# Design Decisions on Data and Guards in MSC2000

**Andre Engels**

Eindhoven University of Technology

## Contents

- History
- Decisions on Data
- Guards
- Semantics of Guards
- Conclusions

- MSC 1992: Messages, Actions
- MSC 1996: References, HMSC
- MSC 2000: Time, Object Orientation, Data

Incorporating data:

Early attempts (Grabowski et al. 1995, Baker & Jervis 1997) tried to combine MSC and ASN.1.

Feijs & Mauw (1998): General interface between MSC and data language.

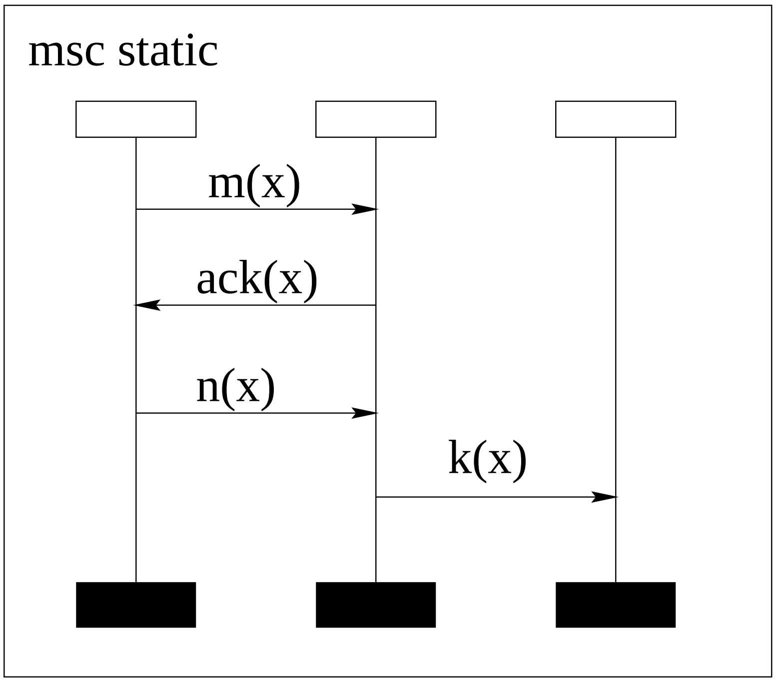
Engels, Feijs & Mauw (1999): Extension with dynamic variables

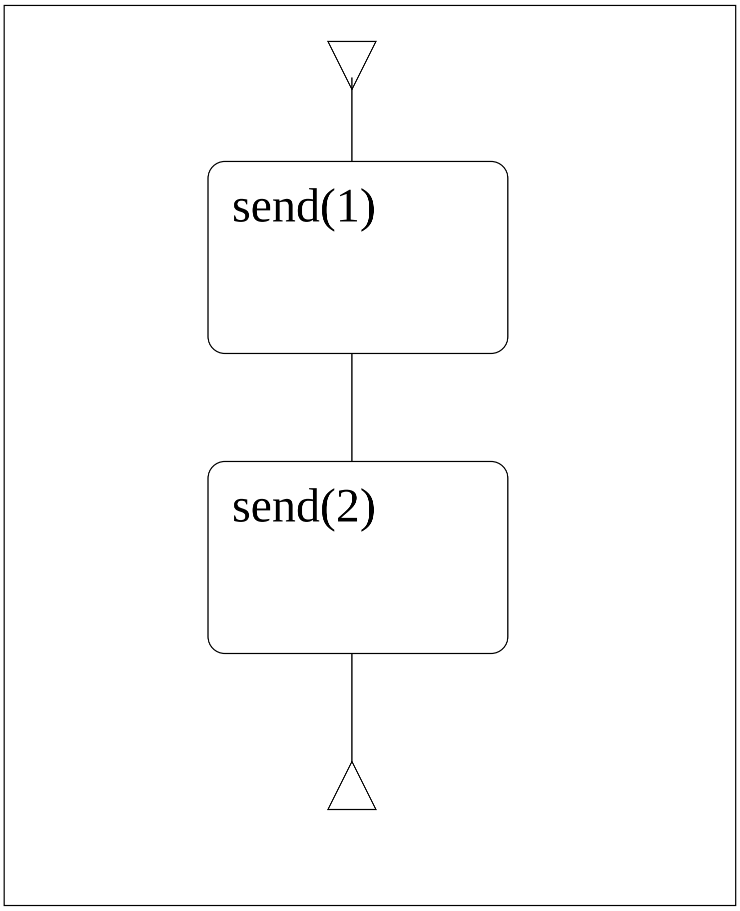
Eindhoven (1999): Final choices

- Backward Compatibility
- Intuitive Semantics
- Semantic Simplicity
- Correspondence to Existing Practice

