Refinement II

Inherent nondeterminism versus underspecification

And weak sequencing

Ketil Stølen

Outline

- Two kinds of nondeterminism
  - Underspecification
  - Inherent (explicit) nondeterminism
- The need for both alt and xalt
- Semantics in the general case
- Refinement in the general case

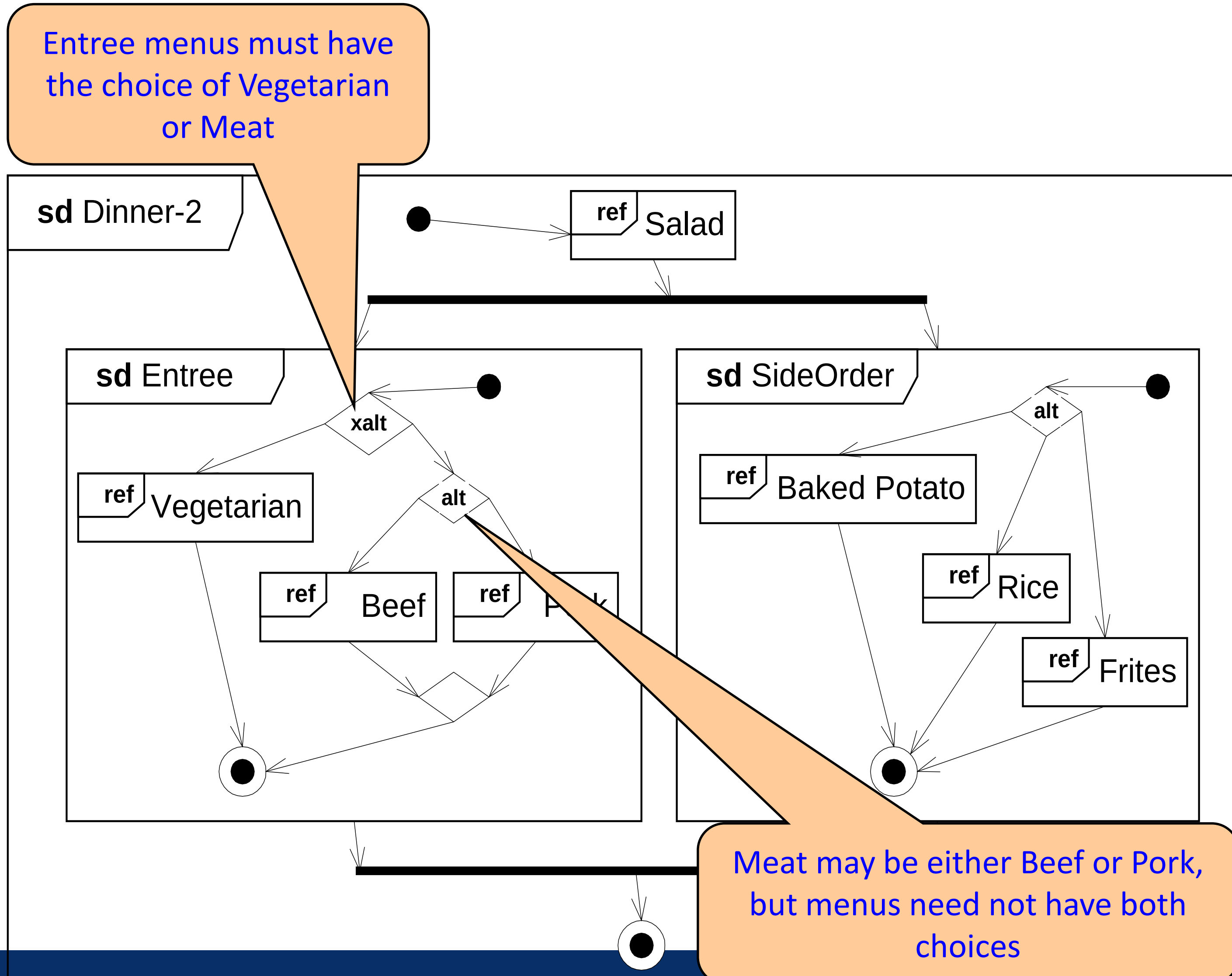
Underspecification and inherent nondeterminism

- Underspecification:
  - Several alternative behaviours are considered equivalent (serve the same purpose)
- Inherent nondeterminism:
  - Alternative behaviours that must all be possible for the implementation
- These two should be described differently!

