Automatic Generation of Optimal Reductions of Distributions
Liyong Lin, Tomáš Masopust, W. Murray Wonham, Rong Su

Abstract—A reduction of a source distribution is a collection of smaller sized distributions that are collectively equivalent to the source distribution with respect to the property of decomposability. That is, an arbitrary language is decomposable with respect to the source distribution if and only if it is decomposable with respect to each smaller sized distribution (in the reduction). The notion of reduction of distributions has previously been proposed to improve the complexity of decomposability verification. In this work, we address the problem of generating (optimal) reductions of distributions automatically. A (partial) solution to this problem is provided, which consists of 1) an incremental algorithm for the production of candidate reductions and 2) a reduction validation procedure. In the incremental production stage, backtracking is applied whenever a candidate reduction that cannot be validated is produced. A strengthened substitution-based proof technique is used for reduction validation, while a fixed template of candidate counter examples is used for reduction refutation; put together, they constitute our (partial) solution to the reduction verification problem. In addition, we show that a recursive approach for the generation of (small) reductions is easily supported.

Index Terms—discrete-event systems, decentralized supervisor synthesis, decomposability, co-observability, complexity

I. INTRODUCTION

A language $L \subseteq \Sigma^*$ is said to be decomposable with respect to a distribution, i.e., a tuple of non-empty sub-alphabets, of $\Sigma$ if $L$ is equal to the synchronous product of its projections onto the respective sub-alphabets. The notion of decomposability is well studied and applied in the literature on supervisory control theory, see for example [1], [2], [3], [4], [5], [6], and synthesis of synchronous products of transition systems [7]. As a special case of co-observability [6], the property of decomposability is already quite expensive to verify in general [1], [3]. Indeed, in general, it is quite unlikely that a polynomial time algorithm would exist since the problem of decomposability verification is PSPACE-complete, even when $L$ is prefix-closed (Theorem 4.24 in [7]). A basic problem of interest is to identify tractable fragments of the decomposability verification problem, based on structures of the distributions. The first such result is given in [3] for the verification of conditional decomposability, i.e., decomposability with a rather restricted structure imposed on the distributions. There are also some techniques that exploit both structures of the distributions and prefix-closeness of the test languages. The earliest such result that we know of appears in [1], which has been generalized in the Appendix of [8]. We shall not require prefix-closeness of the test languages in this work, to exploit solely the structures of distributions.

Recently, a notion of reduction of distributions [8], [9] has been proposed to improve the complexity of decomposability verification and support the parallel verification of decomposability, which is heavily inspired by (and generalizes) the result on polynomial time verification of conditional decomposability in [3]. Intuitively, a reduction of a given (source) distribution is a set of smaller sized distributions satisfying the property that an arbitrary language $L$ is decomposable with respect to the source distribution if and only if it is decomposable with respect to each smaller sized distribution in the reduction. If such a reduction exists, then we can verify the decomposability of $L$ with respect to every smaller sized distribution (in the reduction) to determine the decomposability of $L$ with respect to the source distribution. This divide-and-conquer approach allows us to reduce the verification complexity for a large class of distributions [3], [8]. Several basic challenges encountered in this approach include the problem of efficient production of candidate reductions and the problem of reduction verification. It is of considerable interest for us to find algorithmic solutions to these two basic problems, which can be used for automatic generation of (optimal) reductions. Optimal reductions of distributions then correspond to optimal reduction of verification complexity in our approach.

The main motivation for our study on the decomposability verification problem is its close relationship with the problem of existence of decentralized supervisors [2] (see for example [5], [6], [8]). In particular, a structural approach for improving the complexity of decomposability verification can lead to a better understanding of the complexity for deciding the existence of decentralized supervisors, in relation to the structures of the decentralized control architectures [8], [9]. A similar line of research for understanding the decidability of the decentralized supervisor synthesis problem appears in [10] and [11], which is based on some central structural results in trace theory [12]. The problem of reduction verification is related to the problem of finding decision-theoretic reduction functions (in descriptive complexity) [13], [8]. Indeed, verifying a candidate reduction of a distribution is equivalent to verifying the identity function to be a reduction (function) between two decision problems (of fixed format) about decomposability [8]. In general, however, even the problem of verifying whether the identity function is a reduction between two given decision problems is undecidable [14]. It is of interest to know whether the same conclusion also holds on our special problem setup.

1This statement assumes that reduction computation is not the bottleneck, which is often the case. In practice, we can run direct verification (w.r.t. $\Delta$) and the reduction-based verification in parallel and output the first answer that becomes available.

2The decomposability verification problem is a special case of the problem of existence of decentralized supervisors [8]: an instance of the first problem with prefix-closed test language $L \subseteq \Sigma^*$ and distribution $\Delta = (\Sigma_i)_{i=1}^{n}$ is an instance of the second problem with specification $L$, decentralized control architecture $(A_i)_{i=1}^{n}$, where $A_i = (\Sigma_i, X_i)$, and plant $G$, where $L(G) = L(G) = \Sigma^*$ (see Proposition 43 in [8] for more details).

3We conjecture that the problem of reduction verification is decidable.
Some basic results have been established in [8]. In this work, we continue to develop some results to address the following technical problems listed in [8], with a particular emphasis on Technical Problem 3:

1) Given any distribution $\Delta$ and any set $P$ of distributions, determine whether $P$ is a reduction of $\Delta$
2) Given any distribution $\Delta$ of $\Sigma$, determine whether it has a reduction
3) Given any distribution $\Delta$, compute an optimal reduction, if there exists a reduction
4) Given any set $P$ of distributions, determine the existence of a distribution $\Delta$ such that $P$ is a reduction of $\Delta$
5) Given any distribution $\Delta$ of $\Sigma$ and any reduction $P$ of $\Delta$, determine whether $P$ is compact, that is, whether $P$ involves no redundancy

The contributions of this study, compared with [8], are the following.

- We show that the two partially ordered sets studied in [8], i.e., the set of all distributions of an alphabet $\Sigma$ and the set of candidate reductions of a distribution $\Delta$, are indeed finite lattices, thus complete. Then, we can talk about the least upper bounds and the greatest lower bounds for sets of distributions and sets of candidate reductions.
- We then show that a set $P$ of distributions is a reduction of $\Delta$ only if $\Delta$ is the greatest lower bound of $P$. It then follows that Technical Problem 4 is reducible to Technical Problem 1; thus, we only need to call the reduction verification procedure for once to solve Technical Problem 4. We also show that the above order-theoretic necessary condition is equivalent to the necessary condition stated in Lemma 25 of [8] (reproduced as Lemma 10 in this work), which is based on the notion of generalized dependence relation.
- We implement a reduction verification procedure, which combines a strengthened version of the substitution-based (automated) proof technique [8] for reduction validation and the candidate counter examples based techniques for reduction refutation. Thus, given any candidate reduction $P$ of $\Delta$, if $\Delta$ can be obtained by using the strengthened substitution-based automated proof, then $P$ is a reduction of $\Delta$; and a fixed template of candidate counter examples is used to refute $P$, which turns out to be quite effective in refuting candidate reductions that are not valid.
- Structural results are also developed for verifying the decomposability of the fixed template of candidate counter examples, which can be used to show that some classes of distributions (e.g., distributions of ring-like structures) have no reductions.
- We develop an incremental algorithm for the production of candidate reductions, which is guided by Lemma 24 and Lemma 25 of [8] (Lemmas 9 and 10 in this work). It is tailored for the efficient production of a subclass of candidate reductions that are meet-consistent\(^{4}\). Backtracking will be enforced whenever a (meet-consistent) candidate reduction that cannot be validated is produced. The idea is to first explore small candidate reductions, i.e., candidate reductions of small width. Thus, it works quite effectively for those distributions which have small reductions. Since small candidate reductions often correspond to those high up elements\(^{5}\) in the lattice of the set of candidate reductions, this incremental production algorithm generates the candidate reductions from the higher up elements to those lower down ones in the lattice.
- For those distributions that do not have small reductions, the same reduction validation procedure may potentially be called for many times before a reduction is computed, using the above incremental production algorithm (with backtracking). To address this problem, we also develop a “conservative” recursive algorithm for computing reductions (guided by Theorem 22 of [8], which is reproduced as Theorem 12 in this work), by generating lower down candidate reductions in the lattice first. Once a candidate reduction is validated, it can be recursively improved by computing reductions of the constituent distributions, if the reductions exist.

While the decidability statuses of the main technical problems still remain unknown, we believe that this work constitutes a major step, compared with [8], towards practically solving the problems of reduction verification, determining the existence of reductions and automatic generation of optimal reductions. We have carried out a number of experiments on determining the existence of reductions. And the results obtained are quite encouraging (see Appendix B for a small list of examples), and we have not found examples where the reduction verification procedure fails to conclude\(^{6}\).

The rest of this paper will be organized as follows. Section II is devoted to preliminaries. In Section III, we provide several basic results. In Section IV, we then present the reduction verification procedure, which consists of both reduction validation and reduction refutation. Section V then addresses the problem of automatic generation of small or (near-)optimal reductions, where the central aim is to achieve efficient production of candidate reductions. We then draw the conclusions in Section VI. Some additional material and experimental results are included in the Appendices.

II. Preliminaries

We assume that the reader is familiar with basic theories of formal languages and finite automata [15]. Some knowledge about order theory [16], [17] is also assumed. In the following, additional notation and terminology are introduced, including some definitions and results from [8].

Let $[1, n]$ denote the set $\{1, 2, \ldots, n\}$. For any two sets $A$ and $B$, we write $A - B$ to denote the set-theoretic difference of $A$ and $B$. The cardinality of any set $A$ is denoted by $|A|$. For any given alphabet $\Sigma$, a distribution of $\Sigma$ of size $n$ is an $n$-tuple $\Delta = (\Sigma_1, \Sigma_2, \ldots, \Sigma_n)$ of non-empty sub-alphabets of $\Sigma$ such that $\Sigma = \bigcup_{i=1}^{n} \Sigma_i$ and the sub-alphabets are pairwise incomparable with respect to set inclusion. We sometimes also view the distribution $\Delta$ as the set $\{\Sigma_1, \Sigma_2, \ldots, \Sigma_n\}$ for convenience and use $|\Delta|$ to denote the size of $\Delta$. Given a distribution $\Delta = (\Sigma_1, \Sigma_2, \ldots, \Sigma_n)$ of $\Sigma$, we have $n$ projections $P_i$ from $\Delta$.

\(^{4}\)The property of meet-consistency shall be discussed later. It is a necessary condition for a candidate reduction to be a valid reduction of a distribution.

\(^{5}\)We have the Hasse diagram, where the top element lies in the top and the bottom element lies in the bottom. The high up elements lie close to the top element and the low down elements lie close to the bottom element.

\(^{6}\)The reduction verification problem is a central problem in the investigation of the decidability. We would like to know whether the reduction verification procedure is complete. We experiment on the easier problem of determining the existence of reductions, which can be reduced to the reduction verification problem [8].
\(\Sigma^*\) to \(\Sigma_i^*\) and \(n\) inverse projections \(P_i^{-1}\) from \(\Sigma_i^*\) to \(2^\Sigma\). Also, both projections and inverse projections are naturally extended to the mappings between languages. The *synchronous product* 
\[ \prod_{i=1}^n L_i \] 
of languages \(L_i\) over \(\Sigma_i\) is defined as \( \bigcap_{i=1}^n P_i^{-1}(L_i) \). 

A language \(L \subseteq \Sigma^*\) is said to be *decomposable* with respect to distribution \(\Delta = (\Sigma_1, \Sigma_2, \ldots, \Sigma_n)\) of \(\Sigma\) if \(L = \prod_{i=1}^n P_i(L_i)\). The straightforward approach for checking the decomposability of \(L\) with respect to \(\Delta\) is of worst case time complexity \(O((n+1)|\Sigma| - \sum_{i=1}^n |\Sigma_i|)m^{n+1}\), where \(m\) is the state size of the given deterministic finite automaton that recognizes \(L\). The *decomposition closure* \(\prod_{i=1}^n P_i(L)\) of \(L\) with respect to \(\Delta\) is denoted by \(L^\Delta\). Let

\[s_1 \shuffle s_2 = \{v_1u_1v_2u_2 \ldots v_ku_k \mid s_1 = v_1v_2 \ldots v_k, s_2 = u_1u_2 \ldots u_k, \text{where } v_i, u_i \in \Sigma^* \text{ for each } i \in [1, k]\}\]
denote the *shuffle* of any two strings \(s_1, s_2 \in \Sigma^*\). For example, the shuffle of strings \(s_1 = abb\) and \(s_2 = a\) is the language \(s_1 \shuffle s_2 = \{aab, abab, abba\}\).

Each distribution \(\Delta = (\Sigma_1, \Sigma_2, \ldots, \Sigma_n)\) induces a *dependence relation* \(D_\Delta \subseteq \Sigma \times \Sigma\), which is defined below.

\[D_\Delta := \{(a, b) \in \Sigma \times \Sigma \mid \exists i \in [1, n], a, b \in \Sigma_i\}\]

Clearly, \(D_\Delta\) is reflexive and symmetric. Then, the *complement* \(I_\Delta := \Sigma \times \Sigma - D_\Delta\) of \(D_\Delta\) is said to be the *independence relation* induced by \(\Delta\), which is irreflexive and symmetric. For any \((a, b) \in I_\Delta\), \(a\) and \(b\) are said to be *independent symbols*. We can use an undirected graph \((\Sigma, I_\Delta)\) for the visualization of an independence relation. For example, the independence relation \(I_\Delta = \{(a, c), (c, a), (b, c), (c, b)\}\) induced by the distribution \(\Delta = \{(a, b), (c)\}\) of \(\Sigma = \{a, b, c\}\) can be visualized in Fig. 1. We may also use the notation \(I_\Delta = \{(a, c), (b, c)\}\), where, for example, \((a, c)\) is used to represent both \((a, c)\) and \((c, a)\). Let \(\Sigma'\) be any non-empty sub-alphabet of \(\Sigma\). We define \(\Sigma' \subseteq \Delta\) if \(\exists i \in [1, n], \Sigma' \subseteq \Sigma_i\). Clearly, for any sub-alphabet \(\Sigma' \subseteq \Sigma\) with cardinality 1, \(\Sigma' \subseteq \Delta\) holds for any distribution \(\Delta\) of \(\Sigma\). Apparently, for any two symbols \(a, b \in \Sigma\), \(\{a, b\} \subseteq \Delta\) if and only if \((a, b) \in D_\Delta\). Thus, \(\Delta\) naturally generalizes the notion of a dependence relation.

