



## STAIRS – Steps to Analyze Interactions with Refinement Semantics

Øystein Haugen  
with K. Stølen, R. Kobro Runde, K. E. Husa  
(SARDAS project)  
Version 051107



## The art of precision

- In formal semantics
  - the meaning should be expressed so that a machine could check the proofs
    - This rarely happens – not even theoretically
  - or the formulas are so complicated that not even the reviewers dare to believe they are incorrect
    - How often this happens, I am not sure
- In practical standardization politics
  - the meaning must be expressed so vaguely that nobody will object
    - This certainly happens a lot!
  - with the hazard of different persons and different tool interpreting the semantics differently
    - sad, but true
  - and the effect that language constructs are decided without the full understanding of their consequences
    - true, but not as critical as formalists may think



## Committee languages and other mastodons

- The practical/technical dilemma of artificial languages
  - Small language = simple, but hardly expressive enough
    - Pascal, Java, MSC-92
      - to be practical the user must add a large number of libraries
  - Big language = complex, expressive, and non-transparent
    - PL/1, Ada, UML 2.0?
      - an abundant amount of teaching is necessary
- The organizational/political dilemma of committee languages (The version 2 syndrome)
  - My users need only this very small addition
    - but to get it others will have to get their additions
    - together these additions may not be consistent



## On semantics of UML 2.0

- All semantics of UML 2.0 expressed in natural (?) English
- There is no common semantics model
  - but a sketch of a common execution model
- The reader will notice that the different behavioral notations, Activities, State Machines and Interactions use three different semantic approaches
  - this is made sufficiently vague so that nobody objected ....
- Interactions have a kind of trace semantics
- State Machines have automata
- Activities use a Petri-net-like approach

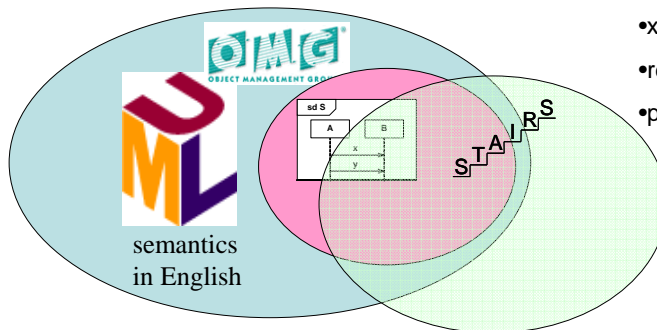


## Requirements to formalizing UML 2.0

- Formalize UML 2.0 and not "My Favorite Language"
  - UML 2.0 does have semantics – but it is written in English
- The semantics must be able to cover a significant part of the language
  - The whole of Interactions; the whole of State Machines; ....
  - Not only "Basic Sequence Diagrams"
- New suggestions
  - must be such that they may be standardized in the future
- Semantics should be such that it can be tool supported
  - or else it will never be used



## UML 2.0 and STAIRS



### Adding:

- xalt
- refuse (modifying neg)
- probabilities

### Can it migrate?

- to state machines?
- to activities?

### Not handling:

- critical (-regions)
- ignore/consider

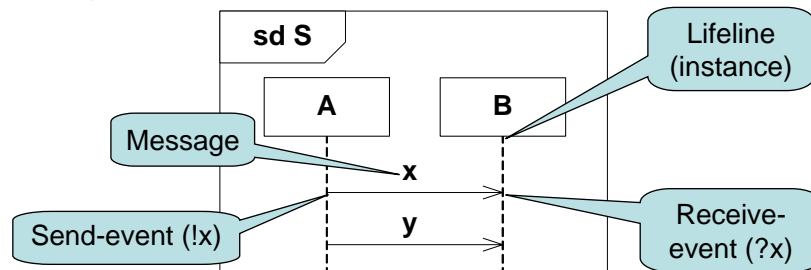


## When developing STAIRS ...

- Formalization may
  - improve understanding
  - reveal inconsistencies
    - **neg** is not exactly what it intended to be
  - motivate missing concepts
    - **xalt** to distinguish mandatory from potential scenarios
- Formalization makes it possible to challenge intuition
  - desirable formal features (of the language)
    - associativity, commutativity and distributivity
    - compositionality
      - result of analysis of one part can be used in a larger context
      - Refinement is monotonic wrt. interaction operators
  - desirable intuitive features
    - refinement should match general development progression

