Branching Bisimilarity on Normed BPA Is EXPTIME-complete

Chaodong He Mingzhang Huang
BASICS, Department of Computer Science and Engineering, Shanghai Jiao Tong University

Abstract—We put forward an exponential-time algorithm for deciding branching bisimilarity on normed BPA (Basic Process Algebra) systems. The decidability of branching (or weak) bisimilarity on normed BPA was once a long standing open problem which was closed by Yuxi Fu in [1]. The EXPTIME-hardness is an inference of a slight modification of the reduction presented by Richard Mayr [2]. Our result claims that this problem is EXPTIME-complete.

I. INTRODUCTION

Basic process algebra (BPA) [3] is a fundamental model of infinite state systems, with its famous counterpart in the theory of formal languages: context free grammars in Greibach normal forms, which generate the entire context free languages. In 1987, Baeten, Bergstra and Klop [4] proved a surprising result at the time that strong bisimilarity on normed BPA is decidable. This result is in sharp contrast to the classical fact that language equivalence is undecidable for context free grammar [5]. After this remarkable discovery, decidability and complexity issues of bisimilarity checking on infinite state systems have been intensively investigated. See [6], [7], [8], [9], [10] for a number of surveys.

As regards strong bisimilarity on normed BPA, Hüttel and Stirling [11] improved the result of Baeten, Bergstra and Klop using a more simplified proof by relating the strong bisimilarity of two normed BPA processes to the existence of a successful tableau system. Later, Huynh and Tian [12] showed that the problem is in \( \Sigma_2^P \), the second level of the polynomial hierarchy. Before long, another significant discovery was made by Hirshfeld, Jerrum and Moller [13] who showed that the problem can even be decided in polynomial time. Improvements on running time was made later in [14], [15], [16].

The decidability of strong bisimilarity on general BPA is affirmed by Christensen, Hüttel and Stirling [17]. 2-EXPTIME is claimed to be an upper bound by Burkart, Caucaal and Steffen [18] and is explicitly proven recently by Jančar [19]. As to the lower bound, Kiefer [20] achieves EXPTIME-hardness, which is an improvement of the previous PSPACE-hardness obtained by Srba [21].

In the presence of silent actions, however, the picture is less clear. The decidability for both weak bisimilarity and branching bisimilarity on normed BPA was once long standing open problems. For weak bisimilarity [22], the problem is still open, while for branching bisimilarity [23], [24], a remarkable discovery is made by Fu [1] recently that the problem is decidable. Very recently, using the key property developed in [1], Czerwiński and Jančar shows that there exists an exponentially large bisimulation base for branching bisimilarity on normed BPA, and by guessing, they show that the complexity of this problem is in NEXPTIME [25]. The current best lowerbound for weak bisimilarity is the EXPTIME-hardness established by Mayr [2], whose proof can be slightly modified to show the EXPTIME-hardness for branching bisimilarity as well. As to the general BPA, decidability of branching bisimilarity is still unknown.

This paper confirms that an exponential time algorithm exists for checking branching bisimilarity on normed BPA. Comparing with the known EXPTIME-hardness result, we get the result of EXPTIME-completeness. Thus the complexity class of branching bisimilarity on normed BPA is completely determined.

Basically, we introduce a family of relative bisimilarities parameterized by the reference sets, which can be represented by a decomposition base defined in this paper. The branching bisimilarity is exactly the relative bisimilarity whose reference set is the empty set. We show that the base can be approximated. The approximation procedure starts from an initial base, which is relatively trivial, and is carried on by repeatedly refining the current base. In order to define the approximation procedure and to ensure that the family of relative bisimilarities is achieved at last, a lot of technical difficulties need overcoming. Some of them are listed here:

- Relative bisimilarities (Section III) defined in this paper is superior to the corresponding concepts in the previous works [1], [25], because it does not rely on the suffix processes.
- We show that a generalized unique decomposition property holds for relative bisimilarities (Theorem 3). In the decompositions, bisimilarities with different reference sets depend and impact on each other. The notion of decomposition bases (Section V) provides an effective representation of an arbitrary family of process equivalences that satisfies the unique decomposition property.
- In an iteration of refinement operation, a new base is constructed from the old (Section VI). That is, a new family of equivalences is obtained from the old. Besides, comparing with the previous works [26], [13], [27], our refinement procedure possesses several hallmarks:
  - The new base is constructed via a globally greedy strategy, in which all the relevant equivalences with different reference sets are dealt with as a whole.
  - The refinement operation in previous works heav-
ily depends on a predefined notions of norms and decreasing transitions, which can be determined beforehand. Such a method does not work at present. Our solution is to define norms in a semantic way (Section IV). Norms, relying on the relevant equivalence relations, are dynamically changed. When we start to construct a new base, no information on norms is available thus we cannot determine whether a transition is decreasing. Our solution is to put the task of computing norms into the global iteration procedure via greedy strategy.

In previous works the order of process constants is defined in advance. In the construction of new bases, the constants are treated in order. There is no such predefined order in our algorithm. The treating order is also decided dynamically.

Equivalence checking on normed BPA is significantly harder than the related problem on totally normed BPA. For totally normed BPA, branching bisimilarity is recently shown polynomial-time decidable [27]. What is obtained in this paper is significantly stronger than previous results [28], [29], [27].

We maintain an online version of the paper [1]. It will contain more illustrative examples and remarks on subtleties.

II. PRELIMINARIES

A. Normed Basic Process Algebra

A basic process algebra (BPA) system $\Gamma$ is a triple $(C, A, \Delta)$, where $C$ is a finite set of process constants ranged over by $X, Y, Z, U, V, W$, $A$ is a finite set of actions, and $\Delta$ is a finite set of transition rules. The processes, ranged over by $\alpha, \beta, \gamma, \delta, \zeta, \eta$, are generated by the following grammar:

$$\alpha ::= \varepsilon \mid X \mid \alpha_1, \alpha_2.$$ 

The syntactic equality is denoted by $\equiv$. We assume that the sequential composition $\alpha_1, \alpha_2$ is associative up to $\equiv$ and $\varepsilon, \alpha = \alpha, \varepsilon = \alpha$. Sometimes $\alpha, \beta$ is shortened as $\alpha\beta$. The set of processes is exactly $C^*$, the finite strings over $C$. There is a special symbol $\tau$ in $A$ for silent transition. $\ell$ is invariably used to denote an arbitrary action, while $a$ are used to denote a visible (i.e. non-silent) action. The operational semantics of the processes are defined by the following labelled transition rules:

$$\Gamma \ni (X \xrightarrow{\ell} \alpha) \in \Delta, \alpha \xrightarrow{\ell} \alpha' \alpha, \beta \xrightarrow{\ell} \alpha', \beta$$

A central dot “$'$” is often used to indicate an arbitrary process. For example, we write $\alpha \xrightarrow{\ell_1}, \tau \xrightarrow{\ell_2} \beta$, or even $\alpha, \beta \xrightarrow{\ell_1}, \tau \xrightarrow{\ell_2} \beta$, to mean that there exists some $\gamma$ such that $\alpha \xrightarrow{\ell_1} \gamma$ and $\gamma \xrightarrow{\ell_2} \beta$.

If $\simeq$ is an equivalence relation on processes, then we will use $\alpha \xrightarrow{\ell} \alpha'$ to denote the fact $\alpha \xrightarrow{\ell} \alpha' \simeq \alpha$, and use $\alpha \xrightarrow{\ell} \alpha'$ to denote the fact $\alpha \xrightarrow{\ell} \alpha' \neq \alpha$. As usual, we write $\iff$ for the reflexive transitive closure of $\xrightarrow{\ell}$. We write $\iff$ for the symmetric closure of $\xrightarrow{\ell}$. Accordingly, $\xrightarrow{\ell}$ is understood as the reflexive transitive closure of $\xrightarrow{\ell}$. In other words, $\alpha \equiv \alpha'$ is understood as $\alpha \xrightarrow{\ell_1}, \ldots, \xrightarrow{\ell_n} \alpha'$.

A process $\alpha$ is normed if $\alpha \xrightarrow{\ell_1}, \ldots, \xrightarrow{\ell_n} \varepsilon$ for some $\ell_1, \ldots, \ell_n$. A BPA system $\Gamma = (C, A, \Delta)$ is normed if all the processes defined in $\Gamma$ are normed. In other words, $X$ is normed for every $X \in C$. In the rest of the paper, we will invariably use $\Gamma = (C, A, \Delta)$ to indicate a normed BPA system. A BPA system $\Gamma$ is called realtime if for every $(X \xrightarrow{\ell} \alpha) \in \Delta$, we have $\ell \neq \tau$.

A process $\alpha$ is called a ground process if $\alpha \equiv \varepsilon$. The set of ground constants is denoted by $C_G$. Apparently $C_G \subseteq C$ and $\alpha \equiv \varepsilon$ if and only if $\alpha \in C^*_G$.

Remark 1. $\Gamma$ is totally normed if rules of the form $X \xrightarrow{\tau} \varepsilon$ are forbidden. $\Gamma$ is totally normed if and only if $C_G = \emptyset$.

B. Bisimulation and Bisimilarity

In the presence of silent actions, branching bisimilarity of van Glabbeek and Weijland [23], [24] is well-known.

Definition 1. Let $\simeq$ be an equivalence relation on processes. $\simeq$ is called a branching bisimulation, if the following bisimulation property hold:

- If $\alpha \equiv \alpha'$, then $\beta \equiv \beta'$ for some $\beta'$ such that $\alpha' \equiv \beta'$.
- If $\alpha \equiv \alpha'$, then $\beta \equiv \beta'$ for some $\beta'$ such that $\alpha' \equiv \beta'$.

The branching bisimilarity $\simeq$ is the largest branching bisimulation.

Remark 2. Branching bisimulations and other bisimulation-like relations in this paper are forced to be equivalence relations. This is a technical convention.

The branching bisimilarity is a congruence relation, and it satisfies the following famous lemma.

Lemma 1 (Computation Lemma [23]). If $\alpha \equiv \alpha' \Rightarrow \alpha'' \simeq \alpha$, then $\alpha' \simeq \alpha$.

If $\Gamma$ is realtime, the branching bisimilarity is the same as the strong bisimilarity. In this paper, branching bisimilarity will be abbreviated as bisimilarity. For realtime systems, the term bisimilarity will also be used to indicate strong bisimilarity.

III. RELATIVIZED BISIMILARITY ON NORMED BPA

A. Retrospection

In [11], Yuxi Fu creates the notion of redundant processes, and discover the following Proposition [5] which turns out to be a very useful tool to show the decidability of branching bisimilarity for normed BPA.

Definition 2. A process $\alpha$ is a $\simeq$-redundant over $\gamma$ if $\alpha \gamma \simeq \gamma$.

We use $\Rd(\gamma) = \{X \mid X \gamma \simeq \gamma\}$ to indicate the set of all constants that is $\simeq$-redundant over $\gamma$. Clearly, $\Rd(\gamma) \subseteq C_G$.

The redundant processes over $\gamma$ are completely determined by the redundant constants.

Lemma 2. $\alpha \gamma \simeq \gamma$ if and only if $\alpha \in (\Rd(\gamma))^*$. 
The crucial observation in [H] is the following fact, which is also the starting point of this paper.

**Proposition 3.** Assume that \( \text{Rd}(\gamma_2) = \text{Rd}(\gamma_2) \), then \( \alpha \gamma_1 \cong \beta \gamma_1 \) if and only if \( \alpha \gamma_2 \cong \beta \gamma_2 \).

Proposition 3 inspires us to define a relativized version of bisimilarity \( \cong_R \) for certain suitable reference set \( R \), which will validate the following theorem.

**Theorem 1.** Let \( R = \text{Rd}(\gamma) \) for some \( \gamma \), then \( \alpha \cong_R \beta \) if and only if \( \alpha \cong \beta \).

Proposition 3 confirms that \( \cong_R \) does not depend on the special choice of \( \gamma \). However, it is wiser not to take Theorem 1 as the definition of \( \cong_R \) from a computational point of view. Here are the reasons.

- We cannot tell beforehand (except that we can decide \( \cong \)) whether, for a given \( R \), there exists \( \gamma \) such that \( R = \text{Rd}(\gamma) \), nor can we tell whether \( R = \text{Rd}(\gamma) \) even if both \( R \) and \( \gamma \) are given.
- The algorithm developed in this paper takes the refinement approach. Suppose that \( \cong \) is an approximation of \( \cong_R \), we can define, for instance, the \( \cong \)-redundant constants \( \text{Rd}^\cong(\gamma) \) accordingly. It is possible to run into the situation where, for a specific \( R \), there is no \( \delta \) such that \( R = \text{Rd}^\cong(\delta) \) even if both \( R \) and \( \gamma \) are given. Therefore it is advisable to make \( \cong_R \) well-defined for any \( R \) such that \( R \subseteq C_G \). Importantly, \( \cong_R \) should be defined without the knowledge of the existence of \( \gamma \).

**B. Definition of R-Bisimilarities**

Now we elaborate on the definition of \( \cong_R \). Some auxiliary notations are introduced to make things clear.

**Definition 3.** Let \( R \subseteq C_G \). Two processes \( \alpha \) and \( \beta \) are \( R \)-equal, denoted by \( \alpha =_R \beta \), if there exist \( \zeta, \alpha', \beta' \) such that \( \alpha = \zeta \alpha', \beta = \zeta \beta' \), and \( \alpha', \beta' \in R^* \).

\( R \)-equality is an equivalence relation. Two processes are \( R \)-equal if they differ only in a suffixes in \( R^* \). Eliminating a suffix in \( R^* \) from a process does not change the \( =_R \)-class.

**Lemma 4.** 1) If \( \alpha =_R \alpha \gamma \) if and only if \( \gamma \in R^* \).

2) \( \alpha =_R \epsilon \) if and only if \( \alpha \in R^* \).

**Definition 4.** \( \alpha \) is in \( R \)-normal-form (R-nf) if

1) either \( \alpha = \epsilon \),

2) or there exist \( \alpha \) and \( X \) such that \( \alpha = \alpha \gamma X \) and \( X \notin R \).

If \( \alpha =_R \alpha' \) and \( \alpha' \) is in R-nf, then \( \alpha' \) is called an \( R \)-nf of \( \alpha \).

The (unique) \( R \)-nf of \( \alpha \) is denoted by \( \alpha_R \).

\( R \)-equality is the syntactic equality on \( R \)-nf’s. In particular, \( \emptyset \)-equality is exactly the normal syntactic equality.

The transition relations can also be relativized as follows.

**Definition 5.** The \( R \)-transition relations between \( R \)-nf’s are defined as follows: We write \( \zeta \xrightarrow{R} \eta \) if there exists \( \alpha \) and \( \beta \) such that \( \zeta = \alpha_R \), \( \eta = \beta_R \), and \( \alpha \xrightarrow{R} \beta \).

Note that, if \( \alpha \xrightarrow{R} \beta \), then both \( \alpha \) and \( \beta \) must be in \( R \)-nf.

