Refinement I

From theory to practice

Ketil Stølen

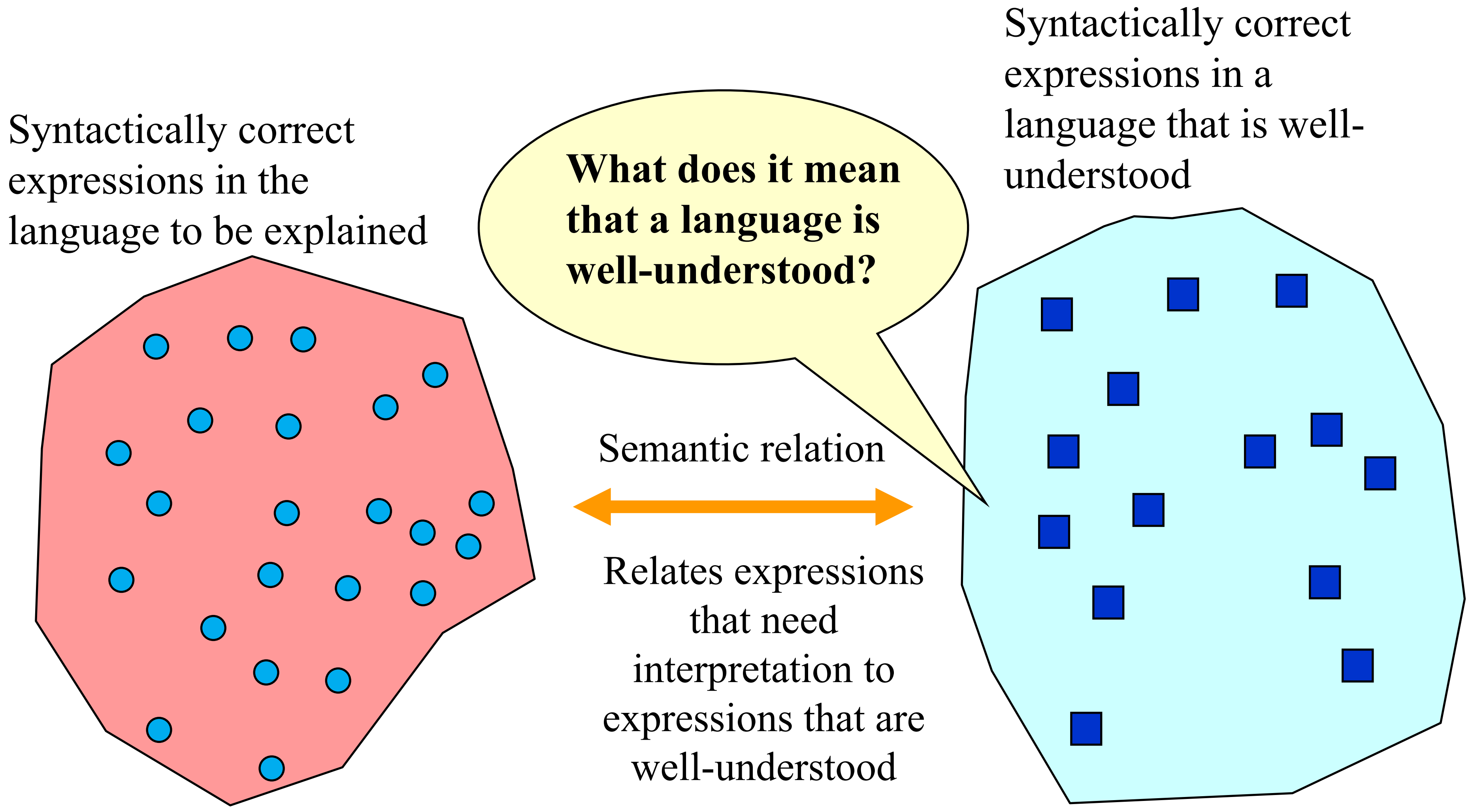
Objectives for the lectures on refinement

- Motivate the role of refinement
- Introduce and relate the following notions of refinement
  - supplementing
  - narrowing
- Illustrate the use of these notions of refinement
  - the interplay between specification and refinement
- Illustrate the translation of theory into practice

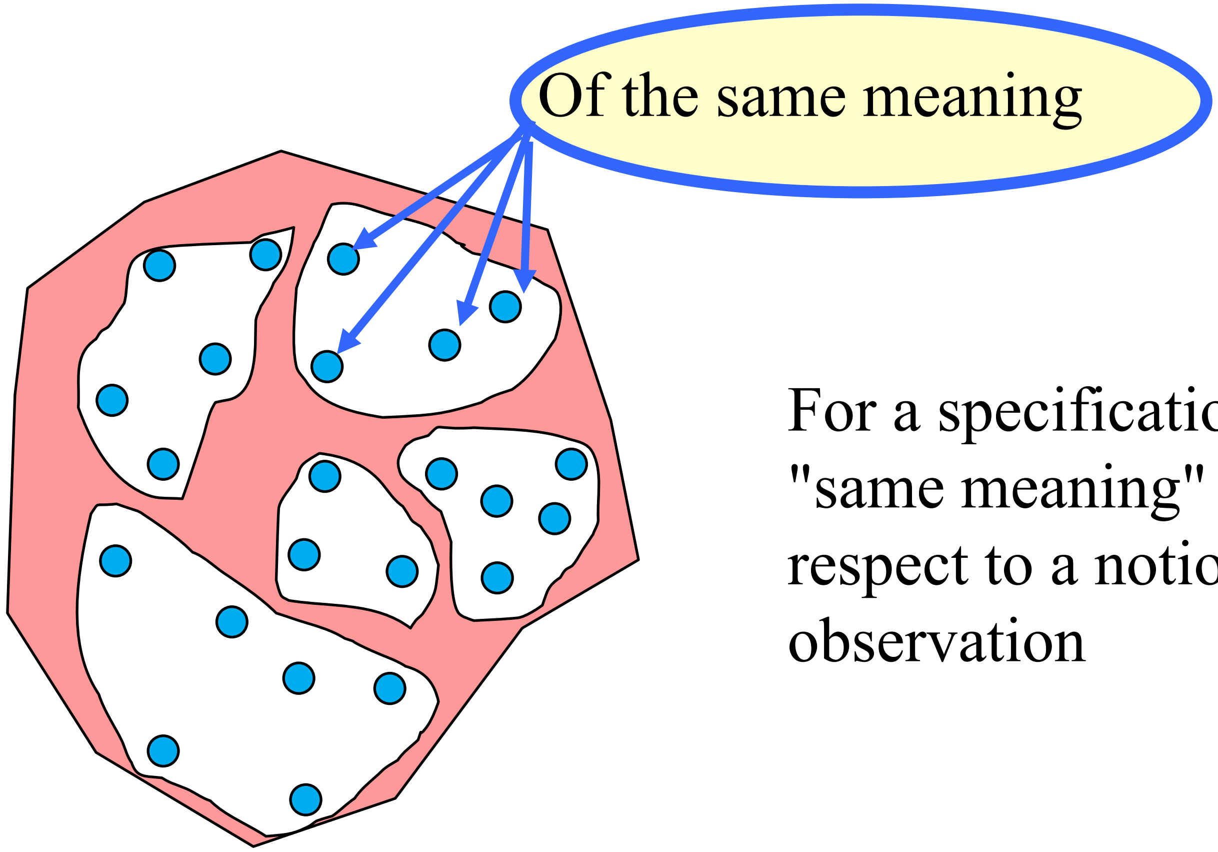
Three main concepts of language theory

- Syntax
  - The relationship between symbols or groups of symbols independent of content, usage and interpretation
- Semantics
  - The rules and conventions that are necessary to interpret and understand the content of language constructs
- Pragmatics
  - The study of the relationship between symbols or groups of symbols and their interpretation and usage

