MSCan
A Tool for Analyzing MSC Specifications

Benedikt Bollig[1] Carsten Kern[2] Markus Schlütter[2]
Volker Stolz[2]

[1] Laboratoire Spécification et Vérification

[2] Lehrstuhl Informatik 2

TACAS 2006, March 30

Outline

1. Introduction to Message Sequence Charts
   - Message Sequence Charts (MSCs)
   - Message Sequence Graphs (MSGs)
2. Implementability
   - The Formal Model of CFMs
   - Examples
3. Tool Presentation
4. Summary and Outlook

Presentation outline

1. Introduction to Message Sequence Charts
   - Message Sequence Charts (MSCs)
   - Message Sequence Graphs (MSGs)
2. Implementability
   - The Formal Model of CFMs
   - Examples
3. Tool Presentation
4. Summary and Outlook

Message Sequence Charts (MSCs)

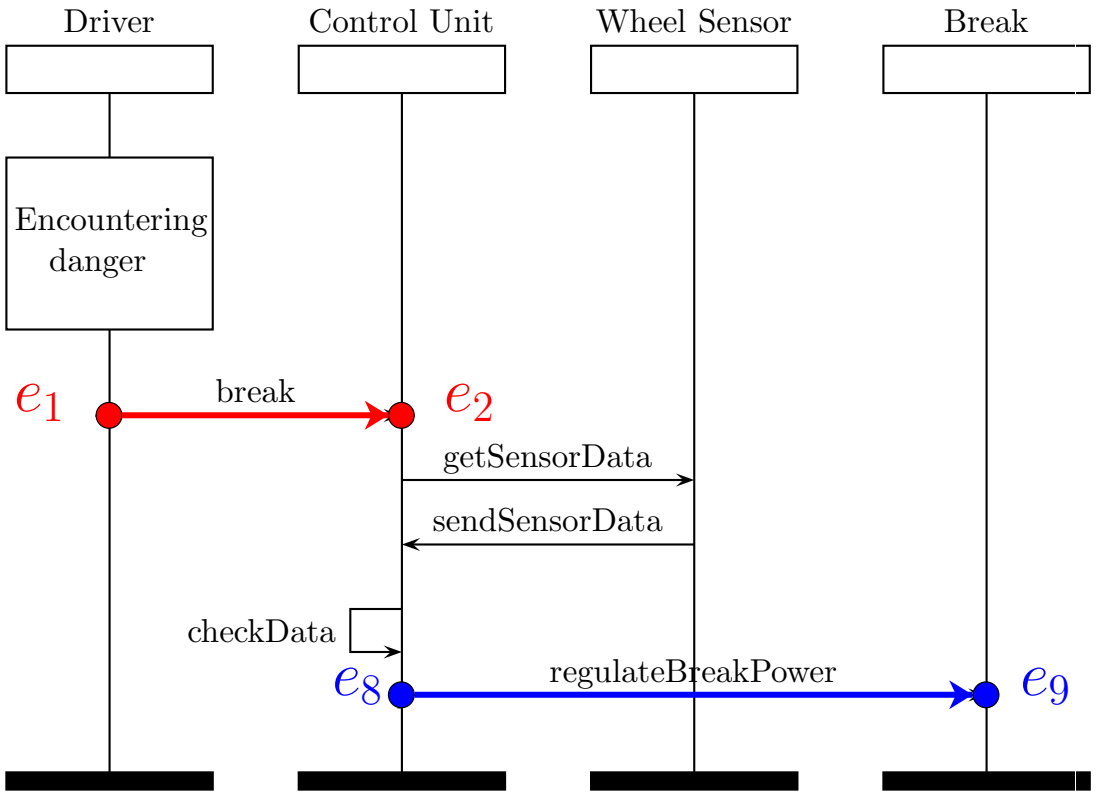
What are Message Sequence Charts

- Visual specification language
- Often used in the design of communication protocols
- Standardized: ITU-TS Recommendation Z.120
- Similar to UML's sequence diagrams

What does our tool provide

- MSCan tests MSC specifications for implementability

An MSC Example



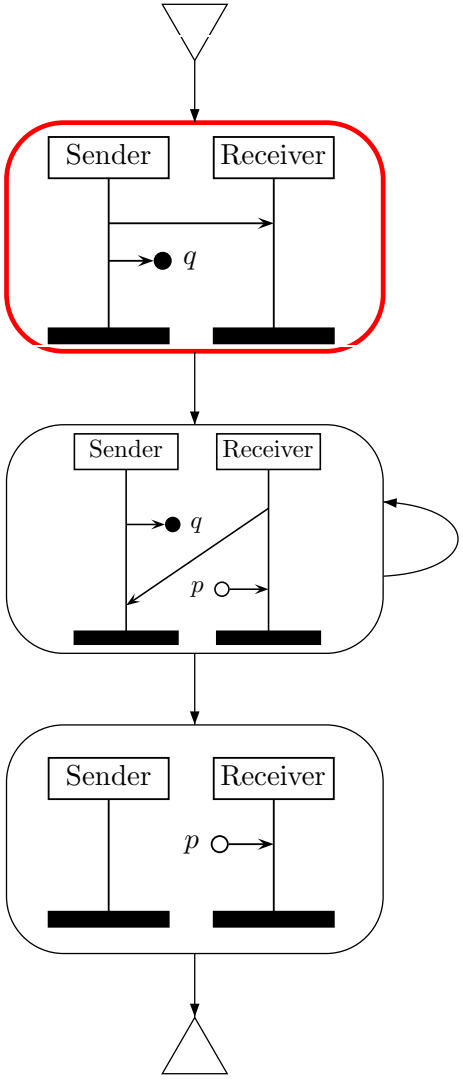
Definition: Compositional MSC (CMSC) [Gunter & Muscholl & Peled TACAS'01]

A CMSC is a tuple $M = \langle \mathcal{P}, E, t, m, \leq \rangle$

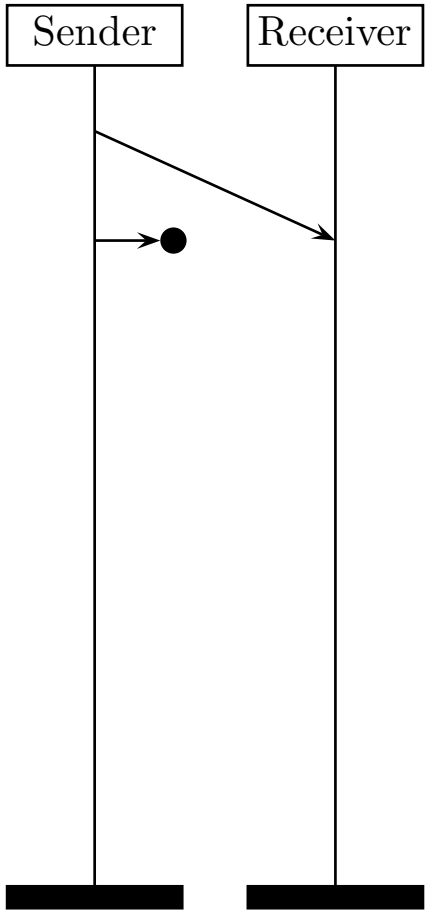
- $\mathcal{P}$ finite set of processes
- $E = \biguplus_{p \in \mathcal{P}} E_p = Send \uplus Rec$ set of events
- $t : E \to \{p!q,\ q?p \mid p, q \in \mathcal{P}\}$ labeling function
- $m : S \rightarrowtail R$ partial injective matching function (respects FIFO)
- $\leq \subseteq E \times E$ partial order on events
- E_p is totally ordered by $\leq$

If m is total and bijective, we call M an MSC

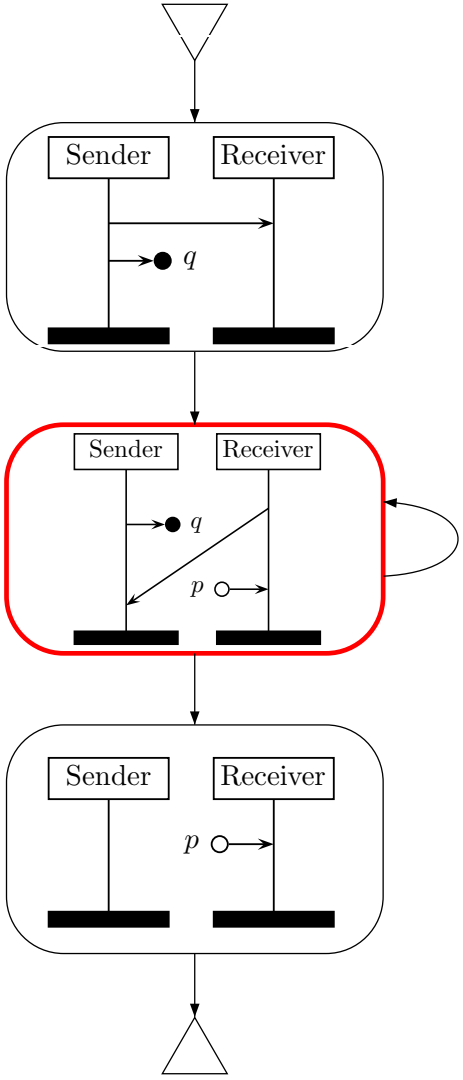
A CMSG Example (Alternating-Bit Protocol)



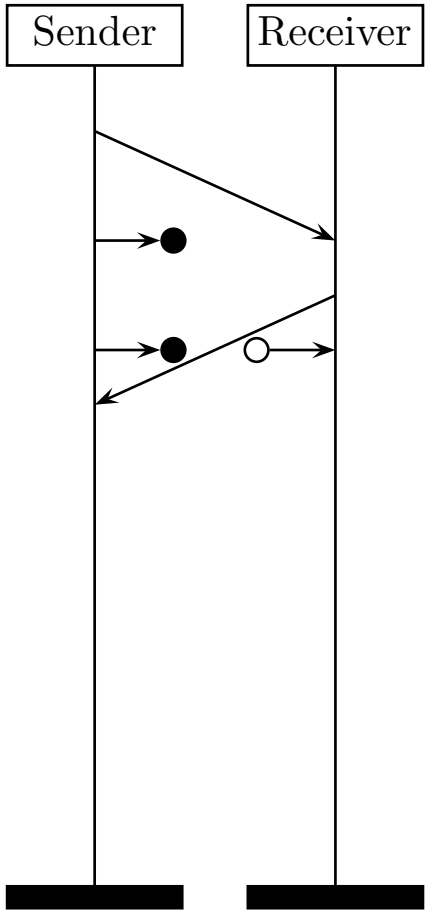
An execution



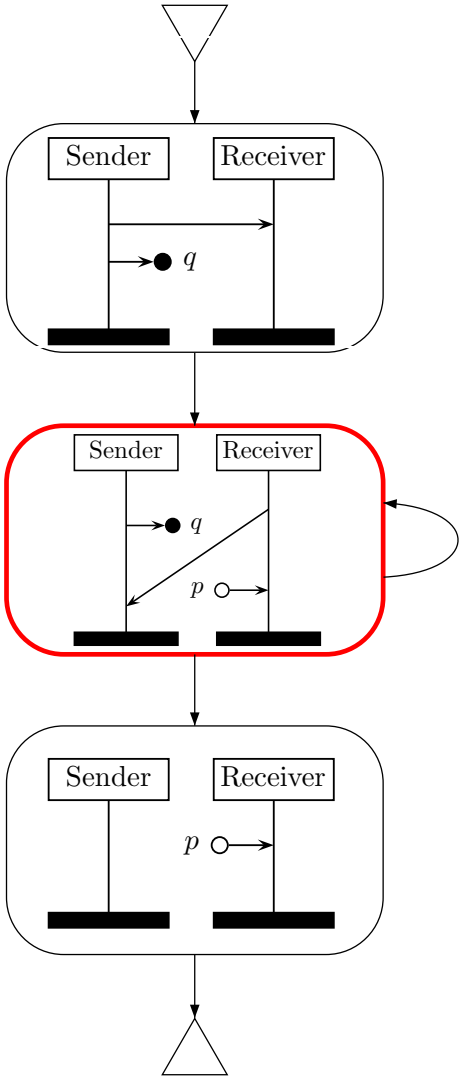
A CMSG Example (Alternating-Bit Protocol)