**Lemma 5.** \( \alpha_R \xrightarrow{R} \beta_R \) if and only if \( \alpha =_R \epsilon \).

Let \( \alpha \) be an \( R \)-nf and \( \alpha \xrightarrow{R} \beta \). Intuitively, if \( \alpha \neq \epsilon \), then \( \alpha \xrightarrow{R} \beta \) is induced by \( \alpha \); if \( \alpha = \epsilon \), then \( \alpha \xrightarrow{R} \beta \) is induced by one of the constants in \( R \). This important fact is formalized in the following lemma.

**Lemma 6.** \( \alpha_R \xrightarrow{R} \beta_R \) if and only if

1) either \( \alpha = \epsilon \) and \( X \xrightarrow{R} \beta' \) for some \( \beta' \) and \( X \in R \) such that \( \beta'_R = \beta_R \).

2) or \( \alpha_R \neq \epsilon \) and \( \alpha \xrightarrow{R} \beta' \) for some \( \beta' \) such that \( \beta'_R = \beta_R \).

As usual, we write \( \equiv_R \) for the reflexive transitive closure of \( \xrightarrow{R} \). We write \( \equiv \) for the symmetric closure of \( \equiv \) (i.e. \( \equiv_R \equiv \equiv_R \cup \equiv^{-1}_R \)). Accordingly, \( \alpha \equiv_R \) is understood as \( \alpha \equiv\equiv_R \equiv \alpha \), and \( \equiv_R \) is the reflexive transitive closure of \( \equiv \).

The next lemma confirms that the ground processes are robust under relativization.

**Lemma 7.** \( \alpha \equiv \epsilon \) if and only if \( \alpha_R \equiv R \epsilon \).

Now it is time for defining \( R \)-bisimilarity.

**Definition 6.** Let \( R \subseteq C_G \) and let \( \xrightarrow{\cong} \) be an equivalence relation on processes such that \( =_R \subseteq \cong \). We say \( \cong \) is an \( R \)-bisimulation, if the following conditions are satisfied whenever \( \alpha \cong \beta \):

1) if \( \alpha \equiv \epsilon \), then \( \beta \equiv \epsilon \).

2) if \( \alpha \xrightarrow{R} \alpha' \), then \( \beta_R \equiv_R \beta_R \xrightarrow{R} \beta' \) for some \( \beta' \) such that \( \alpha' \equiv \beta' \).

3) if \( \alpha \equiv \alpha' \), then \( \beta_R \equiv_R \beta_R \equiv_R \beta' \) for some \( \beta' \) such that \( \alpha' \equiv \beta' \).

The \( R \)-bisimilarity \( \cong_R \) is the largest \( R \)-bisimulation.

If \( R = \emptyset \), then \( \cong \) is exactly the ordinary bisimilarity \( \cong \).

\( R \)-bisimulations can actually be understood as the bisimulations on \( R \)-nf’s under \( R \)-transitions, as is stated below.

**Proposition 8.** \( \cong \) is an \( R \)-bisimulation if and only if whenever \( \alpha \cong \beta \):

1) if \( \alpha \equiv \epsilon \), then \( \beta \equiv \epsilon \);

2) if \( \alpha_R \equiv_R \alpha'_R \), then \( \beta_R \equiv_R \beta_R \equiv_R \beta'_R \) for some \( \beta' \) such that \( \alpha' \equiv \beta' \);

3) if \( \alpha_R \equiv \alpha'_R \), then \( \beta_R \equiv_R \beta_R \equiv_R \beta'_R \) for some \( \beta' \) such that \( \alpha' \equiv \beta' \).

Comparing with the bisimulation (Definition 1), Proposition 8 contains an extra ground preservation condition which says that a ground process cannot be related to a non-ground process in an \( R \)-bisimulation. In the case of bisimulation, this condition is also satisfied but it can be derived from other bisimulation conditions. As to \( R \)-bisimulation, this is not always the case, as is illustrated in the following example.

**Example 1.** Consider the following normed BPA \((C, A, \Delta)\):

- \( C = \{ A_0, A_1 \} \);
- \( A = \{ a, \tau \} \);
- \( \Delta \) is the set of the following rules:
Let \( R = \{ A_1 \} \), and let \( \simeq \) be the equivalence relation which relates every processes defined in \( \Gamma \) to \( \epsilon \). Clearly \( A_0 \not\simeq_R \epsilon \). However, we can show that \( (A_0, \epsilon) \) satisfies \( R \)-bisimulation conditions except for the ground preserving condition:

Considering that the \( R \)-transitions of \( \epsilon \) can be trivially matched by \( A_0 \), it remains to show that \( \epsilon \) can match the \( R \)-transitions of \( A_0 \). The unique \( R \)-transition of \( A_0 \) is \( A_0 \overset{\alpha}{\to} R \epsilon \), which can be matched by \( \epsilon \overset{\alpha}{\to} \gamma R \epsilon \) since \( A_1 \overset{\alpha}{\to} A_1 \) and \( A_1, R = \epsilon \).

The relative bisimilarity \( \simeq_R \) is not a congruence in general. For example, we may not have \( \alpha \gamma \simeq_R \beta \gamma \) even if \( \alpha \simeq_R \beta \). However, we have the following result.

**Lemma 9.** If \( \gamma \simeq_R \delta \) and \( \alpha \simeq \beta \), then \( \alpha \gamma \simeq \beta \delta \). In particular, if \( \gamma \simeq_R \delta \), then \( \alpha \gamma \simeq_R \alpha \delta \).

The computation lemma also hold for \( \simeq_R \).

**Lemma 10** (Computation Lemma for \( \simeq_R \)). If \( \alpha \Rightarrow_R \alpha \gamma \simeq_R \alpha \) then \( \alpha \gamma \simeq_R \alpha \).

**C. \( R \)-identities and Admissible Reference Sets**

\( R \)-bisimilarity has the following basic property.

**Lemma 11.** Let \( R \subseteq C_G \). If \( X \in R \), then \( X \simeq_R \epsilon \).

The converse of Lemma 11 does not hold in general. This fact leads to the following study.

**Definition 7.** Let \( R \subseteq C_G \). A process \( \alpha \) is called a \( \simeq_R \) identity if \( \alpha \simeq_R \epsilon \). We use \( \text{Id}_R \) to denote \( \{ X \mid X \simeq_R \epsilon \} \).

By Lemma 11 \( R \subseteq \text{Id}_R \subseteq C_G \). Below we will demonstrate that, as a reference set, \( \text{Id}_R \) plays a very important role. At first we state the following useful proposition for relative bisimilarities. It says that \( \simeq_R \) is monotone.

**Proposition 12.** Let \( R_1 \subseteq R_2 \subseteq C_G \). If \( \alpha \simeq_{R_1} \beta \), then \( \alpha \simeq_{R_2} \beta \).

**Corollary 13.** Let \( R_1 \subseteq R_2 \subseteq C_G \). Then, \( \text{Id}_{R_1} \subseteq \text{Id}_{R_2} \).

Intuitively, \( \simeq_R \) is the relative bisimilarity induced by regarding the constants in \( R \) as \( \epsilon \) purposely. As a logical conclusion, we have \( X \simeq_{\text{Id}_R} \epsilon \) if and only if \( X \in \text{Id}_R \). This intuitions leads to the following proposition and the corollaries.

**Proposition 14.** \( \alpha \simeq_R \beta \) if and only if \( \alpha \simeq_{\text{Id}_R} \beta \).

**Corollary 15.** Let \( R_1 \subseteq C_G \) and \( R_2 \subseteq C_G \). If \( \text{Id}_{R_1} = \text{Id}_{R_2} \), then \( \alpha \simeq_{R_1} \beta \) if and only if \( \alpha \simeq_{R_2} \beta \).

**Corollary 16.** Let \( R, S \subseteq C_G \) such that \( R \subseteq S \subseteq \text{Id}_R \), then \( \text{ld}_S = \text{ld}_R \).

A direct inference of Corollary 16 is the following fact.

**Lemma 17.** \( X \simeq_{\text{ld}_R} \epsilon \) if and only if \( X \in \text{ld}_R \). In other words, \( \text{ld}_{\text{ld}_R} = \text{ld}_R \).

The above discussions lead to the following definition.

**Definition 8.** An \( R \subseteq C_G \) is called admissible if \( R = \text{ld}_R \).

The significance of Proposition 12 Proposition 14 and their corollaries is the revelation of the following fact: The set \( \{ \simeq_R \}_{R \subseteq C_G} \) of all relative bisimilarity is completely determined by those \( \simeq_R \)'s in which \( R \) is admissible.

**Lemma 18.** For every \( R \subseteq C_G \), \( \text{ld}_R \) is admissible. \( \text{ld}_R \) is the smallest admissible set which contains \( R \).

**D. \( R \)-redundant Constants**

The properties of \( \simeq \)-redundant processes (Definition 2 and Proposition 3 in Section III-A) can now be generalized for the relative bisimilarity \( \simeq_R \).

**Definition 9.** Let \( R \subseteq C_G \). A process \( \alpha \) is \( \simeq_R \)-redundant over \( \gamma \) if \( \alpha \gamma \simeq_R \gamma \). We use \( \text{Rd}_R(\gamma) \) to denote \( \{ X \mid X \gamma \simeq_R \gamma \} \).

Note that \( \text{Rd}(\gamma) \) defined in Section III-A is exactly \( \text{Rd}_0(\gamma) \). Also note that \( \text{ld}_R \) is the same as \( \text{Rd}_R(\epsilon) \).

**Lemma 19.** If \( \gamma \simeq_R \delta \), then \( \text{Rd}_R(\gamma) = \text{Rd}_R(\delta) \).

**Lemma 20.** 1) \( \alpha \simeq_R \epsilon \) if and only if \( \alpha \in (\text{Id}_R)^* \).

2) \( \alpha \gamma \simeq_R \gamma \) if and only if \( \alpha \in (\text{Rd}_R(\gamma))^* \).

**Lemma 19** is a direct inference of Lemma 11. Lemma 20 is the strengthened version of Lemma 3.

Now we can state the fundamental theorem for \( \simeq_R \).

**Theorem 2.** Let \( R' = \text{Rd}_R(\gamma) \), then \( \alpha \simeq_R \beta \) if and only if \( \alpha \gamma \simeq_R \beta \gamma \).

**Proposition 21.** Assume that \( \text{Rd}_R(\gamma_1) = \text{Rd}_R(\gamma_2) \), then \( \alpha \gamma_1 \simeq_R \beta \gamma_1 \) if and only if \( \alpha \gamma_2 \simeq_R \beta \gamma_2 \).

**Proposition 22.** Suppose that \( \gamma \simeq_R \delta \) and let \( R' = \text{Rd}_R(\gamma) = \text{Rd}_R(\delta) \). Then \( \alpha \gamma \simeq_R \beta \delta \) if and only if \( \alpha \simeq_R \beta \).

Theorem 4 and Proposition 41 are the strengthened versions of Theorem 1 and Proposition 1. Proposition 22 is an inference of Lemma 9 and Theorem 2. Theorem 4 and Proposition 22 act as the relativized version of the congruence property and the cancellation law.

The following lemma is an inference of Theorem 1.

**Lemma 23.** \( \text{Rd}(\gamma) = \text{Rd}(\delta) \).

In the following we discuss the significance of the admissible reference sets. First it is easy to see the following fact according to Proposition 14.

**Lemma 24.** \( \text{Rd}_R(\gamma) = \text{Rd}_R(\gamma) \) for every \( \gamma \) and \( R \).

The following Lemma ensures that the admissible set is preserved under the 'redundant' operation.

**Lemma 25.** If \( R = \text{Rd}_R(\gamma) \) for some \( \gamma \), then \( R \) is admissible.

**Remark 3.** Even if \( R \) is admissible, it is not guaranteed that \( R \in \text{Rd}_R(\gamma) \) for some \( \gamma \) and \( R' \).

**E. Unique Decomposition Property for \( R \)-bisimilarities**

When \( \Gamma \) is real-time, the set \( C \) of process constants can be divided into two disjoint sets: primes \( \text{Pr} \) and composites \( \text{Cm} \). Every process \( \alpha \) is bisimilar to a sequential composition of
prime constants $P_1, \ldots, P_r$, and moreover, the prime decomposition is unique (up to bisimilarity). That is, if $P_1, \ldots, P_r \simeq Q_1, \ldots, Q_s$, then $r = s$ and $P_i \sim Q_i$ for every $1 \leq i \leq r$. This property is called unique decomposition property, which is first established by Hirshfeld et al. in [13]. When $\Gamma$ is totally normed, the unique decomposition property still holds [27].

If $\Gamma$ is not totally normed, the existence of redundant processes broken the unique decomposition property in the above sense. However, we expound that, apart from the existence of redundant constants, the relative bisimilarities $\{\simeq_r\}$ still enjoys the unique decomposition property (Theorem 3). This ‘weakened’ version of unique decomposition property is still called unique decomposition property in this paper.

**Definition 10.** Let $R \subseteq C_G$, and $X \in C$.
- $X$ is a $\simeq_R$-composite if $X \simeq_R \alpha X'$ for some $X'$ and $\alpha$ such that $X' \notin {\mathcal{I}}_R$ and $\alpha \notin {\mathcal{R}}_d({X'})$.
- $X$ is a $\simeq_R$-prime if $X$ is neither a $\simeq_R$-identity nor a $\simeq_R$-composite.

According to Definition 10, a constant $X \in C$ must act as one of the three different roles: $\simeq_R$-identity, $\simeq_R$-composite, or $\simeq_R$-prime. We will use $P_{\mathcal{R}}$ and $C_{\mathcal{R}}$ to indicate the set of $\simeq_R$-primes and $\simeq_R$-composites, respectively. Clearly $P_{\mathcal{R}} = P_{\mathcal{R}d}$ and $C_{\mathcal{R}} = C_{\mathcal{R}d}$.

**Definition 11.** We call $P_r, P_{r-1}, \ldots, P_1$ a $\simeq_R$-prime decomposition of $\alpha$, if $\alpha \simeq_R P_r P_{r-1} \ldots P_1$, and $P_i$ is a $\simeq_R$-prime for $1 \leq i \leq r$, if $R_i = {\mathcal{I}}_R$ and $R_{i+1} = {\mathcal{R}}_d(P_i)$ for $1 \leq i < r$.

Note that according to Lemma 25 every $R_i$ for $1 \leq i \leq r$ is admissible.

The following ‘relativized prime process property’ is crucial to the unique decomposition property (Theorem 3).

