Modelling Concurrency with Comtraces and Generalized Comtraces

Ryszard Janicki∗,1

Department of Computing and Software, McMaster University, Hamilton, ON, L8S 4K1 Canada

Dai Tri Man Lê2

Department of Computer Science, University of Toronto, Toronto, ON, M5S 3G4 Canada

Abstract

Comtraces (combined traces) are extensions of Mazurkiewicz traces that can model the “not later than” relationship. In this paper, we first introduce the novel concept of generalized comtraces, extensions of comtraces that can additionally model the “non-simultaneously” relationship. Then we study some basic algebraic properties and canonical forms of both comtraces and generalized comtraces. Finally we analyze the relationship between (generalized) comtraces and (generalized) stratified order structures in detail. The major technical contributions of this paper are the results showing that generalized comtraces and generalized stratified order structures can uniquely represent one another.

Key words: generalized trace theory, trace monoids, step sequences, stratified partial orders, stratified order structures, canonical representations

1. Introduction

Mazurkiewicz traces, or just traces3, are quotient monoids over sequences (or words)2. The theory of traces has been utilized to tackle problems from quite diverse areas including combinatorics, graph theory, algebra, logic and especially concurrency theory5.

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. 9, 11), and thus cannot be modelled in terms of traces. For example, neither traces nor partial orders can model the “not later than” relationship11. 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 \( x = \{ \{ a \} \{ b \}, \{ a, b \} \} \); where step \( \{ a, b \} \) denotes the simultaneous performance of \( a \) and \( b \) and the step sequence \( \{ a \} \{ b \} \) denotes the execution of \( a \) followed by \( b \). But the set \( x \) 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 those limitations the concept of a comtrace (combined trace) was introduced in [12]. Comtraces are finite sets of equivalent step sequences and the congruence is determined by a relation \( \text{ser} \), which is called serializability and in general is not symmetric. Monoid generators are ‘steps’, i.e., finite sets, so they have some internal structure that can be used to define equations that generate the quotient monoid. Set union is used to define comtrace congruence. Comtraces provide a formal language counterpart to stratified order structures (so-structures) and were used to provide a semantics of Petri nets with inhibitor arcs. The paper [12] contains a major result showing that every comtrace uniquely determines a so-structure, yet contains very little algebraic theory of comtraces, and the reciprocal relationship, how a finite so-structure determines an appropriate comtrace, is not discussed at all. We will discuss this reciprocal relationship in detail as well as a formal relationship between traces and comtraces.

A so-structure [6, 10, 12, 13] is a triple \( (X, \prec, \sqsubseteq) \), where \( \prec \) and \( \sqsubseteq \) are binary relations on \( X \). They were invented to model both the “earlier than” (the relation \( \prec \)) and the “not later than” (the relation \( \sqsubseteq \)) relationships, under the 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 [12, 18, 17, 20] and others). However, so far comtraces are used much 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 theory of quotient monoids for comtraces analogous to that of traces.

However, neither comtraces nor so-structures can adequately 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 \([11, 13]\) 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., all of these step sequences \( \{ a \} \{ b \}, \{ b \} \{ a \} \) and \( \{ a, b \} \) are 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 in [7, 9]. A gso-structure is a triple \( (X, \ll, \sqsubseteq) \), where \( \ll \) and \( \sqsubseteq \) are binary relations on \( X \) modelling the “non-simultaneously” and the “not later than” relationships respectively under the assumption that all system runs are modelled by stratified partial orders. In this paper, we propose a language counterpart of gso-structures, called generalized comtraces (g-comtraces). We will analyze in detail the properties of g-comtraces, their canonical representations, and most importantly the formal relationship between g-comtraces and gso-structures.

This paper is the expansion and revision of our results from [16, 22]. 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 [6, 10, 12, 13, 7, 9]. Comtraces are defined and their relationship to traces is discussed in Section 5 and the 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.
A few properties of both comtrace and g-comtrace languages are presented in Section 9. In Section 10 we discuss both the so-structures defined by comtraces and the comtraces generated by so-structures. The gso-structures generated by g-comtraces are defined and analyzed in Section 11 and the g-comtraces generated by gso-structures are defined and analyzed in Section 12. Concluding remarks are made in Section 13. We also include two Appendixes containing some long and technical proofs of results from Section 11.

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

Let $X$ be a set. The powerset of $X$ will be denoted by $\mathcal{P}(X)$, i.e., $\mathcal{P}(X) \overset{df}{=} \{ Y \mid Y \subseteq X \}$, and the set of all non-empty subsets of $X$, i.e., $\mathcal{P}(X) \setminus \{ \emptyset \}$ will be denoted by $\mathcal{P}(X)$. We let $id_X$ denote the identity relation on a set $X$, i.e., $id_X \overset{df}{=} \{(x, x) \mid x \in X\}$. If $R$ and $S$ are binary relations on a set $X$ (i.e., $R, S \subseteq X \times X$), then their composition $R \circ S$ is defined as $R \circ S \overset{df}{=} \{(x, y) \in X \times X \mid \exists z \in Z. (x, z) \in R \land (z, y) \in S\}$. We also define

$$R^i \overset{df}{=} id_X, \quad R^* \overset{df}{=} R^{-1} \circ R \quad \text{(for } i \geq 1\text{)} \quad R^+ \overset{df}{=} \bigcup_{i \geq 1} P^i \quad R^\ast \overset{df}{=} \bigcup_{i \geq 0} P^i$$

The relations $R^+$ and $R^\ast$ are called the (irreflective) transitive closure and reflexive transitive closure of $R$ respectively.

A binary relation $R \subseteq X \times X$ is an equivalence relation on $X$ if and only if (iff) the following must hold for all $a, b, c \in X$: $a R a$ (reflexive), $a R b \implies b R a$ (symmetric) and $a R b R c \implies a R c$ (transitive). If $R$ is an equivalence relation, then for every $x \in X$, the set $[x]_R \overset{df}{=} \{ y \mid y R x \land y \in X \}$ is the equivalence class of $x$ with respect to $R$. We also define $X/R \overset{df}{=} \{ [x]_R \mid x \in X \}$, i.e., the set of all equivalence classes of $X$ with respect to $R$. 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 (strict) partial order iff for all $a, b, c \in X$, we have: $\neg (a \prec a)$ (irreflective) and $a \prec b \prec c \implies a \prec c$ (transitive). The pair $(X, \prec)$ in this case is called a partially ordered set (also called a poset), i.e., the set $X$ is partially ordered by the relation $\prec$. The pair $(X, \prec)$ is called a finite partially ordered set (finite poset) if $X$ is finite.

Given a poset $(X, \prec)$, we define the binary relations $\preceq, \prec, \succeq \subseteq X \times X$ as follows:

$$a \prec b \overset{df}{\iff} \neg (a \prec b) \land \neg (b \prec a) \land a \neq b$$

$$a \preceq b \overset{df}{\iff} a \prec b \lor a \prec b$$

$$a \succeq b \overset{df}{\iff} a = b \lor a \prec b$$
In other words, we write \( a \succeq b \) if \( a \) and \( b \) are distinctly incomparable elements of \( X \) w.r.t. the partial order \( \prec \); we write \( a \prec b \) if \( a \) and \( b \) are distinct and \( \lnot (b \prec a) \). The relation \( \succeq \) was introduced to make some formulations shorter.

A poset \((X, \prec)\) is

- **total** (or **linear**) iff \( \prec \) is empty, i.e., for all \( a, b \in X \), either \( a \prec b \), or \( b \prec a \), or \( a = b \).

- **stratified** (or **weak**) iff \( \simeq \) 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

\[
\text{Total}_X(\prec) \overset{df}{=} \{ \triangleleft \subseteq X \times X \mid \triangleleft \text{ is a total extension of } \prec \} \\
\text{Strat}_X(\prec) \overset{df}{=} \{ \triangleleft \subseteq X \times X \mid \triangleleft \text{ is a stratified extension of } \prec \}
\]

**Theorem 2.1 (Szpiroin’s Theorem [23]).** *For every poset \((X, \prec)\), \( \prec = \bigcap_{\triangleleft \in \text{Total}_X(\prec)} \triangleleft \).* \(\square\)

Szpiroin’s Theorem states that every partial order is uniquely determined by the intersection of all of its total extensions. The same is also true for stratified extensions.

**Corollary 2.1.** *For every poset \((X, \prec)\), \( \prec = \bigcap_{\triangleleft \in \text{Strat}_X(\prec)} \triangleleft \).*  

**Proof.** Since for each \( \triangleleft \in \text{Strat}_X(\prec) \) we have \( \prec \subseteq \triangleleft \), then \( \prec \subseteq \bigcap_{\triangleleft \in \text{Strat}_X(\prec)} \triangleleft \). Since each total order is a stratified order, it follows that \( \text{Total}_X(\prec) \subseteq \text{Strat}_X(\prec) \). Thus, we have \( \bigcap_{\triangleleft \in \text{Strat}_X(\prec)} \triangleleft \subseteq \bigcap_{\triangleleft \in \text{Total}_X(\prec)} \triangleleft = \prec \). \(\square\)

### 2.2. Monoids and Equational Monoids

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

A nonempty equivalence relation \( \sim \subseteq X \times X \) is a **congruence** in the monoid \((X, *, 1)\) if for all \( a_1, a_2, b_1, b_2 \in X \), \( a_1 \sim b_1 \land a_2 \sim b_2 \Rightarrow (a_1 * a_2) \sim (b_1 * b_2) \).

The triple \((X / \sim, *, [1])\), where \([a] * [b] = [a * b]\), 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 \( \sim \) (for more details see for example [3]). The symbols \( * \) and \( \oplus \) are often omitted if this does not lead to any discrepancy.

**Definition 2.1 (Equation monoid (cf. [16, 25])).** Let \( M = (X, *, 1) \) be a monoid and let \( EQ = \{ x_i = y_i \mid i = 1, \ldots, n \} \) be a finite set of equations. Define \( \equiv_{EQ} \) to be the least congruence on \( M \) satisfying, \( x_i = y_i \Rightarrow x_i \equiv_{EQ} y_i \), for every equation \( x_i = y_i \in EQ \). We call the relation \( \equiv_{EQ} \) the congruence defined by \( EQ \), or \( EQ \)-congruence.

The **quotient monoid** \( M_{\equiv_{EQ}} = (X / \equiv_{EQ}, *, [1]) \), where \([x] * [y] = [x * y]\), is called an **equational monoid**. \(\blacksquare\)
The following folklore result shows that the relation \(\equiv_{EQ}\) can also be uniquely defined in an explicit way.

**Proposition 2.1.** For equational monoids, we have the \(EQ\)-congruence \(\equiv = (\sim \cup \sim^{-1})^*\), where the relation \(\sim \subseteq X \times X\) is defined as:

\[
x \sim y \iff \exists x_1, x_2 \in X. (u \equiv w) \in EQ, x = x_1 * u * x_2 \wedge y = x_1 * w * x_2.
\]

**Proof.** Define \(\equiv = \sim \cup \sim^{-1}\). Clearly \((\equiv)^*\) is an equivalence relation. Let \(x_1 \equiv y_1\) and \(x_2 \equiv y_2\). This means \(x_1 (\equiv)^k y_1\) and \(x_2 (\equiv)^l y_2\) for some \(k, l \geq 0\). Hence, \(x_1 * x_2 \equiv y_1 * y_2\), i.e., \(x_1 * x_2 \equiv y_1 * y_2\). Thus, \(\equiv\) is a congruence. Let \(\sim\) be a congruence satisfying for all \((u \equiv w) \in EQ\), \(u \sim w\). Clearly we have \(x \sim y \implies x \sim y\). Hence, \(x \equiv y \iff x (\equiv)^m y \implies x \sim^m y \implies x \sim y\). Thus, the congruence \(\equiv\) is the least. \(\square\)

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 \(\Sigma\), let \(\lambda\) 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 \(\cdot\) denote the sequence concatenation operator (usually omitted). Since the sequence concatenation operator is associative, the triple \((\Sigma^*, \cdot, \lambda)\) is a **monoid** (of sequences).

Consider an alphabet \(S \subseteq \mathcal{P}(X)\) for some finite \(X\). The elements of \(S\) are called **steps** and the elements of \(S^*\) are called **step sequences**. For example if \(S = \left\{\{a,b,c\}, \{a,b\}, \{a\}, \{c\}\right\}\) then \(\{a,b\}\{c\}\{a,b,c\} \in S^*\) is a step sequence. The triple \((S^*, \cdot, \lambda)\), where \(\cdot^*\) denotes the step sequence concatenation operator (usually omitted), is a **monoid** (of step sequences), since the step sequence operator is also associative.

We will now show the formal relationship between step sequences and stratified orders. Let \(t = A_1 \ldots A_k\) be a step sequence. We define \(|t|_a\), the number of occurrences of an event \(a\) in \(t\), as \(|t|_a = |\left\{A_i \mid 1 \leq i \leq k \wedge a \in A_i\right\}|\), where \(|X|\) denotes the cardinality of the set \(X\). Then:

- We can uniquely construct its **enumerated step sequence** \(\overline{t}\) as
  \[
  \overline{t} \equiv \overline{A_1} \ldots \overline{A_k}, \text{ where } \overline{A_i} \equiv \left\{e^{l[A_1 \ldots A_i \ldots e^{l+1}]} \mid e \in A_i\right\}.
  \]
  We will call such \(\alpha = e^{l[A_1 \ldots A_i \ldots e^{l+1}]}\) an **event occurrence of** \(e\). For each event occurrence \(\alpha = e^{l[A_1]}, \text{ let } l(\alpha)\) denote the label of \(\alpha\), i.e., \(l(\alpha) = l(e^{l[A_1]}) = e\). For instance, if \(u = \{a,b\}\{b,c\}\{c,a\}\{a\}\), then \(u = \{a^{(1)}, b^{(1)}\}\{b^{(2)}, c^{(1)}\}\{a^{(2)}, c^{(2)}\}\{a^{(3)}\}\). Conversely, from an enumerated step sequence \(\overline{t} = \overline{A_1} \ldots \overline{A_k}\), we can also uniquely reconstruct its step sequence \(t = l[\overline{A_1}] \ldots l[\overline{A_k}]\).

- We 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 \(t = \{a,b\}\{b,c\}\{c,a\}\{a\}\), \(\Sigma_t = \{a^{(1)}, a^{(2)}, a^{(3)}\}\{b^{(1)}, b^{(2)}\}\{c^{(1)}, c^{(2)}\}\).

- For each \(\alpha \in \Sigma_t\), we let \(pos_t(\alpha)\) denote the consecutive number of a step where \(\alpha\) belongs, i.e., if \(\alpha \in A_j\) 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 $<_u, \simeq_u \subseteq \Sigma_u \times \Sigma_u$ as:

\[ \alpha <_u \beta \iff \text{pos}_u(\alpha) < \text{pos}_u(\beta) \quad \text{and} \quad \alpha \simeq_u \beta \iff \text{pos}_u(\alpha) = \text{pos}_u(\beta). \]

Since $<_{\Sigma_u} = <_u \cup \sim_u$, we have $\alpha <_{\Sigma_u} \beta \iff (\alpha \neq \beta \land \text{pos}_u(\alpha) \leq \text{pos}_u(\beta))$. Note that $<_u$ is a stratified order iff $\simeq_u$ is an equivalence relation on $\Sigma_u$. The two propositions below are known folklore results (which are rarely formally proved). We will provide proofs to make the paper self-sufficient. The first proposition shows that $<_u$ is indeed a stratified order.

**Proposition 2.2.** Given a step sequence $u = B_1 \ldots B_n$, the relation $\simeq_u$ is an equivalence relation.

**Proof.** Since $\alpha, \beta \in \Sigma_u$, it follows that $\alpha, \beta \in \overline{B_i}$ for some $1 \leq i \leq n$. Hence, $\sim_u$ is an equivalence relation induced by the partitions $\overline{B_1}, \ldots, \overline{B_n}$ of $\Sigma_u$. □

The stratified order $<_u$ is an order generated by the step sequence $u$.

Conversely, let $<$ be a stratified order on a set $\Sigma$. The set $\Sigma$ can be represented as a sequence of equivalence classes $\Omega_\prec = B_1 \ldots B_k (k \geq 0)$ such that

\[ \prec = \bigcup_{i<j} (B_i \times B_j) \quad \text{and} \quad \simeq_\prec = \bigcup_i (B_i \times B_i). \]

The sequence $\Omega_\prec$ is a step sequence representing $\prec$.

The correctness of the existence of $\Omega_\prec$ is shown by the next folklore proposition.

**Proposition 2.3.** If $\prec$ is a stratified order on a set $\Sigma$ and $A, B$ are two distinct equivalence classes of $\simeq_\prec$, then either $A \times B \subseteq \prec$ or $B \times A \subseteq \prec$.

**Proof.** Since both equivalence classes $A$ and $B$ are non-empty, we let $a \in A$ and $b \in B$. Clearly, $a <_\prec b$ or $b <_\prec a$; otherwise, $a \sim_\prec b$, which contradicts that $a, b$ are elements from two distinct equivalence classes. There are two cases:

1. If $a < b$: we want to show $A \times B \subseteq \prec$. Let $c \in A$ and $d \in B$, it suffices to show $c < d$. Assume for contradiction that $\neg(c < d)$. Since $c \not\in_\prec d$, it follows that $d < c$. There are three subcases:
   (a) If $a = c$, then $d < a$ and $a < b$. Hence, $d < b$. This contradicts that $b, d \in B$.
   (b) If $b = d$, then $b < c$ and $a < b$. Hence, $a < c$. This contradicts that $a, c \in A$.
   (c) If $a \neq c$ and $b \neq d$, then $a \sim_\prec c$ and $b \sim_\prec d$ and $\neg(a \sim_\prec d)$ and $\neg(c \sim_\prec b)$. Since $\neg(a \sim_\prec d)$, either $a <_\prec d$ or $d <_\prec a$.
      - If $a <_\prec d$: since $d <_\prec c$, it follows $a <_\prec c$. This contradicts $a \sim_\prec c$.
      - If $d <_\prec a$: since $a <_\prec b$, it follows $d <_\prec b$. This contradicts $d \sim_\prec b$.

   Therefore, we conclude $A \times B \subseteq \prec$.

2. If $b < a$: using a dual argument to (1), we can show that $B \times A \subseteq \prec$. □

The idea of Proposition 2.3 is that if we define a relation $\prec$ on the set of equivalence classes $\{B_1, \ldots, B_n\}$, then $\prec$ is a total order on $\{B_1, \ldots, B_n\}$. Hence, Propositions 2.2 and 2.3 are fundamental for understanding the equivalence of stratified partial orders and step sequences.

Note that since sequences are special cases of step sequences (step sequences of singletons) and total orders are special cases 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 $<_x$ denote the total order generated by $x$, and for every total order $\prec$, we let $\Omega_\prec$ denote the sequence generating $\prec$. 

6
3. Traces vs. Partial Orders

Traces or partially commutative monoids \([2, 3, 23, 24]\) 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 will discuss the relationship between traces and finite partial orders.

The theory of traces has been utilized to tackle problems from quite diverse areas including combinatorics, graph theory, algebra, logic and, especially (due to the relationship to partial orders) concurrency theory \([5, 23, 24]\).

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

**Definition 3.1** \([5, 24]\). Let \(M = (E^*, \ast, \lambda)\) be a free monoid generated by \(E\), and let the relation \(\text{ind} \subseteq E \times E\) be an irreflexive and symmetric relation (called independency or commutation), and \(EQ \overset{df}{=} \{ab = ba \mid (a,b) \in \text{ind}\}\). Let \(\equiv\text{ind}\), called trace congruence, be the congruence defined by \(EQ\). Then the equational monoid \(M_{\equiv\text{ind}} = (E^*/\equiv\text{ind}, \ast, \lambda)\) is a monoid of traces (or a free partially commutative monoid). The pair \((E, \text{ind})\) is called a trace alphabet. ■

We will omit the subscript \(\text{ind}\) from trace congruence and write \(\equiv\) if it causes no ambiguity.

**Example 3.1.** Let \(E = \{a, b, c\}\), \(\text{ind} = \{(b,c),(c,b)\}\), i.e., \(EQ = \{bc = cb\}\). For example, \(abcbca \equiv cbbcba\) (since \(abcbca \approx cbbcba \approx accbba\)). Also we have \(t_1 = [abcbca] = \{abcba,abcaba,acacbca,acbcbca,abbcba,abcba,acbcbca\}\), \(t_2 = [abc] = \{abc,acb\}\) and \(t_3 = [bca] = \{bca,aca\}\) are traces. Note that \(t_1 = t_2 \odot t_3\) since \([abcbca] = [abc] \odot [bca]\). ■

Each trace can be interpreted as a finite partial order. Let \(t = \{x_1, \ldots, x_k\}\) be a trace, and let \(\prec_{x_i}\) denotes the total order induced by the sequence \(x_i\), \(i = 1, \ldots, k\). Note that \(\Sigma_i = \Sigma_{x_i}\) for all \(i, j = 1, \ldots, n\), so we can define \(\Sigma = \bigcup_{i=1}^{k} \Sigma_i\). For example for \(t_1\) from Example 3.1 we have \(\Sigma_i = \{a^{(1)}, b^{(1)}, c^{(1)}, a^{(2)}, b^{(2)}, c^{(2)}\}\). Clearly \(\prec_{t_1} \subseteq \Sigma_1 \times \Sigma_2\). The partial order generated by \(t\) can then be defined as \(\prec_t = \bigcap_{i=1}^{k} \prec_{x_i}\). In fact, the set \(\{\prec_{t_1}, \ldots, \prec_{t_k}\}\) consists of all the total extensions of \(\prec_t\) (see \([23, 24]\)).

For example, the trace \(t_1 = [abcbca]\) from Example 3.1 can be interpreted as a partial order \(\prec_{t_1}\) depicted in the following diagram (arcs inferred from transitivity are omitted for simplicity):

\[
\begin{array}{c}
\node{a^{(1)}} \node{b^{(1)}} \node{b^{(2)}} \\
\node{c^{(1)}} \node{a^{(2)}} \node{c^{(2)}}
\end{array}
\]

**Remark 3.1.** 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 \((\Sigma_{[s]}, \prec_{[s]}\)) on the sequence \(s\) and \((x, y) \notin \text{ind}\). 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. \([5]\)). 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.

Conversely, each finite partial order can be represented by a trace as follows. Given a finite set $X$, let $(X, \prec)$ be a poset and let $\{\prec_1, \ldots, \prec_k\}$ be the set of all total extensions of $\prec$. Let $x_i \in X^*$ be a sequence that represents $\prec_i$, for $i = 1, \ldots, k$. Then the set $t_\prec = \{x_1, \ldots, x_k\}$ is a trace over the trace alphabet $(X, \preceq)$, i.e., $t_\prec = [x_i]/\equiv_{\prec}$ for any $i = 1, \ldots, k$.

From the concurrency point of view, the fundamental advantage of traces is that in many cases it is simpler and more fruitful to analyze sequences equipped with an independency relation $\text{ind}$ and their underlying quotient trace monoid than their equivalent partial order representations. This is especially the case when we want to study the formal linguistic aspects of concurrent behaviors, e.g., Ochmanski’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 [5].

