
Semantics of UML 2.0 Interactions with Variabilities

María Victoria Cengarle
TU München

Peter Graubmann
Siemens AG

Stefan Wagner
TU München

October 24, 2005

Introduction

Interactions with
Variabilities

Abstract Syntax

Denotational
Semantics

Case Study

Conclusions

Introduction

Interactions with Variabilities

Abstract Syntax

Denotational Semantics

Case Study

Conclusions

Introduction

Motivation
Concepts and Related
Work

Interactions with
Variabilities

Abstract Syntax

Denotational
Semantics

Case Study

Conclusions

Introduction

Introduction

Motivation

Concepts and Related Work

Interactions with Variabilities

Abstract Syntax

Denotational Semantics

Case Study

Conclusions

- System family development is a way of systematic planning and reuse of commonalities between different products
- Focus on variability in domain and application engineering
- UML interactions are a possibility to describe component behaviour
- Various forms (MSCs) are widely used in practice
- However, they lack an explicit notion of variability

Introduction

Motivation

Concepts and Related Work

Interactions with Variabilities

Abstract Syntax

Denotational Semantics

Case Study

Conclusions

- Constructs for an adequate language extension
- With formal semantics
- Uses UML but transferable to MSCs
- Operator variant that specifies optional behaviour depending on the system family configuration
- Operator repeat to reduplicate instances with message exchange patterns
- Variable scoping concept
- Ziadi et al. propose «optionalLifeline» and «variant»
- «variant» in Kobra
- We emphasise parameteric nature of variability and use operators
- Formal semantics

Introduction

**Interactions with
Variabilities**

Features

Variation Points and
Variants

Interactions with
Variants

Repeat

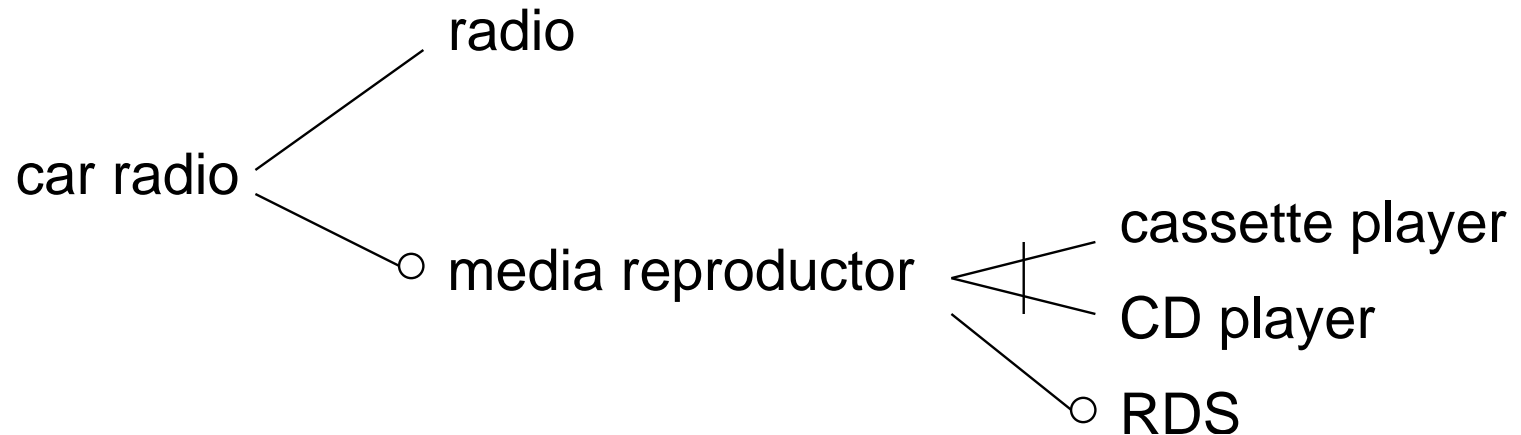
Abstract Syntax

Denotational
Semantics

Case Study

Conclusions

Interactions with Variabilities



- An essential aspect or characteristic of a system in a domain
- Distinctively identifiable abstractions
- A system family is a collection of software products that are similar but have varying features
- A feature can be mandatory, optional or alternative to other features
- Approaches (FODA, FORM) that organise features in and/or trees

Introduction

Interactions with Variabilities

Features

Variation Points and Variants

Interactions with Variants

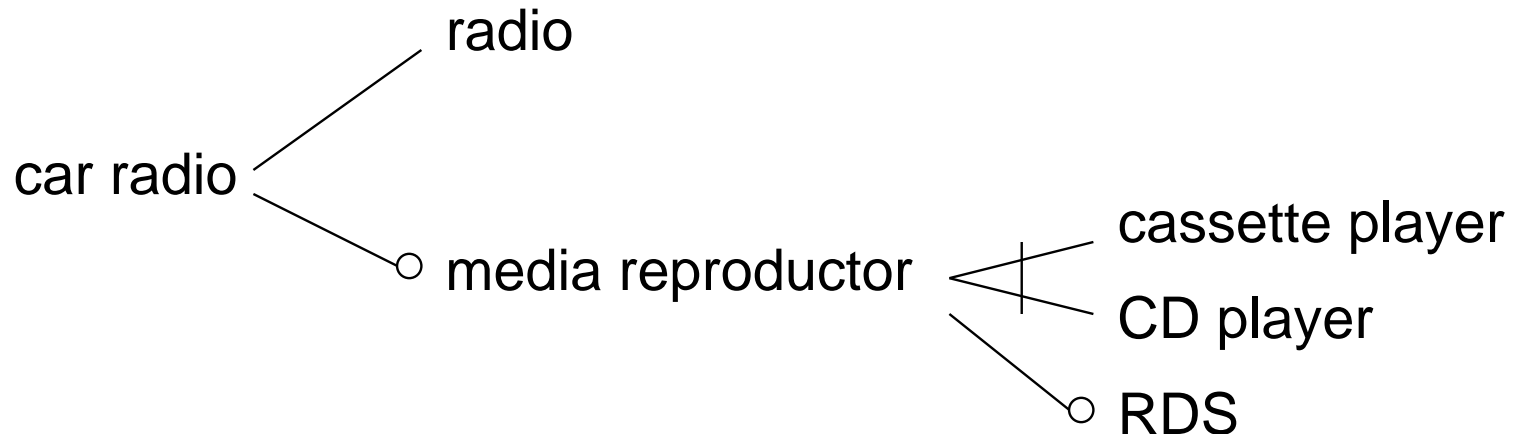
Repeat

Abstract Syntax

Denotational Semantics

Case Study

Conclusions



- In use cases variability is mirrored by *variation points*
- Location where a variation occurs
- Captured in one or more *variants*

