Information and Computation 209 (2011) 1355–1389

Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

# Modelling concurrency with comtraces and generalized comtraces

Ryszard Janicki [a,*], Dai Tri Man Lê [b]

[a] Department of Computing and Software, McMaster University, Hamilton, ON, L8S 4K1 Canada
[b] Department of Computer Science, University of Toronto, Toronto, ON, M5S 3G4 Canada

ARTICLE INFO

*Article history:*
Received 24 August 2009
Revised 4 May 2011
Available online 17 August 2011

*Keywords:*
Generalized trace theory
Trace monoid
Step sequence
Stratified partial order
Stratified order structure
Canonical representation

ABSTRACT

*Comtraces* (*com*bined *traces*) are extensions of Mazurkiewicz traces that can model the "not later than" relationship. In this paper, we first introduce the novel notion of *generalized comtraces*, extensions of comtraces that can additionally model the "non-simultaneously" relationship. Then we study some basic algebraic properties and canonical representations of comtraces and generalized comtraces. Finally we analyze the relationship between generalized comtraces and generalized stratified order structures. The major technical contribution of this paper is a proof showing that generalized comtraces can be represented by generalized stratified order structures.



## 1. Introduction

Mazurkiewicz traces, or just traces,[1] are quotient monoids over sequences (or words) [2,23,4]. The theory of traces has been utilized to tackle problems from diverse areas including combinatorics, graph theory, algebra, logic and especially concurrency theory [4].

As a language representation of finite partial orders, traces can sufficiently model "true concurrency" in various aspects of concurrency theory. However, some aspects of concurrency cannot be adequately modelled by partial orders (cf. [8,10]), and thus cannot be modelled by traces. For example, neither traces nor partial orders can model the "not later than" relationship [10]. If an event $a$ is performed "not later than" an event $b$, then this "not later than" relationship can be modelled by the following set of two step sequences $\mathbf{x} = \{\{a\}\{b\}, \{a, b\}\}$; where *step* $\{a, b\}$ denotes the simultaneous execution of $a$ and $b$ and the step sequence $\{a\}\{b\}$ denotes the execution of $a$ followed by $b$. But the set $\mathbf{x}$ cannot be represented by any trace (or equivalently any partial order), even if the generators, i.e. elements of the trace alphabet, are sets and the underlying monoid is the monoid of step sequences (as in [29]).

To overcome these limitations, Janicki and Koutny proposed the *comtrace* (*com*bined *trace*) notion [11]. First the set of all possible steps that generates step sequences is identified by a relation *sim*, which is called *simultaneity*. Second a congruence relation is determined by a relation *ser*, which is called *serializability* and is in general *not* symmetric. Then a comtrace is defined as a finite set of congruent step sequences. Comtraces were invented to provide a formal linguistic counterpart of *stratified order structures* (*so-structures*), analogously to how traces relate to partial orders.

A so-structure [5,9,11,12] is a triple $(X, \prec, \sqsubset)$, where $\prec$ and $\sqsubset$ are binary relations on the set $X$. So-structures were invented to model both the "earlier than" (the relation $\prec$) and the "not later than" (the relation $\sqsubset$) relationships, under the

* Corresponding author.
*E-mail addresses:* janicki@mcmaster.ca (R. Janicki), ledt@cs.toronto.edu (D.T.M. Lê).

[1] The word "trace" has many different meanings in computer science and software engineering. In this paper, we reserve the word "trace" for *Mazurkiewicz trace*, which is different from "traces" used in Hoare's CSP [7].

0890-5401/$ – see front matter 
doi:10.1016/j.ic.2011.08.001

assumption that all system runs are modelled by stratified partial orders, i.e., step sequences. They have been successfully applied to model inhibitor and priority systems, asynchronous races, synthesis problems, etc. (see for example [11,26,17–20]).

The paper [11] contains a major result showing that every comtrace uniquely determines a labeled so-structure, and then use comtraces to provide a semantics of Petri nets with inhibitor arcs. However, so far comtraces are used less often than so-structures, even though in many cases they appear to be more natural than so-structures. Perhaps this is due to the lack of a sufficiently developed quotient monoid theory for comtraces similar to that of traces.

However, neither comtraces nor so-structures are enough to model the "non-simultaneously" relationship, which could be defined by the set of step sequences $\{\{a\}\{b\},\{b\}\{a\}\}$ with the additional assumption that the step $\{a,b\}$ is not allowed. In fact, both comtraces and so-structures can adequately model concurrent histories only when paradigm $\pi_3$ of [10,12] is satisfied. Intuitively, paradigm $\pi_3$ formalizes the class of concurrent histories satisfying the condition that if both $\{a\}\{b\}$ and $\{b\}\{a\}$ belong to the concurrent history, then so does $\{a,b\}$ (i.e., these three step sequences $\{a\}\{b\}$, $\{b\}\{a\}$ and $\{a,b\}$ are all equivalent observations).

To model the general case that includes the "non-simultaneously" relationship, we need the concept of *generalized stratified order structures* (*gso-structures*), which were introduced and analyzed by Guo and Janicki in [6,8]. A gso-structure is a triple $(X,\Diamond,\sqsubset)$, where $\Diamond$ and $\sqsubset$ are binary relations on $X$ modelling the "non-simultaneously" and "not later than" relationships respectively, under the assumption that all system runs are modelled by stratified partial orders.

To provide the reader with a high level view of the main motivation and intuition behind the use of so-structures as well as the need of gso-structures, we will consider a motivating example (adapted from [8]).

### 1.1. A motivating example

We will illustrate our basic concepts and constructions by analyzing four simple concurrent programs. Three of these programs will involve the concepts of simultaneous executions, which is essential to our model. We would like to point out that the theory presented in this paper is especially for models where simultaneity is well justified, for example for the models with a discrete time.

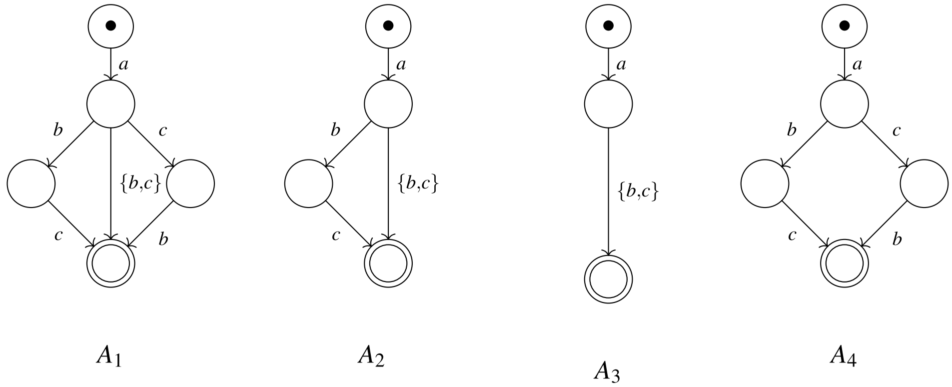
All four programs in this example are written using a mixture of *cobegin, coend* and a version of *concurrent guarded commands.*

**Example 1.**

```
P1: begin int x,y;
       a: begin x:=0; y:=0 end;
       cobegin b: x:=x+1, c: y:=y+1 coend
      end P1.

P2: begin int x,y;
       a: begin x:=0; y:=0 end;
       cobegin b: x=0 → y:=y+1, c: x:=x+1 coend
      end P2.

P3: begin int x,y;
       a: begin x:=0; y:=0 end;
       cobegin b: y=0 → x:=x+1, c: x=0 → y:=y+1 coend
      end P3.

P4: begin int x;
       a: x:=0;
       cobegin b: x:=x+1, c: x:=x+2 coend
      end P4.
```

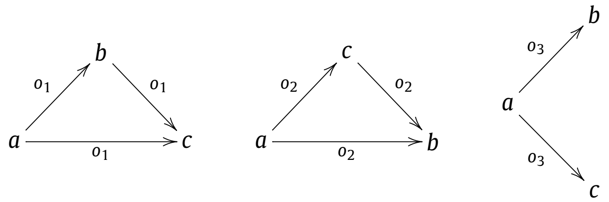
Each program is a different composition of three events (actions) called $a$, $b$, and $c$ ($a_i$, $b_i$, $c_i$, $i=1,\ldots,4$, to be exact, but a restriction to $a$, $b$, $c$ does not change the validity of the analysis below, while simplifying the notation). Transition systems modelling these programs are shown in Fig. 1.

Let $obs(P_i)$ denote the set of all program runs involving the actions $a$, $b$, $c$ that can be observed. Assume that simultaneous executions can be observed. In this simple case all runs (or observations) can be modelled by *step sequences*. Let us denote $o_1=\{a\}\{b\}\{c\}$, $o_2=\{a\}\{c\}\{b\}$, $o_3=\{a\}\{b,c\}$. Each $o_i$ can be equivalently seen as a stratified partial order $o_i=(\{a,b,c\},\xrightarrow{o_i})$ where:




| | | | |
|---|---|---|---|
| $\prec_1 = \{(a,b),(a,c)\}$ | $\prec_2 = \{(a,b),(a,c)\}$ | $\prec_3 = \{(a,b),(a,c)\}$ | $\prec_4 = \{(a,b),(a,c)\}$ |
| $\sqsubset_1 = \{(a,b),(a,c)\}$ | $\sqsubset_2 = \{(a,b),(a,c),(b,c)\}$ | $\sqsubset_3 = \{(a,b),(a,c),(b,c),(c,b)\}$ | $\sqsubset_4 = \{(a,b),(a,c)\}$ |
| $\Diamond_1 = \sqsubset_1 \cup \sqsubset_1^{-1}$ | $\Diamond_2 = \prec_2 \cup \prec_2^{-1}$ | $\Diamond_3 = \prec_3 \cup \prec_3^{-1}$ | $\Diamond_4 = \{(a,b),(b,a),(a,c),(c,a),(b,c),(c,b)\}$ |
| $obs(P_1) \asymp obs(A_1) \asymp \{\prec_1\} \asymp \{\prec_1, \sqsubset_1\} \asymp \{\Diamond_1, \sqsubset_1\}$ | $obs(P_2) \asymp obs(A_2) \asymp \{\prec_2, \sqsubset_2\} \asymp \{\Diamond_2, \sqsubset_2\}$ | $obs(P_3) \asymp obs(A_3) \asymp \{\prec_3, \sqsubset_3\} \asymp \{\Diamond_3, \sqsubset_3\}$ | $obs(P_4) \asymp obs(A_4) \asymp \{\Diamond_4, \sqsubset_4\}$ |

**Fig. 1.** Examples of *causality*, *weak causality*, and *commutativity*. Each program $P_i$ can be modelled by a labeled transition system (automaton) $A_i$. The step $\{a, b\}$ denotes the *simultaneous* execution of $a$ and $b$.




We can now write $obs(P_1) = \{o_1, o_2, o_3\}$, $obs(P_2) = \{o_1, o_3\}$, $obs(P_3) = \{o_3\}$, $obs(P_4) = \{o_1, o_2\}$. Note that for every $i = 1, \ldots, 4$, all runs from the set $obs(P_i)$ yield exactly the same outcome. Hence, each $obs(P_i)$ is called the *concurrent history* of $P_i$.

An abstract model of such an outcome is called a *concurrent behavior*, and now we will discuss how causality, weak causality and commutativity relations are used to construct concurrent behavior.

*1.1.1. Program $P_1$*

In the set $obs(P_1)$, for each run, $a$ always precedes both $b$ and $c$, and there is no *causal* relationship between $b$ and $c$. This *causality* relation, $\prec$, is the partial order defined as $\prec = \{(a, b), (a, c)\}$. In general $\prec$ is defined by: $x \prec y$ iff for each run $o$ we have $x \xrightarrow{o} y$. Hence for $P_1$, $\prec$ is the intersection of $o_1$, $o_2$ and $o_3$, and $\{o_1, o_2, o_3\}$ is the set of all stratified extensions of the relation $\prec$.

Thus, in this case, the causality relation $\prec$ models the concurrent behavior corresponding to the set of (equivalent) runs $obs(P_1)$. We will say that $obs(P_1)$ and $\prec$ are *tantamount*[2] and write $obs(P_1) \asymp \{\prec\}$ or $obs(P_1) \asymp (\{a, b, c\}, \prec)$. Having $obs(P_1)$ one may construct $\prec$ (as an intersection of all orders from $obs(P_1)$), and then reconstruct $obs(P_1)$ (as the set of all stratified extensions of $\prec$). This is a classical case of the "true" concurrency approach, where concurrent behavior is modelled by a causality relation.

Before considering the remaining cases, note that the causality relation $\prec$ is exactly the same in all four cases, i.e., $\prec_i = \{(a, b), (a, c)\}$, for $i = 1, \ldots, 4$, so we may omit the index $i$.

*1.1.2. Programs $P_2$ and $P_3$*

To deal with $obs(P_2)$ and $obs(P_3)$, $\prec$ is insufficient because $o_2 \notin obs(P_2)$ and $o_1, o_2 \notin obs(P_2)$. Thus, we need a weak causality relation $\sqsubset$ defined in this context as $x \sqsubset y$ iff for each run $o$ we have $\neg(y \xrightarrow{o} x)$ (*$x$ is never executed after $y$*). For our four cases we have $\sqsubset_2 = \{(a, b), (a, c), (b, c)\}$, $\sqsubset_1 = \sqsubset_4 = \prec$, and $\sqsubset_3 = \{(a, b), (a, c), (b, c), (c, b)\}$. Notice again that for $i = 2, 3$, the pair of relations $\{\prec, \sqsubset_i\}$ and the set $obs(P_i)$ are *tantamount* as each is definable from the other. (The set

[2] Following [8], we are using the word "tantamount" instead of "equivalent" as the latter usually implies that the entities are of the same type, as "equivalent automata", "equivalent expressions", etc. Tantamount entities can be of different types.

$obs(P_i)$ can be defined as the greatest set $PO$ of partial orders built from $a$, $b$ and $c$ satisfying $x \prec y \Rightarrow \forall o \in PO.x \xrightarrow{o} y$ and $x \sqsubset_i y \Rightarrow \forall o \in PO.\neg(y \xrightarrow{o} x)$.)

Hence again in these cases ($i = 2, 3$) $obs(P_i)$ and $\{\prec, \sqsubset_i\}$ are *tantamount*, $obs(P_i) \asymp \{\prec, \sqsubset_i\}$, and so the pair $\{\prec, \sqsubset_i\}$, $i = 2, 3$, models the concurrent behavior described by $obs(P_i)$. Note that $\sqsubset_i$ alone is not sufficient, since (for instance) $obs(P_2)$ and $obs(P_2) \cup \{\{a, b, c\}\}$ define the same relation $\sqsubset$.

#### *1.1.3. Program $P_4$*

The causality relation $\prec$ does not model the concurrent behavior of $P_4$ correctly[3] since $o_3$ does not belong to $obs(P_4)$. The commutativity relation $\Leftrightarrow$ is defined in this context as $x \Leftrightarrow y$ iff for each run $o$ either $x \xrightarrow{o} y$ or $y \xrightarrow{o} x$. For the set $obs(P_4)$, the relation $\Leftrightarrow_4$ looks like $\Leftrightarrow_4 = \{(a, b), (b, a), (a, c), (c, a), (b, c), (c, b)\}$. The pair of relations $\{\Leftrightarrow_4, \prec\}$ and the set $obs(P_4)$ are *tantamount* as each is definable from the other. (The set $obs(P_4)$ is the greatest set $PO$ of partial orders built from $a$, $b$ and $c$ satisfying $x \Leftrightarrow_4 y \Rightarrow \forall o \in PO.x \xrightarrow{o} y \vee y \xrightarrow{o} x$ and $x \prec y \Rightarrow \forall o \in PO.x \xrightarrow{o} y$.) In other words, $obs(P_4)$ and $\{\Leftrightarrow_4, \prec\}$ are tantamount, so we may say that in this case the relations $\{\Leftrightarrow_4, \prec\}$ model the concurrent behavior described by $obs(P_4)$.

Note that $\Leftrightarrow_1 = \prec \cup \prec^{-1}$ and the pair $\{\Leftrightarrow_1, \prec\}$ also model the concurrent behavior described by $obs(P_1)$.

#### *1.1.4. Summary of analysis of $P_1$, $P_2$, $P_3$ and $P_4$*

For each $P_i$ the state transition model $A_i$ and their respective concurrent histories and concurrent behaviors are summarized in Fig. 1. Thus, we can make the following observations:

1. $obs(P_1)$ can be modelled by the relation $\prec$ alone, and $obs(P_1) \asymp \{\prec\}$.
2. $obs(P_i)$, for $i = 1, 2, 3$ can also be modelled by the appropriate pairs of relations $\{\prec, \sqsubset_i\}$, and $obs(P_i) \asymp \{\prec, \sqsubset_i\}$.
3. All sets of observations $obs(P_i)$, for $i = 1, 2, 3, 4$ are modelled by the appropriate pairs of relations $\{\Leftrightarrow_i, \sqsubset_i\}$, and $obs(P_i) \asymp \{\Leftrightarrow_i, \sqsubset_i\}$.

Note that the relation $\prec$ is not independent from the relations $\Leftrightarrow$, $\sqsubset$, since it can be proven (see [10]) that $\prec = \Leftrightarrow \cap \sqsubset$. Intuitively, since $\Leftrightarrow$ and $\sqsubset$ are the abstraction of the "earlier than or later than" and "not later than" relations, it follows that their intersection is the abstraction of the "earlier than" relation.

#### *1.1.5. Intuition for comtraces and generalized comtraces*

We may also try to model the concurrent behaviors of the programs $P_1$, $P_2$, $P_3$ and $P_4$ only in *terms of algebra of step sequences*. To do this we need to introduce an equivalence relation on step sequences such that the sets $obs(P_i)$, for $i = 1, \ldots, 4$, interpreted as *sets of step sequences* and not partial orders, are appropriate equivalence classes. A particular instance of this equivalence relation should depend on the structure of a particular program, or its labeled transition system representation.

It turns out that in such an approach the program $P_4$ needs to be treated differently than $P_1$, $P_2$ and $P_3$. In order to avoid ambiguity, we will write $obs_{\text{step}}(P_i)$ to denote the same set of system runs as $obs(P_i)$, but with runs now modelled by step sequences instead of partial orders.

For all four cases we need two relations $sim_i$ and $ser_i$, $i = 1, \ldots, 4$, on the set $\{a, b, c\}$. The relations $sim_i$, called *simultaneity*, are symmetric and indicate which actions can be executed simultaneously, i.e. in one step. It is easy to see that $sim_1 = sim_2 = sim_3 = \{(b, c), (c, b)\}$, but $sim_4 = \emptyset$. The relations $ser_i$, called *serializability*, may not be symmetric, must satisfy $ser_i \subseteq sim_i$, and indicate how steps can equivalently be executed in some sequence. In principle if $(\alpha, \beta) \in ser$ then the step $\{\alpha, \beta\}$ is equivalent to the sequence $\{\alpha\}\{\beta\}$. For our four cases we have $ser_1 = sim_1 = \{(b, c), (c, b)\}$, $ser_2 = \{(b, c)\}$, $ser_3 = ser_4 = \emptyset$.

Let $A, B, C$ be steps such that $A = B \cup C$ and $B \cap C = \emptyset$. For example $A = \{b, c\}$, $B = \{b\}$ and $C = \{c\}$. We will say that the step $A$ and the step sequence $BC$ are *equivalent*, $A \approx_i BC$, if $B \times C \subseteq sim_i$. For example we have $\{b, c\} \approx_i \{b\}\{c\}$ for $i = 1, 2$ and $\{b, c\} \approx_i \{c\}\{b\}$ for $i = 1$. The relations $\approx_3$ and $\approx_4$ are empty.

Let $\equiv_i$ be the smallest equivalence relation on the whole set of events containing $\approx_i$, and for each step sequence $A_1 \ldots A_k$, let $[A_1 \ldots A_k]_{\equiv_i}$ denote the equivalence class of $\equiv_i$ containing the step sequence $A_1 \ldots A_k$.

For our four cases, we have:

1. $[\{a\}\{b\}\{c\}]_{\equiv_1} = \{\{a\}\{b\}\{c\}, \{a\}\{c\}\{b\}, \{a\}\{b, c\}\} = obs_{\text{step}}(P_1) \asymp obs(P_1)$.
2. $[\{a\}\{b\}\{c\}]_{\equiv_2} = \{\{a\}\{b\}\{c\}, \{a\}\{b, c\}\} = obs_{\text{step}}(P_2) \asymp obs(P_2)$.
3. $[\{a\}\{b\}\{c\}]_{\equiv_3} = \{\{a\}\{b, c\}\} = obs_{\text{step}}(P_3) \asymp obs(P_3)$.
4. $[\{a\}\{b\}\{c\}]_{\equiv_4} = \{\{a\}\{b\}\{c\}\} \neq obs_{\text{step}}(P_4)$.

Strictly speaking the statement $obs_{\text{step}}(P_i) = obs(P_i)$ is false, but obviously $obs_{\text{step}}(P_i) \asymp obs(P_i)$, for $i = 1, \ldots, 4$.

[3] Unless we assume that simultaneity is not allowed, or not observed, in which case $obs(P_1) = obs(P_4) = \{o_1, o_2\}$, $obs(P_2) = \{o_1\}$, $obs(P_3) = \emptyset$.

For $i = 1, \ldots, 3$, equivalence classes of each relation $\equiv_i$ are generated by relations $sim_i$ and $ser_i$. These equivalence classes are called *comtraces* (introduced in [11] as a generalization of Mazurkiewicz traces) and can be used to model concurrent histories of the systems or programs like $P_1$, $P_2$ and $P_3$.

In order to model the concurrent history of $P_4$ with equivalent step sequences, we need a third relation $inl_4$ on the set of events $\{a, b, c\}$ that is symmetric and satisfies $inl_4 \cap sim_4 = \emptyset$. The relation $inl_4$ is called *interleaving*, and if $(x, y) \in inl$ then events $x$ and $y$ cannot be executed simultaneously, but the execution of $x$ followed by $y$ and the execution of $y$ followed by $x$ are equivalent. For program $P_4$ we have $inl_4 = \{(b, c), (c, b)\}$.

We can now define a relation $\approx'_4$ on step sequences of length two, as $BC \approx'_4 CB$ if $B \times C \subseteq inl$, which for this simple case gives $\approx'_4 = \{(\{b\}\{c\}, \{c\}\{b\}), (\{c\}\{b\}, \{b\}\{c\})\}$. Let $\equiv_4$ be the smallest equivalence relation on the whole set of events containing $\approx_4$ and $\approx'_4$. Then we have

$$\big[\{a\}\{b\}\{c\}\big]_{\equiv_4} = \big\{\{a\}\{b\}\{c\}, \{a\}\{c\}\{b\}\big\} = obs_{\text{step}}(P_4) \asymp obs(P_4).$$

Equivalence classes of relations like $\equiv_4$, generated by the relations like $sim_4$, $ser_4$ *and* $inl_4$ are called *generalized comtraces* (*g-comtraces*, introduced in [15]) and they can be used to model concurrent histories of the systems or programs like $P_4$.

### 1.2. Summary of contributions

This paper is an expansion and revision of our results from [15,21]. We propose a formal-language counterpart of gso-structures, called *generalized comtraces* (*g-comtraces*). We will revisit and expand the algebraic theory of comtraces, especially various types of canonical forms and the formal relationship between traces and comtraces. We analyze in detail the properties of g-comtraces, their canonical representations, and most importantly the formal relationship between g-comtraces and gso-structures.

### 1.3. Organization

The content of the paper is organized as follows. In the next section, we review some basic concepts of order theory and monoid theory. Section 3 recalls the concept of Mazurkiewicz traces and discusses its relationship to finite partial orders. Section 4 surveys some basic background on the relational structures model of concurrency [5,9,11,12,6,8].

Comtraces are defined and their relationship to traces is discussed in Section 5, and g-comtraces are introduced in Section 6.

Various basic algebraic properties of both comtrace and g-comtrace congruences are discussed in Section 7. Section 8 is devoted to canonical representations of traces, comtraces and g-comtraces. In Section 9 we recall some results on the so-structures defined by comtraces. The gso-structures generated by g-comtraces are defined and analyzed in Section 10. Concluding remarks are made in Section 11. We also include two appendices containing some long and technical proofs of results from Section 10.

## 2. Orders, monoids, sequences and step sequences

In this section, we recall some standard notations, definitions and results which are used extensively in this paper.

### 2.1. Relations, orders and equivalences

The *powerset* of a set $X$ will be denoted by $\wp(X)$. The set of all *non-empty* subsets of $X$ will be denoted by $\wp^{\setminus\{\emptyset\}}(X)$. In other words, $\wp^{\setminus\{\emptyset\}}(X) \triangleq \wp(X) \setminus \{\emptyset\}$.

Let $f : A \to B$ be a function, then for every set $C \subseteq A$, we write $f[C]$ to denote the image of the set $C$ under $f$, i.e., $f[C] \triangleq \{f(x) \mid x \in C\}$.

We let $id_X$ denote the *identity relation* on a set $X$. We write $R \circ S$ to denote the *composition* of relations $R$ and $S$. We also write $R^+$ and $R^*$ to denote the (*irreflexive*) *transitive closure* and *reflexive transitive closure* of $R$ respectively.

A binary relation $R \subseteq X \times X$ is an *equivalence relation* on $X$ iff it is reflexive, symmetric and transitive. If $R$ is an equivalence relation, we write $[x]_R$ to denote the equivalence class of $x$ with respect to $R$, and the set of all equivalence classes in $X$ is denoted as $X/R$ and called the *quotient set* of $X$ by $R$. We drop the subscript and write $[x]$ to denote the equivalence class of $x$ when $R$ is clear from the context.

A binary relation $\prec \subseteq X \times X$ is a *partial order* iff $R$ is *irreflexive* and *transitive*. The pair $(X, \prec)$ in this case is called a *partially ordered set* (*poset*). The pair $(X, \prec)$ is called a *finite poset* if $X$ is finite. For convenience, we define:

$$\simeq_\prec \triangleq \big\{(a, b) \in X \times X \;\big|\; a \not\prec b \wedge b \not\prec a\big\} \qquad (\textit{incomparable}),$$

$$\frown_\prec \triangleq \big\{(a, b) \in X \times X \;\big|\; a \simeq_\prec b \wedge a \neq b\big\} \qquad (\textit{distinctly incomparable}),$$

$$\prec^\frown \triangleq \big\{(a, b) \in X \times X \;\big|\; a \prec b \vee a \frown_\prec b\big\} \qquad (\textit{not greater}).$$

A poset $(X, \prec)$ is *total* iff $\frown_\prec$ is empty; and *stratified* iff $\simeq_\prec$ is an equivalence relation. Evidently every total order is stratified.

