# Towards Self-Adaptable Languages

<u>Gwendal Jouneaux</u><sup>1</sup> Olivier Barais<sup>1</sup> Benoit Combemale<sup>1</sup> Gunter Mussbacher<sup>2</sup>

<sup>1</sup>Univ. Rennes, Inria, IRISA – Rennes, France

<sup>2</sup>McGill University – Montreal, Canada



ALE Seminar — May 24, 2022



Towards Self-Adaptable Languages - Jouneaux et al.

Introduction •			
-			

#### Context

Software ...



Introduction	

**SEALS** 000000 becifying Adaptive Semantics

Conclusion

# Context

Software ...

Evolve in complex/changing environment (e.g, Cloud, embedded systems)



Self-Adaptable Virtual Machine

SEALS 000000 ecifying Adaptive Semantics

Conclusior

# Context

Software ...

- Evolve in complex/changing environment (e.g, Cloud, embedded systems)
- Need dynamic adaptation to best deliver the service (e.g., Waymo<sup>1</sup>, Netflix<sup>1</sup>)

1 Cf. https://waymo.com, https://www.netflix.com

Self-Adaptable Virtual Machine

SEALS 000000 ecifying Adaptive Semantics

Conclusior

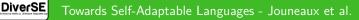
# Context

Software ...

- Evolve in complex/changing environment (e.g, Cloud, embedded systems)
- Need dynamic adaptation to best deliver the service (e.g., Waymo<sup>1</sup>, Netflix<sup>1</sup>)

Software languages ...

1 Cf. https://waymo.com, https://www.netflix.com



SEALS 000000

### Context

Software ...

- Evolve in complex/changing environment (e.g, Cloud, embedded systems)
- Need dynamic adaptation to best deliver the service (e.g., Waymo<sup>1</sup>, Netflix<sup>1</sup>)

#### Software languages ...

Can abstract concerns into high level constructs (e.g., memory management)

1 Cf. https://waymo.com, https://www.netflix.com

# Context

Software ...

DiverSE

- Evolve in complex/changing environment (e.g, Cloud, embedded systems)
- Need dynamic adaptation to best deliver the service (e.g., Waymo<sup>1</sup>, Netflix<sup>1</sup>)

#### Software languages ...

Can abstract concerns into high level constructs (e.g., memory management)

Vision : abstract self-adaption into high level language constructs

<sup>1</sup> Cf. https://waymo.com, https://www.netflix.com

SEALS 000000 cifying Adaptive Semantion

Conclusion

### What is a Self-Adaptable Language ?

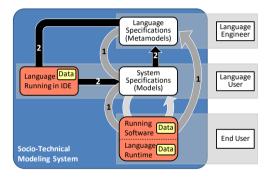
" A software language that abstracts the design and execution of feedback loops in the design-time environment and the run-time environment "

- 1. Free the language user from the implementation of :
  - The feedback loop
  - The trade-off analysis
- 2. Allow continuous and automatic evolution of itself

0.00

Self-Adaptable Language

#### L-MODA | Languages, Models, and Data



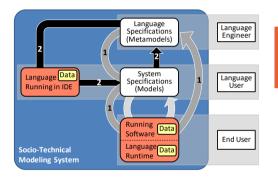


Self-Adaptable Language ●○ elf-Adaptable Virtual Machines

SEALS 000000 ecifying Adaptive Semantic

Conclusion

### L-MODA | Languages, Models, and Data



1) Runtime Feedback Loop

Use run-time data, model & metamodel  $\rightarrow$  adaptation of language semantics

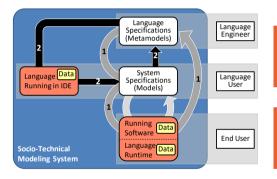


Self-Adaptable Language ○●○ elf-Adaptable Virtual Machines

SEALS 000000 cifying Adaptive Semantic

Conclusion

#### L-MODA | Languages, Models, and Data



L-MODA Conceptual Framework for Self-Adaptable Languages

#### 1) Runtime Feedback Loop

Use run-time data, model & metamodel  $\rightarrow$  adaptation of language semantics

