COMPUTING SCIENCE

Folded Hasse Diagrams of Combined Traces

Lukasz Mikulski and Maciej Koutny

TECHNICAL REPORT SERIES

No. CS-TR-1357 November 2012

TECHNICAL REPORT SERIES

No. CS-TR-1357 November, 2012

Folded Hasse Diagrams of Combined Traces

L. Mikulski and M. Koutny

Abstract

To represent concurrent behaviours one can use concepts originating from language theory, including traces and comtraces. Traces can express notions such as concurrency and causality, whereas comtraces can also capture weak causality and simultaneity. This paper is concerned with the development of efficient data structures and algorithms for manipulating comtraces. We introduce and investigate folded Hasse diagrams of comtraces which generalise Hasse diagrams defined for partial orders and traces. We also develop an efficient on-line algorithm for deriving Hasse diagrams from language theoretic representations of comtraces. Finally, we briefly discuss how folded Hasse diagrams could be used to implement efficiently some basic operations on comtraces.

Printed and published by Newcastle University,
Computing Science, Claremont Tower, Claremont Road,
Newcastle upon Tyne, NE1 7RU, England.

Bibliographical details

MIKULSKI, L., KOUTNY, M.

Folded Hasse Diagrams of Combined Traces
[By] L. Mikulski and M. Koutny

Newcastle upon Tyne: Newcastle University: Computing Science, 2012.

(Newcastle University, Computing Science, Technical Report Series, No. CS-TR-1357)

Added entries

NEWCASTLE UNIVERSITY
Computing Science. Technical Report Series. CS-TR-1357

Abstract

To represent concurrent behaviours one can use concepts originating from language theory, including traces and comtraces. Traces can express notions such as concurrency and causality, whereas comtraces can also capture weak causality and simultaneity. This paper is concerned with the development of efficient data structures and algorithms for manipulating comtraces. We introduce and investigate folded Hasse diagrams of comtraces which generalise Hasse diagrams defined for partial orders and traces. We also develop an efficient on-line algorithm for deriving Hasse diagrams from language theoretic representations of comtraces. Finally, we briefly discuss how folded Hasse diagrams could be used to implement efficiently some basic operations on comtraces.

About the authors

Lukasz Mikulski obtained his PhD in 2012 from the University of Warsaw, Poland. He now works as a Lecturer at the Faculty of Mathematics and Computer Science of the Nicolaus Copernicus University, Torun, Poland.

Maciej Koutny obtained his MSc (1982) and PhD (1984) from the Warsaw University of Technology. In 1985 he joined the then Computing Laboratory of the University of Newcastle upon Tyne to work as a Research Associate. In 1986 he became a Lecturer in Computing Science at Newcastle, and in 1994 was promoted to an established Readership at Newcastle. In 2000 he became a Professor of Computing Science.

Suggested keywords

COMTRACE, TRACE
HASSE DIAGRAM
STRATIFIED ORDER STRUCTURE
PARTIAL ORDER
CONCURRENCY
CAUSALITY
WEAK CAUSALITY
INDEPENDENCE
ALGORITHMIC COMPLEXITY
PETRI NET
INHIBITOR ARC

Folded Hasse Diagrams of Combined Traces

Łukasz Mikulski[1,2] and Maciej Koutny[2]

[1] Faculty of Mathematics and Computer Science
Nicolaus Copernicus University
Toruń, Chopina 12/18, Poland
lukasz.mikulski@mat.umk.pl
[2] School of Computing Science
Newcastle University
Newcastle upon Tyne, NE1 7RU, United Kingdom
{maciej.koutny,lukasz.mikulski}@ncl.ac.uk

Abstract. To represent concurrent behaviours one can use concepts originating from language theory, including traces and comtraces. Traces can express notions such as concurrency and causality, whereas comtraces can also capture weak causality and simultaneity. This paper is concerned with the development of efficient data structures and algorithms for manipulating comtraces. We introduce and investigate folded Hasse diagrams of comtraces which generalise Hasse diagrams defined for partial orders and traces. We also develop an efficient on-line algorithm for deriving Hasse diagrams from language theoretic representations of comtraces. Finally, we briefly discuss how folded Hasse diagrams could be used to implement efficiently some basic operations on comtraces.
Keywords: comtrace, trace, Hasse diagram, stratified order structure, partial order, concurrency, causality, weak causality, independence, algorithmic complexity, Petri net, inhibitor arc

1 Introduction

The dynamic behaviours of concurrent systems represented, e.g., by Petri nets, are usually modelled as ongoing evolutions involving actions that take place at the interface with the environment. The simplest representations of such evolutions are sequences (or words) of executed actions, leading to a formal language semantics of Petri nets. However, words alone cannot express concurrency and causality between executed actions which are features of paramount importance if one wants to understand or efficiently analyse concurrent behaviours. To address this issue, one may consider keeping an additional information about the relevant properties of behaviours, for example, in the form of causal dependencies between actions.

This approach underpins the trace model of concurrent behaviour [1, 8]. Traces are not sufficient, however, when one needs to deal, e.g., with Petri nets with inhibitor arcs. To deal with such systems, one may extend traces with additional information about intrinsic relationships between executed actions in the form of *weak causality* (where a weakly precedes b if it can be executed earlier or simultaneously with b). The resulting model of combined traces [4, 6, 5] (or *comtraces*) enjoys properties similar to those which hold for traces.

In this paper, we are concerned with the development of efficient data structures and algorithms for manipulating comtraces. We first introduce and investigate *folded Hasse diagrams* of comtraces which generalise Hasse diagrams [12] defined for partial orders and traces. We then develop an efficient algorithm for deriving them from individual step sequence representatives of comtraces. We also explain how the proposed representation of comtraces can be used to implement efficiently some basic operations on comtraces. Note that our aims are different from those pursued in papers like [3] where the main concern is to develop visually pleasing ways of drawing Hasse diagrams. Here, we are focused on their semantical extensions and effective manipulation.

The paper is organised as follows. Section 2 provides basic notation and terminology. Section 3 discusses comtraces and introduces their folded Hasse diagrams. Section 4 presents an efficient on-line construction of folded Hasse diagrams from individual step sequence representatives of comtraces. Section 5 outlines possible applications of the results contained in this paper, and Section 6 briefly discusses directions for further research.

2 Preliminaries

Throughout the paper we use the standard notions of the set theory and formal language theory. In particular, $\uplus$ denotes disjoint set union, and by an *alphabet* we mean a nonempty finite set Σ, the elements of which are called *(atomic) actions*. Finite sequences over Σ are called *words*. The set of all words is denoted by Σ^*.

A *directed acyclic graph* is a pair $dag = (X, R)$, where X is a finite set and R is an acyclic irreflexive binary relation on X. In a diagrammatical representation, X is the set of vertices while R the set of arcs. A *linearisation* of dag is any sequence $u = x_1 \dots x_n$ of distinct elements of dag such that $X = \{x_1, \dots, x_n\}$ and, for all $1 \leq i < j \leq n$ the predicate $x_j(R^+)x_i$ is false.

A directed acyclic graph $po = (X, \prec)$ is a *poset* if the relation $\prec$ is transitive. Moreover, a directed acyclic graph $(X, \prec^{\mathsf{pre}})$ is a *po-diagram* if $\prec = (\prec^{\mathsf{pre}})^*$. Among all the *po*-diagrams, we can distinguish the smallest one (i.e., that with the smallest relation $\prec^{\mathsf{pre}}$), denoted by $H(po) = (X, \prec^{\mathsf{red}})$ and called the *Hasse diagram* of po. Note that $\prec^{\mathsf{red}}$ can be obtained from $\prec$ by simply removing all the arcs implied by the transitivity of $\prec$; in other words, $\prec^{\mathsf{red}}$ is equal to $\prec \setminus (\prec \circ \prec)$. Moreover, if (X, R) is a *po*-diagram, then $\prec^{\mathsf{red}} = R \setminus \bigcup_{i \geq 2} R^i$.

