Specification of labelled reconfigurable graphs in Marge

José Proença
based on work with Alexandre Madeira, Manuel Martins, David Tinoco @ Univ. Aveiro, Portugal
ReacTS @ SEFM, Aveiro, 5 November 2024

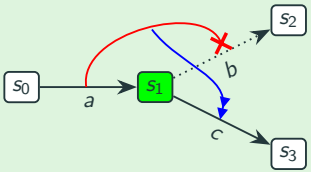
CISTER & U.Porto, Porto, Portugal

CISTER - Research Centre in
Real-Time & Embedded
Computing Systems

U.PORTO
FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

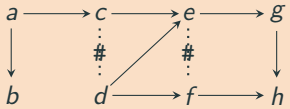
Reconfigurable Graphs and Event Structures

Reconfigurable graphs in Marge



by Tinoco, Madeira, Martins, Proença [FACS'24]

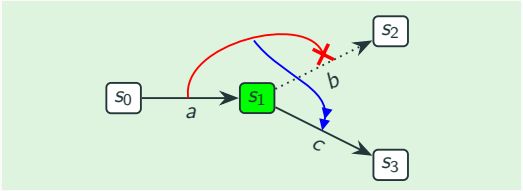
(Bundle) Event structures



by Nielsen, Plotkin, and Winskel [TCS'81] *and Langerak* [FORTE'92]

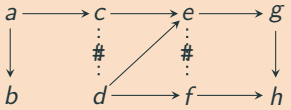
Reconfigurable Graphs and Event Structures

Reconfigurable graphs in Marge



by Tinoco, Madeira, Martins, Proença [FACS'24]

(Bundle) Event structures



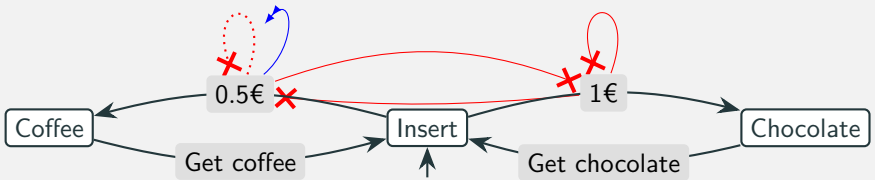
by Nielsen, Plotkin, and Winskel [TCS'81] *and Langerak* [FORTE'92]

Goal: Get insights over reconfigurable graphs with Marge;
Investigate dependencies & conflicts between reconfigurable graphs

Reconfigurable Graphs

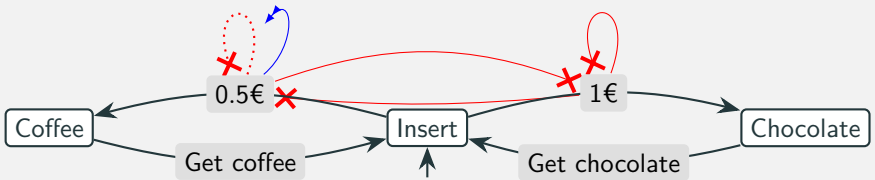
Reconfigurable Coffee Machine

The reconfigurable graph

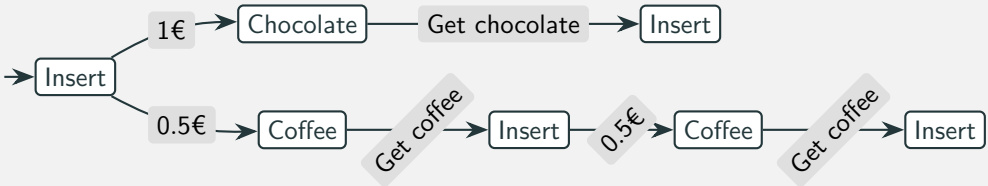


Reconfigurable Coffee Machine

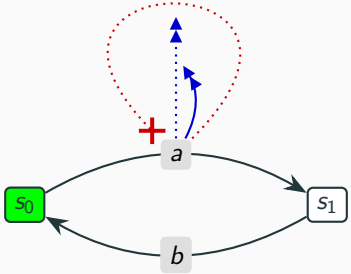
The reconfigurable graph



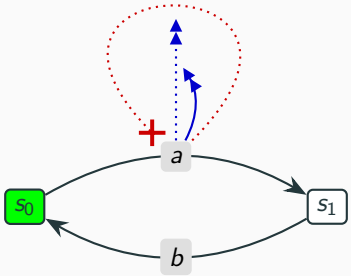
can be encoded as



Reconfigurable Counter



Reconfigurable Counter



When can these be useful?

