Refinement II

Ketil Stølen

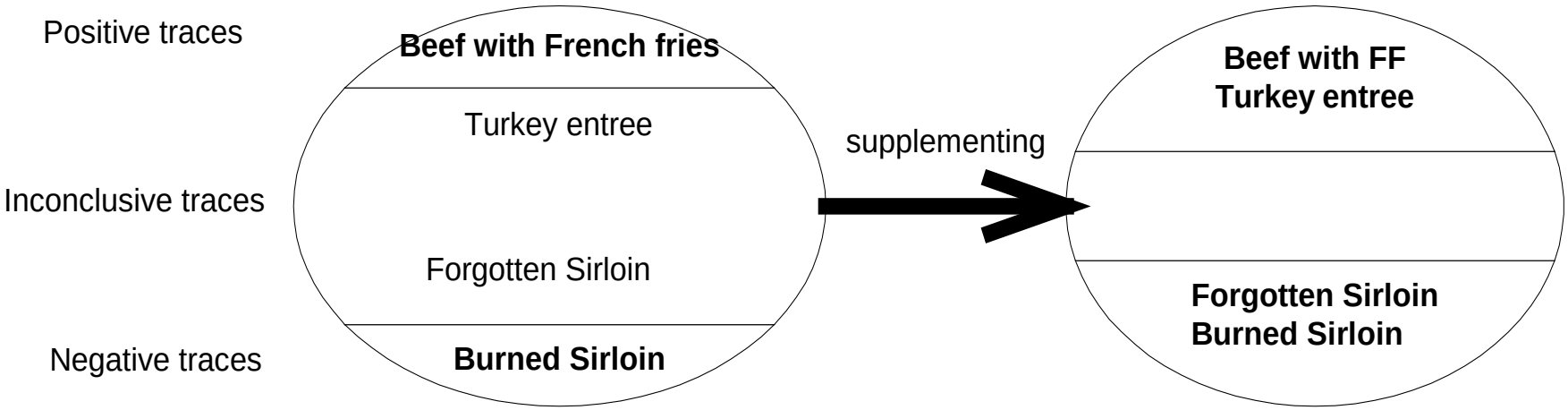
Outline

- Refinement summarized
- Inherent non-determinism (also called explicit non-determinism)