#### 2) Design Feedback Loop

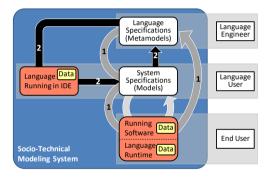
Use design-time data, models & metamodel  $\rightarrow$  adaptation of syntax, pragmatics & semantics



Self-Adaptable Language ○○● Self-Adaptable Virtual Machines

SEALS 000000 pecifying Adaptive Semanti 0000 Conclusion

### L-MODA | Stakeholders





Self-Adaptable Language ○○● Self-Adaptable Virtual Machines

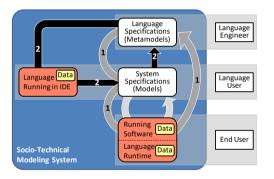
SEALS

ecifying Adaptive Semantic

Conclusion

# L-MODA | Stakeholders

#### Various uses of the feedback loops ...





Self-Adaptable Language ○○● Self-Adaptable Virtual Machines

SEALS 000000 pecifying Adaptive Seman

Conclusion

# L-MODA | Stakeholders

Various uses of the feedback loops ...

Language Specifications Language Engineer (Metamodels) Language Data System 1 Language Specifications **Running in IDE** User (Models) Running Data Software End User Language Data Socio-Technical Runtime **Modeling System** 

Examples for the Runtime Feedback Loop :



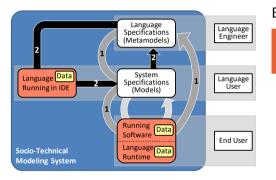
Self-Adaptable Language ○○● Self-Adaptable Virtual Machines

SEALS 000000 pecifying Adaptive Semant

Conclusion

# L-MODA | Stakeholders

Various uses of the feedback loops ...



 ${\sf Examples \ for \ the \ Runtime \ Feedback \ Loop}:$ 

Language engineer in complete control Tailor the language to a particular trade-off

L-MODA Conceptual Framework for Self-Adaptable Languages



Delegation of responsabilities

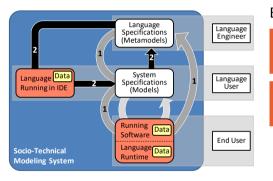
Self-Adaptable Language ○○● Self-Adaptable Virtual Machines

SEALS 000000 pecifying Adaptive Seman

Conclusion

# L-MODA | Stakeholders

Various uses of the feedback loops ...



Examples for the Runtime Feedback Loop :

Language engineer in complete control Tailor the language to a particular trade-off

**K** Language user custom adaptations Configure the adaptations for a system Delegation of responsabilities



Self-Adaptable Language ○○● Self-Adaptable Virtual Machines

SEALS 000000 pecifying Adaptive Seman

Conclusion

# L-MODA | Stakeholders

Various uses of the feedback loops ...

Language Language Engineer Specifications (Metamodels) Language Data System Language 1 Specifications **Running in IDE** User (Models) Running Data Software End User Language Data Socio-Technical Runtime **Modeling System** 

L-MODA Conceptual Framework for Self-Adaptable Languages Examples for the Runtime Feedback Loop :

Language engineer in complete control Tailor the language to a particular trade-off

**K** Language user custom adaptations Configure the adaptations for a system

**C** End-user preferences Indicate preference for trade-offs



elf-Adaptable Language

Self-Adaptable Virtual Machines

SEALS 000000 ecifying Adaptive Semantic

Conclusion

# Experimentation

#### The case of Self-Adaptable Virtual Machines



SEALS

ecifying Adaptive Semantic

Conclusion

#### What are Self-Adaptable Virtual Machines

- A specific case of Self-Adaptable Languages
- Runtime Feedback loop in language operational semantics
- ▶ In our experiment : Pluggable architecture with delegation of responsibilities

