

2IW05 – Software Specification

Message Sequence Charts (MSCs): Part I

Mohammad Mousavi

Formal Systems Analysis (FSA)
Model-Driven Software Engineering
Department of Computer Science
TU/Eindhoven

November 21, 2011

- ① Group compositions are posted on the web page; please check.
- ② There seems to be a conflict in the first exam time: for those students, we may hold an exam in the morning of January 30.

Outline

1 MSC

2 Basic MSCs

3 Extensions

4 Consistency

Description of Functionality

Functionality specifies the goal concerning the behavior of the system:

Active relation between input and output (business information system)

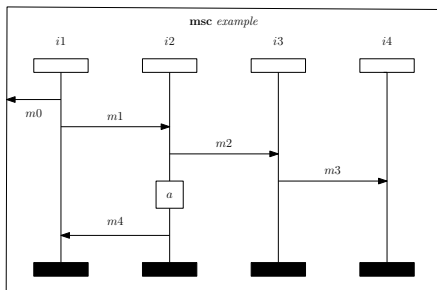
Reactive constant interaction with environment (protocols, Embedded Systems)

Message Sequence Charts

- standardized by International Telecommunication Union (ITU) since 1993

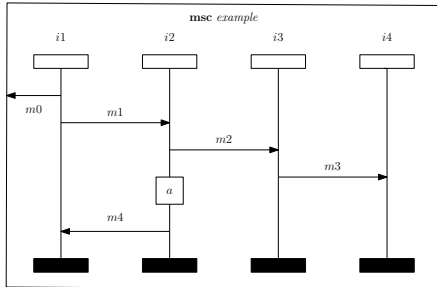
Message Sequence Charts

- standardized by International Telecommunication Union (ITU) since 1993
- graphical language



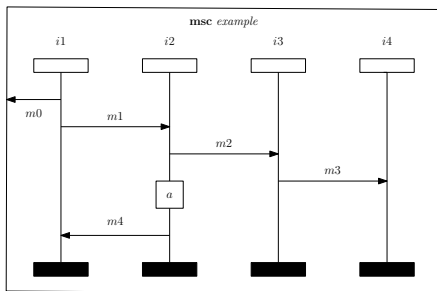
Message Sequence Charts

- standardized by International Telecommunication Union (ITU) since 1993
- graphical language
- description of interactions between system and environment



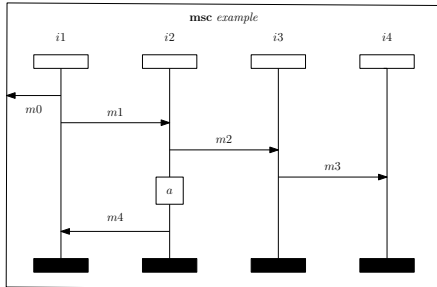
Message Sequence Charts

- standardized by International Telecommunication Union (ITU) since 1993
- graphical language
- description of interactions between system and environment
- description of interactions between system components



Message Sequence Charts

- standardized by International Telecommunication Union (ITU) since 1993
- graphical language
- description of interactions between system and environment
- description of interactions between system components
- communication is asynchronous



Requirements specification express the scenarios in use cases

Requirements specification express the scenarios in use cases

Visualization and simulation overview of communication between entities
in a simulation of a distributed system

- Requirements specification** express the scenarios in use cases
- Visualization and simulation** overview of communication between entities in a simulation of a distributed system
- Validation and verification** check whether a system can execute the sequences of events described by an MSC; consistency check w.r.t. a state-based behavioural description