4. Relational Structures Model of Concurrency

Even though partial orders are a principle tool for modelling “true concurrency,” they have some limitations. While they can sufficiently model the “earlier than” relationship, they cannot model neither the “not later than” relationship nor the “non-simultaneously” relationship. It was shown in [11] 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 [7, 9, 13] (see [9] for a detailed bibliography and history).

In this chapter, we review the theory of stratified order structures of [13] and generalized stratified order structures of [7, 9]. 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 [11] 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 sophisticated sets of stratified observations are used to describe concurrent behaviors, e.g., to model the “not later than” relationship, we need to use appropriate relational structures [11].

4.1. Stratified Order Structure

By a relational structure, we will 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 4.1 (Stratified order structure [13]). A stratified order structure (so-structure) is a relational structure \( S = (X, \prec, \sqsubseteq) \), such that for all \( a, b, c \in X \), the following hold:

\[
\begin{align*}
S1: & \quad a \not\sqsubseteq a \\
S2: & \quad a \prec b \Rightarrow a \sqsubseteq b \\
S3: & \quad a \sqsubseteq b \sqsubseteq c \wedge a \neq c \Rightarrow a \sqsubseteq c \\
S4: & \quad a \sqsubseteq b \prec c \vee a \prec b \sqsubseteq c \Rightarrow a \prec c
\end{align*}
\]

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 \( a \prec b \Rightarrow b \not\sqsubseteq a \). The relation \( \prec \) is called causality and represents the “earlier than” relationship, and the relation \( \sqsubseteq \) is called weak causality and represents the “not later than” relationship. The axioms C1–C4 model the mutual relationship between “earlier than” and “not later than” relations, provided that the system runs are defined as stratified orders.

The concept of so-structures were independently introduced in [6] and [10] (the axioms are slightly different from S1–S4, although equivalent). Their comprehensive theory has been presented in [13]. They have been successfully applied to model inhibitor and priority systems, asynchronous races, synthesis problems, etc. (see for example [12, 19, 20, 26] and others).

The name partially follows from the following result.

Proposition 4.1 ([11]). For every stratified order \( \prec \) on \( X \), the triple \( S_{\prec\sqsubseteq} = (X, \prec, \sqsubseteq) \) is a so-structure. ■

Definition 4.2 (Stratified extension of a so-structure [13]). Let \( S = (X, \prec, \sqsubseteq) \) be a so-structure. A stratified order \( \prec \) on \( X \) is a stratified extension of \( S \) if for all \( \alpha, \beta \in X \),

\[
\alpha \prec \beta \implies \alpha \prec \beta \quad \text{and} \quad \alpha \sqsubseteq \beta \implies \alpha \prec \beta
\]

The set of all stratified extensions of \( S \) is denoted as \( \text{ext}(S) \). ■

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 4.1 ([13, Theorem 2.9]). Let \( S = (X, \prec, \sqsubseteq) \) be a so-structure. Then

\[
S = \left( X, \bigcap_{\prec \in \text{ext}(S)} \prec, \bigcap_{\sqsubseteq \in \text{ext}(S)} \sqsubseteq \right)
\]

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

**Theorem 4.2 (11).** Let \( S = (X, \prec, \sqsubset) \) be a so-structure. Then for every \( a, b \in X \)
\[
\left( \exists \alpha \in ext(S). a \prec b \right) \land \left( \exists \beta \in ext(S). b \prec a \right) \implies \left( \exists \gamma \in ext(S). a \prec \gamma \prec b \right).
\]

The classification of concurrent behaviors provided in (11) says that a concurrent behavior conforms to the paradigm \( \pi_3 \) if it has the same property as stated in Theorem 4.2 for \( ext(S) \). In other words, Theorem 4.2 states that the set \( ext(S) \) conforms to paradigm \( \pi_3 \) of (11).

### 4.2. Generalized Stratified Order Structure

The stratified order structures can adequately model concurrent histories only when paradigm \( \pi_3 \) is satisfied. For the general case, we need gso-structures introduced in (7) also under the assumption that the system runs are defined as stratified orders.

**Definition 4.3 (Generalized stratified order structure [7, 9]).** A generalized stratified order structure (gso-structure) is a relational structure \( G = (X, \prec, \sqsubset) \) such that \( \sqsubset \) is irreflexive, \( \prec \) is symmetric and irreflexive, and the triple \( S_G = (X, \prec_G, \sqsubset_G) \), where \( \prec_G = \prec \cap \sqsubset \), is a so-structure.

Such relational structure \( S_G \) is called the so-structure induced by \( G \). When \( X \) is finite, \( G \) is called a finite gso-structure.

The relation \( \prec \) 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 \( R^{sym} \) denote the symmetric closure of \( R \) and \( R^{sym} \) can be defined as \( R^{sym} \overset{df}{=} R \cup R^{-1} \).

**Definition 4.4 (Stratified extension of a gso-structure [7, 9]).** Let \( G = (X, \prec, \sqsubset) \) be a gso-structure. A stratified order \( \prec \) on \( X \) is a stratified extension of \( G \) if for all \( \alpha, \beta \in X \),
\[
\alpha \prec \beta \implies \alpha \prec^{sym} \beta \quad \text{and} \quad \alpha \sqsubset \beta \implies \alpha \prec \gamma \sqsubset \beta.
\]
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 following.

**Theorem 4.3 (7, 9).** Let \( G = (X, \prec, \sqsubset) \) be a gso-structure. Then
\[
G = \left( X, \bigcap_{\alpha \in ext(G)} \prec^{sym}, \bigcap_{\beta \in ext(G)} \sqsubset \right).
\]

The gso-structures do not have an equivalence of Theorem 4.2, which makes proving properties about gso-structures more difficult, but they can model the most general concurrent behaviors (provided that observations are stratified orders) [9].

---

*A paradigm is a supposition or statement about the structure of a concurrent behavior (concurrent history) involving a treatment of simultaneity. See [3, 11] for more details.*
4.3. Motivating Example

To understand the main motivation and intuition behind the use of s0-structures and gso-structures, we will consider four simple programs in the following example (from [9]).

Example 4.1. All the programs are written using a mixture of cobegin, coend and a version of concurrent guarded commands.

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 (a1, b1, c1, i = 1, ..., 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 Figure 1. ■

Let obs(Pi) 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 o1 = {a}{b}{c}, o2 = {a}{c}{b}, o3 = {a}{b,c}. Each o1 can be equivalently seen as a stratified partial order o1 = ((a,b,c), ai) where:

We can now write obs(P1) = {o1,o2,o3}, obs(P2) = {o1,o3}, obs(P3) = {o3}, obs(P4) = {o1,o2}. Note that for every i = 1, ..., 4, all runs from the set obs(Pi) yield exactly the same outcome. Hence, each obs(Pi) is called the concurrent history of Pi.

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.
The relationship between modelled by a labelled transition system (automaton) \( \{ \text{execution of ou} \} \prec \text{different types.} \)

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 \( i \).

\[ \text{Program } P_1: \]

In the set \( \text{obs}(P_1) \), 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 \( x \overset{o}{\rightarrow} y \). Hence for \( P_1 \), \( \prec \) is the intersection of \( \alpha_1, \alpha_2 \) and \( \alpha_3 \), and \( \{\alpha_1, \alpha_2, \alpha_3\} \) 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 \( \text{obs}(P_1) \). We will say that \( \text{obs}(P_1) \) and \( \prec \) are tantamount\footnote{Following \cite{9} 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.} and write \( \text{obs}(P_1) \cong \{\prec\} \) or \( \text{obs}(P_1) \cong \{\{(a, b, c)\}, \prec\} \). Having \( \text{obs}(P_1) \) one may construct \( \prec \) (as an intersection), and hence construct \( \text{obs}(P_1) \) (as the set of all stratified extensions). This is a classical case of the “true” concurrency approach, where concurrent behavior is modelled by a causality relation.

Figure 1: Examples of causality, weak causality, and commutativity. Each program \( P_i \) can be modelled by a labelled transition system (automaton) \( A_i \). The set \( \{a, b\} \) denotes the simultaneous execution of \( a \) and \( b \).
5. Comtraces

To deal with $\text{obs}(P_2)$ and $\text{obs}(P_3)$, $\prec$ is insufficient because $o_2 \notin \text{obs}(P_2)$ and $o_1, o_2 \notin \text{obs}(P_2)$. Thus, we need a weak causality relation $\sqsubseteq$ defined in this context as $x \sqsubseteq y$ iff for each run $o$ we have $\neg(y \xrightarrow{a} x)$ ($x$ is never executed after $y$). For our four cases we have

$\sqsubseteq_2 = \{(a, b), (a, c), (b, c)\}$, $\sqsubseteq_4 = \{(a, b), (a, c), (b, c)\}$. Notice again that for $i = 2, 3$, the pair of relations $\langle \prec, \sqsubseteq_i \rangle$ and the set $\text{obs}(P_i)$ are equivalent in the sense that each is definable from the other. (The set $\text{obs}(P_4)$ can be defined as the greatest set $\text{PO}$ of partial orders built from $a, b$, and $c$ satisfying $x \prec y \Rightarrow \forall o \in \text{PO}. x \xrightarrow{a} y$ and $x \sqsubseteq_i y \Rightarrow \forall o \in \text{PO}. \neg(y \xrightarrow{a} x)$.)

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

Program $P_4$:

The causality relation $\prec$ does not model the concurrent behavior of $P_4$ correctly since $o_3$ does not belong to $\text{obs}(P_4)$. The commutativity relation $\ll$ is defined in this context as $x \ll y$ iff for each run $o$ either $x \xrightarrow{a} y$ or $y \xrightarrow{a} x$. For the set $\text{obs}(P_4)$, the relation $\ll_4$ looks like $\ll_4 = \{(a, b), (a, c), (c, a), (b, c), (c, b)\}$. The pair of relations $\langle \ll_4, \prec \rangle$ and the set $\text{obs}(P_4)$ are equivalent in the sense that each is definable from the other. (The set $\text{obs}(P_4)$ is the greatest set $\text{PO}$ of partial orders built from $a, b$, and $c$ satisfying $x \ll y \Rightarrow \forall o \in \text{PO}. x \xrightarrow{a} y \lor y \xrightarrow{a} x$ and $x \prec y \Rightarrow \forall o \in \text{PO}. x \xrightarrow{a} y$.) In other words, $\text{obs}(P_4)$ and $\langle \ll_4, \prec \rangle$ are tantamount, $\text{obs}(P_4) \succ \langle \ll_4, \prec \rangle$, so we may say that in this case the relations $\langle \ll_4, \prec \rangle$ model the concurrent behavior described by $\text{obs}(P_4)$.

Note also that $\ll_1 = \prec \cup \ll_1$ and the pair $\langle \ll_1, \ll_1 \rangle$ also models the concurrent behavior described by $\text{obs}(P_1)$.

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

1. $\text{obs}(P_1)$ can be modelled by the relation $\prec$ alone, and $\text{obs}(P_1) \succ \langle \prec \rangle$.
2. $\text{obs}(P_i)$, for $i = 1, 2, 3$ can also be modelled by appropriate pairs of relations $\langle \prec, \sqsubseteq_i \rangle$, and $\text{obs}(P_i) \succ \langle \prec, \sqsubseteq_i \rangle$.
3. all sets of observations $\text{obs}(P_i)$, for $i = 1, 2, 3, 4$ are modelled by appropriate pairs of relations $\langle \ll_i, \sqsubseteq_i \rangle$, and $\text{obs}(P_i) \succ \langle \ll_i, \sqsubseteq_i \rangle$.

Note that the relations $\prec, \ll, \sqsubseteq$ are not independent, since it can be proved (see [11]) that $\ll = \ll \cap \sqsubseteq$. Intuitively, since $\ll$ and $\sqsubseteq$ 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.

5. Comtraces

The standard definition of a free monoid $(E^*, \ast, \lambda)$ assumes the elements of $E$ have no internal structure (or their internal structure does not affect any monoidal properties), and they are

---

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

13
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 (combined traces), were introduced in [12] 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 5.1 (Comtrace alphabet [12]). Let $E$ be a finite set (of events) and let $ser \subseteq sim \subseteq E \times E$ be two relations called serializability and simultaneity respectively and the relation $sim$ is irreflexive and symmetric. Then the triple $(E, sim, ser)$ is called the 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 [13]), while $(a, b) \in ser$ means that $a$ and $b$ may occur simultaneously or $a$ may occur before $b$ (i.e., both executions are equivalent). We define $S$, the set of all (potential) steps, as the set of all cliques of the graph $(E, sim)$, i.e.,

$$S \equiv \{ A \mid A \neq \emptyset \land \forall a, b \in A. (a = b \lor (a, b) \in sim) \}.$$

Definition 5.2 (Comtrace congruence [12]). Let $\theta = (E, sim, ser)$ be a comtrace alphabet and let $\approx_{ser}$, called comtrace congruence, be the $EQ$-congruence defined by the set of equations

$$EQ \equiv \{ A = BC \mid A = B \cup C \in S \land B \times C \subseteq ser \}.$$

Then the equational monoid $(S^*/\approx_{ser}, \cdot, [\lambda])$ 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 2.1 the comtrace congruence relation can also be defined explicitly in non-equational form as follows.

Proposition 5.1. Let $\theta = (E, sim, ser)$ be a comtrace alphabet and let $S^*$ be the set of all step sequences defined on $\theta$. Let $\approx_{ser} \subseteq S^* \times S^*$ be the relation comprising all pairs $(t, u)$ of step sequences such that $t = wAz$ and $u = wBCz$, where $w, z \in S^*$ and $A, B, C$ are steps satisfying $B \cup C = A$ and $B \times C \subseteq ser$. Then $\approx_{ser} = (\approx_{ser} \cup \approx_{ser}^{-1})^\delta$.

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

Example 5.1. Let $E = \{a, b, c\}$ where $a$, $b$ and $c$ are three atomic operations, where

\[
\begin{align*}
a : & \quad y \leftarrow x + y \\
b : & \quad x \leftarrow y + 2 \\
c : & \quad y \leftarrow y + 1
\end{align*}
\]

Assuming simultaneous reading is allowed, then only $b$ and $c$ can be performed simultaneously, and the simultaneous execution of $b$ and $c$ gives the same outcome as executing $b$ followed by $c$ (see also the program $P_2$ of Example 4.1). We can then define $sim = \{(b, c), (c, b)\}$ and $ser = \{(b, c)\}$, and we have $S = \{\{a\}, \{b\}, \{c\}, \{b, c\}\}$, $EQ = \{\{b, c\} = \{b\}\{c\}\}$. For example, $x = \{(a)\{b, c\}\} = \{(a)\{b, c\}, \{a\}\{b\}\{c\}\}$ is a comtrace. Note that $\{a\}\{c\}\{b\} \notin x$. ■
Proposition 5.2. Note that Lemma 5.1 guarantees that this definition is correct.

Synchrony are very close to that used to deal with simultaneity. Even though traces are quotient monoids over sequences and comtraces are quotient monoids over step sequences, 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 \((E, \text{ind}, \text{ser})\) and \((E, \text{sim}, \text{ser})\) be trace and comtrace alphabets respectively. For each sequence \( x = a_1 \ldots a_n \in E^* \), we define \( x^1 = \{a_1\} \ldots \{a_n\} \) to be its corresponded sequence of singleton sets.

Lemma 5.1 (Relationship between traces and comtraces).

1. If \( \text{ser} = \text{sim} \), then for each comtrace \( t \in \mathcal{S}^*/\equiv_{\text{ser}} \) there is a step sequence \( x = \{a_1\} \ldots \{a_k\} \in \mathcal{S}^* \) such that \( t = [x]_{\equiv_{\text{ser}}} \).
2. If \( \text{ser} = \text{sim} = \text{ind} \), then for each \( x, y \in E^* \), we have \( x \equiv_{\text{ind}} y \iff x^1 \equiv_{\text{ser}} y^1 \).

Proof. 1. Let \( t = [A_1 \ldots A_m] \). Then \( t = [A_1] \ldots [A_m] \). Let \( A_i = \{a_i^1, \ldots, a_i^n\} \). Then since \( \text{ser} = \text{sim} \), we have \( [A_i] = \{[a_i^1], \ldots, [a_i^n]\} \), for \( i = 1, \ldots, m \).

2. It suffices to show that \( x \equiv_{\text{ind}} y \iff x^1 \equiv_{\text{ser}} y^1 \). Note that if \( \text{sim} = \text{ser} \) then \( \text{ser} \) is symmetric. We have

\[
x \equiv_{\text{ind}} y \iff x = wabz \land y = wbaz \land (a, b) \in \text{ind} \quad \text{(Definition of } \equiv_{\text{ind}} \text{)}
\]

\[
\iff x^1 = w^1\{a\}\{b\}z^1 \land y^1 = w^1\{b\}\{a\}z^1 \\
\land \{a\} \times \{b\} \subseteq \text{ser}
\iff x^1 \equiv_{\text{ser}} w^1\{a, b\}z^1 \land w^1\{a, b\}z^1 \equiv_{\text{ser}} y^1
\quad \text{(Definition of } \equiv_{\text{ser}} \text{)}
\]

Let \( t \) be a trace over \((E, \text{ind})\) and let \( v \) be a comtrace over \((E, \text{sim}, \text{ser})\).

- We say that \( t \) and \( v \) are equivalent if \( \text{sim} = \text{ser} = \text{ind} \) and there is \( x \in E^* \) such that \( t = [x]_{\equiv_{\text{ind}}} \) and \( v = [x^1]_{\equiv_{\text{ser}}} \). If a trace \( t \) and a comtrace \( v \) are equivalent we will write \( t \equiv v \).

Note that Lemma 5.1 guarantees that this definition is correct.

Proposition 5.2. Let \( t, r \) be traces and \( v, w \) be comtraces. Then

1. \( t \equiv v \land t \equiv w \implies v = w \).
2. \( t \equiv v \land r \equiv w \implies t = r \). \quad \square

Equivalent traces and comtraces generate identical partial orders. However, we will postpone the discussion of this issue to Section 10. Hence traces can be regarded as a special case of comtraces.

It appears that the concept of the comtrace can be very useful to formalize the concept of synchrony (cf. [18]). In principle events \( a_1, \ldots, a_k \) are synchronous if they can be executed in one step \( \{a_1, \ldots, a_k\} \) but this execution cannot be modeled by any sequence of proper subsets of \( \{a_1, \ldots, a_k\} \). Note that in general ‘synchrony’ is not necessarily ‘simultaneity’ as it does not include the concept of time [10]. It appears, however, that the mathematics used to deal with synchrony are very close to that used to deal with simultaneity.
Definition 5.3 (Independency and synchrony). Let \((E, \text{sim}, \text{ser})\) be a given comtrace alphabet. We define the relations \(\text{ind}, \text{syn}\) and the set \(\mathbb{S}_{\text{syn}}\) as follows:

- \(\text{ind} \subseteq E \times E\), called independency, and defined as \(\text{ind} = \text{ser} \cap \text{ser}^{-1}\),
- \(\text{syn} \subseteq E \times E\), called synchrony, and defined as:
  \[(a, b) \in \text{syn} \iff (a, b) \in \text{sim} \setminus \text{ser}^{\text{sym}},\]
- \(\mathbb{S}_{\text{syn}} \subseteq \mathbb{S}\), called synchronous steps, and defined as:
  \[A \in \mathbb{S}_{\text{syn}} \iff A \neq \emptyset \land (\forall a, b \in A. \ (a, b) \in \text{syn}).\]

If \((a, b) \in \text{ind}\) 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 \((a, b) \in \text{syn}\) 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 is not equivalent to either \(a\) followed by \(b\), or \(b\) followed by \(a\). In principle, the relation \(\text{syn}\) is a counterpart of ‘synchrony’ (cf. \([18]\)). If \(A \in \mathbb{S}_{\text{syn}}\), then the set of events \(A\) can be executed as one step, but it cannot be simulated by any sequence of its subsets.

Example 5.2. Assume we have \(E = \{a, b, c, d, e\}, \text{sim} = \{(a, b), (b, a), (a, c), (c, a), (a, d), (d, a)\},\) and \(\text{ser} = \{(a, b), (b, a), (a, c)\}\). Hence, \(\mathbb{S} = \{\{a, b\}, \{a, c\}, \{a, d\}, \{a\}, \{b\}, \{c\}, \{e\}\}\), and

\[
\text{ind} = \{(a, b), (b, a)\} \quad \text{syn} = \{(a, d), (d, a)\} \quad \mathbb{S}_{\text{syn}} = \{\{a, d\}\}
\]

Since \(\{a, d\} \in \mathbb{S}_{\text{syn}}\) the step \(\{a, d\}\) cannot be split. For example the comtraces \(x_1 = \{[a, b] \{c\} \{a\}\}, x_2 = \{[e] \{a, d\} \{a, c\}\},\) and \(x_3 = \{[a, b] \{c\} \{a\} \{e\} \{a, d\} \{a, c\}\}\) are the following sets of step sequences:

\[
\begin{align*}
\text{x}_1 &= \{\{a, b\} \{c\} \{a\}, \{a\} \{b\} \{c\} \{a\}, \{b\} \{a\} \{c\} \{a\}, \{b\} \{a\} \{c\} \{a\}\} \\
\text{x}_2 &= \{\{e\} \{a, d\} \{a, c\}, \{e\} \{a, d\} \{a, c\}\} \\
\text{x}_3 &= \{\{a, b\} \{c\} \{a\} \{e\} \{a, d\} \{a, c\}, \{a\} \{b\} \{c\} \{a\} \{e\} \{a, d\} \{a, c\}, \\
& \quad \{b\} \{a\} \{c\} \{a\} \{e\} \{a, d\} \{a, c\}, \{b\} \{a\} \{c\} \{a\} \{e\} \{a, d\} \{a, c\}\}
\end{align*}
\]

We also have \(x_3 = x_1 \otimes x_2\). Note that since \((c, a) \notin \text{ser}\), \((a, c) \equiv_{\text{ser}} \{a\} \{c\} \neq_{\text{ser}} \{c\} \{a\}\).

6. Generalized Comtraces

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

Example 6.1. Let \(E = \{a, b, c\}\) where \(a\), \(b\) and \(c\) are three atomic operations defined as follows (we assume simultaneous reading is allowed):

\[
\begin{align*}
a &: x \leftarrow x + 1 \\
b &: x \leftarrow x + 2 \\
c &: y \leftarrow y + 1
\end{align*}
\]
It is reasonable to consider them all as ‘concurrent’ as any order of their executions yields exactly the same results (see [11, 13] for more motivation and formal considerations as well as the program \( P \_4 \) of Example [4, 1]). Note that while simultaneous execution of \( \{ a, c \} \) and \( \{ b, c \} \) are allowed, the step \( \{ a, b \} \) is not, 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,

\[
x = \{ \{ a \} \{ b \} \{ c \}, \{ a \} \{ c \} \{ b \}, \{ b \} \{ a \} \{ c \}, \{ b \} \{ c \} \{ a \}, \{ c \} \{ a \} \{ b \}, \{ c \} \{ b \} \{ a \}, \\
\{ a, c \} \{ b \}, \{ b, c \} \{ a \}, \{ b \} \{ a, c \}, \{ a \} \{ b, c \} \}
\]

is a valid concurrent history or behavior [11, 13]. 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 no comtrace can represent this situation.

