Towards Self-Adaptable Languages

Gwendal Jouneaux [1] Olivier Barais [1]
Benoit Combemale [1] Gunter Mussbacher [2]

[1]Univ. Rennes, Inria, IRISA – Rennes, France

[2]McGill University – Montreal, Canada

ALE Seminar — May 24, 2022

Context

Software ...

Context

Software ...

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

Context

Software ...

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

[1] Cf. `https://waymo.com`, `https://www.netflix.com`

Context

Software ...

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

Software languages ...

[1] Cf. `https://waymo.com`, `https://www.netflix.com`

Context

Software ...

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

Software languages ...

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

[1] Cf. `https://waymo.com`, `https://www.netflix.com`

Context

Software ...

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

Software languages ...

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

Vision : abstract self-adaption into high level language constructs

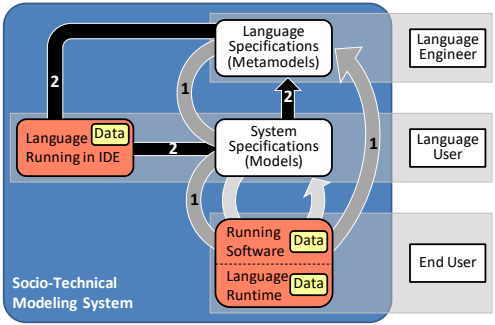
[1] Cf. `https://waymo.com`, `https://www.netflix.com`

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* "

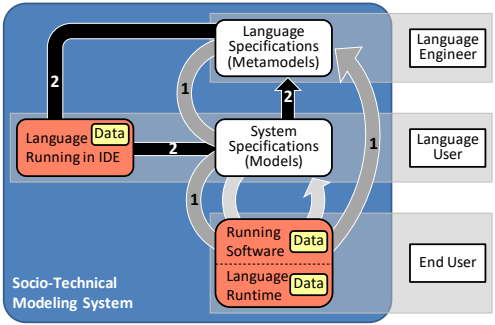
1. Free the language user from the implementation of :
   - The feedback loop
   - The trade-off analysis
2. Allow continuous and automatic evolution of itself

L-MODA | Languages, Models, and Data



L-MODA Conceptual Framework for Self-Adaptable Languages

L-MODA | Languages, Models, and Data

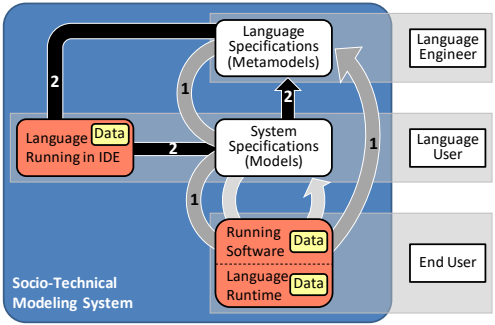


L-MODA Conceptual Framework for Self-Adaptable Languages

1) Runtime Feedback Loop

Use run-time data, model & metamodel
→ adaptation of language semantics

L-MODA | Languages, Models, and Data



L-MODA Conceptual Framework for Self-Adaptable Languages

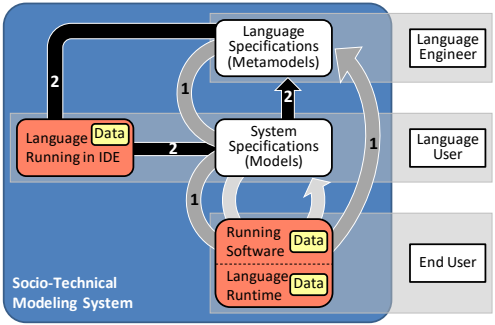
1) Runtime Feedback Loop

Use run-time data, model & metamodel
→ adaptation of language semantics

2) Design Feedback Loop

Use design-time data, models & metamodel
→ adaptation of syntax, pragmatics & semantics