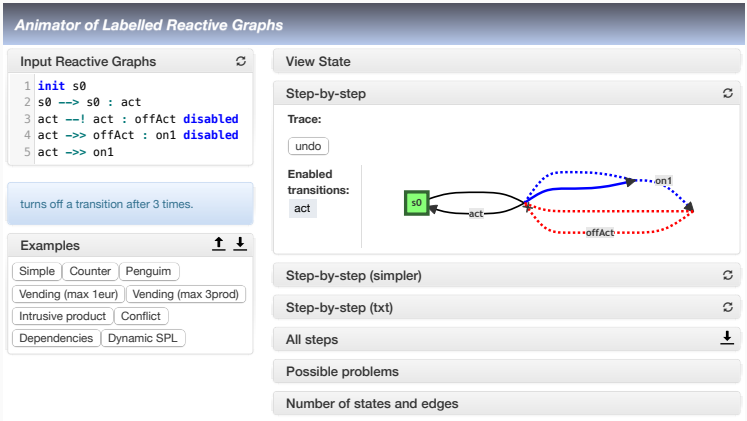
Tool to experiment with semantics/compositions:

https://fm-dcc.github.io/marge

Marge – animator of Multi Action Reactive Graphs

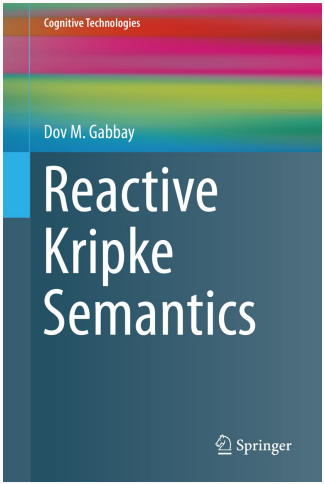
- Developed in Scala, using CAOS (generating JavaScript with ScalaJS)
- Static website that loads a compiled JavaScript (fully offline, no server)

- Input or load example
- Run step-by-step
- Run all steps
- Find possible problems
- Count states/edges



https://fm-dcc.github.io/MARGe

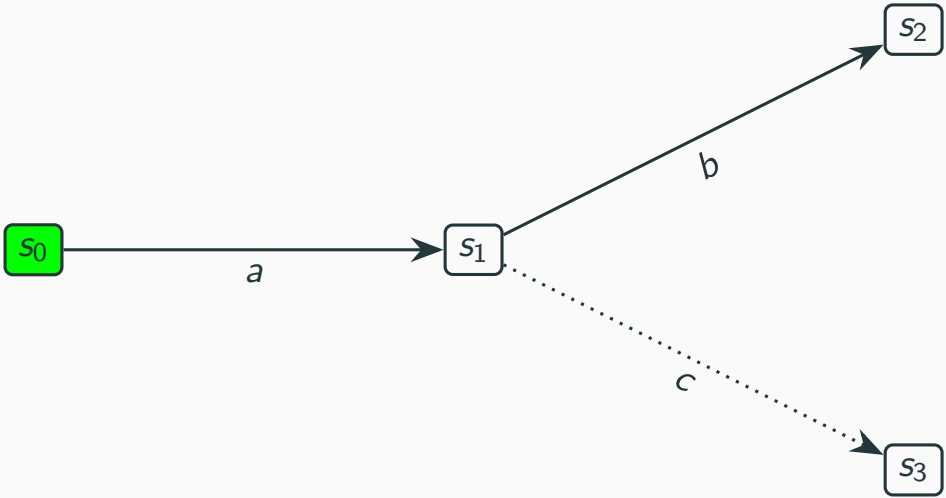
Reactive graphs: A reconfigurability dimension on LTS

