Checking Equality and Regularity for Normed BPA with Silent Moves

Yuxi Fu

BASICS, Department of Computer Science
Shanghai Jiaotong University, Shanghai 200030, China

17 February, 2013

Abstract

The decidability of weak bisimilarity on normed BPA is a long standing open problem. It is proved in this paper that branching bisimilarity, a standard refinement of weak bisimilarity, is decidable for normed BPA and that the associated regularity problem is also decidable.

Contents

1 Introduction

In [BBK87, BBK93] Baeten, Bergstra and Klop proved a surprising result that strong bisimilarity between context free grammars without empty productions is decidable. The proof exploits periodic tree structure of process graphs associated to Greibach normal forms. The decidability is in sharp contrast to the well known fact that language equivalence between these grammars is undecidable [HU79]. Seeing from another perspective, Baeten, Bergstra and Klop's work has extended the decidability result of bisimulation equivalence from finite state systems [Mil84, Mil89b, vG93] to infinite state systems. After [BBK87] decidability and complexity issues of equivalence checking of infinite systems *à la* process algebra have been intensively investigated. See [JM99, BCMS01, Srb04, MSS04, KJ06, JS08] for a number of surveys. As regards BPA, Hüttel and Stirling [HS91] improved Baeten, Bergstra and Klop's proof by a more straightforward one using tableau system. Hüttel [H9̈1] then repeated the tableau construction for branching bisimilarity on totally normed BPA processes. Later Hirshfeld [Hir96] applied the tableau method to the weak bisimilarity on the totally normed BPA. An affirmative answer to the decidability of the strong bisimilarity on general BPA is given by Christensen, Hüttel and Stirling by applying the technique of bisimulation base [CHS92, CHS95].

The complexity aspect of BPA has also been investigated over the years. By constructing an NC-reduction from a variant of the Boolean Circuit Value Problem to the strong bisimilarity problem on a finite labeled transition system, Balcazar, Gabarro and Santha [BGS92] pointed out that strong bisimilarity is P-hard, reaffirming our intuition about the sequential nature of bisimulation. Huynh and Tian [HT94] showed that the problem is in Σ_2^p, the second level of the polynomial hierarchy [Pap94]. Hirshfeld, Jerrum and Moller [HJM96] completed the picture by offering a remarkable polynomial algorithm for the strong bisimilarity of normed BPA. For the general BPA, Burkart, Caucal and Steffen [BCS95] showed that the strong bisimilarity problem is elementary. They claimed that their algorithm can be optimized to get a 2-EXPTIME upper bound. A further elaboration of the 2-EXPTIME upper bound is given in [Jan12] with the introduction of infinite regular words. The current known best lower bound of the problem, EXPTIME, is obtained by Kiefer [Kie12], improving both the PSPACE lower bound result and its proof of Srba [Srb02b]. Kiefer observed that the problem of determining the winner of a so-called hit-or-run game is EXPTIME complete and that it is reducible to the strong bisimilarity problem of BPA. An obvious challenge now is to close the gap between the EXPTIME lower bound and the 2-EXPTIME upper bound. Much less is known about the weak bisimilarity on BPA. Stříbrná's PSPACE lower bound [Stř98] is subsumed by both the result of Srba [Srb02b] and that of Mayr [May03]. Mayr reduced the acceptance problem of Alternating Linear Bound Automaton, which is known to be EXPTIME-complete, to the complement of the weak bisimilarity problem of normed BPA, showing that the latter is EXPTIME-hard. These lower bound results on BPA are subsumed by Kiefer's recent result. A slight modification of Mayr's reduction produces a reduction from the acceptance problem of Alternating Linear Bound Automaton to the complement of the branching bisimilarity problem of normed BPA. A summary of some of the afore-mentioned results is given in Figure 1, where $\sim$, $\simeq$ and $\approx$ are respectively strong bisimilarity, branching bisimilarity and weak bisimilarity.

	BPA	Normed **BPA**
$\sim$	Decidable [CHS92] 2-EXPTIME [BCS95] EXPTIME-hard [Kie12] PSPACE-hard [Srb02b]	Decidable [BBK87] Decidable [HS91] P-complete [BGS92][HJM96]
$\simeq$	? EXPTIME-hard [May03]	? EXPTIME-hard [May03]
$\approx$	? EXPTIME-hard [May03] PSPACE-hard [Stř98]	? EXPTIME-hard [May03] PSPACE-hard [Stř98]

Figure 1: Decidability of BPA

It is generally believed that weak bisimilarity, as well as branching bisimilarity, on BPA is decidable. There has been however a lack of technique to resolve the difficulties caused by silent transitions. All currently known decidability results on BPA with silent transitions are achieved by placing restrictions on the model. This paper answers affirmatively to one of the questions posed in Fig. 1. We will show that branching bisimilarity on normed BPA and its associated regularity problem are decidable. The crux of the proof is the observation that the tree consisting of the state preserving silent transitions from a normed BPA process is essentially finite. In fact it is effectively finite. The effective finite tree property allows one to approximate branching bisimilarity of normed BPA via a sequence of finite branching bisimulations. Consequently a semi-decidable procedure for the complement of branching bisimilarity of normed BPA is readily available. The effective finite tree property also suggests to investigate tableau method for branching bisimilarity of normed BPA. It turns out that normed BPA satisfies some form of cancelation property that can be used to control the size of a tableau. Based on this observation a tableau based decidable procedure can be designed to check the branching bisimilarity of two normed BPA processes. Remarkably the cancelation property also offers an interesting way of deciding if a normed BPA process is branching bisimilar to some finite state process.

The rest of the paper is organized as follows: Section 2 lays down the preliminaries. Section 3 points out that state preserving silent transitions in normed BPA enjoy finite tree property and derives that the complement of branching bisimilarity on normed BPA processes is semi-decidable. Building upon the results obtained in Section 3, Section 4 defines tableau method for branching bisimilarity on normed BPA processes and proves that branching bisimilarity on normed BPA processes is decidable. Section 5 demonstrates that the regularity problem for normed BPA processes is decidable by exploiting the decidability property of the branching bisimilarity. Section 6 comments on some future research issues.

2 Branching Bisimilarity for BPA

In this section we fix the terminologies and notations for BPA and introduces the technical preliminaries necessary in the rest of the paper.

2.1 Basic Process Algebra

A *basic process algebra* (BPA for short) Γ is a triple $(\mathcal{V}, \mathcal{A}, \Delta)$ where $\mathcal{V} = \{X_1, \ldots . X_n\}$ is a finite set of *variables*, $\mathcal{A} = \{a_1, \ldots . a_m\} \cup \{\tau\}$ is a finite set of *actions* ranged over by ℓ, and Δ is a finite set of *transition rules*. The special symbol τ denotes a *silent* action. A *BPA process* defined in Γ is an element of the set $\mathcal{V}^*$ of finite string of element of $\mathcal{V}$. The set $\mathcal{V}$ will be ranged over by capital letters and $\mathcal{V}^*$ by lower case Greek letters. The empty string is denoted by ϵ. We think of $\mathcal{V}^*$ as constructed from *sequential operator* '.'. So $\alpha.\beta$ is a sequential process which acts as α and invokes β only after α has become ϵ by performing a finite number of actions. This informal semantics points out that the sequential operator is associative and ϵ is the unit. The algebraic property allows us to ignore the operator completely at the syntactical level. We will use $=$ for the grammar equality on $\mathcal{V}^*$. A transition rule is of the form $X \xrightarrow{\ell} \alpha$, where ℓ ranges over $\mathcal{A}$. The transitional semantics is closed under *composition* in the sense that $X\gamma \xrightarrow{\ell} \alpha\gamma$ for all γ whenever $X \xrightarrow{\ell} \alpha$. We shall assume that every variable of a BPA is defined by at least one transition rule and every action in $\mathcal{A}$ appears in some transition rule. Accordingly we sometimes refer to a BPA by its set of transition rules. We write $\longrightarrow$ for $\xrightarrow{\tau}$ and $\Longrightarrow$ for the reflexive transitive closure of $\xrightarrow{\tau}$. The set $\mathcal{A}^*$ will be ranged over by ℓ^*. If $\ell^* = \ell_1 \ldots \ell_k$ for some $k \geq 0$, then $\alpha \xrightarrow{\ell^*} \alpha'$ stands for $\alpha \xrightarrow{\ell_1} \alpha_1 \ldots \xrightarrow{\ell_{k-1}} \alpha_{k-1} \xrightarrow{\ell_k} \alpha'$ for some $\alpha_1, \ldots, \alpha_{k-1}$. We say that α' is a *descendant* of α if $\alpha \xrightarrow{\ell^*} \alpha'$ for some ℓ^*.

We use finite branching labeled trees to describe operational behaviors of processes, in which the label of a node is a BPA process and the label of an edge is an action. We let $\mathbf{l}, \mathbf{m}, \mathbf{n}$ to range over the set of nodes. We often confuse a node with its label. Suppose $\mathcal{T}, \mathcal{T}'$ are labeled trees. The notation $\mathcal{T} \subseteq \mathcal{T}'$ indicates that each path from the root of $\mathcal{T}$ is also a path from the root of $\mathcal{T}'$. If in addition $\mathcal{T}$ is finite, we write $\mathcal{T} \subseteq_f \mathcal{T}'$. A *transition tree* $\mathcal{T}^\alpha$ rooted at α is composed from the transition sequences starting from α. The subtree consists of all silent transition sequences from α, denoted by $\mathcal{T}^{\alpha,\tau}$, is called the *$\tau$-tree* of α. We may think of $\mathcal{T}^{\alpha,\tau}$ as a tree without any labels on its edges. Given a variable X in a BPA, the set $rv(X)$ of variables reachable from X consists of every variable that appears as the first variable of a node in $\mathcal{T}^X$. Similarly the set $rv^\tau(X)$ of variables reachable from X via silent transition consists of every variable that appears as the first variable of a node in $\mathcal{T}^{X,\tau}$.

A BPA process α is *normed* if there are some actions $\ell_1, \ldots \ell_j$ such that $\alpha \xrightarrow{\ell_1} \ldots \xrightarrow{\ell_j} \epsilon$. A process is *unnormed* if it is not normed. Normedness is a desirable feature for languages since generations of infinitely long words are considered unnecessary. The *norm* of a BPA process α, denoted by $\|\alpha\|$, is the least k such that $\alpha \xrightarrow{\ell_1} \ldots \xrightarrow{\ell_k} \epsilon$ for some $\ell_1, \ldots \ell_k$. A *normed BPA*, or *nBPA*, is one in which every variable is normed.

For each given BPA Δ, we introduce the following notations:

- m_Δ is the number of transition rules.
- n_Δ is the number of variables, and the variables will be referred to as $X_1, \ldots, X_{n_\Delta}$.
- r_Δ is $\max \left\{ |\gamma| \,\middle|\, X \xrightarrow{\lambda} \gamma \in \Delta \right\}$, where $|\gamma|$ denotes the size, or the length of γ. We will assume that $r_\Delta > 0$ for otherwise the BPA Δ would be a submodel of the finite state CCS whose equivalence checking problem has already been resolved.
- t_Δ is $\max_{1 \le i \le n_\Delta} \{t(X_i) \mid t(X_i) \text{ is defined}\}$. The notation $t(X_i)$ stands for the length of the shortest silent transition sequence from X_i to ϵ. We take $t(X)$ as undefined if there is no silent transition sequence leading from X to ϵ. Clearly $t_\Delta > 0$.
- $||\Delta||$ is $\max \{||X_i|| \mid 1 \le i \le n_\Delta \text{ and } X_i \text{ is normed}\}$.

Each of m_Δ, n_Δ, r_Δ, t_Δ and $||\Delta||$ can be effectively calculated from Δ. For example algorithms for t_Δ and $||\Delta||$ can be designed using dynamic programming technique.

2.2 Branching Bisimilarity

The idea of the branching bisimilarity of van Glabbeek and Weijland [vGW89] is that not all silent actions can be ignored. What can be ignored are those that do not change system states irreversibly. The following definition is from [vGW89]. We write $\mathcal{R}^{-1}$ for the reverse relation of $\mathcal{R}$ and we often use infix notation $x\mathcal{R}y$ for $(x, y) \in \mathcal{R}$.

Definition 1. *A binary relation $\mathcal{R}$ on BPA processes is a van-Glabbeek-Weijland* branching bisimulation *if the following statements are valid whenever $\alpha\mathcal{R}\beta$:*

1. *If $\beta\mathcal{R}^{-1}\alpha \xrightarrow{\ell} \alpha'$ then one of the following statements is valid:*
   - *(i) $\ell = \tau$ and $\alpha'\mathcal{R}\beta$.*
   - *(ii) $\beta \Longrightarrow \beta''\mathcal{R}^{-1}\alpha$ for some β'' such that $\beta'' \xrightarrow{\ell} \beta'\mathcal{R}^{-1}\alpha'$ for some β'.*
2. *If $\alpha\mathcal{R}\beta \xrightarrow{\ell} \beta'$ then one of the following statements is valid:*
   - *(i) $\ell = \tau$ and $\alpha\mathcal{R}\beta'$.*
   - *(ii) $\alpha \Longrightarrow \alpha''\mathcal{R}\beta$ for some α'' such that $\alpha'' \xrightarrow{\ell} \alpha'\mathcal{R}\beta'$ for some α'.*

The van-Glabbeek-Weijland branching bisimilarity $\simeq_{vGW}$ *is the largest van-Glabbeek-Weijland branching bisimulation.*

For the purpose of defining an equality for BPA the above definition has to be modified, the reason being that $\simeq_{vGW}$ is not a congruence relation for sequential processes. The simplest counter example is given by the process ϵ and the process Ω defined by the BPA $(\Omega, \{\tau\}, \{\Omega \xrightarrow{\tau} \Omega\})$. One clearly has $\Omega \simeq_{vGW} \epsilon$, but $\Omega\alpha \not\simeq_{vGW} \epsilon\alpha$ for every α that may perform an external action. The modification we adopt imposes a condition easily checkable algorithmically.

Definition 2. *A binary relation $\mathcal{R}$ is a* branching bisimulation *for BPA if it is a van-Glabbeek-Weijland branching bisimulation and the following is valid whenever $\alpha\mathcal{R}\beta$:*

3. If $\alpha = \epsilon$ then $\beta \Longrightarrow \epsilon$, and if $\beta = \epsilon$ then $\alpha \Longrightarrow \epsilon$.

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

The branching bisimilarity $\simeq$ satisfies the standard properties of observational equivalence stated in the next two lemmas.

Lemma 3 (Computation Lemma)**.** *Suppose $\alpha_0 \xrightarrow{\tau} \alpha_1 \xrightarrow{\tau} \alpha_2 \xrightarrow{\tau} \dots \xrightarrow{\tau} \alpha_k \simeq \alpha_0$. Then $\alpha_0 \simeq \alpha_1 \simeq \alpha_2 \simeq \dots \simeq \alpha_k$.*

Lemma 4 (Bisimulation Lemma)**.** *Suppose $\alpha \Longrightarrow \alpha' \simeq \beta$ and $\beta \Longrightarrow \beta' \simeq \alpha$. Then $\alpha' \simeq \alpha \simeq \beta \simeq \beta'$.*

As far as we know Lemma 3 appears for the first time in [vGW89] and Lemma 4 in [DNMV90]. Using Computation Lemma one easily sees that whenever $\beta \simeq \alpha \xrightarrow{\ell} \alpha'$ is simulated by $\beta \xrightarrow{\tau} \beta_1 \xrightarrow{\tau} \beta_2 \dots \xrightarrow{\tau} \beta_k \xrightarrow{\ell} \beta'$ such that $\beta_k \simeq \alpha$ and $\beta' \simeq \alpha'$ then $\beta \simeq \beta_1 \simeq \dots \simeq \beta_k$. Using the same lemma it is easy to prove that $\simeq$ is a congruence.

