac{\boxed{\Pi_0}}{T, \mathsf{enc}(T',K')} \qquad \dfrac{\boxed{\Pi_1}}{K, K'}\,(\text{decR})'}{\mathsf{dec}(T,K), T'} \quad \dfrac{\dfrac{\boxed{\Pi_0}}{T, T'} \qquad \dfrac{\boxed{\Pi_1}}{K, K'}\,(\text{dec})}{\mathsf{dec}(T,K), \mathsf{dec}(T',K')}$ in all those cases, we have $U = \mathsf{dec}(T,K)$ ot $U' = \mathsf{dec}(T,K)$ whereas we assumed that neither U nor U' contain any dec ✓.

In other words, when trying to prove that a biterm U, U' is bideducible and that those terms do not contain a dec symbol, then we can work as in the destructor/constructor model: the proof are synchronized on both sides.

As was already remarked with terms of the form $\mathsf{dec}(\mathsf{enc}(n,k),k),n'$, it suffices that one of the two terms contain a dec to lead to a unsynchronized proof.

Corollary 35 : The restriction of the bideduction problem in $E_{\text{enc}}^{\text{atomic}}$ (but also in E_{enc}, notice we didn't use the atomicity assumption in the proof of Proposition 34) where the target biterm do not contain any dec symbol is decidable.

Proof. We can restrict the search of proof to the system with only (decLR), (enc) and (input), which corresponds to a destructor/constructor proof system.

Algorithm 36 :

```
BideduceEncAtomDecLR (φ, φ') :
Φ := zip(φ, φ')
Until Φ doesn't change :
    For each (enc(M, k), enc(M', k')) ∈ Φ :
        If (M, M') ∉ Φ :
            If (k, k') ∈ Φ :
                Φ := Φ ⊔ {(M, M')}
Return Φ

bideduceEncAtomAllKeysL (Φ, k) :
keys' := ∅
For All (T, T') ∈ Φ:
    If T == k&&T' is atomic:
        keys' ∪ {T'}
Return keys'
```

```
BideduceEncAtomAllKeysR (Φ, k') :
keys := ∅
For All (T, T') ∈ Φ:
  If T' == k'&&T is atomic:
    keys ∪ {T}
Return keys

BideduceEncAtomConstr(Φ, T, T') :
If (T, T') ∈ Φ :
  Return true
Else
  If T, T' == n, n' :
    Return BideduceEncAtomKeysFrom(Φ, n, n')
  If T, T' == enc(M, k), enc(M', k') :
    Return BideduceEncAtomConstr(Φ, M, M')&& BideduceEncAtomConstr(Φ, k, k')
  If T == dec(M, k), :
    keys' := bideduceEncAtomAllKeysL(Φ, k)
      For All k' ∈ keys':
        If BideduceEncAtomConstr(Φ, M, enc(T', k')):
          Return true
  If T' == dec(M', k') :
    keys := bideduceEncAtomAllKeysR(Φ, k')
      For All k ∈ keys:
        If BideduceEncAtomConstr(Φ, enc(T, k), M'):
          Return true
  If T, T' == dec(M, k), dec(M', k') :
    If BideduceEncAtomConstr(Φ, M, M')&& BideduceEncAtomConstr(Φ, k, k'):
      Return true
  Return false

BideduceEncAtom(φ, φ', T, T') :
bideducibleDestr := BideduceEncAtomDecLR(φ, φ')
Return BideduceEncAtomConstr(bideducibleDestr, T, T')
```

Lemma 37 : Let Π be a proof tree in half-decomposition-composition form of $\mathcal{I}^{E^{\text{atomic strict}}_{\text{enc}}}_{\phi,\phi'}$. If the last rule of Π is (decLR), then there is a proof Π' of the same conclusion such that all rules in Π' are (decLR) or (input).

Proof. Let's proceed by induction on Π.

- If Π is simply an application of (input), the tree is already of the required form.
- $$\dfrac{\dfrac{\triangledown\Pi_0\triangle}{\mathsf{enc}(T,k),\mathsf{enc}(T',k')} \qquad \dfrac{\triangledown\Pi_1\triangle}{k,k'}}{T,T'}\,(\text{decLR})^{\cdot}$$
  - First, Π_0 is also in half-decomposition-composition form by definition, so we can apply the I.H. to Π_0 and obtain Π'_0 proving $\mathsf{enc}(T,k),\mathsf{enc}(T',k')$ of the required form.
  - Π_1 proves k,k' that do not contain any dec symbol, thus, by Proposition 34, there is a proof $\widetilde{\Pi_1}$ that contain only rule (input), (decLR) and (enc). Thus, $\widetilde{\Pi_1}$ is basically a proof in a destructor/constructor proof system, so we can rewrite it into $\widetilde{\Pi'_1}$ in decomposition-composition form (which is *a fortiori* in half-decomposition-composition form), and since k,k' is not of the form $\mathsf{enc}(\sqcup,\sqcup),\mathsf{enc}(\sqcup,\sqcup)$, the last rule of $\widetilde{\Pi'_1}$ must be (decLR), so we can also apply the I.H. to $\widetilde{\Pi'_1}$ to obtain Π'_1 of the required form.

Putting everything together, the proof $\Pi' = \dfrac{\dfrac{\boxed{\Pi'_0}}{\mathsf{enc}(T,k),\mathsf{enc}(T',k')} \qquad \dfrac{\boxed{\Pi'_1}}{k,k'}}{T,T'}\,(\text{decLR})$. is of the required form.

Definition 38 : Let

$$\text{DestrBideducible}_{\phi,\phi'} = \left\{(T,T') \mid T,T' \text{ is derivable in } \mathcal{I}^{E^{\text{atomic strict}}_{\text{enc}}}_{\phi,\phi'} \text{ with only (decLR) and (input)}\right\}$$

Lemma 39 : The Algorithm 36 has the following specifications:

1. BideduceEncAtomDecLR(ϕ,ϕ') returns the set of biterm provable in $\mathcal{I}^{E^{\text{atomic strict}}_{\text{enc}}}_{\phi,\phi'}$ using only rule (decLR).
2. If $\Phi = \text{DestrBideducible}_{\phi,\phi'}$, then we have that bideduceEncAtomAllKeysL(Φ,k) returns the set of names n' such that k,n' is provable in $\mathcal{I}^{E^{\text{atomic strict}}_{\text{enc}}}_{\phi,\phi'}$.
3. If $\Phi = \text{DestrBideducible}_{\phi,\phi'}$, then we have that bideduceEncAtomAllKeysR(Φ,k') returns the set of names n such that n,k' is provable in $\mathcal{I}^{E^{\text{atomic strict}}_{\text{enc}}}_{\phi,\phi'}$.
4. If $\Phi = \text{DestrBideducible}_{\phi,\phi'}$, then we have that BideduceEncAtomConstr(Φ,T,T') returns `true` iff T,T' is provable in $\mathcal{I}^{E^{\text{atomic strict}}_{\text{enc}}}_{\phi,\phi'}$.

Proof.

1. BideduceEncAtomDecLR(ϕ,ϕ') saturates $\mathtt{zip}(\phi,\phi')$ with applications of (decLR)
   - Clearly, all biterm in the returned set is in DestrBideducible ✓.
   - Conversely, by Lemma 37 we only need to apply (decLR) to the hypothesis from the frames ϕ,ϕ', and the height of each proof tree proving a biterm of DestrBideducible being finite, BideduceEncAtomDecLR will find each provable biterm eventually. It can be proven properly by an induction on the height of proof trees ✓.
2. Let's show the specification of the returned set.
   - Clearly, if $(k,n') \in \Phi = \text{DestrBideducible}_{\phi,\phi'}$, then k,n' is provable in $\mathcal{I}^{E^{\text{atomic strict}}_{\text{enc}}}_{\phi,\phi'}$ ✓.
   - Let k,n' be provable in $\mathcal{I}^{E^{\text{atomic strict}}_{\text{enc}}}_{\phi,\phi'}$. By Proposition 34, since k,n' do not contain any dec symbol, there is a proof of k,n' in $\mathcal{I}^{E^{\text{atomic strict}}_{\text{enc}}}_{\phi,\phi'}$ with only rules (decLR), (enc), (input). Therefore this proof is a proof in a destructor/constructor system, so it can be rewritten in decomposition-composition form, which is *a fortiori* in half-decomposition-composition form. This proof must end with a (decLR) since the toplevel constructors of k and n' is not enc, thus by Lemma 37, k,n' is provable using only (decLR) and (input) thus $(k,n') \in \text{DestrBideducible}_{\phi,\phi'} = \Phi$ (this is the same argument as in the proof of Lemma 37) ✓.
3. Similar to the case of bideduceEncAtomAllKeysL ✓.
4. Notice each condition correspond to testing the application of a rule: l.6-7 corresponds to (enc), l.8-12 to (decR), l.13-17 to (decL) and l.18-20 to (dec). The **Return** statement l.2 corresponds to the application of a proof of a biterm in DestrBideducible.
   - With this remark, it should be clear that if BideduceEncAtomConstr$\big(\text{DestrBideducible}_{\phi,\phi'},T,T'\big)$ returns `true` then T,T' is provable in $\mathcal{I}^{E^{\text{atomic strict}}_{\text{enc}}}_{\phi,\phi'}$ ✓.
   - Assuming there is a proof Π (w.l.o.g. in half-decomposition-composition form) of T,T' in $\mathcal{I}^{E^{\text{atomic strict}}_{\text{enc}}}_{\phi,\phi'}$. We proceed by induction on Π.
     - $\dfrac{}{T,T'}\,(\text{input})$ then by definition $(T,T') \in \mathtt{zip}(\phi,\phi') \subseteq \text{DestrBideducible}_{\phi,\phi'}$, thus the algorithm returns `true` on line 2 ✓.

- $\dfrac{\overset{\Pi_0}{\vdots}}{\dfrac{\mathsf{enc}(T,k),\mathsf{enc}(T',k')}{T,T'}}\ \dfrac{\overset{\Pi_1}{\vdots}}{k,k'}$ (decLR). In this case, by Lemma 37, there is a Π' containing only (decLR) and (input), thus $(T,T') \in \text{DestrBideducible}_{\phi,\phi'}$ and the algorithm returns `true` on line 2 ✓.
- $\dfrac{\overset{\Pi_0}{\vdots}}{\dfrac{M,M'}{\mathsf{enc}(M,k),\mathsf{enc}(M',k')}}\ \dfrac{\overset{\Pi_1}{\vdots}}{k,k'}$ (enc). In this case we'll reach l.7 and the recursive calls BideduceEncAtomConstr(Φ,T,T'), BideduceEncAtomConstr(Φ,k,k') will return `true` by I.H. on Π_0 and Π_1 ✓.
- $\dfrac{\overset{\Pi_0}{\vdots}}{\dfrac{M,\mathsf{enc}(T',k')}{\mathsf{dec}(M,k),T'}}\ \dfrac{\overset{\Pi_1}{\vdots}}{k,k'}$ (decR). By syntaxic constraints on $T == \mathsf{dec}(M,k)$, we'll reach the condition l.8 and go through to the "then" branch. Since Π_1 proves k,k', by the specification of bideduceEncAtomAllKeysL, $k' \in$ keys' after l.9. Therefore,
  - either we'll reach the iteration of the **For All** loop with k', in which case by I.H. on Π_0, the recursive call BideduceEncAtomConstr$(\Phi,M,\mathsf{enc}(T',k'))$ will return `true` and the algorithm will return `true` ✓ or
  - the algorithm returns `true` at a previous iteration of the loop ✓.
- $\dfrac{\overset{\Pi_0}{\vdots}}{\dfrac{\mathsf{enc}(M,k),M'}{T,\mathsf{dec}(M',k')}}\ \dfrac{\overset{\Pi_1}{\vdots}}{k,k'}$ (decL). Similar to the case of (decL) ✓.
- $\dfrac{\overset{\Pi_0}{\vdots}}{\dfrac{M,M'}{\mathsf{dec}(M,k),\mathsf{dec}(M',k')}}\ \dfrac{\overset{\Pi_1}{\vdots}}{k,k'}$ (dec). In this case either the algorithm returns true before reaching the condition l. 18 or we'll reach l.19 and the recursive call BideduceEncAtomConstr(Φ,M,M'), BideduceEncAtomConstr(Φ,k,k') will return `true` by I.H. on Π_0 and Π_1 ✓.