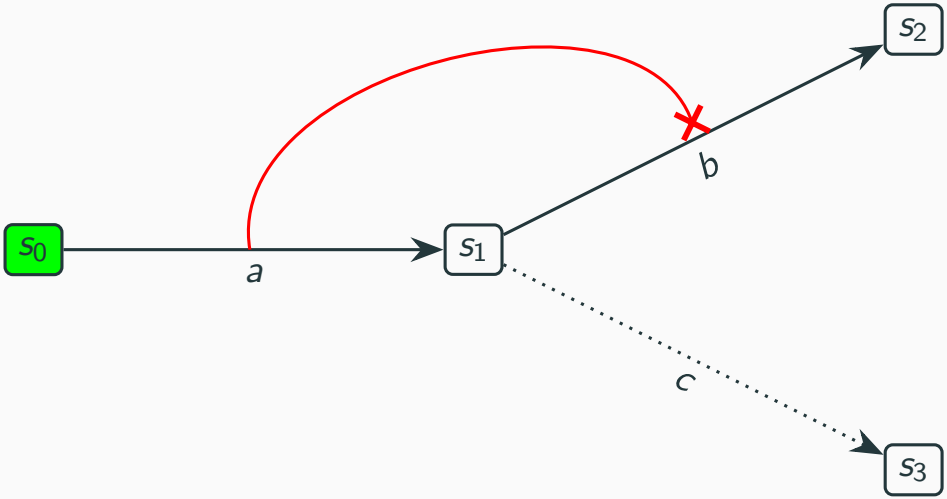


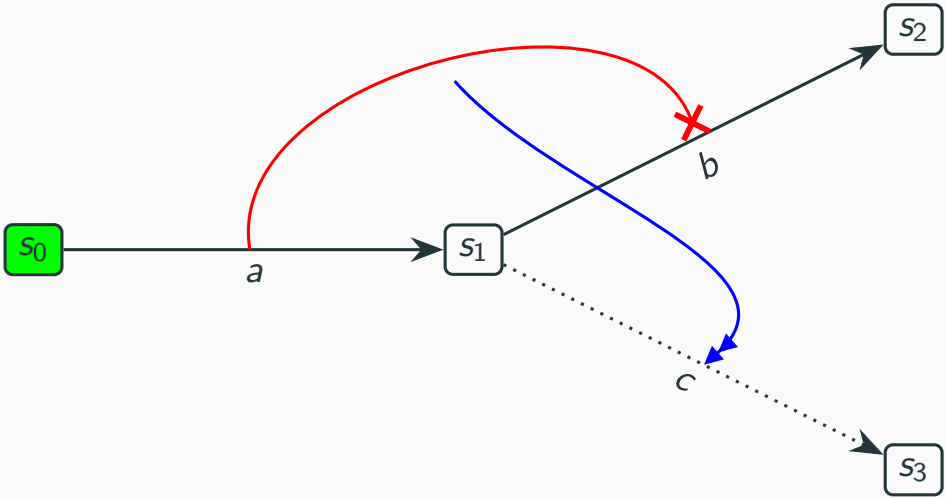
"In computer science the word reactivity has been used to denote systems that react to their environment and are not meant to terminate, as coined by Pnueli and Harel in [On the development of reactive systems, 1985]. In this work the word has a different meaning, reactive systems are history-dependent relational structures, where the accessibility relation is determined not only by the point where one is, but also by the previous transitions"

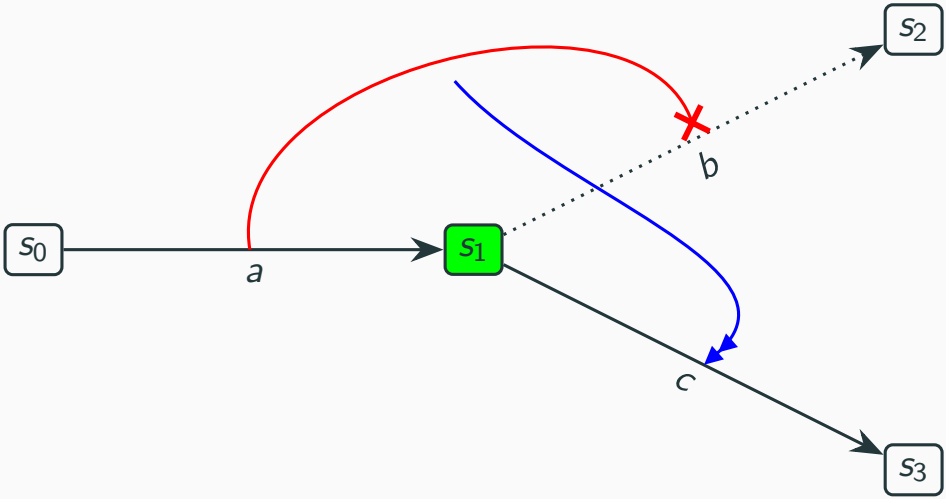
Dov M. Gabbay (2013)

I call Reconfigurable Graph instead of Reactive Graph in this talk


s_0
s_1
s_2
s_3
a
b
c


s_0
s_1
s_2
s_3
a
b
c


s_2
b
s_0
a
s_1
c
s_3

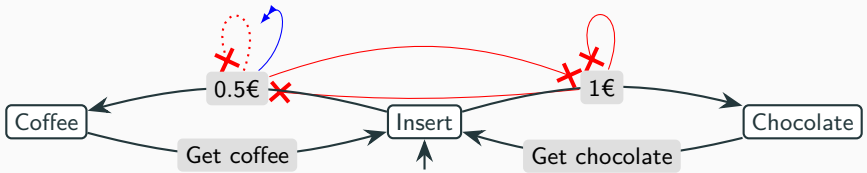

s_2
b
s_0
a
s_1
c
s_3

A labelled version of a reactive graph

A Multi-Actions Reactive Graph is a tuple $M = (W, Act, E, \rightarrow, \twoheadrightarrow, \multimap, \dot{\cdot}, w_0, \alpha_0)$ where:

- W – states
- Act – actions
- E – edges
- $w_0 \in W$ – initial state;
- $\alpha_0 \subseteq E$ – initially active edges

