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**

Mazurkiewicz Traces (Mazurkiewicz 1977)

- (E, ind): an alphabet with an independence relation
- Independent symbols can be commuted.
  - If $(a, b) \in ind$, 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

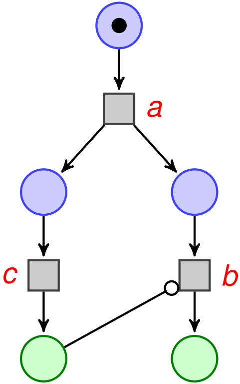
1. as a special class of labeled partial order, traces cannot model more complicated causality relationships
2. elements of trace alphabet have no visible internal structure

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

Elementary Net with Inhibitor Arcs [JK ’95]

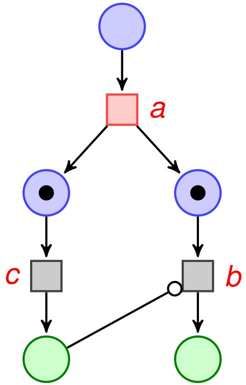


Equivalent step sequence runs

1.
2.

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

Elementary Net with Inhibitor Arcs [JK '95]

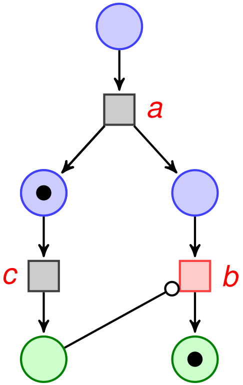


Equivalent step sequence runs

1. $\{a\}$
2.

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

Elementary Net with Inhibitor Arcs [JK '95]

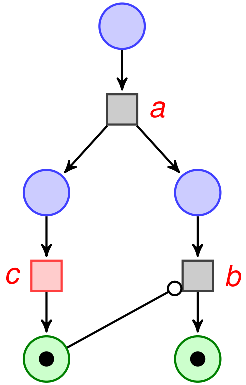


Equivalent step sequence runs

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

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

Elementary Net with Inhibitor Arcs [JK '95]

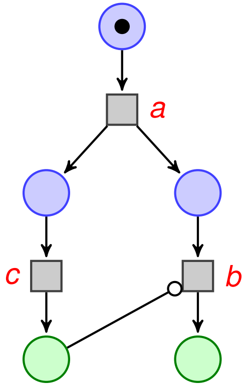


Equivalent step sequence runs

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

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

Elementary Net with Inhibitor Arcs [JK ’95]

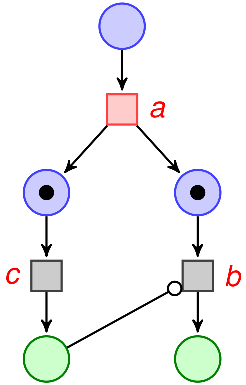


Equivalent step sequence runs

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

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

Elementary Net with Inhibitor Arcs [JK '95]

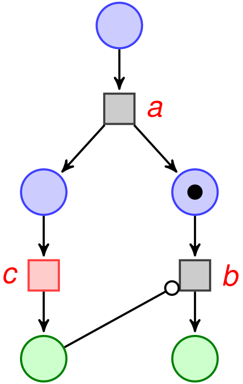


Equivalent step sequence runs

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

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

Elementary Net with Inhibitor Arcs [JK '95]

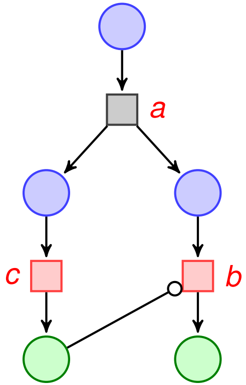


Equivalent step sequence runs

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

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

Elementary Net with Inhibitor Arcs [JK '95]

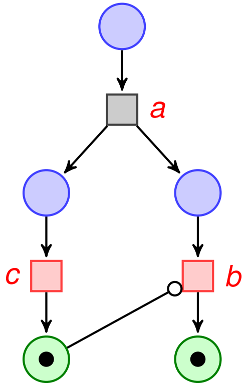


Equivalent step sequence runs

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

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

Elementary Net with Inhibitor Arcs [JK '95]

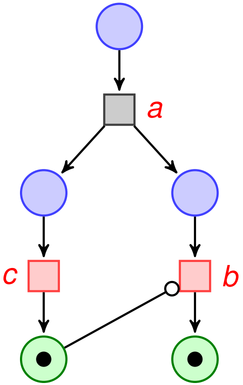


Equivalent step sequence runs

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

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

Elementary Net with Inhibitor Arcs [JK '95]



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, sim, ser), where

- sim and ser model pairwise simutaneity and serializability
- $ser \subseteq sim \subseteq E \times E$
- sim is irreflexive and symmetric
  - defines the valid steps $\Longrightarrow$ valid step sequences.

