Refinement – formal design with sequence diagrams

Ketil Stølen

SINTEF & University of Oslo

September 21, 2007

Overview

- Obligatory Exercise No. 1
- Motivation
  - How can we incrementally develop UML specifications
- Requirements to STAIRS
  - What should we require from a stepwise method for developing UML specifications
- Explanation through an example
  - A Dinner Restaurant
- Refinement
  - Comparison with traditional pre-post paradigm

Obligatory Exercise No. 1

- Should be solved individually by each student
- Will be made available today
- Refinement exam from last year

- October 15, 9.00 AM is the HARD deadline

- October 19: Walkthrough of Oblig 1

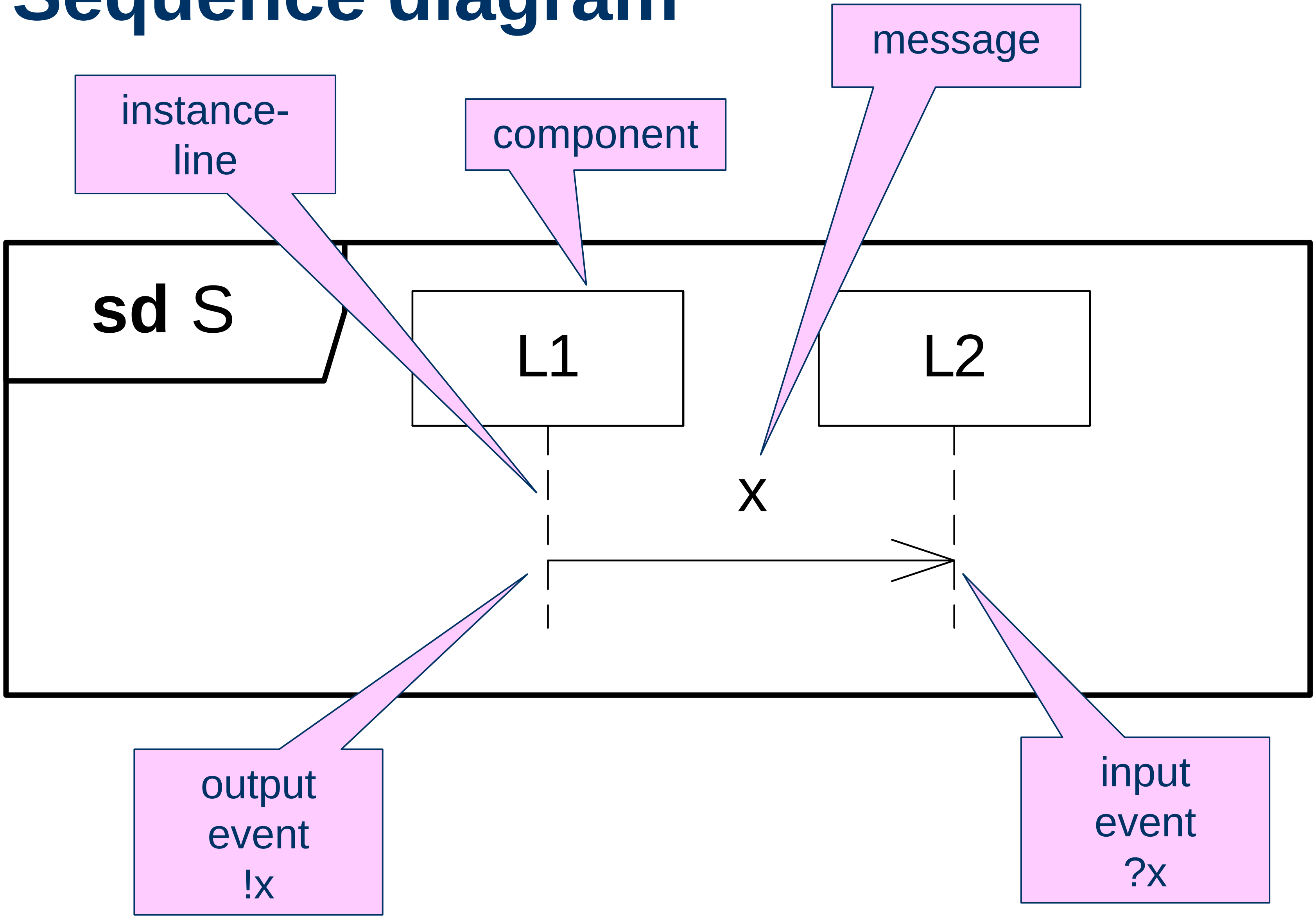
Motivation

- Exploit classical theory of refinement in a practical UML setting
  - From theory to practice, and not the other way around
- Briefly summarized: we aim to explain how classical theory of refinement can be applied to refine specifications expressed with the help of sequence diagrams
- Sequence diagrams can be used to capture the meaning of other UML description techniques for behavior
- By defining refinement for sequence diagrams we therefore implicitly define refinement for UML

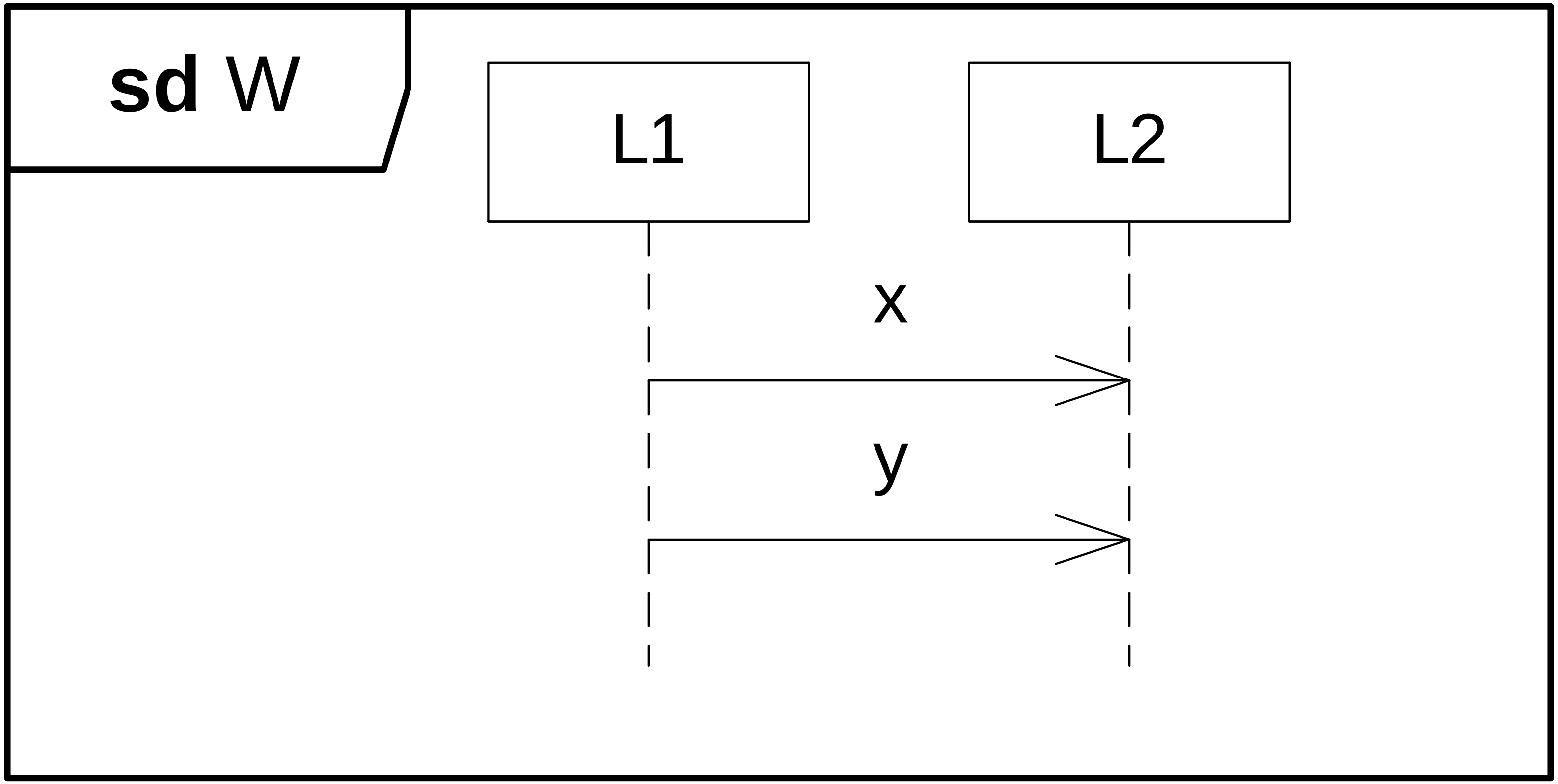
Requirements to STAIRS

- Should allow specification of potential behavior
  - Support for under-specification
- Should allow specification of mandatory behavior
  - Support for information hiding (inherent non-determinism, unpredictability)
- Should allow specification of negative behavior in addition to positive behavior
  - Support for threat modeling
- Should capture the notion of refinement
- Should formalize incremental development
- Should support compositional analysis, verification and testing

Sequence diagram



Weak sequencing



<!x,?x,!y,?y>
<!x,!y,?x,?y>

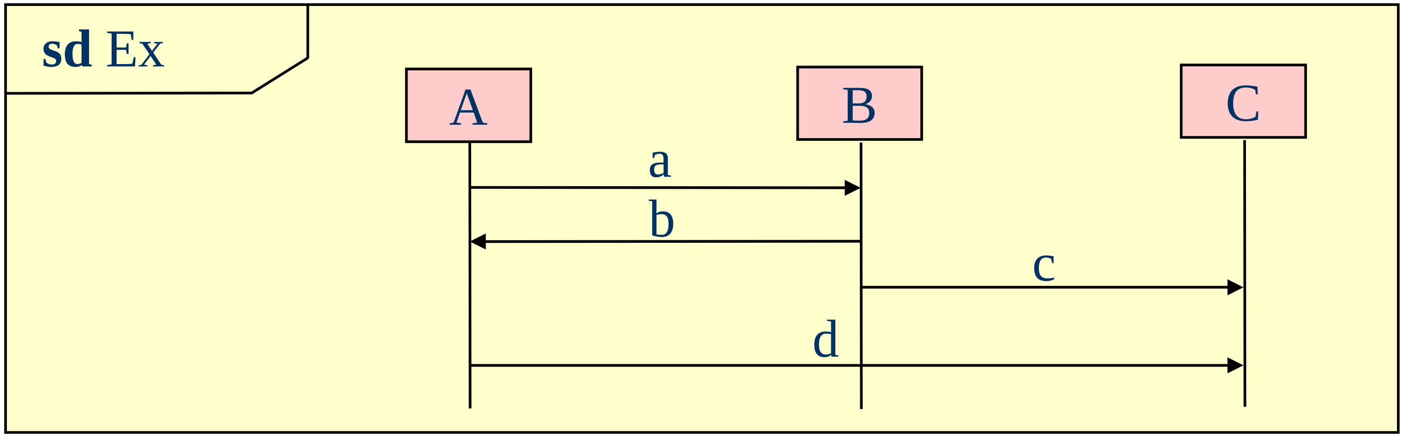
Traces

- Traces are used to capture executions (behaviors) semantically
- Within the field of formal methods there are many variants of traces
- In STAIRS traces are sequences of events

<e1, e2, e3, e4, e4, e1, e2, e5, ……………>

- An event represent either the transmission or reception of messages
  - ?m - reception of message m
  - !m - transmission of message m
- Events are instantaneous
- A trace may be finite
  - termination, deadlock, infinite waiting, crash
- A trace may also be infinite
  - infinite loop, intended non termination