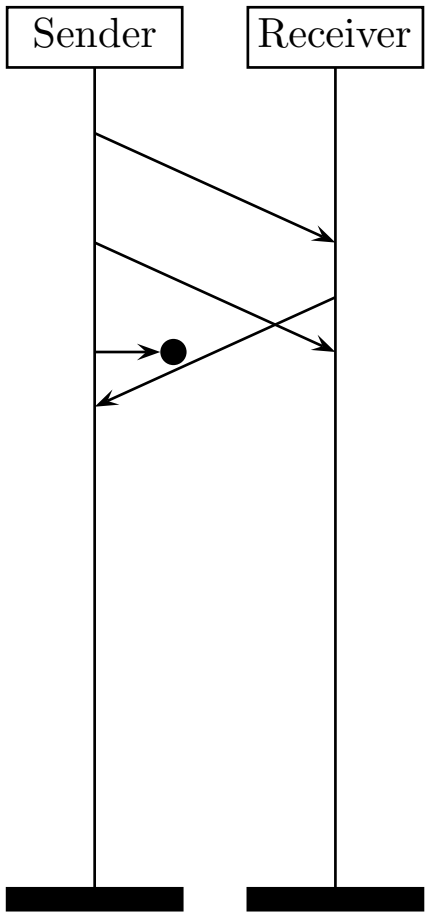
An execution



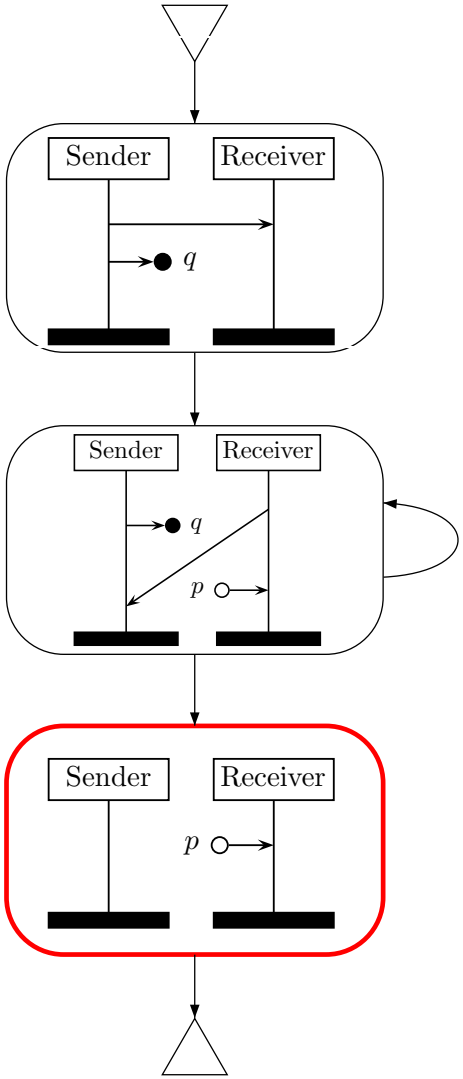
A CMSG Example (Alternating-Bit Protocol)



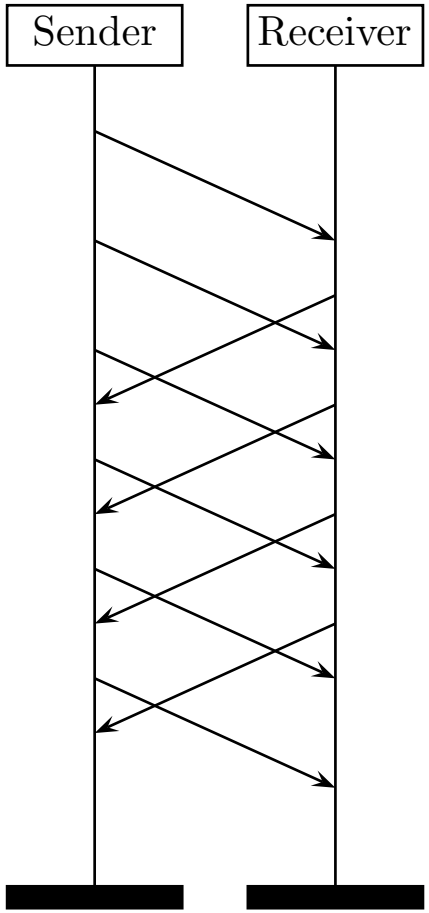
An execution



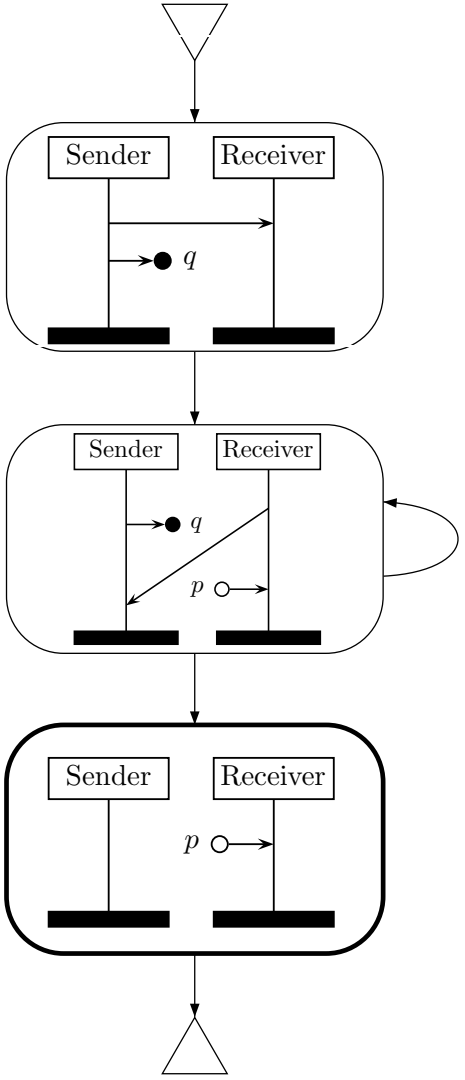
A CMSG Example (Alternating-Bit Protocol)



An execution

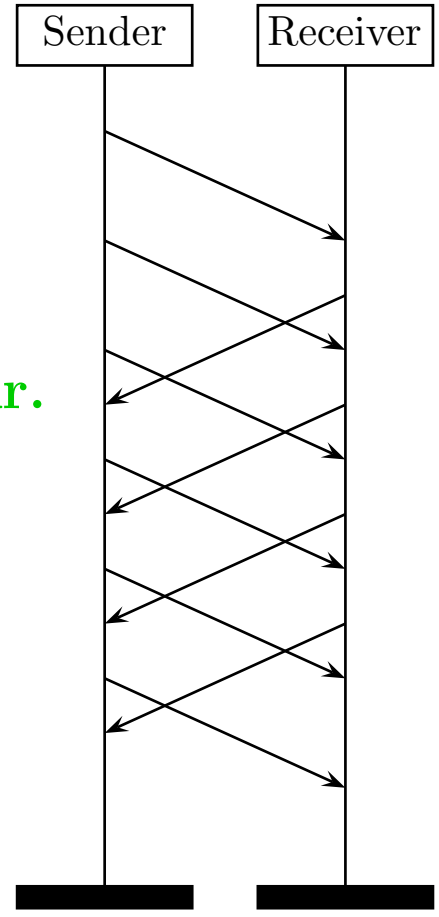


A CMSG Example (Alternating-Bit Protocol)



is safe and regular.

An execution



Definition: Compositional MSGs (CMSGs) [Gunter & Muscholl & Peled TACAS'01]

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

- $\langle V, R \rangle$: finite graph ($R \subseteq V \times V$)
- $V^0 \subseteq V$: set of start nodes
- $V^f \subseteq V$: set of end nodes
- $\lambda : V \to \mathbb{CMSC}$: node labeling function

If λ maps to $\mathbb{MSC}$, then G is called an MSG.

Presentation outline

1. Introduction to Message Sequence Charts
   - Message Sequence Charts (MSCs)
   - Message Sequence Graphs (MSGs)
2. Implementability
   - The Formal Model of CFMs
   - Examples
3. Tool Presentation
4. Summary and Outlook

Definition: Communicating Finite-State Machine (CFM)

Let $\mathcal{P}$ be a given finite set.

A CFM $\mathcal{A} = \langle (\mathcal{A}_p)_{p \in \mathcal{P}}, F \rangle$

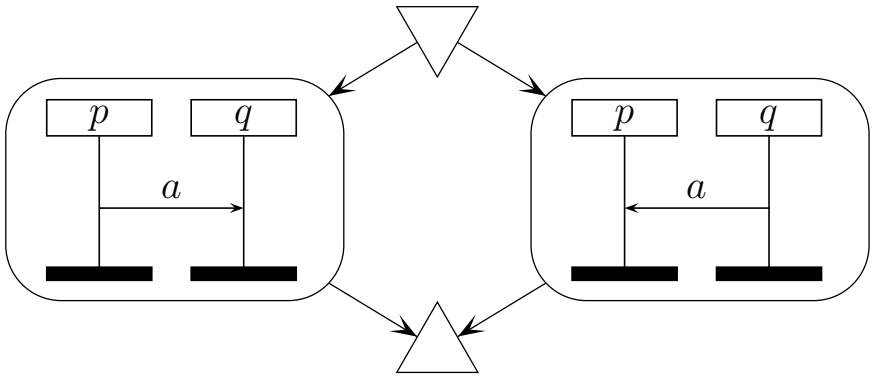
- $(\mathcal{A}_p)_{p \in \mathcal{P}}$: collection of local automata
- $F \subseteq \prod_{p \in \mathcal{P}} S_p$: set of global final states

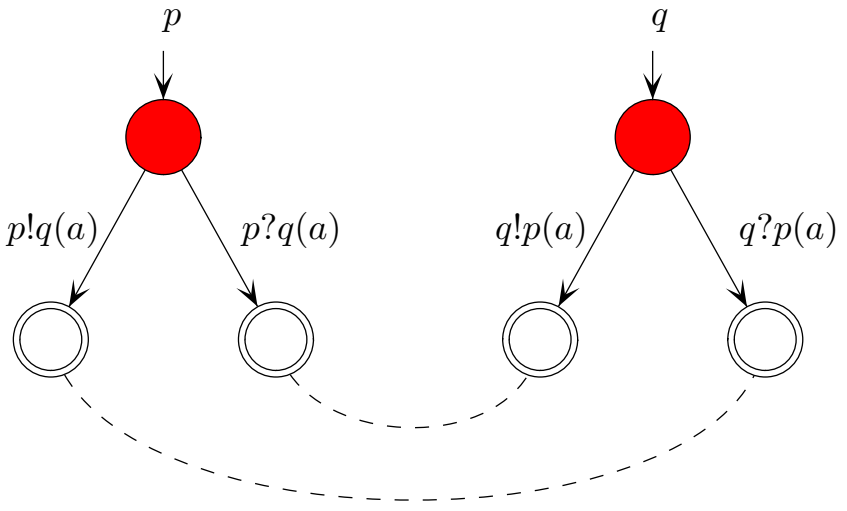
$\mathcal{A}_p = \langle S_p, s_p, \longrightarrow_p \rangle$

finite automaton

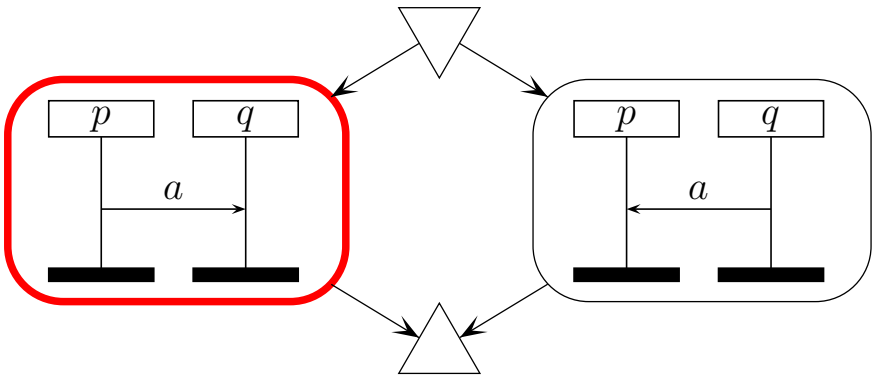
- S_p: finite set of local states
- $s_p \in S_p$: local start state
- $\longrightarrow_p \subseteq S_p \times \{p!q,\ p?q \mid q \in \mathcal{P}\} \times S_p$: local transition relation