Refinement summarized

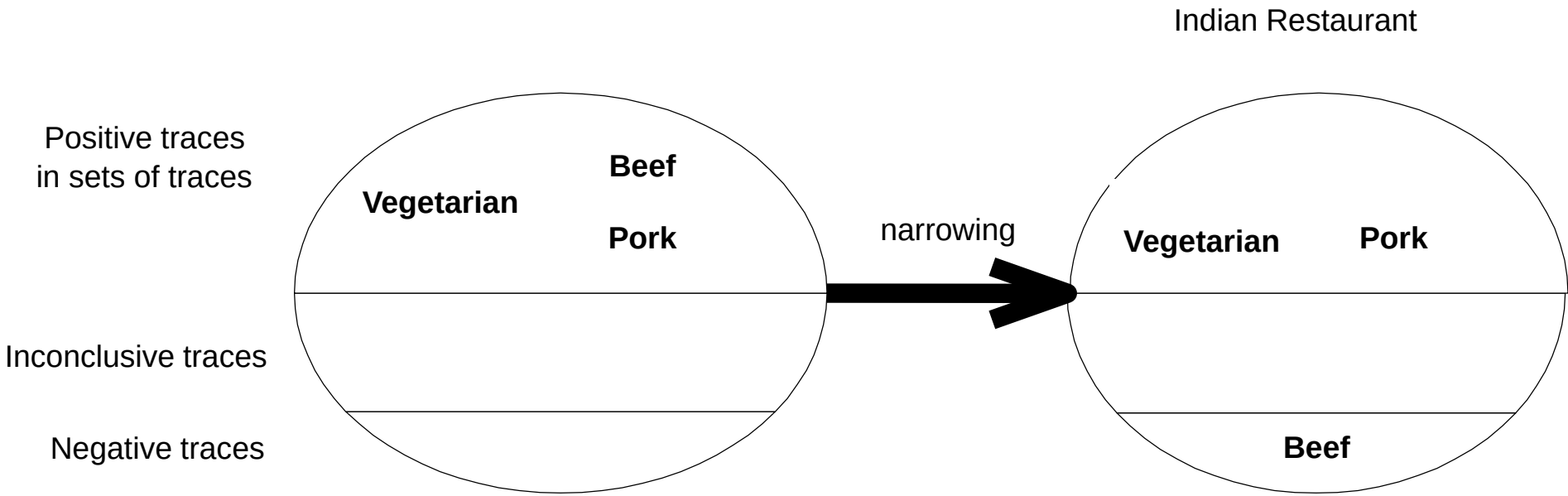
Supplementing

- Supplementing involves reducing the set of inconclusive traces by redefining inconclusive traces as either positive or negative
  - Positive trace remains positive
  - Negative trace remains negative



Narrowing

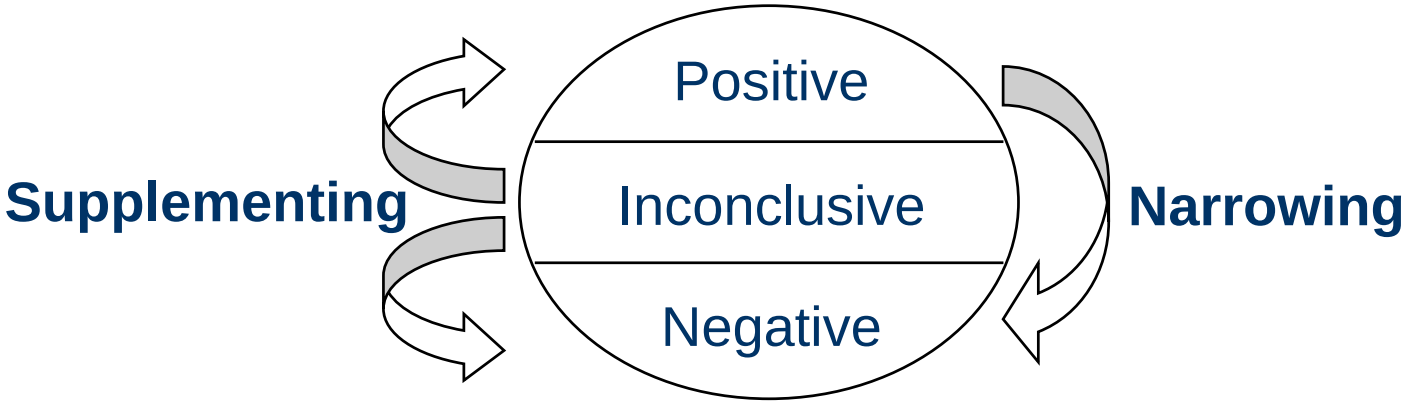
- Narrowing involves reducing the set of positive traces by redefining them as negative
  - Inconclusive traces remain inconclusive
  - Negative traces remain negative



Direct definition of refinement

- A sequence diagram B is a refinement of a sequence diagram A if
  - every trace classified as negative by A is also classified as negative by B
  - every trace classified as positive by A is classified as either positive or negative by B

Refinement formalized



- An interaction obligation o'=(p',n') is a refinement of an interaction obligation o=(p,n) iff
  - n $\subseteq$ n'
  - p $\subseteq$ p'Un'