Lemma 40 : The algorithms (Algorithm 36) terminate.

Proof.

1. BideduceEncAtomDecLR terminates since it saturate Φ, which grows at each iteration, and $\Phi \subseteq \text{DestrBideducible}_{\phi,\phi'}$ which is finite.
2. bideduceEncAtomAllKeysL and bideduceEncAtomAllKeysR terminate since it's only a **For** loop over a finite set.
3. BideduceEncAtomConstr terminates since it only explore the AST of T,T' simultaneously.
4. BideduceEncAtom terminates since the former terminate and it only call those.

Theorem 41 : Bideduction in $E_{\text{enc}}^{\text{atomic strict}}$ is decidable.

5. Pair/Projections

In light of the previous section, we try to show a decidability result on an equational theory outside the destructor semantics. Since with enc/dec the problem is that keys can be "hidden" behind a $\mathsf{dec}(\mathsf{enc}(\sqcup,\sqcup),\sqcup)$, we consider a more straightforward theory: pairs and projections.

Definition 42 : Let $\Sigma_{\text{pair}} = \{\langle \sqcup,\sqcup\rangle/2, \mathsf{fst}/1, \mathsf{snd}/1\}$ and E_{pair} the theory generated by the following rewriting rules:

- $\mathsf{fst}(\langle x,y\rangle) \to x$; and
- $\mathsf{snd}(\langle x,y\rangle) \to y$.

Definition 43 : Let $\mathcal{I}^{E_{\text{pair}}}_{\phi,\phi'}$ be the following inference system:

$$(T_i, T_i') \in \mathsf{zip}(\phi,\phi') \frac{}{T_i, T_i'}\,(\text{input})$$

$$\frac{\langle T_0,T_1\rangle, \langle T_0',T_1'\rangle}{T_0, T_0'}\,(\text{fstLR}) \qquad \frac{\langle T_0,T_1\rangle, \langle T_0',T_1'\rangle}{T_1, T_1'}\,(\text{sndLR})$$

$$T' \neq \langle \sqcup,\sqcup\rangle \frac{\langle T_0,T_1\rangle, T'}{T_0, \mathsf{fst}(T')}\,(\text{fstL}) \qquad T \neq \langle \sqcup,\sqcup\rangle \frac{T, \langle T_0',T_1'\rangle}{\mathsf{fst}(T), T_0'}\,(\text{fstR})$$

$$T' \neq \langle \sqcup,\sqcup\rangle \frac{\langle T_0,T_1\rangle, T'}{T_1, \mathsf{snd}(T')}\,(\text{sndL}) \qquad T \neq \langle \sqcup,\sqcup\rangle \frac{T, \langle T_0',T_1'\rangle}{\mathsf{snd}(T), T_1'}\,(\text{sndR})$$

$$\begin{matrix} T \neq \langle \sqcup,\sqcup\rangle \\ T' \neq \langle \sqcup,\sqcup\rangle \end{matrix} \frac{T, T'}{\mathsf{fst}(T), \mathsf{fst}(T')}\,(\text{fst}) \qquad \begin{matrix} T \neq \langle \sqcup,\sqcup\rangle \\ T' \neq \langle \sqcup,\sqcup\rangle \end{matrix} \frac{T, T'}{\mathsf{snd}(T), \mathsf{snd}(T')}\,(\text{snd})$$

$$\frac{T_0, T_0' \qquad T_1, T_1'}{\langle T_0,T_1\rangle, \langle T_0',T_1'\rangle}\,(\text{pair})$$

Proposition 44 : $\mathcal{I}^{E_{\text{pair}}}_{\phi,\phi'}$ is sound and complete w.r.t. bideduction.

Proof. Similar to Proposition 22.

Definition 45 : A proof in $\mathcal{I}^{E_{\text{pair}}}_{\phi,\phi'}$ is in full-decomposition-composition form if:

- eah premise of a rule (fst), (snd) is *not* proven by a (pair);
- each premise of a rule (fstL), (sndL) (resp. (fstR) or (sndR)) is proven by (fstLR), (sndLR), (input), (fstL) or (sndL), (resp. (fstR) or (sndR)); and
- each premise of (fstLR) or (sndLR) is proven by (input), (fstLR) or (sndLR).

Proposition 46 : If T, T' is derivable in $\mathcal{I}^{E_{\text{pair}}}_{\phi,\phi'}$ then there is a proof Π in full-decomposition-composition form proving T, T'.

Proof. Let's show by induction on proof trees that for all Π' proof tree of T, T', we have a proof in full-decomposition-composition form of T, T'.

- (input), (pair): straightforward by applying the I.H. on the subproofs✓.
- (fst): we apply the I.H. to the direct subproof, by syntaxic constraint the last rule of the resulting proof cannot be (pair).
- (fstL): We have $\Pi = \dfrac{\begin{matrix} \\ \boxed{\Pi'} \end{matrix}}{T' \neq \langle \sqcup,\sqcup\rangle \dfrac{\langle T_0,T_1\rangle, T'}{T_0, \mathsf{fst}(T')}\,(\text{fstL})}$. We apply the I.H. on Π'. Since $T' \neq \langle \sqcup,\sqcup\rangle$, the last rule of Π' is not pair. Plus since the other term in the premise is a pair, the last rule of Π' is not (fstR) or (sndR). Therefore, it must be (input), (fstLR), (sndLR), (fstL) or (sndL) ✓.
- (fstR) is similar to the (fstL) case.
- (fstLR): we examine the last rule proving the premise.

- (input), (fstLR), (sndLR): straightforward by applying the I.H. on the subproof, if any✓.
- (fstL), (fstR), (sndL), (sndR), (fst), (snd): the conclusion of one of those rules must contain fst or snd at toplevel (at least on one of the two sides), therefore the toplevel constructor is not a pair in one of the two sides, whereas the premise of (fstLR) must be a biterm of two pairs ✓.
- (pair): In this case, the proof tree is $\dfrac{\dfrac{\boxed{\Pi'_0} \quad \boxed{\Pi'_1}}{T_0, T'_0 \qquad T_1, T'_1}\,(\text{pair})}{\dfrac{\langle T_0,T_1\rangle, \langle T'_0,T'_1\rangle}{T_0, T'_0}\,(\text{fstLR})}$ and we can apply the I.H. on Π'_0 ✓.

- (snd), (sndL), (sndR) and (sndLR) cases are similar to the corresponding rules for fst.

We therefore have three phases in a bideduction proof (from the leafs):
1. Application of destructors rules reducing on both sides (fstLR), (sndLR);
2. Application of destructors rules reducing on one (fstL), (fstR), (sndL), (sndR);
3. Application of destructors rules reducing no side (fst), (snd);
4. Application of constructor rules (pair).

We now prove a lemma that will help during the search of proofs from the conclusion. First notice that when applying a destructor rule that reduces only on one side e.g. (fstL) on a conclusion of the form, say $T, \mathsf{fst}(T')$, we have to guess a term T_1 such that the premise of the rule is $\langle T,T_1\rangle, T'$. The following lemma says we can restrict the search of such a T_1: we can consider only terms T_1 such that $\langle T,T_1\rangle$ appear as a subterm of one of the hypotheses.

Lemma 47 : Let Π be a proof of T, T' in full-decomposition-composition form in $\mathcal{I}_{\phi,\phi'}^{E_{\text{pair}}}$. If the last rule used in Π is (fstL), (sndL), (resp. (fstR) or (sndR)), then there is U, U' a biterm appearing in a node of Π, which is:
- proved by (fstLR), (sndLR) or (input); and
- is a premise of (fstL) or (sndL) (resp. (fstR) or (sndR));

such that:
- if the last rule in Π is (fstL), there is a term T_1 such that $\langle T,T_1\rangle \in \mathsf{st}(U)$;
- if it is (sndL), there is a term T_0 such that $\langle T_0,T\rangle \in \mathsf{st}(U)$;
- if it is (fstR), there is a term T'_1 such that $\langle T',T'_1\rangle \in \mathsf{st}(U')$;
- if it is (sndR), there is a term T'_0 such that $\langle T'_0,T'\rangle \in \mathsf{st}(U')$.

Proof. We prove this result by induction on Π.
- If Π is of the form $\dfrac{\boxed{\Pi'}}{\langle T,T_1\rangle, T'}\,(\text{fstL})$. We discuss each possible last rule of Π' — which by Definition 45 is either (fstL), (sndL), (fstLR), (sndLR) or (input):
  - (input), (fstLR), (sndLR): $U, U' = \langle T,T_1\rangle, T'$ and T satisfy the conditions.
  - (fstL): by I.H. applied to the direct subproof, we have a biterm U, U' appearing in the required position in Π such that there is T_2 such that $\langle\langle T,T_1\rangle,T_2\rangle \in \mathsf{st}(U)$. Hence U, U' and T_1 satisfy the conditions.
  - (sndL): similar.
- The other cases (last rule in Π is (sndL), (sndR), (fstR)) are similar.