Lemma 5. *The branching bisimilarity $\simeq$ is equivalent and congruent. It is the largest equivalent congruence contained in $\simeq_{vGW}$.*

Proof. Let $\mathcal{R} \subseteq \simeq_{vGW}$ be an equivalence and a congruence and let α be such that $\alpha\mathcal{R}\epsilon$. Define A by the labeled transition system $\{A \xrightarrow{a} A\}$, where a is some label that is not used by α. Now $\alpha A\mathcal{R}A$ by congruence. So $\alpha A \simeq_{vGW} A$. Hence $\alpha \Longrightarrow \epsilon$. □

Having defined an equality for BPA, we can formally draw a line between the silent actions that change the capacity of systems and those that do not. We say that a silent action $\alpha \xrightarrow{\tau} \alpha'$ is *state preserving* if $\alpha \simeq \alpha'$; it is a *change of state* if $\alpha \not\simeq \alpha'$. We will write $\alpha \to \alpha'$ if $\alpha \xrightarrow{\tau} \alpha'$ is state preserving and $\alpha \xrightarrow{\iota} \alpha'$ if it is a change of state. The reflexive and transitive closure of $\to$ is denoted by $\to^*$. Since both external actions and change-of-state silent actions must be explicitly bisimulated, we let $\jmath$ range over the set $(\mathcal{A} \setminus \{\tau\}) \cup \{\iota\}$. So $\alpha \xrightarrow{\jmath} \alpha'$ means either $\alpha \xrightarrow{a} \alpha'$ for some $a \neq \tau$ or $\alpha \xrightarrow{\iota} \alpha'$. The following lemma is an easy consequence of Computation Lemma.

Lemma 6. *If there exists a state preserving silent transition sequence from α to β, then all silent transition sequences from α to β are state preserving.*

It is time to see some BPA's and some examples of branching bisimilar processes.

Example 7. The BPA Γ_0 is defined by the following transition rules:

$$X \xrightarrow{a} \epsilon, \qquad Y \xrightarrow{b} \epsilon, \qquad Z \xrightarrow{a} V, \qquad V \xrightarrow{b} Z.$$

It is easy to see that $XYZ \simeq Z$ but $XZ \not\simeq Z \not\simeq YZ$. The variables Z, V are unnormed. We remark that the set $\{\alpha \mid \alpha Z \simeq Z\}$ is infinite but is finitely generated by $\{XY\}$. At the moment we do not know the answer to the following question: For an unnormed variable U defined in a general BPA, is the set $\{\alpha \mid \alpha U \simeq U\}$ finitely generated?

Example 8. The BPA Γ_1 is defined by the following transition rules:

$$H \xrightarrow{c} \epsilon, \quad H \xrightarrow{c} U, \quad H \xrightarrow{c} W, \quad U \xrightarrow{a} U, \quad W \xrightarrow{b} W.$$

Although $U \not\simeq W$, one has $HU \simeq HW$. In this example both U and W are unnormed.

Example 9. The BPA Γ_2 is defined by the following transition rules:

$$\begin{array}{llll} A \xrightarrow{a} A, & A \xrightarrow{\tau} \epsilon, & B \xrightarrow{b} B, & B \xrightarrow{\tau} \epsilon, \\ C \xrightarrow{a} C, & C \xrightarrow{b} C, & C \xrightarrow{\tau} \epsilon, & \\ D \xrightarrow{d} A, & E \xrightarrow{d} B. & & \end{array}$$

It is easy to see that $AC \simeq BC$ and $DC \simeq EC$, although $A \not\simeq B$ and $D \not\simeq E$. In this example all variables are normed.

Example 10. The BPA Γ_3 is defined by the following transition rules:

$$\begin{array}{llllll} I \xrightarrow{a} J, & J \xrightarrow{a} I, & I \xrightarrow{c} \epsilon, & J \xrightarrow{c} K, & K \xrightarrow{b} \epsilon, & K \xrightarrow{\tau} \epsilon, \\ L \xrightarrow{b} M, & M \xrightarrow{b} L, & L \xrightarrow{d} \epsilon, & M \xrightarrow{d} N, & N \xrightarrow{a} \epsilon, & N \xrightarrow{\tau} \epsilon, \\ Q \xrightarrow{a} Q, & Q \xrightarrow{b} Q, & Q \xrightarrow{\tau} \epsilon. & & & \end{array}$$

It is not difficult to see that $ILIQ \simeq JLIQ$ and $ILILQ \simeq JLILQ$. However $ILI \not\simeq JLI$ and $ILIL \not\simeq JLIL$.

The equalities and inequalities stated in Example 7 and Example 8 are also valid for strong bisimilarity.

2.3 Bisimulation Base

An *axiom system* $\mathcal{A}$ is a finite set of equalities on normed BPA processes. An element $\alpha = \beta$ of $\mathcal{A}$ is called an *axiom*. Write $\mathcal{A} \vdash \alpha = \beta$ if the equality $\alpha = \beta$ can be derived from the axioms of $\mathcal{A}$ by repetitive use of any of the three equivalence rules and two congruence rules. For our purpose the most useful axiom systems are those that generate branching bisimulations. These are bisimulation bases originally due to Caucal (see the survey paper [BCMS01] for more background). The following definition is Hüttel's adaptation to the branching scenario [H9̈1].

Definition 11. *A finite axiom system $\mathcal{A}$ is a* bisimulation base *if the following bisimulation base property hold for every axiom* (α_0, β_0) *of $\mathcal{A}$:*

1. *If $\beta_0 \mathcal{A}^{-1} \alpha_0 \longrightarrow \alpha_1 \longrightarrow \ldots \longrightarrow \alpha_n \xrightarrow{\ell} \alpha'$ then there are $\beta_1, \ldots, \beta_n, \beta'$ such that $\mathcal{A} \vdash \alpha_1 = \beta_1, \ldots, \mathcal{A} \vdash \alpha_n = \beta_n$, $\mathcal{A} \vdash \alpha' = \beta'$ and the following hold:*

   (i) For each i with $0 \leq i < n$, either $\beta_i = \beta_{i+1}$, or $\beta_i \longrightarrow \beta_{i+1}$, or there are $\beta_i^1, \ldots, \beta_i^{k_i}$ such that $\beta_i \longrightarrow \beta_i^1 \longrightarrow \ldots \longrightarrow \beta_i^{k_i} \longrightarrow \beta_{i+1}$ and $\mathcal{A} \vdash \alpha_i = \beta_i^1$, $\ldots$, $\mathcal{A} \vdash \alpha_i = \beta_i^{k_i}$.

(ii) *Either $\ell = \tau$ and $\beta_n = \beta'$, or $\beta_n \xrightarrow{\ell} \beta'$, or there are $\beta_n^1, \ldots, \beta_n^{k_n}$ such that $\beta_n \longrightarrow \beta_n^1 \longrightarrow \ldots \longrightarrow \beta_n^{k_n} \xrightarrow{\ell} \beta'$ and $\mathcal{A} \vdash \alpha_n = \beta_n^1, \ldots, \mathcal{A} \vdash \alpha_n = \beta_n^{k_n}$.*

2. *If $\alpha_0 \mathcal{A} \beta_0 \longrightarrow \beta_1 \longrightarrow \ldots \longrightarrow \beta_n \xrightarrow{\ell} \beta'$ then there are $\alpha_1, \ldots, \alpha_n, \alpha'$ such that $\mathcal{A} \vdash \beta_1 = \alpha_1, \ldots, \mathcal{A} \vdash \beta_n = \alpha_n$, $\mathcal{A} \vdash \beta' = \alpha'$ and the following hold:*

   (i) *For each i with $0 \leq i < n$, either $\alpha_i = \alpha_{i+1}$, or $\alpha_i \longrightarrow \alpha_{i+1}$, or there are $\alpha_i^1, \ldots, \alpha_i^{k_i}$ such that $\alpha_i \longrightarrow \alpha_i^1 \longrightarrow \ldots \longrightarrow \alpha_i^{k_i} \longrightarrow \alpha_{i+1}$ and $\mathcal{A} \vdash \beta_i = \alpha_i^1$, $\ldots$, $\mathcal{A} \vdash \beta_i = \alpha_i^{k_i}$.*

   (ii) *Either $\ell = \tau$ and $\alpha_n = \alpha'$, or $\alpha_n \xrightarrow{\ell} \alpha'$, or there are $\alpha_n^1, \ldots, \alpha_n^{k_n}$ such that $\alpha_n \longrightarrow \alpha_n^1 \longrightarrow \ldots \longrightarrow \alpha_n^{k_n} \xrightarrow{\ell} \alpha'$ and $\mathcal{A} \vdash \beta_n = \alpha_n^1, \ldots, \mathcal{A} \vdash \beta_n = \alpha_n^{k_n}$.*

3. *If $\alpha_0 = \epsilon$ then either $\beta_0 = \epsilon$ or $\beta_0 \longrightarrow \beta_1 \longrightarrow \ldots \longrightarrow \beta_k \longrightarrow \epsilon$ for some $\beta_1, \ldots, \beta_k$ with $k \geq 0$ such that $\mathcal{A} \vdash \epsilon = \beta_1, \ldots, \mathcal{A} \vdash \epsilon = \beta_k$.*

4. *If $\beta_0 = \epsilon$ then either $\alpha_0 = \epsilon$ or $\alpha_0 \longrightarrow \alpha_1 \longrightarrow \ldots \longrightarrow \alpha_k \longrightarrow \epsilon$ for some $\alpha_1, \ldots, \alpha_k$ with $k \geq 0$ such that $\mathcal{A} \vdash \alpha_1 = \epsilon, \ldots, \mathcal{A} \vdash \alpha_k = \epsilon$.*

The next lemma points out the importance of bisimulation base [Ḧ91]. It also explains the terminology 'bisimulation base'.

Lemma 12. *If $\mathcal{A}$ is a bisimulation base then $\mathcal{A}^{\vdash} = \{(\alpha, \beta) \mid \mathcal{A} \vdash \alpha = \beta\}$ is a branching bisimulation.*

Proof. We prove that $\mathcal{A}^{\vdash}$ satisfies the bisimulation base property, which implies that it is a branching bisimulation. Suppose $\mathcal{A} \vdash \alpha = \beta$. It can be easily shown by induction that there must exist

$$\gamma_1\delta_1\lambda_1, \gamma_2\delta_2\lambda_2, \gamma_3\delta_3\lambda_3, \ldots, \gamma_{k-1}\delta_{k-1}\lambda_{k-1}, \gamma_k\delta_k\lambda_k \text{ and } \delta_1', \ldots, \delta_k'$$

for $k \geq 1$ such that $\alpha = \gamma_1\delta_1\lambda_1$, $\gamma_k\delta_k'\lambda_k = \beta$ and the following hold:

$$\gamma_1\delta_1\lambda_1 \;\mathcal{A}\; \gamma_1\delta_1'\lambda_1 = \gamma_2\delta_2\lambda_2 \;\mathcal{A}\; \gamma_2\delta_2'\lambda_2 = \gamma_3\delta_3\lambda_3 \ldots \gamma_{k-1}\delta_{k-1}'\lambda_{k-1} = \gamma_k\delta_k\lambda_k \;\mathcal{A}\; \gamma_k\delta_k'\lambda_k.$$

Now suppose $\alpha \longrightarrow \alpha_1 \longrightarrow \ldots \longrightarrow \alpha_n \xrightarrow{\ell} \alpha'$. The last action ℓ could be performed by γ_1 or δ_1 or λ_1. If it is caused by γ_1, the process $\gamma_1\delta_1'\lambda_1$ can trivially bisimulate the action sequence. If it is caused by δ_1, it suffices to make use of the bisimulation base property. The third case is similar to the second one. Property 3 and property 4 of Definition 11 are necessary in the last case. By repeating the argument we eventually get a transition sequence from β that bisimulates $\alpha \longrightarrow \alpha_1 \longrightarrow \ldots \longrightarrow \alpha_n \xrightarrow{\ell} \alpha'$ in the style prescribed in Definition 11. □

Given a bisimulation base $\mathcal{A}$, it is semi-decidable to check if $\mathcal{A} \vdash \alpha = \beta$. This property, together with Lemma 12, explains why bisimulation base has often been used to produce (semi-)decidable procedure. For the application of this technique to the decidability study of strong bisimilarity on BPA and branching bisimilarity on totally normed BPA, see [HS91, Ḧ91, CHS92].

3 Approximation of Branching Bisimilarity

To look at the algebraic property of the branching bisimilarity $\simeq$ more closely, we introduce a notion of normedness appropriate for the equivalence.

Definition 13. *The* branching norm *of a BPA process α is the least number k such that $\exists j_1 \dots j_k . \exists \alpha_1 \dots \alpha_k . \alpha \rightarrow^* \xrightarrow{j_1} \alpha_1 \rightarrow^* \xrightarrow{j_2} \dots \alpha_{k-1} \rightarrow^* \xrightarrow{j_k} \alpha_k \rightarrow^* \epsilon$. The branching norm of α is denoted by $\|\alpha\|_b$.*

Clearly if $\alpha \rightarrow^* \alpha'$ then $\|\alpha'\|_b = \|\alpha\|_b$. It is also clear that $\|\alpha\|_b \leq \|\alpha\|$. The inequality is useful since we can replace $\|\alpha\|_b$ by $\|\alpha\|$ when we look for effective upper bound. Let's look at a number of examples:

- The branching norm of A defined by $\{A \xrightarrow{\tau} \epsilon\}$ is 0; the branching norm of B defined by $\{B \xrightarrow{a} B, B \xrightarrow{\tau} \epsilon\}$ is 1; and the branching norm of C defined by $\{C \xrightarrow{\tau} Y, Y \xrightarrow{b} Y, Y \xrightarrow{\tau} \epsilon\}$ is 1.
- The branching norm of D defined by $\{D \xrightarrow{a} D\}$ is ∞.
- The branching norm of E defined by $\{E \xrightarrow{\tau} E\}$ is tricky. An algorithm that tries to calculate the branching norm would never stop. We will say that the branching norm of E is undefined. We denote this fact by writing $\|E\|_b = \perp$.

We now state several simple lemmas about branching norm. The first is standard.

Lemma 14. *If $\alpha \simeq \beta$ then $\|\alpha\|_b = \|\beta\|_b$.*

The second one is expected.

Lemma 15. *The following hold:*

1. *If $\|\alpha\|_b = 0$ then $\alpha \simeq \epsilon$.*
2. *If $\|\alpha\|_b = \perp$ then $\alpha \simeq \Omega$.*
3. *If $\|\alpha\|_b = \infty$ then $\alpha\gamma \simeq \alpha$.*

Proof. If α cannot perform any external action, then its branching norm must be zero if it can terminate and must be undefined if it can only diverge. If α can never reach ϵ, then everything after α is useless. □

The next lemma is more interesting.