Use of MSC

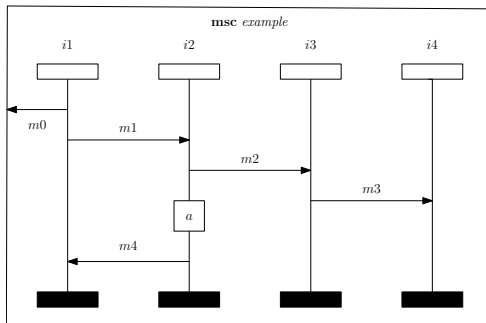
- Requirements specification** express the scenarios in use cases
- Visualization and simulation** overview of communication between entities in a simulation of a distributed system
- Validation and verification** check whether a system can execute the sequences of events described by an MSC; consistency check w.r.t. a state-based behavioural description
- Conformance testing** MSCs can be used for describing test purpose

Outline

- 1 MSC
- 2 Basic MSCs
- 3 Extensions
- 4 Consistency

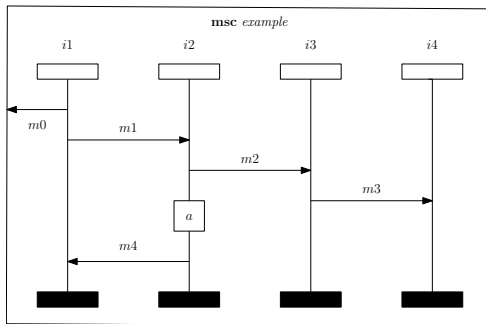
Basic Message Sequence Charts

- finite collection of instances
 - objects (instances) of classes
 - logical partitioning of the system



Basic Message Sequence Charts

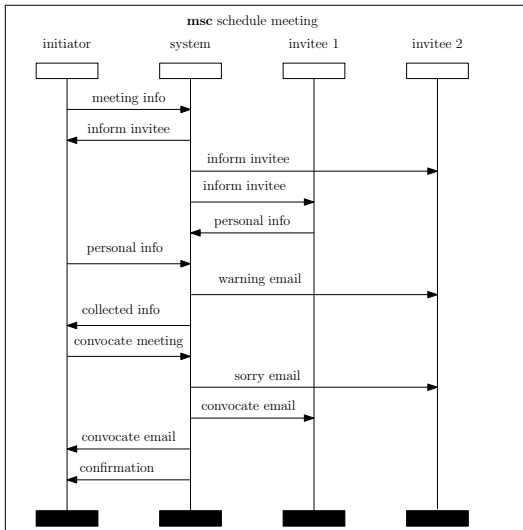
- finite collection of instances
 - objects (instances) of classes
 - logical partitioning of the system
- actions
 - operations
 - messages to/from the outside world



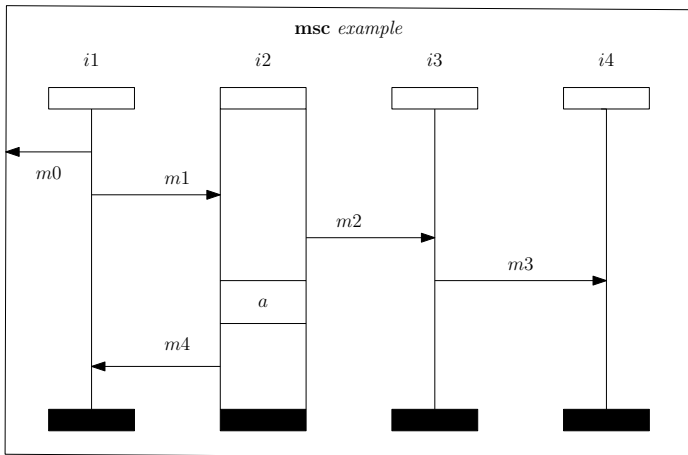
Use case **Schedule Meeting**

- ...
- **Description**
 - ① The initiator sends meeting information to the system
 - ② The system responds with inviting the participants to the meeting
 - ③ The invitees respond with their preferences
 - ④ If not all invitees respond within a predefined time-frame, the system will send a gentle reminder to those invitees
 - ⑤ The system sends the collected info to the initiator and issues an apology to all invitees that did not respond timely
 - ⑥ The system then sends convocations to all invitees that did respond
 - ⑦ Finally, the system sends a confirmation to the initiator
- **Exceptions**

