Interactive Model Derivation with External Constraints

Mikoláš Janota
University College Dublin, Ireland

Victoria Kuzina
Andrzej Wąsowski
IT University of Copenhagen, Denmark
Interactive Model Derivation

Possible Operations

Analysis

Perform

Edit Operation

Choice

Possible Operations

Designer
Interactive Model Derivation

Possible Operations

Analysis

Edit Operation

Perform

Choice

Model

Possible Operations

Designer
Interactive Model Derivation

Possible Operations

Analysis

Model

Perform

Edit Operation

Choice

Possible Operations

Designer
Interactive Model Derivation

Possible Operations

Analysis

Edit Operation

Perform

Choice

Designer

Possible Operations
Soundness-preserving derivation seen in instance derivation

Completeness-preserving derivation will be illustrated by a prototype for feature diagrams

Semantics-preserving derivation will be illustrated by a prototype for feature models
Soundness-preserving derivation seen in instance derivation

Completeness-preserving derivation will be illustrated by a prototype for feature diagrams

Semantics-preserving derivation will be illustrated by a prototype for feature models
Classification of Derivation

- Soundness-preserving derivation seen in instance derivation
- Completeness-preserving derivation will be illustrated by a prototype for feature diagrams
- Semantics-preserving derivation will be illustrated by a prototype for feature models
model $\rightarrow$ an instance
Soundness-Preserving Derivation

\[ \varphi \vdash [M_0] \]

model → an instance
Soundness-Preserving Derivation

$\phi \llbracket M_0 \rrbracket \llbracket M_1 \rrbracket$

model $\rightarrow$ an instance
Soundness-Preserving Derivation

\[ \langle M_0 \rangle \phi \rightarrow \langle M_1 \rangle \rightarrow \ldots \rightarrow \langle M_n \rangle \]

model \rightarrow an instance
C1  Each USB must contain exactly one instance of PC.

C2  Every device is connected to a port or to the PC instance.

C3  Every USB has a keyboard connected or a free port to connect one.
1. PC

2. PC -- hub-1

3. PC -- hub-1

4. PC -- hub-1 -- printer-1

device -- hub-2

socket

device -- hub-2

socket
Deriving a USB PC hub-1 device hub-2 hub-1 PC socket device printer-1 device hub-2 PC socket socket printer-2

1. 

2. 

3. 

4. 

5. 

PC

hub-1

PC

hub-2

device

PC

hub-1

PC

hub-1

PC

hub-1

device

printer-1

socket

device

printer-2

socket

device

socket

PC

hub-1
Validity of advice: no sequence of operations leads to an invalid model

For the USB language: all derivable USBs satisfy the language constraints (the diagram and C1–C3)

Exhaustiveness of advice: all conforming instances are derivable.

For the USB language: all legal USBs can be derived (van der Meer 2006)
**Validity of advice:** no sequence of operations leads to an invalid model

For the USB language: all derivable USBs satisfy the language constraints (the diagram and C1–C3)

**Exhaustiveness of advice:** all conforming instances are derivable.

For the USB language: all legal USBs can be derived (van der Meer 2006)
Validity of advice: no sequence of operations leads to an invalid model

For the USB language: all derivable USBs satisfy the language constraints (the diagram and C1–C3)

Exhaustiveness of advice: all conforming instances are derivable.

For the USB language: all legal USBs can be derived (van der Meer 2006)
Validity of advice: no sequence of operations leads to an invalid model

For the USB language: all derivable USBs satisfy the language constraints (the diagram and C1–C3)

Exhaustiveness of advice: all conforming instances are derivable.

For the USB language: all legal USBs can be derived (van der Meer 2006)
Soundness-Preserving Derivation

\[ \llbracket \text{M} \rrbracket \phi / \llbracket \text{M}_0 \rrbracket / \llbracket \text{M}_1 \rrbracket / \ldots / \llbracket \text{M}_n \rrbracket \]

model \(\rightarrow\) an instance
$[\mathcal{L}_0]$
Completeness-Preserving Derivation

\[ [\mathcal{L}_0] \bigg| [\mathcal{L}_1] \bigg| \phi \]
Completeness-Preserving Derivation

$[\mathcal{L}_0] [\mathcal{L}_1] \cdots [\mathcal{L}_n] \phi$
Feature Model Example

Root Feature
Mandatory
Sub-feature
Optional
Sub-feature
or-group
xor-group

additional constraint:

electric → automatic
additional constraint:

electric → automatic
Feature Model approximating a constraint
Feature Model approximating a constraint
Feature Model approximating a constraint
Feature Model approximating a constraint
The algorithm has the properties of validity and exhaustiveness of advice.