Introduction

Interactions with Variabilities

Features

Variation Points and Variants

Interactions with Variants

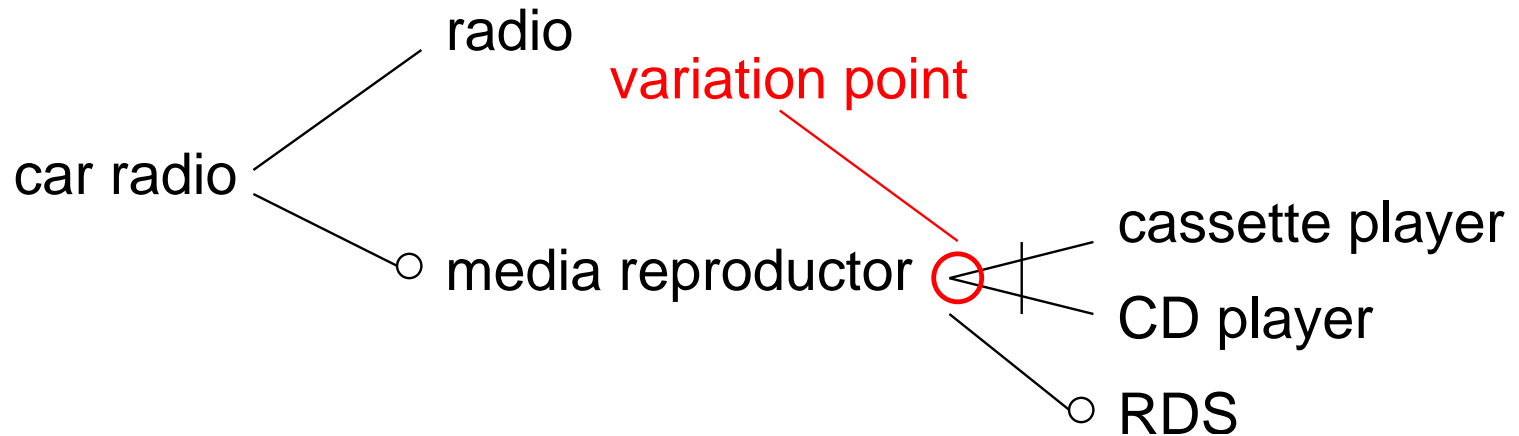
Repeat

Abstract Syntax

Denotational Semantics

Case Study

Conclusions



- In use cases variability is mirrored by *variation points*
- Location where a variation occurs
- Captured in one or more *variants*