**Lemma 26.** Suppose that $X$, $Y$ are $\simeq_R$-primes and $\alpha X \simeq_R \beta Y$. Then $X \simeq_R Y$.

**Theorem 3** (Unique Decomposition Property for $R$-bisimilarities). Let $P_1, P_{r-1}, \ldots, P_1$ and $Q_1, Q_{s-1}, \ldots, Q_1$ be $\simeq_R$-prime decompositions. Let $R_i$, $S_i = {\mathcal{I}}_R$ and let $R_{i+1} = {\mathcal{R}}_d(P_i)$ for $1 \leq i < r$ and $S_{j+1} = {\mathcal{R}}_d(Q_j)$ for $1 \leq j < s$. Then, $r = s$, $R_i = S_i$ and $P_i \simeq_R Q_i$ for $1 \leq i \leq r$.

IV. NORMS AND DECREASING BISIMULATIONS

A. Syntactic Norms vs. Semantic Norms

When $\Gamma$ is realtime, a natural number called norm is assigned to every process. The norm is the least number $k$ such that $\alpha \rightarrow^{a_1} \rightarrow^{a_2} \ldots \rightarrow^{a_k} \epsilon$ for some $a_1, a_2, \ldots, a_k$.

The norm for realtime systems is both syntactic (static) and semantic (dynamic). It is syntactic because its definition does not rely on bisimilarity, and it can be efficiently calculated via greedy strategy merely with the knowledge of rules in $\Delta$. It is semantic, because the norm of a realtime process $\alpha$ is the least number $k$ such that $\alpha \rightarrow^{a_1} \rightarrow^{a_2} \ldots \rightarrow^{a_k} \epsilon$ for some $a_1, a_2, \ldots, a_k$.

Therefore, we get the coincidence of the syntactic norm and semantic norm for realtime systems. For non-realtime systems, however, the syntactic norms and the semantic ones do not coincide any more. They must be studied separately.

B. Strong Norms and Weak Norms

We define two syntactic norms for non-realtime systems. The strong norm takes silent actions into account while the weak norm neglects the contribution of silent actions.

**Definition 12.** The strong norm of $\alpha$, denoted by $|\alpha|_{st}$, is the least number $k$ such that $\alpha \rightarrow^{\ell_1} \rightarrow^{\ell_2} \ldots \rightarrow^{\ell_k} \epsilon$ for some $\ell_1, \ell_2, \ldots, \ell_k$.

The weak norm of $\alpha$, denoted by $|\alpha|_{wk}$, is the least number $k$ such that $\alpha \rightarrow^{a_1} \rightarrow^{a_2} \ldots \rightarrow^{a_k} \epsilon$ for some $a_1, a_2, \ldots, a_k$.

**Lemma 27.**
1) $|\alpha|_{st} = |\epsilon|_{wk} = 0$;
2) $|\alpha|_{st} + |\beta|_{st} = |\alpha|_{wk} + |\beta|_{wk}$.

**Lemma 28.**
1) $|\alpha|_{st} = 0$ if and only if $\alpha = \epsilon$;
2) $|\alpha|_{wk} = 0$ if and only if $\alpha \in C_G^1$ (i.e. $\alpha \to \epsilon$).

**Lemma 29.** If $\alpha \simeq_R \beta$, then $|\alpha|_{wk} = |\beta|_{wk}$.

**Lemma 30.** $|X|_{st}$ is exponentially bounded for every $X$.

C. The Semantic Norms

The semantic norms play an important role in our algorithm. They depend on the involved semantic equivalence. Let $\simeq$ be a process equivalence. A transition $\alpha \rightarrow^{\ell} \alpha'$ is called $\simeq$-preserving if $\alpha' \simeq \alpha$.

**Definition 13.** Let $\simeq$ be a process equivalence. The $\simeq$-norm of $\alpha$, denoted by $||\alpha||_{\simeq}$, is the least number $k$ such that

$$
\alpha \simeq . \rightarrow^{\ell_1} . \simeq . \rightarrow^{\ell_2} . \simeq . \ldots . \rightarrow^{\ell_k} . \simeq . \epsilon
$$

for some $\ell_1, \ell_2, \ldots, \ell_k$. If $||\alpha||_{\simeq} = k$, then any transition sequence of the form

$$
\alpha \simeq . \rightarrow^{\ell_1} . \simeq . \rightarrow^{\ell_2} . \simeq . \ldots . \rightarrow^{\ell_k} . \simeq . \epsilon
$$

is called a witness path of $\simeq$-norm for $\alpha$. The length of the witness path is $k$.

Clearly, the $\simeq$-norms have the following basic fact:

**Lemma 31.** If $\alpha \simeq \beta$, then $||\alpha||_{\simeq} = ||\beta||_{\simeq}$.

If $\simeq$ is an arbitrary equivalence relation, the witness path does not always exist, because it is not always the case $\alpha \rightarrow^{\ell} \beta$ whenever $\alpha \simeq \beta$. This is one of the motivations of the forthcoming notion of decreasing bisimulation (Definition15). For the moment, we introduce the $\simeq$-decreasing transitions.

**Definition 14.** A transition $\alpha \rightarrow^{\ell} \alpha'$ is $\simeq$-decreasing if $||\alpha'||_{\simeq} < ||\alpha||_{\simeq}$.

According to Definition13 $||\alpha'||_{\simeq} = ||\alpha||_{\simeq} - 1$ if $\alpha \rightarrow^{\ell} \alpha'$ is a $\simeq$-decreasing transition. In witness path (1), every transition $\rightarrow^{\ell}$ must be $\simeq$-decreasing for $1 \leq i < k$.

**Definition 15.** An process equivalence $\simeq$ is a decreasing bisimulation, if the following conditions are satisfied:

1) If $\alpha \simeq \epsilon$, then $\alpha \to \epsilon$. 
2) If $\alpha \simeq \beta$ and $\alpha \xrightarrow{\ell} \alpha'$ is a $\simeq$-decreasing transition, then there exist $\beta''$ and $\beta'$ such that $\beta \xrightarrow{\simeq} \beta'' \xrightarrow{\ell} \beta'$ and $\alpha' \simeq \beta'$.

Decreasing bisimulation is a weaker version of bisimulation. The difference lies in that only decreasing transitions need to be matched. Be aware that the transition $\beta'' \xrightarrow{\ell} \beta'$ in Definition 15 is forced to be $\simeq$-decreasing.

Let $\simeq$ be a decreasing bisimulation. Then any $\simeq$-decreasing transition of $\alpha$ can be extended to a witness path of $\simeq$-norm of $\alpha$. The norm $\|\alpha\|_{\simeq}$ is equal to the least number of decreasing transitions from $\alpha$ to $\epsilon$.

Nearly all equivalences appearing in this paper are decreasing bisimulation. For example:

**Proposition 32.** $\simeq_R$ is a decreasing bisimulation for every $R \subseteq C_G$.

There is no need to define the so-called $R$-decreasing bisimulation. The following lemma confirms that, for decreasing transitions, $\xrightarrow{\ell} R$ and $\xrightarrow{\ell}$ are essentially the same.

**Lemma 33.** If $\alpha \xrightarrow{\ell} R \beta$ is $\simeq_R$-decreasing, then $\alpha \xrightarrow{\ell} \beta'$ for some $\beta'$ such that $\beta' R = \beta$.

The following Lemma provides a bound for semantic norms.

**Lemma 34.** If $\simeq$ is a decreasing bisimulation, then $\|\alpha\|_{\simeq} \leq |\alpha|_{st}$ for every $\alpha$.

**D. Decreasing Bisimulation with $R$-Expansion of $\simeq$**

Based on decreasing transitions, we can define a special notion called decreasing bisimulation with $R$-expansion of $\simeq$, which will be taken as the refinement operation in our algorithm. This notion is crucial to the correctness of the refinement operation. The readers are suggested to review Definition 6 and Definition 15 before going on.

**Definition 16.** Let $\simeq$ and $\approx$ be two equivalences on processes such that $\equiv R \subseteq \simeq \subseteq \approx$. We say that $\simeq$ is an $R$-expansion of $\approx$ if the following conditions hold whenever $\alpha \simeq \beta$:

1) If $\alpha \Rightarrow \epsilon$ if and only if $\beta \Rightarrow \epsilon$.
2) If $\alpha \mathrel{\not\Rightarrow} \alpha'$, then either $\beta R \mathrel{\not\Rightarrow} R \beta'$ for some $\beta'$ such that $\alpha' \simeq \beta'$.
3) If $\alpha \Rightarrow \alpha'$, then $\beta R \Rightarrow R \beta'$ for some $\beta'$ such that $\alpha' \simeq \beta'$.

We say that $\approx$ is an decreasing bisimulation with $R$-expansion of $\simeq$ if $\simeq$ is both a decreasing bisimulation and an $R$-expansion of $\approx$.

The following lemma provides another characterization of the decreasing bisimulation with $R$-expansion of $\approx$.

**Lemma 35.** Assume that $= R \subseteq \simeq \subseteq \approx \subseteq = \simeq$ is an decreasing bisimulation with $R$-expansion of $\approx$ if and only if following conditions hold whenever $\alpha \simeq \beta$ and $\alpha, \beta$ are in $R$-nf:

1) if $\alpha \Rightarrow R \epsilon$, then $\beta \Rightarrow R \epsilon$;
2) if $\alpha \not\Rightarrow R \alpha'$, being $\simeq$-decreasing, then $\beta \Rightarrow R \beta'$ for some $\beta'$ such that $\alpha' \simeq \beta'$;
3) if $\alpha \not\Rightarrow R \alpha'$, not being $\simeq$-decreasing, then $\beta \Rightarrow R \beta'$ for some $\beta'$ such that $\alpha' \simeq \beta'$;
4) if $\alpha \Rightarrow R \alpha'$, being $\simeq$-decreasing, then $\beta \Rightarrow R \beta'$ for some $\beta'$ such that $\alpha' \simeq \beta'$;
5) if $\alpha \Rightarrow R \alpha'$, not being $\simeq$-decreasing, then $\beta \Rightarrow R \beta'$ for some $\beta'$ such that $\alpha' \simeq \beta'$.

**V. Decomposition Bases**

In this section, we define a way for finitely representing a family of equivalences which satisfies unique decomposition property in the sense of Theorem 3. Such family of equivalences include $\{\simeq_R\}_R$ and all the intermediate families of equivalences constructed during the iterations. This finite representation is named decomposition base.

**A. $R$-blocks and $R$-orders**

To make our algorithm easy to formulate, we need some technical preparations. The reason will be clear later.

**Definition 17.** Let $R \subseteq C_G$. We call that $\alpha$ is $R$-associate to $\beta$ if $\alpha \Rightarrow R \beta$. Let $X \in C \setminus \{\}$. The $R$-block related to $X$, denoted by $[X]_R$ is the set of all the constants which is $R$-associate to $X$. Namely, $[X]_R \overset{\text{def}}{=} \{Y \mid X \Rightarrow R Y\}$. We use the term block to specify any $R$-block for $R \subseteq C_G$.

Clearly, two $R$-blocks coincide when they overlap. Thus $R$-blocks compose a partition of $C \setminus \{\}$. The partition is denoted by $C_R \overset{\text{def}}{=} \{[X]_R \mid X \in C \setminus \{\}\}$.

We call a reference set $R$ qualified if $X \Rightarrow R \epsilon$ cannot happen for every $X \not\in R$. The unqualified $R$'s can be predetermined. They are useless from now to the end of this paper. From now on we assume that every reference set $R$ is qualified. For example when we write ‘for every $R \subseteq C_G$', we refer to every qualified $R$ which is a subset of $C_G$. In particular, every admissible set is qualified.

**Lemma 36.** All constants in a block $[X]_R$ are $R$-bisimilar.

**Lemma 37.** If $[X]_R \neq [Y]_R$ and $X \Rightarrow R Y$, then $Y \not\Rightarrow R X$.

The behaviours of $[X]_R$ can be more than the total behaviours of its member constants. All the processes associate to $X$ should be taken into account. It is possible that $X \Rightarrow R \zeta X'$ for some ground process $\zeta$. For instance, we can have $X \Rightarrow R Z \Rightarrow R \zeta Y \Rightarrow R Y \Rightarrow R X$. In this example, $X, Y, Z, \zeta, X, \zeta Y, \zeta Z$ are mutually $R$-associate. Thus the behaviour of $\zeta$ must be taken into account.

**Definition 18.** $Y$ is an $R$-propagating of $X$ (or $[X]_R$) if $X \Rightarrow R Y \zeta X'$ for some $\zeta$ and $X'$. (In this case we must have $X' \Rightarrow X$, and $Y \zeta$ are ground.)

**Lemma 38.** If $Y$ is an $R$-propagating of $X$, then $Y \in R^d(X)$.

**Lemma 39.** Suppose $X \Rightarrow R \zeta X' \xrightarrow{\ell} R \zeta' X'$ such that $\zeta' X' \not\Rightarrow R X$. Then $X' \in [X]_R$, and $\zeta = Y \gamma$ for some $Y$ and $\gamma$ such that

- $Y$ is an $R$-propagating of $[X]_R$. 

\[ Y \xrightarrow{\ell} \alpha \quad \text{and} \quad \zeta' = \alpha \gamma. \]
\[ X \simeq_R \gamma X. \text{ (i.e. } \gamma \in (\text{Rd}_R(X))^*) \]
\[ Y, X \xrightarrow{\ell} R \alpha X \quad \text{with} \quad Y, X \simeq_R \zeta X' \simeq_R X \quad \text{and} \quad \alpha X \simeq_R \zeta' X \approx \zeta' X'. \]

Lemma 39 shows that the behaviours of \([X]_R\) are completely determined by the associate constants and the propagating constants of \(X\), which leads to the following definition.

**Definition 19.** The R-derived transition \( \xrightarrow{\ell} \) is defined as follows:

1. Let \( \hat{X} \in [X]_R \) and \( \hat{X} \xrightarrow{\ell} R \alpha \). If either \( \ell \neq \tau \), or \( \ell = \tau \) and \( \alpha \neq x \), then \( [X]_R \xrightarrow{\ell} R \alpha \).
2. Let \( Y \) be an \( R \)-propagating of \([X]_R\) and \( Y \xrightarrow{\ell} R \alpha \). If either \( \ell \neq \tau \), or \( \ell = \tau \) and \( \alpha \neq x \), then \( [X]_R \xrightarrow{\ell} R \alpha X \).

**Lemma 40.** Suppose \( X \xleftrightarrow{\ell} \cdot \xrightarrow{\ell} R \alpha \). If either \( \ell \neq \tau \), or \( \ell = \tau \) and \( \alpha \neq x \), then \( [X]_R \xleftrightarrow{\ell} R \cdot \simeq R \alpha \).

It is technically convenient to treat the \( R \)-blocks as the basic objects in the algorithm, because of the following lemma.

**Lemma 41.** If \([X]_R \xrightarrow{\tau} R \cdot \xrightarrow{\ell} Y \), then \([Y]_R \neq [X]_R \).

Finally, we can define an order on \( R \)-blocks based on Lemma 37. For every \( R \), we fix a linear order less than \( R \) such that whenever \([X]_R \prec [Y]_R \), we have \( X \neq Y \).