Two strings \(s, s'\) over \(\Sigma\) are said to be *trace equivalent* with respect to the independence relation \(I_\Delta\), denoted by \(s \sim_{I_\Delta} s'\), if there exist strings \(v_0, v_1, \ldots, v_k\) such that \(s = v_0, s' = v_k\) and for each \(i \in [1, k]\), there exist some \(u_i, u'_i \in \Sigma^*\) such that \((a_i, b_i) \in I_\Delta\), \(v_{i-1} = u_iu'_i\) and \(v_i = u_iu'_i\). Intuitively, two strings are trace equivalent if each one of them can be obtained from the other by a sequence of permutations of adjacent symbols that are independent. And the set of trace equivalent strings of \(s\) for \(I_\Delta\) is said to be the *trace closure* of \(s\) with respect to \(I_\Delta\), denoted by \([s]_{I_\Delta}\). The *trace closure* \([L]_{I_\Delta}\) of a language \(L \subseteq \Sigma^*\) is defined to be the set \(\bigcup_{s \in L} [s]_{I_\Delta}\). A language \(L\) is said to be *trace-closed* with respect to \(I_\Delta\) if \(L = [L]_{I_\Delta}\).

For example, consider the independence relation \(I_\Delta = \{(a, c), (c, a), (b, c), (c, b)\}\) induced by \(\Delta = \{(a, b), (c)\}\). The strings \(bac\) and \(cba\) are trace equivalent with respect to \(I_\Delta\) since \(bac \sim_{I_\Delta} cba\). Let \(L = \{bac, cba, bca\}\) is trace-closed with respect to \(I_\Delta\) since we have \([L]_{I_\Delta} = L = \{bac, cba, bca\}\). We shall often identify each singleton with the unique element it contains.

We recall the following well-established result (see Proposition 4.3 in [18], where a detailed proof is given).

---

A distribution \(\Delta\) is said to be *reducible* if it has a reduction. It is said to be \((l, k)\)-reducible if it has a reduction \(P = \{\Delta_1, \Delta_2, \ldots, \Delta_l\}\), where \(\max_{i \in [1, l]} |\Delta_i| = k\); and the reduction \(P\) will be referred to as an \((l, k)\)-*reduction of \(\Delta\)*, in which \(l\) is called the *height*, \(k\) the width, and \((l, k)\) the *dimension*. The width is the main determining factor for the worst case time complexity, and the height of a reduction corresponds to the number of parallel processors needed in the parallel verification. A reduction of \(\Delta\) is said to be *compact* if none of its proper subsets is also a reduction of \(\Delta\). Formally, we say that a reduction \(P\) of distribution \(\Delta\) is *optimal*\(^8\) if for any other reduction \(P'\) of \(\Delta\), it holds that either 1) the width of \(P'\) is greater than the width of \(P\), or 2) if the width of \(P'\) is equal to the width of \(P\), then the height of \(P'\) is greater than or equal to the height of \(P\). Clearly, an optimal reduction is necessarily compact. The next example [8] shows that it is possible to have two or more compact reductions of the same width, but of different heights. Thus, a compact reduction is not necessarily optimal.

**Example 3.** Consider the distribution \(\Delta = (\Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4)\) of \(\{a, b, c, d, e, f\}\), where \(\Sigma_1 = \{a, b\}, \Sigma_2 = \{b, c\}, \Sigma_3 = \{d, e\}\) and \(\Sigma_4 = \{e, f\}\). It is known that \(\{\Delta'_1, \Delta'_2, \Delta'_3\}\) is a compact \((3, 2)\)-reduction of \(\Delta\) [8], where

\[
\Delta'_1 = \{(a, b), (b, c, d, e, f)\}
\]
\[
\Delta'_2 = \{(a, b, c), (d, e, f)\}
\]
\[
\Delta'_3 = \{(d, e), (a, b, c, e, f)\}
\]

and \(\{\Delta''_1, \Delta''_2\}\) is a compact \((2, 2)\)-reduction of \(\Delta\), where

\[
\Delta''_1 = \{(a, b, c, e, f), (b, c, d, e)\}
\]
\[
\Delta''_2 = \{(a, b, c), (d, e, f)\}
\]

\(\{\Delta''_1, \Delta''_2\}\) is an optimal reduction of \(\Delta\) of dimension \((2, 2)\). Then, we can conclude that \(\{\Delta'_1, \Delta'_2, \Delta'_3\}\) is not an optimal reduction of \(\Delta\).
A partial ordering $\leq_\Sigma$ on the set $\Delta(\Sigma)$ of all distributions of $\Sigma$ (lifting the partial ordering $\subseteq$) is defined as follows.

**Definition 4 ($\leq_\Sigma$ [8]).** Given any two distributions $\Delta$ and $\Delta'$ of $\Sigma$, we define $\Delta \leq_\Sigma \Delta'$ if $\forall i \in \Delta, \exists \Sigma_i' \in \Delta'$, $\Sigma_i \subseteq \Sigma_i'$. We have the following lemma.

**Lemma 5** ([8]). Let $\Delta$ and $\Delta'$ be any two distributions of $\Sigma$. Then, $\Delta \leq_\Sigma \Delta'$ if and only if for an arbitrary language $L \subseteq \Sigma^*$, the decomposability of $L$ with respect to $\Delta$ implies the decomposability of $L$ with respect to $\Delta'$. The next corollary follows from Lemma 5.

**Corollary 6** ([8]). Let $\{\Delta_1, \Delta_2, \ldots, \Delta_i\}$ be any reduction of $\Delta$. Then, for each $i \in [1, l]$, we have $\Delta \leq_\Sigma \Delta_i$.

A distribution $\Delta' = (\Sigma_1', \Sigma_2', \ldots, \Sigma_n')$ is said to be merged from $\Delta$ if $\Delta'$ can be obtained from $\Delta = (\Sigma_1, \Sigma_2, \ldots, \Sigma_n)$ by 1) merging the sub-alphabets according to a proper partition of $\{1, n\}$, and followed by 2) keeping only those maximal sub-alphabets in the resulting tuple [8]. This is illustrated by the next example [8].

**Example 7.** Consider the alphabet $\Sigma = \{a, b, c, d, e\}$ and the distribution $\Delta = (\Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4)$ of $\Sigma$, where $\Sigma_1 = \{a, b\}$, $\Sigma_2 = \{b, c\}$, $\Sigma_3 = \{c, d\}$ and $\Sigma_4 = \{d, e\}$. If we merge $\Sigma_1$ and $\Sigma_3$, which corresponds to the partition $\{\{1, 3\}, \{2\}, \{4\}\}$ of $\{1, 4\}$, then we obtain the tuple $\Sigma_1 \cup \Sigma_3, \Sigma_2, \Sigma_4)$. By keeping only those maximal sub-alphabets, we obtain the distribution $\Delta' = (\Sigma_1 \cup \Sigma_3, \Sigma_2, \Sigma_4)$ of $\Sigma$. Then, $\Delta'$ is merged from $\Delta$, by the above definition. For convenience, we say $\Delta'$ is obtained from $\Delta$ by merging $\Sigma_1$ and $\Sigma_3$. And the partition is denoted by $P_A(\Delta')$, that is, $P_A(\Delta') = \{\{1, 3\}, \{2\}, \{4\}\}$.

Let $M(\Delta)$ denote the set of distributions that are merged from $\Delta$. We only need to consider those candidate reductions in which the distributions all belong to $M(\Delta)$ and are pairwise $\leq_{\Sigma}$ incomparable [8]. Here, two distributions $\Delta, \Delta'$ are $\leq_{\Sigma}$ incomparable if $\Delta \not\leq_\Sigma \Delta'$ and $\Delta' \not\leq_\Sigma \Delta$. Then, let the search space for candidate reductions of $\Delta$ be $S_D := \{P \in 2^{\Delta(\Delta)} | \forall \Delta', \Delta'' \in P, (\Delta' \not\leq_\Sigma \Delta'' \Rightarrow \Delta' \not\leq_\Sigma \Delta'')\}$

Compared with [8], $\emptyset$ is included in $S_D$ here for the purpose of technical convenience (see Theorem 18).

The following partial ordering $\leq_{\Sigma}$ on $S_D$ (lifting the partial ordering $\leq_{\Sigma}$) is defined.

**Definition 8** ($\leq_{\Sigma}$ [8]). Given any two elements $P, P'$ in $S_D$, we define $P \leq_{\Sigma} P'$ if $\forall \Delta_j' \in P', \exists \Delta_i \in P, \Delta_i \leq_\Sigma \Delta_j'$.

We assume $M(\Delta)$ is non-empty, since otherwise $\Delta$ cannot have a reduction. In the partial order $(S_D, \leq_{\Sigma})$, the bottom\(^9\) element exists and is indeed equal to $[M(\Delta)] \in S_D$ [8]. Here, we have used the notation that, for any set $P$ of distributions, $[P]$ stands for the set of distributions that is obtained from $P$ by keeping only those minimal distributions (in terms of the partial ordering $\leq_\Sigma$). A distribution $\Delta'$ is said to be minimally merged from $\Delta$ if it can be obtained from $\Delta$ by merging two sub-alphabets $\Sigma_i$ and $\Sigma_j$ of $\Delta$, where $i \neq j$. And let $\bot(\Delta) \subseteq M(\Delta)$ denote the set of minimally merged distributions from $\Delta$. $\bot(\Delta)$ is also the bottom element of $S_D$, since $[M(\Delta)] = [\bot(\Delta)]$ [8].

The next two lemmas provide necessary conditions for a set (in $S_D$) of distributions to be a reduction of $\Delta$. Lemma 10 is a generalization of Lemma 9, since $\subseteq$ naturally generalizes the notion of a dependence relation.

**Lemma 9** ([8]). If $\{\Delta_1, \Delta_2, \ldots, \Delta_l\}$ is a reduction of $\Delta$, then $I_\Delta = \bigcup_{i \in [1, l]} I_{\Delta_i}$, or, equivalently, $D_\Delta = \bigcap_{i \in [1, l]} D_{\Delta_i}$.

**Lemma 10** ([8]). Let $\Delta$ be any distribution of $\Sigma$ and let $\Delta_i$ be any distribution of $\Sigma$ for each $i \in [1, l]$. If $\{\Delta_1, \Delta_2, \ldots, \Delta_l\}$ is a reduction of $\Delta$, then for each subset $\Sigma_i$ of $\Sigma$ of cardinality at least 2, $\forall i \in [1, l], \Sigma_i \subseteq \Delta_i \iff \Sigma_i \subseteq \Delta$.

The following lemma states that the set of reductions of any given distribution $\Delta$ within $S_D$ is indeed a downward closed set with respect to the partial ordering $\leq_{\Sigma}$.

**Lemma 11** ([8]). Let $P$ and $Q$ be any two elements of $S_D$. If $P \leq_{\Sigma} Q$ and $Q$ is a reduction of $\Delta$, then $P$ is also a reduction of $\Delta$.

Since $\bot(\Delta)$ is the bottom element of $S_D$, we then have the next result, by Lemma 11.

**Theorem 12** ([8]). Let $\Delta$ be any distribution of $\Sigma$. Then, $\Delta$ has a reduction if and only if $\bot(\Delta)$ is a reduction of $\Delta$.

Thus, to determine the existence of a reduction for any given distribution $\Delta$, we only need to check whether the set $\{\bot(\Delta)\}$ is a reduction (of $\Delta$). That is, Technical Problem 2 is reduced to Technical Problem 1, i.e., we only need to call the reduction verification procedure for once.

**Example 13** ([8]). For the distribution $\Delta$ in Example 3, we have $\bot(\Delta) = \{\Delta_{34}, \Delta_{12}, \Delta_{13}, \Delta_{14}, \Delta_{23}, \Delta_{24}\}$, where

\[
\begin{align*}
\Delta_{14} &:= \{(a, b), \{b, c\}, \{d, e, f\}\} \\
\Delta_{12} &:= \{(d, e), \{e, f\}, \{a, b\}\} \\
\Delta_{13} &:= \{(a, b, d, e), \{b, c\}, \{e, f\}\} \\
\Delta_{14} &:= \{(a, b, e, f), \{b, c\}, \{d, e\}\} \\
\Delta_{23} &:= \{\{b, c, d, e\}, \{a, b\}, \{e, f\}\} \\
\Delta_{24} &:= \{\{b, c, e, f\}, \{a, b\}, \{d, e\}\}
\end{align*}
\]

Here $\Delta_i$ denotes the distribution that is obtained from $\Delta$ by merging $\Sigma_i$ and $\Sigma_j$. In this example, distributions in $\bot(\Delta)$ are $\leq_{\Sigma}$ incomparable. Theorem 12 implies that the distribution $\Delta$ in Example 3 has a reduction if and only if $[\bot(\Delta)] = \bot(\Delta)$ shown above is a reduction.

### III. Some Basic Results

In this section, we shall show that $(\Delta(\Sigma), \leq_{\Sigma})$ and $(S_D, \leq_{\Sigma})$ are indeed finite lattices, thus complete. We shall also provide an order-theoretic necessary condition for a set of distributions to be a reduction (of a source distribution), which will make it obvious that Technical Problem 4 can be reduced to Technical Problem 10.1, i.e., we only need to call the reduction verification procedure for once.

\(^9\)This claim can be verified. It is also an easy consequence of Theorem 18, which will be shown later.
A. Ordering on Distributions

It is not difficult to check that the relation $\preceq_\Sigma$ is a partial ordering on $\Delta(\Sigma)$. Indeed, we shall now show that $(\Delta(\Sigma), \preceq_\Sigma)$ is a finite lattice.

**Theorem 14.** $(\Delta(\Sigma), \preceq_\Sigma)$ is a finite lattice.

**Proof:** It is not difficult to see that $(\Delta(\Sigma), \preceq_\Sigma)$ is a partial order. In the partial order $(\Delta(\Sigma), \preceq_\Sigma)$,

1. the meet $\Delta \land \Delta'$ of any two distributions $\Delta$, $\Delta'$ always exists, and it is defined to be the distribution $\Delta''$ that is obtained from the set $\{\Sigma_i \cap \Sigma'_j \mid \Sigma_i \in \Delta, \Sigma'_j \in \Delta'\}$ by keeping only those maximal sub-alphabets. We now show the following:
   a) $\Delta'' \preceq_\Sigma \Delta$ and $\Delta'' \preceq_\Sigma \Delta'$: for any $\Sigma_i \cap \Sigma'_j \in \Delta''$, since $\Sigma_i \cap \Sigma'_j \subseteq \Sigma_i \in \Delta$, we have $\Delta'' \preceq_\Sigma \Delta$; and similarly, $\Delta'' \preceq_\Sigma \Delta'$.
   b) for any distribution $\Delta''$, if $\Delta'' \preceq_\Sigma \Delta$ and $\Delta'' \preceq_\Sigma \Delta'$, then $\Delta'' \preceq_\Sigma \Delta$.