The algorithm is efficient compared to approaches based on Constraint Satisfaction or Logic Programming.
Completeness-Preserving Derivation

\[ [\mathcal{M}_0] \subseteq [\mathcal{M}_1] \subseteq \ldots \subseteq [\mathcal{M}_n] \]

\[ [\mathcal{L}_0] \subseteq [\mathcal{L}_1] \subseteq \ldots \subseteq [\mathcal{L}_n] \]
Completeness-Preserving Derivation

\[ \phi \llbracket \mathcal{M}_0 \rrbracket \llbracket \mathcal{M}_1 \rrbracket \cdots \llbracket \mathcal{M}_n \rrbracket \]

\[ \llbracket \mathcal{L}_0 \rrbracket \llbracket \mathcal{L}_1 \rrbracket \cdots \llbracket \mathcal{L}_n \rrbracket \phi \]
Completeness-Preserving Derivation

\[ \phi \llbracket M_0 \rrbracket \llbracket M_1 \rrbracket \ldots \llbracket M_n \rrbracket \]

\[ \llbracket M_0 \rrbracket \llbracket M_1 \rrbracket \llbracket M_n \rrbracket \phi \]

model \rightarrow \text{an instance}

\[ [\mathcal{L}_0] \ [\mathcal{L}_1] \ [\mathcal{L}_n] \]

\[ \phi \]

meta-model \rightarrow \text{model}
semantics-preserving =
completeness-preserving + soundness-preserving
semantics-preserving =
completeness-preserving
+ soundness-preserving

Example: any refactoring
semantics-preserving = completeness-preserving + soundness-preserving

Example: any refactoring
Example: feature model derivation
the model comprises two components
feature diagram \((M)\)
additional constraint \((\psi)\)
overall semantics must be preserved
"\(M + \psi = M' + \psi'\)"

or

\[
(\mathcal{M}, \psi) \xrightarrow{\phi} (\mathcal{M}', \psi')
\]
the model comprises two components
feature diagram ($\mathcal{M}$)
additional constraint ($\psi$)
overall semantics must be preserved
"$\mathcal{M} + \psi = \mathcal{M}' + \psi'$"

or

$$(\mathcal{M}, \psi) \xrightarrow{\cdot} (\mathcal{M}', \psi')$$

$$(\mathcal{M}', \psi') \xleftarrow{\cdot} (\mathcal{M}, \psi)$$
Practically Speaking

Java - Eclipse SDK

Feature Model Derivation

1--Select Example -->>

14--Feature Diagram

0--Car -->>

Feature Graph to Clauses:

- Gear --> Manual | Automatic
- Automatic --> Gear
- Electric --> Engine
- Manual --> Gear
- Electric --> Automatic
- !Manual | !Automatic
- Gas | Electric
- Keyless --> PowerLocks

Left-over constraints:

- Body
- Engine
- Gear
- Manual | Automatic
- Electric --> Automatic
- !Manual | !Automatic
- Gas | Electric
- Keyless --> PowerLocks
Practically Speaking

Feature Model Derivation

1 -- Select Example

14 -- Feature Diagram

0 -- Car

3 -- Gear

11 --

7 -- Automatic
6 -- Manual

8 -- Keyless
9 -- PowerLocks

1 -- Body

Left-over constraints:

Electric => Automatic
Keyless => PowerLocks

Feature Graph to Clauses:

Gear => Manual | Automatic
Automatic => Gear
Electric => Engine
Manual => Gear
Keyless => PowerLocks
Body => Car
Car => Body
PowerLocks => Car
Gear => Car
Car => Gear
- glossary
  soundness-preserving derivation
  completeness-preserving derivation
  semantics-preserving derivation
  validity of advice
  exhaustiveness of advice

- Work this out for a rich subset of ECORE models, not only for Feature Models