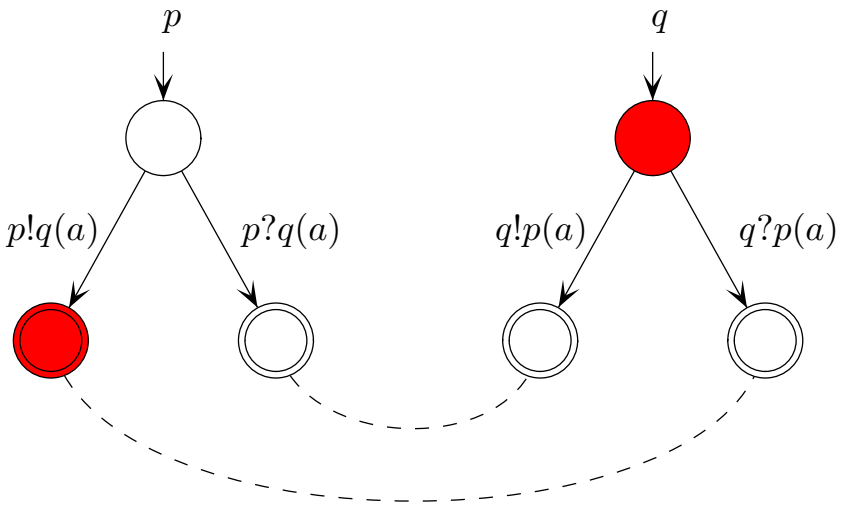
An easy example



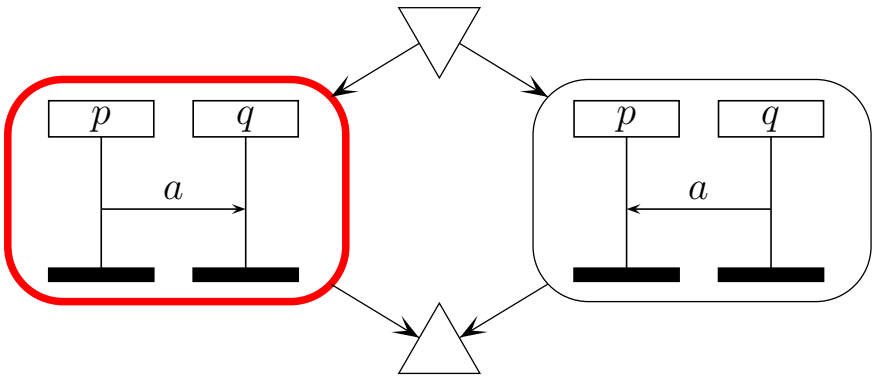


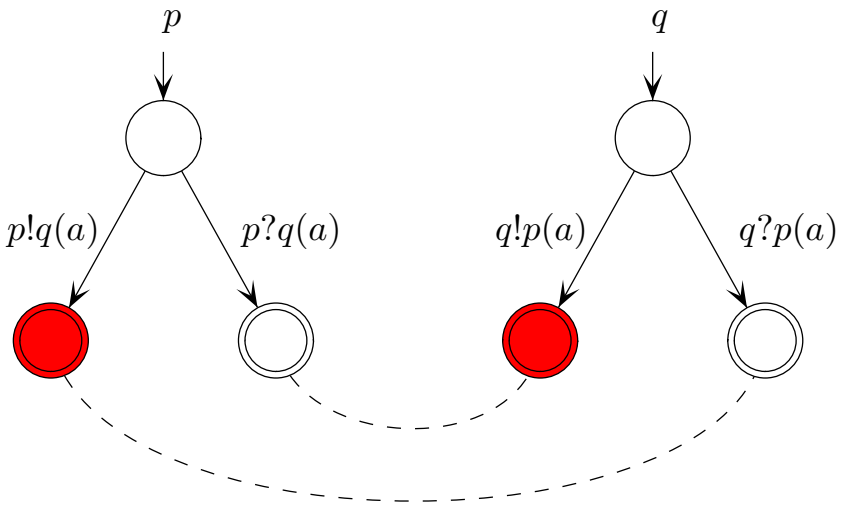
An easy example



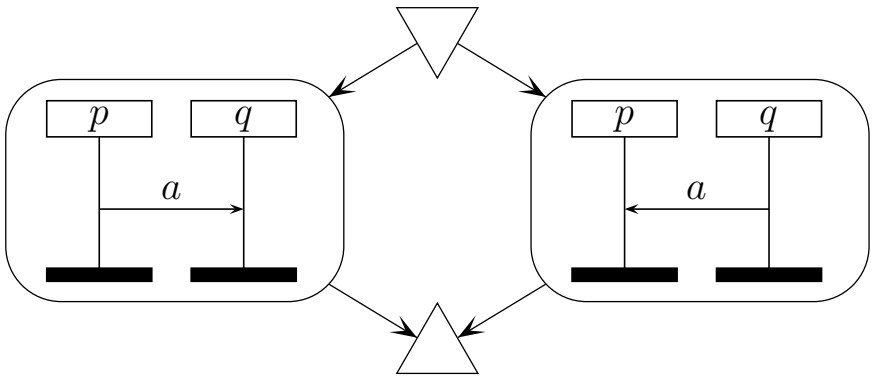


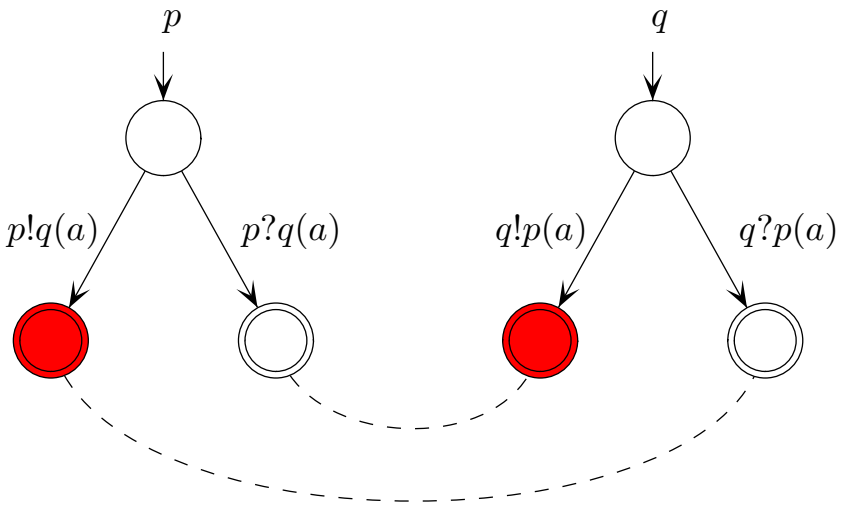
An easy example





An easy example





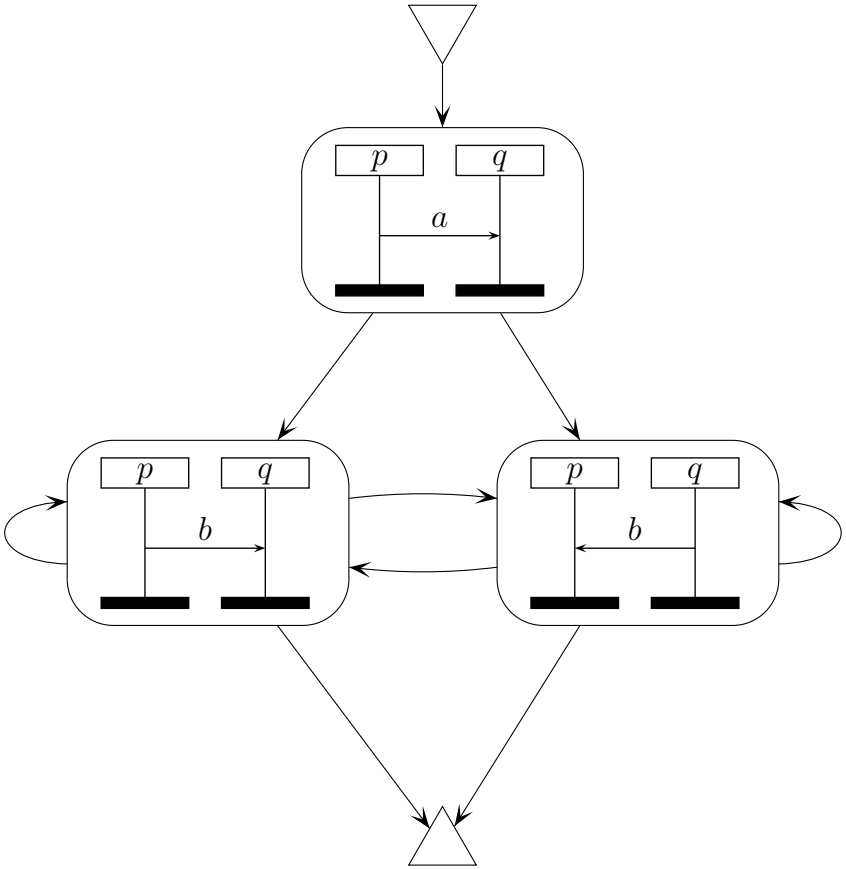
The problem we face:

Deadlocks

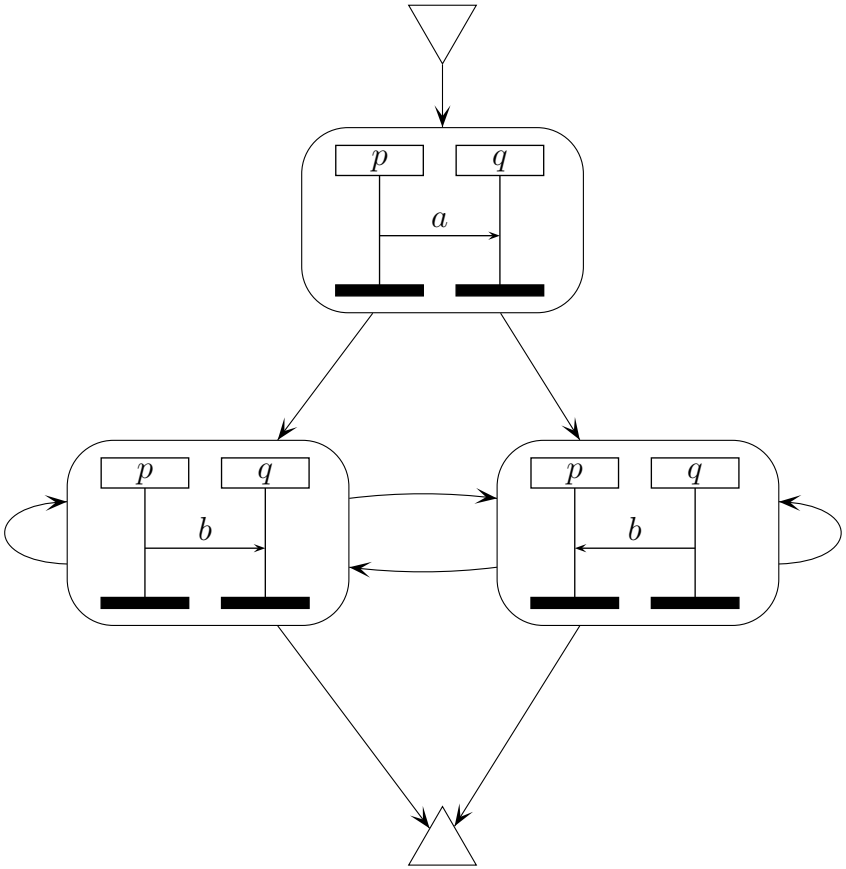
The solution:

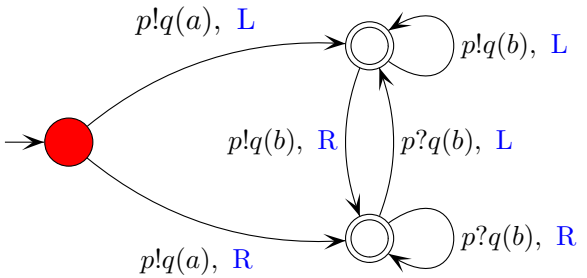
Local-choice MSGs

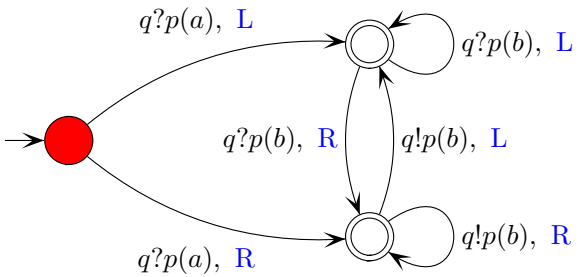
Local-choice CMSG [Genest TACAS'05; Ben-Abdallah & Leue TACAS'97]



Local-choice CMSG [Genest TACAS'05; Ben-Abdallah & Leue TACAS'97]

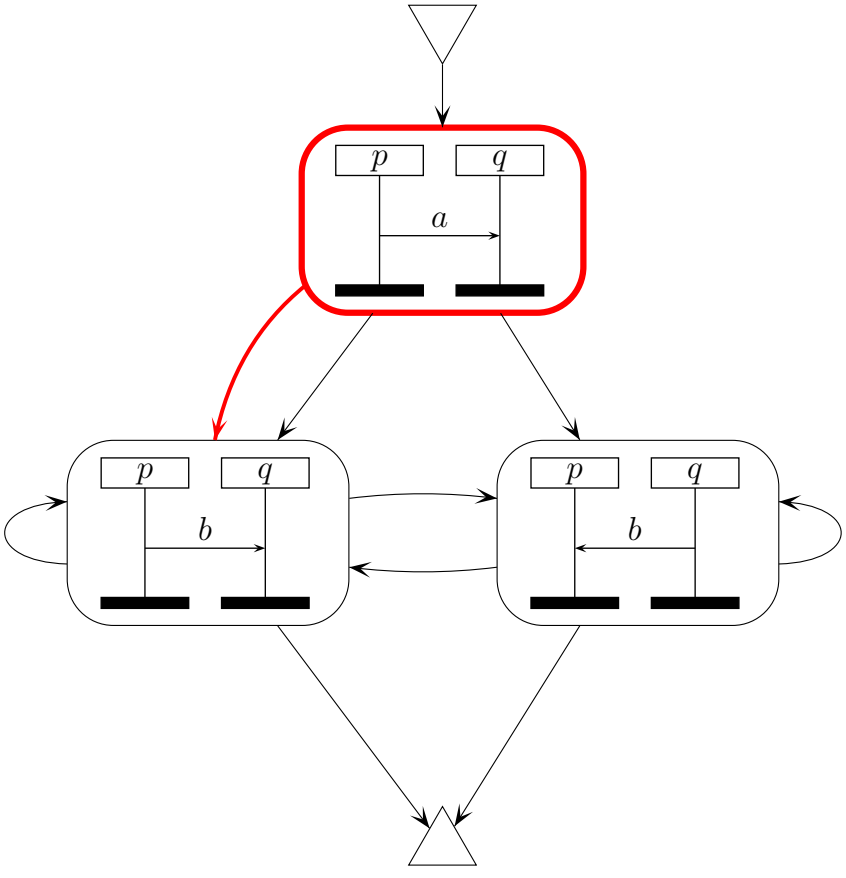


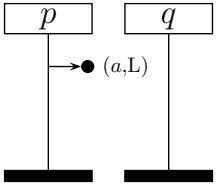


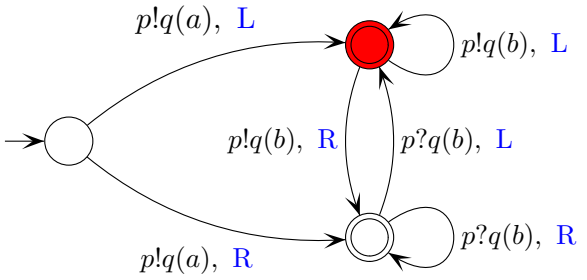


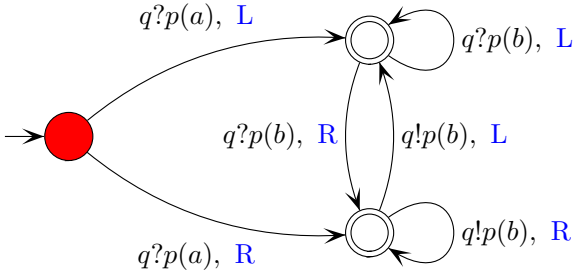
$p \rightarrow q$:
$q \rightarrow p$:

Local-choice CMSG [Genest TACAS'05; Ben-Abdallah & Leue TACAS'97]



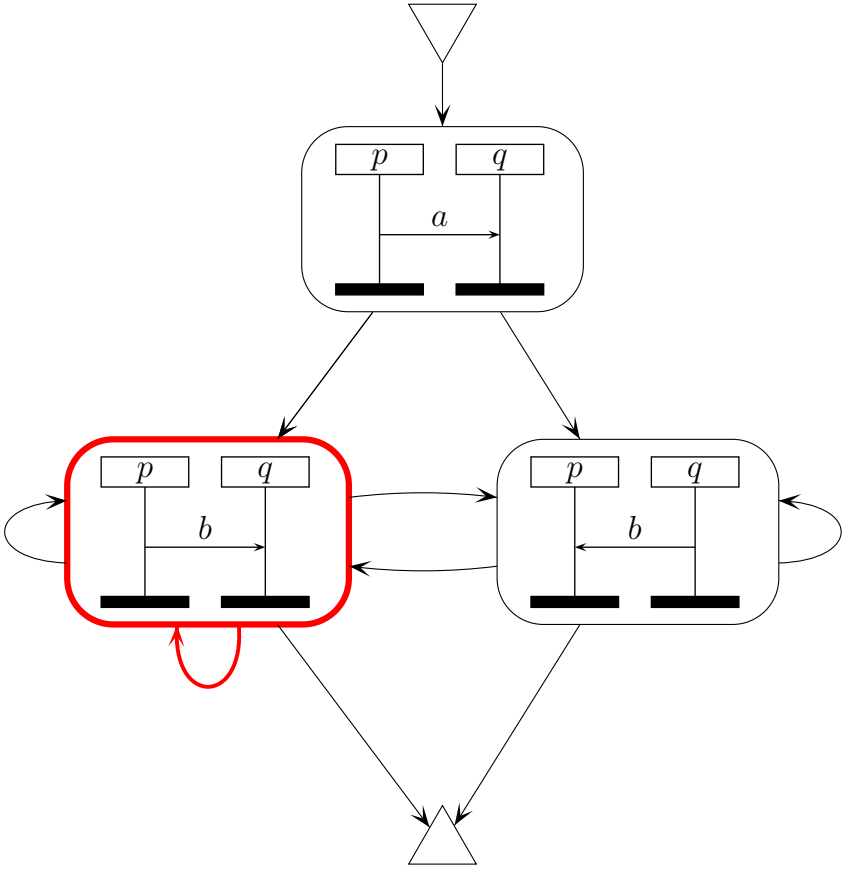


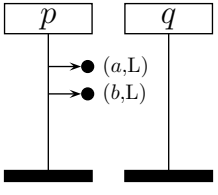


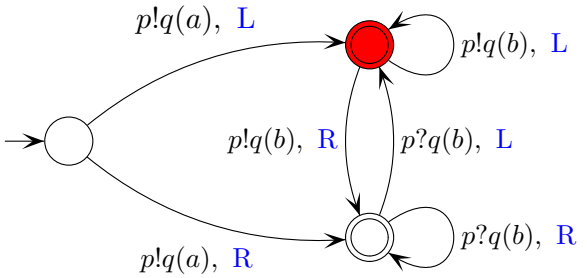


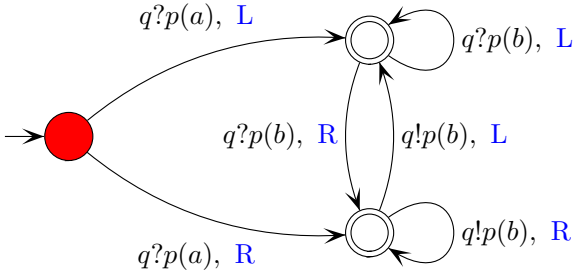
$p \to q : (a,\mathrm{L})$
$q \to p :$

Local-choice CMSG [Genest TACAS'05; Ben-Abdallah & Leue TACAS'97]



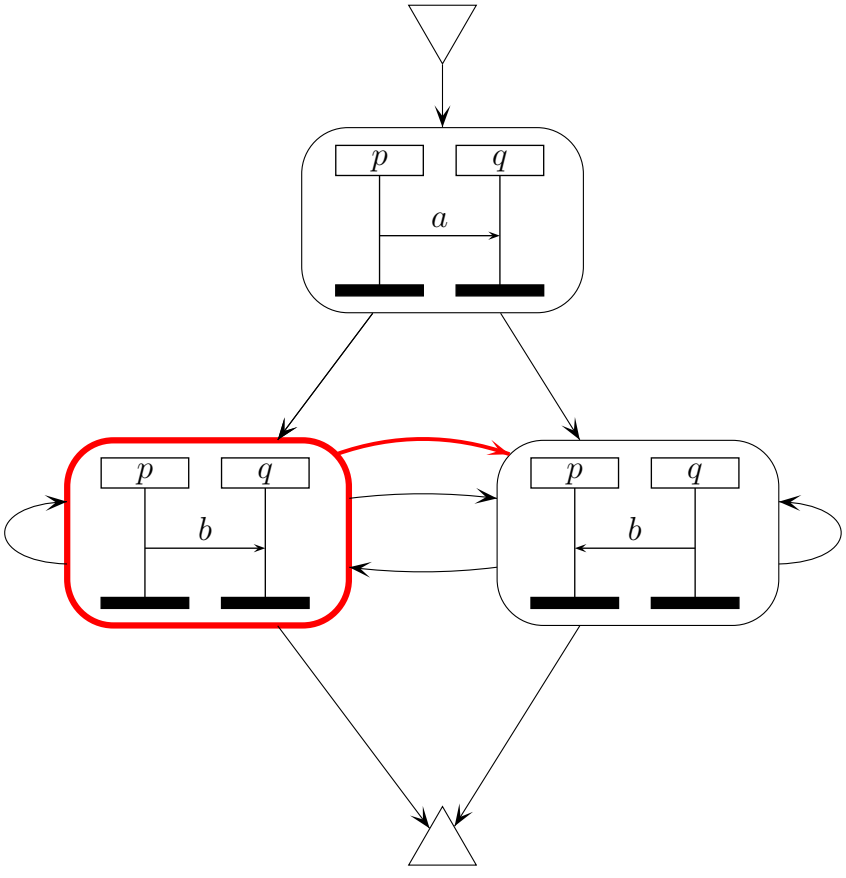


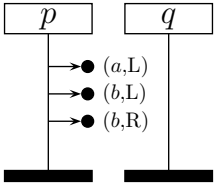


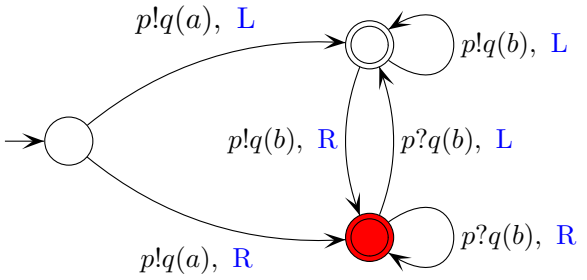


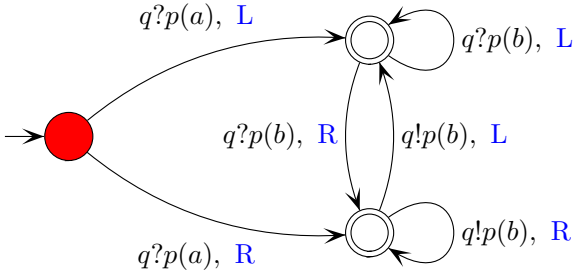
$p \to q : (a,\mathrm{L})\ (b,\mathrm{L})$
$q \to p :$

