Overview

- Obligatory Exercise No. 2
- 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. 2

- Should be solved individually by each student
- The exercise will be made available next week

- The deadline is October 25, 16.00 PM
- You should send your individual solutions by email to kst@sintef.no as an attachment in pdf-format

November 2:
- We will walk through the obligatory exercise and return the individual solutions in the group session October 2

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

SD S

L1

L2

x

Instance-line
Component
Message
Input event ?x
Output event !x
Weak sequencing

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
  
  \(<e_1, e_2, e_3, e_4, e_4, e_1, e_2, e_5, \ldots>\>

- An event represents 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, !b, !c, !d, ?c, ?d>
<!a, !a, !b, !c, ?b, ?c, !d, ?d>
<!a, !a, !b, !c, ?b, ?c, !d, ?d>
```

Alternative

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

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

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

![Diagram showing sequence of events in cooking a main dish, with positive traces, negative traces, and inconclusive traces highlighted.]

- Positive traces: main dish: sirloin
- Negative traces: smell of burned meat
- Inconclusive traces: heat is adequate

**Beef with French fries**
- Turkey entree
- Forgotten Sirloin
- Burned Sirloin
Pre-post specifications

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

\[
\text{var } \text{dividend, divisor, quotient, rest : Nat}
\]

\[
\text{pre } \text{divisor} \neq 0 \quad \text{Assumption about the state at the moment the execution is initiated}
\]

\[
\text{post} \ ( \text{dividend} = (\text{quotient'} \times \text{divisor}) + \text{rest'}) \land \text{rest'} < \text{divisor} \quad \text{Guarantee with respect to the state at the moment of termination}
\]

Semantics of pre-post specifications

- **pre false initially**
  - no constraints on state at termination
  - Legal, but arbitrary behavior

- **pre true initially**
  - post true at termination
  - Legal system behavior

- **post false at termination**
  - Illegal system behavior
Comparing STAIRS with pre-post

<table>
<thead>
<tr>
<th></th>
<th>pre=false</th>
<th>pre=true</th>
</tr>
</thead>
<tbody>
<tr>
<td>inconclusive</td>
<td>post=false</td>
<td>positive</td>
</tr>
<tr>
<td></td>
<td>post=true</td>
<td>negative</td>
</tr>
</tbody>
</table>

Refinement in pre-post

- Weakening pre
- Strengthening post

- pre false initially
- post true at termination
- post false at termination
- no constraint on state at termination
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

<table>
<thead>
<tr>
<th>pre=false</th>
<th>pre=true</th>
</tr>
</thead>
<tbody>
<tr>
<td>inconclusive</td>
<td>post=true positive</td>
</tr>
<tr>
<td>post=false negative</td>
<td></td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>pre=false</th>
<th>pre=true</th>
</tr>
</thead>
<tbody>
<tr>
<td>inconclusive</td>
<td>positive</td>
</tr>
<tr>
<td>post=false</td>
<td>negative</td>
</tr>
</tbody>
</table>

Assumption

strengthening the 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?

![Diagram showing sequence diagrams A and B]
Is B a refinement of A?

[Diagram showing state diagrams for A and B, comparing their states and transitions]
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

- An interaction obligation \( o'=(p',n') \) is a refinement of an interaction obligation \( o=(p,n) \) iff
  - \( n \subseteq n' \)
  - \( p \subseteq p' \cup n' \)
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 \textit{alt} and \textit{xalt}

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

- Mandatory non-determinism captured by \textit{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

Positive
Inconclusive
Negative
alt vs xalt

- Assume
  \[[ d1 \] = \{(p1, n1)\} \quad \text{[} d2 \] = \{(p2, n2)\}

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

- xalt specifies mandatory behaviour:
  \[[ d1 \text{ xalt } d2 \] = \[[ d1 \] U \[[ d2 \]
  = \{(p1,n1)\} U \{(p2,n2)\}

Example: Network communication
alt vs xalt

Mandatory requirements STAIRS


NOTE:
- Next Friday: Refinement III