Inherent non-determinism

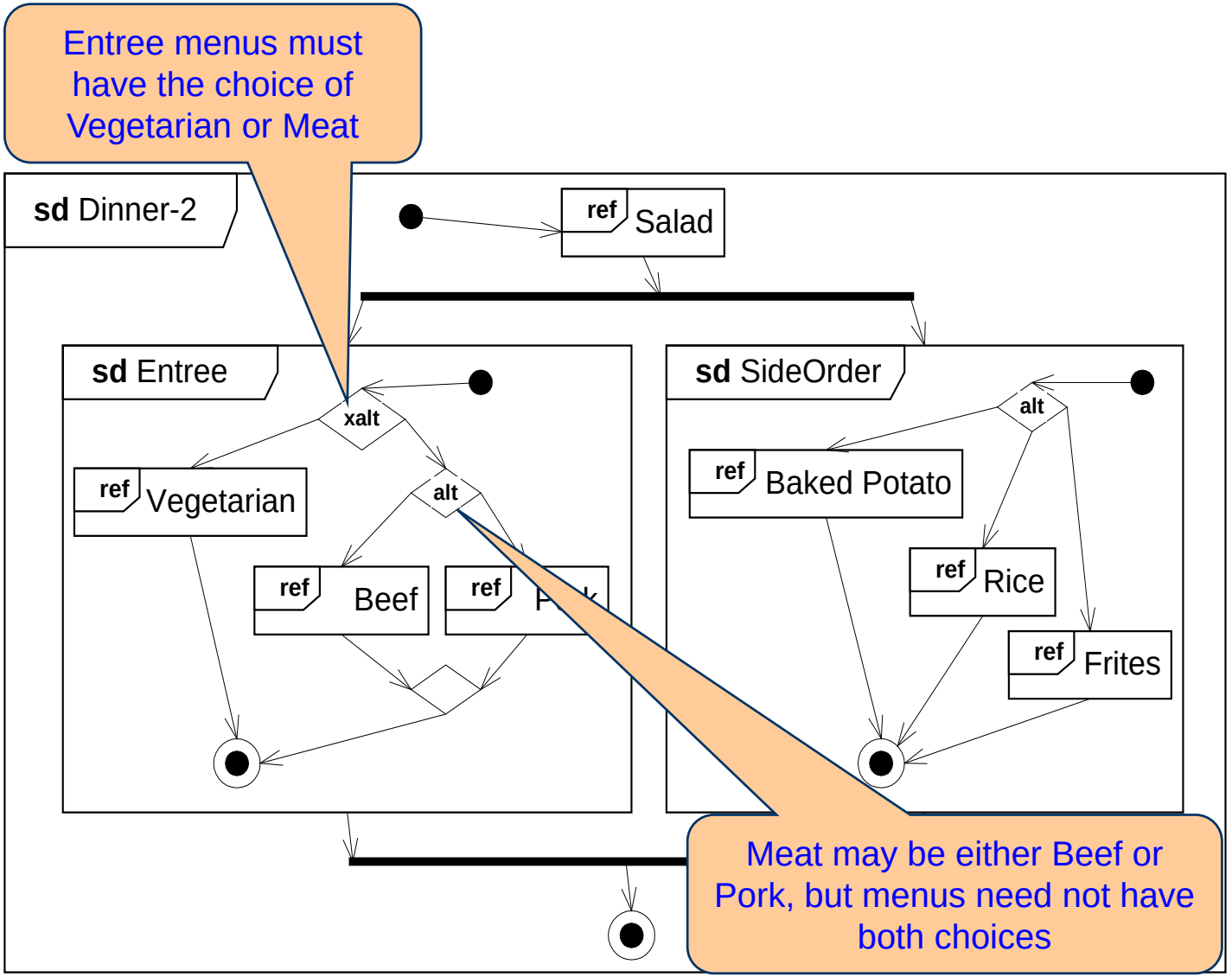
Underspecification and inherent non-determinism

- Underspecification: Several alternative behaviours are considered equivalent (serve the same purpose)
- Inherent non-determinism: 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
  - Under-specification
  - Non-critical design decisions may be postponed
- Inherent or explicit non-determinism captured by xalt characterizes non-determinism that must be reflected in every correct implementation in one way or another
  - Makes it possible to specify games
  - Important in relation to security
  - Also helpful as a means of abstraction