Introduction

Interactions with Variabilities

Features

Variation Points and Variants

Interactions with Variants

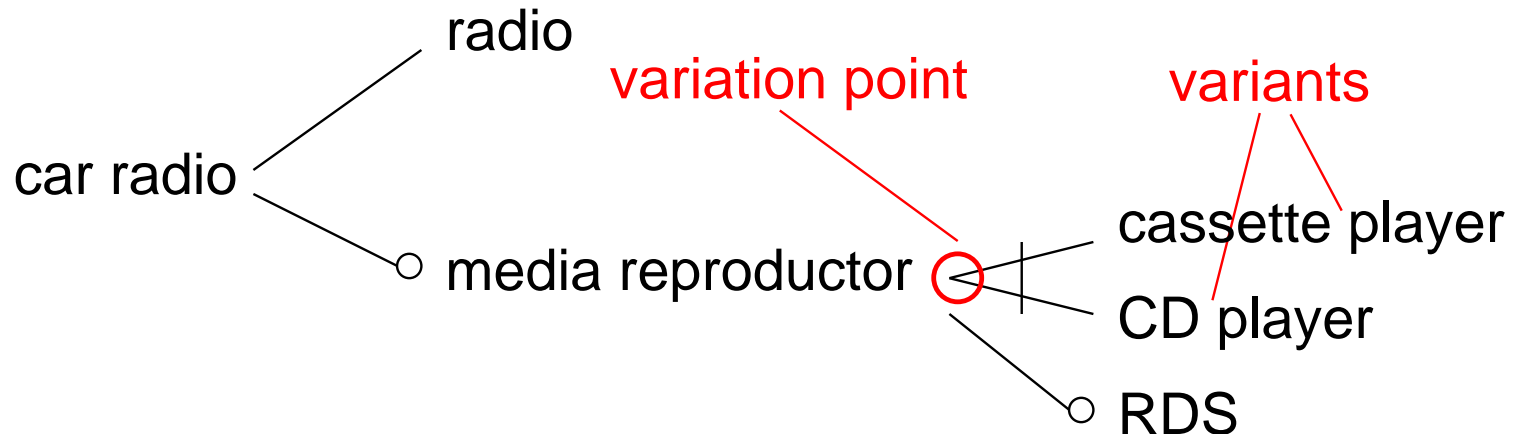
Repeat

Abstract Syntax

Denotational Semantics

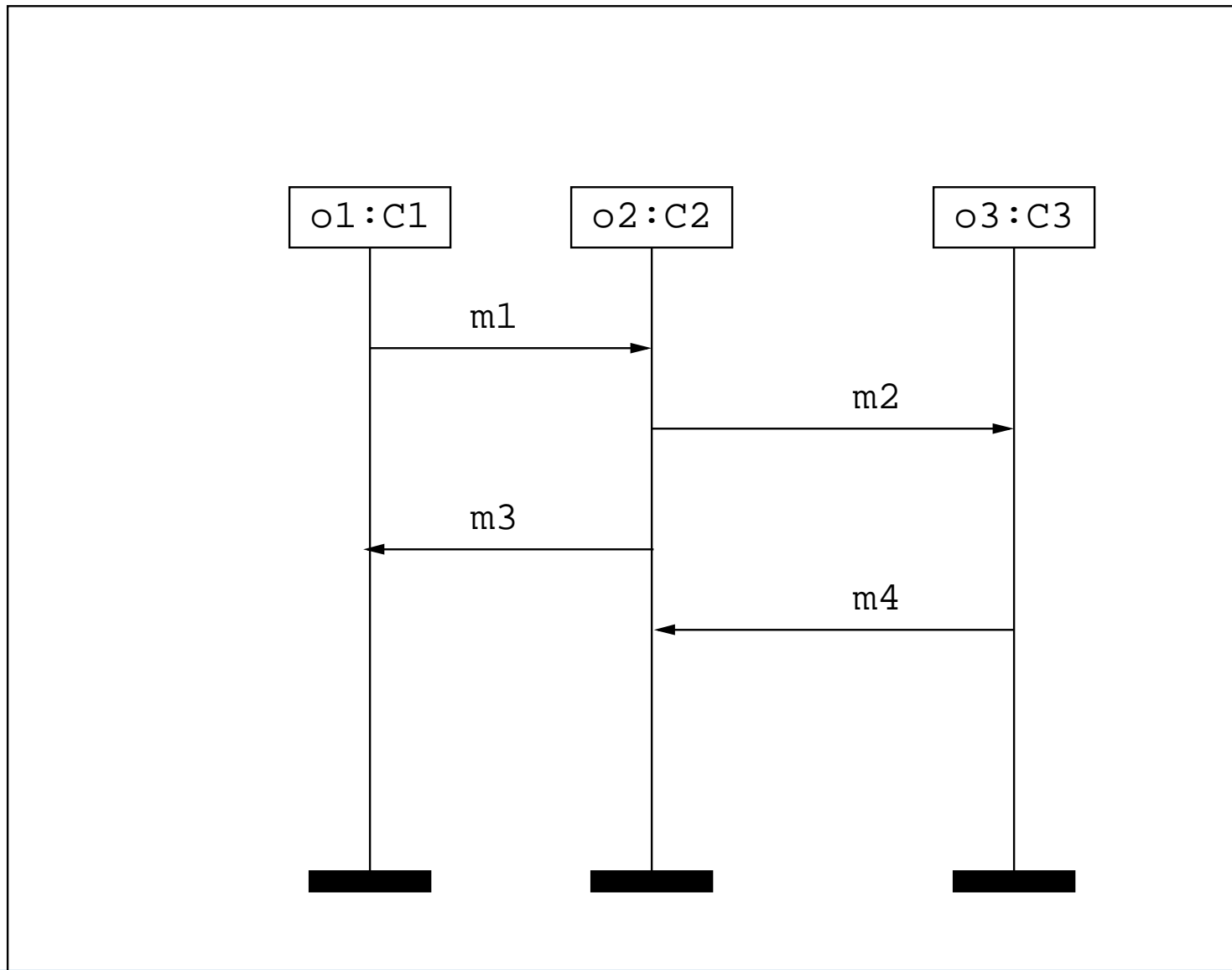
Case Study

Conclusions

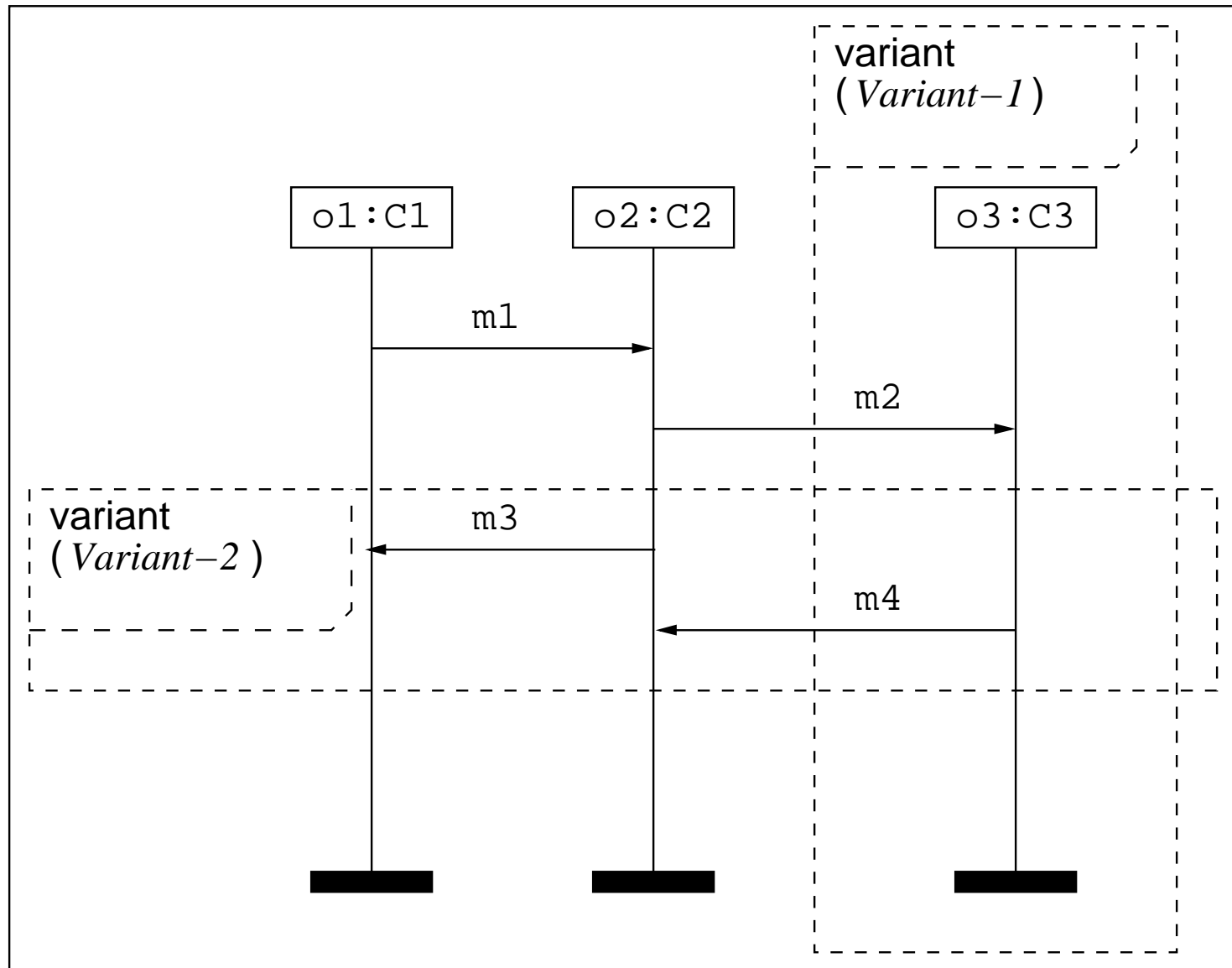


- In use cases variability is mirrored by *variation points*
- Location where a variation occurs
- Captured in one or more *variants*

- Introduction
- Interactions with Variabilities
- Features
- Variation Points and Variants
- Interactions with Variants**
- Repeat
- Abstract Syntax
- Denotational Semantics
- Case Study
- Conclusions



- Introduction
- Interactions with Variabilities
- Features
- Variation Points and Variants
- Interactions with Variants**
- Repeat
- Abstract Syntax
- Denotational Semantics
- Case Study
- Conclusions