**Lemma 42.** If \([X]_R \xrightarrow{\tau} R \cdot \xrightarrow{\ell} Y \), then \([Y]_R \prec [X]_R \).

### B. Decomposition Bases

A decomposition base \( B \) is a family of \( \{B_R\}_{R \subseteq C_G} \) in which every \( B_R \) is a quintuple \((Id_R^B, Pr_R^B, Cm_R^B, dcmp_R, Rd_R^B)\).

- \( Id_R^B \) is a subset of ground constants called \( B_R \)-identities.
- \( Cm_R^B \) specifies the set of \( B_R \)-composites. A \( B_R \)-composite is an \( Id_R^B \)-block.
- \( Pr_R^B \) specifies the set of \( B_R \)-primes. A \( B_R \)-prime is an \( Id_R^B \)-block.
- \( Rd_R^B \) is a function whose domain is \( Pr_R^B \). Let \([X]Id_R^B \) be a \( B_R \)-prime. The value \( Rd_R^B([X]Id_R^B) \) is a set of ground constants which are called \( B_R \)-redundant over \([X]Id_R^B \).
- \( dcmp_R \) is a function whose domain is \( Cm_R^B \). Let \([X]Id_R^B \) be a \( B_R \)-composite. The value \( dcmp_R([X]Id_R^B) \) is the \( B_R \)-decomposition of \([X]Id_R^B \), which is a set of blocks \( [X_1]R_1, [X_2]R_2, \ldots, [X_r]R_r \) with \( r \geq 1 \), \( R_1 \models Id_R^B \), \( X_1, \ldots, X_r \in Pr_R^B \), and \( R_{i+1} = Rd_R^B([X_i]R_i) \) for every \( 1 \leq i < r \).

To make a decomposition base \( B \) work properly, we need the following constraints:

1. \( R \subseteq Id_R^B \subseteq C_G \).
2. If \( R \subseteq S \), then \( Id_R^S \subseteq Id_S^B \). If \( R \subseteq S \subseteq Id_R^B \), then \( Id_R^B = Id_S^B \). In particular, \( Id_{Id_R^B}^B = Id_R^B \).
3. \( B_R = B_{Id_R^B} \) for every \( R \). When \( R = Id_R^B \), \( R \) is called \( B \)-admissible. A base is completely determined by those \( B_R \) in which \( R \) is \( B \)-admissible.

4. If \( R \) is \( B \)-admissible, then \( Cm_R^B \) and \( Pr_R^B \) are a partition of \( R \)-blocks: \( Cm_R^B \cup Pr_R^B = C_R \) and \( Cm_R^B \cap Pr_R^B = \emptyset \).
5. \( Rd_R^B([X]Id_R^B) \) is \( B \)-admissible provided that \([X]Id_R^B \) is a \( B \)-prime. Thus \( dcmp_B \) is well-defined.

A decomposition base \( B \) defines a family of string rewriting system \( \{ \xrightarrow{\tau} \}_{R \subseteq C_G} \). The family of \( B_R \)-reductions are defined according to the following structural rules.

A \( B_R \)-reduction relation is deterministic. Thus for any process \( \alpha \), the \( B_R \)-normal-form (in the sense of string rewriting systems) is unique, and it is called the \( B_R \)-decomposition of \( \alpha \). We use the notation \( dcmp^B_R(\alpha) \) to indicate the \( B_R \)-decomposition of \( \alpha \). Two processes \( \alpha \) and \( \beta \) are \( B_R \)-equivalent, notation \( \alpha \equiv B_R \beta \), if they have the same \( B_R \)-decomposition.

**Lemma 43.** \( \alpha \equiv B_R \beta \) if and only if \( dcmp^B_R(\alpha) = dcmp^B_R(\beta) \).

According to \( B_R \)-reduction rules, we have the following characterization of \( dcmp^B_R(\alpha) \).

**Lemma 44.** If \( R \) is \( B \)-admissible, then

- \( dcmp^B_R(\varepsilon) = \varepsilon \).
- If \( X \in R \), then \( dcmp^B_R(\gamma X) = dcmp^B_R(\gamma) \).
- If \( X \not\in R \), then \( dcmp^B_R(\gamma X) = dcmp^B_R(\gamma, [X]R) \).
- If \([X]_R \in Cm_R^B \), then \( dcmp^B_R(\gamma, [X]R) = dcmp^B_R(\gamma, Cm_R^B([X]R)) \).

If \( R \) is not \( B \)-admissible, then \( dcmp^B_R(\alpha) = dcmp_{Id_R^B}(\alpha) \).

We list some basic facts.

**Lemma 45.** \( \alpha \equiv B_R \varepsilon \) if and only if \( \alpha \in Id_R^B \). When \( R \) is \( B \)-admissible, \( \alpha \equiv B_R \varepsilon \) if and only if \( \alpha \in R \).

**Lemma 46.** If \( X_1, X_2 \in [X]_R \), then \( X_1 \equiv B_R X_2 \) and \( \|X_1\|_R = \|X_2\|_R \).

We can write \( \|X\|_R \) for any \( \hat{X} \in [X]_R \).

**Lemma 47.** \( \|X\|_R \geq 1 \) if \( R \) is \( B \)-admissible and \( X \not\in R \).

**Lemma 48.** If \( \equiv B_R \) is a decreasing bisimulation, then the size of \( dcmp^B_R([X]R) \) is exponentially bounded.

In the following the superscript \( B \) will often be omitted if \( B \) is clear from the context. For example sometimes we write \( Pr_R^B \) for \( Pr_R^B \).

### C. Representing \( \simeq_R \) via Decomposition Base

We define a decomposition base \( \hat{B} \) which can represent \( \simeq_R \) for every \( R \). That is, \( \equiv B \beta \) if and only if \( \alpha \simeq_R \beta \).
is crucial. Moreover, there are other subtleties which is deserve
to be mentioned.

Lemma 49. All constants in a block \([X]_{\id_R}\) are \(R\)-bisimilar.

The description of \(\bar{B} = \{\id_R, \pr_R, \cm_R, \dcr_R, \rd_R\}_R\) relies on the family of orders \(\prec_R\) defined in Section [V-A]
It contains three steps:

- In the first step, we determine \(\id_R\) for every \(R\): \(\id_R = \id_R\). According to Proposition [12] Proposition [13] and
their corollaries, \(\id_R\) satisfies constraints 1–3 in Section [V-B] In particular, \(R\) is admissible if and only if \(R\) is \(\bar{B}\)-admissible.
- In the second step, we determine other constituents of \(\bar{B}_R\) for every \(\bar{B}\)-admissible \(R\):
  \[
  \pr_R = \left\{ [X]_R \mid X \in \pr_R \text{ and } X \not\prec_R Y \text{ for every } Y <_R X \right\}
  \]
  \[
  \cm_R = \left\{ [X]_R \mid X \in \cm_R, \text{ or } X \in \pr_R \text{ and } Y \prec_R X \text{ for some } Y \right\}
  \]
  \[
  \text{if } [X]_R \in \pr_R, \text{ then } \rd_R([X]_R) = \{Y \mid YX \approx_R X\}. \text{ Be aware that } \rd_R([X]_R) \text{ is admissible (also }
  \bar{B}\text{-admissible) according to Lemma [25].}
  \]
- In the third step, for every non-\(\bar{B}\)-admissible \(R\), \(\bar{B}_\text{id}_R\) is assigned to \(\bar{B}_R\). That is, \(\pr_R := \pr_\text{id}_R\), \(\cm_R := \cm_\text{id}_R\), and so on.

Pay special attention to the descriptions of \(\pr_R\) and \(\cm_R\). They have slightly different from \(\pr_R\) and \(\cm_R\). Semantically,
if \(X \in \pr_R\) and \(X \approx_R Y\), then \(Y \in \pr_R\). In the syntactic description of \(\pr_R\) and \(\cm_R\), we need the \(\bar{B}\)-\(R\)-primes to be
absolutely unique, which is accomplished via \(<_R\). The orders
\(\prec_R\) take effects in double means: Let \(R\) be admissible, then

1) Among the \(R\)-blocks of \(\approx_R\)-primes, there is exactly one distinguished \(R\)-block that is qualified as a \(\bar{B}\)-prime,
which is the \(<_R\)-minimum one in the related \(\approx_R\)-class.
2) Let \([X]_R\) be a \(\bar{B}\)-\(R\)-prime. If \([X]_R \not\approx_R \alpha\), then \(X \not\approx_R \alpha\).

If \(X \implies_R Y \not\implies_R X\), then \(X \not\approx_R Y\).

Every decomposition base constructed during the refinement
procedure will satisfy these two properties.

Ultimately we have the following coincidence result.

Proposition 50. \(\alpha \approx_R \beta\) if and only if \(\alpha \approx_R^\bar{B} \beta\).

VI. DESCRIPTION OF THE ALGORITHM

Our algorithm takes the partition refinement approach. The
purpose is to figure out the \(\bar{B}\) defined in Section [V-C] The strategy
is to start with a special initial base \(B_0\) satisfying \(\bar{B} \subseteq B_0\) and iteratively refine it. We will use notation \(B \subseteq D\)
to mean that \(\bar{B} \subseteq D\) for every \(R\). The refinement operation
will be denoted by \(\text{Ref}\). By taking \(B_{i+1} = \text{Ref}(B_i)\), we have
a sequence of decomposition bases \(B_0, B_1, B_2, \ldots\) such that
\(B_0 \supseteq B_1 \supseteq B_2 \supseteq \ldots\). The correctness of the refinement
operation adopted in this paper depends on the following
requirements, which will be proved gradually:

1) \(\bar{B} \subseteq B_0\).
2) \(\text{Ref}(\bar{B}) = \bar{B}\).
3) If \(\bar{B} \subseteq B\), then \(\bar{B} \subseteq \text{Ref}(B) \subseteq B\).

According to the above three requirements, once the sequence
\(\{B_i\}_{i \in \omega}\) becomes stable, say \(B_i = B_{i+1}\) for some \(i\), we can
affirm that \(\bar{B} = B_i\).

On the whole, our algorithm is an iteration:

1) Compute the initial base \(B_0\) and let \(D := B_0\).
2) Compute the new base \(B\) from the old base \(D\).
3) If \(B\) equals \(D\) then halt and return \(B\).
4) \(D := B\) and go to step 2.

Apparently, the algorithm relies on the initial base and the refinement step which computes \(B = \text{Ref}(D)\) from \(D\).

A. Relationships between Old and New Bases

Before describing the algorithm in details, we investigate the
relationship between two bases \(B\) and \(\bar{D}\) assume that \(B \subseteq D\).

Lemma 51. If \(B \subseteq D\), then \(\id_R^B \subseteq \id_R^\bar{D}\) for every \(R\).

Remark 4. An interesting consequence according to
Lemma [51] is that, if \(R = \id_R^D\), then \(R \subseteq \id_R^B\) must hold
because \(R \subseteq \id_R^B \subseteq \id_R^\bar{D}\). This confirms the fact that, during
the iteration of refinement, once \(R\) becomes \(\bar{B}\)-admissible, it
preserves admissibility in the future.

Lemma 52. If \(B \subseteq D\) and \(\id_R^B \subseteq \id_R^\bar{D}\), then \(\pr_R^B \subseteq \pr_R^\bar{D}\).

Lemma 53. If \(B \subseteq D\), \(\id_R^B = \id_R^\bar{D}\), and \(\pr_R^B = \pr_R^\bar{D}\), then
\(\pr_R^B \subseteq \pr_R^\bar{D}\).

Lemma 54. If \(B \subseteq D\), and moreover \(\id_R^B = \id_R^\bar{D}\), \(\pr_R^B = \pr_R^\bar{D}\),
and \(\rd_R^B \subseteq \rd_R^\bar{D}\) for every \(R\), then \(B = D\).

The purpose of Lemma [51] to Lemma [54] is to get the following
fact.

Proposition 55. The total number of iterations (i.e. refinement
operations) in our algorithm is exponentially bounded.

B. The Initial Base

The initial base \(B_0\) is defined as follows:

- \(\id_R := C_G\) is \{\(X \in C \mid \|X\|_{wk} = 0\) for every \(R\).

Thus \(C_G\) is the only \(B_0\)-admissible set.

- If \(C_G = C\), then \(\pr_R = \emptyset\); else, \(\pr_R := \{[P]_{C_G}\}
where \([P]_{C_G}\) is the \(<_{C_G}\)-minimum \(C_G\)-block satisfying
\([P]_{wk} = 1\).

- \(\cm_R = C_G \setminus \pr_R\).

- \(\dcr_R([X]_{C_G}) = [P]_{C_G} \ldots [P]_{C_G}\) if \([X]_{C_G} \in \cm_R\).

- \(\rd_R([P]_{\id_R}) = C_G\), if \(\pr_R = \{[P]_{\id_R}\}\).

Notice that for every \(R\), \(B_{0,R}\) is the same.

Lemma 56. \(\alpha \approx_R \beta\) if and only if \(\|\alpha\|_{wk} = \|\beta\|_{wk}\).
Lemma 57. \( \hat{B} \subseteq B_0 \). Namely, \( \simeq_R \subseteq \hat{B}_R \) for every \( R \).

One can check that all the five constraints described in Section VI-B are satisfied by \( \hat{B} \).

C. \( \hat{B}_R \) as Decreasing Bisimulations with R-Expansion of \( \frac{\hat{D}}{R} \)

We start to define new base \( \hat{B} \) from the old base \( \hat{D} \). This is the core of our algorithm. The newly constructed \( \hat{B}_R \) is made to be a decreasing bisimulation with \( R \)-expansion of \( \frac{\hat{D}}{R} \). Referring to Lemma [35], we have \( \hat{B}_R \subseteq \frac{\hat{D}}{R} \), and for every \( \alpha, \beta \) are in \( R \)-nf, the following conditions hold whenever \( \alpha \hat{\equiv}_R \beta \):

1) if \( \alpha \xrightarrow{R} \epsilon \), then \( \beta \xrightarrow{R} \epsilon \);
2) Whenever \( \alpha \xrightarrow{R} \alpha' \),
   a) if \( \alpha \xrightarrow{R} \alpha' \) is \( \hat{B}_R \)-decreasing, then \( \beta \xrightarrow{R} \cdot \alpha' \) for some \( \beta' \) such that \( \alpha' \hat{\equiv}_R \beta' \);
   b) if \( \alpha \xrightarrow{R} \alpha' \) is not \( \hat{B}_R \)-decreasing and \( \alpha \hat{\equiv}_R \alpha' \), then \( \beta \hat{\equiv}_R \beta' \) for some \( \beta' \) such that \( \alpha' \hat{\equiv}_R \beta' \);
3) Whenever \( \alpha \xrightarrow{R} \alpha' \),
   a) if \( \alpha \xrightarrow{R} \alpha' \) is \( \hat{B}_R \)-decreasing, then \( \beta \xrightarrow{R} \cdot \alpha' \) for some \( \beta' \) such that \( \alpha' \hat{\equiv}_R \beta' \);
   b) if \( \alpha \xrightarrow{R} \alpha' \) is not \( \hat{B}_R \)-decreasing, then \( \beta \hat{\equiv}_R \beta' \) for some \( \beta' \) such that \( \alpha' \hat{\equiv}_R \beta' \).