In this section, we will introduce the concept of generalized comtraces (g-comtraces), an extension of comtraces, also equational monoids of step sequences. The g-comtraces will be able to model “non-simultaneously” relationship similar to the one from Example [6, 1].

**Definition 6.1 (Generalized comtrace alphabet).** Let \( E \) be a finite set (of events). Let \( \text{ser}, \sim \) and \( \text{inl} \) be three relations on \( E \) called *serializability*, *simultaneity* and *interleaving* respectively satisfying the following conditions:

- \( \sim \) and \( \text{inl} \) are irreflexive and symmetric,
- \( \text{ser} \subseteq \sim \), and
- \( \sim \cap \text{inl} = \emptyset \).

Then the triple \((E, \sim, \text{ser}, \text{inl})\) is called a *g-comtrace alphabet*.

The interpretation of the relations \( \sim \) and \( \text{ser} \) is as in Definition [5, 1] and \( (a, b) \in \text{inl} \) means \( a \) and \( b \) cannot occur simultaneously, but their occurrence in any order is equivalent. As for comtraces, we define \( S \), the set of all (potential) steps, as the set of all cliques of the graph \((E, \sim)\).

**Definition 6.2 (Generalized comtrace congruence).** Let \( \Theta = (E, \sim, \text{ser}, \text{inl}) \) be a g-comtrace alphabet and let \( \equiv _{(\text{ser}, \text{inl})} \), called *g-comtrace congruence*, be the \( \text{EQ} \)-congruence defined by the set of equations \( \text{EQ} = \text{EQ}_1 \cup \text{EQ}_2 \), where

\[
\text{EQ}_1 \overset{df}{=} \{ A = BC \mid A \in S \land B \subseteq C \subseteq \text{ser} \},
\]

\[
\text{EQ}_2 \overset{df}{=} \{ BA = AB \mid A \in S \land B \subseteq A \times B \subseteq \text{inl} \}.
\]

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

Since \( \text{ser} \) and \( \text{inl} \) are irreflexive, \( (A = BC) \in \text{EQ}_1 \) implies \( B \cap C = \emptyset \), and \( (AB = BA) \in \text{EQ}_2 \) implies \( A \cap B = \emptyset \). Since \( \text{inl} \cap \sim = \emptyset \), we also have that if \( (AB = BA) \in \text{EQ}_2 \), then \( A \cup B \notin S \).

By Proposition [2, 1] g-comtrace congruence relations can also be defined explicitly in non-equational form as follows.
Proposition 6.1. Let $\Theta = (E, \text{sim}, \text{ser}, \text{inl})$ be a g-comtrace alphabet and let $S^*$ be the set of all step sequences defined on $\Theta$.

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

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

We define $\approx_{\text{ser, inl}} \overset{\text{df}}{=} \approx_1 \cup \approx_2$. Then $\equiv_{\text{ser, inl}} = \left( \approx_{\text{ser, inl}} \cup \approx_{\text{ser, inl}}^{-1} \right)^*$. \hfill $\Box$

The name “generalized comtraces” comes from that fact that when $\text{inl} = \emptyset$, Definition 6.2 coincides with Definition 5.2 of comtrace monoids. Hence comtraces can be regarded as a special case of generalized comtraces. We will omit the subscript $\{\text{ser, inl}\}$ from the g-comtrace congruence and $\approx_{\text{ser, inl}}$, and write $\equiv$ and $\approx$ if it causes no ambiguity.

Example 6.2. The set $x$ from Example 6.1 is a g-comtrace with $E = \{a,b,c\}$, $\text{ser} = \text{sim} = \{(a,c),(c,a),(b,c),(c,b)\}$, $\text{inl} = \{(a,b),(b,a)\}$, and $S = \{(a,c),(b,c),(a),(b),(c)\}$. So we write $x = \{(a,c)\{b\}\}$. \hfill $\blacksquare$

It is worth noticing 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 $\{a\}$ 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 result. 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 [24]. They are intuitive and powerful tools with many applications in practice. In this section we will present equivalent properties for both comtrace and g-comtrace congruence. The basic obstacle is switching from sequences to step sequences.

7.1. Properties of Comtrace Congruence

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

For each step sequence or enumerated step sequence $x = X_1 \ldots X_t$, we define the step sequence weight of $x$ as weight$(x) \overset{\text{df}}{=} \sum_{i=1}^{tk} |X_i|$. We also define $|\emptyset(x)\overset{\text{df}}{=} \bigcup_{i=1}^{t} X_i$. 

18
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 \[5\]. 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 \(\text{ser}\) is usually not commutative. Example \[5.1\] works as a counter example since \(\{a\}\{b,c\} \equiv \{a\}\{b\}\{c\}\) but \(\{b,c\}\{a\} \neq \{c\}\{b\}\{a\}\). 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:

\[
\lambda \div_r a \overset{df}{=} \lambda, \quad wA \div_r a \overset{df}{=} \begin{cases} (w \div_r a)A & \text{if } a \notin A \\ w & \text{if } A = \{a\} \\ w(A \setminus \{a\}) & \text{otherwise.} \end{cases}
\]

Symmetrically, a step sequence left cancellation operator \(\div_l\) is defined as follows:

\[
\lambda \div_l a \overset{df}{=} \lambda, \quad Aw \div_l a \overset{df}{=} \begin{cases} A(w \div_l a) & \text{if } a \notin A \\ w & \text{if } A = \{a\} \\ (A \setminus \{a\})w & \text{otherwise.} \end{cases}
\]

Finally, for each \(D \subseteq E\), we define the function \(\pi_D : \mathbb{S}^+ \rightarrow \mathbb{S}^+\), step sequence projection onto \(D\), as follows:

\[
\pi_D(\lambda) \overset{df}{=} \lambda, \quad \pi_D(wA) \overset{df}{=} \begin{cases} \pi_D(w) & \text{if } A \cap D = \emptyset \\ \pi_D(w)(A \cap D) & \text{otherwise.} \end{cases}
\]

The below result shows that the algebraic properties of comtraces are similar to the algebraic properties of traces \[24\].

**Proposition 7.1.**

1. \(u \equiv v \implies \text{weight}(u) = \text{weight}(v)\). (step sequence weight equality)
2. \(u \equiv v \implies |u|_a = |v|_a\). (event-preserving)
3. \(u \equiv v \implies u \div_r a \equiv v \div_r a\). (right cancellation)
4. \(u \equiv v \implies u \div_l a \equiv v \div_l a\). (left cancellation)
5. \(u \equiv v \iff \forall s,t \in \mathbb{S}^+, \text{ust} \equiv \text{svt}\). (step subsequence cancellation)
6. \(u \equiv v \implies \pi_D(u) \equiv \pi_D(v)\). (projection rule)

**Proof.** For all except (5), it suffices to show that \(u \equiv v\) implies the right hand side of (1)–(6). Note that \(u \equiv v\) means \(u = xAy, v = xBCy\), where \(A = B \cup C, B \cap C = \emptyset\) and \(B \times C \subseteq \text{ser}\).

1. Since \(A = B \cup C\) and \(B \cap C = \emptyset\), we have \(\text{weight}(A) = |A| = |B| + |C| = \text{weight}(BC)\). Hence, \(\text{weight}(u) = \text{weight}(x) + \text{weight}(A) + \text{weight}(z) = \text{weight}(x) + \text{weight}(BC) + \text{weight}(z) = \text{weight}(v)\).

2. There are two cases:
   - \(a \in A\): Then \(a \notin B \cap C\) because \(B \cap C = \emptyset\). Since \(A = B \cup C\), either \(a \in B\) or \(a \in C\). Then \(|A|_a = |BC|_a\). Therefore, \(|u|_a = |x|_a + |A|_a + |z|_a = |x|_a + |BC|_a + |z|_a = |v|_a\).
Notice that when $u \approx v$, the case $u = xAy \approx xBCy$ follows from Proposition 7.1. So we only need to consider the case $u = xABy$ and $v = xBAy$, where $A \times B \subseteq \text{inl}$ and $A \cap B = \emptyset$. The rest can be proved similarly to the proof of Proposition 7.1. 

\section{Properties of Generalized Comtrace Congruence}

We now show that g-comtrace congruence has virtually the same algebraic properties as comtrace congruence.

\begin{proposition}
Let $S$ be the set of all steps over a g-comtrace alphabet $(E, \text{sim}, \text{ser}, \text{inl})$ and $u, v \in S^*$. Then
\begin{enumerate}
  \item $u \equiv v \implies \text{weight}(u) = \text{weight}(v)$. (step sequence weight equality)
  \item $u \equiv v \implies |u|_a = |v|_a$. (event-preserving)
  \item $u \equiv v \implies u \equiv v \equiv \overline{R}a$. (right cancellation)
  \item $u \equiv v \implies u \equiv v \equiv \overline{L}a$. (left cancellation)
  \item $u \equiv v \implies \forall s, t \in S^*, sut \equiv vst$. (step subsequence cancellation)
  \item $u \equiv v \implies \pi_D(u) \equiv \pi_D(v)$. (projection rule)
\end{enumerate}
\end{proposition}

\begin{proof}
For all except (5), it suffices to show that $u \approx v$ implies the right hand side of (1)–(6). Notice that when $u \approx v$, the case $u = xAy \approx xBCy$ follows from Proposition 7.1. So we only need to consider the case $u = xABy$ and $v = xBAy$, where $A \times B \subseteq \text{inl}$ and $A \cap B = \emptyset$. The rest can be proved similarly to the proof of Proposition 7.1. 
\end{proof}

Corollary 7.1. For all $u, v, x \in S^*$, we have
\begin{enumerate}
  \item $u \equiv v \implies u \equiv v \equiv \overline{R}x$.
  \item $u \equiv v \implies u \equiv v \equiv \overline{L}x$.
\end{enumerate}
Corollary 7.2. For all step sequences \(u, v, x\) over a g-comtrace alphabet \((E, \text{sim}, \text{ser}, \text{inl})\),

1. \(u \equiv v \implies u \div R x \equiv v \div R x\).
2. \(u \equiv v \implies u \div L x \equiv v \div L x\).

The following proposition ensures that if any relation from the set \(\{\leq, \geq, <, >, =, \neq\}\) between the positions of two event occurrences holds after applying cancellation or projection operations on a g-comtrace \([\mathcal{U}]\), then it also holds for the whole \([\mathcal{U}]\). In other words, both cancellation and projection preserve ordering in the stratified orders defined by g-comtraces.

Proposition 7.3. Let \(\pi\) be an enumerated step sequence over a g-comtrace alphabet \((E, \text{sim}, \text{ser}, \text{inl})\) and \(\alpha, \beta, \gamma \in \Sigma_a\) such that \(\gamma \notin \{\alpha, \beta\}\). Let \(\mathcal{R} \in \{\leq, \geq, <, >, =, \neq\}\). Then

1. If \(\forall \mathcal{V} \in [\mathcal{U} \div L \gamma]. \ pos_{\pi}(\alpha) \mathcal{R} \ pos_{\pi}(\beta)\), then \(\forall \mathcal{W} \in [\mathcal{U}]. \ pos_{\pi}(\alpha) \mathcal{R} \ pos_{\pi}(\beta)\).
2. If \(\forall \mathcal{V} \in [\mathcal{U} \div R \gamma]. \ pos_{\pi}(\alpha) \mathcal{R} \ pos_{\pi}(\beta)\), then \(\forall \mathcal{W} \in [\mathcal{U}]. \ pos_{\pi}(\alpha) \mathcal{R} \ pos_{\pi}(\beta)\).
3. If \(S \subseteq \Sigma_a\) such that \(\{\alpha, \beta\} \subseteq S\), then
\[
(\forall \mathcal{V} \in [\pi_S(\pi)]. \ pos_{\pi}(\alpha) \mathcal{R} \ pos_{\pi}(\beta)) \implies (\forall \mathcal{W} \in [\mathcal{U}]. \ pos_{\pi}(\alpha) \mathcal{R} \ pos_{\pi}(\beta)).
\]

Proof. 1. Assume that
\[
\forall \mathcal{V} \in [\mathcal{U} \div L \gamma]. \ pos_{\pi}(\alpha) \mathcal{R} \ pos_{\pi}(\beta) \quad (7.1)
\]
Suppose for a contradiction that \(\exists \mathcal{W} \in [\mathcal{V}]. \neg(pos_{\pi}(\alpha) \mathcal{R} pos_{\pi}(\beta))\). Since \(\gamma \notin \{\alpha, \beta\}\), we have \(\neg(pos_{\pi \div L \gamma}(\alpha) \mathcal{R} pos_{\pi \div L \gamma}(\beta))\). But \(\mathcal{W} \in [\mathcal{V}]\) implies \(\mathcal{V} \div L \gamma \equiv \mathcal{U} \div L \gamma\). Hence, \(\mathcal{W} \div L \gamma \in [\mathcal{U} \div L \gamma]\) and \(\neg(pos_{\pi \div L \gamma}(\alpha) \mathcal{R} pos_{\pi \div L \gamma}(\beta))\), contradicting (7.1).

2. Dually to part (1).

3. Assume that
\[
\forall \mathcal{V} \in [\pi_S(\pi)]. \ pos_{\pi}(\alpha) \mathcal{R} \ pos_{\pi}(\beta) \quad (7.2)
\]
Suppose for a contradiction that \(\exists \mathcal{W} \in [\mathcal{V}]. \neg(pos_{\pi}(\alpha) \mathcal{R} pos_{\pi}(\beta))\). Since \(\{\alpha, \beta\} \subseteq S\), we have \(\neg(pos_{\pi_S(\pi)}(\alpha) \mathcal{R} pos_{\pi_S(\pi)}(\beta))\). But \(\mathcal{W} \in [\mathcal{V}]\) implies \(\pi_S(\mathcal{W}) \equiv \pi_S(\mathcal{V})\). Hence, \(\pi_S(\mathcal{W}) \in [\pi_S(\pi)]\) and \(\neg(pos_{\pi_S(\pi)}(\alpha) \mathcal{R} pos_{\pi_S(\pi)}(\beta))\), contradicting (7.2).

Clearly the above results also holds for comtraces as they are just g-comtraces with \(\text{inl} = \emptyset\).

8. Maximally Concurrent and Canonical Representations

We will show that both traces, comtraces and g-comtraces have some special representations, that intuitively correspond to maximally concurrent execution of concurrent histories, i.e., “executing as much as possible in parallel”\(^7\). 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 will consider comtraces and traces as special and more regular cases.

\(^7\)This kind of semantics is formally defined and analyzed for example in [3].
8.1. Representations of Generalized Comtraces

Let $\Theta = (E, \text{sim}, \text{ser}, \text{inl})$ be a g-comtrace alphabet and $S$ 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 [4] for an alternative version of traces called “vector firing sequences” (see [15, 27]).

**Definition 8.1 (Greedy maximally concurrent form).** A step sequence $u = A_1 \ldots A_k \in S^*$ is in greedy maximally concurrent form (GMC-form) if and only if for each $i = 1, \ldots, k$:

$$\exists\, y_i \\
(B_i y_i = A_1 \ldots A_k) \implies |B_i| \leq |A_i|,$$

where for all $i = 1, \ldots, k$, $A_i, B_i \in S$, and $y_i \in S^*$.

**Proposition 8.1.** For each g-comtrace $u$ over $\Theta$, there is a step sequence $u \in S^*$ in GMC-form such that $u = |u|$.

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

1. Initialize $i \leftarrow 0$ and $v_0 \leftarrow \varnothing$
2. while $v_i \neq \lambda$ do
3. $i \leftarrow i + 1$
4. Find $A_i$ such that for each $B_i y_i = v_{i-1}$, $|B_i| \leq |A_i|
5. $v_i \leftarrow v_{i-1} \cup L A_i$
6. end while
7. $k \leftarrow i - 1$.

Since $\text{weight}(v_{i+1}) < \text{weight}(v_i)$ the above algorithm always terminates. Clearly $u = A_1 \ldots A_k$ is in GMC-form and $u \in u$. \hfill \Box

The algorithm from the proof of Proposition 8.1 used to generate $A_1, \ldots, A_k$ justifies the prefix “greedy” in Definition 8.1. However, the GMC representation of g-comtraces is seldom unique and often intuitively “not maximally concurrent”. Consider the following two examples.

**Example 8.1.** Let $E = \{a, b, c\}$, $\text{sim} = \{(a, c), (c, a)\}$, $\text{ser} = \text{sim}$ and $\text{inl} = \{(a, b), (b, a)\}$ and $u = [\{a\} \{b\} \{c\}] = \{\{a\} \{b\} \{c\}, \{b\} \{a\} \{c\}, \{b\} \{a, c\}\}$.

Note that every element of $u$ is in GMC-form, but only $\{b\} \{a, c\}$ can intuitively be interpreted as maximally concurrent. \hfill \Box

**Example 8.2.**

Let $E = \{a, b, c, d, e\}$ and $\text{sim} = \text{ser}, \text{inl}$ be as in the picture on the right, and let $u = [\{a\} \{b, c, d, e\}]$. One can easily verify by inspection that $\{a\} \{b, c, d, e\}$ is the shortest element of $u$ and the only element of $u$ 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 $\{b, c, d, e\}$. \hfill \Box

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

---

22
Let \( x = A_1 \ldots A_k \) be a step sequence. We define \( \text{length}(A_1 \ldots A_k) \equiv k \).
We also say that \( A_i \) is maximally concurrent in \( x \) if \( \forall i \neq j \exists A_i,A_j \implies |B_i| \leq |A_i| \).

Note that \( A_k \) is always maximally concurrent in \( x \), which makes the following definition correct.

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

**Definition 8.2.** A step sequence \( u = A_1 \ldots A_k \) is maximally concurrent (MC-) if and only if

1. \( v \equiv u \implies \text{length}(u) \leq \text{length}(v) \),
2. for all \( i = 1, \ldots, k \) and for all \( w \),
   \[
   (u = A_1 \ldots A_k \equiv w \land \text{length}(u_i) = \text{length}(w)) \implies mc(u_i) \leq mc(w).
   \]

**Theorem 8.1.** For every g-comtrace \( u \), there exists a step sequence \( u \in u \) such that \( u \) is maximally concurrent.

**Proof.** Let \( u_1 \in u \) be a step sequence such that for each \( v, v \equiv u_1 \implies \text{length}(u_1) \leq \text{length}(v) \), and \( (v \equiv u_1 \land \text{length}(u_1) = \text{length}(v)) \implies mc(u_1) \leq mc(v) \). Obviously such \( u_1 \) exists for every g-comtrace \( u \). Assume that \( u_1 = A_1w_1 \) and \( \text{length}(u_1) = k \). Let \( u_2 \) be a step sequence satisfying \( u_2 \equiv w_1, u_2 \equiv v \implies \text{length}(u_2) \leq \text{length}(v) \), and \( (v \equiv u_2 \land \text{length}(u_2) = \text{length}(v)) \implies mc(u_2) \leq mc(v) \). Assume that \( u_2 = A_2w_3 \). We repeat this process \( k - 1 \) times. Note that \( u_k = A_k \in \mathcal{S} \). The step sequence \( u = A_1 \ldots A_k \) is maximally concurrent and \( u \in u \). \( \square \)

For the case of Example 8.1 the step sequence \( \{b\} \{a, c\} \) is maximally concurrent and for the case of Example 8.2 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 unique representation is often very 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 unique representation must also have an easy to handle definition. For g-comtraces we can get unique representation by introducing some total ordering of steps and then apply this ordering on either GMC-forms or MC-forms. To achieve this purpose, we just need an arbitrary total order on the set of events \( E \). However the definition of GMC-form is local (step by step) and easier to handle than the definition of MC-form, which is global as it requires comparing the length of all step sequences in a given g-comtrace. Because of “greediness”, ordering different GMC-representations is also simpler than ordering different MC-representations. Hence, we will base our unique representation on the idea of GMC-form.

**Definition 8.3 (Lexicographical ordering).** Assume that we have a total order \( <_E \) on \( E \).

1. We define a step order \( <^s \) on \( \mathcal{S} \) as follows:
   \[
   A <^s B \iff |A| > |B| \lor (|A| = |B| \land A \neq B \land \min_{<_E} (A \setminus B) < E \min_{<_E} (B \setminus A)),
   \]
   where \( \min_{<_E} (X) \) denotes the least element of the set \( X \subseteq E \) w.r.t. \( <_E \).
2. Let $A_1 \ldots A_n$ and $B_1 \ldots B_m$ be two sequences in $S^*$. We define a lexicographical order $<_\text{lex}$ on step sequences in a natural way as the lexicographical order induced by $<_\text{st}$, i.e.,

$$A_1 \ldots A_n <_{\text{lex}} B_1 \ldots B_m \iff \exists k > 0. \left( (\forall i < k. A_i = B_i) \land (A_k <_\text{st} B_k \lor n < k \leq m) \right).$$

Directly from the above definition we have the desired properties of $<_\text{st}$ and $<_\text{lex}$.

**Corollary 8.1.**

1. The step order $<_\text{st}$ is a total order on $S$.
2. The lexicographical order $<_\text{lex}$ is a total order on $S^*$.

**Example 8.3.** Assume that $a <_E b <_E c <_E d <_E e$. Then we have $\{a, b, c, e\} <_\text{st} \{b, c, d\}$ since $\{a, b, c, e\} \setminus \{b, c, d\} = \{a\}$, $\{b, c, d\} \setminus \{a, b, c, e\} = \{d\}$, and $a <_E d$. And $\{a, c\} \{b, c\} \{d\} \{c, d\} <_{\text{lex}} \{a, c\} \{b\} \{c, d, e\}$ since $|\{b, c\}| > |\{b\}|$.

**Definition 8.4 (g-Canonical step sequence).** A step sequence $x \in S^*$ is g-canonical if for every step sequence $y \in S^*$, we have $(x \equiv y \land x \neq y) \implies x <_{\text{lex}} y$.

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

**Corollary 8.2.**

1. Each g-canonical step-sequence is in GMC-form.
2. For every step sequence $x \in S^*$, there exists exactly one g-canonical sequence $u$ such that $x \equiv u$.

All of the concepts and results discussed so far in this section hold also for general equational monoids derived form the step sequence monoid (like those considered in [16]).

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, 4, 12, 16].

8.2. Canonical Representations of Comtraces

First note that comtraces are just g-comtraces with $\text{inl}$ being empty relation, so all definitions for g-comtraces also hold for comtraces.

Let $\theta = (E, \text{sim}, \text{ser})$ be a comtrace alphabet (i.e. $\text{inl} = \emptyset$) and $S$ be the set of all steps over $\theta$. In principle, $(a, b) \in \text{ser}$ 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 allow such replacement.