Introduction

Interactions with Variabilities

Features
Variation Points and Variants

Interactions with Variants

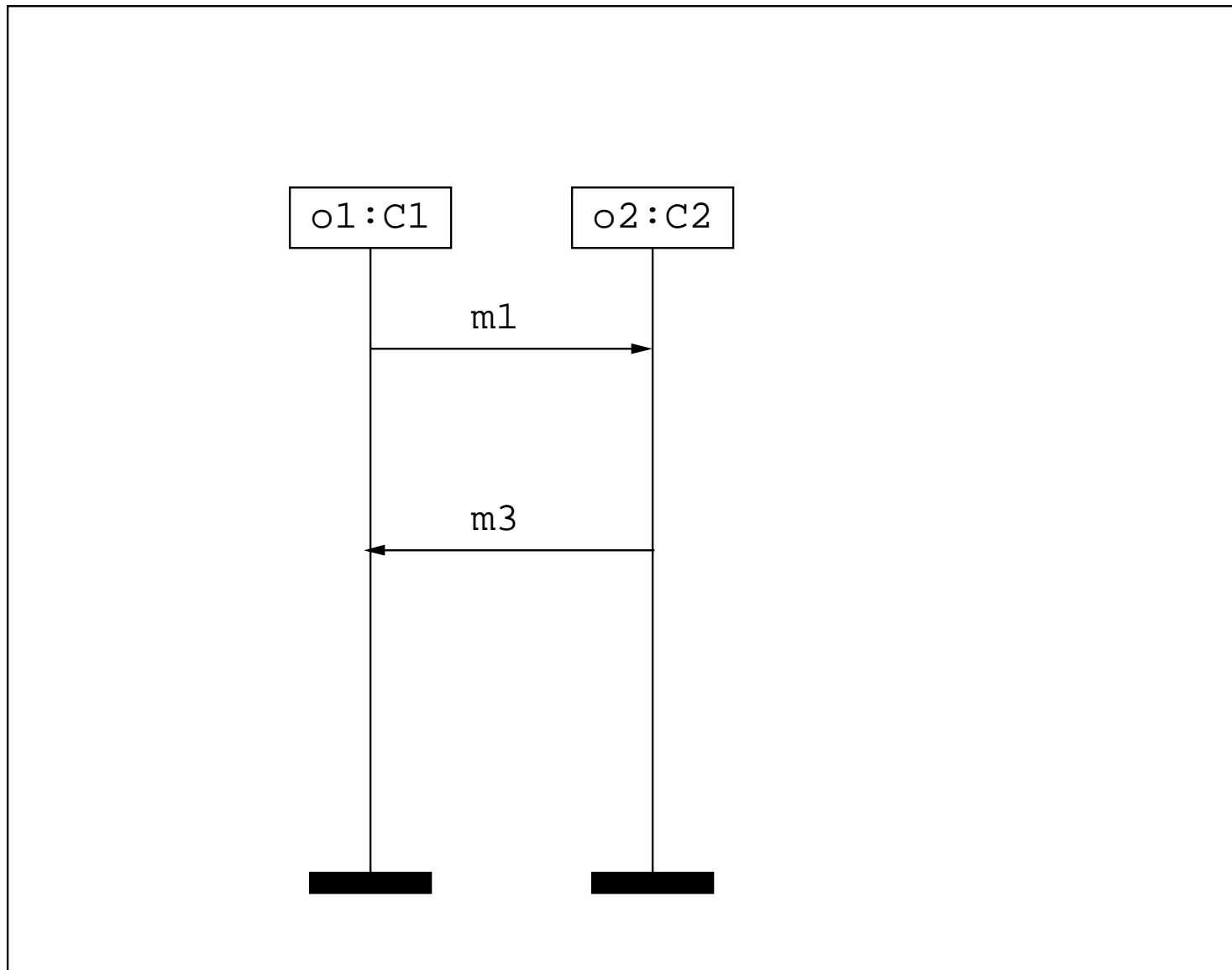
Repeat

Abstract Syntax

Denotational Semantics

Case Study

Conclusions



Introduction

Interactions with Variabilities

Features
Variation Points and Variants

Interactions with Variants

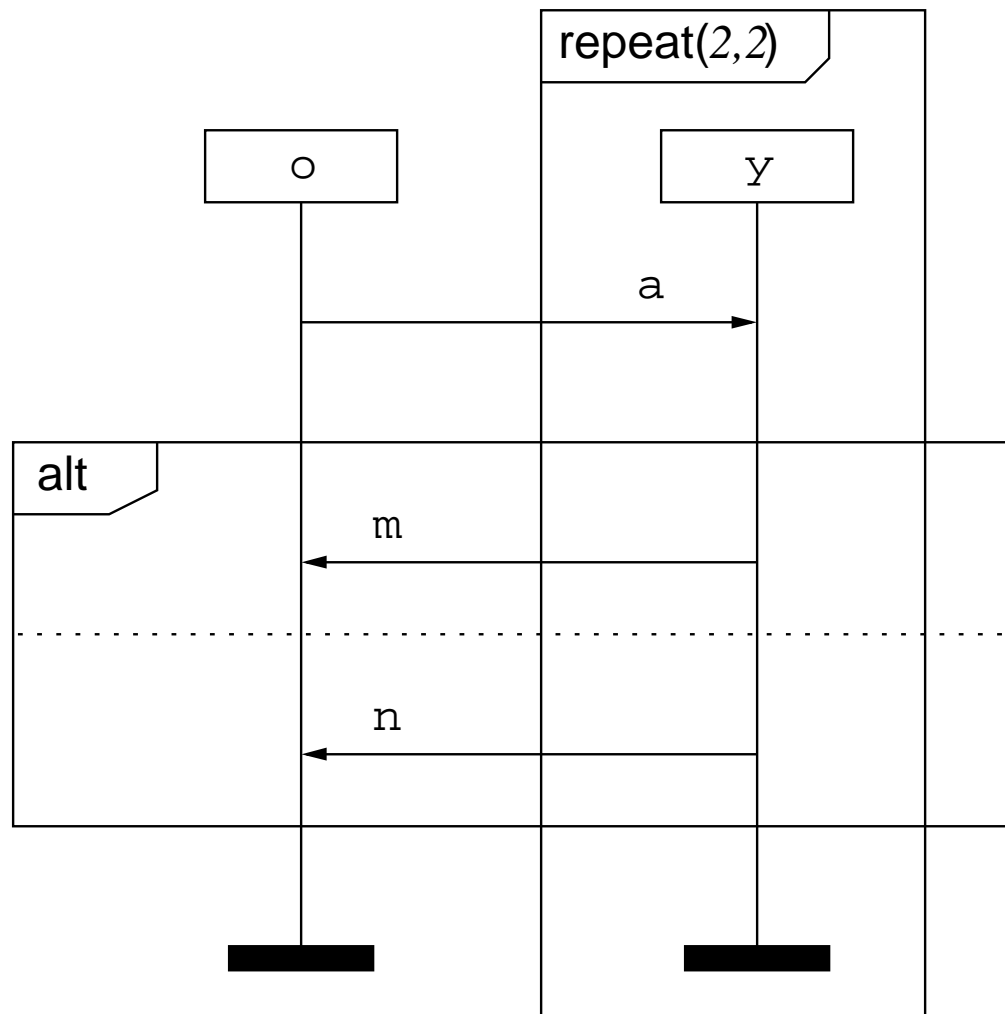
Repeat

Abstract Syntax

Denotational Semantics

Case Study

Conclusions



Introduction

Interactions with Variabilities

Features
Variation Points and Variants

Interactions with Variants

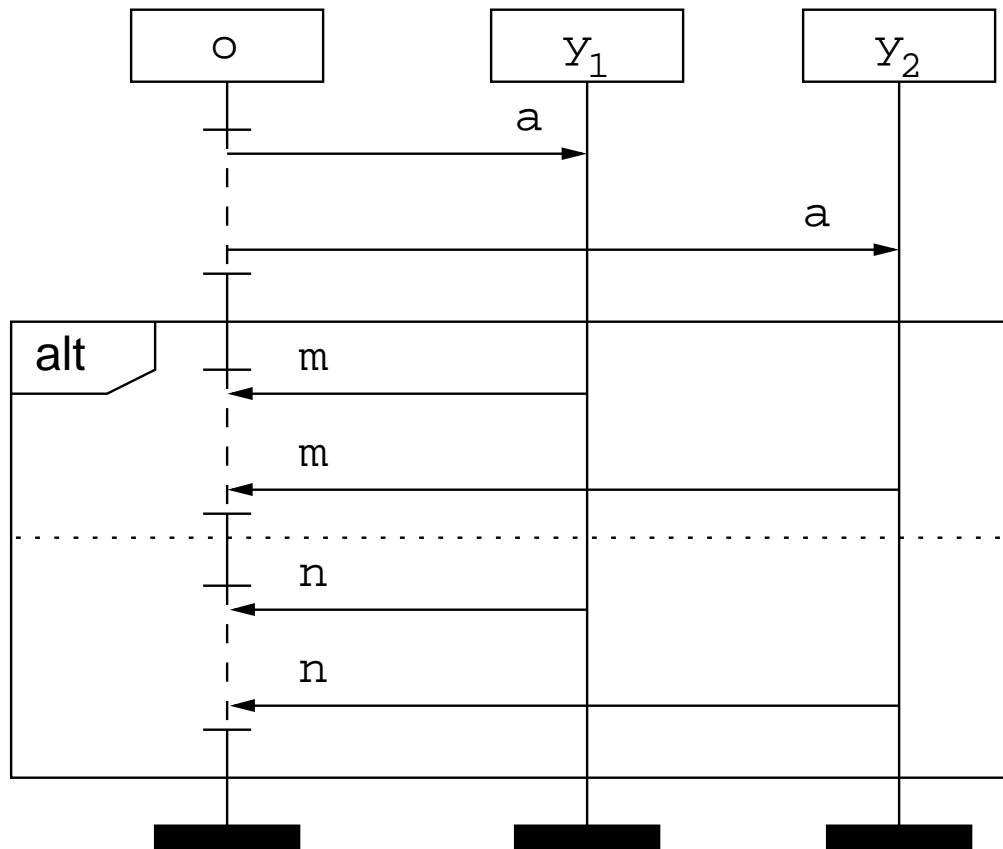
Repeat

Abstract Syntax

Denotational Semantics

Case Study

Conclusions



Introduction

Interactions with
Variabilities

Abstract Syntax

Use

Syntax Fragment

Denotational
Semantics

Case Study

Conclusions

Abstract Syntax

Introduction

Interactions with
Variabilities

Abstract Syntax

Use

Syntax Fragment

Denotational
Semantics

Case Study

Conclusions

- Graphical notations are often ambiguous
- UML interactions can also be interpreted in different ways
- We introduce an abstract syntax to describe the interactions
- Textual and unambiguous
- Basis are *send* and *receive* events

Introduction

Interactions with
Variabilities

Abstract Syntax

Use

Syntax Fragment

Denotational
Semantics

Case Study

Conclusions

$$\begin{array}{l}
 \textit{Interaction} ::= \textit{Event} \\
 \qquad \qquad \qquad | \textit{CombinedFragment} \\
 \textit{CombinedFragment} ::= \textit{sd}(\{\textit{Instances}\})(\textit{Interaction}) \\
 \qquad \qquad \qquad | \textit{seq}(\textit{Interaction}, \textit{Interaction}) \\
 \qquad \qquad \qquad | \textit{par}(\textit{Interaction}, \textit{Interaction}) \\
 \qquad \qquad \qquad | \textit{alt}(\textit{Interaction}, \textit{Interaction}) \\
 \qquad \qquad \qquad | \textit{repeat}(\{\textit{Instances}\})(\textit{Times}, (\textit{Times} \mid \infty), \\
 \qquad \qquad \qquad \textit{Interaction}) \\
 \qquad \qquad \qquad | \textit{variant}(\textit{BExp}, \textit{Interaction}) \\
 \textit{Instances} ::= \textit{Instance}, \textit{Instances} \\