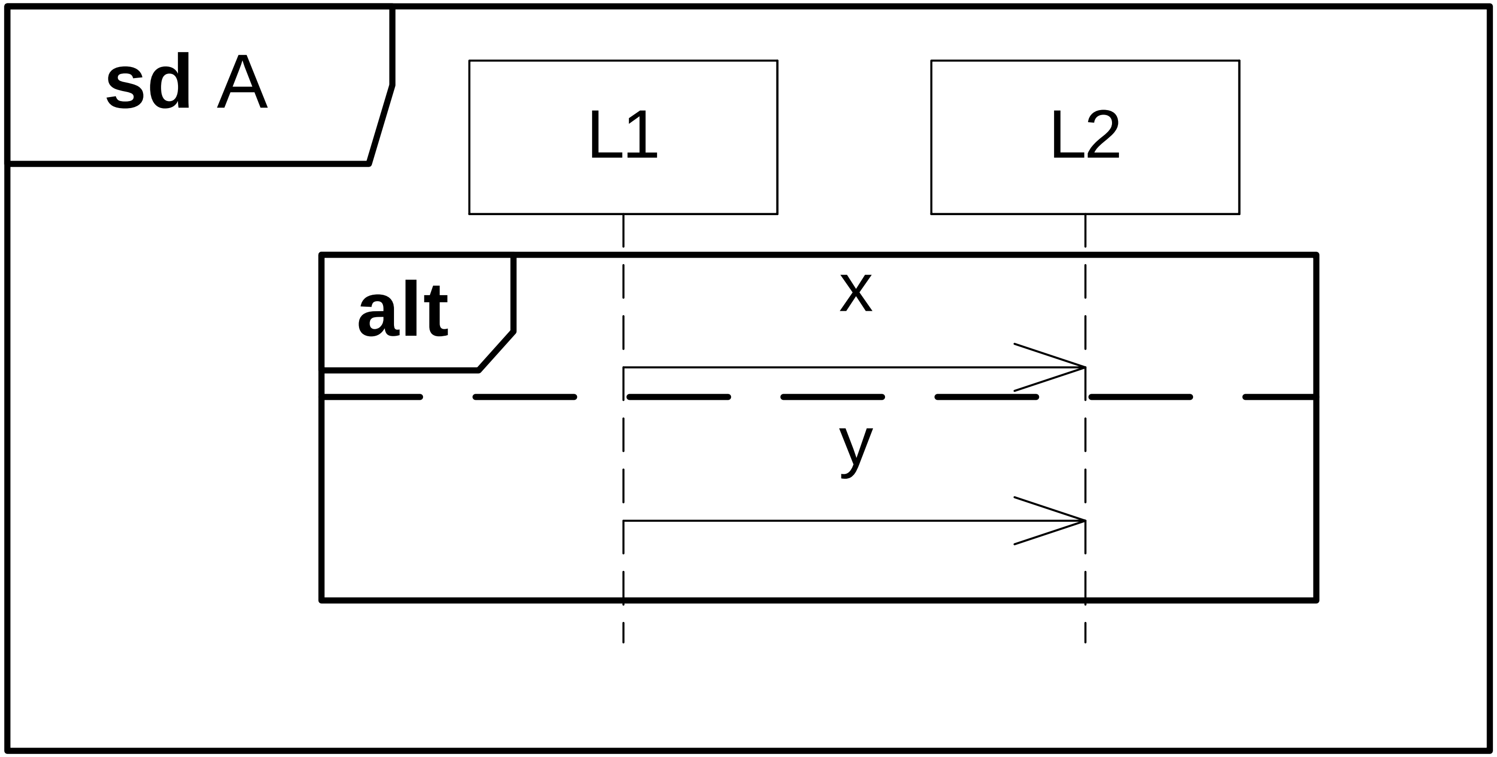
Example



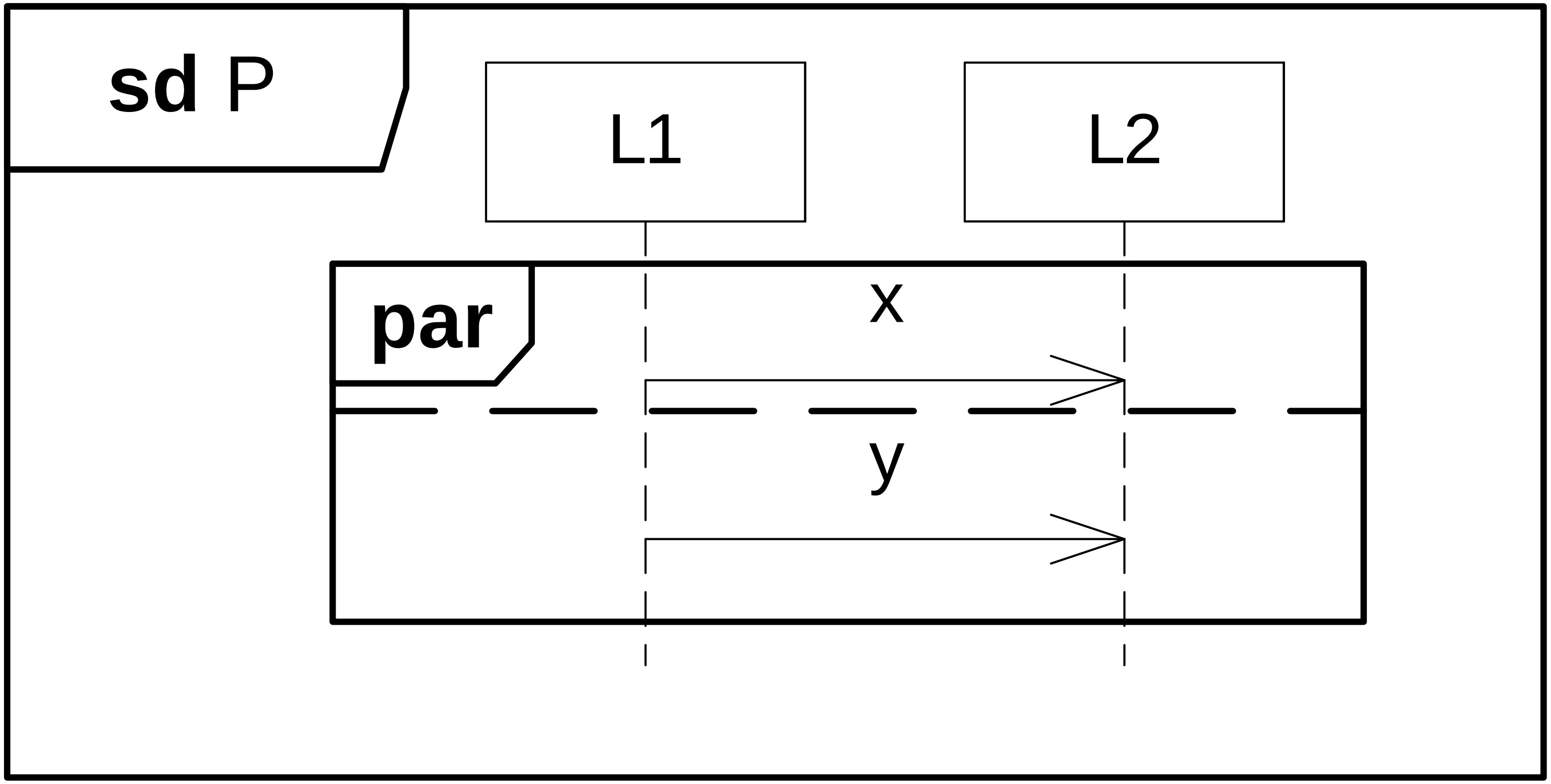
This sequence diagram has six traces:

<!a, ?a, !b, ?b, !c, ?c, !d, ?d>
<!a, ?a, !b, ?b, !c, !d, ?c, ?d>
<!a, ?a, !b, ?b, !d, !c, ?c, ?d>
<!a, ?a, !b, !c, ?b, ?c, !d, ?d>
<!a, ?a, !b, !c, ?b, !d, ?c, ?d>
<!a, ?a, !b, !c, ?c, ?b, !d, ?d>