**Definition 8.5 (Forward dependency).** Let $\text{FD} \subseteq S \times S$ be a relation comprising all pairs of steps $(A, B)$ such there exists a step $C \in S$ such that

$$C \subseteq B \land A \times C \subseteq \text{ser} \land C \times (B \setminus C) \subseteq \text{ser}.$$ 

The relation $\text{FD}$ is called forward dependency on steps.
Definition 8.6 (Comtrace canonical step sequence [12]).

\[ a \in A \]

Assume Proposition 7.1(6), (Lemma 8.1. prove that \( \{ a, b \} \in A \). Assume \( A \cup B \subseteq \mathbb{S} \).

For the former case:

\[ \{ a, b \} = \pi_{a,b}(A \cup B) = \pi_{a,b}(AB) = \{ a \} \{ b \}. \]

But \( \{ a, b \} = \{ a \} \{ b \} \) means \( (a, b) \in B \). Therefore \( A \cup B \subseteq \mathbb{S} \).

Assume \( A \cup C \subseteq \mathbb{S} \) and consequently \( A \cap B = \emptyset \). Assume \( A \cup C = \emptyset \). By Proposition 7.1(6), \( \{ a, c \} = \pi_{a,c}(A \cup C) \). Assume \( A \cup B \subseteq \mathbb{S} \).

We will now recall the definition of a canonical step sequence for con-traces.

Definition 8.6 (Comtrace canonical step sequence [12]). A step sequence \( u = A_1 \ldots A_k \)

is canonical if we have \( A_i, A_{i+1} \notin \mathbb{F} \) for all \( i, 1 \leq i < k \).

The next results shows that the canonical step sequence for con-traces is in fact "greedy".

Lemma 8.2. For each canonical step sequence \( u = A_1 \ldots A_k \), we have

\[ A_1 = \{ a \mid \exists w \in [u], w = C_1 \ldots C_m \wedge a \in C_1 \}. \]

Proof. Let \( A = \{ a \mid \exists w \in [u], w = C_1 \ldots C_m \wedge a \in C_1 \}. \) Since \( u \in [u], A \subseteq A \). We need to prove that \( A \subseteq A_1 \). Let \( A_1 \in A \) if \( k = 1 \), so assume \( k > 1 \). Suppose that \( a \in A \setminus A_1, a \in A_j, 1 < j \leq k \) and \( a \notin A_i \) for \( i < j \). Note that \( A_1 \) is also canonical and \( u' = A_1 \ldots A_j = (u \setminus_R (A_1 \ldots A_k)) \setminus_L (A_1 \ldots A_{j-2}) \). Let \( v' = (v \setminus_R (A_1 \ldots A_k)) \setminus_L (A_1 \ldots A_{j-2}) \). We have \( v' = B'x' \) where \( a \notin B' \). By \( \{ a, c \} \subseteq B \), \( c \notin B \), \( (a, c) \notin B \). For the former case:

\[ \pi_{a,c}(u') = \{ c \} \{ a \} \] (if \( c \notin A_j \)) or \( \pi_{a,c}(u') = \{ a \} \{ c \} \) (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 \( c \notin B' \)).

In both cases \( \pi_{a,c}(u') \neq \pi_{a,c}(v') \), contradicting Proposition 7.1(6). If \( \pi_{a,c}(u') = \{ a \} \{ c \} \) then \( \pi_{a,c}(v') \) equals either \( \{ a, c \} \{ c \} \) (if \( c \in B' \)) or \( \{ a \} \{ c \} \{ c \} \) (if \( c \notin B' \)).

In both cases \( \pi_{a,c}(u') \neq \pi_{a,c}(v') \), contradicting Proposition 7.1(6).

For the latter case, let \( d \in A_{j-1} \).

\[ \pi_{a,b,d}(u') = \{ d \} \{ a, b \} \] (if \( d \notin A_j \)), or \( \pi_{a,b,d}(u') = \{ d \} \{ a, b, d \} \) (if \( d \in A_j \)). If \( \pi_{a,b,d}(u') = \{ d \} \{ a, b, d \} \) then \( \pi_{a,b,d}(v') \) is one of the following

\[ \{ a, b, d \}, \{ a, b \} \{ d \}, \{ a, d \} \{ b \}, \{ a, b \} \{ d \} \{ a \} \{ d \} \{ b \}, \{ a \} \{ d \} \{ b \}, \{ a \} \{ d \} \{ b \}, \{ a \} \{ d \} \{ b \}, \{ a \} \{ d \} \{ b \}, \{ a \} \{ d \} \{ b \}. \]

However in any of these cases we have \( \pi_{a,b,d}(u') \neq \pi_{a,b,d}(v') \), contradicting Proposition 7.1(6).
We will now show that for comtraces the canonical form as defined by Definition 8.6 and GMC-form are equivalent, and that each comtrace has a unique canonical representation.

**Theorem 8.2.** A step sequence $u$ is in GMC-form if and only if it is canonical.

**Proof.** ($\Rightarrow$) Suppose that $u = A_1 \ldots A_k$ is canonical. By Lemma 8.3, we have that for each $B_1 y_1 \equiv A_1 \ldots A_k$, $|B_1| \leq |A_1|$. Since each $A_1 \ldots A_k$ is also canonical, $A_2 \ldots A_k$ is canonical so by Lemma 8.2, again we have that for each $B_2 y_2 \equiv A_2 \ldots A_k$, $|B_2| \leq |A_2|$. 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 $(A_j, A_{j + 1}) \in FD$. Hence $A_1 \ldots A_{j - 1}$ is canonical, and, by (1) of this Theorem, in GMC-form. By Lemma 8.1, either there is a non empty $C \subseteq A_{j + 1}$ such that $(A_j \cup C)(A_{j + 1} \setminus B) \equiv A_j A_{j + 1}$, or $A_j \cup A_{j + 1} \equiv A_1 A_{j + 1}$. In the first case since $C \neq \emptyset$, then $|A_j \cup C| > |A_j|$, 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 8.3 (implicit in [12]).** For each step sequence $v$ there is a unique canonical step sequence $u$ such that $v \equiv u$.

**Proof.** The existence follows from Proposition 8.1 and Theorem 8.2. 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 8.2, 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 $A_j = B_j$ for $i = 2, \ldots, k$ and $k = m$.

The result of Theorem 8.3 was not stated explicitly in [12], but it can be derived from the results of Propositions 3.1, 4.8 and 4.9 of [12]. However Propositions 3.1 and 4.8 of [12] involve the concepts of partial orders and stratified order structures, while the proof of Theorem 8.3 uses only the algebraic properties of step sequences and comtraces.

Immediately from Theorems 8.2 and 8.3 we get the following result.

**Corollary 8.3.** 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 8.3.** If a step sequence $u$ is canonical and $u \equiv v$, then $\text{length}(u) \leq \text{length}(v)$.

**Proof.** By induction on $\text{length}(v)$. Obvious for $\text{length}(v) = 1$ as then $u = v$. Assume it is true for all $v$ such that $\text{length}(v) \leq r - 1$, $r \geq 2$. 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 7.1, $v_1 \equiv u \div L A_1 = A_2 \ldots A_k$, and $A_2 \ldots A_k$ is clearly canonical. Hence by induction assumption $k - 1 = \text{length}(A_2 \ldots A_k) \leq s$. By Lemma 8.2, $B_1 \subseteq A_1$, hence $v_1 = v \div L A_1 = B_2 \ldots B_r \div L A_1 = C_1 \ldots C_s$, which means $s \leq r - 1$. Therefore $k - 1 \leq s \leq r - 1$, i.e. $k \leq r$, which ends the proof.

26
Theorem 8.4. A step sequence $u$ is maximally concurrent if and only if it is canonical.

Proof. ($\Leftarrow$) Let $u$ be canonical. From Lemma 8.3, it follows the condition (1) of Definition 8.2 is satisfied. By Theorem 8.2, $u$ is in GMC-form, so the condition (2) of Definition 8.2 is satisfied as well.

($\Rightarrow$) By induction on $\text{length}(u)$. It is obviously true for $u = A_1$. Suppose it is true for $\text{length}(u) = k$. Let $u = A_1 A_2 ... A_k A_{k+1}$ be maximally concurrent. The step sequence $A_2 ... A_{k+1}$ is also maximally concurrent and canonical by the induction assumption. If $A_1 A_2 ... A_{k+1}$ is not canonical, then $(A_1, A_2) \in \mathcal{F_D}$. By Lemma 8.1, either there is non-empty $C \subseteq B$ 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 $(A_1 \cup C)(A_2 \setminus C) A_3 ... A_{k+1} \equiv A_1...A_{k+1} = u$ or $(A_1 \cup A_2) A_3 ... A_{k+1} \equiv A_1...A_{k+1} = u$. The former contradicts the condition (2) of Definition 8.2, the latter one contradicts the condition (1) of Definition 8.2 so $u$ is not maximally concurrent, which means $(A_1, A_2) \notin \mathcal{F_D}$, so $u = A_1 ... A_{k+1}$ is canonical. \hfill $\square$

Summing up, as far as canonical representation is concerned, comtraces are very regular. All three forms for g-comtraces, GMC-form, MC-form and g-canonical form, collapse to one comtrace canonical form if $\text{inl} = \emptyset$.

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

Corollary 8.4. If $x = a_1 ... a_k \in E^*$ is fully commutative and $y = a_{i_1} ... a_{i_k}$ is any permutation of $a_1 ... a_k$, then $x \equiv y$. \hfill $\square$

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

A fully commutative sequence $y$ is maximal in $x \in E^*$ if either $x = y b z$, or $x = \text{way}$ or $x = \text{waybzc}$, for some $a, b \in E$, $w, z \in E^*$ and neither $yb$ nor $az$ are fully commutative.

Lemma 8.4. Each sequence $x$ has a unique decomposition $x = x_1 ... x_k$ such that each fully commutative $x_1$ is maximal in $x$.

Proof. By induction on $x$. Obvious for $x = a \in E$. Assume it is true for $x$. Consider $xa$. We have $xa = x_1 ... x_k a$ where $x_1 ... x_k$ is a unique decomposition of $x$. If $x_n a$ is fully commutative then $x_1 ... x_{k-1} x'_k a$, where $x'_k = x_k a$ is the unique decomposition of $xa$. If $x_k a$ is not fully commutative, then $x = x_1 ... x_k x_{k+1}$, where $x_{k+1} = a$ is the unique decomposition of $xa$. \hfill $\square$

Definition 8.7 (Greedy maximally concurrent form for traces [2, 4]). A sequence $x \in E^*$ is in greedy maximally concurrent form (GMC-form) if $x = \lambda$ or $x = x_1 ... x_n$ such that

1. each $x_i$ is fully commutative, for $i = 1, ..., n$,
2. for each $1 \leq i \leq n - 1$ and for each element $a$ of $x_{i+1}$ there exists an element $b$ of $x_i$ such that $a \neq b$ and $(a, b) \notin \text{ind}$. \hfill $\blacksquare$
Corollary \[8.4\] and Lemma \[8.4\] explain and justify the name. Often the form from the above definition is called “canonical” \[4, 15, 16\].

**Theorem 8.5 \([2, 4]\).** For every trace \(t \in E^*/\equiv\), there exists \(x \in E^*\) such that \(t = [x]\) 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 \(t_1 = [abecba]\) from Example \[3.1\] has four GMC representations: \(abecba, acbba, abccba,\) and \(acbcb\). The GMC-form is however unique when traces are represented as vector firing sequences \[4, 15, 27\], where each fully commutative sequence is represented by a single unique vector of events (so the name “canonical” used in \[4, 15\] is justified). To get uniqueness in standard Mazurkiewicz trace formalism, 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 \(x = x_1 \ldots x_n\), each \(x_i\) is minimal w.r.t. the lexicographic ordering. The GMC-form with this additional condition is called Foata canonical form.

**Theorem 8.6 \([2]\).** 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 con-traces.

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

For each sequence \(x\) such that its maximal fully commutative composition is \(x = x_1 \ldots x_k\), define \(x^{(\text{max})} = \text{st}(x_1) \ldots \text{st}(x_k) \in S^*\), its maximally concurrent step sequence representation. The name is formally justified by the following result (which also follows implicitly from \[4\]).

**Proposition 8.2.**

1. A sequence \(x\) is in GMC-form in \((E, \text{ind})\) if and only if the step sequence \(x^{(\text{max})}\) is in GMC-form (or canonical form) in \((E, \text{sim}, \text{ser})\) where \(\text{sim} = \text{ser} = \text{ind}\).

2. \([x]_{\equiv \text{ind}} \overset{\text{trunc}}{\equiv} [x^{(\text{max})}]_{\equiv \text{ser}}\).

**Proof.**

1. Let \(x = x_1 \ldots x_k\) be maximally fully commutative representation of \(x\). If \(x\) is not in GMC-form then by (2) of Definition \[8.7\] there are \(x_i, x_{i+1}\) and \(a, b \in E\) such that \(a \in \text{st}(x_i)\), \(b \in \text{st}(x_{i+1})\) and \((a, b) \in \text{ind}\). Since \(\text{ser} = \text{ind}\) this means that \((\text{st}(x_i), \text{st}(x_{i+1})) \in \text{FD}\), so \(x^{(\text{max})}\) is not canonical. Suppose that \(x^{(\text{max})}\) is not canonical, i.e. \((\text{st}(x_i), \text{st}(x_{i+1})) \in \text{FD}\) for some \(i\). This means there is a non-empty \(C \subseteq \text{st}(x_{i+1})\) such that \(\text{st}(x_i) \times C \subseteq \text{ser}\) and \(C \times (\text{st}(x_{i+1}) \setminus C) \subseteq \text{ser}\). Let \(a \in \text{st}(x_i)\) and \(b \in C \subseteq \text{st}(x_{i+1})\). Since \(\text{ind} = \text{ser}\), then \((a, b) \in \text{ind}\), so \(x\) is not in GMC-form.

---

*The vector firing sequences were introduced by Mike Shields in 1979 \[27\] as an alternative equivalent representation of Mazurkiewicz traces.*

28
2. By the definition \( [x]/\equiv_{ind} \equiv \equiv_{ser}^{t_{con}} [x^{l}] \equiv_{ser} \). Let \( a_1 \ldots a_n \) be a fully commutative sequence. Since \( ser = ind \{ a_1 \} \ldots \{ a_n \} \equiv_{ser} \{ a_1, \ldots, a_n \} \). Hence, for each sequence \( x, x^{l} \equiv_{ser} x^{max}, \) i.e. \( [x^{l}] \equiv_{ser} = [x^{max}] \equiv_{ser} \). □

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. Generalized Comtrace and its Languages

As for traces, we can easily extend the concepts of comtraces and g-comtraces to the level of languages, with similar potential applications. We will give only the definitions and the results for g-comtraces, as they are practically identical in both cases.

Let \( \Theta = (E, \sim, ser, inl) \) be a g-comtrace alphabet and \( \mathcal{S} \) be the set of all possible steps over \( \Theta \). Any subset \( L \) of \( \mathcal{S}^* \) is a step sequence language over \( \Theta \), while any subset \( \mathcal{L} \) of \( \mathcal{S}^*/\equiv \) is a g-comtrace language over \( \Theta \).

For any step sequence language \( L \), we define a g-comtrace language \( [L]_{\Theta} \) (or just \( [L] \)) as \( [L] \overset{df}{=} \{ [u] \mid u \in L \} \), and \( [L] \) is called the g-comtrace language generated by \( L \).

For any g-comtrace language \( \mathcal{L} \), we define \( \cup \mathcal{L} \overset{df}{=} \{ u \mid [u] \in \mathcal{L} \} \). Given step sequence languages \( L_1, L_2 \) and g-comtrace languages \( \mathcal{L}_1, \mathcal{L}_2 \) over the alphabet \( \Theta \), the composition of languages are defined as following:

\[
L_1L_2 \overset{df}{=} \{ s_1 \ast s_2 \mid s_1 \in L_1 \land s_2 \in L_2 \} \quad \mathcal{L}_1 \mathcal{L}_2 \overset{df}{=} \{ t_1 \oplus t_2 \mid t_1 \in \mathcal{L}_1 \land t_2 \in \mathcal{L}_2 \}
\]

(Recall \( \ast \) and \( \oplus \) denote the operators for step sequence and g-comtrace monoids respectively.)

We let \( L^* \) and \( \mathcal{L}^* \) denote the Kleene closure of the step sequence language \( L \) and the g-comtrace language \( \mathcal{L} \). We define \( L^* \overset{df}{=} \bigcup_{n \geq 0} L^n \), where \( L^0 \overset{df}{=} \{ \lambda \} \) and \( L^{n+1} \overset{df}{=} L^nL \). We define \( \mathcal{L}^* \overset{df}{=} \bigcup_{n \geq 0} \mathcal{L}^n \), where \( \mathcal{L}^0 \overset{df}{=} \{ \lambda \} \) and \( \mathcal{L}^{n+1} \overset{df}{=} \mathcal{L}^n \mathcal{L} \).

Since g-comtrace languages are sets, one can use the standard set operations: union, intersection, difference, etc. The following result is a direct consequence of the g-comtrace language definition and the properties of g-comtrace composition.

**Proposition 9.1.** Let \( L, L_1, L_2 \) and \( L_i \) for \( i \in I \) be step sequence languages, and let \( \mathcal{L} \) be a g-comtrace language. Then:

1. \( [\emptyset] = \emptyset \)
2. \( [L_1][L_2] = [L_1L_2] \)
3. \( L_1 \subseteq L_2 \Rightarrow [L_1] \subseteq [L_2] \)
4. \( L \subseteq \bigcup [L] \)
5. \( \mathcal{L} = \bigcup \mathcal{L} \)
6. \( [L_1] \cup [L_2] = [L_1 \cup L_2] \)
7. \( \bigcup_{i \in I} [L_i] = [\bigcup_{i \in I} L_i] \)
8. \( [L]^* = [L^*] \).

**PROOF.** The proof is the same to the case of traces in [24]. □

When \( inl = \emptyset \), we have the case of comtrace languages. The languages of comtraces and g-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 \((E, \text{sim}, \text{ser}, \text{inl})\) from the system, and the gcomtrace (comtrace) language \([L]\) defines the structural semantics of the system.

**Example 9.1.** 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. [14]), usually under the interpretation that \( a = \text{‘Error Message’}, b = \text{‘Stop And Restart’}, \) and \( c = \text{‘Some Action’}. \) It can be formally specified in various notations including Priority and Inhibitor Nets (cf. [10, 13]). Its operational semantics (easily found in any model) can be defined by the following language of step sequences

\[
L_{\text{Priority}} \overset{\text{df}}{=} \text{Pref}(\left\{\text{c}\right\}^* \cup \left\{a\right\}\left\{b\right\} \cup \left\{a, c\right\}\left\{b\right\}^*),
\]

where \( \text{Pref}(L) \overset{\text{df}}{=} \bigcup_{w \in L} \{u \in L \mid \exists v. uv = w\} \) denotes the prefix closure of \( L \).

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

**Remark 9.1.** As opposed to the case of trace languages, we know very little about the properties of comtrace languages, not to mention the properties of g-comtrace languages. In particular deep results analogous to Zielonka’s theorem [30] are unknown.

### 10. Comtraces and Stratified Order Structures

This section consists of two parts. In the first part we will recall the major result of [12] that shows how comtraces define appropriate so-structures. The second part contains the new result showing how arbitrary finite so-structures can be represented by comtraces. This problem was not analyzed in [12].

We will start with the definition of \( \Diamond \)-closure construction that plays a substantial role in most of the applications of so-structures for modelling concurrent systems (cf. [12, 20]).

**Definition 10.1 (Diamond closure of relational structures [12]).**
Given a relational structure \( S = (X, R_1, R_2) \), we define \( S^\Diamond \), the \( \Diamond \)-closure of \( S \), as

\[
S^\Diamond \overset{\text{df}}{=} (X, \prec_{R_1, R_2}, \sqcap_{R_1, R_2}),
\]

where \( \prec_{R_1, R_2} \overset{\text{df}}{=} (R_1 \cup R_2)^* \circ R_1 \circ (R_1 \cup R_2)^* \) and \( \sqcap_{R_1, R_2} \overset{\text{df}}{=} (R_1 \cup R_2)^* \setminus \text{id}_X. \)
The motivation behind the above definition is the following. For 'reasonable' $R_1$ and $R_2$, the relational structure $(X, R_1, R_2)^\odot$ should satisfy the axioms S1–S4 of the so-structure definition. Intuitively, $\odot$-closure is a generalization of the transitive closure constructions for relations to so-structures. Note that if $R_1 = R_2$ then $(X, R_1, R_2)^\odot = (X, R_1, R_1)$. The following result shows that the properties of $\odot$-closure are very close to the appropriate properties of transitive closure.

**Theorem 10.1 (Closure properties of $\odot$-closure [12]).**

Let $S = (X, R_1, R_2)$ be a relational structure.

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

Each comtrace is a set of equivalent step sequences and each 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 $\pi_2$, 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 $[u]$ except $u$ 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 [12] and depend heavily on the $\odot$-closure construction and its properties.

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

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

**Definition 10.2.** Let $u \in S^*$. We define the relations $\prec_u, \sqsubseteq_u \subseteq \Sigma_u \times \Sigma_u$ as:

1. $\alpha \prec_u \beta \overset{df}{\iff} \alpha \prec_u \beta \land (l(\alpha), l(\beta)) \notin \text{ser}$,
2. $\alpha \sqsubseteq_u \beta \overset{df}{\iff} \alpha \prec_u \beta \land (l(\beta), l(\alpha)) \notin \text{ser}$. $\blacksquare$

**Lemma 10.1 ([12, Lemma 4.7]).** For all $u, v \in S^*$, if $u \equiv v$, then $\prec_u = \prec_v$ and $\sqsubseteq_u = \sqsubseteq_v$. $\square$

Definition 10.2 together with Lemma 10.1 describes two basic local invariants of the elements of $\Sigma_u$. The relation $\prec_u$ captures the situation when $\alpha$ always precedes $\beta$, and the relation $\sqsubseteq_u$ captures the situation when $\alpha$ never follows $\beta$.

**Definition 10.3.** Given a step sequence $u \in S^*$ and its respective comtrace $u = [u] \in S^*/\equiv$. We define the relational structures $S^{(u)}$ and $S_u$ as follows:

$$S^{(u)} \overset{df}{=} (\Sigma_u, \prec_u, \sqsubseteq_u)^\odot \quad S_u \overset{df}{=} \left(\Sigma_u, \bigcap_{x \in u} \prec_x, \bigcap_{x \in u} \sqsubseteq_x\right)$$
In contrast, a direct use of the relations $\prec, \sqsubseteq$ defined by the comtrace $\{a,b\}\{a,d\}$ can be reduced to computing transitive closure of relations, and can be done very efficiently.