Comtrace Equivalence

the least congruence $\equiv$ satisfying for all steps A, B and C,

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

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

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:
     - 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) . . .

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) ...

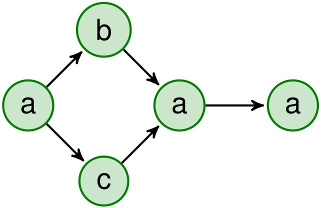
Observation:

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

Traces as Labeled Posets

Question:

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

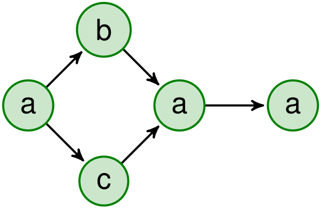


Hasse diagram

Traces as Labeled Posets

Question:

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



Hasse diagram

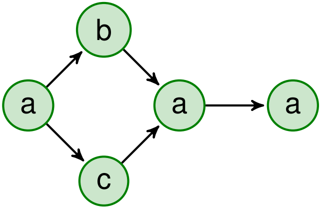
Two main conditions:

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

Traces as Labeled Posets

Question:

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



Hasse diagram

Two main conditions:

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

Our goal

To give a similar order-theoretic characterization for comtraces!

Stratified Order Structures

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

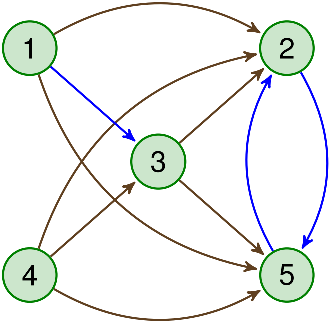
a triple $S = (X, \prec, \sqsubset)$ and binary relations $\prec, \sqsubset \subseteq X \times X$ satisfying

1. $a \not\sqsubset a$
2. $a \prec b \implies a \sqsubset b$
3. $a \sqsubset b \sqsubset c \ \wedge \ a \neq c \implies a \sqsubset c$
4. $a \sqsubset b \prec c \ \vee \ a \prec b \sqsubset c \implies a \prec c$

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

"not later than" = "earlier than" or "simultaneous"

Example of so-structure



"earlier than" ∩ "not later than": ⟶

"not later than": ⟶

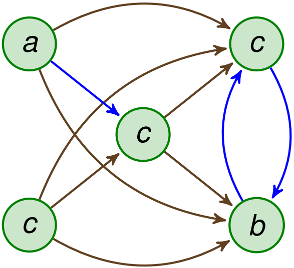
Theorem (JK '95)

Every comtrace uniquely defines a labeled so-structure.

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?



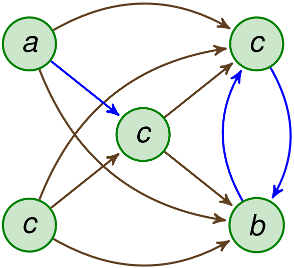
"earlier than" ∩ "not later than": ⟶

"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" ∩ "not later than": $\longrightarrow$

"not later than": $\longrightarrow$

Obstacles

1. complication of having both $\longrightarrow$ and $\longrightarrow$
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 $S = (X, \prec, \sqsubset)$ and binary relations $\prec, \sqsubset \subseteq X \times X$ satisfying

1. $a \not\sqsubset a$
2. $a \prec b \implies a \sqsubset b$
3. $a \sqsubset b \sqsubset c \wedge a \neq c \implies a \sqsubset c$
4. $a \sqsubset b \prec c \vee a \prec b \sqsubset c \implies a \prec c$

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

Stratified Order Structures

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

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

1. $a \not\sqsubset a$
2. $a \prec b \implies a \sqsubset b$
3. $a \sqsubset b \sqsubset c \wedge a \neq c \implies a \sqsubset c$
4. $a \sqsubset b \prec c \vee a \prec b \sqsubset c \implies a \prec c$

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

Observation

The "not later than" relation $\sqsubset$ is a strict pre-order!