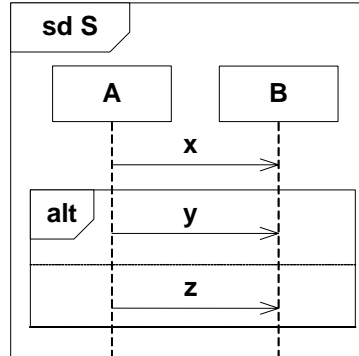


## Background: UML interactions



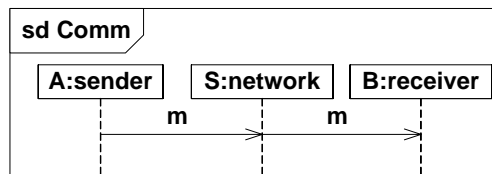
- Partial ordering of events:
  - The send event is ordered before the corresponding receive event.
  - Events on the same lifeline are ordered from the top and downwards.
- S specifies the two traces:
  - $\langle !x, ?x, !y, ?y \rangle$
  - $\langle !x, !y, ?x, ?y \rangle$

## Alternatives

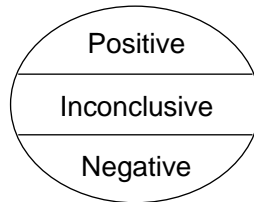


- S specifies the four traces:
    - < lx, ?x, !y, ?y >
    - < lx, !y, ?x, ?y >
    - < lx, ?x, !z, ?z >
    - < lx, !z, ?x, ?z >
- First alternative operand
- Second alternative operand

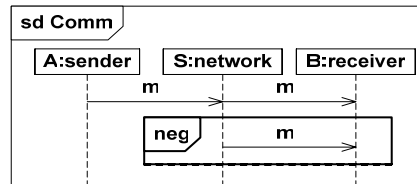
## Example: Network communication



- Interactions = example runs!
  - Specifies a set of positive and/or negative behaviours.



## Negative behavior



Positive:

 $\langle !m_{AS}, ?m_{AS}, !m_{SB}, ?m_{SB} \rangle$ 

Negative:

 $\langle !m_{AS}, ?m_{AS}, !m_{SB}, ?m_{SB}, !m_{SB}, ?m_{SB} \rangle$  $\langle !m_{AS}, ?m_{AS}, !m_{SB}, !m_{SB}, ?m_{SB}, ?m_{SB} \rangle$ 

- Formally:  
 $(p1, n1) \supseteq (p2, n2) =$   
 $(p1 \supseteq p2, (p1 \supseteq n2) \cup (n1 \supseteq p2) \cup (n1 \supseteq n2))$
- Note:
  - Inconclusive + positive/negative = inconclusive
  - Positive + negative = negative

## Underspecification and non-determinism

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



### alt vs xalt

- Assume

$$[[ d1 ]] = (p1, n1)$$

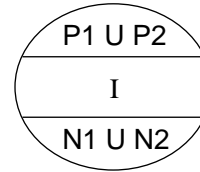
$$[[ d2 ]] = (p2, n2)$$

- alt specifies potential behavior:

$$[[ d1 \text{ alt } d2 ]]$$

$$= [[ d1 ]] + [[ d2 ]]$$

$$= (p1 \cup p2, n1 \cup n2)$$

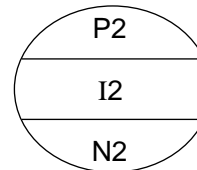
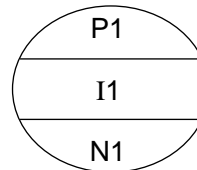


- xalt specifies mandatory behavior:

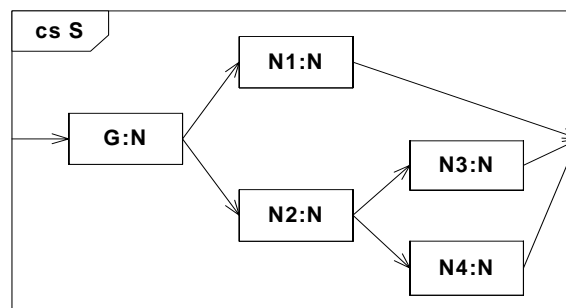
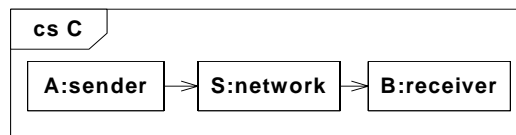
$$[[ d1 \text{ xalt } d2 ]]$$

$$= [[ d1 ]] \cup [[ d2 ]]$$

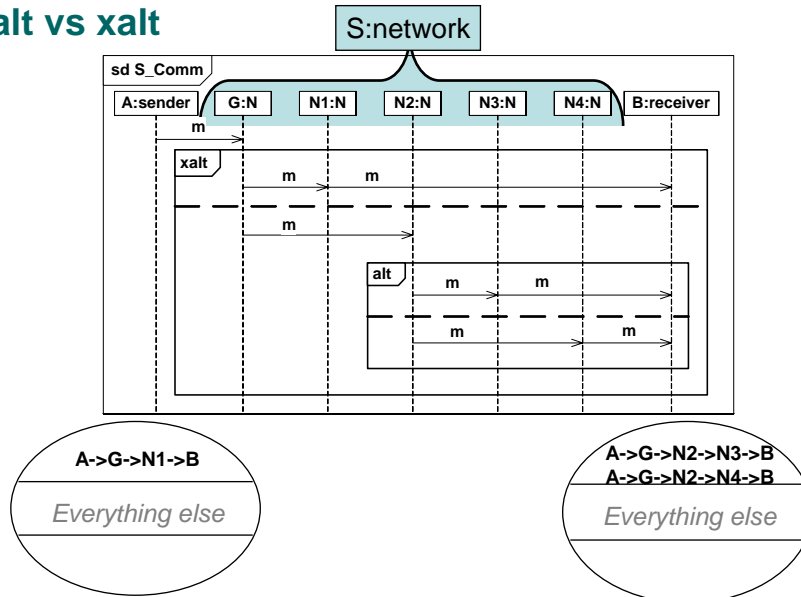
$$= (p1, n1) \cup (p2, n2)$$



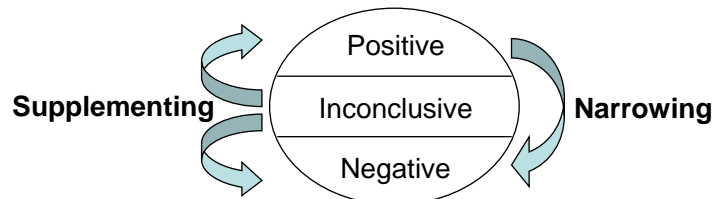
### Example: Network communication



## alt vs xalt



## Refinement in STAIRS

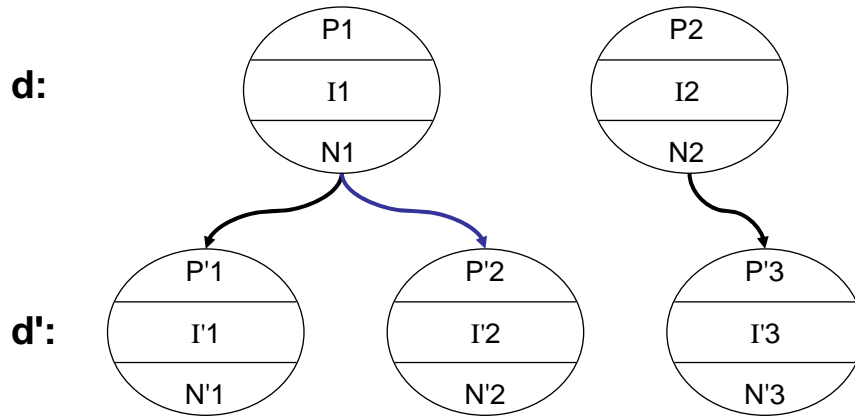


- An interaction obligation  $o'=(p',n')$  is a refinement of an interaction obligation  $o=(p,n)$  iff
  - $n \subseteq n'$
  - $p \subseteq p \cup n'$