Restaurant example with both alt and xalt



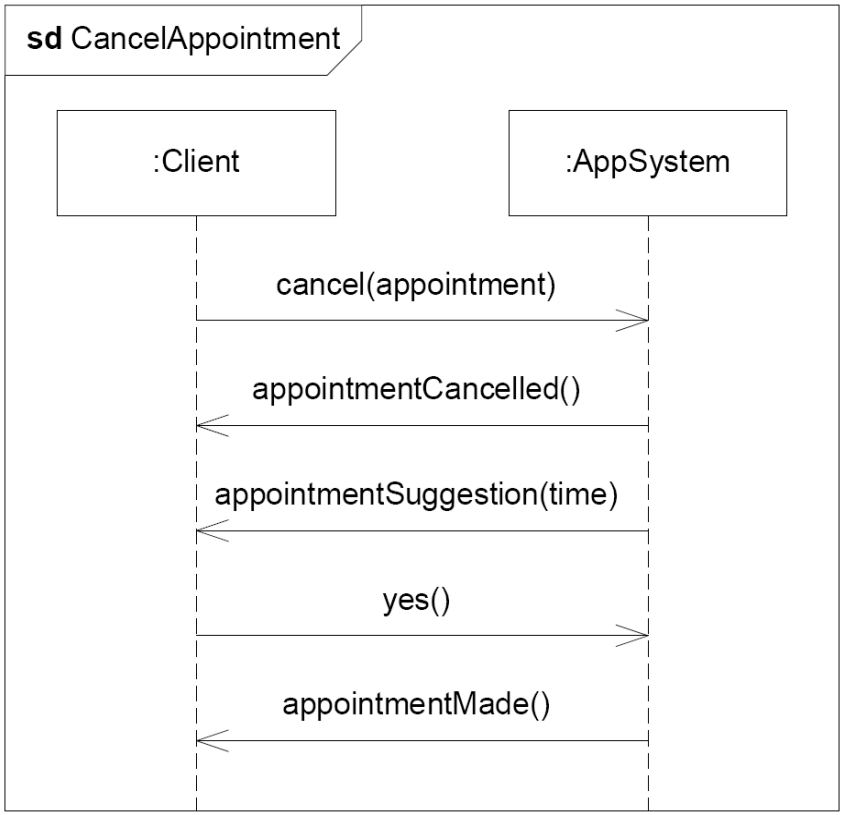
Example: an appointment system

- 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
  - Payment: The system may send an invoice message asking the client to pay for the previous or an unused appointment.