 \qquad \qquad \qquad | \textit{Instance}
 \end{array}$$

...

Introduction

Interactions with
Variabilities

Abstract Syntax

**Denotational
Semantics**

Why a Formal
Semantics?

Main Concepts

Satisfaction Relations

Case Study

Conclusions

Denotational Semantics

Introduction

Interactions with
Variabilities

Abstract Syntax

Denotational
Semantics

**Why a Formal
Semantics?**

Main Concepts

Satisfaction Relations

Case Study

Conclusions

- Yet another UML semantics?
- UML 2.0 interactions are not handled by other semantics
- More precision and unambiguity in language extensions
- Important for effective tool support

Introduction

Interactions with
Variabilities

Abstract Syntax

Denotational
Semantics

Why a Formal
Semantics?

Main Concepts

Satisfaction Relations

Case Study

Conclusions

- Based on semantics of plain interactions from Cengarle and Knapp
- The semantics of a plain interaction S states whether a trace t is *positive* or *negative* for the interaction, written $t \models_p S$ and $t \models_n S$, respectively
- if t is neither positive nor negative for S , then t is called *inconclusive* for S .
- The semantics of an extended interaction S depends on a configuration \mathcal{C} .
- $t \models_p^{\mathcal{C}} S$ and $t \models_n^{\mathcal{C}} S$

$$\begin{array}{ll}
 t \models_p^{\mathcal{C}} \text{variant}(B, S) & \text{if either } \mathcal{C} \models B \text{ and } t \models_p^{\mathcal{C}} S \\
 & \text{or } \mathcal{C} \not\models B \text{ and } t \models_p^{\mathcal{C}} \text{skip} \\
 t \models_p^{\mathcal{C}} \text{repeat}(Y)(m, \bar{n}, S) & \text{if } \exists \sigma : \cup_{i=1}^n \{y_i : y \in Y\} \rightarrow \mathbb{I}. \sigma \\
 & \text{is injective and} \\
 & t \models_p^{\mathcal{C}} \text{par}_{i=1}^{\mathcal{C}(m, \bar{n})} (S[y \mapsto \sigma(y_i) : y \in Y]) \\
 t \models_p^{\mathcal{C}} \text{sd}(Y)(S) & \text{if } \exists \sigma : Y \rightarrow \mathbb{I}. \sigma \text{ is injective and} \\
 & t \models_p^{\mathcal{C}} S[y \mapsto \sigma(y) : y \in Y]
 \end{array}$$

- The negative satisfaction relations are written accordingly.
- The boolean operators are defined as usual.