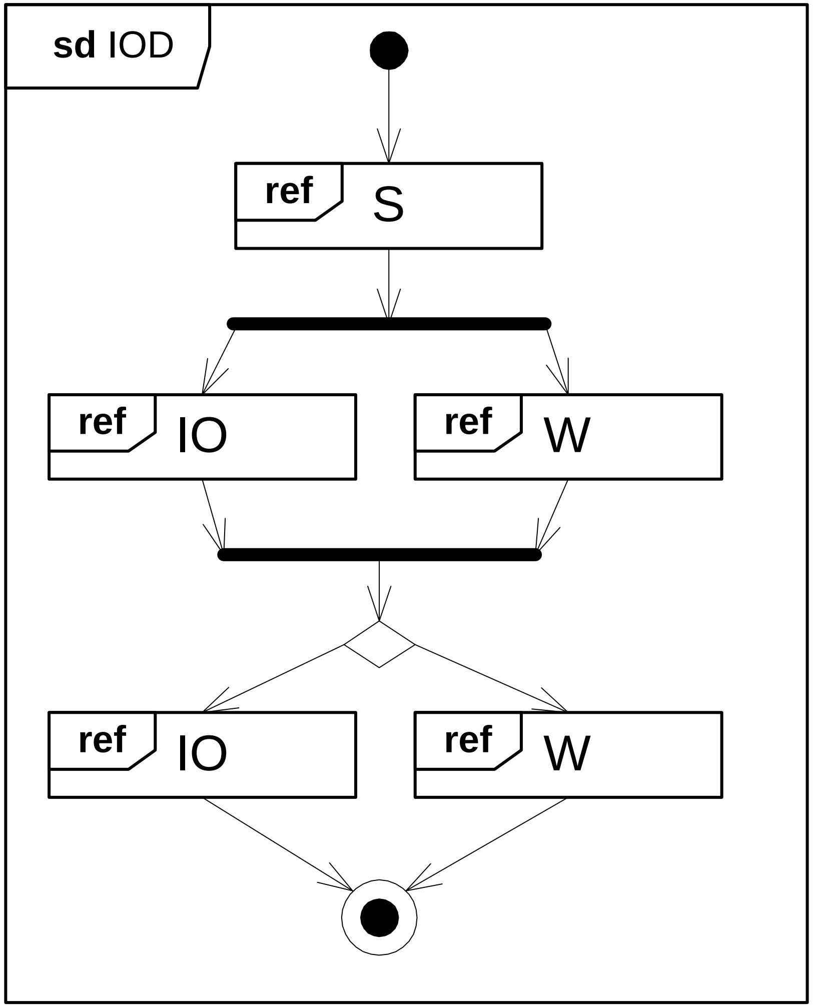
Alternative



Parallel execution

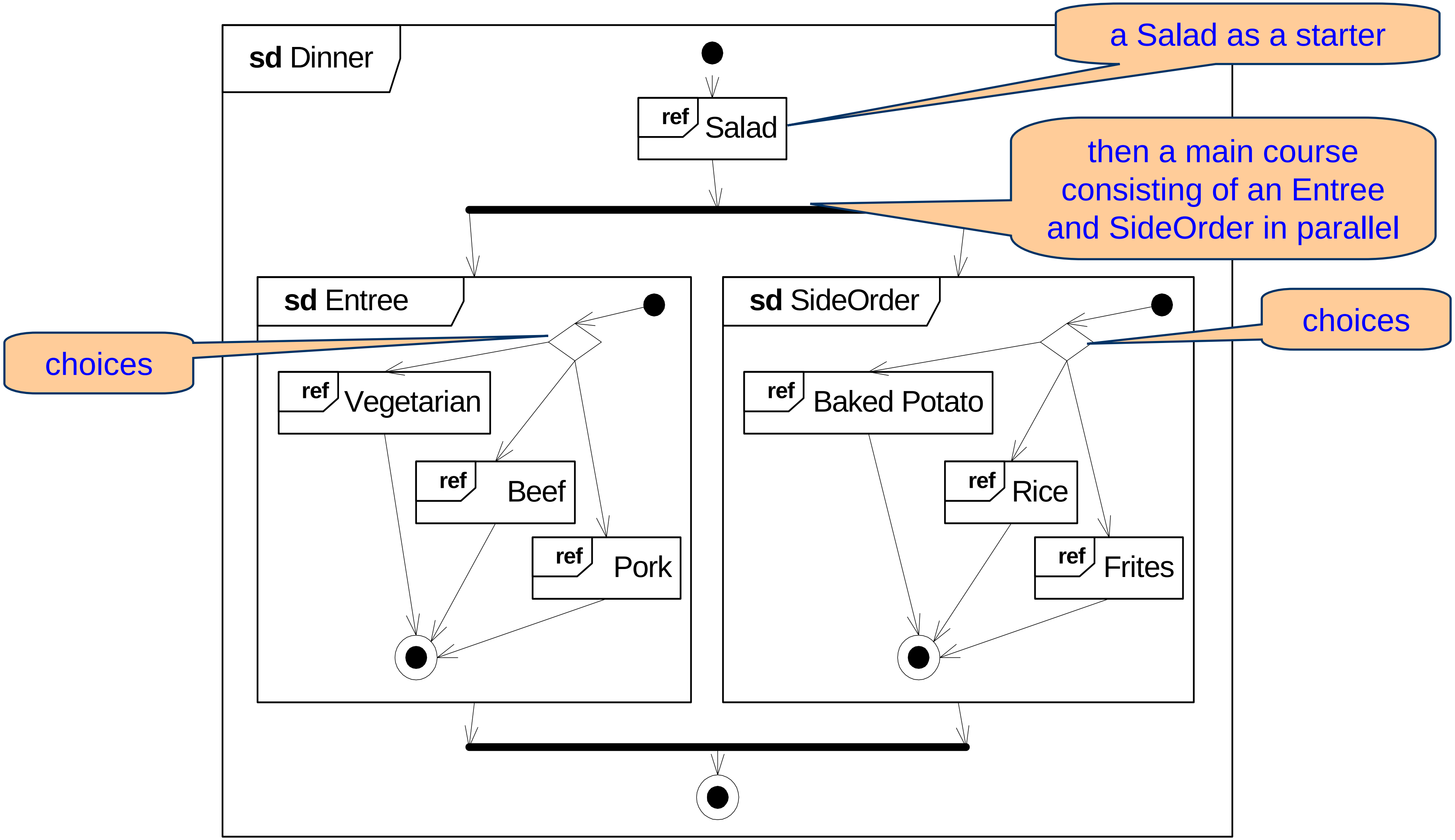


Interaction overview diagram



S **seq** (IO **par** W) **seq** (IO **alt** W)

Dinner



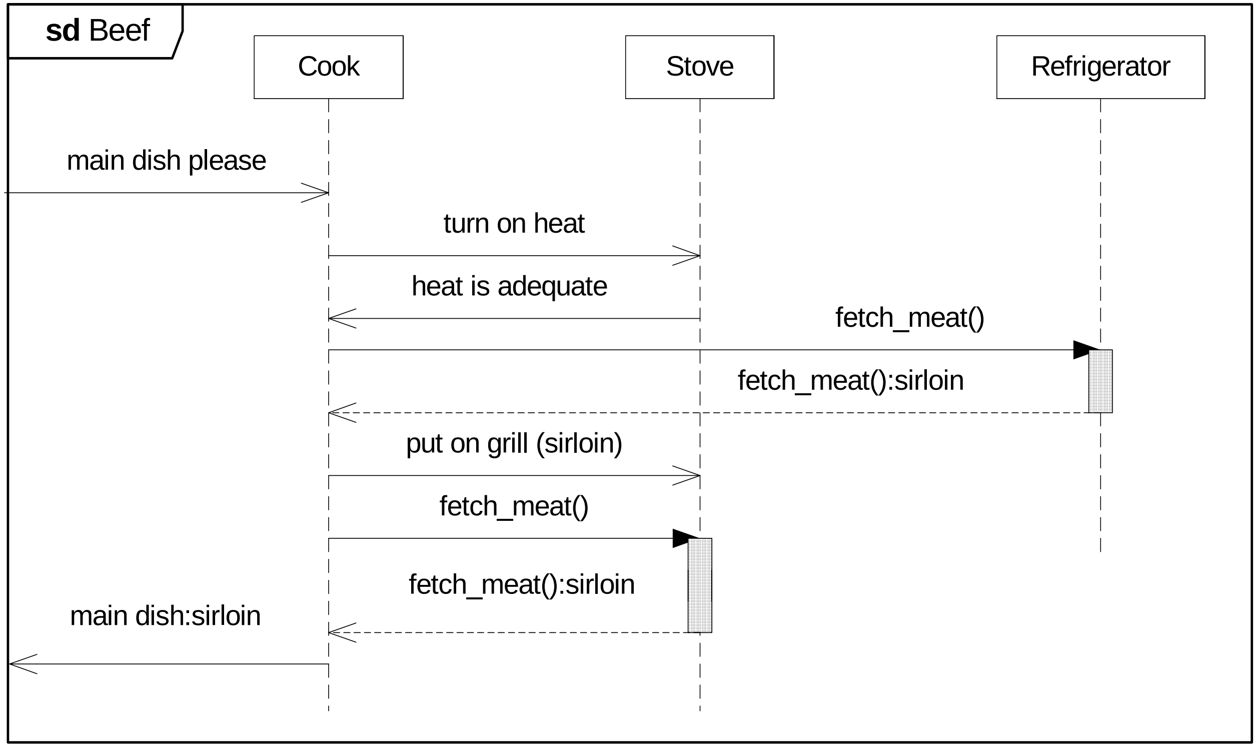
a Salad as a starter

then a main course consisting of an Entree and SideOrder in parallel

choices

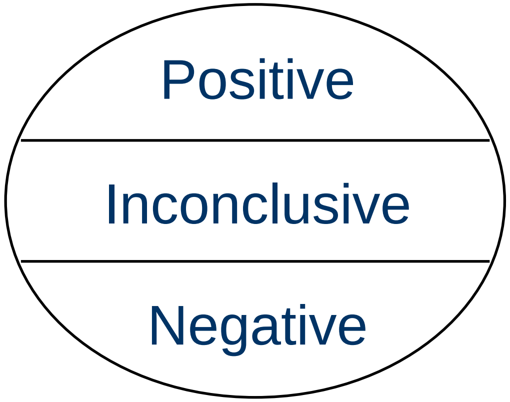
choices

Some potential positive traces of Beef



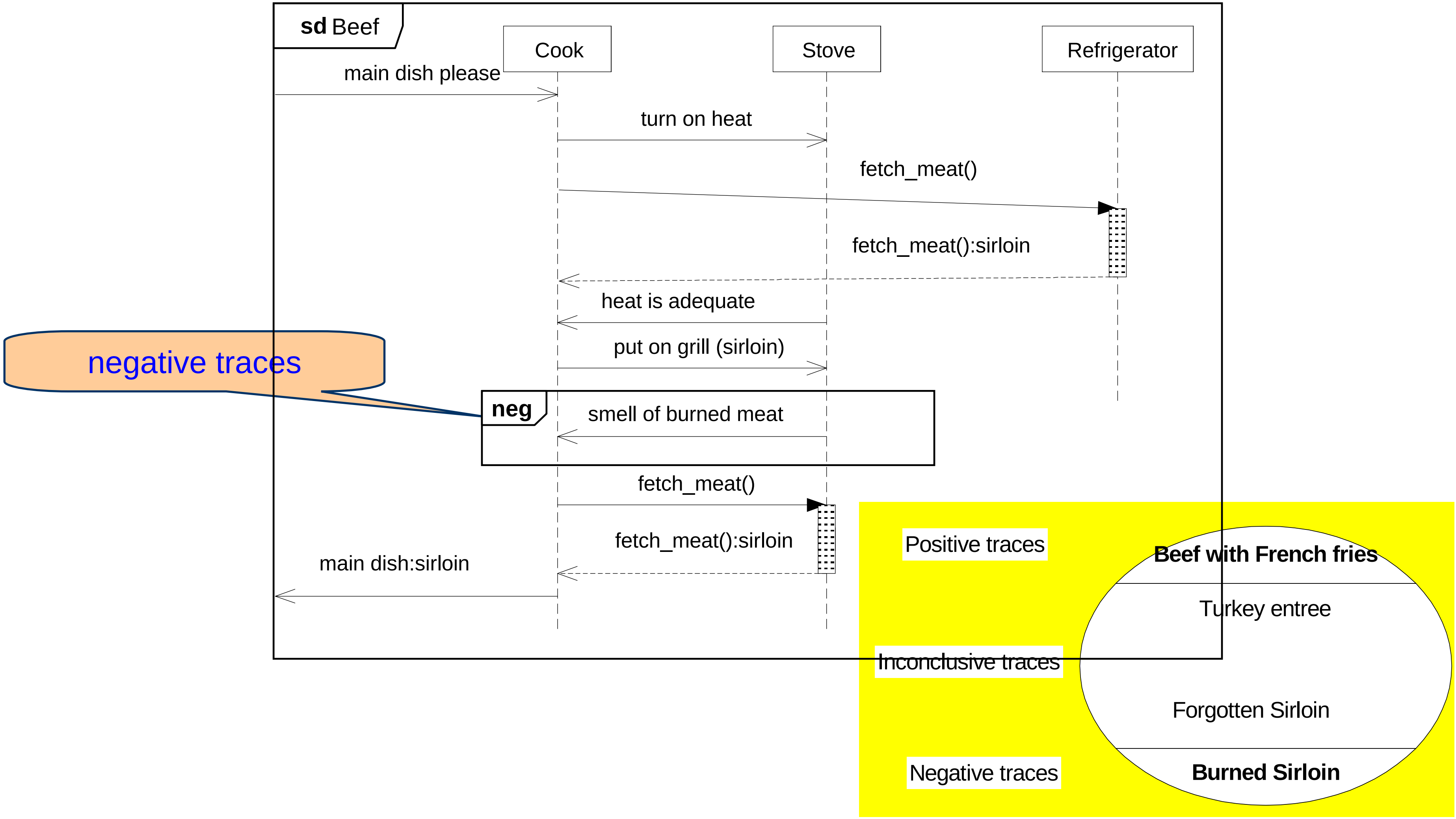
STAIRS semantics: simple case

- Each positive execution is represented by a trace
- Each negative execution is represented by a trace
- The semantics of a sequence diagram is a pair of sets of traces (Positive, Negative)