The need for both **alt** and **xalt**

- Potential non-determinism captured by **alt** allows abstraction and inessential non-determinism
- Inherent or explicit non-determinism captured by **xalt** characterizes non-determinism that must be reflected in every correct implementation in one way or another

Restaurant example with both **alt** and **xalt**



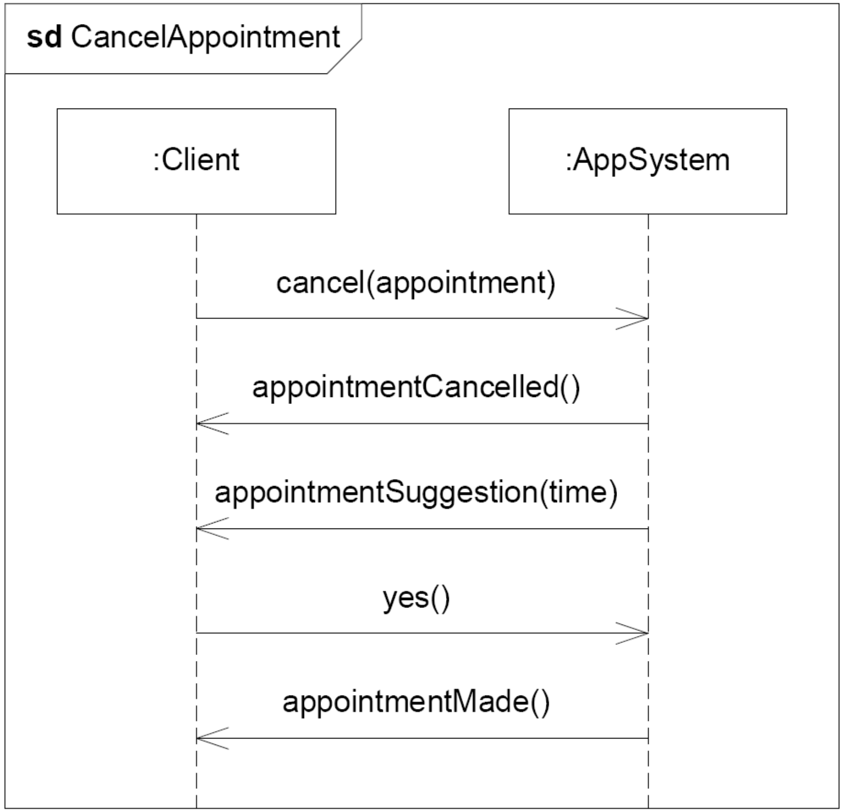
Example: an appointment system

- A system for booking appointments used by e.g. dentists

- Functionality:
  - MakeAppointment: The client may ask for an appointment
  - CancelAppointment: The client may cancel an appointment