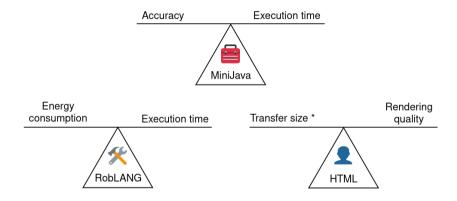


SEALS 000000 pecifying Adaptive Semantic

Conclusion

### Motivating Examples

DiverSE



\* Transfer size is proportional to energy consumption (Cf. https://www.websitecarbon.com/)

Introduction	Self-Adaptable Virtual Machines	
0	000000	

#### Adaptations

DiverSE

2

Towards Self-Adaptable Languages - Jouneaux et al.

elf-Adaptable Language 00 Self-Adaptable Virtual Machines

SEALS 000000 cifying Adaptive Semantic

Conclusion

### Adaptations (MiniJava)

Applied Approximate Loop Unrolling [1] on image processing algorithm (Sobel)



Standard output



#### Approximated output

[1] M. Rodriguez-Cancio, B. Combemale, and B. Baudry, "Approximate loop unrolling," in *Proceedings of the* 16th ACM International Conference on Computing Frontiers, ACM, 2019



Towards Self-Adaptable Languages - Jouneaux et al.

SEALS 000000 ecifying Adaptive Semantic

Conclusion

# Adaptations (RobLANG)

Applied a motor speed reduction on basic actions

Rational : 
$$P_i = P_{max}(\frac{Speed_i}{Speed_{max}})^3$$
 [2]

Three programs studied :

- Move forward/backward
- Turn left/right
- Combination of moves and turns (square patterns)

[2] A. Al-Mofleh, S. Taib, W. Salah, et al., "Importance of energy efficiency: From the perspective of electrical equipments," in *Proceedings of the 2nd International Conference on Science and Technology (ICSTIE)*, 2008



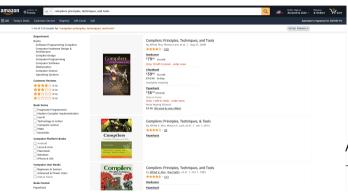
If-Adaptable Language

Self-Adaptable Virtual Machines

SEALS 000000 cifying Adaptive Semantic

Conclusion

# Adaptations (HTML)



- Conditional loading of resources
- Perforation of HTML lists
- Image degradation

Applied on the top 100 websites ightarrow 45 still deliver the content

#### Standard website





Self-Adaptable Language

# Adaptations (HTML)

Self-Adaptable Virtual Machines

SEALS 000000 ecifying Adaptive Semanti

Conclusion



1-16 of 70 results for"compilers principles, techniques, and tools" Sort by: Teatured



Compilers: Principles, Techniques, and Tools

by Alfred Abo, Montoa Lam, et al. | Aug 31, 2006 4.2 out of 5 stars318 Bardcover 579,36573,3610 rent Only 16 loft in stock - order scon.

eTexthook \$39.99\$39.99to rent \$74.99to buy Available instantly





Conditional loading of resources

Perforation of HTML lists

Image degradation

Applied on the top 100 websites ightarrow 45 still deliver the content

Adapted website



Towards Self-Adaptable Languages - Jouneaux et al.

SEALS

ecifying Adaptive Semantics

Conclusion

#### Evaluate the relevance of proposed adaptation

TL;DR : Good results but ...

- Correct adaptations of MiniJava
- Up to 10x more actions on RobLANG
- Energy reduction from -8.7% to 97.2% with a mean of 63.8% [54.2%, 73.4%]

- Performance overhead
- Lack of control on the adaptations
- Deal with the diversity of programs oblivious of the adaptations performed



EALS

ifying Adaptive Semantics

Conclusion

#### Compare Language-level vs System-level abstractions

Attempt to implement as library of the language

VMs	Feedback loop	Trade-off reasoning	Feedback loop calls	Interaction with the domain
MiniJava	=	=	+	+
RobLang	=	+ +	+	-
HTML (JS)	=	=	=	-