Let $\prec_1$ and $\prec_2$ be partial orders on a set $X$. Then $\prec_2$ is an *extension* of $\prec_1$ if $\prec_1 \subseteq \prec_2$. The relation $\prec_2$ is a *total extension* (*stratified extension*) of $\prec_1$ if $\prec_2$ is total (stratified) and $\prec_1 \subseteq \prec_2$.

For a poset $(X, \prec)$, we define

$$Total_X(\prec) \triangleq \{\lhd \subseteq X \times X \mid \lhd \text{ is a total extension of } \prec\}.$$

**Theorem 1.** *(See Szpilrajn [28].) For every poset $(X, \prec)$, $\prec = \bigcap_{\lhd \in Total_X(\prec)} \lhd$.* $\square$

Szpilrajn's theorem states that every partial order can be uniquely reconstructed by taking the intersection of all of its total extensions.

### 2.2. Monoids and equational monoids

A triple $(X, *, \mathbb{1})$, where $X$ is a set, $*$ is a total binary operation on $X$, and $\mathbb{1} \in X$, is called a *monoid*, if $(a*b)*c = a*(b*c)$ and $a * \mathbb{1} = \mathbb{1} * a = a$, for all $a, b, c \in X$.

An equivalence relation $\sim \subseteq X \times X$ is a *congruence* in the monoid $(X, *, \mathbb{1})$ if for all elements $a_1, a_2, b_1, b_2$ of $X$, $a_1 \sim b_1 \wedge a_2 \sim b_2 \Rightarrow (a_1 * a_2) \sim (b_1 * b_2)$.

The triple $(X/\sim, \circledast, [\mathbb{1}])$, where $[a] \circledast [b] = [a * b]$, is called the *quotient monoid* of $(X, *, 1)$ under the congruence $\sim$. The mapping $\phi : X \to X/\sim$ defined as $\phi(a) = [a]$ is called the *natural homomorphism* generated by the congruence $\sim$. We usually omit the symbols $*$ and $\circledast$.

**Definition 1** *(Equation monoid).* Given a monoid $M = (X, *, \mathbb{1})$ and a finite set of *equations* $EQ = \{x_i = y_i \mid i = 1, \ldots, n\}$, define $\equiv_{EQ}$ to be the least congruence on $M$ satisfying

$$x_i = y_i \quad \Longrightarrow \quad x_i \equiv_{EQ} y_i$$

for every equation $x_i = y_i \in EQ$. We call the relation $\equiv_{EQ}$ the congruence defined by the set of equation $EQ$, or $EQ$-congruence. The quotient monoid $M_{\equiv_{EQ}} = (X/\equiv_{EQ}, \circledast, [\mathbb{1}])$, where $[x] \circledast [y] = [x * y]$, is called an *equational monoid*.

The following folklore result shows that the relation $\equiv_{EQ}$ can also be *uniquely* defined in an explicit way.

**Proposition 1.** *(Cf. [21].) Given a monoid $M = (X, *, \mathbb{1})$ and a set of equations EQ, define the relation $\approx \subseteq X \times X$ as:*

$$x \approx y \quad \stackrel{df}{\Longleftrightarrow} \quad \exists x_1, x_2 \in X. \exists (u = w) \in EQ. x = x_1 * u * x_2 \wedge y = x_1 * w * x_2,$$

*then the EQ-congruence $\equiv$ is $(\approx \cup \approx^{-1})^*$, the symmetric irreflexive transitive closure of $\approx$.* $\square$

We will see later in this paper that monoids of traces, comtraces and generalized comtraces are all special cases of equational monoids.

### 2.3. Sequences, step sequences and partial orders

By an *alphabet* we shall understand any finite set. For an alphabet $\Sigma$, let $\Sigma^*$ denote the set of all *finite* sequences of elements (words) of $\Sigma$, $\lambda$ denotes the empty sequence, and any subset of $\Sigma^*$ is called a *language*. In the scope of this paper, we only deal with *finite* sequences. Let the operator $\_ \cdot \_$ denote the sequence concatenation (usually omitted). Since the sequence concatenation operator is associative and $\lambda$ is neutral, the triple $(\Sigma^*, \cdot, \lambda)$ is a *monoid* (of sequences).

Consider an alphabet $\mathbb{S} \subseteq \wp^{\setminus\{\emptyset\}}(X)$ for some alphabet $\Sigma$. The elements of $\mathbb{S}$ are called *steps* and the elements of $\mathbb{S}^*$ are called *step sequences*. For example if $\mathbb{S} = \{\{a, b, c\}, \{a, b\}, \{a\}, \{c\}\}$ then $\{a, b\}\{c\}\{a, b, c\} \in \mathbb{S}^*$ is a step sequence. The triple $(\mathbb{S}^*, \cdot, \lambda)$, is a *monoid* (of step sequences), since the step sequence concatenation is associative and $\lambda$ is neutral.

We will now show the formal relationship between step sequences and stratified orders. Let $t = A_1 \ldots A_k$ be a step sequence in $\mathbb{S}^*$. We define $|t|_a$, the number of occurrences of an event $a$ in $t$, as $|t|_a \triangleq |\{A_i \mid 1 \leqslant i \leqslant k \wedge a \in A_i\}|$, where $|X|$ denotes the cardinality of the set $X$.

- We can uniquely construct its *enumerated step sequence* $\bar{t}$ as

  $$\bar{t} \triangleq \overline{A_1} \ldots \overline{A_k}, \quad \text{where } \overline{A_i} \triangleq \left\{e^{(|A_1 \ldots A_{i-1}|_e + 1)} \;\middle|\; e \in A_i\right\}.$$

  We call such $\alpha = e^{(i)} \in \overline{A_i}$ an *event occurrence* of $e$. E.g., if $t = \{a, b\}\{b, c\}\{c, a\}\{a\}$, then $\bar{t} = \{a^{(1)}, b^{(1)}\}\{b^{(2)}, c^{(1)}\}\{a^{(2)}, c^{(2)}\}\{a^{(3)}\}$ is its enumerated step sequence.
- Let $\Sigma_t = \bigcup_{i=1}^{k} \overline{A_i}$ denote the set of all event occurrences in all steps of $t$. For example, when $t = \{a, b\}\{b, c\}\{c, a\}\{a\}$, we have $\Sigma_t = \{a^{(1)}, a^{(2)}, a^{(3)}, b^{(1)}, b^{(2)}, c^{(1)}, c^{(2)}\}$.

- Define $l : \Sigma_t \to \Sigma$ to be the function that returns the label of an event occurrence. In other words, for each event occurrence $\alpha = e^{(i)}$, $l(\alpha)$ returns the *label* $e$ of $\alpha$. From an enumerated step sequence $\bar{t} = \overline{A_1} \dots \overline{A_k}$, we can uniquely recover its step sequence as $t = l[\overline{A_1}] \dots l[\overline{A_k}]$.
- For each $\alpha \in \Sigma_t$, let $pos_t(\alpha)$ denote the index number of the step where $\alpha$ occurs, i.e., if $\alpha \in \overline{A_j}$ then $pos_t(\alpha) = j$. For our example, $pos_t(a^{(2)}) = 3$, $pos_t(b^{(2)}) = 2$, etc.

Given a step sequence $u$, we define two relations $\lhd_u, \simeq_u \subseteq \Sigma_u \times \Sigma_u$ as:

$$\alpha \lhd_u \beta \quad \overset{df}{\Longleftrightarrow} \quad pos_u(\alpha) < pos_u(\beta) \quad \text{and} \quad \alpha \simeq_u \beta \quad \overset{df}{\Longleftrightarrow} \quad pos_u(\alpha) = pos_u(\beta).$$

Since $\lhd_u^{\frown}$ is the union of $\lhd_u$ and $\frown_u$, we have

$$\alpha \lhd_u^{\frown} \beta \quad \Longleftrightarrow \quad \big(\alpha \neq \beta \wedge pos_u(\alpha) \leqslant pos_u(\beta)\big).$$

The two propositions below are folklore results (see [21] for detailed proofs), which are fundamental for understanding why stratified partial orders and step sequences are two interchangeable concepts. The first proposition shows that $\lhd_u$ is indeed a stratified order.

**Proposition 2.** *Given a step sequence u, the relation $\simeq_u$ is an equivalence relation and $\lhd_u$ is a stratified order.* □

We will call $\lhd_u$ the stratified order *generated by the step sequence u*. Conversely, let $\lhd$ be a stratified order on a set $\Sigma$. Then the second proposition says:

**Proposition 3.** *If $\lhd$ is a stratified order on a set $\Sigma$ and $A$, $B$ are two distinct equivalence classes of $\simeq_\lhd$, then either $A \times B \subseteq \lhd$ or $B \times A \subseteq \lhd$.* □

In other words, Proposition 3 implies that if we define a binary relation $\widehat{\lhd}$ on the quotient set $\Sigma/\simeq_\lhd$ as

$$A \,\widehat{\lhd}\, B \quad \overset{df}{\Longleftrightarrow} \quad A \times B \subseteq \lhd,$$

then $\widehat{\lhd}$ totally orders $\Sigma/\simeq_\lhd$ into a sequence of equivalence classes $\Omega_\lhd = B_1 \dots B_k$ ($k \geqslant 0$). We will call the sequence $\Omega_\lhd$ as the step sequence *representing* $\lhd$.

Since sequences are a special case of step sequences and total orders are a special case of stratified orders, the above results can be applied to sequences and finite total orders as well. Hence, for each sequence $x \in \Sigma^*$, we let $\lhd_x$ denote the *total order generated* by $x$, and for every total order $\lhd$, we let $\Omega_\lhd$ denote the *sequence generating* $\lhd$. Furthermore, $\Sigma_x$ will denote the alphabet of the sequence $x$.

## 3. Traces vs. partial orders

Traces or partially commutative monoids [2,4,23,24] are *equational monoids over sequences*. In the previous section we have shown how sequences correspond to finite total orders and how step sequences correspond to finite stratified orders. In this section we discuss the relationship between traces and finite partial orders.

The theory of traces has been utilized to tackle problems from diverse areas including combinatorics, graph theory, algebra, logic and, especially (due to the relationship to partial orders) concurrency theory [4,23,24].

Since traces constitute a *sequence representation of partial orders*, they can effectively model "true concurrency" in various aspects of concurrency theory using simple and intuitive means. We will now recall the definition of a *trace monoid*.

**Definition 2.** (See [4,24].) Let $M = (E^*, *, \lambda)$ be a *monoid* generated by finite $E$, and let the relation $ind \subseteq E \times E$ be an irreflexive and symmetric relation (called *independency* or *commutation*), and $EQ \triangleq \{ab = ba \mid (a, b) \in ind\}$. Let $\equiv_{ind}$, called *trace congruence*, be the congruence defined by $EQ$. Then the equational monoid $M_{\equiv_{ind}} = (E^*/\equiv_{ind}, \circledast, [\lambda])$ is a monoid of *traces* (or a *free partially commutative monoid*). The pair $(E, ind)$ is called a *trace alphabet*.

We will omit the subscript *ind* from trace congruence and write $\equiv$ if it causes no ambiguity.

**Example 2.** Let $E = \{a, b, c\}$, $ind = \{(b, c), (c, b)\}$, i.e., $EQ = \{bc = cb\}$.[4] For example, $abcbca \equiv accbba$ (since $abcbca \approx acbbca \approx acbcba \approx accbba$). Also we have $\mathbf{t}_1 = [abcbca] = \{abcbca, abccba, acbbca, acbcba, abbcca, accbba\}$, $\mathbf{t}_2 = [abc] = \{abc, acb\}$ and $\mathbf{t}_3 = [bca] = \{bca, cba\}$ are traces. Note that $\mathbf{t}_1 = \mathbf{t}_2 \circledast \mathbf{t}_3$ since $[abcbca] = [abc] \circledast [bca]$.

[4] Strictly speaking $EQ = \{bc = cb, cb = bc\}$ but standardly we consider the equations $bc = cb$ and $cb = bc$ as identical.

$$\begin{tikzcd}
 & b^{(1)} \arrow[r] & b^{(2)} \arrow[dr] & \\
a^{(1)} \arrow[ur] \arrow[dr] & & & a^{(2)} \\
 & c^{(1)} \arrow[r] & c^{(2)} \arrow[ur] &
\end{tikzcd}$$

**Fig. 2.** Partial order generated by the trace [*abcbca*].

Each trace can be interpreted as a finite partial order. Let $\mathbf{t} = \{x_1, \ldots, x_k\}$ be a trace, and let $\lhd_{x_i}$ denote the total order induced by the sequence $x_i$, $i = 1, \ldots, k$. Note that $\Sigma_{x_i} = \Sigma_{x_j}$ for all $i, j = 1, \ldots, n$, so we can define $\Sigma_t = \Sigma_{x_i}$, $i = 1, \ldots, n$. For example, the set of event occurrences of the trace $\mathbf{t}_1$ from Example 2 is $\Sigma_{\mathbf{t}_1} = \{a^{(1)}, b^{(1)}, c^{(1)}, a^{(2)}, b^{(2)}, c^{(2)}\}$. Each $\lhd_i$ is a total order on $\Sigma_{\mathbf{t}}$. The partial order generated by $\mathbf{t}$ can then be defined as $\prec_{\mathbf{t}} = \bigcap_{i=1}^{k} \lhd_{x_i}$. In fact, the set $\{\lhd_{x_1}, \ldots, \lhd_{x_k}\}$ consists of all total extensions of $\prec_{\mathbf{t}}$ (see [23,24]). Thus, the trace $\mathbf{t}_1 = [abcbca]$ from Example 2 can be interpreted as the partial order $\prec_{\mathbf{t}_1}$ depicted in Fig. 2 (arcs inferred from transitivity are omitted for simplicity).

**Remark 1.** Given a sequence $s$, to construct the partial order $\prec_{[s]}$ generated by $[s]$, we *do not* need to build up to exponentially many elements of $[s]$. We can simply construct the direct acyclic graph $(\Sigma_{[s]}, \prec_s)$, where $x^{(i)} \prec_s y^{(j)}$ iff $x^{(i)}$ occurs before $y^{(j)}$ on the sequence $s$ and $(x, y) \notin ind$. The relation $\prec_s$ is usually *not* the same as the partial order $\prec_{[s]}$. However, after applying the *transitive closure* operator, we have $\prec_{[s]} = \prec_s^+$ (cf. [4]). We will later see how this idea is generalized to the construction of so-structures and gso-structures from their "trace" representations. Note that to do so, it is inevitable that we have to generalize the *transitive closure* operator to these order structures.

From the concurrency point of view, the trace quotient monoid representation has a fundamental advantage over its labeled poset representation when studying the formal linguistic aspects of concurrent behaviors, e.g., Ochmański's characterization of recognizable trace language [25] and Zielonka's theory of asynchronous automata [30]. For more details on traces and their various properties, the reader is referred to the monograph [4]. The reader is also referred to [1] for interesting discussions on the trade-offs: traces vs. labeled partial order models that allow auto-concurrency, e.g., pomsets.

## 4. Relational structures model of concurrency

Even though partial orders are one of the main tools for modelling "true concurrency", they have some limitations. While they can sufficiently model the "earlier than" relationship, they can model neither the "not later than" relationship nor the "non-simultaneously" relationship. It was shown in [10] that any reasonable concurrent behavior can be modelled by an appropriate *pair of relations*. This leads to the theory of *relational structures models of concurrency* [12,6,8] (see [8] for a detailed bibliography and history).

In this section, we review the theory of *stratified order structures* of [12] and *generalized stratified order structures* of [6, 8]. The former can model both the "earlier than" and the "not later than" relationships, but not the "non-simultaneously" relationship. The latter can model all three relationships.

While traces provide sequence representations of causal partial orders, their extensions, comtraces and generalized comtraces discussed in the following sections, are *step sequence* representations of stratified order structures and generalized stratified order structures respectively.

Since the theory of relational order structures is far less known than the theory of causal partial orders, we will not only give appropriate definitions but also introduce some intuition and motivation behind those definitions using simple examples.

We start with the concept of an *observation*:

An *observation* (also called a *run* or an *instance of concurrent behavior*) is an abstract model of the execution of a concurrent system.

It was argued in [10] that *an observation must be a total, stratified or interval order* (interval orders are not used in this paper). Totally ordered observations can be represented by sequences while stratified observations can be represented by step sequences.

The next concept is a *concurrent behavior*:

A *concurrent behavior* (*concurrent history*) is a set of equivalent observations.

When totally ordered observations are sufficient to define whole concurrent behaviors, then the concurrent behaviors can entirely be described by causal partial orders. However if concurrent behaviors consist of more sophisticated sets of stratified observations, e.g., to model the "not later than" relationship or the "non-simultaneously" relationship, then we need relational structures [10].

### 4.1. Stratified order structure

By a *relational structure*, we mean a triple $T = (X, R_1, R_2)$, where $X$ is a set and $R_1$, $R_2$ are binary relations on $X$. A relational structure $T' = (X', R'_1, R'_2)$ is an *extension* of $T$, denoted as $T \subseteq T'$, iff $X = X'$, $R_1 \subseteq R'_1$ and $R_2 \subseteq R'_2$.

**Definition 3** *(Stratified order structure).* (See [12].) A *stratified order structure* (*so-structure*) is a relational structure $S = (X, \prec, \sqsubset)$, such that for all $a, b, c \in X$, the following hold:

$$\begin{array}{llll} \text{S1:} & a \not\sqsubset a, & \text{S3:} & a \sqsubset b \sqsubset c \wedge a \neq c \implies a \sqsubset c, \\ \text{S2:} & a \prec b \implies a \sqsubset b, & \text{S4:} & a \sqsubset b \prec c \vee a \prec b \sqsubset c \implies a \prec c. \end{array}$$

When $X$ is finite, $S$ is called a *finite so-structure*.

Note that the axioms S1–S4 imply that $(X, \prec)$ is a poset and $a \prec b \Rightarrow b \not\sqsubset a$. The relation $\prec$ is called *causality* and represents the "earlier than" relationship, and the relation $\sqsubset$ is called *weak causality* and represents the "not later than" relationship. The axioms S1–S4 model the mutual relationship between "earlier than" and "not later than" relations, *provided that the system runs are modelled by stratified orders*.

The concept of so-structures were independently introduced in [5] and [9] (the axioms are slightly different from S1–S4, although equivalent). Their comprehensive theory has been presented in [12]. They have been successfully applied to model inhibitor and priority systems, asynchronous races, synthesis problems, etc. (see for example [11,26,17,18,16,19,20]). The name follows from the following result.

**Proposition 4.** *(See [10].) For every stratified order $\lhd$ on $X$, the triple $S_\lhd = (X, \lhd, \lhd^\frown)$ is a so-structure.* $\square$

**Definition 4** *(Stratified extension of so-structure).* (See [12].) A *stratified* order $\lhd$ on $X$ is a *stratified extension* of a so-structure $S = (X, \prec, \sqsubset)$ if for all $\alpha, \beta \in X$,

$$\alpha \prec \beta \implies \alpha \lhd \beta \quad \text{and} \quad \alpha \sqsubset \beta \implies \alpha \lhd^\frown \beta.$$

The set of all stratified extensions of $S$ is denoted as $ext(S)$.

According to Szpilrajn's theorem, every poset can be reconstructed by taking the intersection of all of its total extensions. A similar result holds for so-structures and stratified extensions.

**Theorem 2.** *(See [12, Theorem 2.9].) Let $S = (X, \prec, \sqsubset)$ be a so-structure. Then*

$$S = \left( X, \bigcap_{\lhd \in ext(S)} \lhd, \bigcap_{\lhd \in ext(S)} \lhd^\frown \right). \qquad \square$$

The set $ext(S)$ also has the following internal property that will be useful in various proofs.

**Theorem 3.** *(See [10].) Let $S = (X, \prec, \sqsubset)$ be a so-structure. Then for every $a, b \in X$,*

$$\big(\exists \lhd \in ext(S).a \lhd b\big) \wedge \big(\exists \lhd \in ext(S).b \lhd a\big) \implies \exists \lhd \in ext(S).a \frown_\lhd b. \qquad \square$$

The classification of concurrent behaviors provided in [10] says that a concurrent behavior conforms to the paradigm[5] $\pi_3$ if it has the same property as stated in Theorem 3 for $ext(S)$. In other words, Theorem 3 states that the set $ext(S)$ conforms to the paradigm $\pi_3$.

### 4.2. Generalized stratified order structure

The stratified order structures can adequately model concurrent histories only when the paradigm $\pi_3$ is satisfied. For the general case, we need *gso-structures* introduced in [6] also under the assumption that the system runs are defined as stratified orders.

**Definition 5** *(Generalized stratified order structure).* (See [6,8].) A *generalized stratified order structure* (*gso-structure*) is a relational structure $G = (X, <>, \sqsubset)$ such that $\sqsubset$ is irreflexive, $<>$ is symmetric and irreflexive, and the triple $S_G = (X, \prec_G, \sqsubset)$, where $\prec_G = <> \cap \sqsubset$, is a so-structure, called the *so-structure induced by G*. When $X$ is finite, $G$ is called a finite gso-structure.

[5] A *paradigm* is a supposition or statement about the structure of a concurrent behavior (concurrent history) involving a treatment of *simultaneity*. See [8,10] for more details.

The relation $<>$ is called *commutativity* and represents the "non-simultaneously" relationship, while the relation $\sqsubset$ is called *weak causality* and represents the "not later than" relationship.

For a binary relation $R$ on $X$, we let $R^{\mathrm{sym}} \triangleq R \cup R^{-1}$ denote the *symmetric closure* of $R$.

**Definition 6** *(Stratified extension of gso-structure).* (See [6,8].) A stratified order $\lhd$ on $X$ is a *stratified extension* of a gso-structure $G = (X, <>, \sqsubset)$ if for all $\alpha, \beta \in X$,

$$\alpha <> \beta \implies \alpha \lhd^{\mathrm{sym}} \beta \quad \text{and} \quad \alpha \sqsubset \beta \implies \alpha \lhd^{\frown} \beta.$$

The set of all stratified extensions of $G$ is denoted as $ext(G)$.

Every gso-structure can also be uniquely reconstructed from its stratified extensions. The generalization of Szpilrajn's theorem for gso-structures can be stated as the following.

**Theorem 4.** *(See [6,8].) Let $G = (X, <>, \sqsubset)$ be a gso-structure. Then*

$$G = \left( X, \bigcap_{\lhd \in ext(G)} \lhd^{\mathrm{sym}}, \bigcap_{\lhd \in ext(G)} \lhd^{\frown} \right). \qquad \square$$

The gso-structures *do not* have an equivalent of Theorem 3. As a counter-example consider $G = (\{a, b, c\}, <>_4, \sqsubset_4)$ where $<>_4$ and $\sqsubset_4$ are those from Fig. 1. Hence $ext(G) = obs(P_4) = \{o_1, o_2\}$, where $o_1 = \{a\}\{b\}\{c\}$ and $o_2 = \{a\}\{c\}\{b\}$. For this gso-structure we have $b \xrightarrow{o_1} c$ and $c \xrightarrow{o_2} b$, but neither $o_1$ nor $o_2$ contains the step $\{b, c\}$, so Theorem 3 does not hold. The lack of an equivalent of Theorem 3 makes proving properties about gso-structures more difficult, but they can model the most general concurrent behaviors provided that observations are modelled by stratified orders [8].

## 5. Comtraces

The standard definition of a free monoid $(E^*, *, \lambda)$ assumes that the elements of $E$ have no internal structure (or their internal structure does not affect any monoidal properties), and they are often called 'letters', 'symbols', 'names', etc. When we assume the elements of $E$ have some internal structure, for instance that they are sets, this internal structure may be used when defining the set of equations $EQ$. This idea is exploited in the concept of a *comtrace*.

*Comtraces* (*com*bined *traces*), introduced in [11] as an extension of traces to distinguish between "earlier than" and "not later than" phenomena, are equational monoids of step sequence monoids. The equations $EQ$ are in this case defined implicitly via two relations: *simultaneity* and *serializability*.

**Definition 7** *(Comtrace alphabet).* (See [11].) Let $E$ be a finite set (of events) and let $ser \subseteq sim \subset E \times E$ be two relations called *serializability* and *simultaneity* respectively and the relation $sim$ is irreflexive and symmetric. Then the triple $(E, sim, ser)$ is called a *comtrace alphabet*.

Intuitively, if $(a, b) \in sim$ then $a$ and $b$ can occur simultaneously (or be a part of a *synchronous* occurrence in the sense of [17]), while $(a, b) \in ser$ means that $a$ and $b$ may occur simultaneously and also $a$ may occur before $b$ (i.e., both executions are equivalent). We define $\mathbb{S}$, the set of all (potential) *steps*, as the set of all cliques of the graph $(E, sim)$, i.e.,

$$\mathbb{S} \triangleq \left\{ A \mid A \neq \emptyset \wedge \forall a, b \in A. \big(a = b \vee (a, b) \in sim\big) \right\}.$$

**Definition 8** *(Comtrace congruence).* (See [11].) Let $\theta = (E, sim, ser)$ be a comtrace alphabet and let $\equiv_{ser}$, called *comtrace congruence*, be the $EQ$-congruence defined by the set of equations

$$EQ \triangleq \left\{ A = BC \mid A = B \cup C \in \mathbb{S} \wedge B \times C \subseteq ser \right\}.$$

Then the equational monoid $(\mathbb{S}^*/\equiv_{ser}, \circledast, [\lambda])$ is called a monoid of *comtraces* over $\theta$.

Since $ser$ is irreflexive, for each $(A = BC) \in EQ$ we have $B \cap C = \emptyset$. By Proposition 1, the comtrace congruence relation can also be defined explicitly in non-equational form as follows.

**Proposition 5.** *Let $\theta = (E, sim, ser)$ be a comtrace alphabet and let $\mathbb{S}^*$ be the set of all step sequences defined on $\theta$. Let $\approx_{ser} \subseteq \mathbb{S}^* \times \mathbb{S}^*$ be the relation comprising all pairs $(t, u)$ of step sequences such that $t = wAz$ and $u = wBCz$, where $w, z \in \mathbb{S}^*$ and $A$, $B$, $C$ are steps satisfying $B \cup C = A$ and $B \times C \subseteq ser$. Then $\equiv_{ser} = (\approx_{ser} \cup \approx_{ser}^{-1})^*$.* $\square$

We will omit the subscript $ser$ from comtrace congruence and $\approx_{ser}$, and only write $\equiv$ and $\approx$ if it causes no ambiguity.

