A Polynomial Time Algorithm for Deciding Branching Bisimilarity on Totally Normed BPA

Chaodong He

BASICS, Department of Computer Science, Shanghai Jiao Tong University

Abstract. Strong bisimilarity on normed BPA is polynomial-time decidable, while weak bisimilarity on totally normed BPA is NP-hard. It is natural to ask where the computational complexity of branching bisimilarity on totally normed BPA lies. This paper confirms that this problem is polynomial-time decidable. To our knowledge, in the presence of silent transitions, this is the first bisimilarity checking algorithm on infinite state systems which runs in polynomial time. This result spots an instance in which branching bisimilarity and weak bisimilarity are both decidable but lie in different complexity classes (unless NP = P), which is not known before.

The algorithm takes the partition refinement approach and the final implementation can be thought of as a generalization of the previous algorithm of Czerwiński and Lasota [10,7]. However, unexpectedly, the correctness of the algorithm cannot be directly generalized from previous works, and the correctness proof turns out to be subtle. The proof depends on the existence of a carefully defined refinement operation fitted for our algorithm and the proposal of elaborately developed techniques, which are quite different from previous works.

1 Introduction

Basic process algebra (BPA) [2] 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 [3,4] proved a surprising result 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 [15]. After this remarkable discovery, decidability and complexity issues of bisimilarity checking on infinite state systems have been intensively investigated. See [19,6,28,29,20] for a number of surveys.

As regards to the strong bisimilarity checking on normed BPA, Hüttel and Stirling [17] 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 [18] showed that the problem is in Σ_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, with the

complexity $\mathcal{O}(N^{13})$. The running time was later improved [21,10]. All these algorithms take the approach of partition refinement, relying on the unique decomposition property and some efficient way of equality checking on compressed long strings. It deserves special mention that Czerwiński and Lasota [10] create a different refinement scheme. This refinement scheme was previously used in developing an polynomial-time algorithm of checking strong bisimilarity on normed basic parallel processes (normed BPP) [14]. In this way Czerwiński and Lasota improve the running time to $\mathcal{O}(N^5)$. Hitherto, the best algorithm was reported in [7], whose running time is $\mathcal{O}(N^4\text{polylog}(N))$.

In the presence of silent actions the picture is less clear. Even the decidability for weak bisimilarity is still open. A remarkable discovery is made by Fu [11] recently that branching bisimilarity [31], a standard refined alternative of weak bisimilarity, is decidable on normed BPA. Very recently, Czerwiński and Jančar confirm this problem to be in NEXPTIME [9]. The current best lowerbound for weak bisimilarity is the EXPTIME-hardness established by Mayr [24], whose proof can be slightly modified to show the EXPTIME-hardness for branching bisimilarity as well.

In retrospect one cannot help thinking that more attention should have been paid to the branching bisimilarity. Going back to the original motivation to equivalence checking, one would agree that a specification *spec* normally contains no silent actions because silent actions are about how-to-do. It follows that *spec* is weakly bisimilar to an implementation *impl* if and only if *spec* is branching bisimilar to *impl* (Theorem 5.8.18 in [2]). In addition, in majority of practical examples, the branching bisimilarity and the weak bisimilarity coincide. What these observations tell us is that as far as verification is concerned the branching bisimilarity ought to play a bigger role than the weak bisimilarity, especially in the situations where branching bismilarity is easily decided.

One major difficulty of checking weak or branching bisimilarity on normed BPA stems from the lack of nice structural properties such as unique decomposition property. By forcing the final action of every process to be observable, we have an important subset of normed BPA, called totally normed BPA, in which unique decomposition property still holds for branching bisimilarity. The bisimilarity checking on totally normed BPA also has a long history. In 1991, Hüttel [16] repeated the tableau construction developed in [17] for branching bisimilarity on totally normed BPA. Although Hüttel's construction is not sound for weak bisimilarity, the relevant decidability can also be established [12]. For the lower bound, NP-hardness is established by Stříbrná [30] for weak bisimilarity via a reduction from the knapsack problem. By inspecting Stříbrná's proof, we are aware that the NP-hardness still holds for any other bisimilarity, such as delay bisimilarity, η-bisimilarity, and even quasi-branching bisimilarity [31], except for branching bisimilarity. The requirement of branching bisimilarity that change-of-state silent actions must be explicitly bisimulated makes it impossible to realize nondeterminism by designing some gadgets via a bisimulation game. These crucial observations inspire us to rethink the possibility of designing more

efficient algorithm for the problem of checking branching bisimilarity on totally normed BPA.

The paper provides a polynomial time algorithm for checking branching bisimilarity on totally normed BPA. Therefore an instance is spotted that branching bisimilarity and weak bisimilarity are both decidable but lie in different complexity classes.

For brevity, in the rest of this paper, 'branching bisimilarity' will usually be referred to as 'bisimilarity'. We avoid using the term 'strong bisimilarity', since the strong bisimilarity can be interpreted as the bisimilarity for 'realtime' processes. A realtime process is a process which can perform no silent action.

The algorithm developed in this paper takes a similar partition refinement approach and the framework adopted in [10,7], which was designed to decide bisimilarity for realtime normed BPA. This algorithm is called CL algorithm in this paper. The final efficient implementation of our algorithm is a generalized version of CL algorithm in the sense that, for realtime systems, our algorithm and CL algorithm are essentially the same.

Our algorithm heavily relies on the technique of dynamic programming, which makes our implementation has the same computational complexity as CL algorithm. Although our algorithm seems very similar to the previous one, the technical details, including the definition of expansion and refinement operation, the theoretical development of its correctness are quite difficult than the previous CL algorithm.

Without doubt, the consecutive silent transitions in the definition of branching bisimilarity cause severe problems in two aspects: the correctness and the efficiency. Note that the totally normedness guarantees that the number of consecutive silent actions are bounded by the number of constants. It is not hard to use this observation, together with the game theoretical view of branching bisimilarity, to design an algorithm which runs in polynomial space. However, the consecutive silent actions, which cause nondeterminism, did make checking branching bisimulation property take exponential time if the naive way was taken. The only way to overcome this difficulty is a proper usage of the technique of dynamic programming. When consecutive silent actions are eliminated by means of dynamic programming, we have a severe problem: why is the resulting algorithm still correct?

In the situation of CL algorithm for realtime normed BPA, there is a predefined refinement operation, $\mathsf{Ref}(\equiv)$. In that situation, we had a canonical definition of expansion and a canonical definition of relative decreasing bisimilarity (in our terminology). The final refinement operation $\mathsf{Ref}(\equiv)$ was defined as the decreasing bisimilarity wrt. the expansion of $\equiv$. The refined equivalence relation was then constructed by a greedy algorithm, in each step of which two memberships were efficiently tested. Therefore, the correctness of CL algorithm was comparatively obvious.

Unfortunately it is unlikely, if not impossible, to generate the above proof structure for CL algorithm to our algorithm, because there is no clear way to define the expansion relation like that in CL algorithm. Note that the expansion

relation should be both correct and efficient. We had several aborted attempts before finally we decided to take some other ways.

The correctness of CL algorithm depends on a clearly defined refinement operation which relies on two steps of operations: the expansion operation and the relative decreasing bisimilarity. Our crucial insight is that, there is no need to separate these two steps of operations. The main technical line is briefly outlined below. It takes several stages:

- At first, for realtime systems, we define the refinement operations by a way of combining the two steps of operation which was taken in CL algorithm into a cohesive whole. In this way, we have noticed that we defines exactly the same refinement operation as that in CL algorithm.
- Then, the refinement operation defined in the above way is smoothly generated for the style of branching bisimilarity. In this stage, our attention is centred on the property of the refined relation. The efficiency is never cared about. We prove that the refinement operation preserves congruence and the unique decomposition property.
- Then a characterization theorem is established for the refined congruence. In this characterization, the consecutive silent actions are completely eliminated. Thus the problem of efficiency is mainly solved. Using this characterization, the correctness proof for realtime systems can be obtained. But for systems with silent actions, it is not enough.
- Finally, the proof is finished by developing a simpler characterization which corresponds to our algorithm directly. In this stage, a special property of branching bisimilarity for processes in prime decomposition turns out to be quite useful.

The rest of the paper is organized as follows. Section 2 lays down the preliminaries. Section 3 focuses on the unique decomposition property for branching bisimilarity on totally normed BPA. Then we describe our algorithm in Section 4. The suitable definition of refinement steps are discussed in Section 5, and the correctness proof are provided in Section 6. Finally, Section 7 gives additional remarks.

2 Preliminaries

Basic Process Algebra A *basic process algebra* (BPA) system is a triple $(\mathbf{C}, \mathcal{A}, \Delta)$, where $\mathbf{C} = \{X_1, \ldots, X_n\}$ is a finite set of process constants, $\mathcal{A}$ is a finite set of actions, and Δ is a finite set of transition rules. The *processes*, ranged over by $\alpha, \beta, \gamma, \delta$, are generated by the following grammar:

$$\alpha \ ::= \ \epsilon \ \mid \ X \ \mid \ \alpha_1 \cdot \alpha_2.$$

The syntactic equality is denoted by $=$. We assume that the sequential composition $\alpha_1 \cdot \alpha_2$ is associative up to $=$ and $\epsilon \cdot \alpha = \alpha \cdot \epsilon = \alpha$. Sometimes $\alpha \cdot \beta$ is shortened as $\alpha\beta$. The set of processes is exactly $\mathbf{C}^*$, the strings over $\mathbf{C}$. There

can be a special symbol τ in $\mathcal{A}$ for silent transition. Typically, ℓ is used to denote actions, while a are used to denote visible (i.e. non-silent) actions. The transition rules in Δ are of the form $X \xrightarrow{\ell} \alpha$. The following labelled transition rules define the operational semantics of the processes.

$$\frac{X \xrightarrow{\ell} P \in \Delta}{X \xrightarrow{\ell} \alpha} \qquad \frac{\alpha \xrightarrow{\ell} \alpha'}{\alpha \cdot \beta \xrightarrow{\ell} \alpha' \cdot \beta}$$

The operational semantics is structural, meaning that $\alpha \cdot \beta \xrightarrow{\ell} \alpha' \cdot \beta$ whenever $\alpha \xrightarrow{\ell} \alpha'$. We write $\Longrightarrow$ for the reflexive transitive closure of $\xrightarrow{\tau}$, and $\stackrel{\widehat{\ell}}{\Longrightarrow}$ for $\Longrightarrow\xrightarrow{\ell}\Longrightarrow$ if $\ell \neq \tau$ and for $\Longrightarrow$ otherwise.

A process α is *normed* if $\alpha \xrightarrow{\ell_1} \cdots \xrightarrow{\ell_n} \epsilon$ for some $\ell_1, \ldots, \ell_n$. A process α is *totally normed* if it is normed, and moreover, $\ell_n \neq \tau$ whenever $\alpha \xrightarrow{\ell_1} \cdots \xrightarrow{\ell_n} \epsilon$. A BPA definition $(\mathbf{C}, \mathcal{A}, \Delta)$ is (totally) normed if all processes defined in it are (totally) normed. We write (t)(n)BPA for the (totally)(normed) basic process algebra model. In other words, a tnBPA system is a nBPA system in which rules of the form $X \xrightarrow{\tau} \epsilon$ are forbidden.

We call a BPA system *realtime* if $\tau \notin \mathcal{A}$. That is to say, a realtime system can not perform silent actions. Clearly, realtime totally normed BPA is exactly realtime normed BPA.

Bisimulations and Bisimilarities In the presence of silent actions two well known process equalities are the branching bisimilarity [31] and the weak bisimilarity [26].

Definition 1. *Let $\mathcal{R}$ be a relation on processes. $\mathcal{R}$ is a* branching bisimulation, *if the following hold whenever $\alpha\mathcal{R}\beta$:*

1. *If $\alpha \xrightarrow{\ell} \alpha'$, then either*
   (a) *$\ell = \tau$ and $\alpha'\mathcal{R}\beta$; or*
   (b) *$\beta \Longrightarrow \beta'' \xrightarrow{\ell} \beta'$ and $\alpha'\mathcal{R}\beta'$ and $\alpha\mathcal{R}\beta''$ for some β', β''.*
2. *If $\beta \xrightarrow{\ell} \beta'$, then either*
   (a) *$\ell = \tau$ and $\alpha\mathcal{R}\beta'$; or*
   (b) *$\alpha \Longrightarrow \alpha'' \xrightarrow{\ell} \alpha'$ and $\alpha'\mathcal{R}\beta'$ and $\alpha''\mathcal{R}\beta$ for some α', α''.*

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

Definition 2. *A relation $\mathcal{R}$ is a* weak bisimulation *if the following are valid:*

1. *Whenever $\alpha\mathcal{R}\beta$ and $\alpha \xrightarrow{\ell} \alpha'$, then $\beta \stackrel{\widehat{\ell}}{\Longrightarrow} \beta'$ and $\alpha'\mathcal{R}\beta'$ for some β'.*
2. *Whenever $\alpha\mathcal{R}\beta$ and $\beta \xrightarrow{\ell} \beta'$, then $\alpha \stackrel{\widehat{\ell}}{\Longrightarrow} \alpha'$ and $\alpha'\mathcal{R}\beta'$ for some α'.*

The weak bisimilarity $\approx$ *is the largest weak bisimulation.*

Both $\simeq$ and $\approx$ are congruence relations for (t)nBPA. We remark that transitivity of $\simeq$ is not straightforward according to Definition 1, because the branching bisimulation $\mathcal{R}$ defined in Definition 1 need not be transitive [5]. To solve this problem, van Glabbeek and Weijland [31] introduce a slightly different notion called *semi-branching bisimulation*.

Definition 3. *Let $\mathcal{R}$ be a relation on processes. $\mathcal{R}$ is a* semi-branching bisimulation *if the following hold whenever $\alpha\mathcal{R}\beta$:*

1. *If $\alpha \xrightarrow{\ell} \alpha'$, then either*
   (a) $\ell = \tau$ and $\beta \Longrightarrow \beta'$ for some β' such that $\alpha\mathcal{R}\beta'$ and $\alpha'\mathcal{R}\beta'$; or
   (b) $\beta \Longrightarrow \beta'' \xrightarrow{\ell} \beta'$ and $\alpha'\mathcal{R}\beta'$ and $\alpha\mathcal{R}\beta''$ for some β', β''.
2. *If $\beta \xrightarrow{\ell} \beta'$, then either*
   (a) $\ell = \tau$ and $\alpha \Longrightarrow \alpha'$ for some α' such that $\alpha'\mathcal{R}\beta$ and $\alpha'\mathcal{R}\beta'$; or
   (b) $\alpha \Longrightarrow \alpha'' \xrightarrow{\ell} \alpha'$ and $\alpha'\mathcal{R}\beta'$ and $\alpha''\mathcal{R}\beta$ for some α', α''.

Then it is easy to establish the following facts:

1. A branching bisimulation is a semi-branching bisimulation.
2. A semi-branching bisimulation is transitive.
3. The largest semi-branching bisimulation is an equivalence.
4. The largest semi-branching bisimulation is a branching bisimulation.

Now the largest semi-branching bisimulation is the same as $\simeq$, the largest branching bisimulation.

If the involved system is realtime, then the branching bisimilarity and the weak bisimilarity are coincident. They are called the *strong bisimilarity* and are denoted by $\sim$ in literature. In this paper, branching bisimilarity is often abbreviated as *bisimilarity*. If the system is realtime, we also use the term bisimilarity to indicate strong bisimilarity. However, we tend to use the term 'branching bisimilarity' in the situation of discussing on its relationship with weak bisimilarity.