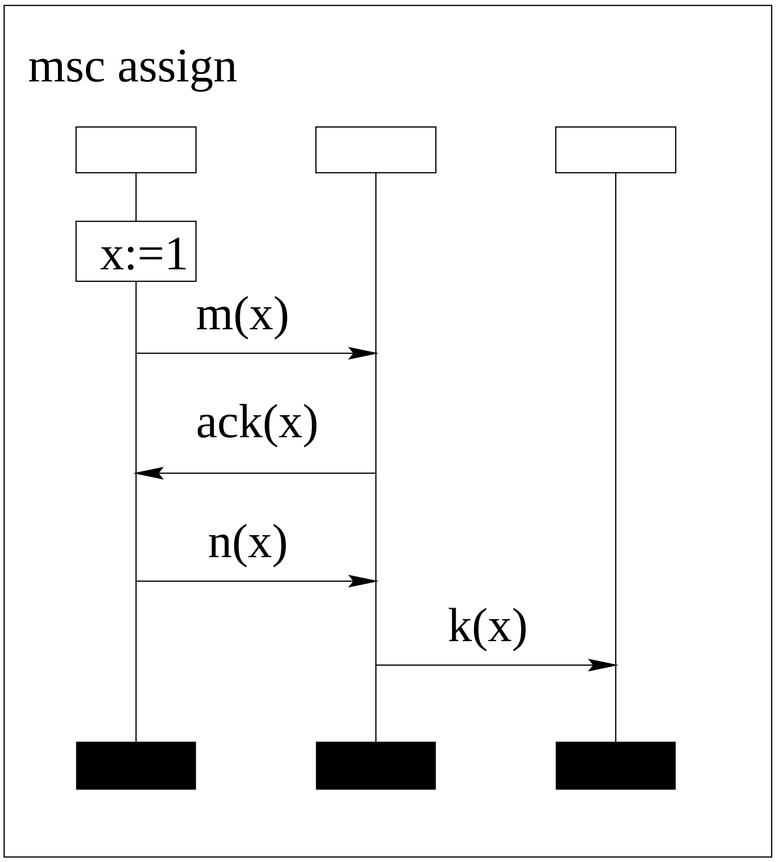
## Data: Static vs. Dynamic Variables

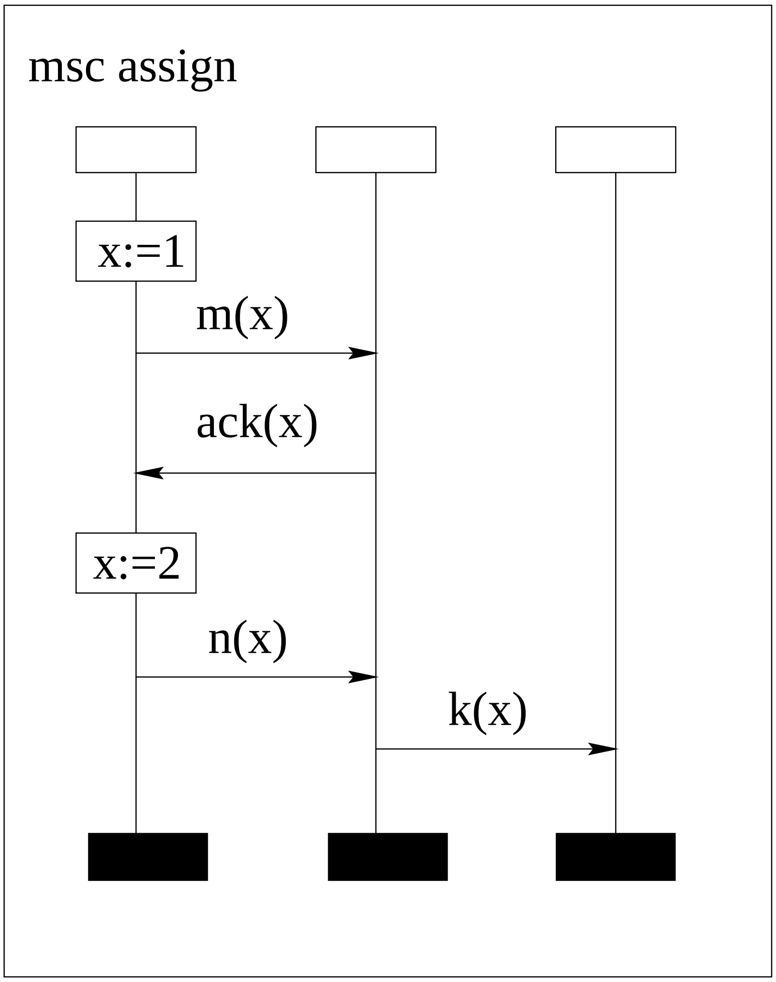
- Fully static variables (Feijs & Mauw 1998)
- Parameter variables
- Single assignment
- Multiple assignment


