The Theory of Sequential Value Passing

Eduardo Costa Martins
Eindhoven University of Technology
Eindhoven, The Netherlands

Recently, sequential value passing was used to prove a process-theoretic generalisation of the classical theorem of the correspondence of pushdown automata and context-free grammars. We correct the proof, further simplify the sequential value passing process algebra used, and provide a sound and ground-complete axiomatisation of this process algebra.

1 Introduction

In [6], context-free grammars are interpreted in the process theoretic setting as recursive process specifications over the process algebra TSP (Theory of Sequential Processes), and the classical correspondence between context-free grammars and pushdown automata is considered in bisimulation semantics. It is found that recursive TSP processes and pushdown automata have incomparable expressive power in bisimulation semantics; there are recursive TSP processes that are not bisimilar to any pushdown automaton, and likewise there are pushdown automata that are not bisimilar to any recursive TSP process. By replacing the sequential composition operator of TSP with the sequencing operator, we get the theory $\mathrm{TSP}^{;}$, and the expressive power of TSP is reduced such that every $\mathrm{TSP}^{;}$ process is bisimilar to some pushdown automaton [8]. To recover the full correspondence, the authors of [4] find that $\mathrm{TSP}^{;}$ needs to be extended with a notion of state awareness. This is incorporated using a data transferring mechanism known as 'sequential value passing.'

The variant of sequential value passing introduced in [4] uses a two-sorted syntax for describing processes. Determining the data state of a process required reasoning over propositional logic and defining an 'effect function' over actions. Moreover, the correspondence proof itself required an additional assumption about this effect function. The authors present a simplification in [7] that does away with these complexities. We find in [10] that the theory in [7] lacks certain properties, such as the associativity of sequencing [10]. Moreover, it uses almost twice as many operational rules compared to [4].

In this paper, we present the Theory of Sequential Value Passing, TSVP, together with a sound and ground-complete axiomatisation. TSVP combines the beneficial aspects of both of the previous theories whilst using fewer operational rules. We re-establish the correspondence with pushdown automata. Importantly, we are unable to prove that TSVP processes and pushdown automata are equally expressive in bisimulation semantics, despite the constructions we use being similar to the ones in [4, 7]. We resort to branching bisimilarity, which abstracts from internal behaviour (τ-steps). Our constructions use τ-transitions to abstract from subtle differences between the semantics of TSVP processes and pushdown automata. These differences are not considered in the constructions in [4, 7].

Outline. In Section 2, we introduce the notions necessary to formalise a process-theoretic version of the classical correspondence between context-free grammars and pushdown automata, including the process algebra $\mathrm{TSP}^{;}$. In Section 3, we extend $\mathrm{TSP}^{;}$ with sequential value passing using a different approach to [4, 7]. In Section 4 we provide a sound and ground-complete axiomatisation of TSVP. Lastly, we repair the correspondence proof in Section 5, and conclude in Section 6. Proofs are in the appendix.

Acknowledgements. Thanks to Bas Luttik and Jos Baeten for the correspondence about [4, 7].

Submitted to:
Express/SOS 2026

2 Preliminaries

We use a *labelled transition system* as a common semantic framework.

Definition 1. A *transition system space* is a 4-tuple $\langle S, \mathscr{A}, \rightarrow, \downarrow \rangle$, where

1. S is a set of states;
2. $\mathscr{A}$ is a set of actions; $\tau \in \mathscr{A}$ is the invisible action;
3. $\rightarrow \subseteq S \times \mathscr{A} \times S$ is an $\mathscr{A}$-labelled *transition relation*; and
4. $\downarrow \subseteq S$ is the set of *final* or *accepting* states.

A *labelled transition system* (LTS) is a transition system space equipped with a designated *root* or *initial* state.

The transition relation is written infix and acceptance is written postfix, i.e. we write $p \xrightarrow{a} q$ and $p\downarrow$ for $(p,a,q) \in \rightarrow$ and $p \in \downarrow$, respectively. Additionally, we write $p \xrightarrow{a}$ iff there exists some $q \in S$ such that $p \xrightarrow{a} q$, and $p \rightarrow$ iff there exists some $a \in \mathscr{A}$ such that $p \xrightarrow{a}$. Lastly, we write $p \not\xrightarrow{a}$, $p \not\rightarrow$, and $p\not\downarrow$ for the negation of $p \xrightarrow{a}$, $p \rightarrow$, and $p\downarrow$, respectively.

An equivalence defined on states of an LTS is extended to LTSs in the usual way: LTSs L and L' are equivalent iff in the disjoint union of L and L', their initial states are equivalent. Whereas the classical correspondence between CFGs and PDAs uses language equivalence, the de facto standard equivalence in process theory is (strong) bisimilarity.

Definition 2. A symmetric binary relation $\mathscr{R}$ on S is a *bisimulation* iff for all $p\mathscr{R}q$ and $a \in \mathscr{A}$:

1. if $p \xrightarrow{a} p'$ for some $p' \in S$, then there exists $q' \in S$ such that $q \xrightarrow{a} q'$ and $p'\mathscr{R}q'$; and
2. if $p\downarrow$ then $q\downarrow$.

We say that p is *bisimilar* to q, or $p \leftrightarrows q$, if and only if there exists a bisimulation, $\mathscr{R}$, such that $p\mathscr{R}q$.

We use the acceptance by final state notion of a pushdown automaton rather than the acceptance by empty stack notion. Classically, these interpretations are equally expressive, but in our context the final state interpretation is more expressive [5].

Definition 3. A *pushdown automaton* (PDA) $M = \langle S, \mathscr{A}, \mathscr{D}_M, \rightarrow, \uparrow, \downarrow \rangle$ is a sextuple where:

1. S is a finite set of states;
2. $\mathscr{A}$ is a finite input alphabet; $\tau \in \mathscr{A}$ is the invisible action;
3. $\mathscr{D}_M$ is a finite data alphabet;
4. $\rightarrow \subseteq S \times \mathscr{A} \times (\mathscr{D}_M \cup \{\epsilon\}) \times \mathscr{D}_M^* \times S$ is a finite set of *transitions* or *steps*;
5. $\uparrow \in S$ is the initial state; and
6. $\downarrow \subseteq S$ is the set of final or accepting states.

We write $p \xrightarrow{a[\delta/x]} q$ if $(p,a,\delta,x,q) \in \rightarrow$. If $\delta \neq \epsilon$, then this has the meaning 'p can pop δ, consume input symbol a, and push x to transition to q'. If $\delta = \epsilon$, then the meaning is 'if the stack is empty, then p can consume input symbol a, and push x to transition to q'. We associate an LTS with a PDA by considering the underlying transition relation that defines its language.

Definition 4. Let $M = \langle S, \mathscr{A}, \mathscr{D}_M, \rightarrow, \uparrow, \downarrow \rangle$ be a pushdown automaton. The LTS associated with M is $\mathscr{L}(M) = \langle S_{\mathscr{L}(M)}, \mathscr{A}, \rightarrow_{\mathscr{L}(M)}, \downarrow_{\mathscr{L}(M)}, \uparrow_{\mathscr{L}(M)} \rangle$ defined as follows:

1. $S_{\mathscr{L}(M)} = \{(s,x) \mid s \in S \wedge x \in \mathscr{D}_M^*\}$;
2. $\rightarrow_{\mathscr{L}(M)} \subseteq S_{\mathscr{L}(M)} \times \mathscr{A} \times S_{\mathscr{L}(M)}$ is the least relation such that for all $s,s' \in \mathscr{S}$, $a \in \mathscr{A}$, $d \in \mathscr{D}_M$, and $x,x' \in \mathscr{D}_M^*$ we have
   - $(s,dx) \xrightarrow{a}_{\mathscr{L}(M)} (t,x'x)$ if, and only if, $s \xrightarrow{a[d/x']} t$;
   - $(s,\epsilon) \xrightarrow{a}_{\mathscr{L}(M)} (t,x)$ if, and only if, $s \xrightarrow{a[\epsilon/x]} t$;
3. $\downarrow_{\mathscr{L}(M)} = \{(s,x) \mid s \in \downarrow \wedge x \in \mathscr{D}_M^*\}$; and
4. $\uparrow_{\mathscr{L}(M)} = (\uparrow, \epsilon)$.

An LTS associated with a pushdown automaton is called a *pushdown process* (PDP).

We now present the process algebra TSP$^;$ [8], which we extend in the next section. TSP$^;$ has the parameter $\mathscr{A}$, the set of possible actions, and the parameter $\mathscr{P}$, a *finite* set of *process identifiers*. The set of TSP$^;$ expressions is generated by the following grammar ($a \in \mathscr{A}, X \in \mathscr{P}$):

$$p ::= \mathbf{0} \mid \mathbf{1} \mid a.p \mid p+p \mid p;p \mid X$$

A recursive specification is a mapping Δ from $\mathscr{P}$ to the set of process expressions. We typically think of Δ as a set of defining equations, writing $X \stackrel{\text{def}}{=} p$ rather than $\Delta(X) = p$.

To reduce the number of parentheses when writing TSP$^;$ expressions we use the rule that action prefix ($a.p$) binds strongest, and alternative composition ($p+p$) binds weakest. For example, $a.\mathbf{1}+b.\mathbf{0}$ represents $(a.\mathbf{1})+(b.\mathbf{0})$, and not $a.(\mathbf{1}+b.\mathbf{0})$.

The semantics of TSP$^;$ is defined by means of structural operational semantics using a unary acceptance predicate $\downarrow$ (written postfix) and, for each $a \in \mathscr{A}$, a binary transition relation $\xrightarrow{a}$ (written infix). The operational rules of the transition system specification in Table 1 should be read as follows: if the predicates above the line are derivable for a given substitution, then the conclusion(s) below the line are derivable for the same substitution. The negative premise, $x \nrightarrow$, holds iff there is no $a \in \mathscr{A}$ and TSP$^;$ expression x' for which $x \xrightarrow{a} x'$ is derivable.

Table 1: Operational rules for TSP$^;$.

$$\frac{}{\mathbf{1}\downarrow}\ \text{T1} \qquad \frac{}{a.x \xrightarrow{a} x}\ \text{AP1}$$

$$\frac{x \xrightarrow{a} x'}{x+y \xrightarrow{a} x' \quad y+x \xrightarrow{a} x'}\ \text{AC1} \qquad \frac{x\downarrow}{x+y\downarrow \quad y+x\downarrow}\ \text{AC2}$$

$$\frac{x \xrightarrow{a} x'}{x;y \xrightarrow{a} x';y}\ \text{S1} \qquad \frac{x\downarrow \quad x \nrightarrow \quad y \xrightarrow{a} y'}{x;y \xrightarrow{a} y'}\ \text{S2} \qquad \frac{x\downarrow \quad y\downarrow}{x;y\downarrow}\ \text{S3}$$

$$\frac{x \xrightarrow{a} x' \quad X \stackrel{\text{def}}{=} x}{X \xrightarrow{a} x'}\ \text{REC1} \qquad \frac{x\downarrow \quad X \stackrel{\text{def}}{=} x}{X\downarrow}\ \text{REC2}$$

The transition system specification associates a transition system space with TSP$^;$ as follows:

- the set of states consists of all TSP$^;$ process expressions;

- the transition relation consists of exactly those transitions that are derivable using the operational rules; and
- likewise, the set of accepting states consists of exactly the states for which acceptance is derivable using the operational rules.

The LTS associated with a TSP$^;$ process expression p is the fragment of the TSP$^;$ transition system space reachable from p, with p designated as the root state.

The $\_;\_$ symbol represents sequencing. For $p = p_1; p_2; \ldots; p_n$ (sequencing is associative) we call p_1 the *head* of p and $p_2; \ldots; p_n$ the *tail* of p. The semantics of sequencing are like that of the sequential composition in TSP, with the exception that the semantics for sequential composition do not require the $x \not\rightarrow$ premise in the rule S2. For this reason, it is useful to make the distinction between a terminating process and an accepting process. Whereas the notion of 'terminating' is typically synonymous with 'accepting,' we now say that a process is *terminating* iff it is accepting *and* cannot execute any actions.

It is well-known that the existence of negative premises such as $x \not\rightarrow$ in a transition system specification means the transition system specification may not define a unique transition relation that agrees with provability. Fortunately, TSP$^;$ is stratifiable, and as a consequence, this is not the case [9]. The extended theory, TSVP, is also stratifiable using a stratification similar to the one shown in [10]. Further details about stratifiability are beyond the scope of this paper. As usual, we must restrict our scope to *guarded* recursive specifications: a process expression is guarded if occurrence of a process identifier is within the scope of an action prefix, and a recursive specification is guarded if each defining equation is guarded.

3 Theory of Sequential Value Passing

We now extend TSP$^;$ with a notion of data, using an approach similar to the one in [7]. For our notion of data, we use *attributes*, which range over a *finite* data set, $\mathscr{D}$, a parameter of the theory. To assign attributes to processes, we add the unary *attribute assignment operator*, $d{:}\blacktriangleright\_$, to the syntax of the theory, one for each $d \in \mathscr{D}$. It has the intuitive meaning that $d{:}\blacktriangleright p$ behaves as the process p with assigned attribute d. For example, the process $d{:}\blacktriangleright a.\mathbf{1}$ represents the process with attribute d that admits an a-transition to an accepting state. After the a-transition, the process loses the attribute.

Next, to introduce conditional behaviour, we add the unary *guarded command* operator, $\delta :\rightarrow \_$, to the syntax of the theory, one for each $\delta \in \mathscr{D} \cup \{\ominus\}$. We use the symbol $\ominus$ to express the absence of an attribute, and write $\mathscr{D}_\ominus$ to denote $\mathscr{D} \cup \{\ominus\}$. The process $\delta :\rightarrow p$ intuitively describes the process that behaves as p upon being assigned attribute δ and that otherwise exhibits no behaviour. So $d :\rightarrow a.\mathbf{1}$ describes a process that may execute an a-transition to an accepting state upon being assigned attribute d, and is otherwise deadlocked. We allow $\ominus$ as a guard, in order to describe behaviour that is only possible in the absence of an attribute.

The operational rules use the following auxiliary predicates:

- There is a consistency predicate, $\mathscr{C}\delta$ for each $\delta \in \mathscr{D}_\ominus$. For a process p, $p\mathscr{C}\delta$ has the meaning 'p is consistent and has attribute value δ,' or if $\delta = \ominus$, then the meaning is 'p is consistent and has no attribute value.' We say p is *consistent* and write $p\mathscr{C}$ iff $p\mathscr{C}\delta$ for some $\delta \in \mathscr{D}_\ominus$, and otherwise say p is *inconsistent*.
- There is a conditional transition relation, $\xrightarrow{\delta,a}$ for each $\delta \in \mathscr{D}_\ominus, a \in \mathscr{A}$. Then $p \xrightarrow{\delta,a} q$ has the meaning 'if p has attribute δ, then it can execute an a transition and proceed as q.'
- Likewise, there is a conditional acceptance predicate, ${}^\delta\!\downarrow$ for each $\delta \in \mathscr{D}_\ominus$. Then $p{}^\delta\!\downarrow$ has the meaning 'if p has attribute δ, then it is accepting.'

Additionally, we write $p \xrightarrow{\delta,a}$ iff there exists some TSVP expression q such that $p \xrightarrow{\delta,a} q$, and $p \xrightarrow{\delta,}$ iff there exists some $a \in \mathscr{A}$ such that $p \xrightarrow{\delta,a}$. Lastly, we write $p \xrightarrow{\delta,a}\!\!\!\!\!\!/\;$, $p \xrightarrow{\delta,}\!\!\!\!\!\!/\;$, and $p^{\delta}\!\not\downarrow$ for the negation of $p \xrightarrow{\delta,a}$, $p \xrightarrow{\delta,}$, and $p^{\delta}\downarrow$, respectively.

The predicates $\xrightarrow{\ominus,a}$ and ${}^{\ominus}\downarrow$ describe conditional behaviour that is only possible in the absence of an attribute value. The *concrete behaviour* $\downarrow$ and $\xrightarrow{a}$ that defines the TSVP transition system space is derived using the operational rules in Table 2.