The following lemma, first noticed by van Glabbeek and Weijland [31], plays a fundamental role in the study of bisimilarity.

Lemma 1. *If $\alpha \Longrightarrow \alpha' \Longrightarrow \alpha'' \simeq \alpha$ then $\alpha' \simeq \alpha$.*

Let $\cong$ be a process equivalence. A silent action $\alpha \xrightarrow{\tau} \alpha'$ is *state-preserving* with regards to $\cong$ if $\alpha' \cong \alpha$; it is *change-of-state* with regards to $\cong$ if $\alpha' \ncong \alpha$. Branching bisimilarity strictly refines weak bisimilarity in the sense that only state-preserving silent actions can be ignored; a change-of-state must be explicitly bisimulated. Suppose that $\alpha \simeq \beta$ and $\alpha \xrightarrow{\ell} \alpha'$ is matched by the transition sequence $\beta \xrightarrow{\tau} \cdots \xrightarrow{\tau} \beta^i \xrightarrow{\tau} \cdots \xrightarrow{\tau} \beta'' \xrightarrow{\ell} \beta'$. By definition one has $\alpha \simeq \beta''$. It follows from Lemma 1 that $\alpha \simeq \beta^i$, meaning that all silent actions in $\beta \Longrightarrow \beta''$ are necessarily state-preserving. This property fails for the weak bisimilarity as the following example demonstrates.

Example 1. Consider the tnBPA system whose rules are defined by

$$\{X \xrightarrow{b} \epsilon,\ X \xrightarrow{\tau} X',\ X' \xrightarrow{a} \epsilon,\ X \xrightarrow{a} \epsilon;\ Y \xrightarrow{b} \epsilon,\ Y \xrightarrow{\tau} Y',\ Y' \xrightarrow{a} \epsilon\}.$$

One has $X \approx Y$. However $X \not\simeq Y$ since $Y \not\simeq Y'$.

Norm Given an tnBPA system $(\mathbf{C}, \mathcal{A}, \Delta)$. We relate a natural number $\mathtt{norm}(X)$, the *norm* of X, to every constant X, defined as the least k such that $X \Longrightarrow \xrightarrow{a_1} \Longrightarrow \ldots \Longrightarrow \xrightarrow{a_k} \epsilon$. Silent actions contribute zero to norm. $\mathtt{norm}$ is extended to processes by taking $\mathtt{norm}(\epsilon) = 0$ and $\mathtt{norm}(X \cdot \alpha) = \mathtt{norm}(X) + \mathtt{norm}(\alpha)$.

Lemma 2. *In a* tnBPA *system,* $\mathtt{norm}(\alpha) = 0$ *if and only if* $\alpha = \epsilon$.

A transition $\alpha \xrightarrow{\ell} \alpha'$ is *decreasing*, denoted by $\alpha \xrightarrow{\ell}_{\mathrm{dec}} \alpha'$ if either $\ell \neq \tau$ and $\mathtt{norm}(\alpha) = \mathtt{norm}(\alpha') + 1$, or $\ell = \tau$ and $\mathtt{norm}(\alpha) = \mathtt{norm}(\alpha')$. The notion of decreasing transitions formalizes the intuition that a transition can be extended to a path which witnesses the norm of α.

Standard Input For technical convenience, we require the input tnBPA system $(\mathbf{C}, \mathcal{A}, \Delta)$ to be *standard*, which have the following two additional properties:

1. The constants in $\mathbf{C} = \{X_i\}_{i=1}^n$ are ordered by non-decreasing norm, that is:
$$\mathtt{norm}(X_1) \leq \mathtt{norm}(X_2) \leq \ldots \leq \mathtt{norm}(X_n).$$
2. Let $\mathbf{C}_i$ be the set $\{X_1, X_2, \ldots, X_i\}$ for $i = 0, 1, \ldots, n$. In particular, $\mathbf{C}_0 = \emptyset$ and $\mathbf{C}_n = \mathbf{C}$. Assume $X_i \xrightarrow{\ell}_{\mathrm{dec}} \alpha$, we need the property $\alpha \in \mathbf{C}_{i-1}^*$. This property does not hold in general because of the existence of loops like $X_i \Longrightarrow X_j \Longrightarrow X_i$. In this case we have $X_i \simeq X_j$ by Lemma 1, and we can transform the system by contracting X_i and X_j into one constant (removing X_j and substituting all occurrences of X_j in Δ by X_i) and eliminating the loop rules. All loops can be eliminated in this way. (By totally normedness, $X \overset{\tau}{\Longrightarrow}_{\mathrm{dec}} X \cdot Y$ is impossible.) Afterwards, we specify a partial order $\preceq \in \mathbf{C} \times \mathbf{C}$ such that $X \prec X'$ if and only if either $\mathtt{norm}(X) < \mathtt{norm}(X')$ or $X' \Longrightarrow_{\mathrm{dec}} X$. Then the order of constants are chosen to be any total order which extends $\prec$. These works can be done by computing the 'dependency graph' and then calling an algorithm for topological sort.

The size of a tnBPA system $(\mathbf{C}, \mathcal{A}, \Delta)$ is denoted by $|\Delta|$. A procedure is said to be *efficient* if it runs in polynomial time. The above discussion confirms that any tnBPA system can be efficiently transformed to a standard one with no size growing.

Lemma 3. *For every* tnBPA *system* $(\{X_1, X_2, \ldots . X_n\}, \mathcal{A}, \Delta)$*, there is a standard* tnBPA *system* $(\{X_1', X_2', \ldots, X_m'\}, \mathcal{A}, \Delta')$ *computable in at most* $\mathcal{O}(|\Delta|^2)$ *time, in which* $m \leq n$ *and* $|\Delta'| \leq |\Delta|$.

From now on, the input tnBPA system is supposed to be standard, and is fixed as $(\mathbf{C}, \mathcal{A}, \Delta)$ where $\mathbf{C} = \{X_1, X_2, \ldots . X_n\}$. We will invariantly use n to denote the size of $\mathbf{C}$, and N to denote the size of the related tnBPA system.

The problem is formally defined as follows:

Problem:	Branching Bisimilarity on tnBPA
Instance:	A standard tnBPA system $(\mathbf{C} = \{X_i\}_{i=1}^n, \mathcal{A}, \Delta)$, and $\alpha, \beta \in \mathbf{C}^*$.
Question:	$\alpha \simeq \beta$?

We restate the important property for standard systems as the following lemma.

Lemma 4. *Assume* $X_i \xrightarrow{\ell}_{\mathrm{dec}} \alpha$*, we have* $\alpha \in \mathbf{C}^*_{i-1}$.

Other Conventions We will always use notation $\equiv$ to denote an equivalence/congruence relation on $\mathbf{C}^*$. An equivalence/congruence relation $\equiv$ is *norm-preserving* if $\mathtt{norm}(\alpha) = \mathtt{norm}(\alpha')$ whenever $\alpha \equiv \alpha'$. In this paper, all the equivalence/congruence relations are supposed to be norm-preserving. This fact is not always explicitly stated.

3 Finite Representations

In this section, we propose a convenient way of representing bisimilarity and the approximating congruences [1].

From the algebraic view, the set of processes of tnBPA is exactly the free monoid generated by $\mathbf{C}$. The question is how to represent a congruence relation on $\mathbf{C}^*$. We will show that the bisimilarity $\simeq$ is a very special congruence. Not only is it finitely generated, but it enjoys a highly structured property called *unique decomposition property*.

3.1 Unique Decomposition Property of $\simeq$

Unique decomposition property plays a central role in all the algorithms for bisimilarity checking on realtime nBPA. This important property also holds for bisimilarity on tnBPA.

Recall that a congruence $\equiv$ is *norm-preserving* if $\mathtt{norm}(\alpha) = \mathtt{norm}(\beta)$ whenever $\alpha \equiv \beta$. The following lemma is a direct consequence of Definition 1.

Lemma 5. $\simeq$ *is a norm-preserving congruence.*

Let $\equiv \subseteq \mathbf{C}^* \times \mathbf{C}^*$ be an arbitrary norm-preserving congruence. Intuitively, a constant process X_i is a composite if $X_i \equiv \alpha\beta$ for some $\alpha, \beta \neq \epsilon$. In this case we also have $\mathtt{norm}(\alpha), \mathtt{norm}(\beta) < \mathtt{norm}(X_i)$ from Lemma 2. For technical convenience we will define X_i to be a *composite* modulo $\equiv$ if $X_i \equiv \alpha$ for some $\alpha \in \mathbf{C}^*_{i-1}$. Otherwise, X_i is called a *prime* modulo $\equiv$.

Let $\mathbf{P} \subseteq \mathbf{C}$ be the set of primes modulo $\equiv$. By Lemma 5 and the well-foundedness of natural numbers, every $X \in \mathbf{C}$ has a *prime decomposition* $\alpha \in \mathbf{P}^*$ such that $X_i \equiv \alpha$. We say that $\equiv$ has unique decomposition property, or simply $\equiv$ is *decompositional* if every process has exactly one prime decomposition.

It is the time to establish the unique decomposition property of $\simeq$. The following Lemma 6 and Theorem 1 is standard, as is in the case of bisimilarity for realtime nBPA [13]. The *right cancellation* property is established first.

[1] The proofs in this section is a generalization of the corresponding work for realtime normed BPA, say [13]. The readers familiar with these former works can only skim this part.

Lemma 6 (Right Cancellation). *$\alpha\gamma \simeq \beta\gamma$ entails $\alpha \simeq \beta$.*

Proof. $\{(\alpha, \beta) : \alpha\gamma \simeq \beta\gamma \text{ for some } \gamma\}$ is a bisimulation. □

Theorem 1 (Unique Decomposition Property of $\simeq$). *$\simeq$ is decompositional. Let $X_{i_1} \dots X_{i_p}$ and $X_{j_1} \dots X_{j_q}$ be two irreducible decompositions such that $X_{i_1} \dots X_{i_p} \simeq X_{j_1} \dots X_{j_q}$. Then, $p = q$ and $X_{i_t} \simeq X_{j_t}$ for every $1 \leq t \leq p$.*

Proof. Assume on the contrary that $X_{i_1} \dots X_{i_p}$ and $X_{j_1} \dots X_{j_q}$ be two different irreducible decompositions with the least norm such that

$$X_{i_1} \dots X_{i_p} \simeq X_{j_1} \dots X_{j_q}.$$

Suppose that

$$X_{i_1} \dots X_{i_p} \Longrightarrow_{\mathrm{dec}} \xrightarrow{a}_{\mathrm{dec}} \gamma X_{i_2} \dots X_{i_p}. \tag{1}$$

These actions must be bisimulated (matched) by

$$X_{j_1} \dots X_{j_q} \Longrightarrow_{\mathrm{dec}} \xrightarrow{a}_{\mathrm{dec}} \delta X_{j_2} \dots X_{j_q} \tag{2}$$

for some δ such that $\gamma X_{i_2} \dots X_{i_p} \simeq \delta X_{j_2} \dots X_{j_q}$. Since the norm of $\gamma X_{i_2} \dots X_{i_p}$ and $\delta X_{j_2} \dots X_{j_q}$ is strictly decremented, we have $X_{i_p} \simeq X_{j_q}$ from the induction hypothesis. Now by right cancellation lemma, $X_{i_1} \dots X_{i_{p-1}} \simeq X_{j_1} \dots X_{j_{q-1}}$. This contradicts with the minimum norm assumption. □

On the other direction, right or left cancellation property is an implication of unique decomposition property.

Lemma 7. *Let $\equiv$ be decompositional. Then $\alpha\gamma \equiv \beta\gamma$ (or $\gamma\alpha \equiv \gamma\beta$) implies $\alpha \equiv \beta$.*

Remark 1. The proof of Lemma 6 and Theorem 1 is standard [3,13]. Although the proof is fairly straightforward, it heavily depends on *branching* bisimilarity and *totally* normedness. For example in the above proof when actions coming from X_{i_1} in (1) are matched by the actions in (2), the crucial point is that X_{j_2} is never used. This cannot be proved in the case of weak bisimilarity, or in the case without totally normedness. We will have the following two counterexamples if branching bisimilarity is replaced by *weak* bisimilarity, or if the condition of totally normedness is abandoned.

Example 2. This counterexample is borrowed from [16]. Consider the tnBPA system $(\{X, Y, B, A\}, \{a\}, \Delta)$, with

$$\Delta = \{X \xrightarrow{a} Y, Y \xrightarrow{a} \epsilon, Y \xrightarrow{\tau} X, A \xrightarrow{a} \epsilon, A \xrightarrow{\tau} B, B \xrightarrow{a} \epsilon\}.$$

Clearly, $AY \approx BY$ but $A \not\approx B$. Right cancellation property does not hold, neither does the unique decomposition property hold.

Example 3. Consider the nBPA system $(\{X\}, \{a\}, \Delta)$, with

$$\Delta = \{X \xrightarrow{a} X, X \xrightarrow{\tau} \epsilon\}.$$

Clearly, $X \simeq XX \simeq XXX \simeq \dots$. Unique decomposition property fails in this example merely because the existence of *idempotent* processes.

3.2 Decomposition Bases

A decompositional congruence over $\mathbf{C}^*$ can be represented by a decomposition base. A *decomposition base* $\mathcal{B}$ is a pair $(\mathbf{P}, \mathbf{E})$, in which $\mathbf{P} \subseteq \mathbf{C}$ specifies the set of primes, and $\mathbf{E}$ is a finite set of equations of the form $X = \alpha_X$ for every $X \in \mathbf{C} - \mathbf{P}$ and $\alpha_X \in \mathbf{P}^*$. The equation $X = \alpha_X$ realizes the fact that every composite X is equal to a string of primes α_X which is the *prime decomposition* of X. The congruence relation generated by $\mathcal{B}$ is denoted by $\overset{\mathcal{B}}{\equiv}$.

The *prime decomposition* of a process α with regard to $\mathcal{B}$ is denoted by $\mathtt{dcmp}_{\mathcal{B}}(\alpha)$. Formally, we set $\mathtt{dcmp}_{\mathcal{B}}(X) = X$ when $X \in \mathbf{P}$, and $\mathtt{dcmp}_{\mathcal{B}}(X) = \alpha_X$ wherever the equation $X = \alpha_X$ is in $\mathbf{E}$. The domain of $\mathtt{dcmp}_{\mathcal{B}}$ is extended to $\mathbf{C}^*$ naturally by setting $\mathtt{dcmp}_{\mathcal{B}}(\epsilon) = \epsilon$ and $\mathtt{dcmp}_{\mathcal{B}}(\alpha \cdot \beta) = \mathtt{dcmp}_{\mathcal{B}}(\alpha) \cdot \mathtt{dcmp}_{\mathcal{B}}(\beta)$.

The following lemma makes checking $\alpha \overset{\mathcal{B}}{\equiv} \beta$ fairly easy by only computing the prime decompositions of α and β.

Lemma 8. *$\alpha \overset{\mathcal{B}}{\equiv} \beta$ if and only if $\mathtt{dcmp}_{\mathcal{B}}(\alpha) = \mathtt{dcmp}_{\mathcal{B}}(\beta)$.*

In the rest of the paper, every congruence $\mathcal{B} = (\mathbf{P}, \mathbf{E})$ generated by a decomposition base $\mathcal{B}$ is assumed to be norm-preserving. Thus we must have $\mathtt{norm}(X) = \mathtt{norm}(\alpha_X)$ if the equation $X = \alpha_X$ is in $\mathbf{E}$.