3 Comtraces and Their Hasse Diagrams

A *comtrace alphabet* is a triple $\Theta = (\Sigma, sim, ser)$, where Σ is an alphabet and $ser \subseteq sim \subset \Sigma \times \Sigma$ are two relations, respectively called *serialisability* and *simultaneity*; it is assumed that sim is irreflexive and symmetric. Intuitively, if $(a, b) \in sim$ then a and b may occur simultaneously, whereas $(a, b) \in ser$

means that in such a case a may also occur before b (with both executions being equivalent). The set of (potential) steps over Θ, called *step alphabet*, is then defined as the set $\mathbb{S}$ comprising all nonempty sets of actions $A \subseteq \Sigma$ such that $(a, b) \in \mathit{sim}$, for all distinct $a, b \in A$. To avoid confusion with the well-established operation of set concatenation, we follow [2] and denote a step containing actions a and b by (ab) rather then $\{a, b\}$, etc. Finite sequences in $\mathbb{S}^*$, including the empty one denoted by λ, are called *step sequences*.

We now present a number of notions and notations for step sequences. In what follows, $\Theta = (\Sigma, \mathit{sim}, \mathit{ser})$ is a *fixed* comtrace alphabet.

Let $w = A_1 \dots A_n$ and $v = B_1 \dots B_m$ be two step sequences. Then $w \circ v = wv = A_1 \dots A_n B_1 \dots B_m$ is the concatenation of w and v. The alphabet $\mathit{alph}(w)$ of w comprises all actions occurring within w, and $\#_a(w)$ is the number of occurrences of an action a within w. The set $\mathit{occ}(w)$ of *action occurrences* of w comprises all pairs (a, i) with $a \in \mathit{alph}(w)$ and $1 \leq i \leq \#_a(w)$.

The position $\mathit{pos}_w(\alpha)$ of an action occurrence $\alpha = (a, i) \in \mathit{occ}(w)$ is given as the smallest integer j such that $\#_a(A_1 \dots A_j) = i$. In such a case, we also denote $\alpha \in \mathit{occ}_j(w)$. Hence $\mathit{occ}(w) = \mathit{occ}_1(w) \uplus \cdots \uplus \mathit{occ}_n(w)$.

The default label of an action occurrence $\alpha = (a, i)$ is $\ell(\alpha) = a$. We can also apply ℓ to sets of action occurrences and sequences of sets of action occurrences in the usual way, e.g., $\ell(\{\alpha_1, \dots, \alpha_n\}) = \{\ell(\alpha_1), \dots, \ell(\alpha_n)\}$.

The *head* (first action occurrences) and *tail* (last action occurrences) of a step sequence w are two sets defined by:

$$
\begin{aligned}
\mathit{head}(w) &= \{(a, 1) \mid a \in \mathit{alph}(w)\} \\
\mathit{tail}(w) &= \{(a, \#_a(w)) \mid a \in \mathit{alph}(w)\} \ .
\end{aligned}
$$

3.1 Comtraces

Let $\Theta = (\Sigma, \mathit{sim}, \mathit{ser})$ be a *fixed* comtrace alphabet. The *comtrace congruence* over Θ, denoted by $\equiv_\Theta$, is the reflexive, symmetric and transitive closure of the relation $\sim_\Theta \subseteq \mathbb{S}^* \times \mathbb{S}^*$ defined in such a way that $w \sim_\Theta v$ if there are $u, z \in \mathbb{S}^*$ and $A, B, C \in \mathbb{S}$ satisfying

$$
w = uAz \quad v = uBCz \quad A = B \cup C \quad B \times C \subseteq \mathit{ser} \ .
$$

Note that $B \cap C = \varnothing$ as ser is irreflexive.

Equivalence classes of the relation $\equiv_\Theta$ are called *comtraces* (see [5]), and the comtrace containing a given step sequence w is denoted by $[w]$. The set of all comtraces is denoted by $\mathbb{S}^*/_{\equiv_\Theta}$, and the pair $(\mathbb{S}^*/_{\equiv_\Theta}, \circ)$ is a (comtrace) monoid, where $\tau \circ \phi = [w \circ w']$, for any step sequences $w \in \tau$ and $w' \in \phi$. Comtrace concatenation is well-defined as $[w \circ w'] = [v \circ v']$, for all $w, v \in \tau$ and $w', v' \in \phi$. Similarly, for every comtrace τ and every $a \in \Sigma$, we can define $\mathit{alph}(\tau) = \mathit{alph}(w)$, $\#_a(\tau) = \#_a(w)$ and $\mathit{occ}(\tau) = \mathit{occ}(w)$, where $w \in \tau$ is arbitrarily chosen. A comtrace τ is a prefix of a comtrace ϕ if there is a comtrace ϕ' such that $\tau \circ \phi' = \phi$.

The normal form of a comtrace captures a greedy, maximally concurrent, execution of the actions occurring in the comtrace conforming to the simultaneity and serialisability relations. A step sequence $w = A_1 \dots A_n \in \mathbb{S}^*$ is in (Foata) *normal form* if, for each $i \leq n$, whenever $Av \equiv_\Theta A_i \dots A_n$ for some $A \in \mathbb{S}$ and $v \in \mathbb{S}^*$, then $|A| \leq |A_i|$. One can see that each comtrace comprises a unique step sequence in normal form. Note that an alternative (equivalent) definition of normal form requires that, for every $i < n$, there is no $\varnothing \neq A \subseteq A_{i+1}$ such that $A_i \times A \subseteq ser$ and $A \times (A_{i+1} \backslash A) \subseteq ser$.

3.2 Stratified order structures

Comtraces can be represented by so-structures, in a similar way as traces can be represented by posets.

A *stratified order structure* (or so-structure) is a tuple $sos = (X, \prec, \sqsubset)$ comprising two binary relations, $\prec$ (*causality*) and $\sqsubset$ (*weak causality*), on a finite set X such that, for all $x, y, z \in X$:

$$\begin{array}{ll} S1: & x \not\sqsubset x \\ S2: & x \prec y \implies x \sqsubset y \\ S3: & x \sqsubset y \sqsubset z \ \wedge\ x \neq z \implies x \sqsubset z \\ S4: & x \sqsubset y \prec z \ \vee\ x \prec y \sqsubset z \implies x \prec z\,. \end{array}$$

Intuitively, $\prec$ represents the 'earlier than' relationship, and $\sqsubset$ the 'not later than' relationship, when the elements of X are being interpreted as events which have occurred in an execution of some concurrent system. Note that $\prec$ is a partial order, and $x \prec y$ implies $y \not\sqsubset x$. In diagrams, $\prec$ is depicted by solid arcs, and $\sqsubset$ by dashed arcs.

Let $\varrho = (X, \prec^{\mathsf{pre}}, \sqsubset^{\mathsf{pre}})$ be a triple such that $\prec^{\mathsf{pre}}$ and $\sqsubset^{\mathsf{pre}}$ are irreflexive binary relations on a finite set X. Then the *so-closure* of ϱ is defined as

$$\varrho^{\diamondsuit} = (X, \gamma \circ \prec^{\mathsf{pre}} \circ \gamma, \gamma \setminus id_X)\ ,$$

where $\gamma = (\prec^{\mathsf{pre}} \cup \sqsubset^{\mathsf{pre}})^*$. If $\varrho^{\diamondsuit} = sos$, where sos is an so-structure, then ϱ is called an *sos-diagram*.