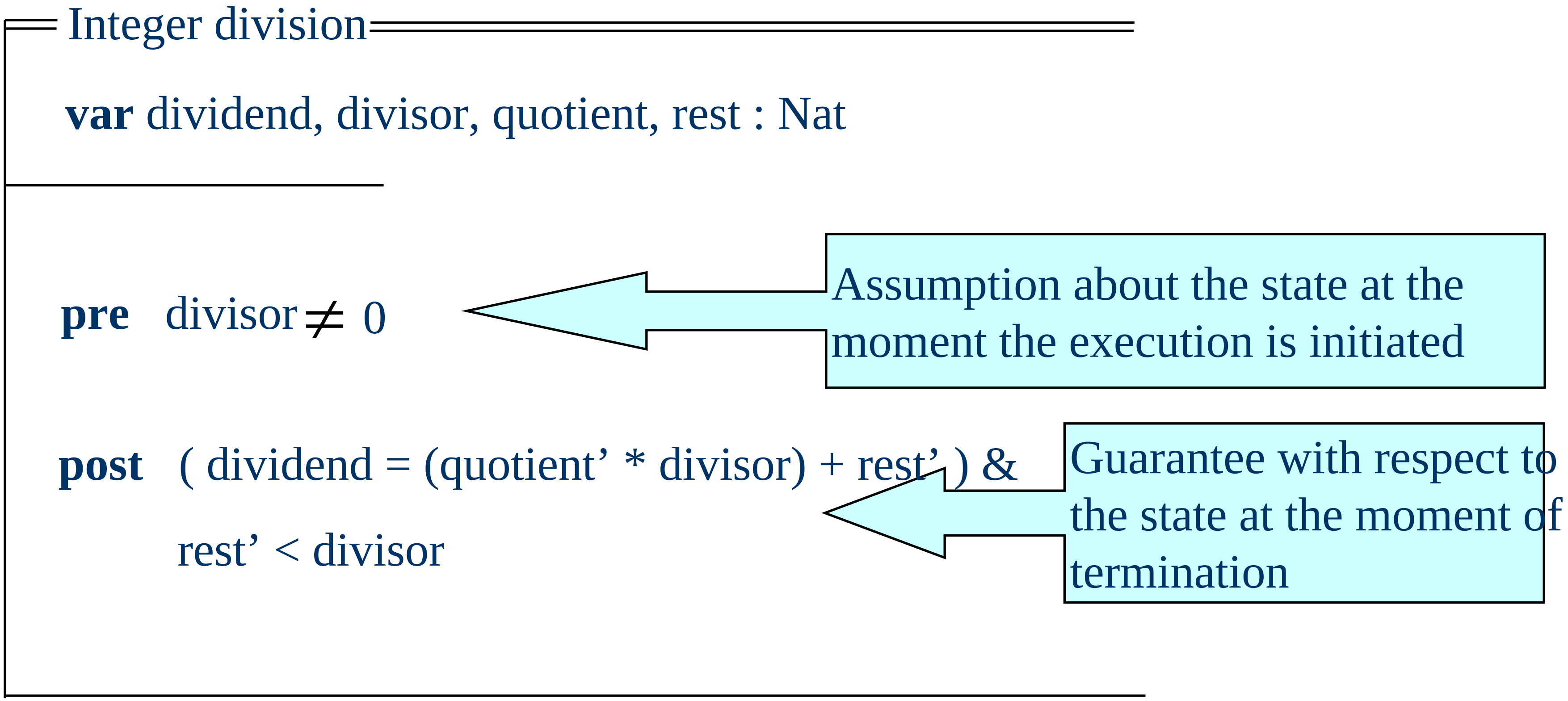
- All other traces over the actual alphabet of events are inconclusive

Potential negative Beef experiences



Pre-post specifications

Pre-post specifications are based on the assumption-guarantee paradigm



Integer division

var dividend, divisor, quotient, rest : Nat

pre divisor ≠ 0

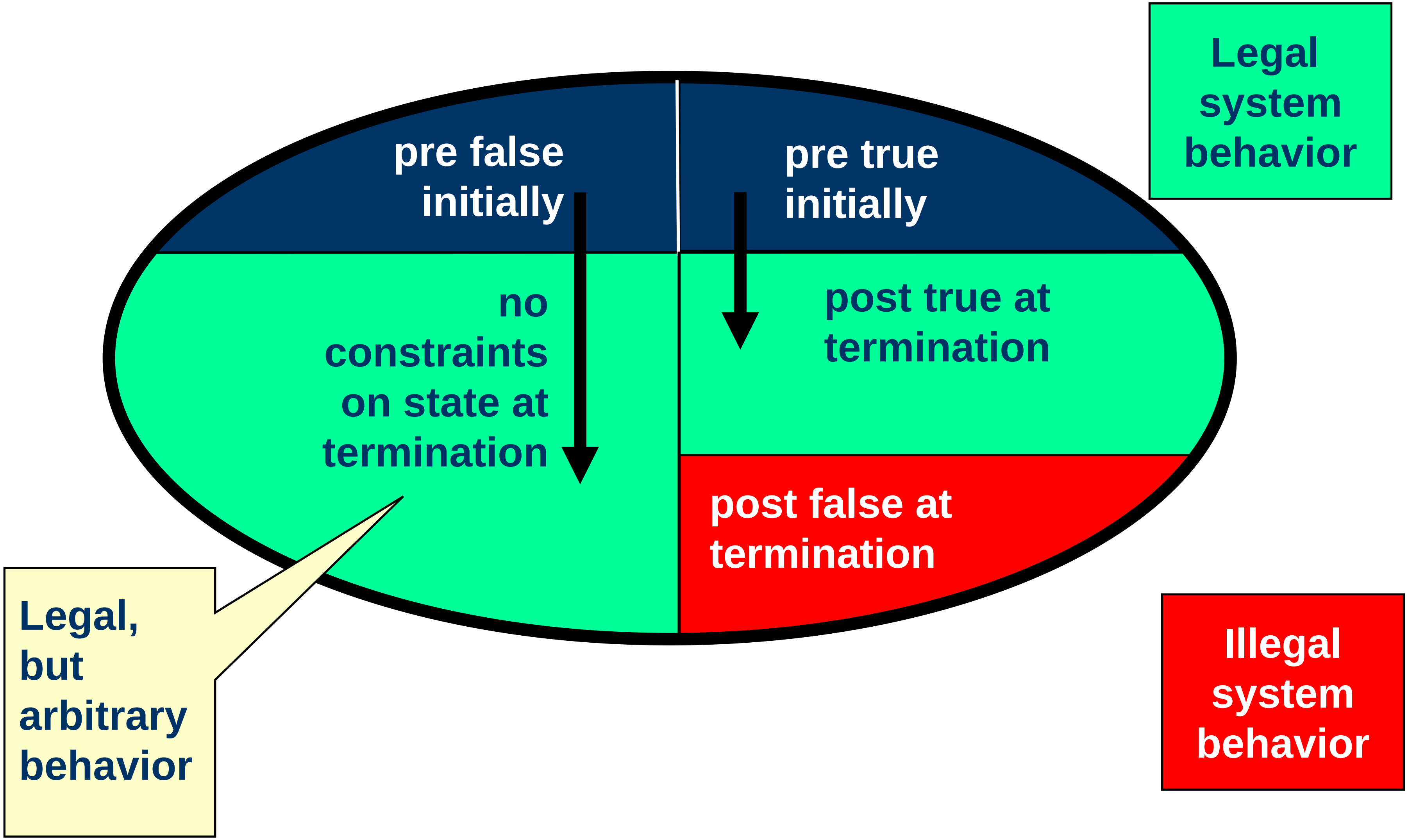
post (dividend = (quotient’ * divisor) + rest’) &