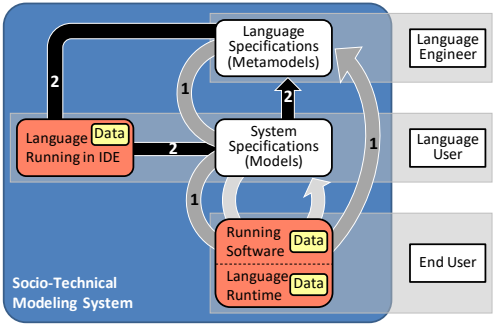
L-MODA | Stakeholders



L-MODA Conceptual Framework for Self-Adaptable Languages

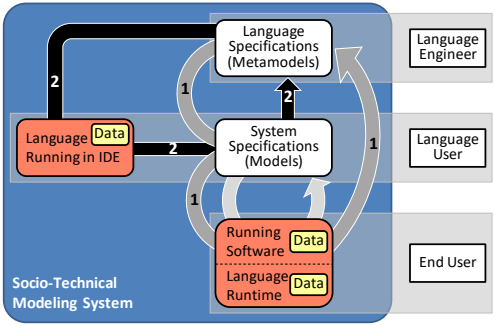
L-MODA | Stakeholders

Various uses of the feedback loops ...



L-MODA Conceptual Framework for Self-Adaptable Languages

L-MODA | Stakeholders



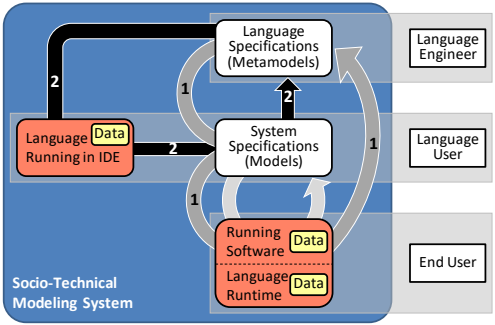
L-MODA Conceptual Framework for Self-Adaptable Languages

Various uses of the feedback loops ...

Examples for the Runtime Feedback Loop :

Delegation of responsabilities

L-MODA | Stakeholders



L-MODA Conceptual Framework for Self-Adaptable Languages

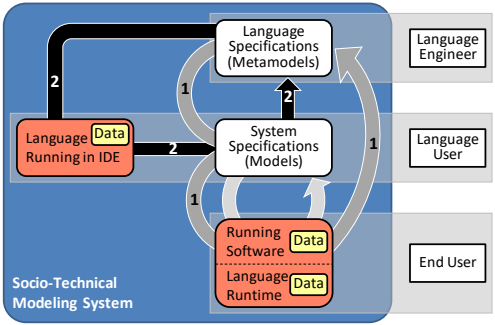
Various uses of the feedback loops ...

Examples for the Runtime Feedback Loop :

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

Delegation of responsabilities

L-MODA | Stakeholders



L-MODA Conceptual Framework for Self-Adaptable Languages

Various uses of the feedback loops ...

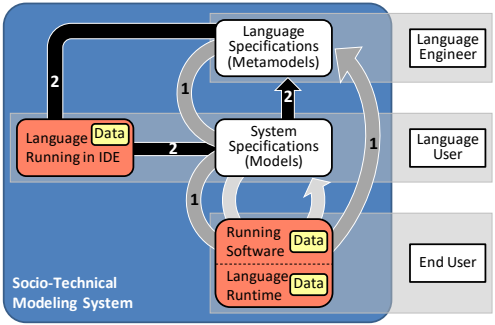
Examples for the Runtime Feedback Loop :

