Message Refinement: Describing Multi-Level Protocols in MSC

Andre Engels

Eindhoven University of Technology

Contents

- Motivation
- History
- Message Refinement
- When is Message Refinement Allowed?
- Semantics
- Conclusions

- Levels of communication
- Forgetting about lower levels can cause problems
- But adding them can as well
- Solution: Give MSC without lower levels, but show how they should be added

Some methods exist to compose/refine MSCs:

- Instance Refinement
- Reference MSC
- High-Level MSCs

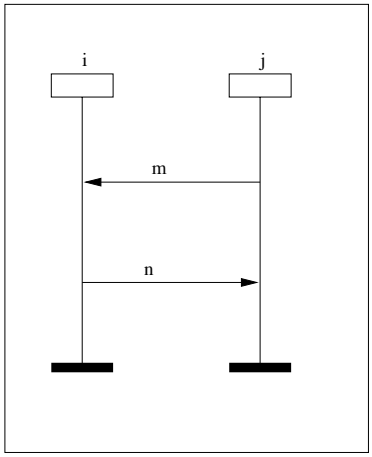
Maybe we can extend the idea of Refinement to:

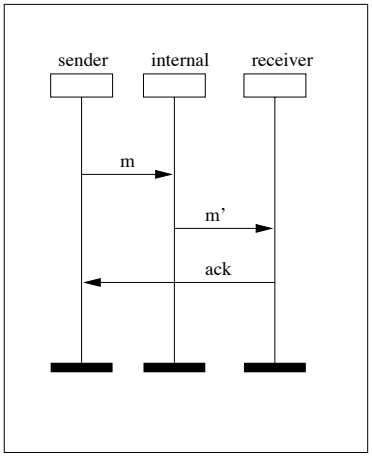
- Action Refinement
- Message Refinement

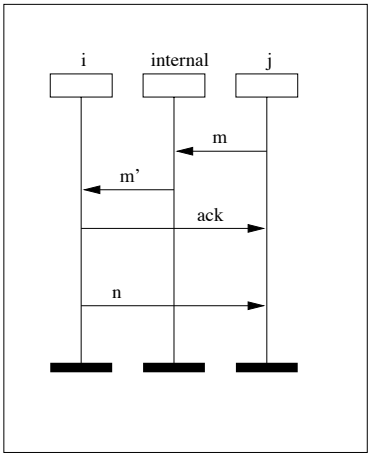
The basic idea

Replace a message in an MSC by another, complete, MSC.

We call the refining MSC a 'Protocol MSC'







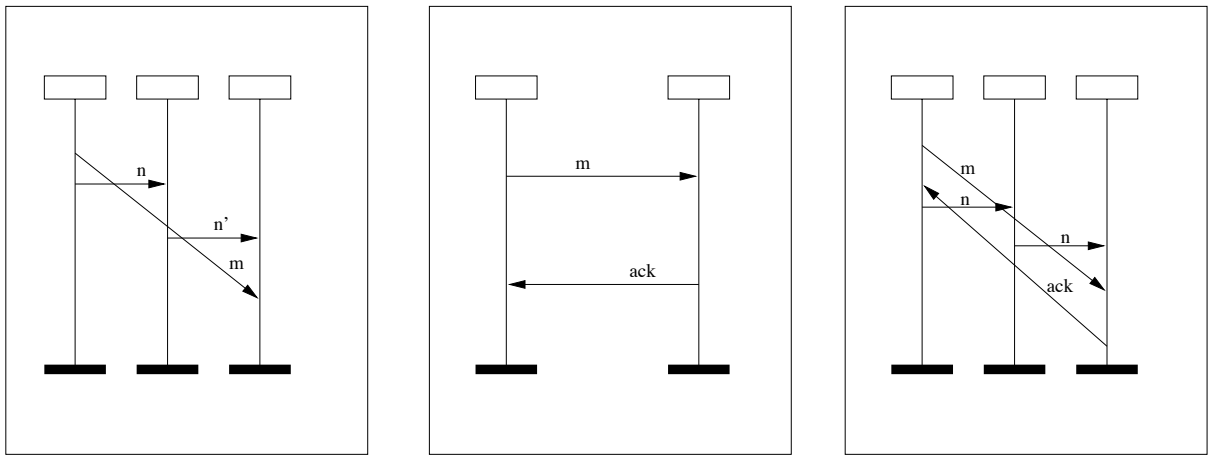
What properties should a Protocol MSC have?

- Two special instances, sender and receiver
- Events e_1 on sender and e_2 on receiver such that $e_1 \ll e_2$.
- No deadlock, no lifelock

What does the refined MSC look like?

- All instances of the original MSC, all instances except sender and receiver of the protocol MSC
- All events of the original MSC except sending and receiving the message to be refined, all events of the protocol MSC
- All orderings of the original MSC and the protocol MSC, where the sending/receiving of the refined message are replaced by the events on the sender/receiver of the protocol MSC

A Problem



Even though the Original MSC and the Protocol MSC are 'neat', the refined MSC contains a deadlock.

A protocol is bidirectional if there is an event e at the receiver and an event e' on the sender such that $e \ll e'$, and unidirectional otherwise.

Rule:

- Any message may be replaced by a unidirectional protocol.

- A message may be replaced by a bidirectional protocol, if $!m$ and $?m$ are not on the same instance, and there is no event e with $!m \ll e \ll ?m$.

A better idea might be to add Synchronous Communication to the language, that is, a message for which its sending and receival can be regarded a single action. In that case we can say:

- A 'normal' message may only be refined by a unidirectional protocol
- A synchronous message may only be refined by a bydirectional protocol

A semantics for Message Refinement can be given, but it is complicated. It is better to define Message Refinement (and possibly other composition/refinement operators as well) as an operator *on* instead of *in* the language, that is as a way to form out of two MSCs a new one.

Synchronous communication can be modelled as a single event that falls in the ordering of two different instances.

Conclusions

- Message Refinement provides a new way to combine a number of smaller MSCs into one large MSC
- The difference between unidirectional and bidirectional is important in deciding when Message Refinement is allowed.
- Addition of Synchronous Communication would help clarifying this difference.
- It is better to define composition techniques like Message Refinement as operators on the language than as operators in the language.