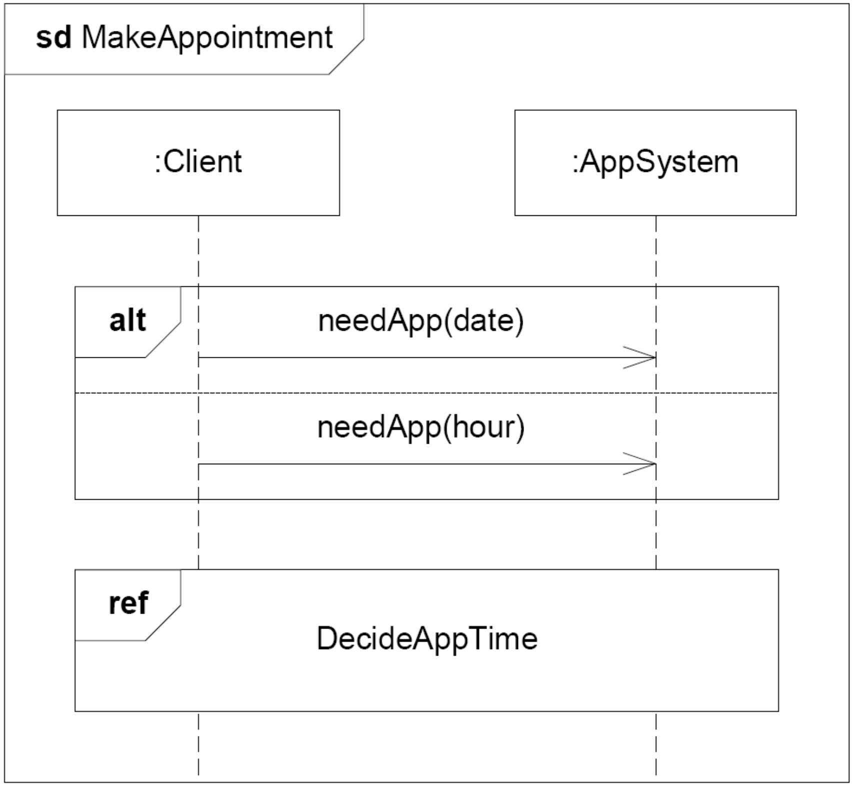
CancelAppointment

- This specification has two positive traces
- Whether reception of appointmentCancelled() occurs before or after sending of appointmentSuggestion(...) is not important
- Underspecification due to weak sequencing



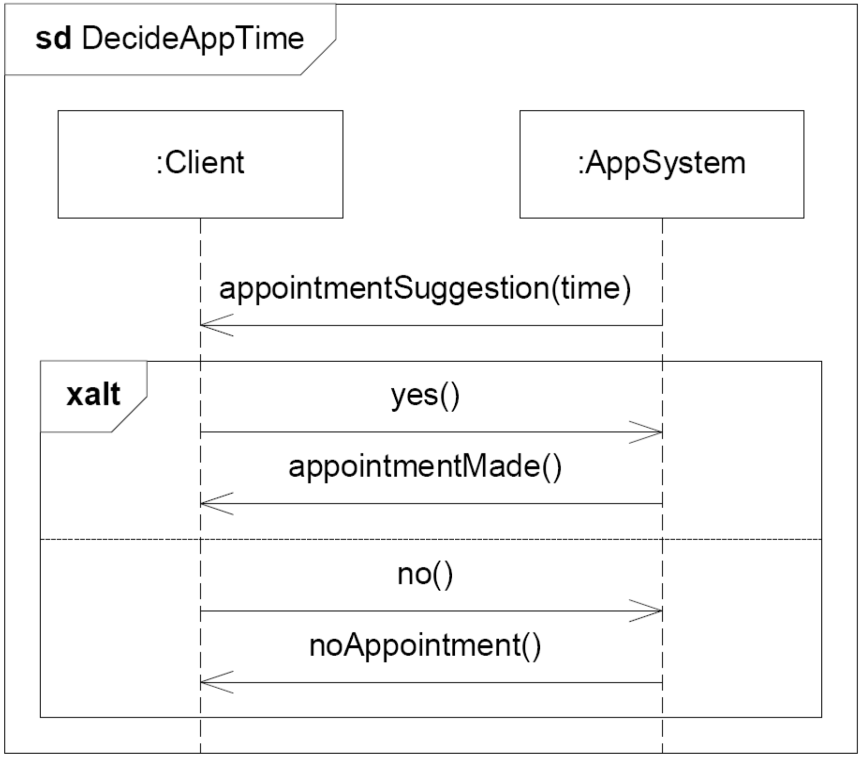
MakeAppointment

- May ask for either a specific date or a specific hour of the day (e.g. in the lunch break)
- The system is not required to offer both alternatives
- Underspecification expressed by the **alt** operator