Semantic relation


Syntactically correct expressions in the language to be explained
What does it mean that a language is well-understood?
Syntactically correct expressions in a language that is well-understood
Semantic relation
Relates expressions that need interpretation to expressions that are well-understood

The need for a notion of observation



For a specification language "same meaning" is defined with respect to a notion of observation

Our notion of observation

- May observe only external behavior
- May observe that nothing bad happens
- May observe that something eventually happens
- May observe any potential behavior
- May observe time with respect to a global clock

May our notion of observation be implemented by a human being?

Pre-post specifications

The origins of refinement

Pre-post specifications

Integer division

var dividend, divisor, quotient, rest : Nat

pre divisor $\neq$ 0

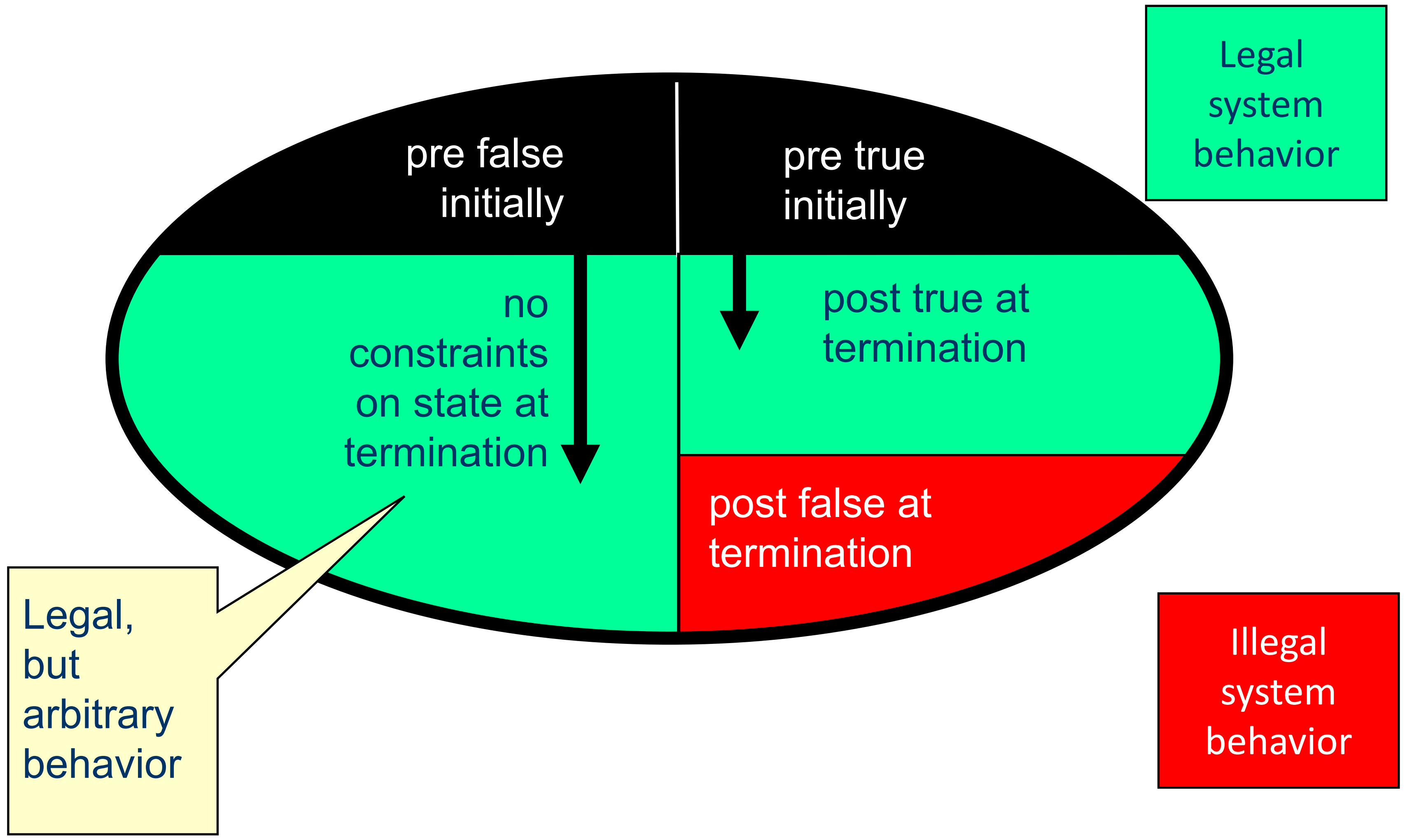
Assumption about the state at the moment the execution is initiated