**Example 3.** Let $E = \{a, b, c\}$ where $a$, $b$ and $c$ are three atomic operations, where

$$a: \quad y := x + y, \qquad b: \quad x := y + 2, \qquad c: \quad y := y + 1.$$

Assume simultaneous reading is allowed, but simultaneous writing is not allowed. Then the events $b$ and $c$ can be performed simultaneously, and the execution of the step $\{b, c\}$ gives the same outcome as executing $b$ followed by $c$. The events $a$ and $b$ can also be performed simultaneously, but the outcome of executing the step $\{a, b\}$ is not the same as executing $a$ followed by $b$, or $b$ followed by $a$. Note that although executing the steps $\{a, b\}$ and $\{b, c\}$ is allowed, we cannot execute the step $\{a, c\}$ since that would require writing on the same variable $y$.

Let $E = \{a, b, c\}$ be the set of events. Then we can define the comtrace alphabet $\theta = (E, sim, ser)$, where $sim = \{(a, b), (b, a), (b, c), (c, b)\}$ and $ser = \{(b, c)\}$. Thus the set of all possible steps is

$$\mathbb{S}_\theta = \big\{\{a\}, \{b\}, \{c\}, \{a, b\}, \{b, c\}\big\}.$$

We observe that the set $\mathbf{t} = [\{a\}\{a, b\}\{b, c\}] = \{\{a\}\{a, b\}\{b, c\}, \{a\}\{a, b\}\{b\}\{c\}\}$ is a comtrace. But the step sequence $\{a\}\{a, b\}\{c\}\{b\}$ is not an element of $\mathbf{t}$ because $(c, b) \notin ser$.

Even though traces are quotient monoids over sequences and comtraces are quotient monoids over step sequences (and the fact that steps are sets is used in the definition of quotient congruence), traces can be regarded as a special case of comtraces. In principle, each trace commutativity equation $ab = ba$ corresponds to two comtrace equations $\{a, b\} = \{a\}\{b\}$ and $\{a, b\} = \{b\}\{a\}$. This relationship can formally be formulated as follows.

Let $(E, ind)$ and $(E, sim, ser)$ be trace and comtrace alphabets respectively. For each sequence $x = a_1 \dots a_n \in E^*$, we define $x^{\{\}} = \{a_1\} \dots \{a_n\}$ to be its corresponding step sequence, which in this case consists of only singleton steps.

**Lemma 1.**

1. *Assume $ser = sim$. Then for each comtrace $\mathbf{t} \in \mathbb{S}^*/{\equiv_{ser}}$ there exists a step sequence $x = \{a_1\} \dots \{a_k\} \in \mathbb{S}^*$ such that $\mathbf{t} = [x]_{\equiv_{ser}}$.*
2. *If $ser = sim = ind$, then for each $x, y \in E^*$, we have $x \equiv_{ind} y \iff x^{\{\}} \equiv_{ser} y^{\{\}}$.*

**Proof.** (1) follows from the fact that if $ser = sim$, then for each $A = \{a_1, \dots, a_k\} \in \mathbb{S}$, we have $A \equiv_{ser} \{a_1\} \dots \{a_k\}$. (2) is a simple consequence of the definition of $x^{\{\}}$. $\square$

Let $\mathbf{t}$ be a trace over $(E, ind)$ and let $\mathbf{v}$ be a comtrace over $(E, sim, ser)$. We say that $\mathbf{t}$ and $\mathbf{v}$ are *tantamount* if $sim = ser = ind$ and there is $x \in E^*$ such that $\mathbf{t} = [x]_{\equiv_{ind}}$ and $\mathbf{v} = [x^{\{\}}]_{\equiv_{ser}}$. If a trace $\mathbf{t}$ and a comtrace $\mathbf{v}$ are equivalent we will write $\mathbf{t} \overset{\mathsf{t}\leftrightsquigarrow\mathsf{c}}{\equiv} \mathbf{v}$. Note that Lemma 1 guarantees that this definition is valid.

**Proposition 6.** *Let $\mathbf{t}, \mathbf{r}$ be traces and $\mathbf{v}, \mathbf{w}$ be comtraces. Then*

1. $\mathbf{t} \overset{\mathsf{t}\leftrightsquigarrow\mathsf{c}}{\equiv} \mathbf{v} \wedge \mathbf{t} \overset{\mathsf{t}\leftrightsquigarrow\mathsf{c}}{\equiv} \mathbf{w} \Longrightarrow \mathbf{v} = \mathbf{w}$.
2. $\mathbf{t} \overset{\mathsf{t}\leftrightsquigarrow\mathsf{c}}{\equiv} \mathbf{v} \wedge \mathbf{r} \overset{\mathsf{t}\leftrightsquigarrow\mathsf{c}}{\equiv} \mathbf{v} \Longrightarrow \mathbf{t} = \mathbf{r}$.

**Proof.** 1. $\mathbf{t} \overset{\mathsf{t}\leftrightsquigarrow\mathsf{c}}{\equiv} \mathbf{v}$ means that there is $x \in E^*$ such that $\mathbf{t} = [x]_{\equiv_{ind}}$ and $\mathbf{v} = [x^{\{\}}]_{\equiv_{ser}}$, and $\mathbf{t} \overset{\mathsf{t}\leftrightsquigarrow\mathsf{c}}{\equiv} \mathbf{w}$ means that there is $y \in E^*$ such that $\mathbf{t} = [y]_{\equiv_{ind}}$ and $\mathbf{w} = [y^{\{\}}]_{\equiv_{ser}}$. Since $\mathbf{t} = [x]_{\equiv_{ind}} = [y]_{\equiv_{ind}}$ then $x \equiv_{ind} y$ and by Lemma 1(2), $x^{\{\}} \equiv_{ser} y^{\{\}}$, i.e. $\mathbf{v} = \mathbf{w}$.

2. Similarly as (1). $\square$

Equivalent traces and comtraces generate identical partial orders. However, we will postpone the discussion of this issue to Section 9. Hence *traces can be regarded as a special case of comtraces.*

Note that comtrace might be a useful notion to formalize the concept of *synchrony* (cf. [17]). In principle, events $a_1, \dots, a_k$ are *synchronous* if they can be executed in one step $\{a_1, \dots, a_k\}$ but this execution cannot be modelled by any sequence of proper subsets of $\{a_1, \dots, a_k\}$. Note that in general 'synchrony' is not necessarily 'simultaneity' as it does not include the concept of time [15]. It appears, however, that the mathematics to deal with synchrony are close to that to deal with simultaneity.

**Definition 9** *(Independency and synchrony).* Let $(E, sim, ser)$ be a given comtrace alphabet. We define the relations $ind$, $syn$ and the set $\mathbb{S}_{syn}$ as follows:

- $ind \subseteq E \times E$, called *independency*, and defined as $ind = ser \cap ser^{-1}$,
- $syn \subseteq E \times E$, called *synchrony*, and defined as:

$$(a, b) \in syn \quad \overset{df}{\iff} \quad (a, b) \in sim \setminus ser^{\mathsf{sym}},$$

- $\mathbb{S}_{syn} \subseteq \mathbb{S}$, called *synchronous steps*, and defined as:

$$A \in \mathbb{S}_{syn} \quad \overset{df}{\Longleftrightarrow} \quad A \neq \emptyset \wedge (\forall a, b \in A.(a, b) \in syn).$$

If $(a, b) \in ind$ then $a$ and $b$ are *independent*, i.e., executing them either simultaneously, or $a$ followed by $b$, or $b$ followed by $a$, will yield exactly the same result. If $(a, b) \in syn$ then $a$ and $b$ are *synchronous*, which means they might be executed in one step, either $\{a, b\}$ or as a part of bigger step, but such an execution of $\{a, b\}$ is not equivalent to either $a$ followed by $b$, or $b$ followed by $a$. In principle, the relation $syn$ is a counterpart of 'synchrony' (cf. [17]). If $A \in \mathbb{S}_{syn}$, then the set of events $A$ can be executed as one step, but it *cannot* be simulated by any sequence of its subsets.

**Example 4.** Assume we have $E = \{a, b, c, d, e\}$, $sim = \{(a, b), (b, a), (a, c), (c, a), (a, d), (d, a)\}$, and $ser = \{(a, b), (b, a), (a, c)\}$. Hence, $\mathbb{S} = \{\{a, b\}, \{a, c\}, \{a, d\}, \{a\}, \{b\}, \{c\}, \{e\}\}$, and

$$ind = \big\{(a, b), (b, a)\big\}, \qquad syn = \big\{(a, d), (d, a)\big\}, \qquad \mathbb{S}_{syn} = \big\{\{a, d\}\big\}.$$

Since $\{a, d\} \in \mathbb{S}_{syn}$, the step $\{a, d\}$ *cannot* be split into smaller steps. For example the comtraces $\mathbf{x}_1 = [\{a, b\}\{c\}\{a\}]$, $\mathbf{x}_2 = [\{e\}\{a, d\}\{a, c\}]$, and $\mathbf{x}_3 = [\{a, b\}\{c\}\{a\}\{e\}\{a, d\}\{a, c\}]$ are respectively the following sets of step sequences:

$$\mathbf{x}_1 = \big\{\{a, b\}\{c\}\{a\}, \{a\}\{b\}\{c\}\{a\}, \{b\}\{a\}\{c\}\{a\}, \{b\}\{a, c\}\{a\}\big\},$$
$$\mathbf{x}_2 = \big\{\{e\}\{a, d\}\{a, c\}, \{e\}\{a, d\}\{a\}\{c\}\big\},$$
$$\mathbf{x}_3 = \left\{\begin{array}{l} \{a, b\}\{c\}\{a\}\{e\}\{a, d\}\{a, c\}, \{a\}\{b\}\{c\}\{a\}\{e\}\{a, d\}\{a, c\}, \\ \quad \{b\}\{a\}\{c\}\{a\}\{e\}\{a, d\}\{a, c\}, \{b\}\{a, c\}\{a\}\{e\}\{a, d\}\{a, c\}, \\ \quad \{a, b\}\{c\}\{a\}\{e\}\{a, d\}\{a\}\{c\}, \{a\}\{b\}\{c\}\{a\}\{e\}\{a, d\}\{a\}\{c\}, \\ \quad \{b\}\{a\}\{c\}\{a\}\{e\}\{a, d\}\{a\}\{c\}, \{b\}\{a, c\}\{a\}\{e\}\{a, d\}\{a\}\{c\} \end{array}\right\}.$$

We also have $\mathbf{x}_3 = \mathbf{x}_1 \circledast \mathbf{x}_2$. Note that since $(c, a) \notin ser$, $\{a, c\} \equiv_{ser} \{a\}\{c\} \not\equiv_{ser} \{c\}\{a\}$.

We can easily extend the concepts of comtraces to the level of languages, with potential applications similar to traces. For any step sequence language $L$, we define a comtrace language $[L]_\Theta$ (or just $[L]$) to be the set $\{[u] \mid u \in L\}$. The languages of comtraces provide a bridge between operational and structural semantics. In other words, if a step sequence language $L$ describes an operational semantics of a given concurrent system, we only need to derive the comtrace alphabet $(E, sim, ser)$ from the system, and the comtrace language $[L]$ defines the structural semantics of the system.

**Example 5.** Consider the following simple concurrent system Priority, which comprises two sequential subsystems such that

- the first subsystem can cyclically engage in event $a$ followed by event $b$,
- the second subsystem can cyclically engage in event $b$ or in event $c$,
- the two systems synchronize by means of handshake communication,
- there is a priority constraint stating that if it is possible to execute event $b$, then $c$ must not be executed.

This example has often been analyzed in the literature (cf. [13]), usually under the interpretation that $a =$ 'Error Message', $b =$ 'Stop And Restart', and $c =$ 'Some Action'. It can be formally specified in various notations including *Priority* and *Inhibitor Nets* (cf. [9,12]). Its operational semantics (easily found in any model) can be defined by the following step sequence language

$$L_{\mathsf{Priority}} \triangleq Pref\Big(\big(\{c\}^* \cup \{a\}\{b\} \cup \{a, c\}\{b\}\big)^*\Big),$$

where $Pref(L) \triangleq \bigcup_{w \in L} \{u \in L \mid \exists v.uv = w\}$ denotes the *prefix closure* of $L$.

The rules for deriving the comtrace alphabet $(E, sim, ser)$ depend on the model, and for Priority, the set of possible steps is $\mathbb{S} = \{\{a\}, \{b\}, \{c\}, \{a, c\}\}$, and $ser = \{(c, a)\}$ and $sim = \{(a, c), (c, a)\}$. Then, $[L_{\mathsf{Priority}}]$ defines the structural comtrace semantics of Priority. For instance, the comtrace $[\{a, c\}\{b\}] = \{\{c\}\{a\}\{b\}, \{a, c\}\{b\}\}$ is in the language $[L_{\mathsf{Priority}}]$.

## 6. Generalized comtraces

There are reasonable concurrent behaviors that cannot be modelled by any comtrace. Let us analyze the following example.

**Example 6.** Let $E = \{a, b, c\}$ where $a$, $b$ and $c$ are three atomic operations defined as follows (we assume simultaneous reading is allowed):

$$a:\quad x := x + 1, \qquad b:\quad x := x + 2, \qquad c:\quad y := y + 1.$$

It is reasonable to consider them all as 'concurrent' as any order of their executions yields exactly the same results (see [10,12] for more motivation and formal considerations as well as the program $P_4$ of Example 1). Assume that simultaneous reading is allowed, but simultaneous writing is not. Then while simultaneous executions of $\{a, c\}$ and $\{b, c\}$ are allowed, we cannot execute $\{a, b\}$, since simultaneous writing on the same variable $x$ is not allowed.

The set of all equivalent executions (or runs) involving one occurrence of the operations $a$, $b$ and $c$, and modelling the above case,

$$\mathbf{x} = \left\{ \begin{array}{c} \{a\}\{b\}\{c\}, \{a\}\{c\}\{b\}, \{b\}\{a\}\{c\}, \{b\}\{c\}\{a\}, \{c\}\{a\}\{b\}, \\ \{c\}\{b\}\{a\}, \{a, c\}\{b\}, \{b, c\}\{a\}, \{b\}\{a, c\}, \{a\}\{b, c\} \end{array} \right\},$$

is a valid concurrent history [10,12]. However $x$ is *not* a comtrace. The problem is that we have $\{a\}\{b\} \equiv \{b\}\{a\}$ but $\{a, b\}$ *is not* a valid step, so comtrace cannot represent this situation.

In this section, we will introduce the *generalized comtrace* notion (*g-comtrace*), an extension of comtrace, which is also defined over step sequences. The g-comtraces will be able to model "non-simultaneously" relationship similar to the one from Example 6.

**Definition 10** *(Generalized comtrace alphabet).* Let $E$ be a finite set (of events). Let *ser*, *sim* and *inl* be three relations on $E$ called *serializability*, *simultaneity* and *interleaving* respectively satisfying the following conditions:

- *sim* and *inl* are irreflexive and symmetric,
- $ser \subseteq sim$, and
- $sim \cap inl = \emptyset$.

Then the triple $(E, sim, ser, inl)$ is called a *g-comtrace alphabet*.

The interpretation of the relations *sim* and *ser* is as in Definition 7, and $(a, b) \in inl$ means $a$ and $b$ cannot occur simultaneously, but their occurrence in any order is equivalent. As for comtraces, we define the set $\mathbb{S}$ of all (potential) *steps* as the set of all cliques of the graph $(E, sim)$.

**Definition 11** *(Generalized comtrace congruence).* Let $\Theta = (E, sim, ser, inl)$ be a g-comtrace alphabet and let $\equiv_{\{ser,inl\}}$, called *g-comtrace congruence*, be the *EQ*-congruence defined by the set of equations $EQ = EQ_1 \cup EQ_2$, where

$$EQ_1 \triangleq \{A = BC \mid A = B \cup C \in \mathbb{S} \wedge B \times C \subseteq ser\},$$

$$EQ_2 \triangleq \{BA = AB \mid A \in \mathbb{S} \wedge B \in \mathbb{S} \wedge A \times B \subseteq inl\}.$$

The equational monoid $(\mathbb{S}^*/\equiv_{\{ser,inl\}}, \circledast, [\lambda])$ is called a *monoid of g-comtraces* over $\Theta$.

Since *ser* and *inl* are irreflexive, $(A = BC) \in EQ_1$ implies $B \cap C = \emptyset$, and $(AB = BA) \in EQ_2$ implies $A \cap B = \emptyset$. Since $inl \cap sim = \emptyset$, we also have that if $(AB = BA) \in EQ_2$, then $A \cup B \notin \mathbb{S}$.

By Proposition 1, the g-comtrace congruence relations can also be defined explicitly in non-equational form as follows.

**Definition 12.** Let $\Theta = (E, sim, ser, inl)$ be a g-comtrace alphabet and let $\mathbb{S}^*$ be the set of all step sequences defined on $\Theta$.

- Let $\approx_1 \subseteq \mathbb{S}^* \times \mathbb{S}^*$ be the relation comprising all pairs $(t, u)$ of step sequences such that $t = wAz$ and $u = wBCz$ where $w, z \in \mathbb{S}^*$ and $A$, $B$, $C$ are steps satisfying $B \cup C = A$ and $B \times C \subseteq ser$.
- Let $\approx_2 \subseteq \mathbb{S}^* \times \mathbb{S}^*$ be the relation comprising all pairs $(t, u)$ of step sequences such that $t = wABz$ and $u = wBAz$ where $w, z \in \mathbb{S}^*$ and $A$, $B$ are steps satisfying $A \times B \subseteq inl$.

We define $\approx_{\{ser,inl\}}$ as $\approx_{\{ser,inl\}} \triangleq \approx_1 \cup \approx_2$.

**Proposition 7.** *For each g-comtrace alphabet* $\Theta = (E, sim, ser, inl)$

$$\equiv_{\{ser,inl\}} = \left(\approx_{\{ser,inl\}} \cup \approx_{\{ser,inl\}}^{-1}\right)^*.$$

**Proof.** Follows from Proposition 1. $\square$

The name "generalized comtraces" comes from the fact that when $inl = \emptyset$, Definition 11 coincides with Definition 8 of a comtrace monoid. We will omit the subscript $\{ser, inl\}$ from $\equiv_{\{ser,inl\}}$ and $\approx_{\{ser,inl\}}$, and write $\equiv$ and $\approx$ when causing no ambiguity.

**Example 7.** The set $\mathbf{x}$ from Example 6 is a g-comtrace, where we have $E = \{a, b, c\}$, $ser = sim = \{(a, c), (c, a), (b, c), (c, b)\}$, $inl = \{(a, b), (b, a)\}$, and $\mathbb{S} = \{\{a, c\}, \{b, c\}, \{a\}, \{b\}, \{c\}\}$.

It is worthnoting that there is an *important difference* between the equation $ab = ba$ for traces, and the equation $\{a\}\{b\} = \{b\}\{a\}$ for g-comtrace monoids. For traces, the equation $ab = ba$, when translated into step sequences, corresponds to two equations $\{a, b\} = \{a\}\{b\}$ and $\{a, b\} = \{b\}\{a\}$, which implies $\{a\}\{b\} \equiv \{a, b\} \equiv \{b\}\{a\}$. For g-comtrace monoids, the equation $\{a\}\{b\} = \{b\}\{a\}$ implies that $\{a, b\}$ *is not a step*, i.e., neither the equation $\{a, b\} = \{a\}\{b\}$ nor the equation $\{a, b\} = \{b\}\{a\}$ belongs to the set of equations. In other words, for traces the equation $ab = ba$ means 'independency', i.e., executing $a$ and $b$ in any order or simultaneously will yield the same consequence. For g-comtrace monoids, the equation $\{a\}\{b\} = \{b\}\{a\}$ means that execution of $a$ and $b$ in any order yields the same result, but executing of $a$ and $b$ in any order is *not* equivalent to executing them simultaneously.

## 7. Algebraic properties of comtrace and generalized comtrace congruences

Algebraic properties of trace congruence operations such as *left/right cancellation* and *projection* are well understood. They are intuitive and simple tools with many applications [24]. In this section we will generalize these cancellation and projection properties to comtrace and g-comtrace. The basic obstacle is switching from sequences to step sequences.

### 7.1. Properties of comtrace congruence

Let us consider a comtrace alphabet $\theta = (E, sim, ser)$ where we reserve $\mathbb{S}$ to denote the set of all possible steps of $\theta$ throughout this section.

For each step sequence or enumerated step sequence $x = X_1 \dots X_k$, we define the *step sequence weight* of $x$ as $weight(x) \triangleq \sum_{i=1}^{k} |X_i|$. We also define $\biguplus(x) \triangleq \bigcup_{i=1}^{k} X_i$.

Due to the commutativity of the independency relation for traces, the *mirror rule*, which says if two sequences are congruent, then their *reverses* are also congruent, holds for *trace congruence* [4]. Hence, in trace theory, we only need a *right cancellation* operation to produce congruent *subsequences* from congruent sequences, since the *left cancellation* comes from the right cancellation of the reverses.

However, the *mirror rule* does not hold for comtrace congruence since the relation *ser* is usually not commutative. Example 3 works as a counter-example since $\{a\}\{b, c\} \equiv \{a\}\{b\}\{c\}$ but $\{b, c\}\{a\} \not\equiv \{c\}\{b\}\{a\}$. Thus, we define separate left and right cancellation operators for comtraces.

Let $a \in E$, $A \in \mathbb{S}$ and $w \in \mathbb{S}^*$. The operator $\div_R$, *step sequence right cancellation*, is defined as follows:

$$\lambda \div_R a \triangleq \lambda, \qquad wA \div_R a \triangleq \begin{cases} (w \div_R a)A & \text{if } a \notin A, \\ w & \text{if } A = \{a\}, \\ w(A \setminus \{a\}) & \text{otherwise.} \end{cases}$$

Symmetrically, a *step sequence left cancellation* operator $\div_L$ is defined as follows:

$$\lambda \div_L a \triangleq \lambda, \qquad Aw \div_L a \triangleq \begin{cases} A(w \div_L a) & \text{if } a \notin A, \\ w & \text{if } A = \{a\}, \\ (A \setminus \{a\})w & \text{otherwise.} \end{cases}$$

Finally, for each $D \subseteq E$, we define the function $\pi_D : \mathbb{S}^* \to \mathbb{S}^*$, *step sequence projection* onto $D$, as follows:

$$\pi_D(\lambda) \triangleq \lambda, \qquad \pi_D(wA) \triangleq \begin{cases} \pi_D(w) & \text{if } A \cap D = \emptyset, \\ \pi_D(w)(A \cap D) & \text{otherwise.} \end{cases}$$

The algebraic properties of comtraces are similar to those of traces [24].

**Proposition 8.**

1. $u \equiv v \Longrightarrow weight(u) = weight(v)$ (*step sequence weight equality*),
2. $u \equiv v \Longrightarrow |u|_a = |v|_a$ (*event-preserving*),
3. $u \equiv v \Longrightarrow u \div_R a \equiv v \div_R a$ (*right cancellation*),
4. $u \equiv v \Longrightarrow u \div_L a \equiv v \div_L a$ (*left cancellation*),
5. $u \equiv v \Longleftrightarrow \forall s, t \in \mathbb{S}^*. sut \equiv svt$ (*step subsequence cancellation*),
6. $u \equiv v \Longrightarrow \pi_D(u) \equiv \pi_D(v)$ (*projection rule*).

**Proof.** The proofs use the same techniques as in [24]. We would like recall only the following key observation that simplifies the proof of this proposition: since $\equiv$ is the symmetric transitive closure of $\approx$, it suffices to show that $u \approx v$ implies the right-hand side of (1)–(6). The rest follows naturally from the definition of comtrace $\approx$ and the congruence $\equiv$. $\square$

Note that $(w \div_R a) \div_R b = (w \div_R b) \div_R a$, so we define

$$w \div_R \{a_1, \ldots, a_k\} \triangleq \big(\ldots\big((w \div_R a_1) \div_R a_2\big)\ldots\big) \div_R a_k, \quad \text{and}$$
$$w \div_R A_1 \ldots A_k \triangleq \big(\ldots\big((w \div_R A_1) \div_R A_2\big)\ldots\big) \div_R A_k.$$

We define dually for $\div_L$. Hence Proposition 8(4) and (5) can be generalized as follows.

**Corollary 1.** *For all $u, v, x \in \mathbb{S}^*$, we have*

1. $u \equiv v \Longrightarrow u \div_R x \equiv v \div_R x$.
2. $u \equiv v \Longrightarrow u \div_L x \equiv v \div_L x$. $\square$

*7.2. Properties of generalized comtrace congruence*

Using the same proof technique as in Proposition 8, we can show that g-comtrace congruence has the same algebraic properties as comtrace congruence.

**Proposition 9.** *Let $\mathbb{S}$ be the set of all steps over a g-comtrace alphabet $(E, sim, ser, inl)$ and $u, v \in \mathbb{S}^*$. Then*

1. $u \equiv v \Longrightarrow weight(u) = weight(v)$ (*step sequence weight equality*),
2. $u \equiv v \Longrightarrow |u|_a = |v|_a$ (*event-preserving*),
3. $u \equiv v \Longrightarrow u \div_R a \equiv v \div_R a$ (*right cancellation*),
4. $u \equiv v \Longrightarrow u \div_L a \equiv v \div_L a$ (*left cancellation*),
5. $u \equiv v \Longleftrightarrow \forall s, t \in \mathbb{S}^*. sut \equiv svt$ (*step subsequence cancellation*),
6. $u \equiv v \Longrightarrow \pi_D(u) \equiv \pi_D(v)$ (*projection rule*).

**Corollary 2.** *For all step sequences $u, v, x$ over a g-comtrace alphabet $(E, sim, ser, inl)$,*

1. $u \equiv v \Longrightarrow u \div_R x \equiv v \div_R x$,
2. $u \equiv v \Longrightarrow u \div_L x \equiv v \div_L x$. $\square$

The following proposition ensures that if any relation from the set $\{\leqslant, \geqslant, <, >, =, \neq\}$ holds for the positions of two event occurrences after applying cancellation or projection operations on a g-comtrace $[\bar{u}]$, then it also holds for the whole $[\bar{u}]$.