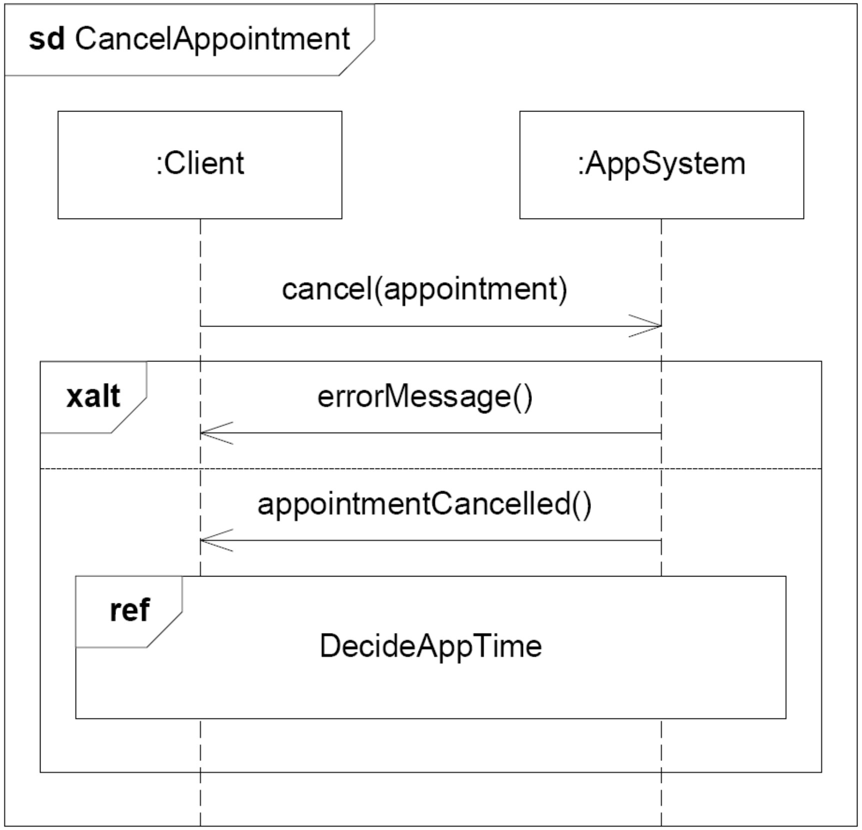
DecideAppTime

- The system must be able to handle *both* yes() and no() as reply messages from the client
- This is *not* underspecification
- Therefore the alternatives are expressed by the **xalt** operator



CancelAppointment - revised

- The condition for choosing errorMessage() or appointmentCancelled() is not shown
- Both alternatives should be possible
- The choice is made by the system



Use of **alt** versus **xalt**

The crucial question when specifying alternatives:

- Do these alternatives represent similar traces in the sense that implementing only one is sufficient?

When to use **alt**

- Use **alt** to specify alternatives that represent similar traces i.e. to model
  - underspecification

When to use **xalt**

- Use **xalt** to specify alternatives that must all be present in an implementation, i.e. to model
  - inherent nondeterminism, as in the specification of a coin toss or a password generator
  - alternative traces due to different inputs that the system must be able to handle (as in DecideAppTime)
  - alternative traces where the conditions for these being positive are abstracted away (as in revised version of CancelAppointment)

Semantics

So far I have told you that the semantics of a sequence diagram is an interaction obligation

- In the general case, this is not sufficiently expressive

Semantics – general case

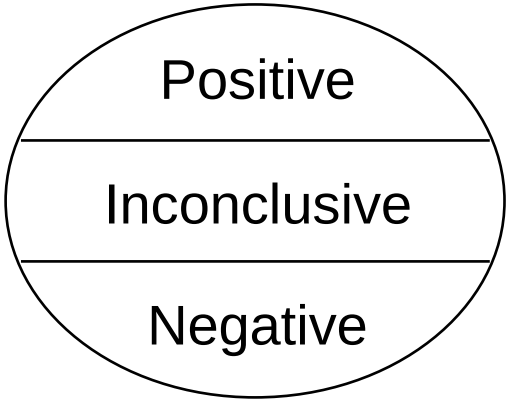
- The semantics of a sequence diagram *without occurrences* of **xalt** is a set of a single interaction obligation