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

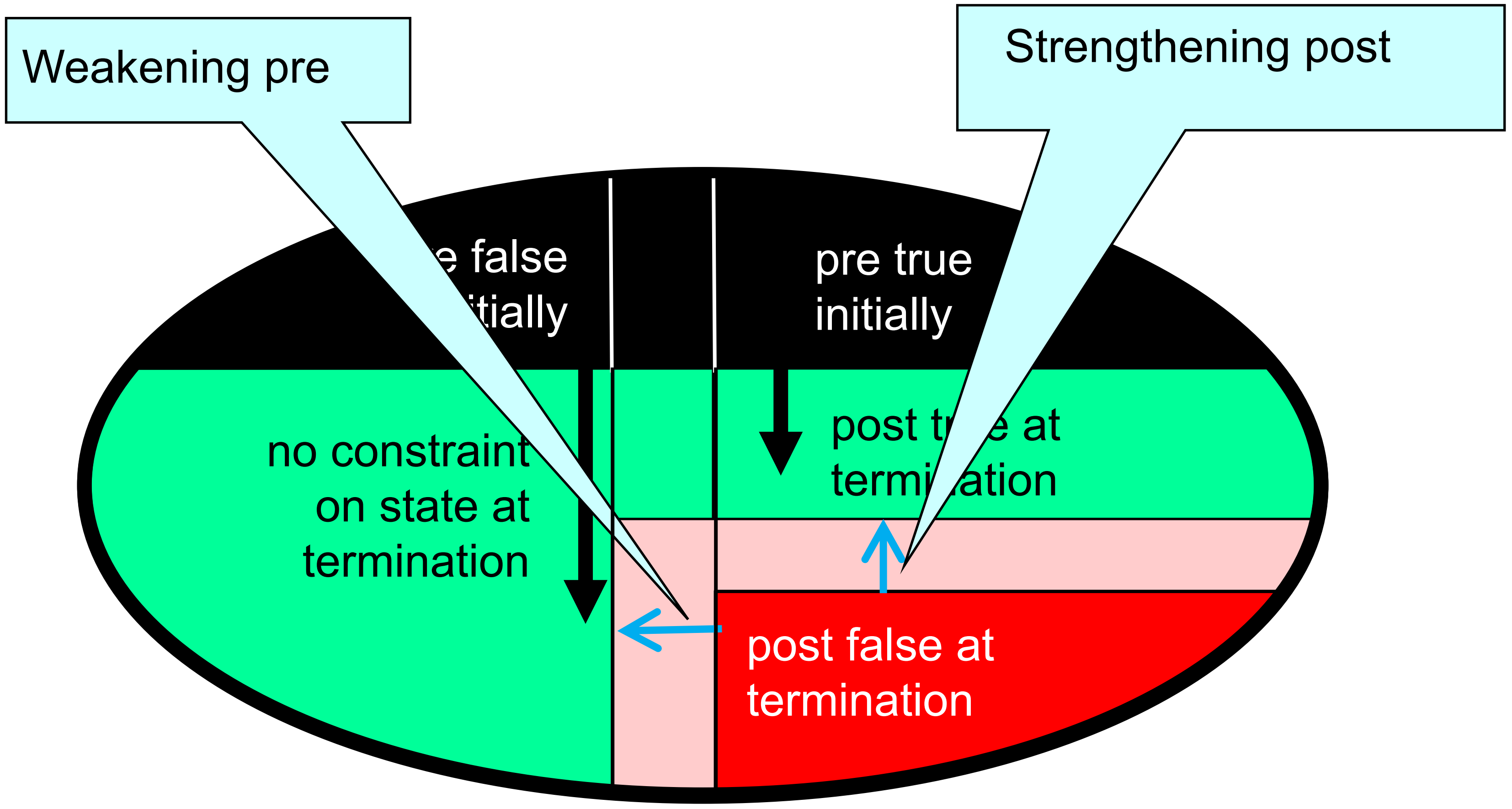
rest’ < divisor

Guarantee with respect to the state at the moment of termination

Semantics of pre-post specifications



Refinement in pre-post



Weakening the pre-condition (the assumption)

```
Integer division
    var dividend, divisor, quotient, rest : Nat

    pre   true

    post
          if divisor ≠ 0 then
             ( dividend = (quotient' * divisor) + rest' ) & rest' < divisor
          else quotient' = 0
```

Strengthening the post-condition (the guarantee)

