Partial Models: A Position Paper

Michalis Famelis, Shoham Ben-David, Marsha Chechik and Rick Salay

University of Toronto

October 17, 2011
Motivating Example

Bob and Alice are building a network controller:

Bob's behavioral model

Alice's architectural model

<table>
<thead>
<tr>
<th>Controller</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ disconnected : boolean</td>
</tr>
<tr>
<td>+ on() : void</td>
</tr>
<tr>
<td>+ off() : void</td>
</tr>
<tr>
<td>+ beep() : void</td>
</tr>
</tbody>
</table>
Motivating Example

Bob and Alice are building a network controller:

Constraint (C1): "No sink states."
Bob’s Alternative Fixes

1. **Recover back to On:**
   - Recover back to On.
   - Control: on()
   - Connected: [disconnected == False]

2. **Log an error and turn Off:**
   - Log an error and turn Off.
   - Control: off()
   - Connected: [disconnected == False]

3. **Get rid of Warning:**
   - Get rid of Warning.
   - Control: on()
   - Connected: [disconnected == False]
Uncertainty: Which Alternative?

Bob has a problem:

- Requirements are unclear about recovery.
- Any changes to the architectural model must be approved by Alice.

What are his options?

- Stop and wait for more information.
- Make an (informed) guess, risk backtracking.
- Work with the entire set of alternatives.
- Use Partial Models! :)

Note: Inconsistency fixing is merely an example.

There could be other sources of uncertainty!
What Is A Partial Model?

\[ \Phi_p = (\neg A \land B \land C \land D \land \neg E \land \neg F \land G) \lor \\
(\neg A \land B \land \neg C \land D \land E \land F \land \neg G) \lor \\
(A \land \neg B \land \neg C \land \neg D \land \neg E \land \neg F \land \neg G) \]
What Is A Partial Model?

Optional elements

\[ \Phi_p = (\neg A \land B \land C \land D \land \neg E \land \neg F \land G) \lor \\
(\neg A \land B \land \neg C \land D \land E \land F \land \neg G) \lor \\
(A \land \neg B \land \neg C \land \neg D \land \neg E \land \neg F \land \neg G) \]
What Is A Partial Model?

Allowable configurations

Control

[disconnected==True] beep()

Controller

+disconnected : boolean
+on() : void
+off() : void
+beep() : void
+error() : void
+recover() : void

Φp = (¬A \land B \land C \land D \land ¬E \land ¬F \land G) \lor
(¬A \land B \land ¬C \land D \land E \land F \land ¬G) \lor
(A \land ¬B \land ¬C \land ¬D \land ¬E \land ¬F \land ¬G)
What Is A Partial Model?

Concretization

\[ \Phi_p = (-A \land B \land C \land D \land \neg E \land \neg F \land G) \lor (-A \land B \land \neg C \land D \land E \land F \land \neg G) \lor (A \land \neg B \land \neg C \land \neg D \land \neg E \land \neg F \land \neg G) \]

Motivating Example

Working with Partial Models

Reasoning

Transformation

Conclusion
Facilitate decision deferral in the presence of uncertainty by using Partial Models, that represent sets of alternatives, as first-class development artifacts.

What do we mean by “first-class development artifact”?  
- Checking of properties.  
- Transformation and refinement.
Comments On Partial Models

Other important characteristics:

- Compact and exact representation of a set.
- Metamodel/language independence.

Status:

- Different kinds of partiality, submitted [SCF11].
  - May, Abs, Var, OW
- Preliminary implementation with Alloy/KodKod.
1 Motivating Example

2 Working with Partial Models
   Reasoning
   Transformation

3 Conclusion
Checking Properties

Check C1 ("no sink states") on the partial model $M_1$:

\[ \Phi_p = (\neg A \land B \land C \land D \land \neg E \land \neg F \land \neg G) \lor (\neg A \land B \land \neg C \land D \land E \land F \land \neg G) \lor (A \land \neg B \land \neg C \land \neg D \land \neg E \land \neg F \land \neg G) \]

It holds for all concretizations. Result: **True**.
Checking Properties

Check C2 ("no transitions with identical source and target"): 

\[
\Phi_p = (\neg A \land B \land C \land D \land \neg E \land \neg F \land G) \lor \\
(\neg A \land B \land \neg C \land D \land E \land F \land \neg G) \lor \\
(A \land \neg B \land \neg C \land \neg D \land \neg E \land \neg F \land \neg G)
\]