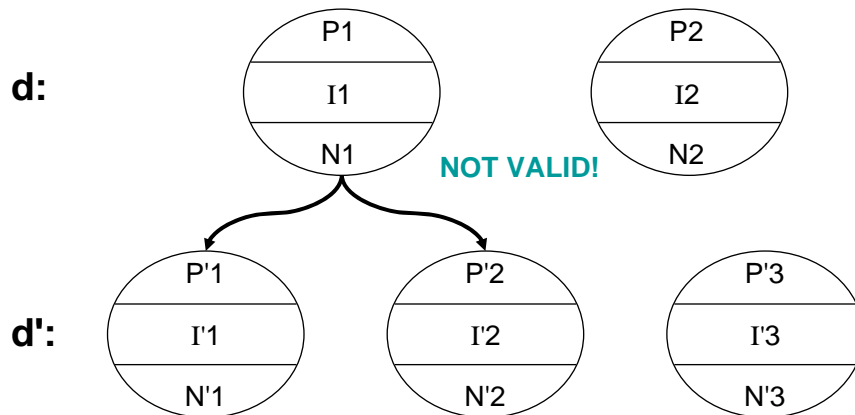
### Valid Refinement

- An interaction d' is a refinement of an interaction d iff  $\forall o \in [[d]]: \exists o' \in [[d']]: o \sim o'$



### Invalid Refinement

- An interaction d' is a refinement of an interaction d iff  $\forall o \in [[d]]: \exists o' \in [[d']]: o \sim o'$





## Literature on STAIRS (1/2)

- Øystein Haugen, Ketil Stølen:  
*STAIRS – Steps to analyze interactions with refinement semantics* (UML'2003, LNCS 2863).
  - Distinguishes between mandatory and potential behaviour
- Øystein Haugen, Knut Eilif Husa, Ragnhild Kobro Runde, Ketil Stølen:  
*STAIRS towards formal design with sequence diagrams* (SOSYM, Online First, 2005).
  - Denotational trace semantics for interactions
  - Formalizes the refinement relations in STAIRS
- Øystein Haugen, Knut Eilif Husa, Ragnhild Kobro Runde, Ketil Stølen:  
*Why timed sequence diagrams require three-event semantics* (Dagstuhl post-proc., LNCS 3466). Extended version as research report 309.
  - Extends STAIRS with time and three-event semantics



## Literature on STAIRS (2/2)

- Ragnhild Kobro Runde, Øystein Haugen, Ketil Stølen:  
*Refining UML interactions with explicit and implicit nondeterminism* (Nordic Journal of Computing, to appear).
  - Extends STAIRS with data and guards
  - More on mandatory vs potential behaviour
- Ragnhild Kobro Runde, Øystein Haugen, Ketil Stølen:  
*How to transform UML neg into a useful construct* (NIK'2005, to appear).
  - Investigates various formal definitions for negation
- Atle Refsdal, Knut Eilif Husa, Ketil Stølen:  
*Specification and refinement of soft real-time requirements using sequence diagrams* (FORMATS'05).
  - Extends STAIRS with probabilistic alternatives



## Summary

- Interactions are partial specifications:
  - Distinguish between positive and inconclusive traces
- Distinguish between underspecification (alt) and inherent non-determinism (xalt)
- Refinement also of partial interactions.
  - Supplementing
  - Narrowing
- Introducing guards should be a valid refinement step.
  - Traces with a false guards should be negative
- Especially timing requirements motivates more detail:
  - three event semantics