Algorithm 48 :

```
BideducePairProjLR(φ, φ′):
Φ := zip(φ, φ′)
Until Φ doesn't change:
  For (T, T′) ∈ Φ:
    If T == ⟨T₀,T₁⟩ and T′ == ⟨T₀′,T₁′⟩:
      If (T₀, T₀′) ∉ Φ && (T₁, T₁′) ∉ Φ:
        Φ := Φ ⊔ {(T₀, T₀′), (T₁, T₁′)}
Return Φ

BideducePairProjL(Φ, T, T′):
If (T, T′) ∈ Φ:
  Return true
Match T′:
  fst(T₀′) →
    For (U, U′) ∈ Φ:
      For S ∈ st(U) :
        If S is of the form ⟨T,T₁⟩ for some T₁:
          If BideducePairProjL(Φ, ⟨T,T₁⟩, T₀′):
            Return true
  snd(T₀′) →
    For (U, U′) ∈ Φ:
      For S ∈ st(U) :
        If S is of the form ⟨T₀,T⟩ for some T₀:
          If BideducePairProjL(Φ, ⟨T₀,T⟩, T₀′):
            Return true
  _ → Return false
Return false

BideducePairProjR(Φ, T, T′):
If (T, T′) ∈ Φ:
  Return true
Match T:
  fst(T₀) →
    For (U, U′) ∈ Φ:
      For S′ ∈ st(U′) :
        If S′ is of the form ⟨T′,T₁′⟩ for some T₁′:
          If BideducePairProjR(Φ, T₀, ⟨T′,T₁′⟩):
            Return true
  snd(T₀) →
    For (U, U′) ∈ Φ:
      For S′ ∈ st(U′) :
        If S′ is of the form ⟨T₀′,T′⟩ for some T₀′:
          If BideducePairProjR(Φ, T₀, ⟨T₀′,T′⟩):
            Return true
  _ → Return false
Return false

BideducePairProjLxorR(Φ, T, T′):
Return BideducePairProjL(Φ, T, T′) || BideducePairProjR(Φ, T, T′)
```

BideducePairProj(Φ, T, T'):
1 **If** BideducePairProjLxorR(Φ, T, T') :
2 | **Return** `true`
3 **Else If** $T ==$ fst(T_0) and $T' ==$ fst(T'_0):
4 | **Return** BideducePairProj(Φ, T_0, T'_0)
5 **Else If** $T ==$ snd(T_1) and $T' ==$ snd(T'_1):
6 | **Return** BideducePairProj(Φ, T_1, T'_1)

BideducePairPair(Φ, T, T'):
1 **If** BideducePairProj(Φ, T, T') :
2 | **Return** `true`
3 **Else If** $T == \langle T_0,T_1\rangle$ and $T' == \langle T'_0,T'_1\rangle$:
4 | **Return** BideducePairPair(Φ, T_0, T'_0) && BideducePairPair(Φ, T_1, T'_1)

BideducePair(ϕ, ϕ', T, T'):
1 Φ := BideducePairProjLR(ϕ, ϕ')
2 **Return** BideducePairPair(Φ, T, T')

Lemma 49 : In the specifications below, we assume that Φ is a set of provable biterms in $\mathcal{I}_{\phi,\phi'}^{E_{\text{pair}}}$ using only rules (input), (fstLR), (sndLR). We show the specification of each algorithm:

1. BideducePairProjLR(ϕ, ϕ') returns the set of biterms provable in $\mathcal{I}_{\phi,\phi'}^{E_{\text{pair}}}$ using only (input), (fstLR) and (sndLR).
2. BideducePairProjL(Φ, T, T') returns `true` iff there is an incomplete proof Π of T, T' using only (sndL) and (fstL) rules, and whose leafs are in Φ.
3. BideducePairProjR(Φ, T, T') returns `true` iff there is an incomplete proof Π of T, T' using only (sndR) and (fstR) rules, and whose leafs are in Φ.
4. BideducePairProjLxorR(Φ, T, T') returns `true` iff there is an incomplete proof Π of T, T' using only (sndL), (fstL), (sndR) and (fstR) rules, and whose leafs are in Φ.
5. BideducePairProj(Φ, T, T') returns `true` iff there is an incomplete proof Π of T, T' in full-decomposition-composition form, that uses neither (pair) nor (fstLR), (sndLR), and whose leafs are in Φ.
6. BideducePairPair(Φ, T, T') returns `true` iff there is an incomplete proof Π of T, T' in full-decomposition-composition form, that doesn't use (fstLR), (sndLR) nor (input) and whose leafs are in Φ.

Proof. We prove each specification sequentially from (1) to (6) (the proof of one of these specifications may rely on the previous specification, in that order).

1. We have two directions to prove: that each biterm in the final Φ is indeed provable this way, and that each biterm provable this way is in Φ.
   - Clearly, if $(\langle T_0,T_1\rangle, \langle T'_0,T'_1\rangle)$ is provable, then (T_0, T'_0) and (T_1, T'_1) too by an application of (fstLR) (resp. (sndLR)) ✓.
   - Conversely, if T, T' is probable with only (input), (fstLr) and (sndLR) rules, there is a finite proof tree using those rules. We can show by a straightforward induction that at iteration i of the loop, Φ contains all biterms provable with a tree of size i, which concludes the proof ✓.
2. - First, by the specification (4), and since recursive calls in this function corresponds to the application of (fstL) or (sndL), if BideducePairProjL(Φ, T, T') returns `true`, then Φ, T, T' satisfy the specification ✓.
   - Assuming Φ, T, T' satisfy the specification, we proceed by induciton on Π the incomplete proof of T, T'.

- $\Pi = T, T'$ without any rule applied, then T, T' is a leaf of Π thus by the specification $T, T' \in \Phi$ and BideducePairProjL(Φ, T, T') terminates after the first condition and returns `true`.
- $\Pi = \dfrac{\boxed{\Pi'}}{T, \mathsf{snd}(T')}$ (sndL). We want to apply the I.H. to the direct subproof, so we need to prove that the corresponding recursive call will be reached. We thus start by showing the condition l.13 of BideducePairProjL is met.

  Since this incomplete proof can be completed with proofs of biterms in Φ by assumption, we can apply Lemma 47 to T, T' and the completed proof. Hence, there is a term T_0 and U, U' appearing between a (fstLR), (sndLR) or (input) and a (fstL) or (sndL) such that $\langle T_0, T\rangle \in \mathsf{st}(U)$. By definition of Π and Φ, U, U' must appear as a leaf in Π, thus $U, U' \in \Phi$ and the condition must be met when the loop l.11 considers (U, U').

  Hence, by the I.H. applied to the recursive call BideducePairProjL$(\Phi, \langle T_0, T\rangle, T')$ and Π', this recursive call returns `true` so the initial call must also return `true` ✓.

3. Similar to BideducePairProjL ✓.
4. • First, by the specification (2) and (3), if BideducePairProjLxorR(Φ, T, T') returns `true`, then Φ, T, T' satisfy the specification ✓.
   • Now, assume we have Π an incomplete proof in full-decomposition-composition form proving T, T' using only (fstL), (fstR), (sndL), (sndR). Consider the last rule in Π. If it is (fstL) or (sndL) (resp. (fstR) or (sndR)), then by Definition 45, each rule used in Π is (fstL) or (sndL) (resp. (fstR) or (sndR)), therefore either BideducePairProjL(Φ, T, T') or BideducePairProjR(Φ, T, T') returns `true`, and BideducePairProjLxorR(Φ, T, T') returns `true` ✓.
5. • First, by the specification (4), and since recursive calls in this function corresponds to the application of (fst) or (snd), if BideducePairProj(Φ, T, T') returns `true`, then Φ, T, T' satisfy the specification ✓.
   • Assuming Φ, T, T' satisfy the specification, let Π be an incomplete proof of T, T' in full-decomposition-composition form of T, T' that does not use (pair), (fstLR), nor (sndLR) and whose leaf are in Φ. We prove the result by induction on Π:
   - If it is (fst) or (snd), then either BideducePairProj(Φ, T, T') returns `true` before the recursive calls, or w.l.o.g., the last rule in Π is (fst), thus $T = \mathsf{fst}(T_0)$ and $T' = \mathsf{fst}(T_0')$ for some T_0, T_0' and T_0, T_0' is proved by the direct subproof of Π. In this case, BideducePairProj(Φ, T, T') recursively calls BideducePairProj(Φ, T_0, T_0') and we conclude by aplpying the I.H. on the subproof ✓.
   - If it is another rule, by Definition 45, Π does not contain any rule (fst) or (snd), thus by the specification (4), BideducePairProjLxorR(Φ, T, T') return `true` ✓.
6. • First, by the specification (5), and since recursive calls in this function corresponds to the application of (pair), if BideducePairPair(Φ, T, T') returns `true`, then Φ, T, T' satisfy the specification ✓.
   • This case is similar to (5), each recursive call corresponding to an application of the rule (pair) ✓.

Lemma 50 : BideducePair terminates on all inputs.

Proof. We prove that each function terminates, from BideducePairProjLR to BideducePair.

1. BideducePairProjLR saturates `zip`(ϕ, ϕ') by applications of (fstLR)/(sndLR). The number of iterations of the **Until** loop is bounded by the maximum depth of a term in `zip`(ϕ, ϕ').
2. BideducePairProjL terminates since its recursive calls simply explore the AST of T'
3. BideducePairProjR terminates (similar to BideducePairProjL)
4. BideducePairProjLxorR terminates since BideducePairProjL and BideducePairProjR terminates.

5. BideducePairProj and BideducePairPair: the recursive calls simply explore the AST of T and T' simultaneously, there is at most $\min(|T|, |T'|)$ recursive calls. Plus the calls to BideducePairProjLxorR terminate.
6. BideducePair terminates since BideducePairProjLR and BideducePairPair terminates.

Theorem 51 : BideducePair decides bideduction in E_{pair}.

Proof. First, BideducePair always terminates by Lemma 50

Let's show that BideducePair(ϕ, ϕ', T, T') returns `true` iff there is a proof tree Π of T, T' in $\mathcal{I}_{\phi,\phi'}^{E_{\text{pair}}}$.

- First, by Lemma 49, if BideducePair(ϕ, ϕ', T, T') returns `true` then a proof tree of T, T' with leafs in BideducePairProjLR (ϕ, ϕ') exists. However, leafs in this set are also provable in `inferencePair` (ϕ, ϕ'), again by Lemma 49.
- Let's now prove the converse. Assume T, T' is provable in `inferencePair`(ϕ, ϕ'). By Proposition 46, there is a proof Π of T, T' in full-decomposition-composition form. Notice such a proof can be split in two parts:
  (i) the part near the root that only does not use (fstLR), (sndLR) nor (input), which is a cut of Π, let's call it $\Pi_{\text{composition}}$; and
  (ii) several subproofs of Π, $\Pi^0_{\text{full decomposition}}, ..., \Pi^{k-1}_{\text{full decomposition}}$ proving $(T_0, T'_0), ..., (T_{k-1}, T'_{k-1})$ containing only rules (fstLR), (sndLR) and (input) such that the conclusions of the $\Pi^i_{\text{full decomposition}}$ are the unproved premises of $\Pi_{\text{composition}}$.

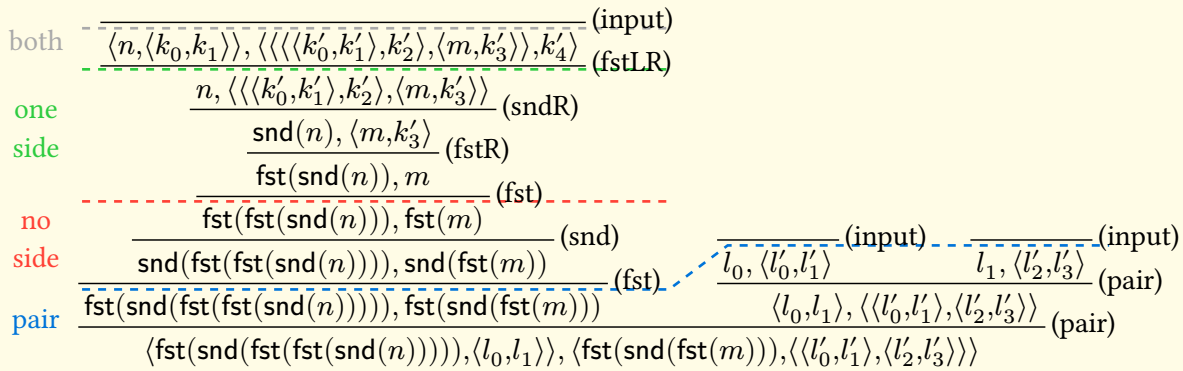
  By (1) of Lemma 49 and (ii), all the T_i, T'_i are in BideducePairProjLR(ϕ, ϕ') and by (6), since $\Pi_{\text{composition}}$ is an incomplete proof of T, T' in full-decomposition-composition form that doesn't use (fstLR), (sndLR) nor (input), BideducePairPair$(\phi,, \phi', T, T')$ returns `true` ✓.