The following lemma formalizes the important observation that prime constants do not have state-preserving silent actions.

Lemma 9. *Let $\mathcal{B} = (\mathbf{P}, \mathbf{E})$ be a decomposition base, and $X_i \in \mathbf{P}$. Assume $X_i \xrightarrow{\ell}_{\mathrm{dec}} \alpha$, we have $X_i \overset{\mathcal{B}}{\not\equiv} \alpha$.*

Proof. According to Lemma 4, $\alpha \in \mathbf{C}_{i-1}$.

1. If $\ell = \tau$ and $\mathtt{norm}(X_i) = \mathtt{norm}(\alpha)$. In this case, if we have $X_i \overset{\mathcal{B}}{\equiv} \alpha$, then according to the fact that X_i being prime and Lemma 8, $X_i = \mathtt{dcmp}_{\mathcal{B}}(X_i) = \mathtt{dcmp}_{\mathcal{B}}(\alpha)$. This is a contradiction.
2. If $\ell \neq \tau$ and $\mathtt{norm}(X_i) = \mathtt{norm}(\alpha) + 1$, we cannot have $X_i \overset{\mathcal{B}}{\equiv} \alpha$ because $\overset{\mathcal{B}}{\equiv}$ is norm preserving. □

The above property can be lifted from constants to processes, regarding Lemma 7.

Lemma 10. *Let $\mathcal{B} = (\mathbf{P}, \mathbf{E})$ be a decomposition base, and $\alpha \in \mathbf{P}^*$. Assume $\alpha \xrightarrow{\ell}_{\mathrm{dec}} \gamma$, we have $\alpha \overset{\mathcal{B}}{\not\equiv} \gamma$.*

Remark 2. Algebraically, a decomposition base $\mathcal{B}$ can be understood as a *finite presentation* of a monoid. In fact, $\mathcal{B}$ specifies the quotient monoid $\mathbf{C}^* / \overset{\mathcal{B}}{\equiv}$. Moreover, the unique decomposition property says that the quotient monoid $\mathbf{C}^* / \overset{\mathcal{B}}{\equiv}$ is a free monoid. From computational point of view, $\mathcal{B}$ is a *string rewriting system*. Rewriting rules are exact the equations in $\mathbf{E}$ from left to right. Strings in *normal forms* are exact $\mathbf{P}^*$, the free monoid generated by $\mathbf{P}$. All composites

can be reduced to its prime decompositions. Any $\alpha \in \mathbf{C}^*$ has a normal form. Church-Rosser property is guaranteed by the unique decomposition property, which makes checking $\alpha \overset{\mathcal{B}}{\equiv} \beta$ fairly easy by merely rewriting α and β to their normal forms.

4 Description of the Algorithm

This section serves as the description of our algorithm. The algorithm takes the partition refinement approach. It is a generalized version of the one in [10], which we call CL algorithm. However, unlike the original CL algorithm, the correctness of our algorithm is not obvious and is much more difficult to prove. This is the reason why we describe the algorithm before we prove its correctness. During the description, we also show some properties and requirements which make the algorithm work. A few properties are not proved until Section 5.

4.1 Partition Refinements with Decomposition Bases

In order to decide whether $\alpha \simeq \beta$, we start with an initial congruence relation $\equiv_0$, and iteratively refine it. The refinement operation will be denoted by Ref. By taking $\equiv_{i+1} = \mathsf{Ref}(\equiv_i)$, we have a sequence of congruence relations

$$\equiv_0, \equiv_1, \equiv_2, \ldots$$

which satisfy

$$\equiv_0 \supseteq \equiv_1 \supseteq \equiv_2 \supseteq \ldots.$$

The correctness of the refinement operation adopted in this paper depends on the following requirements:

1. $\simeq \subseteq \equiv_0$.
2. $\mathsf{Ref}(\simeq) = \simeq$.
3. If $\simeq \subsetneq \equiv$, then $\simeq \subseteq \mathsf{Ref}(\equiv) \subsetneq \equiv$.

Once the sequence becomes stable, say $\equiv_i = \equiv_{i+1}$, we have $\simeq = \equiv_i$.

Remark 3. The refinement operation taken in this paper leads to a monotonic sequence $\{\equiv_i\}_{i \in \omega}$. Namely,

$$\equiv_0 \supseteq \equiv_1 \supseteq \equiv_2 \supseteq \ldots.$$

This property is not necessary in a general framework of refinement. One alternative is to replace the third requirement above by the following two:

3'. Ref is monotone. $\mathsf{Ref}(\equiv) \subseteq \mathsf{Ref}(\equiv')$ whenever $\equiv \subseteq \equiv'$.
4'. If $\simeq \subsetneq \equiv$, then $\mathsf{Ref}(\equiv) \neq \equiv$.

In the algorithm, the congruences $\simeq$ and $\equiv_i$'s are all represented by decomposition bases. That is, all the intermediate $\equiv_i$ must be decompositional congruences. In the following, we will develop an implementation of the refinement steps in polynomial time.

On the whole, the algorithm is an iteration:

1. Compute the initial base $\mathcal{B}_{\text{init}}$ and set $\mathcal{B} = \mathcal{B}_{\text{init}}$.
2. Compute the base $\mathcal{B}'$ from $\mathcal{B}$.
3. If $\mathcal{B}'$ equals $\mathcal{B}$ then halt and return $\mathcal{B}$.
4. Assign new base $\mathcal{B}'$ to $\mathcal{B}$ and go to step 2.

Apparently, the algorithm relies on the base $\mathcal{B}_{\text{init}}$ of the initial congruence $\equiv_0$ and the refinement step, computing $\mathcal{B}'$ from $\mathcal{B}$.

4.2 Outline of the Algorithm

The framework of the algorithm is described as Fig. 1.

Initial Congruence The base $\mathcal{B}_{\text{init}} = (\mathbf{P}_{\text{init}}, \mathbf{E}_{\text{init}})$ of the initial congruence $\equiv_0$ is set as:

- $\mathbf{P}_{\text{init}} = X_1$,
- $\mathbf{E}_{\text{init}}$ contains $X_i = \underbrace{X_1 \cdot X_1 \cdot \ldots \cdot X_1}_{\mathtt{norm}(X_i) \text{ times}}$ for every $i > 1$.

For $\equiv_0$, we have the following properties.

Lemma 11. *$\alpha \equiv_0 \beta$ if and only if $\mathtt{norm}(\alpha) = \mathtt{norm}(\beta)$.*

Lemma 12. *1. $\equiv_0 \supseteq \simeq$.*
2. $\equiv_0$ is a norm-preserving and decompositional congruence.

Properties of Refinement Steps In order to understand the framework of the algorithm, We need to investigate the relationship between $\mathcal{B}' = (\mathbf{P}', \mathbf{E}')$ and $\mathcal{B} = (\mathbf{P}, \mathbf{E})$ in step 2. Later from the algorithm, we will confirm that $\overset{\mathcal{B}'}{\equiv} \subseteq \overset{\mathcal{B}}{\equiv}$. Under this condition, we have the following key observation.

Lemma 13. *Let $\mathcal{B} = (\mathbf{P}, \mathbf{E})$ and $\mathcal{B}' = (\mathbf{P}', \mathbf{E}')$ be two decomposition bases.*

1. *If $\overset{\mathcal{B}'}{\equiv} \subseteq \overset{\mathcal{B}}{\equiv}$, then $\mathbf{P} \subseteq \mathbf{P}'$.*
2. *If $\mathbf{P}' = \mathbf{P}$, then $\mathcal{B}' = \mathcal{B}$.*

Proof. 1. Suppose $X_i \notin \mathbf{P}'$, we show $X_i \notin \mathbf{P}$. Since $X_i \notin \mathbf{P}'$, there is an equation $X_i = \alpha$ in $\mathbf{E}'$ for some $\alpha \in \mathbf{C}_{i-1}^*$, which means $X_i \overset{\mathcal{B}'}{\equiv} \alpha$. Because $\overset{\mathcal{B}'}{\equiv} \subseteq \overset{\mathcal{B}}{\equiv}$, we have $X_i \overset{\mathcal{B}}{\equiv} \alpha$, which means that X_i is not a prime modulo $\overset{\mathcal{B}}{\equiv}$. That is, $X_i \notin \mathbf{P}$.

Framework of the algorithm:

1. Initialize $\mathcal{B} = (\mathbf{P}, \mathbf{E})$;
2. $\mathbf{P}' \coloneqq \mathbf{P}$;
3. **repeat**
4. $\quad \mathbf{P} \coloneqq \mathbf{P}'$; $\mathbf{E} \coloneqq \mathbf{E}'$; $\mathbf{E}' \coloneqq \emptyset$;
5. $\quad$ **for** each $X_i \in \mathbf{C} \setminus \mathbf{P}$ **do**
6. $\quad\quad s \coloneqq \mathtt{dcmp}_{(\mathbf{P}', \mathbf{E}')}(\alpha_i)$;
7. $\quad\quad flag \coloneqq$ **true**;
8. $\quad\quad k \coloneqq \mathtt{lpfindex}_{(\mathbf{P}, \mathbf{E})}(X_i)$;
9. $\quad\quad$ **for** each $X_j \in \{\mathtt{lpf}_{(\mathbf{P}, \mathbf{E})}(X_i)\} \cup \{X_{k+1}, \ldots, X_{i-1}\} \cap (\mathbf{P}' \setminus \mathbf{P})$ **do**
10. $\quad\quad\quad$ **if** $\mathtt{lpftest}_{(\mathbf{P}', \mathbf{E}')}(X_i, X_j)$ **then**
11. $\quad\quad\quad\quad \mathbf{E}' \coloneqq \mathbf{E}' \cup \{X_i = X_j \cdot \mathtt{sffx}(\mathtt{norm}(X_i) - \mathtt{norm}(X_j); s)\}$;
12. $\quad\quad\quad\quad flag \coloneqq$ **false**;
13. $\quad\quad\quad$ **end if**
14. $\quad\quad$ **end for**
15. $\quad\quad$ **if** $flag$ **then**
16. $\quad\quad\quad \mathbf{P}' \coloneqq \mathbf{P}' \cup \{X_i\}$;
17. $\quad\quad$ **end if**
18. $\quad$ **end for**
19. **until** $\mathbf{P} = \mathbf{P}'$

Fig. 1. Framework of Efficient Algorithm

2. Suppose that $\mathcal{B}' \subsetneq \mathcal{B}$. Then there is some X_i such that $\mathtt{dcmp}_{\mathcal{B}}(X_i) \neq \mathtt{dcmp}_{\mathcal{B}'}(X_i)$. We have $X_i \overset{\mathcal{B}}{\equiv} \mathtt{dcmp}_{\mathcal{B}}(X_i)$ and $X_i \overset{\mathcal{B}'}{\equiv} \mathtt{dcmp}_{\mathcal{B}'}(X_i)$ Since $\overset{\mathcal{B}'}{\equiv} \subseteq \overset{\mathcal{B}}{\equiv}$, we have $X_i \overset{\mathcal{B}}{\equiv} \mathtt{dcmp}_{\mathcal{B}'}(X_i)$, thus $\mathtt{dcmp}_{\mathcal{B}}(X_i) \overset{\mathcal{B}}{\equiv} \mathtt{dcmp}_{\mathcal{B}'}(X_i)$. Since $\mathtt{dcmp}_{\mathcal{B}}(X_i)$ and $\mathtt{dcmp}_{\mathcal{B}'}(X_i)$ are both in $\mathbf{P}^*$, we have $\mathtt{dcmp}_{\mathcal{B}}(X_i) = \mathtt{dcmp}_{\mathcal{B}'}(X_i)$, a contradiction. □

According to Lemma 13, we call constants in $\mathbf{P}$ *old primes* and constants in $\mathbf{P}' \setminus \mathbf{P}$ *new primes*. During the iterative procedure of refinement, once a constant becomes prime, it is a prime thereafter. If at certain step of iteration there is no new prime to add, the algorithm terminates. Thus we have the following property.

Proposition 1. *There can be at most n steps of iteration in the algorithm.*

This confirms the termination of the algorithm and provides an implementation of the step 3 by checking if there are new primes. The remaining thing is to study the implementation of step 2.

Computing $\mathcal{B}'$ from $\mathcal{B}$ Computation of $\mathcal{B}'$ proceeds as follows. First we assign $\mathbf{P}' = \mathbf{P}$ and $\mathbf{E}' = \emptyset$. Then we add appropriate constants to $\mathbf{P}'$ and appropriate equations to $\mathbf{E}'$. For every $i = 2, \ldots, n$ with $X_i \in \mathbf{C} \setminus \mathbf{P}$, we check whether there

exists $\delta \in (\mathbf{P}' \cap \mathbf{C}_{i-1})^*$ such that $X_i \overset{\mathcal{B}'}{\equiv} \delta$. If not, we add X_i to $\mathbf{P}'$, otherwise we add the appropriate equation $X_i = \alpha$ to $\mathbf{E}'$. We emphasize that at the time X_i is treated, we have already known whether $X_j \in \mathbf{P}'$ and $\mathtt{dcmp}_{\mathcal{B}'}(X_j)$ for every $j < i$.

The efficient computation of $\mathcal{B}'$ from $\mathcal{B}$ relies on the following three aspects:

1. The candidates δ for testing $X_i \overset{\mathcal{B}'}{\equiv} \delta$ must be 'small'.
2. We need an correct and efficient way of deciding whether (X_i, δ) can be put into $\mathbf{E}'$, i.e. $X_i \overset{\mathcal{B}'}{\equiv} \delta$.
3. We need an efficient representation and manipulation on strings.

The representation and operations on long strings can be implemented in a systematic way and will be discussed shortly in Section 4.5. For the moment, we suppose that all the operations on strings appears in the algorithm are polynomial time computable.

4.3 Small Set of Candidates

Now we confirm that, for every X_i, there is a small number of δ's which are required to determine whether $X_i \overset{\mathcal{B}'}{\equiv} \delta$. In the case of realtime nBPA, this is a significant discovery in CL algorithm, for it greatly reduces the expense of the algorithm. The same way is taken here, but the rationality will be confirmed later.

Let $\mathcal{B} = (\mathbf{P}, \mathbf{E})$ be a decomposition base. We say that prime constant $X_j \in \mathbf{P}$ is the *leftmost prime factor* of X_i wrt. $\mathcal{B}$, denoted by $\mathtt{lpf}_{\mathcal{B}}(X_i) = X_j$, if $\mathtt{dcmp}_{\mathcal{B}}(X_i) = X_j \cdot \gamma$ for some γ. Clearly, $\mathtt{lpf}_{\mathcal{B}}(X_i)$ is unique.

Now fix one decreasing transition rule $X_i \xrightarrow{\ell_i}_{\mathrm{dec}} \alpha_i$ ($\ell_i = \tau$ is allowed.) for every $X_i \in \mathbf{C}$. We use $\mathtt{sffx}(h; \alpha)$ to denote the suffix of string α with norm h. Note that $\mathtt{sffx}(h; \alpha)$ is undefined unless α has such a suffix with norm h.