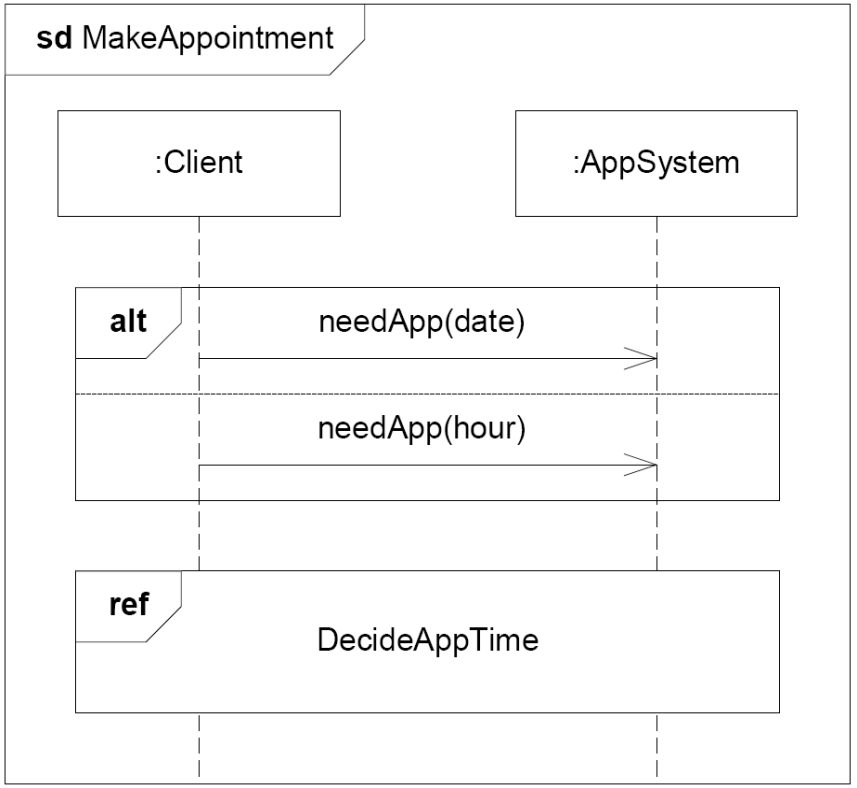
xalt vs alt (1): CancelAppointment

- 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



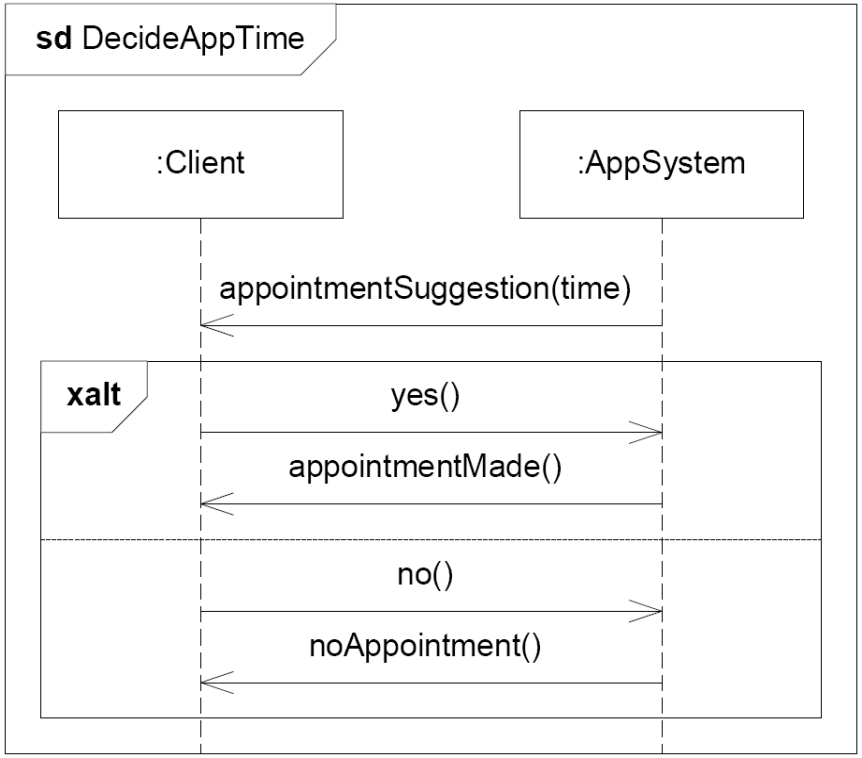
xalt vs alt (2): MakeAppointment

- 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



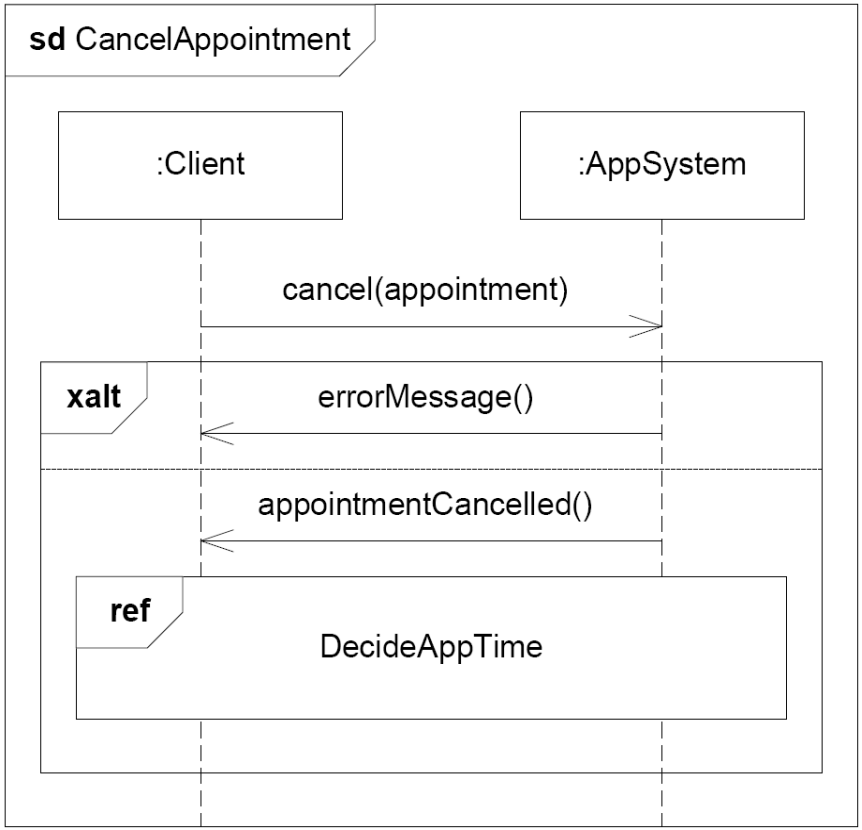
xalt vs alt (3): DecideAppTime

- 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



xalt vs alt (4): CancelAppointment

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



xalt vs alt (5)