2. the join $\Delta \lor \Delta'$ of any two distributions $\Delta$, $\Delta'$ always exists, and it is defined to be the distribution $\Delta''$ whose components are exactly the maximal elements (in terms of set inclusion) of $\Delta \cup \Delta'$. We now show the following:
   a) $\Delta \preceq_\Sigma \Delta''$ and $\Delta' \preceq_\Sigma \Delta''$: this is straightforward from the definition of $\Delta''$ and the definition of $\preceq_\Sigma$. Indeed, for any $\Sigma_i \in \Delta$, if $\Sigma_i \subseteq \Delta''$, then $\Sigma_i \subseteq \Sigma_i \subseteq \Delta$; otherwise, by the definition of $\Delta''$, there exists some $\Sigma''_j \subseteq \Delta''$ such that $\Sigma_i \subseteq \Sigma''_j \subseteq \Delta''$. Thus, $\Sigma_i \subseteq \Delta''$,
   b) for any distribution $\Delta''$, if $\Delta \preceq_\Sigma \Delta''$, then $\Delta'' \preceq_\Sigma \Delta$.

We have the next result, which strengthens Corollary 6: $\Delta$ not only is a lower bound of its reduction $\{\Delta_1, \Delta_2, \ldots, \Delta_l\}$, but it is also the greatest lower bound.

**Proposition 15.** Let $\{\Delta_1, \Delta_2, \ldots, \Delta_l\}$ be any reduction of $\Delta$. We have $\Delta = \bigwedge_{i=1}^l \Delta_i$.

**Proof:** By Theorem 14, $\bigwedge_{i=1}^l \Delta_i$ exists. By Corollary 6, we have $\Delta \preceq_\Sigma \bigwedge_{i=1}^l \Delta_i$, since $\{\Delta_1, \Delta_2, \ldots, \Delta_l\}$ is a reduction of $\Delta$. For any language $L \subseteq \Sigma'$ that is decomposable with respect to distribution $\bigwedge_{i=1}^l \Delta_i$, we conclude that $L$ is also decomposable with respect to each $\Delta_j$, by Lemma 5, since $\bigwedge_{j=1}^l \Delta_j \preceq_\Sigma \Delta_j$ for each $j \in [1, l]$. Since $\{\Delta_1, \Delta_2, \ldots, \Delta_l\}$ is a reduction of $\Delta$, we then conclude that $L$ is decomposable with respect to $\Delta$. That is, for any language $L \subseteq \Sigma'$ that is decomposable with respect to $\bigwedge_{i=1}^l \Delta_i$, $L$ is decomposable with respect to $\Delta$. By Lemma 5, we then have $\bigwedge_{i=1}^l \Delta_i \preceq_\Sigma \Delta$. Thus, $\Delta = \bigwedge_{i=1}^l \Delta_i$.

Proposition 15 provides another necessary condition for a set of distributions to be a reduction of a source distribution $\Delta$. It is of interest to compare which one is stronger, Lemma 10 or Proposition 15. Indeed, they are equivalent.

**Theorem 16.** Let $\Delta$ be any distribution of $\Sigma$ and let $\Delta_i$, be any distribution of $\Sigma$, for each $i \in [1, l]$, $\Delta = \bigwedge_{i=1}^l \Delta_i$ if and only if for each subset $\Sigma'$ of $\Sigma$ of cardinality at least 2, $(\forall i \in [1, l], \Sigma' \preceq_\Sigma \Delta_i) \Rightarrow \Sigma' \preceq_\Sigma \Delta$.

**Proof:** Suppose $\Delta = \bigwedge_{i=1}^l \Delta_i$. Let $\Sigma'$ be any subset of $\Sigma$ with $|\Sigma'| \geq 2$. Suppose for each $i \in [1, l], \Sigma' \preceq_\Sigma \Delta_i$. Then we have $\Sigma' \subseteq \bigwedge_{i=1}^l \Delta_i = \Delta$, by the definitions of $\subseteq$ and the meet operation $\land$. The direction $\Sigma' \preceq_\Sigma \Delta \Rightarrow (\forall i \in [1, l], \Sigma' \preceq_\Sigma \Delta_i)$ is automatic from $\Delta = \bigwedge_{i=1}^l \Delta_i \preceq_\Sigma \Delta_i$, for each $j \in [1, l]$. On the other hand, suppose that for each subset $\Sigma'$ of $\Sigma$ of cardinality at least 2, $(\forall i \in [1, l], \Sigma' \subseteq \Delta_i) \Rightarrow \Sigma' \subseteq \Delta$. We shall first show that $\Delta \preceq_\Sigma \bigwedge_{i=1}^l \Delta_i$, for each $i \in [1, l]$. This is not difficult. Indeed, for any $\Sigma' \subseteq \Delta$, we have $\Sigma' \subseteq \bigwedge_{i=1}^l \Delta_i$ by definition. We conclude $\Sigma' \subseteq \Delta$, for each $i \in [1, l]$, i.e., for each $\Sigma' \subseteq \Delta$, there exists $\Sigma' \subseteq \Delta_i$ such that $\Sigma' \subseteq \Sigma_i$. Thus, we have $\Delta \preceq_\Sigma \bigwedge_{i=1}^l \Delta_i$. For any $\Sigma' \subseteq \bigwedge_{i=1}^l \Delta_i$, by the definitions of $\subseteq$ and $\bigwedge$, we have $\Sigma' \subseteq \Delta_i$ for each $i \in [1, l]$. Thus, we conclude $\Sigma' \subseteq \Delta_i$. That is, for any $\Sigma' \subseteq \bigwedge_{i=1}^l \Delta_i$, there exists $\Sigma' \subseteq \Delta$ such that $\Sigma' \subseteq \Sigma_i$. We have $\bigwedge_{i=1}^l \Delta_i \preceq_\Sigma \Delta$. Thus, $\Delta = \bigwedge_{i=1}^l \Delta_i$.

Another immediate consequence of Proposition 15 is that a set of distributions can only be the reduction of their greatest lower bound in the lattice $(\Delta(\Sigma), \preceq_\Sigma)$. Thus, Technical Problem 4 can be reduced to Technical Problem 1 in the following way: $\{\Delta_1, \Delta_2, \ldots, \Delta_l\}$ is a reduction of some distribution if and only if $\{\Delta_1, \Delta_2, \ldots, \Delta_l\}$ is a reduction of $\bigwedge_{i=1}^l \Delta_i$.

The necessary condition given in Lemma 10 is essentially order-theoretic, as has been shown in Theorem 16. It turns out that it is also possible to reformulate the necessary condition provided in Lemma 9. The key is the following.

**Lemma 17.** For any set $\{\Delta_1, \Delta_2, \ldots, \Delta_l\}$ of distributions of $\Sigma$, we have $\bigcup_{i \in [1, l]} \Delta_i = I_{\bigwedge_{i=1}^l \Delta_i}$.

**Proof:** Since $\bigwedge_{i=1}^l \Delta_i \preceq_\Sigma \Delta_i$, for each $i \in [1, l]$, it holds that $D_{\bigwedge_{i=1}^l \Delta_i} \subseteq D_{\Delta_i}$. Thus, it holds that $I_{\Delta_i} \subseteq I_{\bigwedge_{i=1}^l \Delta_i}$ for each $i \in [1, l]$. We then have $\bigcup_{i \in [1, l]} I_{\Delta_i} \subseteq I_{\bigwedge_{i=1}^l \Delta_i}$.

On the other hand, let $(a, b) \in I_{\bigwedge_{i=1}^l \Delta_i}$ be any independent symbols for $\bigwedge_{i=1}^l \Delta_i$. We must have that $(a, b) \in \bigcup_{i \in [1, l]} I_{\Delta_i}$. Indeed, suppose on the contrary that $(a, b) \not\in I_{\Delta_i}$, for each $i \in [1, l]$. It follows
that \( \{a, b\} \subseteq \bigwedge_{i=1}^{l} \Delta_i \) and thus \( (a, b) \notin I_{\bigwedge_{i=1}^{l} \Delta_i} \), which is a contradiction.

By Lemma 17, \( \bigcup_{i \in [1, l]} I_{\Delta_i} = I_\Delta \) if and only if \( I_{\bigwedge_{i=1}^{l} \Delta_i} = I_\Delta \) if and only if \( D_{\bigwedge_{i=1}^{l} \Delta_i} = D_\Delta \). So, we can view the gap between Lemma 9 and Lemma 10 from a different perspective. Lemma 10 requires the equality \( \bigwedge_{i=1}^{l} \Delta_i = \Delta \), and Lemma 9 only requires the dependence relation (resp., the independence relation) induced by \( \bigwedge_{i=1}^{l} \Delta_i \) to be equal to the dependence relation (resp., the independence relation) induced by \( \Delta \).

**B. Ordering on Candidate Reductions**

It is not difficult to check that the relation \( \leq_\Delta \) is a partial ordering. Indeed, we shall now show that \( (S_\Delta, \leq_\Delta) \) is a finite lattice. The proof follows closely that of Theorem 14.

**Theorem 18.** \( (S_\Delta, \leq_\Delta) \) is a finite lattice.

**Proof:** It is not difficult to check that \( (S_\Delta, \leq_\Delta) \) is a partial order. In the partial order \( (S_\Delta, \leq_\Delta) \),

1) the meet \( P \land P' \) of any two elements \( P, P' \in S_\Delta \) always exists, and it is defined to be the element \( \{P \cup P'\} \in S_\Delta \).

Recall that \( P'' = \{P \cup P'\} \in S_\Delta \) is obtained from \( P \cup P' \) by keeping only the minimal distributions (in terms of the partial ordering \( \leq_\Sigma \)). We now show the following.

a) \( P'' \leq_\Delta P \) and \( P'' \leq_\Delta P' \): this is straightforward from the definition of \( P'' \) and the definition of \( \leq_\Delta \). Indeed, for any \( \Delta_i \in P \), if \( \Delta_i \in P'' \), then \( \Delta_i \leq_\Sigma \Delta_i' \); otherwise, by the definition of \( P'' \), there must exist some \( \Delta_i' \in P'' \) such that \( \Delta_i' \leq_\Sigma \Delta_i \) (\( \Delta_i \notin P'' \)), it must have been removed due to the presence of some \( \Delta_i' \in P'' \) where \( \Delta_i' \leq_\Sigma \Delta_i \). Thus, for any \( \Delta_i \in P \), there exists some \( \Delta_i'' \in P'' \) such that \( \Delta_i'' \leq_\Sigma \Delta_i' \). That is, \( P'' \leq_\Delta P \). Similarly, we have \( P'' \leq_\Delta P' \).

b) for any \( P'' \in S_\Delta \), if \( P'' \leq_\Delta P \) and \( P'' \leq_\Delta P' \), then \( P'' \leq_\Delta P'' \): this is straightforward from the definition of \( P'' \) and the definition of \( \leq_\Delta \). Indeed, for any \( \Delta_i'' \in P'' \), it holds that \( \Delta_i'' \in P \) or \( \Delta_i'' \in P' \), by the definition of \( P'' \). If \( \Delta_i'' \in P \), then there exists some \( \Delta_i''' \in P'' \) such that \( \Delta_i''' \leq_\Sigma \Delta_i'' \), since \( P'' \leq_\Delta P \). If \( \Delta_i'' \in P' \), then there exists \( \Delta_i''' \in P'' \) such that \( \Delta_i''' \leq_\Sigma \Delta_i'' \), since \( P'' \leq_\Delta P' \). That is, for any \( \Delta_i'' \in P'' \), there exists some \( \Delta_i''' \in P'' \) such that \( \Delta_i''' \leq_\Sigma \Delta_i'' \). Thus, \( P'' \leq_\Delta P'' \).

2) the join \( P \lor P' \) of any two elements \( P, P' \in S_\Delta \) always exists, and it is defined to be the element \( P'' \in S_\Delta \) which is obtained from the join of only those minimal distributions (in terms of the partial ordering \( \leq_\Sigma \)). We now show the following.

a) \( P \leq_\Sigma P'' \) and \( P' \leq_\Sigma P'' \): this is straightforward from the definition of \( P'' \) and the definition of \( \leq_\Sigma \). Indeed, for any \( \Delta_i'' \in P'' \), we have \( \Delta_i'' \leq_\Sigma \Delta_i''' \leq_\Sigma \Delta_i'' \) for some \( \Delta_i \in P \) and \( \Delta_i \in P' \). It then follows that \( \Delta_i \leq_\Sigma \Delta_i'' \\( \Delta_i' \leq_\Sigma \Delta_i'' \). Thus, \( P \leq_\Sigma P'' \) and \( P' \leq_\Sigma P'' \).

b) for any \( P'' \in S_\Delta \), if \( P \leq_\Sigma P'' \) and \( P' \leq_\Sigma P'' \), then \( P'' \leq_\Sigma P'' \): this is straightforward from the definition of \( P'' \) and the definition of \( \leq_\Sigma \). Indeed, for any \( \Delta_i'' \in P'' \), there exists some \( \Delta_i''' \in P'' \) such that \( \Delta_i''' \leq_\Sigma \Delta_i'' \\( \Delta_i''' \leq_\Sigma \Delta_i'' \). Therefore, \( \Delta_i''' \leq_\Sigma \Delta_i'' \), and thus \( P'' \leq_\Sigma P'' \).

As an application, the greatest lower bound of \( S_\Delta \) is equal to \( \bigwedge_{\Delta \in S_\Delta} P = \{\bigcup_{\Delta \in S_\Delta} \} \in [M(\Delta)] \), since we have \( \{\Delta\} \in S_\Delta \) for each \( \Delta \in [M(\Delta)] \). Thus, with Lemma 11, we obtain the result that \( \Delta \) has a reduction if and only if \( [M(\Delta)] \) is a reduction of \( \Delta \) (cf. Theorem 12). And this is in fact the key result behind the conclusion that Technical Problem 2 can be reduced to Technical Problem 1.