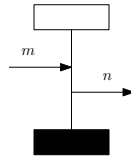
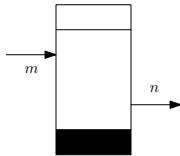
Scenario specification - Meeting Scheduler



Column-form instances



Column-form instances (II)



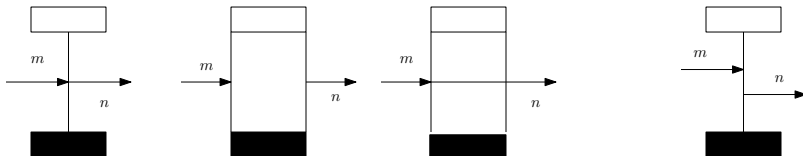
Events at same time point

Attaching two or more events at the same point (same height) to an axis is not allowed.

Events at same time point

Attaching two or more events at the same point (same height) to an axis is not allowed.

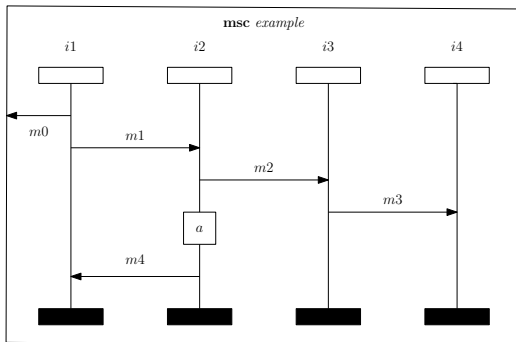
Exception to this rule:



Semantics of BMSC

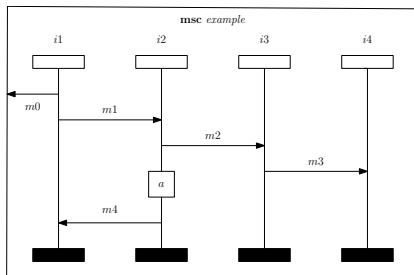
(Simplifying) Assumptions:

- 1 Every message is uniquely identified by its name.
- 2 Every local action is uniquely identified by its name.



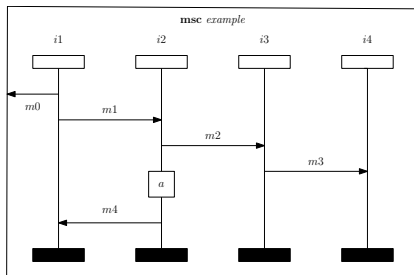
Events

- 1 Communication is asynchronous hence a message (between instances) with name m leads to two events
 - message outputs: $!m1, !m2, !m3, !m4$
 - message inputs: $?m1, ?m2, ?m3, ?m4$



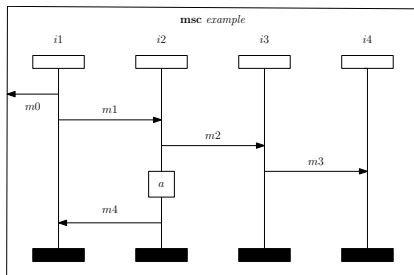
Events

- 1 Communication is asynchronous hence a message (between instances) with name m leads to two events
 - message outputs: $!m1, !m2, !m3, !m4$
 - message inputs: $?m1, ?m2, ?m3, ?m4$
- 2 Communication with the environment describes only a message output or only a message input: $!m0$



Events

- 1 Communication is asynchronous hence a message (between instances) with name m leads to two events
 - message outputs: $!m1, !m2, !m3, !m4$
 - message inputs: $?m1, ?m2, ?m3, ?m4$
- 2 Communication with the environment describes only a message output or only a message input: $!m0$
- 3 A local action is considered to occur instantaneously: a



Ordering the events:

- 1 On an instance events are ordered from top to bottom

Ordering the events:

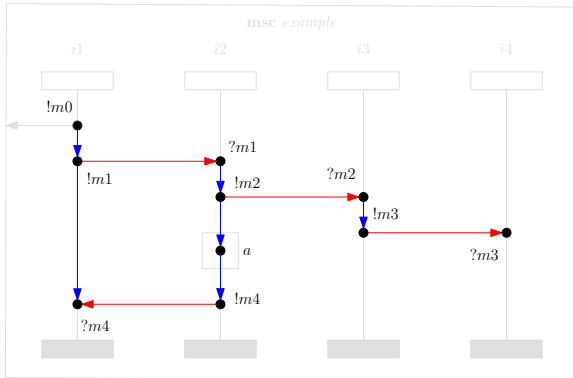
- ① On an instance events are ordered from top to bottom
- ② Vertical ordering on MSC frame does not enforce ordering on events in the environment

Ordering the events:

- ① On an instance events are ordered from top to bottom
- ② Vertical ordering on MSC frame does not enforce ordering on events in the environment
- ③ Message output occurs before corresponding message input

Ordering the events:

- 1 On an instance events are ordered from top to bottom
- 2 Vertical ordering on MSC frame does not enforce ordering on events in the environment
- 3 Message output occurs before corresponding message input



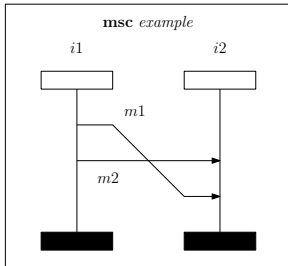
Ordering the events:

- ① On an instance events are ordered from top to bottom
- ② Vertical ordering on MSC frame does not enforce ordering on events in the environment
- ③ Message output occurs before corresponding message input

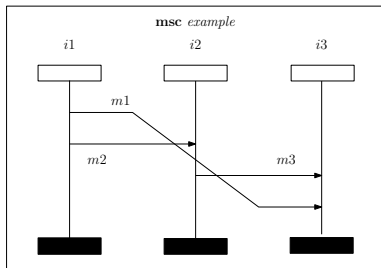
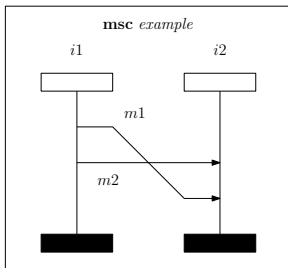
Definition

Let $<$ denote the ordering described by an MSC M . M is *consistent* iff the transitive closure of $<$ is irreflexive. Otherwise M is called *inconsistent*.

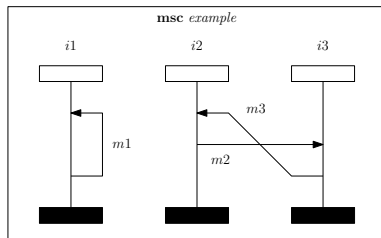
Overtaking of messages



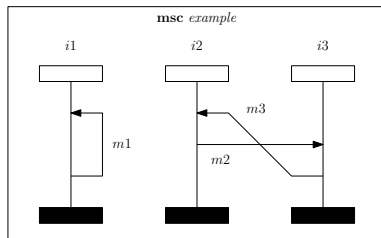
Overtaking of messages



Cyclic dependencies



Cyclic dependencies



Such MSCs are called inconsistent!

