Analyzing Message Sequence Charts

Anca Muscholl

LIAFA, Université Paris 7

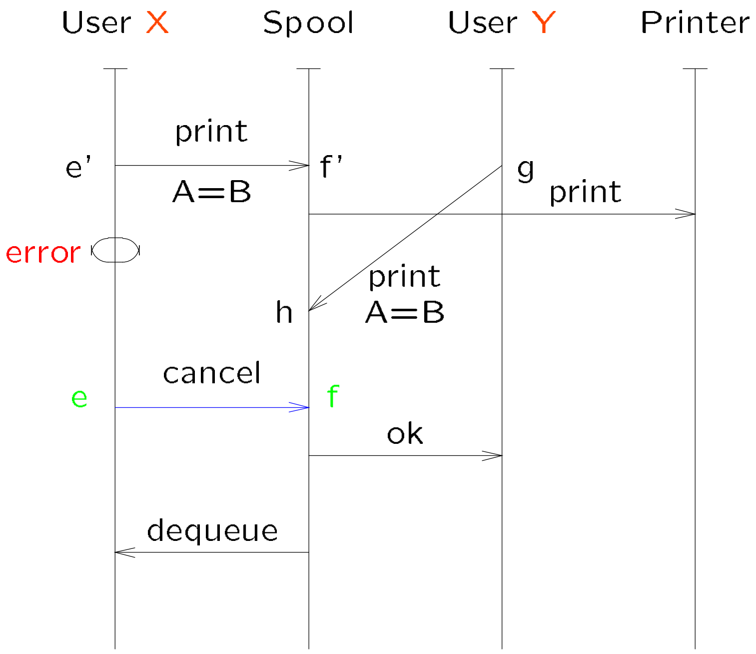
joint work with Doron Peled (Bell Labs)

Contents

Message Sequence Charts (time diagrams, ...)

- Interaction between processes / objects (instances)
- Visual description of requirements
- Concurrent systems (partial order semantics)
- Scenarios in early system design: sunny day or rainy day
- tools for analysis: uBET, MESA,...

MSC syntax



- instances (processes) X, Y, Spool, Printer
- events $e, f, g, h, \ldots$
- messages $(e, f), (g, h), \ldots$
- local condition (error)

MSC syntax (cont.)

MSC: partially ordered set of labeled events

events: send, receive or local

partial order: e *precedes* f, if

- (e, f) message

- e **before** f on the same process line

Visual order: partial order of *precedence*

Ex.

$$\begin{aligned} e' &< f' \\ f' &< h \\ f' &< f \end{aligned}$$

HMSC

Single MSCs composed to HMSC: branching, iteration, ...

- Message Sequence Graphs (HMSC)
- hierarchical MSC (nesting)