**IV. REDUCTION VERIFICATION**

In this section, we address the problem of reduction verification. In Section IV-A, we shall first develop a technique for reduction refutation, by using candidate counter examples. In Section IV-B, we will then focus on reduction validation. The reduction verification procedure is then given in Section IV-C.

**A. Candidate Counter Examples for Reduction Refutation**

Suppose a candidate reduction \( P \) is not a reduction of the distribution \( \Delta \). Then, we know that it is possible to construct a finite language counter example to refute \( P \) [8] with a possible reduced finite language over \( \Sigma \) such that it is decomposable with respect to each distribution in \( P \) but not decomposable with respect to \( \Delta \). In this section, we provide a template \( L_{\text{cand}} \) of candidate counter examples for any distribution \( \Delta \), which turns out to be quite effective in refuting candidate reductions that are not valid. We first need the next result, which has been shown in the proof of Proposition 29 in [8].

**Lemma 19.** If \( P \in S_\Delta \) is not a reduction of \( \Delta \), then every counter example for refuting \( P \) is trace-closed with respect to \( I_\Delta \) but not decomposable with respect to \( \Delta \).

Thus, as a candidate counter example \( L_{\text{cand}} \) of \( P \), we need to ensure that \( L_{\text{cand}} \) is indeed trace-closed with respect to \( I_\Delta \) but not decomposable with respect to \( \Delta \). For any alphabet \( \Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_{|\Sigma|}\} \) and any distribution \( \Delta = (\Delta_1, \Delta_2, \ldots, \Delta_n) \) of \( \Sigma \), we generate the following set \( L(\Sigma_j) \) of strings for each \( j \in [1, n] \).

\[
L(\Sigma_j) := a^h_1(\Sigma_j) \cup a^h_2(\Sigma_j) \cup \ldots \cup a^h_{|\Sigma_j|}(\Sigma_j)
\]

where, for each \( i \in [1, |\Sigma_j|] \), \( h_j(i) := 1 \) if \( \sigma_i \in \Sigma_j \), \( h_j(i) := j + 1 \) if \( \sigma_i \notin \Sigma_j \). Now, let \( L_{\text{cand}} = \bigcup_{j=1}^n L(\Sigma_j) \). We have the following.

**Lemma 20.** \( L_{\text{cand}} \) is trace-closed with respect to \( I_\Delta \) but not decomposable with respect to \( \Delta \).

**Proof:** By the construction, each \( L(\Sigma_j) \) is trace-closed with respect to \( I_\Delta \). Thus, \( L_{\text{cand}} \) is also trace-closed with respect to \( I_\Delta \). Let \( s_0 := \sigma_1 \sigma_2 \ldots \sigma_{|\Sigma|} \). By definition, \( s_0 \notin L_{\text{cand}} \). It is clear that, for each \( j \in [1, n] \), \( P_j(s_0) \in P_j(L(\Sigma_j)) \). Thus, \( s_0 \in \bigcup_{j=1}^n P_j(s_0) \subseteq \bigcup_{j=1}^n P_j(L(\Sigma_j)) \subseteq \bigcup_{j=1}^n P_j(L_{\text{cand}}) \).
Thus, $s_0$ is a witness of the fact that $L_{\text{cand}}$ is not decomposable with respect to $\Delta$.

We now use the next example to illustrate the construction of $L_{\text{cand}}$.

**Example 21.** Consider distribution $\Delta = (\Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4)$ of $\Sigma = \{a, b, c, d\}$, where $\Sigma_1 = \{a, b\}$, $\Sigma_2 = \{b, c\}$, $\Sigma_3 = \{c, d\}$ and $\Sigma_4 = \{d, a\}$. Then, $L_{\text{cand}} = (a \cup b \cup c \cup d^2) \cup (a^2 \cup b \cup c \cup d) \cup (a^3 \cup b^2 \cup c \cup d) \cup (a \cup b^2 \cup c^2 \cup d)$

We will first present an application of the candidate counter example $L_{\text{cand}}$ to a distribution of ring-like structure of size 5.

**Example 22.** Consider distribution $\Delta = (\Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \Sigma_5)$ of $\Sigma = \{a, b, c, d, e\}$, where $\Sigma_1 = \{a, b\}$, $\Sigma_2 = \{b, c\}$, $\Sigma_3 = \{c, d\}$, $\Sigma_4 = \{d, e\}$ and $\Sigma_5 = \{e, a\}$. We can use $L_{\text{cand}}$ to show that $\Delta$ has no reduction. Indeed, we only need to show that $[\perp (\Delta)] = \{\Delta_1, \Delta_2, \Delta_3, \Delta_4, \Delta_5\}$ is not a reduction of $\Delta$, by Theorem 12, where

$$
\begin{align*}
\Delta_1 &= \{(a, b, c), (c, d), (d, e), (e, a)\} \\
\Delta_2 &= \{(b, c, d), (a, b, c), (d, e), (e, a)\} \\
\Delta_3 &= \{(c, d, e), (a, b, c), (d, e), (e, a)\} \\
\Delta_4 &= \{(d, e, a), (a, b, c), (d, e), (e, a)\} \\
\Delta_5 &= \{(a, b, c), (b, c, d), (c, d), (d, e)\}
\end{align*}
$$

By Lemma 20, we only need to show $L_{\text{cand}}$ is decomposable with respect to each $\Delta_i$ in $[\perp (\Delta)]$. In this case, $L_{\text{cand}} = (a \cup b \cup c \cup d^2) \cup (a^2 \cup b \cup c \cup d) \cup (a^3 \cup b^2 \cup c \cup d) \cup (a^4 \cup b^4 \cup c \cup d) \cup (a^5 \cup b^5 \cup c \cup d) \cup (a^6 \cup b^6 \cup c \cup d)$

It is not difficult\(^{14}\) to verify that $L_{\text{cand}}$ is indeed decomposable with respect to each distribution $\Delta_i$ in $[\perp (\Delta)]$. Thus, $[\perp (\Delta)]$ is not a reduction of $\Delta$, and $\Delta$ has no reduction. We note that, in this example, Proposition 15 fails to refute $[\perp (\Delta)]$.

One may be tempted to think that the reason why the above distribution $\Delta$ has no reduction is because each symbol in $\Sigma$ is shared and the coupling between sub-alphabets in $\Delta$ is tight. This is not entirely correct, as can be seen in the next example.

**Example 23.** Consider the distribution $\Delta = (\Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4)$ of $\Sigma = \{a, b, c, d\}$ of size 4, where $\Sigma_1 = \{a, b\}$, $\Sigma_2 = \{b, c\}$, $\Sigma_3 = \{c, d\}$ and $\Sigma_4 = \{d, a\}$. $\Delta$ is of ring-like structure of size 4. $\Delta$ does not have a reduction. This can be shown by using $L_{\text{cand}}$ on $[\perp (\Delta)]$ again, as in Example 22. On the other hand, consider the distribution $\Delta' = (\Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \Sigma_5)$ of $\Sigma = \{a, b, c, d\}$ of size 5, where $\Sigma_1 = \{a, b\}$, $\Sigma_2 = \{b, c\}$, $\Sigma_3 = \{c, d\}$, $\Sigma_4 = \{d, a\}$ and $\Sigma_5 = \{e, a\}$. It is straightforward to see, by using the substitution-proof technique (see [8] or Section IV-B for more details), that $\{\Delta_1, \Delta_2\}$ is a (2, 3)-reduction of $\Delta'$, where

$$
\begin{align*}
\Delta_1 &= \{(a, b), (b, c), (a, c, d)\} \\
\Delta_2 &= \{(a, b), (b, c), (c, d), (d, a)\}
\end{align*}
$$

In the remaining of this section, we will study the following two related problems.

A) Given any candidate reduction $P$ of $\Delta$, which we would like to refute via $L_{\text{cand}}$, we only need to show $L_{\text{cand}}$ is decomposable with respect to each distribution $\Delta_i$ in $P$. This can be achieved by directly applying the definition of decomposability, that is, by checking $L_{\text{cand}} = L_{\text{cand}}^\downarrow$. Since $L_{\text{cand}}$ has a very special structure, it is possible\(^{15}\) to discharge the verification obligations by using some efficient reasoning rules. This may potentially lead to a more manageable approach to verify the decomposability of $L_{\text{cand}}$, especially for the more difficult parameterized setup. B) will provide such an example, where we need to reason about the decomposability for an infinite number of $L_{\text{cand}}$’s, to show a family of distributions of ring-like structures do not have reductions. However, one could come up with several different reasoning rules (or, explanations) for the decomposability of $L_{\text{cand}}$. It is of much interest to develop an intuitive reasoning rule that is also powerful: If $L_{\text{cand}}$ is decomposable with respect to $\Delta_i$’s, then the reasoning rule shall often conclude it. Preferably, $L_{\text{cand}}$ is abstracted away in the analysis and we only need to work with sub-alphabets in $\Delta$ and $\Delta_i$’s, that is, explore structures of the distributions. Indeed, we believe that this can lead to a better understanding of the reason why a distribution does not have a reduction.

To provide the intuition, we shall now analyze why $L_{\text{cand}}$ is decomposable with respect to each $\Delta_i$ in Example 22. We only need to explain the decomposability of $L_{\text{cand}}$ with respect to distribution $\Delta_1$; for other distributions $\Delta_i$’s, the explanation is similar.

**Example 24.** Consider the distributions $\Delta$ and $\Delta_1$ given in Example 22. We recall here that $\Delta = (\Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \Sigma_5)$, where $\Sigma_1 = \{a, b\}$, $\Sigma_2 = \{b, c\}$, $\Sigma_3 = \{c, d\}$, $\Sigma_4 = \{d, e\}$ and $\Sigma_5 = \{e, a\}$. And, $\Delta_1 = \{(a, b, c), (c, d), (d, e), (e, a)\}$. We have $L_{\text{cand}} = L(\Sigma_1) \cup L(\Sigma_2) \cup L(\Sigma_3) \cup L(\Sigma_4) \cup L(\Sigma_5) = (a \cup b \cup c \cup d^2) \cup (a^2 \cup b \cup c \cup d) \cup (a^3 \cup b^2 \cup c \cup d) \cup (a^4 \cup b^4 \cup c \cup d)$

To verify the decomposability of $L_{\text{cand}}$ with respect to $\Delta_1$, we only need to check whether $L_{\text{cand}} \subseteq L_{\text{cand}}$. The decomposition closure $L_{\text{cand}}$ of $L_{\text{cand}}$ is by definition equal to

$$
\bigcup_{\text{Dom}} P_{(a,b,c)}(s_1) \| P_{(c,d)}(s_2) \| P_{(a,c)}(s_3) \| P_{(e,a)}(s_4)$$

where $\text{Dom} := \bigwedge_{i \in [1, 4]} s_i \in L_{\text{cand}}$ and we have used $P_{(e,a)}$ to denote the projection from $\Sigma^* \setminus \Sigma'$ for $\Sigma' \subseteq \Sigma$. We only need to show that, for any $s_1, s_2, s_3, s_4$ that satisfies $\text{Dom}$, if

$$P_{(a,b,c)}(s_1) \| P_{(c,d)}(s_2) \| P_{(a,c)}(s_3) \| P_{(e,a)}(s_4) \neq \emptyset,$$

then we must have

$$P_{(a,b,c)}(s_1) \| P_{(c,d)}(s_2) \| P_{(a,c)}(s_3) \| P_{(e,a)}(s_4) \subseteq L_{\text{cand}}.$$

We shall perform case analysis on $s_1 \in L_{\text{cand}}$; and there are five cases that correspond to $s_1 \in L(\Sigma_j)$, where $j \in [1, 5]$.

\(^{14}\)We shall provide an analysis later in Example 24.

\(^{15}\)We shall only show two reasoning rules (see Theorem 26 in this section and Theorem 44 in Appendix A, which is an improvement of Theorem 26).
We shall only analyze the cases \( s_1 \in L(\Sigma_1) \) and \( s_1 \in L(\Sigma_2) \); and the rest cases can be analyzed in a similar way.

(\( s_1 \in L(\Sigma_1) \)): Suppose we select \( s_1 \in L(\Sigma_1) = a \, \urcorner \, b \, \urcorner \, c^2 \, \urcorner \, d^2 \, \urcorner \, e^3 \), then we have \( P_{\{a,b,c\}}(s_1) = a \, \urcorner \, b \, \urcorner \, c, d,e \). We then need to select \( s_2, s_3, s_4 \) so that

\[
P_{\{a,b,c\}}(s_1) \cap P_{\{c,d\}}(s_2) \cap P_{\{d,e\}}(s_3) \cap P_{\{e,a\}}(s_4) \neq \emptyset.
\]

This determines that the only choice for \( s_2 \) is to choose \( s_2 \in L(\Sigma_1) = a \, \urcorner \, b \, \urcorner \, c \, \urcorner \, d \, \urcorner \, e^2 \), since any other choice of \( s_2 \) does not contain exactly two occurrences of the symbol \( c \) which matches \( c^2 \) in \( P_{\{a,b,c\}}(s_1) \). Then, \( P_{\{c,d\}}(s_2) \in c^2 \, \urcorner \, d^2 \). Then, we need to select \( s_3, s_4 \) so that

\[
P_{\{a,b,c\}}(s_1) \cap P_{\{c,d\}}(s_2) \cap P_{\{d,e\}}(s_3) \cap P_{\{e,a\}}(s_4) \neq \emptyset.
\]

Thus, in this case, \( P_{\{a,b,c\}}(s_1) \) cannot be used to determine symbol \( d \) by matching \( c \). Instead, in these cases, \( P_{\{a,b,c\}}(s_1) \) can be used to first determine symbol \( c \), through \( P_{\{a,b\}}(s_4) \), by matching \( a \) (after the matching, \( s_1 \) and \( s_4 \) belong to the same \( L(\Sigma_j) \), \( j = 2, 3 \)). Then, \( P_{\{a,e\}}(s_4) \) can be used to determine the symbol \( d \), through \( P_{\{d,e\}}(s_3) \), by matching \( e \). The chain of the matching can be performed since neither \( a \) nor \( e \) belongs to \( \Sigma_2 \). Thus, \( (\text{the number of occurrences of } a \text{ and } e \text{ in } L(\Sigma_2) \text{ and } L(\Sigma_3) \text{ are thus distinctive, which can be used for matching).}

The case \( s_1 \in L(\Sigma_2) \) discussed before is an illustration.