Comparison of implementation simplicity (+ in favor of language-level)

tion Self-Adaptable Lan

DiverSE

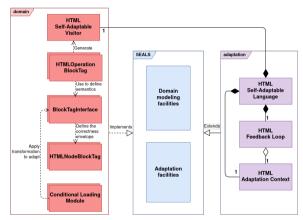
Self-Adaptable Virtual Machines

SEALS

ecifying Adaptive Semantics

Conclusion

### SEALS : A Framework for Building Self-Adaptable Virtual Machines



Approach overview on the HTML use case

DiverSE

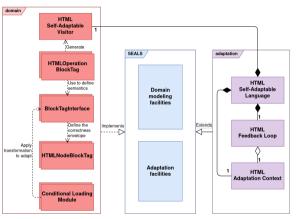
Self-Adaptable Virtual Machines

SEALS •00000 ecifying Adaptive Semantic

Conclusion

### SEALS : A Framework for Building Self-Adaptable Virtual Machines

- Modeling of domain concepts
  - 1. Define the abstract syntax
  - 2. Create the correctness envelope
  - 3. Implement the operational semantics



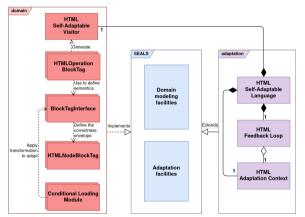
Approach overview on the HTML use case

SEALS •00000 ecifying Adaptive Semantic

Conclusion

# SEALS : A Framework for Building Self-Adaptable Virtual Machines

- Modeling of domain concepts
  - $1. \ \mbox{Define the abstract syntax}$
  - 2. Create the correctness envelope
  - 3. Implement the operational semantics
- Adaptation process' specialization
  - $1. \ \mbox{Specialize the Adaptation Context}$
  - 2. Specialize the Feedback loop
  - 3. Connect the components



Approach overview on the HTML use case



lf-Adaptable Language

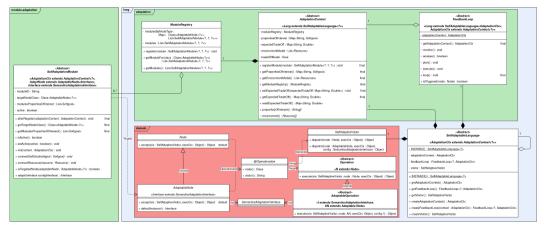
Self-Adaptable Virtual Machines

SEALS

becifying Adaptive Semantic

Conclusion

#### Framework implementation - Global view



#### Class diagram of the SEALS Framework

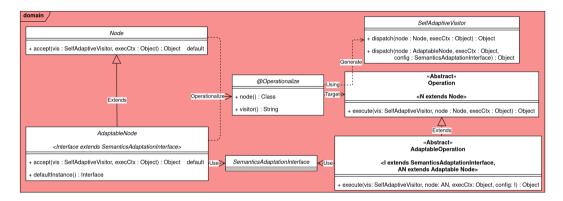


SEALS

occifying Adaptive Semantics

Conclusion

#### Framework implementation - Domain modeling



Class diagram of the SEALS Framework (domain package)



SEALS

ecifying Adaptive Semantics

Conclusion

#### Framework implementation - Adaptation process



Class diagram of the SEALS Framework (adaptation package)



SEALS

pecifying Adaptive Semantics

Conclusion

#### Framework implementation - Adaptation modules

«Abstract» SelfAdaptationModule	
<adaptationctx adaptationcontext<?="" extends="">, AdaptNode extends AdaptableNode<interface>, Interface extends SemanticsAdaptationInterface&gt;</interface></adaptationctx>	
- moduleID : String	
- targetNodeClass : Class <adaptablenode<?>&gt;</adaptablenode<?>	
<ul> <li>modulesPropertiesOfInterest : List<softgoal></softgoal></li> </ul>	
- active : boolean	
+ afterRegister(adaptationContext : AdaptationContext) : void	final
+ getTargetNodeClass() : Class <adaptablenode<?>&gt;</adaptablenode<?>	final
+ getModulesPropertiesOfInterest() : List <softgoal></softgoal>	final
+ isActive() : boolean	
+ setActive(active : boolean) : void	
+ init(context : AdaptationCtx) : void	
+ connectSoftGoal(softgoal : Softgoal) : void	
+ connectResource(resource : Resource) : void	
+ isTargetedNode(adaptableNode : AdaptableNode ) : boolean	
+ adapt(Interface configInterface) : Interface	