Integer division

var dividend, divisor, quotient, rest : Nat

pre divisor $\neq$ 0

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

rest’ < divisor & dividend’ = dividend &

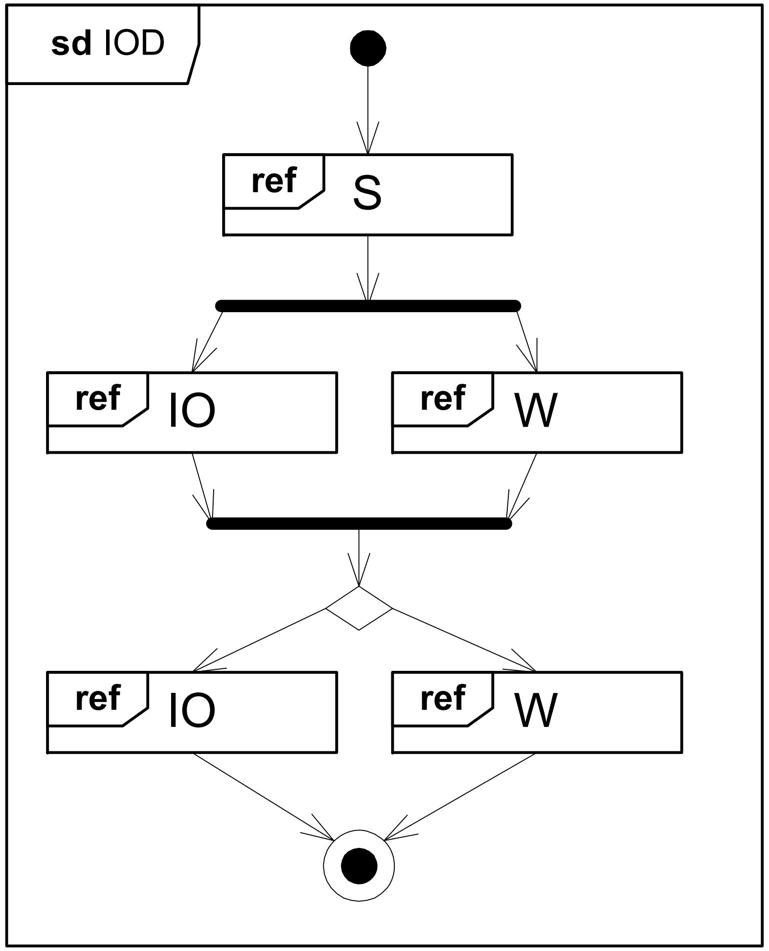
divisor’ = divisor

Refinement in UML

Motivation

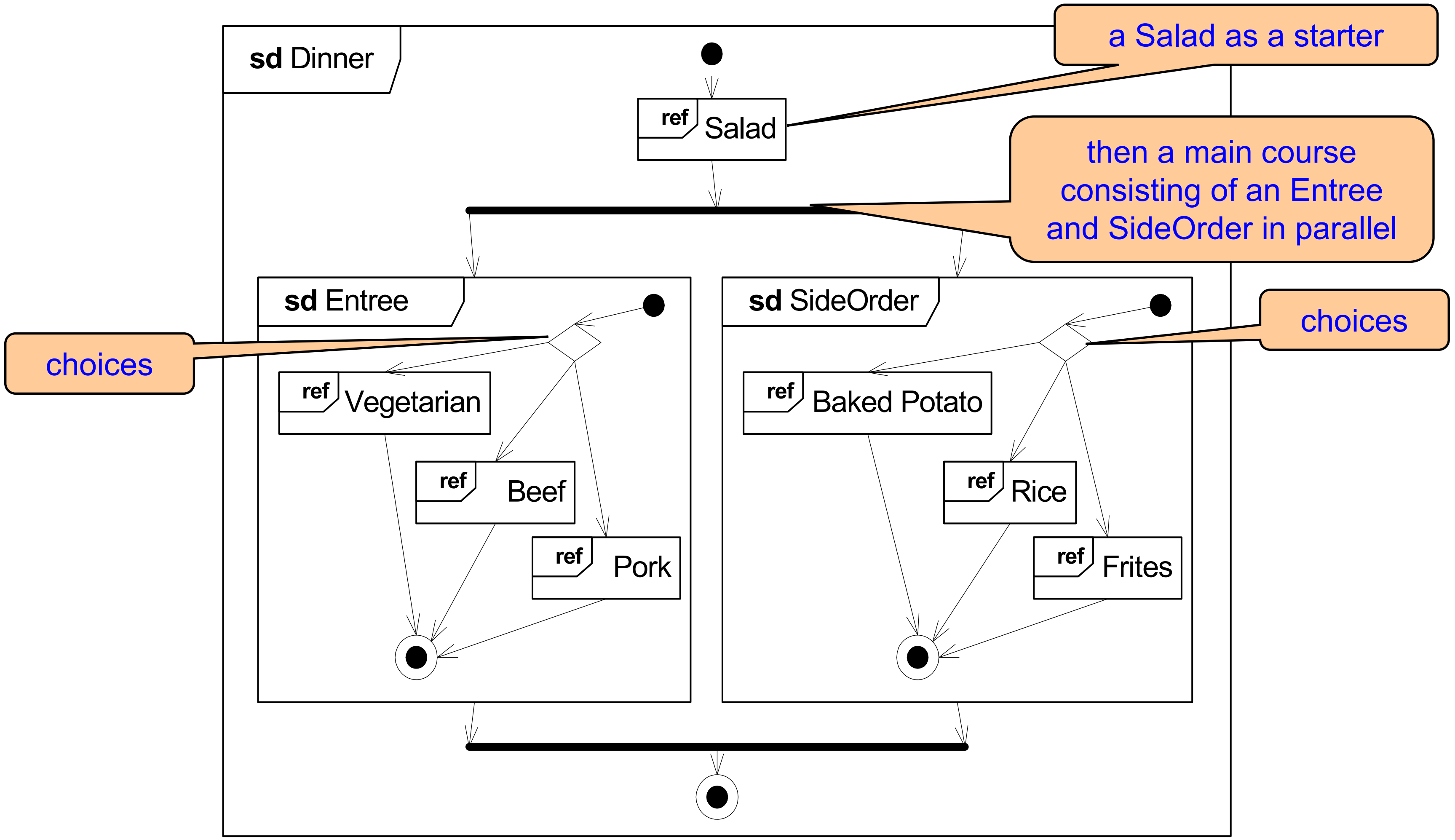
- Exploit classical theory of refinement in a practical UML setting
  - From theory to practice, and not the other way around
- Sequence diagrams can be used to capture the meaning of other UML description techniques for behavior
- By defining refinement for sequence diagrams we therefore implicitly define refinement for UML

Interaction overview diagram

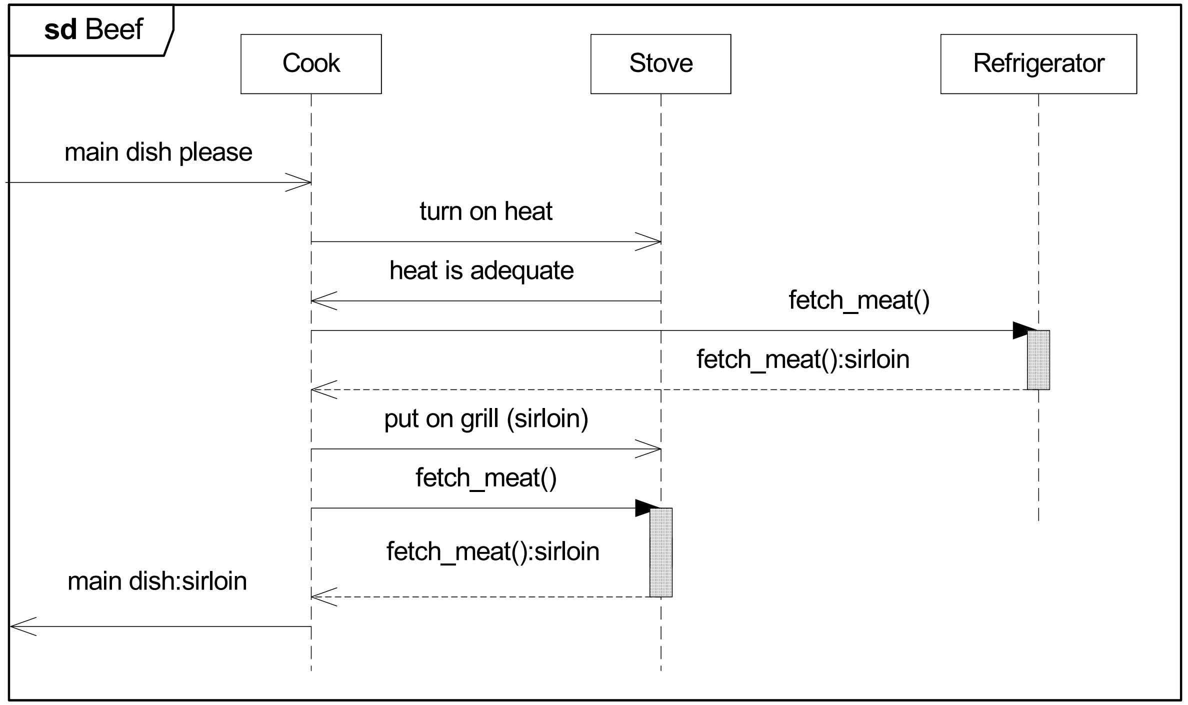


S **seq** (IO **par** W) **seq** (IO **alt** W)