The above conditions will be called expansion conditions in the following. Our task is to construct \( \hat{B} \) from \( \hat{D} \) and validate these expansion conditions. From expansion conditions we can see that, in case \( \hat{B}_R = \frac{\hat{D}}{R} \), \( \hat{B}_R \) must be an \( R \)-bisimulation. Thus when \( \simeq_R \subseteq \frac{\hat{D}}{R} \), we must have \( \hat{B}_R \subseteq \frac{\hat{D}}{R} \).

Basicly, the construction contains three steps:

1) Determine \( \text{Id}_R \) for every qualified \( R \). After that, we know whether a given \( R \) is \( B \)-admissible. Note that some \( R \)'s which are not \( D \)-admissible can be \( B \)-admissible.

2) Determine other constituents of \( B_R \) for every \( B \)-admissible \( R \).

3) For non-\( B \)-admissible \( R \)'s, \( \text{Id}_{\text{Id}_R} \) is copied to \( B_R \).

The third step is relatively trivial. Its correctness depends on the following lemma.

Lemma 58. If \( \simeq_{\text{Id}_R} \subseteq \hat{\equiv}_{\text{Id}_R} \), then \( \simeq_R \subseteq \hat{\equiv}_R \).

The first and second steps of the construction are described in Section VI-D and Section VI-E.

D. Determining \( \text{Id}_R^B \)

First of all, we must determine what \( \text{Id}_R^B \) is. This problem asks when we can believe that \( X \hat{\equiv}_R \epsilon \) for \( X \subseteq C_G \). Be aware that Lemma [35] confirms that \( \text{Id}_R^B \subseteq \text{Id}_R^D \). The basic idea is to make use of the expansion conditions.

Definition 20. Let \( S \) be a set that makes \( R \subseteq S \subseteq \text{Id}_R^D \). We call \( S \) an \( \text{Id}_R^B \)-candidate if the following conditions are satisfied whenever \( X \in S \setminus R \):

1) If \( X \xrightarrow{R} \alpha \) and \( \alpha \not\in \text{Id}_R^B \), then \( \epsilon \xrightarrow{R} \beta \) for some \( \beta \) such that \( \text{dcmp}_R(\alpha) = \text{dcmp}_R(\beta) \).

2) If \( X \xrightarrow{R} \alpha \), then \( \epsilon \xrightarrow{R} \beta \) for some \( \beta \) such that \( \text{dcmp}_R(\alpha) = \text{dcmp}_R(\beta) \).

According to Definition 20:

1. \( R \) is an \( \text{Id}_R^B \)-candidate.

2. \( \text{Id}_R^B \)-candidates are closed under union.

\( \text{Id}_R^B \) is defined as the largest \( \text{Id}_R^B \)-candidate. One fast way of computing \( \text{Id}_R^B \) is described as procedure COMPUTINGID(\( R \)) in Fig. 1. It is easy to check the following properties.

Lemma 59.

1. \( R \subseteq \text{Id}_R^B \subseteq \text{Id}_R^D \).

2. If \( R \subseteq S \), then \( \text{Id}_R^B \subseteq \text{Id}_S^B \). If \( R \subseteq S \subseteq \text{Id}_R^D \), then \( \text{Id}_R^B = \text{Id}_S^B \). In particular, \( \text{Id}_{\text{Id}_R^D} = \text{Id}_R^B \).

According to \( B \)-reduction rules in Section VI-B, \( \alpha \hat{\equiv}_R \epsilon \) if and only \( \alpha \in (\text{Id}_R^B)^* \). Thus \( (\text{Id}_R^B)^* \) is the only class that the \( \hat{B}_R \)-norm of whose members are zero. The correctness of the construction of \( \text{Id}_R^B \) depends on the following lemma.

Lemma 60. Assume that \( \simeq_R \subseteq \hat{\equiv}_R \). Then \( \alpha \hat{\equiv}_R \epsilon \) implies \( \alpha \hat{\equiv}_R \epsilon \).

E. Determining Other Constituents of \( B_R \)

When \( \text{Id}_R^B \) is determined for every \( R \), we can construct the whole \( B \) via a greedy strategy. Since \( B_R \) is completely determined by \( B_{\text{Id}_R^B} \), we only need to construct those \( B_R \)'s in which \( R \) is \( B \)-admissible.

The algorithm in Fig. 2 constructs \( B_R \) and compute the \( \hat{B}_R \)-norms of \( [X]_R \)'s for every \( B \)-admissible \( R \) at the same time via the greedy strategy. When the program starts an iteration of repeat-block at line 4, it attempts to find all the blocks \( [X]_R \)'s such that \( \|X\|_R = m \). In the algorithm \( d_R([X]_R) \) is used to indicate \( [X]_B \). We also write \( d_R(\alpha) \) for \( \|\alpha\|_R \).

Precisely,

\[
d_R(\alpha) \overset{\text{def}}{=} \sum_{i=1}^{r} d_R([X_i]_R), \quad (2)
\]

if \( \text{dcmp}_R(\alpha) = [X_r]_R \cdots [X_{r-1}]_R \cdots [X_1]_R \). The algorithm maintains two sets \( U \) and \( V \). They form a partition of all the \( B \)-admissible blocks. (A block \( [X]_R \) is \( B \)-admissible if \( R \) is \( B \)-admissible.) During the execution of the algorithm, we can move a certain \( [X]_R \) from \( U \) to \( V \). At that time, we define the related information for \( [X]_R \) to determine whether \( [X]_R \) is a \( B_R \)-prime or a \( B_R \)-composite; compute \( \text{DcD}_R([X]_R) \) if it is a \( B_R \)-prime; compute \( \text{DcD}_R([X]_R) \) if it is a \( B_R \)-composite.

In the following, we say that a block \( [X]_R \) is treated if \( [X]_R \in V \). When \( [X]_R \) is selected during the execution of the algorithm, it is called undeterring. Every time \( [X]_R \) under treating, we confirm the following fact:

- If \( [X]_R \xrightarrow{R} \alpha \) with \( d_R(\alpha) = m - 1 \), \( (\hat{B}_R \)-decreasing \)
- or if \( [X]_R \xrightarrow{R} \alpha \) with \( d_R(\alpha) = m \), (possibly \( \hat{B}_R \)-preserving \)

then all the blocks in the \( B_R \)-decomposition of \( \alpha \) are treated, thus the related information for \( \alpha \) is already known. The first case is guaranteed by the non-decrease of \( m \). The second case is by the aid of the order of \( <_R \). These are two cases which
**Computing ID(R):**

1) $\text{Id}^R_R := \text{Id}^R_R$

2) while there exists $X \in \text{Id}^R_R - R$ such that one of the followings is violated:
   - If $X \xrightarrow{\alpha} \beta$ and $\alpha \notin \text{Id}^R_R$, then $\epsilon \xrightarrow{\alpha} \beta$ for some $\beta$ such that $\alpha = R \beta$.
   - If $X \xrightarrow{\alpha} \beta$, then $\epsilon \xrightarrow{\alpha} \beta$ for some $\beta$ such that $\alpha = R \beta$.
   do remove $Y$ from $\text{Id}^R_R$ for every $Y \Rightarrow R \zeta \Rightarrow R \epsilon$ and $X$ appears in $\zeta$.
end while

**Initializing:**

1) for every $R$-admissible $R$
   $\text{Pr}_R^R := \emptyset$ ; $\text{Cm}_R^R := \emptyset$.
   for every $[X]_R \in C_R$
   $\text{De}_R([X]_R) := \perp$ ; $\text{Rd}_R([X]_R) := \perp$.
   \[d_R([X]_R) := \perp.\]
end for

2) $U := \{[X]_R | R = \text{Id}^R_R$ and $[X]_R \in C_R\}$;
   $V := \emptyset$;
   $T := \emptyset$.

**Expanding $R(X, [Y_k]_{R_k} \ldots [Y_1]_{R_1})$:**

1) if $X \nRightarrow R Y_k \ldots Y_1$ then
   return false.
endif

2) if the followings conditions are met:
   - Whenever $[X]_R \Rightarrow R \alpha$, then
     a) if $d_R(\alpha) = m - 1$, then $[Y_k]_{R_k} \Rightarrow R \zeta$ such that $\alpha = R \zeta, Y_k, \ldots, Y_1$.
     b) else, either $\zeta = \tau$ and $\alpha = R \zeta, Y_k, \ldots, Y_1$, or $[Y_k]_{R_k} \Rightarrow R \zeta$ such that $\alpha = R \zeta, Y_k, \ldots, Y_1$.
   - Either $[X]_R \Rightarrow R \alpha$ for some $\alpha$ such that $\alpha = R Y_k, \ldots, Y_1$; or whenever $[Y_k]_{R_k} \Rightarrow R \zeta$.
     a) if $d_R(\gamma, Y_{k-1}, \ldots, Y_1) = m - 1$, then $[X]_R \Rightarrow R \alpha$ for some $\alpha$ such that $\alpha = R \gamma, Y_{k-1}, \ldots, Y_1$.
     b) else, $[X]_R \Rightarrow R \alpha$ for some $\alpha$ such that $\alpha = R Y_{k-1}, \ldots, Y_1$.
endif
then
return true.
else
return false.
endif

**Computing $R^*_R([X]_R)$:**

1) $T := \{W \mid W.X \nRightarrow R X\}$ ; $R^*_R([X]_R) := T$.

2) while there exists $Y \in R^*_R([X]_R)$ such that one of the followings are violated:
   - If $Y \Rightarrow R \zeta$ and $\zeta \notin T$, then $[X]_R \Rightarrow R \beta$ for some $\beta$ such that $\zeta = R \beta$.
   - If $Y \Rightarrow R \zeta$, then $[X]_R \Rightarrow R \beta$ for some $\beta$ such that $\zeta = R \beta$.
   do remove $Y$ from $R^*_R([X]_R)$ for every $Y \Rightarrow R \zeta \Rightarrow R \epsilon$ and $X$ appears in $\zeta$.
end while

**Constructing New Base:**

1) for every $R$
   **Computing ID(R).**
end for

2) Initializing.

3) $m := 1$.

4) repeat

5) while there exists $[X]_R \in U$ such that $[X]_R \Rightarrow R \gamma$ and $d_R(\gamma) = m - 1$,
   do
   select one of such $[X]_R$ which is $R$-minimum.
   $d_R([X]_R) := m$.
   if there exists $\delta$ such that $\text{Expand}_R(X, dcmp^B_R(\delta))$, then
   put $[X]_R$ into $\text{Cm}_R^R$.
   $\text{De}_R([X]_R) := \text{dcmp}^B_R(\delta)$.
   else
   put $[X]_R$ into $\text{Pr}_R^R$.
   **Computing $R^*_R([X]_R)$**.
end if
move $[X]_R$ from $U$ to $V$.
end while

6) while there exists $[X]_R \in U$ such that $[X]_R \Rightarrow R \gamma$ and $d_R(\gamma) = m$,
   do
   if $\text{Expand}_R(X, dcmp^B_R(\gamma))$, then
   put $[X]_R$ into $\text{Cm}_R^R$.
   $d_R([X]_R) := m$.
   $\text{De}_R([X]_R) := \text{dcmp}^B_R(\gamma)$.
   move $[X]_R$ from $U$ to $V$.
else
move $[X]_R$ from $U$ to $T$.
end if
end while

7) put every block in $T$ into $U$.

8) $m := m + 1$.

until $U = \emptyset$

9) for every non-$B$-admissible $R$
   $B'_R := B'_{\text{Id}^R_R}$.
end for

---

Figure 1. Constructing New Base: Part I

Figure 2. Constructing New Base: Part II

correspond to two different possibilities that the $B_R$-norm of $[X]_R$ can be declared as $m$.

1) Treating $[X]_R$: The First Possibility. : There exists a witness path of $[X]_R$ starting with a $\nRightarrow R$-decreasing transition. That is, $[X]_R \Rightarrow R \gamma$ for some $\gamma$ such that $\|\gamma\|_R = m - 1$. This possibility is treated via the while-block at line 5.

At the time we have known $\text{dcmp}^B_R(\gamma)$. The first problem is to decide whether $[X]_R$ is a prime or a composite. To this end, we try to guess a candidate for decomposition of $[X]_R$, say $[Y_k]_{R_k}[Y_{k-1}]_{R_{k-1}} \ldots [Y_1]_{R_1}$ with $R_1 = R$ and $R_{i+1} =
Let \( X \subseteq R \) and \( Y \) be the set \{W | W \subseteq R \} and let \( S \subseteq T \). We call \( S \) an \( R^B \) candidate if the following conditions are satisfied whenever \( Y \subseteq S \):

1. If \( Y \xrightarrow{\tau} \zeta \) and \( \zeta \not\in T^* \), then \( [X]_R \xrightarrow{\tau} \beta \) for some \( \beta \) such that \( \zeta, X \xrightarrow{\beta} \beta \).
2. If \( Y \xrightarrow{\tau} \zeta \), then \( [X]_R \xrightarrow{\tau} \beta \) for some \( \beta \) such that \( \zeta, X \xrightarrow{\beta} \beta \).

According to Definition 27:

1. \( \emptyset \) is an \( R^B \) candidate.
2. \( R^B \) candidates are closed under union.

\( R^B \) is defined as the largest \( R^B \) candidate. One fast way of computing \( R^B \) is described as procedure COMPUTING\( R^B \) in Fig. 1.

4) Basic Properties of the Construction: We point out the following important properties.

**Lemma 61.** \( R^B \) constructed above is \( B \)-admissible.

**Lemma 62.** \( d_R([X]_R) \) computed in our algorithm is equal to \(|X| \cdot \epsilon_R \cdot | \alpha | \). As an inference, \( d_R(\alpha) = | \alpha | \cdot \epsilon_R \).

**Lemma 63.** \( B \subseteq D \). If \( \tilde{B} \subseteq B \), then \( \tilde{B} \subseteq D \).

**F. The Correctness of the Refinement Operation**

Remember Lemma 27 and Lemma 63. The remain thing is to confirm the following fact.

**Theorem 4.** Suppose that \( \tilde{B} \subseteq D \), then \( \tilde{B} \subseteq B \). Namely, \( \alpha \xrightarrow{\beta} \beta \) implies \( \alpha \xrightarrow{B} \beta \) for every \( R \).

It is enough to prove Theorem 4 under the assumption that \( R \) is \( B \)-admissible. If \( \xrightarrow{\beta} \beta \) for every \( B \)-admissible \( R \)'s, then for non-\( B \)-admissible \( R \) we have \( \xrightarrow{\beta} \beta \subseteq \xi d_R(B) = \epsilon_R \). Thus in the following we assume \( R \) to be \( B \)-admissible.

The correctness of Theorem 4 relies on some important observations.