Example. Let

$$\begin{aligned}\phi &= (\quad \langle n, \langle k_0, k_1\rangle\rangle \quad , \quad l_0 \quad , \quad l_1 \quad)\\ \phi' &= (\langle\langle\langle\langle k'_0, k'_1\rangle, k'_2\rangle, \langle m, k'_3\rangle\rangle, k'_4\rangle,\ \langle l'_0, l'_1\rangle\ ,\ \langle l'_2, l'_3\rangle\ )\end{aligned}$$

Here is a proof of $\phi, \phi' \rhd \langle \mathsf{fst}(\mathsf{snd}(\mathsf{fst}(\mathsf{fst}(\mathsf{snd}(n))))), \langle l_0, l_1\rangle\rangle, \langle \mathsf{fst}(\mathsf{snd}(\mathsf{fst}(m))), \langle\langle l'_0, l'_1\rangle, \langle l'_2, l'_3\rangle\rangle\rangle$.

As is formally explained in Proposition 46, we split the proofin four phases: the applications of (pair) constructor (pair), the applications of projections reducing on no side (no side), the applications of projections reducing on only one side (one side), and the applications of projections reducing on both sides (both).



6. (Un)decidability of $\triangleright$

In this section, we show several results of undecidability and reductions regarding the bideduction problem. Through this, we try to see how bideduction relates to similar problems: deduction and static equivalence, whose decidability has been thoroughly studied [1].

6.1. $\vdash$ reduces to $\triangleright$

We begin with a rather straightforward reduction: deduction reduces to bideduction.

Reduction 52 : Let ϕ, T be an instance of the deduction problem. $\phi \vdash T$ iff $\phi, \phi \triangleright T, T$.

Proof. Let ϕ, T.

- If there is a recipe R such that $R[\phi] = T$, then $R[\phi] = T$ and $R[\phi] = T$ thus $\phi, \phi \triangleright T, T$.
- Conversely, if there is R such taht $R[\phi] = T$ and $R[\phi] = T$, then $R[\phi] = T$ thus $\phi \vdash T$.

Plus, since there are equational theory in which $\vdash$ is undecidable [1, Section 3.1],

6.2. $\approx_s$ does not reduce to $\triangleright$

There is an equational theory [1, Section 3.3] such that we can encode an undecidable problem about Turing machines as a static equivalence problem.

In this theory though, for all ϕ, T, there is only a finite number of recipes that can verify $R[\phi] = T$. Therefore, deduction *and* bideduction are decidable in this theory.

In other words, static equivalence is at least harder than bideduction.

6.3. $\triangleright$ does not reduce to $\vdash$

We adapt the same equational theory as was previously used by Cortier and Abadi to show that deduction was undecidable [1, Section 3.1], reducing problems in the symbolic models to the Post correspondance problem (Pcp). Although here, the goal is to find a theory where deduction is decidable, but bideduction is not.

Definition 53 : Met $\Sigma_{\text{Pcp}'} = \{\mathsf{ok}_0/0, \mathsf{ok}_1/0, [\sqcup, \sqcup]^{\sqcup}/3, \mathsf{testpcpeq}/1, \sqcup \cdot \sqcup/2\}$ and $E_{\text{Pcp}'}$ generated by the following rules :

$$
\begin{aligned}
[x, y]^{\mathsf{ok}_0} \cdot [x', y']^{\mathsf{ok}_0} &\to [x \cdot x', y \cdot y']^{\mathsf{ok}_0} \\
[x, y]^{\mathsf{ok}_1} \cdot [x', y']^{\mathsf{ok}_1} &\to [x \cdot x', y \cdot y']^{\mathsf{ok}_1} \\
x \cdot (y \cdot z) &\to (x \cdot y) \cdot z \\
\mathsf{testpcpeq}\big([x, x]^{\mathsf{ok}_0}\big) &\to \mathsf{ok}_0 \\
\mathsf{testpcpeq}\big([x, x]^{\mathsf{ok}_1}\big) &\to \mathsf{ok}_1
\end{aligned}
$$

Notice this system is confluent and terminates, therefore convergent, we can thus define the value of a term.

Definition 54 : We say that T and T' are equal up to associativy, written $T \underset{A}{=} T'$, if these terms are equal regarding only the rule $x \cdot (y \cdot z) \to (x \cdot y) \cdot z$.

6.3.1. Undecidability of $\triangleright$

We reduce the Post correspondance problem to bideduction with for instance a two-letters alphabet contained in the names $\{n_0, n_1\} \subseteq \mathcal{N}$. We consider words $w_0 ... w_{k-1} \in \{n_0, n_1\}^*$ and corresponding terms $w_0 \cdot ... \cdot w_{k-1}$ are equal (whatever the order of the $\sqcup \cdot \sqcup$ operators in the terms).

Reduction 55 : Let an instance of Pcp $P = ((u_0, v_0), ..., (u_{n-1}, v_{n-1}))$. Let $\mathsf{BiPCP}(P)$ be the following instance of bideduction :

$$\phi = \left([u_0, v_0]^{\mathsf{ok}_0}, ..., [u_{n-1}, v_{n-1}]^{\mathsf{ok}_0}\right) \qquad T = \mathsf{ok}_0$$
$$\phi' = \left([u_0, v_0]^{\mathsf{ok}_1}, ..., [u_{n-1}, v_{n-1}]^{\mathsf{ok}_1}\right) \qquad T' = \mathsf{ok}_1$$

We now define a set of terms on a different signature that will be able to represent a recipe applied to an instance of $\mathsf{BiPCP}(P)$.

Definition 56 : We define the set of **well-formed pre-terms** (these are not terms since they use a symbol ok that is not in $\Sigma_{\text{Pcp}'}$) :

$$\mathsf{WF}((u_i, v_i)) \ni W, W', W'', ... ::= \mathsf{testpcpeq}(W) \mid [W, W']^{W''} \mid W \cdot W' \mid \left[u_{i_0} \cdot ... \cdot u_{i_{k-1}}, v_{i_0} \cdot ... \cdot v_{i_{k-1}}\right]^{\mathsf{ok}} \mid \mathsf{ok}_b$$

For $T \in \mathsf{WF}(P)$, we define by induction terms T_0 and T_1, that are T where each occurence of ok are replaced by the corresponding ok_b.

Let $\mathsf{WF}_b(P) = \{T_b \mid T \in \mathsf{WF}(P)\}$.

We naturally extend notions of size, subterms (we'll call them subpre-terms) and $\underset{A}{=}$ to well-formed pre-terms.

Proposition 57 : Let ϕ (resp. ϕ') from $\mathsf{BiPCP}(P)$, R a recipe. There is $T \in \mathsf{WF}(P)$ such that $R[\phi] \underset{E_{\text{Pcp}'}}{=} T_0$ and $R[\phi'] \underset{E_{\text{Pcp}'}}{=} T_1$.

Proof. We proceed by induction on recipes.
- ok_b : we directly apply the fifth rule of Definition 56 ✓.
- x_i : we directly apply the fourth rule of Definition 56 ✓.
- $\mathsf{testpcpeq}(R')$: We apply the I.H. on R' and get T' such that $T'_0 \underset{E_{\text{Pcp}'}}{=} R'[\phi]$ and $T'_1 \underset{E_{\text{Pcp}'}}{=} R'[\phi']$, notice $T = \mathsf{testpcpeq}(R')$ satisfy the conditions ✓.
- the other inductive cases are similar.

Lemma 58 : Let $T \in \mathsf{WF}(P)$. If $T_b{\downarrow} = \mathsf{ok}_b$, then T is (syntactically) of the form $\mathsf{testpcpeq}(T')$ with $T'_b{\downarrow} \Downarrow [V_b, V_b]^{\mathsf{ok}_b}$ for some values V_0, V_1.

Proof. $T \not== \mathsf{ok}_b$ since otherwise, $\mathsf{ok}_0 == \mathsf{ok}_1$. Therefore, the toplevel constructor must be $\mathsf{testpcpeq}$ (consider each case of Definition 56), since $\mathsf{testpcpeq}(\square)$ is the result of no rule.

By definition of the rules reducing $\mathsf{testpcpeq}$, we get the second part of the lemma.

We wish to show Lemma 60 to conclude on the reduction. We begin by generalizing it to be able to prove a result by induction.

Lemma 59 : Let $\tilde{T} \in \mathsf{WF}(P)$ of minimal size such that there are two values V_0, V_1 such that $\tilde{T}_b{\downarrow} = [V_b, V_b]^{\mathsf{ok}_b}$.

Let $T_0, ..., T_{k-1}$ subpre-terms of $\tilde{T}$ whose toplevel constructor is not $\cdot$ and such that $T_0 \cdot ... \cdot T_{k-1} \underset{A}{=} \tilde{T}$.

For all $i \in \{0, ..., k-1\}$, T_i is a variable.

Proof. Let $T_0, ..., T_{k-1}$ as in the lemma, we consider one of the T_i. First, notice that there are $V_0^i, V_1^i, V_0'^i$ and $V_1'^i$ such that $T_{ib} \Downarrow [V_b^i, V_b'^i]^{\mathsf{ok}_b}$.

We consider each case depending on the toplevel constructor of T_i.

- $T_i == x_i : T_i$ is already of the required form ✓.
- $T_i == [T_i', T_i'']^{T_i'''}$, by the previous remark, we must have $T_{ib}''' \Downarrow \mathsf{ok}_b$, however T''' is a subpre-term of $\tilde{T}$ which is assumed to be minimal among terms evaluating to ok_b, we thus conclude by ex falso ✓.
- $T_i == T_i' \cdot T_i''$: is not possible: we assumed T_i didn't have $\cdot$ as a toplevel constructor ✓.
- $T_i == \mathsf{testpcpeq}(T_i')$. We examine the possible values of T_i :
  - $T_{ib}' \Downarrow [V_{ib}, V_{ib}]^{V_{ib}'}$ with V_{i0}, V_{i1} values, $V_{i0}'' \in \{\mathsf{ok}_0, \mathsf{ok}_1\}$ and $V_{i1}' \in \{\mathsf{ok}_0, \mathsf{ok}_1\}$. However, $T_{ib}{\downarrow} \in \{\mathsf{ok}_0, \mathsf{ok}_1\}$ thus $T_{ib}{\downarrow}\left(= [V_b^i, V_b'^i]^{\mathsf{ok}_b}\right) \neq [\sqcup, \sqcup]^{\sqcup}$, and we conclude.
  - $T_i{\downarrow} = \mathsf{testpcpeq}(T_i'){\downarrow} = \mathsf{testpcpeq}(T_i'{\downarrow})\left(= [V_b^i, V_b'^i]^{\mathsf{ok}_b}\right) \neq [\sqcup, \sqcup]^{\sqcup}$, we conclude by ex falso.