C2 holds for some concretizations.
Checking Properties

But C2 does not hold for others:

\[ \Phi_p = (\neg A \land B \land C \land D \land \neg E \land \neg F \land G) \lor 
(\neg A \land B \land \neg C \land D \land E \land F \land \neg G) \lor 
(A \land \neg B \land \neg C \land \neg D \land \neg E \land \neg F \land \neg G) \]

The result is therefore **Maybe**.
How Is Checking Done?

• High level algorithm:
  
  1. Express the *entire* partial model as a formula $\Phi_{M1}$:
     
     $\Phi_{M1} = \Phi_P \land \text{Control} \land \text{Off} \ldots \land \text{Controller} \land \text{on()} \land \ldots$

  2. Express the property as a propositional formula $\Phi_{C2}$.

  3. Check $\Phi_{M1} \land \Phi_{C2}$ and $\Phi_{M1} \land \neg \Phi_{C2}$ for SAT.

• If SAT, we also get counterexamples for feedback.

• We can reason about *all* concretizations together, with two queries to the SAT solver.

• Study of feasibility and scalability, submitted [FSC11].
1 Motivating Example

2 Working with Partial Models
   Reasoning
   Transformation

3 Conclusion
Two Kinds Of Transformations

1. Classical transformations adapted to work for Partial Models.
   - “Detail-adding” (DA) refinements, refactoring
   - Allowing development to continue, even in the presence of uncertainty.

2. Transformations specific to Partial Models.
   - “Uncertainty-removing” (UR) refinements.
   - A systematic way to incorporate new information.
Using Adapted Transformations

Bob elaborates *Warning* by DA refinement.

\[ \Phi p' = ( (\neg A \land B \land C \land D \land \neg E \land \neg F \land G) \lor 
(\neg A \land B \land \neg C \land D \land E \land F \land \neg G) \lor 
(A \land \neg B \land \neg C \land \neg D \land \neg E \land \neg F \land \neg G) 
) \land D \Rightarrow (H \land J) \]
Why “Adapt” Transformations?

Not all concretizations have a Warning state!

\[
\Phi_p' = \left( \neg A \land B \land C \land D \land \neg E \land \neg F \land G \right) \lor \\
\left( \neg A \land B \land \neg C \land D \land E \land F \land \neg G \right) \lor \\
\left( A \land \neg B \land \neg C \land \neg D \land \neg E \land \neg F \land \neg G \right) \\
\land D \implies (H \land J)
\]
Some “Just Work”

Bob can do some kinds of refactoring straightforwardly:

![Diagram of a control system with states Off and On, actions off(), on(), recover(), and methods beep() and isDisconnected().]
Comments On Adapted Transformations

• Detail-adding refinements should result in models with “more information”, same level of uncertainty.

• Refactoring should not add or remove information and/or uncertainty.

• Adapted versions of classical transformations must be total and surjective.

• Such transformations preserve True existential and False universal properties.
Removing Uncertainty

Once Bob and Alice have negotiated, they can return to classical models:

“Uncertainty-removing” refinements:
Transformations specific to Partial Models.
Comments On UR Refinements

- For May partiality, optional elements can be kept optional, removed or made mandatory.
- Completely remove uncertainty: *Concretization*
- UR refinement: a generic refinement mechanism, with well understood properties [SCF11]:
  - True (False) properties remain True (False).
  - Maybe properties can be changed into True or False or remain unaffected.
Summary

• Goal: Facilitate decision deferral in the presence of uncertainty.

• Approach: Use Partial Models to represent sets of alternatives.

• How: Partial Models are first-class development artifacts.
  - Property checking.
  - Adapted transformations.
  - Partial Model-specific transformations.
Conclusion

• Important contributions:
  • Decision deferral in the presence of uncertainty.
  • Compact and exact representation of a set.
  • Metamodel independence.

• In the paper: 14 specific Research Questions.

• Preliminary work on Representation, Property Checking.

• Some prototype tooling.

• Main focus now: Transformations.
Bibliography


Available at http://www.cs.toronto.edu/~famelis
Questions?