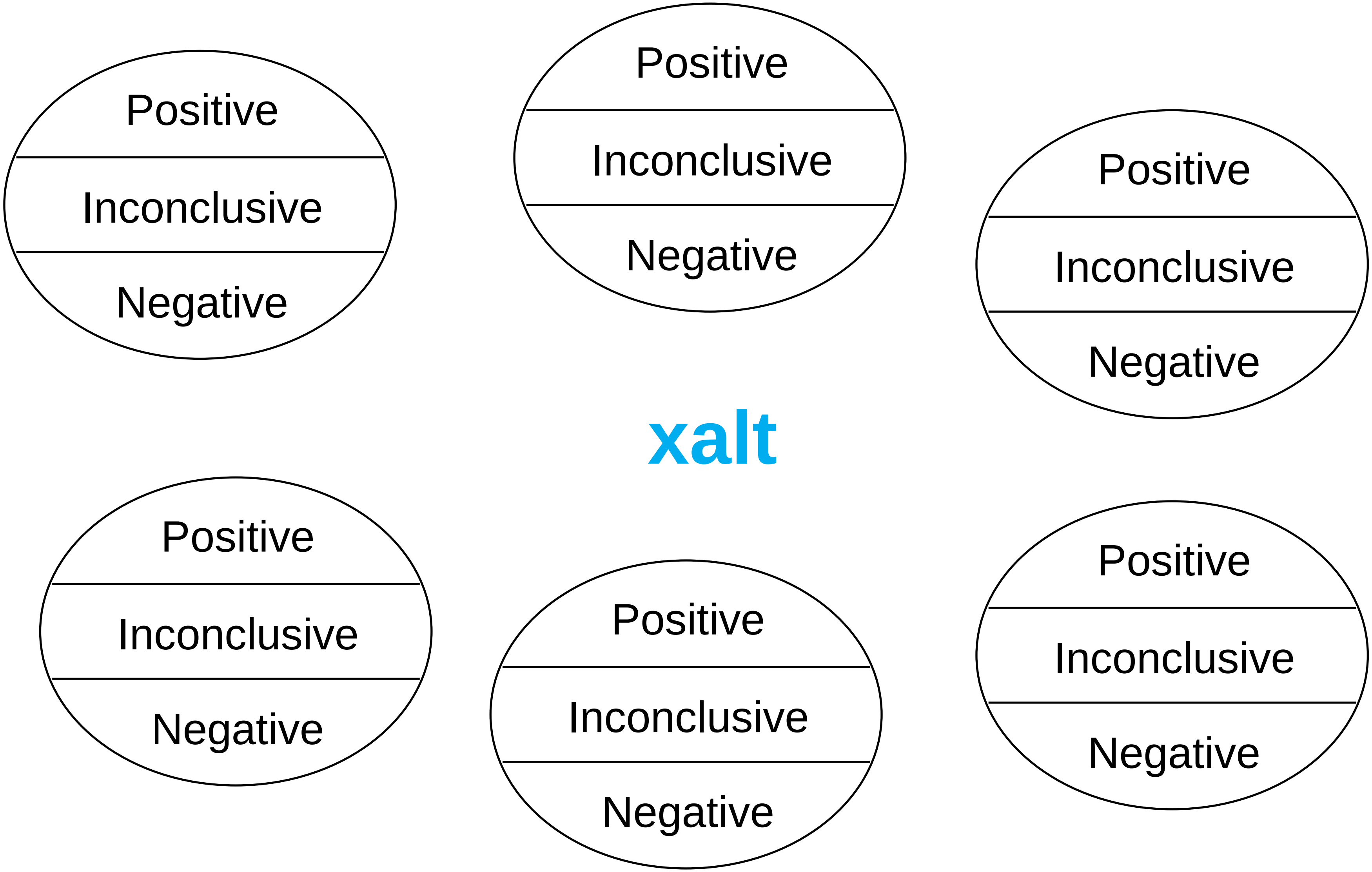
  { (p,n) }

- The semantics of a sequence diagram *with occurrences* of **xalt** is a set of arbitrarily many interaction obligations

  { (p1,n1), (p2,n2), … , (pK,nK) }

alt




Positive
Inconclusive
Negative
Positive
Inconclusive
Negative
Positive
Inconclusive
Negative
xalt
Positive
Inconclusive
Negative
Positive
Inconclusive
Negative
Positive
Inconclusive
Negative

As before

For any sequence diagram d, [[d]] denotes its semantics

We may think of [[ ]] as a function of the following type

- [[ ]]: **SequenceDiagram → Set of InteractionObligation**

$\uplus$ - the inner union operator

The inner union of two interaction obligations yields the interaction obligation whose