Introduction

Interactions with
Variabilities

Abstract Syntax

Denotational
Semantics

Case Study

Overview

Interaction

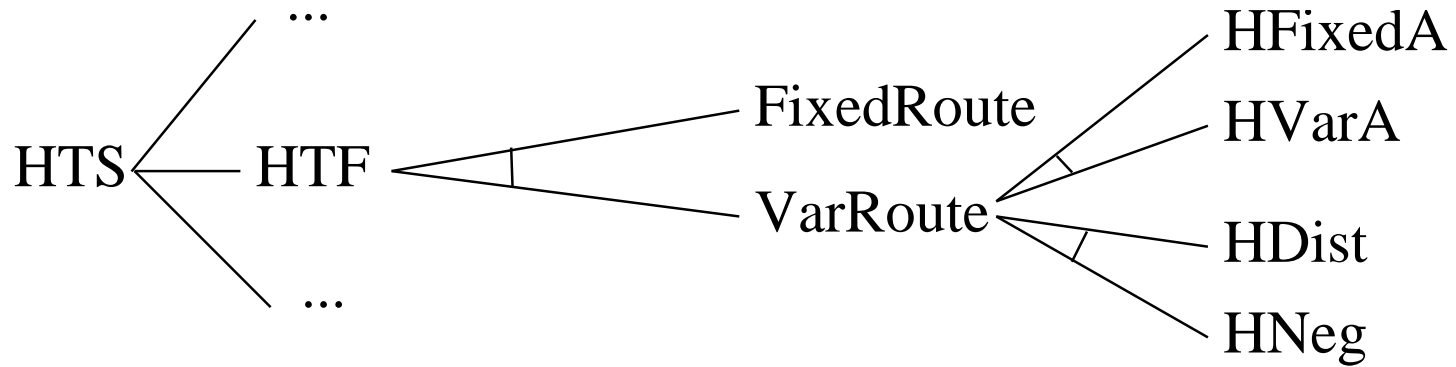
Textual Representation

Trace

Conclusions

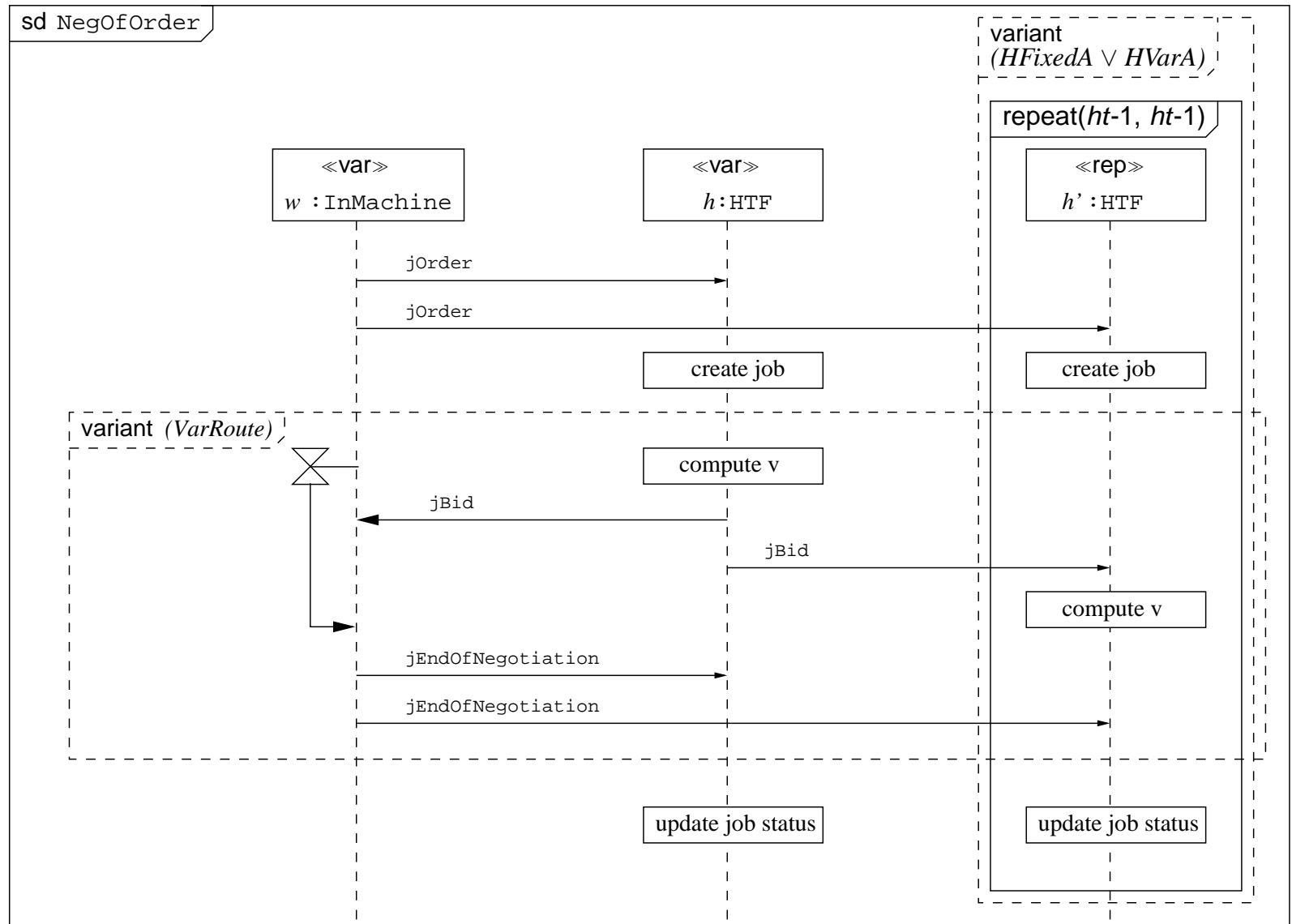
Case Study

- [Introduction](#)
- [Interactions with Variabilities](#)
- [Abstract Syntax](#)
- [Denotational Semantics](#)
- [Case Study](#)
- [Overview](#)
- [Interaction](#)
- [Textual Representation](#)
- [Trace](#)
- [Conclusions](#)



- Extensive case study of a holonic flow of material in a production system
- Autonomous vehicles (HTFs) transport engine parts between machine tools where they are processed
- Several variations concerning the process are possible