Lemma 16. *Suppose α is normed. Then $\alpha \simeq \delta\alpha$ if and only if $\|\alpha\|_b = \|\delta\alpha\|_b$.*

Proof. If $\|\alpha\|_b = \|\delta\alpha\|_b$ then every silent action sequence from $\delta\alpha$ to α must contain only state preserving silent transitions according to Computation Lemma. Moreover there must exist such a silent action path for otherwise $\|\alpha\|_b < \|\delta\alpha\|_b$. □

It does not follow from $\alpha \simeq \delta\alpha$ that $\delta \simeq \epsilon$. This is the most tricky situation as far as equivalence checking is concerned. A counter example is given by the BPA defined in Example 9. One has $AC \simeq C \simeq BC$. But clearly $\epsilon \not\simeq A \not\simeq B \not\simeq \epsilon$. To deal with situations like this we need the notion of relative norm.

3.1 Relative Norm

Definition 17. *The* relative norm $||\alpha||_b^\sigma$ *of α with respect to σ is the least k such that $\alpha\sigma \rightarrow^* \xrightarrow{j_1} \alpha^1\sigma \ldots \rightarrow^* \xrightarrow{j_{k-1}} \alpha^{k-1}\sigma \rightarrow^* \xrightarrow{j_k} \alpha^k\sigma \rightarrow^* \sigma$ for some $j_1, \ldots, j_k, \alpha_1, \ldots, \alpha_k$.*

In other words, the relative norm of α with respect to σ is the least total number of external actions and change-of-state silent actions $\alpha\sigma$ must perform in order to reach σ. Obviously $0 \le ||\alpha||_b^\sigma \le ||\alpha||_b$. Returning to the BPA Γ_2 defined in Example 9 again, we see that $||A||_b^B = 1$ and $||A||_b^C = 0$.

Using the notion of relative norm we can introduce the following terminologies:

- A transition $X\sigma \xrightarrow{\ell} \eta\sigma$ is *norm consistent* if either $||\eta||_b^\sigma = ||X||_b^\sigma$ and $\ell = \tau$ or $||\eta||_b^\sigma = ||X||_b^\sigma - 1$ and $\ell \neq \tau \vee \ell = \iota$.
- If $X\sigma \longrightarrow \eta\sigma$ is norm consistent with $||X||_b^\sigma > 0$, then it is *norm splitting* if there are at least two variables in η whose relative norms in $\eta\sigma$ are greater than 0.

For a normed BPA Δ no silent transition sequence contains more than $||\Delta||_b$ norm splitting transitions, where $||\Delta||_b$ denotes $\max\{||X_i||_b \mid 1 \le i \le n_\Delta \text{ and } X_i \text{ is normed}\}$.

The crucial property about relative norm is described in the following lemma.

Lemma 18. *Suppose $\alpha, \beta, \delta, \gamma$ are normed and $||\alpha||_b^\gamma = ||\beta||_b^\delta$. If $\alpha\gamma \simeq \beta\delta$ then $\gamma \simeq \delta$.*

Proof. Suppose $||\alpha||_b^\gamma = ||\beta||_b^\delta$. Now $||\alpha||_b^\gamma + ||\gamma||_b = ||\alpha\gamma||_b = ||\beta\delta||_b = ||\beta||_b^\delta + ||\delta||_b$. Therefore $||\gamma||_b = ||\delta||_b$. A norm consistent action sequence $\alpha\gamma \rightarrow^* \xrightarrow{j_1} \ldots \rightarrow^* \xrightarrow{j_k} \rightarrow^* \gamma$ must be matched up by $\beta\delta \rightarrow^* \xrightarrow{j_1} \ldots \rightarrow^* \xrightarrow{j_k} \beta'\delta$ for some β'. Clearly $||\beta'\delta||_b = ||\gamma||_b = ||\delta||_b$. It follows from Lemma 16 that $\delta \simeq \beta'\delta \simeq \gamma$. □

It should be remarked that the above lemma fails if δ, γ are unnormed. Indeed the BPA defined in Example 8 renders a counter example. Lemma 18 describes a weak form of left cancelation property. The general left cancelation property fails. The normed process A defined in Example 9 satisfies $AA \simeq A$. But clearly $A \not\simeq \epsilon$. Obviously this counter example also shows that right cancelation fails as well. In the decidability proof of the strong bisimilarity for normed BPA, the right cancelation property plays a crucial role. One wonders if some useful form of right cancelation holds. We dwelt upon the following conjecture for a few days: If $X\sigma \not\simeq \sigma$ and $\alpha X\sigma \simeq \beta X\sigma$ then $\alpha X \simeq \beta X$. We realized that it is a false conjecture when we came up with the BPA defined in Example 10. The search for an alternative to the right cancelation property led to the discovery of a nice and simple property of nBPA that allows us to control the size of common suffix of a pair of bisimilar processes. The basic idea is to remove redundant variables from two bisimilar processes while maintaining bisimilarity.

Definition 19. *A process α is* irredundant over γ *if $||\alpha||_b^\gamma > 0$. It is* redundant over γ *if $||\alpha||_b^\gamma = 0$. A process α is* head irredundant *if either $\alpha = \epsilon$ or $\alpha = X\alpha'$ for some X, α' such that $\alpha \not\simeq \alpha'$. It is head redundant otherwise. We write $Hirred(\alpha)$ to indicate that α is head irredundant. A process α is* completely irredundant *if every suffix of α is head irredundant. We write $Cirred(\alpha)$ to mean that α is completely irredundant.*

If α is normed, then α is irredundant over γ if and only if $\alpha\gamma \not\simeq \gamma$. The next lemma says that a redundant process consists solely of redundant variables.

Lemma 20. *Suppose $X_1, \ldots, X_k, \sigma$ are normed. Then $X_1 \ldots X_k$ is redundant over σ if and only if X_i is redundant over σ for every $X_i \in \{X_1, \ldots, X_k\}$.*

Proof. Suppose $X_1, \ldots, X_k, \sigma$ are normed and $X_1 \ldots X_k$ is redundant over σ. Then

$$X_1 \ldots X_k\sigma \Longrightarrow X_2 \ldots X_k\sigma \Longrightarrow \ldots \Longrightarrow X_k\sigma \Longrightarrow \sigma \simeq X_1 \ldots X_k\sigma.$$

It follows from Computation Lemma that $X_1 \ldots X_k\sigma \simeq X_2 \ldots X_k\sigma \simeq \ldots \simeq X_k\sigma \simeq \sigma$. We are done by using the congruence property. The implication in the other direction is due to congruence. □

The normedness assumption in Lemma 20 is crucial. The lemma would be invalid without the assumption. A counter example is given by the BPA defined in Example 7. The lemma suggests to introduce, for each σ, the *redundant set* $\mathcal{R}_\sigma$ of σ defined by

$$\mathcal{R}_\sigma \stackrel{\text{def}}{=} \{X \mid X\sigma \simeq \sigma\}.$$

In other words $\mathcal{R}_\sigma$ is the set of the redundant variables over σ. Let $\mathcal{V}(\alpha)$ be the set of variables appearing in α. We have two useful corollaries. The first is simple.

Corollary 21. *Suppose α, σ are normed. Then $\alpha\sigma \simeq \sigma$ if and only if $\mathcal{V}(\alpha) \subseteq \mathcal{R}_\sigma$.*

The second is instructive to our construction of tableau.

Corollary 22. *Suppose $\alpha, \beta, \sigma_0, \sigma_1$ are defined in a normed BPA and $\mathcal{R}_{\sigma_0} = \mathcal{R}_{\sigma_1}$. Then $\alpha\sigma_0 \simeq \beta\sigma_0$ if and only if $\alpha\sigma_1 \simeq \beta\sigma_1$.*

Proof. Suppose $\mathcal{R}_{\sigma_0} = \mathcal{R}_{\sigma_1}$. Let $\mathcal{S}$ be $\{(\alpha\sigma_0, \beta\sigma_0) \mid \alpha\sigma_1 \simeq \beta\sigma_1\}$. It is not difficult to see that $\mathcal{S} \cup \simeq$ is a branching bisimulation. □

3.2 Finite Tree Property

We now take a close look at the τ-trees of normed BPA processes. We are particularly interested in the subtrees of τ-trees containing only state preserving silent transitions. It turns out that all such subtrees are essentially finite. Let's explain what we mean by 'essentially finite'. Suppose $\beta = X\omega \simeq \alpha \xrightarrow{\ell} \alpha'$ and the action is bisimulated by $X\omega \to^* Z\theta \xrightarrow{\ell} \eta\theta$. Generally the length of the state preserving transition sequence $X\omega \to^* Z\theta$ is unbounded, and consequently we have to consider an infinite number of simulating sequences. But if there is some θ' such that $X\omega \to^* Z\theta'$ and $\theta' \simeq \theta$, and moreover the length of $X\omega \to^* Z\theta'$ is smaller than that of $X\omega \to^* Z\theta$, then we can safely abandon the bisimulating sequence $X\omega \to^* Z\theta \xrightarrow{\ell} \eta\theta$ in favor of $X\omega \to^* Z\theta' \xrightarrow{\ell} \eta\theta'$. We say that β has the finite tree property, and that the subtree of $\mathcal{T}^{\beta,\tau}$ containing only state preserving silent transitions is essentially finite, if there is a finite subtree $\mathcal{T}_\beta$ of $\mathcal{T}^{\beta,\tau}$ such that whenever $\beta \to^* Y\delta$ then there is some node $Y\delta'$ in $\mathcal{T}_\beta$ with $\delta' \simeq \delta$. Before discussing the finite tree property for nBPA, let's see in how many steps a variable V can reach silently every variable that can ever be reached from V silently. A variable Z is said to be reachable silently from a process γ if $\gamma \Longrightarrow Z\gamma'$ for some γ'. Suppose $V \to V_{k_1} \ldots V_{k_j}$. It takes at most $(r_\Delta - 1)t_\Delta + 1$ silent transitions to reach from V to V_{k_i} with $1 \leq i \leq j$. It follows that in fewer than $((r_\Delta - 1)t_\Delta + 1)(n_\Delta - 1)$ steps every variable W reachable via silent transitions from V can be reached. Recall that we have assumed that $r_\Delta > 0$ and $t_\Delta > 0$. So the previous formula is bounded by $r_\Delta t_\Delta(n_\Delta - 1)$.

Next we establish two technical lemmas. The first is about state preserving transitions caused by an irredundant variable.

Lemma 23. *Suppose $X\omega \to^* Y\omega'$ and $\|X\|_b^\omega = \|Y\|_b^{\omega'} > 0$. Then there is some ω'' such that $\omega'' \simeq \omega'$, $X\omega \to^* Y\omega''$ and the length of $X\omega \to^* Y\omega''$ is bounded by $r_\Delta t_\Delta(n_\Delta - 1)$.*

Proof. First notice that if $X\omega \to^* Y\omega'$ and $\|X\|_b^\omega = \|Y\|_b^{\omega'} > 0$ then $\omega' = \delta\omega$ for some δ and $X\omega \to^* Y\omega'$ is induced by $X \Longrightarrow Y\delta$. Notice that if

$$X\omega \to^* V\gamma_1\omega \to^* W\gamma_2\omega \to^* Y\omega'$$

for some head irredundant processes $V\gamma_1\omega, W\gamma_2\omega$ such that the processes on the path from $V\gamma_1\omega$ to $W\gamma_2\omega$, excluding both $V\gamma_1\omega$ and $W\gamma_2\omega$, are all head redundant, then by Lemma 6 we may assume that $V\gamma_1\omega \to^* W\gamma_2\omega$ is the shortest silent transition sequence from $V\gamma_1\omega$ to $W\gamma_2\omega$, whose length is bounded by $(r_\Delta - 1)t_\Delta + 1$. It follows that we may assume that the length of the transition sequence between every pair of neighboring head irreducible processes in $X\omega \to^* Y\omega'$ is bounded by $(r_\Delta - 1)t_\Delta + 1$. Under this assumption if the length of $X \Longrightarrow Y\delta$ is greater than $r_\Delta t_\Delta(n_\Delta - 1) \geq ((r_\Delta - 1)t_\Delta + 1)(n_\Delta - 1)$, then there must be two head irreducible processes $Z\gamma_0\omega, Z\gamma_1\omega$ such that

$$X \Longrightarrow Z\gamma_0 \Longrightarrow Z\gamma_1 \Longrightarrow Y\delta$$

and

$$X\omega \to^* Z\gamma_0\omega \to^* Z\gamma_1\omega \to^* Y\delta\omega.$$

It follows from $\|X\|_b^\omega = \|Y\|_b^{\omega'} > 0$ that $\|Z\|_b^{\gamma_0\omega} = \|Z\|_b^{\gamma_1\omega} = \|Y\|_b^{\omega'}$. Using the fact that $\|Z\|_b^{\gamma_0\omega} > 0$ one sees immediately that $\gamma_1 = \gamma_1'\gamma_0$ for some γ_1'. By removing the part γ_1' from every process in $Z\gamma_1\omega \to^* Y\delta\omega$ we can remove $Z\gamma_0\omega \to^* Z\gamma_1\omega$ from $X\omega \to^* Z\gamma_0\omega \to^* Z\gamma_1\omega \to^* Y\delta\omega$ and get a shorter sequence $X\omega \to^* Z\gamma_0\omega \to^* Y\delta'\omega$ for some δ' with $\delta'\omega \simeq \omega \simeq \delta\omega$ by Lemma 18. The above procedure can be repeated until we reach a transition sequence whose length is strictly bounded by $r_\Delta t_\Delta(n_\Delta - 1)$. □

The second deals with state preserving transitions caused by a redundant variable.

Lemma 24. *Suppose $Z_0 \Longrightarrow Z\delta$, $Z_0\omega \to^* Z\delta\omega$ and $\|Z_0\|_b^\omega = 0$. Then $Z_0 \Longrightarrow Z\delta'$ for some δ' such that $Z_0\omega \to^* Z\delta'\omega$ and the length of $Z_0 \Longrightarrow Z\delta'$ is bounded by $r_\Delta t_\Delta(n_\Delta - 1)$.*

Proof. It is important to notice that every silent transition sequence $Z_0 \Longrightarrow Z\delta$ must take the following shape

$$\begin{aligned}
Z_0 &\longrightarrow \eta_l^1 Z_1 \eta_r^1 \Longrightarrow Z_1\eta_r^1 \\
&\longrightarrow \eta_l^2 Z_2 \eta_r^2 \eta_r^1 \Longrightarrow Z_2\eta_r^2\eta_r^1 \\
&\longrightarrow \Longrightarrow \ldots \\
&\longrightarrow \eta_l^k Z_k \eta_r^k \ldots \eta_r^2\eta_r^1 \Longrightarrow Z_k\eta_r^k \ldots \eta_r^2\eta_r^1 = Z\delta
\end{aligned}$$

for some $\eta_l^1, Z_1, \eta_r^1, \eta_l^2, Z_2, \eta_r^2, \ldots, \eta_l^k, Z_k, \eta_r^k$ such that none of $\eta_r^k, \ldots \eta_r^2, \eta_r^1$ is affected in the transition sequence. By the assumption $\|Z\delta\|_b^\omega = 0$, one derives from Lemma 3 that $\eta_l^1, Z_1, \eta_r^1, \eta_l^2, Z_2, \eta_r^2, \ldots, \eta_l^k, Z_k, \eta_r^k$ are all redundant over ω. If $k \geq n_\Delta$ then there must be $i < j$ such that $Z_i = Z_j$. In this case we can remove the transition sequence $Z_i\eta_r^i \ldots \eta_r^2\eta_r^1 \Longrightarrow Z_j\eta_r^j \ldots \eta_r^2\eta_r^1$ from $Z_0 \Longrightarrow Z\delta$ in the following manner:

$$\begin{aligned}
Z_0 &\longrightarrow \eta_l^1 Z_1 \eta_r^1 \Longrightarrow Z_1\eta_r^1 \\
&\longrightarrow \Longrightarrow \ldots \\
&\longrightarrow \eta_l^i Z_i \eta_r^i \ldots \eta_r^2\eta_r^1 \Longrightarrow Z_i\eta_r^i \ldots \eta_r^2\eta_r^1 = Z_j\eta_r^i \ldots \eta_r^2\eta_r^1 \\
&\longrightarrow \Longrightarrow \ldots \\
&\longrightarrow \eta_l^k Z_k \eta_r^k \ldots \eta_r^{j+1}\eta_r^i \ldots \eta_r^1 \Longrightarrow Z_k\eta_r^k \ldots \eta_r^{j+1}\eta_r^i \ldots \eta_r^1.
\end{aligned}$$

So we may as well assume that $k < n_\Delta$. According to Lemma 6, we may assume further that, for each $h \in \{1, \ldots, k\}$, the transition sequence $\eta_l^h Z_h \eta_r^h \Longrightarrow Z_h\eta_r^h$ is the shortest among all the silent transition sequences from $\eta_l^h Z_h \eta_r^h$ to $Z_h\eta_r^h$, which is bounded by $(r_\Delta - 1)t_\Delta$. It follows that some δ' exists such that $\delta' \simeq \delta$ and $Z_0 \rightarrow^* Z\delta'$ with its legnth bounded by $(r_\Delta - 1)t_\Delta(n_\Delta - 1) + (n_\Delta - 1) \leq r_\Delta t_\Delta(n_\Delta - 1)$. □

We are now in a position to prove that all normed BPA processes enjoy the finite tree property and that there are effective bounds on the size of the finite trees.