- $\rightarrow \subseteq W \times Act \times W$ – ground edges
- $\twoheadrightarrow \subseteq E \times E$ – activating edges
- $\multimap \subseteq E \times E$ – deactivating edges
- $\dot{\cdot} : E \longrightarrow (\rightarrow \cup \twoheadrightarrow \cup \multimap)$ – internal details of edges



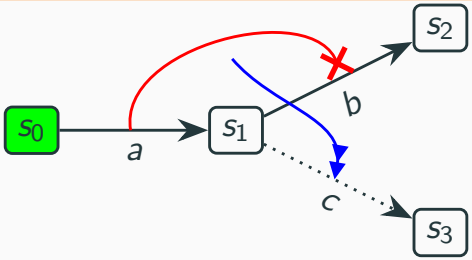
Semantics

A reconfigurable graph M can evolve its configuration $\langle w_0, \alpha_0 \rangle$ by the rule

$$\frac{\exists e \in \alpha \quad \cdot \quad \overline{e} = w \xrightarrow{a} w' \quad \wedge \quad \alpha' = \big(\alpha \cup \mathsf{on}(e, \alpha)\big) \backslash \mathsf{off}(e, \alpha)}{\langle w, \alpha \rangle \quad \xrightarrow{a}_M \quad \langle w', \alpha' \rangle}$$

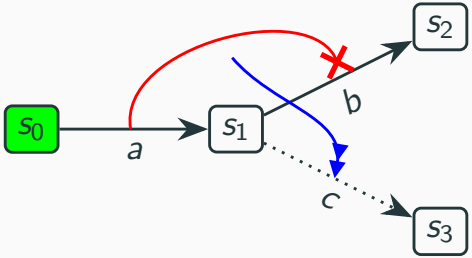
A reconfigurable graph M can evolve its configuration $\langle w_0, \alpha_0 \rangle$ by the rule

$$\frac{\exists e \in \alpha \quad \cdot \quad \overline{e} = w \xrightarrow{a} w' \quad \wedge \quad \alpha' = \big(\alpha \cup \mathsf{on}(e,\alpha)\big) \backslash \mathsf{off}(e,\alpha)}{\langle w, \alpha \rangle \quad \xrightarrow{a}_M \quad \langle w', \alpha' \rangle}$$



A reconfigurable graph M can evolve its configuration $\langle w_0, \alpha_0 \rangle$ by the rule

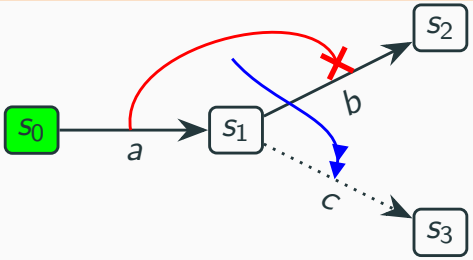
$$\frac{\exists e \in \alpha \quad \cdot \quad \overline{e} = w \xrightarrow{a} w' \quad \wedge \quad \alpha' = \big(\alpha \cup \mathsf{on}(e,\alpha)\big) \backslash \mathsf{off}(e,\alpha)}{\langle w, \alpha \rangle \quad \xrightarrow{a}_M \quad \langle w', \alpha' \rangle}$$



$\langle s_0, \{ s_0 \xrightarrow{a} s_1, s_1 \xrightarrow{b} s_2, \ldots \} \rangle \quad \xrightarrow{a}$

A reconfigurable graph M can evolve its configuration $\langle w_0, \alpha_0 \rangle$ by the rule

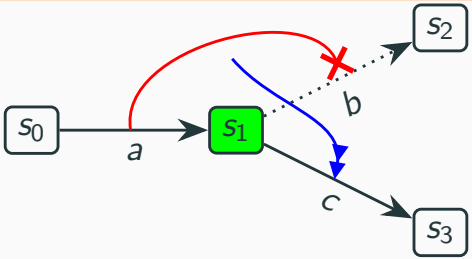
$$\frac{\exists e \in \alpha \quad \cdot \quad \overline{e} = w \xrightarrow{a} w' \quad \wedge \quad \alpha' = \big(\alpha \cup \mathsf{on}(e,\alpha)\big) \backslash \mathsf{off}(e,\alpha)}{\langle w, \alpha \rangle \quad \xrightarrow{a}_M \quad \langle w', \alpha' \rangle}$$



$$\langle s_0, \{s_0 \xrightarrow{a} s_1, s_1 \xrightarrow{b} s_2, \ldots\} \rangle \quad \xrightarrow{a} \quad \langle s_1, (\{s_0 \xrightarrow{a} s_1, s_1 \xrightarrow{b} s_2, \ldots\} \cup \{s_1 \xrightarrow{c} s_3\}) \setminus \{s_1 \xrightarrow{b} s_2\} \rangle$$

