

# **A Specification-based Test Case Generation Method for UML/OCL**

Achim D. Brucker, Matthias P. Krieger,  
Delphine Longuet, and Burkhart Wolff

Université Paris-Sud

# Motivation: Why Automated test data generation ?

- Model-based Testing fits to the paradigm of, well, Modeling !
- (Approximative) Alternative to Code-**Verification**, particularly for a mix of (dirty) prog. languages.
- Can be used for the **Validation** of Models against Legacy problems ...
- Suffers from State-Explosion Problems like Model-Checking or Automated Theorem Proving

# Outline

- Recap UML/OCL Semantics
- Example
- Compiling UML/OCL to Logics
- DNF Computation
- Alias-Closure
- Test-Data-Selection
- Perspective: Black-box-Testing in HOL-TestGen

# Recap UML/OCL Semantics

- Semantic Interpretation and Validity:

$$(\sigma, \sigma') \models \phi \quad \equiv \quad (I \llbracket \phi \rrbracket (\sigma, \sigma') = \perp \text{true} \perp)$$

where the interpretation function  $I$  is

built over all built-in operators of OCL  
(+, ->includes, Set{a}, ...)

Recall: operations usually strict, i.e.

$$f(x_1, \dots, \perp, \dots, x_n) \equiv \perp$$

(where  $\perp$  is a notation for “invalid”)

# Recap UML/OCL Semantics (ctd I)

- Semantics Invariants:

$$\begin{aligned} \llbracket \text{context } c : C \text{ inv } n : \phi(c) \rrbracket (\sigma, \sigma') = \\ (\sigma, \sigma') \models (C \text{ .allInstances() } \rightarrow \text{forall}(x | \phi(x))) \wedge \\ (\sigma, \sigma') \models (C \text{ .allInstances() } \rightarrow \text{forall}(x | \phi(x)))_{\text{pre}} \end{aligned}$$

Consequence: in

context C inv m: a > 0

a must be defined, i.e. not equal  $\perp$  !!!

# Recap UML/OCL Semantics (ctd II)

- Semantics of Operation Contracts:

$\llbracket \text{context } C :: \text{op}(a_1, \dots, a_n) : T$   
 $\text{pre } \phi(\text{self}, a_1, \dots, a_n)$

$\text{post } \psi(\text{self}, a_1, \dots, a_n, \text{result}) \rrbracket (\sigma, \sigma') \equiv$   
 $\forall s, x_1, \dots, x_n.$

$\Delta(s, x_1, \dots, x_n) \wedge (\sigma, \sigma') \models \phi(\text{self}, x_1, \dots, x_n)$

$\rightarrow (\sigma, \sigma') \models \psi(s, x_1, \dots, x_n, s.\text{op}(x_1, \dots, x_n)) \wedge$

$\neg \Delta(s, x_1, \dots, x_n) \rightarrow s.\text{op}(x_1, \dots, x_n) \equiv \perp$

# Recap UML/OCL Semantics (ctd III)

- Global theory context  $\Gamma$  consists of
  - compiled invariants of the spec
  - compiled contracts of the spec
- Verification Question :

$$\Gamma \vdash (\sigma, \sigma') \models \phi$$

is  $\phi$  derivable via a conventional Gentzen Calculus from invariants and contracts ?

# Recap UML/OCL Semantics (ctd III)

- Validation Question :

$$\Gamma \vdash (\sigma, \sigma') \models \phi$$

can we derive from a *test-specification* automatically a number of ground - formulas that we run against the real system ?

# Recap UML/OCL Semantics (ctd III)

- Validation Question :

$$\Gamma \vdash (\sigma, \sigma') \models \phi \rightarrow TC_1 \vee \dots \vee TC_n$$

can we derive from a *test-specification* automatically a number of ground - formulas that we run against the real system ?