rest’ < divisor

Assumption about the state at the moment the execution is initiated

Guarantee with respect to the state at the moment of termination

Semantics of pre-post specifications



pre false initially

pre true initially

no constraints on state at termination

post true at termination

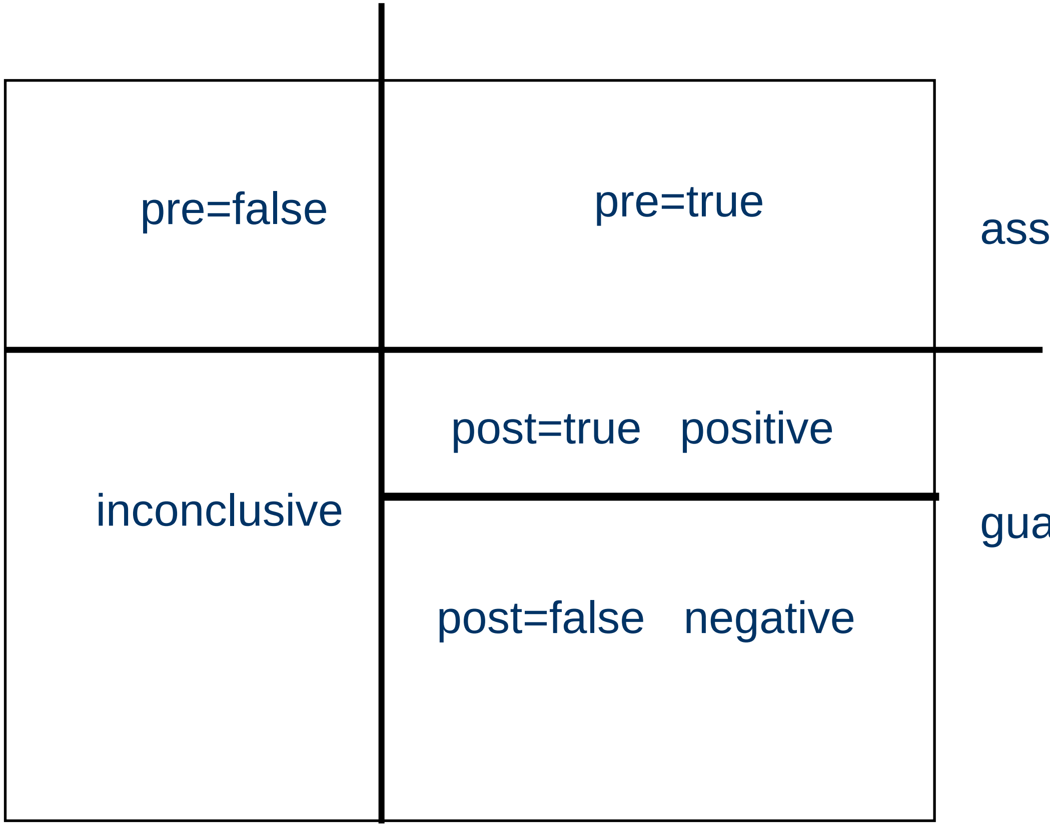
post false at termination

Legal system behavior

Illegal system behavior

Legal, but arbitrary behavior

Comparing STAIRS with pre-post



pre=false

pre=true

assumption

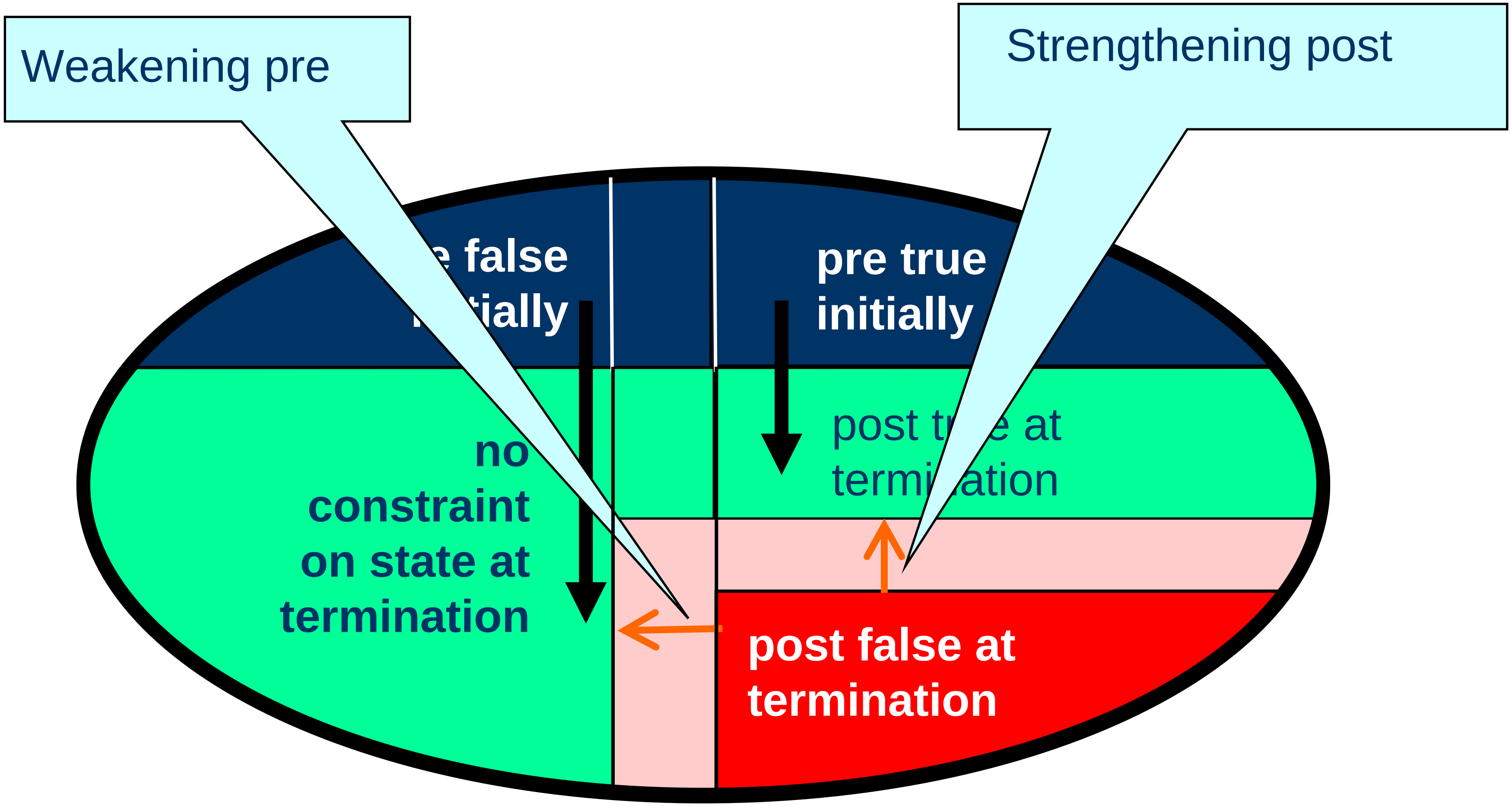
post=true positive

inconclusive

guarantee

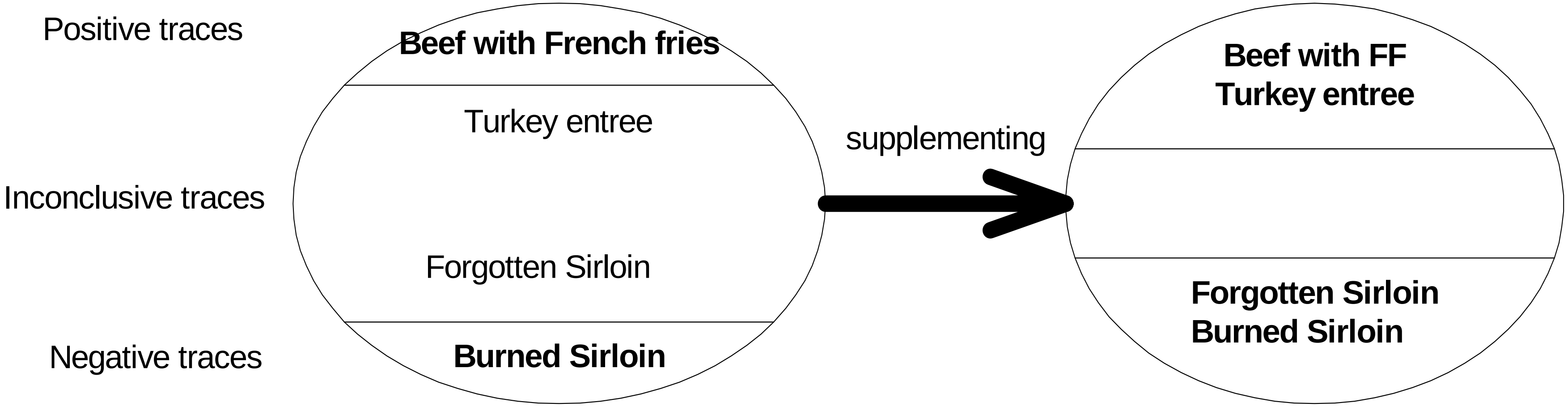
post=false negative

Refinement in pre-post

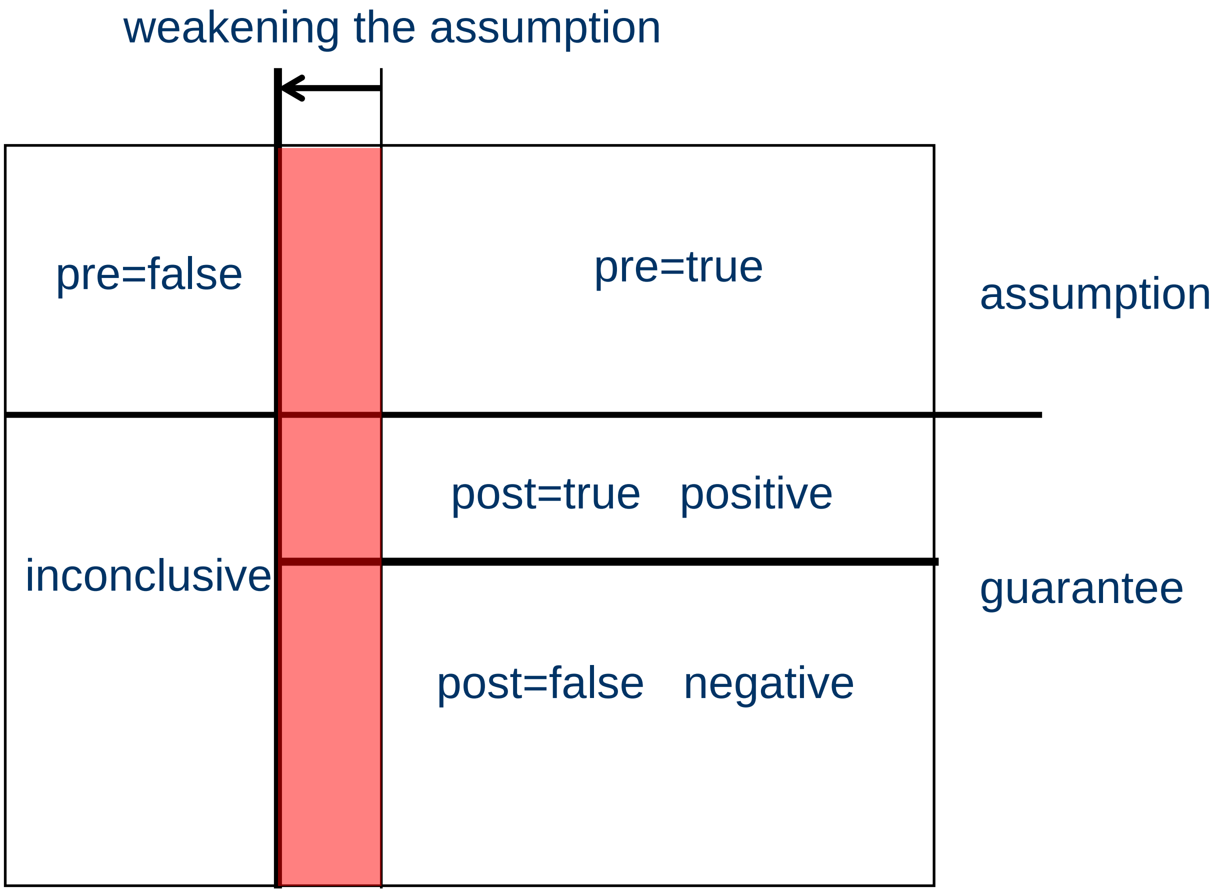


STAIRS: supplementing

- 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



Supplementing in pre-post



weakening the assumption

pre=false

pre=true

assumption

post=true positive

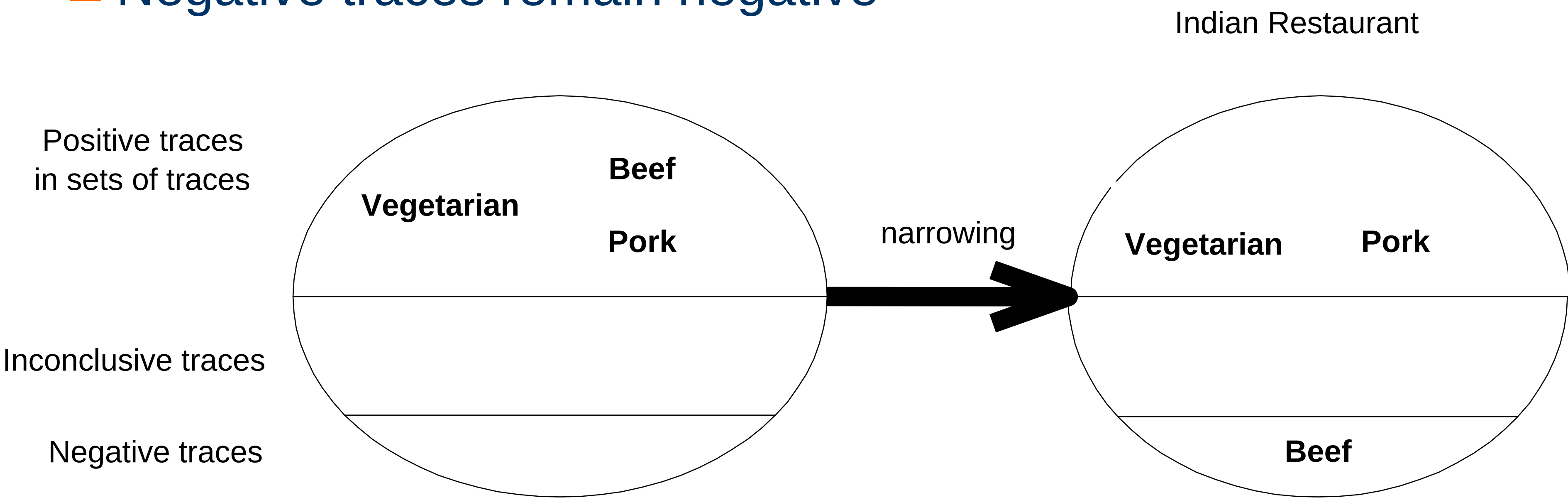
inconclusive

guarantee