HMSC = transition system labeled by single MSCs

$(S, \longrightarrow, c)$

$c(s)$ = single MSC labeling state s

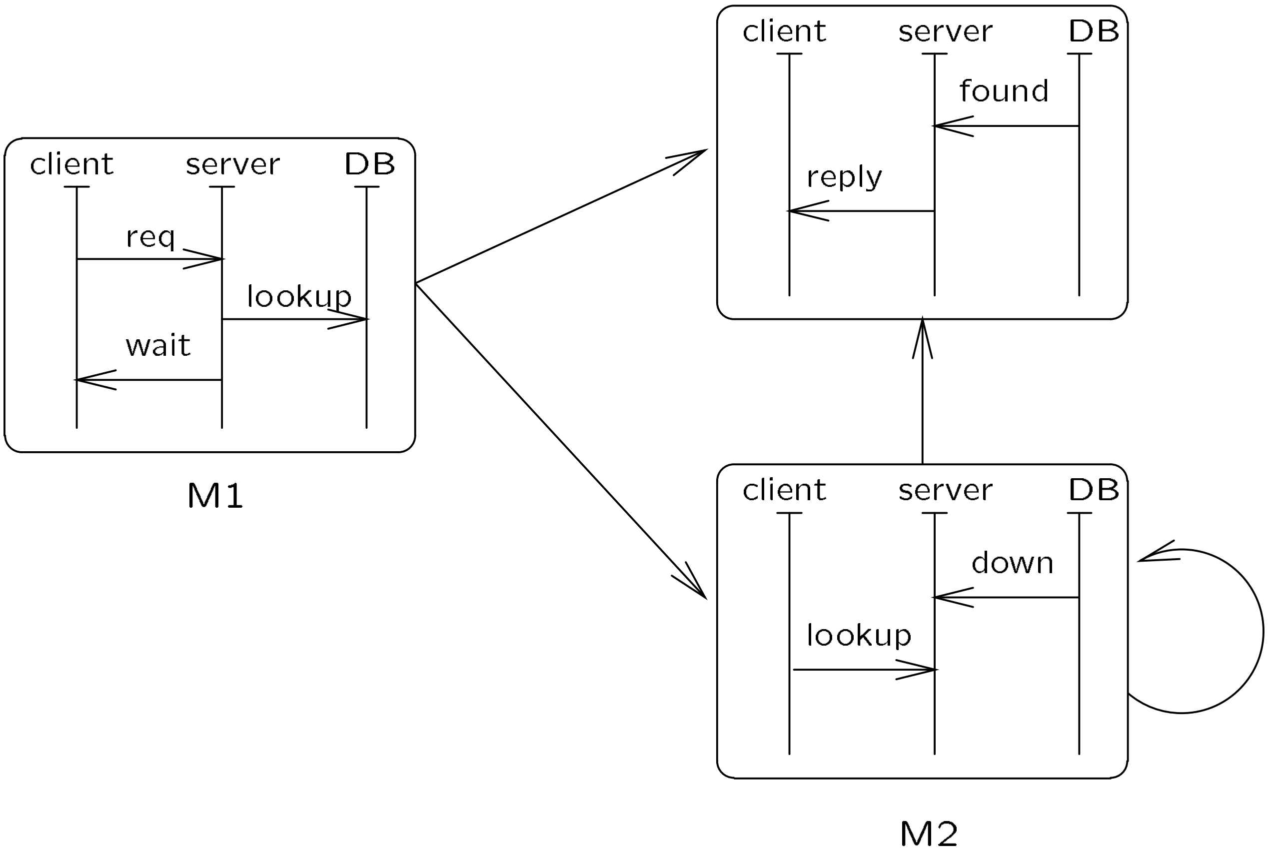
maximal path ...

$$s_1 \longrightarrow s_2 \longrightarrow \ldots$$

... defines an execution MSC

$$c(s_1) \circ c(s_2) \circ \ldots$$

HMSC (cont.)