Proposition 2. *Let $\mathcal{B}$ be a decomposition base such that $\overset{\mathcal{B}}{\equiv}$ is a decreasing branching bisimulation (Definition 8, Section 5.3). If $\mathtt{lpf}_{\mathcal{B}}(X_i) = X_j$, then*

$$X_i \overset{\mathcal{B}}{\equiv} X_j \cdot \mathtt{sffx}(\mathtt{norm}(X_i) - \mathtt{norm}(X_j); \mathtt{dcmp}_{\mathcal{B}}(\alpha_i)).$$

Proof. From $\mathtt{lpf}_{\mathcal{B}}(X_i) = X_j$, we have $X_i \overset{\mathcal{B}}{\equiv} X_j \cdot \alpha$ for some α satisfying $\mathtt{norm}(\alpha) = \mathtt{norm}(X_i) - \mathtt{norm}(X_j)$. Knowing $\overset{\mathcal{B}}{\equiv}$ is a decreasing branching bisimulation, we consider the transition $X_i \xrightarrow{\ell_i}_{\mathrm{dec}} \alpha_i$. There are two cases:

- $\ell_i = \tau$ and $\alpha_i \overset{\mathcal{B}}{\equiv} X_j \cdot \alpha$. In this case, let $\beta = X_j$ and we have $\alpha_i \overset{\mathcal{B}}{\equiv} \beta \cdot \alpha$.
- $\ell_i \neq \tau$ or $\alpha_i \overset{\mathcal{B}}{\not\equiv} X_j \cdot \alpha$. In this case, we have $X_j \Longrightarrow_{\mathrm{dec}} \xrightarrow{\ell_i}_{\mathrm{dec}} \beta$ such that $\alpha_i \overset{\mathcal{B}}{\equiv} \beta \cdot \alpha$.

In either case, we have $\alpha_i \overset{\mathcal{B}}{\equiv} \beta \cdot \alpha$ for some β. According to the fact that $\overset{\mathcal{B}}{\equiv}$ is decompositional, we get $\mathtt{dcmp}_{\mathcal{B}}(\alpha_i) = \mathtt{dcmp}_{\mathcal{B}}(\beta) \cdot \mathtt{dcmp}_{\mathcal{B}}(\alpha)$, and consequently $\mathtt{dcmp}_{\mathcal{B}}(\alpha) = \mathtt{sffx}(\mathtt{norm}(\alpha); \mathtt{dcmp}_{\mathcal{B}}(\alpha_i)) = \mathtt{sffx}(\mathtt{norm}(X_i) - \mathtt{norm}(X_j); \mathtt{dcmp}_{\mathcal{B}}(\alpha_i))$, hence $\alpha \overset{\mathcal{B}}{\equiv} \mathtt{sffx}(\mathtt{norm}(X_i) - \mathtt{norm}(X_j); \mathtt{dcmp}_{\mathcal{B}}(\alpha_i))$. Recall that $X_i \overset{\mathcal{B}}{\equiv} X_j \cdot \alpha$, we get $X_i \overset{\mathcal{B}}{\equiv} X_j \cdot \mathtt{sffx}(\mathtt{norm}(X_i) - \mathtt{norm}(X_j); \mathtt{dcmp}_{\mathcal{B}}(\alpha_i))$. □

Assume that $\overset{\mathcal{B}'}{\equiv} \subseteq \overset{\mathcal{B}}{\equiv}$. Comparing $\mathtt{lpf}_{\mathcal{B}'}(X_i)$ with $\mathtt{lpf}_{\mathcal{B}}(X_i)$, there are two possibilities: $\mathtt{lpf}_{\mathcal{B}'}(X_i) = \mathtt{lpf}_{\mathcal{B}}(X_i)$ or $\mathtt{lpf}_{\mathcal{B}'}(X_i) \neq \mathtt{lpf}_{\mathcal{B}}(X_i)$. If $\mathtt{lpf}_{\mathcal{B}'}(X_i) \neq \mathtt{lpf}_{\mathcal{B}}(X_i)$, the following property confirms that $\mathtt{lpf}_{\mathcal{B}'}(X_i)$ must be a new prime.

Proposition 3. *Assume that* $\overset{\mathcal{B}'}{\equiv} \subseteq \overset{\mathcal{B}}{\equiv}$. *Let* $X_{j'} = \mathtt{lpf}_{\mathcal{B}'}(X_i)$ *and* $X_j = \mathtt{lpf}_{\mathcal{B}}(X_i)$. *If* $j' \neq j$, *then* $j' > j$ *and* $X_{j'} \in \mathbf{P}' \setminus \mathbf{P}$.

Proof. Assume on the contrary that $X_{j'} \in \mathbf{P}$, we have $X_i \overset{\mathcal{B}}{\equiv} X_j \cdot \gamma \overset{\mathcal{B}}{\equiv} X_{j'} \cdot \gamma'$, which violates unique decomposition property of $\overset{\mathcal{B}}{\equiv}$. □

Now we can illustrate the algorithm framework in Fig. 1. The **repeat** block at line 3 realize the procedure of iteration. At every iteration, $\mathcal{B} = (\mathbf{P}, \mathbf{E})$ is updated to $\mathcal{B}' = (\mathbf{P}', \mathbf{E}')$. During an iteration, every constant X_i which is current composite is treated in the fixed index order via the outer **for** block at line 5. Note that, when X_i is treated, $\mathtt{dcmp}_{\mathcal{B}'}(\alpha_i)$ can be determined. Then the inner **for** block at line 9 is used for discovering a new decomposition of X_i for $\mathcal{B}'$ by determining the leftmost prime factor X_j of X_i. By Proposition 3, X_j can be unchanged (in the case $X_j = \mathtt{lpf}_{(\mathbf{P},\mathbf{E})}(X_i)$), or be a new prime less than X_i (in the case $X_j \in (\mathbf{P}' \setminus \mathbf{P})$ and $\mathtt{lpfindex}_{\mathcal{B}}(X_i) < j < i$), or be X_i itself (in the case no X_j is found in the inner **for** block at line 9). In the last case, variable $flag$ which is set true at line 7 remains being **true** and X_i is added to the set $\mathbf{P}'$ of new primes (line 15). The operation $\mathtt{lpfindex}_{\mathcal{B}}(X_i)$ returns index k such that $\mathtt{lpf}_{\mathcal{B}}(X_i) = X_k$. Using Proposition 2 and Proposition 3, the set of candidates can be confined into the form of $X_j \cdot \mathtt{sffx}(\mathtt{norm}(X_i) - \mathtt{norm}(X_j); \mathtt{dcmp}_{\mathcal{B}'}(\alpha_i))$, Note that, in the inner **for** block, procedure $\mathtt{lpftest}_{\mathcal{B}'}(X_i, X_j)$ is used to check whether X_j is the leftmost prime factor of X_i modulo $\overset{\mathcal{B}'}{\equiv}$. In fact, it tests whether

$$X_i \overset{\mathcal{B}'}{\equiv} X_j \cdot \mathtt{sffx}(\mathtt{norm}(X_i) - \mathtt{norm}(X_j); \mathtt{dcmp}_{\mathcal{B}'}(\alpha_i)). \tag{3}$$

In the rest part of this paper, the right hand side of Equation (3) is denoted by δ. We remark that $\delta \in \mathbf{P}' \cap \mathbf{C}_{i-1}$. Our goal is to find an efficient way to check whether $X_i \overset{\mathcal{B}'}{\equiv} \delta$.

Remark 4. The small number of candidates of δ relies on Proposition 2, which requires that $\overset{\mathcal{B}'}{\equiv}$ be a decreasing bisimulation. The definition of decreasing bisimulation will be introduced in Section 5.3. According to the refinement operation defined in Section 5, $\overset{\mathcal{B}'}{\equiv}$ is assured to be a decreasing bisimulation.

Checking $X_i \overset{\mathcal{B}'}{\equiv} \delta$:

1. **test** $\mathtt{dcmp}_{\mathcal{B}}(X_i) = \mathtt{dcmp}_{\mathcal{B}}(\delta)$. if so, goto step 2; else **reject** (X_i, δ).
2. **test** for every $X_i \xrightarrow{\ell}_{\mathrm{dec}} \alpha$, we have
   (a) either $\ell = \tau$ and $\mathtt{dcmp}_{\mathcal{B}'}(\alpha) = \delta$;
   (b) or $\delta \xrightarrow{\ell}_{\mathrm{dec}} \beta$ for some β and $\mathtt{dcmp}_{\mathcal{B}'}(\alpha) = \mathtt{dcmp}_{\mathcal{B}'}(\beta)$.
   If so, goto step 3; else, **reject** (X_i, δ).
3. **test** for every $X_i \xrightarrow{\ell}_{\mathrm{inc}} \alpha$, we have
   $\delta \xrightarrow{\ell}_{\mathrm{inc}} \beta$ for some β and $\mathtt{dcmp}_{\mathcal{B}}(\alpha) = \mathtt{dcmp}_{\mathcal{B}}(\beta)$.
   If so, goto step 4; else, **reject** (X_i, δ).
4. **test** whether $X_i \xrightarrow{\tau}_{\mathrm{dec}} \alpha$ for some α such that $\mathtt{dcmp}_{\mathcal{B}'}(\alpha) = \delta$.
   If so, goto step 7; else, goto step 5.
5. **test** for every $\delta \xrightarrow{\ell}_{\mathrm{dec}} \beta$, we have
   $X_i \xrightarrow{\ell}_{\mathrm{dec}} \alpha$ for some α such that $\mathtt{dcmp}_{\mathcal{B}'}(\alpha) = \mathtt{dcmp}_{\mathcal{B}'}(\beta)$.
   If so, goto step 6; else, **reject** (X_i, δ).
6. **test** for every $\delta \xrightarrow{\ell}_{\mathrm{inc}} \beta$, we have
   $X_i \xrightarrow{\ell}_{\mathrm{inc}} \alpha$ for some α such that $\mathtt{dcmp}_{\mathcal{B}}(\alpha) = \mathtt{dcmp}_{\mathcal{B}}(\beta)$.
   If so, goto step 7; else, **reject** (X_i, δ).
7. **accept** (X_i, δ).

Fig. 2. Checking $X_i \overset{\mathcal{B}'}{\equiv} \delta$

4.4 Efficient Way of Testing $X_i \overset{\mathcal{B}'}{\equiv} \delta$

The algorithm framework described in Fig. 1 tells us an efficient way for the implementation of partition refinement on the unique decomposition congruences. Up to now, we have not discuss how the refinement operation is and how shall we realize it efficiently. That is, how $\mathtt{lpftest}_{(\mathbf{P}', \mathbf{E}')}(X_i, X_j)$ at line 10 is implemented. Now we present the details. That is , we present an efficient way to check whether $X_i \overset{\mathcal{B}'}{\equiv} \delta$. In this way, we define $\mathcal{B}'$ from $\mathcal{B}$ via the algorithm.

The whole testing is described in Fig. 2. In later sections, we have further discussions on this implementation. For now, we only remark that, in the situation of realtime nBPA, this implementation coincides with CL algorithm. The proof of correctness is deferred to Section 5.

We can state two properties which need to be used to make the whole framework Fig. 1 work.

Lemma 14. *In every iteration of Fig. 1, we get a decomposition base $\mathcal{B}'$ from $\mathcal{B}$. The following hold:*

1. $\overset{\mathcal{B}'}{\equiv} \subseteq \overset{\mathcal{B}}{\equiv}$.

2. $\overset{\mathcal{B}'}{\equiv}$ *is a decreasing bisimulation.*

Proof. Item 1 is an inference directly from Fig. 2. Item 2 will be discussed in detail in Section 5. □

4.5 Operations on Long Strings

In the algorithm, we meet quite a few operations on strings whose length is exponential. Thus we need an efficient way to represent and manipulate them. This sort of improvement actually appears in all the previous work on strong bisimilarity checking on normed BPA. There are many different ways to do so, and nothing special in our situation. Thus we only sketch the idea and provide some literature.

In the previous work [13,21,10], a long string is represented by a *straight-line program* (SLP), a context-free grammar (typically in Chomsky normal form) which generates only one word. The efficient algorithms rely on an efficient implementation of equality checking on SLP-compressed strings, which is typically implemented (as a special case) by an efficient algorithm of compressed pattern matching such as [27,22]. Lohrey [23] gives a nice survey on algorithms on SLP-compressed strings.

One deficiency of the above scheme is that the procedure for string equality checking is called every time two strings need to compare, and previous computations are completely ignored. In [25] and its improved version [1], a data structure for finite set of strings is maintained, which supports concatenation, splitting, and equality checking operations. Czerwiński [7] uses this technique to improve his previous algorithm [10].

4.6 Analysis of Time Complexity

Now we give a very brief discussion of the time complexity of the whole algorithm. Some less important factors are deliberately neglected. Readers are referred to Czerwiński [7].

Consider the algorithm described in Fig. 1. The dominating factor is the operation $\texttt{lpftest}_{(\mathbf{P}',\mathbf{E}')}(X_i, X_j)$ at line 10. We claim that there are totally $\mathcal{O}(n^2)$ invocations of `lpftest`.

In the implementation of `lpftest`, we call the procedures described in Fig. 2. The procedure treats processes as *normed strings*. Therefore, the time consumed depends on the costs of the operations on normed strings. We suppose that there are three operations of 'normed' strings: $\texttt{Concatenate}(\sigma_1, \sigma_2)$, $\texttt{Split}(\sigma, h)$, and $\texttt{Equal}(\sigma_1, \sigma_2)$, which are supposed to spend time $\mathsf{C}(N)$, $\mathsf{S}(N)$, and $\mathsf{E}(N)$, respectively. Claimed in [7], the best implementation is $\mathsf{C}(N) = \mathcal{O}(N \cdot \text{polylog}N)$, $\mathsf{S}(N) = \mathcal{O}(N \cdot \text{polylog}N)$, and $\mathsf{E}(N) = \mathcal{O}(\text{polylog}N)$.

Consider the procedures in Fig. 2. The most time-consuming part is still the part of matching, which can perform $\mathcal{O}(N^2)$ times of `Equal` operations. This makes the total time of checking branching bisimilarity no difference from checking strong bisimilarity. The overall running time is $\mathcal{O}(N^4 \cdot \text{polylog}N)$.

5 The Refinement Operation

Now, we start to discuss the correctness of the algorithm. In order to prove the correctness, we need to answer two questions:

1. What is the refinement operation corresponding to a step of iteration in our algorithm?
2. How our algorithm can be derived from the refinement operation.

In this section we answer the first question, and the second question will be answered in Section 6.

Actually how to define the refinement operation for our algorithm is really not clear at the first glance. Thus we review the refinement operation adopted in CL algorithm in Section 5.1. Then in Section 5.1 we find another way to define and understand the refinement operation in Section 5.2. Following this understanding, we attempt to define the refinement operation which turns out to be suitable for our algorithm in Section 5.3, and then show some basic properties.

5.1 The Refinement Operation for Realtime nBPA

Before going into the tricky part of our definition of the refinement relation, let us review the reason why the algorithm is correct for the realtime nBPA. This special case is comparatively easy. For convenience, we describe the procedure of checking $X_i \overset{\mathcal{B}'}{\equiv} \delta$ for realtime nBPA in Fig. 3. This is nothing but a special case of Fig. 2, and it is a slightly simplified version of the corresponding procedure in CL algorithm.

At first we review the framework of the correctness proof for CL algorithm.

In the case of bisimilarity for realtime nBPA, we can define the following well-known *expansion* relation directly from the definition of bisimulation.