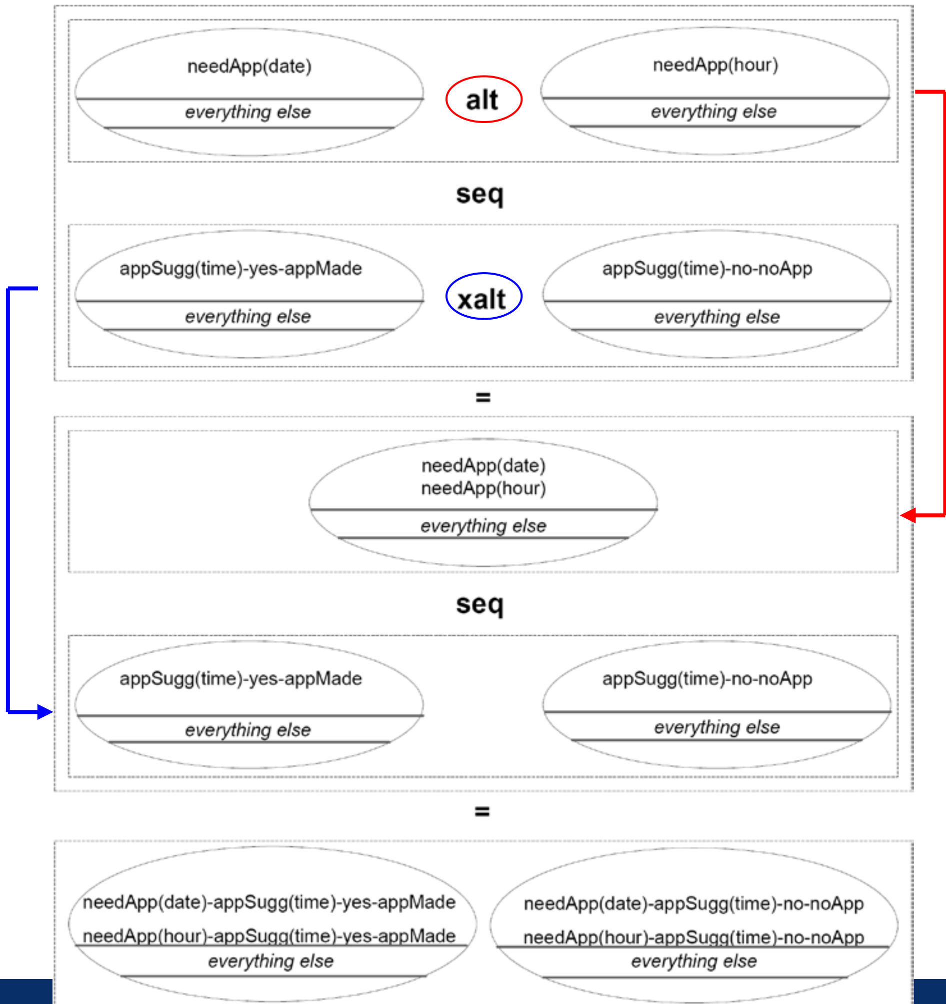
- positive set = the union of the argument's positive sets
- negative set = the union of the argument's negative sets

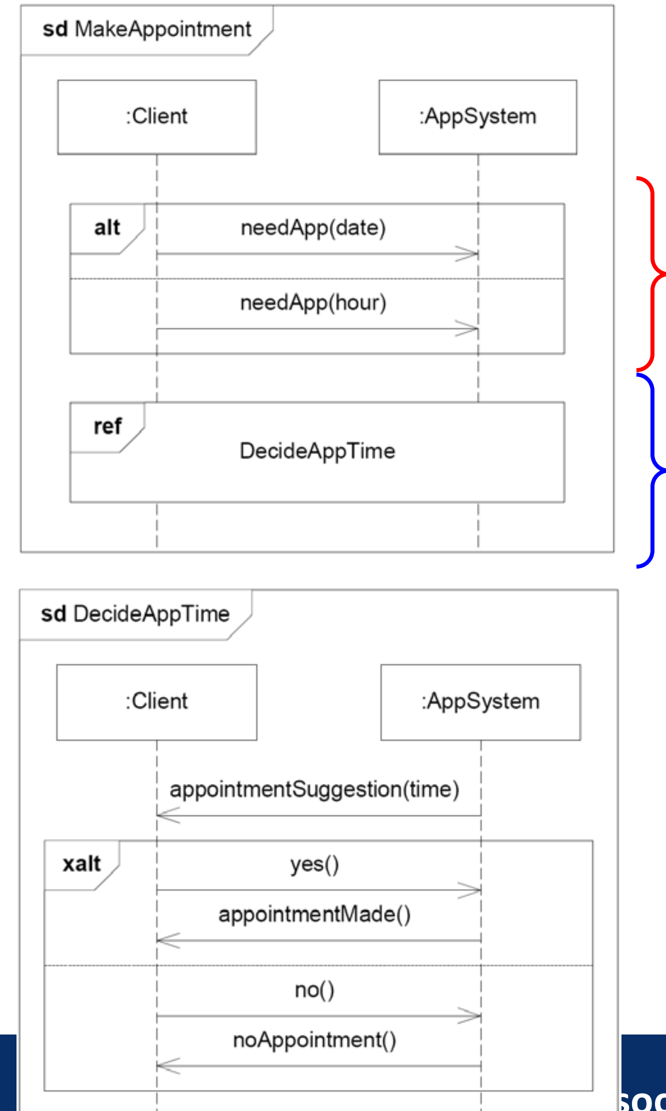
$$(p_1, n_1) \uplus (p_2, n_2) \stackrel{\text{def}}{=} (p_1 \cup p_2, n_1 \cup n_2)$$