Lemma 60 : Let $\tilde{T} \in \mathsf{WF}(P)$. If $\tilde{T}_0 \underset{E_{\text{PCP}'}}{=} \mathsf{ok}_0$ and $\tilde{T}_1 \underset{E_{\text{PCP}'}}{=} \mathsf{ok}_1$, then there is U of the form $\mathsf{testpcpeq}\left(U^0 \cdot ... \cdot U^{l-1}\right)$ such that the U^i are variables of $x_0, ..., x_{n-1}$ such that $U_0 \underset{E_{\text{PCP}'}}{=} \mathsf{ok}_b$ and $U_1 \underset{E_{\text{PCP}'}}{=} \mathsf{ok}_b$.

Proof. First, the toplevel constructor must be $\mathsf{testpcpeq}$ by Lemma 58.

Hence, we have $\tilde{T} = \mathsf{testpcpeq}(T)$ with $T \in \mathsf{WF}(P)$ such that $T_b \Downarrow [V_b, V_b]^{\mathsf{ok}_b}$. We show by induction on $|T'|$ that "If there are values V_0, V_1 such that $T \Downarrow [V_b, V_b]^{\mathsf{ok}_b}$ then there is U of the form $\mathsf{testpcpeq}\left(U^0 \cdot ... \cdot U^{l-1}\right)$ with U_i variables among $x_0, ...x_{n-1}$ such that $U_0 \underset{E_{\text{PCP}'}}{=} \mathsf{ok}_b$ et $U_1 \underset{E_{\text{PCP}'}}{=} \mathsf{ok}_b$"

- If $|T| = 1$ then T is a variable x_i such that $x_{ib} \Downarrow [V_b, V_b]^{\mathsf{ok}_b}$, thus $U \overset{?}{=\!=} (x_i)$ satisfy the conditions ✓.
- If $|T| > 1$, then let $T_0, ..., T_{k-1}$ be subpre-terms of T whose toplevel constructor is not $\cdot$ and such that $T_0 \cdot ... \cdot T_{k-1} \underset{A}{=\!=} T$.

  Notice, as in the proof of Lemma 59, that there are $V_0^i, V_1^i, V_0'^i$ and $V_1'^i$ such that $T_{ib} \Downarrow [V_b^i, V_b'^i]^{\mathsf{ok}_b}$.
  - If on of the T_i is of the form $[T_i', T_i'']^{T_i'''}$, we have $T_i''' \Downarrow \mathsf{ok}_b$, thus by applying Lemma 58 we get a subpre-term T_i^4 of T_i (thus of T), satisfying :
    1. $|T^4| < |T|$; and
    2. $T_b^4 \Downarrow \left[V_b^4, V_b^4\right]^{\mathsf{ok}_b}$ for some values V_0^4 et V_1^4.

    We thus get the result by applying the I.H. on T^4 ✓.
  - If no T_i is of this form, then, as in the proof of Lemma 59, all the T_i are variables, therefore $U \overset{?}{=\!=} (T_0 \cdot ... \cdot T_{k-1})$ satisfy the conditions ✓.

Proposition 61 : Bideduction is undecidable in $E_{\text{PCP}'}$.

Proof. Let's show that P is a positive instance of PCP iff $\mathsf{BiPCP}(P)$ is a positive instance of the bideduction problem.

- If P is a positive instance of PCP, consider $i_1, ..., i_k$ such that $u_{i_1}...u_{i_k} = v_{i_1}...v_{i_k}$, we directly have that $R = \mathsf{testpcpeq}\left(x_{i_1} \cdot ... \cdot x_{i_k}\right)$ witnesses $\phi, \phi' \rhd \mathsf{ok}_0, \mathsf{ok}_1$.
- If $\phi, \phi' \rhd \mathsf{ok}_0, \mathsf{ok}_1$, then we have a recipe R such that $R[\phi] \underset{\text{PCP}'}{=} \mathsf{ok}_0$ and $R[\phi'] \underset{\text{PCP}'}{=} \mathsf{ok}_1$.

  By Proposition 57, we have $T \in \mathsf{WF}(P)$ such that $T_0 \underset{E_{\text{PCP}'}}{=} \mathsf{ok}_0$ and $T_1 \underset{E_{\text{PCP}'}}{=} \mathsf{ok}_1$.

  Then, by Lemma 60, we can assume that T is of the form $\mathsf{testpcpeq}\left(\left[u_{i_0}, v_{i_0}\right]^v \cdot ... \cdot \left[u_{i_{l-1}}, v_{i_{l-1}}\right]^v\right)$, thus we have that $T_b \underset{E_{\text{PCP}'}}{=} \mathsf{testpcpeq}\left(\left[u_{i_0} \cdot ... \cdot u_{i_{l-1}}, v_{i_0} \cdot ... \cdot v_{i_{l-1}}\right]_b^{\mathsf{ok}}\right) \underset{\text{PCP}'}{=} \mathsf{ok}_b$, gence, we must have, by definition of the rules, $u_{i_0} \cdot ... \cdot u_{i_{l-1}} = v_{i_0} \cdot ... \cdot v_{i_{l-1}}$, which provides a witness of the positivity of the instance for PCP.

6.3.2. Decidability of $\vdash$

Proposition 62 : Deducibility is decidable in $E_{\text{PCP}'}$.

Proof.

Consider the following algorithm:

$\textsc{DeducePcp}(\phi = (M_0, .., M_{n-1}), T)$:
 $V := T\downarrow$
 Si il existe i tq $V = M_i\downarrow$:
  Renvoyer x_i
 Filtrer V :
  $\mathsf{ok}_b \to$ **Renvoyer** ok_b
  $[T_0, T_1]^{T_2} \to$
   $R_0, R_1, R_2 := \textsc{DeducePcp}(\phi, T_0), \textsc{DeducePcp}(\phi, T_1), \textsc{DeducePcp}(\phi, T_2)$
   Si $R_0, R_1, R_2 \neq$ `fail` :
    Renvoyer $[R_0, R_1]^{R_2}$
   Sinon
    $T_0^0, ..., T_0^{k-1} :=$ valeurs sans $\cdot$ toplevel tq $T_0^0 \cdot ... \cdot T_0^{k-1} \underset{A}{=} T_0$
    $T_1^0, ..., T_1^{l-1} :=$ valeurs sans $\cdot$ toplevel tq $T_1^0 \cdot ... \cdot T_1^{k-1} \underset{A}{=} T_1$
    Pour i tq $0 \leq i < k$:
     Pour j tq $0 \leq j < l$:
      $R_0 := \textsc{DeducePcp}\left(\phi, \left[T_0^0 \cdot ... \cdot T_0^{i-1}, T_1^0 \cdot ... \cdot T_1^{j-1}\right]^{T_2}\right)$
      $R_1 := \textsc{DeducePcp}\left(\phi, \left[T_0^i \cdot ... \cdot T_0^{k-1}, T_1^j \cdot ... \cdot T_1^{l-1}\right]^{T_2}\right)$
      Si $R_0, R_1 \neq$ `fail` :
       Renvoyer $R_0 \cdot R_1$
  $T_0 \cdot ... \cdot T_{n-1} \to$ // tel qu'aucun des T_i n'ait $\cdot$ en constructeur toplevel
   Pour i tq $0 \leq i < k$:
    $R_0, R_1 := \textsc{DeducePcp}(\phi, T_0 \cdot ... \cdot T_{i-1}), \textsc{DeducePcp}(\phi, T_i \cdot ... \cdot T_{k-1})$
    Si $R_0, R_1 \neq$ `fail` :
     Renvoyer $R_0 \cdot R_1$
   Renvoyer `fail`
  $\mathsf{testpcpeq}(T_0) \to$
   $R_0 := \textsc{DeducePcp}(\phi, T_0)$
   Si $R_0 \neq$ `fail` :
    Renvoyer $\mathsf{testpcpeq}(R_0)$
   Sinon
    Renvoyer `fail`

If $\textsc{DeducePcp}(\phi, T) \neq$ `fail`, it should be clear that the returned recipe witnesses $\phi \vdash T$.

Conversely, let R be a recipe on which no rule can be applied directly (without filling the holes) such that $R[\phi] \underset{E_{\text{Pcp}'}}{=} T$. Let's show by induction on R that there is R' a recipe such that :
1. $R'[\phi] \underset{E_{\text{Pcp}'}}{=} T$ and $\textsc{DeducePcp}(T) = R'$.

- If R is a constant ok_b it's straightforward with $R' = R$ ✓.
- If $R = \mathsf{testpcpeq}(R_0)$
  - if $R[\phi]\downarrow = \mathsf{testpcpeq}(R_0\downarrow[\phi]) = T\downarrow$, by applying the I.H. to R_0 we get the result ✓.
  - otherwise, after examining the rules, we see that we must have $R[\phi]\downarrow (= T\downarrow) = \mathsf{ok}_b$, and we directly have $R' = \mathsf{ok}_b$ satisfying the conditions ✓.
- If $R = [R_0, R_1]^{R_2}$, since $[\sqcup, \sqcup]^{\sqcup}$ is the left hand side of no rewriting rule, $R[\phi]\downarrow = [R_0\downarrow[\phi], R_1\downarrow[\phi]]^{R_2\downarrow[\phi]} = T\downarrow$. We thus get the result by applying the I.H. on R_0, R_1 and R_2 ✓.
- If $R = R_0 \cdot R_1$, then :
  - if $V = V_0 \cdot ... \cdot V_{k-1}$

- if $R_1[\phi]\downarrow = V_0 \cdot V_1$, in which case $R[\phi]\downarrow = (R_0[\phi]\downarrow \cdot V_0) \cdot V_1 = T\downarrow$. By applying the I.H. on R_1, we get R'_1 such that $R'_1[\phi] = V_0 \cdot V_1$ with $\cdot$ as the toplevel constructor i.e. $R'_1 = R'^0_1 \cdot R'^1_1$. Thus, with $R' = (R_0 \cdot R'^0_1) \cdot R'^1_1$ we get the result ✓.
- otherwise, if $R_0[\phi]\downarrow = [V_0, V_1]^{\mathsf{ok}_b}$ and $R_1[\phi] = [V'_0, V'_1]^{\mathsf{ok}_b}\downarrow$, then $R[\phi]\downarrow = [(V_0 \cdot V'_0)\downarrow, (V_1 \cdot V'_1)\downarrow]^{\mathsf{ok}_b}$.

  By applying the I.H. on R_0 and R_1, we get R'^0_0, R'^1_0, and R'^2_0 but also R'^0_1, R'^1_1, and R'^2_1 such that $R'_0 = \left[R'^0_0, R'^1_0\right]^{R'^2_0}$ and $R'_1 = \left[R'^0_1, R'^1_1\right]^{R'^2_1}$ satisfying the conditions.