**Lemma 64.** Let \( R \) be \( B \)-admissible. Let \( \gamma = Y_i \ldots Y_1 \) and \( \delta = Z_l \ldots Z_1 \) such that \([Y_k]_R \subseteq [Y_{k-1}]_R \ldots [Y_1]_R \) and \([Z_j]_R \subseteq [Z_{j-1}]_R \ldots [Z_1]_R \) are two \( B \)-decompositions, in which \( R_1, S_1 = R \) and \( R_{i+1} = R^B(Y_i) \) for \( 1 \leq i < k \) and \( S_{j+1} = R^B(Z_j) \) for \( 1 \leq j < l \). If \( \gamma \) and \( \delta \) satisfy the expansion conditions for \( B \), then we have \( h = l, R_i = S_i \) and \([Y_k]_R \subseteq [Z_k]_R \) for \( 1 \leq i < k \).

**Lemma 64** confirms that, when \([X]_R \) is being treated, at most one decomposition candidate \([Y_k]_R \subseteq [Y_{k-1}]_R \ldots [Y_1]_R \) can make \( X \) \( B \)-prime. Decreasing bisimulation property is crucial to validate Lemma 64. This is why \( R \) must be constructed as a decreasing bisimulation with \( R \)-expansion of \( P \) (Section VI-C), rather than simply defined as \( R \)-expansion of \( P \).

**Construction**

**II.** Let \( S \) be an arbitrary \( B \)-admissible set. Suppose that \( \beta \) is \( B \)-admissible and \( d_R([X]_R) = [W_u]_S \ldots [W_1]_S \). Then \( W_u \ldots W_1 \) is \( B \)-applicable, and \( d_R([X]_R) = [W_u]_S \ldots [W_1]_S \). That is, \( \alpha \not\in S \), \( W_u \ldots W_1 \). When \( V \) contains all blocks, we can get Theorem 4 by induction. Making use of II, we can establish the following.

**Lemma 65.** Suppose \( S \) is \( B \)-admissible and \( d_R([X]_R) = [W_u]_S \ldots [W_1]_S \), then \( W_u \ldots W_1 \) is \( B \)-applicable.
1) If $\|W_u \ldots W_1\|_{B_R^S} < m$, then $[W]_S \in V$ and $W \equiv_S W_u \ldots W_1$.

2) If $\|W_u \ldots W_1\|_{B_R^S} = m$ and $W <_S X$, then $[W]_S \in V$ and $W \equiv_S W_u \ldots W_1$.

With Lemma 65 and IH, we can show the following.

**Lemma 66.** Suppose $[X]_R$ be a $\hat{B}_R$-composite, and assume that $\text{Dec}_R^B([X]_R) = [Z_l]_R \ldots [Z_1]_R$. Then $Z_l \ldots Z_1$ is $B_R, V$-applicable.

With Lemma 65 and Lemma 64 we can prove the following key property.

**Proposition 67.** Suppose $[X]_R$ be a $\hat{B}_R$-composite, and assume that $\text{Dec}_R^B([X]_R) = [Z_l]_R \ldots [Z_1]_R$. Then $X \equiv_R Z_l \ldots Z_1$.

**G. The Time Complexity**

The running time of our algorithm is exponentially bounded, according to its description, together with Lemma 34, Lemma 48 and Proposition 55.

**VII. EXAMPLES**

To be announced.

**REFERENCES**


APPENDIX A
PROOFS IN SECTION [III]

A. Proof of Proposition [12]

Suppose \( R_1 \subseteq R_2 \). We show that \( \simeq \overset{\text{def}}{=} (\simeq_{R_1} \cup \simeq_{R_2})^* \) is an \( R_2 \)-bisimulation. Readers should keep in mind that \( \simeq_{R_1} \subseteq \simeq \) and \( \simeq_{R_2} \subseteq \simeq \).

Suppose there are \( \alpha \) and \( \beta \) such that \( \alpha \simeq_{R_1} \beta \). According to Definition [6] we have:
1) if \( \alpha \rightarrow \epsilon \), then \( \beta \rightarrow \epsilon \);
2) if \( \alpha \frac{\nu}{\nu} \alpha' \), then \( \beta R_1 \frac{\nu}{\nu} R_1 \frac{\nu}{\nu} R_1 \beta' \) for some \( \beta' \) such that \( \alpha' \simeq_{R_1} \beta' \);
3) if \( \alpha \rightarrow \alpha' \), then \( \beta R_1 \simeq_{R_1} R_1 \frac{\nu}{\nu} \alpha \), for some \( \beta' \) such that \( \alpha' \simeq_{R_1} \beta' \).

Now we need to show that
1) if \( \alpha \rightarrow \epsilon \), then \( \beta \rightarrow \epsilon \);
2) if \( \alpha \frac{\nu}{\nu} \alpha' \), then \( \beta R_2 \frac{\nu}{\nu} R_2 \frac{\nu}{\nu} R_2 \beta' \) for some \( \beta' \) such that \( \alpha' \simeq_{R_2} \beta' \);
3) if \( \alpha \rightarrow \alpha' \), then \( \beta R_2 \simeq_{R_2} R_2 \frac{\nu}{\nu} \alpha \), for some \( \beta' \) such that \( \alpha' \simeq_{R_2} \beta' \).

Condition 1 is trivial. Other two conditions have the same structures. As usual we choose to prove Condition 2. Assume \( \alpha \frac{\nu}{\nu} \alpha' \). We have
\( \beta S \overset{\nu}{\rightarrow} \beta' \) for some \( \beta' \) such that \( \alpha' \simeq S \beta' \).

Now there are two cases:

- \( \beta S = \epsilon \). In this case, we have \( \beta \in S^* \) thus \( \beta \simeq_{R_1} \epsilon \) and \( \beta_{S} \simeq_{R} \epsilon \). Consider any \( \beta'' \) such that \( \beta_{S} \overset{\nu}{\rightarrow} \beta'' \). We have the following properties.
  - \( \beta'' \rightarrow \epsilon \), because \( \beta'' \simeq_{R} \epsilon \).
  - There exists \( X \in S \) such that \( X \overset{\nu}{\rightarrow} \gamma \) and \( \gamma_{S} = \beta'' \).

Now we have \( \epsilon \overset{\nu}{\rightarrow} X \overset{\nu}{\rightarrow} \gamma \rightarrow \epsilon \) and therefore by Computation Lemma, \( \gamma \overset{\nu}{\rightarrow} R \epsilon \) and therefore \( S = \epsilon \). Therefore \( \beta'' = \gamma_{S} = \epsilon \). This fact is important in the sense that whenever \( \beta \overset{\nu}{\rightarrow} \beta'' \), \( \beta'' \) must be \( \epsilon \). Now according to Lemma [6] there exists \( X \in S \) such that \( X \overset{\nu}{\rightarrow} \hat{\beta} \) for some \( \beta' \) and \( \beta_{S} = \beta' \). Knowing \( \simeq_{R} \subseteq \simeq_{S} \), we have \( \hat{\beta} = \beta_{S} \). Now that \( \hat{\beta} \overset{\nu}{\rightarrow} \beta'' \) because \( \beta \in S \) and \( \beta \simeq_{R_1} \epsilon \) and \( \beta \simeq_{R_2} \epsilon \).

- \( \beta \neq \epsilon \). In this case we can show by applying Lemma [6] that \( \beta \overset{\nu}{\rightarrow} \beta'' \) for some \( \beta' \) such that \( \alpha' \simeq S \beta' \). By Definition [5] we have \( \beta_{R} \overset{\nu}{\rightarrow} \beta_{R} \overset{\nu}{\rightarrow} \beta' \). Knowing that \( \beta_{R} \simeq \beta \) and \( \beta_{R} \subseteq \simeq_{S} \), we have \( \beta_{R} \simeq \beta \) and thus \( \alpha' \simeq \beta' \).

B. Proof of Proposition [14]

Because \( R \subseteq \text{Id}_{R} \), by Proposition [12] we know that \( \simeq_{R} \subseteq \simeq_{R} \). Thus it suffices to show \( \simeq_{R} \subseteq \simeq_{R} \). Let \( S = \text{Id}_{R} \), it suffices to show that \( S \) is an \( R \)-bisimulation.

Suppose \( \alpha \simeq_{S} \beta \). According to Definition [6] we have:
1) if \( \alpha \rightarrow \epsilon \), then \( \beta \rightarrow \epsilon \);
2) if \( \alpha \frac{\nu}{\nu} \alpha' \), then \( \beta_{S} \overset{\nu}{\rightarrow} \beta_{S} \frac{\nu}{\nu} \beta' \) for some \( \beta' \) such that \( \alpha' \simeq_{S} \beta' \);
3) if \( \alpha \rightarrow \alpha' \), then \( \beta_{S} \simeq_{S} R_{S} \frac{\nu}{\nu} \alpha \), for some \( \beta' \) such that \( \alpha' \simeq_{S} \beta' \).

Now assume that \( \alpha \leq \beta \). Then we have \( \alpha \simeq_{R_{1}} \alpha \simeq_{R_{2}} \beta \). If we can use transitivity to get the result of the proposition.

C. Proof of Theorem [1] and Theorem [2]

Since Theorem [1] is a special case of Theorem [2] we only prove Theorem [2]. The proof is divided into the following Lemma [68] and Lemma [69].

Lemma 68. Let \( S = \text{Rd}_{R}(\gamma) \), then \( \alpha \simeq_{S} \beta \) implies \( \alpha_{\gamma} \overset{R_{\gamma}}{\longrightarrow} \beta_{\gamma} \).

Proof. We can assume that \( \gamma \notin R^{*} \). If not, we will have \( S = \text{Id}_{R} \) and thus according to Proposition [12] \( \simeq_{S} = \simeq_{R} \). The result of this lemma holds accordingly.

Let \( I \) be identical relation on \( C^{*} \). That is, \( I = \{ (\zeta, \zeta) \mid \zeta \in C^{*} \} \). Let
\( \approx_{S} \overset{\text{def}}{=} \{ (\alpha_{\gamma}, \beta_{\gamma}) \mid \alpha \simeq_{S} \beta \} \).
Define the relation
\[ \equiv \overset{\text{def}}{=} (\simeq \cup \simeq_R)^* \]

Clearly \( \simeq \) is an equivalence relation. We show that \( \equiv \) is an \( R \)-bismulation. We call attention to readers that \( \simeq \subseteq \equiv \), and \( \equiv \circ \equiv \subseteq \equiv \).

Consider an arbitrary pair \((\zeta, \eta)\) such that \( \zeta \simeq \eta \). We prove
- if \( \zeta \overset{\ell}{\rightarrow} \), then \( \eta \overset{\ell}{\rightarrow} \) (trivially relabeling)
- if \( \zeta \overset{\ell}{\rightarrow} \zeta' \), then \( \eta_R \overset{\ell}{\rightarrow} \eta'_R \) for some \( \eta' \) such that \( \zeta' \simeq \eta' \);
- if \( \zeta \overset{\ell}{\rightarrow} \zeta' \), then \( \eta_R \overset{\ell}{\rightarrow} \eta'_R \) for some \( \eta' \) such that \( \zeta' \simeq \eta' \).

If \((\zeta, \eta) \in \equiv \). Now we must have \( \zeta = \alpha \gamma \) and \( \eta = \beta \gamma \) such that \( \alpha \simeq \beta \). There are two cases:

1. \( \alpha \neq \epsilon \). In this case \( \zeta \overset{\ell}{\rightarrow} \zeta' \) is induced by \( \alpha \gamma \overset{\ell}{\rightarrow} \alpha' \gamma \).
   - If \( \ell = \tau \) and \( \alpha \gamma \neq \alpha' \gamma \). In this case, we must have \( \alpha' \notin \equiv \alpha \). According to the fact \( \alpha \simeq \beta \) and Definition 6, \( \beta_S \overset{\ell}{\rightarrow} \beta' \) for some \( \beta' \) such that \( \alpha' \simeq \beta' \). Consider the above path from \( \beta' \) to \( \beta \). This path can be written as follows:
   \[ \beta_S = \beta_0 \overset{\zeta_S}{\rightarrow} \beta_1 \overset{\zeta_S}{\rightarrow} \ldots \overset{\zeta_S}{\rightarrow} \beta_k \overset{\zeta_S}{\rightarrow} \beta \]
   We have two possibilities:
   - \( \beta_i \neq \epsilon \) for every \( 0 \leq i \leq k \). According to Lemma 6, we have \( \beta \overset{\equiv}{\rightarrow} \beta' \simeq \beta \) for some \( \beta' \). Actually we have \( \beta \overset{\equiv}{\rightarrow} \beta' \overset{\tau}{\rightarrow} \beta' \) with \( \alpha' \simeq \beta' \). Therefore we have \( \alpha' \overset{\tau}{\rightarrow} \beta' \). Furthermore, because \( \equiv \subseteq \beta \subseteq \beta \), we have \( \beta' \simeq \beta \).
   - \( \beta_i = \epsilon \) for some \( 0 \leq i \leq k \). Choose the largest \( i \).

2. \( \alpha = \epsilon \). In this case \( \zeta \overset{\ell}{\rightarrow} \zeta' \) is induced by \( \gamma \overset{\ell}{\rightarrow} \gamma' \).
   Now we have \( \beta \equiv \gamma \). Thus \( \beta' \overset{\tau}{\rightarrow} \gamma' \) with \( \alpha' = \gamma \) and \( \gamma' \).

In all, we have \( \beta \gamma \overset{\equiv}{\rightarrow} \gamma_R \overset{\equiv}{\rightarrow} \gamma' \) such that \( \alpha \gamma \simeq \gamma' \) and \( \alpha' \gamma \simeq \gamma' \). We must have \( \gamma' \neq \gamma' \) because \( \gamma' \neq \gamma' \neq \alpha' \gamma \simeq \gamma' \).

- \( \ell \neq \tau \). This case can be proved in the same way as the case \( \ell = \tau \) and \( \alpha \neq \alpha' \).

2) \( \alpha = \epsilon \). In this case \( \zeta \overset{\ell}{\rightarrow} \zeta' \) is induced by \( \gamma \overset{\ell}{\rightarrow} \gamma' \).
   Now we have \( \beta \equiv \gamma \). Thus \( \beta' \overset{\tau}{\rightarrow} \gamma' \) with \( \alpha' = \gamma \) and \( \gamma' \).

Lemma 69. Suppose \( S = Rd_R(\gamma) \), then \( \alpha \gamma \simeq \beta \gamma \) implies \( \alpha \simeq \beta \).

Proof. Define the set
\[ \overset{\equiv}{=} \overset{\text{def}}{=} \{ (\alpha, \beta) \mid \alpha \gamma \simeq \beta \gamma \} \]
As before we can assume \( \gamma \notin R^* \). Otherwise the conclusion of the lemma is relatively trivial.

We show that \( \simeq \) is an \( S \)-bismulation. It is easy to see that \( \simeq \) is an equivalence relation indeed.