- A third use of xalt: to specify inherent nondeterminism
  - for example when specifying a password generator

- The crucial question when specifying alternatives: Do these alternatives represent similar traces in the sense that implementing only one is sufficient?
  - if yes, use alt
  - otherwise, use xalt

The pragmatics of alt vs xalt

- Use alt to specify alternatives that represent similar traces, i.e. to model
  - underspecification
- 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
  - 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 CancelAppointment on slide 12)

Semantics – general case

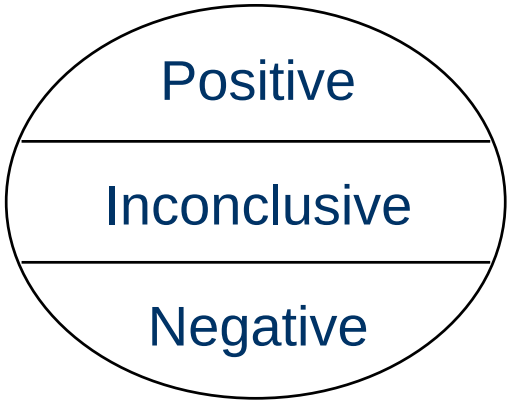
- The semantics of a sequence diagram *without occurrences* of **xalt** is a set of a single interaction obligation

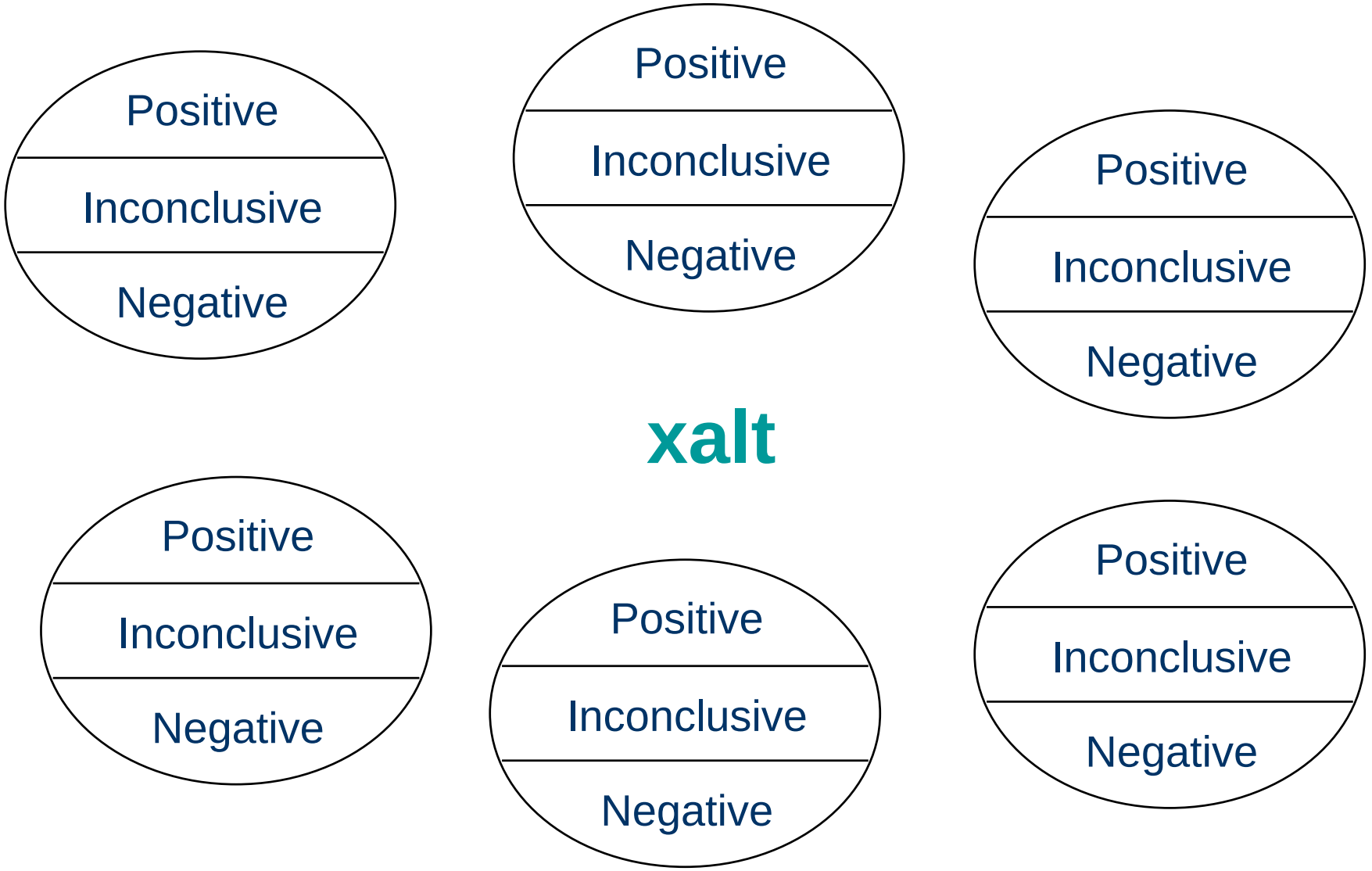
  {(p,n)}

- 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

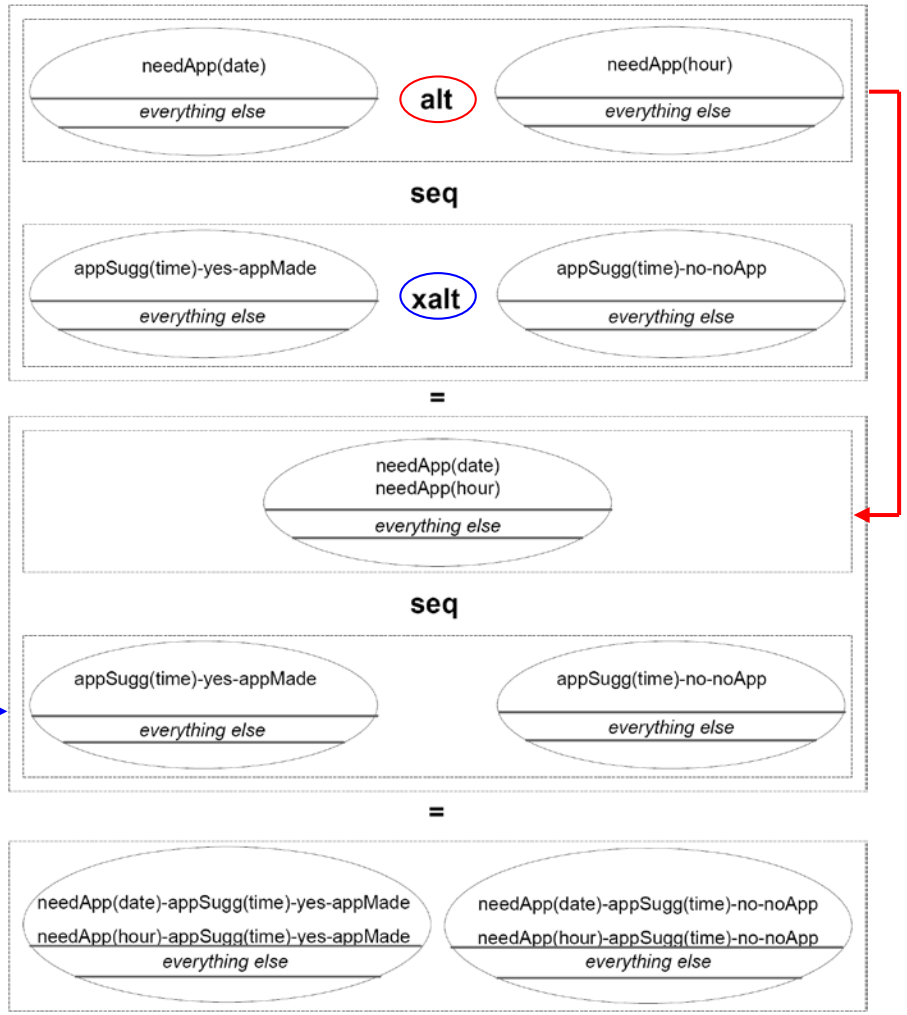
Notational convention

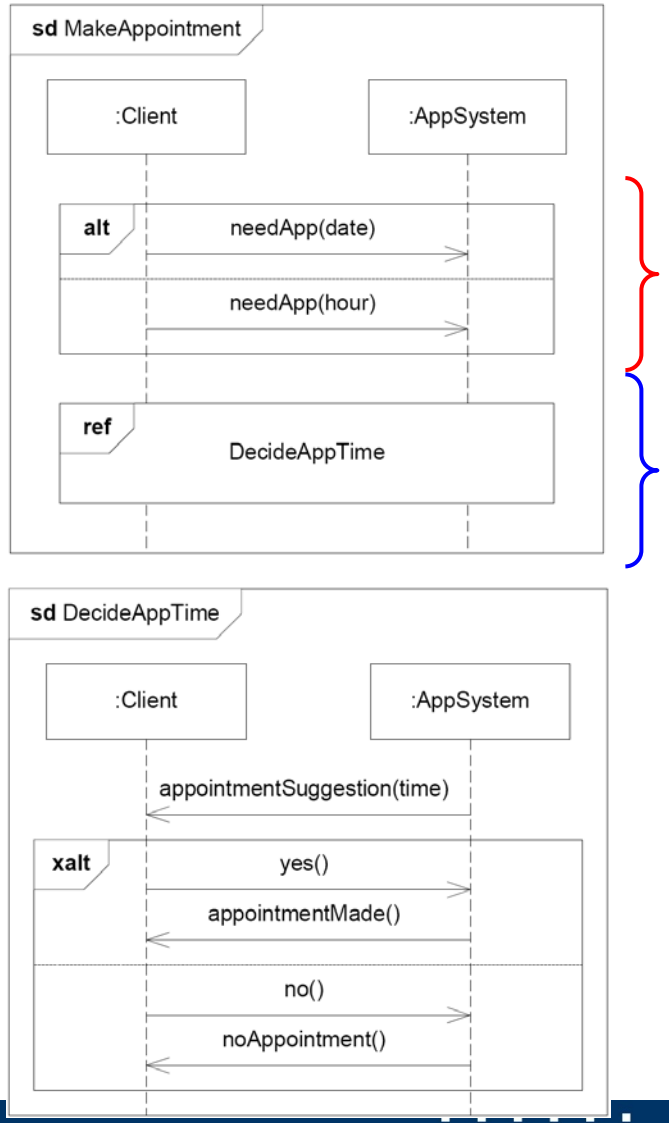
- For any sequence diagram d, [[d]] denotes its sematics
- We may think of [[ ]] as a function of the following type
- [[ ]]: **SequenceDiagram → Set of InteractionObligation**

Formal semantics of alt and xalt

- Alt combines interaction obligations:
  - $[[d_1 \text{ alt } d_2]] \stackrel{\text{def}}{=} \{o_1 \uplus o_2 \mid o_1 \in [[d_1]] \wedge o_2 \in [[d_2]]\}$
    - Inner union of interaction obligations $\uplus$:
    - $(p_1,n_1) \uplus (p_2,n_2) \stackrel{\text{def}}{=} (p_1 \cup p_2, n_1 \cup n_2)$
- Xalt results in distinct interaction obligations:
  - $[[d_1 \text{ xalt } d_2]] \stackrel{\text{def}}{=} [[d_1]] \cup [[d_2]]$

Informal illustration of MakeAppointment





Reading on refinement

- Haugen, Husa, Runde, Stølen: *STAIRS towards formal design with sequence diagrams*, 2005. SoSyM, Springer.
- Runde, Haugen, Stølen: *The Pragmatics of STAIRS*, 2006. Springer-Verlag. LNCS 4111.