Table 2: Operational rules for concrete behaviour.

$$\frac{x \xrightarrow{\delta,a} x' \quad x \mathscr{C} \delta}{x \xrightarrow{a} x'}\ \text{O1} \qquad \frac{x^{\delta}\downarrow \quad x \mathscr{C} \delta}{x\downarrow}\ \text{O2}$$

No other operational rules can be used to conclude $p \xrightarrow{a} p'$ or $p\downarrow$. This means that we disregard all the rules in Table 1. Instead, we obtain the rules in Table 3 by taking each rule in Table 1, adding a consistency premise, and replacing each $\xrightarrow{a}$ and $\downarrow$ with its conditional variant. The conditional variant of $x \nrightarrow$ is $x \xrightarrow{\delta,}\!\!\!\!\!\!/\;$, which holds iff there is no a, x' for which $x \xrightarrow{\delta,a} x'$ is derivable.

Table 3: Conditional semantics of TSP: operators.

$$\frac{\mathbf{1} \mathscr{C}}{\mathbf{1}^{\delta}\downarrow}\ \text{T1} \qquad \frac{a.x \mathscr{C} \quad x \mathscr{C}}{a.x \xrightarrow{\delta,a} x}\ \text{AP1}$$

$$\frac{x \xrightarrow{\delta,a} x' \quad x+y \mathscr{C}}{x+y \xrightarrow{\delta,a} x' \qquad y+x \xrightarrow{\delta,a} x'}\ \text{AC1} \qquad \frac{x^{\delta}\downarrow \quad x+y \mathscr{C}}{x+y^{\delta}\downarrow \qquad y+x^{\delta}\downarrow}\ \text{AC2}$$

$$\frac{x \xrightarrow{\delta,a} x' \quad x;y \mathscr{C}}{x;y \xrightarrow{\delta,a} x';y}\ \text{S1} \qquad \frac{x^{\delta}\downarrow \quad x \xrightarrow{\delta,}\!\!\!\!\!\!/\; \quad y \xrightarrow{\delta,a} y' \quad x;y \mathscr{C}}{x;y \xrightarrow{\delta,a} y'}\ \text{S2} \qquad \frac{x^{\delta}\downarrow \quad y^{\delta}\downarrow \quad x;y \mathscr{C}}{x;y^{\delta}\downarrow}\ \text{S3}$$

$$\frac{x \xrightarrow{\delta,a} x' \quad X \overset{\text{def}}{=} x \quad X \mathscr{C}}{X \xrightarrow{\delta,a} x'}\ \text{REC1} \qquad \frac{x^{\delta}\downarrow \quad X \overset{\text{def}}{=} x \quad X \mathscr{C}}{X^{\delta}\downarrow}\ \text{REC2}$$

The operational rules for attribute assignment and guarded command are given in Table 4.

The operational rules for consistency are given in Table 5. These are such that each consistent process has at most one attribute value. The operational rule S0 for the consistency of a sequencing $p;q$ requires that the right-hand side argument of the sequencing does not have an attribute value. This way, if $p;q$ is consistent and p can reach some state p', then also $p';q$ is consistent. As a consequence, we use fewer operational rules for consistency in comparison to [7], and achieve the same result.

The operational rules in Tables 2 to 5 are all the operational rules of TSVP. The following propositions verify that a consistent process expression has at most one attribute, and only consistent processes exhibit any form of behaviour.

Proposition 1. *Let p be some process expression. There exists at most one $\delta \in \mathscr{D}$ such that $p \mathscr{C} \delta$.*

Table 4: Conditional semantics of attribute assignment and guarded command.

$$\frac{x \xrightarrow{\delta,a} x' \quad d^{\frown\blacktriangle} x \mathscr{C}}{d^{\frown\blacktriangle} x \xrightarrow{\delta,a} x'} \text{AA1} \qquad \frac{x \xrightarrow{\delta,a} x' \quad \delta :\!\rightarrow x \mathscr{C}}{\delta :\!\rightarrow x \xrightarrow{\delta,a} x'} \text{G1}$$

$$\frac{x^{\delta}\downarrow \quad d^{\frown\blacktriangle} x \mathscr{C}}{d^{\frown\blacktriangle} x^{\delta}\downarrow} \text{AA2} \qquad \frac{x^{\delta}\downarrow \quad \delta :\!\rightarrow x \mathscr{C}}{\delta :\!\rightarrow x^{\delta}\downarrow} \text{G2}$$

Table 5: Operational rules for consistency.

$$\frac{}{\mathbf{0} \mathscr{C} \ominus} \text{Z0} \qquad \frac{}{\mathbf{1} \mathscr{C} \ominus} \text{T0}$$

$$\frac{}{a.x \mathscr{C} \ominus} \text{AP0} \qquad \frac{x \mathscr{C} \gamma \quad y \mathscr{C} \delta \quad \gamma \in \{\ominus, \delta\}}{x + y \mathscr{C} \delta \qquad y + x \mathscr{C} \delta} \text{AC0}$$

$$\frac{x \mathscr{C} \delta \quad y \mathscr{C} \ominus}{x; y \mathscr{C} \delta} \text{S0} \qquad \frac{x \mathscr{C} \delta \quad X \stackrel{\text{def}}{=} x}{X \mathscr{C} \delta} \text{REC0}$$

$$\frac{x \mathscr{C} \delta \quad \delta \in \{\ominus, d\}}{d^{\frown\blacktriangle} x \mathscr{C} d} \text{AA0} \qquad \frac{x \mathscr{C} \gamma \quad \gamma \in \{\ominus, \delta\}}{\delta :\!\rightarrow x \mathscr{C} \ominus} \text{G0}$$

Proposition 2. *Let p and q be process expressions. If for some $\delta \in \mathscr{D}_{\ominus}$, and $a \in \mathscr{A}$ it holds that $p \xrightarrow{\delta,a} q$, $p^{\delta}\downarrow$, or $q \xrightarrow{\delta,a} p$, then $p \mathscr{C}$.*

We note that some operational rules here can be simplified. For example, the $\mathbf{1} \mathscr{C}$ and $a.x \mathscr{C}$ premises in operational rules T1 and AP1 are superfluous due to the operational rules T0 and AP0, and the $x; y \mathscr{C}$ premise in rules S1–S3 can be reduced to $y \mathscr{C} \ominus$ by Proposition 2. We chose not to do so, in order to keep the presentation of operational rules consistent.

Example 1. We illustrate the notion of 'sequential value passing' with a coin toss game that is won if the result of the coin toss is heads. Consider the following process expressions:

$$\text{Toss} = \text{toss}.h^{\frown\blacktriangle}\mathbf{1} + \text{toss}.t^{\frown\blacktriangle}\mathbf{1}$$
$$\text{Result} = h :\!\rightarrow \text{win}.\mathbf{1} + t :\!\rightarrow \text{lose}.\mathbf{0}$$

The transition system associated with the process Toss;Result is shown in Figure 1. For illustrative purposes, the states are annotated with the attribute value of the underlying expression. We emphasise that the attribute values are not part of the LTS structure.

It is quite straightforward to prove that Toss;Result $\xrightarrow{\text{toss}}$ $(h^{\frown\blacktriangle}\mathbf{1})$;Result. Then, $h^{\frown\blacktriangle}\mathbf{1}^{h}\downarrow$ and $h^{\frown\blacktriangle}\mathbf{1} \xrightarrow{h,}$. Since $h^{\frown\blacktriangle}\mathbf{1} \mathscr{C} h$, it holds that $h^{\frown\blacktriangle}\mathbf{1}$ is terminating by O1 and O2, and moreover $(h^{\frown\blacktriangle}\mathbf{1})$;Result $\mathscr{C} h$ by S0. Therefore, S2 and O1 derive $(h^{\frown\blacktriangle}\mathbf{1})$;Result $\xrightarrow{\text{win}}$ $\mathbf{1}$. Essentially, the attribute value of a terminating head is passed to the tail of the sequencing. It ends up being that $(h^{\frown\blacktriangle}\mathbf{1})$;Result behaves the same as $h^{\frown\blacktriangle}$Result, and this property is known as 'sequential value passing.'

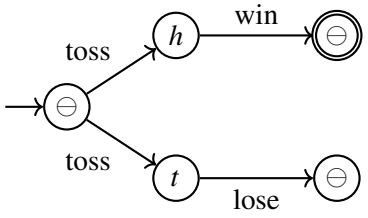

Figure 1: Coin toss example with an added winning condition. A double circle such as the one after the 'win' transition denotes an accepting state.

Proposition 3. *Bisimilarity is not a congruence for* TSVP.

Proof. Consider the process Result from the previous example. Note that $(h^{\blacktriangle}\mathbf{1});\text{Result} \xrightarrow{\text{win}}$, whereas $(t^{\blacktriangle}\mathbf{1});\text{Result} \xrightarrow{\text{win}}$. Hence, the two processes are not bisimilar, even though $h^{\blacktriangle}\mathbf{1} \leftrightarrows t^{\blacktriangle}\mathbf{1}$. □

Bisimilarity is not a congruence for TSVP because it does not capture the change in behaviour resulting from the assignment of attributes. This is common for process algebras with a notion of data, so the authors of [14] provide a number of equivalences to remedy this. We find that the notion of *stateless bisimilarity* serves as a congruence for TSVP, however we must adapt the definition given in [14] as it is for process algebras with a two-sorted syntax such as the one in [4]. We arrive at the following definition.

Definition 5. A symmetric binary relation $\mathscr{R}$ on TSVP terms is a *stateless bisimulation* if for each $\delta \in \mathscr{D}_{\ominus}, a \in \mathscr{A}$, and TSVP term p', whenever $p\mathscr{R}q$ then:

1. $p\mathscr{C}\delta$ implies $q\mathscr{C}\delta$;
2. $p \xrightarrow{\delta,a} p'$ implies there exists a TSVP term q' such that $q \xrightarrow{\delta,a} q'$ and $p'\mathscr{R}q'$; and
3. $p^{\delta}\downarrow$ implies $q^{\delta}\downarrow$.

We say that p is *stateless bisimilar* to q, or $p \leftrightarrows_{sl} q$, if and only if there exists a stateless bisimulation, $\mathscr{R}$, such that $p\mathscr{R}q$.

Proposition 4. *Stateless bisimilarity is finer than bisimilarity, and it is a congruence for* TSVP. *Moreover, $\leftrightarrows_{sl}$ is a stateless bisimulation.*

The idea of using the operational rules in Table 2 (and only those rules) to define the TSVP transition system space brings about many benefits compared to the theory in [7]. In short, it reduces the number of operational rules required, and it means we do not need to introduce any extra operational rules for the function symbols of TSP$^{:}$, i.e. the rules in Table 1 correspond one-to-one to the rules in Table 3. It also provides a clear workflow for deriving the concrete behaviour of a TSVP process: first determine its attribute value, and then the conditional behaviour w.r.t. that attribute. As we show in the following example, this together with the ability to describe behaviour conditioned on the absence of an attribute is what leads to sequencing being associative in TSVP, whereas it was not in [7].

Example 2. Consider the following process expressions, with $d \in \mathscr{D}, \delta \in \mathscr{D}_{\ominus}$:

$$\begin{aligned} p &= \delta^{\blacktriangle}\mathbf{1} \\ q &= \mathbf{1} + d :\!\rightarrow a.\mathbf{0} \\ r &= b.\mathbf{0} \quad . \end{aligned}$$

We will consider the processes $p;(q;r)$ and $(p;q);r$. It turns out that, due to sequential value passing, we may simply consider the equivalent processes $\delta \triangleright (q;r)$ and $(\delta \triangleright q);r$. This is formalised in the next section, and is also valid for the theory in [7].

First, consider $(\delta \triangleright q);r$. It holds that $(\delta \triangleright q);r \xrightarrow{b}$ iff $\delta \neq d$. This is because, if $\delta = d$, the attribute δ allows the a-transition of q, thereby disabling the b-transition of r since $\delta \triangleright q$ is not terminating. The theory in [7] agrees on this.

Next, consider $\delta \triangleright (q;r)$. In both theories, it holds that $q;r \xrightarrow{b}$. For the theory in [7], a transition of the form $\xrightarrow{b}$ is viewed as unconditional, meaning that this transition is propagated to $\delta \triangleright (q;r)$, i.e. $\delta \triangleright (q;r) \xrightarrow{b}$ regardless of the value of δ. However, for sequencing to be associative, we require $\delta \triangleright (q;r) \xrightarrow{b}$ iff $\delta \neq d$. This can be expressed in TSVP semantics since $q;r \xrightarrow{\varepsilon,b}$ for all $\varepsilon \neq d$. This is not applicable to the theory in [7], since there is no $\xrightarrow{\ominus,b}$ predicate. In other words, TSVP can express the condition for disabling an action, whereas the theory in [7] cannot since it can only express unconditional actions, and conditions for enabling an action.

For more details, the reader may consult [10]. There, an alternative semantics is considered where $q;r$ cannot execute this b-action due to the presence of the conditional a-action. However, a 'dual' counter-example shows that sequencing remains not associative.

4 Equational Theory

We now consider the equational theory of the recursion-free fragment of TSVP. We use a countably infinite set of variables, $\mathscr{V}$, to express equational properties. The set of TSVP terms is generated by the following grammar ($a \in \mathscr{A}, d \in \mathscr{D}, \delta \in \mathscr{D}_{\ominus}, x \in \mathscr{V}$):

$$t ::= \mathbf{0} \mid \mathbf{1} \mid a.t \mid t + t \mid d \triangleright t \mid \delta :\to t \mid t;t \mid x$$

Note that a process expression, as considered in the previous section, does not contain variables, whereas here, a term may contain variables. A term without variables is also called a *closed term.* A *closed substitution* is a mapping σ from variables to closed terms. If t is a TSVP term and σ is a closed substitution, then by $\sigma(t)$ we denote the process expression obtained by replacing every occurrence of a variable x in t by $\sigma(x)$.

Equational reasoning allows us to reason over TSVP expressions syntactically rather than semantically. We see the usefulness in the next section, where we use equational reasoning to establish the correspondence between TSVP processes and pushdown automata. Thus, we seek a set of equations, called an *axiomatisation*, that allows us to derive all valid stateless bisimilarities between TSVP processes, and only those that are valid.

Definition 6 (Soundness and Ground-completeness)**.** Let t, u be TSVP terms. An equation $t = u$ is *sound* iff for all closed substitutions σ, it holds that $\sigma(t) \leftrightarrows_{sl} \sigma(u)$.

A set E of equations is *ground-complete* iff for all TSVP process expressions p, q, it holds that $p \leftrightarrows_{sl} q$ implies $p = q$ is derivable using the equations in E and the rules of equational logic.

Much like TSP$^:$, we find that TSVP is not finitely axiomatisable, for the same reasons as shown in [9]. The authors of [9] introduce an auxiliary *non-acceptance* operator to the syntax of of TSP$^:$ in order to obtain a finite axiomatisation. The operational rules for this operator are shown in Table 6, adapted slightly to account for the introduction of attributes and conditionals.

Table 6: Operational rules for non-acceptance in TSVP.

$$\frac{x\mathscr{C}\delta}{NA(x)\mathscr{C}\delta}\ \text{NA0} \qquad \frac{x \xrightarrow{\delta,a} x' \quad NA(x)\mathscr{C}}{NA(x) \xrightarrow{\delta,a} x'}\ \text{NA1}$$

Intuitively, $NA(p)$ describes a process that behaves like p, except without (conditional) acceptance. Note that $NA(p)$ may transition into an accepting state, but $NA(p)$ itself is not accepting.