Outline

- 1 MSC
- 2 Basic MSCs
- 3 Extensions**
- 4 Consistency

Additional features

- ① Ordering facilities
 - ① Coregions

① Ordering facilities

① Coregions

② Causal orderings

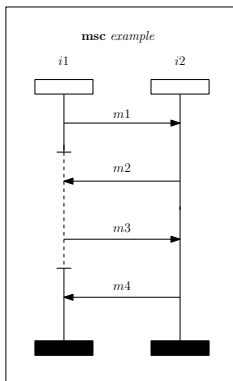
① between events from the same instance

② between events from different instances

Additional features

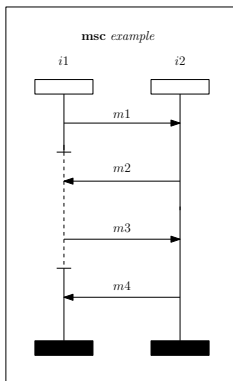
- 1 Ordering facilities
 - 1 Coregions
 - 2 Causal orderings
 - 1 between events from the same instance
 - 2 between events from different instances
- 2 Compositions of MSCs

Coregions



What are the orderings described on instance *i1*?

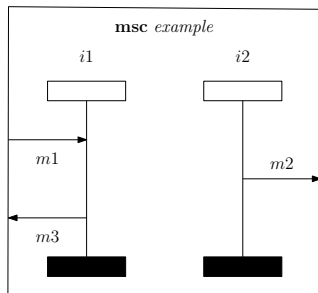
Coregions



What are the orderings described on instance $i1$?

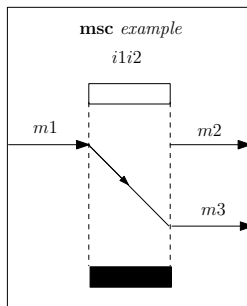
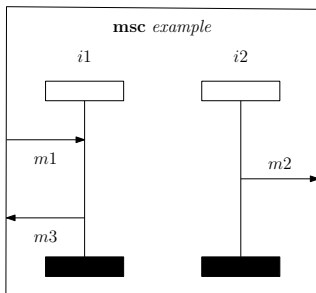
$!m1 < ?m2$ $!m1 < !m3$ $?m2 < ?m4$ $!m3 < ?m4$

Causal orderings



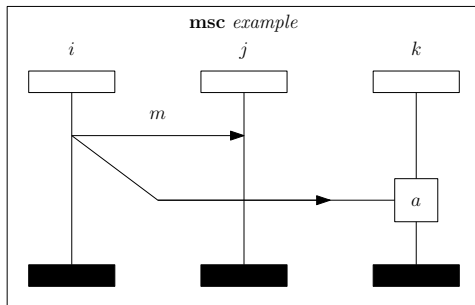
Abstract from distinction between instances $i1$ and $i2$, i.e., combine them into one instance $i12$. How to represent the resulting instance?

Causal orderings



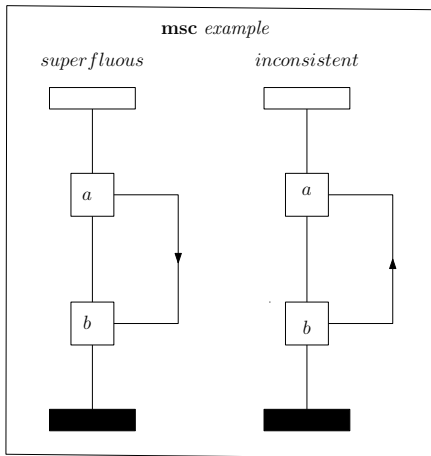
Abstract from distinction between instances $i1$ and $i2$, i.e., combine them into one instance $i12$. How to represent the resulting instance?

Causal ordering on events from different instances

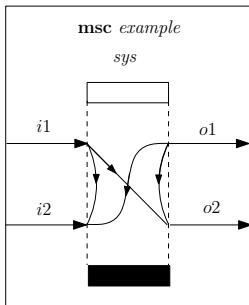


The reason for the ordering is abstracted from!