Local-choice CMSG [Genest TACAS'05; Ben-Abdallah & Leue TACAS'97]



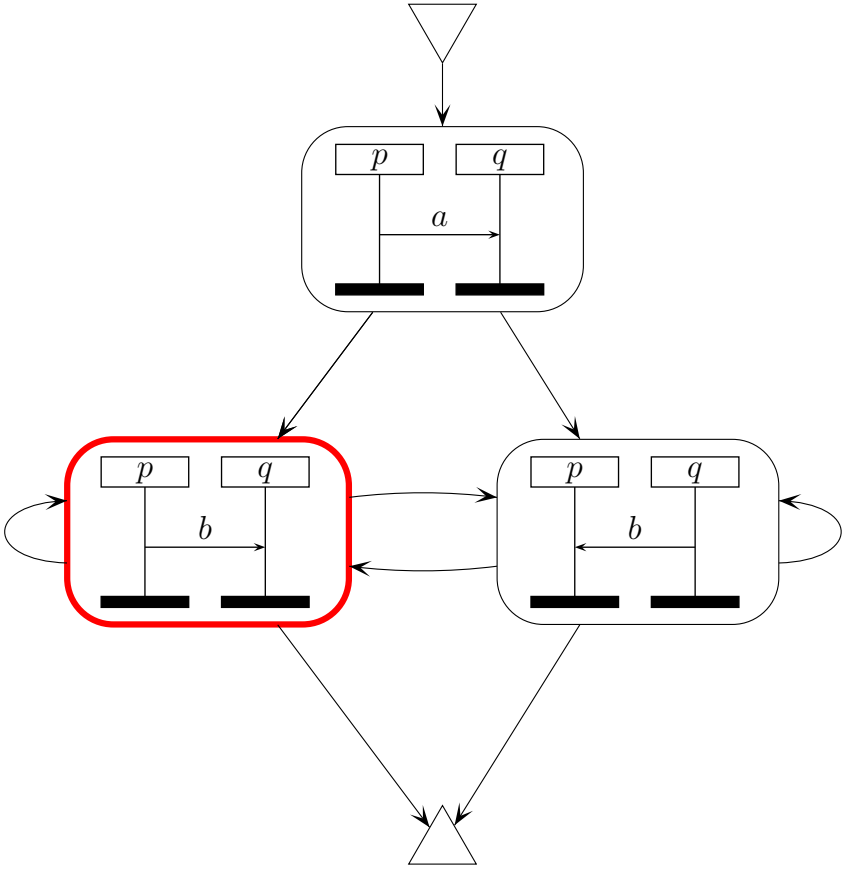


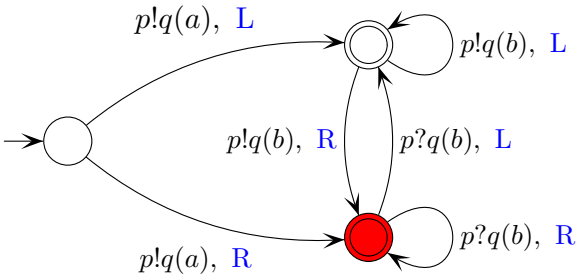


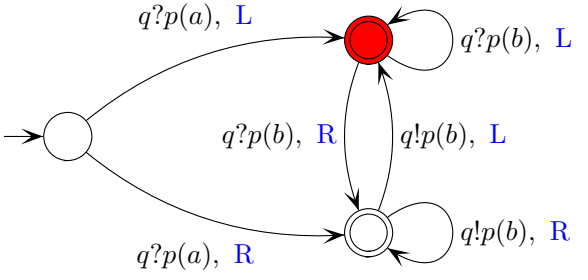


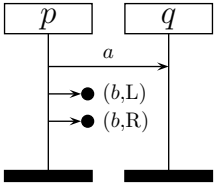
$p \to q : (a,\mathrm{L})\ (b,\mathrm{L})\ (b,\mathrm{R})$
$q \to p :$

Local-choice CMSG [Genest TACAS'05; Ben-Abdallah & Leue TACAS'97]



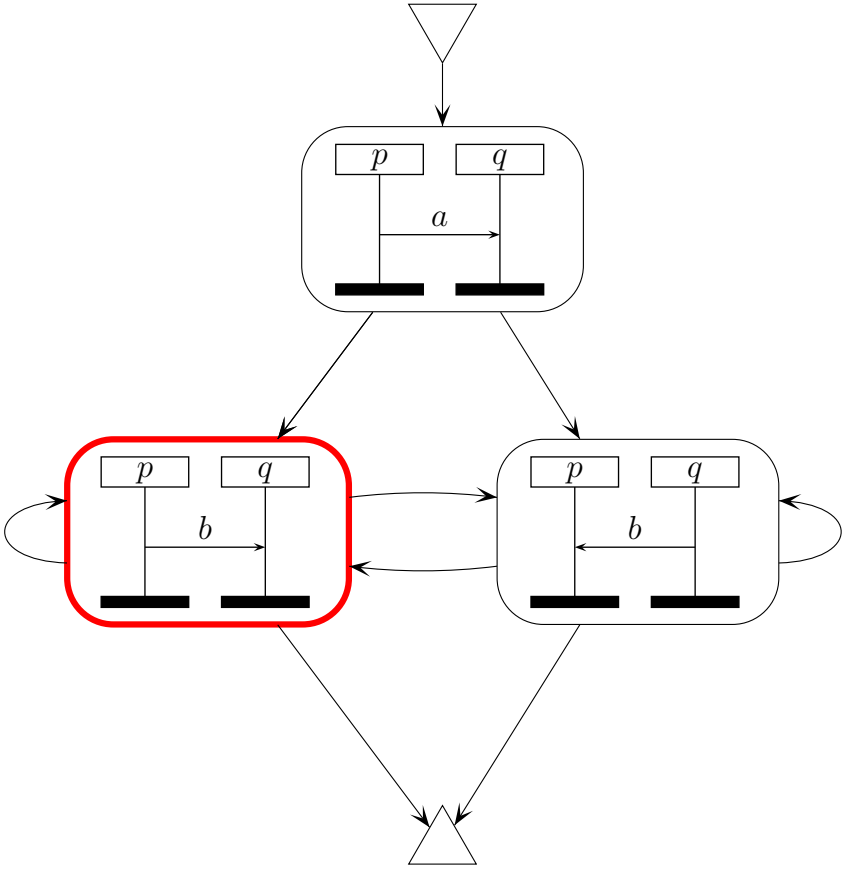


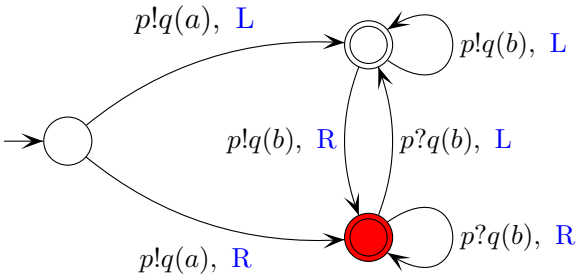


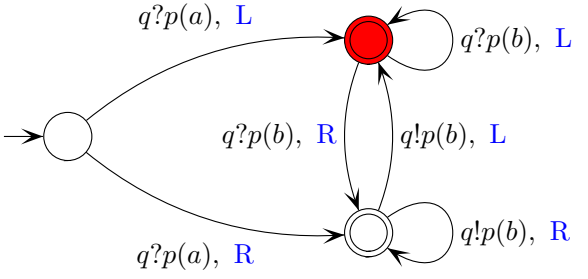


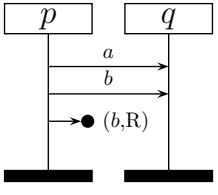
$p \to q : (b,\mathrm{L})\ (b,\mathrm{R})$
$q \to p :$

Local-choice CMSG [Genest TACAS'05; Ben-Abdallah & Leue TACAS'97]



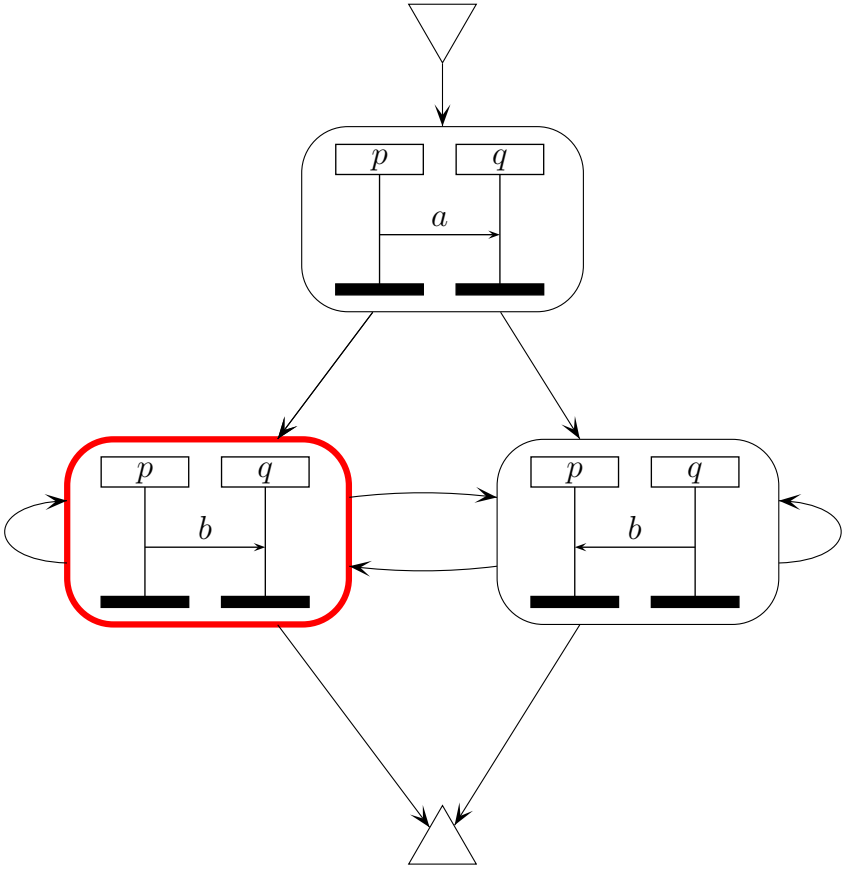


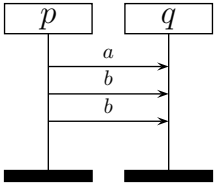


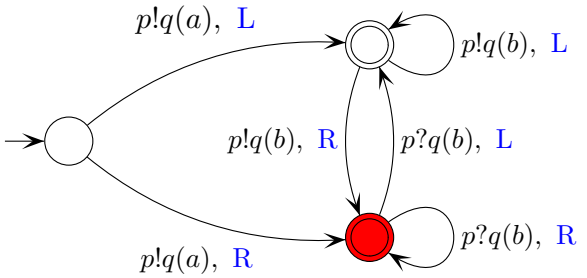


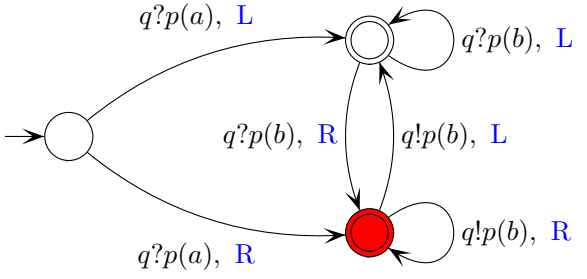
$p \to q : (b,\mathrm{R})$
$q \to p :$

Local-choice CMSG [Genest TACAS'05; Ben-Abdallah & Leue TACAS'97]