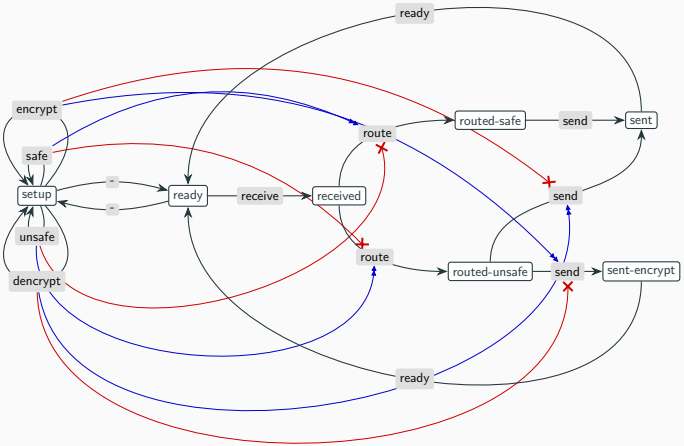
A reconfigurable graph M can evolve its configuration $\langle w_0, \alpha_0 \rangle$ by the rule

$$\frac{\exists e \in \alpha \quad \cdot \quad \overline{e} = w \xrightarrow{a} w' \quad \wedge \quad \alpha' = \big(\alpha \cup \mathsf{on}(e,\alpha)\big) \backslash \mathsf{off}(e,\alpha)}{\langle w, \alpha \rangle \quad \xrightarrow{a}_M \quad \langle w', \alpha' \rangle}$$



$$\langle s_0, \{s_0 \xrightarrow{a} s_1, s_1 \xrightarrow{b} s_2, \ldots\} \rangle \quad \xrightarrow{a} \quad \langle s_1, (\{s_0 \xrightarrow{a} s_1, s_1 \xrightarrow{c} s_3, \ldots\} \rangle$$

Applicable in dynamic software product lines?



Example adapted from Cordy, Classen, Heymans, Legay, Schobbens: *Model checking adaptive software with featured transition systems* (ASAS 2013).

Composition of models

Composition of reconfigurable graphs

Goal

Help building complex systems by composing simpler modules.

Composition of reconfigurable graphs

Goal

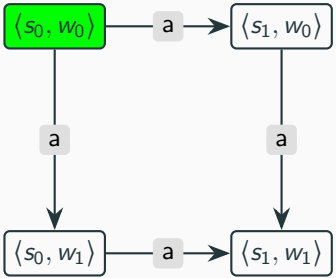
Help building complex systems by composing simpler modules.

Four products of reactive graphs

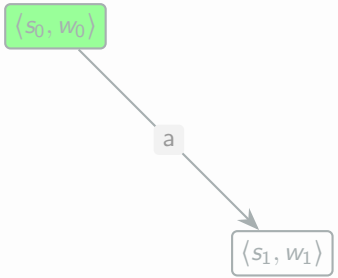
- asynchronous and synchronous
- with and without intrusive transitions

Traditional composition

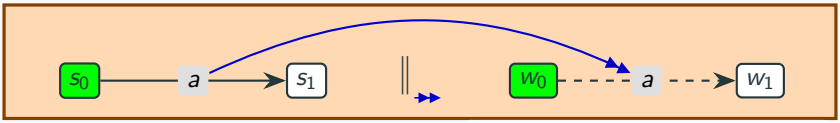
Asynchronous (interleaving)



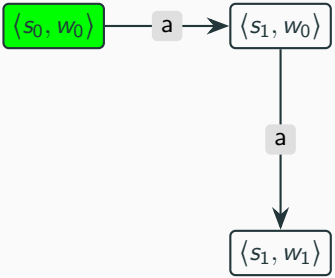
Synchronous (communicating)



Composition with intrusive transitions (example 1)

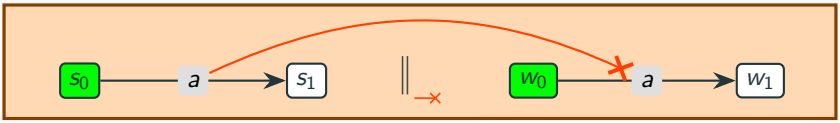


Asynchronous (interleaving)

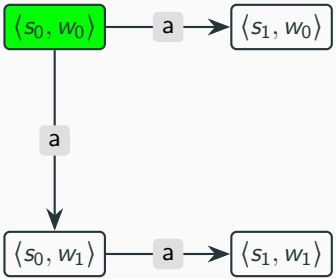


Synchronous (communicating)

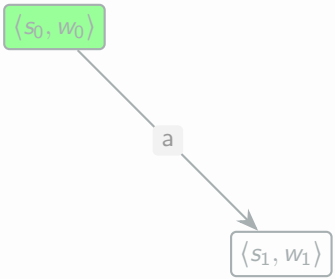
Composition with intrusive transitions (example 2)