Formal semantics of **alt** and **xalt**

- **alt** joins interaction obligations:
  - $[[d_1 \text{ alt } d_2]] \stackrel{\text{def}}{=} \{o_1 \uplus o_2 \mid o_1 \in [[d_1]] \wedge o_2 \in [[d_2]]\}$
- **xalt** keeps the interaction obligations:
  - $[[d_1 \text{ xalt } d_2]] \stackrel{\text{def}}{=} [[d_1]] \cup [[d_2]]$

Informal illustration of MakeAppointment





Sequential composition

Basic rules

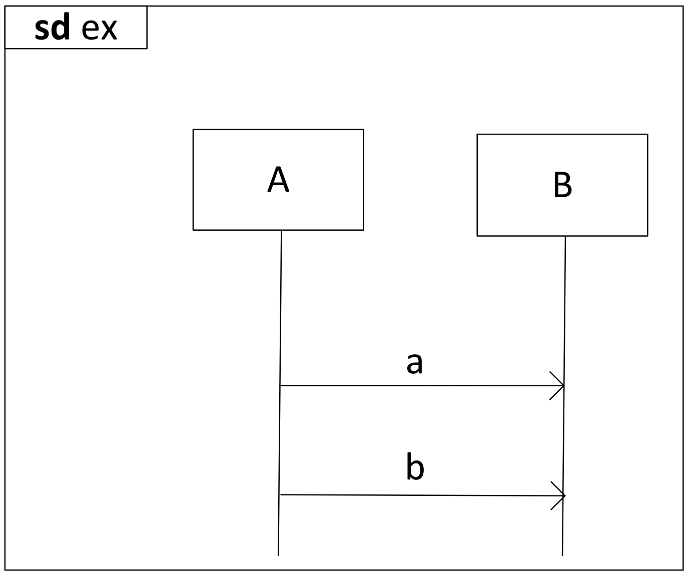
Causality

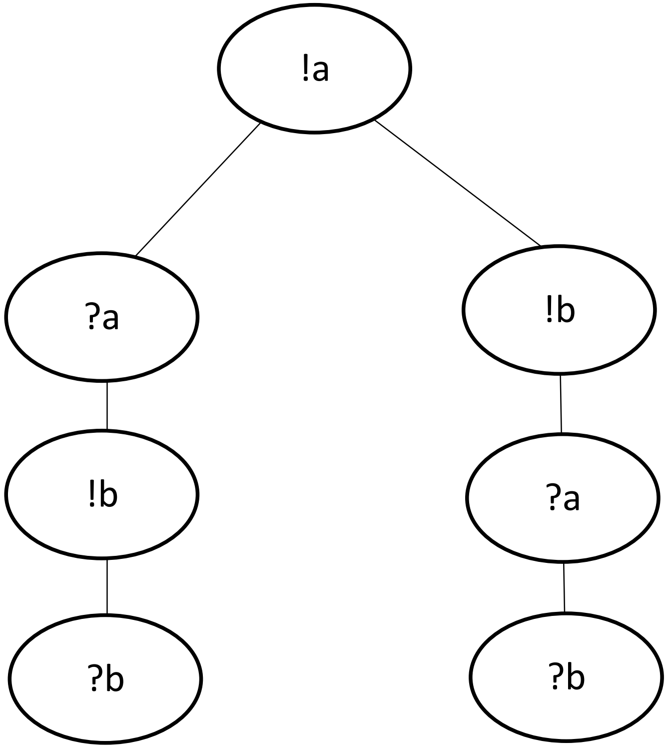
- a message can never be received before it has been transmitted
- the transmission event for a message is therefore always ordered before the reception event for the same message

Weak sequencing

- events from the same lifeline are ordered in the trace in the same order as on the lifeline (from top to bottom)

Example





Mathematically !a and ?a (etc.) are shorthands for !(a,A,B) and ?(a,A,B)
Hence, each event contains the names of its sending and receiving lifelines