- Introduction
- Interactions with Variabilities
- Abstract Syntax
- Denotational Semantics
- Case Study
- Overview
- Interaction**
- Textual Representation
- Trace
- Conclusions



$$S_{NegOfOrder} = sd(\{w, h\}, seq(S_{ord}, S_1, S_2))$$

$$S_{ord} = seq(snd(w, h, jOrder), rcv(w, h, jOrder))$$

$$S_1 = \text{variant}(VarRoute \wedge \neg HFixedA \wedge \neg HVarA, seq($$

$$\quad snd(h, w, jBid), rcv(h, w, jBid),$$

$$\quad snd(w, h, jEndOfNegotiation),$$

$$\quad rcv(w, h, jEndOfNegotiation)))$$

$$S_2 = \text{variant}(HFixedA \vee HVarA, repeat(\{h'\}, ht - 1, ht - 1, seq($$

$$\quad snd(w, h', jOrder), rcv(w, h', jOrder), S'_1)))$$

Introduction

Interactions with
Variabilities

Abstract Syntax

Denotational
Semantics

Case Study

Overview

Interaction

Textual Representation

Trace

Conclusions

$$S'_1 = \text{variant}(\text{VarRoute}, \text{seq}(\text{snd}(h, w, \text{jBid}), \text{rcv}(h, w, \text{jBid}), \text{snd}(h, h', \text{jBid}), \text{rcv}(h, h', \text{jBid}), \text{snd}(w, h, \text{jEndOfNegotiation}), \text{rcv}(w, h, \text{jEndOfNegotiation}), \text{snd}(w, h', \text{jEndOfNegotiation}), \text{rcv}(w, h', \text{jEndOfNegotiation})))$$

Introduction

Interactions with Variabilities

Abstract Syntax

Denotational Semantics

Case Study

Overview

Interaction

Textual Representation

Trace

Conclusions

$$\begin{aligned}
 t_1 = & \text{snd}(w3 : \text{InMachine}, h1 : \text{HTF}, j\text{Order}) \cdot \\
 & \text{snd}(w3 : \text{InMachine}, h2 : \text{HTF}, j\text{Order}) \cdot \\
 & \text{snd}(w3 : \text{InMachine}, h3 : \text{HTF}, j\text{Order}) \cdot \\
 & \text{rcv}(w3 : \text{InMachine}, h1 : \text{HTF}, j\text{Order}) \cdot \\
 & \text{rcv}(w3 : \text{InMachine}, h3 : \text{HTF}, j\text{Order}) \cdot \\
 & \dots \\
 & \text{snd}(w3 : \text{InMachine}, h1 : \text{HTF}, j\text{EndOfNegotiation}) \cdot \\
 & \text{snd}(w3 : \text{InMachine}, h3 : \text{HTF}, j\text{EndOfNegotiation}) \cdot \\
 & \text{snd}(w3 : \text{InMachine}, h2 : \text{HTF}, j\text{EndOfNegotiation}) \cdot \\
 & \text{rcv}(w3 : \text{InMachine}, h3 : \text{HTF}, j\text{EndOfNegotiation}) \cdot \\
 & \text{rcv}(w3 : \text{InMachine}, h1 : \text{HTF}, j\text{EndOfNegotiation})
 \end{aligned}$$

Introduction

Interactions with Variabilities

Abstract Syntax

Denotational Semantics

Case Study

Overview

Interaction

Textual Representation

Trace

Conclusions

$$\begin{aligned}
 t_1 = & \text{snd}(w3 : \text{InMachine}, h1 : \text{HTF}, j\text{Order}) \cdot \\
 & \text{snd}(w3 : \text{InMachine}, h2 : \text{HTF}, j\text{Order}) \cdot \\
 & \text{snd}(w3 : \text{InMachine}, h3 : \text{HTF}, j\text{Order}) \cdot \\
 & \text{rcv}(w3 : \text{InMachine}, h1 : \text{HTF}, j\text{Order}) \cdot \\
 & \text{rcv}(w3 : \text{InMachine}, h3 : \text{HTF}, j\text{Order}) \cdot \\
 & \dots \\
 & \text{snd}(w3 : \text{InMachine}, h1 : \text{HTF}, j\text{EndOfNegotiation}) \cdot \\
 & \text{snd}(w3 : \text{InMachine}, h3 : \text{HTF}, j\text{EndOfNegotiation}) \cdot \\
 & \text{snd}(w3 : \text{InMachine}, h2 : \text{HTF}, j\text{EndOfNegotiation}) \cdot \\
 & \text{rcv}(w3 : \text{InMachine}, h3 : \text{HTF}, j\text{EndOfNegotiation}) \cdot \\
 & \text{rcv}(w3 : \text{InMachine}, h1 : \text{HTF}, j\text{EndOfNegotiation})
 \end{aligned}$$

$t_1 \models_n^c S_{\text{NegOfOrder}}$, since in t_1 the receipt of message $j\text{EndOfNegotiation}$ from the `InMachine` to the first repeated instance `h2` is missing.

Introduction

Interactions with
Variabilities

Abstract Syntax

Denotational
Semantics

Case Study

Conclusions

Summary

Future Work

Conclusions

Introduction

Interactions with
Variabilities

Abstract Syntax

Denotational
Semantics

Case Study

Conclusions

Summary

Future Work

- Extensions to the UML interactions to represent variability
- Operator variant specifies an optional interaction that depends on a configuration
- Operator repeat for copying analogous lifelines with their message exchange patterns
- The number of lifelines depends then on the configuration
- Binding operator sd for the declaration of instance variables
- Sufficient means for the purposes of our case study
- There may be other variations that we did not consider

Introduction

Interactions with
Variabilities

Abstract Syntax

Denotational
Semantics

Case Study

Conclusions

Summary

Future Work

- Dynamic reconfiguration
- Incorporation of MSC connectors
- Interplay with other UML diagrams