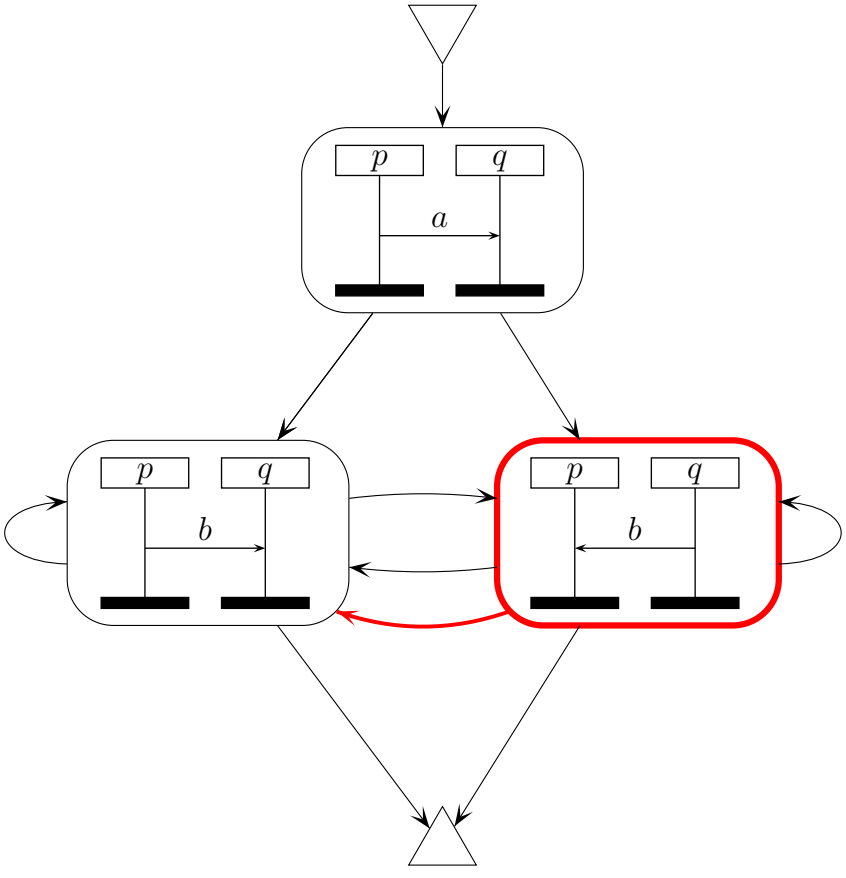


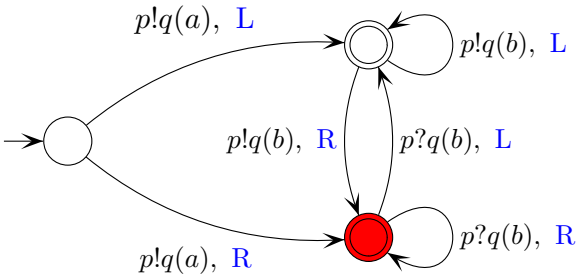


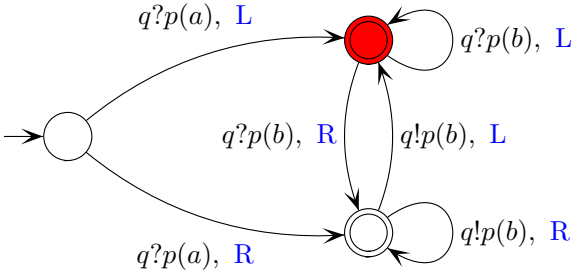
$p \to q$:

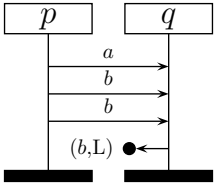
$q \to p$:

Local-choice CMSG [Genest TACAS'05; Ben-Abdallah & Leue TACAS'97]



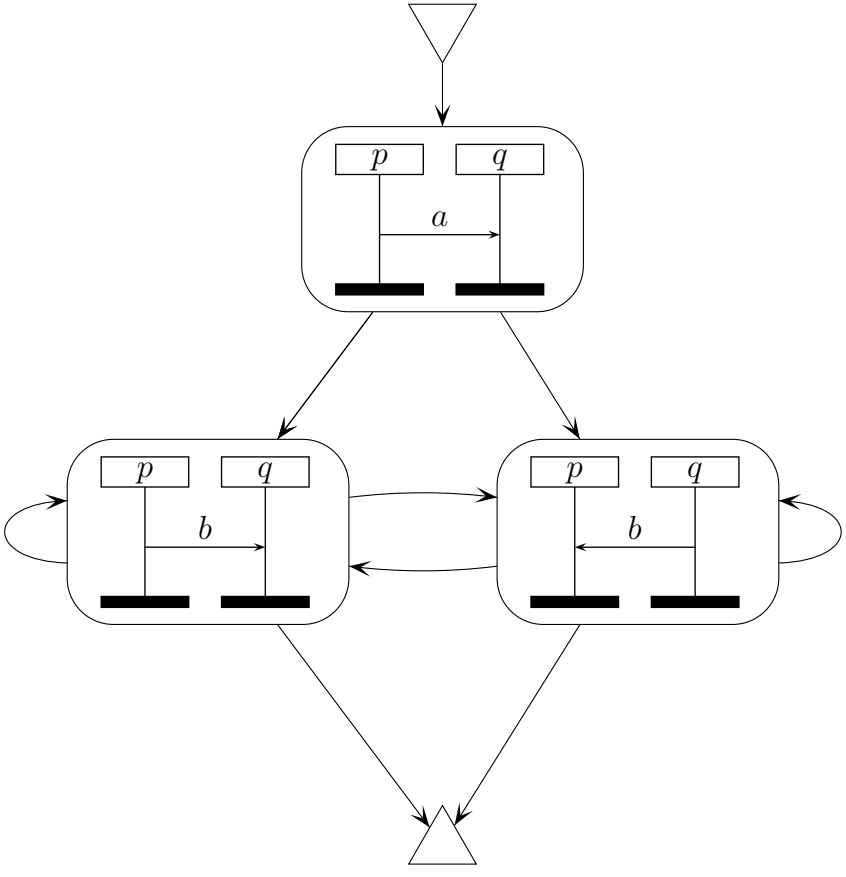


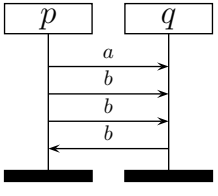


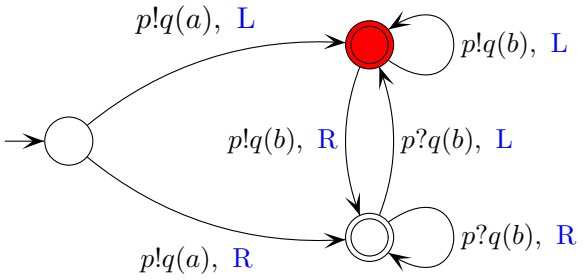


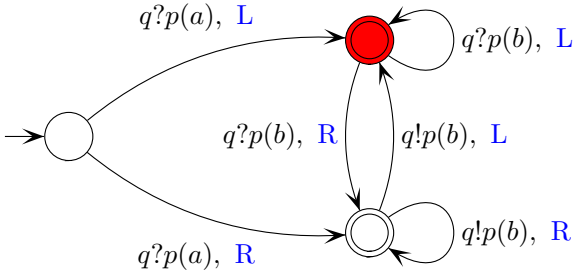
$p \rightarrow q$:
$q \rightarrow p$: (b,L)

Local-choice CMSG [Genest TACAS'05; Ben-Abdallah & Leue TACAS'97]





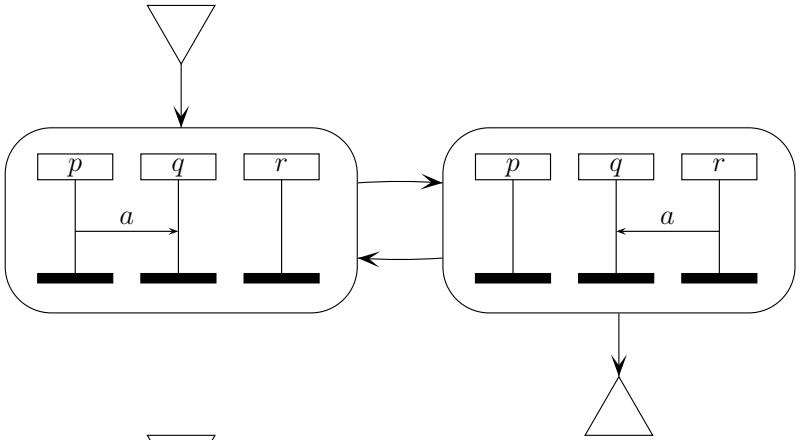




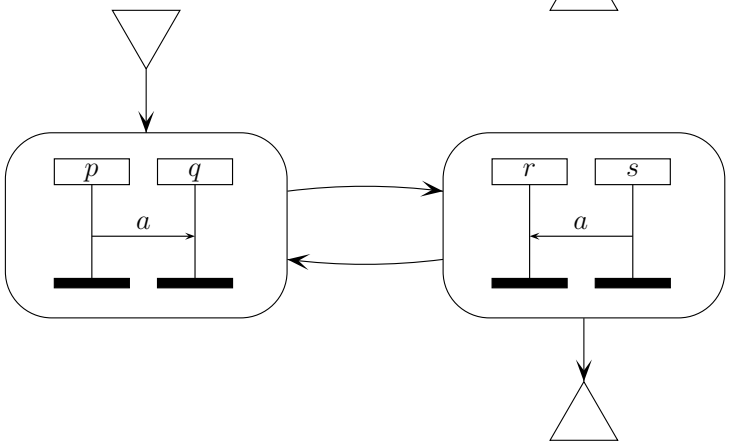
$p \to q$:

$q \to p$:

Globally cooperative CMSGs [Genest et al. ICALP'02; Morin STACS'02]