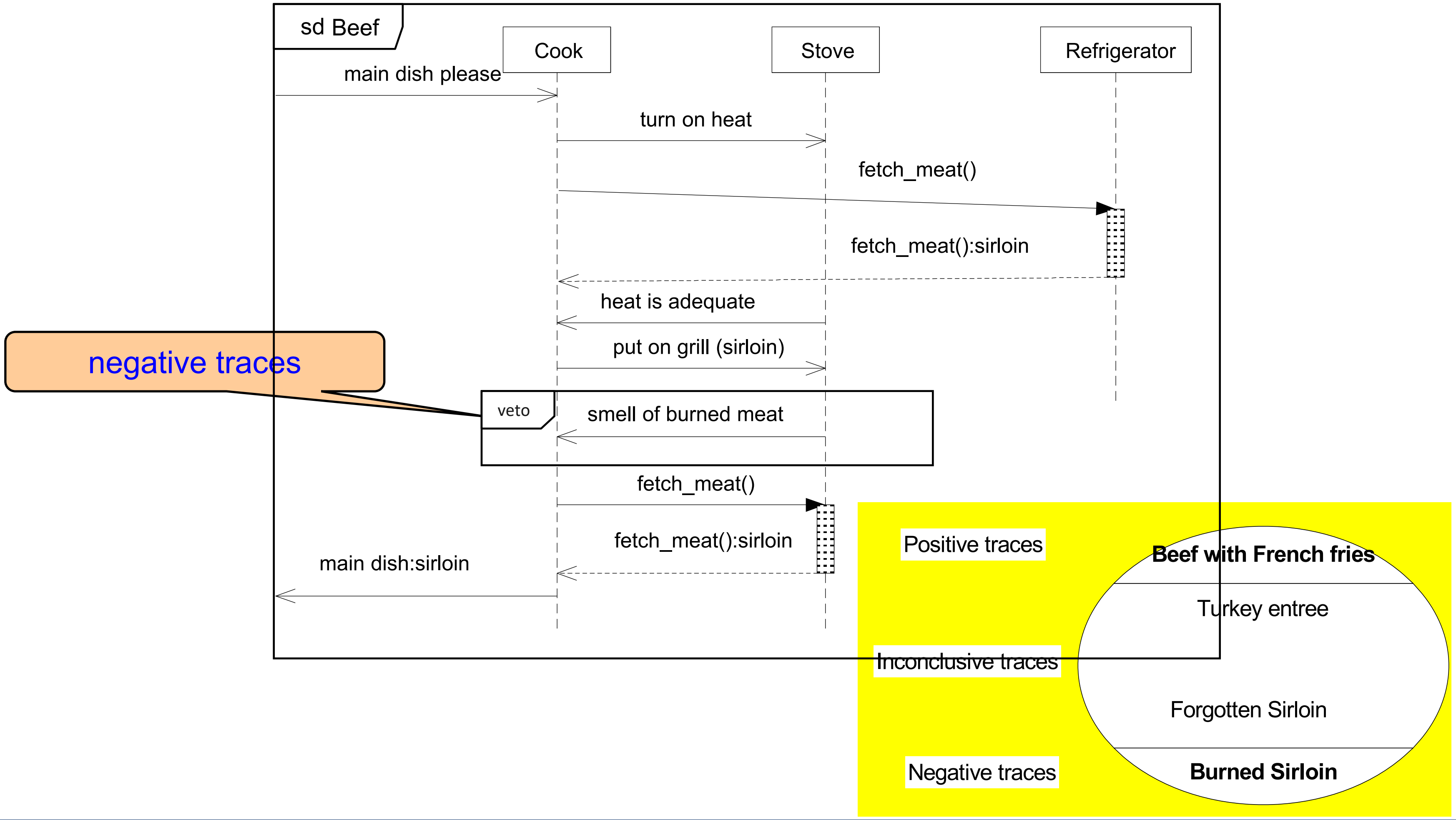
Dinner


sd Dinner
ref Salad
sd Entree
ref Vegetarian
ref Beef
ref Pork
sd SideOrder
ref Baked Potato
ref Rice
ref Frites
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

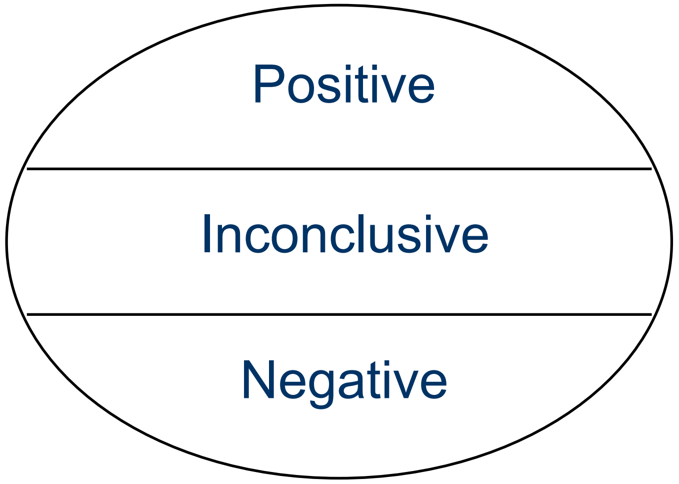


Potential negative Beef experiences



Positive, negative and inconclusive behaviour

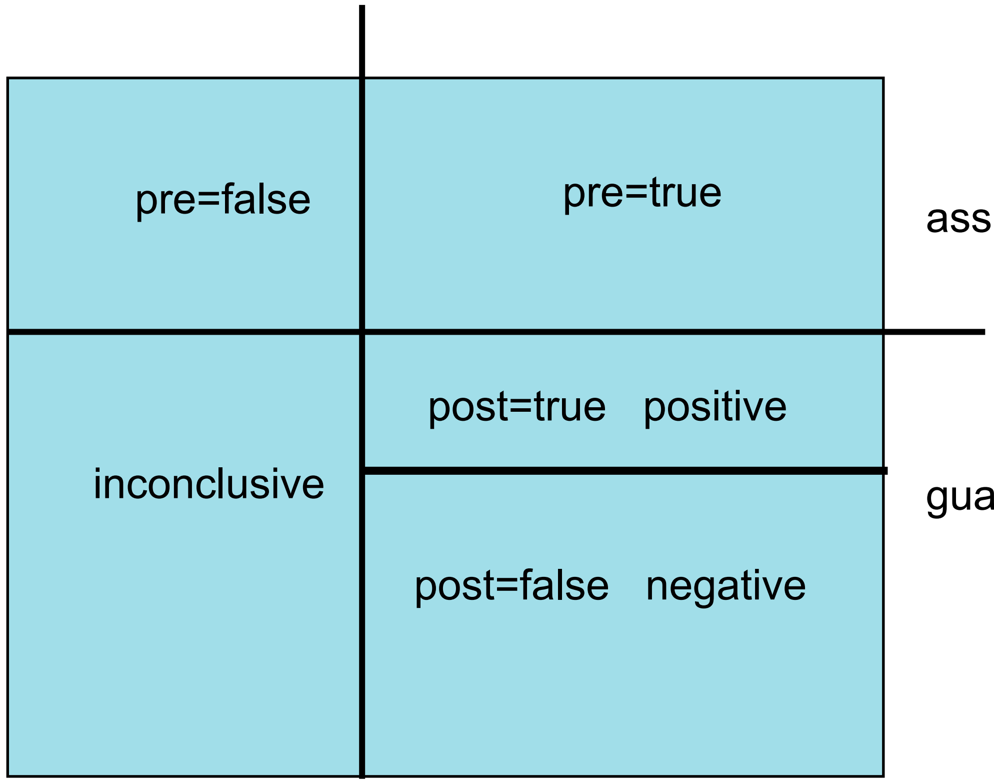
- Each positive execution is represented by a trace
- Each negative execution is represented by a trace
- All other traces over the actual alphabet of events are inconclusive



Interaction obligation

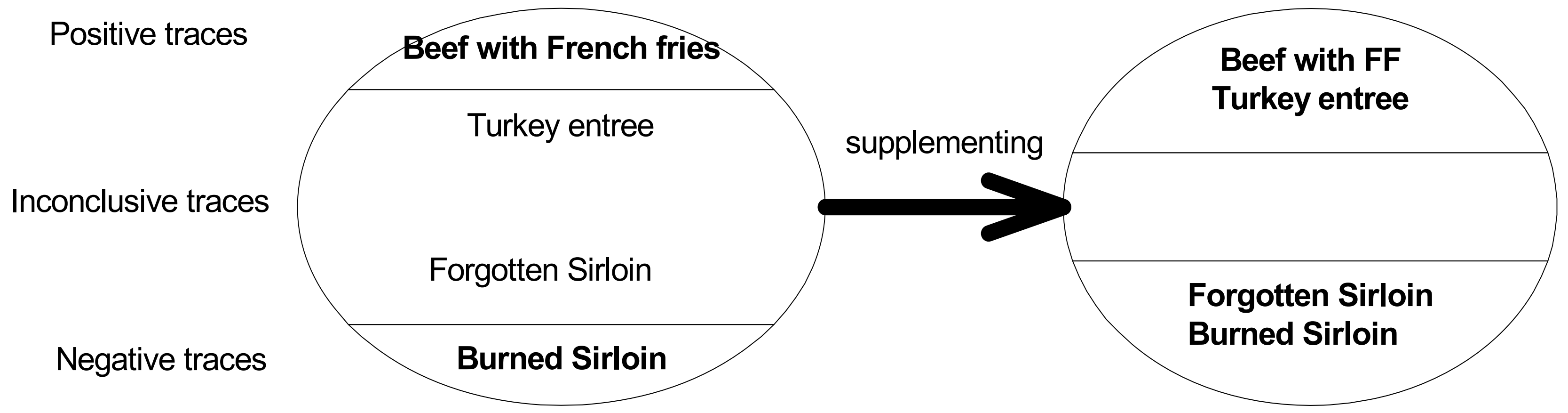
- The semantics of a basic sequence diagram is a pair of trace sets
  - (Positive, Negative)
- We refer to such pairs as interaction obligations
- For any sequence diagram S we use [[S]] to denote its interaction obligation