Definition 4. *Let $\mathcal{R}$ be a binary relation on* ***realtime*** *processes. The* expansion *of $\mathcal{R}$, $\mathsf{Exp}(\mathcal{R})$, contains all pairs (α, β) satisfying the following conditions:*

1. *Whenever $\alpha \xrightarrow{a} \alpha'$, then $\beta \xrightarrow{a} \beta'$ and $\alpha' \mathcal{R} \beta'$ for some β'.*
2. *Whenever $\beta \xrightarrow{a} \beta'$, then $\alpha \xrightarrow{a} \alpha'$ and $\alpha' \mathcal{R} \beta'$ for some α'.*

For realtime system, a relation $\mathcal{R}$ is a bisimulation if and only if $\mathcal{R} \subseteq \mathsf{Exp}(\mathcal{R})$. Bisimilarity $\simeq$ is the largest relation $\mathcal{R}$ which satisfies $\mathcal{R} = \mathsf{Exp}(\mathcal{R})$.

Definition 4 is well-behaved in the sense that $\mathsf{Exp}(\equiv) \cap \equiv \subsetneq \equiv$ if $\equiv$ is not a bisimulation, and $\mathsf{Exp}(\equiv)$ is a norm-preserving congruence suppose that $\equiv$ is. However, we cannot simply define the refinement relation $\mathsf{Ref}(\equiv)$ to be $\mathsf{Exp}(\equiv) \cap \equiv$, because $\mathsf{Exp}(\equiv) \cap \equiv$ may not be a decompositional congruence even if $\equiv$ is. In other words, we cannot always find a $\mathcal{B}'$ such that $\overset{\mathcal{B}'}{\equiv} = \mathsf{Exp}(\overset{\mathcal{B}}{\equiv}) \cap \overset{\mathcal{B}}{\equiv}$. The way to solve this problem is to find a decompositional congruence $\overset{\mathcal{B}'}{\equiv} = \mathsf{Ref}(\overset{\mathcal{B}}{\equiv})$ which lies between $\simeq$ and $\mathsf{Exp}(\overset{\mathcal{B}}{\equiv}) \cap \overset{\mathcal{B}}{\equiv}$. The way suggested in [14] is that $\mathsf{Ref}(\equiv)$ be the decreasing bisimilarity wrt. $\mathsf{Exp}(\equiv) \cap \equiv$.

Checking $X_i \overset{\mathcal{B}'}{\equiv} \delta$ in the case of REALTIME systems:

1. **test** $\mathtt{dcmp}_{\mathcal{B}}(X_i) = \mathtt{dcmp}_{\mathcal{B}}(\delta)$.
   If so, goto step 2; else **reject** (X_i, δ).
2. **test** for every $X_i \xrightarrow{\ell}_{\mathrm{dec}} \alpha$, we have
   $\delta \xrightarrow{\ell}_{\mathrm{dec}} \beta$ for some β and $\mathtt{dcmp}_{\mathcal{B}'}(\alpha) = \mathtt{dcmp}_{\mathcal{B}'}(\beta)$.
   If so, goto step 3; else, **reject** (X_i, δ).
3. **test** for every $X_i \xrightarrow{\ell}_{\mathrm{inc}} \alpha$, we have
   $\delta \xrightarrow{\ell}_{\mathrm{inc}} \beta$ for some β and $\mathtt{dcmp}_{\mathcal{B}}(\alpha) = \mathtt{dcmp}_{\mathcal{B}}(\beta)$.
   If so, goto step 4; else, **reject** (X_i, δ).
4. **test** for every $\delta \xrightarrow{\ell}_{\mathrm{dec}} \beta$, we have
   $X_i \xrightarrow{\ell}_{\mathrm{dec}} \alpha$ for some α such that $\mathtt{dcmp}_{\mathcal{B}'}(\alpha) = \mathtt{dcmp}_{\mathcal{B}'}(\beta)$.
   If so, goto step 5; else, **reject** (X_i, δ).
5. **test** for every $\delta \xrightarrow{\ell}_{\mathrm{inc}} \beta$, we have
   $X_i \xrightarrow{\ell}_{\mathrm{inc}} \alpha$ for some α such that $\mathtt{dcmp}_{\mathcal{B}}(\alpha) = \mathtt{dcmp}_{\mathcal{B}}(\beta)$.
   If so, goto step 6; else, **reject** (X_i, δ).
6. **accept** (X_i, δ).

Fig. 3. Checking $X_i \overset{\mathcal{B}'}{\equiv} \delta$ for Realtime Systems

Definition 5. *Let $\mathcal{R}$ be a relation on **realtime** processes. $\mathcal{R}$ is a* decreasing bisimulation *if the following hold whenever $\alpha\mathcal{R}\beta$:*

1. *Whenever $\alpha \xrightarrow{\ell}_{\mathrm{dec}} \alpha'$, then $\beta \xrightarrow{\ell}_{\mathrm{dec}} \beta'$ such that $\alpha'\mathcal{R}\beta'$.*
2. *Whenever $\beta \xrightarrow{\ell}_{\mathrm{dec}} \beta'$, then $\alpha \xrightarrow{\ell}_{\mathrm{dec}} \alpha'$ such that $\alpha'\mathcal{R}\beta'$.*

Let $\equiv$ be a norm-preserving congruence. The decreasing bisimilarity wrt. $\equiv$*, denoted by $\simeq_{\mathrm{dec}}^{\equiv}$, is the largest decreasing bisimulation contained in $\equiv$.*

We do not justify the rationality of the relation $\simeq_{\mathrm{dec}}^{\equiv}$. The fact is that $\simeq_{\mathrm{dec}}^{\equiv}$ is a congruence, and moreover, it satisfies the following:

1. $\simeq_{\mathrm{dec}}^{\equiv}$ is decompositional if $\equiv$ is right-cancellative.
2. $\mathsf{Exp}(\equiv)$ and also $\mathsf{Exp}(\equiv) \cap \equiv$ is right-cancellative if $\equiv$ is decompositional.

According to these two facts, $\simeq_{\mathrm{dec}}^{\mathsf{Exp}(\equiv)\cap\equiv}$ is decompositional whenever $\equiv$ is. From here, we can define $\mathsf{Ref}(\equiv)$ to be $\simeq_{\mathrm{dec}}^{\mathsf{Exp}(\equiv)\cap\equiv}$.

In order to get a characterization of $\simeq_{\mathrm{dec}}^{\mathsf{Exp}(\equiv)\cap\equiv}$, we need the following expansion relation for decreasing bisimilarity.

Definition 6. *Let $\mathcal{R}$ be a binary relation on **realtime** processes. The* decreasing expansion *of $\mathcal{R}$, $\mathsf{Exp}_{\mathrm{dec}}(\mathcal{R})$, contains all pairs (α, β) satisfying the following conditions:*

1. *Whenever* $\alpha \xrightarrow{\ell}_{\mathrm{dec}} \alpha'$, *then* $\beta \xrightarrow{\ell}_{\mathrm{dec}} \beta'$ *and* $\alpha' \mathcal{R} \beta'$ *for some* β'.
2. *Whenever* $\beta \xrightarrow{\ell}_{\mathrm{dec}} \beta'$, *then* $\alpha \xrightarrow{\ell}_{\mathrm{dec}} \alpha'$ *and* $\alpha' \mathcal{R} \beta'$ *for some* α'.

Then we can establish the following important property for realtime nBPA: $(\alpha, \beta) \in \simeq_{\mathrm{dec}}^{\equiv}$ if and only if

$$\alpha \equiv \beta \text{ and } (\alpha, \beta) \in \mathsf{Exp}_{\mathrm{dec}}(\simeq_{\mathrm{dec}}^{\equiv}).$$

From this fact, considering that $\mathsf{Ref}(\equiv) = \simeq_{\mathrm{dec}}^{\mathsf{Exp}(\equiv) \cap \equiv}$, we have: $(\alpha, \beta) \in \mathsf{Ref}(\equiv)$ if and only if

$$\alpha \equiv \beta \text{ and } (\alpha, \beta) \in \mathsf{Exp}(\equiv) \text{ and } (\alpha, \beta) \in \mathsf{Exp}_{\mathrm{dec}}(\mathsf{Ref}(\equiv)).$$

Now, to prove $\overset{\mathcal{B}'}{\equiv} = \mathsf{Ref}(\overset{\mathcal{B}}{\equiv})$, it suffices to prove:

$$\alpha \overset{\mathcal{B}'}{\equiv} \beta \text{ if and only if } \alpha \overset{\mathcal{B}}{\equiv} \beta \text{ and } (\alpha, \beta) \in \mathsf{Exp}(\overset{\mathcal{B}}{\equiv}) \text{ and } (\alpha, \beta) \in \mathsf{Exp}_{\mathrm{dec}}(\overset{\mathcal{B}'}{\equiv}).$$

According to this characterization, apparently we have $\overset{\mathcal{B}'}{\equiv} \subseteq \overset{\mathcal{B}}{\equiv}$.

Now it is time to explain that the procedure in Fig. 3 is actually based on this characterization. Suppose we want to check whether $X_i \overset{\mathcal{B}'}{\equiv} \delta$. It suffices to check the following three conditions:

1. $X_i \overset{\mathcal{B}}{\equiv} \delta$.
2. $(X_i, \delta) \in \mathsf{Exp}_{\mathrm{dec}}(\overset{\mathcal{B}'}{\equiv})$.
3. $(X_i, \delta) \in \mathsf{Exp}(\overset{\mathcal{B}}{\equiv})$.

Notice that these three conditions are deliberately arranged in the above order. Now we study the procedure described in Fig. 3. Step 1 corresponds to Condition 1: checking $X_i \overset{\mathcal{B}}{\equiv} \delta$. Step 2 and Step 4 correspond to Condition 2: checking $(X_i, \delta) \in \mathsf{Exp}_{\mathrm{dec}}(\overset{\mathcal{B}'}{\equiv})$. Step 3 and Step 5 partly correspond to Condition 3: checking $(X_i, \delta) \in \mathsf{Exp}(\overset{\mathcal{B}}{\equiv})$. In Step 3 and Step 5, we find that only increasing transitions are treated. This is because the decreasing transitions are already treated in Step 2 and Step 4, in which stricter requirements are tested, considering $\overset{\mathcal{B}'}{\equiv} \subseteq \overset{\mathcal{B}}{\equiv}$.

5.2 Another Understanding of the Refinement Operation

The characterization of the refinement operation defined in Section 5.1 is fine. However, currently we do not know how to generalize this characterization to non-realtime systems. The main problem is that we cannot find a feasible way to define the expansion relation. This is because the technique of dynamic programming is used in the algorithm. This makes the expansion of $\overset{\mathcal{B}}{\equiv}$, if there is a way to define, not only depend on $\overset{\mathcal{B}}{\equiv}$, but also depend on $\overset{\mathcal{B}'}{\equiv}$. This fact makes

it very difficult to generalize the correctness proof in the way taken in CL algorithm. Thus we hope to find another better way to prove the correctness of our algorithm.

Before doing this in non-realtime systems, the attempt is first made in realtime systems. That is, we develop another characterization of the refinement operation for the procedure in Fig. 3.

The basic idea is to integrate the three parts into a whole concept, which we called *decreasing bisimilarity with expansion*.

To avoid confusion, readers are suggested to forget the terminologies and notations taken in Section 5.1, because the forms of the following terminologies and notations can be close to the ones in Section 5.1, but their meanings are different.

We do not provide proofs for the lemmas and theorems below, because they are special cases for those in Section 5.3.

Definition 7. *Let $\equiv$ be a norm-preserving congruence on* ***realtime*** *processes, and let $\mathcal{R} \subseteq \equiv$ be a relation on* ***realtime*** *processes. We say $\mathcal{R}$ is a* decreasing bisimulation with expansion of $\equiv$ *if the following conditions hold whenever $\alpha \mathcal{R} \beta$:*

1. *Whenever $\alpha \xrightarrow{\ell} \alpha'$,*
   (a) *if $\alpha \xrightarrow{\ell}_{\mathrm{dec}} \alpha'$, then $\beta \xrightarrow{\ell}_{\mathrm{dec}} \beta'$ for some β' such that $\alpha' \mathcal{R} \beta'$;*
   (b) *if $\alpha \xrightarrow{\ell}_{\mathrm{inc}} \alpha'$, then $\beta \xrightarrow{\ell}_{\mathrm{inc}} \beta'$ for some β' such that $\alpha' \equiv \beta'$.*
2. *Whenever $\beta \xrightarrow{\ell} \beta'$,*
   (a) *if $\beta \xrightarrow{\ell}_{\mathrm{dec}} \beta'$, then $\alpha \xrightarrow{\ell}_{\mathrm{dec}} \alpha'$ for some α' such that $\alpha' \mathcal{R} \beta'$.*
   (b) *if $\beta \xrightarrow{\ell}_{\mathrm{inc}} \beta'$, then $\alpha \xrightarrow{\ell}_{\mathrm{inc}} \alpha'$ for some α' such that $\alpha' \equiv \beta'$.*

The decreasing bisimilarity with expansion of $\equiv$*, denoted by $\simeq^{\equiv}$, is the largest decreasing bisimulation with expansion of $\equiv$.*

The following lemma confirms the validity of Definition 7.

Lemma 15. *The following properties hold:*

1. *The identity relation is a decreasing bisimulation with expansion of $\equiv$.*
2. *Let $\mathcal{R}$ be a decreasing bisimulation with expansion of $\equiv$. Then, $\mathcal{R}^{-1}$ is also a decreasing bisimulation with expansion of $\equiv$.*
3. *Let $\mathcal{R}_1$ and $\mathcal{R}_2$ be two decreasing bisimulation with expansion of $\equiv$. Then, $\mathcal{R}_1 \circ \mathcal{R}_2$ is also a decreasing bisimulation with expansion of $\equiv$.*
4. *Let $\{\mathcal{R}_\lambda\}_{\lambda \in I}$ be a set of decreasing bisimulation with expansion of $\equiv$. Then, $\bigcup_{\lambda \in I} \mathcal{R}_\lambda$ is a decreasing bisimulation with expansion of $\equiv$.*

According to Lemma 15, $\simeq^{\equiv}$ is an equivalence relation. According to Definition 7, any decreasing bisimulation with expansion of $\equiv$ must be norm-preserving, thus $\simeq^{\equiv}$ is also norm-preserving. Moreover, we have

Lemma 16. *$\simeq^{\equiv}$ is a norm-preserving congruence.*

Now we can define $\mathsf{Ref}(\equiv) = \simeq^{\equiv}$. The validity depends on the following two properties.

Lemma 17. *1.* $\simeq^{\simeq} = \simeq$.
2. If $\simeq \subsetneq \equiv$, *then* $\simeq \subseteq \simeq^{\equiv} \subsetneq \equiv$.

The unique decomposition property of $\simeq^{\equiv}$ can be established in the same way as that of $\simeq$, but relies on the right cancellation property of $\equiv$.

Theorem 2 (Unique Decomposition Property of $\simeq^{\equiv}$). *Let $\equiv$ be a norm-preserving congruence which is right-cancellative. Then, $\simeq^{\equiv}$ is decompositional.*

It is not hard to establish the following characterization theorem of $\simeq^{\equiv}$.