  Let $R' = \left[R'^0_0 \cdot R'^0_1, R'^1_0 \cdot R'^1_1\right]^{\mathsf{ok}_b}$ from the I.H., $R'^0_0[\phi] = V_0$, $R'^1_0[\phi] = V_1$ and the same goes for R'^0_1, R'^1_1.

  Hence, $R'[\phi]\downarrow = \left[(R'^0_0[\phi] \cdot R'^0_1[\phi])\downarrow, (R'^1_0[\phi] \cdot R'^1_1[\phi])\downarrow\right]^{\mathsf{ok}_b} = [(V_0 \cdot V'_0)\downarrow, (V_1 \cdot V'_1)\downarrow]^{\mathsf{ok}_b} = T\downarrow$.

  R' thus satisfy the three conditions ✓.
- otherwise, we must have $R_0[\phi]\downarrow \cdot R_1\downarrow[\phi]\downarrow \neq R_0[\phi]\downarrow \cdot R_1\downarrow[\phi]$ and we applied some other rules than associativity. We thus must have applied the rewriting rule $[x, y]^{\mathsf{ok}_b} \cdot [x', y']^{\mathsf{ok}_b} \to [x \cdot x', y \cdot y']^{\mathsf{ok}_b}$, therefore $V = [V_0, V_1]^{V_2}$. Henceforth, we have $V^0_0, V^1_0, V^0_1, V^1_1$ such that :
  1. $R_0[\phi]\downarrow = \left[V^0_0, V^0_1\right]^{V_2}$ and $R_0[\phi]\downarrow = \left[V^1_0, V^1_1\right]^{V_2}$; and
  2. $V_0 = V^0_0 \cdot V^1_0$ and $V_1 = V^0_1 \cdot V^1_1$.

  Since we backtrack across all possible splitting in the algorithm DeducePcp, either we find a recipe such that $R[\phi] = V$ sooner, or we'll consider this splitting of V_0 and V_1, and by I.H. on R_0 and R_1, we'll find $R'_0 \cdot R'_1$ witnessing the deduction ✓.

Plus, it is rather straightforward to notice that the algorithm terminates, the size of T being a decreasing variant across recursive calls, we get that $\vdash$ is decidable in $E_{\text{Pcp}'}$.

6.3.3. (Un)decidability of $\approx_s$

It seems that $\approx_s$ is undecidable in $E_{\text{Pcp}'}$, but we couldn't prove it.

It should be clear that if P is a positive instance of Pcp, then $\mathsf{BiPCP}(P)$ is negative for static equivalence: let R a recipe proving bideduction of $\mathsf{BiPCP}(P)$, we have $R_0 = R$ and $R_1 = \mathsf{ok}_0$ distinguishing ϕ and ϕ'.

It remains to prove that if P is negative, then the frames of $\mathsf{BiPCP}(P)$ are indistinguishable.

The problem is there can be differences between the two frames, e.g. with $R_0 = \square_0 \cdot [\mathsf{ok}_0, \mathsf{ok}_0]^{\mathsf{ok}_0}$ and $R_1 = \square_0 \cdot [\mathsf{ok}_1, \mathsf{ok}_1]^{\mathsf{ok}_1}$, the first term of each frame merges with $[\mathsf{ok}_b, \mathsf{ok}_b]^{\mathsf{ok}_b}$ iff it reduces to a term of the form $[\_, \_]^{\mathsf{ok}_b}$, which may be the case for ϕ but not for ϕ'. Although, we couldn't use these differences to distinguish the two frames, and our intuitions is that we cannot, and that $\mathsf{BiPCP}(P)$ must yield indistinguishable frames in this case.

6.4. Conclusion on reductions and decidabilities

The following diagram sums up reductions and decidability results. Consider E_{trivial} an empty equational theory, all problems are decidable in this theory.

Each ellipse contains the decidable problems in the corresponding theory.

Note the green ellipse has this form only if the conjecture in Section 6.3.3 is true.

Notice furthermore that, whereas the reduction from $\vdash$ to $\rhd$ is carried out without changing the signature nor the theory, the reduction from $\vdash$ to $\approx_s$ adds a symbol to the signature without changing

the theory [1, Section 3.2]. To our knowledge it is not known whether this last reduction can be done without changing the signature.

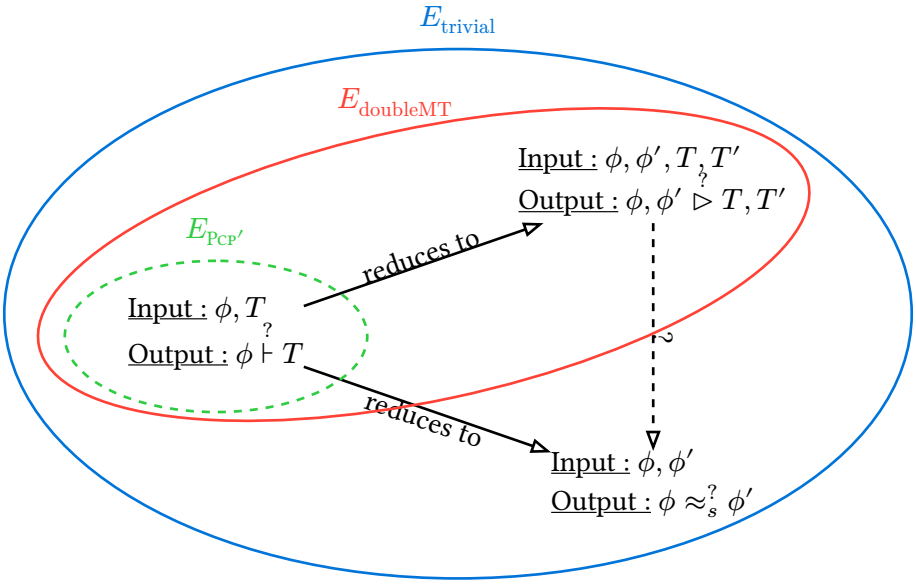


Figure 1: Réductions, théories & les problèmes qu'elles rendent décidables

7. With a set of ground equalities assumed to hold

7.1. Definition of the extended bideduction problem

Definition 63 : Let $X = \{T_0 = U_0, ..., T_{N-1} = U_{N-1}\}$ a set of ground equalities of which we consider $\mathcal{E}_X$ its closure by application of context, reflexivity, transitivity and symmetry.

Let Σ be a signature, E an equational theory. We say that T and T' are equal up to E and $\mathcal{E}$, written $T \overset{\mathcal{E}}{\underset{E}{=}} T'$, if there is a sequence of terms $T_0, ..., T_{n-1}$ such that $(T_i = T_{i+1}) \in E \cup \mathcal{E}$ for all $i, 0 \leq i < n-1$ and such that $T == T_0$ and $T_n == T'$. We write $T \overset{\mathcal{E}}{=} T'$ if all equalities used are in $\mathcal{E}$.

Let $\mathcal{E}, \mathcal{E}'$ be sets of ground equalities, ϕ, ϕ' sequences of terms and T, T' terms. We write $\phi, \phi' \overset{\mathcal{E},\, \mathcal{E}'}{\triangleright} T, T'$ if there is a recipe R such that $R[\phi] \overset{\mathcal{E},\, \mathcal{E}'}{\underset{E}{=}} T$ and $R[\phi] \overset{\mathcal{E},\, \mathcal{E}'}{\underset{E}{=}} T'$.

We write $\overset{\mathcal{E}}{\triangleright}$ if $\mathcal{E} = \mathcal{E}'$.

Now that we know how to decide some bideduction problems, we would like to consider the following extension of the problem:

Definition 64 : We define a generalization of the bideduction decision problem to ground equalities.
DecBideduceGround$_{E,\, \mathcal{E},\mathcal{E}'}$
Input : ϕ, ϕ' sequences of messages, T, T' target terms.
Output : Does $\phi, \phi' \overset{E,\, \mathcal{E}}{\triangleright} T, T'$ hold ? i.e. is there R a recipe such that $R[\phi] \overset{\mathcal{E}}{\underset{E}{=}} T$ and $R[\phi'] \overset{\mathcal{E}'}{\underset{E}{=}} T'$.

Note we consider here a variant where ground equalities need not be the same on the left and the right side. This is the more general problem, as all instances where the same set of ground equalities is assumed to hold on both sides are instances of this problem too.

To grasp a better understanding of groun equalities in bideduction, we first consider the restriction where the equational theory is empty. We don't consider the full problem here.

If we could completely separate the application of ground equalities from the application of equations of the theory, the problem could be split into two parts, however this is actually not possible, even in simple theory.

7.2. A counter example to separation of ground and theory equalities with only enc/dec

Example. Even in this theory, there may be cases where we need to alternate between rewriting steps and application of ground equalities.

$$X = \{\mathsf{enc}(n, l') = \mathsf{enc}(m, k)\} \quad \phi = (\mathsf{enc}(n, l), l, l', k) \quad R = \mathsf{dec}(\mathsf{enc}(\mathsf{dec}(\square_0, \square_1), \square_2), \square_3)$$

$$R[\phi] = \mathsf{dec}(\mathsf{enc}(\mathsf{dec}(\mathsf{enc}(n, l), l), l'), k) \to$$

$$\mathsf{dec}(\mathsf{enc}(n, l'), k) \overset{\mathcal{E}_X}{=}$$

$$\mathsf{dec}(\mathsf{enc}(m, k), k) \to$$

$$m$$

7.3. Restriction on the ground equalities

We first show some results about how many terms can be obtain from a term applying only ground equalities, that will later be used to show a first decidability result.

Definition 65 : Let X be a set of ground equalities, we say that $\mathcal{E}_X$ is *recursive* if there is an infinite sequence of terms $(T_i)_{i\in\mathbb{N}}$ such that for all i, $T_i \overset{\mathcal{E}_X}{=} T_{i+1}$ and the T_i are pairwise distinct (syntactically).

Example. A set of ground equalities with $\langle n,m\rangle = \langle\langle n,m\rangle,m\rangle$, $f(n) = f(f(m))$ and $m = n$, or $n = \langle m,m'\rangle$ and $m = \langle m,n\rangle$ or $f(x) = x$ will allow, from a ground term T, to generate infinitely many terms U such that $T \overset{\mathcal{E}}{=} U$.

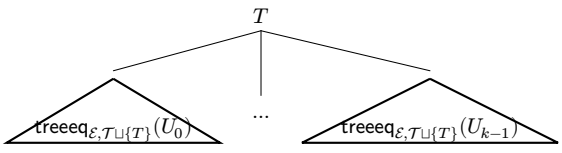
Proposition 66 : $\mathcal{E}$ is recursive iff there is a term T such that there is infinitely many terms equal to T w.r.t. $\mathcal{E}$.

Proof.

- First, notice that if $\mathcal{E}$ is recusive, we have infinitely many distinc terms $(T_i)_{i\in\mathbb{N}}$ such that $T_i \overset{\mathcal{E}}{=}_a T_{i+1}$, thus *a fortiori* infinitely many terms equal to T_0 up to $\mathcal{E}$.
- Conversely, let's show by contraposition that if $\mathcal{E}$ *is not* recursive, then there is a finite number of terms equal to T up to $\mathcal{E}$.