**Proposition 10.** *Let $\bar{u}$ be an enumerated step sequence over a g-comtrace alphabet $(E, sim, ser, inl)$ and $\alpha, \beta, \gamma \in \Sigma_u$ such that $\gamma \notin \{\alpha, \beta\}$. Let $\mathcal{R} \in \{\leqslant, \geqslant, <, >, =, \neq\}$. Then*

1. *if $\forall \bar{v} \in [\bar{u} \div_L \gamma]. pos_{\bar{v}}(\alpha) \mathcal{R}\, pos_{\bar{v}}(\beta)$, then $\forall \overline{w} \in [\bar{u}]. pos_{\overline{w}}(\alpha) \mathcal{R}\, pos_{\overline{w}}(\beta)$,*
2. *if $\forall \bar{v} \in [\bar{u} \div_R \gamma]. pos_{\bar{v}}(\alpha) \mathcal{R}\, pos_{\bar{v}}(\beta)$, then $\forall \overline{w} \in [\bar{u}]. pos_{\overline{w}}(\alpha) \mathcal{R}\, pos_{\overline{w}}(\beta)$,*
3. *if $S \subseteq \Sigma_u$ such that $\{\alpha, \beta\} \subseteq S$, then*

$$\big(\forall \bar{v} \in \big[\pi_S(\bar{u})\big]. pos_{\bar{v}}(\alpha) \mathcal{R}\, pos_{\bar{v}}(\beta)\big) \quad \Longrightarrow \quad \big(\forall \overline{w} \in [\bar{u}]. pos_{\overline{w}}(\alpha) \mathcal{R}\, pos_{\overline{w}}(\beta)\big).$$

**Proof.** 1. Assume that

$$\forall \bar{v} \in [\bar{u} \div_L \gamma]. pos_{\bar{v}}(\alpha) \mathcal{R}\, pos_{\bar{v}}(\beta). \tag{7.1}$$

Suppose for a contradiction that $\exists \overline{w} \in [\bar{u}]. \neg(pos_{\overline{w}}(\alpha) \mathcal{R}\, pos_{\overline{w}}(\beta))$. Since $\gamma \notin \{\alpha, \beta\}$, we have $\neg(pos_{\overline{w} \div_L \gamma}(\alpha) \mathcal{R}\, pos_{\overline{w} \div_L \gamma}(\beta))$. But $\overline{w} \in [\bar{u}]$ implies $\overline{w} \div_L \gamma \equiv \bar{u} \div_L \gamma$. Hence, $\overline{w} \div_L \gamma \in [\bar{u} \div_L \gamma]$ and $\neg(pos_{\overline{w} \div_L \gamma}(\alpha) \mathcal{R}\, pos_{\overline{w} \div_L \gamma}(\beta))$, contradicting (7.1).

2. Dually to part (1).

3. Assume that

$$\forall \bar{v} \in \big[\pi_S(\bar{u})\big]. pos_{\bar{v}}(\alpha) \mathcal{R}\, pos_{\bar{v}}(\beta). \tag{7.2}$$

Suppose for a contradiction that $\exists \overline{w} \in [\bar{u}]. \neg(pos_{\overline{w}}(\alpha) \mathcal{R}\, pos_{\overline{w}}(\beta))$. Since $\{\alpha, \beta\} \subseteq S$, we have $\neg(pos_{\pi_S(\overline{w})}(\alpha) \mathcal{R}\, pos_{\pi_S(\overline{w})}(\beta))$. But $\overline{w} \in [\bar{v}]$ implies $\pi_S(\overline{w}) \equiv \pi_S(\bar{u})$. Hence, $\pi_S(\overline{w}) \in [\pi_S(\bar{u})]$ and $\neg(pos_{\pi_S(\overline{w})}(\alpha) \mathcal{R}\, pos_{\pi_S(\overline{w})}(\beta))$, contradicting (7.2). $\square$

Clearly the above results also hold for comtraces as they are just g-comtraces with $inl = \emptyset$.

## 8. Maximally concurrent and canonical representations

In this section, we show that traces, comtraces and g-comtraces all have some special representations, that intuitively correspond to *maximally concurrent execution of concurrent histories*, i.e., "executing as much as possible in parallel". This kind of semantics is formally defined and analyzed for example in [3]. However such representations are truly unique only for comtraces. For traces and g-comtraces, unique (or *canonical*) representations are obtained by adding some arbitrary total ordering on their alphabets.

In this section we will start with the general case of g-comtraces and then consider comtraces and traces as a special case.

### 8.1. Representations of generalized comtraces

Let $\Theta = (E, sim, ser, inl)$ be a g-comtrace alphabet and $\mathbb{S}$ be the set of all steps over $\Theta$. We will start with the most "natural" definition which is the straightforward application of the approach used in [3] for an alternative version of traces called "vector firing sequences" (see [14,27]).

**Definition 13** *(Greedy maximally concurrent form).* A step sequence $u = A_1 \dots A_k \in \mathbb{S}^*$ is in *greedy maximally concurrent form* (*GMC-form*) if and only if for each $i = 1, \dots, k$:

$$(B_i y_i \equiv A_i \dots A_k) \quad \Longrightarrow \quad |B_i| \leqslant |A_i|,$$

where for all $i = 1, \dots, k$, $A_i, B_i \in \mathbb{S}$, and $y_i \in \mathbb{S}^*$.

**Proposition 11.** *For each g-comtrace* $\mathbf{u}$ *over* $\Theta$ *there is a step sequence* $u \in \mathbb{S}^*$ *in GMC-form such that* $\mathbf{u} = [u]$.

**Proof.** Let $u = A_1 \dots A_k$, where the steps $A_1, \dots, A_k$ are generated by the following simple greedy algorithm:

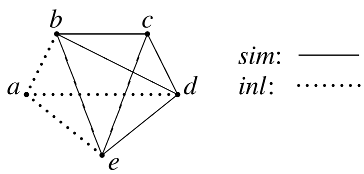
Initialize $i \leftarrow 0$ and $u_0 \leftarrow u$
**while** $u_i \neq \lambda$ **do**
    $i \leftarrow i + 1$
    Find $A_i$ such that there exists $y$ such that $A_i y \equiv u_{i-1}$ and for each $Bz \equiv A_i y \equiv u_{i-1}$, $|B| \leqslant |A_i|$
    $u_i \leftarrow u_{i-1} \div_L A_i$
**end while**
$k \leftarrow i - 1$.

Since $weight(u_{i+1}) < weight(u_i)$ the above algorithm always terminates. Clearly $u = A_1 \dots A_k$ is in GMC-form and $u \in \mathbf{u}$. □

The algorithm from the proof of Proposition 11 used to generate $A_1, \dots, A_k$ justifies the prefix "greedy" in Definition 13. However the GMC representation of g-comtraces is seldom unique and often not "maximally concurrent". Consider the following two examples.

**Example 8.** Let $E = \{a, b, c\}$, $sim = \{(a, c), (c, a)\}$, $ser = sim$ and $inl = \{(a, b), (b, a)\}$ and $\mathbf{u} = [\{a\}\{b\}\{c\}] = \{\{a\}\{b\}\{c\}, \{b\}\{a\}\{c\}, \{b\}\{a, c\}\}$. Note that both $\{a\}\{b\}\{c\}$ and $\{b\}\{a, c\}$ are in GMC-form, but only $\{b\}\{a, c\}$ can intuitively be interpreted as maximally concurrent.

**Example 9.** Let $E = \{a, b, c, d, e\}$, and $sim = ser$, $inl$ be as in the picture below, and let $\mathbf{u} = [\{a\}\{b, c, d, e\}]$. One can easily verify by inspection that $\{a\}\{b, c, d, e\}$ is the shortest element of $\mathbf{u}$ and the only element of $\mathbf{u}$ in GMC-form is $\{b, e, d\}\{a\}\{c\}$. The step sequence $\{b, e, d\}\{a\}\{c\}$ is longer and intuitively less maximally concurrent than the step sequence $\{a\}\{b, c, d, e\}$.




Hence for g-comtraces the greedy maximal concurrency notion is not necessarily the global maximal concurrency notion, so we will try another approach.

Let $x = A_1 \dots A_k$ be a step sequence. We define $length(A_1 \dots A_k) \triangleq k$. We also say that $A_i$ is *maximally concurrent in* $x$ if $B_i y_i \equiv A_i \dots A_k \Longrightarrow |B_i| \leqslant |A_i|$. Note that $A_k$ *is always maximally concurrent in* $x$, which makes the following definition correct.

For every step sequence $x = A_1 \dots A_k$, let $mc(x)$ be the smallest $i$ such that $A_i$ is maximally concurrent in $x$.

**Definition 14.** A step sequence $u = A_1 \dots A_k$ is *maximally concurrent* (*MC*-) iff

1. $v \equiv u \Longrightarrow length(u) \leqslant length(v)$,
2. for all $i = 1, \dots, k$ and for all $w$,

$$\big(u_i = A_i \dots A_k \equiv w \wedge length(u_i) = length(w)\big) \quad \Longrightarrow \quad mc(u_i) \leqslant mc(w).$$

**Theorem 5.** *For every g-comtrace* $\mathbf{u}$*, there exists a step sequence* $u \in \mathbf{u}$ *such that* $u$ *is maximally concurrent.*

**Proof.** Let $u_1 \in \mathbf{u}$ be a step sequence such that for each $v$, $v \equiv u_1 \Longrightarrow length(u_1) \leqslant length(v)$, and $(v \equiv u_1 \wedge length(u_1) = length(v)) \Longrightarrow mc(u_1) \leqslant mc(v)$. Obviously such $u_1$ exists for every g-comtrace $\mathbf{u}$. Assume that $u_1 = A_1 w_1$ and $length(u_1) = k$. Let $u_2$ be a step sequence satisfying $u_2 \equiv w_1$, $u_2 \equiv v \Longrightarrow length(u_2) \leqslant length(v)$, and $(v \equiv u_2 \wedge length(u_2) = length(v)) \Longrightarrow mc(u_2) \leqslant mc(v)$. Assume that $u_2 = A_2 w_3$. We repeat this process $k-1$ times. Note that $u_k = A_k \in \mathbb{S}$. The step sequence $u = A_1 \dots A_k$ is maximally concurrent and $u \in \mathbf{u}$. $\square$

For the case of Example 8 the step sequence $\{b\}\{a, c\}$ is maximally concurrent and for the case of Example 9 the step sequence $\{a\}\{b, c, d, e\}$ is maximally concurrent. There may be more than one maximally concurrent step sequences in a g-comtrace. For example if $E = \{a, b\}$, $sim = ser = \emptyset$, $inl = \{(a, b), (b, a)\}$, then the g-comtrace $t = [\{a\}\{b\}] = \{\{a\}\{b\}, \{b\}\{a\}\}$ and both $\{a\}\{b\}$ and $\{b\}\{a\}$ are maximally concurrent.

Having a canonical (unique) representation is often useful in proving properties about g-comtraces since it allows us to *uniquely identify* a g-comtrace. Furthermore, to be really useful in proofs, a canonical representation should be easy to construct and manipulate. For g-comtraces, it turns out that a natural way to get a canonical representation is: fix a total order on the alphabet, extend it to a lexicographical ordering on step sequences, and then simply choose the lexicographically least element.

**Definition 15** *(Lexicographical ordering).* Assume that we have a *total order* $<_E$ on $E$.

1. We define a *step order* $<^{st}$ on $\mathbb{S}$ as follows:

$$A <^{st} B \quad \stackrel{df}{\Longleftrightarrow} \quad |A| > |B| \vee \Big(|A| = |B| \wedge A \neq B \wedge \min_{<_E}(A \setminus B) <_E \min_{<_E}(B \setminus A)\Big),$$

   where $\min_{<_E}(X)$ denotes the least element of the set $X \subseteq E$ w.r.t. $<_E$.
2. Let $A_1 \dots A_n$ and $B_1 \dots B_m$ be two sequences in $\mathbb{S}^*$. We define a *lexicographical order* $<^{lex}$ on step sequences in a natural way as the lexicographical order induced by $<^{st}$, i.e.,

$$A_1 \dots A_n <^{lex} B_1 \dots B_m \quad \stackrel{df}{\Longleftrightarrow} \quad \exists k > 0 \quad \forall i < k. \big(A_i = B_i \wedge (A_k <^{st} B_k \vee n < k \leqslant m)\big).$$

Directly from the above definition, it follows that $<^{st}$ totally orders the set of possible steps $\mathbb{S}$ and $<^{lex}$ totally orders the set of possible step sequences $\mathbb{S}^*$.

**Example 10.** Assume that $a <_E b <_E c <_E d <_E e$. Then we have $\{a, b, c, e\} <^{st} \{b, c, d\}$ since $\{a, b, c, e\} \setminus \{b, c, d\} = \{a\}$, $\{b, c, d\} \setminus \{a, b, c, e\} = \{d\}$, and $a <_E d$. And $\{a, c\}\{b, c\}\{d\}\{d, c\} <^{lex} \{a, c\}\{b\}\{c, d, e\}$ since $|\{b, c\}| > |\{b\}|$.

**Definition 16** *(g-canonical step sequence).* A step sequence $x \in \mathbb{S}^*$ is *g-canonical* if for every step sequence $y \in \mathbb{S}^*$, we have $(x \equiv y \wedge x \neq y) \Longrightarrow x <^{lex} y$.

In other words, $x$ is g-canonical if it is the least element in the g-comtrace $[x]$ with respect to the lexicographical ordering $<^{lex}$.

**Corollary 3.**

1. *Each g-canonical step sequence is in GMC-form.*
2. *For every step sequence* $x \in \mathbb{S}^*$*, there exists a unique g-canonical sequence* $u \equiv x$*.* $\square$

All of the concepts and results discussed so far in this section hold also for general equational monoids derived from the step sequence monoid (like those considered in [15]). We will now show that for both comtraces and traces, the GMC-form, MC-form and g-canonical form correspond to the canonical form discussed in [2,3,11,15].

### 8.2. Canonical representations of comtraces

First note that comtraces are just g-comtraces with an empty relation *inl*, so all definitions for g-comtraces also hold for comtraces.

Let $\theta = (E, sim, ser)$ be a comtrace alphabet (i.e. $inl = \emptyset$) and $\mathbb{S}$ be the set of all steps over $\theta$. In principle, $(a,b) \in ser$ means that the sequence $\{a\}\{b\}$ can be replaced by the set $\{a,b\}$ (and vice versa). We start with the definition of a relation between steps that allows such replacement.

**Definition 17** *(Forward dependency).* Let $\mathbb{FD} \subseteq \mathbb{S} \times \mathbb{S}$ be a relation comprising all pairs of steps $(A, B)$ such that there exists a step $C \in \mathbb{S}$ such that

$$C \subseteq B \wedge A \times C \subseteq ser \wedge C \times (B \setminus C) \subseteq ser.$$

The relation $\mathbb{FD}$ is called *forward dependency* on steps.

Note that in this definition $C \in \mathbb{S}$ implies $C \neq \emptyset$, but $C = B$ is allowed. The next result explains the name "forward dependency" of $\mathbb{FD}$. If $(A, B) \in \mathbb{FD}$, then some elements from $B$ can be moved to $A$ and the outcome will still be equivalent to $AB$.

**Lemma 2.** $(A, B) \in \mathbb{FD} \iff (\exists C \in \wp^{\setminus\{\emptyset\}}(B).(A \cup C)(B \setminus C) \equiv AB) \vee A \cup B \equiv AB.$

**Proof.** ($\Rightarrow$) If $C = B$ then $A \cup B \approx AB$ which implies $A \cup B \equiv AB$. If $C \subset B$ and $C \neq \emptyset$ then we have $(A \cup C)(B \setminus C) \approx AC(B \setminus C) \approx AB$, i.e. $(A \cup C)(B \setminus C) \equiv AB$.

($\Leftarrow$) Assume $A \cup B \equiv AB$. This means $A \cup B \in \mathbb{S}$ and consequently $A \cap B = \emptyset$, $A \times B \subseteq ser$. Let $a \in A$, $b \in B$. By Proposition 8(6), $\{a,b\} = \pi_{\{a,b\}}(A \cup B) \equiv \pi_{\{a,b\}}(AB) = \{a\}\{b\}$. But $\{a,b\} \equiv \{a\}\{b\}$ means $(a,b) \in ser$. Therefore $A \times B \subseteq ser$, i.e. $(A, B) \in \mathbb{FD}$.

Assume $C \subset B$, $C \neq \emptyset$ and $(A \cup C)(B \setminus C) \equiv AB$. This implies $A \cup C \in \mathbb{S}$ and $A \cap C = \emptyset$. Let $a \in A$ and $c \in C$. By Proposition 8(6), $\{a,c\} = \pi_{\{a,c\}}(A \cup C)(B \setminus C) \equiv \pi_{\{a,c\}}(AB) = \{a\}\{c\}$. But $\{a,c\} \equiv \{a\}\{c\}$ means $(a,c) \in ser$. Hence $A \times C \subseteq ser$. Let $b \in B \setminus C$ and $c \in C$. By Proposition 8(6), $\{c\}\{b\} = \pi_{\{b,c\}}(A \cup C)(B \setminus C) \equiv \pi_{\{b,c\}}(AB) = \{b,c\}$. Thus $\{c\}\{b\} \equiv \{b,c\}$, which means $(c,b) \in ser$, i.e. $C \times (B \setminus C) \subseteq ser$. Hence $(A, B) \in \mathbb{FD}$. $\square$

We will now recall the definition of a canonical step sequence for comtraces.

**Definition 18** *(Comtrace canonical step sequence).* (See [11].) A step sequence $u = A_1 \ldots A_k$ is *canonical* if we have $(A_i, A_{i+1}) \notin \mathbb{FD}$ for all $i$, $1 \leqslant i < k$.

The next results show that a canonical step sequence for comtraces is in fact "greedy".

**Lemma 3.** *For each non-empty canonical step sequence $u = A_1 \ldots A_k$, we have*

$$A_1 = \left\{a \;\middle|\; \exists w \in [u].w = C_1 \ldots C_m \wedge a \in C_1\right\}.$$

**Proof.** Let $A = \{a \mid \exists w \in [u].w = C_1 \ldots C_m \wedge a \in C_1\}$. Since $u \in [u]$, $A_1 \subseteq A$. We need to prove that $A \subseteq A_1$. Definitely $A = A_1$ if $k = 1$, so assume $k > 1$. Suppose that $a \in A \setminus A_1$, $a \in A_j$, $1 < j \leqslant k$, and $a \notin A_i$ for $i < j$. Since $a \in A$, there is $v = Bx \in [u]$ such that $a \in B$. Note that $A_{j-1}A_j$ is also *canonical* and $u' = A_{j-1}A_j = (u \div_R (A_{j+1} \ldots A_k)) \div_L (A_1 \ldots A_{j-2})$. Let $v' = (v \div_R (A_{j+1} \ldots A_k)) \div_L (A_1 \ldots A_{j-2})$. We have $v' = B'x'$ where $a \in B'$. By Corollary 1, $u' \equiv v'$. Since $u' = A_{j-1}A_j$ is canonical then $\exists c \in A_{j-1}.(c,a) \notin ser$ or $\exists b \in A_j.(a,b) \notin ser$.

- For the former case: $\pi_{\{a,c\}}(u') = \{c\}\{a\}$ (if $c \notin A_j$) or $\pi_{\{a,c\}}(u') = \{c\}\{a,c\}$ (if $c \in A_j$). If $\pi_{\{a,c\}}(u') = \{c\}\{a\}$ then $\pi_{\{a,c\}}(v')$ equals either $\{a,c\}$ (if $c \in B'$) or $\{a\}\{c\}$ (if $c \notin B'$), i.e., in both cases $\pi_{\{a,c\}}(u') \not\equiv \pi_{\{a,c\}}(v')$, contradicting Proposition 8(6). If $\pi_{\{a,c\}}(u') = \{c\}\{a,c\}$ then $\pi_{\{a,c\}}(v')$ equals either $\{a,c\}\{c\}$ (if $c \in B'$) or $\{a\}\{c\}\{c\}$ (if $c \notin B'$). However in both cases $\pi_{\{a,c\}}(u') \not\equiv \pi_{\{a,c\}}(v')$, contradicting Proposition 8(6). For the latter case, let $d \in A_{j-1}$. Then $\pi_{\{a,b,d\}}(u') = \{d\}\{a,b\}$ (if $d \notin A_j$), or $\pi_{\{a,b,d\}}(u') = \{d\}\{a,b,d\}$ (if $d \in A_j$). If $\pi_{\{a,b,d\}}(u') = \{d\}\{a,b\}$ then $\pi_{\{a,b,d\}}(v')$ is one of the following $\{a,b,d\}$, $\{a,b\}\{d\}$, $\{a,d\}\{b\}$, $\{a\}\{b\}\{d\}$ or $\{a\}\{d\}\{b\}$, and in either case $\pi_{\{a,b,d\}}(u') \not\equiv \pi_{\{a,b,d\}}(v')$, again contradicting Proposition 8(6).
- If $\pi_{\{a,b,d\}}(u') = \{d\}\{a,b,d\}$, then we know $\pi_{\{a,b,d\}}(v')$ is one of the following $\{a,b,d\}\{d\}$, $\{a,b\}\{d\}\{d\}$, $\{a,d\}\{b,d\}$, $\{a,d\}\{b\}\{d\}$, $\{a,d\}\{d\}\{b\}$, $\{a\}\{b\}\{d\}\{d\}$, $\{a\}\{d\}\{b\}\{d\}$, or $\{a\}\{d\}\{d\}\{b\}$. However in any of these cases we have $\pi_{\{a,b,d\}}(u') \not\equiv \pi_{\{a,b,d\}}(v')$, contradicting Proposition 8(6) as well. $\square$

We will now show that for comtraces the canonical form from Definition 18 and GMC-form are equivalent, and that each comtrace has a unique canonical representation.

**Theorem 6.** *A step sequence u is in GMC-form if and only if it is canonical.*

**Proof.** ($\Leftarrow$) Suppose that $u = A_1 \ldots A_k$ is canonical. By Lemma 3 we have that for each $B_1 y_1 \equiv A_1 \ldots A_k$, $|B_1| \leqslant |A_1|$. Since each $A_i \ldots A_k$ is also canonical, $A_2 \ldots A_k$ is canonical so by Lemma 3 again we have that for each $B_2 y_2 \equiv A_2 \ldots A_k$, $|B_2| \leqslant |A_2|$. And so on, i.e. $u = A_1 \ldots A_k$ is in GMC-form.

($\Rightarrow$) Suppose that $u = A_1 \ldots A_k$ is not canonical, and $j$ is the smallest number such that $(A_j, A_{j+1}) \in \mathbb{FD}$. Hence $A_1 \ldots A_{j-1}$ is canonical, and, by ($\Leftarrow$) of this theorem, in GMC-form. By Lemma 2, either there is a non-empty $C \subset A_{j+1}$ such that $(A_j \cup C)(A_{j+1} \setminus B) \equiv A_j A_{j+1}$, or $A_j \cup A_{j+1} \equiv A_j A_{j+1}$. In the first case since $C \neq \emptyset$, $|A_j \cup C| > |A_j|$; in the second case $|A_j \cup A_{j+1}| > |A_j|$, so $A_j \ldots A_k$ is not in GMC-form, which means $u = A_1 \ldots A_k$ is not in GMC-form either. □

**Theorem 7.** *(Implicit in [11].) For each step sequence v there is a unique canonical step sequence u such that $v \equiv u$.*

**Proof.** The existence follows from Proposition 11 and Theorem 6. We only need to show uniqueness. Suppose that $u = A_1 \ldots A_k$ and $v = B_1 \ldots B_m$ are both canonical step sequences and $u \equiv v$. By induction on $k = |u|$ we will show that $u = v$. By Lemma 3, we have $B_1 = A_1$. If $k = 1$, this ends the proof. Otherwise, let $u' = A_2 \ldots A_k$ and $w' = B_2 \ldots B_m$ and $u', v'$ are both canonical step sequences of $[u']$. Since $|u'| < |u|$, by the induction hypothesis, we obtain $A_i = B_i$ for $i = 2, \ldots, k$ and $k = m$. □

The result of Theorem 7 was not stated explicitly in [11], but it can be derived from the results of Propositions 3.1, 4.8 and 4.9 of [11]. However Propositions 3.1 and 4.8 of [11] involve the concepts of partial orders and stratified order structures, while the proof of Theorem 7 uses only the algebraic properties of step sequences and comtraces.

Immediately from Theorems 6 and 7 we get the following result.

**Corollary 4.** *A step sequence u is canonical if and only if it is g-canonical.* □

It turns out that for comtraces the canonical representation and MC representation are also equivalent.

**Lemma 4.** *If a step sequence u is canonical and $u \equiv v$, then $length(u) \leqslant length(v)$.*

**Proof.** By induction on $length(v)$. Obvious for $length(v) = 1$ as then $u = v$. Assume it is true for all $v$ such that $length(v) \leqslant r - 1$, $r \geqslant 2$. Consider $v = B_1 B_2 \ldots B_r$ and let $u = A_1 A_2 \ldots A_k$ be a canonical step sequence such that $v \equiv u$. Let $v_1 = v \div_L A_1 = C_1 \ldots C_s$. By Corollary 1(2), $v_1 \equiv u \div_L A_1 = A_2 \ldots A_k$, and $A_2 \ldots A_k$ is clearly canonical. Hence by induction assumption $k - 1 = length(A_2 \ldots A_k) \leqslant s$. By Lemma 3, $B_1 \subseteq A_1$, hence $v_1 = v \div_L A_1 = B_2 \ldots B_r \div_L A_1 = C_1 \ldots C_s$, which means $s \leqslant r - 1$. Therefore $k - 1 \leqslant s \leqslant r - 1$, i.e. $k \leqslant r$, which ends the proof. □

**Theorem 8.** *A step sequence u is maximally concurrent if and only if it is canonical.*

**Proof.** ($\Leftarrow$) Let $u$ be canonical. From Lemma 4 it follows the condition (1) of Definition 14 is satisfied. By Theorem 6, $u$ is in GMC-form, so the condition (2) of Definition 14 is satisfied as well.