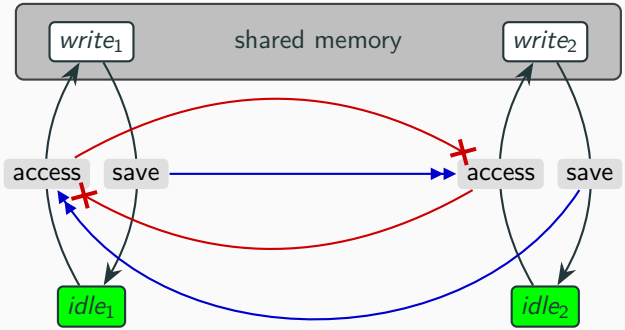
Asynchronous



Synchronous



Intrusive transitions – a different way to communicate



Products – formally

Asynchronous product with intrusive transitions

Given $c_1 = \langle s_1, \alpha_1 \rangle$ and $c_2 = \langle s_2, \alpha_2 \rangle$, the product, $c_1 \parallel_{\Gamma^{\oplus}, \Gamma^{\ominus}} c_2$ is defined as follows:

$$\alpha_i(\Gamma^{\oplus}, \Gamma^{\ominus}, e) \quad = \quad \big(\alpha_i \cup \mathsf{on}(e, \alpha_i) \cup \Gamma^{\oplus}(e)\big) \setminus \big(\mathsf{off}(e, \alpha_i) \cup \Gamma^{\ominus}(e)\big)$$

$$\frac{\exists\ e \in \alpha_1 \quad \cdot \quad \overline{e} = s_1 \xrightarrow{a} s_1' \quad \wedge \quad \alpha_1' = \big(\alpha_1 \cup \mathsf{on}(e, \alpha_1)\big) \backslash \mathsf{off}(e, \alpha_1) \quad \wedge \quad \alpha_2' = \alpha_2(\Gamma^{\oplus}, \Gamma^{\ominus}, e)}{\langle s_1, \alpha_1 \rangle \parallel_{\Gamma^{\oplus}, \Gamma^{\ominus}} \langle s_2, \alpha_2 \rangle \quad \xrightarrow{a} \quad \langle s_1', \alpha_1' \rangle \parallel_{\Gamma^{\oplus}, \Gamma^{\ominus}} \langle s_2, \alpha_2' \rangle}$$

$$\frac{\exists\ e \in \alpha_2 \quad \cdot \quad \overline{e} = s_2 \xrightarrow{a} s_2' \quad \wedge \quad \alpha_2' = \big(\alpha_2 \cup \mathsf{on}(e, \alpha_2)\big) \backslash \mathsf{off}(e, \alpha_2) \quad \wedge \quad \alpha_1' = \alpha_1(\Gamma^{\oplus}, \Gamma^{\ominus}, e)}{\langle s_1, \alpha_1 \rangle \parallel_{\Gamma^{\oplus}, \Gamma^{\ominus}} \langle s_2, \alpha_2 \rangle \quad \xrightarrow{a} \quad \langle s_1, \alpha_1' \rangle \parallel_{\Gamma^{\oplus}, \Gamma^{\ominus}} \langle s_2', \alpha_2' \rangle}$$

Variation of products of reconfigurable graphs – many initial states

A NETWORK of Multi-Actions Reactive Graph is a tuple $M = (W, Act, E, \rightarrow, \twoheadrightarrow, \multimap, \dot{\tau}, W_0, \alpha_0)$:

- W – states
- Act – actions
- E – edges
- $W_0 \subseteq W$ – (multi)set of initial states;
- $\alpha_0 \subseteq E$ – initially active edges

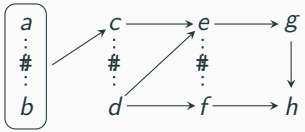
- $\rightarrow \subseteq W \times Act \times W$ – ground edges
- $\twoheadrightarrow \subseteq E \times E$ – activating edges
- $\multimap \subseteq E \times E$ – deactivating edges
- $\dot{\tau} : E \longrightarrow (\rightarrow \cup \twoheadrightarrow \cup \multimap)$ – internal details of edges

Evolving a configuration $\langle W_0, \alpha_0 \rangle$

$$\frac{\exists e \in \alpha \quad \cdot \quad \overline{e} = w \xrightarrow{a} w' \quad \wedge \quad \alpha' = \big(\alpha \cup \mathrm{on}(e,\alpha)\big) \backslash \mathrm{off}(e,\alpha)}{\langle W \cup \{w\}, \alpha \rangle \quad \xrightarrow{a} \quad \langle W \cup \{w'\}, \alpha' \rangle}$$

Event Structures

Bundle event structures by Example



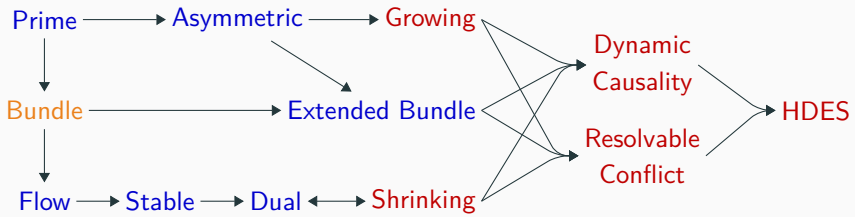
Valid traces

- a·c·e·g·h
- d·b·e·g·h
- d·f·a·h
- ...