A *stratification* of an so-structure $sos = (X, \prec, \sqsubset)$ is any sequence $u = X_1 \dots X_n$ of nonempty subsets of X such that $X = X_1 \uplus \dots \uplus X_n$,

- $(X_j \times X_i) \cap \prec = \varnothing$, for all $1 \leq i \leq j \leq n$; and
- $(X_j \times X_i) \cap \sqsubset = \varnothing$, for all $1 \leq i < j \leq n$.

We denote this by $u \in str(sos)$.

The so-structure *induced* by a comtrace τ is defined as

$$sos(\tau) = presos(w)^{\diamondsuit}\ ,$$

where w is any step sequence $w \in \tau$, and

$$presos(w) = (occ(w), \prec_w, \sqsubset_w)$$

is a triple such that $\prec_w$ and $\sqsubset_w$ are irreflexive binary relations on $occ(w)$ satisfying, for all distinct $\alpha, \beta \in occ(w)$,

– $\alpha \prec_w \beta$ if $pos_w(\alpha) < pos_w(\beta)$ and $(\ell(\alpha), \ell(\beta)) \notin ser$; and
– $\alpha \sqsubset_w \beta$ if $pos_w(\alpha) \leq pos_w(\beta)$ and $(\ell(\beta), \ell(\alpha)) \notin ser$.

The soundness of the last definition stems from the fact that $presos(w) = presos(v)$, for all step sequences $w, v \in \tau$. Crucially, one can see ([6]) that the induced so-structure provides an alternative representation of τ as we have:

$$\tau = \{\ell(u) \mid u \in str(sos(\tau))\}\ . \tag{1}$$

Note that if wA is a step sequence then appending a step A results in proper extension of $presos(w)$:

$$\begin{aligned} \prec_w &= \prec_{wA} |_{occ(w)\times occ(w)} \\ \sqsubset_w &= \sqsubset_{wA} |_{occ(w)\times occ(w)}\ . \end{aligned} \tag{2}$$

Moreover, only forward relationships are added:

$$\begin{aligned} \prec_{wA} \cap\, occ_{|w|+1}(wA) \times occ(wA) &= \varnothing \\ \sqsubset_{wA} \cap\, occ_{|w|+1}(wA) \times occ(w) \quad &= \varnothing\ . \end{aligned} \tag{3}$$

As a result, $presos(w)$ is a *vertex-induced* subgraph of $presos(wA)$, i.e., the vertices of $presos(wA)$ contain those of $presos(w)$, and if we take all the arcs of $presos(wA)$ joining the vertices of $presos(w)$, then we obtain exactly the arcs of $presos(w)$.

3.3 Folded so-structures

Weak causality is a pre-order rather than a partial order relation, and so it can be advantageous to work with a quotient so-structure derived from $sos(\tau) = (occ(\tau), \prec, \sqsubset)$ induced by a comtrace τ. First, for each action occurrence $\alpha \in occ(\tau)$, we denote by $\langle\alpha\rangle$ the equivalence class of the $\sqsubset$-cycle relation comprising α, i.e., α together with the set of all $\beta \in occ(\tau)$ satisfying $\alpha \sqsubset \beta \sqsubset \alpha$. Each such $\Psi = \langle\alpha\rangle$ will be called a *folded action*, and their set will be denoted by $\widehat{occ}(\tau)$. Note that folded actions are also called indivisible steps in [9], and the idea of indivisibility of steps was utilised, in the case of traces, in [11].

The *folded so-structure induced* by a comtrace τ is $\widehat{sos}(\tau) = (\widehat{occ}(\tau), \widehat{\prec}, \widehat{\sqsubset})$, where, for all $\Psi, \Phi \in \widehat{occ}(\tau)$:

– $\Psi \widehat{\prec} \Phi$ if $(\Psi \times \Phi) \cap \prec \neq \varnothing$; and
– $\Psi \widehat{\sqsubset} \Phi$ if $(\Psi \times \Phi) \cap \sqsubset \neq \varnothing$ and $\Psi \neq \Phi$.

By *S4*, $\Psi \widehat{\prec} \Phi$ means that $\Psi \times \Phi$ is included in $\prec$. By *S3*, $\Psi \widehat{\sqsubset} \Phi$ means that $\Psi \times \Phi$ is included in $\sqsubset$. And, by *S2*–*S4*, $\widehat{sos}(\tau)$ is an so-structure, and $\widehat{\sqsubset}$ is a poset containing $\widehat{\prec}$.

It turns out that different comtraces induce different folded so-structures. Moreover, there is a straightforward way of recovering $sos(\tau)$ from $\widehat{sos}(\tau)$, as we have, for all $\alpha, \beta \in occ(\tau)$:

– $\alpha \prec \beta$ iff $\langle\alpha\rangle \widehat{\prec} \langle\beta\rangle$; and

– $\alpha \sqsubset \beta$ iff $\langle\alpha\rangle \widehat{\sqsubset} \langle\beta\rangle$, or $\alpha \neq \beta \wedge \langle\alpha\rangle = \langle\beta\rangle$.

It can also be shown that, for every step sequence $w \in \tau$,

$$\widehat{sos}(\tau) = \widehat{presos}(w)^{\diamondsuit} \ , \tag{4}$$

where $\widehat{presos}(w) = (\widehat{occ}(w), \widehat{\prec}_w, \widehat{\sqsubset}_w)$ is a triple such that $\widehat{\prec}_w$ and $\widehat{\sqsubset}_w$ are irreflexive binary relations on $\widehat{occ}(w)$ satisfying, for all distinct $\Phi, \Psi \in \widehat{occ}(w)$,

- $\Phi \widehat{\prec}_w \Psi$ if $(\Phi \times \Psi) \cap \prec_w \neq \varnothing$; and
- $\Phi \widehat{\sqsubset}_w \Psi$ if $(\Phi \times \Psi) \cap \sqsubset_w \neq \varnothing$.

Note that $\widehat{presos}(w) = \widehat{presos}(v)$, for all $w, v \in \tau$, and that $\widehat{presos}(w)$ is an $\widehat{sos}(\tau)$-diagram.

Action occurrences of a step can always be partitioned into folded actions.

Proposition 1. *Let $v = wAu$ be a step sequence. Then the set of action occurrences $occ_{|w|+1}(v)$ can be partitioned into a set Γ of folded actions, each of which is a strongly connected component of the directed graph*

$$(occ_{|w|+1}(v), \sqsubset_v |_{occ_{|w|+1}(v) \times occ_{|w|+1}(v)}) \tag{5}$$

and

$$(\Gamma, \widehat{\sqsubset}_v|_{\Gamma \times \Gamma}) \tag{6}$$

is a directed acyclic graph. Moreover, for any linearisation $\Psi_1 \dots \Psi_m$ of the graph in (6), $w \circ \ell(\Psi_1) \dots \ell(\Psi_m) \circ u \in [v]$.

Proof. Let $sos([v]) = (occ(\tau), \prec, \sqsubset)$ and $\alpha \in occ_{|w|+1}(v)$. We first observe that if $\beta \in occ(v)$ is such that $\alpha \sqsubset \beta \sqsubset \alpha$ then $pos_v(\alpha) \leq pos_v(\beta) \leq pos_v(\alpha)$ which means that $\beta \in occ_{|w|+1}(v)$. Hence the elements of any $\sqsubset$-cycle comprising α belong to $occ_{|w|+1}(v)$. This further implies that if β is an element of a $\sqsubset$-cycle comprising α, then β is also an element of some $\sqsubset_v$-cycle comprising α and, moreover, the elements of such a cycle belong to $occ_{|w|+1}(v)$. Hence $\langle\alpha\rangle$ is a strongly connected component of the graph in (5).

