A Characterization of Combined Traces using Labeled Stratified Order Structures

Dai Tri Man Lê

Department of Computer Science
University of Toronto
Canada

Petri Nets 2010
Outline

1 Background
   - Mazurkiewicz Traces
   - Combined Traces (Comtraces)

2 Comtraces as Labeled Stratified Order Structures
   - Motivation
   - Construction
   - Representation Theorems

3 Conclusion
(\(E, \text{ind}\)): an alphabet with an independence relation
Independent symbols can be commuted.
If \((a, b) \in \text{ind}\), then \(xaby \equiv xbay\).
A trace is an equivalence class of words.
Each equivalence class describes a partial order run of the system
Simple and elegant algebraic tool providing “true concurrency” semantics for concurrent systems with a static architecture.
E.g., elementary net systems, 1-safe Petri nets...
Combined Traces (Comtraces)

Limitation of Traces

1. as a special class of labeled partial order, traces cannot model more complicated causality relationships
2. elements of trace alphabet have no visible internal structure
Combined Traces (Comtraces)

Limitation of Traces

1. as a special class of labeled partial order, traces cannot model more complicated causality relationships.
2. elements of trace alphabet have no visible internal structure.

Main Ideas

comtraces (combined traces) [Janicki and Koutny 1995]

1. quotient of step sequence monoid
2. formal-linguistic representation of stratified order structures [Gaifman and Pratt 1987] [Janicki and Koutny 1991]
3. capture so-structure runs of the system
“a-priori semantics”: event completion takes some time.
"a-priori semantics": event completion takes some time.
Equivalent step sequence runs

1. \{a\}{b}

2.

“a-priori semantics”: event completion takes some time.
“a-priori semantics”: event completion takes some time.

Equivalent step sequence runs

1. \{a\}\{b\}\{c\}
2.
“a-priori semantics”: event completion takes some time.
 Equivalent step sequence runs

1. \{a\}{b}{c}
2. \{a\}

“a-priori semantics”: event completion takes some time.
"a-priori semantics": event completion takes some time.

Equivalent step sequence runs

1. \{a\}, \{b\}, \{c\}
2. \{a\}
Equivalent step sequence runs

1. \{a\}{b}\{c\}
2. \{a\}

“a-priori semantics”: event completion takes some time.
Elementary Net with Inhibitor Arcs [JK ’95]

Equivalent step sequence runs

1 \{a\}{b}{c}
2 \{a\}{b, c}

“a-priori semantics”: event completion takes some time.
Equivalent step sequence runs

1. \{a\}{b}\{c\}
2. \{a\}\{b, c\}

But NOT equivalent to \{a\}\{c\}\{b\}

“a-priori semantics”: event completion takes some time.
**Comtrace**

**Comtrace Concurrent Alphabet**

A tuple \((E, \text{sim}, \text{ser})\), where

- \(\text{sim}\) and \(\text{ser}\) model pairwise simultaneity and serializability
- \(\text{ser} \subseteq \text{sim} \subseteq E \times E\)
- \(\text{sim}\) is irreflexive and symmetric
  - defines the valid steps \(\implies\) valid step sequences.

**Comtrace Equivalence**

The least congruence \(\equiv\) satisfying for all steps \(A, B\) and \(C\),

if \(A \times B \subseteq \text{ser}\) and \(C = A \cup B\) then \(uCv \equiv uABv\)

Each equivalence class of \(\equiv\) is called a **comtrace**.
Comtrace example

From the previous example

- Define $sim = \{(b, c), (c, b)\}$ and $ser = \{(b, c)\}$
- The equivalent runs
  1. $\{a\}{b}{c}$
  2. $\{a\}{b, c}$

  can be grouped together into a comtrace (an equivalence class)

$$[[\{a\}{b}{c}\}] = \{\{a\}{b}{c}\}, \{a\}{b, c}\}$$
Motivations

Main research direction
Lift results and techniques from Mazurkiewicz traces to comtraces
Motivations