- $a \cdots\#\cdots b$ – a excludes b
- $a \longrightarrow b$ – b must come after a
- $\{a, b\} \longrightarrow c$ – c must come after a and b, and $a \cdots\#\cdots b$
- all events must be used

Event structures

Landscape (partial): static and dynamic classes of event structures.

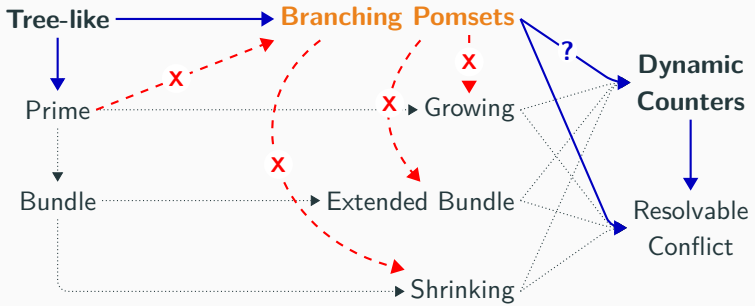


Arrows represent (strict) inclusion in terms of expressiveness

Used to represent different classes of Petri nets

Arbach, Karcher, Peters, Nestmann, Dynamic causality in event structures [FORTE 2015/LMCS 2018]

Landscape (partial): static and dynamic classes of event structures.

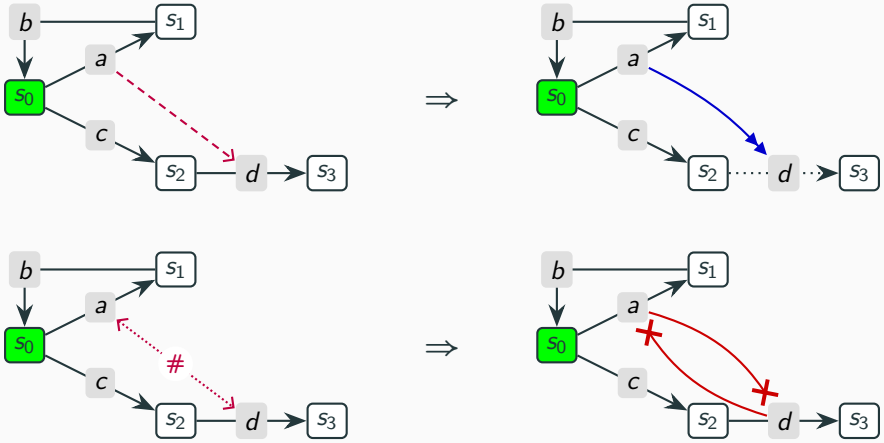


Used also to relate branching pomsets – *Edixhoven, Jongmans, Proença, Castellani, Branching pomsets: design, expressiveness and applications to choreographies* [JLAMP 2024]