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

Language user custom adaptations
Configure the adaptations for a system

Delegation of responsabilities

L-MODA | Stakeholders



L-MODA Conceptual Framework for Self-Adaptable Languages

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

Language user custom adaptations
Configure the adaptations for a system

End-user preferences
Indicate preference for trade-offs

Delegation of responsabilities

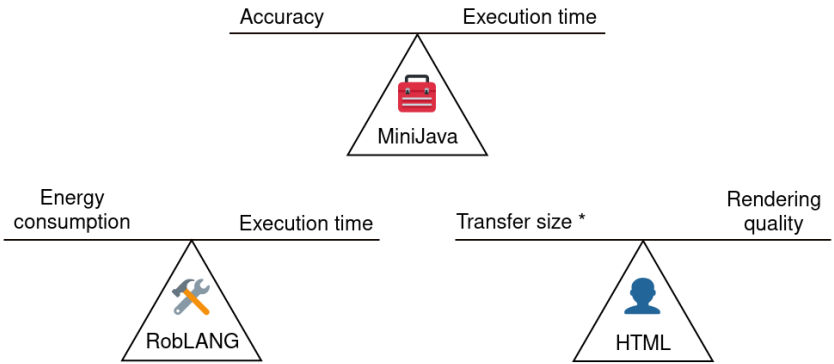
Experimentation

The case of Self-Adaptable Virtual Machines

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

Motivating Examples

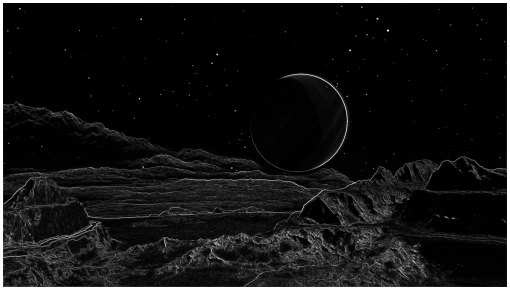


* Transfer size is proportional to energy consumption (Cf. `https://www.websitecarbon.com/`)

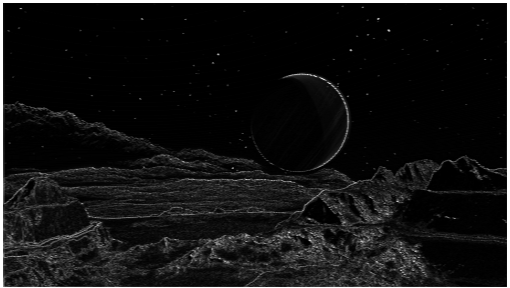
Adaptations

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

Adaptations (RobLANG)

Applied a motor speed reduction on basic actions

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

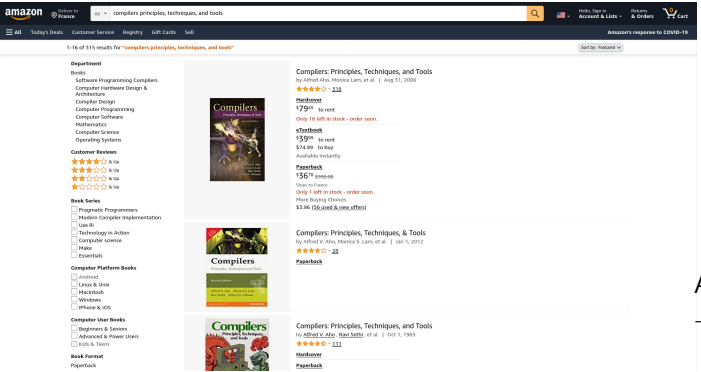
Three programs studied :

- 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

Adaptations (HTML)

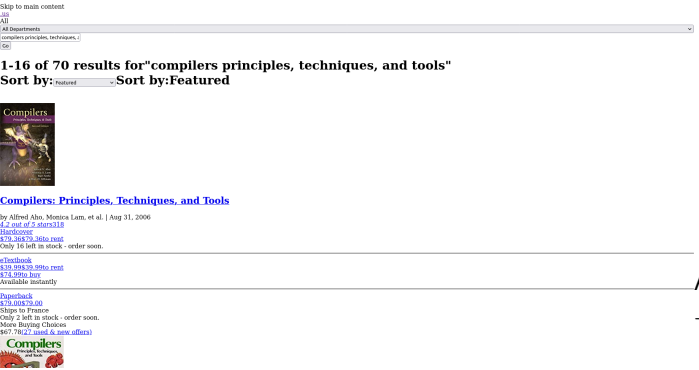


Standard website

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

Applied on the top 100 websites
$\rightarrow$ 45 still deliver the content

Adaptations (HTML)