Thus, in Case 1, \( s_1 \) is determined via the chain \( (a, e)(c, d) \), where \( (a, e) \) and \( (c, d) \) are elements of \( D_{\Delta_1} \). For the purpose of matching, both \( a \) and \( e \) need to be distinctive. To make a distinctive, we require \( s_1 \) to be in \( L(\Sigma_2) \), \( L(\Sigma_3) \) or \( L(\Sigma_4) \). To make \( e \) distinctive, we require \( s_1 \) to belong to \( L(\Sigma_1) \), \( L(\Sigma_2) \) or \( L(\Sigma_3) \). Thus, \( s_1 \) is determined via the chain \( (a, e)(c, d) \) when \( s_1 \) is in \( L(\Sigma_2) \) or \( L(\Sigma_3) \).

Thus, for any choice of \( s_1 \), symbol \( d \) is determined. A similar analysis shows that, for any choice of \( s_1 \), \( c \) is determined. That is, for any choice of \( s_1 \), we can cover each symbol \( a, b, c, d, e \) in \( \Sigma \). Thus, we can now conclude that \( L_{\text{cand}} \) is decomposable with respect to \( \Delta_1 \). The analysis result is summarized in the following table. The second column shows how \( d \) is determined for \( s_1 \in L(\Sigma_j) \), \( j \in \{1, 5\} \); and the third column shows how \( e \) is determined.

| \( s_1 \) \( L(\Sigma_j) \) | \( d \) \( (c, d) \) \( (c, d) \) \( (a, e) \) \( (a, e) \) \( (a, e) \) |
|----------------|----------------|----------------|----------------|----------------|----------------|
| \( s_1 \in L(\Sigma_1) \) | \( (c, d) \) \( (c, d) \) \( (a, e) \) \( (a, e) \) \( (a, e) \) |
| \( s_1 \in L(\Sigma_2) \) | \( (a, e) \) \( (a, e) \) \( (a, e) \) \( (a, e) \) \( (a, e) \) |
| \( s_1 \in L(\Sigma_3) \) | \( (a, e) \) \( (a, e) \) \( (a, e) \) \( (a, e) \) \( (a, e) \) |
| \( s_1 \in L(\Sigma_4) \) | \( (a, e) \) \( (a, e) \) \( (a, e) \) \( (a, e) \) \( (a, e) \) |
| \( s_1 \in L(\Sigma_5) \) | \( (a, e) \) \( (a, e) \) \( (a, e) \) \( (a, e) \) \( (a, e) \) |

Such a table can be viewed as a proof of the decomposability of \( L_{\text{cand}} \) with respect to \( \Delta_1 \). Thus, our goal is to synthesize such a table, if possible, to discharge the verification obligation. We shall note that essentially we only need to produce certain paths (satisfying certain constraints) in the graph \( (\Sigma, D_{\Delta_1}) \) to synthesize such a table (c.f. second and third columns). This motivates us to develop a dependence graph based technique for verifying the decomposability of \( L_{\text{cand}} \) with respect to the distributions in \( M(\Delta) \). The reason why we analyze the choice of \( s_1 \) is that the corresponding sub-alphabet \( \{a, b, c\} \) in \( \Delta_1 \) does not appear in \( \Delta \). Thus, there must exist some distinctive symbols

\[16\] The number of occurrences of \( c \) is not distinctive for \( s_1 \in L(\Sigma_2) \), since strings in \( L(\Sigma_2) \) and \( L(\Sigma_3) \) have the same number of occurrences of \( c \). The same conclusion holds for \( L(\Sigma_4) \). For simplicity, we also say \( c \) is a distinctive symbol for \( s_1 \in L(\Sigma_j) \), \( j = 1, 4, 5 \). Clearly, \( c \) is a distinctive symbol for \( s_1 \in L(\Sigma_j) \) iff \( c \notin \Sigma_j \).
We conclude that otherwise, \( i \) is a distinctive symbol for \( \sigma \). Given any symbol \( \sigma \in \Sigma \), let \( \Delta \subseteq \Sigma \) be any distribution of \( \Sigma \) and let \( \Delta' \Delta = \Delta \subseteq \Sigma \). Then, \( \Delta' \Delta = \Delta \subseteq \Sigma \) is decomposable with respect to \( \Delta \) if there exists a sub-alphabet \( \Delta' \subseteq \Delta \) such that \( \Delta' \subseteq \Delta \) and \( \Delta \subseteq \Sigma \) is decomposable with respect to \( \Delta' \). In addition, we also define the graph \( (\Sigma, \Delta') = (\Delta, \Delta') \) and for each path \( p = (\Delta, \Delta') \) in \( (\Sigma, \Delta') \), record the set of all simple paths from \( \sigma \) to \( \sigma \) in graph \( \Sigma \). Since \( \sigma \) is an arbitrary symbol in \( \Sigma \) and all symbols in \( \Sigma \) are covered, it is possible to avoid unnecessary computation involved in the same analysis. It is clear that \( \Delta \subseteq \Sigma \) and, therefore, follows that \( \Delta \subseteq \Sigma \). Thus, \( \Delta \subseteq \Sigma \) and, for any choice of \( \sigma \in \Sigma \), all symbols in \( \Sigma \) are covered. Since \( \sigma \) is an arbitrary symbol in \( \Sigma \), we then conclude that, for any choice of \( \sigma \in \Sigma \), all symbols in \( \Sigma \) are covered.
1) The set \( B_{\Delta'}(\{(a, b, c)\}) \) of boundary symbols in \( \{a, b, c\} \) is \( \{a, c\} \). Thus, \( C_{\Delta'}(\{(a, b, c), d\}) = C_{\Delta'}(\{(a, b), c\}) \cup C_{\Delta'}(\{(c), d\}) \).

Now, the set \( S_{\Delta'}(\{(a, b, c), d\}) \) of all simple paths from \( a \) to \( d \) in the graph \( (\Sigma, D_{\Delta'}) \) that do not pass through any symbol within \( \{a, b, c\} \) (except for \( a \)) is the singleton \( \{acd\} \). Thus, \( C_{\Delta'}(\{(a, b, c), d\}) = C_{\Delta'}(\{acd\}) = \mathcal{N}(a) \cup \mathcal{N}(c) = \{2, 3\} \). Similarly, \( C_{\Delta'}(\{(c), d\}) = C_{\Delta'}(\{c\}) = \mathcal{N}(c) = \{1, 4, 5\} \). Thus, we have \( C_{\Delta'}(\{(a, b, c), d\}) = \{1, 5\} \).

2) Similarly, \( C_{\Delta'}(\{(a, b, c), e\}) = C_{\Delta'}(\{(a, e), c\}) \cup C_{\Delta'}(\{(c), d\}) \).

It is easy to check that \( C_{\Delta'}(\{(a, e), c\}) = \mathcal{N}(a) = \{2, 3, 4\} \), and \( C_{\Delta'}(\{(c), d\}) = \mathcal{N}(c) = \{1, 5\} \). Thus, we have \( C_{\Delta'}(\{(a, b, c), e\}) = \{1, 5\} \).

The decomposability of \( L_{\text{cond}} \) with respect to \( \Delta_2, \Delta_3, \Delta_4 \) and \( \Delta_5 \) can also be easily explained by using Theorem 26.

For any distribution \( \Delta' \in \mathcal{M}(\Delta) \), \( D_{\Delta} \subseteq D_{\Delta'} \) holds. Thus, \( (\Sigma, D_{\Delta'}) \) has strictly more edges than \( (\Sigma, D_{\Delta}) \) does. Instead of computing \( C_{\Delta}(\Sigma_{\iota}, \sigma') \), we can compute \( C_{\Delta'}(\Sigma_{\iota}, \sigma') \), which is also well-defined for any sub-alphabet \( \Sigma_{\iota} \) in \( \Delta' \) and any \( \sigma \in \Sigma - \Sigma_{\iota} \). Furthermore, \( C_{\Delta}(\Sigma_{\iota}, \sigma') \subseteq C_{\Delta'}(\Sigma_{\iota}, \sigma') \).

We immediately have the following, which allows us to work on the graph \( (\Sigma, D_{\Delta'}) \) for each \( \sigma' \in \Delta(\Delta) \).

**Corollary 29.** Let \( \Delta' = (\Sigma_{\iota}', \Sigma_{\jot}', \ldots, \Sigma_{\kappa}') \in \mathcal{M}(\Delta) \). \( L_{\text{cond}} \) is decomposable with respect to \( \Delta' \) if there exists a sub-alphabet \( \Sigma_{\iota}' \) in \( \Delta' - \Delta \) such that for each \( \sigma' \in \Sigma - \Sigma_{\iota}' \), \( C_{\Delta}(\Sigma_{\iota}', \sigma') = [1, n] \), where \( \Delta = (\Sigma_1, \Sigma_2, \ldots, \Sigma_n) \).

**Proof:** This immediately follows from Theorem 26 and the fact that \( C_{\Delta}(\Sigma_{\iota}', \sigma') \subseteq C_{\Delta'}(\Sigma_{\iota}', \sigma') \) for each \( \Sigma_{\iota}' \in \Delta' \) and each \( \sigma' \in \Sigma - \Sigma_{\iota}' \).

To prove \( \Delta \) has no reduction, we have the following result.

**Corollary 30.** \( \Delta \) does not have a reduction if for any two different sub-alphabets \( \Sigma_i, \Sigma_j \in \Delta \) such that \( \Sigma_i \cup \Sigma_j \neq \Sigma \), it holds that for each \( \sigma' \in \Sigma - (\Sigma_i \cup \Sigma_j) \), \( C_{\Delta}(\Sigma_i \cup \Sigma_j, \sigma') = [1, n] \), where \( \Delta = (\Sigma_1, \Sigma_2, \ldots, \Sigma_n) \).

**Proof:** We have that \( \Delta \) does not have a reduction iff \( (\Delta, \Delta) \) is not a reduction of \( \Delta \). If \( (\Delta, \Delta) \) is decomposable with respect to each distribution in \( \Delta \) itself, the only sub-alphabet in \( \Delta' \) is \( \Sigma_{\iota}' \), and \( \Delta \) is the union of some sub-alphabets \( \Sigma_i, \Sigma_j \) in \( \Delta \) such that \( \Sigma_i \cup \Sigma_j \neq \Sigma \). The statement then immediately follows from Corollary 29.

Corollary 30 can be used to show the following "parameterized" result.

**Theorem 31.** For any \( n \geq 3 \), the distribution \( \Delta = (\{a_0, a_1\}, \{a_1, a_2\}, \ldots, \{a_{n-2}, a_{n-1}\}, \{a_{n-1}, a_0\}) \) of \( \Sigma = \{a_0, a_1, \ldots, a_{n-1}\} \) does not have a reduction.

**Proof:** \( \Delta \) has no reduction when \( n = 3 \), since \( \mathcal{M}(\Delta) = \emptyset \). And, from Example 23 and Example 22, we know that \( \Delta \) does not have a reduction when \( n = 4 \) or \( n = 5 \). Thus, we assume \( n \geq 6 \). Let \( \Sigma_i := \{a_{i}, a_{i+1}\} \) for each \( i \in [0, n-1] \), where the addition \( i + 1 \) is modulo \( n \). Let \( i_1 < i_2 \) be any two indexes in \( [0, n-1] \).

![Fig. 3: graph (\Sigma, D_{\Delta})](image)

Clearly, \( \Sigma_{i_1} \cup \Sigma_{i_2} \neq \Sigma \). We only need to show that for each \( \sigma' \in \Sigma - (\Sigma_{i_1} \cup \Sigma_{i_2}) \), \( C_{\Delta}(\Sigma_{i_1} \cup \Sigma_{i_2}, \sigma') = [1, n] \).

We first observe that each \( a_i \) only belongs to \( \Sigma_{i_1} \) and \( \Sigma_{i_2} \). Thus, \( \mathcal{N}(a_i) = [1, n] \) \( \setminus \{i_1, i_2\} \). We have the following case analysis:

1) \( 2 < i_2 - i_1 < n - 2 \): In this case, the boundary symbols in \( \Sigma_{i_1} \cup \Sigma_{i_2} \) are \( a_{i_1}, a_{i_1+1}, a_{i_2}, a_{i_2+1} \) (see Fig. 3).
   a) For any symbol \( a_j \) that lies between \( a_{i_1+1} \) and \( a_{i_2} \), we have \( C_{\Delta}(\Sigma_{i_1} \cup \Sigma_{i_2}, a_j) = \mathcal{C}_{\Delta}(a_{i_1+1}a_{i_1+2} \ldots a_{j-2}a_{j-1}a_{j}) \cup \mathcal{C}_{\Delta}(a_{j}a_{j+1}a_{j+2} \ldots a_{i_2}) \).
   b) For any symbol \( a_j \) that lies between \( a_{i_2} \) and \( a_{i_1+1} \), we have \( C_{\Delta}(\Sigma_{i_1} \cup \Sigma_{i_2}, a_j) = C_{\Delta}(\Sigma_{i_1} \cup \Sigma_{i_2}, a_j) \).

2) \( 1 \leq i_2 - i_1 \leq 2 \): In this case, the boundary symbols in \( \Sigma_{i_1} \cup \Sigma_{i_2} \) are \( a_{i_2}, a_{i_1+1} \) and \( a_{i_2+1} \). The proof is exactly the same as in 1) b) with \( a_{i_1+1} \) and \( a_{i_2+1} \). We only need to show that \( C_{\Delta}(\Sigma_{i_1} \cup \Sigma_{i_2}, a_j) = [1, n] \) for each \( a_j \) between \( a_{i_2} \) and \( a_{i_1+1} \).

3) \( n - 2 \leq i_2 - i_1 \leq n - 1 \): In this case, the boundary symbols in \( \Sigma_{i_1} \cup \Sigma_{i_2} \) are \( a_{i_2}, a_{i_2+1}, a_{i_1+1}, a_{i_2+1} \). The proof is exactly the same as in 1) a) with \( a_{i_1+1} \) and \( a_{i_2+1} \). We only need to show that \( C_{\Delta}(\Sigma_{i_1} \cup \Sigma_{i_2}, a_j) = [1, n] \) for each \( a_j \) between \( a_{i_1+1} \) and \( a_{i_2+1} \).

**Remark 32.** It is difficult to manage the proofs for results like Theorem 31 by using \( l_{\text{cond}} \) directly, since an infinite number of decomposability verifications would need to be carried out.

The advantage of Theorem 26, Corollaries 29 and 30 is that...
only structures of $\Delta$ and $\Delta_i$’s are required for the reasoning, using graphs of the dependence relations. This can facilitate reduction refutation for the parameterized setup, as illustrated in Theorem 31.