Lemma 25. *For each normed BPA process $\alpha = X\omega$, one can effectively construct a finite tree $\mathcal{T}_\alpha \subseteq_f \mathcal{T}^{\alpha,\tau}$ of height less than an effective bound H_α uniformly computable from α such that whenever $\alpha \rightarrow^* V\theta$ then $\theta \simeq \eta$ for some node $V\eta$ of $\mathcal{T}_\alpha$.*

Proof. Suppose $\alpha = X\omega$. We take a closer look at the state preserving silent transitions caused by X. There are four possibilities:

1. $\|X\|_b^\omega = 0$. In this case an upper bound is given by Lemma 24.

2. $\|X\|_b^\omega > 0$. $X\omega \rightarrow^* Z\delta\omega$ for some Z, δ such that $\|Z\|_b^{\delta\omega} = \|X\|_b^\omega$. In this case an upper bound is given by Lemma 23.

3. $\|X\|_b^\omega > 0$. $X\omega \rightarrow^* Z\delta\omega$ for some Z, δ such that $X \Longrightarrow Z\delta$, $\|Z\|_b^{\delta\omega} = 0$ and all the processes on the path, except $X\omega$, are head redundant. Using the same argument as in the proof of Lemma 24 we can easily show that $X \Longrightarrow Z\delta'$ for some δ' such that (i) $X\omega \rightarrow^* Z\delta'\omega$, (ii) apart from $X\omega$ all the processes in $X\omega \rightarrow^* Z\delta\omega$ are head redundant and (iii) the length of $X \Longrightarrow Z\delta'$ is bounded by $r_\Delta t_\Delta n_\Delta$.

4. $\|X\|_b^\omega > 0$. $X\omega \rightarrow^* Y\gamma\omega \xrightarrow{\tau} \eta\gamma\omega$ is caused by $X \rightarrow^* Y\gamma \xrightarrow{\tau} \eta\gamma$ for some Y, γ, η such that $Y\gamma\omega \xrightarrow{\tau} \eta\gamma\omega$ is a norm splitting transition and no transition in $X\omega \rightarrow^* Y\gamma\omega$ is norm splitting. By Lemma 23 we know that in no more than $r_\Delta t_\Delta(n_\Delta - 1) + 1$ steps we can encounter the norm splitting transition.

The bounds derived in the above four cases are no greater than $r_\Delta t_\Delta n_\Delta$.

If $X\omega$ is head irredundant then $\alpha \rightarrow^* V\theta$ may contain at most $\|\Delta\|_b - 1$ norm splitting transitions and that no transition in the sequence can affect ω. After the last norm splitting transition it takes fewer than $r_\Delta t_\Delta n_\Delta$ steps to exhaust all possibilities.

Therefore some η exists such that $\eta \simeq \theta$ and $\alpha \rightarrow^* V\eta$ with its length bounded by $r_\Delta t_\Delta n_\Delta \|\Delta\|_b$. If $X\omega$ is head redundant, then either every variable in α is redundant or there are at most $|\alpha| - 1$ consecutive redundant variables in α. Consequently we have the bound $\max\{r_\Delta t_\Delta n_\Delta |\alpha|, r_\Delta t_\Delta n_\Delta (|\alpha| - 1 + \|\Delta\|_b)\}$. So we can set $H_\alpha = r_\Delta t_\Delta n_\Delta (|\alpha| + \|\Delta\|_b)$; and we can let $\mathcal{T}_\alpha$ be the finite tree consisting of all silent transition sequences whose length is no more than H_α. □

The effective bound H_α defined in the above proof could be too generous. A much more economic analysis is necessary if one intends to study the complexity aspect of the problem. For the purpose of this paper it is sufficient to know that the number of possible simulation sequences of a given transition is effectively bounded.

Corollary 26. *Suppose $\alpha, \beta\gamma$ are normed BPA processes and $\gamma \not\simeq \beta\gamma$. If $\beta\gamma \simeq \alpha \xrightarrow{J} \alpha'$, then there is a transition sequence $\beta\gamma \rightarrow^* \beta''\gamma \xrightarrow{J} \beta'\gamma$ with its length bounded by H_β such that $\beta''\gamma \simeq \alpha$ and $\beta'\gamma \simeq \alpha'$.*

Proof. Under the assumption $\gamma \not\simeq \beta\gamma$ we can repeat the proof of Lemma 25 for $\beta\gamma$ in a way that γ is not affected. So the computable bound is independent γ. □

We are now in a position to prove the following.

Proposition 27. *The relation $\not\simeq$ on normed BPA processes is semi-decidable.*

Proof. We define $\simeq_k$, the branching bisimilarity up to depth k, by exploiting Corollary 26. The inductive definition is as follows:

- $\alpha \simeq_0 \beta$ for all α, β.
- $\alpha \simeq_{i+1} \beta$ if the following hold:
    1. If $\beta \simeq_i^{-1} \alpha \xrightarrow{\ell} \alpha'$ then one of the following statements is valid:
        (i) $\ell = \tau$ and $\alpha' \simeq_i \beta$.
        (ii) $\beta \Longrightarrow \beta'' \simeq_i^{-1} \alpha$ for some β'' such that $\beta'' \xrightarrow{\ell} \beta' \simeq_i^{-1} \alpha'$ for some β' and the length of $\beta \Longrightarrow \beta''$ is bounded by H_β.
    2. If $\alpha \simeq_i \beta \xrightarrow{\ell} \beta'$ then one of the following statements is valid:
        (i) $\ell = \tau$ and $\alpha \simeq_i \beta'$.
        (ii) $\alpha \Longrightarrow \alpha'' \simeq_i \beta$ for some α'' such that $\alpha'' \xrightarrow{\ell} \alpha' \simeq_i \beta'$ for some α' and the length of $\alpha \Longrightarrow \alpha''$ is bounded by H_α.

Since we are only dealing with normed BPA, we do not need the following clause in the above definition: If $\alpha = \epsilon$ then $\beta \Longrightarrow \epsilon$, and if $\beta = \epsilon$ then $\alpha \Longrightarrow \epsilon$. Each $\simeq_k$ is decidable because (i) the finite branching property holds for the transition system, (ii) the length of every simulating sequence is bounded effectively, and (iii) the property 3 in the definition of $\simeq_{i+1}$ is decidable. Using Corollary 26 one easily sees that $\simeq \subseteq \bigcap_{k\in\omega} \simeq_k$. The converse inclusion can be proved by showing that the relation

$$\{(\alpha, \beta) \mid \alpha \simeq_k \beta \text{ for infinitely many } k\}$$

is a branching bisimulation, which is standard. The semi-decidability of $\not\simeq$ then follows from the coincidence of $\simeq$ with $\bigcap_{k\in\omega} \simeq_k$ and the decidability of $\simeq_k$ for every $k \geq 0$. □

4 Equality Checking

Milner's approach to the proof of bisimulation equivalence between two finite state processes is by fixpoint induction [Mil84, Mil89b]. To prove that P is weakly congruent to Q, we unfold P and Q simultaneously to produce a tree of equality. If P is really weakly congruent to Q, we get a finite tree. A leaf of the tree is either a pair of identical processes or a pair (M, N) that is the same as one of its ancestors. Now by folding up leaves of the latter form, we get a guarded equation satisfied by both P and Q. The fixpoint induction then allows us to conclude that P, Q are provably equal. The tableau approach can be seen as a generalization of the fixpoint induction from finite state systems to infinite state systems. A tree of this kind has been called a tableau system [HS91, H9̈1]. The goal of this section is to give a semi-decidable procedure for the branching bisimilarity on normed BPA processes using tableau method.

4.1 Tableau Method

Suppose $\alpha_0\alpha \not\simeq \alpha$ and $\beta_0\beta \not\simeq \beta$. A *match* for an equality $\alpha_0\alpha = \beta_0\beta$ over (α,β) is a finite set $\{\gamma_i\alpha = \lambda_i\beta\}_{i=1}^k$ containing only those equalities accounted for in the following two reciprocal conditions:

1. For each transition $\alpha_0\alpha \xrightarrow{\ell} \alpha'\alpha$, one of the following holds:
   - $\ell = \tau$ and $\alpha'\alpha = \beta_0\beta \in \{\gamma_i\alpha = \lambda_i\beta\}_{i=1}^k$;
   - there is a sequence $\beta_0\beta \xrightarrow{\tau} \beta_1\beta \xrightarrow{\tau} \ldots \xrightarrow{\tau} \beta_n\beta \xrightarrow{\ell} \beta'\beta$, for $n < H_{\beta_0}$, such that $\{\alpha_0\alpha = \beta_1\beta, \ldots, \alpha_0\alpha = \beta_n\beta, \alpha'\alpha = \beta'\beta\} \subseteq \{\gamma_i\alpha = \lambda_i\beta\}_{i=1}^k$ and $\beta_i\beta \not\simeq \beta$ for all $i \in \{1, \ldots, n\}$.
2. For each transition $\beta_0\beta \xrightarrow{\ell} \beta'\beta$, one of the following holds:
   - $\ell = \tau$ and $\alpha_0\alpha = \beta'\beta \in \{\gamma_i\alpha = \lambda_i\beta\}_{i=1}^k$;
   - there is a sequence $\alpha_0\alpha \xrightarrow{\tau} \alpha_1\alpha \xrightarrow{\tau} \ldots \xrightarrow{\tau} \alpha_n\alpha \xrightarrow{\ell} \alpha'\alpha$, for $n < H_{\alpha_0}$, such that $\{\alpha_1\alpha = \beta_0\beta, \ldots, \alpha_n\alpha = \beta_0\beta, \alpha'\alpha = \beta'\beta\} \subseteq \{\gamma_i\alpha = \lambda_i\beta\}_{i=1}^k$ and $\alpha_i\alpha \not\simeq \alpha$ for all $i \in \{1, \ldots, n\}$.

If $\alpha_0\sigma \not\simeq \sigma$ and $\beta_0\sigma \not\simeq \sigma$, a match for $\alpha_0\sigma = \beta_0\sigma$ over (σ, σ) is said to be a match for $\alpha_0\sigma = \beta_0\sigma$ over σ. The conditions $\beta_i\beta \not\simeq \beta$ and $\alpha_i\alpha \not\simeq \alpha$ can be dropped. We include them for conceptual clarity. The presence of the computable bounds $H_{\alpha_0}, H_{\beta_0}$, which are reasonable in view of Corollary 26, guarantees that the number of matches for $\alpha_0\alpha = \beta_0\beta$ over (α,β) is effectively bounded.

Given a normed BPA process α, it is easy to check if $\alpha \simeq \epsilon$. It amounts to checking if $X \simeq \epsilon$ for every $X \in \mathcal{V}(\alpha)$, which is the same as checking if $X \Longrightarrow \epsilon$ and if X can only perform silent actions. So we shall focus on the equality between two nontrivial normed BPA processes, say α_0, β_0. A *tableau* for $\alpha_0 = \beta_0$ is a tree with each of its nodes labeled by an equality between two normed BPA processes. The root of the tableau is labeled by $\alpha_0 = \beta_0$. We shall distinguish between *global tableau*, *local tableau* and *ambient tableau*. The global tableau is the overall tableau whose root is

Decmp
$$\frac{\gamma\alpha = \lambda\beta}{\alpha = \beta \quad \{U\alpha = \alpha\}_{U\in\mathcal{V}(\gamma)} \quad \{V\beta = \beta\}_{V\in\mathcal{V}(\lambda)}} \quad \begin{array}{l}|\gamma| + |\lambda| > 0,\\ \forall U \in \mathcal{V}(\gamma).U \Longrightarrow \epsilon,\\ \forall V \in \mathcal{V}(\lambda).V \Longrightarrow \epsilon.\end{array}$$

SDecmp
$$\frac{\gamma\alpha = \lambda\beta}{\alpha = \beta \quad \{U\alpha = \alpha\}_{U\in\mathcal{V}(\gamma)} \quad \{V\beta = \beta\}_{V\in\mathcal{V}(\lambda)}} \quad \begin{array}{l}|\gamma| + |\lambda| > 0,\\ \mathit{Hirred}(\alpha),\ \mathit{Hirred}(\beta),\\ \forall U \in \mathcal{V}(\gamma).U \Longrightarrow \epsilon,\\ \forall V \in \mathcal{V}(\lambda).V \Longrightarrow \epsilon.\end{array}$$

Match
$$\frac{\gamma\alpha = \lambda\beta}{\alpha_1\alpha = \beta_1\beta \ \ldots \ \alpha_k\alpha = \beta_k\beta} \quad \begin{array}{l}\gamma\alpha \neq \alpha,\ \lambda\beta \neq \beta, \text{ and } \{\alpha_i\alpha = \beta_i\beta\}_{i=1}^{k}\\ \text{is a match for } \gamma\alpha = \lambda\beta \text{ over } (\alpha,\beta).\end{array}$$

SubstL
$$\frac{\gamma\alpha = \lambda\beta}{\gamma\delta\beta = \lambda\beta} \quad \alpha = \delta\beta \text{ is the residual.}$$

SubstR
$$\frac{\gamma\alpha = \lambda\beta}{\gamma\alpha = \lambda\delta\alpha} \quad \delta\alpha = \beta \text{ is the residual.}$$

ContrL
$$\frac{\gamma Z\delta = \lambda}{\gamma\delta = \lambda \quad Z\delta = \delta} \quad \mathit{Hirred}(\delta),\ Z \Longrightarrow \epsilon \text{ and } |\gamma Z\delta| > \max\{|\alpha_0|, |\beta_0|\}||\Delta||.$$