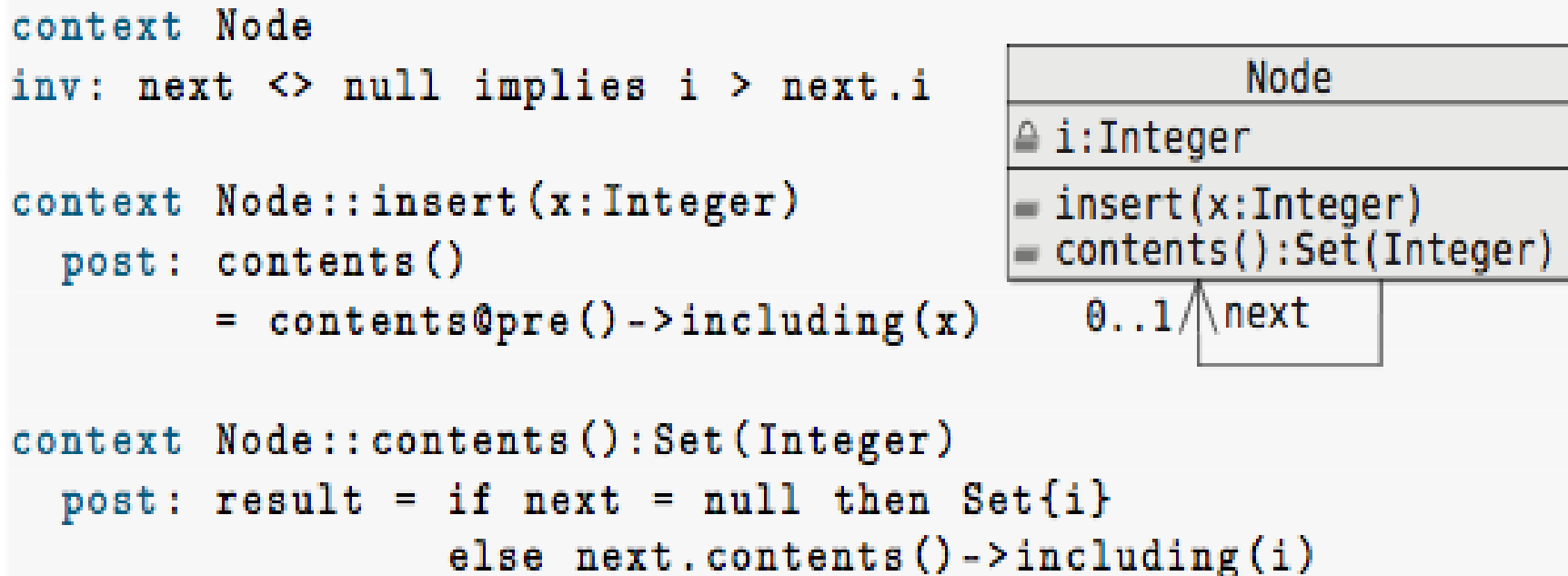
# Recap UML/OCL Semantics (ctd III)

- Validation Question :

$$\Gamma \vdash (\sigma, \sigma') \models \phi \rightarrow \text{TC}_1 \vee \dots \vee \text{TC}_n$$

- **Yes**
  - by DNF construction and simplification wrt. OCL Semantics and  $\Gamma$
  - Constraint Solving for the Test-Cases  $\text{TC}_i$

# Example



**Fig. 1.** A Singly-linked list specified in OCL (excerpt).

# Compiling UML/OCL to Logics

- Turning OCL-Logic into Standard Logic
- Converting Invariants into Recursive Predicates
- Converting Contracts into Pre-Condition-Free (recursive) Formulas
- Making all implicit Definedness Explicit

# Compiling UML/OCL to Logics

Turning OCL-Logic into Standard Logic:

From:

inv: (next=null or next<>null) and i<>null

inv: next<>null implies i>next.i

We can infer definedness (isvalid,  $\partial$ ) of self.next

and self.i :  $(\sigma, \sigma') \models \partial$  self.next ...