msc static
m(x)
ack(x)
n(x)
k(x)


send(1)
send(2)

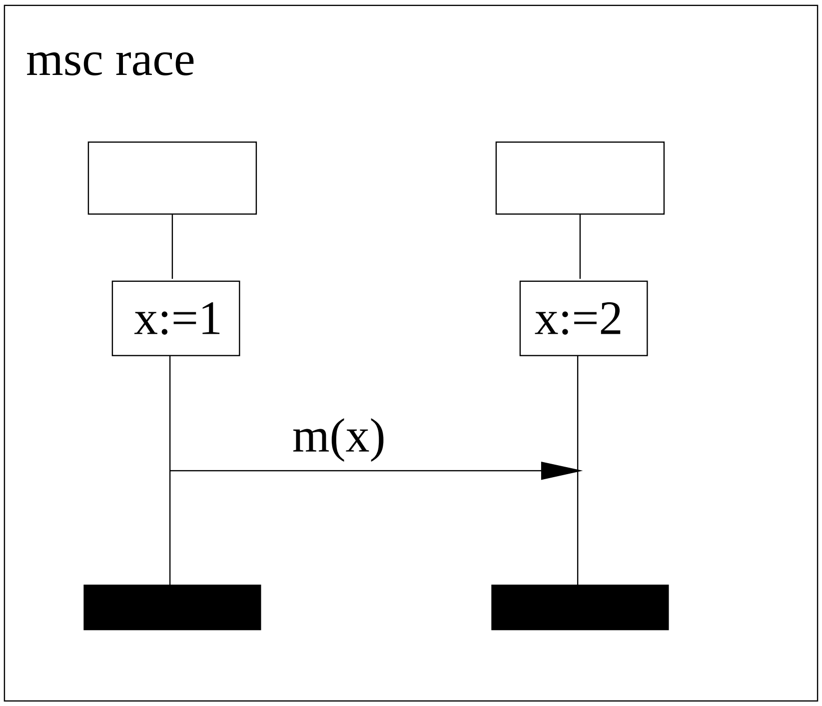
# Single Assignment




msc assign
x:=1
m(x)
ack(x)
x:=2
n(x)
k(x)

For MSC multiple assignments have been chosen; these give the most possibilities. Parametric variables are also allowed, static ones are not.