It is not difficult to see that Proposition 15 and the candidate counter examples based reduction refutation technique are not comparable (see Example 22 and Example 33), and thus they are complementary for reduction refutation.

**Example 33.** Consider the distribution $\Delta = (\Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4)$, where $\Sigma_1 = \{a, b\}$, $\Sigma_2 = \{b, c\}$, $\Sigma_3 = \{c, d\}$ and $\Sigma_4 = \{d, e\}$. Let $\{\Delta_1, \Delta_2\}$ be a candidate reduction of $\Delta$, where

\[
\Delta_1 = \{(a, b, c), \{c, d, e\}\}
\]

\[
\Delta_2 = \{(a, b, c, d), \{d, e\}\}
\]

Proposition 15 can be used to easily refute $\{\Delta_1, \Delta_2\}$. However, $L_{\text{cand}}$ fails to refute $\{\Delta_1, \Delta_2\}$, since it is not decomposable with respect to $\Delta_1, \Delta_2$.

**B. Strengthened Substitution-based Proof for Reduction Validation**

Let $P$ be any candidate reduction of $\Delta$. Suppose it passes the tests of Proposition 15 and the candidate counter example $L_{\text{cand}}$. Then, it is of interest to try validating $P$. A substitution-based proof technique has been proposed in [8] for automated reduction validation. We here recall the main idea and provide a formal soundness proof. Then, we proceed to show that the substitution-based proof technique can be improved with the help of Lemma 11.

The main idea: for any distribution $\Delta’ = (\Sigma_1, \Sigma_2, \ldots, \Sigma_m)$ of $\Sigma$, let $A(\Delta’) := \bigcup_{i\neq j}(\Sigma_i \cap \Sigma_j)$ denote the set of shared symbols in distribution $\Delta’$. Let $\Delta_n = (\Sigma_1, \Sigma_2, \ldots, \Sigma_m)$ be another distribution of $\Sigma$, $\Delta’$ is said to be substitutable into the $i$-th sub-alphabet of $\Delta_n$, denoted $\Delta’ \rightarrow_i \Delta_n$, if $A(\Delta’) \subseteq \Sigma_i$ holds. $\Delta’$ is said to be substitutable into $\Delta_n$, denoted $\Delta’ \rightarrow \Delta_n$, if $\Delta’ \rightarrow_i \Delta_n$ for some $i \in [1, l]$. The result of substituting $\Delta’$ into the $i$-th sub-alphabet of $\Delta_n$ is denoted by $(\Delta’ \vdash_i \Delta_n)$, which is defined to be the distribution obtained from

\[
(\Sigma_1', \Sigma_2', \ldots, \Sigma_{i-1}', \Sigma_i' \cap \Sigma_i, \Sigma_i' \cap \Sigma_{i+1}', \ldots, \Sigma_m')
\]

by keeping only the maximal components. We denote the result of substituting $\Delta’$ into $\Delta_n$ by $(\Delta’ \vdash \Delta_n)$, which is defined to be the set $\{(\Delta’ \vdash_i \Delta_n) \mid \Delta’ \rightarrow_i \Delta_n, i \in [1, l]\}$ of distributions. Given any candidate reduction $\{\Delta_1, \Delta_2, \ldots, \Delta_m\}$ of a source distribution $\Delta$, the idea of the proof technique [8] is to derive new distributions by using substitutions and maintain the set of currently available distributions. If $\Delta$ can be obtained from $\{\Delta_1, \Delta_2, \ldots, \Delta_m\}$ in this way and also $\Delta \leq \Delta$, holds for each $i \in [1, m]$, then $\{\Delta_1, \Delta_2, \ldots, \Delta_m\}$ is a reduction of $\Delta$, using Lemma 1 and Lemma 5. In fact, let $\Delta’$ and $\Delta_n$ be any two distributions of $\Sigma$, where $\Delta’ \rightarrow_i \Delta_n$. For any language $L$ over $\Sigma$, if $L$ is decomposable with respect to $\Delta’$ and $\Delta_n$, then $L$ is also decomposable with respect to the distribution $(\Delta’ \vdash_i \Delta_n)$, using Lemma 1.

**Lemma 34.** Let $\Delta’, \Delta_n$ be any two distributions of $\Sigma$, where $\Delta’ \rightarrow_i \Delta_n$. For any language $L$ over $\Sigma$, if $L$ is decomposable with respect to $\Delta’$ and $\Delta_n$, then $L$ is also decomposable with respect to the distribution $(\Delta’ \vdash_i \Delta_n)$.

**Proof:** By definition, $(\Delta’ \vdash_i \Delta_n)$ is the distribution obtained from

\[
(\Sigma_1', \Sigma_2', \ldots, \Sigma_{i-1}', \Sigma_i' \cap \Sigma_i, \Sigma_i' \cap \Sigma_{i+1}', \ldots, \Sigma_m')
\]

by keeping only the maximal components. If $L$ is decomposable with respect to $\Delta’$ and $\Delta_n$, then we have

\[
\{ L = P_{\Sigma_1'}(L)\| P_{\Sigma_2'}(L) \| \ldots \| P_{\Sigma_{i-1}'}(L) \} \}
\]

By substituting (1) to (2), we have, by Lemma 1,

\[
\{ L = P_{\Sigma_1'}(L)\| P_{\Sigma_2'}(L) \| \ldots \| P_{\Sigma_{i-1}'}(L) \}
\]

For any term $P_{\Sigma}(L)$ at the right hand side of the last equality, it can be removed, if there exists another term $P_{\Sigma'}(L)$ such that $\Sigma' \subseteq \Sigma''$, and the equality still holds. Thus, we only need to keep those terms $P_{\Sigma}(L)$ where $\Sigma''$ is maximal.

We then conclude that $L$ is decomposable with respect to the distribution $(\Delta’ \vdash_i \Delta_n)$.

Thus, if $L$ is decomposable with respect to each $\Delta_i$, then the above substitution procedure would ensure that $L$ is also decomposable with respect to the newly generated distribution $\Delta$. If $\Delta \leq \Sigma \Delta_i$ for each $i \in [1, m]$, then the decomposability of $L$ with respect to $\Delta$ also implies the decomposability of $L$ with respect to each $\Delta_i$, by Lemma 5. We then conclude that $\{\Delta_1, \Delta_2, \ldots, \Delta_m\}$ is a reduction of $\Delta$. Thus, the substitution-based proof technique is sound.

We now use the next example to show that the substitution-based proof technique is not complete, and it can be improved with the help of Lemma 11.

**Example 35.** Consider $\Delta = (\Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \Sigma_5, \Sigma_6)$, where $\Sigma_1 = \{a, b\}$, $\Sigma_2 = \{b, c\}$, $\Sigma_3 = \{a, c\}$, $\Sigma_4 = \{d, e\}$, $\Sigma_5 = \{e, f\}$ and $\Sigma_6 = \{d, f\}$. Let $\{\Delta_1, \Delta_2\}$ be a candidate reduction of $\Delta$, where

\[
\Delta_1 = \{\{a, b, d, e\}, \{b, c, e, f\}, \{a, c, d, f\}\}
\]

\[
\Delta_2 = \{\{a, b, c\}, \{d, e\}, \{e, f\}, \{d, f\}\}
\]

It is clear that the substitution-based proof cannot be applied on $\{\Delta_1, \Delta_2\}$. Furthermore, neither the necessary condition in Lemma 10 nor the candidate counter example $L_{\text{cand}}$ can refute $\{\Delta_1, \Delta_2\}$. Let $P = \{\Delta_1, \Delta_2\}$ and let $Q = \{\Delta_3, \Delta_5\}$, where $\Delta_2 = \{\{a, b, c\}, \{d, e\}\}$. It is clear that $\Delta_2 \leq \Sigma_2 \Delta_5$ and thus $P \leq \Sigma_2 Q$. It is possible to show, by using the substitution-based proof technique, that $Q$ is indeed a reduction of $\Delta$. Thus, by Lemma 11, $P$ is also a reduction of $\Delta$. Thus, the substitution-based proof technique is not complete.

Based on Example 35, we have the next observation. If the substitution-based proof technique cannot validate $P$ and the reduction refutation techniques fail to refute $P$, then we can check whether there exists a set $Q \subseteq \mathcal{M}(\Delta)$ of distributions such that $P \leq Q$ and the substitution-based proof technique works on $Q$ (see Proposition 36). $Q$ does not need to be an element in $S_\Delta$, that is, we do not require the distributions in $Q$ to be $\leq \Sigma$-incomparable. The definition of $P \leq Q$ can be naturally extended in this case: $\forall \Delta_j \in Q, \Delta_i \in P, \Delta_i \leq \Sigma \Delta_j$. In general, there are many different choices of $Q$. It turns out that we only need to apply the substitution-based proof on the following set.
of distributions, where \( U(\Delta^\prime) \) is defined as the union of \( \Delta^\prime \) with any \( \Delta \) that \( \Delta \leq \Delta^\prime \) and \( \Delta \in \mathcal{M}(\Delta) \).

\[
U(P) := \{ \Delta^\prime \in \mathcal{M}(\Delta) \mid \Delta^\prime \leq \Delta \} = \bigcup_{\Delta^\prime \in P} U(\Delta^\prime)
\]

We note that the above approach actually proves the stronger result that \( P \) is a reduction of \( \Delta \). We call this new approach the strengthened substitution-based proof. However, it is more expensive to apply the substitution-based proof on \( \mathcal{L}(P) \). In the worst case, the substitution-based proof has to be applied on the set \( \mathcal{M}(\Delta) \) of all merged distributions, if \( P = \mathcal{L}(\Delta) \). Thus, in the implementation, we only apply the strengthened proof if the (ordinary) substitution-based proof fails to prove \( P \). We remark that the substitution-based proof on \( Q \subseteq \mathcal{M}(\Delta) \), where \( P \leq \Delta Q \), can be viewed as a heuristic for the proof on \( \mathcal{L}(P) \) (by working on the subset \( Q \subseteq \mathcal{L}(P) \)). The same remark is also applicable to \( P \subseteq \mathcal{L}(P) \).

C. Reduction Verification Procedure

Up to now, we have provided some techniques for reduction validation and reduction refutation. In this subsection, we are ready to present the reduction verification procedure.

We shall now summarize the main steps of the overall procedure involved in reduction verification. Given any candidate reduction \( P \) of \( \Delta \).

1. Check whether \( \mathcal{L}_{\delta_1} \subseteq \mathcal{L}_\Delta \) (Proposition 15). If not, return NO.
2. Check whether the candidate counter example \( L_{\text{cand}} \) is decomposable with respect to each distribution in \( P \). If yes, then return NO.
3. Apply the substitution-based proof on \( P \). If \( \Delta \) is obtained, then return YES.
4. Apply the substitution-based proof on \( \mathcal{L}(P) \), i.e., apply the strengthened substitution-based proof on \( P \). If \( \Delta \) is obtained, then return YES.
5. return UNKNOWN

We can compute \( \bigwedge_{\Delta^\prime \in P} \Delta^\prime \), incrementally, thanks to the property of associativity: \( \bigwedge_{i=1}^t (\Delta_1 \wedge \Delta_2) \wedge \Delta_3 \wedge \ldots \wedge \Delta_{t-1} \wedge \Delta_t \) (Step 2) can use the decomposability verification algorithm [1] or the structure-based technique in Theorem 26 (Theorem 44 in Appendix A is an improvement). We note that the first two steps constitute the reduction refutation procedure; the third and the forth steps constitute the reduction validation procedure. It is worth mentioning that the first four steps could be executed in parallel.

There are still many open problems that are left unanswered for the above reduction verification procedure. In particular, we do not know whether it is complete. Indeed, it is of interest to first address following open question.

Open Question: The above reduction verification procedure works on \( \mathcal{L}(P) \), in order to prove that \( P \) is a reduction of \( \Delta \). It is possible to come up with a proof scheme that is at least as powerful. Indeed, we can apply the (substitution-based) proof on \( \mathcal{L}(P) \) instead.

\[
\mathcal{L}(P) = \{ \Delta^\prime \in \mathcal{M}(\Sigma) \mid \exists \Delta^\prime \subseteq P, \Delta^\prime \leq \Sigma, \Delta^\prime \}
\]

We can then execute the reduction verification procedure on \( \mathcal{L}(P) \).

V. TOWARDS AUTOMATIC GENERATION OF OPTIMAL REDUCTIONS

In this section, we shall address the problem of computing small or (near-)optimal reductions of distributions. We present an incremental production algorithm in Section V-A for generating candidate reductions. When combined with the reduction validation procedure presented in Section IV-C, this allows for a rather fast generation of (near-)optimal reductions, if small reductions do exist. Then, a recursive algorithm for computing small reductions is shown in Section V-B.

A. Incremental Production of Candidate Reductions

The problem of producing candidate reductions is a solved problem, if we do not worry about optimality of the reductions. Indeed, by Theorem 12, \( \mathcal{L}(\Delta) \) is already a “perfect” candidate reduction of \( \Delta \), if the search space \( S_\Delta \) is non-empty. It is of interest, however, to consider the production of candidate reductions with smaller/the smallest width, in order to compute (near-)optimal reductions. We shall now explain how Lemma 9 can be used for the efficient production of candidate reductions in an incremental manner (Incremental Collection), which may be followed by an incremental refinement procedure guided by Lemma 10 (Incremental Refinement).

By Lemma 9, any reduction \( \{ \Delta_1, \Delta_2, \ldots, \Delta_l \} \) of \( \Delta \) has to satisfy the condition \( \bigcup_{i \in [1,l]} \mathcal{L}_\Delta = \mathcal{L}_\Delta \). To produce a candidate reduction, we incrementally collect pairs of independent symbols in \( \mathcal{L}_\Delta \), which correspond to edges in the graph \( (\Sigma, \mathcal{L}_\Delta) \), using distributions \( \Delta_i \) that are merged from \( \Delta \). We shall use the next example as an illustration of the idea.

Example 37. Consider the distribution \( \Delta = \{ \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4 \} \) of \( \{ a, b, c, d, e \} \), where \( \Sigma_1 = \{ a, b \}, \Sigma_2 = \{ b, c \}, \Sigma_3 = \{ c, d \}, \Sigma_4 = \{ d, e \} \). The independence relation \( \mathcal{I}_\Delta \) induced by \( \Delta \) is represented in Fig. 4. By Lemma 9, we need to produce a set \( \{ \Delta_1, \Delta_2, \ldots, \Delta_l \} \) of distributions such that \( \mathcal{I}_\Delta = \bigcup_{i \in [1,l]} \mathcal{I}_{\Delta_i} \).