From (e.g.!)

$\tau \models \phi$  and  $\psi$  we can thus infer:  $\tau \models \phi \wedge \tau \models \psi$

# Compiling UML/OCL to Logics

- Converting Invariants into Recursive Predicates:

```
∀ self. ⊢ ∂ self ∧ ⊢ self ≠ null → ⊢ invNode(self)
⇔ ⊢ self.next = null ∨ (⊢ self.next ≠ null
    ∧ ⊢ self.i > self.next.i ∧ ⊢ invNode(self.next))
```

where  $\models \phi$  is an abbrev for  $\tau \models \phi$  or  $(\sigma, \sigma') \models \phi$

# Compiling UML/OCL to Logics

- Converting Contracts into Pre-Condition-Free (recursive) Formulas

```
∀ self. Δ(self) → ⊢ self.contents() ≐  
    if self.next ≐ null then Set{i}  
    else self.next.contents()->including(i)  
∧ ¬Δ(self) → ⊢ self.contents() ≐ ⊥
```

# DNF Computation

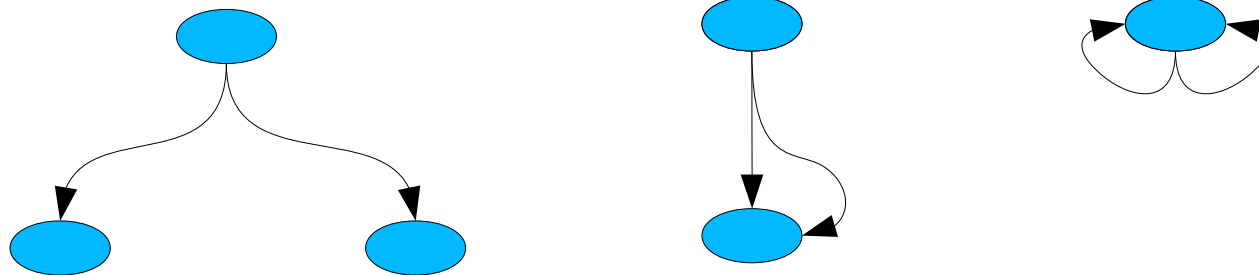
```
 $\Delta(s, x) \wedge \models \text{inv}_{\text{Node}@pre}(s) \wedge \models \text{inv}_{\text{Node}}(s)$   
 $\wedge \models s.\text{contents}() \doteq s.\text{contents}@pre() \rightarrow \text{including}(x)$ 
```

- Unfold inv
- recompute DNF
- goto 1 ...

Stop if nothing changes or paths  
(and therefore size of models exceeds  
a certain bound)

# Alias-Closure

- OO-Specs Talk on Graphs, not Trees  
(not “free algebras”, but “free co-algebras”)



$$\{ p \stackrel{\Delta}{=} q \vee p \stackrel{\nabla}{\neq} q \mid p, q \in \text{Path}(\varphi) \wedge p \text{ non-identical to } q \}$$

# Test-Data-Selection

- Test Case:

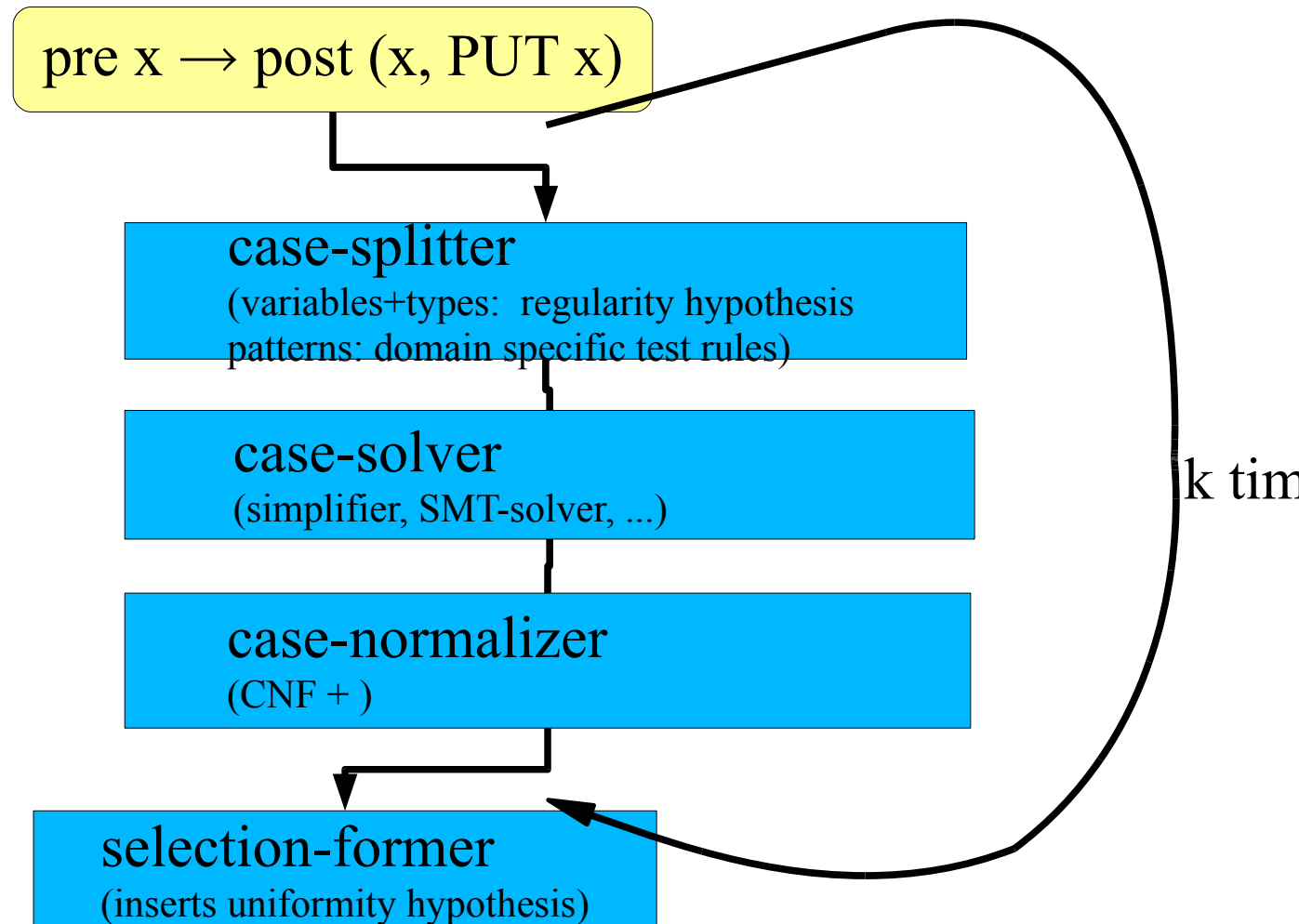
```
 $\Delta(s, x)$   
 $\wedge \models s.next \neq null \wedge \models s.i > s.next.i \wedge \models s.next.next \doteq null$   
 $\wedge \models s.next@pre \doteq null$   
 $\wedge \models Set\{s.next.i\} \rightarrow including(s.i) \doteq Set\{s.i@pre\}$   
 $\rightarrow including(x)$ 
```

- Test - Data - Selection  
(By using a constraint solver ...)