  Let $\mathsf{eq}_{\mathcal{E}}(T) = \left\{U \mid T \overset{\mathcal{E}}{=} U\right\}$. We recursively define a finitely-branching tree, whose set of labels is precisely $\mathsf{eq}_{\mathcal{E}}(T)$, representing possible choices on the applications of atomic equalities transforming T into another term $U \in \mathsf{eq}_{\mathcal{E}}(T)$.

  $\mathsf{treeeq}_{\mathcal{E},\mathcal{T}}(T) =$

  where $\{U_0, ..., U_{k-1}\} = \left\{U \mid T \overset{\mathcal{E}}{=}_a U, U \notin \mathcal{T}\right\}$. Let's first show that this tree is well-defined. First, it is finitely-branching (with $k < +\infty$ branch above) since the application of an atomic equality for $\mathcal{E}_X$ correspond to a choice of a node in the AST of T, and the choice of an equality in X, therefore we have at most $\#X \cdot |T| < +\infty$ terms in atomic equality with T.

  Then, this tree is finite. Since $\mathcal{E}$ is not recursive, there is no infinite branch in this tree since the argument $\mathcal{T}$ ensures that each branch contains pairwise distinct terms. Hence, this is a finitely branching tree with no infinite branch, we conclude by König's lemma.

Definition 67 : Let T, U be terms and X a set of ground equalities, we say that T and U are root equal, written $T \overset{\varepsilon_X}{=}_{\text{root}} U$, if there are $N \in \mathbb{N}$ and $T_0, ..., T_N$ such that $T == T_0$, for $i, 0 \leq i < N - 1$ we have $T_i \overset{\varepsilon_X}{=}_a T_{i+1}, T_N == U$ and there exists $i_0, 0 \leq i_0 < N - 1$ such that $(T_i = T_{i+1}) \in X$.

We furthermore say that the equality is *strict*, written $T \overset{\varepsilon_X}{\Leftrightarrow}_{\text{root}}$, if $T \not\equiv U$.

Hence, $T \overset{\varepsilon_X}{=}_{\text{root}} U$ means that we applied at leas an atomic equality at the root of a term between T and U i.e. there is no context C and subterm T' of T such that $C[T'] = T, T' \overset{\mathcal{E}}{=} U'$ and $C[U'] = U$.

Definition 68 : Let ϕ, ϕ' be sequences of messages of the same length, we let

$$\mathsf{bignd}_{\mathcal{E},\,\mathcal{E}'}(\phi, \phi') = \left\{ (T, T') \;\middle|\; \exists R, \begin{pmatrix} R[\phi] \overset{\mathcal{E}}{\Leftrightarrow}_{\text{root}} T \\ R[\phi'] \overset{\mathcal{E}'}{=} T' \end{pmatrix} \text{ ou } \begin{pmatrix} R[\phi] \overset{\mathcal{E}}{=} T \\ R[\phi'] \overset{\mathcal{E}'}{\Leftrightarrow}_{\text{root}} T' \end{pmatrix} \right\}$$

Definition 69 : Let $\mathcal{E}, \mathcal{E}'$ be sets of ground equalities. We say $\mathsf{bignd}_{\mathcal{E},\,\mathcal{E}'}(\sqcup, \sqcup)$ is *effectively sup-computable* if there is an algorithm $\mathcal{R}_{\mathcal{E},\mathcal{E}'}$ such that for all ϕ, ϕ' sequences of messages, $\mathcal{R}_{\mathcal{E},\mathcal{E}'}(\phi, \phi')$ represents a set of recipes, and for all $(T, T') \in \mathsf{bignd}_{\mathcal{E},\,\mathcal{E}'}(\phi, \phi')$, there is $(U, U') \in \left\{(R[\phi], R[\phi']) \mid R \in \mathcal{R}_{\mathcal{E},\mathcal{E}'}(\phi, \phi')\right\}$ such that $T \overset{\mathcal{E}}{=} U$ and $T' \overset{\mathcal{E}'}{=} U'$ (is a subset up to equalities in $\mathcal{E}, \mathcal{E}'$).

Lemma 70 : If $\mathcal{E}$ is recursive, then there is a term T such that there is an infinite set $\mathcal{T}$ such that for each $U \in \mathcal{T}, T \overset{\varepsilon}{=}_{\text{root}} U$.

Proof. By Proposition 66, if $\mathcal{E}$ is recursive, we have T a term with infintely many terms equal to T w.r.t. $\mathcal{E}$. Let $\mathcal{T} = \left\{U \mid T \overset{\mathcal{E}}{=} U\right\}$. We proceed by induction on T to show that there is a term T' with the same property, but restricted to root equalities.

- if T is a variable or a name, then all equalities from T are ground equalities.
- otherwise, $T = f(T_0, ..., T_{k-1})$. We have two subcases.
  - If there is infintely many terms in $\mathcal{T}$ root equal to T, then it's done.
  - Otherwise, if there is only finitely many terms in $\mathcal{T}$ which are root equal to T, then $\mathcal{T} \supseteq \mathcal{ST}$ where $\mathcal{ST} = \left\{f(U_0, ..., U_{k-1}) \mid U_0 \overset{\mathcal{E}}{=} T_0 \text{ and } ... \text{ and } U_{k-1} \overset{\mathcal{E}}{=} T_{k-1}\right\}$ and $\mathcal{ST}$ contains infintely many terms. Thus there is at least one T_{i_0} such that $\left\{U \mid U \overset{\mathcal{E}}{=} T_{i_0}\right\}$ is infinite, because otherwise $\#\mathcal{ST} = \prod_{i=0}^{k-1} \#\left\{U \mid U \overset{\mathcal{E}}{=} T_i\right\} < +\infty$. We apply the I.H. on this T_{i_0}.

Definition 71 : Let X be ground equalities. We say $\mathcal{E}_X$ is *only atomic* if for each $(T = U) \in X$, T (resp. U) is neither a subterm of U (resp. T) nor of any other term appearing in X.

Proposition 72 : Let $\mathcal{E}$ be only atomic. If $T \overset{\varepsilon}{=}_{\text{root}} U$, then either $T == U$ or $T \overset{\varepsilon}{=}_a U$.

Proof. Let T_0, T_1, T_2 be terms such that $T_0 \overset{\varepsilon_X}{=}_a T_1$ and $T_1 \overset{\varepsilon_X}{=}_a T_2$, let's show that $T_0 == T_2$. By definition, one of these two atomic equalities are applied at the root w.l.o.g. assume that $T_0 \overset{\varepsilon_X}{=}_{\text{root}} T_1$. Therefore, $(T_0 = T_1) \in X$.

Let's show that the equality applied between T_1 and T_2 is $(T_1 = T_0)$. Indeed, for any other equalities $(U = U') \in X$, neither U nor U' is a subterm of T_1.

Finally, we have that $T_0 == T_2$, which concludes the proof.

Corollary 73 : If $\mathcal{E}$ is only atomic, then it is not recursive.

Lemma 74 : Let $\mathcal{E}, \mathcal{E}'$ be ground equalities. If $\mathcal{E}$ and $\mathcal{E}'$ are only atomic, then $\mathsf{bignd}_{\mathcal{E},\,\mathcal{E}}(\sqcup, \sqcup)$ is effectively supcomputable.

Proof. Let X, X' be sets of ground equalities. Consider the following algorithm.

$\mathcal{R}_{\mathcal{E}_X,\mathcal{E}_{X'}}(\phi,\phi')$
$s_X := \max\{|T| \mid \exists U, (T=U)\in X \vee (U=T)\in X\}$
$s_{X'} := \max\{|T| \mid \exists U, (T=U)\in X \vee (U=T)\in X\}$
res $:= \varnothing$
For each R recipes such that $|R| \leq \max(s_X, s_{X'})$:
| res $:=$ res $\cup \{R\}$
Return res

Let's show that $\mathcal{R}_{\mathcal{E}_X,\mathcal{E}_{X'}}(\sqcup,\sqcup)$ effectively supcomputes $\mathsf{bignd}_{\mathcal{E}_X,\,\mathcal{E}_{X'}}(\sqcup,\sqcup)$.

Let ϕ,ϕ' be sequences of terms, let $R \in \mathsf{bignd}_{\mathcal{E}_X,\,\mathcal{E}_{X'}}(\phi,\phi')$. By definition, $(R[\phi] \overset{\mathcal{E}_X}{\Leftrightarrow}_{\text{root}} T$ and $R[\phi'] \overset{\mathcal{E}_{X'}}{=} T')$ or $(R[\phi] \overset{\mathcal{E}_X}{=} T$ and $R[\phi'] \overset{\mathcal{E}_{X'}}{\Leftrightarrow}_{\text{root}} T')$. Assume w.l.o.g. that the former two equalities hold.

We therefore have $R[\phi] \overset{\mathcal{E}_X}{\Leftrightarrow}_{\text{root}} T$ and, by Proposition 72, $R[\phi] \overset{\mathcal{E}_X}{=}_a$, thus $(R[\phi]=T)\in X$, and therefore, $|R| \leq \max\{|T| \mid \exists U, (T=U)\in X \vee (U=T)\in X\} \leq \max(s_X, s_{X'})$ in the algorithm, thus $R \in \mathcal{R}_{\mathcal{E}_X,\mathcal{E}_{X'}}(\phi,\phi')$, which concludes the proof.

Example. Continuing the example with the ground equality $f(n)=n$.

Consider $\phi=(n)$ and $\phi'=(m)$, we have that for all $k\in\mathbb{N}$, we cab apply each recipe of the form $f^k(\square)$, and get all pairs of the form $\left(f^l(n), f^k(m)\right)$ for every $k\in\mathbb{N}$ and $l, 0\leq l\leq k$.

7.4. A first result of decidability with ground equalities

We now focus on the annouced restriction, i.e. taking $E=\varnothing$.

Definition 75 : We consider the problem $\textsc{BideductionOnlyGnd}_{\Sigma,\,\mathcal{E},\mathcal{E}'}$
Input : ϕ,ϕ',T,T'
Output : Is there a recipe R such that $R[\phi] \overset{\mathcal{E}}{=} T$ and $R[\phi] \overset{\mathcal{E}}{=} T'$?

Lemma 76 : Let ϕ,ϕ',T,T' and R such that $R[\phi] \overset{\mathcal{E}}{=} T$ and $R[\phi] \overset{\mathcal{E}}{=} T'$. We have R_{hat} with k hole and $R^0_{\text{root}},\dots,R^{k-1}_{\text{root}}$ such that:
- $R = R_{\text{hat}}[R^0_{\text{root}},\dots,R^{k-1}_{\text{root}}]$;
- $R[\phi] \Downarrow R_{\text{hat}}[R^0_{\text{root}}[\phi]\downarrow,\dots,R^{k-1}_{\text{root}}[\phi]\downarrow]$, $R[\phi'] \Downarrow R_{\text{hat}}[R_{\text{root}}[\phi']\downarrow]$; and
- for all $i, 0\leq i<k$, either:
  - $R^i_{\text{root}}[\phi] \overset{\mathcal{E}}{=}_{\text{root}} U_i$ such that $R_{\text{hat}}[R^0_{\text{root}}[\phi]\downarrow,\dots,U_i,\dots,R^{k-1}_{\text{root}}[\phi]\downarrow] = T$; or
  - $R^i_{\text{root}}[\phi'] \overset{\mathcal{E}}{=}_{\text{root}} U'_i$ such that $R_{\text{hat}}[R^0_{\text{root}}[\phi]\downarrow,\dots,U'_i,\dots,R^{k-1}_{\text{root}}[\phi]\downarrow] = T'$; or
  - R^i_{root} is directly a hole.