Now we check the properties of Definition 6

1. We show \( =_S \subseteq \simeq \). Let \( \alpha = S \beta \). According to Definition 3 there exist \( \alpha', \beta' \in S^* = (Rd_R(\gamma))^* \) and a process \( \zeta \) such that \( \alpha = \zeta \alpha' \) and \( \beta = \zeta \beta' \). Now \( \alpha' \overset{\zeta}{\rightarrow} \beta' \) with \( \alpha' \simeq \beta' \).

2. If \( \alpha \simeq \beta \) and \( \alpha \equiv \beta \), we show \( \beta \equiv \beta \). According to the definition of \( \equiv \), \( \alpha \gamma \simeq \beta \gamma \). Now \( \alpha \gamma \simeq \beta \gamma \). We must be matched by \( \beta \gamma R \). Let us suppose that the matching is \( \beta \gamma R \cap \beta' \gamma R \). Which is induced by \( \beta \equiv \beta' \).

3. If \( \alpha \simeq \beta \) and \( \alpha \overset{\ell}{\rightarrow} \alpha' \), then we show that \( \beta_S \overset{\equiv}{\rightarrow} \beta_S \) for some \( \beta' \) such that \( \alpha' \simeq \beta' \).

According to the definition of \( \equiv \), we have \( \alpha \gamma \simeq \beta \gamma \). Moreover, \( \alpha \overset{\ell}{\rightarrow} \alpha' \) is equivalent to \( \alpha \overset{\ell}{\rightarrow} \gamma \alpha' \gamma \). There are two cases:

- \( \alpha \notin (Rd_R(\gamma))^* \). In this case we also have \( \beta \notin (Rd_R(\gamma))^* \). Then the action \( \alpha \gamma \overset{\ell}{\rightarrow} \alpha' \gamma \) must be matched by \( \beta \gamma R \overset{\equiv}{\rightarrow} \beta' \gamma R \) for some \( \beta' \) with \( \alpha' \simeq \beta' \gamma \). This is equal to \( \beta \overset{\equiv}{\rightarrow} \beta' \). Therefore \( \alpha \overset{\ell}{\rightarrow} \alpha' \gamma \).

- \( \alpha \in (Rd_R(\gamma))^* \). In this case we have \( \alpha \gamma \overset{\equiv}{\rightarrow} \gamma_R \). Hence \( \beta \overset{\equiv}{\rightarrow} \beta \gamma R \). In other words, \( \alpha, \beta \in S^* \).

Now we have \( \beta \overset{\equiv}{\rightarrow} \beta \gamma R \), or equivalently \( \beta \overset{\equiv}{\rightarrow} \beta \).

Now remember \( \epsilon = S \alpha \) and \( \alpha \overset{\ell}{\rightarrow} \alpha' \). Combine these transitions we have \( \beta \overset{\equiv}{\rightarrow} \epsilon = S \alpha \overset{\ell}{\rightarrow} \alpha' \), and thus \( \beta_S \overset{\equiv}{\rightarrow} \beta_S \). We are done.
4) If \( \alpha \simeq \beta \) and \( \alpha \xrightarrow{\alpha'} \beta' \), then we show \( \beta \xrightarrow{\beta'} \) for some \( \beta' \) such that \( \alpha' \simeq \beta' \). This case can be treated in the same way as the previous one.

D. Proofs Concerning R-redundancy

Proof of Lemma 25: Suppose \( X \in R_d \). According to Lemma 22, we have \( X \simeq R_d \) if and only if \( X \in C_m \). This is equivalent to \( X \simeq R_d \). By Theorem 23, \( X \simeq R_d \) if and only if \( X \in R_d \). The proof of the other direction is from the fact that the above reasoning steps are reversible.

Proof of Lemma 52: Consider the following sequence of transitions:

\[
\alpha X \xrightarrow{\alpha_1} \alpha X \xrightarrow{\alpha_2} \cdots \xrightarrow{\alpha_k} \alpha X \xrightarrow{\alpha_\ell} \alpha' X
\]

with \( \alpha, X \not\simeq \alpha' X \) for \( 1 \leq i \leq k \) and \( \alpha' X \simeq X \). This sequence can be matched by \( \alpha' Y \) via

\[
\beta Y \xrightarrow{\beta_1} \beta_1 Y \xrightarrow{\beta_2} \cdots \xrightarrow{\beta_k} \beta_k Y \xrightarrow{\beta_\ell} \beta' Y,
\]

such that

- \( \beta_1 Y \not\simeq \beta_1 Y \) for \( 1 \leq i \leq k \) and \( \beta Y \simeq \beta Y \);
- \( \alpha_1 X \simeq \beta Y \) for \( 1 \leq i \leq k \) and \( \alpha_1 X \simeq \beta Y \).

Accordingly \( \alpha \simeq \alpha \beta \), which implies \( \alpha \simeq \alpha \beta Y \). According to Theorem 24, we can reduce the problem to the case where \( \beta Y \) is the only possible match for \( \alpha_1 X \). Since \( \beta Y \) is \( \beta_1 Y \), we can use induction to show that \( X \simeq R_d \).

Proof of Lemma 53: It suffices to prove \( R_d \subseteq R_d \) for \( B \)-admissible \( (B \)-admissible \( R \)-prime). Let \( [P_i] \subseteq \{ P_i \} \) be \( B \)-prime. According to the rules of \( B \)-reduction defined in Section V.B, we can reduce \( X \in R_d \) if and only if \( X \simeq X \). Thus \( X \simeq X \) whenever \( X \in R_d \), since \( \beta \subseteq \beta \).

Proof of Lemma 54: First we point out that \( C_m \beta \) is \( \beta \)-admissible whenever \( \beta \subseteq \beta \). Then \( \beta \subseteq \beta \) is \( \beta \)-admissible. Hence \( \alpha \beta \), which implies \( \alpha_1 X \simeq \beta Y \), and \( \alpha_1 X \simeq \beta Y \).

Proof of Lemma 55: By induction on \( r \). Suppose \( B \subseteq D \). There is some \( \beta \) and some \( B \)-admissible \( (B \)-admissible \( R \)-prime). Let \( [P_i] \subseteq \{ P_i \} \) be \( B \)-prime. According to the rules defined in Section V.B, we can reduce \( X \in R_d \) if and only if \( X \simeq X \). Thus \( X \simeq X \) whenever \( X \in R_d \), since \( \beta \subseteq \beta \).

Proof of Lemma 56: We need to show that \( \beta \subseteq \beta \). The proof is accomplished by using induction hypothesis.

APPENDIX B

PROOFS IN SECTION VI

A. Proofs Concerning Relationships between Bases

Proof of Lemma 57: If \( X \in \text{Id}_R \), then \( X \simeq R \). Since \( \beta \subseteq \beta \), we have \( X \simeq R \), hence \( X \in \text{Id}_R \).

Proof of Lemma 58: Clearly \( R \subseteq \text{Id}_R \). According to Proposition 22, \( \beta \subseteq \text{Id}_R \). On the other hand, \( \beta \subseteq \text{Id}_R \). Therefore, \( \beta \subseteq \text{Id}_R \).

Proofs of Lemma 59: It is a routine work to check that \( \beta \) is an \( \beta \)-candidate.

Proofs of Lemma 60: We need to show that \( \beta \subseteq \text{Id}_R \). The proof is accomplished by using induction hypothesis.

As before we let \( T \) be \( \{ W \mid W \subseteq X \} \). First we confirm \( \beta \subseteq \text{Id}_R \subseteq T \). By induction \( T \) is a \( \beta \)-admissible
set thus it is also $B$-admissible, hence $\text{Id}^B_R = T$. Because $\text{Rd}^B_R([X]_R) \subseteq T$, we have $\text{Id}^B_{\text{Rd}^B_R([X]_R)} \subseteq \text{Id}^B_R = T$.

Now we check the conditions of $\text{Rd}^B_R([X]_R)$-candidate. Let $Y \in \text{Id}^B_{\text{Rd}^B_R([X]_R)}$. If $Y \in \text{Rd}^B_R([X]_R)$, then nothing need to do. Now we suppose that $Y \not\in \text{Rd}^B_R([X]_R)$.

1) If $Y \xleftarrow{\tau} \zeta$ and $\zeta \not\in T^*$. In this case, because $\text{Id}^B_{\text{Rd}^B_R([X]_R)} \subseteq T$, we have $\zeta \not\in (\text{Id}^B_{\text{Rd}^B_R([X]_R)})^*$. Thus $\text{Rd}^B_R([Y]_R) \xleftarrow{\tau} \hat{\eta}$ for some $\hat{\eta}$. Now according to the definition of $\text{Id}^B_{\text{Rd}^B_R([X]_R)}$, we have $\epsilon \xrightarrow{\tau} \hat{\eta}$ for some $\hat{\eta}$ such that $\text{dcmp}^R_{\text{Rd}^B_R([X]_R)}(\hat{\eta}) = \epsilon$. In other words, there is $Z \in \text{Rd}^B_R([X]_R)$ and $Z \xrightarrow{\tau} \eta$ for some $\eta$ such that $\text{dcmp}^R_{\text{Rd}^B_R([X]_R)}(\eta) = \text{dcmp}^R_{\text{Rd}^B_R([X]_R)}(\eta)$. This makes $\zeta \xleftarrow{\tau} \epsilon$. Now, we use the fact that $\text{Rd}^B_R([X]_R)$ itself is an $\text{Rd}^B_R([X]_R)$-candidate. Thus it satisfies the relevant conditions. That is, $[X]_{\text{Id}^B_R} \xleftarrow{\tau} \beta$ for some $\beta$ such that $\eta, X \xrightarrow{\epsilon} \beta$. And finally we have $\zeta \xrightarrow{\tau} \epsilon$.

2) If $Y \xrightarrow{T^*}. \zeta$. The proof is complete the same as the first case.

Proof of Lemma 62

According to the algorithm, it is clear that $d_R([X]_R) \geq \|X\|_R$. The reason is elaborated as follows.

1) If $d_R([X]_R) = m$ via the fact $[X]_R \xrightarrow{\ell} \gamma$ and $d_R(\gamma) = m - 1$. Then there is a path from $X$ to $\gamma$ with length 1 and there is a path from $\gamma$ to $\epsilon$ with length $m - 1$. Thus totally we have a path from $X$ to $\epsilon$ with length $m$.

2) If $d_R([X]_R) = m$ via the fact $[X]_R \xrightarrow{\ell} \gamma$ and $d_R(\gamma) = m$ and $X \xrightarrow{\epsilon} \gamma$. Then there is a path from $X$ to $\gamma$ with length 0 and there is a path from $\gamma$ to $\epsilon$ with length $m$. Thus totally we have a path from $X$ to $\epsilon$ with length $m$.

Thus in both case we have $d_R([X]_R) \geq \|X\|_R$.

Now assume, for contradiction, that $d_R([X]_R) > \|X\|_R$. Then according to induction hypothesis, for every $[Y]_R \in V$, $d_R([Y]_R) = \|Y\|_R$. Now consider the time when $m$ is assigned to $d_R([X]_R)$. There are two possibilities:

1) $[X]_R \xrightarrow{\ell} \gamma$ and $d_R(\gamma) = m - 1$. In this case, by induction $d_R(\gamma) = \|\gamma\|_R = m - 1$. There can not be other transition of $[X]_R$ such as $[X]_R \xrightarrow{\ell} \zeta$ that

a) either $d_R(\zeta) < m - 1$,

b) or $d_R(\zeta) = m - 1$ and $\zeta \neq X$.

If so, $m - 1$ would be assigned to $d_R([X]_R)$ and the block $[X]_R$ should have already been put into $V$. This is a contradiction.

2) $[X]_R \xrightarrow{\ell} \gamma$, $d_R(\gamma) = m$ and $X \xrightarrow{\epsilon} \gamma$. In this case, by induction $d_R(\gamma) = \|\gamma\|_R = m$, thus we must have $\|X\|_R = \|\gamma\|_R = m$. This is a contradiction.

APPENDIX C

PROOF OF THEOREM

A. Proof of Lemma 64

There are several cases according to the values of $k$ and $l$:

- If $k, l > 1$. In this case we can assume that $\|\gamma\|_R = \|\delta\|_R$ which is already known, and we can tell whether a given action is $B$-decreasing. Suppose we have a decreasing transition of $\gamma \iff \ell \xrightarrow{R} \eta, Y_{k-1} \ldots Y_1$ which is induced by $[Y_k]_R \xrightarrow{R} \eta, Y_{k-1} \ldots Y_1$ which is induced by $[Y_l]_R \xrightarrow{R} \eta$. Moreover, we have $\eta, Y_{k-1} \ldots Y_1 = R \subseteq R, Z_{l-1} \ldots Z_1$. Now we have $\eta, Y_{k-1} \ldots Y_1 \subseteq R, Z_{l-1} \ldots Z_1$

- If $k = 1$ and $l > 1$. In this case, $[Y_1]_R$ must not be a prime. According to our algorithm, one of the candidates will be defined as $\text{Dc}^B_{\text{Id}^B_R([Y_1]_R)}$ if there exist some candidates which can pass the expansion testing.

- If $k = l = 1$. If $[Z_l]_R \subsetneq [Y_1]_R$, then this is the same as the above case. Otherwise we can change the role of $Y_1$ and $Z_1$.

B. Preparations for the Proof of Theorem

To make things clear, we introduce some new terminologies. Note that the program in Fig. 2 maintains a set $V$ of the blocks which have already been treated. During the execution of the algorithm, $V$ start from $\emptyset$ and get larger and larger. Intuitively these blocks in $V$ contain part of information of $B$. Formally, we can define the partial decomposition base $B_V = \{B_{R, V} \mid R \subseteq C_G\}$ in which $B_{R, V} = (\text{Id}^B_{\text{Pr}^B_R \cup V}, \text{Cm}^B_{\text{Pr}^B_R \cup V}, \text{Dc}^B_{\text{Pr}^B_R \cup V}, \text{Rd}^B_{\text{Pr}^B_R \cup V})$ where

- $\text{Pr}^B_R \cup V$.

- $\text{Cm}^B_{\text{Pr}^B_R \cup V}$.

- $\text{Dc}^B_{\text{Pr}^B_R \cup V}$.

- $\text{Rd}^B_{\text{Pr}^B_R \cup V}$.

$B_V$ is called partial in the sense that $\text{Pr}^B_R \cup \text{Cm}^B_{\text{Pr}^B_R \cup V} = C_R \cap V \subseteq C_R$. Comparatively, $\text{Pr}^B_R \cup \text{Cm}^B_{\text{Pr}^B_R \cup V} = C_R$.