Quotient construction

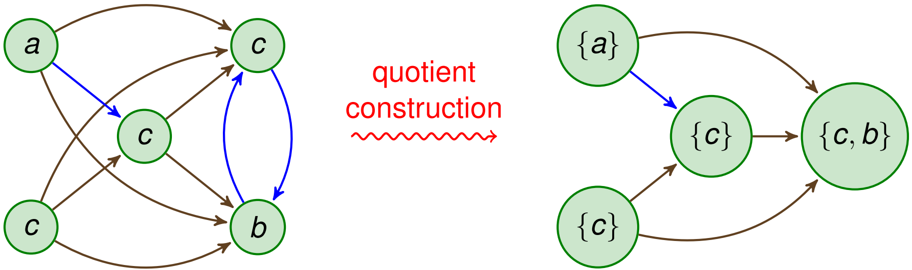
Definition ($\sqsubset$-cycle equivalence relation)

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

Quotient construction

Definition ($\sqsubset$-cycle equivalence relation)

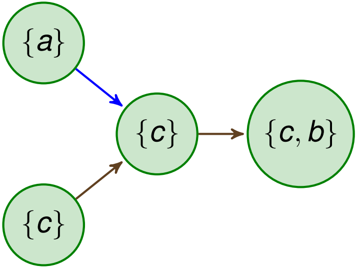
Vertices α and β are $\sqsubset$-cycle equivalent if and only if $\alpha \sqsubset \beta$ and $\beta \sqsubset \alpha$.



Comtraces as labeled so-structures

Question:

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

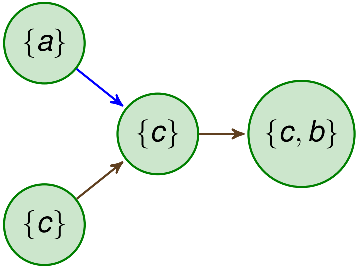


Hasse diagram

Comtraces as labeled so-structures

Question:

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



Hasse diagram

Conditions (Definition 10)

λ denotes the labeling function

1. adjacent nodes $[\alpha] \longrightarrow [\beta]$ satisfies $\lambda([\alpha]) \times \lambda([\beta]) \not\subseteq ser$
2. adjacent nodes $[\alpha] \longrightarrow [\beta]$ satisfies $\lambda([\beta]) \times \lambda([\alpha]) \not\subseteq ser$
3. label set of a node $[\alpha]$ can't be serializable w.r.t. ser
4. ...
5. ...

Representation Theorems

Theorem 3

Given a comtrace alphabet θ, let

- $\mathbb{S}^*/\equiv_{\boldsymbol{\theta}}$: comtraces over θ,
- $\mathsf{LCT}(\theta)$: lsos-comtraces over θ.

Then the following diagram commutes

$$\begin{tikzcd}
\mathbb{S}^*/\equiv_{\boldsymbol{\theta}} \arrow[loop left, "\mathrm{id}_{\mathbb{S}^*/\equiv_\theta}"] \arrow[rr, bend left, "\mathrm{ct2lct}"] & & \mathsf{LCT}(\theta) \arrow[loop right, "\mathrm{id}_{\mathsf{LCT}(\theta)}"] \arrow[ll, bend left, "\mathrm{lct2ct}"]
\end{tikzcd}$$

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

Representation Theorems

Theorem 4

Given a comtrace alphabet θ, let

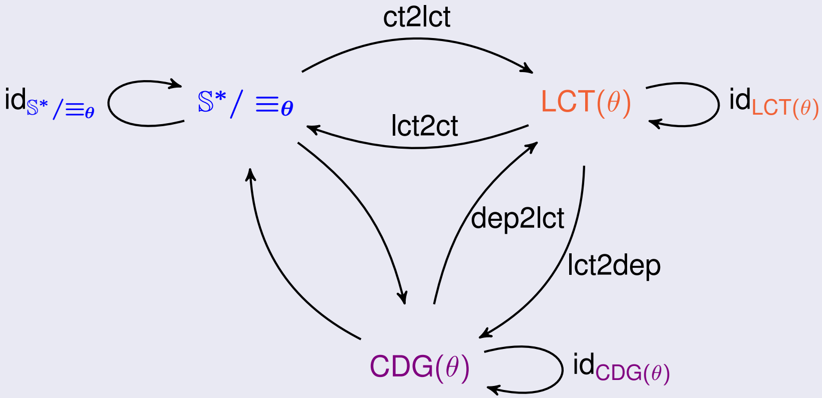
- LCT(θ): lsos-comtraces over θ
- CDG(θ): combined dependency graphs [Kleijn-Koutny '08] over θ
  - analogous to dependency graphs for Mazurkiewicz traces

Then the following diagram commutes

$$\begin{tikzcd}
\mathrm{LCT}(\theta) \arrow[loop left, "\mathrm{id}_{\mathrm{LCT}(\theta)}"] \arrow[rr, bend left, "\mathrm{lct2dep}"] & & \mathrm{CDG}(\theta) \arrow[loop right, "\mathrm{id}_{\mathrm{CDG}(\theta)}"] \arrow[ll, bend left, "\mathrm{dep2lct}"]
\end{tikzcd}$$

Representation Theorems

- Theorems 3 and 4: the following diagram commutes



- 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.

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 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.

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!