That (6) is a directed acyclic graph follows from the acyclicity of $\widehat{\sqsubset}_v$. The last part follows directly from the definitions.

Folded so-structures can also be used to recover all the step sequences belonging to a comtrace.

Proposition 2. *Let τ be a comtrace. Then*

$$\tau = \{\widehat{\ell}(v) \mid v \in str(\widehat{sos}(\tau))\} \ , \tag{7}$$

where, for every $v = \Gamma_1 \dots \Gamma_k \in str(\widehat{sos}(\tau))$,

$$\widehat{\ell}(v) = \left(\bigcup\nolimits_{\Psi \in \Gamma_1} \ell(\Psi)\right) \dots \left(\bigcup\nolimits_{\Psi \in \Gamma_k} \ell(\Psi)\right) \ .$$

Proof. ($\supseteq$) Let $v = \Gamma_1 \dots \Gamma_k \in str(\widehat{sos}(\tau))$ and:

$$w = A_1 \dots A_k = \left(\bigcup\nolimits_{\Psi \in \Gamma_1} \Psi\right) \dots \left(\bigcup\nolimits_{\Psi \in \Gamma_k} \Psi\right) .$$

We have that $\Gamma_1, \dots, \Gamma_k$ are nonempty sets of folded actions such that $\widehat{occ}(\tau) = \Gamma_1 \uplus \dots \uplus \Gamma_k$,

- $(\Gamma_j \times \Gamma_i) \cap \widehat{\prec} = \varnothing$, for all $1 \leq i \leq j \leq k$; and
- $(\Gamma_j \times \Gamma_i) \cap \widehat{\sqsubset} = \varnothing$, for all $1 \leq i < j \leq k$.

Moreover, each folded action occurring in v is an equivalence class, and so different folded actions occurring in v are disjoint sets. Hence $A_1, \dots, A_k$ are nonempty sets of actions occurrences such that $occ(\tau) = A_1 \uplus \dots \uplus A_k$,

- $(A_j \times A_i) \cap \prec = \varnothing$, for all $1 \leq i \leq j \leq k$; and
- $(A_j \times A_i) \cap \sqsubset = \varnothing$, for all $1 \leq i < j \leq k$.

As a result, $w \in str(sos(\tau))$. Since $\widehat{\ell}(v) = \ell(w)$, we obtain, by (1), $\widehat{\ell}(v) \in \tau$.

($\subseteq$) To show the reverse inclusion, assume that $w = A_1 \dots A_k \in str(sos(\tau))$. Then each A_i can be partitioned into a set of folded actions Γ_i (see Proposition 1). One can see that $v = \Gamma_1 \dots \Gamma_k \in str(\widehat{sos}(\tau))$ satisfies $\widehat{\ell}(v) = \ell(w)$. Hence the inclusion follows from (1).

Folded actions can be derived directly from step sequences forming a comtrace.

Proposition 3. *Let α and β be two action occurrences of a comtrace τ. Then $\langle\alpha\rangle = \langle\beta\rangle$ iff $pos_w(\alpha) = pos_w(\beta)$, for every step sequence $w \in \tau$.*

Proof. ($\Longrightarrow$) $\langle\alpha\rangle = \langle\beta\rangle$ implies that, for every step sequence $v = X_1 \dots X_n \in str(sos(\tau))$, α and β belong to the same set X_i. Hence, by (1), $pos_w(\alpha) = pos_w(\beta)$, for every step sequence $w \in \tau$.
($\Longleftarrow$) Let $sos(\tau) = (occ(\tau), \prec, \sqsubset)$. The implication follows from the fact (see [6]) that if $\alpha \not\sqsubset \beta$ then there is a step sequence $w \in \tau$ such that $pos_w(\alpha) > pos_w(\beta)$.

Proposition 4. *For every step sequence wA, $\widehat{presos}(w)$ is a vertex-induced subgraph of $\widehat{presos}(wA)$.*

Proof. By Proposition 1 and (2,3), $\widehat{occ}(wA) = \widehat{occ}(w) \uplus \Gamma$, where Γ is a partition of $occ_{|w|+1}(wA)$ onto folded actions (as in Proposition 1). Furthermore,

$$\begin{aligned} \widehat{\prec}_w &= \widehat{\prec}_{wA}|_{\widehat{occ}(w) \times \widehat{occ}(w)} \\ \widehat{\sqsubset}_w &= \widehat{\sqsubset}_{wA}|_{\widehat{occ}(w) \times \widehat{occ}(w)} \end{aligned} \tag{8}$$

as well as

$$\begin{aligned} \widehat{\prec}_{wA} \cap \Gamma \times \widehat{occ}(wA) &= \varnothing \\ \widehat{\sqsubset}_{wA} \cap \Gamma \times \widehat{occ}(w) \quad &= \varnothing \, . \end{aligned} \tag{9}$$

As a result, $\widehat{presos}(w)$ is a vertex-induced subgraph of $\widehat{presos}(wA)$.

3.4 Hasse diagrams of comtraces

As a folded so-structure $\widehat{sos}(\tau) = (\widehat{occ}(\tau), \widehat{\prec}, \widehat{\sqsubset})$ comprises two nested posets, defining the *folded Hasse diagram* (or, simply, Hasse diagram) of a comtrace τ is straightforward:

$$H(\tau) = (\widehat{occ}(\tau), \widehat{\prec}^{\mathsf{h}}, \widehat{\sqsubset}^{\mathsf{h}}) \ ,$$

where

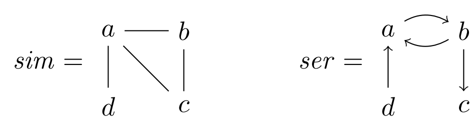
- $\widehat{\prec}^{\mathsf{h}} = \widehat{\prec} \setminus ((\widehat{\prec} \circ \widehat{\prec}) \cup (\widehat{\prec} \circ \widehat{\sqsubset}) \cup (\widehat{\sqsubset} \circ \widehat{\prec}))$; and
- $\widehat{\sqsubset}^{\mathsf{h}} = (\widehat{\sqsubset} \setminus (\widehat{\sqsubset} \circ \widehat{\sqsubset})) \setminus \widehat{\prec}^{\mathsf{h}}$.

Below we denote $H(w) = H(\tau) = (\widehat{occ}(w), \widehat{\prec}^{\mathsf{h}}_w, \widehat{\sqsubset}^{\mathsf{h}}_w)$, for every step sequence $w \in \tau$.

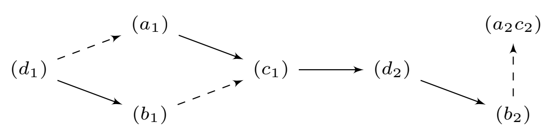
One can see that $H(\tau)$ is the smallest $\widehat{sos}(\tau)$-diagram which, in particular, implies the following.