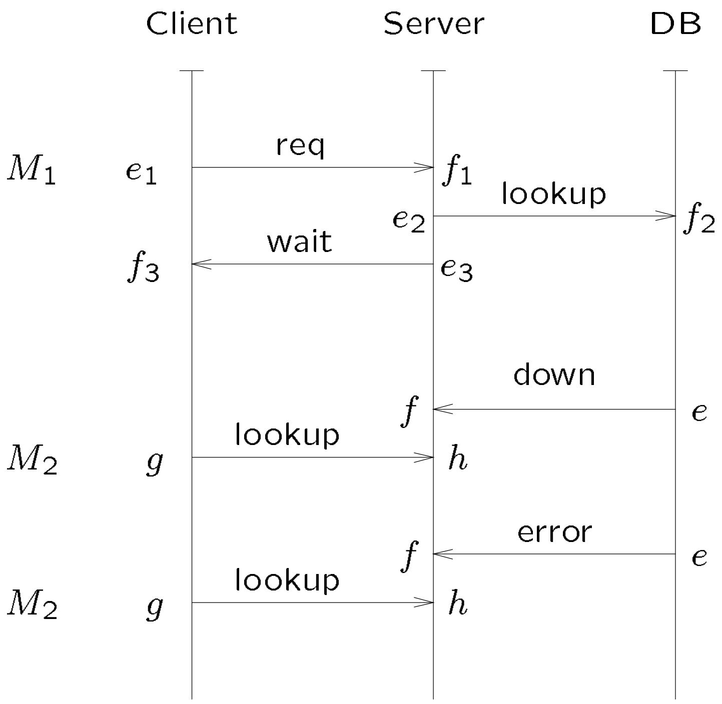


$M_1 \circ M_2$

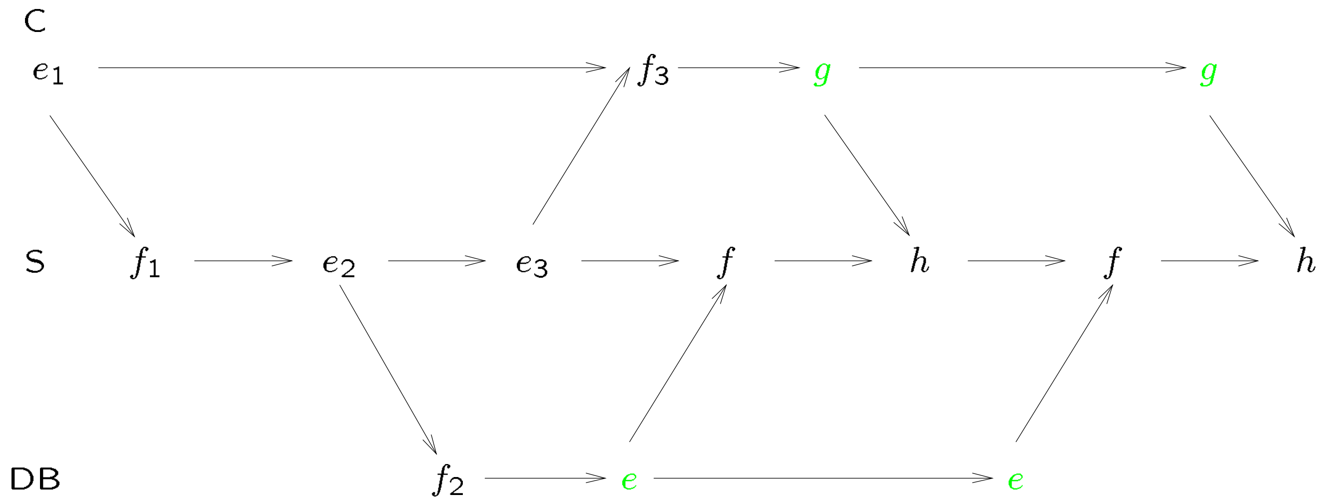
- synchronized execution: all events of M_1 **before** events of M_2 (global condition) $\longrightarrow$ finite state space

- unsynchronized execution: $\longrightarrow$ infinite state space

HMSC (cont.): $M_1 \circ M_2 \circ M_2$



partial order



Ex. linearization (sequence of events)

$$e_1 f_1\, e_2 f_2\, e_3 f_3\, eg\, eg \;\Big|\; fh\, fh$$

Metasemantics

What do we need for specification?

- scenarios: mandatory or optional?
  Damm/Harel '99 live sequence charts
  $\longrightarrow$ hot/cold conditions, events, ...

- universal and existential branching: AND-OR sequence charts
  $\longrightarrow$ overlapping of scenarios

- MSG specifies *either* positive *or* negative properties, but not both

This talk:

- branching is optional

- message passing: asynchronous, FIFO, no message loss, one buffer for each one-way link

Analyzing Specifications

HMSC specifications:

- complex scenarios, might be contradictory

- concurrent behavior, unwanted executions

Need for reliable specification...

Can we apply automatic techniques for verification?

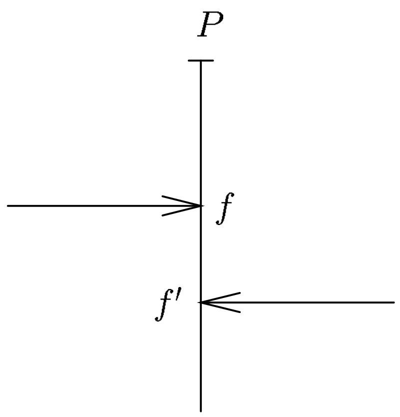
This talk:

- typical verification problems: races, model-checking, synthesis

- limits in verification

Incomplete Specification: Races

System implementation might be less controlled than suggested by the MSC specification



$$f \quad \leq \quad f'$$

From visual order to ...
... the (relaxed) causal order $\preceq$:

$$x \quad \preceq \quad y \qquad \text{implies} \qquad x \quad \leq \quad y$$

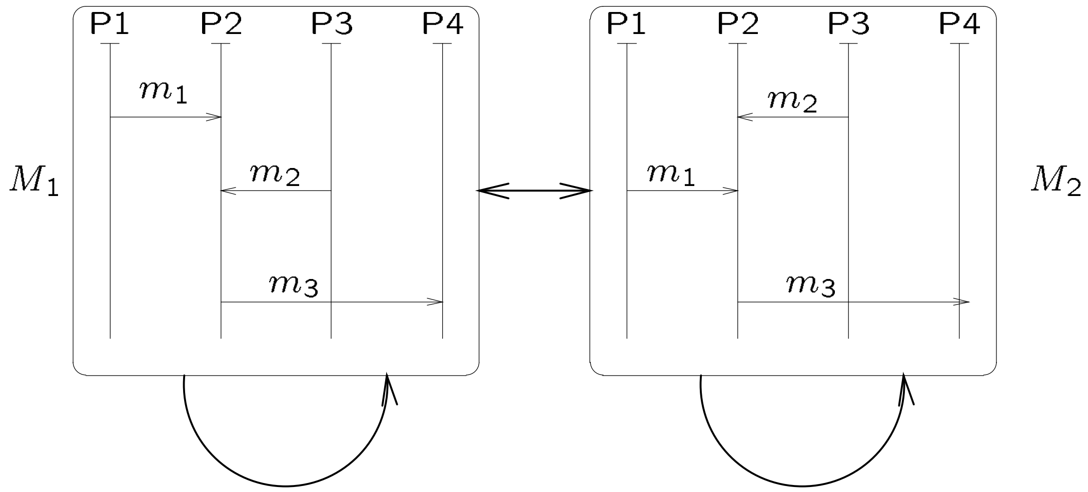
Example: only send events are triggered

process P receives from different processes R, S:
receive events are causally independent

$$f \quad \not\preceq \quad f'$$

Races in HMSC

No races in HMSC:



Example:

- Both M_1 and M_2 contain a race (receives on P_2).

- HMSC has no race (is complete)

$m_1m_2m_3$ and $m_2m_1m_3$ cannot be distinguished by the implementation
(causal order equivalence)

but: they are different for the specification
(visual order)

Races (cont.)

- A single MSC contains races if $\preceq$ is strictly less restrictive than the visual order.

  Checking races in single MSCs:
  quadratic algorithm (Alur et al., 1996)

  Implemented in uBET

- An HMSC contains no races, if it is complete under causal equivalence.

  Example:

  $(m_1 m_2 m_3)(m_1 m_2 m_3) \cdots$ equivalent to $(m_2 m_1 m_3)(m_2 m_1 m_3) \cdots$

M./Peled '99:

There is no algorithm for detecting races in HMSC.

Model checking

- specification HMSC M

- property: given by HMSC (positive/negative) or temporal logic formula

Model checking:

- positive property P

$$M \subseteq P$$

- negative property N

$$M \cap N \neq \emptyset$$

Example: can it happen that after 2 error messages, there is still no error reporting?

Alur/Yannakakis, M./Peled '99:

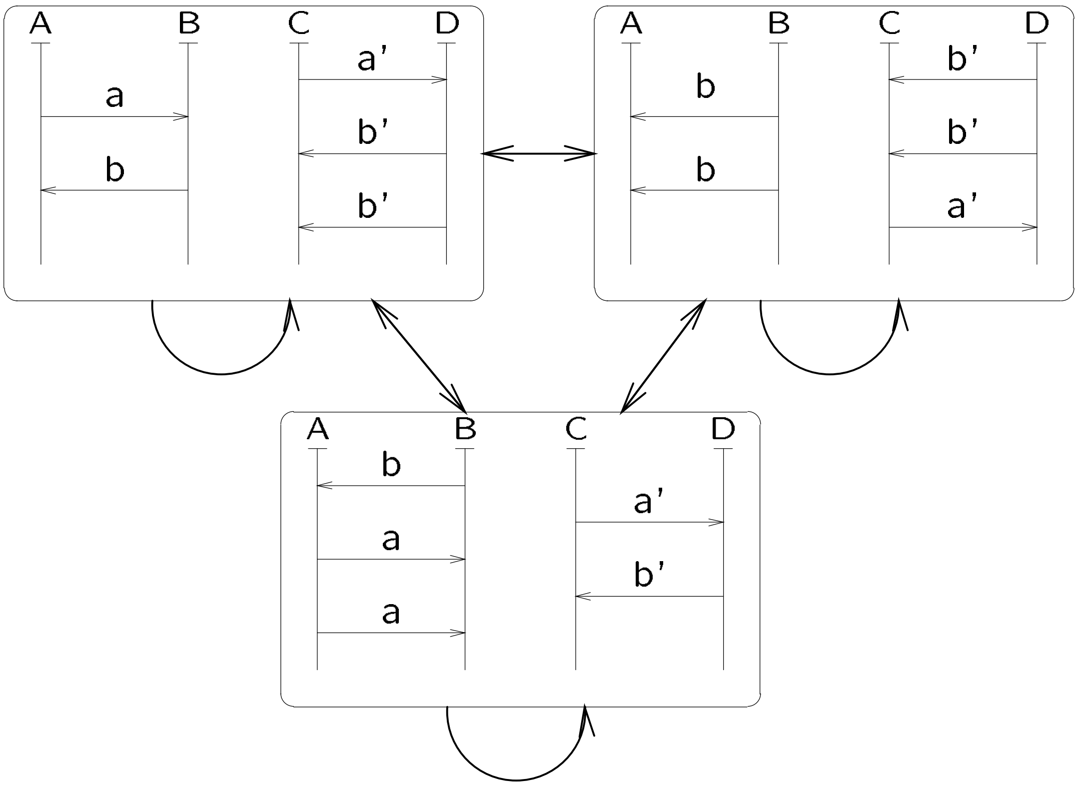
There is no algorithm for model checking HMSC properties.

Model checking (cont.)

Post correspondence problem

$$(ab, abb), (bb, bba), (baa, ab)$$

HMSC



positive property

$$((a+b)(a'+b'))^*(ab'+ba')(a+b+a'+b')^*$$
$$+(a+b+a'+b')^\omega$$

Limits in verification

- Are HMSCs too expressive?

  No, they are even weaker than communicating state machines (cf. synthesis problem).

- Can we decide any reasonable property about HMSC specifications?

  Yes, by asking the right questions.

Rule: don't ask questions which discriminate between different linearizations of the executions.

Good questions:

- Matching with gaps: weak model checking negative properties.

- Property formulated with partial order temporal logics.

Partial order properties

Example: can it happen that after 2 error messages, there is still no error reporting?

$N = \ldots$ error (no report)* error (no report)*

N partial order property,
M HMSC specification.

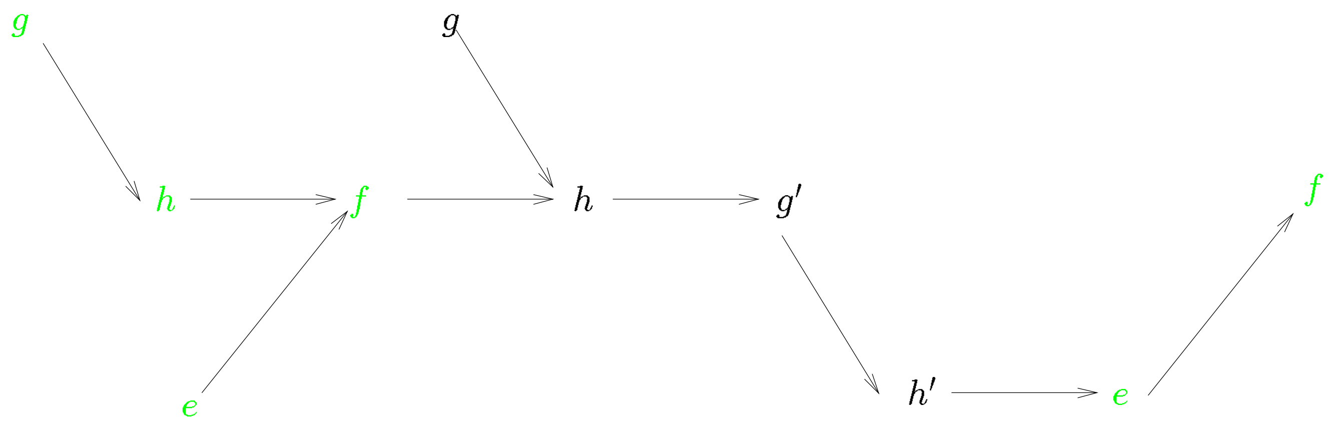
Question:

$$N \cap M \stackrel{?}{=} \emptyset$$

Algorithm: Ignore partial order of M and view M as a transition system (e.g., topological sort of events in every MSC).

