Theoretical Foundations of the UML

Lecture 16: A Realisation Algorithm for Local Choice MSGs

Joost-Pieter Katoen

Lehrstuhl für Informatik 2
Software Modeling and Verification Group

moves.rwth-aachen.de/teaching/ws-1718/fuml/

17. Dezember 2017

RWTH AACHEN UNIVERSITY

Outline

RWTH AACHEN UNIVERSITY

Overview

RWTH AACHEN UNIVERSITY

Regular expressions vs. CFMs

Definition (Finitely generated)

Set of MSCs $\mathcal{M} \subseteq \mathbb{M}$ is finitely generated if there is a finite set of MSCs $\widehat{\mathcal{M}} \subseteq \mathbb{M}$ such that $\mathcal{M} \subseteq \widehat{\mathcal{M}}^*$.

Theorem [Morin 2002]

Let $\mathcal{M}$ be finitely generated. Then:

$\mathcal{M}$ is realisable

iff

there exists a star-connected regular expression α with $\mathcal{L}(\alpha) = \mathcal{M}$.

RWTH AACHEN UNIVERSITY

Some remaining questions

- How do we obtain a CFM realising an MSG algorithmically?
  - in particular, for non-local choice MSGs

This algorithm exploits synchronisation messages

- recall that weak CFMs do not involve synchronisation messages

RWTH AACHEN UNIVERSITY

Today's topics

Today's lecture

An algorithm to realise local-choice MSGs using CFM with synchronisation messages.

Results:

1. An algorithm that generates a CFM from local-choice MSG.

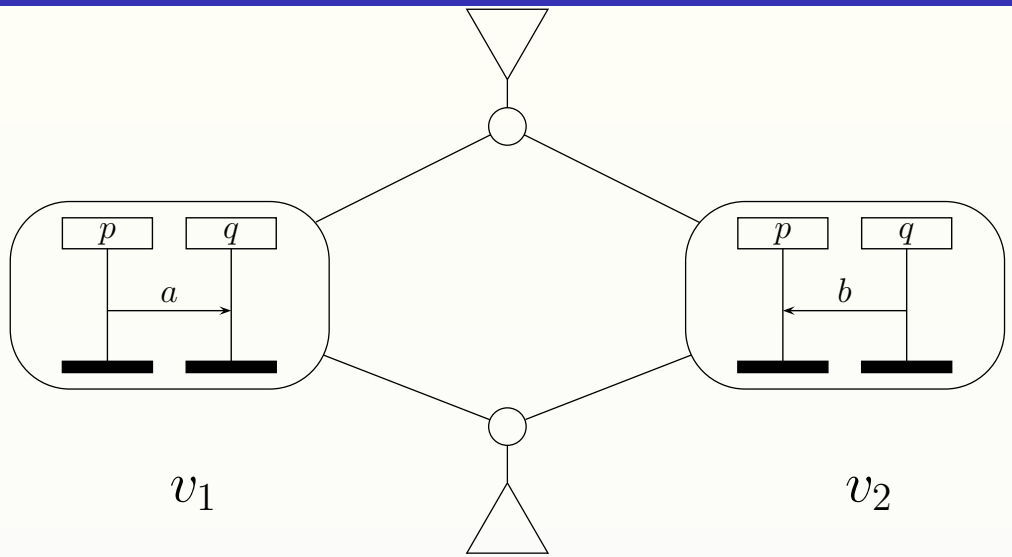
RWTH AACHEN UNIVERSITY

Overview

RWTH AACHEN UNIVERSITY

Non-local choice

G:



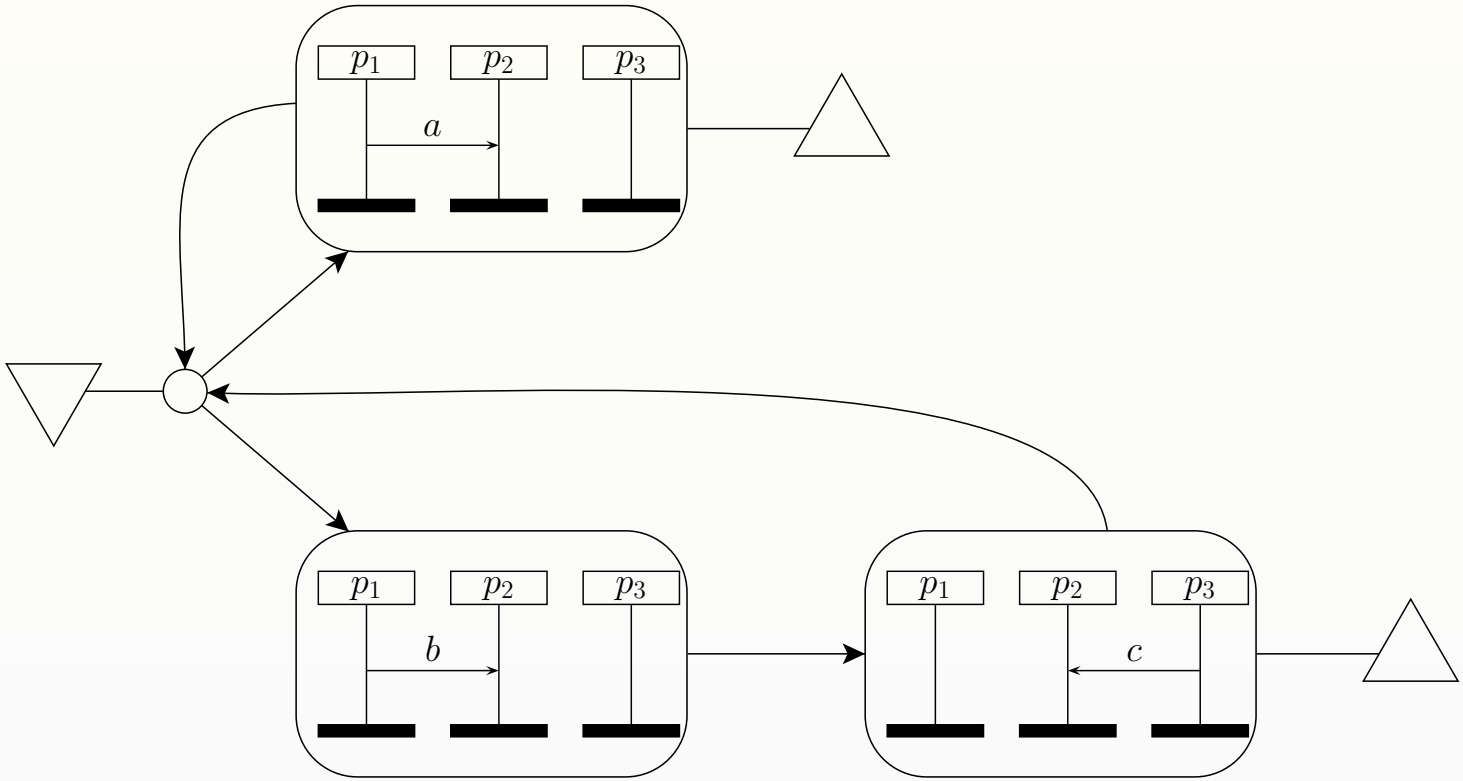
Inconsistency if process p behaves according to vertex v_1 and process q behaves according to vertex v_2

$\Longrightarrow$ realisation by a CFM may yield a deadlock

Problem:

Subsequent behavior in G is determined by distinct processes. When several processes independently decide to initiate behavior, they might start executing different successor MSCs (= vertices). This is called a non-local choice.

A (more involved) non-local choice



Problem:

Inconsistency if p_1 decides to send a and p_3 decides to send c.
Which branch to take in the initial vertex?

Definition (Minimal event)

Let $(E, \preceq)$ be a poset. Event $e \in E$ is a minimal event wrt. $\preceq$ if $\neg(\exists e' \neq e.\ e' \preceq e)$.

Intuition: there is no event that has to happen before e happens.
That is to say: the occurrence of e does not depend on any other event.

Definition (Partial order of a path)

For finite path $\pi = v_1 \dots v_n$ in MSG G, let $<_{M(\pi)}$ be the partial order of the MSC $M(\pi) = \lambda(v_1) \bullet \dots \bullet \lambda(v_n)$.