Figure 2: An example of the relations $\sim, \ser$ on $E = \{a,b,c,d\}$, and the so-structure $(X, \prec, \sqsubseteq)$ defined by the comtrace $\{a,b\}\{a,d\} = \{\{a,b\}\{a,d\}, \{a\}\{b\}\{c\}\{a,d\}, \{a\}\{b\}\{c\}\{a,d\}\}$. The relational structure $S^{(u)}$ is the so-structure induced by the single step sequence $u$ and $S_u$ 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 10.2 ([12] and [13, Theorem 4.10 and Theorem 4.12]).** For all $u, v \in \mathcal{S}$, we have

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

In principle, Theorem 10.2 states that the so-structures $S^{(u)}$ and $S^{(v)}$ from Definition 10.3 are identical and their set of stratified extensions is exactly the comtrace $[u]$ with step sequences interpreted as stratified orders. However, from an algorithmic point of view, the definition of $S^{(u)}$ is much more interesting, since building the relations $\prec_u$ and $\sqsubseteq_u$ and getting their $\Diamond$-closures, which in turn can be reduced to computing transitive closure of relations, can be done very efficiently. In contrast, a direct use of the $S^{(u)}$ definition requires precomputing up to exponentially many elements of the comtrace $[u]$.

Figure 2 shows an example of a comtrace and the so-structure it generates.

Theorem 10.2 characterizes so-structures derived from a given comtrace and was proved mainly in [12]. However, the reciprocal problem which characterizes the comtrace derived from a given finite so-structure has not been formally dealt with so far. We will now study this problem.

Let $S = (X, \prec, \sqsubseteq)$ be a so-structure. For each stratified order $\prec \in \operatorname{ext}(S)$, recall that $\Omega_{\prec}$ denotes the step sequence defined by $\prec$. Note that each element appearing in $\Omega_{\prec}$ is unique so enumeration of elements in $X$ are not needed.

We will start with the definitions of relations simultaneity and serializability that are defined by a given so-structure $S = (X, \prec, \sqsubseteq)$. 

32
Definition 10.4. For each \(a, b \in X\):

1. \((a, b) \in \text{sim}_S \iff a <_\prec b\),
2. \((a, b) \in \text{ser}_S \iff a <_\prec b \land \neg(b \sqsubseteq a)\).

The names \textit{simultaneity} and \textit{serializability} are justified by the following result.

Proposition 10.1. For each \(a, b \in X\):

1. \((a, b) \in \text{sim}_S \iff \exists \varphi \in \text{ext}(S). a <_\varphi b\),
2. \((a, b) \in \text{ser}_S \iff (\exists \varphi \in \text{ext}(S). a <_\varphi b) \land (\exists \varphi \in \text{ext}(S). a <_\varphi b)\),
3. \((a, b) \not\in \text{ser}_S \iff a \not<_\varphi b \lor b <_\varphi a\).

PROOF. 1. This is a consequence of Theorem 4.1 and Theorem 4.2. We have:

\[
(a, b) \in \text{sim}_S \iff \\
\neg(a <_\varphi b) \land \neg(b <_\varphi a) \\
\neg(\forall \varphi \in \text{ext}(S). a <_\varphi b) \land \neg(\forall \varphi \in \text{ext}(S). b <_\varphi a) \\
(\exists \varphi \in \text{ext}(S). a <_\varphi b) \lor (\exists \varphi \in \text{ext}(S). a <_\varphi b) \\
\iff \exists \varphi \in \text{ext}(S). a <_\varphi b
\]  

(Definition 10.4) (Theorem 4.1) (Theorem 4.2)

2. Follows from (1) and Theorem 4.1
3. Follows from Definition 10.2, Theorem 4.1, and (2).

If stratified orders from \(\text{ext}(S)\) are interpreted as observations of concurrent histories (see Section 4 and [1, 11]), then \((a, b) \in \text{sim}_S\) means there is an observation in \(\text{ext}(S)\) where \(a\) and \(b\) are executed simultaneously and \((a, b) \in \text{ser}_S\) means there are equivalent observations where in one observation \(a\) and \(b\) are executed simultaneously, and in another \(b\) follows \(a\). Proposition 10.1 will often be used in the subsequent proofs.

Since each \(\varphi \in \text{ext}(S)\) can be interpreted as a step sequence, we define a relational structure induced by \(\varphi\) similarly to the definition of \(S^{\{u\}}\) for a given step sequence \(u\), but this time with great simplification.

Definition 10.5. For each \(\varphi \in \text{ext}(S)\), define \(S^{\{\varphi\}} \overset{df}{=} (X, \varphi \setminus \text{ser}_S, \varphi \lessdot \text{ser}_S^{-1})\).

It is worth noticing that the relational structure \(S^{\{\varphi\}}\) is a so-structure induced by \(\varphi\), and the relations \(\varphi \setminus \text{ser}_S, \varphi \lessdot \text{ser}_S^{-1}\) play the similar roles as \(\prec_u, \sqsubseteq_u\) from Definition 10.2 except the \(\forall\)-closure is not needed. The subtle reason behind this simplification is shown in the following proposition.

Proposition 10.2. For every \(\varphi \in \text{ext}(S)\), we have:

1. \(\varphi \setminus \text{ser}_S = \varphi = \varphi \setminus \text{ser}_S^{-1}\),
2. \(\varphi \lessdot \text{ser}_S^{-1} = \sqsubseteq = \varphi \lessdot \text{ser}_S^{-1}\).
3. \( S^{\langle \cdot \rangle} = (X, \prec, \sqsubset) \).

**Proof.** 1. We now show the first equality.

\[
a \prec b \land (a, b) \notin \text{ser}_S
\]

\[
\iff a \prec b \land (a \prec b \lor b \sqsubset a)
\quad \quad \quad \quad \quad \quad \text{ (Proposition 10.1.3) }
\]

\[
\iff (a \prec b \land a \prec b) \lor (a \prec b \land b \sqsubset a) \iff a < b \lor \text{False}
\quad \quad \quad \quad \quad \quad \text{ (Theorem 4.1) }
\]

For the second equality, we have \( < \setminus \text{ser}_S = (< \setminus \text{ser}_S) \setminus \text{ser}_S = < \setminus \text{ser}_S = < \).

2. We will show the first equality.

\[
a < \sim b \land (b, a) \notin \text{ser}_S
\]

\[
\iff a < \sim b \land (b \prec a \lor a \sqsubset b)
\quad \quad \quad \quad \quad \quad \text{ (Proposition 10.1.3) }
\]

\[
\iff (a < \sim b \land b \prec a) \lor (a < \sim b \land a \sqsubset b) \iff \text{False} \lor a \sqsubset b
\quad \quad \quad \quad \quad \quad \text{ (Theorem 4.1) }
\]

The second equality immediately follows.

3. Follows from (1) and (2). \( \square \)

We have just shown that every so-structure \( S = (X, \prec, \sqsubset) \) is equal to \( S^{\langle \cdot \rangle} \) for any \( \prec \in \text{ext}(S) \). Note that this proof does not assume that \( S \) is finite. Hence, Proposition 10.2 also holds when \( S \) is an infinite so-structure. This proposition can be interpreted as a generalization of the following folklore result on recovering a partial order from any of its total extension (by replacing partial orders with so-structures and total extensions with stratified extensions).

**Proposition 10.3.** For every partial order \( < \) and every total order \( \preceq \in \text{Total}(\langle \cdot \rangle) \), \( < \setminus \prec = < \).

**Proof.** \( a(< \setminus \prec) b \iff (a < b \land (a < b \lor b < a)) \iff a < b \). \( \square \)

We will end this section by proving that if \( S \) is a finite so-structure, then the set \( \text{ext}(S) \), when interpreted as a set of step sequences, is truly a comtrace over the comtrace alphabet \( (X, \text{sim}_S, \text{ser}_S) \). Moreover, the so-structure generated by this particular comtrace is exactly \( S \).

**Definition 10.6.** For every finite so-structure \( S = (X, \prec, \sqsubset) \), we define:

1. \( \Theta_S = (X, \text{sim}_S, \text{ser}_S) \).
2. \( \mathcal{C}(S) = \{ \Omega \prec \mid \prec \in \text{ext}(S) \} \).

Note that since \( \text{sim}_S \) and \( \text{ser}_S \) clearly satisfy the properties from Definition 5.1, the triple \( \Theta_S \) is a comtrace alphabet. So we can define the comtrace congruence \( \equiv_{\text{ser}_S} \) with respect to \( \Theta_S \). We will call \( \mathcal{C}(S) \) the comtrace generated by \( S = (X, \prec, \sqsubset) \). Theorem 10.3 the main result of this section will justify this name.

**Theorem 10.3.** Let \( S = (X, \prec, \sqsubset) \) be a finite so-structure. For every \( \prec \in \text{ext}(S) \), we have:

1. \( S^{\langle \Omega \prec \rangle} = S^{\langle \Omega \prec \rangle} = S^{\langle \cdot \rangle} = S \).
2. \( \mathcal{C}(S) = [\Omega \prec] \).

34
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 gso-structure and each finite gso-structure can be represented by a
g-comtrace. However the proofs and even formulations of those results are much longer and more complex than in the case of the relationship between contraces and so-structures. The difficulties stem mainly from the following facts:

- The definition of gso-structures is implicit, it involves using the induced so-structures (see Definition 4.3), which makes practically all definitions much more complex (especially the counterpart of \(-\)-closure), and the use of Theorem 4.3 more difficult than the use of Theorem 4.1.
- The internal property expressed by Theorem 4.2 which says that ext\((S)\) conforming to paradigm \(\pi_3\) of [11], does not hold for gso-structures.
- g-Contraces do not have a ‘natural’ canonical form with a well understood interpretation.
- The relation \(\text{inl}\) introduces plenty of irregularity and substantially increases the numbers of cases that need to be considered in many proofs.

In this chapter, we will prove the analogue of Theorem 10.2 showing that every g-comtrace uniquely determines a finite gso-structure.

11.1. 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 \(-\)-closure (see Definition 10.1 which was used in [12] and the previous section to construct finite so-structures from single step sequences or stratified orders.

**Definition 11.1 (Commutative Closure).**

Let \(G = (X, R_1, R_2)\) be any relational structure, and let \(R_3 = R_1 \cap R_2^\circ\). Using the notation from Definition 10.1 the commutative closure of the relational structure \(G\) is defined as

\[
G^\uparrow = \left( X, (\preceq_{R_3})_{\text{sym}} \cup R_1, \sqcap_{R_3R_2} \right).
\]

The motivation behind the above definition is similar to that for \(-\)-closure: for ‘reasonable’ \(R_1\) and \(R_2\), \((X, R_1, R_2)^\uparrow\) should be a gso-structure. Intuitively the \(\uparrow\)-closure is also a generalization of transitive closure for relations. Note that if \(R_1 = R_2\) then \((X, R_1, R_2)^\uparrow = (X, (R_1^\circ)^{\text{sym}}, R_1^\circ)\). Since the definition of gso-structures involves the definition of so-structures (see Definition 4.3), the definition of \(\uparrow\)-closure uses the concept of \(-\)-closure.

Note that we do not have an equivalence of Theorem 10.1 for \(\uparrow\)-closure. The reason is that \(\uparrow\)-closure is tailored to simplify the proofs that we will show in the next section rather than to be a closure operator by itself. Nevertheless, \(\uparrow\)-closure does have some general properties which are extremely useful in our proofs.

The first property shows that the relationship between \(\uparrow\)-closure and \(-\)-closure corresponds to the relationship between gso-structures and so-structures as stated in Definition 4.3.

**Proposition 11.1.** Let \((X, R_1, R_2)\) be a relational structure and \(R_3 = R_1 \cap R_2^\circ\). If \((X, \preceq_0, \sqsubseteq_0) = (X, R_3, R_2)^\uparrow\) is a so-structure, then \(\preceq_0 = (\preceq_{R_3})_{\text{sym}} \cup R_2\)\(\sqcap \sqsubseteq_0\).

**Proof.** (\(\subseteq\)) Since \((X, \preceq_0, \sqsubseteq_0) = (X, R_3, R_2)^\uparrow\), by definition of \(-\)-closure, \(\preceq_0 \subseteq \sqsubseteq_0\). Since we also have \(\preceq_0 \subseteq (\preceq_0 \cup R_2)\), it follows that \(\preceq_0 \subseteq (\preceq_{R_3})_{\text{sym}} \cup R_2\)\(\sqcap \sqsubseteq_0\).

(\(\supseteq\)) Suppose that \((x, y) \in (\preceq_{R_3})_{\text{sym}} \cup R_2\)\(\sqcap \sqsubseteq_0\) and \(\neg (x \preceq_0 y)\). There are two cases to consider:
in the similar manner as Definition 11.2. 

The following two useful operators for relations. We will elaborate and will require full use of the notation from Section 2.3 that allows us to define the formal relationship between step sequences and (labelled) stratified orders. We will also need the previous section. First we construct some relational invariants and next we will use Proposition 11.2. Generalized Stratified Order Structure Generated by a Step Sequence

Proposition 11.2. 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 $G_1^{\infty} \subseteq G_2^{\infty}$.

Proof. Let $R_3 = R_1 \cap R_2^{\circ}$ and $Q_3 = Q_1 \cap Q_2^{\circ}$. Since $R_1 \subseteq Q_1$ and $R_1 \subseteq Q_1$, we have $R_3 \subseteq Q_3$, and $(X, R_1, R_2)^{\circ} \subseteq (X, Q_1, Q_2)^{\circ}$, i.e., $\bowtie R_2 \subseteq \bowtie Q_2$ and $\bowtie R_3 \subseteq \bowtie Q_3$. This immediately implies $G_1^{\infty} \subseteq G_2^{\infty}$.

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

Proposition 11.3. If $G = (X, \bowtie, \bowtie)$ is a gso-structure then $G = G^{\bowtie}$.

Proof. Since $G$ is a gso-structure, by Definition 4.3, $S_G = (X, \bowtie, \bowtie)$ is a so-structure. Hence, by Theorem 10.14, $S_G = S_G^{\bowtie}$, which implies $\bowtie = (\bowtie \cup \bowtie)^{\bowtie}$, i.e., $\bowtie R_2 \subseteq \bowtie Q_2$ and $\bowtie R_3 \subseteq \bowtie Q_3$. This immediately implies $G^{\bowtie} = G^{\bowtie}$.

Hence, $(X, \bowtie, \bowtie) = (X, \bowtie, \bowtie)$ is a so-structure. By Theorem 10.14, we know $(X, \bowtie, \bowtie) = (X, \bowtie, \bowtie)^{\bowtie}$. So from Definition 11.3, $G^{\bowtie} = (X, \bowtie^{\bowtie} \cup \bowtie, \bowtie)$. Since $\bowtie$ is symmetric and $\bowtie \subseteq \bowtie$, we have $\bowtie^{\bowtie} \cup \bowtie = \bowtie$. Thus, $G = G^{\bowtie}$.

The remaining properties of $\bowtie$-closure have to be proved when needed for specific relations $R_1$ and $R_2$.

11.2. 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 $\bowt$-closure was used for $S^{(u)}$. However the construction will be more elaborate and will require full use of the notation from Section 2.3 that allows us to define the formal relationship between step sequences and (labelled) stratified orders. We will also need the following two useful operators for relations.

Definition 11.2. Let $R$ be a binary relation on $X$. We define the

- symmetric intersection of $R$ as $R^{\cap} \overset{df}{=} R \cap R^{-1}$, and
- the complement of $R$ as $R^{C} \overset{df}{=} (X \times X) \setminus R$.

37
Let \( \Theta = (E, \text{sim}, \text{ser}, \text{inl}) \) be a g-comtrace alphabet. Note that if \( u \equiv w \) then \( \Sigma_u = \Sigma_w \) so for every g-comtrace \( s = [s] \in \Sigma^*/\equiv \), we can define \( \Sigma_s = \Sigma_s \).

**Definition 11.3.** Given a step sequence \( s \in \Sigma^* \).

1. Let the relations \( \ll, \sqsubset, \ll, \sqsubset \subseteq \Sigma_s \times \Sigma_s \) be defined as follows:

\[
\alpha \ll \beta \iff (l(\alpha), l(\beta)) \in \text{inl} \quad (11.1)
\]

\[
\alpha \sqsubset \beta \iff \alpha \ll \beta \land (l(\beta), l(\alpha)) \notin \text{ser} \cup \text{inl} \quad (11.2)
\]

\[
\alpha \ll \beta \iff \alpha \ll \beta \land \forall (\alpha, \beta) \in \ll \cap (\ll)^\theta \circ \ll \circ (\ll)^\theta \setminus \text{ser} \cup \text{inl}
\]

\[
\forall ((l(\alpha), l(\beta)) \notin \text{ser} \cup \text{inl}) \land \exists \delta, \gamma \in \Sigma_s. \left( \delta \ll \gamma \land (l(\delta), l(\gamma)) \notin \text{ser} \land \delta \ll \gamma \land (l(\delta), l(\gamma)) \notin \text{ser} \land \alpha \ll \beta \right) \quad (11.3)
\]

2. The triple

\[
G^s \iff (\Sigma_s, \ll, \ll) \quad (11.4)
\]

is called the relational structure induced by the step sequence \( s \).

The intuition of Definition 11.3 is similar to that of Definition 10.2. Given a step sequence \( s \) and the g-comtrace alphabet \( (E, \text{sim}, \text{ser}, \text{inl}) \), without analyzing any other elements of \( [s] \) except \( s \) itself, we would like to be able to construct the same gso-structure as the one defined by the whole g-comtrace. So we will define appropriate “local” invariants \( \ll, \ll \) from the sequence \( s \).

(a) Equation 11.1 is used to construct the relationship \( \ll \), where two event occurrences \( \alpha \) and \( \beta \) might possibly be commutative because they are related by the inl relation.

(b) Equation 11.2 define 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) Equation 11.3 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 \text{ser}) and are not commutative ((\( \alpha, \beta \)) \notin \text{inl}).

(ii) \( \alpha \) and \( \beta \) are supposed to be commutative but they can be flipped 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 some distinct event occurrences \( \gamma \) and \( \delta \) which are squeezed between \( \alpha \) and \( \beta \) (always occur between \( \alpha \) and \( \delta \)) such that \( \delta \) occurs before \( \gamma \) and \( (\delta, \gamma) \) is not in ser (\( \delta \) and \( \gamma \) will never be put together into a single step).

38
After building all of these “local” invariants from the step sequence $s$, all of other “global” invariants which can be inferred from the axioms of the gso-structure definition are fully constructed by the commutative closure.

The next proposition will shows that the relations from $G^{(s)}$ really corresponds to positional invariants of all the step sequences from the g-comtrace $[s]$.

**Proposition 11.4.** Let $s \in S^*$, $G^{(s)} = (\Sigma_s, \lhd, \sqsubset)$, and $\equiv \cap \sqsubseteq$. If $\alpha, \beta \in \Sigma_s$, then

1. $\alpha \lhd \beta \iff \forall u \in [s]. \ pos_u(\alpha) \neq pos_u(\beta)$
2. $\alpha \sqsubset \beta \iff \alpha \neq \beta \land \forall u \in [s]. \ pos_u(\alpha) \leq pos_u(\beta)$
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 $\alpha \prec \beta$. \hfill $\Box$

Even though the results of the above proposition are expected and look deceptively simple, the proof is long and highly technical and can be found in Appendix A. We will next show that $G^{(s)}$ is indeed a gso-structure.

**Theorem 11.1.** Let $s \in S^*$. Then $G^{(s)} = (\Sigma_s, \lhd, \sqsubset)$ is a gso-structure.

**Proof.** Since $\lhd = \bigcap_{u \in [s]} <_{u^{sym}}$ and $<_{u^{sym}}$ is irreflexive and symmetric, $\lhd$ is irreflexive and symmetric. Since $\sqsubset = \bigcap_{u \in [s]} <_{u}$ and $<_{u}$ is irreflexive, $\sqsubset$ is irreflexive.

Let $\equiv \cap \sqsubseteq$, it remains to show that $S = (\Sigma, \prec, \sqsubset)$ satisfies the conditions S1–S4 of Definition 4.1. 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

\[
\alpha \sqsubset \beta \sqsubset \gamma \land \alpha \neq \gamma \implies (\alpha, \beta) \in \bigcap_{u \in [s]} <_{u} \land (\beta, \gamma) \in \bigcap_{u \in [s]} <_{u} \land \alpha \neq \gamma \quad \text{(Theorem 11.2)}
\]

\[
\implies \forall u \in [s]. \ pos_u(\alpha) \leq pos_u(\beta) \land pos_u(\beta) \leq pos_u(\gamma) \land \alpha \neq \gamma \quad \text{(Definition of $<_{u}$)}
\]

\[
\implies \alpha \sqsubset \gamma \quad \text{(Proposition 11.4(2))}
\]

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

\[
\alpha \prec \beta \sqsubset \gamma \implies (\alpha, \beta) \in \bigcap_{u \in [s]} (\lhd \cap <_{u^{sym}}) \land (\beta, \gamma) \in \bigcap_{u \in [s]} (\lhd \cap <_{u^{sym}}) \quad \text{(Theorem 11.2)}
\]

\[
\implies (\forall u \in [s]. \ pos_u(\alpha) \leq pos_u(\beta) \land pos_u(\alpha) \neq pos_u(\beta)) \land (\forall u \in [s]. \ pos_u(\beta) \leq pos_u(\gamma) \land pos_u(\beta) \neq pos_u(\gamma)) \quad \text{(Definition of $<_{u}$)}
\]

\[
\implies \forall u \in [s]. \ pos_u(\alpha) < pos_u(\gamma) \quad \text{(Proposition 11.4(3))}
\]

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

Theorem 11.2 justifies the following definition:

**Definition 11.4.** For every step sequence $s$, $G^{(s)} = (\Sigma_s, \lhd \cup \lhd, \sqsubset \cup \sqsubset)$ is the gso-structure induced by $s$. \hfill $\blacksquare$
Note that Proposition 11.4 also implies that we can construct the gso-structure $G^{(s)}$ 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 11.5.** For every $s \in S^*/\equiv$, we define $G_s = \left( \Sigma_s, \bigcap_{u \in s} \preceq^\text{sym}_u, \bigcap_{u \in s} \preceq_u \right)$. ■

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 11.2.** Let $s \in S^*$. Then $G^{(s)} = G_{[s]}$.

**Proof.** Let $G^{(s)} = (\Sigma_s, \ll, \sqsubseteq)$ and $\alpha, \beta \in \Sigma_r$. Then by Proposition 11.4, 1, 2), we have

\[
\alpha \ll \beta \iff \forall u \in |s|, \text{pos}_u(\alpha) \neq \text{pos}_u(\beta) \iff (\alpha, \beta) \in \bigcap_{u \in |s|} \preceq^\text{sym}_u
\]
\[
\alpha \sqsubseteq \beta \iff (\alpha \neq \beta \land \forall u \in |s|, \text{pos}_u(\alpha) \leq \text{pos}_u(\beta)) \iff (\alpha, \beta) \in \bigcap_{u \in |s|} (\preceq^\text{sym}_u)^\sim
\]

Hence, $G^{(s)} = (\Sigma_s, \ll, \sqsubseteq) = \left( \Sigma_s, \bigcap_{u \in |s|} \preceq^\text{sym}_u, \bigcap_{u \in |s|} \preceq_u \right) = G_{[s]}$. ■

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 11.3 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 11.5, we need to know either all the positional invariants or all elements of the g-comtrace. By Theorem 11.2, these two definitions are equivalent. In our proof, Definition 11.3 is more convenient when we want to deduce the properties of the gso-structure from a single step sequence and a g-comtrace alphabet. Definition 11.3 will be used to reconstruct the relations of a gso-structure when some positional invariants of a g-comtrace are known.