($\Rightarrow$) By induction on $length(u)$. It is obviously true for $u = A_1$. Suppose it is true for $length(u) = k$. Let $u = A_1 A_2 \ldots A_k A_{k+1}$ be maximally concurrent. The step sequence $A_2 \ldots A_{k+1}$ is also maximally concurrent and canonical by the induction assumption. If $A_1 A_2 \ldots A_{k+1}$ is not canonical, then $(A_1, A_2) \in \mathbb{FD}$. By Lemma 2, either there is non-empty $C \subset B$ such that $(A_1 \cup C)(A_2 \setminus C) \equiv A_1 A_2$, or $A_1 \cup A_2 \equiv A_1 B_2$. Hence either $(A_1 \cup C)(A_2 \setminus C) A_3 \ldots A_{k+1} \equiv A_1 \ldots A_{k+1} = u$ or $(A \cup A_2) A_3 \ldots A_{k+1} \equiv A_1 \ldots A_{k+1} = u$. The former contradicts the condition (2) of Definition 14, the latter one contradicts the condition (1) of Definition 14, so $u$ is not maximally concurrent, which means $(A_1, A_2) \notin \mathbb{FD}$, so $u = A_1 \ldots A_{k+1}$ is canonical. □

Summing up, as far as canonical representation is concerned, comtraces behave quite nicely. All three forms for g-comtraces, GMC-form, MC-form and g-canonical form, collapse to one comtrace canonical form if $inl = \emptyset$.

### 8.3. Canonical representations of traces

We will show that the canonical representations of traces are conceptually the same as the canonical representations of comtraces. The differences are merely “syntactical”, as traces are sets of sequences, so “maximal concurrency” cannot be expressed explicitly, while comtraces are sets of step sequences.

Let $(E, ind)$ be a trace alphabet and $(E^*/\equiv, \circledast, [\lambda])$ be the corresponding monoid of traces. A sequence $x = a_1 \ldots a_k \in E^*$ is called *fully commutative* if $(a_i, a_j) \in ind$ for all $i \neq j$ and $i, j \in \{1, \ldots, k\}$.

**Corollary 5.** *If $x = a_1 \ldots a_k \in E^*$ is fully commutative and $y = a_{i_1} \ldots a_{i_k}$ is any permutation of $a_1 \ldots a_k$, then $x \equiv y$.* □

The above corollary could be interpreted as saying that if $x = a_1 \dots a_k \in E^*$ is fully commutative than the set of events $\{a_1, \dots, a_k\}$ can be executed simultaneously.

**Definition 19** *(Greedy maximally concurrent form for traces).* (See [2,3].) A sequence $x \in E^*$ is in *greedy maximally concurrent form* (*GMC-form*) if $x = \lambda$ or $x = x_1 \dots x_n$ such that

1. each $x_i$ is fully commutative, for $i = 1, \dots, n$,
2. for each $1 \leqslant i \leqslant n-1$ and for each element $a$ of $x_{i+1}$ there exists an element $b$ of $x_i$ such that $(a, b) \notin ind$.

Often the form from the above definition is called "canonical" [3,14,15].

**Theorem 9.** *(See [2,3].) For every trace $\mathbf{t} \in E^*/\!\equiv$, there exists $x \in E^*$ such that $\mathbf{t} = [x]$ and $x$ is in the GMC-form.* $\square$

The GMC-form as defined above is not unique, a trace may have more than one GMC representation. For instance the trace $\mathbf{t}_1 = [abcbca]$ from Example 2 has four GMC representations: *abcbca*, *acbbca*, *abccba*, and *acbcba*. The GMC-form is however unique when traces are represented as *vector firing sequences*[6] [3,14,27], where each fully commutative sequence is represented by a unique vector of events (so the name "canonical" used in [3,14] is justified). To get uniqueness for Mazurkiewicz traces, it suffices to order fully commutative sequences. For example, we may introduce an arbitrary total order on $E$, extend it lexicographically to $E^*$ and add the condition that in the representation $x = x_1 \dots x_n$, each $x_i$ is minimal w.r.t. the lexicographic ordering. The GMC-form with this additional condition is called *Foata canonical form*.

**Theorem 10.** *(See [2].) Every trace has a unique representation in the Foata canonical form.* $\square$

We will now show the relationship between GMC-form for traces and GMC-form (or canonical form) for comtraces.

Define $\mathbb{S}$, the set of steps generated by $(E, ind)$ as the set of all cliques of the graph of the relation *ind*, and for each fully commutative sequence $x = a_1 \dots a_n$, let $\mathrm{st}(x) = \{a_1, \dots, a_n\} \in \mathbb{S}$ be the step generated by $x$.

For each sequence $x = x_1 \dots x_k$ in GMC-form in $(E, ind)$, we call the step sequence $x^{\{\max\}} = \mathrm{st}(x_1) \dots \mathrm{st}(x_k) \in \mathbb{S}^*$, the *maximally concurrent step sequence representation* of $x$. Note that by Theorem 10, the step sequence $x^{\{\max\}}$ is unique. The name is formally justified by the following result (which also follows implicitly from [3]).

**Proposition 12.**

1. *A sequence $x = x_1 \dots x_n$ is in GMC-form in $(E, ind)$ if and only if the step sequence $x^{\{\max\}} = \mathrm{st}(x_1) \dots \mathrm{st}(x_k)$ is in GMC-form (or canonical form) in $(E, sim, ser)$ where $sim = ser = ind$.*
2. $[x]_{\equiv_{ind}} \overset{\mathsf{t} \leftrightsquigarrow \mathsf{c}}{\equiv} [x^{\{\max\}}]_{\equiv_{ser}}$.

**Proof.** 1. If $x = x_1 \dots x_n$ is not in GMC-form then by (2) of Definition 19, there are $x_i, x_{i+1}$ and $b \in \mathrm{st}(x_{i+1})$ such that for all $a \in \mathrm{st}(x_i)$, $(a, b) \in ind$. Since $ser = ind$ this means that $(\mathrm{st}(x_1), \mathrm{st}(x_{i+1})) \in \mathbb{FD}$, so $x^{\{\max\}}$ is not canonical. Suppose that $x^{\{\max\}}$ is not canonical, i.e. $(\mathrm{st}(x_1), \mathrm{st}(x_{i+1})) \in \mathbb{FD}$ for some $i$. This means there is a non-empty $C \subseteq \mathrm{st}(x_{i+1})$ such that $\mathrm{st}(x_i) \times C \subseteq ser$ and $C \times (\mathrm{st}(x_{i+1}) \setminus C) \subseteq ser$. Let $a \in \mathrm{st}(x_i)$ and $b \in C \subseteq \mathrm{st}(x_{i+1})$. Since $ind = ser$, then $(a, b) \in ind$, so $x = x_1 \dots x_n$ is not in GMC-form.

2. Clearly $[x]_{\equiv_{ind}} \overset{\mathsf{t} \leftrightsquigarrow \mathsf{c}}{\equiv} [x^{\{\}}]_{\equiv_{ser}}$. Let $a_1 \dots a_n$ be a fully commutative sequence. Since $ser = ind$, $\{a_1\} \dots \{a_n\} \equiv_{ser} \{a_1, \dots, a_n\}$. Hence, for each sequence $x$, $x^{\{\}} \equiv_{ser} x^{\{\max\}}$, i.e. $[x^{\{\}}]_{\equiv_{ser}} = [x^{\{\max\}}]_{\equiv_{ser}}$. $\square$

Hence we have proved that the GMC-form (or canonical form) for comtraces and GMC-form for traces are semantically identical concepts. They both describe the greedy maximally concurrent semantics, which for both comtraces and traces is also the global maximally concurrent semantics.

## 9. Comtraces and stratified order structures

In this section we will recall the major result of [11] that shows how comtraces define appropriate so-structures. We will start with the definition of $\Diamond$-*closure* construction that plays a substantial role in most applications of so-structures for modelling concurrent systems (cf. [11,19,17,18]).

**Definition 20** *(Diamond closure of relational structures).* (See [11].) Given a relational structure $S = (X, R_1, R_2)$, we define $S^{\Diamond}$, the $\Diamond$-*closure* of $S$, as

---

[6] Vector firing sequences were introduced by Mike Shields in 1979 [27] as an alternative representation of Mazurkiewicz traces.

$$S^{\diamondsuit} \triangleq (X, \prec_{R_1,R_2}, \sqsubset_{R_1,R_2}),$$

where $\prec_{R_1,R_2} \triangleq (R_1 \cup R_2)^* \circ R_1 \circ (R_1 \cup R_2)^*$ and $\sqsubset_{R_1,R_2} \triangleq (R_1 \cup R_2)^* \setminus id_X$.

The motivation behind the above definition is the following. For 'reasonable' $R_1$ and $R_2$, the relational structure $(X, R_1, R_2)^{\diamondsuit}$ should satisfy the axioms S1–S4 of the so-structure definition. Intuitively, $\diamondsuit$-closure is a generalization of the transitive closure constructions for relations to so-structures. Note that if $R_1 = R_2$ then $(X, R_1, R_2)^{\diamondsuit} = (X, R_1^+, R_1^+)$. The following result shows that the properties of $\diamondsuit$-closure are close to the appropriate properties of transitive closure.

**Theorem 11** *(Closure properties of $\diamondsuit$-closure). (See [11].) For a relational structure $S = (X, R_1, R_2)$,*

1. *if $R_2$ is irreflexive, then $S \subseteq S^{\diamondsuit}$,*
2. $(S^{\diamondsuit})^{\diamondsuit} = S^{\diamondsuit}$,
3. *$S^{\diamondsuit}$ is a so-structure if and only if $\prec_{R_1,R_2} = (R_1 \cup R_2)^* \circ R_1 \circ (R_1 \cup R_2)^*$ is irreflexive,*
4. *if $S$ is a so-structure, then $S = S^{\diamondsuit}$.* □

Every comtrace is a set of equivalent step sequences and every step sequence represents a stratified order, so a comtrace can be interpreted as a set of equivalent stratified orders. From the theory presented in Section 4 and the fact that comtrace satisfies paradigm $\pi_3$, it follows that this set of orders should define a so-structure, which should be called a so-structure defined by a given comtrace. On the other hand, with respect to a comtrace alphabet, every comtrace can be uniquely generated from any step sequence it contains. Thus, we will show that given a step sequence $u$ over a comtrace alphabet, *without analyzing any other elements of the comtrace* $[u]$ *but $u$ itself*, we will be able to construct the same so-structure as the one defined by the whole comtrace. Formulations and proofs of such results are done in [11] and depend heavily on the $\diamondsuit$-closure construction and its properties.

Let $\theta = (E, sim, ser)$ be a comtrace alphabet, and let $u \in \mathbb{S}^*$ be a step sequence and let $\lhd_u \subseteq \Sigma_u \times \Sigma_u$ be the stratified order generated by $u$ as defined in Section 2.3. Note that if $u \equiv w$ then $\Sigma_u = \Sigma_w$. Thus, for every comtrace $\mathbf{x} = [u] \in \mathbb{S}^*/\equiv$, we can define $\Sigma_{\mathbf{x}} = \Sigma_u$.

We will now show how the $\diamondsuit$-closure operator is used to define a so-structure induced by a single step sequence $u$.

**Definition 21.** Let $u \in \mathbb{S}^*$. We define the relations $\prec_u, \sqsubset_u \subseteq \Sigma_{\mathbf{u}} \times \Sigma_{\mathbf{u}}$ as:

1. $\alpha \prec_u \beta \overset{df}{\iff} \alpha \lhd_u \beta \wedge (l(\alpha), l(\beta)) \notin ser$,
2. $\alpha \sqsubset_u \beta \overset{df}{\iff} \alpha \lhd_u^{\frown} \beta \wedge (l(\beta), l(\alpha)) \notin ser$.

**Lemma 5.** *(See [11, Lemma 4.7].) For all $u, v \in \mathbb{S}^*$, if $u \equiv v$, then $\prec_u = \prec_v$ and $\sqsubset_u = \sqsubset_v$.* □

Definition 21 together with Lemma 5 describes two basic *local* invariants of the elements of $\Sigma_{\mathbf{u}}$. The relation $\prec_u$ captures the situation when $\alpha$ *always precedes* $\beta$, and the relation $\sqsubset_u$ captures the situation when $\alpha$ *never follows* $\beta$.

**Definition 22.** Given a comtrace $\mathbf{u} = [u] \in \mathbb{S}^*/\equiv$. We define

$$S^{\{u\}} \triangleq (\Sigma_{\mathbf{u}}, \prec_u, \sqsubset_u)^{\diamondsuit}, \qquad S_{\mathbf{u}} \triangleq \left(\Sigma_{\mathbf{u}}, \bigcap_{x\in\mathbf{u}} \lhd_x, \bigcap_{x\in\mathbf{u}} \lhd_x^{\frown}\right).$$

The relational structure $S^{\{u\}}$ is the so-structure induced by the single step sequence $u$ and $S_{\mathbf{u}}$ is the so-structure defined by the comtrace $\mathbf{u}$. The following theorem justifies the names and summarizes some nontrivial results concerning the so-structures generated by comtraces.

**Theorem 12.** *(See [11,12].) For all $u, v \in \mathbb{S}^*$, we have*

1. *$S^{\{u\}}$ and $S_{[u]}$ are so-structures,*
2. $u \equiv v \iff S^{\{u\}} = S^{\{v\}}$,
3. $S^{\{u\}} = S_{[u]}$,
4. $ext(S_{[u]}) = \{\lhd_x \mid x \in [u]\}$. □

Theorem 12 states that the so-structures $S^{\{u\}}$ and $S_{[u]}$ from Definition 22 are identical and their stratified extensions are exactly the elements of the comtrace $[u]$ with step sequences interpreted as stratified orders. However, from an algorithmic point of view, the definition of $S^{\{u\}}$ is more interesting, since building the relations $\prec_u$ and $\sqsubset_u$ and getting their $\diamondsuit$-closure,

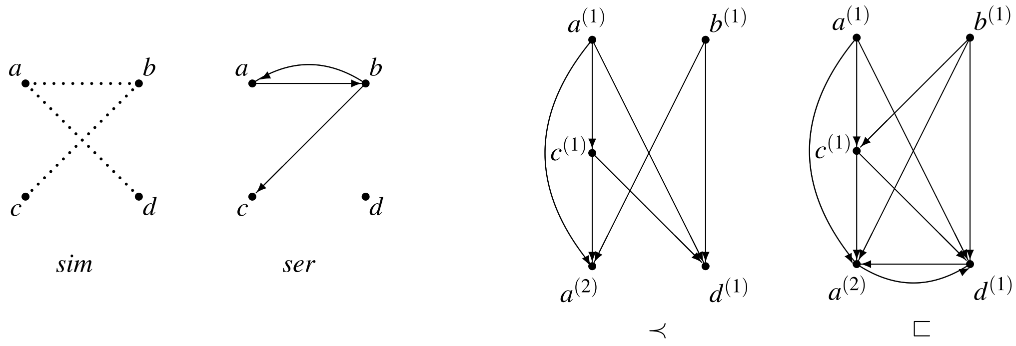



**Fig. 3.** An example of the relations *sim*, *ser* on $E = \{a, b, c, d\}$, and the so-structure $(X, \prec, \sqsubset)$ defined by the comtrace $[\{a, b\}\{c\}\{a, d\}]_{\equiv_{ser}} = \{\{a, b\}\{c\}\{a, d\}, \{a\}\{b\}\{c\}\{a, d\}, \{a\}\{b, c\}\{a, d\}, \{b\}\{a\}\{c\}\{a, d\}\}$.

which in turn can be reduced to computing transitive closure of relations, can be done efficiently. In contrast, a direct use of the $S_{[u]}$ definition requires precomputing up to exponentially many elements of the comtrace $[u]$.

Fig. 3 shows an example of a comtrace and the so-structure it generates.

## 10. Generalized stratified order structures generated by generalized comtraces

The relationship between g-comtraces and gso-structures is in principle the same as the relationship between comtraces and so-structures discussed in the previous section. Each g-comtrace uniquely determines a finite labeled gso-structure. However the formulations and proofs of these analogue results for g-comtraces are more complex. The difficulties are mainly due to the following facts:

- The definition of gso-structure is implicit, it involves using the induced so-structures (see Definition 5), which makes practically all definitions more complex (especially the counterpart of $\diamondsuit$-closure), and the use of Theorem 4 more difficult than the use of Theorem 2.
- The internal property expressed by Theorem 3, which says that $ext(S)$ conforms to paradigm $\pi_3$ of [10], does not hold for gso-structures.
- Generalized comtraces do not have a 'natural' canonical form with a well understood interpretation.
- The relation *inl* introduces plenty of irregularities and increases substantially the number of cases that need to be considered in many proofs.

In this section, we will prove the analogue of Theorem 12 showing that every g-comtrace uniquely determines a finite gso-structure.

### 10.1. Commutative closure of relational structures

We will start with the notion of *commutative closure* of a relational structure. It is an extension of the concept of $\diamondsuit$-closure (see Definition 20) which was used in [11] and the previous section to construct finite so-structures from single step sequences or stratified orders.

**Definition 23** *(Commutative closure).* Let $G = (X, R_1, R_2)$ be any relational structure, and let $R_3 = R_1 \cap R_2^*$. Using the notation from Definition 20, the *commutative closure* of the relational structure $G$ is defined as

$$G^{\bowtie} = \big(X, (\prec_{R_3R_2})^{\mathsf{sym}} \cup R_1, \sqsubset_{R_3R_2}\big).$$

The motivation behind the above definition is similar to that for $\diamondsuit$-closure: for 'reasonable' $R_1$ and $R_2$, $(X, R_1, R_2)^{\bowtie}$ should be a gso-structure. Intuitively the $\bowtie$-closure is also a generalization of transitive closure for relations. Note that if $R_1 = R_2$ then $(X, R_1, R_2)^{\bowtie} = (X, (R_1^+)^{\mathsf{sym}}, R_1^+)$. Since the definition of gso-structures involves the definition of so-structures (see Definition 5), the definition of $\bowtie$-closure uses the concept of $\diamondsuit$-closure.

Note that we do not have an equivalent of Theorem 11 for $\bowtie$-closure. The reason is that $\bowtie$-closure is tailored to simplify the proofs in the next section rather than to be a closure operator by itself. Nevertheless, $\bowtie$-closure satisfies some general properties of a closure operator.

The first property is the monotonicity of $\bowtie$-closure.

**Proposition 13.** *If $G_1 = (X, R_1, R_2)$ and $G_2 = (X, Q_1, Q_2)$ are two relational structures such that $G_1 \subseteq G_2$, then $G_1^{\bowtie} \subseteq G_2^{\bowtie}$.*

**Proof.** Since $R_1 \subseteq Q_1$ and $R_2 \subseteq Q_2$ then $R_3 \subseteq Q_3$, and $(X, R_3, R_2)^{\diamond} \subseteq (X, Q_3, Q_2)^{\diamond}$, i.e. $\prec_{R_3R_2} \subseteq \prec_{Q_3Q_2}$ and $\sqsubset_{R_3R_2} \subseteq \sqsubset_{Q_3Q_2}$, which immediately implies $G_1^{\bowtie} \subseteq G_2^{\bowtie}$. □

Another desirable property of $\bowtie$-closure is that gso-structures are fixed points of $\bowtie$.

**Proposition 14.** *If $G = (X, <\!\!>, \sqsubset)$ is a gso-structure then $G = G^{\bowtie}$.*

**Proof.** Since $G$ is a gso-structure, by Definition 5, $S_G = (X, \prec_G, \sqsubset)$ is a so-structure. Hence, by Theorem 11(4), $S_G = S_G^{\diamond}$, which implies $\sqsubset = (\prec_G \cup \sqsubset)^* \setminus id_X$. But since $S_G$ is a so-structure, $\prec_G \subseteq \sqsubset$. So $\sqsubset = \sqsubset^* \setminus id_X$. Let $\prec = <\!\!> \cap \sqsubset^*$. Then since $<\!\!>$ is irreflexive,

$$\prec = <\!\!> \cap \sqsubset^* = <\!\!> \cap (\sqsubset^* \setminus id_X) = <\!\!> \cap \sqsubset = \prec_G .$$

Hence, $(X, \prec, \sqsubset) = (X, \prec_G, \sqsubset)$ is a so-structure. By Theorem 11(4), we know $(X, \prec, \sqsubset) = (X, \prec, \sqsubset)^{\diamond}$. So from Definition 23, $G^{\bowtie} = (X, \prec^{\mathsf{sym}} \cup <\!\!>, \sqsubset)$. Since $<\!\!>$ is symmetric and $\prec \subseteq <\!\!>$, we have $\prec^{\mathsf{sym}} \cup <\!\!> = <\!\!>$. Thus, $G = G^{\bowtie}$. □

### *10.2. Generalized stratified order structure generated by a step sequence*

We will now introduce a construction that derives a gso-structure from a single step sequence over a given g-comtrace alphabet. The idea of the construction is the same as $S^{\{u\}}$ from the previous section. First we construct some relational invariants and next we will use $\bowtie$-closure in the similar manner as $\diamond$-closure was used for $S^{\{u\}}$. However the construction is more elaborate and requires full use of the notation from Section 2.3 that allows us to define the formal relationship between step sequences and (labeled) stratified orders. We will also need the following two useful operators for relations.

**Definition 24.** Let $R$ be a binary relation on $X$. We define the

- *symmetric intersection* of $R$ as $R^{\Cap} \triangleq R \cap R^{-1}$, and
- the *complement* of $R$ as $R^{\mathsf{C}} \triangleq (X \times X) \setminus R$.

Let $\Theta = (E, sim, ser, inl)$ be a g-comtrace alphabet. Note that if $u \equiv w$ then $\Sigma_u = \Sigma_w$ so for every g-comtrace $\mathbf{s} = [s] \in \mathbb{S}^*/\!\equiv$, we can define $\Sigma_{\mathbf{s}} = \Sigma_s$.

**Definition 25.** Given a step sequence $s \in \mathbb{S}^*$.

1. Let the relations $<\!\!>_s, \sqsubset_s, \prec_s \subseteq \Sigma_{\mathbf{s}} \times \Sigma_{\mathbf{s}}$ be defined as follows:

$$\alpha <\!\!>_s \beta \quad \overset{df}{\iff} \quad \big(l(\alpha), l(\beta)\big) \in inl, \tag{10.1}$$

$$\alpha \sqsubset_s \beta \quad \overset{df}{\iff} \quad \alpha \lhd_s^{\frown} \beta \wedge \big(l(\beta), l(\alpha)\big) \notin ser \cup inl, \tag{10.2}$$

$$\alpha \prec_s \beta \quad \overset{df}{\iff} \quad \alpha \lhd_s \beta \wedge \left( \begin{array}{l} (l(\alpha), l(\beta)) \notin ser \cup inl \\ \quad \vee (\alpha, \beta) \in <\!\!>_s \cap ((\sqsubset_s^*)^{\Cap} \circ <\!\!>_s^{\mathsf{C}} \circ (\sqsubset_s^*)^{\Cap}) \\ \quad \vee \left( \begin{array}{l} (l(\alpha), l(\beta)) \in ser \\ \quad \wedge \exists \delta, \gamma \in \Sigma_s . \left( \begin{array}{l} \delta \lhd_s \gamma \wedge (l(\delta), l(\gamma)) \notin ser \\ \quad \wedge \alpha \sqsubset_s^* \delta \sqsubset_s^* \beta \wedge \alpha \sqsubset_s^* \gamma \sqsubset_s^* \beta \end{array} \right) \end{array} \right) \end{array} \right). \tag{10.3}$$

2. The triple

$$G^{\{s\}} \triangleq (\Sigma_s, \prec_s \cup <\!\!>_s, \prec_s \cup \sqsubset_s)^{\bowtie}$$

is called the *relational structure induced by the step sequence s*.

The intuition of Definition 25 is similar to that of Definition 21. Given a step sequence $s$ and g-comtrace alphabet $(E, sim, ser, inl)$, without analyzing any other elements of $[s]$ except $s$ itself, we would like to construct the gso-structure that is defined by the whole g-comtrace. So we will define appropriate "local" invariants $<\!\!>_s$, $\sqsubset_s$ and $\prec_s$ from the sequence $s$.

(a) Eq. (10.1) is used to construct the relationship $<\!\!>_s$, where two event occurrences $\alpha$ and $\beta$ might possibly be commutative because they are related by the *inl* relation.
(b) Eq. (10.2) defines the not later than relationship and this happens when $\alpha$ occurs not later than $\beta$ on the step sequence $s$ and $\{\alpha, \beta\}$ cannot be serialized into $\{\beta\}\{\alpha\}$, and $\alpha$ and $\beta$ are not commutative.
(c) Eq. (10.3) is the most complicated one, since we want to take into consideration the "earlier than" relationships which are not taken care of by the commutative closure. There are three such cases:

(i) $\alpha$ occurs before $\beta$ on the step sequence $s$, and two event occurrences $\alpha$ and $\beta$ cannot be put together into a single step ($(\alpha,\beta) \notin ser$) and are not commutative ($(\alpha,\beta) \notin inl$).
(ii) $\alpha$ and $\beta$ are supposed to be commutative but they cannot be commuted into $\beta$ and $\alpha$ because $\alpha$ is "synchronous" with some $\gamma$ and $\beta$ is "synchronous" with some $\delta$, and $(\gamma,\delta)$ is not in *inl* ("synchronous" in a sense that they must happen simultaneously).
(iii) $(\alpha,\beta)$ is in *ser* but they can never be put together into a single step because there are two distinct event occurrences $\delta$ and $\gamma$ which are "squeezed" between $\alpha$ and $\beta$ such that $(\delta,\gamma) \notin ser$, and thus $\delta$ and $\gamma$ can never be put together into a single step.

After building all of these "local" invariants from the step sequence $s$, all other "global" invariants which can be inferred from the axioms of the gso-structure definition are fully constructed by the commutative closure.

The next lemma will show that the relations from $G^{\{s\}}$ really correspond to positional invariants of all the step sequences from the g-comtrace $[s]$.

**Lemma 6.** *Let $s \in \mathbb{S}^*$, $G^{\{s\}} = (\Sigma_s, \Diamond, \sqsubset)$, and $\prec = \Diamond \cap \sqsubset$. If $\alpha,\beta \in \Sigma_{\mathbf{s}}$, then*

1. $\alpha \Diamond \beta \iff \forall u \in [s].pos_u(\alpha) \neq pos_u(\beta)$,
2. $\alpha \sqsubset \beta \iff \alpha \neq \beta \wedge \forall u \in [s].pos_u(\alpha) \leqslant pos_u(\beta)$,
3. $\alpha \prec \beta \iff \forall u \in [s].pos_u(\alpha) < pos_u(\beta)$,
4. *if $l(\alpha) = l(\beta)$ and $pos_s(\alpha) < pos_s(\beta)$, then $\alpha \prec \beta$.*

Eventhough the results of the above lemma are expected and look deceptively simple, the proof is long and highly technical and can be found in Appendix A.