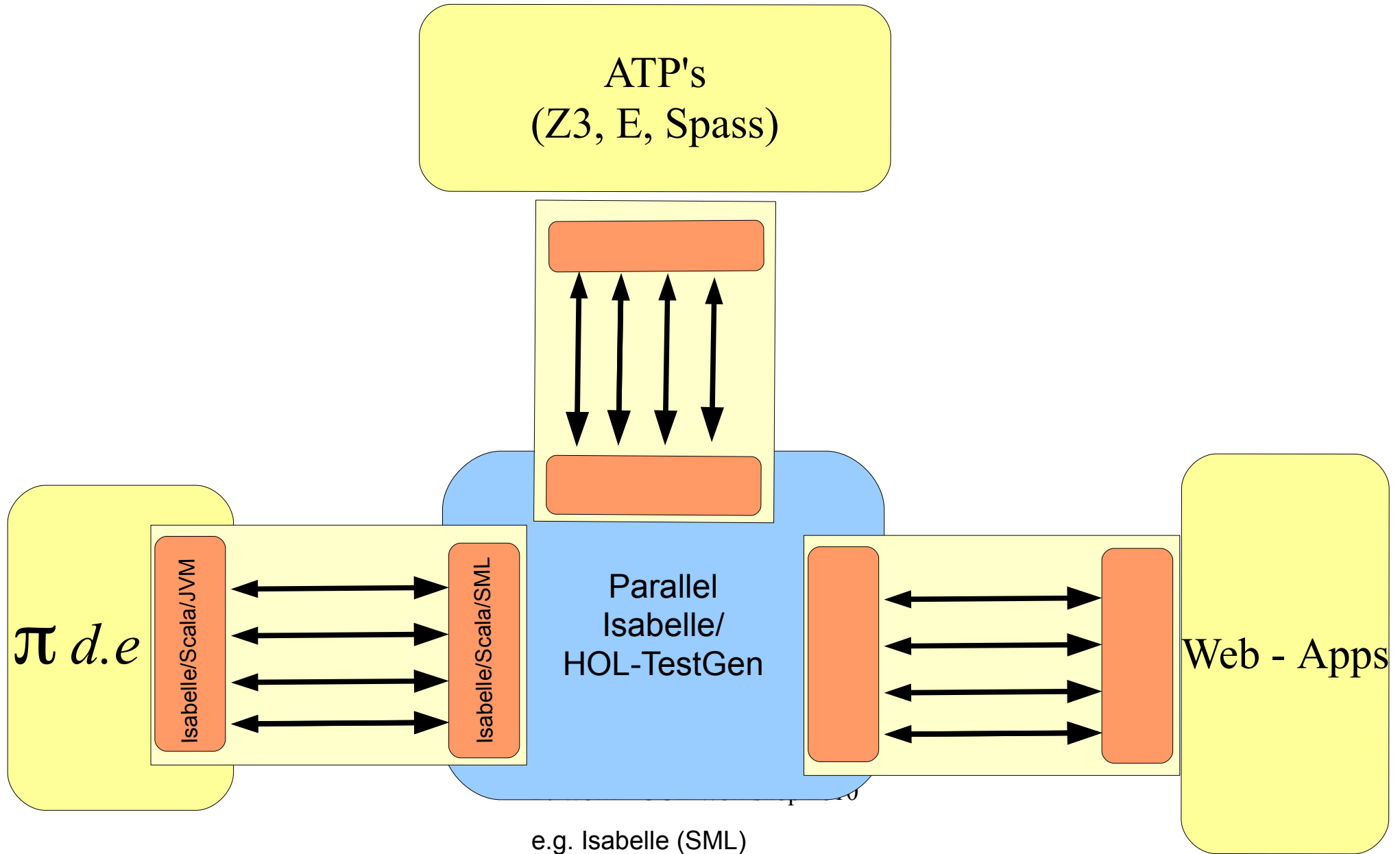


# Perspective: Black-box-Testing in HOL-TestGen

- The HOL-TestGen Derivation Process



# Architecture in the Future



# Conclusions

- Conceptual Study how to Apply DNF Black-box Test-Generation-Method to UML/OCL
- Recursion (on data and operations) taken into account
- Object-oriented Phenomena (Aliases) taken into account
- Done With Hindsight to an Implementation in HOL-TestGen