### 11.3. Generalized Stratified Order Structures Generated by Generalized Comtraces

In this section, we want to show that the construction from Definition 11.3 indeed yields a gso-structure representation of comtraces. But before doing so, we need some preliminary results.

**Proposition 11.5.** Let $s \in S^*$. Then $\preceq_s \in \text{ext}(G^{(s)})$.

**Proof.** Let $G^{(s)} = (\Sigma_s, \ll, \sqsubseteq)$. By Proposition 11.4 for all $\alpha, \beta \in \Sigma$,

\[
\alpha \ll \beta \implies \text{pos}_s(\alpha) \neq \text{pos}_s(\beta) \implies \alpha \ll_s \beta \lor \beta \ll_s \alpha \implies \alpha \ll^\text{sym}_s \beta
\]
\[
\alpha \sqsubseteq \beta \implies \text{pos}_s(\alpha) \leq \text{pos}_s(\beta) \implies \alpha \ll^\sim_s \beta
\]

Hence, by Definition 3.4 we get $\preceq_s \in \text{ext}(G^{(s)})$. ■

**Proposition 11.6.** Let $s \in S^*$. If $\ll \in \text{ext}(G^{(s)})$, then there is a step sequence $u \in S^*$ such that $\ll = \preceq_u$. 

Proof. Let \( G^{(i)} = (\Sigma, \preceq, \sqsubseteq) \) and \( \Omega_{\preceq} = B_1 \ldots B_k \). We will show that \( u = \text{l}[B_1] \ldots \text{l}[B_k] \) is a step sequence such that \( \preceq = \preceq_u \).

Suppose \( \alpha, \beta \in B_i \) be two distinct event occurrences such that \( \text{l}(\alpha), \text{l}(\beta) \notin \text{sim} \). Then \( \text{pos}_u(\alpha) \neq \text{pos}_u(\beta) \), which by Proposition 11.4 implies that \( \alpha \preceq \beta \). Since \( \preceq \in \text{ext}(G^{(i)}) \), by Definition 4.4 \( \alpha \prec \beta \) or \( \beta \prec \alpha \) contradicting that \( \alpha, \beta \in B_i \). Thus, we have shown for all \( B_i \) (1 \( \leq i \leq k \)),

\[
\alpha, \beta \in B_i \land \alpha \neq \beta \implies (\text{l}(\alpha), \text{l}(\beta)) \notin \text{sim}
\]  

(11.4) By Proposition A.1, if \( e^{(i)} \in \Sigma \) and \( i \neq j \) then \( \forall \alpha \in [s] : \text{pos}_u(e^{(i)}) \neq \text{pos}_u(e^{(j)}) \). So it follows from Proposition (11.3, 1) that \( e^{(i)} \sim e^{(j)} \). Since \( \preceq \in \text{ext}(G^{(i)}) \), by Definition 4.4

\[
\text{If } e^{(k_0)} \in B_k \text{ and } e^{(m_0)} \in B_m \text{, then } k_0 \neq m_0 \iff k \neq m
\]  

(11.5) From (11.4) it follows that \( u \) is a step sequence over \( \theta \). Also by (11.5), \( \text{pos}_u^{-1}\{i\} = B_i \) and \( |\text{l}[B_i]| = |B_i| \) for all \( i \). Hence, \( \Omega_{\preceq} = \Omega_{\preceq_u} \), which implies \( \preceq = \preceq_u \). \( \square \)

We want to show that two step sequences over the same \( g \)-comtrace alphabet induce the same gso-structure if and only if they belong to the same \( g \)-comtrace (Theorem 11.3 below). The proof of an analogous result for comtraces from [12] 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 \( < \) (see [12]). Nothing similar holds for g-comtraces. For g-comtraces both natural representations, GMC and MC, are not unique. The g-canonical representation (Definition 8.4) is unique but its uniqueness is artificial, it is induced by some total lexicographical order \( <_{\text{lex}} \) imposed on step sequences (Definition 8.3.3) from all GMC-representations we just have to pick the one that is lexicographically smallest. Nevertheless this lexicographical order \( <_{\text{lex}} \) will be one of the basic tools used in this subsection. However the lack of natural unique representation will make our reasoning much more difficult.

Lemma 11.1. Let \( s \) be a step sequence over a \( g \)-comtrace alphabet \( (E, \text{ser}, \text{sim}, \text{inl}) \) and \( <_E \) be any total order on \( E \). Let \( u = A_1 \ldots A_n \) be the \( g \)-canonical representation of \( [s] \) (i.e., \( u \) is the least element of the \( g \)-comtrace \( [s] \) w.r.t. \( <_{\text{lex}} \)). Let \( G^{(i)} = (\Sigma, \preceq, \sqsubseteq) \) and \( \preceq = \preceq \cap \sqsubseteq \). Let \( \text{mins}_\prec(X) \) denote the set of all minimal elements of \( X \) w.r.t. \( \prec \) and define

\[
Z(X) \overset{df}{=} \left\{ Y \subseteq \text{mins}_{\prec}(X) \mid (\forall \alpha, \beta \in Y. \neg(\alpha \prec \beta)) \land (\forall \alpha \in Y. \forall \beta \in X \setminus Y. \neg(\beta \sqsubseteq \alpha)) \right\}
\]

Let \( \overline{\pi} = A_{1} \ldots A_{n} \) be the enumerated step sequence of \( u \). Then \( A_i \) is the least element of the set \( \{ \text{l}[Y] \mid Y \in Z(\Sigma \cup \{A_{1} \ldots A_{i-1}\}) \} \) w.r.t. \( <_{\text{ext}} \). \( \square \)

Before presenting the proof, we will explain the intuitions behind the definition of the set \( Z(X) \). Let us consider \( Z(\Sigma) \) first. We want \( A_1 \) to be the least element of the set \( \{ \text{l}[Y] \mid Y \in Z(\Sigma) \} \). In short, we want 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 difficult part of this proof is to show that \( A_i \) must be a subset of the set of all minimal elements of \( (\Sigma, \prec) \) satisfying all the constraints of \( Z(\Sigma) \), i.e., each \( Y \in Z(\Sigma) \) satisfies:

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 \( \alpha \).
We then consider only $e^{(i)}$ and $e^{(j)}$. This lemma can be seen as an algorithm to build the g-canonical representation of $G^{(i)}$.

**Proof (Proof of Lemma 11.4).** We first notice that by Proposition 11.4, if $e^{(i)}, e^{(j)} \in \Sigma$ and $i < j$ then $e^{(i)} \prec e^{(j)}$. Hence, for all $\alpha, \beta \in \text{mins}(X)$, 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 $\beta \in \Sigma$, $\text{pos}_s(\beta) \geq \text{pos}_s(\alpha)$. Hence, by Proposition 11.4(3), $\neg(\beta \sim \alpha)$. Thus,

$$
\overline{A_1} \subseteq \text{mins}(X)
$$

For all $\alpha, \beta \in \overline{A_1}$, since $\text{pos}_s(\beta) = \text{pos}_s(\alpha)$, by Proposition 11.4(1), we have

$$
\neg(\alpha \sim \beta)
$$

For any $\alpha \in \overline{A_1}$ and $\beta \in \Sigma \setminus \overline{A_1}$, since $\text{pos}_s(\beta) < \text{pos}_s(\alpha)$, by Proposition 11.4(2),

$$
\neg(\beta \subset \alpha)
$$

From (11.6), (11.7) and (11.8), 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. $\prec^{st}$ is well defined.

Let $Y_0 \in Z(\Sigma)$ such that $B_0 = l(Y_0)$ be the least element of $\{l[Y] \mid Y \in Z(\Sigma)\}$ w.r.t. $\prec^{st}$. We want to show that $A_1 = B_0$. Since $\prec^{st}$ 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 $A_1 \in Z(\Sigma)$ 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 nonempty $W \subseteq Y_0$ there is an enumerated step sequence $v$ such that

$$
\overline{v} = W_0 \overline{W} \equiv \overline{A_1} \ldots \overline{A_k} \text{ and } W \subseteq W_0 \subseteq Y_0
$$

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

**Base case.** When $|W| = 1$, we let $\{\alpha_0\} = W$. We choose $\overline{v} = \overline{E_0} \ldots \overline{E_k}$ such that for all $\overline{v} = \overline{E_0} \ldots \overline{E_k}$, $\alpha_0 \in E_k$ (for $k \geq 0$) such that for all $\overline{v} = \overline{E_0} \ldots \overline{E_k}$, $\alpha_0 \in E_k$, we have

(i) $\text{weight}(\overline{E_0} \ldots \overline{E_k}) \leq \text{weight}(\overline{E_0} \ldots \overline{E_k})$,

(ii) $\text{weight}(\overline{E_{k-1}} \ldots \overline{E_k}) \leq \text{weight}(\overline{E_{k-1}} \ldots \overline{E_k})$.

We then consider only $\overline{W} = \overline{E_0} \ldots \overline{E_k}$. We observe by the way we chose $\overline{W}$, we have $\forall \beta \in [\overline{W}], \beta \neq \alpha_0 \Rightarrow \forall t \in [\overline{w}], \text{pos}_s(\beta) \leq \text{pos}_s(\alpha_0))$. Hence, since $\overline{w} = \overline{u} \oplus R \overline{W}$, it follows from Proposition 11.4(1, 2) that

$$
\forall \beta \in [\overline{w}], (\beta \neq \alpha_0 \Rightarrow \forall t \in [A_1 \ldots A_n], \text{pos}_s(\beta) \leq \text{pos}_s(\alpha_0))
$$

Then it follows from Proposition 11.4(2) that $\forall \beta \in [\overline{w}], (\beta \neq \alpha_0 \Rightarrow \beta \subset \alpha_0)$. But by the way $Y_0$ was chosen, we know that $\forall \alpha \in Y_0, \forall \beta \in \Sigma \setminus Y_0, \neg(\beta \subset \alpha)$. Hence,

$$
\overline{W} = (\overline{E_0} \cup \ldots \cup \overline{E_k}) \subseteq Y_0
$$

We next want to show

$$
\forall \alpha \in \overline{E_i}, \forall \beta \in \overline{E_j}, \{\alpha\} \{\beta\} \equiv \{\alpha, \beta\} \quad (0 \leq i < j \leq k)
$$
Suppose not. Then either \([\{\alpha\}\{\beta\}] = \{\{\alpha\}\{\beta\}\}\) or \([\{\alpha\}\{\beta\}] = \{\{\alpha\}\{\beta\}\} \{\alpha\}\). In either case, we have \(\forall t \in \{\{\alpha\}\{\beta\}\}\). pos(B) \(\neq pos(\beta)\). Since \(\{\alpha\}\{\beta\} \equiv \pi(a,\beta)\), by Proposition 7.3(3), \(\forall t \in \{\{\alpha\}\{\beta\}\}\). pos(B) \(\neq pos(\beta)\). So by Proposition 11.3 \(\alpha \Harr \beta\). This contradicts that \(Y_0 \in Z(\Sigma)\) and \(\alpha, \beta \in \Sigma(\overline{w}) \subseteq Y_0\). Thus, we have shown (11.11), which implies that for all \(\alpha \in E_i\) and \(\beta \in E_j\) \((0 \leq i < j \leq k)\), \((l(\alpha), l(\beta)) \in ser\). Then \(E_0 \ldots E_k \equiv \cup_{i=0}^{k} E_i\). Hence, by (11.10) and (11.11), there exists a step sequence \(v''\) such that \(v''(i) = \left(\cup_{j=0}^{n} E_i, v''(i) = A_1 \ldots A_n\right)\) and \(\{\alpha_0\} \subseteq \cup_{i=0}^{n} E_i \subseteq Y_0\).

**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 \(v_2 = F_0 \equiv A_1 \ldots A_n\), where \(W \setminus \{\beta_0\} \subseteq F_0 \subseteq Y_0\). If \(W \subseteq F_0\), we are done. Otherwise, proceeding like the base case, we construct a step sequence \(v_3\) such that \(v_3 = F_0 \equiv A_1 \ldots A_n\) and \(\{\beta_0\} \subseteq F_1 \subseteq Y_0\). Since \(F_0 \subseteq Y_0\), we have \(W \subseteq F_0 \cup F_1 \subseteq Y_0\). Then similarly to how we proved (11.11), we can show that \(\forall \alpha \in F_0, \forall \beta \in F_1\). \(\{\alpha\}\{\beta\} \equiv \{\alpha\}\). This means that for all \(\alpha \in F_0\) and \(\beta \in F_1\), \((l(\alpha), l(\beta)) \in ser\). Hence, \(F_0 \equiv F_1 \subseteq Y_0\). Hence, there is a step sequence \(v_4\) such that \(v_4 = (F_0 \cup F_1)\equiv A_1 \ldots A_n\) and \(W \subseteq (F_0 \cup F_1) \subseteq Y_0\).

Thus, we have shown (11.9). So by choosing \(W = Y_0\), we get a step sequence \(v\) such that \(v = W_0 \equiv A_1 \ldots A_n\), and \(W_0 \subseteq Y_0\). Hence, \(v = W_0 \equiv A_1 \ldots A_n\). Thus, \(v = B_0 \equiv A_1 \ldots A_n\). 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 \(\{l[Y] \mid Y \in Z(\Sigma \cup \{A_1 \ldots A_{i-1}\}\}\}\) w.r.t. \(^{st}\) by using induction on \(n\), the number of steps of \(A_1 \ldots A_n\). If \(n = 0\), we are done. If \(n \geq 1\), 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 \(p = A_2 \ldots A_n\), \(\Sigma_p = \Sigma \setminus A_1\), and its gso-structure \(\langle \Sigma_p, \subseteq \rangle \subseteq \subseteq \Sigma_p \times \Sigma_p\) \((n \geq 1)\), we get \(A_i\) is the least element of the set \(\{l[Y] \mid Y \in Z(\Sigma \cup \{A_1 \ldots A_{i-1}\}\}\}\) w.r.t. \(^{st}\) for all \(i \geq 2\).

**Theorem 11.3.** 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 11.2 \(G(s) = G(t)\).

\((\Leftarrow)\) By Lemma 11.1 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 \(G(s) = G(t)\) and the construction is unique, we get \(w_1 = w_2\). Hence, \(s \equiv t\).

**Theorem 11.3** justifies the following definition:

**Definition 11.6.** For every g-comtrace \([s]\), \(G([s]) = G(s) = (\Sigma_r, \subseteq, \subseteq, \subseteq \subseteq, \subseteq)\) is the gso-structure induced by the g-comtrace \([s]\).

To end this section, we prove two major results. Theorem 11.4 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 11.5 says that the gso-structure induced by a g-comtrace is uniquely identified by any of its stratified extensions.

**Lemma 11.2.** Let \(s, t \in S^*\) and \(\leq_s \in ext(G([t]))\). Then \(G(s) = G([t])\).
12. Generalized Comtraces Representing Finite Generalized Stratified Order Structures

In this section we will show that every finite gso-structure can be represented by a g-comtrace. Let $G = (\mathcal{X}, \prec, \sqsubseteq)$ be a finite gso-structure and let $\Delta = \{ \Omega, \sigma \mid \sigma \in \text{ext}(G) \}$ be the set of all step sequences defined by the elements of $\text{ext}(G)$. Note that each “event” in $\mathcal{X}$ only occurs in each sequence of $\Delta$ at most once. Hence, we do not need to enumerate the sequences in $\Delta$ or to use the label function $l$ in the definitions and proofs.

We will start with the definition of relations simultaneity, serializability and interleaving induced by a given gso-structure $G = (\mathcal{X}, \prec, \sqsubseteq)$. The following definition is an extension of Definition 10.4 to gso-structures.
Definition 12.1. For each \( a,b \in X \):
1. \((a,b) \in \text{sim}_G \iff \neg(a \ll b)\),
2. \((a,b) \in \text{ser}_G \iff \neg(a \ll b) \land \neg(b \sqsubset a)\),
3. \((a,b) \in \text{inl}_G \iff a \ll b \land \neg(a \sqsubset b \sqcup b \sqsubset a)\).

\[\text{Definition 12.1} \text{ is a generalization of Definition 10.4 as shown in the following proposition.}\]

We recall that from Definition 4.3 we defined \( \sim_G \) and its respective properties. (1), (2) and (3) follow from Theorem 4.3; (4) follows from (1), (2) and Theorem 4.3.

Proposition 12.1.
1. \((a,b) \in \text{sim}_G \iff a \prec_G b\),
2. \((a,b) \in \text{ser}_G \iff (a,b) \in \text{sim} \land \exists \triangle \in \text{ext}(G). a \ll b\),
3. \((a,b) \in \text{inl}_G \iff (a,b) \notin \text{sim} \land \exists \triangle \in \text{ext}(G). a \ll b \land \exists \triangle \in \text{ext}(G). b \ll a\).
4. \((\exists \triangle \in \text{ext}(G). a \ll b) \land (a,b) \notin \text{ser}_G \iff (\forall \triangle \in \text{ext}(G). a \ll \triangle b) \iff a \ll b\)

PROOF. (1), (2) and (3) follow from Theorem 4.3; (4) follows from (1), (2) and Theorem 4.3.

Definition 12.2. For each stratified order \( \triangle \in \text{ext}(G) \), we define
\[G^{\triangle} \overset{df}{=} \left( X, (\triangle \cap \text{ser}_G)^\text{sym} \cup \text{inl}_G, \triangledown \cap (\text{ser}_G^{-1} \cup \text{inl}_G) \right)\]

The following proposition shows that \( G^{\triangle} \) and \( G \) are identical gso-structures. Similarly to the relationship between \( S^{\triangle} \) and \( S \), it is also interesting to observe that we do not need commutative closure to build \( G \) from a stratified order \( \triangle \in \text{ext}(G) \). The reason behind this simplification is explained in the following proposition, which is a generalization of Proposition 10.2.

Proposition 12.3. For every \( \triangle \in \text{ext}(G) \), we have:
1. \((\triangle \cap \text{ser}_G)^\text{sym} \cup \text{inl}_G = \text{ser}_G - \text{ser}_G\text{sym}\),
2. \((\triangle \cap (\text{ser}_G^{-1} \cup \text{inl}_G) = \text{ser}_G \cup (\text{ser}_G^{-1} \cup \text{inl}_G)\),
3. \(G^{\triangle} = (X, \triangle, \sqsubset)\),
4. \(\prec_G = \triangle \cap (\text{ser}_G \cup \text{inl}_G)\),
5. \(\triangledown = \text{ser}_G \cup \text{inl}_G\).
Lemma 12.1. Let $G$ be a finite gso-structure and let $\Delta = \{ \omega_\omega \mid \omega \in \text{ext}(G) \}$. Then we have:

$$(a, b) \in (\prec \setminus \text{ser}_G)^{\text{sym}} \cup \text{inl}_G$$

\[ \iff \quad (a \prec b \land (a, b) \notin \text{ser}_G) \lor (b \prec a \land (b, a) \notin \text{ser}_G) \lor (a, b) \in \text{inl}_G \] (Definition 12.1(3))

\[ \iff \quad a \not\succ b \lor (a \not\prec b \land \neg(b \sqsubseteq a \lor a \sqsubseteq b)) \] (Proposition 12.2(4))

\[ \iff \quad a \not\succ b \lor (a \not\prec b \land \neg(b \sqsubseteq a \lor a \sqsubseteq b)) \] (Definition 12.1(3))

\[ \iff \quad a \not\prec b \lor (a \not\prec b \land \neg(b \sqsubseteq a \lor a \sqsubseteq b)) \] (Theorem 4.3)

3. Follows from (1) and (2).

4. For every $a, b \in X$, we have

$$a, b \in \prec \setminus \text{ser}_G \cup \text{inl}_G$$

\[ \iff \quad a \not\succ b \land (a \sqsubseteq b \lor (a \not\prec b \land b \sqsubseteq a)) \] (Similarly to proof of (2))

\[ \iff \quad a \not\prec b \land (a \not\prec b \land a \sqsubseteq \text{sym} b) \] (Theorem 4.3)

5. Follows from (1), (4) and the fact that $\text{inl}_G$ is symmetric.

We have just shown that every so-structure $G = (X, \prec, \sqsubseteq)$ is equal to $G^{\prec}$ for any stratified order $\prec \in \text{ext}(G)$. Note that since the proof does not assume that $G$ is finite, Proposition 12.3 also holds when $G$ is an infinite gso-structure.

Before stating the main theorem of this section, we need the following definition.

**Definition 12.3.** For each finite gso-structure $G = (X, \prec, \sqsubseteq)$, we define:

1. $\Theta_G \overset{df}{=} (X, \text{sim}_G, \text{ser}_G, \text{inl}_G)$,
2. $g\mathcal{C}(G) \overset{df}{=} \{ \omega_\omega \mid \omega \in \text{ext}(G) \}$.

Observe that $\text{ser}_G \subseteq \text{sim}_G$, the relations $\text{sim}$ and $\text{inl}$ are symmetric, $\text{sim}_G \cap \text{inl}_G = \emptyset$, and all three relations are irreflexive, so $\Theta_G$ is a g-comtrace alphabet. Hence we can define the relations $\overset{\approx}{=}_{\text{ser,inl}}$ and $\overset{\sim}{=}_{\text{ser,inl}}$ with respect to the g-comtrace alphabet $\Theta_G$. We will call $g\mathcal{C}(G)$ the g-comtrace generated by the gso-structure $G$. Theorem 12.1 below will justify this name.

**Lemma 12.1.** Let $G = (X, \prec, \sqsubseteq)$ be a finite gso-structure and let $\Delta = \{ \omega_\omega \mid \omega \in \text{ext}(G) \}$. Then we have:
1. $\Delta$ is a set of step sequences over $\Theta_G$.
2. If $u \in \Delta$, then

$$\preceq_u = \text{inl}_G \quad (12.1)$$
$$\sqsubseteq_u = \preceq \setminus (\text{ser}^{-1}_G \cup \text{inl}_G) = \sqsubseteq \quad (12.2)$$
$$\prec_u = \preceq \setminus (\text{ser}_G \cup \text{inl}_G) = \prec_G \quad (12.3)$$

**Proof (Proof of Lemma 12.1).**

1. We need to check that every element of $\Delta$ is a step sequence over the $g$-comtrace alphabet $\Theta_G$. Let $\Omega_\Theta = A_1 \ldots A_n \in \Delta$. Then since $\prec \in \text{ext}(G)$, for each $A_i \ (1 \leq i \leq n)$, we have if $a, b \in A_i$ and $a \neq b$, then $a \prec b$. Hence, from Proposition 12.2.1, we have $(a, b) \in \text{sim}_G$.
2. The equality from (12.1) immediately follows from (11.1) of Definition 11.3.
   The first and second equalities from (12.2) follow from (11.2) of Definition 11.3 and Proposition 12.3.2.
   The second equality (12.3) follows from Proposition 12.3.3. It remains to show the first equality of (12.3).
   (2) Follows from (11.2) of Definition 11.3.
   (3) We let $\prec$ be the stratified order in $\text{ext}(G)$ such that $u = \Omega_\Theta$. It suffices to show that for every $a, b \in X$, the fact that $a \prec b$ and

$$\left( (a, b) \in \preceq_u \cap \left( (\sqsubseteq_u)^{\circ} \circ \preceq_u \circ (\sqsubseteq_u)^{\circ} \right) \right)$$

leads to a contradiction. There are two cases to consider:

(a) If $a \prec b$ and $(a, b) \in \preceq_u \cap \left( (\sqsubseteq_u)^{\circ} \circ \preceq_u \circ (\sqsubseteq_u)^{\circ} \right)$, then there must be $c, d \in X$ such that $a \sqsubseteq u = c$, $b \sqsubseteq u = d$ and $(c, d) \notin \text{inl}_G$. Since we know from (2) that $\sqsubseteq_u = \sqsubseteq$, we have $a \sqsubseteq_u = c$ and $b \sqsubseteq_u = d$. Since $\sqsubseteq = \sqsubseteq \cup \text{id}_X$, there are three cases to consider.