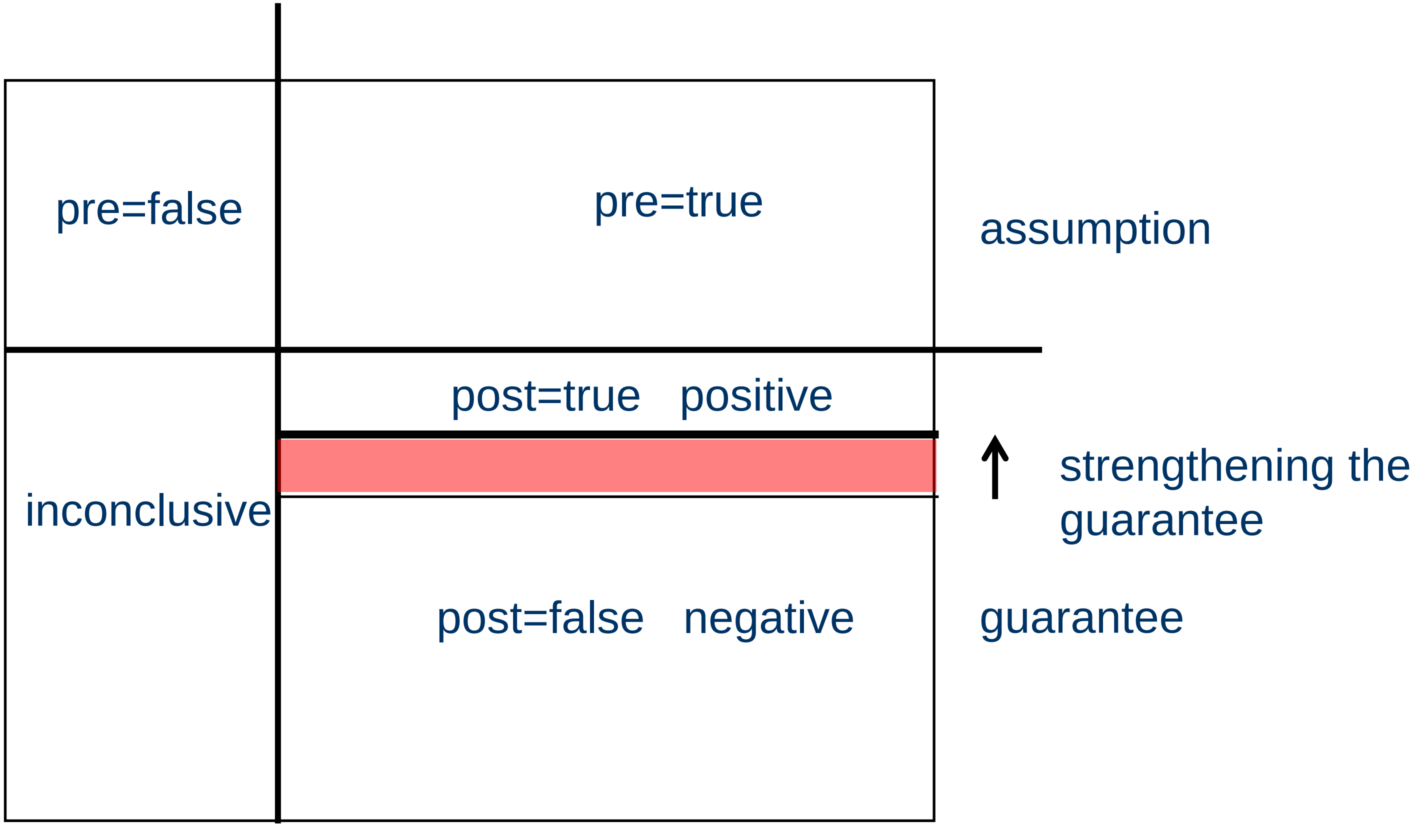
post=false negative

STAIRS: narrowing

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



Narrowing in pre-post



pre=false

pre=true

assumption

post=true positive

strengthening the guarantee

inconclusive

post=false negative

guarantee

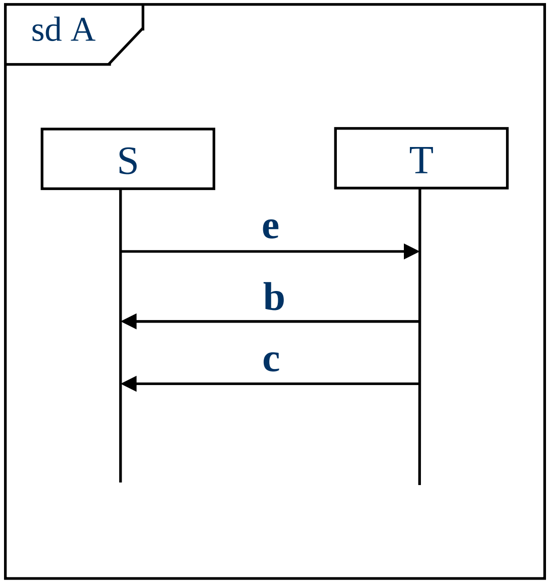
Indirect definition: Refinement in STAIRS

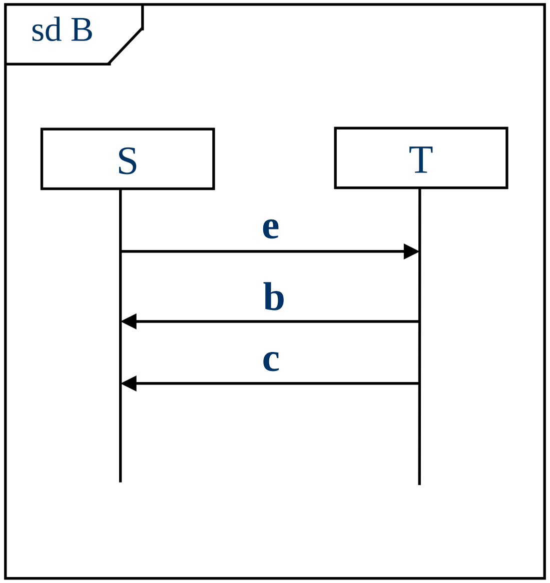
- A sequence diagram B is a general refinement of a sequence diagram A if
  - A and B are semantically identical
  - B can be obtained from A by supplementing
  - B can be obtained from A by narrowing
  - B can be obtained from A by a finite number of steps

    A -> C1 -> C2 -> …. ->Cn->B

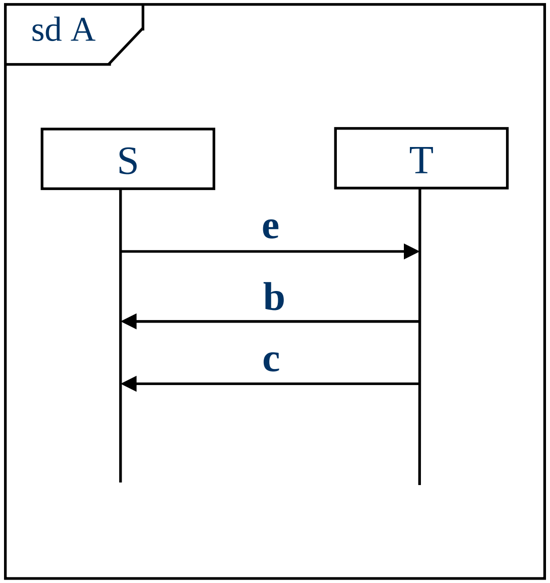
    each of which is either a supplementing or a narrowing

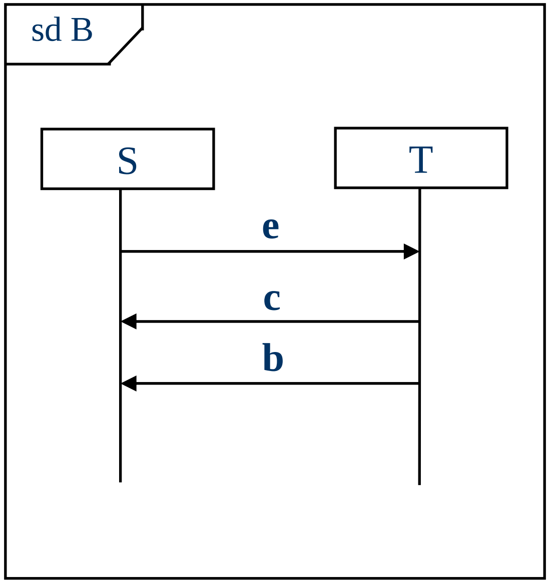
Is B a refinement of A?



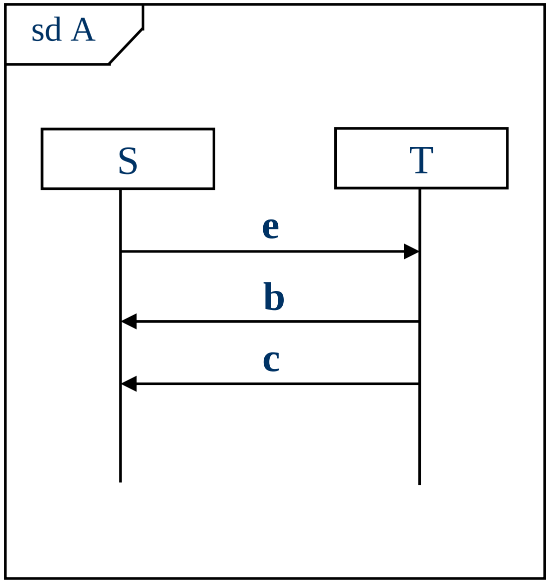


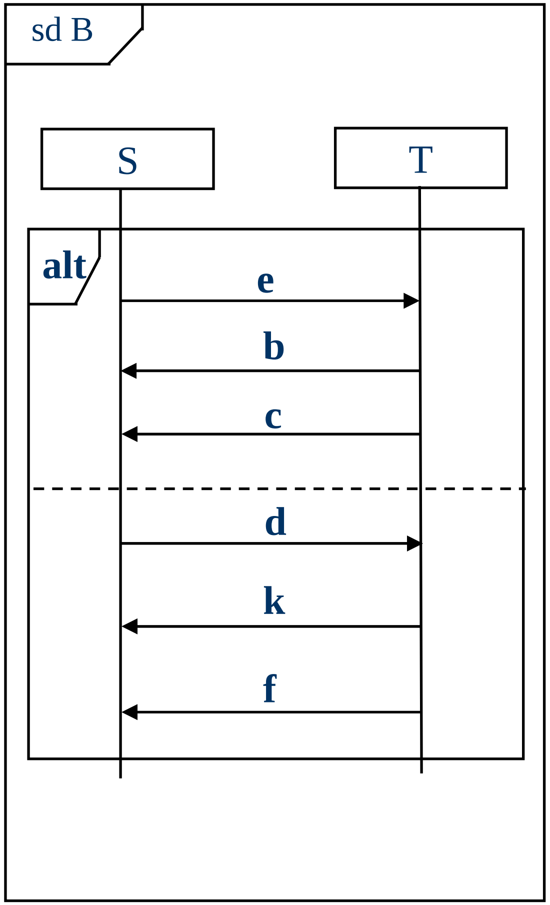
Is B a refinement of A?