Theorem 3. *Let α, β be realtime nBPA processes. Then, $\alpha \simeq^{\equiv} \beta$ if and only if $\alpha \equiv \beta$ and*

1. *Whenever $\alpha \xrightarrow{\ell} \alpha'$,*
   (a) *if $\alpha \xrightarrow{\ell}_{\mathrm{dec}} \alpha'$, then $\beta \xrightarrow{\ell}_{\mathrm{dec}} \beta'$ for some β' such that $\alpha' \simeq^{\equiv} \beta'$;*
   (b) *if $\alpha \xrightarrow{\ell}_{\mathrm{inc}} \alpha'$, then $\beta \xrightarrow{\ell}_{\mathrm{inc}} \beta'$ for some β' such that $\alpha' \equiv \beta'$.*
2. *Whenever $\beta \xrightarrow{\ell} \beta'$,*
   (a) *if $\beta \xrightarrow{\ell}_{\mathrm{dec}} \beta'$, then $\alpha \xrightarrow{\ell}_{\mathrm{dec}} \alpha'$ for some α' such that $\alpha' \simeq^{\equiv} \beta'$.*
   (b) *if $\beta \xrightarrow{\ell}_{\mathrm{inc}} \beta'$, then $\alpha \xrightarrow{\ell}_{\mathrm{inc}} \alpha'$ for some α' such that $\alpha' \equiv \beta'$.*

When $\mathsf{Ref}(\equiv)$ is defined as $\simeq^{\equiv}$, namely $\overset{\mathcal{B}'}{\equiv} = \simeq^{\overset{\mathcal{B}}{\equiv}}$, using Theorem 3, we can get exactly the procedure of checking $X_i \overset{\mathcal{B}'}{\equiv} \delta$ in Fig. 3.

It should be stressed that $\simeq^{\equiv}$ defined in Definition 7 is exact the same as according to $\simeq_{\mathrm{dec}}^{\mathsf{Exp}(\equiv)\cap\equiv}$ in Section 5.1. They are two different understandings of the same refinement operation.

5.3 The Refinement Operation for Non-realtime Systems

We spend a lot of space to discuss the correctness of the algorithm for realtime processes. The reason is that we want to generalize the way to show the correctness of our algorithm of checking branching bisimilarity for totally normed BPA. It turns out that the classical proof for CL algorithm cannot be generalized directly. So we find another characterization of the refinement operation in Section 5.2. It turns out that this one, as expected, can be used to show the correctness of our algorithm described in Fig. 2. In this section we discuss the refinement operation in detail.

We start from the notion of *decreasing bisimilarity with expansion*.

Definition 8. *Let $\equiv$ be a norm-preserving congruence on processes, and let $\mathcal{R} \subseteq \equiv$ be a relation on processes. We say $\mathcal{R}$ is a* decreasing bisimulation with expansion of $\equiv$ *if the following conditions hold whenever $\alpha\mathcal{R}\beta$:*

1. *Whenever $\alpha \xrightarrow{\ell} \alpha'$, then*

(a) *either* $\ell = \tau$ *and* $\beta \xrightarrow{\tau}_{\mathrm{dec}} \beta^1 \xrightarrow{\tau}_{\mathrm{dec}} \cdots \xrightarrow{\tau}_{\mathrm{dec}} \beta^i \xrightarrow{\tau}_{\mathrm{dec}} \cdots \xrightarrow{\tau}_{\mathrm{dec}} \beta^m$ *for some* $m \geq 0$ *and* $\beta^1, \ldots, \beta^m$ *such that* $\alpha' \mathcal{R} \beta^m$ *and* $\alpha \mathcal{R} \beta^k$ *for every* $1 \leq i \leq m$*;*

(b) *or* $\beta \xrightarrow{\tau}_{\mathrm{dec}} \beta^1 \xrightarrow{\tau}_{\mathrm{dec}} \cdots \xrightarrow{\tau}_{\mathrm{dec}} \beta^i \xrightarrow{\tau}_{\mathrm{dec}} \cdots \xrightarrow{\tau}_{\mathrm{dec}} \beta^m \xrightarrow{\ell}_{\mathrm{dec}} \beta'$ *for some* $m \geq 0$ *and* $\beta^1, \ldots, \beta^m$ *and* β' *such that* $\alpha' \mathcal{R} \beta'$ *and* $\alpha \mathcal{R} \beta^k$ *for every* $1 \leq i \leq m$*.*

(c) *or* $\beta \xrightarrow{\tau}_{\mathrm{dec}} \beta^1 \xrightarrow{\tau}_{\mathrm{dec}} \cdots \xrightarrow{\tau}_{\mathrm{dec}} \beta^i \xrightarrow{\tau}_{\mathrm{dec}} \cdots \xrightarrow{\tau}_{\mathrm{dec}} \beta^m \xrightarrow{\ell}_{\mathrm{inc}} \beta'$ *for some* $m \geq 0$ *and* $\beta^1, \ldots, \beta^m$ *and* β' *such that* $\alpha' \equiv \beta'$ *and* $\alpha \mathcal{R} \beta^k$ *for every* $1 \leq i \leq m$*.*

2. *Whenever* $\beta \xrightarrow{\ell} \beta'$*, then*

(a) *either* $\ell = \tau$ *and* $\alpha \xrightarrow{\tau}_{\mathrm{dec}} \alpha^1 \xrightarrow{\tau}_{\mathrm{dec}} \cdots \xrightarrow{\tau}_{\mathrm{dec}} \alpha^i \xrightarrow{\tau}_{\mathrm{dec}} \cdots \xrightarrow{\tau}_{\mathrm{dec}} \alpha^m$ *for some* $m \geq 0$ *and* $\alpha^1, \ldots, \alpha^m$ *such that* $\alpha^m \mathcal{R} \beta'$ *and* $\alpha^k \mathcal{R} \beta$ *for every* $1 \leq i \leq m$*;*

(b) *or* $\alpha \xrightarrow{\tau}_{\mathrm{dec}} \alpha^1 \xrightarrow{\tau}_{\mathrm{dec}} \cdots \xrightarrow{\tau}_{\mathrm{dec}} \alpha^i \xrightarrow{\tau}_{\mathrm{dec}} \cdots \xrightarrow{\tau}_{\mathrm{dec}} \alpha^m \xrightarrow{\ell}_{\mathrm{dec}} \alpha'$ *for some* $m \geq 0$ *and* $\alpha^1, \ldots, \alpha^m$ *and* α' *such that* $\alpha' \mathcal{R} \beta'$ *and* $\alpha^k \mathcal{R} \beta$ *for every* $1 \leq i \leq m$*.*

(c) *or* $\alpha \xrightarrow{\tau}_{\mathrm{dec}} \alpha^1 \xrightarrow{\tau}_{\mathrm{dec}} \cdots \xrightarrow{\tau}_{\mathrm{dec}} \alpha^i \xrightarrow{\tau}_{\mathrm{dec}} \cdots \xrightarrow{\tau}_{\mathrm{dec}} \alpha^m \xrightarrow{\ell}_{\mathrm{inc}} \alpha'$ *for some* $m \geq 0$ *and* $\alpha^1, \ldots, \alpha^m$ *and* α' *such that* $\alpha' \equiv \beta'$ *and* $\alpha^k \mathcal{R} \beta$ *for every* $1 \leq i \leq m$*.*

The decreasing bisimilarity with expansion of $\equiv$*, denoted by* $\simeq^{\equiv}$*, is the largest decreasing bisimulation with expansion of* $\equiv$*.*

If a relation $\mathcal{R} \subseteq \equiv$ *satisfies the above conditions except for 1(c) and 2(c) whenever* $\alpha \,\mathcal{R}\, \beta$*, then we call* $\mathcal{R}$ *a* decreasing bisimulation*.*

Some explanation should be made on Definition 8.

Firstly, assume that $\alpha \xrightarrow{\ell} \alpha'$ for instance. We know the corresponding transition is increasing or decreasing. If the transition is increasing, the only possibility is to take the matched transitions as the item 1(c). If the transition is decreasing, there are two subcases. The item 1(a) corresponds to the situation of $\ell = \tau$ and this silent transition can be vacantly matched. The item 1(b) corresponds to the situation that either ℓ is not silent, or the silent transition must be explicitly matched. Whenever $\alpha \xrightarrow{\tau}_{\mathrm{dec}} \alpha'$, we cannot tell which one of item 1(a) or item 1(b) should be chosen. So we must test the condition 1(a), and if 1(a) does not hold then we test condition 1(b).

Secondly, when a transition $\alpha \xrightarrow{\ell} \alpha'$ is matched by β, Definition 8 takes a different style from Definition 1, the common definition of branching bisimulation. Consider the condition 1(b) for example. In this case we require that the matching sequence of β to be

$$\beta \xrightarrow{\tau}_{\mathrm{dec}} \beta^1 \xrightarrow{\tau}_{\mathrm{dec}} \cdots \xrightarrow{\tau}_{\mathrm{dec}} \beta^i \xrightarrow{\tau}_{\mathrm{dec}} \cdots \xrightarrow{\tau}_{\mathrm{dec}} \beta^m \xrightarrow{\ell}_{\mathrm{dec}} \beta'$$

such that $\alpha' \mathcal{R} \beta'$ and $\alpha \mathcal{R} \beta^i$ for every $1 \leq i \leq m$. That is , every intermediate β^i must be related to α. In Definition 1, however, we take the simplified matching sequence of β:

$$\beta \stackrel{\tau}{\Longrightarrow}_{\mathrm{dec}} \beta'' \xrightarrow{\ell}_{\mathrm{dec}} \beta'$$

such that $\alpha'\mathcal{R}\beta'$ and $\alpha\mathcal{R}\beta''$. The reason is explained as follows. In the normal definition of branching bisimulation, although we do not require $\alpha\mathcal{R}\beta^i$ for every intermediate β^i, the largest bisimulation, $\simeq$, satisfy the Computation Lemma (Lemma 1). Thus if $\mathcal{R}$ is replaced by $\simeq$, namely if $\alpha \simeq \beta$ and $\alpha \xrightarrow{\ell} \alpha'$ is matched by

$$\beta \xrightarrow{\tau}_{\mathrm{dec}} \beta^1 \xrightarrow{\tau}_{\mathrm{dec}} \cdots \xrightarrow{\tau}_{\mathrm{dec}} \beta^i \xrightarrow{\tau}_{\mathrm{dec}} \cdots \xrightarrow{\tau}_{\mathrm{dec}} \beta^m \xrightarrow{\ell}_{\mathrm{dec}} \beta'$$

such that $\alpha' \simeq \beta'$ and $\alpha \simeq \beta^m$, then we immediately have $\alpha \simeq \beta^i$ for every $1 \leq i \leq m$. But at present we cannot establish Computation Lemma for $\simeq^{\equiv}$, since this property depends on another equivalence $\equiv$, and in Definition 8 we do not impose any restrictions on $\equiv$. Thus Computation Lemma could not be established if normal style matchings are taken in Definition 8. Thus one way of defining decreasing bisimulation with expansion of $\equiv$ is to strengthen the relevant requirements. We take this style not because we need the Computation Lemma, but because we need the conditions appearing in Definition 8 to be close to the conditions checked by the algorithm.

Thirdly, the 'semi-branching' style (see Definition 3) is taken in the case of vacant matching. This is not necessary but is helpful to show the transitivity of $\simeq^{\equiv}$.

The following lemma confirms that the relation $\simeq^{\equiv}$ is well-defined.

Lemma 18. *The following properties hold:*

1. *The identity relation is a decreasing bisimulation with expansion of $\equiv$.*
2. *Let $\mathcal{R}$ be a decreasing bisimulation with expansion of $\equiv$. Then, $\mathcal{R}^{-1}$ is also a decreasing bisimulation with expansion of $\equiv$.*
3. *Let $\mathcal{R}_1$ and $\mathcal{R}_2$ be two decreasing bisimulation with expansion of $\equiv$. Then, $\mathcal{R}_1 \circ \mathcal{R}_2$ is also a decreasing bisimulation with expansion of $\equiv$.*
4. *Let $\{\mathcal{R}_\lambda\}_{\lambda\in I}$ be a set of decreasing bisimulation with expansion of $\equiv$. Then, $\bigcup_{\lambda\in I}\mathcal{R}_\lambda$ is a decreasing bisimulation with expansion of $\equiv$.*

According to Lemma 18, $\simeq^{\equiv}$ is an equivalence relation.

Since $\simeq^{\equiv}$ is a decreasing bisimulation with expansion of $\equiv$ according to Definition 8, we have

Lemma 19. $\simeq^{\equiv} \subseteq \equiv$.

According to Definition 8, any decreasing bisimulation with expansion of $\equiv$ must be norm-preserving, thus $\simeq^{\equiv}$ is also norm-preserving. Moreover, $\simeq^{\equiv}$ is a congruence.

Lemma 20. $\simeq^{\equiv}$ *is a norm-preserving congruence.*

Proof. We only show that $\simeq^{\equiv}$ is a congruence. Let

$$\mathcal{S} = \{(\alpha_1 \cdot \alpha_2, \beta_1 \cdot \beta_2) \mid \alpha_1 \simeq^{\equiv} \beta_1 \text{ and } \alpha_2 \simeq^{\equiv} \beta_2\} \cup \simeq^{\equiv}.$$

We show $\mathcal{S}$ is a decreasing bisimulation with expansion of $\equiv$. This is done by checking the conditions in Definition 8 for every $(\alpha_1 \cdot \alpha_2, \beta_1 \cdot \beta_2) \in \mathcal{S}$.

If $\alpha_1 = \epsilon = \beta_1$. This is a trivial case.

If $\alpha_1 \neq \epsilon$ and $\beta_1 \neq \epsilon$. This proof is done by case studies. We study only two cases. Other cases are similar.