   If $a \sqsubseteq c \land c \sqsubseteq u \sqsubseteq a \sqsubseteq b \sqsubseteq b$, then it follows that $\forall \prec \in \text{ext}(G), (a \prec c \land d \prec b)$. Since $a \prec c \sqsubseteq c \prec d \sqsubseteq u \sqsubseteq d$, which means $(a, b) \notin \text{sim}_G \land (\exists \prec \in \text{ext}(S), a \prec b) \land (\exists \prec \in \text{ext}(S), b \prec a)$. Thus, $(c, d) \notin \text{sim}_G \land (\exists \prec \in \text{ext}(S), b \prec a)$, contradicting that $(c, d) \notin \text{inl}_G$. Similarly, we can show that the remaining two cases $(a = c \land d \sqsubseteq a \sqsubseteq b \sqsubseteq d)$ and $(a \sqsubseteq c \land c \sqsubseteq a \sqsubseteq b)$ also lead to a contradiction.

(b) If $a \prec_u b$ and $(a, b) \in \text{ser}_G$ and $\exists c, d \in X$. Then since we know from (2) that $\sqsubseteq_u = \sqsubseteq$, we have $a \sqsubseteq u \sqsubseteq c \sqsubseteq u \sqsubseteq d \sqsubseteq u$.

   Since we have $\sqsubseteq = \sqsubseteq \cup \text{id}_X$, it follows that $\forall w \in \Delta, (\text{pos}_w(a) \leq \text{pos}_w(c) \leq \text{pos}_w(b) \land \text{pos}_w(a) \leq \text{pos}_w(d) \leq \text{pos}_w(b))$. But since $(a, b) \in \text{ser}_G$, there is some $v \in \Delta$ such that $\text{pos}_v(a) = \text{pos}_v(b)$. So $\text{pos}_v(c) = \text{pos}_v(d)$, i.e., $(c, d) \in \text{sim}_G$. This and $c \prec_u d$ imply that $(c, d) \in \text{ser}_G$, a contradiction.

**Theorem 12.1.** Let $G = (X, \prec, \sqsubseteq)$ be a finite gso-structure. For every $\prec \in \text{ext}(G)$, we have:

1. $G[\Omega_\Theta] = G[\Omega_\Theta] = G[\Theta_\Theta] = G$.
2. $g \mathfrak{C}(G) = [\Omega_\Theta]$. □

47
from Definition 11.1, we get ⊏ and (≺)

This implies that are in fact equivalent models.

Lemma 12.1, we get

Figure 5: A gso-structure

13. Conclusion and Future Work

The concept of a comtrace is revisited and its extension, the g-comtrace, is introduced. Both comtraces and g-comtraces can be seen as generalizations of Mazurkiewicz traces. We analyzed some algebraic and linguistic properties of comtraces and g-comtraces, where an interesting application of the algebraic properties of comtraces 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 traces and
comtraces have a natural unique canonical form which corresponds to their maximal concurrent representation, while the only unique canonical representation of a g-comtrace is by choosing the lexicographically least element of the g-comtrace. We also revisit the mutual relationship between comtraces and so-structures from [12] and show that each comtrace can be uniquely represented by a so-structure.

The most important contribution of this paper is the study of the mutual relationship between g-comtraces and gso-structures. The major technical results, Theorems 11.4 and 12.1, can be seen as the generalizations of Stpilrajn’s Theorem in the context of g-comtraces. Furthermore, Theorems 11.4 and 12.1 ensure that g-comtraces and finite gso-structures can uniquely represented by one another. We believe the reason the proofs of 11.4 and 12.1 are more technical than similar theorems of comtraces is that both comtraces and so-structures satisfy paradigm π, while g-comtraces and gso-structures do not. Intuitively, what paradigm π really says is that the underlying structure is really a partial order. 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. Note that 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” relations must belong to a synchronous step. Thus, if we collapse each synchronous set into a single vertex, than the resulting relation is a partial order. When paradigm π is not satisfied, we have much more than a partial order structure, and hence common techniques that depend too much on the underlying partial order structure of comtraces and so-structures will not work for g-comtraces and gso-structures.

Despite some obvious advantages, for instance very handy composition and no need to use labels, quotient monoids (perhaps with some exception of traces) are much 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, advanced quotient monoids, e.g., comtraces and g-comtraces, could 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 as recognisability [24], acceptability [30], etc. are still open.

References

[1] W. Bartussek, D. L. Parnas, Using Assertions About Traces to Write Abstract Specifications for Software Modules, Lecture Notes in Computer Science 65, Springer 1978, pp. 211–236.
[2] P. Cartier and D. Foata, Problèmes combinatoires de commutation et ré-arrangements, Lecture Notes in Mathematics 85, Springer 1969.
[3] P. M. Cohn, Universal Algebra, D. Reidel 1981.
[4] R. Davillers, R. Janicki, M. Koutny, P. E. Lauer, Concurrent and Maximally Concurrent Evolution of Non-sequential Systems, Theoretical Computer Science 43:213-238, 1986.
[5] V. Diekert and G. Rozenberg (eds.), The Book of Traces, World Scientific 1995.
[6] H. Gaifman and V. Pratt, Partial Order Models of Concurrency and the Computation of Function, Proc. of LICS’87, pp. 72-85.
[7] G. Guo and R. Janicki, Modelling Concurrent Behaviours by Commutativity and Weak Causality Relations, Proc. of AMAST’02, LNCS 2422 (2002), 178-191.
[8] C. A. R. Hoare, Communicating Sequential Processes, Prentice-Hall, 1986.
[9] R. Janicki. Relational Structures Model of Concurrency, Acta Informatica, 45(4): 279-320, 2008.
[10] R. Janicki and M. Koutny, Structure of Concurrency Theory, LNC5 506, Springer 1991, pp. 59-74.
[11] R. Janicki and M. Koutny, Structure of Concurrency, Theoretical Computer Science, 112(1): 5–52, 1993.
[12] R. Janicki and M. Koutny, Semantics of Inhibitor Nets, Information and Computation, 123(1): 1–16, 1995.
[13] R. Janicki and M. Koutny, Fundamentals of Modelling Concurrency Using Discrete Relational Structures, Acta Informatica, 34: 367–388, 1997.
Appendix A: Proof of Proposition \[11.4\]

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

1. \( \text{pos}_u(\alpha) \neq \text{pos}_u(\beta) \)
2. If \( \text{pos}_u(\alpha) < \text{pos}_u(\beta) \) and \( v \) is a step sequence satisfying \( v \equiv u \), then \( \text{pos}_v(\alpha) < \text{pos}_v(\beta) \).

Proof. 1. Follows from the fact that \( \text{sim} \) is irreflexive.

2. It suffices to show that if \( \text{pos}_u(\alpha) < \text{pos}_u(\beta) \) and \( \forall \approx \exists !, \) then \( \text{pos}_v(\alpha) < \text{pos}_v(\beta) \). But this is clear from Proposition \[6.1\] and the fact that \( \text{ser} \) and \( \text{inl} \) are irreflexive. \( \square \)

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

1. If \( \text{pos}_u(\alpha) < \text{pos}_u(\beta) \) and \( \text{pos}_w(\alpha) > \text{pos}_w(\beta) \) then there are \( x, y, A, B \) with \( \forall \approx \exists ! \) such that \( \approx \equiv \text{inl}_A \text{inl}_B \approx \text{inl}_A \beta \in \exists ! \), and \( \exists ! \alpha, \beta \in \exists ! \).
2. If \( \text{pos}_u(\alpha) = \text{pos}_u(\beta) \) and \( \text{pos}_w(\alpha) > \text{pos}_w(\beta) \) then there are \( x, y, A, B, C \) with \( \forall \approx \exists ! \) such that \( \approx \equiv \text{inl}_A \text{inl}_B \text{inl}_C \approx \text{inl}_A \beta \in \exists ! \) and \( \exists ! \alpha, \beta \in \exists ! \).

Proof. 1. Assume that \( \text{pos}_u(\alpha) < \text{pos}_u(\beta) \) and \( \text{pos}_w(\alpha) > \text{pos}_w(\beta) \). Since \( u(\approx \cup \approx^{-1})w \), we observe that

- If \( \approx \equiv \text{inl}_A \text{inl}_B \approx \text{inl}_A \beta \in \exists ! \), then \( \forall \approx, \beta \in \exists !(\approx) \), \( \text{pos}_u(\alpha) < \text{pos}_u(\beta) \Rightarrow \text{pos}_w(\alpha) < \text{pos}_w(\beta) \).
Proposition A.3. Let $s$ be a step sequence over a g-comtrace alphabet $(E, \text{sim}, \text{ser}, \text{inl})$. If $\alpha, \beta \in \Sigma_s$, then

1. $\alpha \prec_s \beta \quad \Rightarrow \quad \forall u \in [s], \text{pos}_u(\alpha) \neq \text{pos}_u(\beta)$,
2. $\alpha \sqsubseteq_s \beta \quad \Rightarrow \quad \forall u \in [s], \text{pos}_u(\alpha) \leq \text{pos}_u(\beta)$,
3. $\alpha \prec_s \beta \quad \Rightarrow \quad \forall u \in [s], \text{pos}_u(\alpha) < \text{pos}_u(\beta)$.

Proof. 1. Assume that $\alpha \prec_s \beta$. Then, by (11.1), $(l(\alpha), l(\beta)) \in \text{inl}$. This implies that $l(\alpha) \neq l(\beta)$, so $\alpha \neq \beta$. Also since inl $\cap$ sim = $\emptyset$, there is no step $A$ where \{l(\alpha), l(\beta)\} $\in A$. Hence, $\forall u \in [s], \text{pos}_u(\alpha) \neq \text{pos}_u(\beta)$.

2. Assume that $\alpha \sqsubseteq_s \beta$. Suppose that $\exists u \in [s], \text{pos}_u(\alpha) > \text{pos}_u(\beta)$. Then there must be some $u_1, u_2 \in [s]$ such that $u_1(\approx \cup \approx^1)u_2$ and $\text{pos}_{u_1}(\alpha) \leq \text{pos}_{u_1}(\beta)$ and $\text{pos}_{u_2}(\alpha) > \text{pos}_{u_2}(\beta)$. There are two cases:

a. If $\text{pos}_{u_1}(\alpha) < \text{pos}_{u_1}(\beta)$ and $\text{pos}_{u_2}(\alpha) > \text{pos}_{u_2}(\beta)$, then by Proposition A.2 (1) there are $x, y, A, B$ such that $\pi = xABy(\approx \cup \approx^1)xB\gamma = \pi_1$ and $\alpha \in A, \beta \in B$. Hence, $(l(\alpha), l(\beta)) \in \text{inl}$, which by (11.2) contradicts that $\alpha \sqsubseteq_s \beta$.

b. If $\text{pos}_{u_1}(\alpha) = \text{pos}_{u_1}(\beta)$ and $\text{pos}_{u_2}(\alpha) > \text{pos}_{u_2}(\beta)$, then it follows from Proposition A.2 (2) that there are $x, y, A, B, C$ such that $\pi = xAy \approx \pi_2$ and $\beta \in B$ and $\alpha \in C$. Thus, $(l(\beta), l(\alpha)) \in \text{ser}$, which by (11.2) contradicts that $\alpha \sqsubseteq_s \beta$.

3. Assume that $\alpha \prec_s \beta$. Suppose that $\exists u \in [s], \text{pos}_u(\alpha) \geq \text{pos}_u(\beta)$. Then must be some $u_1, u_2 \in [s]$ such that $u_1(\approx \cup \approx^1)u_2$ and $\text{pos}_{u_1}(\alpha) < \text{pos}_{u_1}(\beta)$ and $\text{pos}_{u_2}(\alpha) > \text{pos}_{u_2}(\beta)$. There are two cases:

a. If $\text{pos}_{u_1}(\alpha) < \text{pos}_{u_1}(\beta)$ and $\text{pos}_{u_2}(\alpha) = \text{pos}_{u_2}(\beta)$, then it follows from Proposition A.2 (2) that there are $x, y, A, B, C$ such that $\pi = xAy \approx \pi_2$ and $\alpha \in B$ and $\beta \in C$. Thus, $(l(\alpha), l(\beta)) \in \text{ser}$ and $(l(\omega), l(\gamma)) \neq (l(\delta), l(\gamma)) \notin \text{ser}$. Hence, it follows from (11.3) that $\exists \delta, \gamma \in \Sigma_s: \text{pos}_0(\delta) < \text{pos}_0(\gamma)$ and $l(\delta), l(\gamma) \notin \text{ser}$. By (2) and transitivity of $\leq$, we have $\text{pos}_u(\alpha) \leq \text{pos}_u(\delta) \leq \text{pos}_u(\beta)$ and $\forall u \in [s], \text{pos}_u(\alpha) \leq \text{pos}_u(\gamma) \leq \text{pos}_u(\beta)$. But since $\alpha, \beta \in \overline{E} \cup \overline{E} = \overline{A}$, it follows that $\{\gamma, \delta\} \subseteq \overline{A}$, which implies $\text{pos}_{\overline{A}}(\gamma) = \text{pos}_{\overline{A}}(\delta)$. Since we also have $\text{pos}_{\overline{A}}(\delta) < \text{pos}_{\overline{A}}(\gamma)$, it follows that Proposition A.2 (2) that there are $\pi, w, D, E, F$ such that $\pi = xAw \approx \pi_2$ and $\delta \in \overline{E}$ and $\gamma \in \overline{F}$. Thus, $(l(\delta), l(\gamma)) \in \text{ser}$, a contradiction.
If \( pos_{\alpha} (\alpha) < pos_{\beta} (\beta) \) and \( pos_{\alpha} (\alpha) > pos_{\beta} (\beta) \), then by Proposition A.3, 1) there are \( x,y,Z,B \) such that \( \overline{Z} = \frac{x\overline{A}A B y}{x \overline{A} A B y} (\approx \cup \approx^{-1}) \frac{x\overline{B}A y}{x \overline{B} A y} = \overline{x}A, B, y \in \overline{B}. \) Thus, \( (l(\alpha), l(\beta)) \in \text{inl}. \) Since we assume \( \alpha <_{\mathcal{R}} \beta, \) by (11.3), we have \( (\alpha, \beta) \in \langle \approx \cup \rangle \cap (\approx \setminus \langle \approx \cup \rangle \circ (\approx \setminus \langle \approx \cup \rangle)). \) So there are some \( \gamma, \delta \) such that \( \alpha (\approx \setminus \langle \approx \cup \rangle) \gamma \sim \delta (\approx \setminus \langle \approx \cup \rangle) \beta. \) Observe that

\[
\alpha (\approx \setminus \langle \approx \cup \rangle) \gamma \Rightarrow \exists \alpha \in \overline{A} \gamma \wedge (\approx \setminus \langle \approx \cup \rangle) \alpha
\]

\[
\forall \alpha \in [\overline{A}], \text{pos}_{\alpha}(\alpha) \leq \text{pos}_{\alpha}(\gamma) \wedge \forall \alpha \in [\overline{A}], \text{pos}_{\alpha} (\gamma) \leq \text{pos}_{\alpha} (\alpha)
\]

\[
\Rightarrow \forall \alpha \in [\overline{A}], \text{pos}_{\alpha} (\alpha) = \text{pos}_{\alpha} (\gamma)
\]

\[
\Rightarrow \{\alpha, \gamma\} \subseteq \overline{A}
\]

Similarly, since \( \delta (\approx \setminus \langle \approx \cup \rangle) \beta, \) we can show that \( \{\delta, \beta\} \subseteq \overline{B}. \) Since \( \overline{\overline{X}} \overline{Y} (\approx \cup \approx^{-1}) \overline{X} \overline{Y}, \) we get \( A \times B \subseteq \text{inl}. \) So \( (l(\gamma), l(\delta)) \in \text{inl}. \) But \( \gamma \sim \delta \) implies that \( (l(\gamma), l(\delta)) \notin \text{inl}, \) a contradiction. \( \square \)

**Proposition A.4.** Let \( s \) be a step sequence over a g-comtrace alphabet \( (E, \sim, \ser, \text{inl}) \) and \( G^{(s)} = (\Sigma, \leftarrow, \circ). \) If \( \alpha, \beta \in \Sigma, \) then

1. \( \alpha \leftarrow \beta \Rightarrow \forall \alpha \in [\overline{A}], \text{pos}_{\alpha}(\alpha) \neq \text{pos}_{\beta}(\beta) \)
2. \( \alpha \circ \beta \Rightarrow (\alpha \neq \beta \wedge \forall \alpha \in [\overline{A}], \text{pos}_{\alpha}(\alpha) \leq \text{pos}_{\beta}(\beta)) \)

**Proof.** Follows from Definitions 11.3 and 11.1 and Proposition A.3. \( \square \)

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

1. Step A is called **serializable** iff

\[
\exists B, C \in \overline{A}, B \cup C = A \wedge B \times C \subseteq \ser.
\]

Step A is called **non-serializable** iff A is not serializable. (Note that every non-serializable step is a synchronous step as defined in Definition 5.5.)

2. Step A is called **serializable to the left of a** iff

\[
\exists B, C \in \overline{A}, B \cup C = A \wedge a \in B \wedge B \times C \subseteq \ser.
\]

Step A is called **non-serializable to the left of a** iff A is not serializable to the left of a, i.e., \( \forall B, C \in \overline{A}, (B \cup C = A \wedge a \in B) \Rightarrow B \times C \not\subseteq \ser. \)

3. Step A is called **serializable to the right of a** iff

\[
\exists B, C \in \overline{A}, B \cup C = A \wedge a \in C \wedge B \times C \subseteq \ser.
\]

Step A is called **non-serializable to the right of a** iff A is not serializable to the right of a, i.e., \( \forall B, C \in \overline{A}, (B \cup C = A \wedge a \in C) \Rightarrow B \times C \not\subseteq \ser. \)

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

1. If A is non-serializable to the left of \( l(\alpha) \) for some \( \alpha \in \overline{A}, \) then \( \alpha \circ_{\overline{A}} \beta \) 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, observe that for all $\alpha, \beta \in \overline{A}$, $(l(\alpha), l(\beta)) \notin \text{inl}$. Hence, by Definition 11.3, we have

$$\alpha \sqsubset_{A} \beta \iff \text{pos}_{A}(\alpha) \leq \text{pos}_{A}(\beta) \wedge (l(\alpha), l(\beta)) \notin \text{ser}.$$ 

**Proof.**

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:

$$RC^{0}(\alpha) \overset{df}{=} \{ \alpha \} \quad RC^{n}(\alpha) \overset{df}{=} \{ \delta \in \overline{A} \mid \exists \gamma \in RC^{n-1}(\alpha) \wedge \gamma \sqsubset_{A} \delta \}$$

We want to prove that if $\overline{A} \setminus RC^{n}(\alpha) \neq \emptyset$ then $|RC^{n+1}(\alpha)| > |RC^{n}(\alpha)|$. Assume that $\overline{A} \setminus RC^{n}(\alpha) \neq \emptyset$. Since $\alpha \in \overline{A}$ and $A$ is non-serializable to the right of $l(\alpha)$, $l(\overline{A} \setminus RC^{n}(\alpha)) \times l[RC^{n}(\alpha)] \subseteq \text{ser}$. Thus there exists some $\gamma \in \overline{A} \setminus RC^{n}(\alpha)$ such that there is some $\delta \in RC^{n}(\alpha)$ satisfying $(l(\gamma), l(\delta)) \notin \text{ser}$. This implies $\delta \sqsubset_{A} \gamma$. Thus, $\gamma \in RC^{n+1}(\alpha)$ where $\gamma \notin RC^{n}(\alpha)$. So $|RC^{n+1}(\alpha)| > |RC^{n}(\alpha)|$ as desired.

Since $A$ is finite and if $\overline{A} \setminus RC^{n}(\alpha) \neq \emptyset$ then $|RC^{n+1}(\alpha)| > |RC^{n}(\alpha)|$, for some $n < |A|$, we must have $RC^{n}(\alpha) = \overline{A}$. Thus, $\beta \in RC^{n}(\alpha)$. By the way the $RC^{n}(\alpha)$ is defined, we have $\alpha \sqsubset_{A} \beta$.

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, for every $\alpha \in \overline{A}$, we have $\forall \beta \in \overline{A}. \alpha \sqsubset_{A} \beta$. \hfill \Box

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 a $g$-comtrace alphabet $\Theta = (E, \text{sim}, \text{ser}, \text{inl})$ 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 $A \neq B \implies A \equiv (A \setminus B)B$.

2. There exists a unique $C \subseteq A$ such that $a \in C$, $C$ is non-serializable to the right of $a$, and $A \neq C \implies A \equiv C(A \setminus C)$.

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:

$$\zeta \overset{df}{=} \{ D \in \hat{\Theta}(A) \mid \exists C \in \hat{\Theta}(A). (C \cup D = A \wedge a \in D \wedge C \times D \subseteq \text{ser}) \}$$

Let $B \in \zeta$ such that $B$ is a minimal element of the poset $(\zeta, \subseteq)$. We claim that $B$ is non-serializable to the left of $a$. Suppose that $B$ is serializable to the left of $a$, then there are some sets $E, F \in \hat{\Theta}(B)$ such that $E \cup F = B \wedge a \in F \wedge E \times F \subseteq \text{ser}$. Since $B \in \zeta$, there is some set $H \in \hat{\Theta}(A)$ such that $H \cup B = A \wedge a \in B \wedge H \times B \subseteq \text{ser}$. Because $H \times B \subseteq \text{ser}$ and $F \subseteq$
We will prove by induction on $B$, it follows that $H \times F \subseteq \text{ser}$. But since $E \times F \subseteq \text{ser}$, we have $(H \cup E) \times F \subseteq \text{ser}$. Hence, $\langle H \cup E \rangle \cup F = A \land a \in F \land (H \cup E) \times F \subseteq \text{ser}$. So $E \in \zeta$ and $E \subseteq B$. This contradicts that $B$ is minimal. Hence, $B$ is non-serializable to the left of $a$.