Partial order properties (cont.)

- Matching with gaps:



Requires that g, h, e, f, e, f occur (scattered) + causality between them

- Partial order logics:

  TLC, temporal logic of causality (Alur, Peled, Penczek '95) $\longrightarrow$ model-checking in exponential-time (PSPACE-complete)

  TrLTL, μ-calculus (Walukiewicz, '96/'99)

Finite-state HMSC

Problem: General HMSC are infinite-state systems (asynchronous communication, concurrency)

Observation: Many HMSC specifications are finite-state.

Solution: Syntactic restriction of HMSC guaranteeing finite-state space $\longrightarrow$ bounded communication.

Communication graph of single MSC:

- vertices: processes
- edges: $P \longrightarrow Q$, if some message from P to Q

Finite-state HMSC (cont.)

(Alur/Yannakakis '99, M./Peled '99)

HMSC M has bounded communication:
for every loop of M:

$$s_1 \longrightarrow s_2 \longrightarrow \cdots s_n \longrightarrow s_1$$

communication graph of

$$c(s_1) \circ c(s_2) \circ \cdots \circ c(s_n)$$

has at most one nontrivial strongly connected component.

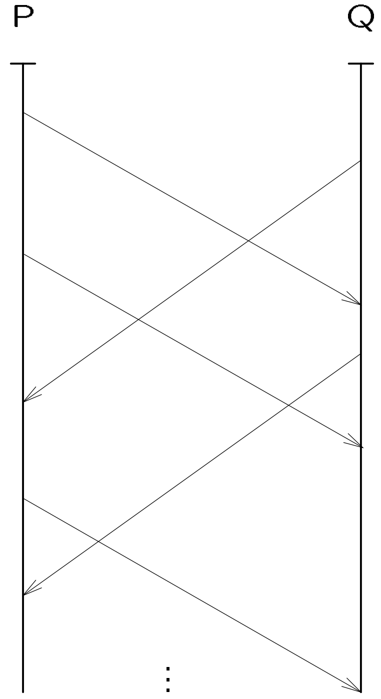
(M./Peled '99)

It can be checked in exponential time whether a HMSC has bounded communication (NP-complete).

Detecting races and model-checking for bounded communication HMSCs can be done in double exponential time (EXPSPACE-complete).

Synthesis of HMSCs

Non HMSC-realizable communication protocol:



Input: finite-state machine A over events $P!Q$ (P sends to Q), $P?Q$ (P receives from Q)

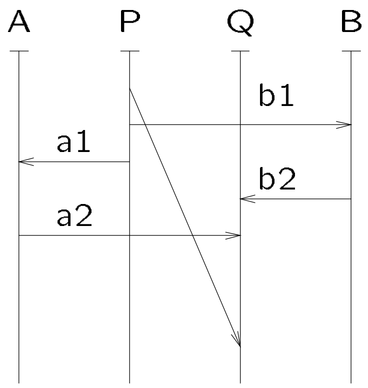
Output: HMSC M equivalent to $L(A)$ (if $L(A)$ is realizable as HMSC)

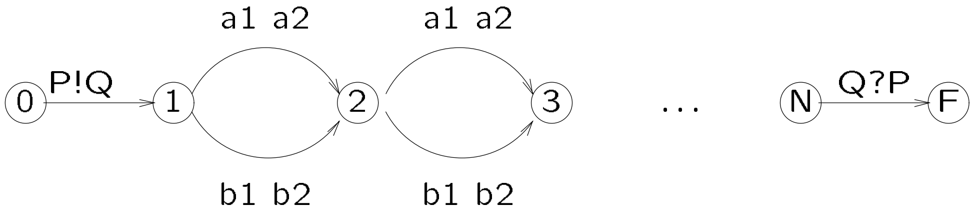
HMSC-realizability (cont.)

Algorithm:

- check whether $L(A)$ is generated by a finite set of MSCs F
- combine $L(A)$ with F to an HMSC

Worst-case: exponential time/size





Conclusion

- Expressiveness: negation is missing

- Verification: extensions to infinite state spaces?

- Need for suitable partial order logic