Note that Lemma 6 also implies that we can construct the relational structure induced by the step sequence $G^{\{s\}}$ (we cannot claim that it is a gso-structure right now) if all the step sequences of a g-comtrace are known. We will first show how to define the gso-structure induced from all the positional invariants of all the step sequences of a g-comtrace.

**Definition 26.** For every $\mathbf{s} \in \mathbb{S}^*/\equiv$, we define $G_{\mathbf{s}} = (\Sigma_{\mathbf{s}}, \bigcap_{u\in\mathbf{s}} \lhd_u^{\mathsf{sym}}, \bigcap_{u\in\mathbf{s}} \lhd_u^{\frown})$.

Note that Theorem 4 *does not* immediately imply that $G_{\mathbf{s}}$ is a gso-structure. It needs to be proved separately.

We will now show that given a step sequence $s$ over a g-comtrace alphabet, the definition of $G^{\{s\}}$ and the definition of $G_{[s]}$ yield exactly the same gso-structure.

**Theorem 13.** *Let $s \in \mathbb{S}^*$. Then $G^{\{s\}} = G_{[s]}$.*

**Proof.** Let $G^{\{s\}} = (\Sigma_s, \Diamond, \sqsubset)$ and $\alpha,\beta \in \Sigma_s$. Then by Lemma 6(1, 2), we have

$$\alpha \Diamond \beta \iff \forall u \in [s].pos_u(\alpha) \neq pos_u(\beta) \iff (\alpha,\beta) \in \bigcap_{u\in[s]} \lhd_u^{\mathsf{sym}},$$

$$\alpha \sqsubset \beta \iff \big(\alpha \neq \beta \wedge \forall u \in [s].pos_u(\alpha) \leqslant pos_u(\beta)\big) \iff (\alpha,\beta) \in \bigcap_{u\in[s]} \big(\lhd_u^{\frown}\big)^{\mathsf{sym}}.$$

Hence, $G^{\{s\}} = (\Sigma_s, \Diamond, \sqsubset) = (\Sigma_s, \bigcap_{u\in[s]} \lhd_u^{\mathsf{sym}}, \bigcap_{u\in[s]} \lhd_u^{\frown}) = G_{[s]}$. $\square$

We will next show that $G^{\{s\}}$ is indeed a gso-structure.

**Theorem 14.** *Let $s \in \mathbb{S}^*$. Then $G^{\{s\}} = (\Sigma_s, \Diamond, \sqsubset)$ is a gso-structure.*

**Proof.** Since $\Diamond = \bigcap_{u\in[s]} \lhd_u^{\mathsf{sym}}$ and $\lhd_u^{\mathsf{sym}}$ is irreflexive and symmetric, $\Diamond$ is irreflexive and symmetric. Since $\sqsubset = \bigcap_{u\in[s]} \lhd_u^{\frown}$ and $\lhd_u^{\frown}$ is irreflexive, $\sqsubset$ is irreflexive.

Let $\prec = \Diamond \cap \sqsubset$, it remains to show that $S = (\Sigma, \prec, \sqsubset)$ satisfies the conditions S1–S4 of Definition 3. Since $\sqsubset$ is irreflexive, S1 is satisfied. Since $\prec \subseteq \sqsubset$, S2 is satisfied. Assume $\alpha \sqsubset \beta \sqsubset \gamma$ and $\alpha \neq \gamma$. Then

$$\begin{aligned}
&\alpha \sqsubset \beta \sqsubset \gamma \wedge \alpha \neq \gamma \\
&\implies (\alpha,\beta) \in \bigcap_{u\in[s]} \lhd_u^{\frown} \wedge (\beta,\gamma) \in \bigcap_{u\in[s]} \lhd_u^{\frown} \wedge \alpha \neq \gamma && \langle \text{Theorem 13} \rangle \\
&\implies \forall u \in [s].pos_u(\alpha) \leqslant pos_u(\beta) \leqslant pos_u(\gamma) \wedge \alpha \neq \gamma && \langle \text{Definition of } \lhd_u \rangle \\
&\implies \alpha \sqsubset \gamma && \big\langle \text{Lemma 6(2)} \big\rangle.
\end{aligned}$$

Hence, S3 is satisfied. Next we assume that $\alpha \prec \beta \sqsubset_s \gamma$. Then

$$\begin{aligned}
&\alpha \prec \beta \sqsubset \gamma \\
&\implies (\alpha,\beta) \in \bigcap_{u\in[s]} \left(\lhd_u^{\frown} \cap \lhd_u^{\mathsf{sym}}\right) \wedge (\beta,\gamma) \in \bigcap_{u\in[s]} \left(\lhd_u^{\frown} \cap \lhd_u^{\mathsf{sym}}\right) && \langle\text{Theorem 13}\rangle \\
&\implies \left(\forall u \in [s].pos_u(\alpha) \leqslant pos_u(\beta) \wedge pos_u(\alpha) \neq pos_u(\beta)\right) \\
&\qquad \wedge \left(\forall u \in [s].pos_u(\beta) \leqslant pos_u(\gamma) \wedge pos_u(\beta) \neq pos_u(\gamma)\right) && \langle\text{Definition of } \lhd_u\rangle \\
&\implies \forall u \in [s].pos_u(\alpha) < pos_u(\gamma) \\
&\implies \alpha \prec \gamma && \langle\text{Lemma 6(3)}\rangle.
\end{aligned}$$

Similarly, we can show $\alpha \sqsubset \beta \prec \gamma \implies \alpha \prec \gamma$. Thus, S4 is satisfied. $\square$

Theorem 14 justifies the following definition.

**Definition 27.** For every step sequence $s$, $G^{\{s\}} = (\Sigma_s, \prec_s \cup <\!>_s, \prec_s \cup \sqsubset_s)^{\bowtie}$ is the *gso-structure induced by s*.

At this point it is worth discussing the roles of the two different definitions of the gso-structures generated from a given g-comtrace. Definition 25 allows us to build the gso-structure by looking at a single step sequence of the g-comtrace and its g-comtrace alphabet. On the other hand, to build the gso-structure from a g-comtrace using Definition 26, we need to know either all the positional invariants or all elements of the g-comtrace. By Theorem 13, these two definitions are equivalent. However, in our proof, Definition 25 is more convenient when we want to deduce the properties of the gso-structure defined from a single step sequence over a given g-comtrace alphabet. On the other hand, Definition 26 will be used to reconstruct the gso-structure when positional invariants of a g-comtrace are known.

### *10.3. Generalized stratified order structures generated by generalized comtraces*

In this section, we want to show that the construction from Definition 25 indeed yields a gso-structure representation of comtraces. But before doing so, we need some preliminary results.

**Proposition 15.** *Let $s \in \mathbb{S}^*$. Then $\lhd_s \in ext(G^{\{s\}})$.*

**Proof.** Let $G^{\{s\}} = (\Sigma, <\!>, \sqsubset)$. By Lemma 6, for all $\alpha, \beta \in \Sigma$,

$$\begin{aligned}
&\alpha <\!> \beta \implies pos_s(\alpha) \neq pos_s(\beta) \implies \alpha \lhd_s \beta \vee \beta \lhd_s \alpha \implies \alpha \lhd_s^{\mathsf{sym}} \beta, \\
&\alpha \sqsubset \beta \implies pos_s(\alpha) \leqslant pos_s(\beta) \implies \alpha \lhd_s^{\frown} \beta.
\end{aligned}$$

Hence, by Definition 6, we get $\lhd_s \in ext(G^{\{s\}})$. $\square$

**Proposition 16.** *Let $s \in \mathbb{S}^*$. If $\lhd \in ext(G^{\{s\}})$, then there exists $u \in \mathbb{S}^*$ such that $\lhd = \lhd_u$.*

**Proof.** Let $G^{\{s\}} = (\Sigma_s, <\!>, \sqsubset)$ and $\Omega_\lhd = B_1 \ldots B_k$. We will show that $u = l[B_1] \ldots l[B_k]$ is a step sequence such that $\lhd = \lhd_u$.
Suppose $\alpha, \beta \in B_i$ are two distinct event occurrences such that $(l(\alpha), l(\beta)) \notin sim$. Then $pos_s(\alpha) \neq pos_s(\beta)$, which by Lemma 6 implies that $\alpha <\!> \beta$. Since $\lhd \in ext(G^{\{s\}})$, by Definition 6, $\alpha \lhd \beta$ or $\beta \lhd \alpha$ contradicting that $\alpha, \beta \in B_i$. Thus, we have shown for all $B_i$ $(1 \leqslant i \leqslant k)$,

$$\alpha, \beta \in B_i \wedge \alpha \neq \beta \implies \left(l(\alpha), l(\beta)\right) \in sim. \tag{10.4}$$

By Proposition A.1(2) (in Appendix A), if $e^{(i)}, e^{(j)} \in \Sigma_s$ and $i \neq j$ then $\forall u \in [s].pos_u(e^{(i)}) \neq pos_u(e^{(j)})$. So it follows from Lemma 6(1) that $e^{(i)} <\!> e^{(j)}$. Since $\lhd \in ext(G^{\{s\}})$, by Definition 6,

$$\text{if} \quad e^{(k_0)} \in B_k \quad \text{and} \quad e^{(m_0)} \in B_m, \quad \text{then} \quad k_0 \neq m_0 \iff k \neq m. \tag{10.5}$$

From (10.4) it follows that $u$ is a step sequence over $\theta$. Also by (10.5), $pos_u^{-1}[\{i\}] = B_i$ and $|l[B_i]| = |B_i|$ for all $i$. Hence, $\Omega_\lhd = \Omega_{\lhd_u}$, which implies $\lhd = \lhd_u$. $\square$

We want to show that two step sequences over the same g-comtrace alphabet induce the same gso-structure iff they belong to the same g-comtrace (Theorem 15 below). The proof of an analogous result for comtraces from [11] is simpler because every comtrace has a unique natural canonical representation that is both greedy and maximally concurrent and can be easily constructed. Moreover the canonical representation for comtraces correspond to the unique greedy stratified extension of appropriate causality relation $\prec$ (see [11]). Nothing similar holds for g-comtraces. For g-comtraces both natural representations, GMC and MC, are not unique. The g-canonical representation (Definition 16) is unique but its uniqueness is artificial and induced by some step sequence lexicographical order $<^{lex}$ (Definition 15). Nevertheless this lexicographical order $<^{lex}$ will be the basic tool used in the next lemma. The lack of natural unique representation will make our reasoning a bit harder.

**Lemma 7.** *Let $s$ be a step sequence over a g-comtrace alphabet $(E, ser, sim, inl)$ and $<_E$ be any total order on $E$. Let $u = A_1 \dots A_n$ be the g-canonical representation of $[s]$ (i.e., $u$ is the least element of the g-comtrace $[s]$ w.r.t. $<^{lex}$). Let $G^{\{s\}} = (\Sigma, \Leftrightarrow, \sqsubset)$ and $\prec = \Leftrightarrow \cap \sqsubset$. For each $X \subseteq \Sigma$, let $\mathsf{mins}_\prec(X)$ denote the set of all minimal elements of $X$ w.r.t. $\prec$ and define*

$$Z(X) \triangleq \big\{ Y \subseteq \mathsf{mins}_\prec(X) \;\big|\; \big(\forall \alpha, \beta \in Y. \neg(\alpha \Leftrightarrow \beta)\big) \wedge \big(\forall \alpha \in Y,\ \forall \beta \in X \setminus Y. \neg(\beta \sqsubset \alpha)\big) \big\}.$$

*Let $\bar{u} = \overline{A_1} \dots \overline{A_n}$ be the enumerated step sequence of $u$. Then $A_i$ is the least element of the set $\{l[Y] \mid Y \in Z(\Sigma \setminus \biguplus(\overline{A_1} \dots \overline{A_{i-1}}))\}$ w.r.t. the ordering $<^{st}$.*

Before presenting the proof, we will explain the intuition behind the definition of the set $Z(X)$. Let us consider $Z(\Sigma)$ first. Then $A_1$ in this lemma is the least element of the set $\{l[Y] \mid Y \in Z(\Sigma)\}$ w.r.t. the ordering $<^{st}$. Our goal is to construct $A_1$ by looking only at the gso-structure $G$ without having to construct up to exponentially many stratified extensions of $G$. The most technical part of this proof is to show that $\overline{A_1}$ actually belongs to the set $Z(\Sigma)$. Recall that to show that $Y \in Z(\Sigma)$, we want to show that $Y$ satisfies the following conditions:

(i) no two elements in $Y$ are commutative,
(ii) for an element $\alpha \in Y$ and $\beta \in \Sigma \setminus Y$, it is not the case that $\beta$ is not later than $\alpha$.

Note that we actually define $Z(X)$ instead of $Z(\Sigma)$, because we want to apply it successively to build *all* the steps $A_i$ of the g-canonical representation $u$ of $G^{\{s\}}$. This lemma can be seen as an algorithm to build the g-canonical representation of $[s]$ by looking only at $G^{\{s\}}$.

**Proof of Lemma 7.** First notice that by Lemma 6(3), for every non-empty $X \subseteq \Sigma$, since $\Sigma$ is finite, we know that $\mathsf{mins}_\prec(X)$ is non-empty and finite. Furthermore by Lemma 6(4), if $e^{(i)}, e^{(j)} \in \Sigma$ and $i < j$, then $e^{(i)} \prec e^{(j)}$. Hence, for all $\alpha, \beta \in \mathsf{mins}_\prec(X)$, where $X \subseteq \Sigma$, we have $l(\alpha) \neq l(\beta)$. This ensures that if $Y \in Z(X)$ and $X \subseteq \Sigma$, then $|Y| = |l[Y]|$.

For all $\alpha \in \overline{A_1}$ and $\beta \in \Sigma$, $pos_s(\beta) \geqslant pos_s(\alpha)$. Hence, by Lemma 6(3), $\neg(\beta \prec \alpha)$. Thus,

$$\overline{A_1} \subseteq \mathsf{mins}_\prec(\Sigma). \tag{10.6}$$

For all $\alpha, \beta \in \overline{A_1}$, since $pos_s(\beta) = pos_s(\alpha)$, by Lemma 6(1), we have

$$\neg(\alpha \Leftrightarrow \beta). \tag{10.7}$$

For any $\alpha \in \overline{A_1}$ and $\beta \in \Sigma \setminus \overline{A_1}$, since $pos_s(\beta) < pos_s(\alpha)$, by Lemma 6(2),

$$\neg(\beta \sqsubset \alpha). \tag{10.8}$$

From (10.6), (10.7) and (10.8), we know that $\overline{A_1} \in Z(\Sigma)$. Hence, $Z(\Sigma) \neq \emptyset$. This ensures the least element of $\{l[Y] \mid Y \in Z(\Sigma)\}$ w.r.t. $<^{st}$ is well defined.

Let $Y_0 \in Z(\Sigma)$ such that $B_0 = l[Y_0]$ is the least element of $\{l[Y] \mid Y \in Z(\Sigma)\}$ w.r.t. $<^{st}$. We want to show that $A_1 = B_0$. Since $<^{st}$ is a total order, we know that $A_1 <^{st} B_0$ or $B_0 <^{st} A_1$ or $A_1 = B_0$. But since $\overline{A_1} \in Z(\Sigma)$ and $B_0$ be the least element of the set $\{l[B] \mid B \in Z(\Sigma)\}$, $\neg(A_1 <^{st} B_0)$. Hence, to show that $A_1 = B_0$, it suffices to show $\neg(B_0 <^{st} A_1)$.

Suppose that $B_0 <^{st} A_1$. We first want to show that for every non-empty $W \subseteq Y_0$ there is an enumerated step sequence $v$ such that

$$\bar{v} = W_0 \overline{v_0} \equiv \overline{A_1} \dots \overline{A_n} \quad \text{and} \quad W \subseteq W_0 \subseteq Y_0. \tag{10.9}$$

We will prove this by induction on $|W|$.

*Base case.* When $|W| = 1$, we let $\{\alpha_0\} = W$. We choose $\overline{v_1} = \overline{E_0} \dots \overline{E_k}\,\overline{y_1} \equiv \overline{A_1} \dots \overline{A_n}$ and $\alpha_0 \in \overline{E_k}$ $(k \geqslant 0)$ such that for all $\overline{v'} = \overline{E'_0} \dots \overline{E'_{k'}}\,\overline{y'_1} \equiv \overline{A_1} \dots \overline{A_n}$ and $\alpha_0 \in \overline{E'_{k'}}$, we have

(i) $weight(\overline{E_0} \dots \overline{E_k}) \leqslant weight(\overline{E'_0} \dots \overline{E'_{k'}})$, and
(ii) $weight(\overline{E_{k-1}}\,\overline{E_k}) \leqslant weight(\overline{E'_{k'-1}}\,\overline{E'_{k'}})$.

We then consider only $\overline{w} = \overline{E_0} \dots \overline{E_k}$. We observe by the way we chose $\overline{v_1}$, we have $\forall \beta \in \biguplus(\overline{w}).(\beta \neq \alpha_0 \Longrightarrow \forall t \in [w].pos_t(\beta) \leqslant pos_t(\alpha_0))$. Hence, since $\overline{w} = \bar{u} \div_R \overline{v_0}$, it follows from Proposition 10(1, 2) that

$$\forall \beta \in \biguplus(\overline{w}).\big(\beta \neq \alpha_0 \Longrightarrow \forall t \in [A_1 \dots A_n].pos_t(\beta) \leqslant pos_t(\alpha_0)\big).$$

Then it follows from Lemma 6(2) that $\forall\beta \in \biguplus(\overline{w}).(\beta \neq \alpha_0 \Longrightarrow \beta \sqsubset \alpha_0)$. But by the way $Y_0$ was chosen, we know that $\forall\alpha \in Y_0.\forall\beta \in \Sigma \setminus Y_0.\neg(\beta \sqsubset \alpha)$. Hence,

$$\biguplus(\overline{w}) = (\overline{E_0} \cup \cdots \cup \overline{E_k}) \subseteq Y_0. \tag{10.10}$$

We next want to show

$$\forall\alpha \in \overline{E_i}.\forall\beta \in \overline{E_j}.\{\alpha\}\{\beta\} \equiv \{\alpha,\beta\} \quad (0 \leqslant i < j \leqslant k). \tag{10.11}$$

Suppose not. Then either $[\{\alpha\}\{\beta\}] = \{\{\alpha\}\{\beta\}\}$ or $[\{\alpha\}\{\beta\}] = \{\{\alpha\}\{\beta\},\{\beta\}\{\alpha\}\}$. In either case, we have $\forall t \in [\{l(\alpha)\}\{l(\beta)\}].pos_t(\alpha) \neq pos_t(\beta)$. Since $\{\alpha\}\{\beta\} \equiv \pi_{\{\alpha,\beta\}}(\bar{u})$, by Proposition 10(3), $\forall t \in [u].pos_t(\alpha) \neq pos_t(\beta)$. So by Lemma 6, $\alpha <> \beta$. This contradicts that $Y_0 \in Z(\Sigma)$ and $\alpha,\beta \in \Sigma(\overline{w}) \subseteq Y_0$. Thus, we have shown (10.11), which implies that for all $\alpha \in \overline{E_i}$ and $\beta \in \overline{E_j}$ $(0 \leqslant i < j \leqslant k)$, $(l(\alpha), l(\beta)) \in ser$. Then $\overline{E_0}\ldots\overline{E_k} \equiv \bigcup_{i=0}^{k}\overline{E_i}$. Hence, by (10.10) and (10.11), there exists a step sequence $v_1''$ such that $\overline{v_1''} = (\bigcup_{i=0}^{k}\overline{E_i})\overline{y_1} \equiv \overline{A_1}\ldots\overline{A_n}$ and $\{\alpha_0\} \subseteq \bigcup_{i=0}^{k}\overline{E_i} \subseteq Y_0$.

*Inductive step.* When $|W| > 1$, we pick an element $\beta_0 \in W$. By applying the induction hypothesis on $W \setminus \{\beta_0\}$, we get a step sequence $v_2$ such that $\overline{v_2} = \overline{F_0}\,\overline{y_2} \equiv \overline{A_1}\ldots\overline{A_n}$ where $W \setminus \{\beta_0\} \subseteq \overline{F_0} \subseteq Y_0$. If $W \subseteq \overline{F_0}$, we are done. Otherwise, proceeding like the base case, we construct a step sequence $v_3$ such that $\overline{v_3} = \overline{F_0}\,\overline{F_1}\,\overline{y_3} \equiv \overline{A_1}\ldots\overline{A_n}$ and $\{\beta_0\} \subseteq \overline{F_1} \subseteq Y_0$. Since $\overline{F_0} \subseteq Y_0$, we have $W \subseteq \overline{F_0} \cup \overline{F_1} \subseteq Y_0$. Then similarly to how we proved (10.11), we can show that $\forall\alpha \in \overline{F_0}.\forall\beta \in \overline{F_1}.\{\alpha\}\{\beta\} \equiv \{\alpha,\beta\}$. This means that for all $\alpha \in \overline{F_0}$ and $\beta \in \overline{F_1}$, $(l(\alpha), l(\beta)) \in ser$. Hence, $\overline{F_0}\,\overline{F_1} \equiv \overline{F_0} \cup \overline{F_1}$. Hence, there is a step sequence $v_4$ such that $\overline{v_4} = (\overline{F_0} \cup \overline{F_1})\overline{y_4} \equiv \overline{A_1}\ldots\overline{A_n}$ and $W \subseteq (\overline{F_0} \cup \overline{F_1}) \subseteq Y_0$.

Thus, we have shown (10.9). So by choosing $W = Y_0$, we get a step sequence $v$ such that $\bar{v} = W_0\overline{v_0} \equiv \overline{A_1}\ldots\overline{A_n}$ and $Y_0 \subseteq W_0 \subseteq Y_0$. Hence, $\bar{v} = W_0\overline{v_0} \equiv \overline{A_1}\ldots\overline{A_n}$. Thus, $v = B_0v_0 \equiv A_1\ldots A_n$. But since $B_0 <^{st} A_1$, this contradicts the fact that $A_1\ldots A_n$ is the least element of $[s]$ w.r.t. $<^{lex}$. Hence, $A_1$ is the least element of $\{l[Y] \mid Y \in Z(\Sigma)\}$ w.r.t. $<^{st}$.

We now prove that $A_i$ is the least element of the set $\{l[Y] \mid Y \in Z(\Sigma \setminus \biguplus(\overline{A_1}\ldots\overline{A_{i-1}}))\}$ w.r.t. $<^{st}$ by induction on $n$, the number of steps of the g-canonical step sequence $u = A_1\ldots A_n$. If $n = 0$, we are done. If $n > 0$, then we have just shown that $A_1$ is the least element of $\{l[Y] \mid Y \in Z(\Sigma)\}$ w.r.t. $<^{st}$. By applying the induction hypothesis on $p = \overline{A_2}\ldots\overline{A_n}$, $\Sigma_p = \Sigma \setminus \overline{A_1}$, and its gso-structure $(\Sigma_p, <> \cap (\Sigma_p \times \Sigma_p), \sqsubset \cap (\Sigma_p \times \Sigma_p))$, we get $A_i$ is the least element of the set $\{l[Y] \mid Y \in Z(\Sigma \setminus \biguplus(\overline{A_1}\ldots\overline{A_{i-1}}))\}$ w.r.t. $<^{st}$ for all $i \geqslant 2$. □

**Theorem 15.** *Let $s$ and $t$ be step sequences over a g-comtrace alphabet $(E, sim, ser, inl)$. Then $s \equiv t$ iff $G^{\{s\}} = G^{\{t\}}$.*

**Proof.** ($\Rightarrow$) If $s \equiv t$, then $[s] = [t]$. Hence, by Theorem 13, $G^{\{s\}} = G^{\{t\}}$.

($\Leftarrow$) By Lemma 7, we can use $G^{\{s\}}$ to construct a unique element $w_1$ such that $w_1$ is the least element of $[s]$ w.r.t. $<^{lex}$, and then use $G^{\{t\}}$ to construct a unique element $w_2$ that is the least element of $[t]$ w.r.t. $<^{lex}$. But since $G^{\{s\}} = G^{\{t\}}$, we get $w_1 = w_2$. Hence, $s \equiv t$. □

Theorem 15 justifies the following definition:

**Definition 28.** For every g-comtrace $[s]$, $G_{[s]} = G^{\{s\}} = (\Sigma_s, \prec_s \cup <>_s, \prec_s \cup \sqsubset_s)^{\bowtie}$ is the *gso-structure induced by the g-comtrace* $[s]$.

To end this section, we prove two major results. Theorem 16 says that the stratified extensions of the gso-structure induced by a g-comtrace $[t]$ are exactly those generated by the step sequences in $[t]$. Theorem 17 says that the gso-structure induced by a g-comtrace is uniquely identified by any of its stratified extensions.

**Lemma 8.** *Let $s, t \in \mathbb{S}^*$ and $\lhd_s \in ext(G^{\{t\}})$. Then $G^{\{s\}} = G^{\{t\}}$.*

The proof of the above lemma uses Definition 25 heavily and thus requires a separate analysis of many cases and was moved to Appendix B.

**Theorem 16.** *Let $s, t \in \mathbb{S}^*$. Then $ext(G^{\{s\}}) = \{\lhd_u \mid u \in [s]\}$.*

**Proof.** ($\subseteq$) Suppose $\lhd \in ext(G^{\{s\}})$. By Proposition 16, there is a step sequence $u$ such that $\lhd_u = \lhd$. Hence, by Lemma 8, we have $G^{\{u\}} = G^{\{s\}}$, which by Theorem 15 implies that $u \equiv s$. Hence, $ext(G^{\{s\}}) \subseteq \{\lhd_u \mid u \in [s]\}$.