Adapted website

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

Applied on the top 100 websites
$\rightarrow$ 45 still deliver the content

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

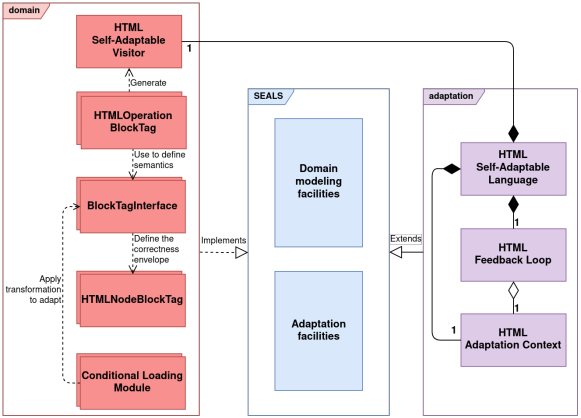
Compare Language-level vs System-level abstractions

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)

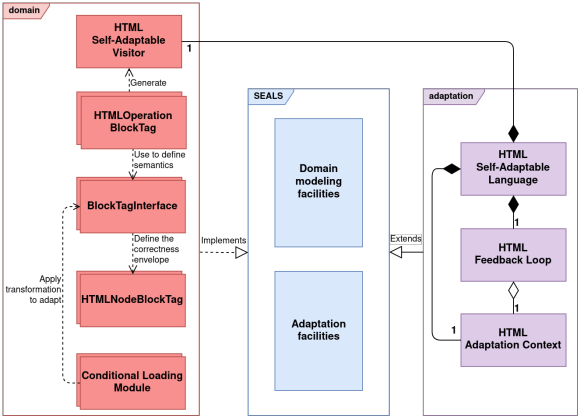
SEALS : A Framework for Building Self-Adaptable Virtual Machines



Approach overview on the HTML use case

SEALS : A Framework for Building Self-Adaptable Virtual Machines

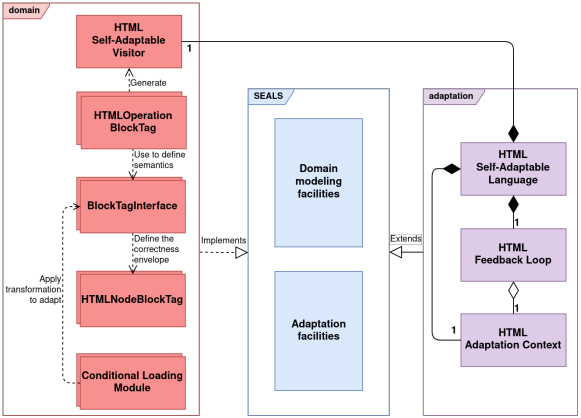
- Modeling of domain concepts
  1. Define the abstract syntax
  2. Create the correctness envelope
  3. Implement the operational semantics



Approach overview on the HTML use case

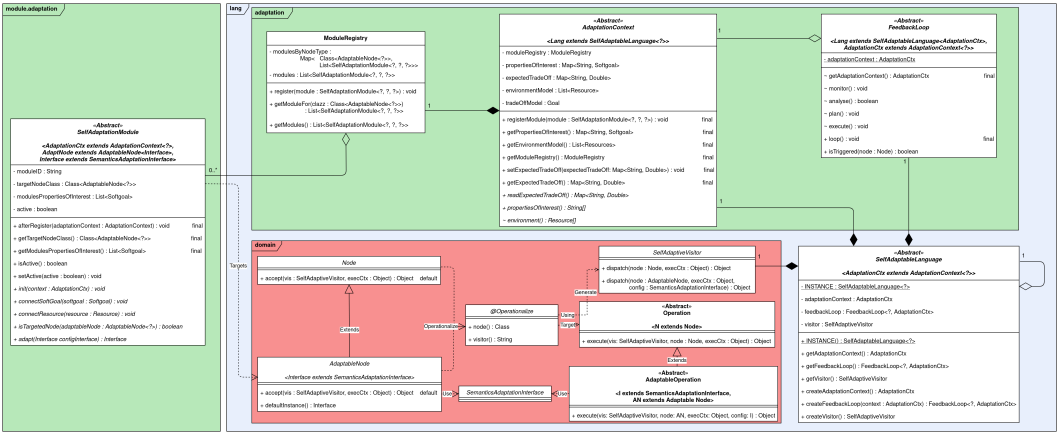
SEALS : A Framework for Building Self-Adaptable Virtual Machines