In order to finitely axiomatise TSVP, we also introduce the *consistency assertion* operator, $\mathscr{C}_\delta(\_)$ for each $\delta \in \mathscr{D}_\ominus$. Intuitively, $\mathscr{C}_\delta(p)$ behaves as p if $p\mathscr{C}\delta$, and otherwise it behaves like the deadlock state with attribute δ. The operational rules for consistency assertion are found in Table 7. We conjecture that this operator is necessary in order to finitely axiomatise TSVP. We also use the constant $\bot$, with no operational rules, to express the inconsistent process. Note that by Proposition 2, all inconsistent processes are in the same stateless bisimilarity equivalence class. Therefore, $\bot$ is used out of convenience rather than necessity, as all instances of $\bot$ in an axiomatisation may be replaced by some other inconsistent process.

Table 7: Operational rules for consistency assertion in TSVP.

$$\frac{}{\mathscr{C}_\gamma(x)\mathscr{C}\gamma}\ \text{C0} \qquad \frac{x \xrightarrow{\delta,a} x' \quad x\mathscr{C}\gamma}{\mathscr{C}_\gamma(x) \xrightarrow{\delta,a} x'}\ \text{C1} \qquad \frac{x^\delta\downarrow \quad x\mathscr{C}\gamma}{\mathscr{C}_\gamma(x)^\delta\downarrow}\ \text{C2}$$

We now present a finite, sound and ground-complete axiomatisation of TSVP. We use the following notation:

- $\ominus^{\blacktriangle}x$ denotes x.
- $\sum_{i=1}^{0} x_i = \mathbf{0}$ and $\sum_{i=1}^{n+1} x_i = \sum_{i=1}^{n} x_i + x_{n+1}$.
- $\sum_{\delta\in\emptyset} x_\delta = \mathbf{0}$ and $\sum_{\delta\in\{\varepsilon\}\cup\mathscr{D}'} x_\delta = \sum_{\delta\in\mathscr{D}'} x_\delta + x_\varepsilon$.
- $\mathscr{D}' :\to x$ with $\mathscr{D}' \subseteq \mathscr{D}_\ominus$ denotes $\sum_{\delta\in\mathscr{D}'} \delta :\to x$.

The order of summation is unimportant since alternative composition is commutative and associative.

First, consider the axioms in Table 8. These are generalisations of the axioms for TSP$^;$ [9]. The changes are as follows:

- Axioms A7, A9 (which describes sequential value passing), and A10 now include an attribute assignment and an assertion that the tail has no attribute value. The assertion is necessary because $p;q$ may be inconsistent even though p and q themselves are both consistent. For example, Axiom A7 does not apply to the inconsistent process $p = \mathbf{0};(d^{\blacktriangle}\mathbf{1})$, since the right-hand side of the axiom is necessarily consistent, but p is inconsistent. The attribute assignment is used to generalise the original axioms, which may be recovered by taking $\delta = \ominus$.
- Axiom A12 describes how acceptance in the head of a sequencing may distribute if the head of the sequencing can execute a transition. This is coded by the $a.\mathscr{C}_\delta(x)$ summand. In TSP$^;$, the consistency assertion is not needed, since $a.x \xrightarrow{a} x$ always holds, but now we require that

x is consistent. Axiom A12 was originally two separate axioms in [9], one where the tail has acceptance, and one where it does not. The two axioms collapse into one, since the latter may be expressed by taking $\mathscr{D}' = \emptyset$.

We note that Axiom A4 is missing. This TSP axiom expressed the left-distributivity of sequential composition over alternative composition. It is not sound in TSP$^;$ (or TSVP), where it is replaced by Axiom A11.

Table 8: TSVP axioms that generalise TSP$^;$ axioms ($a \in \mathscr{A}, \delta \in \mathscr{D}_\ominus$).

$x+y = y+x$	A1	$(x+y)+z = x+(y+z)$	A2
$x+x = x$	A3	$(x;y);z = x;(y;z)$	A5
$x+\mathbf{0} = x$	A6	$(\delta \triangleright \mathbf{0});\mathscr{C}_\ominus(x) = \delta \triangleright \mathbf{0}$	A7
$x;\mathbf{1} = x$	A8	$(\delta \triangleright \mathbf{1});\mathscr{C}_\ominus(x) = \delta \triangleright \mathscr{C}_\ominus(x)$	A9
$(\delta \triangleright a.x);\mathscr{C}_\ominus(y) = \delta \triangleright a.(x;\mathscr{C}_\ominus(y))$	A10	$NA(x+y);z = NA(x);z + NA(y);z$	A11
$(a.\mathscr{C}_\delta(x)+y+\mathbf{1});(NA(z)+\mathscr{D}' :\to \mathbf{1}) = (a.\mathscr{C}_\delta(x)+y);(NA(z)+\mathscr{D}' :\to \mathbf{1})+\mathscr{D}' :\to \mathbf{1}$	A12		

The axioms in Table 9 are for the newly added operators of TSVP, the most notable of which are Axioms G5 and G7. Axiom G5 expresses that unconditional behaviour is the same as conditional behaviour that can be executed under any condition. Axiom G7 is similar to Axiom A11 in the sense that it is a special case of distributivity. It shows that conditional behaviour can be separated.

Table 9: TSVP axioms for attribute assignment and guarded command ($d \in \mathscr{D}, \delta \in \mathscr{D}_\ominus$).

$\delta :\to \mathbf{0} = \mathbf{0}$	G1	$\delta :\to (x+y) = (\delta :\to x)+(\delta :\to y)$		G2
$\delta :\to (\delta :\to x) = \delta :\to x$	G3	$\delta :\to (\varepsilon :\to \mathscr{C}_\ominus(x)) = \mathbf{0}$	$(\delta \neq \varepsilon)$	G4
$\mathscr{D}_\ominus :\to \mathscr{C}_\ominus(x) = \mathscr{C}_\ominus(x)$	G5	$\delta :\to (\delta \triangleright x) = \delta :\to x$		G6
$(\sum_{\gamma\in\mathscr{D}_\ominus} \gamma :\to x_\gamma);y = \sum_{\gamma\in\mathscr{D}_\ominus} \gamma :\to (x_\gamma;y)$				G7
$\delta \triangleright (x+y) = (\delta \triangleright x)+y$				S1
$d \triangleright (d :\to x) = d \triangleright x$	S2	$d \triangleright (\delta :\to \mathscr{C}_\ominus(x)) = d \triangleright \mathbf{0}$	$(d \neq \delta)$	S3

Lemma 1. *Let $\mathscr{D}', \mathscr{D}'' \subseteq \mathscr{D}_\ominus$. The following equations are derivable using the axioms in Table 9:*

$$\mathscr{D}' :\to x + \mathscr{D}'' :\to x = \mathscr{D}' \cup \mathscr{D}'' :\to x$$
$$\mathscr{D}' :\to (\mathscr{D}'' :\to \mathscr{C}_\ominus(x)) = \mathscr{D}' \cap \mathscr{D}'' :\to \mathscr{C}_\ominus(x) \quad .$$

Lastly, the axioms in Table 10 are the axioms for auxiliary operators. These can be used to determine if a process is non-accepting, the attribute value of a process, and the inconsistency of a process.

To show that the set of axioms in Tables 8 to 10 is ground-complete, we use a standard technique as detailed in [2]. We establish a head normal form for TSVP, a standard format for process expressions with reduced syntactic complexity. We show that each process expression is derivably equal to some process expression in head normal form, and then show that two stateless bisimilar processes in head normal form are syntactically equal up to some minor differences.

Table 10: Axioms for auxiliary operators of TSVP ($a \in \mathscr{A}, d, e \in \mathscr{D}, \delta, \varepsilon \in \mathscr{D}_\ominus$).

$NA(\mathbf{0}) = \mathbf{0}$	NA1	$NA(\mathbf{1}) = \mathbf{0}$	NA2
$NA(a.x) = a.x$	NA3	$NA(x+y) = NA(x) + NA(y)$	NA4
$NA(d \curlywedge x) = d \curlywedge NA(x)$	NA5	$NA(d :\to x) = d :\to NA(x)$	NA6
$\mathscr{C}_\ominus(\mathbf{0}) = \mathbf{0}$	C1	$\mathscr{C}_\ominus(\mathbf{1}) = \mathbf{1}$	C2
$\mathscr{C}_\ominus(a.x) = a.x$	C3	$\mathscr{C}_\ominus(\mathscr{C}_\ominus(x) + \mathscr{C}_\ominus(y)) = \mathscr{C}_\ominus(x) + \mathscr{C}_\ominus(y)$	C4
$\mathscr{C}_d(d \curlywedge \mathscr{C}_\ominus(x)) = d \curlywedge \mathscr{C}_\ominus(x)$	C5	$\mathscr{C}_\ominus(\delta :\to \mathscr{C}_\ominus(x)) = \delta :\to \mathscr{C}_\ominus(x)$	C6
$\mathscr{C}_\delta(\mathscr{C}_\delta(x); \mathscr{C}_\ominus(y)) = \mathscr{C}_\delta(x); \mathscr{C}_\ominus(y)$	C7	$\mathscr{C}_\delta(\mathscr{C}_\varepsilon(x)) = \delta \curlywedge \mathbf{0} \quad (\delta \neq \varepsilon)$	C8

$a.\bot = \mathbf{0}$		B1	$x + \bot = \bot$	B2	$d \curlywedge \bot = \bot$	B3	
$d \curlywedge (e \curlywedge x) = \bot$	$(d \neq e)$	B4	$d :\to \bot = \bot$	B5	$\bot; x = \bot$	B6	
$x; (d \curlywedge y) = \bot$		B7	$NA(\bot) = \bot$	B8	$\mathscr{C}_\delta(\bot) = \delta \curlywedge \mathbf{0}$	B9	

Definition 7 (Head Normal Form)**.** We say a TSVP process, p, is in *Head Normal Form* (HNF) iff it is of the form:

$$\begin{aligned} &p = \bot \quad \text{or} \\ &p = \gamma \curlywedge \Big(\sum_{\delta \in \mathscr{D}} \delta :\to p_\delta \Big) \qquad \text{with each } p_\delta \text{ of the form} \\ &p_\delta = \sum_{i=1}^{n} a_i.\mathscr{C}_{\varepsilon_i}(p_i)\ [+\mathbf{1}] \end{aligned}$$

where $n \in \mathbb{N}, a_i \in \mathscr{A}, \varepsilon_i, \gamma \in \mathscr{D}_\ominus$, each p_i is a TSVP process, and $[+\mathbf{1}]$ denoting an optional $\mathbf{1}$ summand. If $p \neq \bot$, then γ is its *attribute value*. The *acceptance condition* of p is the set $\mathbb{1}[p] \subseteq \mathscr{D}_\ominus$ containing $\delta \in \mathscr{D}_\ominus$ iff p_δ has a $\mathbf{1}$ summand.

The definition for head normal form contains some redundancy. If $p \neq \bot$ and $\gamma \neq \ominus$, then the guards may be simplified in accordance with Axioms S2, S3, G1, and A6. We define the head normal form this way to facilitate the use of Axiom G7. When a process is in head normal form, one may easily determine its (in)consistency and the conditions for acceptance, as follows.

Lemma 2. *Let $p \neq \bot$ be in HNF, and let $\gamma \in \mathscr{D}_\ominus$ be its attribute value. Then $p = \mathscr{C}_\gamma(p)$ is derivable using equational logic and the axioms of* TSVP.

Lemma 3. *Let $p \neq \bot$ be in HNF. Then $p = NA(p) + \mathbb{1}[p] :\to \mathbf{1}$ is derivable using equational logic and the axioms of* TSVP.

With these properties established, we now show that each TSVP process expression can be brought to normal form.

Proposition 5. *Every guarded* TSVP *process p is derivably equal to some expression in HNF using equational logic and the axioms of* TSVP.

The HNF allows us to prove some additional axioms, as well as ground-completeness.

Lemma 4. *The following equations are derivable using the axioms in Tables 8 to 10:*

$$\begin{aligned} \delta \curlywedge (x; y) &= (\delta \curlywedge x); y \\ \delta :\to (x; y) &= (\delta :\to x); y \quad . \end{aligned}$$

Theorem 1. *Let p, q be recursion-free* TSVP *processes such that $p \leftrightarrows_{sl} q$. Then $p = q$ is derivable using the axioms in Tables 8 to 10.*

5 Correspondence

We now provide the process-theoretic generalization of the classical correspondence between pushdown automata and context-free grammars. That is, we show that each LTS associated with a pushdown automaton is branching bisimilar to some LTS associated with a TSVP process, and vice-versa. We cannot use strong bisimilarity because the constructions used require τ-transitions in order to abstract from the semantic differences between the two models. We justify the necessity of these τ-transitions with examples where relevant. To define branching bisimilarity, we use $\twoheadrightarrow$ for the reflexive, transitive closure of $\xrightarrow{\tau}$, and write $p \xrightarrow{(a)} q$ iff $p \xrightarrow{a} q$ or $a = \tau \wedge p = q$.

Definition 8. A *branching bisimulation* is a symmetric relation $\mathscr{R} \subseteq S \times S$, such that whenever $p\mathscr{R}q$, then for all $a \in \mathscr{A}$:

1. if $p \xrightarrow{a} p'$ for some $p' \in S$, then there exists $q', q'' \in S$ such that $q \twoheadrightarrow q' \xrightarrow{(a)} q''$, $p\mathscr{R}q'$ and $p'\mathscr{R}q''$; and
2. if $p\downarrow$, then there exists $q' \in S$ such that $q \twoheadrightarrow q'$, $p\mathscr{R}q'$ and $q'\downarrow$.

We say p and q are *branching bisimilar* (written $p \leftrightarrows_b q$), iff such a relation $\mathscr{R}$ exists and $p\mathscr{R}q$.

Much like the classical correspondence between CFGs and PDAs, we require a notion of Greibach Normal Form for the process-theoretic correspondence between TSVP processes and PDPs. For a sequence $\alpha \in \mathscr{P}^*$, we write $\alpha^\dagger$ to denote the process $\mathbf{1}$ if $\alpha = \epsilon$ and $X;\beta^\dagger$ if $\alpha = X\beta$ for some $\beta \in \mathscr{P}^*$ and $X \in \mathscr{P}$. We omit the † superscript when the distinction between the sequence and the associated process expression is clear. Note that by Axiom A8, it is unimportant that for the singleton sequence $X \in \mathscr{P}$, $X^\dagger$ denotes $X;\mathbf{1}$ rather than the process X.

Definition 9 (Value Passing Greibach Normal Form)**.** A process identifier X is in *Value Passing Greibach Normal Form* (VPGNF) iff its defining equation is of the form:

$$X \stackrel{\text{def}}{=} \sum_{i=1}^{n} \delta_i :\rightarrow a_i.\varepsilon_i \mathbin{\curlywedge} \alpha_i + \mathbb{1} :\rightarrow \mathbf{1}$$

with each α_i a sequence of process identifiers in VPGNF, and $\mathbb{1} \subseteq \mathscr{D}_\ominus$. We say a recursive specification Δ is in VPGNF iff each of its identifiers is in VPGNF.

We note that not every guarded specification can be transformed into VPGNF whilst preserving stateless bisimilarity. In fact, we can obtain a specification with a stateless bisimilar initial identifier, where every identifier is in VPGNF, except the initial one. The initial identifier is either of the form $\bot$, or $\gamma \mathbin{\curlywedge} X$ with X in VPGNF. Note that $\gamma \mathbin{\curlywedge} X$ is strong bisimilar to some expression in VPGNF, which can be obtained by resolving the guards in X. For example:

$$d \mathbin{\curlywedge} (d :\rightarrow a.\mathbf{0} + \ominus :\rightarrow b.\mathbf{0}) \leftrightarrows \ominus :\rightarrow a.\mathbf{0} \quad .$$

Moreover, $\bot$ is strongly bisimilar to $\mathbf{0}$, which is in VPGNF. Hence, note that the following proposition uses strong bisimilarity.