22Theoretically, we can work with \( \mathcal{L}(P) \) instead.

23Here, “perfect” means that we do not need to look at the other candidate reductions: if \( \mathcal{L}(\Delta) \) is not a reduction of \( \Delta \), then there is no reduction of \( \Delta \).
Suppose we would like to collect the edge \(\{a, e\}\) in \(I_\Delta\) using \(\Delta_1\), that is, ensure \(\{a, e\} \in I_{\Delta_1}\) holds, then we only need to make sure that, in the process of merging \(\Delta\), the sub-alphabets containing \(a\) will not be merged with those sub-alphabets that contain \(e\). It is not difficult to see that we have the following choices of \(\Delta_1\) of size 2 (ignoring the ordering, which does not matter), which separates \(\Sigma_1\) from \(\Sigma_4\).

1) \(\Delta_1 = (\Sigma_1 \cup \Sigma_2 \cup \Sigma_3, \Sigma_4)\), or
2) \(\Delta_1 = (\Sigma_1 \cup \Sigma_2, \Sigma_3 \cup \Sigma_4)\), or
3) \(\Delta_1 = (\Sigma_1 \cup \Sigma_3, \Sigma_2 \cup \Sigma_4)\), or
4) \(\Delta_1 = (\Sigma_1, \Sigma_4 \cup \Sigma_2 \cup \Sigma_3)\)

Suppose we choose option 1), then \(\Delta_1 = \{(a, b, c, d), \{d, e\}\}\).

Then, \(I_{\Delta_1} = \{(a, e), \{b, c\}, \{c, e\}\}\). The remaining edges are then \(I_\Delta - I_{\Delta_1} = \{(a, c), \{a, d\}, \{b, d\}\}\). Suppose we would like to collect the edge \(\{a, c\}\) in \(I_\Delta - I_{\Delta_1}\) using \(\Delta_2\), i.e., ensure \(\{a, c\} \in I_{\Delta_2}\) holds, then we only need to make sure that, in the process of merging \(\Delta\), the sub-alphabets containing \(a\) are not merged with those sub-alphabets that contain \(c\). It is not difficult to see that we have the following choices of \(\Delta_2\) of size 2, which separates \(\Sigma_1\) from \(\Sigma_2\) and \(\Sigma_3\).

1') \(\Delta_2 = (\Sigma_1 \cup \Sigma_2, \Sigma_2 \cup \Sigma_3)\), or
2') \(\Delta_2 = (\Sigma_1 \cup \Sigma_3, \Sigma_2 \cup \Sigma_1)\)

Suppose we choose option 2'), then \(\Delta_2 = \{(a, b, c, d, e)\}\).

Then, \(I_{\Delta_2} = \{(a, c), \{a, d\}, \{a, e\}\}\). The remaining edges are then \(I_\Delta - I_{\Delta_2} = \{(b, d)\}\). To collect edge \(\{b, d\}\), we only have the following choice of \(\Delta_3\) of size 2, which separates \(\Sigma_1\) and \(\Sigma_2\) from \(\Sigma_3\) and \(\Sigma_4\).

\[\Delta_3 = (\Sigma_1, \Sigma_4 \cup \Sigma_2 \cup \Sigma_3) = \{(a, b, c), \{c, d, e\}\}\]

Clearly, \(\Delta_1, \Delta_2, \Delta_3\) are pairwise \(\leq\)-incomparable. Thus, we have produced a candidate reduction \(\{\Delta_1, \Delta_2, \Delta_3\} \in \mathcal{G}_\Sigma\) that satisfies Lemma 9. It is easy to check, using the substitution-based proof, that \(\{\Delta_1, \Delta_2, \Delta_3\}\) is indeed a reduction of \(\Delta\).

To collect the edge \(\{\sigma_1, \sigma_2\}\) in \(I_\Delta\) using \(\Delta_j\), we only need to make sure that those sub-alphabets that contain symbol \(\sigma_1\) will not be merged with those sub-alphabets that contain \(\sigma_2\). The Incremental Collection stage collects pairs of independent symbols using distributions \(\Delta_j\)'s in an incremental manner; it stops immediately after \(\bigcup_{1 \leq j \leq l} I_{\Delta_j} = I_\Delta\) becomes satisfied. There are often different choices of distributions to be added at each step of the Incremental Collection stage. These decision choices form a tree structure; then, backtracking can be used to enumerate different choices. In order to facilitate the computation of small or (near-)optimal reductions, distributions with small sizes are explored first.

We note that the Incremental Collection stage allows for a quite efficient production of candidate reductions. However, it is still possible that the candidate reduction does not satisfy the necessary condition stated in Proposition 15; the candidate reduction is then not a reduction of \(\Delta\), and we must have \(\Lambda_{1 \leq j \leq l} \Delta_j > \Delta\). If it holds that \(\Lambda_{\Delta' \in [\Delta]} \Delta' = \Delta\), then it is guaranteed that we can collect more distributions in \(\mathcal{M}(\Delta)\) so that the enlarged candidate reduction does meet the necessary condition. The enlarged candidate reduction can be considered a refinement of the original candidate reduction, which has a higher chance of becoming a reduction of \(\Delta\) (c.f. Lemma 11).

And, not surprisingly, Lemma 10 can be used for efficiently guiding the process of refining candidate reductions. The idea is first illustrated using the next example. For convenience, we also refer to the candidate reductions satisfying the necessary condition stated in Proposition 15 as meet-consistent candidate reductions.

Example 38. Consider \(\Delta = (\Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \Sigma_5, \Sigma_6)\), where \(\Sigma_1 = \{a, b\}, \Sigma_2 = \{b, c\}, \Sigma_3 = \{a, c\}\) and \(\Sigma_4 = \{d, e\}, \Sigma_5 = \{e, f\}, \Sigma_6 = \{d, f\}\). Suppose we merge \(\Sigma_1\) and \(\Sigma_2\), then we obtain the distribution \(\Delta_1 = \{(a, b), (d, e), (c, f), (d, f)\}\).

It follows that \(I_{\Delta_1} = I_\Delta\), that is, all the pairs of independent symbols of \(\Delta\) have been collected by \(\Delta_1\). Since the candidate reduction \(\{\Delta_1\}\) is not meet-consistent, we could refine it by adding some more distributions that are merged from \(\Delta\). Since \(\{a, b\} \subseteq \Delta_1\) and \(\{a, b\} \not\subseteq \Delta\), we can use \(\Delta_2\) to break \(\{a, b, c\}\), that is, ensure \(\{a, b\} \not\subseteq \Delta_2\). This would then help to ensure \(\{a, b, c\} \not\subseteq \Delta_1 \land \Delta_2\) and then we have \(\Delta_1 \land \Delta_2 = \Delta\), since \(\{a, b, c\}\) is the only sub-alphabet in \(\Delta_1\) that causes the trouble. To use \(\Delta_2\) to break \(\{a, b, c\}\), we must keep \(\Sigma_2\) and \(\Sigma_3\) separated in \(\Delta_2\). Some possible choices of \(\Delta_2\) (there are other choices that are not enumerated here) are

1) \(\Delta_2 = \{(a, b), \{b, c\}, \{c, e\}, \{d, e\}\}\), or
2) \(\Delta_2 = \{(a, b, d, e, f), \{b, c\}, \{a, c\}\}\), or
3) \(\Delta_2 = \{(a, b), \{b, c, d, e, f\}, \{a, c\}\}\), or
4) \(\Delta_2 = \{(a, b), \{b, c\}, \{a, c, d, e, f\}\}\)

One can verify that the refined candidate reduction \(\{\Delta_1, \Delta_2\}\) is indeed a reduction of \(\Delta\), for any choice of \(\Delta_2\) shown above, by using the substitution-based proof.

If the candidate reduction \(\{\Delta_1, \Delta_2, \ldots, \Delta_l\}\) of distribution \(\Delta = (\Sigma_1, \Sigma_2, \ldots, \Sigma_n)\) generated in the Incremental Collection stage does not satisfy the necessary condition \(\Lambda_{1 \leq i \leq l} \Delta_i = \Delta\), then we shall need to break those troublesome sub-alphabets, according to Lemma 10, by using distributions that are merged from \(\Delta\) (in an incremental manner). Now, suppose \(\Sigma' \subseteq \Sigma\) is some sub-alphabet that needs to be broken. Let \(B(\Sigma')\) denote the minimal elements (in the sense of set inclusion) in \(\{\Sigma \setminus \{\Sigma'\} \mid \Sigma' \subseteq \Sigma\}\). We then add a distribution \(\Delta'\) that is merged from \(\Delta\), with the constraint that no element in \(B(\Sigma')\) can be a subset of some element in the partition \(P_\Sigma(\Delta')\) corresponding to \(\Sigma'\). The same refinement procedure is repeated on the refined candidate reduction \(\{\Delta_1, \Delta_2, \ldots, \Delta_l', \Delta_1\}\). This Incremental Refinement process is continued until the refined candidate reduction becomes meet-consistent. There are often different choices of distributions to be added in each step of the refinement procedure. These decision choices form a tree structure, and backtracking can be used to enumerate different choices. Again, distributions with small sizes are explored first.

Of course, it is also possible to use the necessary condition in Lemma 10 to directly produce a candidate reduction, since collecting pairs of independent symbols is only a special case of breaking sub-alphabets, i.e., breaking sub-alphabets of size 2. And the procedure is the same as that of refining candidate reductions presented above, but starts with candidate reduction \(\{\Delta_1\}\) instead. However, collecting pairs of independent symbols (with backtracking) alone is sufficient in most examples and works quite fast. This is illustrated using the next example.
Example 39. Let us now revisit Example 38. It is possible to generate a reduction in the Incremental Collection stage with a good choice of $\Delta_1$. With backtracking, we can first produce distribution $\Delta_1 = \langle \{a, b\}, \{b, c\}, \{a, c, d, e, f\} \rangle$ instead. Since $I_{\Delta_1}$ is a proper subset of $I_\Delta$, we need more distributions. We can then add $\Delta_2 = \langle \{a, b, c\}, \{d, e\}, \{e, f\}, \{d, f\} \rangle$ so that $I_\Delta = I_{\Delta_1} \cup I_{\Delta_2}$, since $I_{\Delta_2} = I_{\Delta_1}$. It is straightforward to verify that $\{\Delta_1, \Delta_2\}$ is a reduction of $\Delta$. Indeed, the same reduction has appeared in Example 38, but in the reverse order.

We now have the following remark regarding the Incremental Collection stage that collects pairs of independent symbols.

Remark 40. Firstly, the order of production of distributions plays a useful role. To avoid the use of refinement of candidate reductions, one shall avoid $I_{\Delta_1} = I_{\Delta_2}$. Secondly, not all the reductions can be produced by collecting pairs of independent symbols. As a simple example, the reduction $\{\Delta_1, \Delta_2\}$ of distribution $\Delta$ in Example 38 cannot be generated by collecting pairs of independent symbols alone, where

$$
\begin{align*}
\Delta_1 &= \langle \{a, b\}, \{b, c\}, \{a, c\}, \{d, e, f\} \rangle \\
\Delta_2 &= \langle \{a, b, c\}, \{d, e\}, \{e, f\}, \{d, f\} \rangle
\end{align*}
$$

since $I_{\Delta_1} = I_{\Delta_2} = I_\Delta$. Thus, no matter whether $\Delta_1$ or $\Delta_2$ is produced first, the Incremental Collection stage will terminate there. It is clear that $\{\Delta_1, \Delta_2\}$ is not optimal, since $\{\Delta_1', \Delta_2'\}$ is an improved reduction of $\Delta$, where

$$
\begin{align*}
\Delta_1' &= \langle \{a, b\}, \{d, e, f\} \rangle \\
\Delta_2' &= \langle \{a, b, d, e\}, \{b, c, e, f\}, \{a, c, d, f\} \rangle
\end{align*}
$$

Fortunately, the improved reduction $\{\Delta_1', \Delta_2'\}$ can be generated by collecting pairs of independent symbols alone (with backtracking). It is of much interest to know whether there exist some distributions for which the Incremental Refinement stage is strictly necessary for the production of optimal reductions.

A useful property for meet-consistent candidate reductions is the minimal property. A meet-consistent candidate reduction $P$ of $\Delta$ is said to be minimal if every proper subset $P'$ of $P$ is not meet-consistent.

We now summarize the main steps of the overall procedure involved in the automatic generation of small or (near-)optimal reductions using the incremental production approach. Given any distribution $\Delta$,

1. Check whether $\bigwedge_{\Delta' \in \{\bot(\Delta)\}} \Delta' = \Delta$. If not, return NO.
2. Check whether $L_{\text{cond}}$ is decomposable with respect to each distribution in $\{\bot(\Delta)\}$. If yes, return NO.
3. Generate a candidate reduction $P$ of $\Delta$ that has not been seen before, by collecting pairs of independent symbols (Incremental Collection stage). If $P$ does not exist and $\bot(\Delta)$ is minimal, then HALT. If $P$ does not exist and $\bot(\Delta)$ is non-minimal, then goto Step 9).
4. Check whether $\bigwedge_{\Delta, P} \Delta_i = \Delta$. If yes, set $P' := P$, set $\text{flag} := 1$ and goto Step 6); otherwise, set $\text{flag} := 0$.
5. Refine candidate reduction $P$ by breaking sub-alphabets (Incremental Refinement stage) and generate a (refined) meet-consistent candidate reduction $P'$ that has not been seen before. If $P'$ does not exist, then goto Step 3).
6. Apply the substitution-based proof on $P'$ (Section IV-B). If $\Delta$ is obtained, return $P'$.
7. Apply the substitution-based proof on $U(P')$, i.e., apply the strengthened substitution-based proof on $P'$. If $\Delta$ is obtained, return $P'$.
8. If $\text{flag} = 1$, goto Step 3); otherwise, goto Step 5.
9. Apply the substitution-based proof on $\bot(\Delta)$. If $\Delta$ is obtained, return $\bot(\Delta)$.
10. Apply the substitution-based proof on $M(\Delta)$. If $\Delta$ is obtained, return $\bot(\Delta)$; otherwise, HALT.