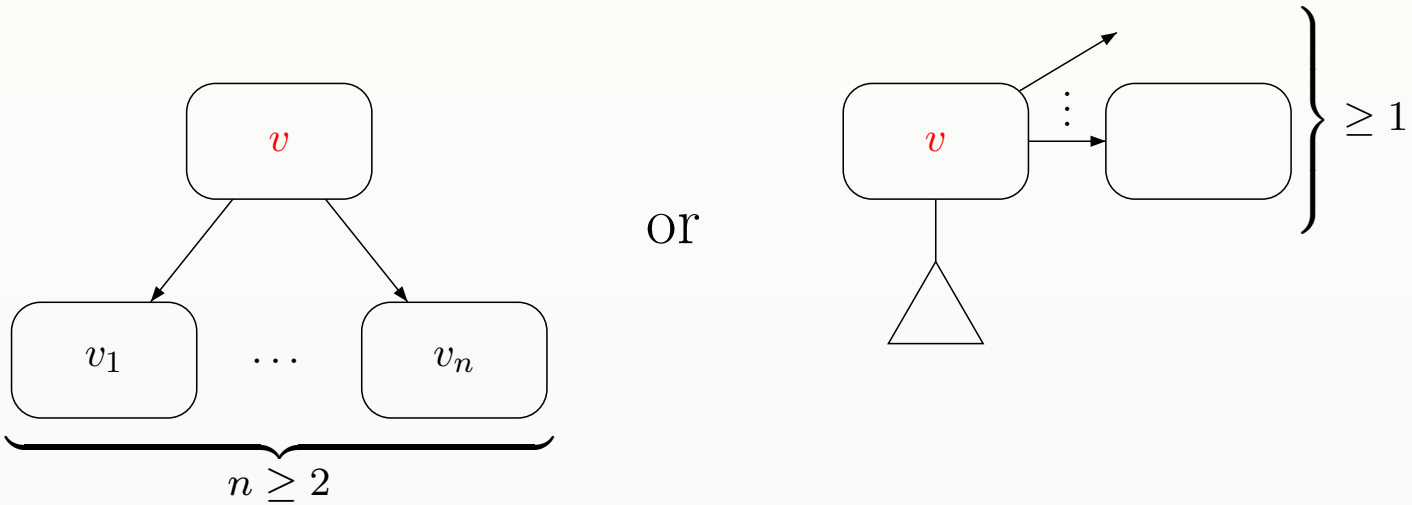
Let $\min(\pi)$ be the set of minimal events wrt. $<_{M(\pi)}$ along finite path π.

RWTH AACHEN UNIVERSITY

Branching vertices

A branching vertex in MSG G either has at least two successors, or is a final vertex with at least one successor.

Pictorially, vertex v is branching if either:



Without loss of generality we assume that branching final vertices do not occur. They can be always be removed at the expense of copying such vertices.

Local choice property

Definition (Local choice)

Let MSG $G = (V, \rightarrow, v_0, F, \lambda)$. MSG G is local choice if for every branching vertex $v \in V$ it holds:

$$\exists \text{process } p.\ \left(\forall \pi \in \text{Paths}(v).\ |\min(\pi')| = 1 \wedge \min(\pi') \subseteq E_p\right)$$

where for $\pi = v v_1 v_2 \dots v_n$ we have $\pi' = v_1 v_2 \dots v_n$.

Intuition:

There is a single process that initiates behavior along every path from the branching vertex v. This process decides how to proceed. In a realisation by a CFM, it can inform the other processes how to proceed.

Local choice or not?

Deciding whether MSG G is local choice or not is in P. (Exercise.)

Overview

RWTH AACHEN UNIVERSITY

Local choice MSGs

An example local-choice MSG on black board.

RWTH AACHEN UNIVERSITY

Realising local choice (C)MSGs

Theorem [Genest *et al.*, 2005]

Any local-choice MSG G is safely realisable by a CFM with additional synchronisation data (which is of size linear in G).