Proposition 6. *For every guarded recursive specification Δ with initial identifier $X_\uparrow$, there exists a recursive specification Δ' in VPGNF with initial identifier $X'_\uparrow$ such that $X_\uparrow \leftrightarrows X'_\uparrow$.*

The following lemma yields that each state p of a VPGNF process is bisimilar to some $\gamma \mathbin{\curlywedge} \alpha$, where $\gamma \in \mathscr{D}_\ominus$ and $\alpha \in \mathscr{P}^+$. The reason we do not describe the exact form of p is because there are many

equivalent ways of writing a sequencing, and p can take on any of these forms. For example, the sequencing $X;(Y;Z)$ is bisimilar to $(X;Y);Z$, $X;((\mathbf{1};Y);Z)$, $\mathbf{1};(\mathbf{1};(X;(Y;Z)))$, and so on. Since we do not wish to consider these details, the correspondence proofs use a type of branching bisimulation known as a branching bisimulation up-to strong bisimilarity [11, 15]. We simplify the figures used in examples by merging these states and simply writing XYZ.

Lemma 5. *Let $\alpha \in \mathscr{P}^+$ be a non-empty sequence of process identifiers in VPGNF, and let $\gamma \in \mathscr{D}_\ominus$. If there exists some $a \in \mathscr{A}$ and* TSVP *process p' such that $\gamma^\frown\!\blacktriangle\alpha \xrightarrow{a} p'$, then there exists $X \in \mathscr{P}, \alpha', \alpha'' \in \mathscr{P}^*, \beta \in \mathscr{P}^+$, and $\varepsilon \in \mathscr{D}_\ominus$ such that:*

- $\alpha = \alpha'' X \alpha'$,
- $Y^\gamma\!\downarrow$ *and* $Y \xrightarrow{\gamma,}$ *for each identifier Y in α'', and*
- $X \xrightarrow{\gamma,a} \varepsilon^\frown\!\blacktriangle\beta$ *and* $p' \leftrightarrows \varepsilon^\frown\!\blacktriangle\beta\alpha'$.

We use the following notation, where x is a sequence over an arbitrary domain X, and α is a sequence of process identifiers:

- $\overleftarrow{x}$ denotes ϵ if $x = \epsilon$ and d if $x = dx'$ with $d \in X, x' \in X^*$.
- $\overrightarrow{x}$ denotes ϵ if $x = \epsilon$ and x' if $x = dx'$ with $d \in X, x' \in X^*$.
- $\mathbb{1}[\alpha]$ denotes the set $\{\gamma \in \mathscr{D}_\ominus \mid \alpha^\gamma\!\downarrow\}$, known as the *acceptance condition* of α.

Since $\mathbb{1}[\alpha\beta] = \mathbb{1}[\alpha] \cap \mathbb{1}[\beta]$ for $\alpha, \beta \in \mathscr{P}^*$, and the acceptance condition of a process identifier in VPGNF can be determined from its defining equation, one can easily compute the acceptance condition for any sequence of identifiers in VPGNF.

5.1 Pushdown Processes as TSVP Processes

In this subsection, we show for a PDA $M = (S, \mathscr{A}, \mathscr{D}_M, \rightarrow, \uparrow, \downarrow)$, how to construct a TSVP specification Δ in VPGNF with initial identifier I such that $\mathscr{L}(M) \leftrightarrows_b I$. By Definition 4, each state of a PDP is of the form (s,x), and by Lemma 5 each state of the TSVP process is of the form $\gamma^\frown\!\blacktriangle\alpha$. Since s and γ are elements of a finite set, and x and α are both sequences, the idea in [4] is for the constructed TSVP process to have the following properties:

1. the set of attributes $\mathscr{D}$ equals S,
2. there is a process identifier X_δ for each $\delta \in \mathscr{D}_M \cup \{\epsilon\}$,
3. $X_\delta \xrightarrow{s,a} t^\frown\!\blacktriangle X_{x'}$ iff $s \xrightarrow{a[\delta/x']} t$, and
4. a state $(s, d_1 \ldots d_n)$ of $\mathscr{L}(M)$ is branching bisimilar to the state $s^\frown\!\blacktriangle X_{d_1} \ldots X_{d_n} X_\epsilon$ of the resulting TSVP process.

There are a few semantics differences between PDPs and VPGNF processes that need to be overcome for this idea to work. The first, which was realised in [4], is about acceptance. By Definition 4, a state $(s, d_1 \ldots d_n)$ of $\mathscr{L}(M)$ is accepting iff $s \in \downarrow$, whereas by operational rules AA2 and S3, $s^\frown\!\blacktriangle X_{d_1} \ldots X_{d_n} X_\epsilon$ is accepting iff $s \in \mathbb{1}[X_{d_1} \ldots X_{d_n} X_\epsilon]$. Thus, every process identifier $X \in \mathscr{P}$ in the constructed TSVP process has a $t :\rightarrow \mathbf{1}$ summand for each $t \in \downarrow$. This way, $\mathbb{1}[X] = \downarrow$, and consequently $\mathbb{1}[X_{d_1} \ldots X_{d_n} X_\epsilon] = \downarrow$.

The second difference is with how the transition relations are determined. By Definition 4, the transitions available from $(s, d_1 \ldots d_n)$ are determined entirely by s and d_1. On the other hand, by Lemma 5, the transitions of $s^\frown\!\blacktriangle X_{d_1} \ldots X_{d_n} X_\epsilon$ are determined by the first identifier X in the sequence $X_{d_1} \ldots X_{d_n} X_\epsilon$ such that $X \xrightarrow{s,}$ or $X^s\!\not\downarrow$. We run into the issue indicated by the following example.

Example 3. Consider the pushdown automaton in Figure 2, and note the $t \xrightarrow{a[\epsilon/\epsilon]} u$ transition, which is redundant because when state t is reached, the stack cannot be empty. By the third property listed above, and since the state t has no transitions with d on the top of the stack, it holds that $X_d \xrightarrow{t}$. Moreover, since t is accepting, it holds that ${X_d}^t\downarrow$. Therefore, X_d is terminating if assigned attribute t, meaning that $t^\frown X_d X_\epsilon$ can execute the redundant $t \xrightarrow{a[\epsilon/\epsilon]} u$ transition associated with X_ϵ. This means the resulting TSVP process would not have the same language. Therefore, we cannot relate $t^\frown X_d X_\epsilon$ to (t,d).

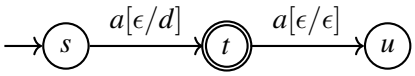

Figure 2: PDA demonstrating that the τ-transitions in Definition 10 are necessary.

The problem shown by this example is that $s^\frown X_{d_1} \ldots X_{d_n} X_\epsilon$ can execute a transition that does not stem from X_{d_1}, because X_{d_1} may be terminating when assigned attribute s. Therefore, our construction ensures a $X_\delta \xrightarrow{s,\tau} s^\frown X_\delta$ transition for each $\delta \in \mathscr{D}_M$, meaning that X_{d_1} can never be terminating. The third property listed above no longer holds. A consequence of this is that $s^\frown X_{d_1} \ldots X_{d_n} X_\epsilon \xrightarrow{\tau} s^\frown X_{d_1} \ldots X_{d_n} X_\epsilon$ always holds for $n \geq 1$, meaning we now introduce divergences.

We arrive at the construction in the following definition, where we use `T` and `F` for the Booleans true and false. The use of the Booleans here is to ensure that the identifier X_ϵ only occurs at the very end of a sequencing. Also note that the initial identifier I is defined such that I is in VPGNF, and bisimilar to $\uparrow^\frown X_\epsilon$.

Definition 10. Let $M = (S, \mathscr{A}, \mathscr{D}_M, \rightarrow, \uparrow, \downarrow)$ be some pushdown automaton. The VPGNF specification, Δ, associated with M has S as its attribute set, and the following identifiers $\{X_\delta \mid \delta \in \mathscr{D}_M \cup \{\epsilon\}\} \cup \{I\}$, with I being initial.

For $x \in \mathscr{D}_M^*$ and $b \in \mathbb{B}$, define the sequence of identifiers α_x^b inductively by $\alpha_x^{\mathtt{T}} = \alpha_x^{\mathtt{F}} X_\epsilon$, $\alpha_\epsilon^{\mathtt{F}} = \epsilon$, and $\alpha_{xd}^{\mathtt{F}} = \alpha_x^{\mathtt{F}} X_d$. Then, for $\delta \in \mathscr{D}_M \cup \{\epsilon\}$, the defining equation of X_δ has the following summands:

- $s :\rightarrow \tau.(s^\frown X_\delta)$ for each state $s \in S$ if $\delta \neq \epsilon$,
- $s :\rightarrow a.(t^\frown \alpha_x^b)$ for each transition $s \xrightarrow{a[\delta/x]} t$ in M, where $b = \mathtt{T} \iff \delta = \epsilon$, and
- $s :\rightarrow \mathbf{1}$ for each state $s \in S$ that is final in M.

Lastly, the defining equation for I has a summand $\ominus :\rightarrow p$ for each summand $\uparrow :\rightarrow p$ of X_ϵ.

Example 4. Consider the following pushdown automaton, $M = (S, \mathscr{A}, \mathscr{D}_M, \rightarrow, s, \downarrow)$.

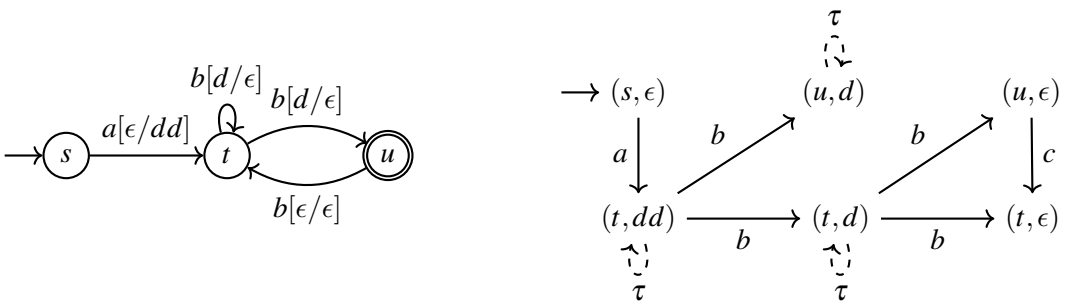

Figure 3: Pushdown automaton for Example 4. The associated pushdown process is on the right, where the accepting states are all those of the form (s',x) with $s' = u$. The dashed τ-transitions are not present in the pushdown process, but are for later reference.

The VPGNF specification associated M is as follows:

$$I \stackrel{\text{def}}{=} \ominus :\to a.t^{\frown\blacktriangle} X_d X_d X_\epsilon$$
$$X_\epsilon \stackrel{\text{def}}{=} s :\to a.t^{\frown\blacktriangle} X_d X_d X_\epsilon + u :\to c.t^{\frown\blacktriangle} X_\epsilon + u :\to \mathbf{1}$$
$$X_d \stackrel{\text{def}}{=} t :\to b.t^{\frown\blacktriangle}\epsilon + t :\to b.u^{\frown\blacktriangle}\epsilon + \sum_{s'\in S} s' :\to \tau.s'^{\frown\blacktriangle} X_d + u :\to \mathbf{1}$$

The transition system associated with I can be obtained from the PDP in Figure 3 as follows. Firstly, replace every state (s',x) with the corresponding VPGNF state $s'^{\frown\blacktriangle}\alpha_x^{\mathrm{T}}$ except for the initial state, which is replaced by I. Then, add in the dashed τ-transitions. The τ-transitions stem from the $\sum_{s'\in S} s' :\to \tau.s'^{\frown\blacktriangle} X_d$ summation in the defining equation for X_d. This is particularly important for the state $u^{\frown\blacktriangle} X_d X_\epsilon$, as without this τ-transition, it would hold that $X_d \stackrel{u,}{\nrightarrow}$, which since ${X_d}^u\downarrow$ would allow for a $u^{\frown\blacktriangle} X_d X_\epsilon \stackrel{c}{\to} t^{\frown\blacktriangle} X_\epsilon$ transition stemming from X_ϵ.

Theorem 2. *For every pushdown process, there exists a branching bisimilar guarded recursive* TSVP *specification.*

5.2 TSVP Processes as Pushdown Processes

Next, we show for a VPGNF specification with initial identifier I and attribute set $\mathscr{D}$, how to construct a PDA $M = (S, \mathscr{A}, \mathscr{D}_M, \to, \uparrow, \downarrow)$ such that $\mathscr{L}(M) \leftrightarrows_b I$. The idea is similar to before, we want:

- $S = \mathscr{D}$,
- $\mathscr{D}_M = \mathscr{P}$, and
- to relate the process $\gamma^{\frown\blacktriangle} X_1 \dots X_n$ to the state $(\gamma, X_1 \dots X_n)$.

We need to overcome the same semantic differences as before, namely:

- $\gamma^{\frown\blacktriangle} X_1 \dots X_n$ is accepting iff $\gamma \in \mathbb{1}[X_1 \dots X_n]$, whereas $(\gamma, X_1 \dots X_n)$ is accepting iff $\gamma \in \downarrow$.
- the transitions of $\gamma^{\frown\blacktriangle} X_1 \dots X_n$ are determined by the first identifier X_i in $X_1 \dots X_n$ such that $X_i \stackrel{\gamma,}{\to}$ or ${X_i}^\gamma\not\downarrow$, whereas the transitions of $(\gamma, X_1 \dots X_n)$ are determined by γ and X_1.

We first consider the difference in the transition relations. In a context without attributes, the authors of [4] use a notion of 'acceptance-irredundance' in order to eliminate terminating identifiers from $X_1 \dots X_n$. This way, the first non-terminating identifier in $X_1 \dots X_n$ is X_1, which is what we want. The following example illustrates why this approach is not suitable once attributes are introduced.

Example 5. Consider the TSVP specification in Figure 4. Note that the process Z is terminating when no attribute is assigned, and therefore the process ZX may execute the a-transition originating from X. This cannot directly be mimicked by a PDP, since Z is above X on the stack. The acceptance-irredundance approach of [4] would not allow for a terminating identifier to be placed between two non-terminating identifiers. We cannot do the same since whether or not Z is terminating depends on the attribute value assigned to it, and this attribute value cannot be known *a priori* when placing Z on the stack. In particular, the process X transitions to the state YZX. The attribute value assigned to ZX in the next state is then determined by Y, and not by X.

$$X \stackrel{\text{def}}{=} a.YZX + d :\rightarrow \mathbf{1}$$
$$Y \stackrel{\text{def}}{=} b.\mathbf{1} + b.(d^{\frown\blacktriangle}\mathbf{1})$$
$$Z \stackrel{\text{def}}{=} \mathbf{1} + d :\rightarrow c.\mathbf{1}$$

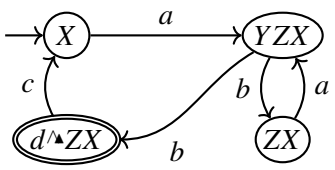

Figure 4: TSVP specification and its LTS, demonstrating the necessity of the τ-transitions in our constructions.

Since we cannot use acceptance-irredundance, our construction uses a $s \xrightarrow{\tau[d/\epsilon]} s$ transition whenever the process identifier associated with d is terminating upon assignment of the attribute associated with s. This essentially emulates sequential value passing.

Next, we consider the differences in acceptance. In a context without attributes, the construction in [4] uses a special type of GNF such that if $\alpha \xrightarrow{a} p'$, then whether or not $p'\downarrow$ can be determined entirely from information given by $\overleftarrow{\alpha}$. As with the previous example, this is no longer applicable since acceptance depends on an attribute value that cannot be known *a priori*. We need to store some additional information on the stack.