Proposition 5. *If $w \in \tau$ then $\widehat{\prec}^{\mathsf{h}} \subseteq \widehat{\prec}_w$ and $\widehat{\sqsubset}^{\mathsf{h}} \subseteq \widehat{\sqsubset}_w$.*

Example 1. Consider a comtrace alphabet Θ with four actions $\Sigma = \{a, b, c, d\}$ together with a simultaneity and serialisability relations, *ser* and *sim*, given by:



The following is the folded Hasse diagram for the comtrace $[w] = (d)(ab)(c)(d)(abc)$, where we denote each action occurrence (x, i) by x_i:



Note that the set (a_2c_2) is the only non-singleton folded action in this case. □

Proposition 6. *For every step sequence wA, $H(w)$ is a vertex-induced subgraph of $H(wA)$.*

Proof. Follows from Propositions 4 and 5.

The last result can be strengthened for step sequences with steps corresponding to single folded actions. In what follows, for every step sequence w,

$$\widehat{tail}(w) = \{\Psi \in \widehat{occ}(w) \mid \Psi \cap tail(w) \neq \varnothing\} \ .$$

Proposition 7. *Let $wA = A_1 \dots A_p A$ be a step sequence such that $\widehat{sos}(w) = (\widehat{occ}(w), \widehat{\prec}, \widehat{\sqsubset})$. Moreover, let*

- *$occ_i(wA) = \Phi_i \in \widehat{occ}(w)$, for each $i \le p$; and*
- *$occ_{|w|+1}(wA) = \Phi \in \widehat{occ}(wA)$.*

Then the following hold:

1. *$\widehat{\prec}^{\mathsf{h}}_{wA} = \widehat{\prec}^{\mathsf{h}}_{w} \cup Z$, where Z is the set of all pairs (Φ_k, Φ) such that*
$$\Phi_k \in \widehat{tail}(w) \text{ and } \Phi_k \widehat{\prec}_{wA} \Phi ,$$
and there is no $\Phi_m \in \widehat{tail}(w)$ satisfying $k < m$ and
$$\Phi_k \widehat{\sqsubset} \Phi_m \widehat{\prec}_{wA} \Phi \quad \text{or} \quad \Phi_k \widehat{\prec} \Phi_m \widehat{\sqsubset}_{wA} \Phi .$$
2. *$\widehat{\sqsubset}^{\mathsf{h}}_{wA} = \widehat{\sqsubset}^{\mathsf{h}}_{w} \cup Z$, where Z is the set of all pairs (Φ_k, Φ) such that*
$$\Phi_k \in \widehat{tail}(w) \text{ and } \Phi_k \widehat{\sqsubset}_{wA} \Phi \text{ and } \neg \Phi_k \widehat{\prec}_{wA} \Phi ,$$
and there is no $\Phi_m \in \widehat{tail}(w)$ satisfying $k < m$ and
$$\Phi_k \widehat{\sqsubset} \Phi_m \widehat{\prec}_{wA} \Phi \quad \text{or} \quad \Phi_k \widehat{\sqsubset} \Phi_m \widehat{\sqsubset}_{wA} \Phi .$$

Proof. We first observe that, for all $\Psi = \Phi_i \in \widehat{occ}(w)$,

$$\Psi \widehat{\prec}^{\mathsf{h}}_{wA} \Phi \;\vee\; \Psi \widehat{\sqsubset}^{\mathsf{h}}_{wA} \Phi \implies \Psi \in \widehat{tail}(w) . \tag{10}$$

Indeed, suppose $\Psi \widehat{\prec}^{\mathsf{h}}_{wA} \Phi$ and $\Psi \notin \widehat{tail}(w)$. By $\Psi \widehat{\prec}^{\mathsf{h}}_{wA} \Phi$, there are $\alpha \in \Psi$ and $\beta \in \Phi$ such that $pos_{wA}(\alpha) < pos_{wA}(\beta) = p+1$ and $(\ell(\alpha), \ell(\beta)) \notin ser$.

Since $\Psi \notin \widehat{tail}(w)$, there are $i < j \le p$ and $\gamma \in \Phi_j$ such that $\Phi_j \in \widehat{tail}(w)$ and $\ell(\gamma) = \ell(\alpha)$. Clearly, $\Psi \widehat{\prec} \Phi_j$. We also have $\gamma \prec_{wA} \beta$, and so $\Phi_j \widehat{\prec}_{wA} \Phi$. Hence we have $\Psi \widehat{\prec} \Phi_j \widehat{\prec}_{wA} \Phi$, and so $\neg \Psi \widehat{\prec}^{\mathsf{h}}_{wA} \Phi$, yielding a contradiction. If $\Psi \widehat{\sqsubset}^{\mathsf{h}}_{wA} \Phi$, we proceed similarly. Thus (10) holds. We also note that, by Proposition 6,

$$\widehat{occ}(w) = \{\Phi_1, \dots, \Phi_p\} \quad \text{and} \quad \widehat{occ}(wA) = \widehat{occ}(w) \cup \{\Phi\} .$$

(1) Suppose that $\Phi_k \widehat{\prec}^{\mathsf{h}}_{wA} \Psi$ and $\neg \Phi_k \widehat{\prec}^{\mathsf{h}}_{w} \Psi$. Then, by Proposition 6, $\Psi = \Phi$. Hence, by (10), we have $\Phi_k \in \widehat{tail}(w)$ and, by Proposition 5, $\Phi_k \widehat{\prec}_{wA} \Phi$. Suppose that there is $\Phi_m \in \widehat{tail}(w)$ satisfying $k < m$ and

$$\Phi_k \widehat{\sqsubset} \Phi_m \widehat{\prec}_{wA} \Phi \quad \text{or} \quad \Phi_k \widehat{\prec} \Phi_m \widehat{\sqsubset}_{wA} \Phi .$$

Then $\neg \Phi_k \widehat{\prec}^{\mathsf{h}}_{wA} \Phi$, yielding a contradiction. Hence we obtained that $(\Phi_k, \Psi) \in Z$, and so $\widehat{\prec}^{\mathsf{h}}_{wA} \subseteq \widehat{\prec}^{\mathsf{h}}_{w} \cup Z$.

To show the reverse inclusion, we first observe that, by Proposition 6, $\widehat{\prec}^{\mathsf{h}}_{w} \subseteq \widehat{\prec}^{\mathsf{h}}_{wA}$. Let $(\Phi_k, \Phi) \in Z$. Then, clearly, $\Phi_k \widehat{\prec} \Phi$. Suppose $\neg \Phi_k \widehat{\prec}^{\mathsf{h}}_{wA} \Phi$. Then, by (10), there is $\Phi_l \in \widehat{tail}(w)$ such that $k < l$ and

$$\Phi_k \widehat{\sqsubset} \Phi_l \widehat{\prec}_{wA} \Phi \quad \text{or} \quad \Phi_k \widehat{\prec} \Phi_l \widehat{\sqsubset}_{wA} \Phi .$$

This, however, yields a contradiction with $(\Phi_k, \Phi) \in Z$.

(2) The proof is similar to that of part (1).

Given a step sequence w, perhaps the most direct way of constructing the graph of $H(w) = (\widehat{occ}(w), \widehat{\prec}^{\mathsf{h}}, \widehat{\sqsubset}^{\mathsf{h}})$ is to follow the definitions. First, we generate the graph of $sos(w) = (occ(w), \prec, \sqsubset)$ by taking the set of vertices $occ(w)$ and processing one-by-one all possible pairs of distinct vertices to derive $\prec_w$ and $\sqsubset_w$. Next, we apply the so-closure to get $sos(w)$, and then compute all folded actions obtaining $\widehat{occ}(w)$ and $\widehat{sos}(w)$. Then we remove all the arcs implied by *S2–S4*, in order to generate $H(w)$. The operations of applying the so-closure and removing unnecessary arcs are straightforward generalisations of the transitive closure of a binary relation. Clearly, the whole procedure is at least quadratic in the number of action occurrences of w. In the rest of the paper, we will provide a more efficient solution which can be regarded as linear.