ContrR
$$\frac{\gamma = \lambda Z\delta}{\gamma = \lambda\delta \quad Z\delta = \delta} \quad \mathit{Hirred}(\delta),\ Z \Longrightarrow \epsilon \text{ and } |\lambda Z\delta| > \max\{|\alpha_0|, |\beta_0|\}||\Delta||.$$

ContrC
$$\frac{\gamma\sigma'\sigma_0\sigma_1 = \lambda\sigma'\sigma_0\sigma_1}{\gamma\sigma'\sigma_1 = \lambda\sigma'\sigma_1 \quad \{V\sigma_1 = \sigma_1\}_{V\in\mathcal{V}(\sigma_0)}} \quad \begin{array}{l}|\sigma'\sigma_0\sigma_1| > 2^{n_\Delta},\ |\sigma_0| > 0,\\ \mathit{Hirred}(\sigma_1),\\ \forall V \in \mathcal{V}(\sigma_0).V \Longrightarrow \epsilon.\end{array}$$

Figure 2: Rules for Global Tableaux

labeled by the goal $\alpha_0 = \beta_0$. Local tableau and ambient tableau are relative notions; their difference will be explained later. A global tableau is constructed from the rules given in Fig. 2. Decmp rule decomposes a goal into several subgoals. We shall find it useful to use SDecmp, which is a stronger version of Decmp. The side condition of SDecmp ensures that it is unnecessary to apply it consecutively since both α and β are irredundant. When applying Decmp rule we assume that an equality $\gamma\sigma = \sigma$, respectively $\sigma = \gamma\sigma$, is always decomposed in the following manner

$$\frac{\gamma\sigma = \sigma}{\sigma = \sigma \quad \{V\sigma = \sigma\}_{V\in\mathcal{V}(\gamma)}} \text{ respectively } \frac{\sigma = \gamma\sigma}{\sigma = \sigma \quad \{V\sigma = \sigma\}_{V\in\mathcal{V}(\gamma)}}.$$

Accordingly $\gamma = \epsilon$, respectively $\epsilon = \gamma$, is always decomposed in the following fashion

$$\frac{\gamma = \epsilon}{\epsilon = \epsilon \quad \{V = \epsilon\}_{V\in\mathcal{V}(\gamma)}} \text{ respectively } \frac{\epsilon = \gamma}{\epsilon = \epsilon \quad \{V = \epsilon\}_{V\in\mathcal{V}(\gamma)}}.$$

Match rule can be applied as long as α respectively β is prefixed by process irredundant over α respectively β. SubstL and SubstR allow one to create common suffix for the two processes in an equality. ContrL and ContrR are used to remove a redundant variable inside a process. In the side conditions of these two rules, α_0, β_0 are the processes appearing in the root label of the global tableau. ContrC deletes redundant variables from the common suffix of a node label whenever the size of the common suffix is over limit. Notice that all the rules are forward and backward sound. Notice also that all the side conditions on the rules are semi-decidable due to the semi-decidability of $\not\simeq$.

In what follows a node $Z\eta = W\kappa$ to which Match rule is applied with the condition $Z\eta \not\simeq \eta \wedge W\kappa \not\simeq \kappa$ is called an *M-node*. A node of the form $Z\sigma = \sigma$ with σ being head irredundant is called a *V-node*. We now describe in detail how a global tableau for $\alpha_0 = \beta_0$ is constructed. Assuming $\alpha_0 = \gamma X\alpha_1$ and $\beta_0 = \lambda Y\beta_1$ such that $X\alpha_1 \not\simeq \alpha_1$ and $Y\beta_1 \not\simeq \beta_1$, we apply the following instance of SDecmp rule to construct the children of the root:

$$\frac{\gamma X\alpha_1 = \lambda Y\beta_1}{X\alpha_1 = Y\beta_1 \quad \{UX\alpha_1 = X\alpha_1\}_{U\in\mathcal{V}(\gamma)} \quad \{VY\beta_1 = Y\beta_1\}_{V\in\mathcal{V}(\lambda)}}.$$

By definition $X\alpha_1 = Y\beta_1$ is an M-node and $\{UX\alpha_1 = X\alpha_1\}_{U\in\mathcal{V}(\gamma)} \cup \{VY\beta_1 = Y\beta_1\}_{V\in\mathcal{V}(\lambda)}$ is a set of V-nodes. These nodes are the roots of new subtableaux. We now explain how these subtableaux are constructed:

I. Starting from $X\alpha_1 = Y\beta_1$ we apply Match rule under the condition that neither α_1 nor β_1 is affected. The application of Match rule is repeated to grow the subtableau rooted at $X\alpha_1 = Y\beta_1$. The construction of the tree is done in a breadth first fashion. So the tree grows level by level. At some stage we apply Decmp rule to all the current leaves. The application of Decmp rule must meet the following conditions:

   – Both α_1 and β_1 must be kept intact in all the current leaves.

   – Either α_1 or β_1 is exposed in at least one current leaf.

   Choose a leaf labeled by either $\alpha_1 = \delta_1\beta_1$ for some δ_1 or by $\delta'_1\alpha_1 = \beta_1$ for some δ'_1 and call it the *residual node* or *R-node*. Suppose the residual node is $\alpha_1 = \delta_1\beta_1$. All the other current leaves, the *non-residual nodes*, must be labeled by an equality of the form $\gamma_1\alpha_1 = \lambda_1\beta_1$. A non-residual node with label $\gamma_1\alpha_1 = \lambda_1\beta_1$ is then attached with a single child labeled by $\gamma_1\delta_1\beta_1 = \lambda_1\beta_1$. This is an application of SubstL rule. Now we can recursively apply the global tableau construction to $\gamma_1\delta_1\beta_1 = \lambda_1\beta_1$ to produce a new subtableau.

   Let's take a look at the size of $\gamma_1\alpha_1 = \lambda_1\beta_1$. The number of times Match rule has been applied in the above construction is bounded by $||\Delta||$. By the definition of Match rule, the maximal length of transition sequence admitted in a match is effectively bounded. So the maximal length of overall transition sequences from the root to the leaves of the subtableau is bounded by a number, say S, effectively computable. It follows that $|\delta_1|$, $|\gamma_1|$ and $|\lambda_1|$ are effectively bounded by $r_\Delta S$.

   The treatment of a V-node child, say $UX\alpha_1 = X\alpha_1$, is similar. We keep applying Match rule over α_1 as long as the side condition is met. At certain stage we apply Decmp rule to all the leaves. The decomposition should meet the following conditions:

- No occurrence of α_1 is affected.
- There is an application of Decmp that takes the following shape

$$\frac{\gamma_1\alpha_1 = \lambda_1\alpha_1}{\alpha_1 = \alpha_1 \quad \{V\alpha_1 = \alpha_1\}_{V\in\mathcal{V}(\gamma_1)} \quad \{V\alpha_1 = \alpha_1\}_{V\in\mathcal{V}(\lambda_1)}}.$$

It is clear that the size bound $r_\Delta S$ still applies to the new leaves. We can then recursively apply the global tableau construction to the current leaves to produce new subtableaux.

The construction of a path in a global tableau ends with either a successful leaf or an unsuccessful leaf. The definition of successful/unsuccessful leaf for the global tableau is as follows:

- A *successful leaf* is either a node labeled by $\varsigma = \varsigma$ for some ς, or a node labeled by $\epsilon = V$ ($V = \epsilon$) with $V \simeq \epsilon$, or a node labeled by $\gamma\sigma = \lambda\sigma$ that meets two conditions: (i) the label is the same as the label of one of its ancestors and (ii) $|\gamma|, |\lambda| \leq 2r_\Delta S$ and $|\sigma| \leq 2^{n_\Delta}$.
- An *unsuccessful leaf* is produced if the node is either labeled by $\epsilon = V$ ($V = \epsilon$) with $V \not\simeq \epsilon$, or labeled by some $\varsigma = \varsigma'$ with distinct ς, ς' such that no rule is applicable to $\varsigma = \varsigma'$.

II. In the above construction the R-node $\alpha_1 = \delta_1\beta_1$ deserves special treatment. It is the root of a new subtableau, which might contain another R-node say $\alpha_2 = \delta_2\beta_2$. In this way new R-nodes are generated one by one. Two things may happen. An R-node shares the same label as one of its ancestors. In this case we get a successful leaf. A different situation arises when we get an R-node whose size is larger than $\max\{|\alpha_0|, |\beta_0|\}\|\Delta\|$. Since the branching norms of the R-nodes strictly decrease, the size violation must be caused by a huge number of redundant variables. Therefore we may apply ContrL and/or ContrR repeatedly to reduce the size of the node. By the end of this procedure we get a leaf whose size is under control.

If after an application of SubstL/SubstR rule we get a node $\alpha'\sigma'\sigma_0\sigma_1 = \beta'\sigma'\sigma_0\sigma_1$ such that ContrC rule is applicable, we get a *C-node*. Once a C-node appears, we immediately apply ContrC rule to reduce the size of its common suffix. An application of ContrC rule may produce some V-nodes and some more C-nodes. We apply ContrC to the new C-nodes if necessary. Intuitively we should apply ContrC sufficiently often so that the common suffix becomes completely irredundant. Eventually either the length of the common suffix has become no more than 2^{n_Δ}, in which case we continue to build up the global tableau, or the side conditions of Localization rule are satisfied, in which case we get an *L-node*. Notice that without ContrC rule we may come to a node $\alpha'\sigma'\sigma_0\sigma_1 = \beta'\sigma'\sigma_0\sigma_1$ where the length of the common suffix surpasses the limit but the condition $Cirred(\sigma'\sigma_0\sigma_1) \wedge Cirred(\sigma'\sigma_1)$ is not satisfied.

$$\text{Localization}\quad \frac{\gamma\sigma'\sigma_0\sigma_1 = \lambda\sigma'\sigma_0\sigma_1}{\begin{array}{l}\gamma\sigma'\sigma_1 = \lambda\sigma'\sigma_1\\ \{X_i\sigma_1 = \sigma_1\}_{i\in I}\\ \{X_i\sigma_0\sigma_1 = \sigma_0\sigma_1\}_{i\in I}\end{array}}\quad \begin{array}{l}|\gamma| > 0 \text{ and } |\lambda| > 0;\ |\sigma'\sigma_0\sigma_1| > 2^{n_\Delta},\\ 2^{n_\Delta} \geq |\sigma_1| > 0 \text{ and } |\sigma_0| > 0;\\ \mathit{Cirred}(\sigma'\sigma_0\sigma_1) \text{ and } \mathit{Cirred}(\sigma'\sigma_1);\\ \gamma\sigma'\sigma_0\sigma_1 \not\approx \sigma'\sigma_0\sigma_1,\ \gamma\sigma'\sigma_1 \not\approx \sigma'\sigma_1;\\ \lambda\sigma'\sigma_0\sigma_1 \not\approx \sigma'\sigma_0\sigma_1,\ \lambda\sigma'\sigma_1 \not\approx \sigma'\sigma_1;\\ I \cap J = \emptyset,\ I \cup J = \{1,\ldots,n_\Delta\};\\ \forall j \in J.\ X_j\sigma_0\sigma_1 \not\approx \sigma_0\sigma_1 \text{ and } X_j\sigma_1 \not\approx \sigma_1;\\ X_i \Longrightarrow \epsilon \text{ for all } i \in I.\end{array}$$

Figure 3: Rule for Local Tableaux

Now suppose an L-node is labeled by $\alpha'\sigma'\sigma_0\sigma_1 = \beta'\sigma'\sigma_0\sigma_1$. In the following application of Localization rule

$$\frac{\alpha'\sigma'\sigma_0\sigma_1 = \beta'\sigma'\sigma_0\sigma_1}{\{X_i\sigma_1 = \sigma_1\}_{i\in I} \quad \alpha'\sigma'\sigma_1 = \beta'\sigma'\sigma_1 \quad \{X_i\sigma_0\sigma_1 = \sigma_0\sigma_1\}_{i\in I}}\ I \cup J = \{1,\ldots,n_\Delta\},$$

the node $\alpha'\sigma'\sigma_1 = \beta'\sigma'\sigma_1$ is a new L-node, $\{X_i\sigma_1 = \sigma_1\}_{i\in I}$ are the *left siblings* of the new L-node, and $\{X_i\sigma_0\sigma_1 = \sigma_0\sigma_1\}_{i\in I}$ are the *right siblings*. We call $\{X_i \mid i \in I\}$ the *R-set* of the new L-node. If the size of the common suffix of $\alpha'\sigma'\sigma_1 = \beta'\sigma'\sigma_1$ is still larger than 2^{n_Δ}, we continue to apply Localization rule. Otherwise we get an *L-root*, which is the root of a local tableau. Now suppose $\alpha'\sigma'\sigma_1 = \beta'\sigma'\sigma_1$ is an L-root. The construction of the local tableau should stick to two principles described as follows:

- *Locality*. No application of Decmp, SDecmp, SubstL, SubsR and ContrC should ever affect $\sigma'\sigma_1$ or any suffix of $\sigma'\sigma_1$. Notice that applications of SubstL or SubstR can never affect $\sigma'\sigma_1$ or any suffix of $\sigma'\sigma_1$.

- *Consistency*. Suppose $\gamma\alpha = \lambda\beta$ is a node to which Match rule is applied using a match over (α,β). Then either $\sigma'\sigma_1$ is a suffix of both α and β, or $\alpha = \beta = \sigma''\sigma_1$ for some σ'' satisfying the following:
  - σ'' is a proper suffix of σ';
  - $\gamma = UZ$ and $\lambda = Z$ such that $Z\sigma''$ is a suffix of σ'; and
  - the match is over $\sigma''\sigma_1$.

The locality and consistency conditions basically say that the choices made in the construction of the local tableau should not contradict to the fact that $\sigma'\sigma_1$ is completely irredundant.

A right sibling may still be subject to an application of Localization rule. But since the size of a right sibling strictly shrinks, we eventually reach a situation in which Localization rule is no longer applicable.

A local tableau may contain another local tableau. In this case we say that the former is an ambient tableau of the latter. A series of nested local tableaux creates a hierarchical structure. A node of a local tableau is only compared to a node within the same local tableau with one exception: An L-root must be compared to the L-roots that are in the path from the root of the global tableau to the present L-root.

A local tableau inherits the definition of successful/unsuccessful leaves from the global tableau (see page 18). In addition a local tableau has two new kind of successful/unsuccessful leaves:

- An L-root is a successful leaf if it shares the same label with one of its ancestors that is also an L-root.
- Suppose $\alpha'\sigma'\sigma_0\sigma_1 = \beta'\sigma'\sigma_0\sigma_1$ is an L-node and its child $\alpha'\sigma'\sigma_1 = \beta'\sigma'\sigma_1$ is an L-root. In the local tableau rooted at $\alpha'\sigma'\sigma_1 = \beta'\sigma'\sigma_1$, a node of the form $Z\sigma_1 = \sigma_1$ is deemed as a leaf. It is a successful leaf if Z is in the R-set of the L-root; it is an unsuccessful leaf otherwise.

The first question one must ask about the tableau construction is if it always terminates. The answer is affirmative.