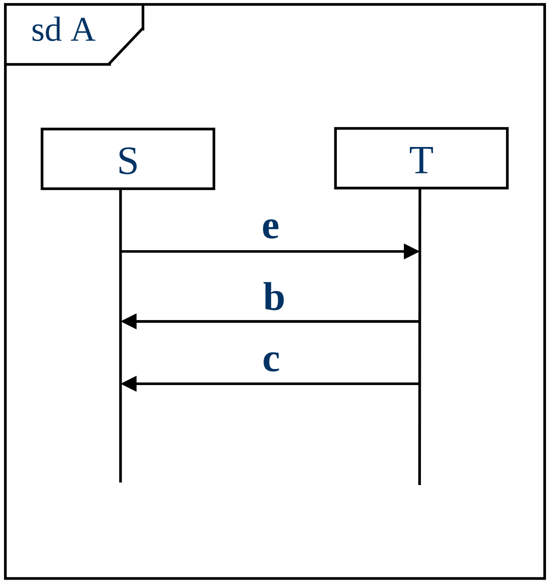


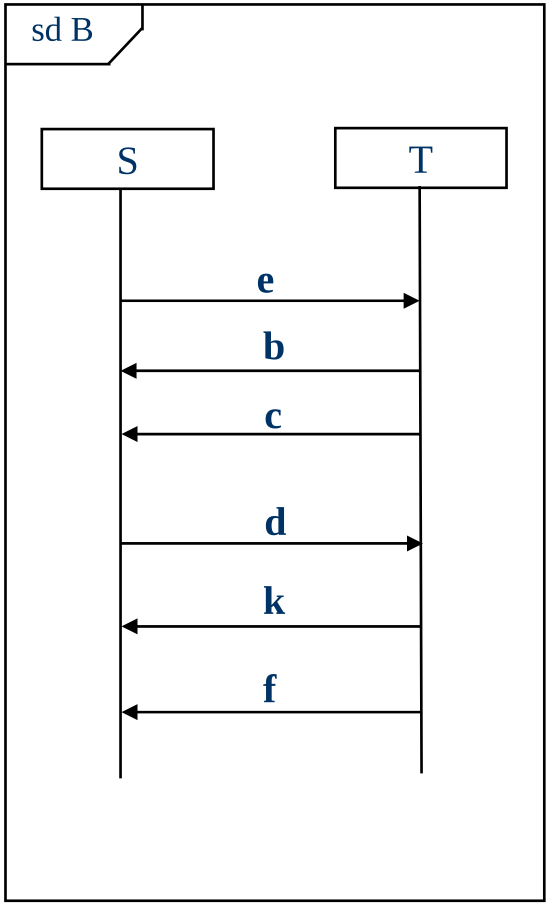
Is B a refinement of A?



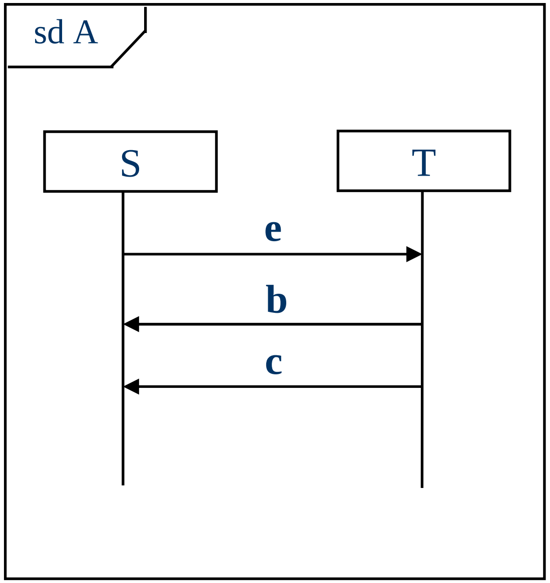


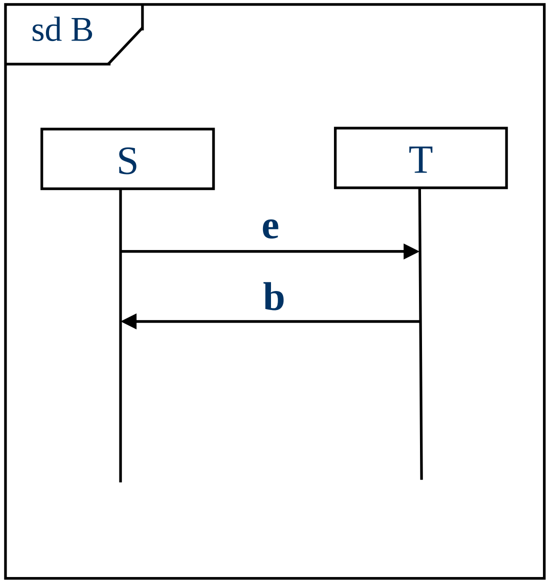
Is B a refinement A?





Is B a refinement of A?

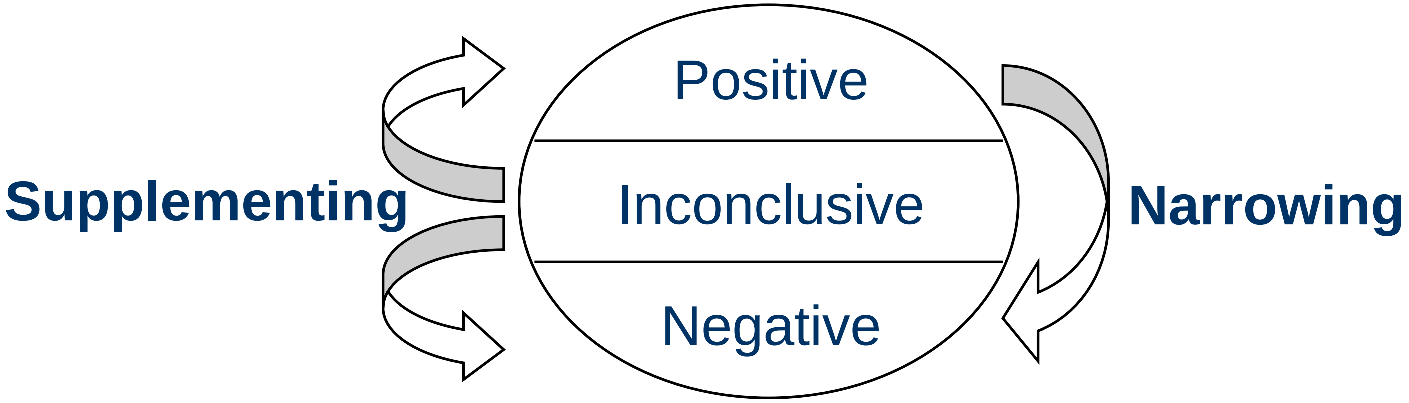




DIRECT DEFINITION: Refinement in STAIRS

- 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 in STAIRS



Supplementing

Positive

Inconclusive

Negative

Narrowing

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

Underspecification and 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!

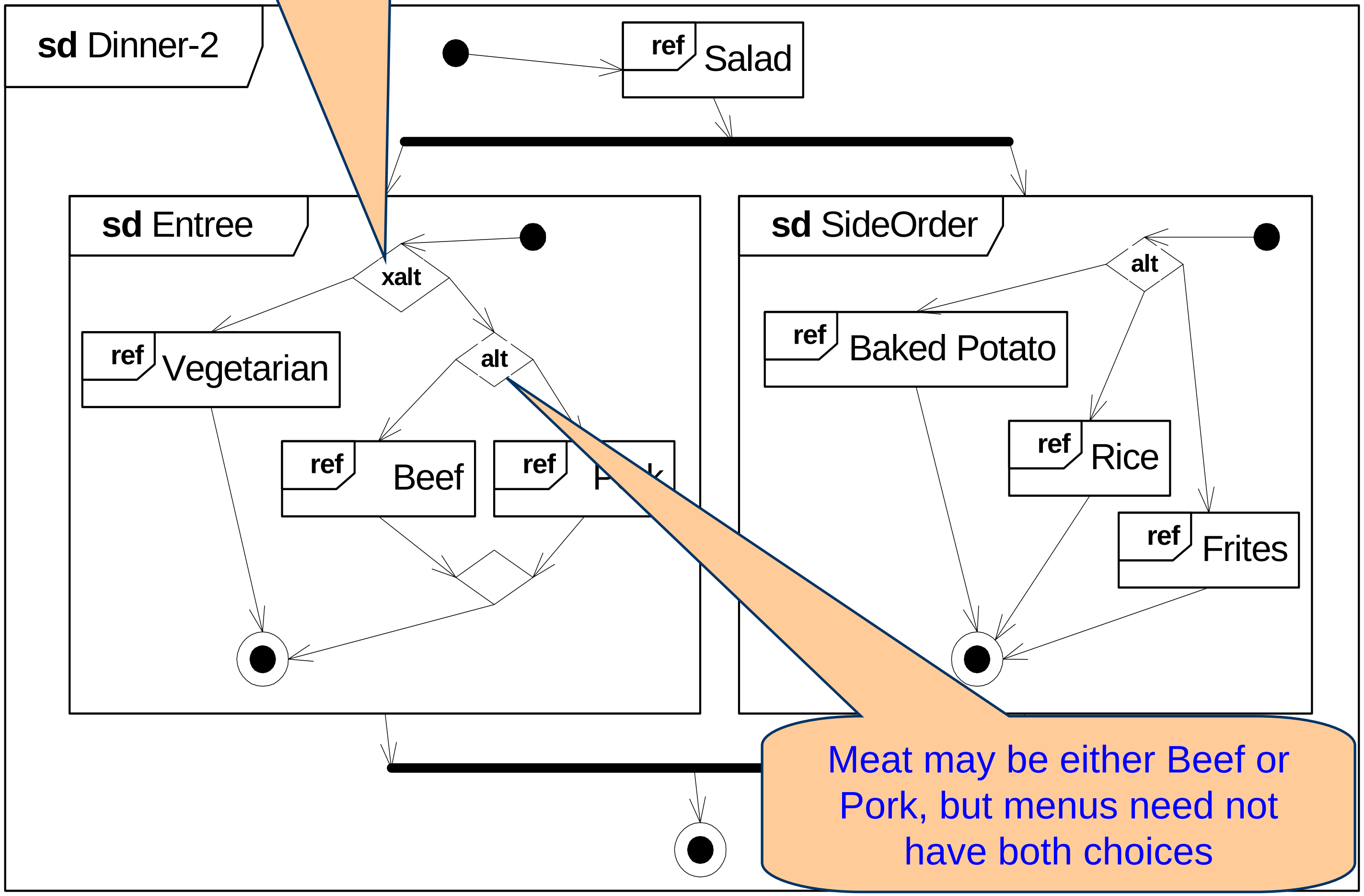
The need for both *alt* and *xalt*

- Potential non-determinism captured by alt allows abstraction and inessential non-determinism
  - Under-specification
  - Non-critical design decisions may be postponed

- Mandatory non-determinism captured by xalt characterizes non-determinism that must be reflected in every correct implementation
  - Makes it possible to specify games
  - Important in relation to security
  - Also helpful as a means of abstraction