Comparing UML with pre-post

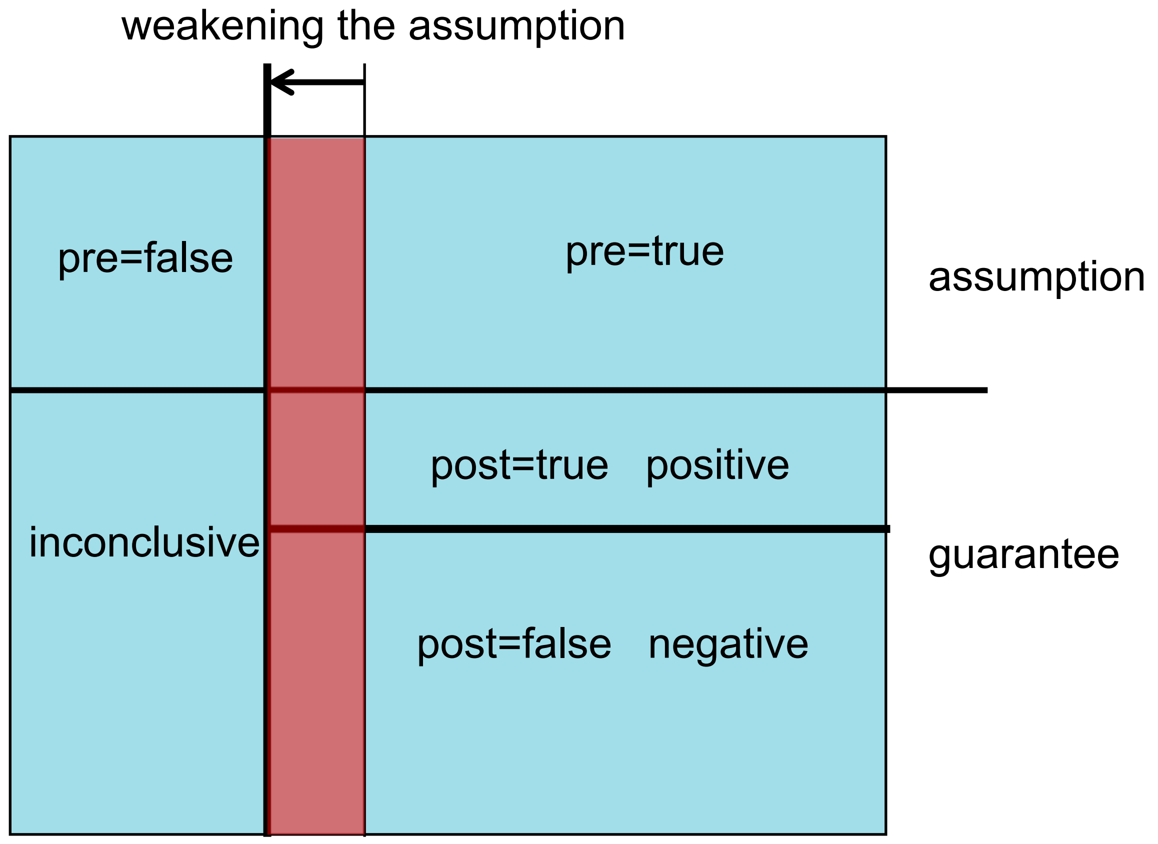


Weakening pre is supplementing in UML

- 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

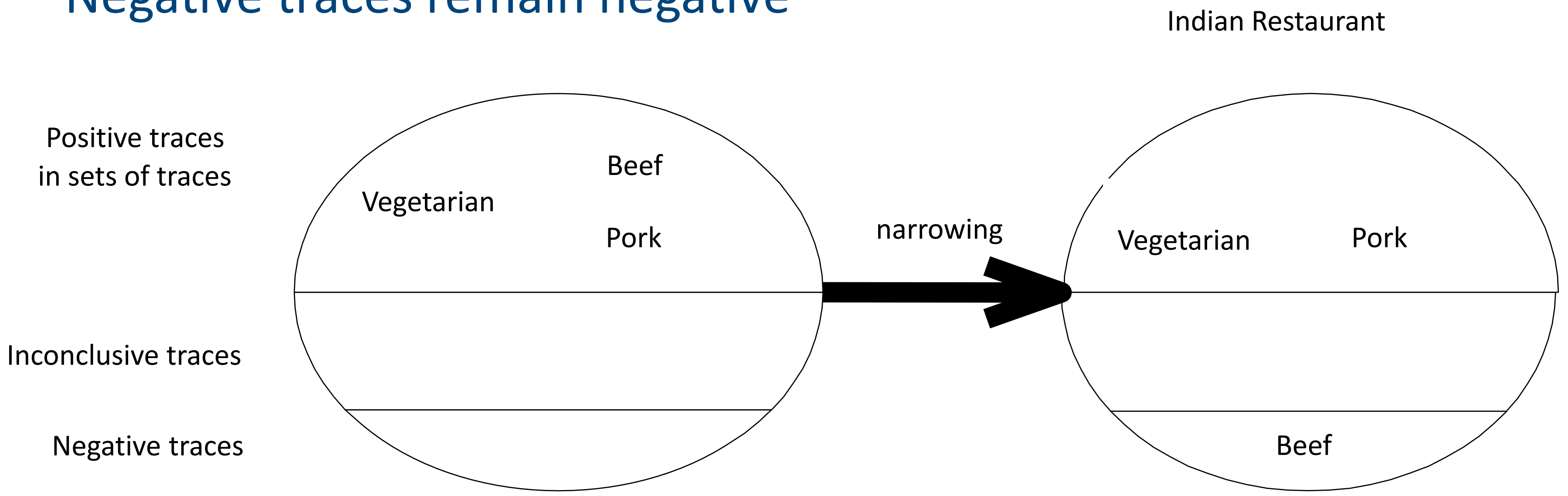


Supplementing in pre-post

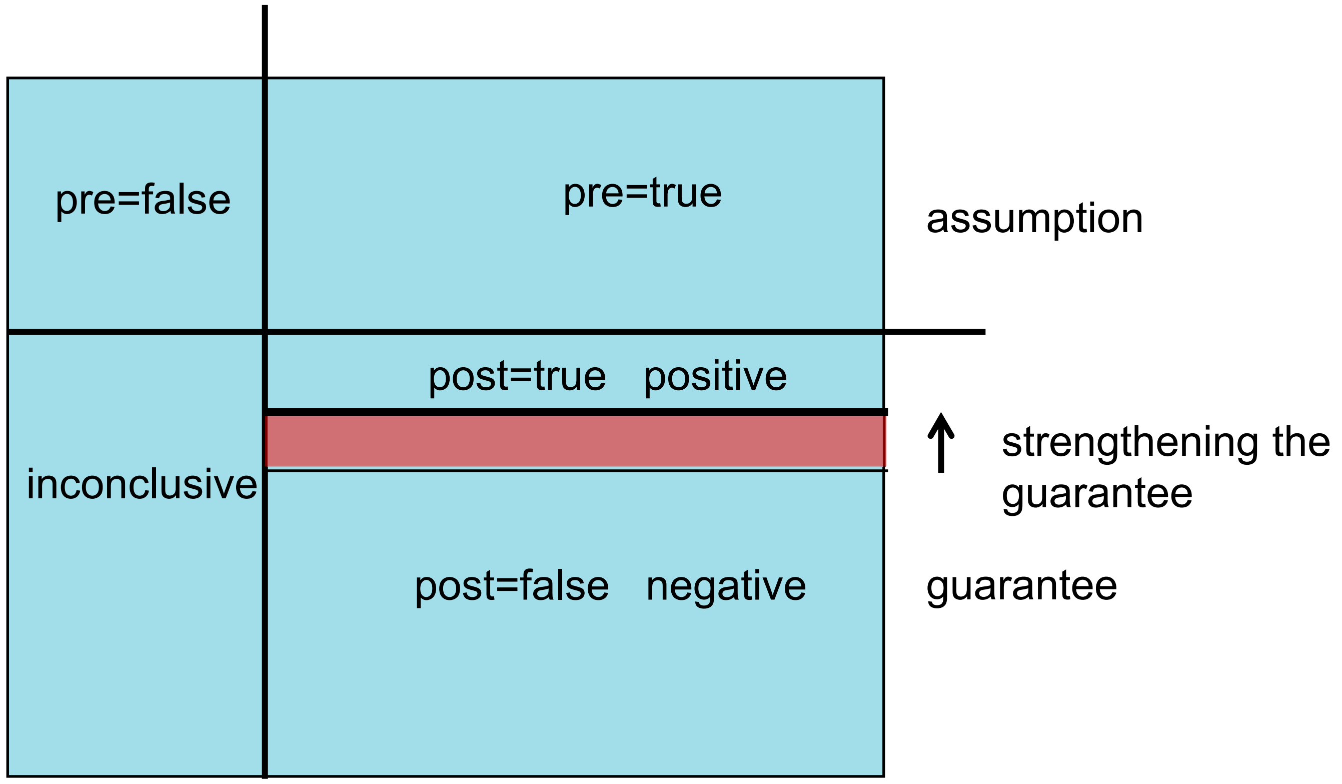


Strengthening the post is narrowing in UML

- 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



Indirect definition of pre-post refinement in UML

A sequence diagram B is a 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