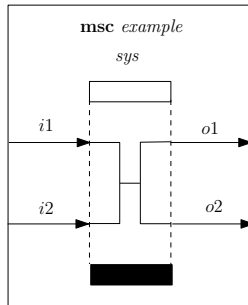
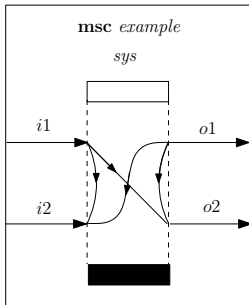
Superfluous and inconsistent causalities



Describing ordering without arrows



Describing ordering without arrows



Outline

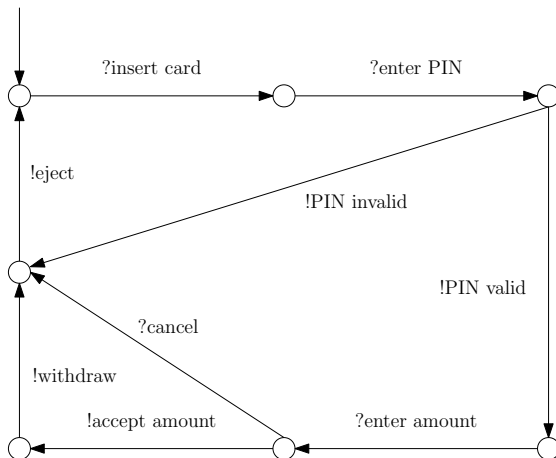
- 1 MSC
- 2 Basic MSCs
- 3 Extensions
- 4 Consistency**

- ① Message Sequence Charts

Description of Behavior

- ① Message Sequence Charts
- ② Labeled Transition Systems

Labeled Transition Systems



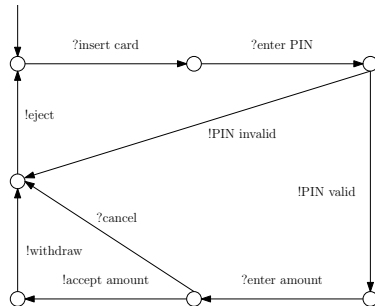
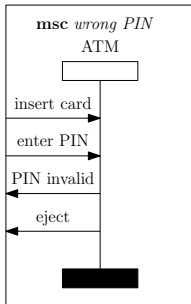
Labeled Transition Systems

Definition (Labeled Transition System (LTS))

A labeled transition system T is a quintuple $(S, A, \rightarrow, \downarrow, s_0)$ where

- S is a set of states;
- A is a set of actions (action labels)
- $\rightarrow \subseteq S \times A \times S$ is a set of transitions
- $\downarrow \subseteq S$ is a set of successfully terminating states
- $s_0 \in S$ is the initial state

Consistency MSC and LTS



Definition (Traces of LTS)

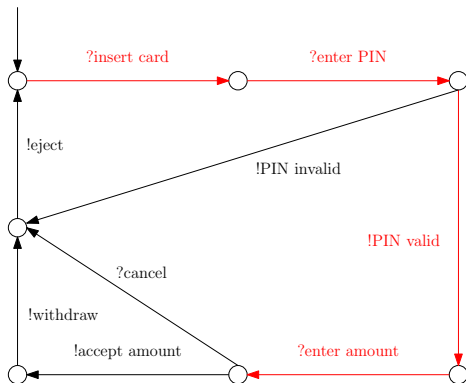
The generalized transition relation $\rightarrow \subseteq S \times A^* \times S$ is the smallest relation that satisfies:

- 1 $s \xrightarrow{\epsilon} s$ for all $s \in S$
- 2 $s \xrightarrow{a} s'$ for all s, a, s' such that $s \xrightarrow{a} s'$
- 3 $s \xrightarrow{\sigma\sigma'} s''$ for any s, σ, σ', s'' such that $s \xrightarrow{\sigma} s'$ and $s' \xrightarrow{\sigma'} s''$ for some $s' \in S$

The set of traces of T , denoted $\text{traces}(T)$ is defined as follows:

$$\text{traces}(T) = \{\sigma \in A^* \mid \exists_{s \in S} s_0 \xrightarrow{\sigma} s\}$$

Consistency MSC and LTS



`?insert card · ?enter PIN · !PIN valid · ?enter amount`

Already infinitely many traces!