4 Direct Construction of Hasse Diagrams

In this section, we present an algorithm aimed at constructing a folded Hasse diagram directly from a given representative v of a comtrace. More precisely, the input to the algorithm is a comtrace alphabet (Σ, sim, ser) and a step sequence $v \in \mathbb{S}^*$ with $\widehat{sos}(v) = (\widehat{occ}(v), \widehat{\prec}, \widehat{\sqsubset})$. We first describe the algorithm and provide its pseudo-code. After that, we evaluate its complexity.

The algorithm is *on-line*, which means that it generates $H(w)$ for the successive prefixes w of v, and during the construction of $H(w)$ the algorithm does not access any information about the (suffix) part of v which does belong to w. Moreover, the algorithm exploits the knowledge of the structure of the intermediate diagrams, following the characterisation of Hasse diagrams captured by Proposition 7. Note that the development of an on-line algorithm is possible thanks to Propositions 4 and 6 as well as the fact that, for each prefix w of v, $presos(w)$ is a vertex-induced subgraph of $presos(v)$ (see Section 3.2).

By Proposition 1, when discussing the operation of the algorithm, we may assume that $v = A_1 \ldots A_r$, where $occ_i(v) = \Phi_i \in \widehat{occ}(v)$, for each $i \leq r$. Ensuring that v is of such a form can be done either through pre-processing in which all steps of the original step sequence are linearlised as described in Proposition 1, or by linearising the currently processed step of the original step sequence, as done in the on-line pseudo-code given later in this section.

We now describe a single phase of the algorithm which starts with the Hasse diagram $H(w)$ for a step sequence $w = A_1 \ldots A_p$ $(p < r)$, and constructs the Hasse diagram $H(wA)$ for $wA = A_1 \ldots A_p A_{p+1}$.

The construction is based on Proposition 7 and, in particular, the set $\widehat{tail}(w)$. A crucial property is that only the elements from $\widehat{tail}(w)$ and $\Phi = \Phi_{p+1}$ can be connected by arcs added in the current phase.

To implement the generation of new arcs captured by Proposition 7, we maintain an auxiliary list TAIL of (pointers to) the elements of $\widehat{tail}(w)$, stored in the order in which they have been processed. That is, TAIL $= \Phi_{i_1} \ldots \Phi_{i_m}$ is such that $\widehat{tail}(w) = \{\Phi_{i_1}, \ldots, \Phi_{i_m}\}$ and $i_1 < \cdots < i_m$.

For each vertex of the diagram being constructed, i.e., each folded action Φ_i, we store two sets of folded actions, called *strong tail predecessors* (STP) and *weak tail predecessors* (WTP). The *invariant* property assumed at the beginning of the current phase (as well any other phase) is that, for each $\Phi_i \in \widehat{tail}(w)$,

$$\begin{aligned} \Phi_i.\mathrm{STP} &= \{\Phi_j \in \widehat{tail}(w) \mid \Phi_j \widehat{\prec} \Phi_i\} \\ \Phi_i.\mathrm{WTP} &= \{\Phi_j \in \widehat{tail}(w) \mid \Phi_j \widehat{\sqsubset} \Phi_i\} \;. \end{aligned} \tag{11}$$

We split the processing of vertex Φ, for which $\Phi.\mathrm{STP} = \Phi.\mathrm{WTP} = \varnothing$ initially, into two parts. In Part 1, we check, following Proposition 7, the necessity of adding arcs from $\widehat{tail}(w)$ to Φ by scanning the list TAIL from right-to-left (i.e., from Φ_{i_m} to Φ_{i_1}). For each Φ_i in TAIL, we attempt to add a new strong arc (in $\widehat{\prec}^{\mathsf{h}}$) or a new weak arc (in $\widehat{\sqsubset}^{\mathsf{h}}$), in the following way:

- If there are action occurrences $\alpha \in \Phi_i$ and $\beta \in \Phi$ such that $(\ell(\alpha), \ell(\beta)) \notin ser$, then we check whether Φ_i belongs to $\Phi.\mathrm{STP}$. If $\Phi_i \notin \Phi.\mathrm{STP}$, we add the arc $\Phi_i \widehat{\prec}^{\mathsf{h}} \Phi$ to $H(wA)$ and then add

$$\{\Phi_i\} \cup \Phi_i.\mathrm{STP} \cup \Phi_i.\mathrm{WTP}$$

  to both $\Phi.\mathrm{STP}$ and $\Phi.\mathrm{WTP}$.
- If there are no action occurrences $\alpha \in \Phi_i$ and $\beta \in \Phi$ such that $(\ell(\alpha), \ell(\beta)) \notin ser$, but there are $\alpha' \in \Phi_i$ and $\beta' \in \Phi$ such that $(\ell(\beta'), \ell(\alpha')) \notin ser$, then we check whether Φ_i belongs to $\Phi.\mathrm{WTP}$. If $\Phi_i \notin \Phi.\mathrm{WTP}$, we add the arc $\Phi_i \widehat{\sqsubset}^{\mathsf{h}}_{wA} \Phi$ to $H(wA)$ and then add $\Phi_i.\mathrm{STP}$ to $\Phi.\mathrm{STP}$, and also add

$$\{\Phi_i\} \cup \Phi_i.\mathrm{STP} \cup \Phi_i.\mathrm{WTP}$$

  to $\Phi.\mathrm{WTP}$.

At this point we have added all the necessary arcs, but TAIL still needs to be updated. In Part 2, we first append Φ at the end of TAIL, and then scan TAIL to check for the presence of Φ_i in $\widehat{tail}(wA)$, for every Φ_i in TAIL $\setminus \{\Phi\}$. Whenever $\Phi_i \notin \widehat{tail}(wA)$ we delete Φ_i from TAIL as well as from the $\Phi_j.\mathrm{STP}$ and $\Phi_j.\mathrm{WTP}$ sets, for all the remaining Φ_j's in TAIL.

To make the updating of TAIL efficient, we maintain information about those action occurrences included in a folded action Ψ which also belong to $tail(w)$. We do this by attaching to Ψ a set $\Psi.\mathrm{OCC}$ of actions which is initialised to $\ell(\Psi)$, i.e., it initially contains labels of all the action occurrences belonging to Ψ. During the check, if a folded action Φ_i still belongs to TAIL, we remove from $\Phi_i.\mathrm{OCC}$ all the elements of $\Phi.\mathrm{OCC}$ (i.e., labels of all the action occurrences belonging to the currently processed folded action). If, as a result, the set $\Phi_i.\mathrm{OCC}$ becomes empty, we know that Φ_i does not belong to $\widehat{tail}(wA)$. In such a case, we remove Φ_i from TAIL as well as from the $\Phi_j.\mathrm{STP}$ and $\Phi_j.\mathrm{WTP}$ sets, for all the Φ_j's in TAIL.

A pseudo-code of the resulting algorithm, divided into three parts, is given next.