Direct definition of pre-post refinement in UML

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 UML formalized

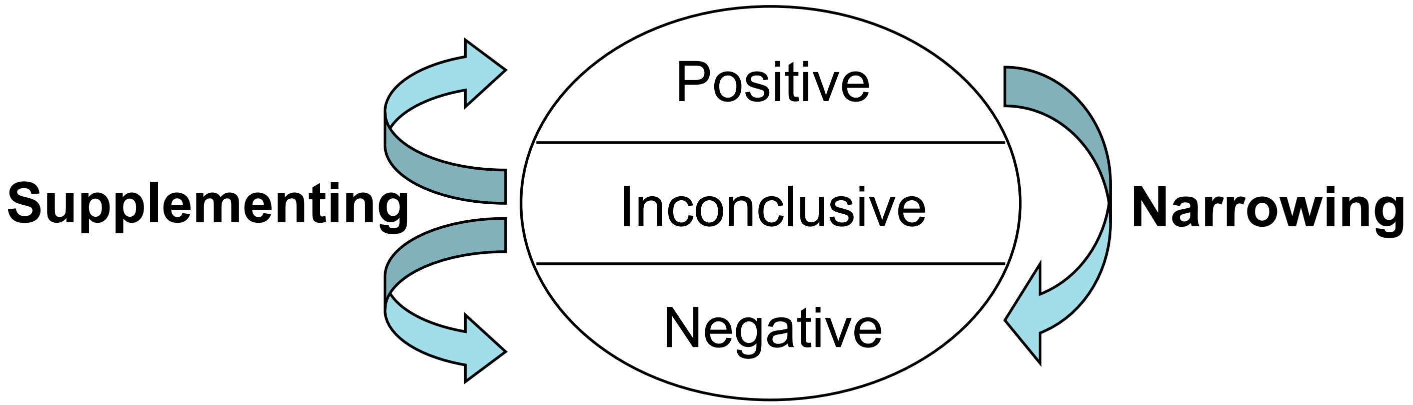
Let A and B be sequence diagrams such that

- [[A]] = (p,n)
- [[B]] = (p',n')

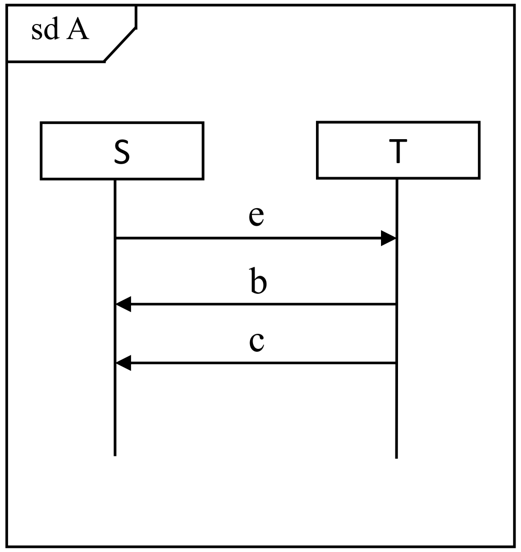
Then B is a refinement of A if

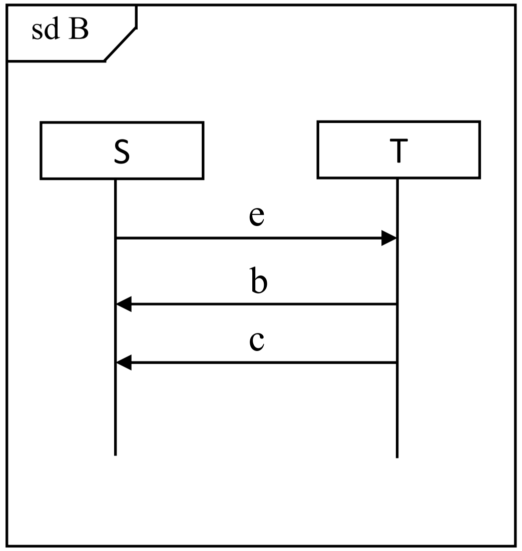
- n is a subset of or equal to n' ($n \subseteq n'$)
- p is a subset of or equal to the union of p' and n' ($p \subseteq p' \cup n'$)