Proof

As MSG G is local choice, at every branch v of G there is a unique process, $p(v)$, say, such that on every path from v the unique minimal event occur at $p(v)$. Then:

1. Process $p(v)$ determines the successor vertex of v.
2. Process $p(v)$ informs all other processes about its decision by adding synchronisation data to the exchanged messages.
3. Synchronisation data is the path (in G) from v to the next branching vertex along the direction chosen by $p(v)$.

Maximal non-branching paths

Definition (Maximal non-branching paths)

For MSG $G = (V, \rightarrow, v_0, F, \lambda)$, let $nbp : V \rightarrow V^*$ be defined by:

$$nbp(v) = \begin{cases} v & \text{if } v \in F \text{ or } v \text{ is a branching vertex} \\ v_1 \dots v_n & \text{otherwise} \end{cases}$$

where $v_1 \dots v_n \in V^*$ is a maximal path (i.e., a path that cannot be prolonged) satisfying:

1. $v_i = v$ for some i, $0 < i \leqslant n$, and
2. $v_n \in F$ or is a branching vertex, and
3. $v_1 = v_0$ or is a direct successor of a branching vertex, and
4. $v_2, \dots, v_{n-1} \notin F$ and are all non-branching vertices

Intuition

$nbp(v)$ is the maximal non-branching path to which v belongs.

Structure of the CFM of local choice MSG G

Let MSG $G = (V, \rightarrow, v_0, F, \lambda)$ be local choice.

Define the CFM $\mathcal{A}_G = (((S_p, \Delta_p))_{p \in \mathcal{P}}, \mathbb{D}, s_{init}, F')$ with:

1. Local automaton $\mathcal{A}_p = (S_p, \Delta_p)$ as defined on next slides

2. $\mathbb{D} = \{ npb(v) \mid v \in V \}$
   synchronisation data = maximal non-branching paths in G

3. $s_{init} = \{ (v_0, \varnothing) \}^n$ where $n = |\mathcal{P}|$
   each local automaton $\mathcal{A}_p$ starts in initial state $(v_0, \varnothing)$, i.e., in initial vertex v_0 while no events of p have been performed

4. $\overline{s} \in F'$ iff for all $p \in \mathcal{P}$, local state $\overline{s}[p] = (v, E)$ with $E \subseteq E_p$ and:
   1. $v \in F$ and E contains a maximal event wrt. $<_p$ in MSC $\lambda(v)$, or
   2. $v \notin F$ and $\pi = v \ldots w$ is a path in G with $w \in F$ and E contains a maximal event wrt. $<_p$ in MSC $\lambda(\pi)$.

RWTH AACHEN UNIVERSITY

State space of local automaton $\mathcal{A}_p$

- $S_p = V \times E_p$ such that for any $s = (v, E) \in S_p$:

$$\forall e, e' \in \lambda(v).\ \big(e <_p e' \text{ and } e' \in E \quad \text{implies} \quad e \in E\big)$$

  that is, E is downward-closed with respect to $<_p$ in MSC $\lambda(v)$

- Intuition: a state (v, E) means that process p is currently in vertex v of MSG G and has already performed the events E of $\lambda(v)$

- Initial state of $\mathcal{A}_p$ is $(v_0, \varnothing)$

RWTH AACHEN UNIVERSITY

Transition relation of local automaton $\mathcal{A}_p$

- Executing events within a vertex of the MSG G:

$$\frac{e \in E_p \cap \lambda(v) \text{ and } e \notin E}{(v, E) \xrightarrow{l(e), nbp(v)}_p (v, E \cup \{ e \})}$$

  Note: since $E \cup \{e\}$ is downward-closed wrt. $<_p$, e is enabled

- Taking an edge (possibly a self-loop) of the MSG G:

$$\frac{\begin{array}{c} E = E_p \cap \lambda(v) \text{ and } e \in E_p \cap \lambda(w) \text{ and} \\ vu_0 \ldots u_n w \in V^* \text{ with } p \text{ not active in } u_0 \ldots u_n \end{array}}{(v, E) \xrightarrow{l(e), nbp(w)}_p (w, \{e\})}$$

  Note: vertex w is the first successor vertex of v on which p is active

RWTH AACHEN UNIVERSITY

Examples

A couple of examples on the black board.

RWTH AACHEN UNIVERSITY