Algorithm 1: Hasse diagram

INPUT: step sequence $v = A_1 \dots A_r$ over (Σ, sim, ser)
OUTPUT: Hasse diagram $H(v)$

for $i := 1$ to r **do**
  compute strongly connected components Γ of $(occ_i(A_1 \dots A_i), \sqsubset_{A_1 \dots A_i}|_{occ_i(A_1 \dots A_i) \times occ_i(A_1 \dots A_i)})$
  compute any linearisation $\Psi_1 \dots \Psi_m$ of $(\Gamma, \widehat{\sqsubset}|_{\Gamma \times \Gamma})$
  for all Φ in $\Psi_1 \dots \Psi_m$ **do**
    add new arcs {Part 1}
    update data structure {Part 2}
  end for
end for

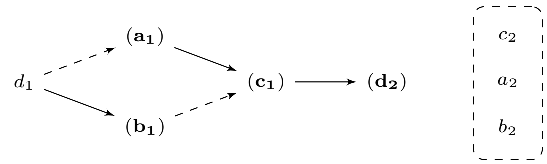
Algorithm 2: Part 1: adding new arcs

add vertex Φ {with OCC $= \ell(\Phi)$ and STP $=$ WTP $= \varnothing$}
for all Φ_i in TAIL {scanned right-to-left} **do**
  if $\Phi_i \widehat{\prec}_v \Phi$ and $\Phi_i \notin \Phi$.STP **then**
    add strong arc $\Phi_i \widehat{\prec}^{\mathsf{h}} \Phi$
    Φ.STP $:= \Phi$.STP $\cup \{\Phi_i\} \cup \Phi_i$.STP $\cup\, \Phi_i$.WTP
    Φ.WTP $:= \Phi$.WTP $\cup \{\Phi_i\} \cup \Phi_i$.STP $\cup\, \Phi_i$.WTP
  end if
  if $\Phi_i \widehat{\sqsubset}_v \Phi$ and $\neg \Phi_i \widehat{\prec}_v \Phi$ and $\Phi_i \notin \Phi$.WTP **then**
    add weak arc $\Phi_i \widehat{\sqsubset}^{\mathsf{h}} \Phi$
    Φ.STP $:= \Phi$.STP $\cup\, \Phi_i$.STP
    Φ.WTP $:= \Phi$.WTP $\cup \{\Phi_i\} \cup \Phi_i$.STP $\cup\, \Phi_i$.WTP
  end if
end for

Algorithm 3: Part 2: updating data structure

add Φ to TAIL
for all Φ_i in TAIL $\setminus \{\Phi\}$ **do**
  Φ_i.OCC $:= \Phi_i$.OCC $\setminus \Phi$.OCC
  if Φ_i.OCC $= \varnothing$ **then**
    remove Φ_i from TAIL
    for all $\Phi_j \in$ TAIL **do**
      remove Φ_i from Φ_j.STP
      remove Φ_i from Φ_j.WTP
    end for
  end if
end for

Example 2. Consider the comtrace alphabet from Example 1, a step sequence $w = (d)(ab)(c)(d)$ and a step $A = (abc)$. Then the Hasse diagram $H(w)$ together with the auxiliary data structure as well as the set A look as follows:



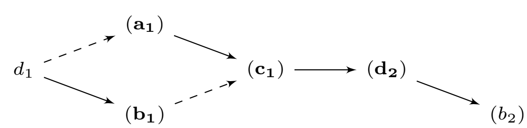
TAIL	STP	WTP
(a_1)	$\varnothing$	$\varnothing$
(b_1)	$\varnothing$	$\varnothing$
(c_1)	$\{(a_1)\}$	$\{(a_1),(b_1)\}$
(d_2)	$\{(a_1),(b_1),(c_1)\}$	$\{(a_1),(b_1),(c_1)\}$

In the diagram, the folded actions of $\widehat{tail}(w)$ are displayed in bold, while the new step A is enclosed within a dashed frame.

Processing the new step starts by computing strongly connected components Γ of the graph

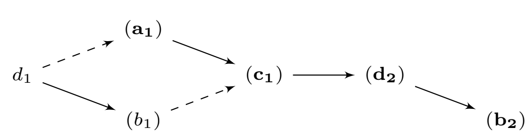
$$\begin{aligned}&(occ_5(wA), \sqsubset_{wA}|_{occ_5(wA)\times occ_5(wA)})\\ &\qquad = (\{a_2,b_2,c_2\},\{(a_2,c_2),(c_2,a_2),(b_2,c_2)\})\ .\end{aligned}$$

There are two strongly connected components in this case, (b_2) and (a_2c_2), and the graph $(\Gamma, \widehat{\sqsubset}|_{\Gamma\times\Gamma})$ contains just one arc, from (b_2) to (a_2c_2). Hence there is only one linearisation of the folded actions in Γ, viz. $(b_2)(a_2c_2)$, and as a result the algorithm will first process (b_2) and after that (a_2c_2).
Applying Part 1 for $\Phi = (b_2)$ leads to:



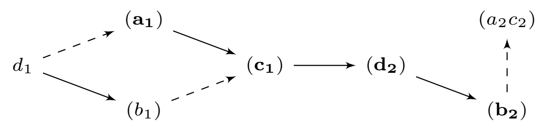
TAIL	STP	WTP
(a_1)	$\varnothing$	$\varnothing$
(b_1)	$\varnothing$	$\varnothing$
(c_1)	$\{(a_1)\}$	$\{(a_1),(b_1)\}$
(d_2)	$\{(a_1),(b_1),(c_1)\}$	$\{(a_1),(b_1),(c_1)\}$
(b_2)	$\{(a_1),(b_1),(c_1),(d_2)\}$	$\{(a_1),(b_1),(c_1),(d_2)\}$

and the subsequent application of Part 2 results in:



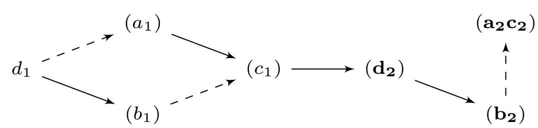
TAIL	STP	WTP
(a_1)	$\varnothing$	$\varnothing$
(c_1)	$\{(a_1)\}$	$\{(a_1)\}$
(d_2)	$\{(a_1),(c_1)\}$	$\{(a_1),(c_1)\}$
(b_2)	$\{(a_1),(c_1),(d_2)\}$	$\{(a_1),(c_1),(d_2)\}$

We then apply Part 1 for $\Phi = (a_2c_2)$ which leads to:



TAIL	STP	WTP
(a_1)	$\varnothing$	$\varnothing$
(c_1)	$\{(a_1)\}$	$\{(a_1)\}$
(d_2)	$\{(a_1),(c_1)\}$	$\{(a_1),(c_1)\}$
(b_2)	$\{(a_1),(c_1),(d_2)\}$	$\{(a_1),(c_1),(d_2)\}$
(a_2c_2)	$\{(a_1),(c_1),(d_2)\}$	$\{(a_1),(c_1),(d_2),(b_2)\}$

and the subsequent application of Part 2 results in:



TAIL	STP	WTP
(d_2)	$\varnothing$	$\varnothing$
(b_2)	$\{(d_2)\}$	$\{(d_2)\}$
(a_2c_2)	$\{(d_2)\}$	$\{(d_2),(b_2)\}$

□

We finally evaluate the complexity of the proposed algorithm. In what follows, d_i denotes the size of the step A_i for each $i \leq r$, k denotes the size of the alphabet Σ, and $n = d_1 + \cdots + d_r$ denotes the size of the input step sequence v.

To start with, the algorithm is on-line, and so we do not have to store the entire step sequence nor the entire Hasse diagram. All we need to do is maintain the structure representing the tail of the current diagram. It contains a list of at most k elements (as $|\widehat{tail}(w)| \leq k$, for every step sequence w) and, since each element has the size $\mathcal{O}(k)$, the memory complexity is $\mathcal{O}(k^2)$.

Evaluating time complexity of the algorithm is more involved. Let us start by evaluating the two parts of a single phase, i.e., Algorithms 2 and 3, when processing A_i.