Among many interesting results on traces

Infinite traces and their applications

Gastin et al. (‘90–‘95) provides excellent theoretical foundation

Applications:

i. message sequence charts (Muscholl et al. ‘98, ‘99) (Mourin ‘02) (Kuske ‘03) (Gazagnaire et al. ‘09)

ii. static analysis of con. programs (Madhusudan et al. ‘06–current)

Temporal logics for finite and infinite traces

(Thiagarajan ‘94) (Mukund-Thiagarajan ‘96)
(Thiagarajan-Walukiewicz ‘97) (Walukiewicz ‘98)
(Leucker ‘02) (Diekert-Gastin ‘00, ‘02, ‘04) (Diekert ‘02)
(Gastin-Mukund ‘02) (Gastin et al. ‘03) (Gastin-Kuske ‘03) . . .

Observation:
Most of the results above utilize labeled-poset definition of traces!
Motivations

Among many interesting results on traces

1. **Infinite traces** and their applications
   - Gastin et al. (’90-’95) provides excellent theoretical foundation
   - Applications:
     - i. message sequence charts (Muscholl et al. ’98, ’99) (Mourin ’02)
       (Kuske ’03) (Gazagnaire et al. ’09)
     - ii. static analysis of con. programs (Madhusudan et al. ’06-current)
Motivations

Among many interesting results on traces

1. **Infinite traces** and their applications
   - Gastin et al. (’90-’95) provides excellent theoretical foundation
   - Applications:
     1. **message sequence charts** (Muscholl et al. ’98, ’99) (Mourin ’02) (Kuske ’03) (Gazagnaire et al. ’09)
     2. static analysis of con. programs (Madhusudan et al. ’06-current)

2. **Temporal logics** for finite and infinite traces
   - (Thiagarajan ’94) (Mukund-Thiagarajan ’96)
   - (Thiagarajan-Walukiewicz ’97) (Walukiewicz ’98)
   - (Leucker ’02) (Diekert-Gastin ’00, ’02, ’04) (Diekert ’02)
   - (Gastin-Mukund ’02) (Gastin et al. ’03) (Gastin-Kuske ’03) ...
Motivations

Among many interesting results on traces

1. **Infinite traces** and their applications
   - Gastin et al. (’90-’95) provides excellent theoretical foundation
   - Applications:
     - i. message sequence charts (Muscholl et al. ’98, ’99) (Mourin ’02) (Kuske ’03) (Gazagnaire et al. ’09)
     - ii. static analysis of con. programs (Madhusudan et al. ’06-current)

2. **Temporal logics** for finite and infinite traces
   - (Thiagarajan ’94) (Mukund-Thiagarajan ’96)
   - (Thiagarajan-Walukiewicz ’97) (Walukiewicz ’98)
   - (Leucker ’02) (Diekert-Gastin ’00, ’02, ’04) (Diekert ’02)
   - (Gastin-Mukund ’02) (Gastin et al. ’03) (Gastin-Kuske ’03) . . .

Observation:
Most of the results above utilize **labeled-poset definition** of traces!
Question:

Given a trace alphabet \((E, \text{ind})\), how to decide if a labeled poset represents a trace?

Hasse diagram
Question:
Given a trace alphabet \((E, \text{ind})\), how to decide if a labeled poset represents a trace?

Two main conditions:

1. Non-connected nodes are labeled with independent events
2. Adjacent nodes are labeled with dependent events

Hasse diagram
Traces as Labeled Posets

Question:
Given a trace alphabet \((E, ind)\), how to decide if a labeled poset represents a trace?