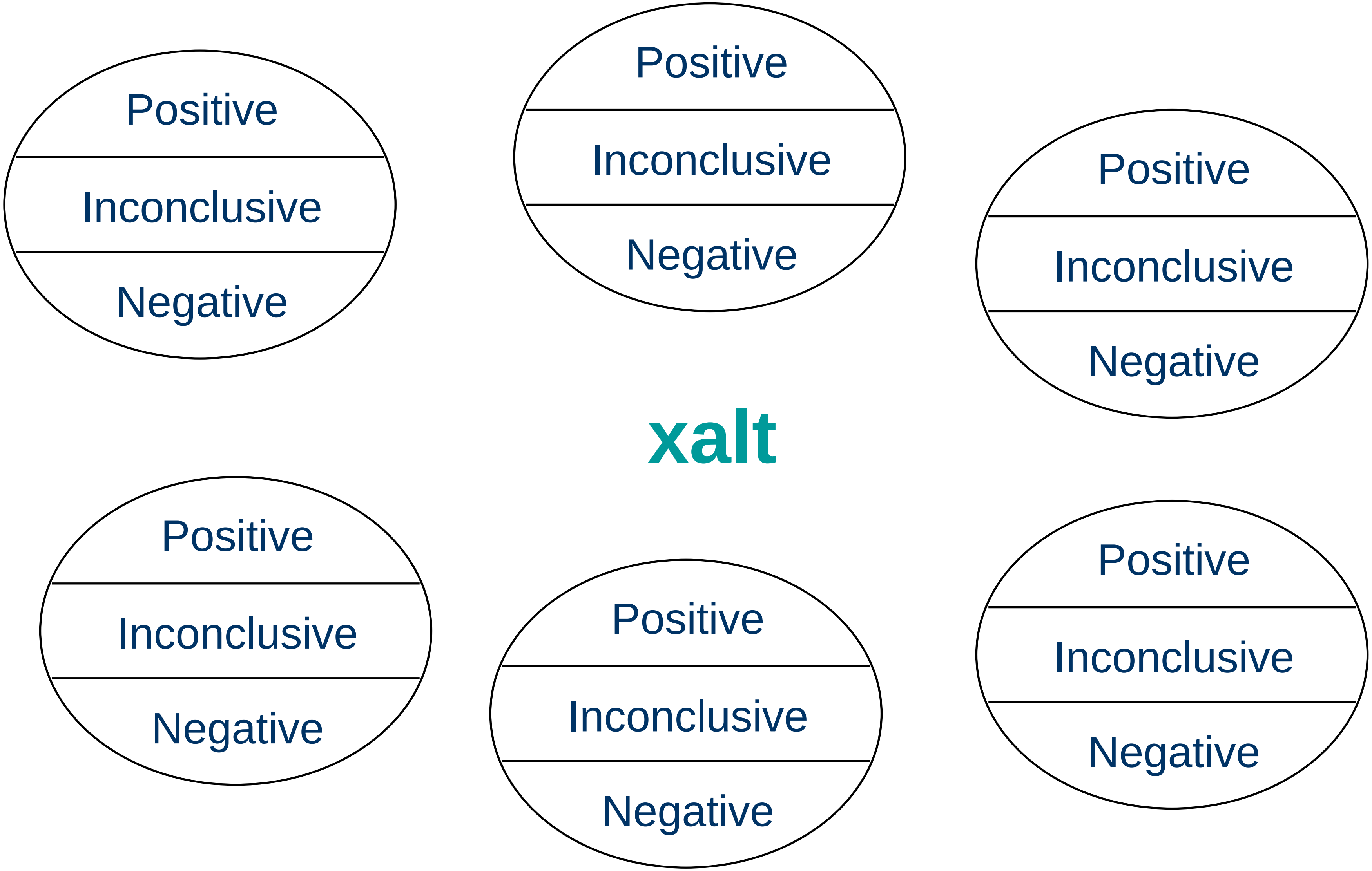
Restaurant example with both alt and xalt

Entree menus must have the choice of Vegetarian or Meat



Meat may be either Beef or Pork, but menus need not have both choices

STAIRS



Positive

Inconclusive

Negative

Positive

Inconclusive

Negative

Positive

Inconclusive

Negative

xalt

Positive

Inconclusive

Negative

Positive

Inconclusive

Negative

Positive

Inconclusive

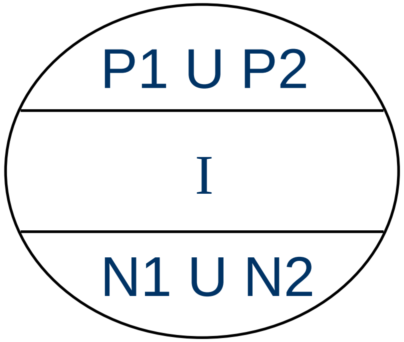
Negative

alt vs xalt

- Assume

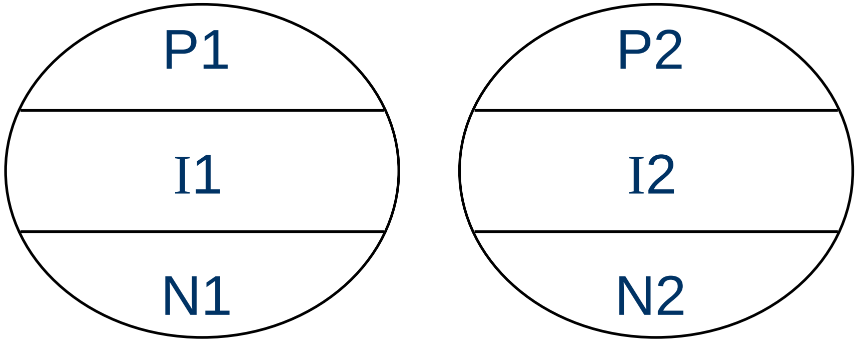
  [[d1]] = {(p1,n1)}   [[d2]] = {(p2,n2)}
- alt specifies potential behaviour:

  [[d1 alt d2]]
  = [[d1]] + [[d2]]
  = {(p1 U p2, n1 U n2)}

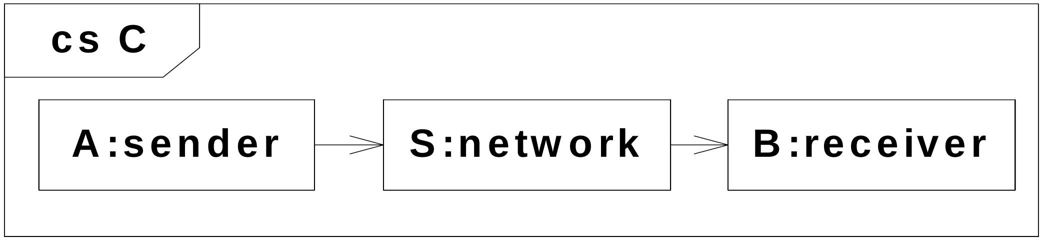


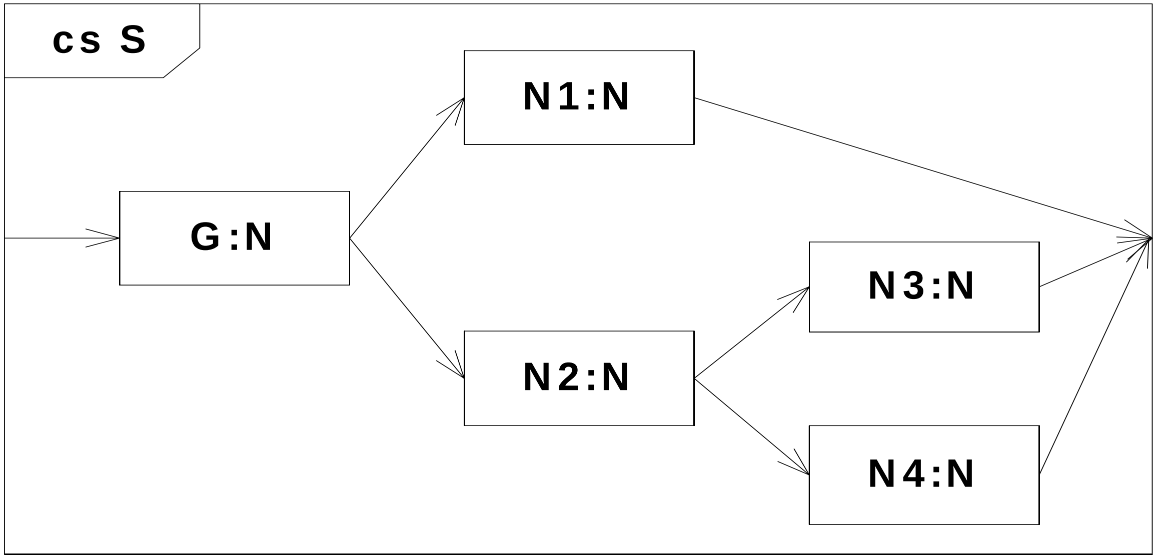
- xalt specifies mandatory behaviour:

  [[d1 xalt d2]]
  = [[d1]] U [[d2]]
  = {(p1,n1)} U {(p2,n2)}

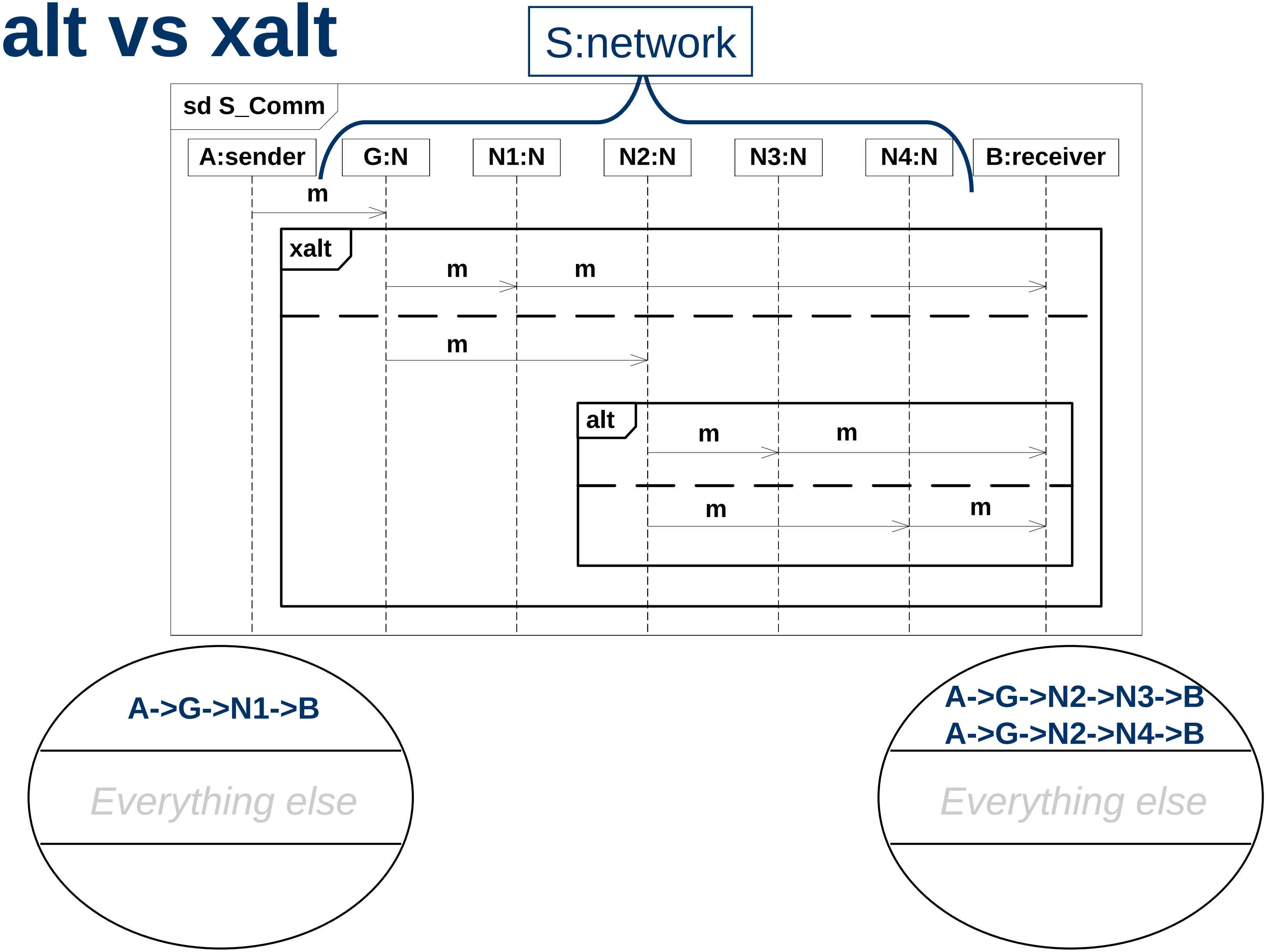


Example: Network communication





alt vs xalt


S:network
sd S_Comm
A:sender
G:N
N1:N
N2:N
N3:N
N4:N
B:receiver
m
xalt
m
m
m
alt
m
m
m
m
A->G->N1->B
Everything else
A->G->N2->N3->B
A->G->N2->N4->B
Everything else

Mandatory requirements STAIRS

- 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.

NOTE:

- Next Tuesday: Group Session Cancelled
- Next Friday: First lecture on security analysis

- Next Lecture on STAIRS: October 5.