Lemma 28. *The size of every tableau for an equality is effectively bounded.*

Proof. A path cannot contain an infinite number of L-nodes since every L-node $\delta\sigma = \delta'\sigma$ satisfies $|\delta|, |\delta'| \leq 2r_\Delta S$ and $\sigma \leq 2r_\Delta S + 2^{n_\Delta}$. It cannot contain an infinite number of L-roots since every L-root $\delta\sigma = \delta'\sigma$ satisfies $|\delta|, |\delta'| \leq 2r_\Delta S$ and $\sigma \leq 2^{n_\Delta}$. In other words the length of a chain of nested local tableaux is effectively bounded. For the same reason a path in the global tableau or within a local tableau contains only an effectively bounded number of M-nodes and V-nodes. Such a path cannot contain an infinite number of R-nodes since that would imply the existence of an infinite number of M-nodes or V-nodes. It is easy to see that the number of R-nodes in such a path is also effectively bounded. Since the branching degree of all tableaux is bounded by a constant, we conclude that every tableau is finite with an effective bound. □

In view of Lemma 28 and the fact that the number of the matches for an equality is effectively bounded, we derive immediately the following corollary.

Corollary 29. *The number of tableaux for an equality is finite and effectively bounded.*

4.2 Completeness Proof

We now turn to tableaux that define bisimulation bases. A tableau is *successful* if all of its leaves, including all the leaves of all local tableau inside it, are successful. It is *unsuccessful* if it is not successful. The significance of the existence of a successful tableau is pointed out in the following proposition.

Proposition 30. *Suppose $X\alpha, Y\beta$ are normed BPA processes. Then $X\alpha \simeq Y\beta$ if and only if there is a successful tableau for $X\alpha = Y\beta$.*

Proof. If $X\alpha \simeq Y\beta$ we can easily construct a tableau using the bisimulation property, Corollary 26 and Corollary 22. Conversely suppose there is a successful tableau $\mathfrak{T}$

for $X\alpha = Y\beta$. Let $\mathcal{A}$ be the axiom system consisting of three parts as defined by the following equation:

$$\mathcal{A} \quad = \quad \mathcal{A}_b \cup \mathcal{A}_l \cup \mathcal{A}_z.$$

The set $\mathcal{A}_b$ of basic axioms is given by the labels of $\mathfrak{T}$:

$$\mathcal{A}_b \quad = \quad \{\gamma = \lambda \mid \gamma = \lambda \text{ is a label of a node in } \mathfrak{T}\}.$$

The sets $\mathcal{A}_l, \mathcal{A}_z$ are defined from $\mathfrak{T}$. To define $\mathcal{A}_l$ we introduce for each L-root **n** a tableau $\mathfrak{T}^{\mathbf{n}}$ obtained by performing some relabeling operation on the local tableau with the root **n**:

1. Let **n** be an L-root whose associated local tableau does not contain any local tableau, and let $\gamma\sigma'\sigma_1 = \lambda\sigma'\sigma_1$ be its label and $\gamma\sigma'\sigma_0\sigma_1 = \lambda\sigma'\sigma_0\sigma_1$ be the label of its parent. Let $\mathfrak{T}^{\mathbf{n}}$ be constructed from the local tableau rooted at **n** by replacing consistently the label $\eta\sigma'\sigma_1 = \kappa\sigma'\sigma_1$ of a node of the local tableau by the label $\eta\sigma'\sigma_0\sigma_1 = \kappa\sigma'\sigma_0\sigma_1$. When this is done we say that the L-root $\gamma\sigma'\sigma_1 = \lambda\sigma'\sigma_1$ and its associated local tableau have been *lifted* to its ambient tableau. Notice that after relabeling the L-root and its parent have the same label. We therefore coerce these two nodes in the ambient tableau.

2. In the inductive step let **m** be an L-root $\gamma'\sigma''\sigma_0'\sigma_1' = \lambda'\sigma''\sigma_0'\sigma_1'$ that has not yet been lifted and that contains only L-roots that have been lifted. Let $\mathbf{m}_1, \ldots, \mathbf{m}_k$ be all the top local tableaux, properly relabeled in previous steps, inside the local tableau rooted at **m**. By a top local tableau inside the local tableau rooted at **m** we mean that the former is not inside another local tableau inside the latter. Now replace in $\mathfrak{T}$ the local tableaux rooted at $\mathbf{m}_1, \ldots, \mathbf{m}_k$ by $\mathfrak{T}^{\mathbf{m}_1}, \ldots, \mathfrak{T}^{\mathbf{m}_k}$ respectively. Again the parent of $\mathbf{m}_i$, an L-node, for each $i \in \{1, \ldots, k\}$, is coerced with the root of $\mathfrak{T}^{\mathbf{m}_i}$. Now we apply the relabeling operation as described in step 1 to the resulting local tableau rooted at **m**. What we get is $\mathfrak{T}^{\mathbf{m}}$.

Let $\mathcal{A}_{\mathbf{n}}$ denote all the labels of $\mathfrak{T}^{\mathbf{n}}$. Now $\mathcal{A}_l$ is defined by

$$\mathcal{A}_l \quad = \quad \bigcup_{\mathbf{n} \text{ is an L node}} \mathcal{A}_{\mathbf{n}}.$$

The set $\mathcal{A}_z$ is defined by

$$\mathcal{A}_z \quad = \quad \left\{ V\sigma = \theta\sigma, \theta\sigma = \sigma \;\middle|\; \begin{array}{l} V\sigma = \sigma \text{ is in } \mathcal{A}_b, \text{ and } V \overset{\tau}{\Longrightarrow} \theta \overset{\tau}{\Longrightarrow} \epsilon \\ \text{is a chosen shortest path from } V \text{ to } \epsilon. \end{array} \right\}.$$

According to Lemma 12, it is sufficient to show that $\mathcal{A}$ is a bisimulation base. The proof is by induction on the nodes of the tableau starting with leaves.

- The axioms $\varsigma = \varsigma$ obviously satisfy the property of bisimulation base.
- The axiom $X = \epsilon$ satisfies the bisimulation base property using $\mathcal{A}_z$.

- Suppose $X_1 \dots X_i\eta = Y_1 \dots Y_j\kappa$ is a node to which Decmp rule is applied:

$$\frac{X_1 \dots X_i\eta = Y_1 \dots Y_j\kappa}{\eta = \kappa \quad X_1\eta = \eta \ \dots \ X_i\eta = \eta \quad Y_1\kappa = \kappa \ \dots \ Y_j\kappa = \kappa}.$$

  Suppose we have the following transition sequence from $X_1 \dots X_i\eta$:

$$X_1 \dots X_i\eta \longrightarrow \eta_1 \longrightarrow \dots \longrightarrow \eta_{k-1} \longrightarrow \eta_k \xrightarrow{\ell} \eta'. \qquad (1)$$

  We prove that (1) can be simulated by a transition sequence from $X_1 \dots X_{i-1}\eta$ that satisfies the bisimulation base property. The last action ℓ could be caused by any one of $X_1, \dots, X_i, \eta$. Suppose (1) is $X_1 \dots X_i\eta \longrightarrow \dots \longrightarrow X_i\eta \overset{\tau}{\Longrightarrow} \eta \xrightarrow{\ell} \eta'$. Now $X_1 \dots X_i\eta \longrightarrow \dots \longrightarrow X_i\eta$ is simulated by $X_1 \dots X_{i-1}\eta \longrightarrow \dots \longrightarrow \eta$ trivially. The transition $X_i\eta \overset{\tau}{\Longrightarrow} \eta$ can be simulated by η vacuously using $\mathcal{A}_z$. And the transition $\eta \xrightarrow{\ell} \eta'$ is simulated of course by $\eta \xrightarrow{\ell} \eta'$. For another case suppose (1) is $X_1 \dots X_i\eta \longrightarrow \dots \longrightarrow X_i\eta \longrightarrow X_i'\eta \xrightarrow{\ell} X_i''\eta$. Here the simulating sequence from $X_1 \dots X_{i-1}\eta$ can be easily produced using induction hypothesis on $X_i\eta = \eta$. Finally if the action ℓ is caused by one of $X_1, \dots, X_{i-1}$, the simulating sequence is trivial. In summary (1) can be simulated by a transition sequence from $X_1 \dots X_{i-1}\eta$ in a way that satisfies the defining property of bisimulation base.

  For similar reason a transition sequence $X_1 \dots X_{i-1}\eta \longrightarrow \dots \longrightarrow\xrightarrow{\ell} \eta''$ can be simulated by a transition from $X_1 \dots X_{i-2}\eta$, and so on and so fourth. By putting all these simulations together we conclude that (1) must be simulated by a transition sequence from η that satisfies the bisimulation base property. The induction hypothesis on $\eta = \kappa$ produces a simulating sequence from κ. This simulating sequence can be simulated by $Y_1 \dots Y_j\kappa$ by the transition sequence $Y_1 \dots Y_j\kappa \overset{\tau}{\Longrightarrow} \kappa$, permitted by A_z, followed by the transition sequence from κ.

- Suppose $\gamma\alpha = \lambda\beta$ is a node to which Match rule is applied. Let $\gamma\alpha \longrightarrow \dots \longrightarrow \alpha'' \xrightarrow{\ell} \alpha'$ be a transition sequence from $\gamma\alpha$. By the definition of Match the whole transition sequence or part of the transition sequence can be simulated in the relation $\mathcal{A}_b$. Assume that $\gamma\alpha \longrightarrow \dots \longrightarrow \alpha_0\alpha \longrightarrow \dots \longrightarrow \alpha'' \xrightarrow{\ell} \alpha'$ is the transition sequence and that $\gamma\alpha \longrightarrow \dots \longrightarrow \alpha_0\alpha$ can be simulated by $\lambda\beta \longrightarrow \dots \longrightarrow \beta_0\beta$ in $\mathcal{A}_b$. By induction hypothesis the transition sequence $\alpha_0\alpha \longrightarrow \dots \longrightarrow \alpha'' \xrightarrow{\ell} \alpha'$ can be simulated by a transition sequence from $\beta_0\beta$.

- If $\alpha = \beta \in \mathcal{A}_b$ is a node to which SubstL/SubstR/ContrL/ContrR/ContrC is applied, we may use simple induction to establish the bisimulation base property.

- Now we prove that the nodes in $\mathcal{A}_l$ satisfy the bisimulation base property. Suppose **l** is an L-root whose associated local tableau contains no local tableaux. It is easy to show that the pairs in $\mathcal{A}_{\mathbf{l}}$ satisfy the bisimulation base property, assuming that all its leaves satisfy the bisimulation base property. The inductive step can be proved as follows. Suppose $\mathbf{n}_0$ is an L-root labeled $\gamma\sigma'\sigma_1 = \lambda\sigma'\sigma_1$ and its parent $\mathbf{m}_0$, an L-node, is labeled by $\gamma\sigma'\sigma_0\sigma_1 = \lambda\sigma'\sigma_0\sigma_1$. A node in $\mathcal{A}_{\mathbf{n}_0}$ is of the form $\delta\sigma\sigma_0\sigma_1 = \theta\sigma\sigma_0\sigma_1$. The pair $(\delta\sigma\sigma_0\sigma_1, \theta\sigma\sigma_0\sigma_1)$ copycats the bisimulation strategy of $(\delta\sigma\sigma_1, \theta\sigma\sigma_1)$ until it reaches a subgoal of the form $(X_i\sigma_0\sigma_1, \sigma_0\sigma_1)$ for X_i in the R-set of $\mathbf{m}_0$. The subdoal can bisimulate each other by construction. This argument relies on the property that bisimulation of $(\delta\sigma\sigma_1, \theta\sigma\sigma_1)$ keeps σ_1 intact until it reaches subgoals of the form $(X_i\sigma_1, \sigma_1)$ for X_i in the R-set of $\mathbf{m}_0$. It is guaranteed by the consistency condition imposed on the local tableau construction. So the bisimulation base property also hold for the pairs in $\mathcal{A}_{\mathbf{m}_0}$.

To give a flavor of what really happens, let's see the above argument in more detail. Suppose in the local tableau rooted at $\mathbf{n}_0$ there is an L-root $\mathbf{n}_1$ labeled by $\gamma'\sigma''\sigma_3 = \lambda'\sigma''\sigma_3$. The parent node $\mathbf{m}_1$ of the L-root, an L-node, is labeled by $\gamma'\sigma'\sigma_0'\sigma_1 = \lambda'\sigma_0'\sigma'\sigma_1$. The tree structure is indicated by the following

$$\dfrac{\mathbf{m}_0 : \ \gamma\sigma'\sigma_0\sigma_1 = \lambda\sigma'\sigma_0\sigma_1}{\{X_i\sigma_1 = \sigma_1\}_{i\in I} \quad \mathbf{n}_0 : \ \gamma\sigma'\sigma_1 = \lambda\sigma'\sigma_1 \quad \{X_i\sigma_0\sigma_1 = \sigma_0\sigma_1\}_{i\in I}}$$
$$\vdots$$
$$\dfrac{\mathbf{m}_1 : \ \gamma'\sigma_0'\sigma'\sigma_1 = \lambda'\sigma_0'\sigma'\sigma_1}{\{X_{i'}\sigma_3 = \sigma_3\}_{i'\in I'} \quad \mathbf{n}_1 : \ \gamma'\sigma''\sigma_3 = \lambda'\sigma''\sigma_3 \quad \{X_{i'}\sigma_2\sigma_3 = \sigma_2\sigma_3\}_{i'\in I'}}$$

where $\sigma_0'\sigma'\sigma_1 = \sigma''\sigma_2\sigma_3$. By induction all nodes in $\mathfrak{T}^{\mathbf{m}_1}$ satisfy the bisimulation base property. The root of $\mathfrak{T}^{\mathbf{m}_1}$ is $\gamma'\sigma''\sigma_2\sigma_3 = \lambda'\sigma''\sigma_2\sigma_3$, which is syntactically the same as $\gamma'\sigma_0'\sigma'\sigma_1 = \lambda'\sigma_0'\sigma'\sigma_1$. The nodes in $\mathfrak{T}^{\mathbf{m}_1}$ must be relabeled in $\mathfrak{T}^{\mathbf{m}_0}$. We need to prove that the nodes in $\mathfrak{T}^{\mathbf{m}_1}$ still satisfy the bisimulation base property after they are lifted into $\mathfrak{T}^{\mathbf{m}_0}$. A node in $\mathfrak{T}^{\mathbf{m}_1}$ lifted from $\mathfrak{T}^{\mathbf{m}_0}$ must be of the form

$$\gamma''\sigma'''\sigma_0'\sigma'\sigma_0\sigma_1 = \lambda''\sigma'''\sigma_0'\sigma'\sigma_0\sigma_1. \tag{2}$$

We show that (2) satisfies the bisimulation base property. This can be demonstrated by mimicking bisimulations in the local tableaux. We look at two cases:

- $|\sigma_1| > |\sigma_3|$. Let $\sigma_{2,1}, \sigma_{1,3}$ be such that $\sigma_2 = \sigma_{2,1}\sigma_{1,3}$ and $\sigma_1 = \sigma_{1,3}\sigma_3$. According to the lift construction, the pair in (2) can copycat the bisimulation of $\gamma''\sigma'''\sigma''\sigma_3 = \lambda''\sigma'''\sigma''\sigma_3$ until it is decomposed into subgoals of the form $X_{i'}\sigma_{2,1}\sigma_0\sigma_1 = \sigma_{2,1}\sigma_0\sigma_1$ for $i' \in I'$. The subgoal $X_{i'}\sigma_{2,1}\sigma_0\sigma_1 = \sigma_{2,1}\sigma_0\sigma_1$ imitates $X_{i'}\sigma_2\sigma_3 = \sigma_2\sigma_3$ until it reaches further subgoals of the form $X_i\sigma_0\sigma_1 = \sigma_0\sigma_1$. The bisimulation base property is satisfied by the pair $(X_i\sigma_0\sigma_1, \sigma_0\sigma_1)$ by construction. Notice that the consistency condition is needed to keep clear for example the boundary at $\sigma_{2,1}$ and $\sigma_{1,3}$.
- $|\sigma_1| < |\sigma_3|$. Let $\sigma_{3,1}$ be such that $\sigma_3 = \sigma_{3,1}\sigma_1$. The pair in (2) can copycat the bisimulation of $\gamma''\sigma'''\sigma''\sigma_3 = \lambda''\sigma'''\sigma''\sigma_3$ until it is decomposed into subgoals of the form $X_{i'}\sigma_2\sigma_{3,1}\sigma_0\sigma_1 = \sigma_2\sigma_{3,1}\sigma_0\sigma_1$ for $i' \in I'$. The latter pair then bisimulate each other like the pair $(X_{i'}\sigma_2\sigma_3, \sigma_2\sigma_3)$ until it is decomposed into further subgoals of the form $(X_i\sigma_0\sigma_1, \sigma_0\sigma_1)$ for $i \in I$. Now $(X_i\sigma_0\sigma_1, \sigma_0\sigma_1)$ satisfies the bisimulation base property by definition. Again the consistency condition is necessary here.

