Refining models expressed in UML sequence diagrams

October 7, 2005
– STAIRS –
Steps to Analyze Sequence Diagrams with Refinement Semantics
Motivation

- Make use of classical refinement theory in a practical UML setting
  - From theory to practice, and not the other way around
- We aim to explain how classical theory of refinement can be used to refine specifications expressed with the help
- Sequence diagrams can be used to explain other kinds of UML diagrams
- By defining refinement for sequence diagrams we implicitly define refinement for the UML as a whole
Traces

- Traces are used to represent system runs mathematically
- In the literature there are many different kinds of traces
- INF 5150 traces are sequences of events

\[ <e_1, e_2, e_3, e_4, e_4, e_1, e_2, e_5, \ldots \ldots \ldots > \]

- Events are instantaneous
- The number of events in a trace may be finite
  - may be caused by: termination, deadlock, infinite waiting, system crash
- The number of events in a trace may be infinite
  - May be cause by: non-termination, livelock, non-termination by purpose
Weak sequencing

\[ \langle !x, ?x, !y, ?y \rangle \]
\[ \langle !x, !y, ?x, ?y \rangle \]
Positive and negative traces

What are the positive and negative traces of this diagram?
Semantics of sequence diagrams (without xalt)

- The formal semantics of a sequence diagram (without xalt) $d$ is a pair $(P,N)$ of sets of traces such that
  - $P$ contains exactly the positive traces of $d$
  - $N$ contains exactly the negative traces of $d$

- For any diagram $d$, we use $[[d]]$ to denote its semantics

- The same trace may be both positive and negative with respect to the same diagram
  - But if this is the case, you have probably not specified what you intended to specify

- A trace that is neither positive nor negative for a sequence diagram $d$ is inconclusive with respect to $d$
Pre-post specifications

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

---

### Integer division

```latex
\begin{align*}
\text{var } & \text{dividend, divisor, quotient, rest : Nat} \\
\text{pre } & \text{divisor} \neq 0 \\
\text{post } & (\text{dividend} = (\text{quotient}' \times \text{divisor}) + \text{rest}') \land \\
& \text{rest}' < \text{divisor}
\end{align*}
```

- **Assumption about the state at the moment the execution is initiated:**
  - `divisor \neq 0`

- **Guarantee with respect to the state at the moment of termination:**
  - `dividend = (quotient' \times divisor) + rest'` and `rest' < divisor`
Semantics for pre-post specification

Legal, arbitrary behavior

Illegal behavior

pre false at initiation

pre true at initiation

everything allowed

post true at the moment of termination

post false at the moment of termination
Property refinement for pre-post specifications

Weaken assumption

Strengthen guarantee

pre false at initiation

pre true at initiation

post true at the moment of termination

post false at the moment of termination

everything allowed
Property refinement of sequence diagrams

positive variations

negative traces

sd Beef

main dish please

Cook

Stove

Refrigerator

turn on heat

fetch_meat()

fetch_meat():sirloin

heat is adequate

put on grill (sirloin)

smell of burned meat

fetch_meat()

fetch_meat():sirloin

main dish:sirloin

Positive traces

Beef with French fries

Turkey entree

Forgotten Sirloin

Burned Sirloin

Negative traces

Inconclusive traces
Supplementing

- To supplement means reducing the set of inconclusive traces by reclassifying inconclusive traces as either positive or negative.
- The already positive traces remain positive.
- The already negative traces remain negative.
Narrowing

- To narrow means reducing the set of positive traces by reclassifying positive traces as negative
- The inconclusive traces remain inconclusive
- The already negative traces remain negative
INDIRECT DEFINITION: Property refinement in STAIRS

A sequence diagram B is a property 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
  \[ A \rightarrow C_1 \rightarrow C_2 \rightarrow \ldots \rightarrow C_n \rightarrow B \]  
each of which is either a supplementing or a narrowing
Is B a property refinement of A?
Is B a property refinement of A?
Is B a property refinement of A?
Is B a property refinement A?
Is B a property refinement of A?
DIRECT DEFINITION: Property refinement in STAIRS