Due to the $s \xrightarrow{\tau[d/\epsilon]} s$ transitions that are used to pop terminating identifiers from the stack, we may restrict our attention to processes of the form $\gamma^{\frown\blacktriangle}\alpha$ with $\alpha = X_1 \ldots X_n$ and $\gamma^{\frown\blacktriangle}X_1$ non-terminating. Lemma 5 tells us that if $\gamma^{\frown\blacktriangle}\alpha \xrightarrow{a} p'$, then p' is of the form $\varepsilon^{\frown\blacktriangle}\beta\overrightarrow{\alpha}$, where ε and β are determined by X_1. Then, since:

- $p'\downarrow$ iff $\varepsilon \in \mathbb{1}[\beta\overrightarrow{\alpha}]$,
- $\varepsilon \in \mathbb{1}[\beta\overrightarrow{\alpha}]$ iff $\varepsilon \in \mathbb{1}[\beta] \cap \mathbb{1}[\overrightarrow{\alpha}]$, and
- ε and β are determined by X_1,

a PDA can determine whether or not $p'\downarrow$ by storing $\mathbb{1}[\overrightarrow{\alpha}]$ alongside X_1 on the stack. Thus, the state $\gamma^{\frown\blacktriangle}\alpha$ of a TSVP process with $\alpha = X_1 \ldots X_n$ will relate to $(\gamma, \overline{\alpha})$ with $\overline{\alpha} = (X_1, \mathbb{1}[X_2 \ldots X_n]) \ldots (X_n, \mathbb{1}[\epsilon])$. To maintain this structure, define for $\alpha \in \mathscr{P}^*$ and $\mathbb{1} \subseteq \mathscr{D}_\ominus$, the sequence $\overline{\alpha}^{\mathbb{1}} \in (\mathscr{P} \times 2^{\mathscr{D}_\ominus})^*$ as:

$$\overline{\alpha}^{\mathbb{1}} = \begin{cases} \epsilon & \text{if } \alpha = \epsilon \\ \overline{\beta}^{\mathbb{1}[X] \cap \mathbb{1}}(X, \mathbb{1}[X] \cap \mathbb{1}) & \text{if } \alpha = \beta X \text{ for some } \beta \in \mathscr{P}^*, X \in \mathscr{P} \end{cases}$$

and note the following properties:

1. $\overline{\alpha}^{\mathscr{D}_\ominus} = \overline{\alpha}$,
2. $\overrightarrow{\overline{\alpha}} = \overline{\overrightarrow{\alpha}}$, and
3. $\overline{\alpha}^{\mathbb{1}[\beta]}\overline{\beta} = \overline{\alpha\beta}$.

We arrive at the construction in the following definition. The constructed PDA has $(\mathbb{B} \times \mathscr{D}_\ominus) \cup \{\uparrow\}$ as its set of states rather than $\mathscr{D}_\ominus$. The Boolean is used to determine the set of accepting states, which is the set of states where this Boolean equal T. The state $\uparrow$ is initial, and is used to prepare the stack.

Definition 11. Let Δ be some TSVP specification in VPGNF with initial identifier I. The PDA associated with Δ is defined as $M = (S, \mathscr{A}, \mathscr{D}_M, \rightarrow, \uparrow, \downarrow)$ where:

- $S = (\mathbb{B} \times \mathscr{D}_\ominus) \cup \{\uparrow\}$,
- $\mathscr{D}_M = \mathscr{P} \times 2^{\mathscr{D}_\ominus}$,

- $\to$ is the least relation such that:
  - $(b,\gamma) \xrightarrow{a[(X,\mathbb{1})/\overline{\alpha}^{\mathbb{1}}]} (b',\varepsilon)$ for each summand $\delta :\to a.\varepsilon \mathbin{\triangleleft} \alpha$ of X with $\delta \preccurlyeq \gamma$, where $b' = \mathtt{T} \iff \varepsilon \in \mathbb{1}[\alpha] \cap \mathbb{1}$,
  - $(b,\gamma) \xrightarrow{\tau[(X,\mathbb{1})/\epsilon]} (b,\gamma)$ iff $\gamma \in \mathbb{1}[X]$ and there is no summand $\delta :\to a.\varepsilon \mathbin{\triangleleft} \alpha$ of X with $\delta \preccurlyeq \gamma$, and
  - $\uparrow \xrightarrow{\tau[\epsilon/\overline{I}]} (b,\ominus)$, where $b = \mathtt{T} \iff \ominus \in \mathbb{1}[I]$.
- $\downarrow$ is the least set such that $\{\mathtt{T}\} \times \mathscr{D}_\ominus \subseteq \downarrow$, and $I\downarrow$ implies $\uparrow \in \downarrow$.

Example 6. Consider the VPGNF specification below.

$$X \stackrel{\text{def}}{=} \ominus :\to a.YZ + \ominus :\to \mathbf{1}$$
$$Y \stackrel{\text{def}}{=} \ominus :\to b.d \mathbin{\triangleleft} Z + \ominus :\to c.Z$$
$$Z \stackrel{\text{def}}{=} \sum_{\delta \in \mathscr{D}_\ominus} \delta :\to \mathbf{1}$$

The LTS associated with X is given on the left of Figure 5. The pushdown automaton M associated with X, as given by Definition 11 is rather large and difficult to interpret. Thus, we provide the PDP $\mathscr{L}(M)$ instead on the right of Figure 5.

Recall that the construction in Definition 11 uses τ-transitions to pop a terminating identifier off the stack. This is visible from the fact that states of the form $(s,\overline{Z\alpha})$ always have a τ-transition to $(s,\overline{\alpha})$, since the identifier Z is unconditionally terminating.

Our construction pushes sequences of the form $\overline{\alpha}^{\mathbb{1}}$ onto the stack. This is done such that the stack contents is always of the form $\overline{\beta}$. This information is used by the PDA to determine the accepting states: note that the only states reachable from $(\uparrow,\epsilon)$ are of the form $((b,\delta),\overline{\alpha})$ where $b = \mathtt{T}$ iff $\delta \in \mathbb{1}[\alpha]$.

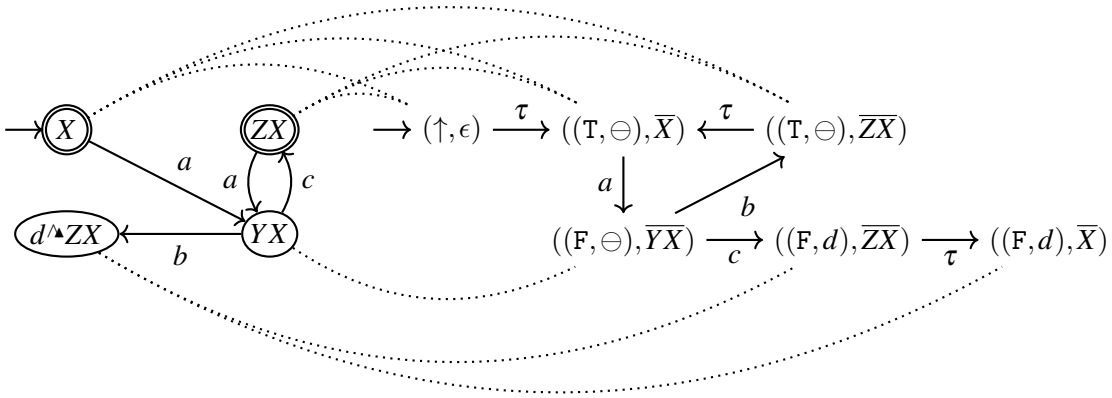

Figure 5: On the left is the LTS associated with the VPGNF specification in Example 6. On the right is the PDP associated with this specification, as described by Definition 11. The accepting states of this PDP are not indicated here, they are the states (s,x) for which $s = \uparrow$ or $s = (\mathtt{T},\delta)$ for some $\delta \in \mathscr{D}_\ominus$. The dotted lines indicate a branching bisimulation, the pairs of which are all (p,q) such that $p \cdots q$.

Theorem 3. *For every guarded recursive* TSVP *specification, there exists a branching bisimilar pushdown process.*

6 Conclusion

We investigated the problem of establishing a process-theoretic counterpart to the classical correspondence between pushdown automata and context-free grammars, which was previously considered solved by the work in [4, 7]. We found some errors in the constructions used, suggesting that a strong bisimilarity correspondence cannot be established.

Before correcting these errors, we first established an alternative semantics for sequential value passing, following the ideas in [7]. These semantics are less involved, using fewer operational rules than either of the previous theories. The operational rules can also be obtained from the rules for $\mathrm{TSP}^{;}$ in a straightforward manner. The resulting semantics correct some implicit assumptions in [7], such as the associativity of sequencing. The key to these semantics is that there are only two rules for deriving the structure of the TSVP transition system space, and that all other operational rules are only for deriving conditional behaviour or consistency.

We then establish a branching bisimilarity correspondence between pushdown processes and recursive TSVP processes. The τ-transitions used in our constructions show how a stack can emulate a sequencing by popping terminating identifiers off the stack, and how a sequencing can emulate a stack by introducing divergences that prevent the head of the sequencing from terminating.

Future Work. Firstly, we were not able to formally prove that TSVP processes and pushdown processes have different expressive power in bisimulation semantics, raising the question of whether or not the usage of branching bisimilarity is really necessary. A bisimilarity correspondence could also be established by restricting the scope to a certain class of PDPs and TSVP processes. For example, in [16] there is the notion of a *normed* pushdown automata, a type of pushdown automaton that can always execute an action when the stack is non-empty. Notably, the pushdown automaton used to justify the τ-transitions in the construction from PDPs to TSVP processes is not a normed pushdown automaton.

References

[1] L. Aceto, W. Fokkink & C. Verhoef (2001): *Structural Operational Semantics*. In: *Handbook of Process Algebra*, Elsevier Science, pp. 197–292, doi:10.1016/B978-044482830-9/50021-7.

[2] L. Aceto, W. J. Fokkink, A. Ingólfsdóttir & B. Luttik (2005): *Finite Equational Bases in Process Algebra: Results and Open Questions*. In A. Middeldorp, V. van Oostrom, F. van Raamsdonk & R. C. de Vrijer, editors: *Processes, Terms and Cycles: Steps on the Road to Infinity, Essays Dedicated to Jan Willem Klop, on the Occasion of His 60th Birthday*, *Lecture Notes in Computer Science* 3838, Springer, pp. 338–367, doi:10.1007/11601548_18.

[3] J. C. M. Baeten, T. Basten & M. A. Reniers (2009): *Process Algebra: Equational Theories of Communicating Processes*. Cambridge Tracts in Theoretical Computer Science, Cambridge University Press, Cambridge, doi:10.1017/CBO9781139195003.

[4] J. C. M. Baeten, C. Carissimo & B. Luttik (2023): *Pushdown Automata and Context-Free Grammars in Bisimulation Semantics*. *Logical Methods in Computer Science* Volume 19, Issue 1, doi:10.46298/lmcs-19(1:15)2023.

[5] J. C. M. Baeten, P. J. L. Cuijpers, B. Luttik & P. J. A. van Tilburg (2010): *A Process-Theoretic Look at Automata*. In F. Arbab & M. Sirjani, editors: *Fundamentals of Software Engineering*, Springer, Berlin, Heidelberg, pp. 1–33, doi:10.1007/978-3-642-11623-0_1.

[6] J. C. M. Baeten, P. J. L. Cuijpers & P. J. A. van Tilburg (2008): *A Context-Free Process as a Pushdown Automaton*. In F. van Breugel & M. Chechik, editors: *CONCUR 2008 - Concurrency Theory*, Springer, Berlin, Heidelberg, pp. 98–113, doi:10.1007/978-3-540-85361-9_11.

[7] J. C. M. Baeten & B. Luttik (2024): *Sequential Value Passing Yields a Kleene Theorem for Processes*. In V. Capretta, R. Krebbers & F. Wiedijk, editors: *Logics and Type Systems in Theory and Practice: Essays Dedicated to Herman Geuvers on The Occasion of His 60th Birthday*, Springer Nature Switzerland, Cham, pp. 1–16, doi:10.1007/978-3-031-61716-4_1.

[8] J. C. M. Baeten, B. Luttik & F. Yang (2017): *Sequential Composition in the Presence of Intermediate Termination (Extended Abstract)*. In: *EXPRESS/SOS 2017*, 255, pp. 1–17, doi:10.4204/EPTCS.255.1. ArXiv:1709.02440 [cs].

[9] A. Belder, B. Luttik & J. C. M. Baeten (2019): *Sequencing and Intermediate Acceptance: Axiomatisation and Decidability of Bisimilarity*. In M. Roggenbach & A. Sokolova, editors: *8th Conference on Algebra and Coalgebra in Computer Science (CALCO 2019)*, *Leibniz International Proceedings in Informatics (LIPIcs)* 139, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, pp. 11:1–11:22, doi:10.4230/LIPIcs.CALCO.2019.11. ISSN: 1868-8969.

[10] E. Costa Martins (2025): *Axiomatising Sequential Value Passing*. Master's thesis, Eindhoven University of Technology. Available at `https://research.tue.nl/en/studentTheses/axiomatising-sequential-value-passing/`.

[11] R. Erkens, J. Rot & B. Luttik (2020): *Up-to Techniques for Branching Bisimilarity*. In A. Chatzigeorgiou, R. Dondi, H. Herodotou, C. Kapoutsis, Y. Manolopoulos, G. A. Papadopoulos & F. Sikora, editors: *SOFSEM 2020: Theory and Practice of Computer Science*, Springer International Publishing, Cham, pp. 285–297, doi:10.1007/978-3-030-38919-2_24.

[12] J. F. Groote (1993): *Transition system specifications with negative premises*. *Theoretical Computer Science* 118(2), pp. 263–299, doi:10.1016/0304-3975(93)90111-6.

[13] R. Milner (1984): *A complete inference system for a class of regular behaviours*. *Journal of Computer and System Sciences* 28(3), pp. 439–466, doi:https://doi.org/10.1016/0022-0000(84)90023-0.

[14] M. Mousavi, M. A. Reniers & J. F. Groote (2004): *Congruence for SOS with data*. *Proceedings 19th IEEE Symposium on Logic in Computer Science (LICS 2004, Turku, Finland, July 14-17, 2004)*, pp. 303–312, doi:10.1109/LICS.2004.1319625. Publisher: Institute of Electrical and Electronics Engineers.

[15] D. Pous & D. Sangiorgi (2011): *Enhancements of the bisimulation proof method*. In D. Sangiorgi & J. Rutten, editors: *Advanced Topics in Bisimulation and Coinduction*, Cambridge Tracts in Theoretical Computer Science, Cambridge University Press, Cambridge, pp. 233–289, doi:10.1017/CBO9780511792588.007.

[16] C. Stirling (1998): *Decidability of bisimulation equivalence for normed pushdown processes*. *Theoretical Computer Science* 195(2), pp. 113–131, doi:10.1016/S0304-3975(97)00216-8.

A Proofs

We shall use the following notation to facilitate our proofs, where $\mathscr{R}$ is a binary relation:

- $\mathscr{R}^s$ denotes the symmetric closure of $\mathscr{R}$, i.e. $\mathscr{R}^s = \mathscr{R} \cup \{(q,p) \mid p\mathscr{R}q\}$.
- $\mathscr{I}$ denotes the identity relation.

The correspondence proofs use the following notion of a branching bisimulation.

Definition 12. A *branching bisimulation up-to strong bisimilarity* is a symmetric relation $\mathscr{R} \subseteq S \times S$, such that whenever $p\mathscr{R}q$, then for all $a \in \mathscr{A}$:

1. if $p \xrightarrow{a} p'$ for some $p' \in S$, then there exists $q', q'' \in S$ such that $q \twoheadrightarrow q' \xrightarrow{(a)} q''$, $p \underline{\leftrightarrow} \circ \mathscr{R} \circ \underline{\leftrightarrow} q'$ and $p' \underline{\leftrightarrow} \circ \mathscr{R} \circ \underline{\leftrightarrow} q''$; and
2. if $p\downarrow$, then there exists $q' \in S$ such that $q \twoheadrightarrow q'$, $p \underline{\leftrightarrow} \circ \mathscr{R} \circ \underline{\leftrightarrow} q'$ and $q'\downarrow$.

Whenever two processes are related by a branching bisimulation up-to strong bisimilarity, they are also branching bisimilar [11, 15].

Proposition 1. *Let p be some process expression. There exists at most one $\delta \in \mathscr{D}$ such that $p \mathscr{C} \delta$.*

Proof. Let p be a TSVP term. We prove by structural induction that there is at most one $\delta \in \mathscr{D}_\ominus$ such that $p \mathscr{C} \delta$.