Class diagram of the SEALS Framework (module package)



SEALS Spec

ecifying Adaptive Semantic

Conclusion

#### Framework implementation - Self-Adaptatble Language

«Abstract» SelfAdaptableLanguage	1
<adaptationctx adaptationcontext<?="" extends="">&gt;</adaptationctx>	
- INSTANCE : SelfAdaptableLanguage	$\triangleright$
- adaptationContext : AdaptationCtx	
- feedbackLoop : FeedbackLoop , AdaptationCtx	
- visitor : SelfAdaptiveVisitor	
+ INSTANCE() : SelfAdaptableLanguage	
+ getAdaptationContext() : AdaptationCtx	
+ getFeedbackLoop() : FeedbackLoop , AdaptationCtx	
+ getVisitor() : SelfAdaptiveVisitor	
+ createAdaptationContext() : AdaptationCtx	
+ createFeedbackLoop(context : AdaptationCtx) : FeedbackLoop , AdaptationCtx	
+ createVisitor() : SelfAdaptiveVisitor	

#### Class diagram of the SEALS Framework (lang package)



Towards Self-Adaptable Languages - Jouneaux et al.

Adaptable Language

Self-Adaptable Virtual Machines

SEALS 000000 Specifying Adaptive Semantics

Conclusion

### Specifying Adaptive Semantics

Based on a language definition :



SEALS 000000 Specifying Adaptive Semantics

Conclusion

## Specifying Adaptive Semantics

Based on a language definition :

- Abstract syntax as metamodel
- Dynamic information merged in the metamodel
- Modular definition of the semantics (I-MSOS)



SEALS 000000 Specifying Adaptive Semantics

Conclusion

## Specifying Adaptive Semantics

Based on a language definition :

- Abstract syntax as metamodel
- Dynamic information merged in the metamodel
- Modular definition of the semantics (I-MSOS)

To make it adaptive we need :



SEALS 000000 Specifying Adaptive Semantics

Conclusion

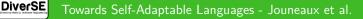
# Specifying Adaptive Semantics

Based on a language definition :

- Abstract syntax as metamodel
- Dynamic information merged in the metamodel
- Modular definition of the semantics (I-MSOS)

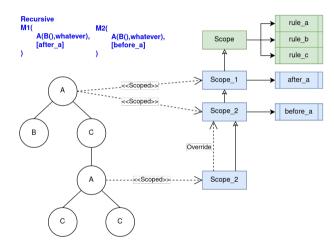
To make it adaptive we need :

- Additional semantics rules for adaptation
- Mechanism for adaptation rule introduction
- Dynamic selection of semantics rule to apply



SEALS 000000 Specifying Adaptive Semantics

Conclusion



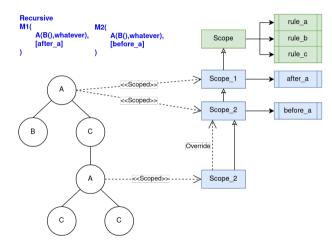


Self-Adaptable Virtual Machines

SEALS 000000 Specifying Adaptive Semantics

Conclusion

#### Scopes of semantics rules

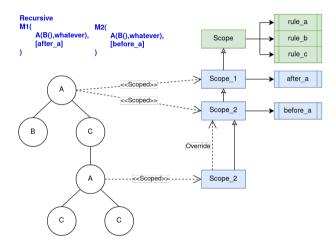


 Original semantics rules defined in the global scope



SEALS 000000 Specifying Adaptive Semantics  $0 \bullet 000$ 

Conclusion

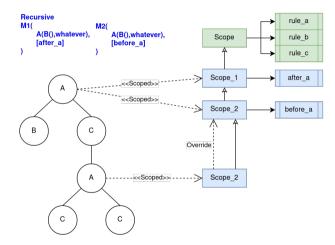


- Original semantics rules defined in the global scope
- Adaptation introduce new rules through new scopes