Step 1) and Step 2) are used to try refuting $\bot(\Delta)$; if $\bot(\Delta)$ is successfully refuted, then $\Delta$ does not have a reduction. The incremental production approach (with backtracking) consists of Step 3) to Step 10). We notice that this approach generates a minimal meet-consistent candidate reduction $P'$, before the (strengthened) substitution-based proof is applied on $P'$. If we cannot validate $P'$, then we backtrack and produce another minimal meet-consistent candidate reduction, if it exists. Since our purpose here is to generate reductions, we will give up on a candidate reduction $P'$ if the strengthened substitution-based proof fails on $P'$; in particular, we will not use $L_{\text{cond}}$ to refute $P'$. Backtracking (using tree structures) can be implemented for Step 3) and Step 5).

We still do not rule out the possibility that $\Delta$ has a reduction while every minimal meet-consistent candidate reduction (in the above sense) is not a reduction of $\Delta$. This is not a serious restriction, since we can apply the (strengthened) substitution-based proof on $\bot(\Delta)$ in case $\bot(\Delta)$ is non-minimal. Steps 9) and 10) are devoted just for that purpose. In particular, we have $U(\bot(\Delta)) = M(\Delta)$ in Step 10); Steps 9) and 10) will be executed if and only if $\bot(\Delta)$ is non-minimal and each minimal meet-consistent candidate reduction cannot be validated in Steps 3) to 8). We note that Steps 1), 2), 9) and 10) can be carried out in parallel and they can be run in parallel with Steps 3) to 8); then, the second part of Step 3) will be changed to “If $P$ does not exist, then HALT”, and $\bot(\Delta)$ does not need to be generated in Steps 3) to 8).

We also need to ensure those distributions in a candidate reduction to be $\leq_{\Sigma}$-incomparable. This could be achieved by replacing $P$ and $P'$ with $[P]$ and $[P']$, respectively. We here shall note that every reduction $P'$ generated using the above approach, if it exists, is compact, due to the minimality.

Remark 41. Unfortunately, we still cannot guarantee that the reduction generated by the above algorithm, if any, is optimal, since we cannot ensure that any candidate reduction that is not validated by the strengthened substitution-based proof, which is thrown away by the above algorithm, is indeed not valid. The algorithm only generates and tests upon minimal meet-consistent candidate reductions; we still do not rule out the possibility that a non-minimal meet-consistent candidate reduction can be optimal. It is easy to modify the algorithm to collect the set of all validated reductions. Then, any optimal reduction among this collection can be chosen as the output.

B. Recursive Generation of Reductions

In this subsection, we shall develop a recursive algorithm that can be used for the production of small or (near-)optimal reductions. The idea is quite straightforward and is based on the next lemma.
Lemma 42. Let \( P \) be any reduction of \( \Delta \). Suppose some distribution \( \Delta' \in P \) has a reduction \( P' \). Then, \( [P \cup P' - \{\Delta'\}] \) is also a reduction of \( \Delta \).

That is, suppose \( P \) is a reduction of \( \Delta \) and \( P' \) is a reduction of some distribution \( \Delta' \in P \). Then, we can replace \( \Delta' \) with \( P' \) and keep only those minimal distributions in \( P \cup P' - \{\Delta'\} \). And the resulting set of distributions is still a reduction of \( \Delta \). Then, Lemma 42 suggests a natural, “conservative” algorithm for the recursive generation of (near)-optimal reductions. The procedure thus shall proceed as follows: we first search for a reduction of \( \Delta \) consisting of a small number of distributions in \( [L(\Delta)] \). This can be easily carried out using the incremental production approach with backtracking, by ensuring that each distribution that is used to collect pairs of independent symbols is minimally merged from \( \Delta \), followed by reduction validation. If such a reduction cannot be found, then we conclude that \( \Delta \) does not have a reduction, by Theorem 12. Now, suppose that some reduction \( P \subseteq [L(\Delta)] \) of \( \Delta \) has been found. Then, we proceed to replace each distribution \( \Delta' \in P \) using a reduction \( P' \subseteq [L(\Delta')] \) of \( \Delta' \), if it exists, as guided by Lemma 42. The same procedure is carried out until no further replacement as above can be made. If there is a reduction of \( \Delta \), then there usually exists a reduction of \( \Delta \) that consists of a small number of distributions in \( [L(\Delta)] \). It is usually much more efficient to verify these candidate reductions, since only a small number of substitutions are needed.

There is an advantage of the recursive generation approach. If there exists a reduction, then it often can be confirmed after a few number of backtracking choices. The optimality of the final reduction depends on the initial reduction and the choices made in each replacement step, which can be enumerated with a tree structure. It is of interest to know whether all the optimal reductions can be computed in this way, which is still open. Again, there is no guarantee of the optimality for the generated reductions, since we are not sure about the completeness of the strengthened substitution-based proof.

VI. CONCLUSIONS

In this work, we have further studied the notion of reduction of distributions and developed procedures for the computation of (near)-optimal reductions. In particular, we have procedures that can be used for the production of candidate reductions and reduction verification. There are still many technical problems that are left open. For example, we do not know whether the strengthened substitution-based proof for reduction validation is complete; and we still do not know whether the problem of reduction verification is decidable, although we conjecture that the answer is positive.

REFERENCES


APPENDIX A

Theorem 26 is quite convenient to apply (cf. Corollary 30). However, the condition stated in Theorem 26 is not necessary for the decomposability of \( \Lambda_{\text{cand}} \) with respect to \( \Delta' \). This shall be illustrated in the next example. Then, we will also explain how Theorem 26 can be improved.

Example 43. Consider the distribution \( \Delta = (\Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4) \), where \( \Sigma_1 = \{a, b, c, g\} \), \( \Sigma_2 = \{c, d, e\} \), \( \Sigma_3 = \{d, e, f\} \) and \( \Sigma_4 = \{e, f, g\} \). \( \Lambda_{\text{cand}} \) (the union of \( a \sqcup b \sqcup c \sqcup d \sqcup e \sqcup f \sqcup g \) is indeed decomposable with respect to the distribution \( \Delta' = (\{a, b, c, g\}, \{c, d, e, f\}, \{e, f, g\}) \), which is obtained from \( \Lambda \) by merging \( \Sigma_2 \) and \( \Sigma_3 \). However, Theorem 26 fails to prove the decomposability of \( \Lambda_{\text{cand}} \) with respect to \( \Delta' \) in this case. Indeed, we notice that the sub-alphabet \( \{c, d, e, f\} \) is the only element in \( \Delta' - \Delta \); and the set \( E_{\Delta'}(\{c, d, e, f\}) \) of boundary symbols in \( \{c, d, e, f\} \) is \( \{c, e, f\} \). It is not difficult to check that \( CR_{\Delta'}(\{c, d, e, f\}, a) = (\{\{c, d, e, f\}, b\} = \{2, 3, 4\} \neq \{1, 4\} \). With Theorem 26, we only conclude that symbol \( g \) is determined, and fail to prove the decomposability of \( \Lambda_{\text{cand}} \) with respect to \( \Delta' \). On the other hand, we observe that we have determined \( \{c, d, e, f, g\} \) after symbol \( g \) is determined. Thus, in particular, the numbers of occurrences of \( c \) and \( g \) are determined. We shall note that the only possibility is the following, which corresponds exactly to \( L(\Sigma_j) \) for \( j \in \{1, 4\} \)

\[ c \sqcup g, c \sqcup g^3, c^4 \sqcup g^4 \text{ and } c^5 \sqcup g^5. \]

Thus, the numbers of occurrences of \( c \) and \( g \) can determine the numbers of occurrences of \( a \) and \( b \) by matching \( c \) and \( g \) via \( P_{a,b,c,g} \). Thus, we have determined all the symbols in \( \Sigma \) and \( \Lambda_{\text{cand}} \) is decomposable with respect to \( \Delta' \). We note that the reason \( c \) and \( g \) together can be used for matching is that they appear together only in the sub-alphabet \( \Sigma_1 \) of \( \Delta \).

In the following, we propose an improved method for testing the decomposability of \( \Lambda_{\text{cand}} \) with respect to distribution \( \Delta' \in \Delta \).
\( \mathcal{M}(\Delta) \). Given any proper sub-alphabet \( \Sigma' \) of \( \Sigma \), we compute \( \mathcal{E}_\Delta(\Sigma') \) with respect to \( \Delta' \) as follows:

1. Set \( \Sigma_d = \Sigma' \).
2. Let \( \Sigma_c \) denote \( \{ \sigma' \in \Sigma - \Sigma_d \mid \mathcal{C}_R(\Sigma_d, \sigma') = [1, n] \} \). If \( \Sigma_c = \emptyset \), then return \( \Sigma_d \); otherwise, let \( \Sigma_d := \Sigma_d \cup \Sigma_c \).
3. If \( \Sigma_d = \Sigma \), then return \( \Sigma \); otherwise, let \( \Sigma_{c'} := \{ \sigma' \in \Sigma - \Sigma_d \mid \exists \Sigma'_d \subseteq \Sigma_d, (\Sigma'_d \geq 2) \land \left( \bigcup_{\sigma \in \Sigma'_d} \mathcal{N}(\sigma) = n - 1 \right) \land (\sigma' \cup \Sigma'_d \subseteq \Delta') \} \).
   - If \( \Sigma_{c'} = \emptyset \), then return \( \Sigma_d \); otherwise, let \( \Sigma_d := \Sigma_d \cup \Sigma_{c'} \) and go to Step 2.

Intuitively, \( \Sigma_d \) denotes the set of currently determined symbols at each step; \( \Sigma_c \) denotes the set of symbols that are determined by matching chains of distinctive symbols (see Example 24); \( \Sigma_{c'} \) denotes the set of symbols that are determined by matching two or more determined symbols in \( \Sigma_d \) that appear together only in one sub-alphabet of \( \Delta \) (see Example 43). We notice that, in Step 3), \( |\bigcup_{\sigma \in \Sigma'_d} \mathcal{N}(\sigma)| = n - 1 \) holds if and only if the symbols in \( \Sigma'_d \) appear together only in one sub-alphabet of \( \Delta \); and then, with \( |\sigma'\cup\Sigma'_d\subseteq\Delta'| \), the symbols in \( \Sigma'_d \) together can be used to determine \( \sigma' \). It is clear that the above computation terminates; and we have the next results (for brevity, we omit the proofs due to space limitation).

**Theorem 44.** Let \( \Delta' = (\Sigma'_1, \Sigma'_2, \ldots, \Sigma'_k) \) be a distribution in \( \mathcal{M}(\Delta) \). Then, \( L_{\text{cand}} \) is decomposable with respect to \( \Delta' \) if there exists a sub-alphabet \( \Sigma'_i \in \Delta' - \Delta \) such that \( \mathcal{E}_\Delta(\Sigma'_i) = \Sigma \), where \( \Delta = (\Sigma_1, \Sigma_2, \ldots, \Sigma_n) \).

Theorem 44 is much more powerful than Theorem 26. It is still an open problem whether the condition in Theorem 44 is necessary.

Let \( \mathcal{E}_\Delta(\Sigma') \) be computed as in Step 1) to Step 3) of \( \mathcal{E}_\Delta(\Sigma') \), with the exception that \( \Delta' \) is replaced with \( \Delta \) throughout. We have the next corollary.

**Corollary 45.** Let \( \Delta' = (\Sigma'_1, \Sigma'_2, \ldots, \Sigma'_k) \) be a distribution in \( \mathcal{M}(\Delta) \). Then, \( L_{\text{cand}} \) is decomposable with respect to \( \Delta' \) if there exists a sub-alphabet \( \Sigma'_i \in \Delta' - \Delta \) such that \( \mathcal{E}_\Delta(\Sigma'_i) = \Sigma \), where \( \Delta = (\Sigma_1, \Sigma_2, \ldots, \Sigma_n) \).

We have the next theorem, which can be used to establish stronger parameterized results.

**Theorem 46.** \( \Delta \) does not have a reduction if for any two different sub-alphabets \( \Sigma_i, \Sigma_j \in \Delta \) such that \( \Sigma_i \cup \Sigma_j \neq \Sigma \), it holds that \( \mathcal{E}_\Delta(\Sigma_i \cup \Sigma_j) = \Sigma \), where \( \Delta = (\Sigma_1, \Sigma_2, \ldots, \Sigma_n) \).

**Proof Sketch:** This immediately follows from Corollary 45 (cf. the proof of Corollary 30).

### APPENDIX B

We list some examples here for determining the existence of reductions for distributions, i.e., checking whether \( \bot(\Delta) \) is a reduction of \( \Delta \). In the third column, \( L_{\text{cand}} \) (app) means that \( L_{\text{cand}} \) is a counter example for \( \bot(\Delta) \) and the reasoning rule in Appendix A can be used; \( L_{\text{cand}} \) is a counter example and Theorem 26 can be used. “sub” means that the (ordinary) substitution-based proof works for \( \bot(\Delta) \). “S sub” means that the strengthened substitution-based proof has to be used (and the ordinary proof fails). For distributions that have reductions, we show the wall-clock time for validating the existence of reductions using a non-optimized Python implementation of the strengthened proof on a laptop equipped with a 1.6GHz Intel Core i5-4200U CPU.

<table>
<thead>
<tr>
<th>Distribution</th>
<th>Existence</th>
<th>Reason; Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>(ab, bc, cd, dae)</td>
<td>no</td>
<td>( L_{\text{cand}} ) (app)</td>
</tr>
<tr>
<td>(abc, be, cd, da)</td>
<td>no</td>
<td>( L_{\text{cand}} ) (app)</td>
</tr>
<tr>
<td>(abc, bce, cde, da)</td>
<td>no</td>
<td>( L_{\text{cand}} )</td>
</tr>
<tr>
<td>(abc, bce, cde, dae)</td>
<td>no</td>
<td>( L_{\text{cand}} )</td>
</tr>
<tr>
<td>(abc, be, cde, dae)</td>
<td>yes</td>
<td>sub; 0.08s</td>
</tr>
<tr>
<td>(abc, bce, cde, dae)</td>
<td>yes</td>
<td>sub; 0.10s</td>
</tr>
<tr>
<td>(abc, bce, cde, dae)</td>
<td>yes</td>
<td>sub; 0.08s</td>
</tr>
<tr>
<td>(abc, bce, cde, dae)</td>
<td>yes</td>
<td>S sub; 2.45s</td>
</tr>
<tr>
<td>(abcg, cde, def, ef g, fgca)</td>
<td>yes</td>
<td>S sub; 0.07s</td>
</tr>
<tr>
<td>(abcg, cde, def, ef g, fggh)</td>
<td>yes</td>
<td>S sub; 0.29s</td>
</tr>
</tbody>
</table>