1. If $p = \mathbf{0}$, $p = \mathbf{1}$, or $p = a.p'$, then only $p \mathscr{C} \ominus$ is derivable.
2. If $p = d :\!\blacktriangleright p'$, $p = \delta :\!\to p'$, or $p = \mathscr{C}_\delta(p')$, then only $p \mathscr{C} d$, $p \mathscr{C} \ominus$, or $p \mathscr{C} \delta$ is derivable, respectively.
3. If $p = NA(p')$, or $p = X$ and $X \stackrel{\text{def}}{=} p'$, then we may inductively assume there is at most one $\delta \in \mathscr{D}_\ominus$ such that $p' \mathscr{C} \delta$. If such a δ exists, then only $p \mathscr{C} \delta$ is derivable. Otherwise, p is inconsistent.
4. If $p = p' + p''$, then inductively there is at most one $(\delta, \varepsilon) \in \mathscr{D}_\ominus^2$ such that $p' \mathscr{C} \delta$ and $p'' \mathscr{C} \varepsilon$. If such a (δ, ε) exists, then $p \mathscr{C} \delta$ iff $\varepsilon \preccurlyeq \delta$, and $p \mathscr{C} \varepsilon$ iff $\delta \preccurlyeq \varepsilon$. If both hold, then $\delta = \varepsilon$. If δ and ε are incomparable, or if such a (δ, ε) does not exist, then p is inconsistent. In any case, p is consistent with at most one attribute.
5. If $p = p'; p''$, then inductively there is at most one $(\delta, \varepsilon) \in \mathscr{D}_\ominus^2$ such that $p' \mathscr{C} \delta$ and $p'' \mathscr{C} \varepsilon$. If such a pair exists, then if $\varepsilon = \ominus$, only $p \mathscr{C} \delta$ is derivable. If $\varepsilon \neq \ominus$, or if such a pair does not exist, then p is inconsistent.
6. If $p = \bot$, then there is no $\delta \in \mathscr{D}_\ominus$ such that $p \mathscr{C} \delta$.

□

Proposition 2. *Let p and q be process expressions. If for some $\delta \in \mathscr{D}_\ominus$, and $a \in \mathscr{A}$ it holds that $p \xrightarrow{\delta, a} q$, $p^\delta\!\downarrow$, or $q \xrightarrow{\delta, a} p$, then $p \mathscr{C}$.*

Proof. The first two statements are clear since every operational rule deriving $p^\delta\!\downarrow$ or $p \xrightarrow{\delta, a} q$ has a premise $p \mathscr{C} \gamma$. Note for AC1 and AC2: $x + y \mathscr{C} \gamma$ iff $y + x \mathscr{C} \gamma$. The last statement we prove by structural induction on q.

1. If $q = \mathbf{0}$, $q = \mathbf{1}$, or $q = \bot$, then the premise is false.
2. If $q = a.q'$ and $q \xrightarrow{\delta, a} p$, then this is derived by AP1, so $p = q'$ and $p \mathscr{C} \gamma$.
3. If $q = q' + q''$ and $q \xrightarrow{\delta, a} p$, then this is derived by AC1. Therefore, either $q' \xrightarrow{\delta, a} p$ or $q'' \xrightarrow{\delta, a} p$. In either case, we obtain inductively that $p \mathscr{C} \gamma$.
4. If $q = q'; q''$ and $q \xrightarrow{\delta, a} p$, then this is derived by S1 or S2.
   - If by S1, then $q' \xrightarrow{\delta, a} p'$, $p = p'; q''$, and $q \mathscr{C} \gamma$. Inductively it holds that $p' \mathscr{C} \gamma'$. Then, $q \mathscr{C} \gamma$ is derived by S0, so it holds that $q'' \mathscr{C} \ominus$. Therefore, $p \mathscr{C} \gamma'$ is derivable by S0.
   - If by S2, then $q'' \xrightarrow{\delta, a} p$ and the result holds inductively.
5. If $q = d :\!\blacktriangleright q'$, $q = \delta :\!\to q'$, $q = NA(q')$, $q = \mathscr{C}_\delta(q')$, or $q = X$ with $X \stackrel{\text{def}}{=} q'$, and $q \xrightarrow{\varepsilon, a} p$, then it must be the case that $q' \xrightarrow{\delta, a} p$, and thus the result holds by induction.

□

Proposition 4. *Stateless bisimilarity is finer than bisimilarity, and it is a congruence for* TSVP. *Moreover, $\leftrightarrows_{sl}$ is a stateless bisimulation.*

Proof. That stateless bisimilarity is an equivalence is clear from its definition. We show that $\leftrightarrows_{sl} \subseteq \leftrightarrow$ by showing that any stateless bisimulation is a bisimulation. Let $\mathscr{R}$ be a stateless bisimulation and let $p\mathscr{R}q$.

1. Suppose $p \xrightarrow{a} p'$. Then this is derived by O1, and hence there is some $\delta \in \mathscr{D}_{\ominus}$ such that $p \mathscr{C} \delta$ and $p \xrightarrow{\delta,a} p'$. Since $\mathscr{R}$ is a stateless bisimulation, it holds that $q \mathscr{C} \delta$ and $q \xrightarrow{\delta,a} q'$ for some q' such that $p'\mathscr{R}q'$. Then, by O1 it holds that $q \xrightarrow{a} q'$, as required.
2. Suppose $p\downarrow$. It follows analogously that $q\downarrow$.

Now, we show that $\leftrightarrows_{sl}$ is a stateless bisimulation. Take $p \leftrightarrows_{sl} q$, and let $\mathscr{R}$ be a stateless bisimulation such that $p\mathscr{R}q$.

1. Suppose $p \mathscr{C} \delta$. Then since $\mathscr{R}$ is a stateless bisimulation, it holds that $q \mathscr{C} \delta$.
2. Suppose $p \xrightarrow{\delta,a} p'$. Since $\mathscr{R}$ is a stateless bisimulation, it holds that $q \xrightarrow{\delta,a} q'$ for some q' such that $p'\mathscr{R}q'$. Since $\mathscr{R}$ is a stateless bisimulation and $p'\mathscr{R}q'$, it holds that $p' \leftrightarrows_{sl} q'$.
3. Analogous to (1).

Lastly, we show that $\leftrightarrows_{sl}$ is a congruence[1]. Let p_1, p_2, q_1, q_2 be TSVP processes such that $p_1 \leftrightarrows_{sl} p_2$ and $q_1 \leftrightarrows_{sl} q_2$. We show compatibility of $\leftrightarrows_{sl}$ with each TSVP operator, including auxiliary ones. Let $\mathscr{R}_p$ and $\mathscr{R}_q$ be stateless bisimulations such that $p_1\mathscr{R}_p p_2$ and $q_1\mathscr{R}_q q_2$.

Action Prefix. We show that $a.p_1 \leftrightarrows_{sl} a.p_2$ by showing that

$$\mathscr{R} \coloneqq \mathscr{R}_p \cup \{(a.p_1, a.p_2)\}^s$$

is a stateless bisimulation. Take $(a.p_1, a.p_2) \in \mathscr{R}$.

1. Clearly $a.p_1 \mathscr{C} \ominus$ and $a.p_2 \mathscr{C} \ominus$.
2. Suppose $a.p_1 \xrightarrow{\delta,b} p_1'$. Then this is derived by AP1, and therefore $a = b, p_1 = p_1'$. By AP1, also $a.p_2 \xrightarrow{\delta,a} p_2$ is derivable, and we are done since $p_1\mathscr{R}p_2$.
3. The premise is false.

The proof for the pair $(a.p_2, a.p_1) \in \mathscr{R}$ is analogous. The proofs for the remaining pair are trivial. Therefore stateless bisimilarity is compatible with action prefix.

Alternative Composition. We show that $p_1 + q_1 \leftrightarrows_{sl} p_2 + q_2$ by showing that

$$\mathscr{R} \coloneqq \mathscr{R}_p \cup \mathscr{R}_q \cup \{(p_1 + q_1, p_2 + q_2)\}^s$$

is a stateless bisimulation. Take $(p_1 + q_1, p_2 + q_2) \in \mathscr{R}$.

1. Suppose $p_1 + q_1 \mathscr{C} \delta$. Then this is derived by AC0, so there is some $\varepsilon \in \mathscr{D}_{\ominus}$ such that $\varepsilon \preccurlyeq \delta$ and either $p_1 \mathscr{C} \delta$ and $q_1 \mathscr{C} \varepsilon$, or vice-versa. W.l.o.g. assume the former. Then since $p_1\mathscr{R}_p p_2$ and $q_1\mathscr{R}_q q_2$, it holds that $p_2 \mathscr{C} \delta$ and $q_2 \mathscr{C} \varepsilon$. Thus, by AC0 it holds that $p_2 + q_2 \mathscr{C} \delta$, as required.
2. Suppose $p_1 + q_1 \xrightarrow{\delta,a} p_1'$. This is derived by AC1, so it is the case that either $p_1 \xrightarrow{\delta,a} p_1'$ or $q_1 \xrightarrow{\delta,a} p_1'$. W.l.o.g. assume the former. Then since $p_1\mathscr{R}_p p_2$, there exists some p_2' such that $p_2 \xrightarrow{\delta,a} p_2'$ and $p_1'\mathscr{R}_p p_2'$. Then by AC1 we derive $p_2 + q_2 \xrightarrow{\delta,a} p_2'$, and we are done since $p_1'\mathscr{R}p_2'$.
3. Analogous.

The proof for the pair $(p_2 + q_2, p_1 + q_1) \in \mathscr{R}$ is analogous. The proofs for the remaining pair are trivial. Therefore stateless bisimilarity is compatible with alternative composition.

[1] That it is a congruence is relatively straightforward from the format of the operational rules.

Sequencing. We show that $p_1;q_1 \leftrightarrows_{sl} p_2;q_2$ by showing that

$$\mathscr{R} \coloneqq \mathscr{R}_q \cup \{(p_1';q_1,p_2';q_2) \mid p_1' \mathscr{R}_p p_2'\}^s$$

is a stateless bisimulation. Take $(p_1';q_1,p_2';q_2) \in \mathscr{R}$.

1. Suppose $p_1';q_1 \mathscr{C} \delta$. This is derived by S0, and therefore $p_1' \mathscr{C} \delta$ and $q_1 \mathscr{C} \ominus$. Since $p_1' \mathscr{R}_p p_2'$ and $q_1 \mathscr{R}_q q_2$, it holds that $p_2' \mathscr{C} \delta$ and $q_2 \mathscr{C} \ominus$. Therefore, S0 derives $p_2';q_2 \mathscr{C} \delta$, as required.
2. Suppose $p_1';q_1 \xrightarrow{\delta,a} q_1'$. This is derived by S1 or S2.
   - If by S1, then $p_1' \xrightarrow{\delta,a} p_1''$ and $q_1' = p_1'';q_1$. Since $p_1' \mathscr{R}_p p_2'$, it holds that $p_2' \xrightarrow{\delta,a} p_2''$ for some p_2'' such that $p_1'' \mathscr{R}_p p_2''$. Therefore, S1 derives $p_2';q_2 \xrightarrow{\delta,a} p_2'';q_2$ and we are done since $(p_1'';q_1,p_2'';q_2) \in \mathscr{R}$.
   - If by S2, then $p_1' \overset{\delta}{\nrightarrow}$, ${p_1'}^{\delta}\downarrow$, and $q_1 \xrightarrow{\delta,a} q_1'$. Since $p_1' \mathscr{R}_p p_2'$ and $q_1 \mathscr{R}_q q_2$, it holds that $p_2' \overset{\delta}{\nrightarrow}$, ${p_2'}^{\delta}\downarrow$, and $q_2 \xrightarrow{\delta,a} q_2'$ for some q_2' such that $q_1' \mathscr{R}_q q_2'$. Then S2 derives $p_2';q_2 \xrightarrow{\delta,a} q_2'$ and we are done since $(q_1',q_2') \in \mathscr{R}$.
3. Suppose $p_1';q_1{}^{\delta}\downarrow$. This is derived by S3, and therefore $p_1'{}^{\delta}\downarrow$ and $q_1{}^{\delta}\downarrow$. Since $p_1' \mathscr{R}_p p_2'$ and $q_1 \mathscr{R}_q q_2$, it holds that $p_2'{}^{\delta}\downarrow$ and $q_2{}^{\delta}\downarrow$. By S3, we derive $p_2';q_2{}^{\delta}\downarrow$, as required.

The proof for the pair $(p_2';q_2,p_1';q_1) \in \mathscr{R}$ is analogous. The proofs for the remaining pair are trivial. Therefore stateless bisimilarity is compatible with sequencing.

Attribute Assignment. We show that $d^{\wedge\blacktriangle}p_1 \leftrightarrows_{sl} d^{\wedge\blacktriangle}p_2$ by showing that

$$\mathscr{R} \coloneqq \mathscr{R}_p \cup \{(d^{\wedge\blacktriangle}p_1, d^{\wedge\blacktriangle}p_2)\}^s$$

is a stateless bisimulation. Take $(d^{\wedge\blacktriangle}p_1, d^{\wedge\blacktriangle}p_2) \in \mathscr{R}$.

1. Suppose $d^{\wedge\blacktriangle}p_1 \mathscr{C} \delta$. Then clearly $\delta = d$, and by AA0 it holds that $p_1 \mathscr{C} \varepsilon$ for some $\varepsilon \preccurlyeq \delta$. Since $p_1 \mathscr{R}_p p_2$, it holds that $p_2 \mathscr{C} \varepsilon$, and hence $d^{\wedge\blacktriangle}p_2 \mathscr{C} d$.
2. Suppose $d^{\wedge\blacktriangle}p_1 \xrightarrow{\delta,a} p_1'$. Then this is derived by AA1, and therefore $p_1 \xrightarrow{\delta,a} p_1'$. Since $p_1 \mathscr{R}_p p_2$, it holds that $p_2 \xrightarrow{\delta,a} p_2'$ for some p_2' such that $p_1' \mathscr{R}_p p_2'$. Then AA1 derives $d^{\wedge\blacktriangle}p_2 \xrightarrow{\delta,a} p_2'$ and we are done since $p_1' \mathscr{R} p_2'$.
3. Analogous.

The proof for the pair $(d^{\wedge\blacktriangle}p_2, d^{\wedge\blacktriangle}p_1) \in \mathscr{R}$ is analogous. The proofs for the remaining pair are trivial. Therefore stateless bisimilarity is compatible with attribute assignment.

Guarded Command. We show that $\gamma{:}\rightarrow p_1 \leftrightarrows_{sl} \gamma{:}\rightarrow p_2$ by showing that

$$\mathscr{R} \coloneqq \mathscr{R}_p \cup \{(\gamma{:}\rightarrow p_1, \gamma{:}\rightarrow p_2)\}^s$$

is a stateless bisimulation. Take $(\gamma{:}\rightarrow p_1, \gamma{:}\rightarrow p_2) \in \mathscr{R}$.

1. Suppose $\gamma{:}\rightarrow p_1 \mathscr{C} \delta$. Then clearly $\delta = \ominus$, and by G0 it holds that $p_1 \mathscr{C} \varepsilon$ for some $\varepsilon \preccurlyeq \gamma$. Since $p_1 \mathscr{R}_p p_2$, it holds that $p_2 \mathscr{C} \varepsilon$, and hence $\gamma{:}\rightarrow p_2 \mathscr{C} \ominus$.

2. Suppose $\gamma :\to p_1 \xrightarrow{\delta,a} p_1'$. Then this is derived by G1, and therefore $\delta = \gamma$ and $p_1 \xrightarrow{\gamma,a} p_1'$. Since $p_1 \mathscr{R}_p p_2$, it holds that $p_2 \xrightarrow{\gamma,a} p_2'$ for some p_2' such that $p_1' \mathscr{R}_p p_2'$. Then G1 derives $\gamma :\to p_2 \xrightarrow{\gamma,a} p_2'$ and we are done since $p_1' \mathscr{R} p_2'$.
3. Analogous.