- A sequence diagram B is a property 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
Property refinement of “tea”

What does it mean to
• supplement?
• narrow?
• both in one go?
Distinguishing two different types of non-determinism

- Classical distinction between mandatory and potential non-determinism

- Most specification languages cannot distinguish mandatory and potential non-determinism
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.
General STAIRS-semantics for sequence diagrams

- The semantics of a sequence diagram is a set of pairs of sets of traces
  - \{(P_1,N_1), (P_2,N_2),..., (P_n,N_n)\}
  - where for each index j
    - \(N_j\) is a set of negative traces
    - \(P_j\) is a set of positive traces

- We refer to these pairs as "interaction obligations"

- Each interaction obligation must be fulfilled by a correct implementation
The semantics of alt and xalt

\[
[d_1 \text{ alt } d_2] \overset{\text{def}}{=} \{(p_1 \cup p_2, n_1 \cup n_2) \mid (p_1, n_1) \in [d_1] \land (p_2, n_2) \in [d_2]\}
\]

\[
[d_1 \text{ xalt } d_2] \overset{\text{def}}{=} [d_1] \cup [d_2]
\]
Property refinement of an interaction obligation

\[(p_1, n_1) \rightsquigarrow (p_2, n_2) \text{ iff } n_1 \subseteq n_2 \land p_1 \subseteq p_2 \cup n_2\]

An interaction obligation \((p_2, n_2)\) is a property refinement of an Interaction obligation \((p_1, n_1)\) if and only if

- \(n_1\) is a subset of or equal to \(n_2\), and
- \(p_1\) is a subset of or equal to the union of \(p_2\) and \(n_2\)
Property refinement in STAIRS – general case

A sequence diagram $d'$ is a refinement of a sequence diagram $d$, written $d \leadsto d'$, iff

$$\forall o \in [d] : \exists o' \in [d'] : o \leadsto o'$$
Property refinement is not enough

**PROBLEM I:** Histories of bits are different from histories of natural numbers

**PROBLEM II:** Only eight natural numbers can be represented by three bits
Requirement: Composition of concrete diagram and translation is a property refinement of abstract diagram.
Interface refinement

Sd AddOnNat

adder

n
l

r:=n+l

Sd AddOnBin+oversettelse

natbin-converter

n

convert nat2bin

n1
n2
n3

convert nat2bin

l1
l2
l3

(r1,r2,r3):=binadd(n1,n2,n3,l1,l2,l3)

convert bin2nat

r

r1
r2
r3

Øystein Haugen

Ketil Stølen
Interface refinement generalizes property refinement

In STAIRS, interface refinement is a special case of detailing.

interface refinement

property refinement

$=\quad$ Interface refinement wrt identity translation
Interface refinement is not enough

PROBLEM I: Histories of bits are different from histories of natural numbers
Solved by interface refinement

PROBLEM II: Only eight natural numbers can be represented by three bits
Remains to be solved
Conditional refinement

Requirement: Composition of concrete spec. and translation is a property refinement of abstract spec. wrt. generator

Abstract diagram

Concrete diagram

Input-output translation

Input generator

Abstract input

Abstract output

Concrete input

Concrete output
Conditional interface refinement (another form of detailing)

Sd AddOnBin + oversettelse

Sd AddOnNat

Constraints
0 < n, l < 5

Characterizes relevant input

r := n + l

(r1, r2, r3) := binadd(n1, n2, n3, l1, l2, l3)

r1
r2
r3
Literature on refinement

- The classics:

- STAIRS is to a large extent based on ideas taken from:
Security analysis

First of three lectures October 28

We will
- classify dependability concepts
- introduce motivate and explain a basic terminology for risk management and risk analysis
- relate risk management to system development
- describe and motivate the different processes of risk management and how they interact