($\supseteq$) If $u \in [s]$, then it follows from Theorem 15 that $G^{\{u\}} = G^{\{s\}}$. This and Proposition 15 imply $\lhd_u \in ext(G^{\{s\}})$. Hence, $ext(G^{\{s\}}) \supseteq \{\lhd_u \mid u \in [s]\}$. □

**Theorem 17.** *Let $s, t \in \mathbb{S}^*$ and $ext(G^{\{s\}}) \cap ext(G^{\{t\}}) \neq \emptyset$. Then $s \equiv t$.*

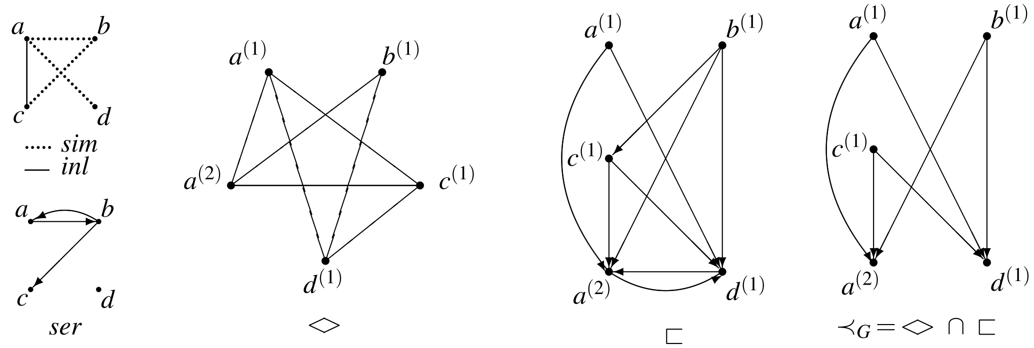



**Fig. 4.** A g-comtrace alphabet $(E, sim, ser, inl)$, where $E = \{a, b, c, d\}$, the gso-structure $G = (X, \Diamond, \sqsubset)$ and $\prec_G = \Diamond \cap \sqsubset$ defined by the g-comtrace $[\{a,b\}\{c\}\{a,d\}] = \{\{a,b\}\{c\}\{a,d\},\ \{a\}\{b\}\{c\}\{a,d\}, \{a\}\{b,c\}\{a,d\}, \{b\}\{a\}\{c\}\{a,d\}, \{b\}\{c\}\{a\}\{a,d\},\ \{b,c\}, \{a\}\{a,d\}\}$.

**Proof.** Let $\lhd \in ext(G^{\{s\}}) \cap ext(G^{\{t\}})$. By Proposition 16, there is a step sequence $u$ such that $\lhd_u = \lhd$. By Lemma 8, we have $G^{\{s\}} = G^{\{u\}} = G^{\{t\}}$. This and Theorem 15 yield $s \equiv t$. $\square$

Summing up, we have proved the analogue of Theorem 12 for g-comtraces. In fact, Theorem 12 is a straightforward consequence of this section for $inl = \emptyset$.

Fig. 4 shows an example of a g-comtrace and the gso-structure it generates.

## 11. Conclusion and future work

The comtrace concept is revisited and its extension, the g-comtrace notion, is introduced. Comtraces and g-comtraces are generalizations of Mazurkiewicz traces, where the concepts of simultaneity, serializability and interleaving are used to define the quotient monoids instead of the usual independency relation in the case of traces. We analyzed some algebraic properties of comtraces and g-comtraces, where an interesting application is the proof of the uniqueness of comtrace canonical representation. We study the canonical representations of traces, comtraces and g-comtraces and their mutual relationships in a more unified framework. We observe that comtraces have a natural unique canonical form which corresponds to their maximal concurrent representation,[7] while the unique canonical representation of g-comtrace can only be obtained by choosing the lexicographically least element.

The most important contribution of this paper, Theorem 16, shows that every g-comtrace uniquely determines a labeled gso-structure. We believe the reason why the proof of Theorem 16 is more technical than the similar theorem for comtraces is that both comtraces and so-structures satisfy paradigm $\pi_3$ while g-comtraces and gso-structures do not. Intuitively, what paradigm $\pi_3$ really says is that the underlying structure consists of partial orders. For comtraces and so-structures, we did augment some more priority relationships into the incomparable elements with respect to the standard causal partial order to produce the not later than relation, and this process might introduce cycles into the graph of the "not later than" relation. However, it is important to observe that any two distinct elements lying on a cycle of the "not later than" relation must belong to the same synchronous set since the "not later than" relation is a *strict preorder*. Thus, if we collapse each synchronous set into a single vertex, then the resulting "quotient" graph of the "not later than" relation is a partial order. The reader is referred to the second author's recent work [22] for more detailed discussion on the preorder property of the "not later than" relation and how this property manifests itself in the comtrace notion. When paradigm $\pi_3$ is not satisfied, as with g-comtraces or gso-structures, we have more than a partial order structure, and hence the usual techniques that depend too on the underlying partial order structure of comtraces and so-structures are often not applicable.

Despite some obvious advantages, for instance, handy composition and no need to use labels, quotient monoids (perhaps with some exception of traces) are less popular for analyzing issues of concurrency than their relational counterparts such as partial orders, so-structures, occurrence graphs, etc. We believe that in many cases, more sophisticated quotient monoids, e.g., comtraces and g-comtraces, can provide simpler and more adequate models of concurrent histories than their relational equivalences.

Much harder future tasks are in the area of comtrace and g-comtrace languages where major problems like recognizability [25], acceptability [30], etc. are still open.


## Acknowledgments

We are grateful to the anonymous referees, who pointed out many typos and suggested us a better way to organize the materials of this paper. We are indebted to many others for their feedbacks and encouragements. These include Steve Cook,


[7] This is also true for traces when they are represented as *vector firing sequences* [3].

Grzegorz Herman, Marek Janicki, Galina Jirásková, Gregory H. Moore, Michael Soltys, Yuli Ye, Marek Zaionc, Nadya Zubkova, and Jeffery Zucker. This work is financially supported by the Ontario Graduate Scholarship and the Natural Sciences and Engineering Research Council of Canada.

## Appendix A. Proof of Lemma 6

**Proposition A.1.** *Let $u$ be a step sequence over a g-comtrace alphabet $(E, sim, ser, inl)$ and $\alpha, \beta \in \Sigma_u$ such that $l(\alpha) = l(\beta)$ and $\alpha \neq \beta$. Then*

1. $pos_u(\alpha) \neq pos_u(\beta)$,
2. *if $pos_u(\alpha) < pos_u(\beta)$ and $v$ is a step sequence satisfying $v \equiv u$, then $pos_v(\alpha) < pos_v(\beta)$.*

**Proof.** 1. Follows from the fact that *sim* is irreflexive.

2. Follows from Proposition 7 and that *ser* and *inl* are irreflexive. □

From the definition of g-comtrace $\approx_{\{ser,inl\}}$ (Definition 12), we can easily show the following proposition, which aims to describe the intuition that if an event $\alpha$ occurs before (or simultaneously with) $\beta$ in the first step sequence and $\alpha$ occurs later than $\beta$ on the second step sequence congruent with the first one, then there must be two "immediately congruent" step sequences, i.e., related by the relation $\approx_{\{ser,inl\}}$ (written as just $\approx$), where this commutation (or serialization) of $\alpha$ and $\beta$ occurs.

**Proposition A.2.** *Let $u, w$ be step sequences over a g-comtrace alphabet $(E, sim, ser, inl)$ such that $u(\approx \cup \approx^{-1})w$. Then*

1. *if $pos_u(\alpha) < pos_u(\beta)$ and $pos_w(\beta) < pos_w(\alpha)$, then there are $x, y, A, B$ such that $\bar{u} = \bar{x}\bar{A}\bar{B}\bar{y}(\approx \cup \approx^{-1})\bar{x}\bar{B}\bar{A}\bar{y} = \bar{w}$ and $\alpha \in \bar{A}$, $\beta \in \bar{B}$. We also have $(l(\alpha), l(\beta)) \in inl$,*
2. *if $pos_u(\alpha) = pos_u(\beta)$ and $pos_w(\beta) < pos_w(\alpha)$, then there are $x, y, A, B, C$ such that $\bar{u} = \bar{x}\bar{A}\bar{y} \approx \bar{x}\bar{B}\bar{C}\bar{y} = \bar{w}$ and $\beta \in \bar{B}$ and $\alpha \in \bar{C}$. This also means $(l(\beta), l(\alpha)) \in ser$.* □

**Proposition A.3.** *Let $s$ be a step sequence over a g-comtrace alphabet $(E, sim, ser, inl)$. If $\alpha, \beta \in \Sigma_s$, then*

1. $\alpha <>_s \beta \Longrightarrow \forall u \in [s].pos_u(\alpha) \neq pos_u(\beta)$,
2. $\alpha \sqsubset_s \beta \Longrightarrow \forall u \in [s].pos_u(\alpha) \leqslant pos_u(\beta)$,
3. $\alpha \prec_s \beta \Longrightarrow \forall u \in [s].pos_u(\alpha) < pos_u(\beta)$,

*and $\alpha \neq \beta$ in all three cases.*

**Proof.** 1. Follows from the fact that $inl \cap sim = \emptyset$.

2. Assume that $\alpha \sqsubset_s \beta$. Suppose that $\exists u \in [s].pos_u(\alpha) > pos_u(\beta)$. Then there must be some $u_1, u_1 \in [s]$ such that $u_1(\approx \cup \approx^{-1})u_2$ and $pos_{u_1}(\alpha) \leqslant pos_{u_1}(\beta)$ and $pos_{u_2}(\alpha) > pos_{u_2}(\beta)$. There are two cases:

(i) If $pos_{u_1}(\alpha) < pos_{u_1}(\beta)$ and $pos_{u_2}(\alpha) > pos_{u_2}(\beta)$, then by Proposition A.2(1), $(l(\alpha), l(\beta)) \in inl$, contradicting that $\alpha \sqsubset_s \beta$.
(ii) If $pos_{u_1}(\alpha) = pos_{u_1}(\beta)$ and $pos_{u_2}(\alpha) > pos_{u_2}(\beta)$, then it follows from Proposition A.2(2), $(l(\beta), l(\alpha)) \in ser$, contradicting that $\alpha \sqsubset_s \beta$.

3. Assume that $\alpha \prec_s \beta$. Suppose that $\exists u \in [s].pos_u(\alpha) \geqslant pos_u(\beta)$. Then there must be some $u_1, u_1 \in [s]$ such that $u_1(\approx \cup \approx^{-1})u_2$ and $pos_{u_1}(\alpha) < pos_{u_1}(\beta)$ and $pos_{u_2}(\alpha) \geqslant pos_{u_2}(\beta)$. There are two cases:

(i) If $pos_{u_1}(\alpha) < pos_{u_1}(\beta)$ and $pos_{u_2}(\alpha) = pos_{u_2}(\beta)$, then it follows from Proposition A.2(2) that $(l(\alpha), l(\beta)) \in ser$ and $\neg(\alpha <>_s \beta)$. Hence, it follows from (10.3) that

$$\exists \delta, \gamma \in \Sigma_s. \begin{pmatrix} pos_s(\delta) < pos_s(\gamma) \wedge (l(\delta), l(\gamma)) \notin ser \\ \wedge\ \alpha \sqsubset_s^* \delta \sqsubset_s^* \beta \wedge \alpha \sqsubset_s^* \gamma \sqsubset_s^* \beta \end{pmatrix}.$$

By (2) and transitivity of $\leqslant$, we have

$$\begin{pmatrix} \gamma \neq \delta \wedge (l(\delta), l(\gamma)) \notin ser \\ \quad \wedge\ (\forall u \in [s].pos_u(\alpha) \leqslant pos_u(\delta) \leqslant pos_u(\beta)) \\ \quad \wedge\ (\forall u \in [s].pos_u(\alpha) \leqslant pos_u(\gamma) \leqslant pos_u(\beta)) \end{pmatrix}.$$

But since $pos_{u_2}(\alpha) = pos_{u_2}(\beta)$, we get $pos_{u_2}(\gamma) = pos_{u_2}(\delta)$. Since we assumed $pos_s(\delta) < pos_s(\gamma)$, it follows from Proposition A.2(2) that $(l(\delta), l(\gamma)) \in ser$, a contradiction.

(ii) If $pos_{u_1}(\alpha) < pos_{u_1}(\beta)$ and $pos_{u_2}(\alpha) > pos_{u_2}(\beta)$, then by Proposition A.2(1), $(l(\alpha), l(\beta)) \in inl$. Since we already assumed $\alpha \prec_s \beta$, by (10.3), $(\alpha, \beta) \in {<\!>_s} \cap ((\sqsubset_s^*)^{\Cap} \circ {<\!>_s^{\complement}} \circ (\sqsubset_s^*)^{\Cap})$. So there are $\gamma, \delta$ such that $\alpha (\sqsubset_s^*)^{\Cap} \gamma <\!>_s^{\complement} \delta (\sqsubset_s^*)^{\Cap} \beta$. Observe that

$$\begin{aligned}
&\alpha (\sqsubset_s^*)^{\Cap} \gamma \\
&\quad \Longrightarrow \quad \alpha (\sqsubset_s^*) \gamma \wedge \gamma (\sqsubset_s^*) \alpha \\
&\quad \Longrightarrow \quad \forall u \in [s].pos_u(\alpha) \leqslant pos_u(\gamma) \wedge \forall u \in [s].pos_u(\gamma) \leqslant pos_u(\alpha) \quad \langle \text{by (2)} \rangle \\
&\quad \Longrightarrow \quad \forall u \in [s].pos_u(\alpha) = pos_u(\gamma) \\
&\quad \Longrightarrow \quad \{\alpha, \gamma\} \subseteq \overline{A} \quad \langle \text{since } \alpha \in \overline{A} \rangle.
\end{aligned}$$

Similarly, since $\delta (\sqsubset_s^*)^{\Cap} \beta$, we can show that $\{\delta, \beta\} \subseteq \overline{B}$. Since $\bar{x}\overline{A}\,\overline{B}\bar{y} (\approx \cup \approx^{-1}) \bar{x}\overline{B}\,\overline{A}\bar{y}$, we get $A \times B \subseteq inl$. So $(l(\gamma), l(\delta)) \in inl$. But $\gamma <\!>_s^{\complement} \delta$ implies that $(l(\gamma), l(\delta)) \notin inl$, a contradiction. $\square$

Immediately from Proposition A.3, we get the following proposition.

**Proposition A.4.** *Let $s$ be a step sequence over a g-comtrace alphabet $(E, sim, ser, inl)$ and $G^{\{s\}} = (\Sigma_s, <\!>, \sqsubset)$. If $\alpha, \beta \in \Sigma_s$, then*

1. $\alpha <\!> \beta \Longrightarrow \forall u \in [s].pos_u(\alpha) \neq pos_u(\beta)$,
2. $\alpha \sqsubset \beta \Longrightarrow (\alpha \neq \beta \wedge \forall u \in [s].pos_u(\alpha) \leqslant pos_u(\beta))$. $\square$

**Definition A.1** *(Serializable and non-serializable steps).* Let $A$ be a step over a g-comtrace alphabet $(E, sim, ser, inl)$ and let $a \in A$ then:

1. Step $A$ is called *serializable* iff
$$\exists B, C \in \wp^{\setminus\{\emptyset\}}(A).B \cup C = A \wedge B \times C \subseteq ser.$$
Step $A$ is called *non-serializable* iff $A$ is not serializable. Every non-serializable step is a synchronous step as defined in Definition 9.
2. Step $A$ is called *serializable to the left of $a$* iff
$$\exists B, C \in \wp^{\setminus\{\emptyset\}}(A).B \cup C = A \wedge a \in B \wedge B \times C \subseteq ser.$$
Step $A$ is called *non-serializable to the left of $a$* iff $A$ is not serializable to the left of $a$, i.e., $\forall B, C \in \wp^{\setminus\{\emptyset\}}(A).(B \cup C = A \wedge a \in B) \Longrightarrow B \times C \nsubseteq ser$.
3. Step $A$ is called *serializable to the right of $a$* iff
$$\exists B, C \in \wp^{\setminus\{\emptyset\}}(A).B \cup C = A \wedge a \in C \wedge B \times C \subseteq ser.$$
Step $A$ is called *non-serializable to the right of $a$* iff $A$ is not serializable to the right of $a$, i.e., $\forall B, C \in \wp^{\setminus\{\emptyset\}}(A).(B \cup C = A \wedge a \in C) \Longrightarrow B \times C \nsubseteq ser$.

**Proposition A.5.** *Let $A$ be a step over a g-comtrace alphabet $(E, sim, ser, inl)$. Then*

1. *if $A$ is non-serializable to the left of $l(\alpha)$ for some $\alpha \in \overline{A}$, then $\alpha \sqsubset_A^* \beta$ for all $\beta \in \overline{A}$,*
2. *if $A$ is non-serializable to the right of $l(\beta)$ for some $\beta \in \overline{A}$, then $\alpha \sqsubset_A^* \beta$ for all $\alpha \in \overline{A}$,*
3. *if $A$ is non-serializable, then $\forall \alpha, \beta \in \overline{A}.\alpha \sqsubset_A^* \beta$.*

Before we proceed with the proof, since for all $\alpha, \beta \in \overline{A}$, $(l(\alpha), l(\beta)) \notin inl$, observe that

$$\alpha \sqsubset_A \beta \quad \Longleftrightarrow \quad pos_A(\alpha) \leqslant pos_A(\beta) \wedge \big(l(\beta), l(\alpha)\big) \notin ser.$$

**Proof of Proposition A.5.** 1. For any $\beta \in \overline{A}$, we have to show that $\alpha \sqsubset_A^* \beta$. We define the $\sqsubset_A$*-right closure set* of $\alpha$ inductively as follows:

$$RC^0(\alpha) \triangleq \{\alpha\}, \qquad RC^n(\alpha) \triangleq \big\{\delta \in \overline{A} \,\big|\, \exists \gamma \in RC^{n-1}(\alpha) \wedge \gamma \sqsubset_A \delta\big\}.$$

Then by induction on $n$, we can show that $|RC^{n+1}(\alpha)| > |RC^n(\alpha)|$ or $RC^n(\alpha) = \overline{A}$. So if $A$ is finite, then for some $n < |A|$, we must have $RC^n(\alpha) = \overline{A}$ and $\beta \in RC^n(\alpha)$. It follows that $\alpha \sqsubset_A^* \beta$.

2. Dually to (1).

3. Since $A$ is non-serializable, it follows that $A$ is non-serializable to the left of $l(\alpha)$ for *every* $\alpha \in \overline{A}$. Hence, the assertion follows. $\square$

The existence of a non-serializable sub-step of a step $A$ to the left/right of an element $a \in A$ can be explained by the following proposition.

**Proposition A.6.** *Let $A$ be a step over an alphabet $\Theta = (E, sim, ser, inl)$ and $a \in A$. Then*

1. *there exists a unique $B \subseteq A$ such that $a \in B$, $B$ is non-serializable to the left of $a$, and $A \neq B \Longrightarrow A \equiv (A \setminus B)B$,*
2. *there exists a unique $C \subseteq A$ such that $a \in C$, $C$ is non-serializable to the right of $a$, and $A \neq C \Longrightarrow A \equiv C(A \setminus C)$,*
3. *there exists a unique $D \subseteq A$ such that $a \in D$, $D$ is non-serializable, and $A \equiv xDy$, where $x$ and $y$ are step sequences over $\Theta$.*

**Proof.** 1. If $A$ is non-serializable to the left of $a$, then $B = A$. If $A$ is serializable to the left of $a$, then the following set is not empty:

$$\zeta \triangleq \left\{D \in \wp^{\setminus\{\emptyset\}}(A) \;\middle|\; \exists C \in \wp^{\setminus\{\emptyset\}}(A).(C \cup D = A \wedge a \in D \wedge C \times D \subseteq ser)\right\}.$$

Let $B \in \zeta$ such that $B$ is a minimal element of the poset $(\zeta, \subset)$. Let $B \in \zeta$ such that $B$ is a minimal element of the poset $(\zeta, \subset)$. We claim that $B$ is non-serializable to the left of $a$. Suppose for a contradiction that $B$ is serializable to the left of $a$, then there are some sets $E, F \in \wp^{\setminus\{\emptyset\}}(()B)$ such that $E \cup F = B \wedge a \in F \wedge E \times F \subseteq ser$. Since $B \in \chi$, there is some set $G \in \wp^{\setminus\{\emptyset\}}(()A)$ such that $G \cup B = A \wedge a \in B \wedge G \times B \subseteq ser$. Because $G \times B \subseteq ser$ and $F \subset B$, it follows that $G \times F \subseteq ser$. But since $E \times F \subseteq ser$, we have $(G \cup E) \times F \subseteq ser$. Hence, $(G \cup E) \cup F = A \wedge a \in F \wedge (G \cup E) \times F \subseteq ser$. So $E \in \zeta$ and $E \subset B$. This contradicts that $B$ is minimal. Hence, $B$ is non-serializable to the left of $a$.

By the way the set $\zeta$ is defined, $A \equiv (A \setminus B)B$. It remains to prove the uniqueness of $B$. Let $B' \in \zeta$ such that $B'$ is a minimal element of the poset $(\zeta, \subset)$. We want to show that $B = B'$.

We first show that $B \subseteq B'$. Suppose that there is some $b \in B$ such that $b \neq a$ and $b \notin B'$. Let $\alpha$ and $\beta$ denote the event occurrences $a^{(1)}$ and $b^{(1)}$ in $\Sigma_A$ respectively. Since $a \in B$ and $b$ is non-serializable to the left of $a$ and $a \neq b$, it follows from Proposition A.5(1) that $\alpha \sqsubset_{[A]} \beta$. Hence, by Proposition A.3(2), we have

$$\forall u \in [A].pos_u(\alpha) \leqslant pos_u(\beta). \tag{A.1}$$

By the way $B'$ is chosen, we know $A \equiv (A \setminus B')B'$ and $b \notin B'$. So it follows that $b \in (A \setminus B')$. Hence, we have $(A \setminus B')B' \in [A]$ and $pos_{(A\setminus B')B'}(\beta) < pos_{(A\setminus B')B'}(\alpha)$, which contradicts (A.1). Thus, $B \subseteq B'$. By reversing the roles of $B$ and $B'$, we can prove that $B \supseteq B'$. Hence, $B = B'$.

2. Dually to (1).

3. By (1) and (2), we choose $D$ to be non-serializable to the left and to the right of $a$. □

**Lemma A.1.** *Let $s$ be a step sequence over a g-comtrace alphabet $(E, sim, ser, inl)$ and $G^{\{s\}} = (\Sigma_s, \Leftrightarrow, \sqsubset)$. Let $\prec = \sqsubset \cup \Leftrightarrow$. If $\alpha, \beta \in \Sigma_s$, then*

1. $\begin{pmatrix} (\forall u \in [s].pos_u(\alpha) \neq pos_u(\beta)) \\ \wedge\ (\exists u \in [s].pos_u(\alpha) < pos_u(\beta)) \\ \wedge\ (\exists u \in [s].pos_u(\alpha) > pos_u(\beta)) \end{pmatrix} \Longrightarrow \alpha \Leftrightarrow \beta,$
2. $(\forall u \in [s].pos_u(\alpha) < pos_u(\beta)) \Longrightarrow \alpha \prec \beta,$
3. $(\alpha \neq \beta \wedge \forall u \in [s].pos_u(\alpha) \leqslant pos_u(\beta)) \Longrightarrow \alpha \sqsubset \beta.$

**Proof.** 1. Assume the left-hand side of the implication. Then by Proposition A.2(1), $(l(\alpha), l(\beta)) \in inl$, which by (10.1) implies that $\alpha \Leftrightarrow_s \beta$. By Definitions 23 and 25, it follows that $\alpha \Leftrightarrow \beta$.

2, 3. Since statements (2) and (3) are mutually related due to the fact that $\prec \subseteq \sqsubset$, we cannot prove each statement separately. The main technical insight is that, to have a stronger induction hypothesis, we need prove both statements simultaneously.

Assume $\forall u \in [s].pos_u(\alpha) \leqslant pos_{\underline{u}}(\beta)$ and $\alpha \neq \beta$. Hence, we can choose $u_0 \in [s]$ where $\overline{u_0} = \overline{x_0}\,\overline{E_1}\ldots\overline{E_k}\,\overline{y_0}$ $(k \geqslant 1)$, $E_1, E_k$ are non-serializable, $\alpha \in \overline{E_1}$, $\beta \in \overline{E_k}$, and

$$\forall u'_0 \in [s].\begin{pmatrix} (\overline{u'_0} = \overline{x'_0}\,\overline{E'_1}\ldots\overline{E'_{k'}}\,\overline{y'_0} \wedge \alpha \in \overline{E'_1} \wedge \beta \in \overline{E'_{k'}}) \\ \Longrightarrow weight(\overline{E_1}\ldots\overline{E_k}) \leqslant weight(\overline{E'_1}\ldots\overline{E'_{k'}}) \end{pmatrix}. \tag{A.2}$$

We will prove by induction on $weight(\overline{E_1}\ldots\overline{E_k})$ that

$$\big(\forall u \in [s].pos_u(\alpha) < pos_u(\beta)\big) \quad \Longrightarrow \quad \alpha \prec \beta, \tag{A.3}$$

$$\big(\alpha \neq \beta \wedge \forall u \in [s].pos_u(\alpha) \leqslant pos_u(\beta)\big) \quad \Longrightarrow \quad \alpha \sqsubset \beta. \tag{A.4}$$

*Base case.* When $weight(\overline{E_1}\ldots\overline{E_k})=2$, then we consider two cases:

- If $\alpha\neq\beta$, $\forall u\in[s].pos_u(\alpha)\leqslant pos_u(\beta)$ and $\exists u\in[s].pos_u(\alpha)=pos_u(\beta)$, then
  - $\overline{u_0}=\overline{x_0}\{\alpha,\beta\}\overline{y_0}$, or
  - $\overline{u_0}=\overline{x_0}\{\alpha\}\{\beta\}\overline{y_0}\equiv\overline{x_0}\{\alpha,\beta\}\overline{y_0}$.

  But since $\forall u\in[s].pos_u(\alpha)\leqslant pos_u(\beta)$, in either case, we must have $\{l(\alpha),l(\beta)\}$ is not serializable to the right of $l(\beta)$. Hence, by Proposition A.5(2), $\alpha(\sqsubset_s)^*\beta$. This by Definitions 23 and 25 implies that $\alpha\sqsubset\beta$.
- If $\forall u\in[s].pos_u(\alpha)<pos_u(\beta)$, then it follows $\overline{u_0}=\overline{x_0}\{\alpha\}\{\beta\}\overline{y_0}$ and $(l(\alpha),l(\beta))\notin ser\cup inl$. This, by (10.3), implies that $\alpha\prec_s\beta$. Hence, from Definitions 23 and 25, we get $\alpha\prec\beta$.

Since $\prec\subseteq\sqsubset$, it follows from these two cases that (A.3) and (A.4) hold.