Refinement in UML illustrated graphically

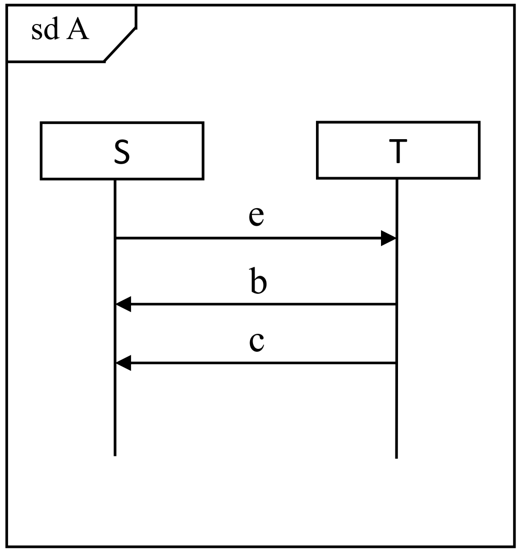


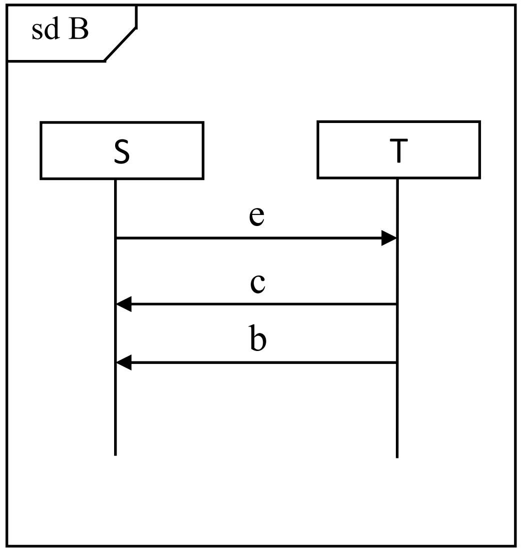
Is B a refinement of A?



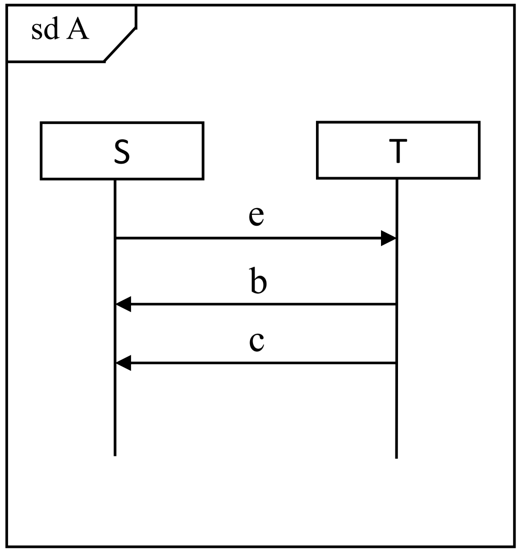


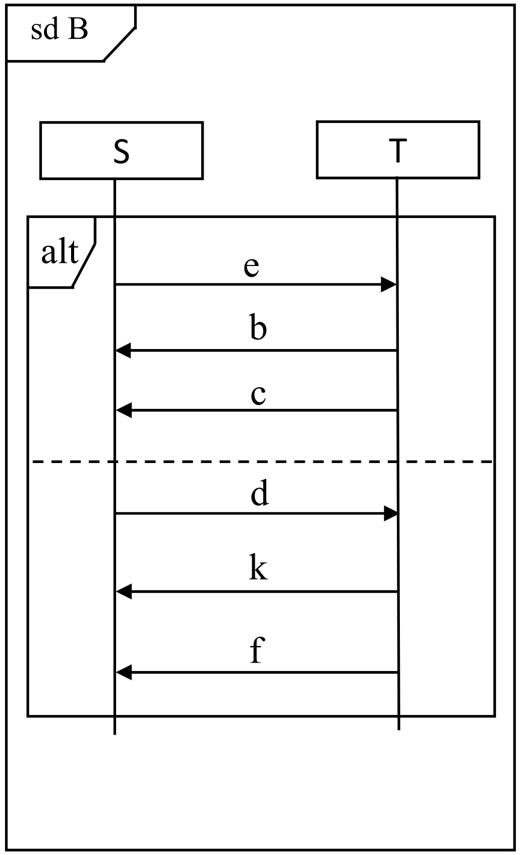
Is B a refinement of A?



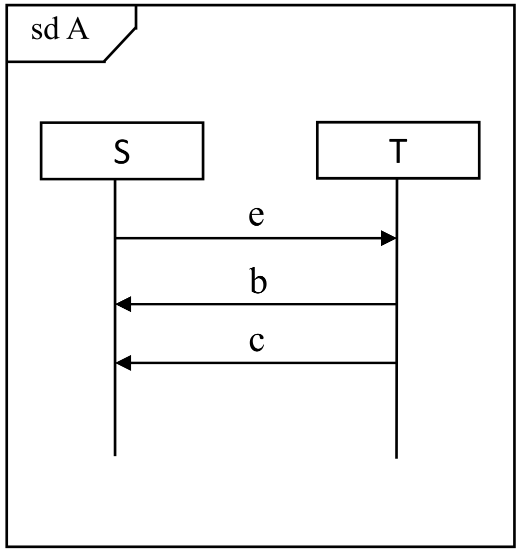


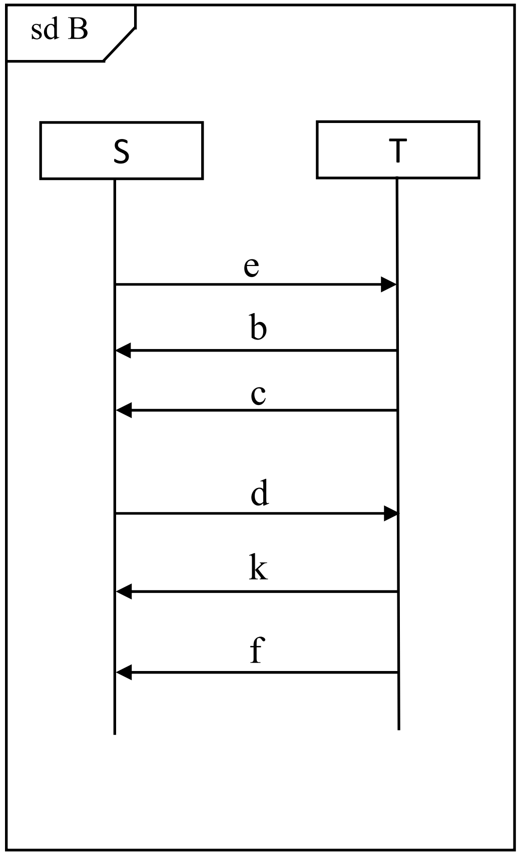
Is B a refinement of A?





Is B a refinement of A?





Is B a refinement of A?