Sequential composition of trace sets s_1 and s_2

$$s_1 \succsim s_2$$

=

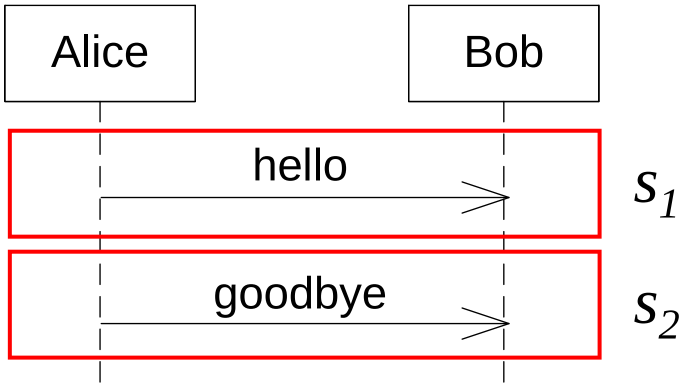
the set of all traces obtained by merging traces

t_1 from s_1 and t_2 from s_2

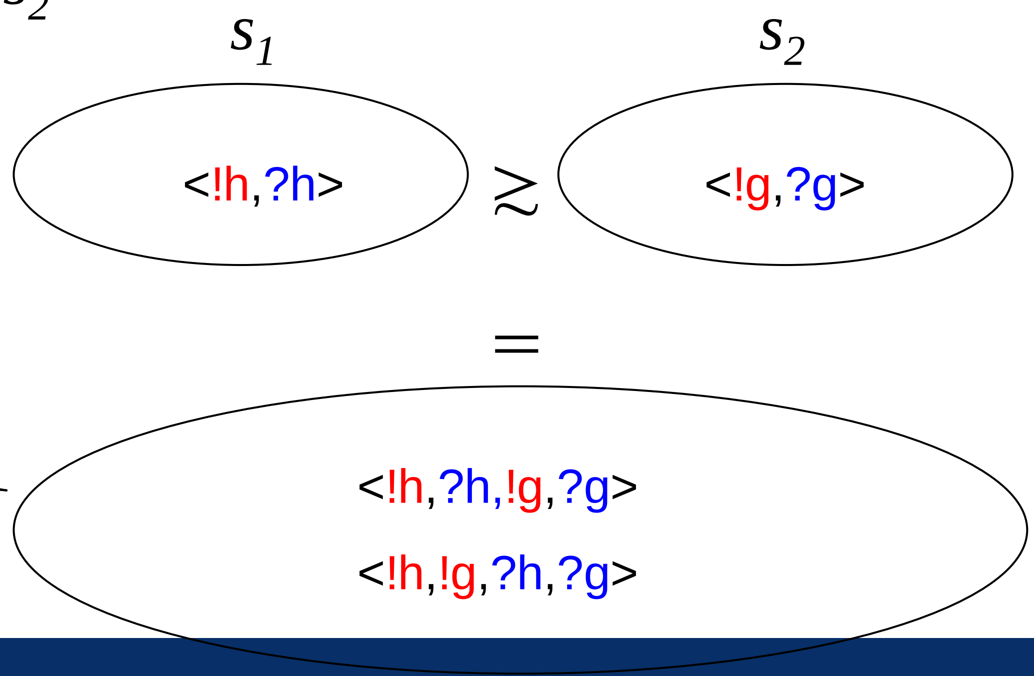
in such a way that for each lifeline,

the events from t_1 comes before the events from t_2

Sequential composition of trace sets



Red events occur on Alice, blue events on Bob



$s_1 \gtrsim s_2$ is the set of positive traces for the diagram

Note

- if s_1 or s_2 is empty then $s_1 \succsim s_2$ is also empty

Sequential composition of interaction obligations

- $(p_1,n_1) \succsim (p_2,n_2) \stackrel{\text{def}}{=} (p_1 \succsim p_2\ ,(n_1 \succsim p_2) \cup (n_1 \succsim n_2) \cup (p_1 \succsim n_2))$

- Traces composed exclusively by positive traces become positive
- Traces composed with at least one negative trace become negative

Formal semantics of **seq**

- $[[d_1 \text{ seq } d_2]] \stackrel{\text{def}}{=} \{o_1 \succsim o_2 \mid o_1 \in [[d_1]] \wedge o_2 \in [[d_2]]\}$

- o_i is shorthand for (p_i, n_i)

Remember: By sequential composition

- positive followed by positive is positive
- positive followed by negative is negative
- negative followed by negative is negative
- negative followed by positive is negative