- Finally let's consider the axioms in $\mathcal{A}_z$. Suppose $V\sigma = \sigma \in \mathcal{A}_b$ and $V\sigma \overset{\tau}{\Longrightarrow} \theta\sigma \overset{\tau}{\Longrightarrow} \sigma$ with $V\sigma = \theta\sigma, \theta\sigma = \sigma \in \mathcal{A}_z$. A transition sequence $\theta\sigma \Longrightarrow\xrightarrow{\ell} \gamma$ is simulated by $V\sigma \xrightarrow{\tau} \theta_0\sigma \xrightarrow{\tau} \theta_1\sigma \xrightarrow{\tau} \ldots \xrightarrow{\tau} \theta_k\sigma \xrightarrow{\tau} \theta\sigma \Longrightarrow\xrightarrow{\ell} \gamma$. Notice that $\mathcal{A}_z \vdash V\sigma = \theta_0\sigma = \sigma = \theta\sigma, \ldots, \mathcal{A}_z \vdash V\sigma = \theta_k\sigma = \sigma = \theta\sigma$. The transition sequence $V\sigma \xrightarrow{\tau} \theta_0\sigma \xrightarrow{\tau} \theta_1\sigma \xrightarrow{\tau} \ldots \xrightarrow{\tau} \theta_k\sigma \xrightarrow{\tau} \theta\sigma \Longrightarrow\xrightarrow{\ell} \gamma$ can be simulated by a transition sequence from σ by induction on $V\sigma = \sigma$. We conclude that both $V\sigma = \theta\sigma$ and $\theta\sigma = \sigma$ satisfy the bisimulation base property.

We have verified that $\mathcal{A}$ is indeed a bisimulation base. It follows from Lemma 12 that $\mathcal{A}$ is part of a branching bisimulation. Consequently $X\alpha \simeq Y\beta$. □

We are now in a position to establish the main result.

Theorem 31. *The branching bisimilarity on normed BPA processes is decidable.*

Proof. The equality between an nBPA process γ and ϵ is decidable. Given two nBPA processes $X\alpha$ and $Y\beta$, our algorithm constructs all tableaux for $X\alpha = Y\beta$ in parallel. At the same time it keeps checking if a successful tableau has been generated. By Proposition 30 the algorithm terminates with a 'yes' answer if and only if $X\alpha \simeq Y\beta$. Since by Lemma 28 there is an effective bound on the number of tableaux for $X\alpha = Y\beta$, the algorithm is capable of answering 'no' if no successful tableau exists. □

We remark that in the presence of Proposition 27 one gets decidability as long as every tableau is finite.

5 Regularity Checking

Given a process α defined in a normed BPA Δ, the *bisimulation class* $[\alpha]$ represented by α is the set $\{\alpha' \mid \alpha' \simeq \alpha\}$. A bisimulation class of Δ is a bisimulation class represented by some process defined in Δ. We say that the BPA Δ is a *finite state system* if the set of bisimulation classes of Δ is finite. We say that α is a *finite state process* if the set of the bisimulation classes represented by the nodes of $\mathcal{T}^\alpha$ is finite. It is an *infinite state process* otherwise. The *regularity problem* of branching bisimilarity for normed BPA asks if a normed BPA process is a finite state process.

The regularity problem for the strong bisimilarity on normed BPA has been settled. Kučera [Kuč96] showed that it is decidable in polynomial time. Srba [Srb02a] observed that it is actually solvable in nondeterministic logarithmic space. In the same paper he also proved that the problem is NL-hard using a reduction from the reachability problem of DAG. The decidability of the regularity problem for the strong bisimilarity on general BPA was proved by Burkart, Caucal and Steffen [BCS95, BCS96]. The problem was shown to be PSPACE-hard by Srba [Srb02a] by reducing from the Quantified Boolean Formula Problem. In the presence of silent actions the picture is far less clear. To the best of our knowledge the decidability of almost all regularity problems of weak/branching bisimilarity, in the setting of process rewriting system [May00], are unknown. The only exception is the undecidability result of the regularity problem for weak bisimilarity of Petri Net, and its extension, established

	BPA	Normed **BPA**
$\sim$	Decidable [BCS95, BCS96] PSPACE-hard [Srb02a]	NL-complete [Srb02a][Kuč96]
$\simeq$	? EXPTIME-hard [May03]	? EXPTIME-hard [May03]
$\approx$	? EXPTIME-hard [May03]	? EXPTIME-hard [May03] NP-hard [Stř98, Srb03]

Figure 4: Regularity of BPA

by Jancar and Esparza [JE96]. This is interesting in view of the fact that the regularity problem for strong bisimilarity of Petri Net is decidable [JE96]. For the totally normed BPA Stříabrna [Stř98] proved that the regularity of weak bisimilarity is NP-hard. Srba [Srb03] improved the result by showing that the problem is both NP-hard and co-NP-hard for normed BPA. Mayr proved that the weak bisimilarity of weakly regular BPA processes is EXPTIME-hard [May03]. A BPA process is weakly regular if it is weakly bisimilar to a finite state. By combining Srba's polynomial reduction from the weak bisimilarity of weakly regular processes to the regularity problem of weak bisimilarity [Srb03], one concludes that the latter is EXPTIME-hard. Mayr's proof of the EXPTIME lower bound [May03] can be modified so that it applies to branching bisimilarity as well. We omit the details here. A summary of these results is given in Fig. 4.

In this section we show that the regularity problem of branching bisimilarity for normed BPA is decidable. We begin with a proof that a related but slightly different problem is decidable.

Proposition 32. *It is decidable to check if a normed BPA Δ defines a finite state system.*

Proof. Let Δ be a normed BPA with n variables. We need to prove that the set of all bisimulation classes of processes defined in Δ is finite. For every process defined by Δ of length 2^n+1, we can algorithmically check if it is completely irredundant. If no such process is completely irredundant, then Δ is a finite state system. Otherwise let κ be such a process. Now there must exist processes $\sigma'', \sigma, \sigma'$ such that $\kappa = \sigma''\sigma\sigma'$ and the redundancy set of $\sigma\sigma'$ is the same as the redundancy set of σ'. The inequality $\sigma\sigma' \not\simeq \sigma'$ holds by the complete irredundancy of κ. Moreover $\sigma\sigma\sigma' \not\simeq \sigma\sigma'$ for otherwise Corollary 22 would imply $\sigma\sigma' \simeq \sigma'$. Meanwhile we also have $\sigma\sigma\sigma' \not\simeq \sigma'$ since $\|\sigma\sigma\sigma'\|_b > \|\sigma\sigma'\|_b > \|\sigma'\|_b$. By induction we can prove that $\sigma^{i+1}\sigma' \not\simeq \sigma^i\sigma'$ and $\|\sigma^{i+1}\sigma'\|_b > \|\sigma^i\sigma'\|_b$ for all $i \geq 0$, where σ^i stands for $\underbrace{\sigma\ldots\sigma}_{i\text{ times}}$. It follows that $\sigma^i\sigma' \not\simeq \sigma^j\sigma'$ whenever $i \neq j$. Therefore Δ defines an infinite state system in this case. □

Proposition 32 does not imply the decidability of the regularity problem. This is because even if a normed BPA defines an infinite state system, a process defined in the BPA can still be a finite state process. To prove the general result we need the following technical lemma.

Lemma 33. *For normed BPA there is an algorithm to calculate the function $\|\_\|_b$.*

Proof. Let α be a process defined by a normed BPA. Consider the transition tree $\mathcal{T}^{\alpha}$. This tree is in general infinite. We can trim this tree down to a finite tree by cutting off a path at some point. The strategy is as follows:

- If $\alpha \xrightarrow{\ell_1} \alpha_1 \ldots \xrightarrow{\ell_k} \alpha_k$ contains $\|\alpha\|$ external actions plus change-of-state silent actions, then remove all the descendants of α_k.
- Suppose $\alpha \xrightarrow{\ell_1} \alpha_1 \ldots \xrightarrow{\ell_k} \alpha_k \to \alpha_{k+1} \to \ldots \to \alpha_h$ is a path in the tree. If $\alpha \xrightarrow{\ell_1} \alpha_1 \ldots \xrightarrow{\ell_k} \alpha_k$ contains at most $\|\alpha\| - 1$ external actions plus change-of-state silent actions and the length of $\alpha_k \to \alpha_{k+1} \to \ldots \to \alpha_h$ is H_{α_k}, then remove all the descendants of α_h.

The first rule is based on the fact $\|\alpha\|_b \leq \|\alpha\|$ and the second rule on Lemma 25. It is now easy to calculate $\|\alpha\|_b$ from the finite tree using Theorem 31. □

It is interesting to notice that, unlike the situation for the strong bisimilarity, it is after the proof of the decidability of $\simeq$ that we are able to show the effectiveness of $\|\_\|_b$. By Lemma 33 the value $\|\Delta\|_b$ is effectively calculable.

In what follows we provide two semi-decidable procedures that would lead to the decidability of the regularity problem. The first semi-decidable procedure allows one to answer positively when the input nBPA process is a finite state process.

Lemma 34. *It is semi-decidable to check if an nBPA process is a finite state process.*

Proof. Suppose α is a normed BPA process. Construct the *branching transition tree* with root label α as follows:

- The tree is constructed in a width first fashion. Suppose β is a leaf of the current tree that is not marked as a leaf of the final tree. Suppose further that $\beta \to^* \beta' \xrightarrow{\ell} \gamma$ is such that the length of $\beta \to^* \beta'$ is less than H_β. Then the node β has a child labeled γ and the edge is labeled by ℓ.
- Whenever a new leaf is generated whose label is branching bisimilar to the label of some inner node of the current tree, then that leaf is marked as a leaf of the final tree. An inner node of the tree is a node that is not a leaf.
- A leaf labeled by ϵ is marked as a leaf of the final tree.

If α is a finite state process then clearly the branching transition tree is finite. Conversely suppose the branching transition tree is finite. Let $\alpha \to^* \xrightarrow{\ell_1} \alpha_1 \ldots \to^* \xrightarrow{\ell_k} \alpha_k$ be a transition sequence such that each α_i with $1 \leq i < k$ is branching bisimilar to some node of the branching transition tree and α_k is not branching bisimilar to any node of the tree. Clearly there must be some node in the sequence $\alpha \to^* \xrightarrow{\ell_1} \alpha_1 \ldots \to^* \xrightarrow{\ell_k} \alpha_k$, say α_{j_1}, that is a leaf of the tree. By definition there must be some inner node β_1 such that $\alpha_{j_1} \simeq \beta_1$. It follows that there is some α_{j_2} such that $\alpha_{j_1} \to^* \xrightarrow{\ell_{j_1+1}} \ldots \to^* \xrightarrow{\ell_{j_2}} \alpha_{j_2}$ is bisimulated by some transition sequence $\beta_1 \to^* \xrightarrow{\ell_{j_1+1}} \ldots \to^* \xrightarrow{\ell_{j_2}} \beta_2$ with the latter sequence satisfying the following:

- $\beta_1 \longrightarrow \ldots \longrightarrow \beta_2$ is a path in the branching transition tree with β_2 being a leaf.
- the edges are labeled respectively by $\ell_{j_1+1}, \ldots, \ell_{j_2}$.

Now β_2 is branching bisimilar to some inner node of the tree. So we may repeat the above argument. Eventually we get some node of the tree that is branching bisimilar to α_k, which is a contradiction. We conclude that every process reachable from α is branching bisimilar to some node in the finite tree. The construction of the branching transition tree is effective due to Lemma 26 and Theorem 31. Hence the semi-decidability of the procedure. □

To simplify the account of the other semi-decidable procedure we introduce the following technical lemma.

Lemma 35. *Suppose $V_0\sigma_0$ is defined in a normed BPA Δ with n variables and $V_0\sigma_0 \xrightarrow{\ell_1} V_1\sigma_1 \xrightarrow{\ell_2} V_2\sigma_2 \ldots \xrightarrow{\ell_i} V_m\sigma_m$ for some $m > n2^n$ such that $|\sigma_0| \leq |\sigma_1| \leq \ldots \leq |\sigma_m|$. If there are $n2^n + 1$ processes $V_{i_0}\sigma_{i_0}, V_{i_1}\sigma_{i_1}, \ldots, V_{i_{n2^n}}\sigma_{i_{n2^n}}$, where $i_0 < i_1 < \ldots < i_{n2^n}$, such that $\|V_{i_0}\sigma_{i_0}\|_b < \|V_{i_1}\sigma_{i_1}\|_b < \ldots < \|V_{i_{n2^n}}\sigma_{i_{n2^n}}\|_b$, then $V_0\sigma_0$ is an infinite state process.*

Proof. By assumption there must exist $j, j' \in \{0, \ldots, n2^n\}$ such that $j < j'$, $V_{i_j} = V_{i_{j'}}$ and $\mathcal{R}_{\sigma_{i_j}} = \mathcal{R}_{\sigma_{i_{j'}}}$. Now $\sigma_{i_{j'}} = \sigma\sigma_{i_j}$ for some σ since $|\sigma_{i_j}| \leq |\sigma_{i_{j'}}|$. Moreover $\|\sigma\|_b^{\sigma_{i_j}} > 0$ due to the inequality $\|V_{i_j}\sigma_{i_j}\|_b < \|V_{i_j}\sigma_{i_{j'}}\|_b$. Clearly $V_{i_j}\sigma^k\sigma_{i_j}$ is a descendant of $V_0\sigma_0$ for all $k \geq 0$. By the proof of Lemma 32 we conclude that $V_0\sigma_0$ is an infinite state process. □

Lemma 36. *There is a semi-decidable procedure to check if a normed BPA process is an infinite state process.*

