UpSTAIRS with Sequence Diagrams

Øystein Haugen,
Ragnhild Kobro Runde, Ketil Stølen

University of Oslo and SINTEF ICT,
Norway
Overview

• Interactions and trace semantics
• Interactions as example runs
• Underspecification and nondeterminism
• Refinement
• Data and guards
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:
  - $< !x, ?x, !y, ?y >$
  - $< !x, !y, ?x, ?y >$
Alternatives

sd S

A

B

x

alt

y

z

- < !x, ?x, !y, ?y >
- < !x, !y, ?x, ?y >
- < !x, ?x, !z, ?z >
- < !x, !z, ?x, ?z >

• S specifies the four traces:

  First alternative

  Second alternative
Example: Network communication

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

Positive:

<!m_{AS}, ?m_{AS}, !m_{SB}, ?m_{SB}>

Negative:

<!m_{AS}, ?m_{AS}, !m_{SB}, ?m_{SB}, !m_{SB}, ?m_{SB}>

<!m_{AS}, ?m_{AS}, !m_{SB}, !m_{SB}, ?m_{SB}>

Formally:

\[(p1,n1) \succ (p2,n2) = (p1 \succ p2, (p1 \succ n2) U (n1 \succ p2) U (n1 \succ n2))\]

Note:

- Inconclusive + positive/negative = inconclusive
- Positive + negative = negative
Overview

• Interactions and trace semantics
• Interactions as example runs
• Underspecification and nondeterminism
• Refinement
• Data and guards
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!
alt vs xalt

• Assume
  
  \[
  [[d_1]] = (p_1, n_1)
  \]
  
  \[
  [[d_2]] = (p_2, n_2)
  \]

• alt specifies potential behaviour:
  
  \[
  [[d_1 \text{ alt } d_2]]
  = [[d_1]] + [[d_2]]
  = (p_1 \cup p_2, n_1 \cup n_2)
  \]

• xalt specifies mandatory behaviour:
  
  \[
  [[d_1 \text{ xalt } d_2]]
  = [[d_1]] \cup [[d_2]]
  = (p_1, n_1) \cup (p_2, n_2)
  \]
Example: Network communication

- **cs C**
  - **A**: sender
  - **S**: network
  - **B**: receiver

- **cs S**
  - **N1**: N
  - **N2**: N
  - **N3**: N
  - **N4**: N
  - **G**: N
alt vs xalt

S:network

sd S_Comm

A: sender

G:N
N1:N
N2:N
N3:N
N4:N
B: receiver

xalt

A->G->N1->B
A->G->N2->N3->B
A->G->N2->N4->B

Everything else
Overview

• Interactions and trace semantics
• Interactions as example runs
• Underspecification and nondeterminism
• Refinement
• Data and guards
Refinement in STAIRS

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

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

sd S_Comm

A: sender  G:N  N1:N  N2:N  N3:N  N4:N  B: receiver

assert

xalt

<table>
<thead>
<tr>
<th>m</th>
<th>m</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>m</td>
<td>m</td>
</tr>
</tbody>
</table>

alt

<table>
<thead>
<tr>
<th>m</th>
<th>m</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>m</td>
<td>m</td>
</tr>
</tbody>
</table>

A->G->N1->B

Everything else

Everything else

A->G->N2->N3->B

A->G->N2->N4->B

Everything else

Everything else
Overview

• Interactions and trace semantics
• Interactions as example runs
• Underspecification and nondeterminism
• Refinement
• Data and guards
Including data

- Two special events:
  - write (for assignments)
  - check (for constraints)

- A state is a total function
  \( \sigma \in \text{Var} \rightarrow \text{Val} \)

- For an expression \( \text{expr} \), \( \text{expr}(\sigma) \) denotes its value in \( \sigma \).

- Assignment:
  \[
  [[\text{assign(var,expr)}]] =
  ( \{ \langle \text{write}(\sigma, \sigma') \rangle \mid \sigma'(\text{var}) = \text{expr}(\sigma) \} , \emptyset )
  \]
Constraints

\[[\text{constr}(c)]\] =  
( \{ \text{check}(\sigma) | c(\sigma) \} , \{ \text{check}(\sigma) | \neg c(\sigma) \} )
Guards

- A special kind of constraint
- May be overlapping
- Need not be exhaustive
Definition of guarded xalt

• Assume:
  
  \[ [[ d1 ]] = (p1,n1) \]
  
  \[ [[ d2 ]] = (p2,n2) \]

• Guarded xalt:
  
  \[ [[ g1->d1 \ xalt \ g2->d2 ]] = \]
  
  \[ [[ constr(g1) \ seq \ d1 ]] \ U \ [[ constr(g2) \ seq \ d2 ]] \]
Definition of guarded xalt

• Assume:
  \[[ d1 \]] = (p1,n1)
  \[[ d2 \]] = (p2,n2)

• Guarded xalt:
  \[[ g1->d1 \ xalt \ g2->d2] \] =
  ( \{ <\text{check}(\sigma)> \models p1 \mid g1(\sigma) \},
    \{ <\text{check}(\sigma)> \models p1 \mid \neg g1(\sigma) \} \cup
    \{ <\text{check}(\sigma)> \models n1 \mid g1(\sigma) \lor \neg g1(\sigma) \} \}
  \cup
  ( \{ <\text{check}(\sigma)> \models p2 \mid g2(\sigma) \},
    \{ <\text{check}(\sigma)> \models p2 \mid \neg g2(\sigma) \} \cup
    \{ <\text{check}(\sigma)> \models n2 \mid g2(\sigma) \lor \neg g2(\sigma) \} \)
sd S_Comm

A: sender
G: N
N1: N
N2: N
N3: N
N4: N
B: receiver

assert

xalt

alt
Narrowing by using guards

sd S_Comm

A: sender
G: N
N1: N
N2: N
N3: N
N4: N
B: receiver

assert

xalt

[N1 capacity ok]
m

m

[N2 capacity ok]
m

alt

A->G->[N1 ok]->N1->B
A->G->[N1 not ok]->N1->B
A->G->[N2 not ok]->N2->... Everything else
A->G->[N2 not ok]->N2->... Everything else
Narrowing by using guards

sd S_Comm

A:sender  G:N  N1:N  N2:N  N3:N  N4:N  B:receiver

A->G->[N2 ok]->N2->[N3 ok]->N3->B
A->G->[N2 ok]->N2->[N4 ok]->N4->B
A->G->[N2 ok]->N2->[N3 not ok and N4 not ok]
A->G->[N2 ok]->N2->[N3 not ok]->N3->B
A->G->[N2 ok]->N2->[N4 not ok]->N4->B

A->G->[N2 not ok]->N2->[N3 ok/not ok]->N3->B
A->G->[N2 not ok]->N2->[N4 ok/not ok]->N4->B
A->G->[N2 ok]->N2->[N3 not ok]->N3->B
A->G->[N2 ok]->N2->[N4 not ok]->N4->B

Everything else
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.
Literature on STAIRS

- 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

  - 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
http://heim.ifi.uio.no/~ragnhilk/stairs/

Thank you!