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



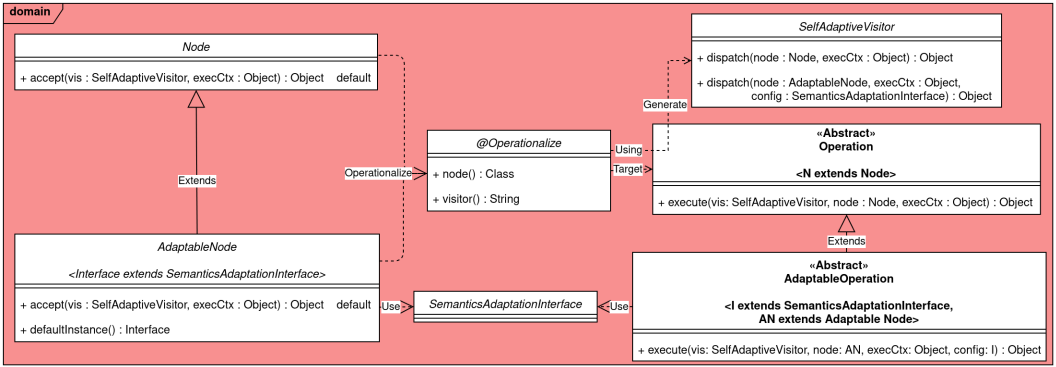
Approach overview on the HTML use case

Framework implementation - Global view



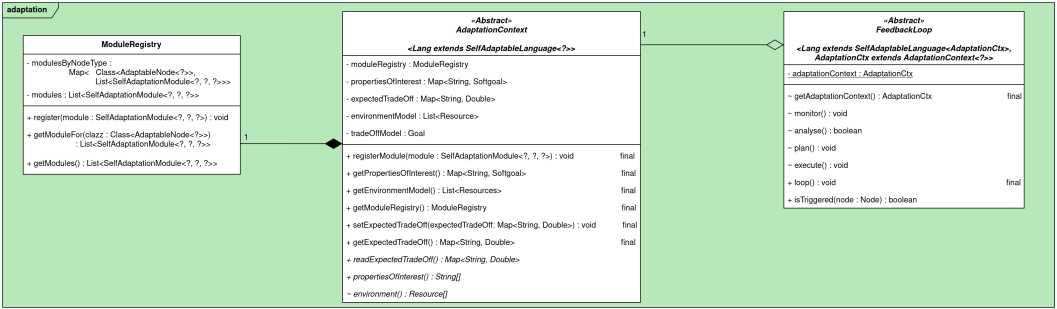
Class diagram of the SEALS Framework

Framework implementation - Domain modeling



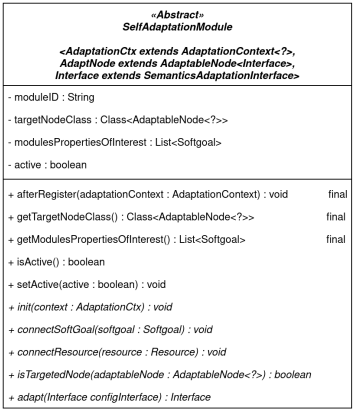
Class diagram of the SEALS Framework (domain package)