# A Problem



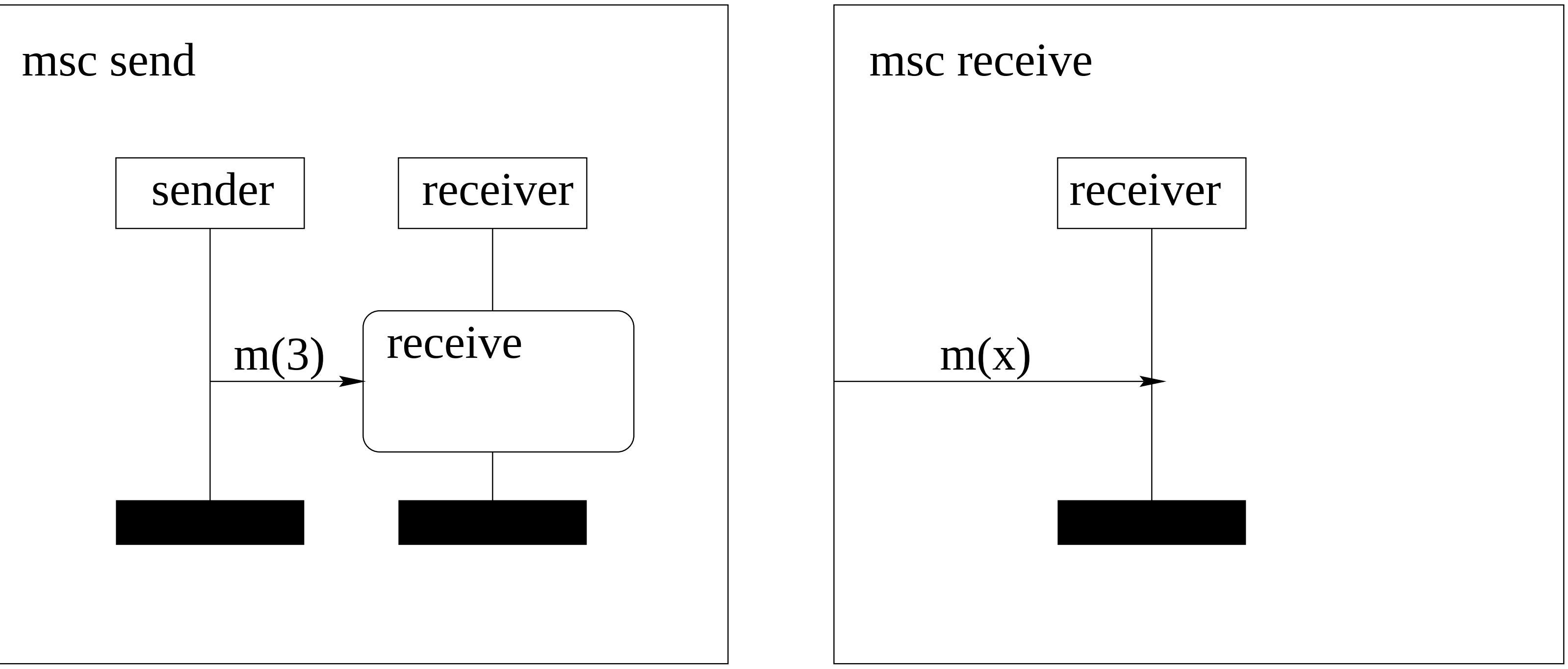

Because of this:

One instance owns a Variable

Others receive the value through messages

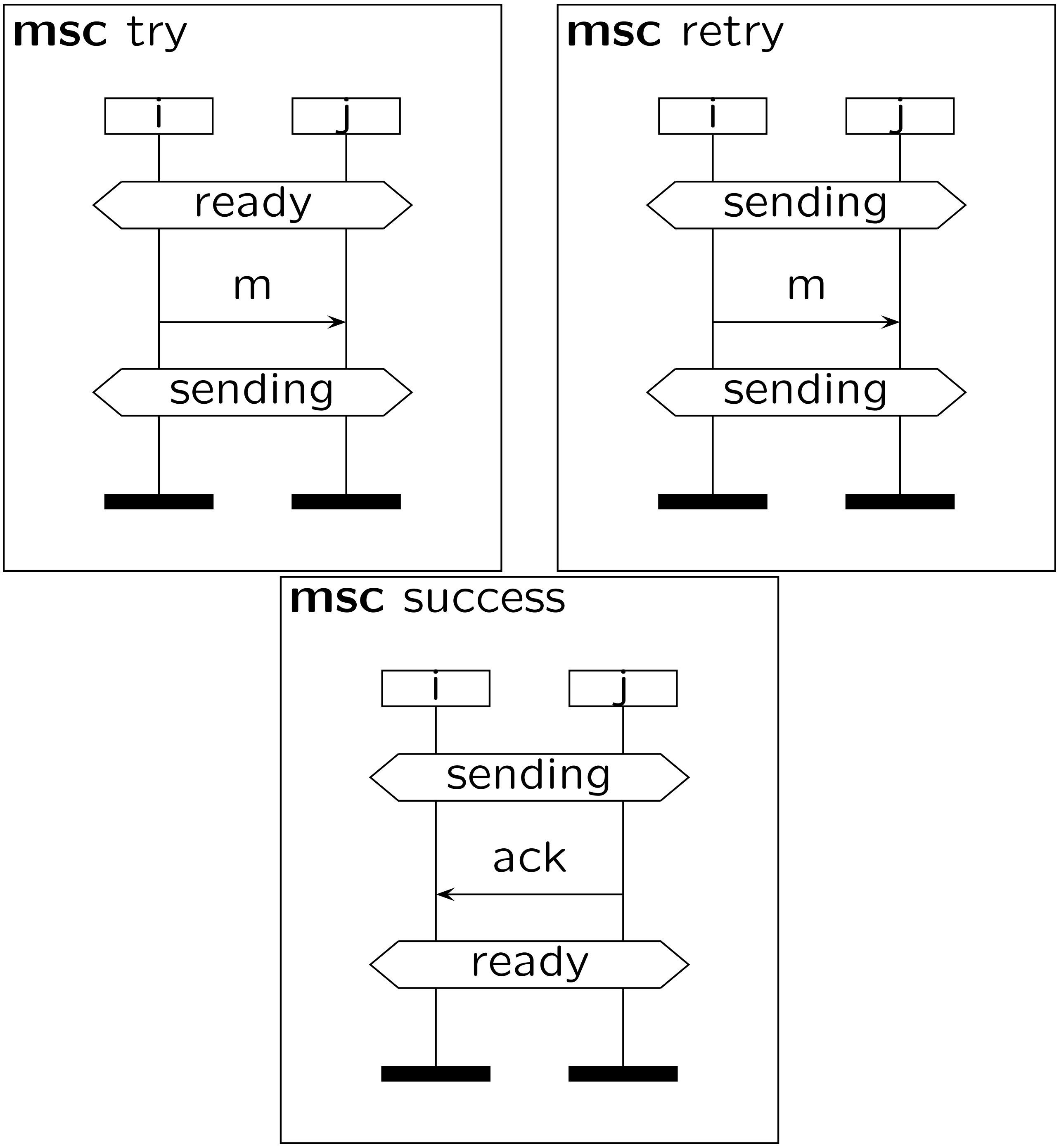
# Place of Assignment

- Action Boxes
- Message Propagation through Gates



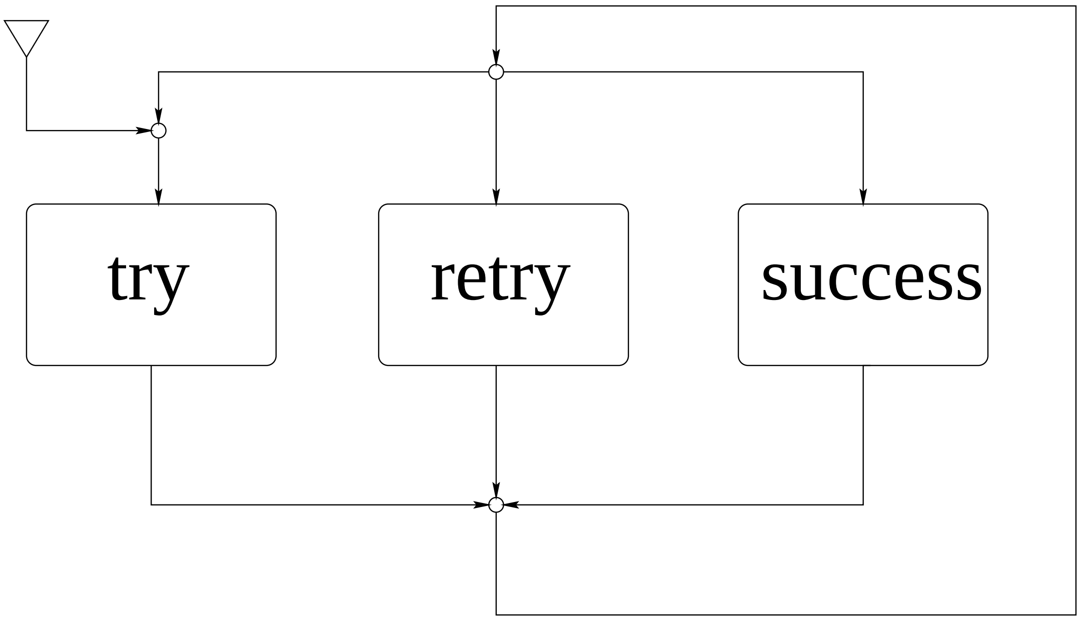
Conditions exist since MSC'92, but had almost no semantical meaning.