Consistency MSC and LTS

The trace corresponding to an acyclic total ordering \ll on events $\{e_1, \dots, e_n\}$ is the trace $e_1 \cdots e_n$ with $e_i \ll e_{i+1}$ for $1 \leq i < n$. We denote this trace by $\text{tr}(\ll)$.

Definition (Traces of MSC)

The traces of an MSC M with ordering $<$ is

$$\text{traces}(M) = \{\text{tr}(\ll) \mid \ll \text{ acyclic and total on } E, < \subseteq \ll\}$$

For trace t and set of events E , $t \upharpoonright E$ denotes the trace t' that is obtained by removing from t all events outside set E . For a set of traces T , $T \upharpoonright E$ denotes the set $\{t \upharpoonright E \mid t \in T\}$.

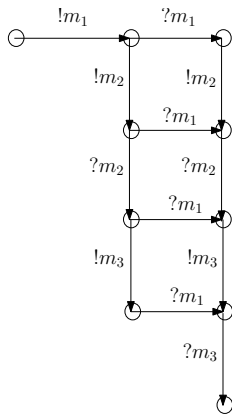
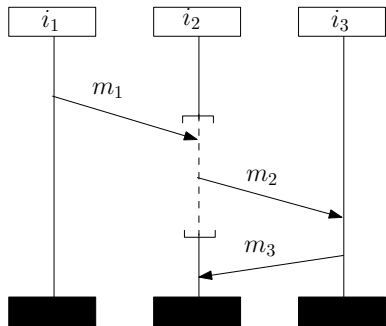
Definition

An MSC is *consistent* with an LTS iff

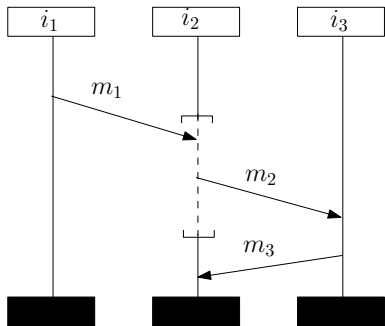
$$\text{traces}(M) \upharpoonright (E_M \cap E_T) \subseteq \text{traces}(T) \upharpoonright (E_M \cap E_T).$$

where E_M is set of events that occur in the MSC and E_T is set of events that occur in the LTS

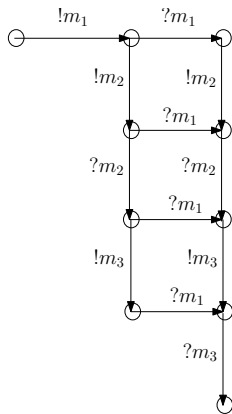
Consistent?



Consistent?

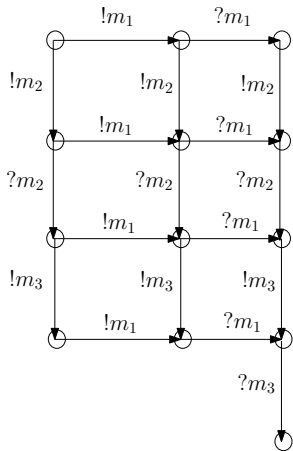


Draw a consistent LTS.



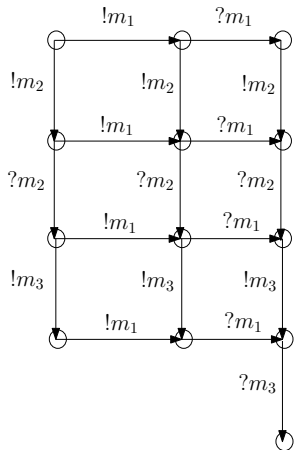
Consistency

Draw a consistent LTS.



Consistency

Draw a consistent LTS.



Simpler alternative?