The proof for the pair $(\gamma :\to p_2, \gamma :\to p_1) \in \mathscr{R}$ is analogous. The proofs for the remaining pair are trivial. Therefore stateless bisimilarity is compatible with guarded command.

Non-acceptance. We show that $NA(p_1) \leftrightarrows_{sl} NA(p_2)$ by showing that

$$\mathscr{R} := \mathscr{R}_p \cup \{(NA(p_1), NA(p_2))\}^s$$

is a stateless bisimulation. Take $(NA(p_1), NA(p_2)) \in \mathscr{R}$.

1. Suppose $NA(p_1) \mathscr{C} \delta$. This is derived by NA0, and hence $p_1 \mathscr{C} \delta$. Since $p_1 \mathscr{R}_p p_2$, it holds that $p_2 \mathscr{C} \delta$, and hence $NA(p_2) \mathscr{C} \delta$.
2. Suppose $NA(p_1) \xrightarrow{\delta,a} p_1'$. Then this is derived by NA1, and therefore $p_1 \xrightarrow{\delta,a} p_1'$. Since $p_1 \mathscr{R}_p p_2$, it holds that $p_2 \xrightarrow{\delta,a} p_2'$ for some p_2' such that $p_1' \mathscr{R}_p p_2'$. Then NA1 derives $NA(p_2) \xrightarrow{\gamma,a} p_2'$ and we are done since $p_1' \mathscr{R} p_2'$.
3. The premise is false.

The proof for the pair $(NA(p_2), NA(p_1)) \in \mathscr{R}$ is analogous. The proofs for the remaining pair are trivial. Therefore stateless bisimilarity is compatible with non-acceptance.

Consistency Assertion. We show that $\mathscr{C}_\delta(p_1) \leftrightarrows_{sl} \mathscr{C}_\delta(p_2)$ by showing that

$$\mathscr{R} := \mathscr{R}_p \cup \{(\mathscr{C}_\delta(p_1), \mathscr{C}_\delta(p_2))\}^s$$

is a stateless bisimulation. Take $(\mathscr{C}_\delta(p_1), \mathscr{C}_\delta(p_2)) \in \mathscr{R}$.

1. Suppose $\mathscr{C}_\delta(p_1) \mathscr{C} \varepsilon$. This is derived by C0, and hence $\varepsilon = \delta$. By G0, also $\mathscr{C}_\delta(p_2) \mathscr{C} \delta$.
2. Suppose $\mathscr{C}_\delta(p_1) \xrightarrow{\varepsilon,a} p_1'$. Then this is derived by C1, and therefore $p_1 \xrightarrow{\varepsilon,a} p_1'$. Since $p_1 \mathscr{R}_p p_2$, it holds that $p_2 \xrightarrow{\varepsilon,a} p_2'$ for some p_2' such that $p_1' \mathscr{R}_p p_2'$. Then C1 derives $\mathscr{C}_\delta(p_2) \xrightarrow{\gamma,a} p_2'$ and we are done since $p_1' \mathscr{R} p_2'$.
3. Analogous

The proof for the pair $(\mathscr{C}_\delta(p_2), \mathscr{C}_\delta(p_1)) \in \mathscr{R}$ is analogous. The proofs for the remaining pair are trivial. Therefore stateless bisimilarity is compatible with consistency assertion. □

Lemma 1. *Let $\mathscr{D}', \mathscr{D}'' \subseteq \mathscr{D}_\ominus$. The following equations are derivable using the axioms in Table 9:*

$$\mathscr{D}' :\to x + \mathscr{D}'' :\to x = \mathscr{D}' \cup \mathscr{D}'' :\to x$$
$$\mathscr{D}' :\to (\mathscr{D}'' :\to \mathscr{C}_\ominus(x)) = \mathscr{D}' \cap \mathscr{D}'' :\to \mathscr{C}_\ominus(x) \quad .$$

Proof. The first equation is merely notation. The second equation follows from Axioms G3 and G4. □

Lemma 2. *Let $p \neq \bot$ be in HNF, and let $\gamma \in \mathscr{D}_\ominus$ be its attribute value. Then $p = \mathscr{C}_\gamma(p)$ is derivable using equational logic and the axioms of* TSVP.

Proof. Straightforward using the axioms for consistency. □

Lemma 3. *Let $p \neq \bot$ be in HNF. Then $p = NA(p) + \mathbb{1}[p] :\to \mathbf{1}$ is derivable using equational logic and the axioms of* TSVP.

Proof. Straightforward using the axioms for non-acceptance. □

Proposition 5. *Every guarded* TSVP *process p is derivably equal to some expression in HNF using equational logic and the axioms of* TSVP.

Proof. Let p be a guarded TSVP term. We prove by induction on the structure of p, that it is derivably equal to some expression in HNF. We only show the case for $p = p';p''$, as it is most involved. It relies on the correctness of the other cases, which are relatively straightforward.

For p to be guarded, both p' and p'' must be guarded. Thus, we inductively assume p' and p'' are in HNF.

Case 1. First, consider the cases where p is inconsistent:

- If $p' = \bot$, then $p = \bot$ is derivable by Axiom B6.
- If $p'' = \bot$, then $p = \bot$ is derivable by Axioms B3 and B7.
- If p'' has attribute value $\gamma'' \neq \ominus$, then $p = \bot$ is derivable by Axiom B7.

It only remains to consider the cases where p is consistent, thus:

$$p' = \gamma'^{\blacktriangle}\Big(\sum_{\delta\in\mathscr{D}} \delta :\to p'_\delta\Big)$$
$$p'' = \sum_{\delta\in\mathscr{D}} \delta :\to p''_\delta \quad .$$

As an intermediary result, we show that $p'_\delta;p''$ can be brought to HNF. Write:

$$p'_\delta = \sum_{i=1}^{n} a_i.\mathscr{C}_{\varepsilon_i}(p_i)\ [+\mathbf{1}] \quad .$$

We distinguish two cases. If $n = 0$, then using Axioms A7 and A9, we have $p'_\delta;p'' = \mathbf{0}$ or $p'_\delta;p'' = p''$, depending on whether or not p'_γ has the $\mathbf{1}$ summand. Since $\mathbf{0}$ can be brought to HNF, and p'' is in HNF, we conclude this case. Now, if $n > 0$, then using Axiom A12, we have

$$p'_\delta;p'' = \Big(\sum_{i=1}^{n} a_i.\mathscr{C}_{\varepsilon_i}(p_i)\Big);p''\ [+\mathbb{1}[p''] :\to \mathbf{1}] \quad ,$$

where the $[+\mathbb{1}[p''] :\to \mathbf{1}]$ summand is there iff p'_γ has the $\mathbf{1}$ summand. Using the axioms for non-acceptance and Axioms A10, A11 and C7, we obtain:

$$p'_\delta;p'' = \sum_{i=1}^{n} a_i.\mathscr{C}_{\varepsilon_i}(p_i;p'')\ [+\mathbb{1}[p''] :\to \mathbf{1}]$$

Lastly, using Axiom G5 and by commuting the individual parts of the $[+\mathbb{1}[p''] :\to \mathbf{1}]$ summand (if it exists), we obtain an expression in HNF, as required.

With the intermediary result established, we distinguish two cases depending on whether or not $\gamma' = \ominus$.

Case 2. First, consider the case where $\gamma' \neq \ominus$. We may use Axioms S1–S3, G1, and A6 to write:

$$p = (\gamma' \curvearrowright p'_{\gamma'}); p'' \quad .$$

Using Axioms A9, A5, and A9 again, we obtain:

$$p = \gamma' \curvearrowright (p'_{\gamma'}; p'') \quad .$$

We have shown above that $p'_{\gamma'}; p''$ can be brought to HNF. Since $\delta \curvearrowright q$ can be brought to HNF whenever q is HNF, we conclude this case.

Case 3. Now, consider the case where $\gamma' = \ominus$. Using Axiom G5, we obtain:

$$p = \sum_{\delta \in \mathscr{D}_\ominus} \delta :\rightarrow (p'_\delta; p'') \quad .$$

Each $p'_\delta; p''$ can be brought to HNF. Since $\delta :\rightarrow q$ and $q + q'$ can be brought to HNF whenever q, q' are in HNF, we conclude this case. □

Lemma 4. *The following equations are derivable using the axioms in Tables 8 to 10:*

$$\delta \curvearrowright (x; y) = (\delta \curvearrowright x); y$$
$$\delta :\rightarrow (x; y) = (\delta :\rightarrow x); y \quad .$$

Proof. First consider $\delta \curvearrowright (x; y)$. Note that if $y \mathscr{C} \ominus$ (which can be determined from the normal form of y), then:

$$\begin{aligned} \delta \curvearrowright (x; y) &\overset{\text{A9}}{=} (\delta \curvearrowright \mathbf{1}); (x; y) \\ &\overset{\text{A5}}{=} ((\delta \curvearrowright \mathbf{1}); x); y \\ &\overset{\text{A9}}{=} (\delta \curvearrowright x); y \quad . \end{aligned}$$

The same result can be obtained if $y \mathscr{C} \ominus$ does not hold, since then both sides of the equation are inconsistent.

Now consider $\delta :\rightarrow (x; y)$. As before, consider $y \mathscr{C} \ominus$. Write x in HNF, and consider the attribute value γ of x. If $\gamma \notin \{\delta, \ominus\}$, then the expression is inconsistent, as is $(\delta :\rightarrow x); y$. Otherwise, if $\gamma = \ominus$, then we may consider the equivalent expression x' with no attribute assignment, due to the previous result and Axiom G4. Then, we can derive $x' = \mathscr{C}_\ominus(x')$. From this, we may use the axioms for consistency and obtain $\delta :\rightarrow (x; y) = \mathscr{C}_\ominus(\delta :\rightarrow (x; y))$. Note that $\varepsilon :\rightarrow (\mathbf{0}; y) = \mathbf{0}$, thus we may consider the expression:

$$\sum_{\varepsilon \in \mathscr{D}_\ominus} \varepsilon :\rightarrow (x_\varepsilon; y)$$

where $x_\varepsilon = x$ if $\varepsilon = \delta$, and otherwise $x_\varepsilon = \mathbf{0}$. Then using G5 and removing the redundant guards, we obtain the desired result, $(\delta :\rightarrow x); y$. □

Theorem 1. *Let p, q be recursion-free* TSVP *processes such that $p \leftrightarrows_{sl} q$. Then $p = q$ is derivable using the axioms in Tables 8 to 10.*

Proof. The proof is similar to the one in [10]. □

Proposition 6. *For every guarded recursive specification Δ with initial identifier $X_\uparrow$, there exists a recursive specification Δ' in VPGNF with initial identifier $X'_\uparrow$ such that $X_\uparrow \leftrightarrows X'_\uparrow$.*

Proof. The proof is given in two parts. First, we show that for each guarded recursive specification Δ with a *consistent* initial identifier $X_\uparrow$, there exists a recursive specification Δ' with initial identifier $X'_\uparrow$ such that $X_\uparrow \leftrightarrows_{sl} X'_\uparrow$, where each defining equation of Δ' is in *Greibach Normal Form* (GNF):

$$X \stackrel{\text{def}}{=} \gamma^{\wedge\blacktriangle}\Big(\sum_{\delta\in\mathscr{D}} \delta :\to p_\delta\Big) \qquad \text{with each } p_\delta \text{ of the form}$$

$$p_\delta = \sum_{i=1}^{n} a_i.\mathscr{C}_{\varepsilon_i}(\alpha_i)\ [+\mathbf{1}]$$

where α_i is a sequence of process identifiers.

We show how to transform such a Δ into a specification Δ' with each defining equation in Δ' in GNF. First, convert each defining equation in Δ to HNF. Then, for each $X \in \mathscr{P}$, each summand $\delta :\to p_\delta$ of X, and each summand $a.\mathscr{C}_\varepsilon(p)$ of p_δ, replace p by a fresh identifier X_p with $\Delta(X_p) = p$. Now, note that each original identifier of $\mathscr{P}$ is in the required form (although the new ones are not). Let S be a stack data structure, consider the set $P \subseteq \mathscr{P}$ of newly created process identifiers, and apply the following subroutine:

1. Push P on the stack S.
2. For each $X_p \in P$, do the following:
   (a) Let f be the outermost function symbol of $\Delta(X_p)$, i.e. $\Delta(X_p) = f(p_1,\ldots,p_n)$ with f of arity n.
   (b) For each p_i that is not an identifier, create a new equation $X_{p_i} \stackrel{\text{def}}{=} p_i$ with X_{p_i} a fresh identifier.
   (c) Redefine $\Delta(X_p) = f(X_{p_1},\ldots,X_{p_n})$.
3. Recurse on the set P' of newly created process identifiers, unless $P' = \emptyset$.

After this subroutine, it holds that for each set P on the stack S, the identifiers in P are defined only in terms of original identifiers of $\mathscr{P}$ (which are in GNF), or identifiers that belong to some P' that are stored above P on S. It can be shown by structural induction that whenever p is a process expression such that $p = f(X_1,\ldots,X_n)$ for f a function symbol of arity n and the defining equations for $X_1,\ldots,X_n$ are in VPGNF, then p can be converted to GNF without introducing any other identifiers. Thus it suffices to pop the set P on the top of S, apply this procedure to $\Delta(X)$ for each $X \in \mathscr{P}$, and repeat until S is empty. The resulting specification is in GNF. It is not too hard to see that each identifier X of a specification in GNF can equivalently be defined in the form:

$$X \stackrel{\text{def}}{=} \gamma^{\wedge\blacktriangle}\left(\sum_{i=1}^{n} \delta_i :\to a_i.\mathscr{C}_{\varepsilon_i}(\alpha)\right) + \mathbb{1} :\to \mathbf{1}$$

by commuting and distributing summands and guards appropriately.

Next, we show how to transform a specification Δ in GNF to VPGNF. We fix process identifiers $X_\varepsilon \stackrel{\text{def}}{=} \varepsilon^{\wedge\blacktriangle}\mathbf{0}$ for each $\varepsilon \in \mathscr{D}_\ominus$. Note that the consistency of a sequence $\alpha \in \mathscr{P}^*$ can easily be determined by the attribute values of the defining equations of each identifier in α. Therefore, each sub-expression $a.\mathscr{C}_\varepsilon(\alpha)$ can be replaced by $a.\alpha$ iff $\alpha\,\mathscr{C}\,\varepsilon$, or $a.X_\varepsilon$ otherwise. Now whenever a sequence α occurs in a defining equation of some identifier, it holds that α is consistent. Therefore, only the first identifier in α

has an attribute value. Now, replace each sequence $\alpha = X\overrightarrow{\alpha}$ in a defining equation by $\varepsilon \blacktriangleright X^{\dagger}\overrightarrow{\alpha}$, where ε is the attribute value of X, and $X^{\dagger}$ is defined as X but without any attribute assigned. It holds that $\alpha \leftrightarrow_{sl} \varepsilon \blacktriangleright X^{\dagger}\overrightarrow{\alpha}$ due to the distributivity of attribute assignment over sequencing. Now the only reachable identifiers (those that occur in the defining equation of the initial identifier, or in defining equations of those identifiers, etc.) are those without any attribute value, except for the initial identifier itself. Remove all non-initial identifiers from the specification which have an attribute value. Now all defining equations are in VPGNF, except for that of the initial identifier, because it still has an attribute γ. First, use Axioms S2 and S3 to remove all guards. Then, the only transformations that does not preserve stateless bisimilarity is the removal of the attribute assignment. Bisimilarity is still preserved since the initial identifier can reach all the same states. Lastly, to bring the initial identifier to VPGNF, reintroduce the guards using Axiom G5.