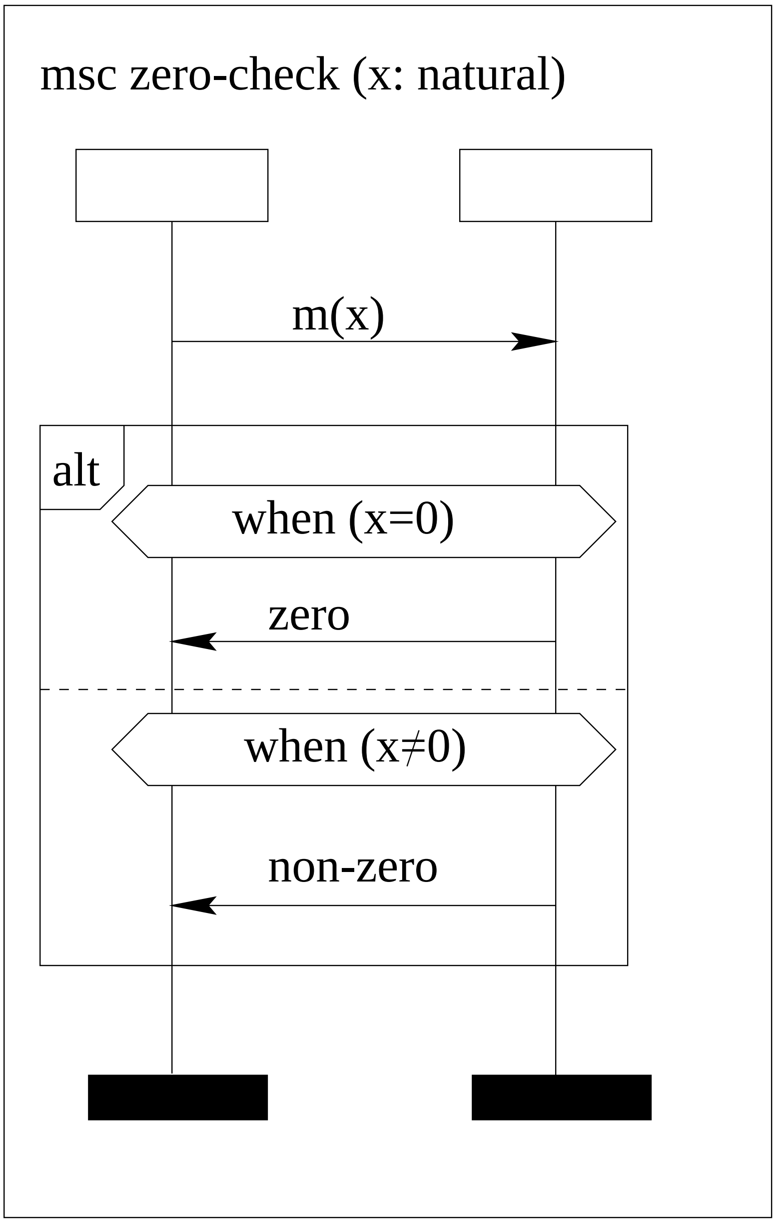
In practice they were used to combine MSCs.

