Analysis and Implementation of MSC-Specifications

Carsten Kern

Lehrstuhl Informatik 2
Software Modeling and Verification
RWTH Aachen

November 29, 2005
Outline

1. Introduction

2. CMSGs and their properties
   - Message Sequence Charts
   - Message Sequence Graphs

3. CMSG hierarchies
   - CMSG Properties
   - Property and Language Hierarchy

4. Implementability

5. Future Work
Presentation outline

1. Introduction

2. CMSGs and their properties
   - Message Sequence Charts
   - Message Sequence Graphs

3. CMSG hierarchies
   - CMSG Properties
   - Property and Language Hierarchy

4. Implementability

5. Future Work
Objects of interest: **Message Sequence Charts**

- standardized modeling language at high level of abstraction
- used for specification of communication protocols
- similar to UML sequence diagrams
Presentation outline

1. Introduction

2. CMSGs and their properties
   - Message Sequence Charts
   - Message Sequence Graphs

3. CMSG hierarchies
   - CMSG Properties
   - Property and Language Hierarchy

4. Implementability

5. Future Work
A CMSC Example (Antiblock System)
A CMSC is a tuple $M = \langle \mathcal{P}, E, t, m, < \rangle$

- $\mathcal{P}$: finite, non-empty set of processes
- $E = \biguplus_{p \in \mathcal{P}} E_p = S \uplus R$: set of events
  
  $(S$: sending and $R$: receiving events)
- $t$: event labeling function ($t : E \rightarrow \text{Act}$)
  
  (e.g.: $t(e) = p!q$ or $t(e') = p?q$)
- $m$: injective and partial matching function ($m : S \rightarrow R$)
  
  (not every sending event needs to have a corresponding receive event)
- $< \subseteq E \times E$: partial order on events

If $m$ is total and bijective we call $M$ an MSC
A CMSG Example (Alternating-Bit Protocol)
A CMSG Example (Alternating-Bit Protocol)
A CMSG Example (Alternating-Bit Protocol)
A CMSG Example (Alternating-Bit Protocol)
Definition: compositional MSGs (CMSGs)

A CMSG is a tuple \( G = \langle V, R, V^0, V^f, \lambda \rangle \)

- \( \langle V, R \rangle \): graph \((V \neq \emptyset, R \subseteq V \times V)\)
- \( V^0 \): non-empty set of start nodes \((V^0 \subseteq V)\)
- \( V^f \): set of end nodes \((V^f \subseteq V)\)
- \( \lambda \): node labeling function \((\lambda : V \rightarrow \text{MSC})\)

If \( \lambda \) maps to \( \text{MSC} \) \( G \) is called a Message Sequence Graph (MSG)
Presentation outline

1. Introduction
2. CMSGs and their properties
   - Message Sequence Charts
   - Message Sequence Graphs
3. CMSG hierarchies
   - CMSG Properties
   - Property and Language Hierarchy
4. Implementability
5. Future Work
What do we need CMSG properties for?

⇒ detecting classes of implementable (C)MSGs
CMSG Property: “Regularity”

Formal Definition:
A CMSG $G$ is called regular iff the communication graph of every cycle is strongly connected.

Intuition of property: Regularity
Regularity means that for each sent message an acknowledgement of the destination process can be received.
Example: CMSG-Property "Regularity"

Communication graph of the loop:
CMSG Properties: “Local-Choice” and “Locality”

Intuition of property: **Local Choice**
In every choice node only one process may decide on the progress of the system (*strong* local-choice).

Intuition of property: **Locality**
The local property assures the local-choice property for every node in the graph.

Observation:
Every local CMSG is also local-choice.
Example: CMSG-Property “Local-Choice”
Property and Language Hierarchy

1: strong local
2: strong local-choice
3: weak local
4: weak local-choice
5: locally cooperative
6: regular
7: globally cooperative

Compressed Language Hierarchy

1, 2, 3, 4
Presentation outline

1. Introduction

2. CMSGs and their properties
   - Message Sequence Charts
   - Message Sequence Graphs

3. CMSG hierarchies
   - CMSG Properties
   - Property and Language Hierarchy

4. Implementability

5. Future Work
Implementation of CMSGs

### Communicating Finite-State Machines $\mathcal{A} = \langle (A_p)_{p \in \mathcal{P}}, F \rangle$

- $(A_p)_{p \in \mathcal{P}}$: set of local automata
- $F \subseteq \prod_{p \in \mathcal{P}} S_p$: set of global final states

### Local Automata $A_p = \langle S_p, s_p, \rightarrow_p \rangle$

- $S_p$: final state set
- $s_p \in S_p$: starting state
- $\rightarrow \subseteq S_p \times Act \times S_p$: transition relation
A CFM Example (Producer-Consumer)
A CFM Example (Producer-Consumer)
A CFM Example (Producer-Consumer)
A CFM Example (Producer-Consumer)
A CFM Example (Producer-Consumer)
A CFM Example (Producer-Consumer)
A CFM Example (Producer-Consumer)
A CFM Example (Producer-Consumer)
A CFM Example (Producer-Consumer)
### Theorem

Every local-choice CMSG is implementable (without deadlocks).
Presentation outline

1. Introduction

2. CMSGs and their properties
   - Message Sequence Charts
   - Message Sequence Graphs

3. CMSG hierarchies
   - CMSG Properties
   - Property and Language Hierarchy

4. Implementability

5. Future Work
Drawbacks of MSCs and MSGs

Problems
- no distinction between what **must** and **may** happen

Possible Approach
- using **Life Sequence Charts (LSCs)** (by Damm and Harel) to distinguish between
  - mandatory, optional and illegal behavior
An LSC example (Antiblock System)

Diagram:
- Display
- Control Unit
- Wheel Sensor
- Break

- Wheel blocks
- Adjust break power
- Inform driver
Outlook

Extension of MSCs and MSGs to simulate quantitative behavior

Thus modeling:
- uncertain events
- unreliable channels
  (integrating probability into message transmission (MSCs))
- probabilistic branching
  (integrating probability into node branching (MSGs))

Approach: extending Life Sequence Charts to cope with quantitative behavior
Goals:

- **extensions** need to be formally defined
- **extensions** need to be equipped with a reasonable **semantics**
- **properties** (like regularity etc.) will be defined and classified
- relation to existing probabilistic extensions of statecharts will be checked