Two main conditions:
1. Non-connected nodes are labeled with independent events
2. Adjacent nodes are labeled with dependent events

Hasse diagram

Our goal
To give a similar order-theoretic characterization for comtraces!
Stratified order structure (so-structure) [Janicki-Koutny ’91]

A triple $S = (X, \prec, \sqsubseteq)$ and binary relations $\prec, \sqsubseteq \subseteq X \times X$ satisfying

1. $a \not\sqsubseteq a$
2. $a \prec b \implies a \sqsubseteq b$
3. $a \sqsubseteq b \sqsubseteq c \land a \neq c \implies a \sqsubseteq c$
4. $a \sqsubseteq b \prec c \lor a \prec b \sqsubseteq c \implies a \prec c$

Intuitively, $\prec$ means "earlier than", and $\sqsubseteq$ means "not later than"

"not later than" = "earlier than" or "simultaneous"
Example of so-structure

Theorem (JK ’95)
Every comtrace uniquely defines a labeled so-structure.

“earlier than” ∩ “not later than”:

“not later than”:
Comtraces as labeled so-structures

Question:
Given a comtrace alphabet \(\{a, b, c\}, \sim, \ser\), how to decide if a labeled so-structure represents a comtrace?

“earlier than” \(\cap\) “not later than”:

“not later than”:
Comtraces as labeled so-structures

Question:
Given a comtrace alphabet \( \{a, b, c\}, sim, ser \), how to decide if a labeled so-structure represents a comtrace?

```
\[ \text{“earlier than” } \cap \text{“not later than”}: \quad \rightarrow \]  
```

Obstacles

1. complication of having both \( \rightarrow \) and \( \rightarrow \)
2. complication of having both \( sim \) and \( ser \)
3. cycles make things less intuitive
Stratified Order Structures

Stratified order structure (so-structure) [Janicki-Koutny ’91]

A triple \( S = (X, \prec, \sqsubseteq) \) and binary relations \( \prec, \sqsubseteq \subseteq X \times X \) satisfying

1. \( a \nprec a \)
2. \( a \prec b \implies a \sqsubseteq b \)
3. \( a \sqsubseteq b \sqsubseteq c \wedge a \neq c \implies a \sqsubseteq c \)
4. \( a \sqsubseteq b \prec c \lor a \prec b \sqsubseteq c \implies a \prec c \)

Intuitively, \( \prec \) means "earlier than", and \( \sqsubseteq \) means "not later than"
Stratified Order Structures

Stratified order structure (so-structure) [Janicki-Koutny ’91]

A triple \( S = (X, \prec, \sqsubseteq) \) and binary relations \( \prec, \sqsubseteq \subseteq X \times X \) satisfying

1. \( a \not\sqsubseteq a \)
2. \( a \prec b \implies a \sqsubseteq b \)
3. \( a \sqsubseteq b \sqsubseteq c \land a \neq c \implies a \sqsubseteq c \)
4. \( a \sqsubseteq b \prec c \lor a \prec b \sqsubseteq c \implies a \prec c \)

Intuitively, \( \prec \) means “earlier than”, and \( \sqsubseteq \) means “not later than”

Observation

The “not later than” relation \( \sqsubseteq \) is a strict pre-order!
Definition ($\sqsubseteq$-cycle equivalence relation)

Vertices $\alpha$ and $\beta$ are $\sqsubseteq$-cycle equivalent if and only if $\alpha \sqsubseteq \beta$ and $\beta \sqsubseteq \alpha$. 
Quotient construction

**Definition (□-cycle equivalence relation)**

Vertices $\alpha$ and $\beta$ are □-cycle equivalent if and only if $\alpha \sqsubset \beta$ and $\beta \sqsubset \alpha$.

Dai Lê (University of Toronto)
Comtraces as labeled so-structures

Question:
Given a comtrace alphabet ($\{a, b, c\}$, $\text{sim}$, $\text{ser}$), how to decide if a labeled so-structure is a comtrace?