At some time in the execution of the algorithm, we get a specific value of $V$, then $B_V$ is already known at that time. Now we can define $\text{dcmp}^B_{\text{Pr}^B_R \cup V}(\alpha)$ for any process $\alpha$. $\text{dcmp}^B_{\text{Pr}^B_R \cup V}(\alpha) = \text{dcmp}^B_{\text{Pr}^B_R \cup V}(\alpha)$ if the derivational of $\alpha \xrightarrow{\text{Pr}^B_R \cup V} \text{dcmp}^B_{\text{Pr}^B_R \cup V}(\alpha)$ (refer to Section V-B) only relies on the information provided in
$B_V$. Otherwise $dcmp^B_R(V)(\alpha)$ is undefined. In the following, a process $\alpha$ is called $B_R$-$\alpha$-applicable if $dcmp^B_R(V)(\alpha)$ is defined.

Now we prepare to confirm the important result: $\alpha \simeq_R \beta$ implies $\alpha \equiv_R \beta$.

First of all, we find that it is enough to prove the result under the assumption that $R$ is $B$-admissible. Note that if $\simeq_R R \subseteq \equiv_R$ for every $B$-admissible $R$, then for every $R$ we have $\simeq_R \subseteq \equiv_{Id^B_0} \subseteq \equiv_R$. Thus in the rest of this section we assume $R$ to be $B$-admissible.

With the help of Lemma 64 we are able to establish Theorem 4.

We take the following approach to prove Theorem 4. Remember that our algorithm maintains a set $V$, containing all the blocks which have been treated. We will suppose that $R$ is $B$-admissible. Let $[X]_R$ be a block which is about to be put into $V$. We try to prove that if $[X]_R$ is a $\bar{B}_R$-composite, and let $dcmp^B_R([X]) = [Z_1]_R, \ldots, [Z_l]_R$, then $X \equiv_R Z_1, \ldots, Z_l$.

Apparently, the proof must be done by induction. However, this is not an easy task. We will choose the following statements as our induction hypotheses:

I. Let $R$ be an arbitrary $B$-admissible set. Suppose that $\gamma$ is $B_R$-$\gamma$-applicable, and $dcmp^B_R(\gamma) = [W_u]_R, \ldots, [W_l]_R$.

Then $W_u \ldots W_l$ is $B_R$-$\gamma$-applicable, and $dcmp^B_R(\gamma) = \equiv_R(W_u \ldots W_l)$. That is, $\equiv_R W_u \ldots W_l$.

We remark that at the time the algorithm terminates when $V$ contains every blocks, the statement I implies Theorem 4.

The readers are suggested to imagine the following picture in mind. Although $R_1, \ldots, R_l$ are all $B$-admissible according to the definition of decomposition base, we cannot draw the conclusion that $R_1, \ldots, R_l$ are all $B$-admissible. It may indeed happen that $Z_i \in Id^B_{R_i}$, which means that $Z_i$ is $B_{R_i}$-redundant. However, since $R_1 = R$ is $B$-admissible, we do have $Z_1 \notin Id^B_{R_1}$. This fact will be used in the proof.

C. Proof of Lemma 65

This fact can be proved by induction on $d = \|W_u \ldots W_l\|_R$, and by studying a witness path of $\simeq_R$-norm for $W$. Then using hypothesis I and by inspecting the algorithm we can show that when $W \simeq_R W_u \ldots W_l$, $W$ should have already been put into $V$.

D. Proof of Lemma 66

We use $\delta$ to indicate $Z_1, \ldots, Z_l$. The lemma confirms that, when $[X]_R$ is about to be put into $V$, $dcmp^B_R(V)(Z_1, \ldots, Z_l)$ has already been defined. This fact is proved by induction, using the induction hypotheses. The way is to choose a process $\gamma$ such that $[X]_R \mapsto_R \gamma$ is on a $\equiv_R$-witness path of $X$. Now $\gamma$ is $B_R$-$\gamma$-applicable, thus we try to use induction hypotheses on $\gamma$. There are two cases:

• If there exist $\gamma$ such that $[X]_R \mapsto_R \gamma$ and $\|\gamma\|_R = m - 1$. In this case, the algorithm is running in the first while-loop in Fig. 2. Since $X \simeq_R \delta$, there is a matching of $[X]_R \mapsto_R \gamma$ from $\delta$, say $\delta \preceq_R \gamma$ for some $\gamma$ that $\gamma \simeq_R \zeta$. Because $\delta$ is itself a $\simeq_R$-prime-decomposition, we must have $\delta \leftrightarrow \cdot \mapsto_R \zeta$, which is induced by $[Z_1]_R \mapsto_R \eta$ for some $\eta$, and $\zeta = \eta, Z_1, \ldots, Z_l$. Suppose $dcmp^B_R(\eta) = [Y_s]_1, \ldots, [Y_l]_1 (s \geq 0)$, then $dcmp^B_R(\gamma)$ must be in the form $[Y_s]_1, \ldots, [Z_{t-1}]_1 R_1, \ldots, [Z_l]_1 R_1$. According to induction hypothesis I, $\gamma \equiv_R \zeta$, we have $dcmp^B_R(\gamma) = [Y_s]_1, \ldots, [Z_{t-1}]_1 R_1, \ldots, [Z_l]_1 R_1$. Thus $\|Y_s \ldots Z_{t-1} \ldots Z_l\|_R = m - 1$. Now since $dcmp^B_R(\zeta) = [Y_s]_1, \ldots, [Z_{t-1}]_1 R_1, \ldots, [Z_l]_1 R_1$, by Lemma 65 $\gamma \equiv_R Y_s, \ldots, Z_{t-1}, \ldots, Z_l$ and thus $\|\zeta\|_R = m - 1$. Since $\delta \leftrightarrow \cdot \mapsto_R \zeta$, we have $\|\delta\|_R \leq m$. In summary, we have:

- $\|\zeta\|_R = m - 1$. Since $\delta \leftrightarrow \cdot \mapsto_R \zeta$, we have $\|\delta\|_R \leq m$.
- $\|\delta\|_R > 0$.

There are two possibilities:

- Either $\|Z_i\|_R < m$ for every $1 \leq i \leq t$. In this case we have $\delta = Z_1, \ldots, Z_l$ is $B_R$-$\gamma$-applicable trivially.
- Or $t = 1$ and $\|Z_1\|_R = m$. In this case, we have $\|Z_1\|_R < \|X\|_R$ and $\|Z_1\|_R = \|X\|_R$. Thus $\|Z_1\|_R \in V$ and thus $\delta = Z_1$ is $B_{R_B}$-$\gamma$-applicable.

• If for every $\gamma$ such that $[X]_R \mapsto_R \gamma$, we do not have $\|\gamma\|_R < m$. In this case, the algorithm is running in the second while-loop in Fig. 2. We are able to find a $\gamma$ which is $B_R$-$\gamma$-applicable such that $[X]_R \mapsto_R \gamma$ and $X \equiv_R \gamma$, and $\|\gamma\|_R = \|X\|_R = m$. If it happens that we can find such a $\gamma$ satisfying $\hat{X} \simeq_R \gamma$, we can use the induction hypothesis I to confirm immediately that $\delta$ is $B_R$-$\gamma$-applicable and $\equiv_R \delta$ (hence $X \equiv_R \delta$). Thus in the following we will assume that $\gamma \neq_R X$. That is, $[X]_R \mapsto_R \gamma \neq_R X$. Since $X \simeq_R \delta$, there is a matching of $[X]_R \mapsto_R \gamma$ from $\delta$, say $\delta \preceq_R \gamma$. There exist such a $\gamma$ that $\gamma \simeq_R \zeta$. Because $\delta$ is itself a $\simeq_R$-prime-decomposition, we must have $\delta \leftrightarrow \cdot \mapsto_R \zeta \simeq_R \gamma$, which is induced by $[Z_1]_R \mapsto_R \eta$ for some $\gamma$, and $\zeta = \eta, Z_1, \ldots, Z_l$. Suppose $dcmp^B_R(\eta) = [Y_s]_1, \ldots, [Y_l]_1 (s \geq 0)$, then $dcmp^B_R(\zeta)$ must be in the form $[Y_s]_1, \ldots, [Z_{t-1}]_1 R_1, \ldots, [Z_l]_1 R_1$. According to induction hypothesis I, $\gamma \equiv_R \zeta$, we have $dcmp^B_R(\gamma) = [Y_s]_1, \ldots, [Z_{t-1}]_1 R_1, \ldots, [Z_l]_1 R_1$. Thus $\|Y_s \ldots Z_{t-1} \ldots Z_l\|_R = m - 1$. Now since $dcmp^B_R(\zeta) = [Y_s]_1, \ldots, [Z_{t-1}]_1 R_1, \ldots, [Z_l]_1 R_1$, by Lemma 65 $\gamma \equiv_R Y_s, \ldots, Z_{t-1}, \ldots, Z_l$ and thus $\|\zeta\|_R = m - 1$. Since $\delta \leftrightarrow \cdot \mapsto_R \zeta$, we have $\|\delta\|_R \leq m$. In summary, we have:

- $\|\zeta\|_R = m - 1$. Since $\delta \leftrightarrow \cdot \mapsto_R \zeta$, we have $\|\delta\|_R \leq m$.
- $\|\zeta\|_R > 0$.
Now we have two possibilities:

- **If** \( t \geq 2 \). Because \( \|Z_1\|_{S, \epsilon} > 0 \), thus \( \|Z_{t-1} \ldots Z_1\|_{S, \epsilon} > 0 \). Let \( S = \text{Rd}_R^B(Z_{t-1} \ldots Z_1) \)

  By induction \( R_t \subseteq S \). Then we have \( |Y_s \ldots Y_1| \|_{S, \epsilon} < m \). Since \( \eta \simeq_R \ Y_s \ldots Y_1 \), it is clear \( \eta \simeq_S \ Y_s \ldots Y_1 \), by Lemma 65 \( \eta \) is \( B_{S, \epsilon} \)-applicable, \( \eta \|_{S, \epsilon} = \|Y_s \ldots Y_1\|_{S, \epsilon} < m \). Now let us investigate \( Z_t \). If \( Z_t \in S \), \( Z_t \ldots Z_1 \) is trivially \( B_{R, \epsilon} \)-applicable. If \( Z_t \notin S \), we have got the fact that \( |Z_t|_S \rightarrow \eta \) and \( |\eta| \|_{S, \epsilon} < m \). This fact tells us that \( |Z_t|_S \) should have been treated before \( |X|_R \).

  In other words, \( |Z_t|_S \in V \), which means that \( \delta = Z_t \ldots Z_1 \) is \( B_{R, \epsilon} \)-applicable.

- **If** \( t = 1 \). In this case, we have the following facts: \( |Z_1|_R \rightarrow \eta \), \( \text{dcmp}_R^B(\eta) = |Y_s \ldots Y_1|_R \), and \( |Y_s \ldots Y_1| \|_{S, \epsilon} = m \). We can show \( |Y_1| \|_{S, \epsilon} > 0 \), using the same argument for proving \( |Z_1| \|_{S, \epsilon} > 0 \) before. Let us say \( \eta = \eta' \cdot W \). \( \eta' \) can be \( \epsilon \), and let \( S = \text{Rd}_R(W) \). Thus \( \text{dcmp}_R^B(W) = |Y_s \ldots Y_1| \) and \( \text{dcmp}_R^B(\eta') = |Y_s \ldots Y_{i+1}| \) for some \( 1 \leq i \leq s \) and \( \eta' \) is \( B_{R, S} \)-applicable by induction.

  1) If \( |Y_s \ldots Y_1| \|_{S, \epsilon} = m \), we can use Lemma 65 to prove:

   - \( W \) is \( B_{R, \epsilon} \)-applicable and \( B^R_{\epsilon} Y_s \ldots Y_1 \).
   - \( S \subseteq \text{Rd}_R^B(W) \).

    Since we know \( \eta' \simeq_R Y_s \ldots Y_{i+1} \), then \( \eta' \simeq_{\text{Rd}_R^B(W)} Y_s \ldots Y_{i+1} \). Because \( |Y_s \ldots Y_{i+1}| \|_{\text{Rd}_R^B(W)} < m \), we can use induction to prove \( \eta' \) is \( B_{\text{Rd}_R^B(W), \epsilon} \)-applicable and \( \eta' \simeq_{\text{Rd}_R^B(W)} Y_s \ldots Y_{i+1} \). In summary, \( \eta \) is \( B_{R, \epsilon} \)-applicable and \( \eta \simeq_R Y_s \ldots Y_1 \).

  2) If \( |Y_s \ldots Y_1| \|_{S, \epsilon} = m \). In this case, \( Y_s \ldots Y_{i+1} \in \text{Rd}_R^B(Y_s \ldots Y_1) \), this implies that \( Y_s \ldots Y_{i+1} \Rightarrow \epsilon \), and therefore

   \[
   \eta = \eta' \cdot W \simeq_R Y_s \ldots Y_1 \implies Y_s \ldots Y_1 \simeq_R W
   \]

   which implies \( \eta \Rightarrow_R W \). Now we have \( Z_1 \Rightarrow_R \eta \Rightarrow_R W \), thus \( [W]_R < X \}_{R, Z_1} \). On the other hand, by \( [Z_1]_R = \text{dcmp}^B_R([X]_R) \), we have \( [Z_1]_R < [X]_R \). Therefore \( W < R Z_1 < R X \). By Lemma 65 \( W \) is \( B_{R, \epsilon} \)-applicable and \( W \simeq_R Y_s \ldots Y_1 \). Now in the same way of case 1, we can prove \( \eta' \) is \( B_{\text{Rd}_R^B(W), \epsilon} \)-applicable and \( \eta' \simeq_{\text{Rd}_R^B(W)} Y_s \ldots Y_{i+1} \). In summary, \( \eta \) is \( B_{R, \epsilon} \)-applicable and \( \eta \simeq_R Y_s \ldots Y_1 \).

   Up to now, we have shown that \( [Z_1]_R < R [X]_R \) and \( [Z_1]_R \rightarrow_R \eta \) such that \( X \simeq_R \gamma \simeq_R \eta \) with \( |\eta|_{S, \epsilon} = m \). This fact means that \( [Z_1]_R \) should be chosen to test the expansion condition in the while-

E. **Proof of Proposition 67**

  We use \( \delta \) to indicate \( Z_t \ldots Z_1 \). By Lemma 66 \( \delta \) is \( B_{R, \epsilon} \)-applicable. In other words, \( \text{dcmp}^B_{R, \epsilon}(\delta) \) is known.

  The proof goes by directly exploring the expansion conditions. Only to remember the following fact:

  1) If \( \alpha \simeq_R \beta \), then \( \alpha \simeq_R \beta \).
  2) If \( \alpha \simeq_R \beta \), and \( \alpha, \beta \) are \( B_{R, \epsilon} \)-applicable, then \( \alpha \simeq_R \beta \).

  By studying the the expansion conditions, we can confirm that \( \text{dcmp}^B_{R, \epsilon}(\delta) \) can successfully pass this testing. Now it is important to take notice of Lemma 64. It ensures that at most one decomposition candidate can pass the testing. Thus we can confirm that \( \text{Dec}^B_{R}(X_{R}) = \text{dcmp}^B_{R, \epsilon}(\delta) \), which implies \( X \simeq_R Z_t \ldots Z_1 \).