Proof. Suppose α is an infinite state process defined by a normed BPA Δ with n variables. There must be an infinite path $\alpha \xrightarrow{\ell_1} \alpha_1 \xrightarrow{\ell_2} \alpha_2 \ldots \xrightarrow{\ell_i} \alpha_i \ldots$ such that the norms of the processes $\alpha, \alpha_1, \ldots, \alpha_i, \ldots$ are unbounded. Assume that $\alpha = V_0\sigma_0$ for some V_0, σ_0 and $\alpha_i = V_i\sigma_i$ for some V_i, σ_i for each $i > 0$. The infinite transition sequence can be rewritten as

$$V_0\sigma_0 \xrightarrow{\ell_1} V_1\sigma_1 \xrightarrow{\ell_2} V_2\sigma_2 \ldots \xrightarrow{\ell_i} V_i\sigma_i \ldots. \quad (3)$$

Choose from (3) a finite subsequence

$$V_j\sigma_j \xrightarrow{\ell_{j+1}} V_{j+1}\sigma_{j+1} \xrightarrow{\ell_{j+2}} V_{j+2}\sigma_{j+2} \ldots \xrightarrow{\ell_k} V_k\sigma_k \quad (4)$$

such that the following are satisfied:

$$\forall j' \in \{j+1, \ldots, k\}.|\sigma_j| < |\sigma_{j'}|, \quad (5)$$

$$\|V_k\sigma_k\|_b > \|V_j\sigma_j\|_b + n^2 2^{2n}\|\Delta\|_b r_\Delta. \quad (6)$$

By Lemma 33 condition (6) is effectively checkable. Condition (5) implies that σ_j is a proper suffix of $\sigma_{j'}$ for all $j' \in \{j+1, \ldots, k\}$. We call $V_p\sigma_p$, for $p > j$, a *minimal* *turning point* if there exists some p' such that $j < p' \leq p$ and $|\sigma_{p'-1}| > |\sigma_{p'}| = \ldots = |\sigma_p| < |\sigma_{p+1}|$. Let $V_{j_1}\sigma_{j_1}$ be a minimal turning point such that for any other minimal turning point $V_p\sigma_p$ the inequality $|\sigma_{j_1}| \leq |\sigma_p|$ holds and if $|\sigma_{j_1}| = |\sigma_p|$ then $j_1 > p$. Since σ_j is a suffix of σ_{j_1}, there must be a maximal initial segment

$$V_j\sigma_j \xrightarrow{\ell_j} V_{j+1}\sigma_{j+1} \ldots \xrightarrow{\ell_{j_1'}} V_{j_1'}\sigma_{j_1'} \tag{7}$$

of (4) that enjoys the property $|\sigma_j| \leq |\sigma_{j+1}| \leq \ldots \leq |\sigma_{j_1'}|$. If there are $n2^n + 1$ processes in (7), say $V^1\sigma^1, \ldots, V^{n2^n+1}\sigma^{n2^n+1}$, such that $\|V^1\sigma^1\|_b < \ldots < \|V^{n2^n+1}\sigma^{n2^n+1}\|_b$, then we conclude from Lemma 35 that α is an infinite state process. Otherwise we derive from the fact $|\sigma_j| < |\sigma_{j_1}| < |\sigma_{j_1'}|$ that there is an initial segment $V_j\sigma_j \xrightarrow{\ell_j} V_{j+1}\sigma_{j+1} \ldots \xrightarrow{\ell_{j_1''}} V_{j_1''}\sigma_{j_1''} \xrightarrow{\ell_{j_1''+1}} V_{j_1''+1}\sigma_{j_1''+1}$ of (7) that meets the following condition

$$|\sigma_{j_1''}| \leq |\sigma_{j_1}| < |\sigma_{j_1''+1}|. \tag{8}$$

Notice that $|\sigma_{j_1}| = |\sigma_{j_1'}|$ is impossible by the way σ_{j_1} is chosen. So the right inequality in (8) must be strict. By normedness one has that

$$V_j\sigma_j \xrightarrow{\ell_j} V_{j+1}\sigma_{j+1} \ldots \xrightarrow{\ell_{j_1''}} V_{j_1''}\sigma_{j_1''} \xrightarrow{\ell_{j_1''+1}} V_{j_1''+1}\sigma_{j_1''+1} \xrightarrow{\ell^*} \sigma_{j_1}$$

for some ℓ^*. Clearly

$$\begin{aligned}
\|V_{j_1}\sigma_{j_1}\|_b - \|V_j\sigma_j\|_b &= \|V_{j_1}\|_b^{\sigma_{j_1}} + \|\sigma_{j_1}\|_b - \|V_j\sigma_j\|_b \\
&\leq \|V_{j_1}\|_b^{\sigma_{j_1}} + \|V_{j_1''+1}\sigma_{j_1''+1}\|_b - \|V_j\sigma_j\|_b \\
&\leq \|V_{j_1}\|_b^{\sigma_{j_1}} + \|V_{j_1'}\sigma_{j_1'}\|_b - \|V_j\sigma_j\|_b \\
&\leq \|\Delta\|_b + (n2^n - 1)\|\Delta\|_b r_\Delta \\
&\leq n2^n\|\Delta\|_b r_\Delta,
\end{aligned}$$

using the assumption that there are at most $(n2^n - 1)$ transitions in (7).

The above construction can be repeated and we eventually get a transition sequence $V_j\sigma_j \xrightarrow{\ell_1^*} V_{j_1}\sigma_{j_1} \xrightarrow{\ell_2^*} V_{j_2}\sigma_{j_2} \ldots \xrightarrow{\ell_m^*} V_{j_m}\sigma_{j_m}$ such that $|\sigma_j| \leq |\sigma_{j_1}| \leq \ldots \leq |\sigma_{j_m}|$ and

$$\|V_j\sigma_j\|_b \leq \|V_{j_1}\sigma_{j_1}\|_b \leq \ldots \leq \|V_{j_m}\sigma_{j_m}\|_b. \tag{9}$$

It follows from condition (6) that $m > n2^n$ and that there are at least $n2^n$ strict inequalities in (9). So we can apply Lemma 35 anyway.

The existence of a transition sequence described above allows one to design a semi-decidable procedure. Given a normed BPA process α the semi-decidable program constructs the transition tree $\mathcal{T}_\alpha$ in a width first fashion and keeps checking if there is a sub-path of a path in the tree satisfying the property described in (5) and (6). If α is an infinite state process then the procedure is bound to find such a sub-path and gives a positive answer. □

In summary we have the following decidability result.

Theorem 37. *The regularity problem of branching bisimilarity on nBPA is decidable.*

6 Remark

For parallel processes, basic parallel processes (BPP) and Petri nets (PN), in which silent transitions are treated as unobservable the only known decidability result concerning equivalence checking is due to Czerwiński, Hofman and Lasota. They have proved in [CHL11] that branching bisimilarity on normed BPP processes is decidable. Their proof is based on a novel technique that reduces every normed BPP process to a normal form. The norm form for a bisimulation class is an element in the class that is minimal with regards to lexicographic order. Other equivalence checking problems about parallel processes are either open or undecidable. This paper provides the first decidability result for the sequential processes, basic process algebras (BPA) and push-down automata (PDA), in which silent transitions are treated as unobservable. The technique used to establish this result is more traditional compared to the one used by Czerwiński, Hofman and Lasota. For further research one could try to apply the technique developed in this paper to the general BPA. One could also try to adapt the methodology to investigate the decidability issue of branching bisimilarity of PDA. The latter problem is particularly interesting in the light of the facts that the strong bisimilarity on PDA is decidable [Sén98, Sti98] whereas the weak bisimilarity on normed PDA is undecidable [JS08]. If either problem turns out to be decidable, the associated regularity problem would then invite immediate study.

Currently we do not see how the techniques used in this paper can be transplanted to weak bisimilarity [Mil89a]. Neither Lemma 18 nor Corollary 22 is known to hold for weak bisimilarity, although Lemma 20 is valid for weak bisimilarity on normed BPA processes. Without Lemma 18 we are not able to establish the finite tree property. And without Corollary 22 we cannot control the size of a tableau. Further investigation is necessary before we can say more about the decidability of the weak bisimilarity.

This paper is a first step in looking for upper bound on branching bisimilarity of normed BPA processes. In next step one could try to answer the following question: Is the problem elementary? In view of Kiefer's result [Kie12] one expects that the EXPTIME lower bound to be improved. At this point it seems appropriate to mention a recent result of Benedikt, Moller, Kiefer and Murawski [BMKM12]. They have proved that strong bisimilarity on normed PDA is non-elementary.

Acknowledgement. I would like to thank the members of BASICS for their interest and valuable discussions. One of the discussions has convinced me that the reduction described in [May03] can be improved to show that branching bisimilarity of nBPA and its associated regularity problem are EXPTIME-hard. In particular my thanks go to Chaodong He for bringing up the issue of transitivity of bisimulation base and for his expert proof-reading, to Mingzhang Huang for informing me of Example 8 and for pointing out a mistake in my earlier formulation of Localization rule, to Huan Long and Qiang Yin for thought provoking discussions, and to Zhimo Shen and Fei Yang for questioning my previous proof of the decidability of the regularity problem. Last but not least I would like to thank Xinxin Liu for pointing out an omission in my previous proof of Lemma 25. The support from NSFC (60873034, 61033002) is gratefully acknowledged.

References

[BBK87] J. Baeten, J. Bergstra, and J. Klop. Deciding of bisimulation equivalence for processes generating context-free languages. pages 94–113. Lecture Notes in Computer Science 259, 1987.

[BBK93] J. Baeten, J. Bergstra, and J. Klop. Deciding of bisimulation equivalence for processes generating context-free languages. *J. ACM*, 40:653–682, 1993.

[BCMS01] O. Burkart, D. Caucal, F. Moller, and B. Steffen. Verification on infinite structures. In J. Bergstra, A. Ponse, and S. Smolka, editors, *Handbook of Process Algebra*, pages 545–623. North-Holland, 2001.

[BCS95] O. Burkart, D. Caucal, and B. Steffen. An elementary bisimulation decision procedure for arbitrary context free processes. *MFCS'95*, pages 423–433, 1995.

[BCS96] O. Burkart, D. Caucal, and B. Steffen. Bisimulation collapse and the process taxonomy. *CONCUR 1996*, pages 247–262, 1996.

[BGS92] J. Balcazar, J. Gabarro, and M. Santha. Deciding bisimilarity is p-complete. *Formal Aspects of Computing*, 4:638–648, 1992.

[BMKM12] M. Benedikt, S. Moller, S. Kiefer, and A. Murawski. Bisimilarity of pushdown systems is nonelementary. 2012.

[CHL11] W. Czerwiński, P. Hofman, and S. Lasota. Decidability of branching bisimulation on normed commutative context-free processes. In *CONCUR 2011*, pages 528–542. Lecture Notes in Computer Science 6901, Springer, 2011.

[CHS92] S. Christensen, H. Hüttel, and C. Stirling. Bisimulation equivalence is decidable for all context-free processes. pages 138–147. Lecture Notes in Computer Science 630, Springer, 1992.

[CHS95] S. Christensen, H. Hüttel, and C. Stirling. Bisimulation equivalence is decidable for all context-free processes. *Information and Computation*, 12:143–148, 1995.

[DNMV90] R. De Nicola, U. Mantanari, and F. Vaandrager. Back and forth bisimulations. In *Proc. CONCUR'90*, volume 458 of *Lecture Notes in Computer Science*, pages 152–165, 1990.

[H9̈1] H. Hüttel. Silence is golden: Branching bisimilarity is decidable for context free processes. pages 2–12. Lecture Notes in Computer Science 575, 1991.

[Hir96] Y. Hirshfeld. Bisimulation trees and the decidability of weak bisimulations. In *INFINITY'96*, 1996.

[HJM96] Y. Hirshfeld, M. Jerrum, and F. Moller. A polynomial algorithm for deciding bisimilarity of normed context free processes. *Theoretical Computer Science*, 158(1-2):143–159, 1996.

[HS91] H. Hüttel and C. Stirling. Actions speak louder than words: Proving bisimilarity for context free processes. pages 376–386, 1991.

[HT94] T. Huynh and L. Tian. Deciding bisimilarity of normed context free processes is in σ_2^p. *Theoretical Computer Science*, 123:83–197, 1994.

[HU79] J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley Publishing Company, 1979.

[Jan12] P. Jančar. Bisimilarity on basic process algebra is in 2-exptime. 2012.

[JE96] P. Jančar and J. Esparza. Deciding finiteness of petri nets up to bisimulation. *Automata, Languages and Programming*, pages 478–489, 1996.

[JM99] P. Jančar and F. Moller. Techniques for decidability and undecidability of bisimilarity. In *Concur'99*, volume 1664 of *Lecture Notes in Computer Science*, pages 30–45, 1999.

[JS08] P. Jančar and J. Srba. Undecidability of bisimilarity by defender's forcing. *Journal of ACM*, 55(1), 2008.

[Kie12] S. Kiefer. Bpa bisimilarity is exptime-hard. 2012.

[KJ06] A. Kučera and P. Jančar. Equivalence-checking on infinite state systems: Techniques and results. *Theory and Practice of Logic Programming*, 6:227–264, 2006.

[Kuč96] A. Kučera. Regularity is decidable for normed bpa and normed bpp processes in polynomial time. In *SOFSEM'96*, pages 377–384. Springer, 1996.

[May00] R. Mayr. Process rewrite systems. *Information and Computation*, 156:264–286, 2000.

[May03] R. Mayr. Weak bisimilarity and regularity of bpa is exptime-hard. In *EXPRESS'03*, 2003.

[Mil84] R. Milner. A complete inference system for a class of regular behaviours. *Journal of Computer and System Science*, 28:439–466, 1984.

[Mil89a] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[Mil89b] R. Milner. A complete axiomatization system for observational congruence of finite state behaviours. *Information and Computation*, 81:227–247, 1989.

[MSS04] F. Moller, S. Smolka, and J. Srba. On the computational complexity of bisimulation, redux. *Information and Computation*, 194:129–143, 2004.

[Pap94] C. Papadimitriou. *Computational Complexity*. Addison-Wesley, Reading, MA, 1994.

[Sén98] G. Sénizergues. Decidability of bisimulation equivalence for equational graphs of finite out-degree. In *Foundations of Computer Science, 1998. Proceedings. 39th Annual Symposium on*, pages 120–129. IEEE, 1998.

[Srb02a] J. Srba. Strong bisimilarity and regularity of basic parallel processes is pspace-hard. In *STACS 2002*, pages 733–733. Springer, 2002.

[Srb02b] J. Srba. Strong bisimilarity and regularity of basic process algebra is PSPACE-hard. In *ICALP'02*, LNCS 2380, pages 716–727, 2002.

[Srb03] J. Srba. Complexity of weak bisimilarity and regularity for BPA and BPP. *Mathematical Structures in Computer Science*, 13:567–587, 2003.

[Srb04] J. Srba. Roadmap of infinite results. In *Formal Models and Semantics, II*. World Scientific Publishing Co., 2004.

[Sti98] C. Stirling. Decidability of bisimulation equivalence for normed push-down processes. *Theoretical Computer Science*, 195(2):113–131, 1998.

[Stř98] J. Stříbrná. Hardness results for weak bisimilarity of simple process algebras. *MFCS'98. Electronic Notes in Theoretical Computer Science*, 18:179–190, 1998.

[vG93] R. van Glabbeek. A complete axiomatization for branching bisimulation congruence of finite-state behaviours. In *Proc. MFCS'93*, volume 711 of *Lecture Notes in Computer Science*, pages 473–484, 1993.

[vGW89] R. van Glabbeek and W. Weijland. Branching time and abstraction in bisimulation semantics. In *Information Processing'89*, pages 613–618. North-Holland, 1989.