The result has been proven for specifications with a consistent initial identifier. If the identifier is inconsistent, then we need only take the specification Δ' with initial identifier $X_{\bot} \stackrel{\text{def}}{=} \mathbf{0}$ since $\bot \leftrightarrow \mathbf{0}$. □

Lemma 5. *Let $\alpha \in \mathscr{P}^{+}$ be a non-empty sequence of process identifiers in VPGNF, and let $\gamma \in \mathscr{D}_{\ominus}$. If there exists some $a \in \mathscr{A}$ and* TSVP *process p' such that $\gamma \blacktriangleright \alpha \xrightarrow{a} p'$, then there exists $X \in \mathscr{P}, \alpha', \alpha'' \in \mathscr{P}^{*}, \beta \in \mathscr{P}^{+}$, and $\varepsilon \in \mathscr{D}_{\ominus}$ such that:*

- $\alpha = \alpha'' X \alpha'$,
- $Y^{\gamma}\!\downarrow$ *and* $Y \xrightarrow{\gamma_{\ominus}}$ *for each identifier Y in α'', and*
- $X \xrightarrow{\gamma,a} \varepsilon \blacktriangleright \beta$ *and* $p' \leftrightarrow \varepsilon \blacktriangleright \beta \alpha'$.

Proof. Let $\alpha \in \mathscr{P}^{+}$, $\gamma \in \mathscr{D}_{\ominus}$, $a \in \mathscr{A}$, and p' be such that $\gamma \blacktriangleright \alpha \xrightarrow{a} p'$. We proceed by induction on α. In case $\alpha = \epsilon$, the statement is vacuously true since $\mathbf{1} \not\rightarrow$. In case $\alpha = X\alpha'$ for some $X \in \mathscr{P}$, $\alpha' \in \mathscr{P}^{*}$, then $\alpha = X;\alpha'$. Then $\gamma \blacktriangleright \alpha \xrightarrow{a} p'$ is derived by O1, meaning $\gamma \blacktriangleright \alpha \xrightarrow{\delta,a} p'$ and $\gamma \blacktriangleright \alpha \mathscr{C} \delta$. Clearly $\delta = \gamma$ since each identifier in VPGNF is consistent with $\ominus$. Then, $\gamma \blacktriangleright \alpha \xrightarrow{\delta,a} p'$ is derived by either S1 or S2.

- If by S1, then it holds that $X \xrightarrow{\gamma,a} p$ and $p' = p;\alpha'$. Since X is in VPGNF, it holds that $p = \varepsilon \blacktriangleright \beta$ for some $\beta \in \mathscr{P}^{+}$ and $\varepsilon \in \mathscr{D}_{\ominus}$. Then we are done, since:
  - $\alpha = \epsilon X \alpha'$, and
  - $X \xrightarrow{\gamma,a} \varepsilon \blacktriangleright \beta$ and $p' \leftrightarrow_{sl} \varepsilon \blacktriangleright \beta \alpha'$ by Lemma 4.
- If by S2, then $X \xrightarrow{\gamma_{\ominus}}, X^{\gamma}\!\downarrow$, and $\alpha' \xrightarrow{\gamma,a} p'$. Inductively, it holds that:
  - $\alpha' = \alpha''' Y \alpha''$,
  - $Z^{\gamma}\!\downarrow$ and $Z \xrightarrow{\gamma_{\ominus}}$ for each identifier Z in α''', and
  - $Y \xrightarrow{\gamma,a} \varepsilon \blacktriangleright \beta$ and $p' \leftrightarrow \varepsilon \blacktriangleright \beta \alpha''$.

  Then we may write $\alpha = X\alpha''' Y \alpha''$, with the prefix being $X\alpha'''$ and the suffix being α'', and we are done.

□

Theorem 2. *For every pushdown process, there exists a branching bisimilar guarded recursive* TSVP *specification.*

Proof. Let $M = \langle S, \mathscr{A}, \mathscr{D}_M, \rightarrow, \uparrow, \downarrow \rangle$ be some pushdown automaton. Let Δ be the TSVP specification associated with M, as given by Definition 10. Let $\mathscr{R}$ be the symmetric closure of the following relation:

$$\{(s \mathbin{^\frown_\blacktriangle} \alpha_x^{\mathsf{T}}, \langle s, x \rangle) \mid s \in S, x \in \mathscr{D}_M^*\}$$

We show that $\mathscr{R}$ is a branching bisimulation up-to strong bisimilarity. First, consider a pair $(s \mathbin{^\frown_\blacktriangle} \alpha_x^{\mathsf{T}}, \langle s, x \rangle)$.

1. Suppose $s \mathbin{^\frown_\blacktriangle} \alpha_x^{\mathsf{T}} \xrightarrow{a} p'$. Note that by construction of Δ, $s \mathbin{^\frown_\blacktriangle} X \xrightarrow{\tau}$ for each identifier $X \neq I$, so $s \mathbin{^\frown_\blacktriangle} \alpha_x^{\mathsf{T}}$ does not have a terminating prefix. From this and Lemma 5, it follows that there exists $x' \in \mathscr{D}_M^*, \delta \in \mathscr{D}_M \cup \{\epsilon\}$ and $t \in S$ such that:
   - $x = \delta \overrightarrow{x'}$,
   - $X_\delta \xrightarrow{s,a} t \mathbin{^\frown_\blacktriangle} \alpha_{x'}^{\delta=\epsilon}$ and $p' \leftrightarrows t \mathbin{^\frown_\blacktriangle} \alpha_{x' \overrightarrow{x}}^{\mathsf{T}}$.

   We distinguish two cases.
   - If $a = \tau, t = s$, and $x' = \delta$, then we are done since $p' \leftrightarrows s \mathbin{^\frown_\blacktriangle} \alpha_x^{\mathsf{T}}$, and $s \mathbin{^\frown_\blacktriangle} \alpha_x^{\mathsf{T}}$ and $\langle s, x \rangle$ are related by $\mathscr{R}$.
   - Otherwise, it holds by construction of Δ that $s \xrightarrow{a[\delta/x']} t$ is a transition of M, and hence $\langle s, x \rangle \xrightarrow{a} \langle t, x'x \rangle$. Then we are done since $t \mathbin{^\frown_\blacktriangle} \alpha_{x' \overrightarrow{x}}^{\mathsf{T}}$ and $\langle t, x'x \rangle$ are related by $\mathscr{R}$.
2. Suppose $s \mathbin{^\frown_\blacktriangle} \alpha_x^{\mathsf{T}} \downarrow$. By the operational semantics, it holds that $X^s \downarrow$ for each identifier X in the sequence α_x^{T}. By construction of Δ, this requires a summand $s :\rightarrow \mathbf{1}$ for some state s that is final in M. Therefore, $\langle s, x \rangle \downarrow$.

Next, consider a pair $(\langle s, x \rangle, s \mathbin{^\frown_\blacktriangle} \alpha_x^{\mathsf{T}})$.

1. Suppose $\langle s, x \rangle \xrightarrow{a} \langle t, y \rangle$. Since $x = \delta \overrightarrow{x'}$ for some $\delta \in \mathscr{D}_M \cup \{\epsilon\}$, we have $y = x' \overrightarrow{x}$ for some transition $s \xrightarrow{a[\delta/x']} t$ of M. By construction of Δ, $X_\delta \xrightarrow{s,a} t \mathbin{^\frown_\blacktriangle} \alpha_{x'}^{\delta=\epsilon}$, and therefore $s \mathbin{^\frown_\blacktriangle} \alpha_x^{\mathsf{T}} \xrightarrow{a} t \mathbin{^\frown_\blacktriangle} \alpha_{x' \overrightarrow{x}}^{\mathsf{T}}$ follows. Then we are done since $\langle t, x' \overrightarrow{x} \rangle$ and $t \mathbin{^\frown_\blacktriangle} \alpha_{x' \overrightarrow{x}}^{\mathsf{T}}$ are related by $\mathscr{R}$.
2. Suppose $\langle s, x \rangle \downarrow$. Then s is final in M, and thus each identifier α_x^{T} has a summand $s :\rightarrow \mathbf{1}$. By the operational semantics, it holds that $s \mathbin{^\frown_\blacktriangle} \alpha_x^{\mathsf{T}} \downarrow$ as required.

Thus, $\mathscr{R}$ is a branching bisimulation up-to strong bisimilarity. Since $\uparrow \mathbin{^\frown_\blacktriangle} X_\epsilon \leftrightarrows_b \langle \uparrow, \epsilon \rangle$ and $I = \uparrow \mathbin{^\frown_\blacktriangle} X_\epsilon$, it holds that $I \leftrightarrows_b \langle \uparrow, \epsilon \rangle$, as required. □

Theorem 3. *For every guarded recursive* TSVP *specification, there exists a branching bisimilar pushdown process.*

Proof. Throughout this proof we shall, as an abuse of notation, use a proposition to denote its underlying Boolean value. Otherwise, it becomes cumbersome to define the Booleans in the branching bisimulation.

Let Δ be some guarded TSVP recursive specification with initial identifier $X_\uparrow$. We assume w.l.o.g. that Δ is in VPGNF. Let M be the PDA associated with Δ, as given by Definition 11. Let $\mathscr{R}$ be the symmetric closure of the following relation.

$$\{(\gamma \mathbin{^\frown_\blacktriangle} \alpha, \langle \langle \gamma \in \mathbb{1}[\alpha], \gamma \rangle, \overline{\alpha} \rangle) \mid \gamma \in \mathscr{D}_\ominus, \alpha \in \mathscr{P}^+\} \cup \{(X_\uparrow, \langle \uparrow', \epsilon \rangle)\}$$

We show that $\mathscr{R}$ is a branching bisimulation up-to strong bisimilarity. Consider the pair $(\gamma \mathbin{^\frown_\blacktriangle} \alpha, \langle \langle \gamma \in \mathbb{1}[\alpha], \gamma \rangle, \overline{\alpha} \rangle)$.

1. Suppose $\gamma \mathbin{^\frown_\blacktriangle} \alpha \xrightarrow{a} p'$. By Lemma 5, there exists $X \in \mathscr{P}, \alpha'', \alpha' \in \mathscr{P}^*, \beta \in \mathscr{P}^+$, and $\varepsilon \in \mathscr{D}_\ominus$ such that:

- $\alpha = \alpha'' X \alpha'$,
- $Y \overset{\gamma}{\downarrow}$ and $Y \overset{\gamma_i}{\mapsto}$ for each identifier Y in α'', and
- $X \xrightarrow{\gamma,a} \varepsilon \curlywedge \beta$ and $p' \underline{\leftrightarrow} \varepsilon \curlywedge \beta \alpha'$.

From the second property, it follows:

- by construction that $\langle\langle \gamma \in \mathbb{1}[\alpha], \gamma\rangle, \overline{\alpha}\rangle \twoheadrightarrow \langle\langle \gamma \in \mathbb{1}[X\alpha'], \gamma\rangle, \overline{X\alpha'}\rangle$, and
- from the fact that α'' is terminating that $\gamma \curlywedge \alpha \underline{\leftrightarrow} \gamma \curlywedge X\alpha'$.

Next, since $X \xrightarrow{\gamma,a} \varepsilon \curlywedge \beta$, it follows that:

$$\langle\langle \gamma \in \mathbb{1}[X\alpha'], \gamma\rangle, \overline{X\alpha'}\rangle \xrightarrow{a} \langle\langle \varepsilon \in \mathbb{1}[\beta\alpha'], \varepsilon\rangle, \overline{\beta\alpha'}\rangle \quad .$$

Then, we are done since:

- $\gamma \curlywedge X\alpha'$ and $\langle\langle \gamma \in \mathbb{1}[X\alpha'], \gamma\rangle, \overline{X\alpha'}\rangle$ are related by $\mathscr{R}$; and
- $\varepsilon \curlywedge \beta\alpha'$ and $\langle\langle \varepsilon \in \mathbb{1}[\beta\alpha'], \varepsilon\rangle, \overline{\beta\alpha'}\rangle$ are related by $\mathscr{R}$.

2. Suppose $\gamma \curlywedge \alpha\downarrow$. By definition of $\mathbb{1}[\alpha]$, it holds that $\gamma \in \mathbb{1}[\alpha]$, and by construction of $\downarrow'$ it holds that $\langle\langle \gamma \in \mathbb{1}[\alpha], \gamma\rangle, \overline{\alpha}\rangle\downarrow$.

Next, consider a pair $(\langle\langle \gamma \in \mathbb{1}[\alpha], \gamma\rangle, \overline{\alpha}\rangle, \gamma \curlywedge \alpha)$.

1. Suppose $\langle\langle \gamma \in \mathbb{1}[\alpha], \gamma\rangle, \overline{\alpha}\rangle \xrightarrow{a} p'$. Since α is non-empty, so is $\overline{\alpha}$. Thus, p' is of the form $\langle t, x' \overrightarrow{\overline{\alpha}}\rangle$ and M has a transition $\langle \gamma \in \mathbb{1}[\alpha], \gamma\rangle \xrightarrow{a[\overleftarrow{\overline{\alpha}}/x']} t$. By definition of M, we may distinguish two cases for the values of x' and t.
   - Suppose $x' = \overline{\beta}^{\mathbb{1}[\overrightarrow{\alpha}]}$ and $t = \langle \varepsilon \in \mathbb{1}[\beta] \cap \mathbb{1}[\overrightarrow{\alpha}], \varepsilon\rangle$ for some summand $\delta :\rightarrow a.\varepsilon \curlywedge \beta$ of $\overleftarrow{\alpha}$ with $\delta \preccurlyeq \gamma$. Then it holds that $\gamma \curlywedge \alpha \xrightarrow{a} \varepsilon \curlywedge \beta \overrightarrow{\alpha}$. Moreover, since $x' \overrightarrow{\overline{\alpha}} = \overline{\beta \overrightarrow{\alpha}}$, and $\varepsilon \in \mathbb{1}[\beta] \cap \mathbb{1}[\overrightarrow{\alpha}]$ iff $\varepsilon \in \mathbb{1}[\beta \overrightarrow{\alpha}]$, it holds that $\varepsilon \curlywedge \beta \overrightarrow{\alpha}$ and p' are related by $\mathscr{R}$.
   - Suppose $x' = \epsilon$ and $t = \langle \gamma \in \mathbb{1}[\alpha], \gamma\rangle$. Then $a = \tau$, $\gamma \in \mathbb{1}[\overleftarrow{\alpha}]$, and moreover there is no summand $\delta :\rightarrow b.\varepsilon \curlywedge \beta$ of $\overleftarrow{\alpha}$ with $\delta \preccurlyeq \gamma$. Therefore, $\gamma \curlywedge \overleftarrow{\alpha}$ is terminating, and consequently $\gamma \curlywedge \alpha \underline{\leftrightarrow} \gamma \curlywedge \overrightarrow{\alpha}$ holds by disregarding any irrelevant conditionals of $\overleftarrow{\alpha}$, and applying Axiom A9. Since $\gamma \in \mathbb{1}[\overleftarrow{\alpha}]$, it holds that $\gamma \in \mathbb{1}[\alpha]$ iff $\gamma \in \mathbb{1}[\overrightarrow{\alpha}]$. Therefore, p' and $\gamma \curlywedge \overrightarrow{\alpha}$ are related by $\mathscr{R}$, and we are done.
2. Suppose $\langle\langle \gamma \in \mathbb{1}[\alpha], \gamma\rangle, \overline{\alpha}\rangle\downarrow$. Then $\gamma \in \mathbb{1}[\alpha]$ holds, and $\gamma \curlywedge \alpha\downarrow$ follows from the operational semantics.

Lastly, we have the pairs $(X_\uparrow, \langle\uparrow', \epsilon\rangle)$ and $(\langle\uparrow', \epsilon\rangle, X_\uparrow)$. It is relatively straightforward to show that these pairs satisfy the required conditions, since $\langle\uparrow', \epsilon\rangle \xrightarrow{\tau} \langle\langle \ominus \in \mathbb{1}[X_\uparrow], \ominus\rangle, \overline{X}_\uparrow\rangle$, and the target state is related to $\ominus \curlywedge X_\uparrow$ in $\mathscr{R}$.

Thus, $\mathscr{R}$ is a branching bisimulation up-to strong bisimilarity, and since $(X_\uparrow, \langle\uparrow', \epsilon\rangle) \in \mathscr{R}$, it holds that $X_\uparrow \underline{\leftrightarrow}_b \langle\uparrow', \epsilon\rangle$. □