- Suppose there is a transition $\alpha_1\alpha_2 \xrightarrow{\ell}_{\mathrm{dec}} \alpha_1'\alpha_2$, we shall find the matching from $\beta_1\beta_2$. Remember $\alpha_1 \simeq^{\equiv} \beta_1$, thus every transition $\alpha_1 \xrightarrow{\ell}_{\mathrm{dec}} \alpha_1'$ has a matching from β_1. Say, we have the matching:

$$\beta_1 \xrightarrow{\tau}_{\mathrm{dec}} \beta_1^1 \xrightarrow{\tau}_{\mathrm{dec}} \cdots \xrightarrow{\tau}_{\mathrm{dec}} \beta_1^i \xrightarrow{\tau}_{\mathrm{dec}} \cdots \xrightarrow{\tau}_{\mathrm{dec}} \beta_1^m \xrightarrow{\ell}_{\mathrm{dec}} \beta_1'$$

  such that $\alpha_1' \simeq^{\equiv} \beta_1'$ and $\alpha_1 \simeq^{\equiv} \beta_1^i$ for every $1 \leq i \leq m$. Then we have

$$\beta_1\beta_2 \xrightarrow{\tau}_{\mathrm{dec}} \beta_1^1\beta_2 \xrightarrow{\tau}_{\mathrm{dec}} \cdots \xrightarrow{\tau}_{\mathrm{dec}} \beta_1^i\beta_2 \xrightarrow{\tau}_{\mathrm{dec}} \cdots \xrightarrow{\tau}_{\mathrm{dec}} \beta_1^m\beta_2 \xrightarrow{\ell}_{\mathrm{dec}} \beta_1'\beta_2.$$

  According to the definition of $\mathcal{S}$ and the fact $\alpha_2 \simeq^{\equiv} \beta_2$, we have $(\alpha_1'\alpha_2, \beta_1'\beta_2) \in \mathcal{S}$, and $(\alpha_1\alpha_2, \beta_1^i\beta_2) \in \mathcal{S}$ for every $1 \leq i \leq m$.
- Suppose there is a transition $\alpha_1\alpha_2 \xrightarrow{\ell}_{\mathrm{inc}} \alpha_1'\alpha_2$, we shall find the matching from $\beta_1\beta_2$. Remember $\alpha_1 \simeq^{\equiv} \beta_1$, thus every transition $\alpha_1 \xrightarrow{\ell}_{\mathrm{inc}} \alpha_1'$ has a matching from β_1. Say, we have the matching:

$$\beta_1 \xrightarrow{\tau}_{\mathrm{dec}} \beta_1^1 \xrightarrow{\tau}_{\mathrm{dec}} \cdots \xrightarrow{\tau}_{\mathrm{dec}} \beta_1^i \xrightarrow{\tau}_{\mathrm{dec}} \cdots \xrightarrow{\tau}_{\mathrm{dec}} \beta_1^m \xrightarrow{\ell}_{\mathrm{inc}} \beta_1'$$

  such that $\alpha_1' \equiv \beta_1'$ and $\alpha_1 \simeq^{\equiv} \beta_1^i$ for every $1 \leq i \leq m$. Then we have

$$\beta_1\beta_2 \xrightarrow{\tau}_{\mathrm{dec}} \beta_1^1\beta_2 \xrightarrow{\tau}_{\mathrm{dec}} \cdots \xrightarrow{\tau}_{\mathrm{dec}} \beta_1^i\beta_2 \xrightarrow{\tau}_{\mathrm{dec}} \cdots \xrightarrow{\tau}_{\mathrm{dec}} \beta_1^m\beta_2 \xrightarrow{\ell}_{\mathrm{inc}} \beta_1'\beta_2.$$

  According to the definition of $\mathcal{S}$ and the fact $\alpha_2 \simeq^{\equiv} \beta_2$, we have $(\alpha_1\alpha_2, \beta_1^i\beta_2) \in \mathcal{S}$ for every $1 \leq i \leq m$. Knowing $\simeq^{\equiv} \subseteq \equiv$, we have $\alpha_2 \equiv \beta_2$, and by congruence of $\equiv$ we have $\alpha_1'\alpha_2 \equiv \beta_1'\beta_2$. □

Now we can define the refinement operation $\mathsf{Ref}(\equiv)$ as $\simeq^{\equiv}$. The validity of this definition depends on the following lemma.

Lemma 21. *The following two properties hold.*

1. $\simeq^{\simeq} = \simeq$.
2. *If* $\simeq \subsetneq \equiv$, *then* $\simeq \subseteq \simeq^{\equiv} \subsetneq \equiv$.

Proof. 1. At first, we show that $\simeq \subseteq \simeq^{\equiv}$ for every $\equiv \supseteq \simeq$. As a special case, we have $\simeq \subset \simeq^{\equiv}$. Assume that $\alpha \simeq \beta$, then we can check that conditions in Definition 8, taking $\mathcal{R} = \simeq$. This is a routine work, by applying the Computation Lemma (Lemma 1). To see why $\simeq^{\simeq} \subseteq \simeq$, we notice $\simeq^{\equiv} \subseteq \equiv$ (Lemma 19), and take $\equiv$ to be $\simeq$.

2. By the proof of the first item and Lemma 19, we already have $\simeq \subseteq \simeq^{\equiv} \subseteq \equiv$ whenever $\simeq \subseteq \equiv$. Now we assume further that $\simeq \subseteq \simeq^{\equiv} = \equiv$, we will show that $\simeq = \simeq^{\equiv} = \equiv$. It suffices to show $\simeq^{\equiv} = \equiv$ is a branching bisimulation (Definition 1). Because $\simeq^{\equiv}$ is a decreasing bisimulation with expansion of $\equiv$, it satisfies the conditions in Definition 8. By taking $\mathcal{R}$ to be both $\simeq^{\equiv}$ and $\equiv$, we see that bisimulation property in Definition 1 can be inferred. □

The unique decomposition property of $\simeq^{\equiv}$ can be established in the same way as that of $\simeq$, but relies on the right cancellation property of $\equiv$.

Theorem 4 (Unique Decomposition Property of $\simeq^{\equiv}$). *Let $\equiv$ be a norm-preserving congruence which is right-cancellative. Then, $\simeq^{\equiv}$ is decompositional.*

Proof. It suffices to show that to show that $\{(\alpha, \beta) : \alpha\gamma \simeq^{\equiv} \beta\gamma \text{ for some } \gamma\}$ is a decreasing branching bisimulation wrt. $\equiv$. In the proof the right cancellativity of $\equiv$ is used. Then the proof goes in the same way as in Theorem 1. □

According to Theorem 4, $\simeq^{\equiv}$ is decompositional whenever $\equiv$ is. This is the key property to define refinement operation. Now, our refinement operation $\mathsf{Ref}(\equiv)$ can be defined as $\simeq^{\equiv}$.

6 The Correctness of the Algorithm

In this section we will show that the $\mathcal{B}'$ constructed from $\mathcal{B}$ during an iteration is exactly the decomposition base of $\mathsf{Ref}(\overset{\mathcal{B}}{\equiv}) = \simeq^{\overset{\mathcal{B}}{\equiv}}$ defined in Section 5.3.

6.1 The Characterization of $\simeq^{\equiv}$

Remember in Section 5.2 we have remarked that the procedure in Fig. 3 is correct for realtime systems. At that time the proof is straightforward, because the procedure checks exactly the conditions in the characterization theorem (Theorem 3), which are exactly the conditions in Definition 7. However, this is not the case now, and there are a number of subtleties.

In the following, we will develop some terminologies, which make us easier to formulate our results. First we need an adequate notion of 'expansion' relation which is suitable for Definition 8 and close to the testing procedure. We call this notion *compound expansion.*

Definition 9. *Let $\equiv$ be a norm-preserving congruence on processes, and let $\mathcal{R} \subseteq \equiv$ be a relation on processes. The* compound expansion *wrt. $\mathcal{R}$ and $\equiv$, denoted by $\mathsf{ComExp}_{\equiv}(\mathcal{R})$, contains all pairs (α, β) which satisfy $\alpha \equiv \beta$ and the following conditions:*

1. *Whenever $\alpha \xrightarrow{\ell} \alpha'$, then either*
   (a) *$\ell = \tau$ and $\alpha'\mathcal{R}\beta$; or*
   (b) *$\beta \xrightarrow{\ell}_{\mathrm{dec}} \beta'$ and $\alpha'\mathcal{R}\beta'$ for some β'; or*
   (c) *$\beta \xrightarrow{\ell}_{\mathrm{inc}} \beta'$ and $\alpha' \equiv \beta'$ for some β'; or*
   (d) *$\beta \xrightarrow{\tau}_{\mathrm{dec}} \beta''$ and $\alpha\mathcal{R}\beta''$ for some β''.*
2. *Whenever $\beta \xrightarrow{\ell} \beta'$, then either*
   (a) *$\ell = \tau$ and $\alpha\mathcal{R}\beta'$; or*
   (b) *$\alpha \xrightarrow{\ell}_{\mathrm{dec}} \alpha'$ and $\alpha'\mathcal{R}\beta'$ for some α'; or*
   (c) *$\alpha \xrightarrow{\ell}_{\mathrm{inc}} \alpha'$ and $\alpha' \equiv \beta'$ for some α'; or*

(d) $\alpha \xrightarrow{\tau}_{\mathrm{dec}} \alpha''$ *and* $\alpha''\mathcal{R}\beta$ *for some* α''.

The correctness of Definition 9 is confirmed by the following lemmas.

Lemma 22. *If* $\mathcal{R}$ *is a decreasing bisimulation with expansion of* $\equiv$ *(see Definition 8), then* $\mathcal{R} \subseteq \mathsf{ComExp}_{\equiv}(\mathcal{R})$. *In particular,* $\simeq^{\equiv} \subseteq \mathsf{ComExp}_{\equiv}(\simeq^{\equiv})$.

Proof. This fact is an inference of Definition 9 and Definition 8. Compare the conditions in these two definitions. When $\mathcal{R}$ is a decreasing bisimulation with expansion of $\equiv$ and $\alpha\mathcal{R}\beta$, (α, β) satisfies the conditions in Definition 8. Then we can find that (α, β) also satisfies the conditions in Definition 9. □

Lemma 23. $\mathsf{ComExp}_{\equiv}(\simeq^{\equiv})$ *is a decreasing bisimulation with expansion of* $\equiv$. *In particular,* $\mathsf{ComExp}_{\equiv}(\simeq^{\equiv}) \subseteq \simeq^{\equiv}$.

Proof. At first, remember that $\mathsf{ComExp}_{\equiv}(\mathcal{R}) \subseteq \equiv$ according to Definition 9. This is the prerequisite of $\mathsf{ComExp}_{\equiv}(\mathcal{R})$ being a decreasing bisimulation with expansion of $\equiv$. This fact will be implicitly used in the remaining proof.

Let $(\alpha, \beta) \in \mathsf{ComExp}_{\equiv}(\simeq^{\equiv})$ and $\alpha \xrightarrow{\ell} \alpha'$. According to the definition of $\mathsf{ComExp}_{\equiv}$ (Definition 9), there are four cases:

1. $\ell = \tau$ and $\alpha' \simeq^{\equiv} \beta$. In this case, we have $(\alpha', \beta) \in \mathsf{ComExp}_{\equiv}(\simeq^{\equiv})$ according to Lemma 22. Thus condition 1(a) of Definition 8 holds (with $m = 0$).
2. $\beta \xrightarrow{\ell}_{\mathrm{dec}} \beta'$ and $\alpha' \simeq^{\equiv} \beta'$ for some β'. In this case, we have $(\alpha', \beta') \in \mathsf{ComExp}_{\equiv}(\simeq^{\equiv})$ according to Lemma 22. This is the special case of the condition 1(b) of Definition 8 in which $m = 0$. Thus condition 1(b) of Definition 8 holds.
3. $\beta \xrightarrow{\ell}_{\mathrm{inc}} \beta'$ and $\alpha' \equiv \beta'$ for some β'. This is the special case of the condition 1(c) of Definition 8 in which $m = 0$. Thus condition 1(c) of Definition 8 holds.
4. $\beta \xrightarrow{\tau}_{\mathrm{dec}} \beta''$ and $\alpha \simeq^{\equiv} \beta''$ for some β''. In this case, we have $(\alpha, \beta'') \in \mathsf{ComExp}_{\equiv}(\simeq^{\equiv})$ according to Lemma 22. We can now use induction hypothesis on the pair (α, β''). Note that this case can not happen forever. Finally, case 1 or case 2 or case 3 must happen.
   - If case 1 happens finally, then we have $\beta \xrightarrow{\tau}_{\mathrm{dec}} \beta_1 \xrightarrow{\tau}_{\mathrm{dec}} \ldots \xrightarrow{\tau}_{\mathrm{dec}} \beta_i \xrightarrow{\tau}_{\mathrm{dec}} \ldots \xrightarrow{\tau}_{\mathrm{dec}} \beta_m$ for some $m > 0$ and $\beta_1, \ldots, \beta_m$ such that $\alpha' \simeq^{\equiv} \beta_m$ and $(\alpha, \beta_k) \in \mathsf{ComExp}_{\equiv}(\simeq^{\equiv})$ for every $1 \leq i \leq m$. Now we also have $(\alpha', \beta_m) \in \mathsf{ComExp}_{\equiv}(\simeq^{\equiv})$ according to Lemma 22. Consequently condition 1(a) of Definition 8 in which $m > 0$ holds.
   - If case 2 happens finally, then we get $\beta \xrightarrow{\tau}_{\mathrm{dec}} \beta_1 \xrightarrow{\tau}_{\mathrm{dec}} \ldots \xrightarrow{\tau}_{\mathrm{dec}} \beta_i \xrightarrow{\tau}_{\mathrm{dec}} \ldots \xrightarrow{\tau}_{\mathrm{dec}} \beta_m \xrightarrow{\ell}_{\mathrm{dec}} \beta'$ for some $m > 0$ and $\beta_1, \ldots, \beta_m$ and β' such that $(\alpha', \beta')\mathsf{ComExp}_{\equiv}(\simeq^{\equiv})$ and $(\alpha, \beta_k) \in \mathsf{ComExp}_{\equiv}(\simeq^{\equiv})$ for every $1 \leq i \leq m$, according to Lemma 22. Consequently condition 1(b) of Definition 8 in which $m > 0$ holds.
   - If case 3 happens finally, then we get $\beta \xrightarrow{\tau}_{\mathrm{dec}} \beta_1 \xrightarrow{\tau}_{\mathrm{dec}} \ldots \xrightarrow{\tau}_{\mathrm{dec}} \beta_i \xrightarrow{\tau}_{\mathrm{dec}} \ldots \xrightarrow{\tau}_{\mathrm{dec}} \beta_m \xrightarrow{\ell}_{\mathrm{inc}} \beta'$ for some $m > 0$ and $\beta_1, \ldots, \beta_m$ and β' such that $\alpha' \equiv \beta'$ and $(\alpha, \beta_k) \in \mathsf{ComExp}_{\equiv}(\simeq^{\equiv})$ for every $1 \leq i \leq m$, according to Lemma 22. Consequently condition 1(c) of Definition 8 in which $m > 0$ holds. □

From Lemma 22 and Lemma 23, we conclude the following important characterization of $\simeq^{\equiv}$.

Theorem 5. *$\alpha \simeq^{\equiv} \beta$ if and only if $(\alpha, \beta) \in \mathsf{ComExp}_{\equiv}(\simeq^{\equiv})$.*

Remark 5. The inverse of Lemma 22 also holds. That is, If a relation $\mathcal{R}$ satisfies $\mathcal{R} \subseteq \mathsf{ComExp}_{\equiv}(\mathcal{R})$, then $\mathcal{R}$ is a decreasing bisimulation with expansion of $\equiv$. According to this fact and Theorem 5, the congruence $\simeq^{\equiv}$ is the greatest fixpoint of $\mathsf{ComExp}_{\equiv}$. Thus the congruence $\simeq^{\equiv}$ can be completely characterized via the operation $\mathsf{ComExp}_{\equiv}$.

Readers may have noticed that the conditions in Definition 9 are quite different from the conditions in Definition 8. In Definition 8 we lay stress on getting a congruence relation from a congruence relation. On the other hand, in Definition 9, the purpose is to give a characterization which makes the conditions easy to check in the algorithm. We do not need $\mathsf{ComExp}_{\equiv}(\mathcal{R})$ to satisfy a lot of favourite properties. The difference between these two definitions must be highlighted, because it does not happen in the case of realtime nBPA, and the existence of silent actions do make things difficult. However, according to Theorem 5, $\mathsf{ComExp}_{\equiv}(\simeq^{\equiv})$ is definitely a favourite congruence.

6.2 The Correctness of the Algorithm

Theorem 5 gives us a potential way to get an implementation of the refinement operation. That is, it provides a potential way to implement $\mathtt{lpftest}_{(\mathbf{P}',\mathbf{E}')}(X_i, X_j)$ at line 10.

In the following discussion, for convenience we presuppose that $\overset{\mathcal{B}'}{\equiv}$ is equal to $\simeq^{\overset{\mathcal{B}}{\equiv}}$. We will develop more properties of $\overset{\mathcal{B}'}{\equiv}$.