By the way the set $\zeta$ is defined, $A \equiv (A \setminus B)$. It remains to prove the uniqueness of $B$. Let $B' \in \zeta$ such that $B'$ is a minimal element of the poset $(\zeta, \subseteq)$. 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 $a$, it follows from Proposition A.3(1) that $\alpha \sqsubseteq^* \beta$. But since $a \neq b$, $\alpha (\sqsubseteq^* \setminus \text{id}_{\Sigma_A}) \beta$. From the definition of $\triangle$-closure, it follows that $\alpha \sqsubseteq [A] \beta$. Hence, by Proposition A.3(2), we have

$$\forall u \in [A]. \ pos_u(\alpha) \leq pos_u(\beta) \quad (A.1)$$

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 $(A \setminus B')B' \in [A] \text{ and } pos_{(A \setminus B')B'}(\beta) < pos_{(A \setminus B')B'}(\alpha)$, which contradicts (A.1).

Thus, $B \subseteq B'$.

By reversing the role of $B$ and $B'$, we can prove that $B \supseteq B'$. Hence, $B = B'$.

2. Dually to (1).

3. By (1) and (2), we only need to choose $D$ such that $D$ is non-serializable to the left and to the right of $a$. 

\[ \square \]

Proposition A.7. Let $s$ be a step sequence over a $g$-contrace alphabet $(E, \text{sim}, \text{ser}, \text{inl})$ and $G^{[s]} = (\Sigma_s, \prec, \sqsubseteq)$. Let $\sqsubseteq \subseteq \cup \prec$. If $\alpha, \beta \in \Sigma_s$, then

1. $(\forall u \in [s]. \ pos_u(\alpha) \neq pos_u(\beta)) \land (\exists u \in [s]. \ pos_u(\alpha) < pos_u(\beta)) \land (\exists u \in [s]. \ pos_u(\alpha) > pos_u(\beta)) \Rightarrow \alpha \nless \beta$

2. $(\forall u \in [s]. \ pos_u(\alpha) < pos_u(\beta)) \Rightarrow \alpha \prec \beta$

3. $(\alpha \neq \beta \land \forall u \in [s]. \ pos_u(\alpha) \leq pos_u(\beta)) \Rightarrow \alpha \sqsubseteq \beta$

\[ \text{PROOF.} \]

1. If $(\forall u \in [s]. \ pos_u(\alpha) \neq pos_u(\beta)) \land (\exists u \in [s]. \ pos_u(\alpha) < pos_u(\beta)) \land (\exists u \in [s]. \ pos_u(\alpha) > pos_u(\beta))$, then by Proposition A.2(1), there are $u_1, u_2 \in [s]$ and $x, y, A, B$ such that $u_1 = \overline{A} \overline{B} \overline{y} (\approx \cup \Rightarrow \approx) \overline{A} \overline{B} \overline{y} = u_2$ and $\alpha \in \overline{A}, \beta \in \overline{B}$. Hence, $([l(\alpha), l(\beta)]) \in \text{inl}$, which by (11.1) implies that $\alpha \nless \beta$. It then follows from Definitions 11.1 and 11.3 that $\alpha \nless \beta$.

2. Assume $(\forall u \in [s]. \ pos_u(\alpha) \leq pos_u(\beta))$ and $\alpha \neq \beta$. Hence, we can choose $u_0 \in [s]$ where $\overline{u_0} = \overline{x_1} \overline{E_1} \ldots \overline{E_k} \overline{y_0} (k \geq 1)$, $E_1, E_k$ are non-serializable, $\alpha \in \overline{E_1}, \beta \in \overline{E_k}$, and

$$\forall u_0 \in [s]. \Rightarrow (\overline{u_0} = \overline{x_0} \overline{E_1} \ldots \overline{E_k} \overline{y_0} \land \alpha \in \overline{E_1} \land \beta \in \overline{E_k}) \leq \text{weight}(\overline{E_1} \ldots \overline{E_k}) \quad (A.2)$$

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

$$\langle \forall u \in [s]. \ pos_u(\alpha) < pos_u(\beta) \rangle \Rightarrow \alpha \prec \beta \quad (A.3)$$

$$\langle \alpha \neq \beta \land \forall u \in [s]. \ pos_u(\alpha) \leq pos_u(\beta) \rangle \Rightarrow \alpha \sqsubseteq \beta \quad (A.4)$$
Base case. When weight\(\{E_1, \ldots, E_k\} = 2\), then we consider two cases:

- If \(\alpha \neq \beta\), \(\forall u \in [s]\. pos_u(\alpha) \leq pos_u(\beta)\) and \(\exists u \in [s]\. pos_u(\alpha) = pos_u(\beta)\), then it follows that
  \[
  \overline{\alpha} = \overline{\gamma} = \overline{\alpha, \beta} \gamma_0, \text{ or } \\
  \overline{\alpha} = \overline{\gamma} = \overline{\alpha} = \overline{\beta} \gamma_0
  \]
  But since \(\forall u \in [s]\. pos_u(\alpha) \leq pos_u(\beta)\), in either case, we must have \(\{l(\alpha), l(\beta)\}\) is not serializable to the right of \(l(\beta)\). Hence, by Proposition \(A.3(2)\), \(\alpha \sqsubseteq \beta\). This by Definitions \(11.1\) and \(11.3\) implies that \(\alpha \sqsubseteq \beta\).

- If \(\forall u \in [s]\. pos_u(\alpha) < pos_u(\beta)\), then it follows \(\overline{\alpha} = \overline{\beta} \gamma_0\). Since we assume that \(\forall u \in [s]\. pos_u(\alpha) < pos_u(\beta)\), we must have \(\{l(\alpha), l(\beta)\} \notin \text{ser or } inl\). This, by \(11.1\), implies that \(\alpha \prec \beta\). Hence, from Definitions \(11.1\) and \(11.3\) we get \(\alpha \prec \beta\).

Inductive step. When weight\(\{E_1, \ldots, E_k\} > 2\), then \(\overline{\alpha} = \overline{\beta} \gamma_0\) where \(k \geq 1\). We need to consider two cases:

Case (i): If \(\alpha \neq \beta\) and \(\forall u \in [s]\. pos_u(\alpha) \leq pos_u(\beta)\) and \(\exists u \in [s]\. pos_u(\alpha) = pos_u(\beta)\), then there is some \(v_0 = E \overline{\alpha} \overline{\beta} \gamma_0\) and \(\alpha, \beta \in E\). Either \(E\) is non-serializable to the right of \(l(\beta)\), or by Proposition \(A.6(2)\) \(\overline{\alpha} = \overline{\beta} \gamma_0\) where \(E\) is non-serializable to the right of \(l(\beta)\). In either case, by Proposition \(A.5(2)\), we have \(\alpha \sqsubseteq \beta\). So it follows from Definitions \(11.1\) and \(11.3\) that \(\alpha \sqsubseteq \beta\).

Case (ii): If \(\forall u \in [s]\. pos_u(\alpha) < pos_u(\beta)\), then it follows \(\overline{\alpha} = \overline{\beta} \gamma_0\) where \(k \geq 2\) and \(\alpha \in E_1\). \(\beta \in E_2\). If \(\{l(\alpha), l(\beta)\} \notin \text{ser or } inl\), then by \(11.1\), \(\alpha \prec \beta\). Hence, from Definitions \(11.1\) and \(11.3\) we get \(\alpha \prec \beta\). So we need to consider only when \(\{l(\alpha), l(\beta)\} \in \text{ser or } inl\). There are three cases to consider:

(a) If \(\overline{\alpha} = \overline{E_1 \beta} \gamma_0\) 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 \(E_1 \times E_2 \notin \text{ser and } E_1 \times E_2 \notin \text{inl}\). Hence, there are \(\alpha_1, \alpha_2 \in E_1\) and \(\beta_1, \beta_2 \in E_2\) such that \(\{l(\alpha_1), l(\beta_1)\} \notin \text{ser and } \{l(\alpha_2), l(\beta_2)\} \notin \text{ser}\). Since \(E_1\) and \(E_2\) are non-serializable, by Proposition \(A.5(3)\), \(\alpha_1 \sqsubseteq \alpha_2\) and \(\beta_1 \sqsubseteq \beta_2\). Also by \(11.2\) we know that \(\alpha_1 \prec \alpha_2\) and \(\beta_1 \prec \beta_2\). Thus, by \(11.3\) we have \(\alpha_1 \prec \beta_1\). Since \(E_1\) and \(E_2\) are non-serializable, by Proposition \(A.5(3)\), \(\alpha_1 \prec \alpha_2 \prec \beta_1 \prec \beta_2\). Hence, by Definitions \(11.1\) and \(11.3\), \(\alpha \prec \beta\).

(b) If \(\overline{\alpha} = \overline{E_1 \ldots E_k} \gamma_0\) where \(k \geq 3\) and \(\{l(\alpha), l(\beta)\} \in \text{inl}\), then let \(\gamma \in E_2\). Observe that we must have

\[
\overline{\alpha} = \overline{E_1 \ldots E_k \gamma_0} = \overline{E_1 \gamma_1} \overline{E_2 \gamma_2} \gamma_3 = \overline{E_1 \gamma_1} \overline{E_2 \gamma_2},
\]

such that \(\gamma \in E\). \(F\) is a non-serializable, and weight\(\{E_1, \ldots, E_k\}\), weight\(\{E_2\}\) satisfy the minimal condition similarly to \(A.2\). Since from the way \(\alpha_0\) is chosen, we know that \(\forall u \in [s]\. pos_u(\alpha) \leq pos_u(\gamma)\) and \(\forall u \in [s]\. pos_u(\gamma) \leq pos_u(\beta)\), by applying the induction hypothesis, we get

\[
\alpha \sqsubseteq \gamma \sqsubseteq \beta
\]

So by transitivity of \(\sqsubseteq\), we get \(\alpha \sqsubseteq \beta\). But since we assume \(\{l(\alpha), l(\beta)\} \in \text{inl}\), it follows that \(\alpha \prec \beta\). Hence, \((\alpha, \beta) \in \text{\(\subset\)} \prec = \prec\).
(c) If \( \overline{m} = \overline{x_0 E_1 \ldots E_k} \) where \( k \geq 3 \) and \( (l(\alpha), l(\beta)) \in \text{ser} \), then we observe from how \( u_0 \) is chosen that

\[
\forall \gamma \in \bigcup \{E_1 \ldots E_k\}, (\forall u \in [s]. \text{pos}_u(\alpha) \leq \text{pos}_u(\gamma) \leq \text{pos}_u(\beta))
\]

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

\[
\forall \gamma \in \bigcup \{E_1 \ldots E_k\} \setminus \{\alpha, \beta\}. \alpha \subset \gamma \subset \beta
\]  
\[(A.6)\]

We next want to show that

\[
\exists \delta, \gamma \in \bigcup \{E_1 \ldots E_k\}. (\text{pos}_u(\delta) < \text{pos}_u(\gamma) \land (l(\delta), l(\gamma)) \notin \text{ser})
\]  
\[(A.7)\]

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

\[
\forall \delta, \gamma \in \bigcup \{E_1 \ldots E_k\}. (\text{pos}_u(\delta) < \text{pos}_u(\gamma) \implies (l(\delta), l(\gamma)) \in \text{ser})
\]

It follows that \( \overline{m} = \overline{x_0 E_1 \ldots E_k} \) \( \equiv \overline{x_0 E_1 \ldots E_k} \), which contradicts that \( \forall u \in [s]. \text{pos}_u(\alpha) < \text{pos}_u(\beta) \). Hence, we have shown \( (A.7) \).

Let \( \delta, \gamma \in \bigcup \{E_1 \ldots E_k\} \) be event occurrences such that \( \text{pos}_u(\delta) < \text{pos}_u(\gamma) \) and \( (l(\delta), l(\gamma)) \notin \text{ser} \). By \( (A.6) \), \( \alpha(\bigcup \text{id}_{E_i}) \delta(\bigcup \text{id}_{E_i}) \beta \) and \( \alpha(\bigcup \text{id}_{E_i}) \gamma(\bigcup \text{id}_{E_i}) \beta \). If \( \alpha < \gamma \) or \( \beta < \gamma \), then by (C4) of Definition \( 11.1 \) \( \alpha < \beta \). Otherwise, by Definitions \( 11.1 \) and \( 11.3 \) we have \( \alpha \sqsubseteq \delta \subseteq \alpha \) and \( \alpha \sqsubseteq \gamma \subseteq \beta \). But since \( \text{pos}_u(\delta) < \text{pos}_u(\gamma) \) and \( (l(\delta), l(\gamma)) \notin \text{ser} \), by Definition \( 11.3 \) \( \alpha < \beta \). So by Definitions \( 11.3 \) and \( 11.3 \) \( \alpha < \beta \).

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

\[\square\]

Proposition \( 11.4 \) Let \( s \) be a step sequence over a \( g \)-comtrace alphabet \( (E, \text{sim}, \text{ser}, \text{inl}) \). Let \( G^{(s)} = (\Sigma_s, \ll, \sqsubseteq, \sqcap) \), and let \( \ll = \ll \cap \sqcap \). Then for every \( \alpha, \beta \in \Sigma_s \), we have

1. \( \alpha \ll \beta \iff \forall u \in [s]. \text{pos}_u(\alpha) \neq \text{pos}_u(\beta) \)
2. \( \alpha \sqsubseteq \beta \iff \forall u \in [s]. \text{pos}_u(\alpha) \leq \text{pos}_u(\beta) \wedge \alpha \neq \beta \)
3. \( \alpha < \beta \iff \forall u \in [s]. \text{pos}_u(\alpha) < \text{pos}_u(\beta) \)
4. If \( l(\alpha) = l(\beta) \) and \( \text{pos}_s(\alpha) < \text{pos}_s(\beta) \), then \( \alpha < \beta \).

Proof. 1. Follows from Proposition \( 11.1 \) and Proposition \( 11.2 \).
2. Follows from Proposition \( 11.1 \) and Proposition \( 11.2 \).
3. Follows from (1) and (2).
4. Assume that \( l(\alpha) = l(\beta) \) and \( \text{pos}_s(\alpha) < \text{pos}_s(\beta) \). Then by Proposition \( 11.1 \), we know \( \forall u \in [s]. \text{pos}_u(\alpha) < \text{pos}_u(\beta) \). Hence, by (3), \( \alpha < \beta \).

\[\square\]

Appendix B: Proof of Lemma \( 11.2 \)

Lemma \( 11.2 \) Let \( s, t \in S^* \) and \( <_s \in \text{ext}(G^{(s)}) \). Then \( G^{(s)} = G^{(t)} \).

Proof. \( (<_t = <_s) \) We have \( \alpha <_t \beta \) iff by Definition \( 11.1 \) \( (l(\alpha), l(\beta)) \in \text{inl} \), which by Definition \( 11.3 \) means \( \alpha <_s \beta \). Hence,

\[
<_t = <_s
\]  
\[(B.8)\]
we have

and $1.3$,

We want to show that

(b) When

(c) There remains only the case when

\(\alpha \prec \beta\) and \(\beta \prec \alpha\). So \(\text{pos}_i(\beta) < \text{pos}_i(\alpha)\),

a contradiction. Thus, \(\sqsubseteq \subseteq \sqsubseteq\).

Together with (B.9), we get

\[\sqsubseteq = \sqsubseteq\]  \hfill (B.10)

(\(\prec_i = \prec_s\) if \(\alpha \prec_i \beta\), then by Definitions 11.1 and 11.3 \(\alpha \prec \beta\). But since \(\prec_i \in \text{ext}(G)\),

we have \(\text{pos}_i(\beta) \leq \text{pos}_s(\beta)\). But since \(\alpha \subseteq \beta\), by Definition 11.3

\((l(\beta), l(\alpha)) \notin \text{ser} \cup \text{inl}\). Hence, by Definition 11.3 \(\alpha \subseteq \beta\).

Thus, \(\therefore\).

\[\therefore \subseteq \subseteq\]  \hfill (B.12)
It remains to show that $\sim_s \subseteq \sim_t$. Let $\alpha \sim_s \beta$. Suppose that $\neg(\alpha \sim_t \beta)$. Since $\alpha \sim_s \beta$, by Definition\textsuperscript{11.3} $pos_s(\alpha) < pos_s(\beta)$ and

\[
\left\{\begin{array}{l}
(l(\alpha), l(\beta)) \notin ser \cup \text{inl} \\
(\alpha, \beta) \in \sqrt{s} \cap ((\Xi^s)^{\circ} \circ (\Xi^s)^{\circ}) \\
(l(\alpha), l(\beta)) \in \text{ser} \\
\wedge \exists \delta, \gamma \in \Sigma_s. \left. \begin{array}{l}
 pos_s(\delta) < pos_s(\gamma) \land (l(\delta), l(\gamma)) \notin \text{ser} \\
\alpha \sqsubseteq^s_\delta \beta \land \alpha \sqsubseteq^s_\gamma \beta
\end{array}\right)\right.\right.
\]

We want to show that $\alpha \sim_t \beta$. We consider three cases:

(a) When $(l(\alpha), l(\beta)) \notin \text{ser} \cup \text{inl}$, we suppose that $\neg(\alpha \sim_t \beta)$. This by Definition\textsuperscript{11.3} implies that $pos_s(\beta) \leq pos_s(\alpha)$. By Definitions\textsuperscript{11.1} and \textsuperscript{11.3} it follows that $\beta \sqsubseteq_t \alpha$ and $\beta \sqsubseteq_t \alpha$. But since $\preceq_s \in \text{ext}(G^{(t)})$, we have $\beta \preceq^t_s \alpha$, which implies $pos_s(\beta) \leq pos_s(\alpha)$, a contradiction.

(b) If $(\alpha, \beta) \in \sqrt{t} \cap ((\Xi^t)^{\circ} \circ (\Xi^t)^{\circ})$, then since $\sqrt{s} = \sqrt{t}$ and $\cap_s = \cap_t$, we have $(\alpha, \beta) \in \sqrt{t} \cap ((\Xi^t)^{\circ} \circ (\Xi^t)^{\circ})$. Since $\alpha \sim_t \beta$, we have $pos_s(\alpha) < pos_s(\beta)$ or $pos_s(\alpha) = pos_s(\beta)$. We claim that $pos_s(\alpha) < pos_s(\beta)$. Suppose for a contradiction that $pos_s(\alpha) < pos_s(\beta)$. Since $(\alpha, \beta) \in \sqrt{t} \cap ((\Xi^t)^{\circ} \circ (\Xi^t)^{\circ})$, and $\sim_t$ is symmetric, we have $(\beta, \alpha) \in \sqrt{t} \cap ((\Xi^t)^{\circ} \circ (\Xi^t)^{\circ})$. Hence, it follows from Definitions\textsuperscript{11.1} and \textsuperscript{11.3} that $\beta \sim_t \alpha$ and $\beta \sim_t \alpha$. But since $\neg \sim_s \in \text{ext}(G^{(t)})$, we have $\beta \sim_t \alpha$, which implies $pos_s(\beta) < pos_s(\alpha)$, a contradiction. We have just shown that $pos_s(\alpha) < pos_s(\beta)$. Since $(\alpha, \beta) \in \sqrt{t} \cap ((\Xi^t)^{\circ} \circ (\Xi^t)^{\circ})$, we get $\alpha \sim_t \beta$.

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

\[
( pos_s(\delta) < pos_s(\gamma) \land (l(\delta), l(\gamma)) \notin \text{ser} \land pos_s(\delta) \leq pos_s(\beta) \land pos_s(\alpha) \leq pos_s(\gamma) \leq pos_s(\beta). \) Since $(l(\delta), l(\gamma)) \notin \text{ser}$, we either have $(l(\delta), l(\gamma)) \in \text{inl}$ or $(l(\delta), l(\gamma)) \notin \text{ser} \cup \text{inl}$.

(i) If $(l(\delta), l(\gamma)) \in \text{inl}$, then $pos_s(\delta) \neq pos_s(\gamma)$. This implies that $(pos_s(\delta) < pos_s(\gamma) \land (l(\delta), l(\gamma)) \notin \text{ser})$ or $(pos_s(\gamma) < pos_s(\delta) \land (l(\delta), l(\gamma)) \notin \text{ser})$. Since $\text{pos}(\delta) \neq \text{pos}(\gamma)$ and $\text{pos}(\alpha) \leq \text{pos}(\delta) \leq \text{pos}(\beta)$ and $\text{pos}(\alpha) \leq \text{pos}(\gamma) \leq \text{pos}(\beta)$, we also have $\text{pos}(\alpha) < \text{pos}(\beta)$. So it follows from Definition\textsuperscript{11.3} that $\alpha \sim_t \beta$.

(ii) If $(l(\delta), l(\gamma)) \notin \text{ser} \cup \text{inl}$, then $(l(\delta), l(\gamma)) \notin \text{ser} \cup \text{inl}$. We want to show that $pos_s(\delta) < pos_s(\gamma)$. Suppose that $pos_s(\delta) \geq pos_s(\gamma)$. Since $(l(\delta), l(\gamma)) \notin \text{ser} \cup \text{inl}$, by Definitions\textsuperscript{11.1} and \textsuperscript{11.3} we have $\gamma \sqsubseteq \delta$ and $\gamma \sqsubseteq \delta$. But since $\neg \sim_s \in \text{ext}(G^{(t)})$, we have $\gamma \sqsubseteq^t_s \delta$, which implies $\text{pos}_s(\gamma) \leq \text{pos}_s(\delta)$, a contradiction. Since $\text{pos}_s(\delta) < \text{pos}_s(\gamma)$ and $\text{pos}_s(\alpha) \leq \text{pos}_s(\delta) \leq \text{pos}_s(\beta)$ and $\text{pos}_s(\alpha) \leq \text{pos}_s(\gamma) \leq \text{pos}_s(\beta)$, we have $\text{pos}_s(\alpha) < \text{pos}_s(\beta)$. Hence, we have $\text{pos}_s(\alpha) < \text{pos}_s(\beta)$ and

\[
(\wedge \alpha \sqsubseteq^s_\delta \beta \land \alpha \sqsubseteq^s_\gamma \beta ) \Rightarrow \alpha \sim_t \beta \) by Definition\textsuperscript{11.3}.

Thus, we have shown $\sim_t \subseteq \sim_t$. This and (B.12) imply

\[
\sim_t = \sim_s \] (B.13)

By (B.8), (B.10) and (B.13), we have $(\Sigma, \preceq_t \cup \preceq_t \cup \preceq_t) = (\Sigma, \preceq_s \cup \preceq_s \cup \preceq_s) \approx (\Sigma, \preceq_s \cup \preceq_s \cup \preceq_s) \approx (G^{(t)})$. Thus, it follows that $G^{(t)} = (\Sigma, \preceq_t \cup \preceq_t \cup \preceq_t) \approx (\Sigma, \preceq_s \cup \preceq_s \cup \preceq_s) \approx (G^{(s)})$. \hfill \qed