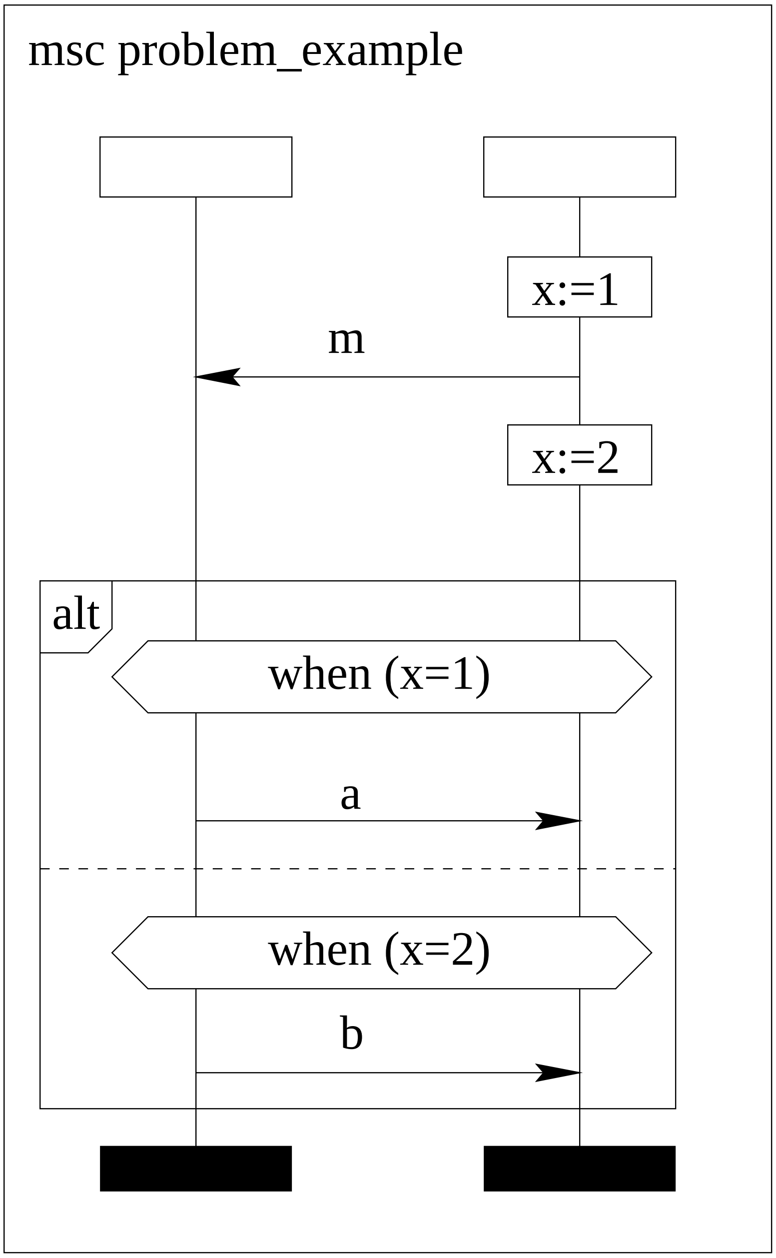

msc try
i
j
ready
m
sending
msc retry
i
j
sending
m
sending
msc success
i
j
sending
ack
ready

To make this type of usage formal, guards were introduced.

1. Syntax: Old fashioned conditions + keyword **when**
2. Semantics: Can only be passed in the correct state or if expression is true
3. Other conditions set the state


try
retry
success


msc zero-check (x: natural)
m(x)
alt
when (x=0)
zero
when (x≠0)
non-zero


msc problem_example
x:=1
m
x:=2
alt
when (x=1)
a
when (x=2)
b

1. 'First to pass the gate'

pro: Semantically simple and good-looking

contra: Sometimes gives counter-intuitive results

2. 'Look upward' ('Owner decides')

pro: Intuitive in usage

contra: Semantics are complex (backtracking) and sometimes absent

Both semantics at some times give unexpected results

Basic Difference: Time of Evaluation

1: When first instance passes the guard

2: When owning instance passes the guard

Neither satisfies

Only allow the cases where both give the same answer

1. Evaluation does not change:
No dynamic variables

2. Evaluation at the same time:
Owning instance is the only ready one

If a guard contains a data expression, then this expression must be of type Boolean. If this expression furthermore contains dynamic variables, it may only cover a single instance, which thus must be the only ready instance of the scope.

- Explanation of Design Choices
- Allowing much where much is useful, little where little is necessary
- Look at Semantics when extending the language
- Explanation of design choices helps users understand