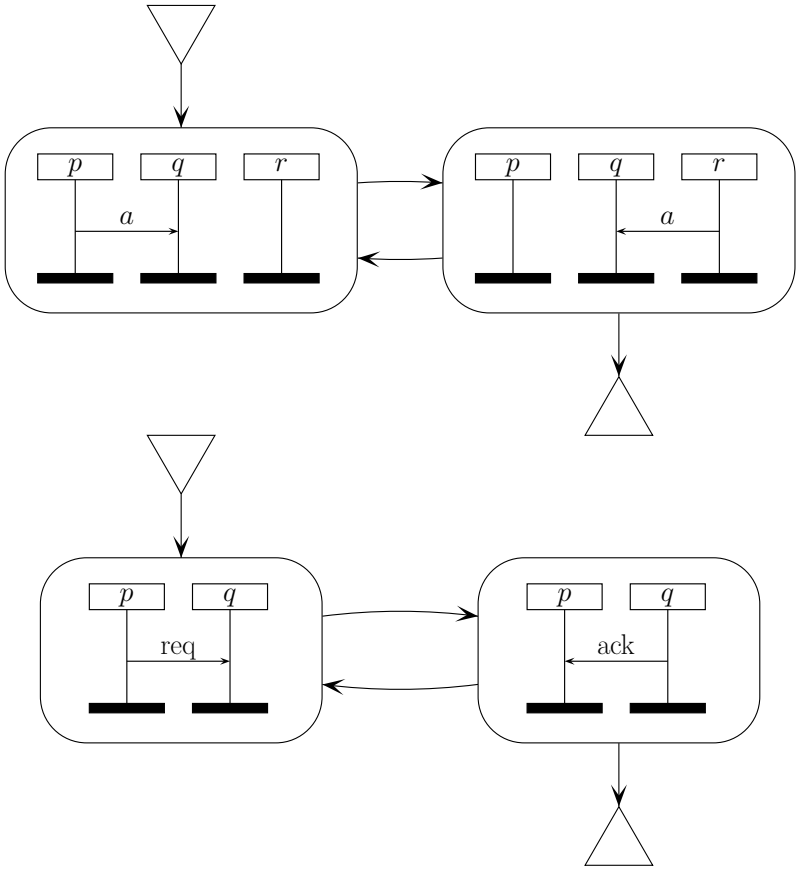


Globally coop. = implementable



Not globally cooperative!

Globally cooperative CMSGs [Genest et al. ICALP'02; Morin STACS'02]



Globally coop. = implementable

Special case: Regularity
[Henriksen et al. ICALP'00]

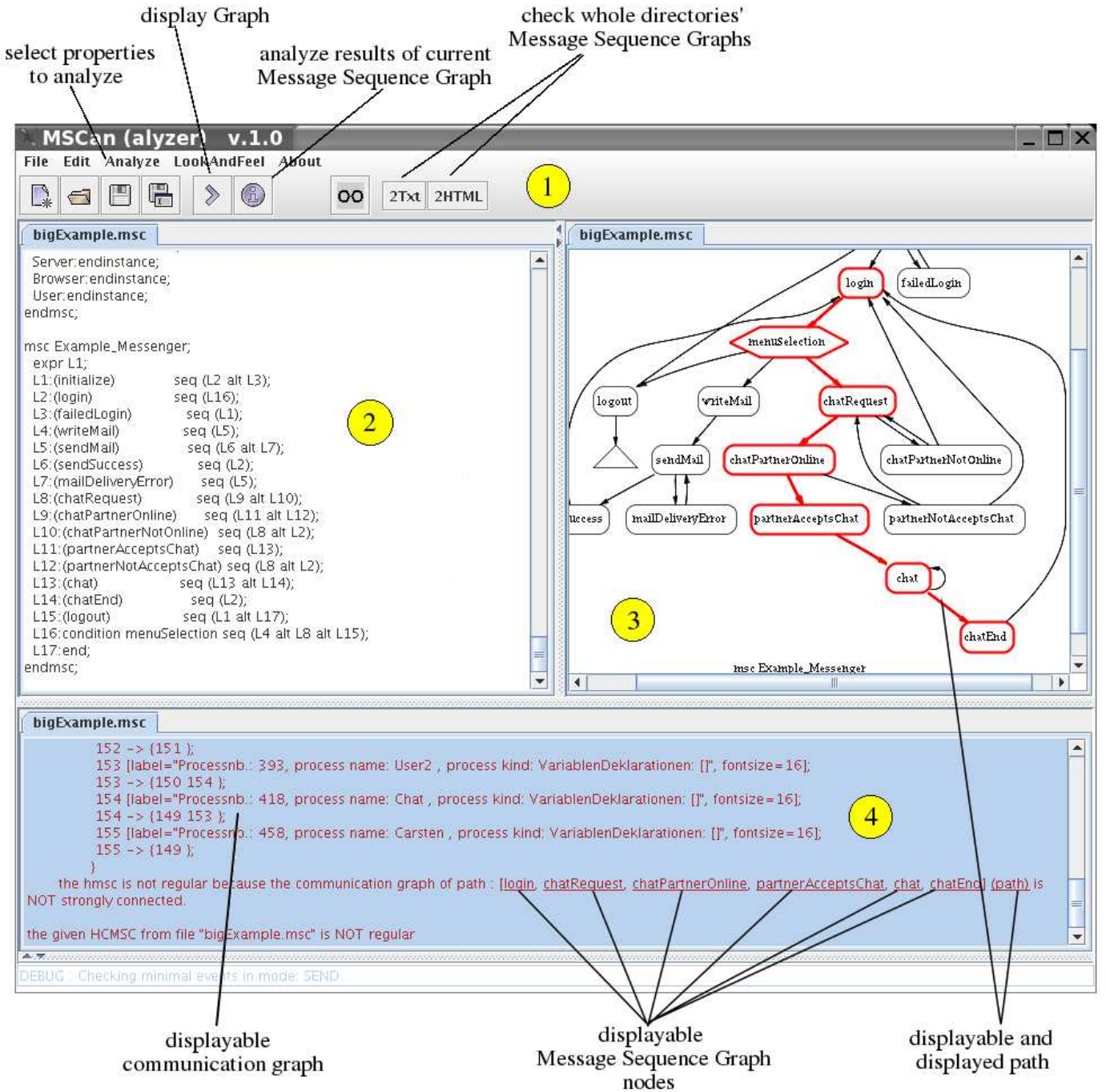
Overview

Property	**Complexity**	**Implementable w/wo deadlocks**
local	P	YES / YES
local-choice	P	YES / YES [Genest et al. ICALP'02]
locally coop.	P [Genest et al. ICALP'02]	YES / NO
regular	Co-NP complete	YES / NO
globally coop.	Co-NP complete [Muscholl & Peled MFCS'99]	YES / NO [Genest & Kuske & Muscholl IC'06]
generally		NO / NO

Safe/globally cooperative CMSGs come along with decidable model-checking problems! [Madhusudan & Meenakshi FSTTCS'01] [Genest & Kuske & Muscholl Information&Computation'06]

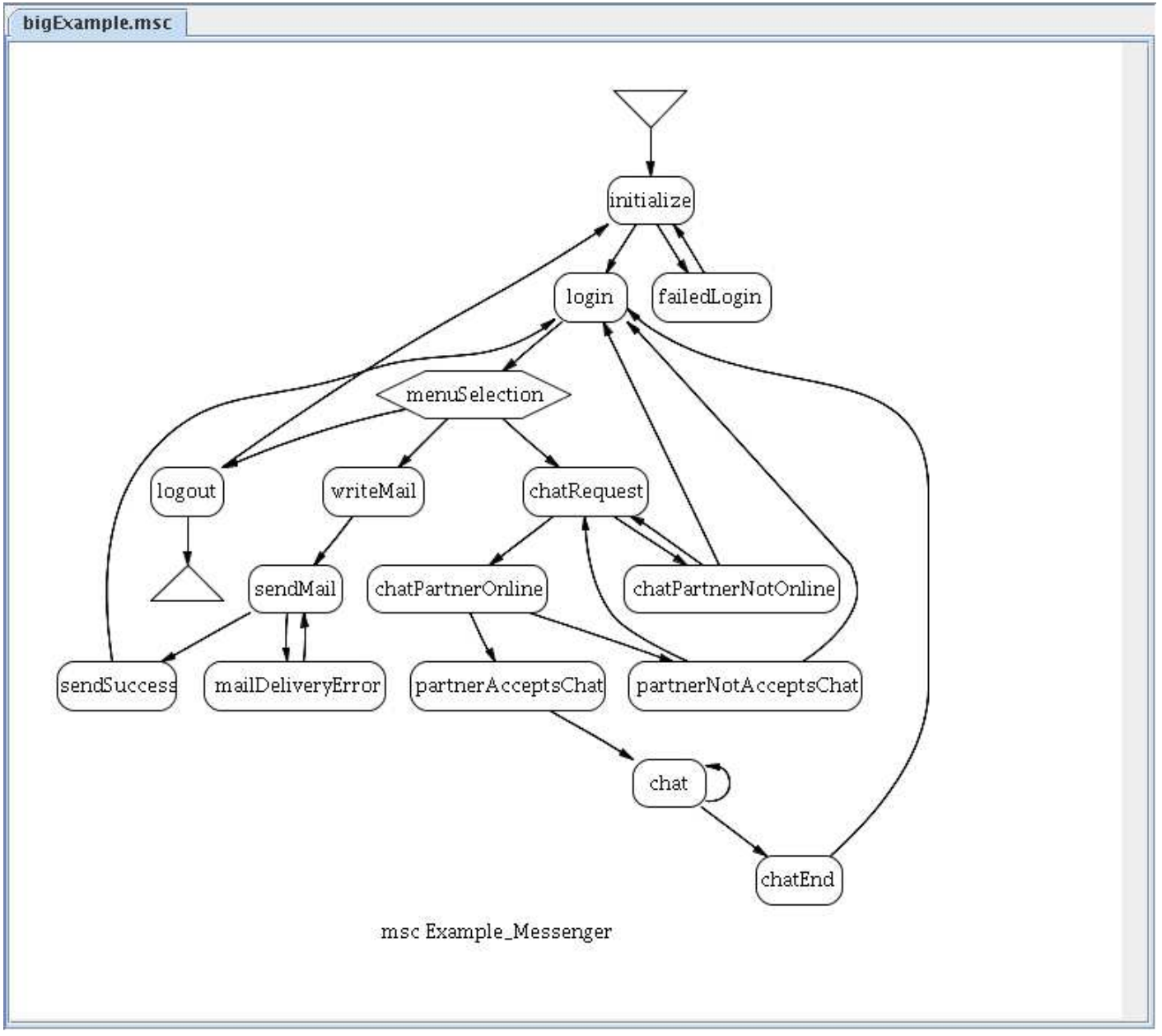
Presentation outline

1. Introduction to Message Sequence Charts
   - Message Sequence Charts (MSCs)
   - Message Sequence Graphs (MSGs)
2. Implementability
   - The Formal Model of CFMs
   - Examples
3. Tool Presentation
4. Summary and Outlook


display Graph
select properties to analyze
analyze results of current Message Sequence Graph
check whole directories' Message Sequence Graphs
MSCan (alyzer) v.1.0
File Edit Analyze LookAndFeel About
2Txt 2HTML
1
bigExample.msc
Server:endinstance;
Browser:endinstance;
User:endinstance;
endmsc;
msc Example_Messenger;
expr L1;
L1:(initialize) seq (L2 alt L3);
L2:(login) seq (L16);
L3:(failedLogin) seq (L1);
L4:(writeMail) seq (L5);
L5:(sendMail) seq (L6 alt L7);
L6:(sendSuccess) seq (L2);
L7:(mailDeliveryError) seq (L5);
L8:(chatRequest) seq (L9 alt L10);
L9:(chatPartnerOnline) seq (L11 alt L12);
L10:(chatPartnerNotOnline) seq (L8 alt L2);
L11:(partnerAcceptsChat) seq (L13);
L12:(partnerNotAcceptsChat) seq (L8 alt L2);
L13:(chat) seq (L13 alt L14);
L14:(chatEnd) seq (L2);
L15:(logout) seq (L1 alt L17);
L16:condition menuSelection seq (L4 alt L8 alt L15);
L17:end;
endmsc;
2
bigExample.msc
login
failedLogin
menuSelection
logout
writeMail
chatRequest
sendMail
chatPartnerOnline
chatPartnerNotOnline
mailDeliveryError
partnerAcceptsChat
partnerNotAcceptsChat
chat
chatEnd
msc Example_Messenger
3
bigExample.msc
152 -> {151 };
153 [label="Processnb.: 393, process name: User2 , process kind: VariablenDeklarationen: []", fontsize=16];
153 -> {150 154 };
154 [label="Processnb.: 418, process name: Chat , process kind: VariablenDeklarationen: []", fontsize=16];
154 -> {149 153 };
155 [label="Processnb.: 458, process name: Carsten , process kind: VariablenDeklarationen: []", fontsize=16];
155 -> {149 };
}
the hmsc is not regular because the communication graph of path : [login, chatRequest, chatPartnerOnline, partnerAcceptsChat, chat, chatEnd] (path) is NOT strongly connected.
the given HCMSC from file "bigExample.msc" is NOT regular
4
DEBUG : Checking minimal events in mode: SEND
displayable communication graph
displayable Message Sequence Graph nodes
displayable and displayed path