In the following, we denote by R_{hat} the greatest such recipe, and $R^0_{\text{root}},\dots,R^{k-1}_{\text{root}}$ the corresponding subrecipes.

Proof. We proceed by induction on the recipe R.
- Whatever the case, if $R[\phi] \overset{\mathcal{E}}{=}_{\text{root}} T$ or $R[\phi] \overset{\mathcal{E}}{=}_{\text{root}} T'$, then $R^{\text{hat}} = \square$ and $R^{\text{root}} = R$ satisfy the conditions ✓.
- If $R=\square_i$: $R^{\text{hat}}=\square$ and $R^0_{\text{root}}=\square_i$ satisfy the condition ✓.
- If $R=f(R_0,\dots,R_{a-1})$, then we discuss further subcases:
  - if $R[\phi] \overset{\mathcal{E}}{=}_{\text{root}} T$ or $R[\phi'] \overset{\mathcal{E}'}{=}_{\text{root}} T'$, then $R_{\text{hat}}=\square$ and $R^0_{\text{root}}=R$ satisfy the conditions;
  - otherwise, T and T' must be of the form $f(T_0,\dots,T_{a-1})$ and $f(T'_0,\dots,T'_{a-1})$ respectively. Plus, we must have $R_i[\phi] \overset{\mathcal{E}}{=} T_i$ and $R_i[\phi'] \overset{\mathcal{E}'}{=} T'_i$. We thus apply the I.H. on each R_i, T_i, T'_i and get recipes
    - $R^0_{\text{hat}},\dots,R^{a-1}_{\text{hat}}$; and
    - for all $i, 0\leq i<a$, $R^{0,i}_{\text{root}},\dots,R^{k_i-1,i}_{\text{root}}$

satisfying the conditions.

We let $R_{\text{hat}} = f\left(R^0_{\text{hat}}, ..., R^{a-1}_{\text{hat}}\right)$ and $(R_{\text{root}}) = \left(R^{0,0}_{\text{root}}, ..., R^{k_0-1,0}_{\text{root}}, ..., R^{0,a-1}_{\text{root}}, ..., R^{k_{a-1}-1,a-1}_{\text{root}}\right)$ and observe it satisfy the conditions ✓.

We consider the greatest such recipes to avoid a case where the root equality is obtained by reflexivity on one side.

Example. Let $X = \{(a = b)\}$, $R = f(\square)$ proving $a, b \rhd f(b), f(b)$. $R^{\text{hat}} = \square$ satisfy the condition since $f(b) \overset{\varepsilon_X}{=}_{\text{root}} f(b)$, but we would rather consider $R^{\text{hat}} = f(\square), R^0_{\text{root}} = \square_0$ where an equality is effectively applied at the root of R^0_{root}.

Lemma 77 : Let $\mathcal{E}, \mathcal{E}', \phi, \phi', T, T'$ such that $\phi, \phi' \overset{\mathcal{E},\,\mathcal{E}'}{\rhd} T, T'$, and let R a recipe proving this bideduction. Let R_{hat} and $R^0_{\text{root}}, ..., R^{k-1}_{\text{root}}$. With the same notation as in Lemma 76, for each $i, 0 \leq i < k$ we have either:
(i) $R^i_{\text{root}}[\phi] \overset{\varepsilon}{\Leftrightarrow}_{\text{root}} U_i$ or $R'^i_{\text{root}}[\phi] \overset{\varepsilon}{\Leftrightarrow}_{\text{root}} U'_i$; or
(ii) $R^i_{\text{root}} = \square_j$.

Proof. Let $R_{\text{hat}}, R^0_{\text{root}}, ..., R^{k-1}_{\text{root}}$ corresponding to ϕ, ϕ', T, T'. Let $i, 0 \leq i < k$. Assume R^i_{root} do not satisfy the first condition.

We have w.l.o.g. that $R^i_{\text{root}} \overset{\varepsilon}{=}_{\text{root}} U_i$ but $R^i_{\text{root}} \overset{\varepsilon}{\Leftrightarrow}_{\text{root}} U_i$. Thus, $R^i_{\text{root}}[\phi] == U_i$. We either have:
- $R^i_{\text{root}} = \square_j$, then it already satisfies the second condition ✓ or
- $R^i_{\text{root}} = f\left(R^i_{\text{root}_0}, ..., R^i_{\text{root}_{a-1}}\right)$. Since $R^i_{\text{root}}[\phi'] \overset{\varepsilon'}{\Leftrightarrow}_{\text{root}} U'_i$, we have $R^i_{\text{root}}[\phi'] \Downarrow f\left(R^i_{\text{root}_0}[\phi]\downarrow, ..., R^i_{\text{root}_{a-1}}[\phi]\downarrow\right)$. Therefore $R_{\text{hat}}[..., f(\square^i_0, ..., \square^i_{a-1}), ...]$ still satisfy Lemma 76 and is greater than R^{hat}, contradicting the assumption ✓.

7.5. A decision procedure when the ground equalities are only atomic

Algorithm 78 :

BideduceFromBignd$_{\mathcal{E},\mathcal{E}'}(\phi, \phi', T, T')$:
  Return BideduceFromBigndRec$_{\mathcal{E},\mathcal{E}'}(\phi, \phi', T, T', \text{Bignd}(\phi, \phi') \cup \{\square_0, ..., \square_{n-1}\})$

BideduceFromBigndRec$_{\mathcal{E},\mathcal{E}'}(\phi, \phi', T, T', \mathcal{R})$:
  For $R \in \mathcal{R}$:
    If $R[\phi] \overset{\mathcal{E}}{=} T$ **and** $R[\phi'] \overset{\mathcal{E}'}{=} T'$:
      Return R
  Match T, T':
    $f\left(T_0, ..., T_{\text{ar}(f)-1}\right), f\left(T'_0, ..., T'_{\text{ar}(f)-1}\right) \rightarrow$
      For $i \in \{0, ..., \text{ar}(f) - 1\}$:
        $R_i :=$ BideduceFromBigndRec$_{\mathcal{E},\mathcal{E}'}(\phi, \phi', T_i, T'_i, \mathcal{R})$
      If $\&\&_{i=0}^{\text{ar}(f)-1} R_i \neq$ `fail`:
        Return $f\left(R_0, ..., R_{\text{ar}(f)-1}\right)$
    _, _ $\rightarrow$ **Pass**
  Return `fail`

We actually show a more general result it suffices to assume that $\text{bignd}_{\mathcal{E},\,\mathcal{E}'}(\sqcup, \sqcup)$ is effectively supcomputable.

Theorem 79 : If $\text{bignd}_{\mathcal{E},\,\mathcal{E}'}(\sqcup, \sqcup)$ is effectively supcomputable, then BideductionOnlyGnd is decidable.

Proof. Assume Bignd effectively supcomputes $\text{bignd}_{\mathcal{E},\,\mathcal{E}'}(\sqcup, \sqcup)$, we claim that BideduceFromBignd decides BideductionOnlyGnd.

First, this algorithm always terminates: the recursive function only executes a finite **For** loop and explore the AST of the terms T, T'.

Let's now prove the correction of this algorithm.

- Clearly, if $\textsc{BideduceFromBigndRec}(\phi, \phi', T, T', \mathcal{R})$ returns a recipe, then it proves $\phi, \phi' \overset{\mathcal{E},\, \mathcal{E}'}{\rhd} T, T'$ ✓.
- Assume there is a recipe R proving $\phi, \phi' \overset{\mathcal{E},\, \mathcal{E}'}{\rhd} T, T'$. By Lemma 76, consider $R_{\text{hat}}, R^0_{\text{root}}, ..., R^{k-1}_{\text{root}}$. Since $R^{\text{hat}}[R^0_{\text{root}}[\phi], ..., R^{k-1}_{\text{root}}[\phi]] \Downarrow R^{\text{hat}}[R^0_{\text{root}}[\phi]\downarrow, ..., R^{k-1}_{\text{root}}[\phi]\downarrow]$ and $R^{\text{hat}}[R^0_{\text{root}}[\phi'], ..., R^{k-1}_{\text{root}}[\phi']] \Downarrow R^{\text{hat}}[R^0_{\text{root}}[\phi']\downarrow, ..., R^{k-1}_{\text{root}}[\phi']\downarrow]$, notice R^{hat} can be seen as a cut of T, T' and that if $\textsc{BideduceFromBigndRec}(\phi, \phi', R^i_{\text{root}}[\phi']\downarrow, R^i_{\text{root}}[\phi']\downarrow, \mathcal{R}) \neq \texttt{fail}$, then the algorithm returns a recipe.

  Indeed, either all the recursive call $\textsc{BideduceFromBigndRec}(\phi, \phi', R^i_{\text{root}}[\phi']\downarrow, R^i_{\text{root}}[\phi']\downarrow, \mathcal{R})$ are reached, or a recipe was found earlier.

  Let $i, 0 \leq i < k$, it now suffices to show that $\textsc{BideduceFromBigndRec}(\phi, \phi', R^i_{\text{root}}[\phi']\downarrow, R^i_{\text{root}}[\phi']\downarrow, \mathcal{R}) \neq \texttt{fail}$.

  According to Lemma 77, either:

  (i) $R^i_{\text{root}}[\phi] \overset{\mathcal{E}}{\Leftrightarrow}_{\text{root}} U_i$ or $R'^i_{\text{root}}[\phi] \overset{\mathcal{E}}{\Leftrightarrow}_{\text{root}} U'_i$ and $R^i_{\text{root}} \in \mathsf{bignd}_{\mathcal{E},\, \mathcal{E}'}(\phi, \phi') \subseteq \textsc{Bignd}(\phi, \phi') \subseteq \textsc{Bignd}(\phi, \phi') \cup \{\square_0, ..., \square_{n-1}\}$; or

  (ii) $R^i_{\text{root}} = \square_j$, and $R^i_{\text{root}} \in \{\square_0, ..., \square_{n-1}\} \subseteq \textsc{Bignd}(\phi, \phi') \cup \{\square_0, ..., \square_{n-1}\}$.

  In both cases, this call returns a recipe, which concludes the proof ✓.

8. Conclusion

We were able to show that several variations of the bideduction problem were decidable, and relate its decidability to static equivalence and deduction. However, in the one hand a general bideduction result, in the setting of decryption/encryption (without destructors) seems out of reach, yet we couldn't prove that it's undecidable.

Plus, it is still unclear how the hardness of bideduction relates to static equivalence. At this point, either bideduction reduces to static equivalence, or their hardness is unrelated (each one can be decidable while the other is not).

Bibliography

[1] M. Abadi and V. Cortier, "Deciding knowledge in security protocols under equational theories."

[2] S. Delaune, "Intruder Deduction Problem."