SEALS 000000 Specifying Adaptive Semantics  $0 \bullet 000$ 

Conclusion



- Original semantics rules defined in the global scope
- Adaptation introduce new rules through new scopes
- Instantiation of scopes defined using pointcuts

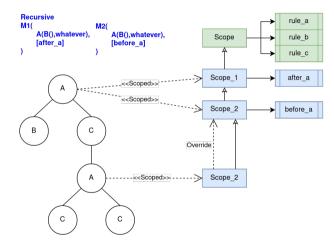


Self-Adaptable Virtual Machines

Towards Self-Adaptable Languages - Jouneaux et al.

SEALS 000000 Specifying Adaptive Semantics  $0 \bullet 000$ 

Conclusior



- Original semantics rules defined in the global scope
- Adaptation introduce new rules through new scopes
- Instantiation of scopes defined using pointcuts
- Two types of scopes :
  - Blocking
  - Recursive

elf-Adaptable Virtual Machines

SEALS 000000 Specifying Adaptive Semantics 00000

Conclusion

## Meta-language for Original Semantics

```
model minijava.ecore
   import helper.java as help
 3
   rule divide_lhs,
        Div(a1, a2) \rightarrow Div(a1', a2)
 6
   where
        a1 \rightarrow a1'
 9
   rule divide_rhs,
        Div(Number(n1), a2) \rightarrow Div(Number(n1), a2')
   where
12
        a2 \rightarrow a2'
13
   rule divide_result ,
14
        Div(Number(n1), Number(n2)) \rightarrow n3
15
16
   where
17
        n^2 = 0
        n3 : help.div(Number(n1), Number(n2))
18
```

Inspired by Spoofax DynSem



SEALS 000000 Specifying Adaptive Semantics 00000

Conclusion

# Meta-language for Original Semantics

```
model minijava.ecore
   import helper.java as help
 3
   rule divide_lhs,
        Div(a1, a2) \rightarrow Div(a1', a2)
 5
   where
        a1 \rightarrow a1'
 9
   rule divide_rhs,
        Div(Number(n1), a2) \rightarrow Div(Number(n1), a2')
   where
12
        a2 \rightarrow a2'
13
14
   rule divide result.
        Div(Number(n1), Number(n2)) \rightarrow n3
15
   where
17
        n^2 = 0
        n3 : help.div(Number(n1), Number(n2))
18
```

- Inspired by Spoofax DynSem
- Model merging metamodel and dynamic information



SEALS 000000 Specifying Adaptive Semantics 00000

Conclusion

## Meta-language for Original Semantics

```
model minijava.ecore
   import helper.java as help
 3
   rule divide_lhs,
        Div(a1, a2) \rightarrow Div(a1', a2)
   where
        a1 \rightarrow a1'
   rule divide_rhs,
 9
        Div(Number(n1), a2) \rightarrow Div(Number(n1), a2')
   where
        a2 \rightarrow a2'
13
14
   rule divide result.
        Div(Number(n1), Number(n2)) \rightarrow n3
15
   where
17
        n^2 = 0
        n3 : help.div(Number(n1), Number(n2))
18
```

- Inspired by Spoofax DynSem
- Model merging metamodel and dynamic information
- Import external operations



elf-Adaptable Virtual Machines

SEALS 000000 Specifying Adaptive Semantics 00000

Conclusion

## Meta-language for Original Semantics

```
model minijava.ecore
   import helper.java as help
 3
   rule divide_lhs,
        Div(a1, a2) \rightarrow Div(a1', a2)
   where
        a1 \rightarrow a1'
   rule divide_rhs,
 9
        Div(Number(n1), a2) \rightarrow Div(Number(n1), a2')
   where
        a2 \rightarrow a2'
13
   rule divide_result ,
14
        Div(Number(n1), Number(n2)) \rightarrow n3
15
   where
17
        n^2 = 0
18
        n3 : help.div(Number(n1), Number(n2))
```