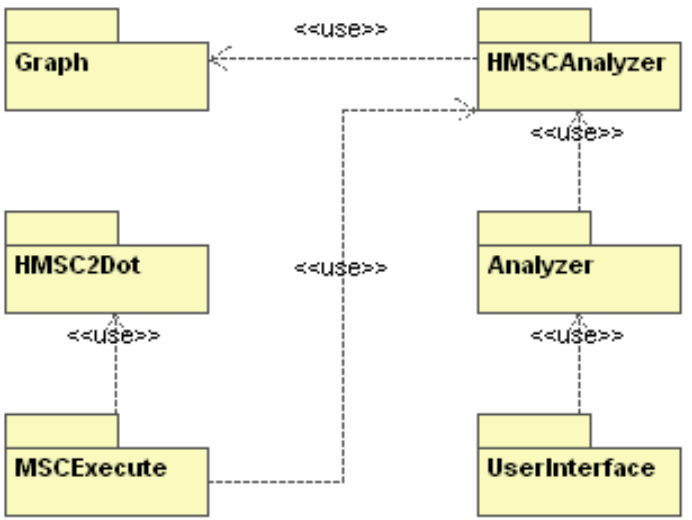
bigExample.msc

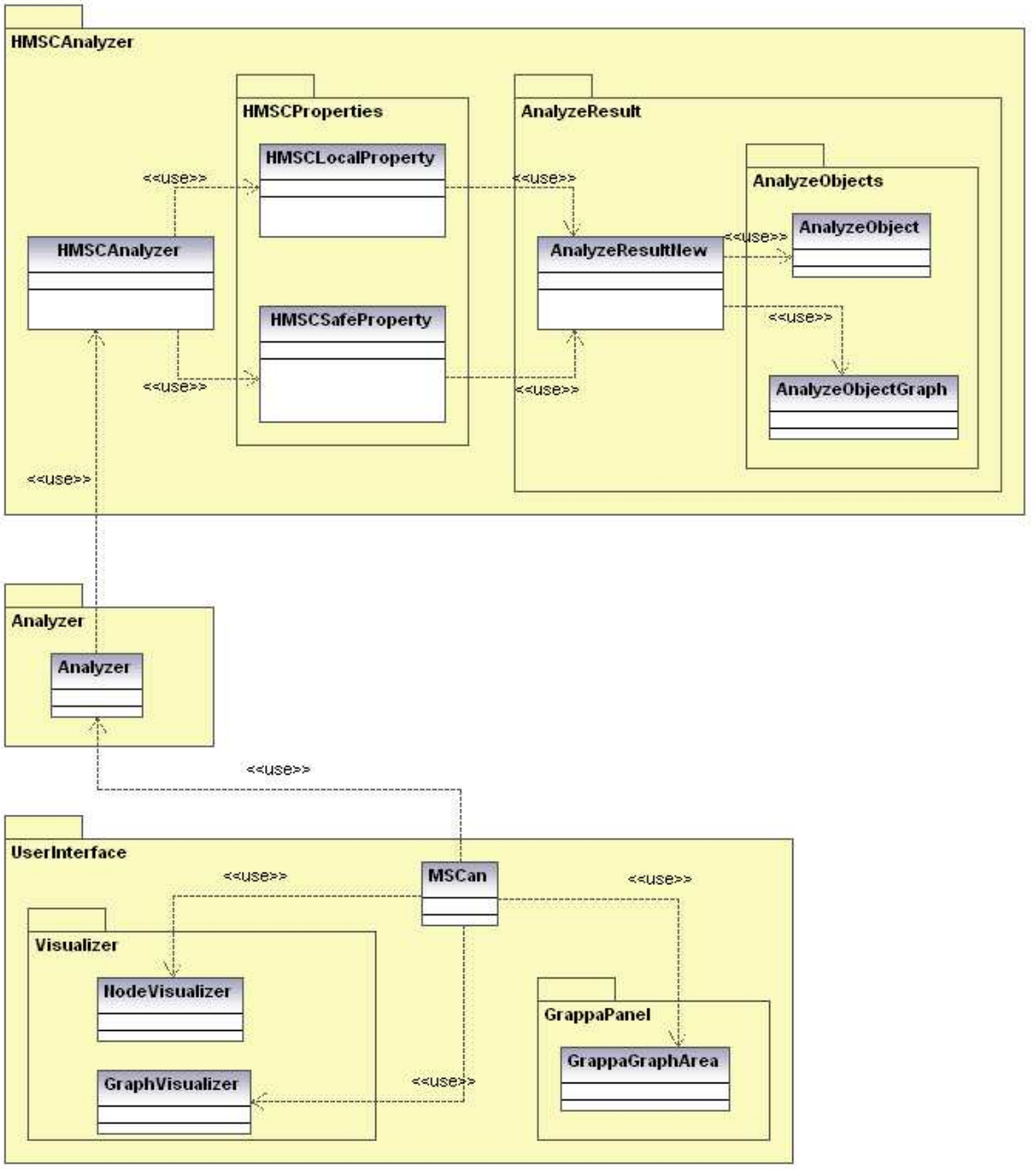
```
endmsc;

msc logout;
  inst User;
  inst Browser;
  inst Server;
  inst Database;
    User: out logout to Browser;
    Browser: in logout from User;
    Browser: out logout to Server;
    Server: in logout from Browser;
    Server: out logoutUser to Database;
    Database: in logoutUser from Server;
    Server: out logoutScreen to Browser;
    Browser: in logoutScreen from Server;
    Server: out closeConnection to User;
    User: in closeConnection from Server;
  Database:endinstance;
  Server:endinstance;
  Browser:endinstance;
  User:endinstance;
endmsc;

msc Example_Messenger;
  expr L1;
  L1:(initialize)            seq (L2 alt L3);
  L2:(login)                 seq (L16);
  L3:(failedLogin)             seq (L1);
  L4:(writeMail)             seq (L5);
  L5:(sendMail)              seq (L6 alt L7);
  L6:(sendSuccess)             seq (L2);
  L7:(mailDeliveryError)       seq (L5);
  L8:(chatRequest)           seq (L9 alt L10);
  L9:(chatPartnerOnline)       seq (L11 alt L12);
  L10:(chatPartnerNotOnline)  seq (L8 alt L2);
  L11:(partnerAcceptsChat)    seq (L13);
  L12:(partnerNotAcceptsChat) seq (L8 alt L2);
  L13:(chat)                 seq (L13 alt L14);
  L14:(chatEnd)              seq (L2);
  L15:(logout)               seq (L1 alt L17);
  L16:condition menuSelection seq (L4 alt L8 alt L15);
  L17:end;
endmsc;
```


bigExample.msc
initialize
login
failedLogin
menuSelection
logout
writeMail
chatRequest
sendMail
chatPartnerOnline
chatPartnerNotOnline
sendSuccess
mailDeliveryError
partnerAcceptsChat
partnerNotAcceptsChat
chat
chatEnd
msc Example_Messenger

bigExample.msc

```
Checking the given HCMSC from file "bigExample.msc for property "regular"
There are 10 loops to check.
    checking Loop 1 of 10 represented by path: [chatRequest, chatPartnerNotOnline] (path) for strongly connected communication graph
    The communication graph is strongly connected.

    checking Loop 2 of 10 represented by path: [sendMail, mailDeliveryError] (path) for strongly connected communication graph
    The communication graph is strongly connected.

    checking Loop 3 of 10 represented by path: [initialize, failedLogin] (path) for strongly connected communication graph
    The communication graph is strongly connected.

    checking Loop 4 of 10 represented by path: [login, writeMail, sendMail, sendSuccess] (path) for strongly connected communication graph
    The communication graph is strongly connected.

    checking Loop 5 of 10 represented by path: [login, chatRequest, chatPartnerOnline, partnerNotAcceptsChat] (path) for strongly connected communication graph
    The communication graph is strongly connected.

    checking Loop 6 of 10 represented by path: [login, chatRequest, chatPartnerOnline, partnerAcceptsChat, chat, chatEnd] (path) for strongly connected communication graph
 digraph dp_graph {
          node [outthreshold=100, inthreshold=100];
          98 [label="Processnb.: 332, process name: User , process kind: VariablenDeklarationen: []", fontsize=16];
          98 -> {99 103 };
          99 [label="Processnb.: 333, process name: Browser , process kind: VariablenDeklarationen: []", fontsize=16];
          99 -> {98 100 102 };
          100 [label="Processnb.: 334, process name: Server , process kind: VariablenDeklarationen: []", fontsize=16];
          100 -> {99 101 103 };
          101 [label="Processnb.: 385, process name: Database , process kind: VariablenDeklarationen: []", fontsize=16];
          101 -> {100 };
          102 [label="Processnb.: 393, process name: User2 , process kind: VariablenDeklarationen: []", fontsize=16];
          102 -> {99 103 };
          103 [label="Processnb.: 418, process name: Chat , process kind: VariablenDeklarationen: []", fontsize=16];
          103 -> {98 102 };
          104 [label="Processnb.: 458, process name: Carsten , process kind: VariablenDeklarationen: []", fontsize=16];
          104 -> {98 };
         }
    the hmsc is not regular because the communication graph of path : [login, chatRequest, chatPartnerOnline, partnerAcceptsChat, chat, chatEnd] (path) is NOT strongly
connected.

the given HCMSC from file "bigExample.msc" is NOT regular
```


Graph
<<use>>
HMSCAnalyzer
<<use>>
HMSC2Dot
<<use>>
Analyzer
<<use>>
<<use>>
MSCExecute
UserInterface


HMSCAnalyzer
HMSCProperties
HMSCLocalProperty
HMSCSafeProperty
AnalyzeResult
AnalyzeObjects
AnalyzeObject
AnalyzeObjectGraph
AnalyzeResultNew
HMSCAnalyzer
<<use>>
Analyzer
Analyzer
UserInterface
MSCan
Visualizer
NodeVisualizer
GraphVisualizer
GrappaPanel
GrappaGraphArea

Presentation outline

1. Introduction to Message Sequence Charts
   - Message Sequence Charts (MSCs)
   - Message Sequence Graphs (MSGs)
2. Implementability
   - The Formal Model of CFMs
   - Examples
3. Tool Presentation
4. Summary and Outlook

Summary

What was done so far

- implementation of GUI and console application
- implementation of several property checks
- tests on small and mid-size examples

Visit our tool web page at:

http://www-i2.informatik.rwth-aachen.de/MSCan/

Outlook

Future Work

- tests on larger examples
- performance tests
- speed-up by exploiting inclusion hierarchies
- realizability without extra data
  [Alur et al. ICSE'00; Lohrey CONCUR'02; Morin STACS'02]
- automatic CFM generation from MSC specifications