Part 1 attempts to add new arcs, for every folded action Ψ in TAIL. Tests carried out in lines 3 and 8 have $\mathcal{O}(d_i \cdot k)$ complexity. Each of the set operations in lines 5,6,10 and 11 has $\mathcal{O}(k)$ complexity, while the time involved in adding an arc is constant. The whole loop (lines 2-13) has therefore $\mathcal{O}(d_i \cdot k^2)$ complexity. Hence the total contribution of Part 1 is equal to

$$\mathcal{O}(d_1 \cdot k^2) + \cdots + \mathcal{O}(d_r \cdot k^2) = \mathcal{O}(n \cdot k^2)\ .$$

Part 2 has also $\mathcal{O}(d_i \cdot k^2)$ complexity. In Algorithm 3, the outer loop has at most k iterations. In each iteration, we carry out some set operations of $\mathcal{O}(k)$ complexity. Moreover, the test in line 4 gives a positive result at most d_i times (as each label from $\ell(\Phi)$ may appear in the Φ_i.OCC's at most once). This means that the inner loop in line 6 may be executed at most d_i times with k repetitions each; thus the set operations in lines 7 and 8 may be executed at most $d_i \cdot k$ times. Therefore the overall time complexity is $\mathcal{O}(d_i \cdot k^2)$. Hence the total contribution of Part 2 is equal to

$$\mathcal{O}(d_1 \cdot k^2) + \cdots + \mathcal{O}(d_r \cdot k^2) = \mathcal{O}(n \cdot k^2)\ .$$

The time complexity of Algorithm 1 can now be calculated in the following way. The main loop has r iterations. The first part involves some operations on graphs of the size $\mathcal{O}(k^2)$ or $\mathcal{O}(k)$ (by the size of a graph we mean a total number of vertices and arcs). Finding strongly connected components and topologically sorting a directed acyclic graph are both linear in its size, and so we can carry out the first part in $\mathcal{O}(k^2)$ time, for each step. This contributes $\mathcal{O}(r \cdot k^2)$ towards the overall time complexity.

We can now add up the above complexity estimates. The total contributions of Part 1 and Part 2 were calculated separately, each providing one $\mathcal{O}(n \cdot k^2)$ component. The third component, corresponding to the pre-processing phase carried out for successive steps of the original step sequence, is $\mathcal{O}(r \cdot k^2)$. This gives:

$$\mathcal{O}(n \cdot k^2) + \mathcal{O}(n \cdot k^2) + \mathcal{O}(r \cdot k^2) = \mathcal{O}((n + r) \cdot k^2)\ .$$

Since $r \leq n$, we finally obtain that the total time complexity of the algorithm described in this section is equal to $\mathcal{O}(n \cdot\ k^2)$.

We can further observe that the factor k is fixed for a given system, and usually much smaller than n. Hence, the algorithm can in practice be considered as linear in the size of its input, and so optimal.

5 Applications

A major advantage of Hasse diagrams and the data structure used by the algorithm described above is a convenient and efficient representation of comtraces. We will now have a brief look at three of its possible applications.

To start with, Hasse diagrams provide an efficient support for checking the equality of two comtraces. This follows from the fact that two step sequences, w and v, belong to the same comtrace iff their Hasse diagrams are equal, i.e., $w \equiv_\Theta v \iff H(w) = H(v)$. Testing for equality of two graphs is linear in their size, and so using Hasse diagrams allows checking comtrace equivalence in $\mathcal{O}(n \cdot k^2)$ time, where n is the total number of action occurrences in the two step sequences, and k is the size of the action alphabet. Hence, for a fixed action alphabet, comtrace equivalence can be checked in linear time.

In addition to keeping explicit representation of $\widehat{tail}(w)$, it is also worth keeping information about the set

$$\widehat{head}(w) = \{\Phi \in \widehat{occ}(w) \mid \Phi \cap head(w) \neq \varnothing\} \ .$$

The extended structure supports an efficient concatenation of two comtraces, $[v]$ and $[w]$. This follows from the fact that the Hasse diagram $H(v \circ w)$ of the concatenated comtrace $[v \circ w]$ is a disjoint union of Hasse diagrams $H(v)$ and $H(w)$ together with some additional arcs, each such arc originating in a folded action of $\widehat{tail}(v)$ and ending in a folded action of $\widehat{head}(w)$.

Using $\widehat{head}(w)$ we can also efficiently generate the normal form of $[w]$. To do so, we first need to generate the set Γ comprising all the folded actions $\Phi \in \widehat{head}(w)$ such that $\Phi \subseteq head(w)$. Having done so, the first step of the normal form of w is $\bigcup_{\Phi \in \Gamma} \ell(\Phi)$. We then iterate the same procedure after deleting from $H(w)$ all the folded actions in Γ. The time complexity of the resulting algorithm is linear in the size of the Hasse diagram of $[w]$, and therefore equal to $\mathcal{O}(n \cdot k)$.

6 Conclusions

In this paper, we presented an efficient way of generating a graph theoretic representations of comtraces. We also provided a number of properties of folded versions of stratified order structures.

In future work we plan to extend our current results to cover also generalised comtraces and generalised so-structures [4, 5]. Another, arguably more challenging, problem is to use Hasse diagrams of comtraces to define an algebra of comtraces with a suitable iteration operator.

Acknowledgements

This research was supported by a fellowship funded by the “Enhancing Educational Potential of Nicolaus Copernicus University in the Disciplines of Mathematical and Natural Sciences” Project POKL.04.01.01-00-081/10.

A preliminary version [10] of this paper was presented at the ACSD’12 conference, Hamburg, June 2012.

References

1. Pierre Cartier and Dominique Foata. *Problèmes Combinatoires de Commutation et Réarrangements*, volume 85 of *LNM*. Springer, Berlin, 1969.
2. Volker Diekert and Yves Métivier. Partial commutation and traces. In *Handbook of Formal Languages*, volume 3, pages 457–533. Springer, 1997.
3. Ralph Freese. Automated lattice drawing. In Peter W. Eklund, editor, *ICFCA*, volume 2961 of *Lecture Notes in Computer Science*, pages 112–127. Springer, 2004.

4. Ryszard Janicki. Relational structures model of concurrency. *Acta Inf.*, 45(4):279–320, 2008.
5. Ryszard Janicki, Jetty Klein, and Maciej Koutny. Quotient monoids and concurrent behaviours. In Carlos Martín-Vide, editor, *Scientific Applications of Language Methods [7]*, chapter 6, pages 313–386. Imperial College Press, London, 2011.
6. Ryszard Janicki and Maciej Koutny. Semantics of inhibitor nets. *Inf. Comput.*, 123(1):1–16, 1995.
7. Carlos Martín-Vide, editor. *Scientific Applications of Language Methods*. Imperial College Press, 2011.
8. Antoni Mazurkiewicz. Concurrent program schemes and their interpretations. Daimi report pb-78, Aarhus University, 1977.
9. Łukasz Mikulski. Algebraic structure of combined traces. In Maciej Koutny and Irek Ulidowski, editors, *CONCUR*, volume 7454 of *Lecture Notes in Computer Science*, pages 456–470. Springer, 2012.
10. Lukasz Mikulski and Maciej Koutny. Hasse diagrams of combined traces. In Jens Brandt and Keijo Heljanko, editors, *ACSD*, pages 92–101. IEEE, 2012.
11. Walter Vogler. A generalization of traces. *ITA*, 25:147–156, 1991.
12. Henri Vogt. *Leçons sur la résolution algébrique des équations.* Cornell University Library historical math monographs. Nony, 1895.