- Inspired by Spoofax DynSem
- Model merging metamodel and dynamic information
- Import external operations
- A set of semantic rules
  - Conclusion as reduction over concepts
  - Reduction premises
  - Side condition
  - Binding computed values

20

Introduction

-Adaptable Language

elf-Adaptable Virtual Machines

SEALS 000000 Specifying Adaptive Semantics  $000 \bullet 0$ 

Conclusion

#### Meta-language for Original Semantics

```
model minijava.ecore
   import helper.java as help
 3
   rule assign_expr,
        Assign(var, expr) \rightarrow Assign(var, expr')
 6
   where
       expr \rightarrow expr'
 8
 9
   rule assign_dec,
10
        Assign(var, Number(n))
11
       ->
12
       var.def.value = Number(n)
13
14
   rule var_reference,
15
        VarRef(def) \rightarrow def.value
```

Accessing dynamic information



f-Adaptable Language O elf-Adaptable Virtual Machines

SEALS

Specifying Adaptive Semantics  $0000 \bullet$ 

Conclusion

## Meta-language for Semantics Adaptation

```
model minijava.ecore
  semantics minijava.sem
3
  recursive ApproximateDouble{
       match Assignement(VarRef(def), expr)
       where def.type = Float
9
   Before binop_rhs rule binop_rhs_float,
       Binop(Double(n1), a2)
      ->
       Binop(Float(n1), a2)
13
  Before binop result rule binop result float.
14
15
       Binop(Number(n1), Double(n2))
16
      ->
17
       Binop(Number(n1), Float(n2))
18 }
```

- Dependence to the semantics
- Pointcut definition
  - Structural matching
  - Additional constraints
- Adaptation rules
  - Kind of adaptation rule
  - Affected rule in semantics
  - Adaptation semantic rule

SEALS 000000 pecifying Adaptive Semantics

Conclusion •00

## Conclusion

► The concept of Self-Adaptable Language and its conceptual framework

- A framework to implement Self-Adaptable Virtual Machines
- Ongoing work on specification of adaptive semantics





SEALS 000000 pecifying Adaptive Semantics

Conclusion

# **Open Questions**

What is the killer app to demonstrate SALs ?

- What is the best design pattern(s) for the implementation ?
- Opinions on specification of adaptive operational semantics

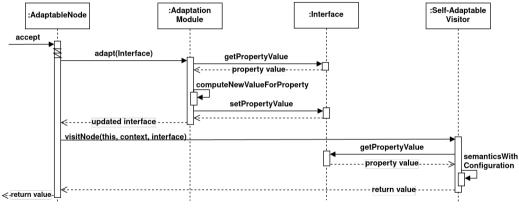


Conclusion 000

# Thanks for your attention !



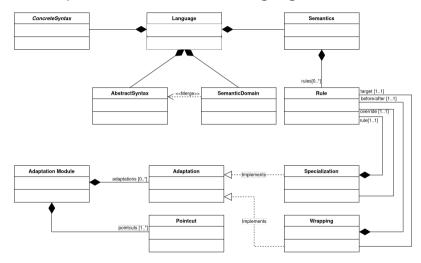
#### Correctness envelope implementation



Example of use of the correctness envelope

Towards Self-Adaptable Languages - Jouneaux et al.

#### Metamodel of adaptive semantics meta languages





## References



M. Rodriguez-Cancio, B. Combemale, and B. Baudry, "Approximate loop unrolling," in *Proceedings of the 16th ACM International Conference on Computing Frontiers*, ACM, 2019.

A. Al-Mofleh, S. Taib, W. Salah, and M. Azizan, "Importance of energy efficiency: From the perspective of electrical equipments," in *Proceedings of the 2nd International Conference on Science and Technology (ICSTIE)*, 2008.