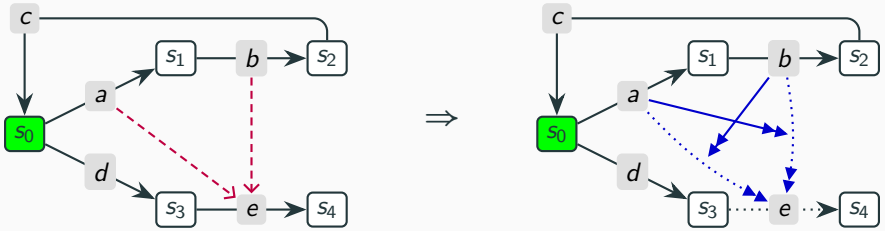
Dependencies as reconfigurations

Dependencies as reconfigurations

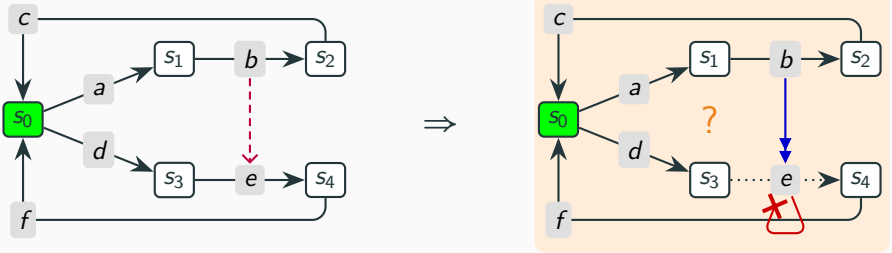
Dependencies as reconfigurations



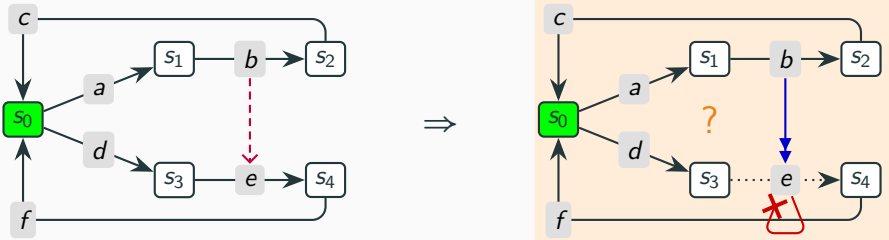
Too many reconfigurations?



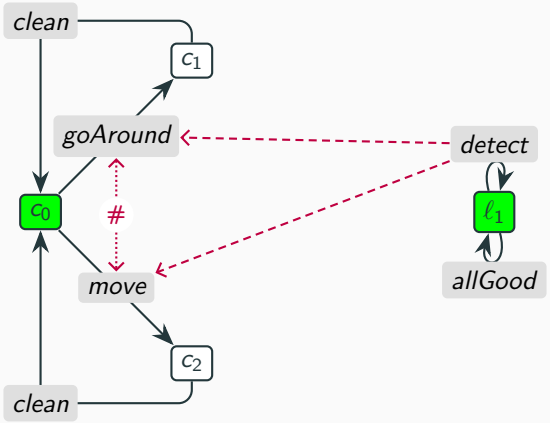
Loops and reset?



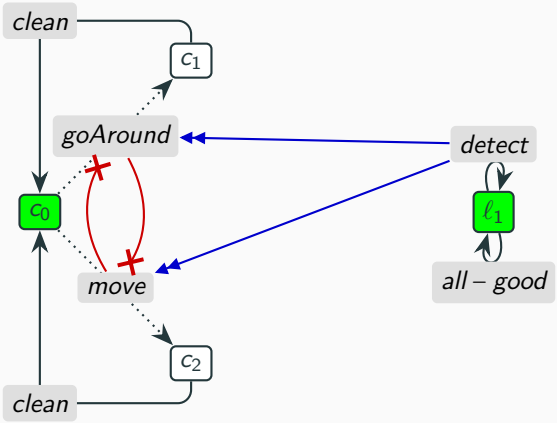
Loops and reset?



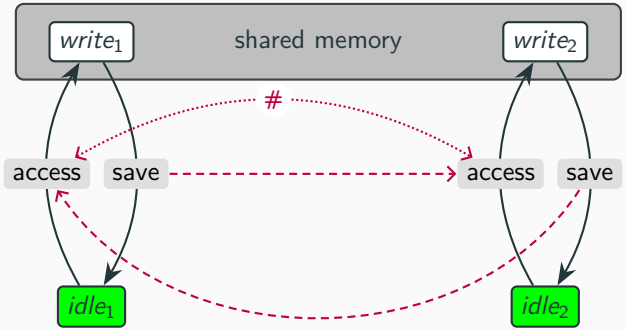
Composing with dependencies?



Composing with dependencies – what should be the interpretation?



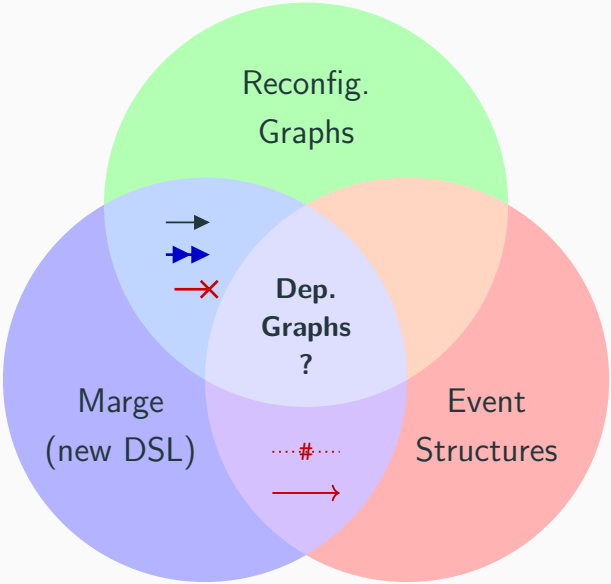
Composing with dependencies (revisiting example)



Other possible challenges and directions of work?

- When to reset a dependency/conflict?
- Different dependency notions (e.g., from different event structures)?
- Should dependability/conflict be a primitive in the model?
  (e.g., keep track of the number of missing activations)
- How compositional are these operators?
- Semantics for reconfigurable graphs (or variation) with Petri nets?

Wrap up – towards dependable graphs in Marge



"...I was told that every good presentation must have a Venn diagram" [Einar, Lima, ICTAC 2023]