Hasse diagram

---

## Conditions (Definition 10)

1. **Adjacent nodes** $[\alpha] \rightarrow [\beta]$ satisfies $\lambda([\alpha]) \times \lambda([\beta]) \not\subseteq \text{ser}$
2. **Adjacent nodes** $[\alpha] \rightarrow [\beta]$ satisfies $\lambda([\beta]) \times \lambda([\alpha]) \not\subseteq \text{ser}$
3. **Label set of a node** $[\alpha]$ can't be serializable w.r.t. $\text{ser}$
4. ...
5. ...

---

Dai Lê (University of Toronto)
Comtraces as labeled so-structures

Question:
Given a comtrace alphabet \(\{a, b, c\}, \text{sim}, \text{ser}\), how to decide if a labeled so-structure is a comtrace?

Conditions (Definition 10)

\(\lambda\) denotes the labeling function

1. adjacent nodes \([\alpha] \rightarrow [\beta]\) satisfies \(
\lambda([\alpha]) \times \lambda([\beta]) \not\subseteq \text{ser}\)

2. adjacent nodes \([\alpha] \rightarrow [\beta]\) satisfies \(
\lambda([\beta]) \times \lambda([\alpha]) \not\subseteq \text{ser}\)

3. label set of a node \([\alpha]\) can’t be serializable w.r.t. \(\text{ser}\)

4. 

5. 

Hasse diagram
Theorem 3

Given a comtrace alphabet \( \theta \), let

- \( \mathbb{S}_{\equiv \theta}^{*} \): comtraces over \( \theta \),
- \( \text{LCT}(\theta) \): Isos-comtraces over \( \theta \).

Then the following diagram commutes

\[
\begin{align*}
\text{id}_{\mathbb{S}_{\equiv \theta}^{*}} & \quad \text{ct2lct} \\
\mathbb{S}_{\equiv \theta}^{*} & \quad \LCT(\theta) \\
\text{lct2ct} & \quad \text{id}_{\LCT(\theta)}
\end{align*}
\]

This is the converse of the main theorem in [JK ’95].
Theorem 4

Given a comtrace alphabet $\theta$, let

- $\text{LCT}(\theta)$: Isos-comtraces over $\theta$
- $\text{CDG}(\theta)$: combined dependency graphs [Kleijn-Koutny ’08] over $\theta$
  - analogous to dependency graphs for Mazurkiewicz traces

Then the following diagram commutes
Theorems 3 and 4: the following diagram commutes

\[ \text{id}_{S^*/\equiv \theta} \quad \text{S^*/} \equiv \theta \quad \text{LCT}(\theta) \quad \text{id}_{\text{LCT}(\theta)} \]

\[ \text{ct2lct} \quad \text{lct2ct} \quad \text{id} \]

\[ \text{CDG}(\theta) \quad \text{dep2lct} \quad \text{lct2dep} \]

Theorems 5 and 6: these mappings are monoid isomorphisms.
Conclusion

Summary

1. We formally show that comtraces and lsos-comtraces and combined dependency graphs are equivalent models.
2. Formal-linguistic, order-theoretic and graph-theoretic respectively.

Future Works

1. Similar results for generalized combined traces [JL '08] [JL '09]
2. Infinite comtraces?
3. Linear temporal logics for comtraces?
4. Applications of comtraces?
Conclusion

Summary

1. We formally show that comtraces and lsos-comtraces and combined dependency graphs are equivalent models.
2. Formal-linguistic, order-theoretic and graph-theoretic respectively.
3. More generalized trace languages using step sequences, where “congruence” is defined from interactions of steps.
Conclusion

Summary

1. We formally show that comtraces and Isos-comtraces and combined dependency graphs are equivalent models.
2. Formal-linguistic, order-theoretic and graph-theoretic respectively.
3. More generalized trace languages using step sequences, where “congruence” is defined from interactions of steps.

Future Works

1. Similar results for generalized combined traces [JL ’08] [JL ’09]
2. Infinite comtraces?
3. Linear temporal logics for comtraces?
4. Applications of comtraces?
Thank you very much for your attention!