*Inductive step.* When $weight(\overline{E_1}\ldots\overline{E_k})>2$, then $\overline{u_0}=\overline{x_0}\overline{E_1}\ldots\overline{E_k}\overline{y_0}$ where $k\geqslant 1$. We need to consider two cases:

*Case* (i): If $\alpha\neq\beta$ and $\forall u\in[s].pos_u(\alpha)\leqslant pos_u(\beta)$ and $\exists u\in[s].pos_u(\alpha)=pos_u(\beta)$, then there is some $v_0$ $\overline{v_0}=\overline{w_0}\overline{E}\overline{z_0}$ and $\alpha,\beta\in\overline{E}$. Either $E$ is non-serializable to the right of $l(\beta)$, or by Proposition A.6(2) $\overline{v_0}=\overline{w_0}\overline{E}\overline{z_0}\equiv\overline{w'_0}\overline{E'}\overline{z'_0}$ where $E'$ is non-serializable to the right of $l(\beta)$. In either case, by Proposition A.5(2), we have $\alpha\sqsubset_s^*\beta$. So by Definitions 23 and 25, $\alpha\sqsubset\beta$.

*Case* (ii): If $\forall u\in[s].pos_u(\alpha)<pos_u(\beta)$, then it follows $\overline{u_0}=\overline{x_0}\overline{E_1}\ldots\overline{E_k}\overline{y_0}$ where $k\geqslant 2$ and $\alpha\in\overline{E_1}$, $\beta\in\overline{E_k}$. If $(l(\alpha),l(\beta))\notin ser\cup inl$, then by (10.3), $\alpha\prec_s\beta$. Hence, from Definitions 23 and 25, we get $\alpha\prec\beta$. So we need to consider only when $(l(\alpha),l(\beta))\in ser$ or $(l(\alpha),l(\beta))\in inl$. There are three cases to consider:

(a) If $\overline{u_0}=\overline{x_0}\overline{E_1}\overline{E_2}\overline{y_0}$ where $E_1$ and $E_2$ are non-serializable, then since we assume $\forall u\in[s].pos_u(\alpha)<pos_u(\beta)$, it follows that $E_1\times E_2\nsubseteq ser$ and $E_1\times E_2\nsubseteq inl$. Hence, there are $\alpha_1,\alpha_2\in\overline{E_1}$ and $\beta_1,\beta_2\in\overline{E_2}$ such that $(l(\alpha_1),l(\beta_1))\notin inl$ and $(l(\alpha_2),l(\beta_2))\notin ser$. Since $E_1$ and $E_2$ are non-serializable, by Proposition A.5(3), $\alpha_1\sqsubset_s^*\alpha_2$ and $\beta_2\sqsubset_s^*\beta_1$. Also by Definition 25, we know that $\alpha_1<\!\!>_s\beta_2$ and $\alpha_2<\!\!>_s^{\complement}\beta_1$. Thus, by Definition 25, we have $\alpha_1\prec_s\beta_2$. Since $E_1$ and $E_2$ are non-serializable, by Proposition A.5(3), $\alpha\sqsubset_s^*\alpha_1\prec_s\beta_2\sqsubset_s^*\beta$. Hence, by Definitions 23 and 25, $\alpha\prec\beta$.

(b) If $\overline{u_0}=\overline{x_0}\overline{E_1}\ldots\overline{E_k}\overline{y_0}$ where $k\geqslant 3$ and $(l(\alpha),l(\beta))\in inl$, then let $\gamma\in\overline{E_2}$. Observe that we must have

$$\overline{u_0}=\overline{x_0}\overline{E_1}\ldots\overline{E_k}\overline{y_0}\equiv\overline{x_1}\overline{E_1}\overline{w_1}\overline{F}\overline{z_1}\overline{E_k}\overline{y_1}\equiv\overline{x_2}\overline{E_1}\overline{w_2}\overline{F}\overline{z_2}\overline{E_k}\overline{y_2}$$

such that $\gamma\in\overline{F}$, $F$ is non-serializable, and $weight(\overline{E_1}\overline{w_1}\overline{F}),weight(\overline{F}\overline{z_2}\overline{E_k})$ satisfy the minimal condition similarly to (A.2). Since from the way $u_0$ is chosen, we know that $\forall u\in[s].pos_u(\alpha)\leqslant pos_u(\gamma)$ and $\forall u\in[s].pos_u(\gamma)\leqslant pos_u(\beta)$, by applying the induction hypothesis, we get

$$\alpha\sqsubset\gamma\sqsubset\beta. \tag{A.5}$$

So by transitivity of $\sqsubset$, we get $\alpha\sqsubset\beta$. But since we assume $(l(\alpha),l(\beta))\in inl$, it follows that $\alpha<\!\!>\beta$. Hence, $(\alpha,\beta)\in\sqsubset\cap<\!\!>=\prec$.

(c) If $\overline{u_0}=\overline{x_0}\overline{E_1}\ldots\overline{E_k}\overline{y_0}$ where $k\geqslant 3$ and $(l(\alpha),l(\beta))\in ser$, then we observe from how $u_0$ is chosen that

$$\forall\gamma\in\biguplus(\overline{E_1}\ldots\overline{E_k}).\big(\forall u\in[s].pos_{u_0}(\alpha)\leqslant pos_{u_0}(\gamma)\leqslant pos_{u_0}(\beta)\big).$$

Similarly to how we show (A.5), we can prove that

$$\forall\gamma\in\biguplus(\overline{E_1}\ldots\overline{E_k})\setminus\{\alpha,\beta\}.\alpha\sqsubset\gamma\sqsubset\beta. \tag{A.6}$$

We next want to show that

$$\exists\delta,\gamma\in\biguplus(\overline{E_1}\ldots\overline{E_k}).\big(pos_{u_0}(\delta)<pos_{u_0}(\gamma)\wedge\big(l(\delta),l(\gamma)\big)\notin ser\big). \tag{A.7}$$

Suppose that (A.7) does not hold, then

$$\forall\delta,\gamma\in\biguplus(\overline{E_1}\ldots\overline{E_k}).\big(pos_{u_0}(\delta)<pos_{u_0}(\gamma)\Longrightarrow\big(l(\delta),l(\gamma)\big)\in ser\big).$$

It follows that $\overline{u_0}=\overline{x_0}\overline{E_1}\ldots\overline{E_k}\overline{y_0}\equiv\overline{x_0}\overline{E}\overline{y_0}$, which contradicts that $\forall u\in[s].pos_u(\alpha)<pos_u(\beta)$. Hence, we have shown (A.7).

Let $\delta,\gamma\in\biguplus(\overline{E_1}\ldots\overline{E_k})$ be event occurrences such that $pos_{u_0}(\delta)<pos_{u_0}(\gamma)$ and $(l(\delta),l(\gamma))\notin ser$. By (A.6), $\alpha(\sqsubset\cup id_{\Sigma_s})\delta(\sqsubset\cup id_{\Sigma_s})\beta$ and $\alpha(\sqsubset\cup id_{\Sigma_s})\gamma(\sqsubset\cup id_{\Sigma_s})\beta$. If $\alpha\prec\delta$ or $\delta\prec\beta$ or $\alpha\prec\gamma$ or $\gamma\prec\beta$, then by S4 of Definition 3, $\alpha\prec\beta$. Otherwise, by Definitions 23 and 25, we have $\alpha\sqsubset_s^*\delta\sqsubset_s^*\beta$ and $\alpha\sqsubset_s^*\gamma\sqsubset_s^*\beta$. But since $pos_{u_0}(\delta)<pos_{u_0}(\gamma)$ and $(l(\delta),l(\gamma))\notin ser$, by Definition 25, $\alpha\prec_s\beta$. So by Definitions 23 and 25, we have $\alpha\prec\beta$.

Thus, we have shown (A.3) and (A.4) as desired. □

**Lemma 6.** *Let s be a step sequence over a g-comtrace alphabet* $(E, sim, ser, inl)$. *Let* $G^{\{s\}} = (\Sigma_s, \Leftrightarrow, \sqsubset)$, *and let* $\prec = \Leftrightarrow \cap \sqsubset$. *Then for every* $\alpha, \beta \in \Sigma_s$, *we have*

1. $\alpha \Leftrightarrow \beta \iff \forall u \in [s].pos_u(\alpha) \neq pos_u(\beta)$,
2. $\alpha \sqsubset \beta \iff \alpha \neq \beta \wedge \forall u \in [s].pos_u(\alpha) \leqslant pos_u(\beta)$,
3. $\alpha \prec \beta \iff \forall u \in [s].pos_u(\alpha) < pos_u(\beta)$,
4. *if* $l(\alpha) = l(\beta)$ *and* $pos_s(\alpha) < pos_s(\beta)$, *then* $\alpha \prec \beta$.

**Proof.** 1. Follows from Proposition A.4(1) and Lemma A.1(1, 2).
2. Follows from Proposition A.4(2) and Lemma A.1(3).
3. Follows from (1) and (2).
4. Follows from Proposition A.1(2). □

## Appendix B. Proof of Lemma 8

**Lemma 8.** *Let* $s, t \in \mathbb{S}^*$ *and* $\lhd_s \in ext(G^{\{t\}})$. *Then* $G^{\{s\}} = G^{\{t\}}$.

**Proof.** To show $G^{\{s\}} = G^{\{t\}}$, it suffices to show that $\Leftrightarrow_t = \Leftrightarrow_s$, $\prec_t = \prec_s$ and $\sqsubset_t = \sqsubset_s$ since this will imply that

$$G^{\{t\}} = (\Sigma, \Leftrightarrow_t \cup \prec_t, \sqsubset_t \cup \prec_t)^{\bowtie} = (\Sigma, \Leftrightarrow_s \cup \prec_s, \sqsubset_s \cup \prec_s)^{\bowtie} = G^{\{s\}}.$$

($\Leftrightarrow_t = \Leftrightarrow_s$) Trivially follows from Definition 25.

($\sqsubset_t = \sqsubset_s$) If $\alpha \sqsubset_t \beta$, then by Definitions 23 and 25, $\alpha \sqsubset \beta$. But since $\lhd_s \in ext(G^{\{t\}})$, we have $\alpha \lhd_s^{\frown} \beta$, which implies $pos_s(\alpha) \leqslant pos_s(\beta)$. But since $\alpha \sqsubset_t \beta$, it follows by Definition 25 that $(l(\beta), l(\alpha)) \notin ser \cup inl$. Hence, by Definition 25, $\alpha \sqsubset_s \beta$. Thus,

$$\sqsubset_t \subseteq \sqsubset_s . \tag{B.1}$$

It remains to show that $\sqsubset_s \subseteq \sqsubset_t$. Let $\alpha \sqsubset_s \beta$, and we suppose that $\neg(\alpha \sqsubset_t \beta)$. Since $\alpha \sqsubset_s \beta$, by Definition 25, $pos_s(\alpha) \leqslant pos_s(\beta)$ and $(l(\beta), l(\alpha)) \notin ser \cup inl$. Since we assume $\neg(\alpha \sqsubset_t \beta)$, by Definition 25, we must have $pos_t(\beta) < pos_t(\alpha)$. Hence, by Definitions 23 and 25, $\beta \prec_t \alpha$ and $\beta \prec \alpha$. But since $\lhd_s \in ext(G^{\{t\}})$, we have $\beta \lhd_s \alpha$. So $pos_s(\beta) < pos_s(\alpha)$, a contradiction. Thus, $\sqsubset_s \subseteq \sqsubset_t$. Together with (B.1), we get $\sqsubset_t = \sqsubset_s$.

($\prec_t = \prec_s$) If $\alpha \prec_t \beta$, then by Definitions 23 and 25, $\alpha \prec \beta$ (of $G^{\{t\}}$). But since $\lhd_s \in ext(G^{\{t\}})$, we have $\alpha \lhd_s \beta$, which implies

$$pos_s(\alpha) < pos_s(\beta). \tag{B.2}$$

Since $\alpha \prec_t \beta$, by Definition 25, we have

$$\begin{aligned} &\big(l(\alpha), l(\beta)\big) \notin ser \cup inl \\ &\quad \vee (\alpha, \beta) \in \Leftrightarrow_t \cap \big((\sqsubset_t^*)^{\Cap} \circ \Leftrightarrow_t^{\complement} \circ (\sqsubset_t^*)^{\Cap}\big) \\ &\quad \vee \left( \begin{array}{l} (l(\alpha), l(\beta)) \in ser \\ \quad \wedge\ \exists \delta, \gamma \in \Sigma_t. \left( \begin{array}{c} pos_t(\delta) < pos_t(\gamma) \wedge (l(\delta), l(\gamma)) \notin ser \\ \wedge\ \alpha \sqsubset_t^* \delta \sqsubset_t^* \beta \wedge \alpha \sqsubset_t^* \gamma \sqsubset_t^* \beta \end{array} \right) \end{array} \right). \end{aligned}$$

We want to show that $\alpha \prec_s \beta$. There are three cases to consider:

(a) When $(l(\alpha), l(\beta)) \notin ser \cup inl$, it follows from (B.2) and Definition 25 that $\alpha \prec_s \beta$.

(b) When $(\alpha, \beta) \in \Leftrightarrow_t \cap ((\sqsubset_t^*)^{\Cap} \circ \Leftrightarrow_t^{\complement} \circ (\sqsubset_t^*)^{\Cap})$, then $\alpha \Leftrightarrow_t \beta$ and there are $\delta, \gamma \in \Sigma$ such that $\alpha (\sqsubset_t^*)^{\Cap} \delta \Leftrightarrow_t^{\complement} \gamma (\sqsubset_t^*)^{\Cap} \beta$. Since $\sqsubset_t = \sqsubset_s$ and $\Leftrightarrow_t = \Leftrightarrow_s$, we have $\alpha \Leftrightarrow_s \beta$ and $\alpha (\sqsubset_s^*)^{\Cap} \delta \Leftrightarrow_s^{\complement} \gamma (\sqsubset_s^*)^{\Cap} \beta$. Thus, it follows from (B.2) and Definition 25 that $\alpha \prec_s \beta$.

(c) There remains only the case when $(l(\alpha), l(\beta)) \in ser$ and there are $\delta, \gamma \in \Sigma_t$ such that

$$\left( \begin{array}{c} pos_t(\delta) < pos_t(\gamma) \wedge (l(\delta), l(\gamma)) \notin ser \\ \wedge\ \alpha \sqsubset_t^* \delta \sqsubset_t^* \beta \wedge \alpha \sqsubset_t^* \gamma \sqsubset_t^* \beta \end{array} \right).$$

Since $\sqsubset_t = \sqsubset_s$, we also have $\alpha \sqsubset_s^* \delta \sqsubset_s^* \beta \wedge \alpha \sqsubset_s^* \gamma \sqsubset_s^* \beta$. Since $(l(\delta), l(\gamma)) \notin ser$, we either have $(l(\delta), l(\gamma)) \in inl$ or $(l(\delta), l(\gamma)) \notin ser \cup inl$.

- If $(l(\delta), l(\gamma)) \in inl$, then $pos_s(\delta) \neq pos_s(\gamma)$. Thus, $(pos_s(\delta) < pos_s(\gamma) \wedge (l(\delta), l(\gamma)) \notin ser)$ or $(pos_s(\gamma) < pos_s(\delta) \wedge (l(\gamma), l(\delta)) \notin ser)$. So it follows from (B.2) and Definition 25 that $\alpha \prec_s \beta$.
- If $(l(\delta), l(\gamma)) \notin inl$, then $(l(\delta), l(\gamma)) \notin ser \cup inl$. Hence, by Definition 25, $\delta \prec_t \gamma$, which by Definitions 23 and 25, $\delta \prec \gamma$. But since $\lhd_s \in ext(G^{\{t\}})$, we have $\delta \lhd_s \gamma$, which implies $pos_s(\delta) < pos_s(\gamma)$. Since $pos_s(\delta) < pos_s(\gamma)$ and $(l(\delta), l(\gamma)) \notin ser$, it follows from (B.2) and Definition 25 that $\alpha \prec_s \beta$.

Thus, we have shown that $\alpha \prec_s \beta$. Hence,

$$\prec_t \subseteq \prec_s . \tag{B.3}$$

It remains to show that $\prec_s \subseteq \prec_t$. Let $\alpha \prec_s \beta$. Suppose that $\neg(\alpha \prec_t \beta)$. Since $\alpha \prec_s \beta$, by Definition 25, we need to consider three cases:

(a) When $(l(\alpha), l(\beta)) \notin ser \cup inl$, we suppose that $\neg(\alpha \prec_t \beta)$. This by Definition 25 implies that $pos_t(\beta) \leqslant pos_t(\alpha)$. By Definitions 23 and 25, it follows that $\beta \sqsubset_t \alpha$ and $\beta \sqsubset \alpha$. But since $\lhd_s \in ext(G^{\{t\}})$, we have $\beta \lhd_s^{\frown} \alpha$, which implies $pos_s(\beta) \leqslant pos_s(\alpha)$, a contradiction.

(b) If $(\alpha, \beta) \in \diamond_s \cap ((\sqsubset_s^*)^{\Cap} \circ \diamond_s^{\complement} \circ (\sqsubset_s^*)^{\Cap})$, then since $\diamond_s = \diamond_t$ and $\sqsubset_s = \sqsubset_t$, we have $(\alpha, \beta) \in \diamond_t \cap ((\sqsubset_t^*)^{\Cap} \circ \diamond_t^{\complement} \circ (\sqsubset_t^*)^{\Cap})$. Since $\alpha \diamond_t \beta$, we have $pos_t(\alpha) < pos_t(\beta)$ or $pos_t(\beta) < pos_t(\alpha)$. We claim that $pos_t(\alpha) < pos_t(\beta)$. Suppose for a contradict that $pos_t(\beta) < pos_t(\alpha)$. Since $(\alpha, \beta) \in \diamond_t \cap ((\sqsubset_t^*)^{\Cap} \circ \diamond_t^{\complement} \circ (\sqsubset_t^*)^{\Cap})$ and $\diamond_t$ is symmetric, we have $(\beta, \alpha) \in \diamond_t \cap ((\sqsubset_t^*)^{\Cap} \circ \diamond_t^{\complement} \circ (\sqsubset_t^*)^{\Cap})$. Hence, it follows from Definitions 23 and 25 that $\beta \prec_t \alpha$ and $\beta \prec \alpha$. But since $\lhd_s \in ext(G^{\{t\}})$, we have $\beta \lhd_s \alpha$, which implies $pos_s(\beta) < pos_s(\alpha)$, a contradiction. Thus, $pos_t(\alpha) < pos_t(\beta)$.
Since $(\alpha, \beta) \in \diamond_t \cap ((\sqsubset_t^*)^{\Cap} \circ \diamond_t^{\complement} \circ (\sqsubset_t^*)^{\Cap})$, we get $\alpha \prec_t \beta$.

(c) There remains only the case when $(l(\alpha), l(\beta)) \in ser$ and there are $\delta, \gamma \in \Sigma_s$ such that

$$\begin{pmatrix} pos_s(\delta) < pos_s(\gamma) \wedge (l(\delta), l(\gamma)) \notin ser \\ \wedge\ \alpha \sqsubset_s^* \delta \sqsubset_s^* \beta \wedge \alpha \sqsubset_s^* \gamma \sqsubset_s^* \beta \end{pmatrix}.$$

Since $\sqsubset_s = \sqsubset_t$, we have $\alpha \sqsubset_t^* \delta \sqsubset_t^* \beta$ and $\alpha \sqsubset_t^* \gamma \sqsubset_t^* \beta$, which by Definition 25 and transitivity of $\leqslant$ implies that $pos_t(\alpha) \leqslant pos_t(\delta) \leqslant pos_t(\beta)$ and $pos_t(\alpha) \leqslant pos_t(\gamma) \leqslant pos_t(\beta)$. Since $(l(\delta), l(\gamma)) \notin ser$, we either have $(l(\delta), l(\gamma)) \in inl$ or $(l(\delta), l(\gamma)) \notin ser \cup inl$.

(i) If $(l(\delta), l(\gamma)) \in inl$, then $pos_t(\delta) \neq pos_t(\gamma)$. This implies that $(pos_t(\delta) < pos_t(\gamma) \wedge (l(\delta), l(\gamma)) \notin ser)$ or $(pos_t(\gamma) < pos_t(\delta) \wedge (l(\gamma), l(\delta)) \notin ser)$. Since $pos_t(\delta) \neq pos_t(\gamma)$ and $pos_t(\alpha) \leqslant pos_t(\delta) \leqslant pos_t(\beta)$ and $pos_t(\alpha) \leqslant pos_t(\gamma) \leqslant pos_t(\beta)$, we also have $pos_t(\alpha) < pos_t(\beta)$. So it follows from Definition 25 that $\alpha \prec_t \beta$.

(ii) If $(l(\delta), l(\gamma)) \notin inl$, then $(l(\delta), l(\gamma)) \notin ser \cup inl$. We want to show that $pos_t(\delta) < pos_t(\gamma)$. Suppose that $pos_s(\delta) \geqslant pos_s(\gamma)$. Since $(l(\delta), l(\gamma)) \notin ser \cup inl$, by Definitions 23 and 25, we have $\gamma \sqsubset_t \delta$ and $\gamma \sqsubset \delta$. But since $\lhd_s \in ext(G\{t\})$, we have $\gamma \lhd_s^{\frown} \delta$, which implies $pos_s(\gamma) \leqslant pos_s(\delta)$, a contradiction. Since $pos_t(\delta) < pos_t(\gamma)$ and $pos_t(\alpha) \leqslant pos_t(\delta) \leqslant pos_t(\beta)$ and $pos_t(\alpha) \leqslant pos_t(\gamma) \leqslant pos_t(\beta)$, we have $pos_t(\alpha) < pos_t(\beta)$. Hence, we have $pos_t(\alpha) < pos_t(\beta)$ and

$$\begin{pmatrix} pos_t(\delta) < pos_t(\gamma) \wedge (l(\delta), l(\gamma)) \notin ser \cup inl \\ \wedge\ \alpha \sqsubset_t^* \delta \sqsubset_t^* \beta \wedge \alpha \sqsubset_t^* \gamma \sqsubset_t^* \beta \end{pmatrix}.$$

So it follows that $\alpha \prec_t \beta$ by Definition 25.

Thus, we have shown $\prec_s \subseteq \prec_t$. This and (B.3) imply $\prec_t = \prec_s$. $\square$

## References


[1] B. Bloom, M.Z. Kwiatkowska, Trade-offs in true concurrency: Pomsets and Mazurkiewicz traces, in: Proc. of MFPS, 1991, pp. 350–375.
[2] P. Cartier, D. Foata, Problèmes combinatoires de commutation et ré-arrangements, Lecture Notes in Math., vol. 85, Springer, 1969.
[3] R. Devillers, R. Janicki, M. Koutny, P.E. Lauer, Concurrent and maximally concurrent evolution of non-sequential systems, Theoret. Comput. Sci. 43 (1986) 213–238.
[4] V. Diekert, G. Rozenberg (Eds.), The Book of Traces, World Scientific, 1995.
[5] H. Gaifman, V. Pratt, Partial order models of concurrency and the computation of function, in: Proc. of LICS'87, pp. 72–85.
[6] G. Guo, R. Janicki, Modelling concurrent behaviours by commutativity and weak causality relations, in: Proc. of AMAST'02, in: Lecture Notes in Comput. Sci., vol. 2422, 2002, pp. 178–191.
[7] C.A.R. Hoare, Communicating Sequential Processes, Prentice Hall, 1986.
[8] R. Janicki, Relational structures model of concurrency, Acta Inform. 45 (4) (2008) 279–320.
[9] R. Janicki, M. Koutny, Invariants and paradigms of concurrency theory, in: Lecture Notes in Comput. Sci., vol. 506, Springer, 1991, pp. 59–74.
[10] R. Janicki, M. Koutny, Structure of concurrency, Theoret. Comput. Sci. 112 (1) (1993) 5–52.
[11] R. Janicki, M. Koutny, Semantics of inhibitor nets, Inform. and Comput. 123 (1) (1995) 1–16.
[12] R. Janicki, M. Koutny, Fundamentals of modelling concurrency using discrete relational structures, Acta Inform. 34 (1997) 367–388.
[13] R. Janicki, M. Koutny, On causality semantics of nets with priorities, Fund. Inform. 34 (1999) 222–255.
[14] R. Janicki, P.E. Lauer, Specification and Analysis of Concurrent Systems: The COSY Approach, Springer, 1992.
[15] R. Janicki, D.T.M. Lê, Modelling concurrency with quotient monoids, in: Proc. of PETRI NETS 2008, in: Lecture Notes in Comput. Sci., vol. 5062, Springer, 2008, pp. 251–269.
[16] G. Juhás, R. Lorenz, C. Neumair, Synthesis of controlled behaviours with modules of signal nets, in: Lecture Notes in Comput. Sci., vol. 3099, Springer, 2004, pp. 233–257.
[17] G. Juhás, R. Lorenz, S. Mauser, Causal semantics of algebraic Petri Nets distinguishing concurrency and synchronicity, Fund. Inform. 86 (3) (2008) 255–298.
[18] G. Juhás, R. Lorenz, S. Mauser, Complete semantics of algebraic Petri Nets, Fund. Inform. 87 (3) (2008) 331–365.
[19] H.C.M. Kleijn, M. Koutny, Process semantics of general inhibitor nets, Inform. and Comput. 190 (2004) 18–69.
[20] J. Kleijn, M. Koutny, Formal languages and concurrent behaviour, Stud. Comput. Intell. 113 (2008) 125–182.
[21] D.T.M. Lê, Studies in comtrace monoids, Master thesis, Dept. of Computing and Software, McMaster University, Hamilton, ON, Canada, August 2008.

[22] Dai Tri Man Lê, A characterization of combined traces using labeled stratified order structures, in: Proceedings of Petri Net 2010, in: Lecture Notes in Comput. Sci., vol. 6128, Springer, 2010, pp. 104–124.
[23] A. Mazurkiewicz, Concurrent program schemes and their interpretation, TR DAIMI PB-78, Comp. Science Depart., Aarhus University, 1977.
[24] A. Mazurkiewicz, Introduction to trace theory, in [4], pp. 3–42.
[25] E. Ochmański, Recognizable trace languages, in [4], pp. 167–204.
[26] M. Pietkiewicz-Koutny, The synthesis problem for elementary net systems, Fund. Inform. 40 (2, 3) (1999) 310–327.
[27] M.W. Shields, Adequate path expressions, in: Lecture Notes in Comput. Sci., vol. 70, Springer, 1979, pp. 249–265.
[28] E. Szpilrajn, Sur l'extension de l'ordre partiel, Fund. Math. 16 (1930) 386–389.
[29] W. Vogler, A generalization of trace theory, RAIRO Inform. Théor. Appl. 25 (2) (1991) 147–156.
[30] W. Zielonka, Notes on finite asynchronous automata, RAIRO Inform. Théor. Appl. 21 (1987) 99–135.