According to Theorem 5, checking $X_i \overset{\mathcal{B}'}{\equiv} \delta$ is equivalent to checking $(X_i, \delta) \in \mathsf{ComExp}_{\overset{\mathcal{B}}{\equiv}}(\overset{\mathcal{B}'}{\equiv})$. Note at first that $\mathsf{ComExp}_{\overset{\mathcal{B}}{\equiv}}(\overset{\mathcal{B}'}{\equiv})$ concerns relation $\mathcal{B}'$ itself, and $\mathcal{B}'$ is not completely known at the moment. Fortunately, we have the following two critical observations.

Observation 1. At the moment of testing on the pair (X_i, δ), we have already known the base $\mathcal{B}$ and a profile of $\mathcal{B}'$ whose constances with indexes less than i. Thus we can suppose that $\mathtt{dcmp}_{\mathcal{B}}(X_i)$ is known for every i such that $1 \leq i \leq n$, and $\mathtt{dcmp}_{\mathcal{B}'}(X_j)$ is known for every j such that $1 \leq j < i$. Therefore we are capable to answer whether $\alpha \overset{\mathcal{B}}{\equiv} \beta$ for any $\alpha, \beta \in \mathbf{C}^*$, and whether $\alpha \overset{\mathcal{B}'}{\equiv} \beta$ for any $\alpha, \beta \in \mathbf{C}^*_{i-1}$.

Observation 2. Whenever decreasing transitions are concerned, say $X_i \xrightarrow{\ell}_{\mathrm{dec}} \beta$, according to Lemma 4, we have $\beta \in \mathbf{C}^*_{i-1}$.

With these two observations, we can develop the efficient procedure for checking $X_i \overset{\mathcal{B}'}{\equiv} \delta$ for realtime system (remember Theorem 3).

But at present, the situation is more complicated. In the presence of silent actions, the above two observations cannot directly lead to the efficient procedure.

The consecutive silent actions do cause inconvenience. Investigate the following scenario. Assume we want to show $X_i \overset{\mathcal{B}'}{\equiv} \delta$ and let $X_i \xrightarrow{\ell} \alpha$ be a transition which is required to be matched by $\delta \xrightarrow{\tau}_{\mathrm{dec}} \beta'' \xrightarrow{\tau}_{\mathrm{dec}} \ldots \xrightarrow{\ell} \beta$ with $X_i \overset{\mathcal{B}'}{\equiv} \beta''$. In this situation we still do not know whether $X_i \overset{\mathcal{B}'}{\equiv} \beta''$ because $\mathtt{dcmp}_{\mathcal{B}'}(X_i)$ still needs computing.

To handle this difficulty, we need some other techniques. Before doing this, we notice the following critical observation:

Observation 3. According to the fact of $\delta \in \mathbf{P}'^{*}$ and Lemma 10, δ has no transition of the form $\delta \xrightarrow{\tau} \beta'$ which satisfies $\delta \overset{\mathcal{B}'}{\equiv} \beta'$.

Whenever $X_i \overset{\mathcal{B}'}{\equiv} \delta$, the above critical observation gives rise to the following lemma.

Lemma 24. *Assume $\overset{\mathcal{B}'}{\equiv} = \simeq^{\overset{\mathcal{B}}{\equiv}}$. When $X_i \overset{\mathcal{B}'}{\equiv} \delta$ and $\delta \xrightarrow{\tau}_{\mathrm{def}} \beta$, then we do **not** have $X_i \overset{\mathcal{B}'}{\equiv} \beta$.*

According to Lemma 24, we can draw the following two assertions. First, when transition $\delta \xrightarrow{\tau}_{\mathrm{def}} \beta$ is matched by X_i, the vacantly matching cannot happen. Second, when transition $X_i \xrightarrow{\ell} \alpha$ is matched by δ, the 'state-preserving' silent transitions cannot occurred.

Within these two assertions, Theorem 5 can be written as follows.

Theorem 6. *Let $\mathcal{B} = (\mathbf{P}, \mathbf{E})$ and $\mathcal{B}' = (\mathbf{P}', \mathbf{E}')$ be two decomposition bases which validate $\overset{\mathcal{B}'}{\equiv} = \simeq^{\overset{\mathcal{B}}{\equiv}}$. Assume $\delta \in \mathbf{P}'^{*}_{i-1}$, then $X_i \overset{\mathcal{B}'}{\equiv} \delta$ if and only if $X_i \overset{\mathcal{B}}{\equiv} \delta$ and the following conditions are satisfied:*

1. *Whenever $X_i \xrightarrow{\ell} \alpha$, then either*
   (a) *$\ell = \tau$ and $\alpha \overset{\mathcal{B}'}{\equiv} \delta$; or*
   (b) *$\delta \xrightarrow{\ell}_{\mathrm{dec}} \beta$ and $\alpha \overset{\mathcal{B}'}{\equiv} \beta$ for some β; or*
   (c) *$\delta \xrightarrow{\ell}_{\mathrm{inc}} \beta$ and $\alpha \overset{\mathcal{B}}{\equiv} \beta$ for some β.*
2. *Either $X_i \xrightarrow{\tau} \alpha$ and $\alpha \overset{\mathcal{B}'}{\equiv} \delta$ for some α;*
   or, whenever $\delta \xrightarrow{\ell} \beta$, either
   (a) *$X_i \xrightarrow{\ell}_{\mathrm{dec}} \alpha$ and $\alpha \overset{\mathcal{B}'}{\equiv} \beta$ for some α, or*
   (b) *$X_i \xrightarrow{\ell}_{\mathrm{inc}} \alpha$ and $\alpha \overset{\mathcal{B}}{\equiv} \beta$ for some α.*

Proof. Remember that Theorem 5 confirms that $X_i \overset{\mathcal{B}'}{\equiv} \delta$ if and only if $(X_i, \delta) \in \mathsf{ComExp}_{\equiv}(\overset{\mathcal{B}'}{\equiv})$. According to Definition 9, $X_i \overset{\mathcal{B}'}{\equiv} \delta$ if and only if $X_i \overset{\mathcal{B}}{\equiv} \delta$ and:

1. Whenever $X_i \xrightarrow{\ell} \alpha$, then either
   (a) $\ell = \tau$ and $\alpha \overset{\mathcal{B}'}{\equiv} \delta$; or

(b) $\delta \xrightarrow{\ell}_{\rm dec} \beta$ and $\alpha \overset{\mathcal{B}'}{\equiv} \beta$ for some β; or
(c) $\delta \xrightarrow{\ell}_{\rm inc} \beta$ and $\alpha' \overset{\mathcal{B}}{\equiv} \beta'$ for some β; or
(d) $\delta \xrightarrow{\tau}_{\rm dec} \beta'$ and $X_i \overset{\mathcal{B}'}{\equiv} \beta'$ for some β'.

2. Whenever $\delta \xrightarrow{\ell} \beta$, then either
(a) $\ell = \tau$ and $X_i \overset{\mathcal{B}'}{\equiv} \beta$; or
(b) $X_i \xrightarrow{\ell}_{\rm dec} \alpha$ and $\alpha \overset{\mathcal{B}'}{\equiv} \beta$ for some α; or
(c) $X_i \xrightarrow{\ell}_{\rm inc} \alpha$ and $\alpha \overset{\mathcal{B}}{\equiv} \beta$ for some α; or
(d) $X_i \xrightarrow{\tau}_{\rm dec} \alpha'$ and $\alpha' \overset{\mathcal{B}'}{\equiv} \beta$ for some α'.

Now making use of Lemma 24, we can draw the conclusion that the case 1(d) and case 2(a) cannot happen! Now the conditions above become the conditions in Theorem 6. □

Comparing with Theorem 5, Theorem 6 has a great advantage. When we need to determine whether $X_i \overset{\mathcal{B}'}{\equiv} \delta$ or not, according to Theorem 6, we only require to checking several conditions which depends only on $\mathcal{B}$ and the profile of $\mathcal{B}'$ in which only constants with index less than i are involved. Thus we can use this fact to construct $\mathcal{B}'$ in the 'bottom-up' way, which is exactly the procedure described in Fig. 2. The proof of correctness of the algorithm is now finished.

7 Remark

7.1 Other Bisimilarities On Totally Normed BPA

Comparing with branching bisimilarity, other bisimilarities tend to be more flexible so that they are currently known to be NP-hard on tnBPA. On the occasion of weak bisimilarity, there are two different problems deserving to consideration. First, it is no longer decompositional, as is shown in Example 2. Second, it is capable to encode NP-complete problem due to its more flexible matching style.

There is a variant of weak bisimilarity called delay bisimilarity, which is still decompositional on tnBPA. Using unique decomposition property, we can confirm that delay bisimilarity is in PSPACE. The way is barely to guess a decomposition base $\mathcal{B} = (\mathbf{P}, \mathbf{E})$ and check that $\overset{\mathcal{B}}{\equiv}$ a delay bisimulation. Still, the bisimulation property needs to carefully defined. Anyway, it is technically much easier than checking branching bisimilarity.

Finally we conjecture that deciding bisimilarities other than branching bisimilarity on tnBPA is PSPACE complete.

7.2 On Branching Bisimilarity Checking

In the situation that silent transitions are treated unobservable, branching bisimilarity arouses interest of researchers. In most of the cases, previous decidability and complexity results for weak bisimilarity still hold for branching bisimilarity.

There are two remarkable exceptions. The decidability of branching bisimilarity is established by Czerwiński, Hofman and Lasota [8] on normed BPP, and by Fu [11] on normed BPA. In these two cases, decidability of weak bisimilarity is unknown. Recently, we have proven that branching (and weak) bisimilarity is undecidable on every model above BPA and BPP in the PRS hierarchy even in the normed case [32]. It is believed that branching bisimilarity is easier to decide than weak bisimilarity. Currently, there is no real instance to support this belief. This paper provides an interesting instance. We expect that more instances will be discovered in the future.

Acknowledgement. The author would like to thank Sławomir Lasota for letting me know the work of Czerwiński [7], the current fastest algorithm for checking strong bisimilarity on normed BPA; to thank the members of BASICS for their helpful discussions on related topics.

References

1. Stephen Alstrup, Gerth Stølting Brodal, and Theis Rauhe. Pattern matching in dynamic texts. In *SODA*, pages 819–828, 2000.
2. J. C. M. Baeten and W. P. Weijland. *Process Algebra*. Cambridge University Press, New York, NY, USA, 1990.
3. Jos C. M. Baeten, Jan A. Bergstra, and Jan Willem Klop. Decidability of bisimulation equivalence for processes generating context-free languages. In *PARLE (2)*, pages 94–111, 1987.
4. Jos C. M. Baeten, Jan A. Bergstra, and Jan Willem Klop. Decidability of bisimulation equivalence for processes generating context-free languages. *J. ACM*, 40(3):653–682, 1993.
5. Twan Basten. Branching bisimilarity is an equivalence indeed! *Inf. Process. Lett.*, 58(3):141–147, 1996.
6. Olaf Burkart, Didier Caucal, Faron Moller, and Bernhard Steffen. Verification on infinite structures, 2000.
7. Wojciech Czerwinski. *Partially-commutative context-free graphs*. PhD thesis, University of Warsaw, 2012.
8. Wojciech Czerwinski, Piotr Hofman, and Slawomir Lasota. Decidability of branching bisimulation on normed commutative context-free processes. In *CONCUR*, pages 528–542, 2011.
9. Wojciech Czerwinski and Petr Jancar. Branching bisimilarity of normed BPA processes is in NEXPTIME. *CoRR*, abs/1407.0645, 2014.
10. Wojciech Czerwinski and Slawomir Lasota. Fast equivalence-checking for normed context-free processes. In *FSTTCS*, pages 260–271, 2010.
11. Yuxi Fu. Checking equality and regularity for normed BPA with silent moves. In *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part II*, pages 238–249, 2013.
12. Yoram Hirshfeld. Bisimulation trees and the decidability of weak bisimulations. *Electr. Notes Theor. Comput. Sci.*, 5:2–13, 1996.
13. Yoram Hirshfeld, Mark Jerrum, and Faron Moller. A polynomial algorithm for deciding bisimilarity of normed context-free processes. *Theor. Comput. Sci.*, 158(1&2):143–159, 1996.

14. Yoram Hirshfeld, Mark Jerrum, and Faron Moller. A polynomial-time algorithm for deciding bisimulation equivalence of normed basic parallel processes. *Mathematical Structures in Computer Science*, 6(3):251–259, 1996.
15. John E. Hopcroft and Jeffrey D. Ullman. *Introduction To Automata Theory, Languages, And Computation*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1990.
16. Hans Hüttel. Silence is golden: Branching bisimilarity is decidable for context-free processes. In *CAV*, pages 2–12, 1991.
17. Hans Hüttel and Colin Stirling. Actions speak louder than words: Proving bisimilarity for context-free processes. In *LICS*, pages 376–386, 1991.
18. Dung T. Huynh and Lu Tian. Deciding bisimilarity of normed context-free processes is in σ_2^p. *Theor. Comput. Sci.*, 123(2):183–197, 1994.
19. Petr Jančar and Faron Moller. Techniques for decidability and undecidability of bisimilarity. In *CONCUR*, pages 30–45, 1999.
20. Antonín Kučera and Petr Jančar. Equivalence-checking on infinite-state systems: Techniques and results. *Theory and Practice of Logic Programming*, 6(201), 2006.
21. Slawomir Lasota and Wojciech Rytter. Faster algorithm for bisimulation equivalence of normed context-free processes. In *MFCS*, pages 646–657, 2006.
22. Yury Lifshits. Solving classical string problems an compressed texts. In *Combinatorial and Algorithmic Foundations of Pattern and Association Discovery*, 2006.
23. Markus Lohrey. Algorithmics on slp-compressed strings: A survey. *Groups Complexity Cryptology*, 4(2):241–299, 2012.
24. Richard Mayr. Weak bisimilarity and regularity of context-free processes is EXPTIME-hard. *Theoretical Computer Science*, 330(3):553–575, February 2005.
25. Kurt Mehlhorn, R. Sundar, and Christian Uhrig. Maintaining dynamic sequences under equality tests in polylogarithmic time. *Algorithmica*, 17(2):183–198, 1997.
26. Robin Milner. *Communication and concurrency*. PHI Series in computer science. Prentice Hall, 1989.
27. Masamichi Miyazaki, Ayumi Shinohara, and Masayuki Takeda. An improved pattern matching algorithm for strings in terms of straight-line programs. In *CPM*, pages 1–11, 1997.
28. Faron Moller, Scott A. Smolka, and Jirí Srba. On the computational complexity of bisimulation, redux. *Inf. Comput.*, 194(2):129–143, 2004.
29. Jiří Srba. Roadmap of infinite results. *Current Trends In Theoretical Computer Science*, 2(201), 2004.
30. Jitka Stříbrná. Hardness results for weak bisimilarity of simple process algebras. *Electr. Notes Theor. Comput. Sci.*, 18:179–190, 1998.
31. Rob J. van Glabbeek and W. P. Weijland. Branching time and abstraction in bisimulation semantics. *J. ACM*, 43(3):555–600, 1996.
32. Qiang Yin, Yuxi Fu, Chaodong He, Mingzhang Huang, and Xiuting Tao. Branching bisimilarity checking for prs. In *ICALP (2)*, pages 363–374, 2014.