Framework implementation - Adaptation process



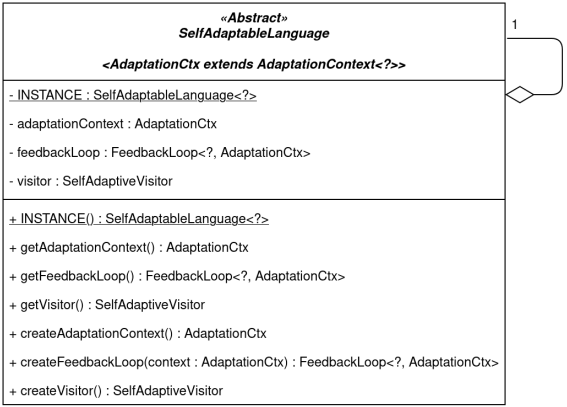
Class diagram of the SEALS Framework (adaptation package)

Framework implementation - Adaptation modules



Class diagram of the SEALS Framework (module package)

Framework implementation - Self-Adaptatble Language



Class diagram of the SEALS Framework (lang package)

Specifying Adaptive Semantics

Based on a language definition :

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)

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 :

Specifying Adaptive Semantics

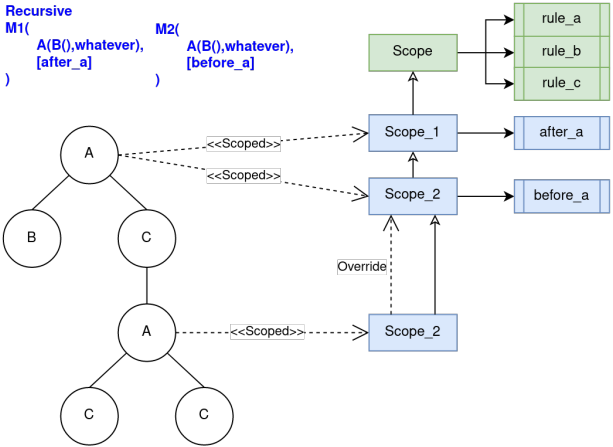
Based on a language definition :

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

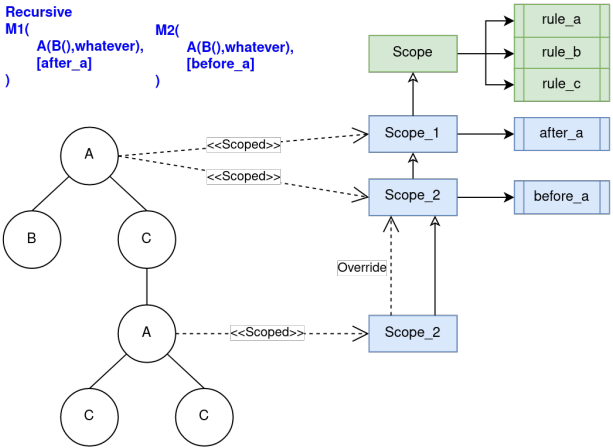
To make it adaptive we need :

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

Scopes of semantics rules

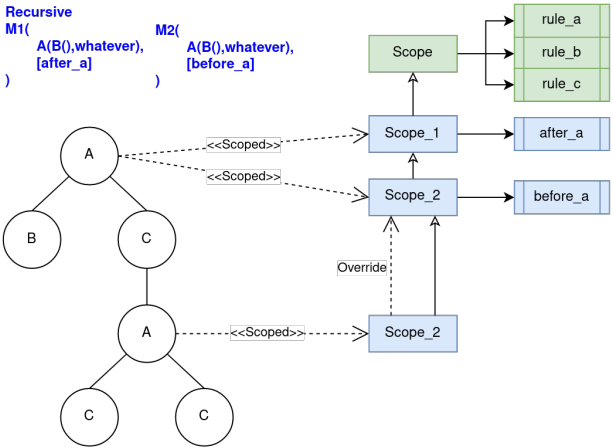


Scopes of semantics rules



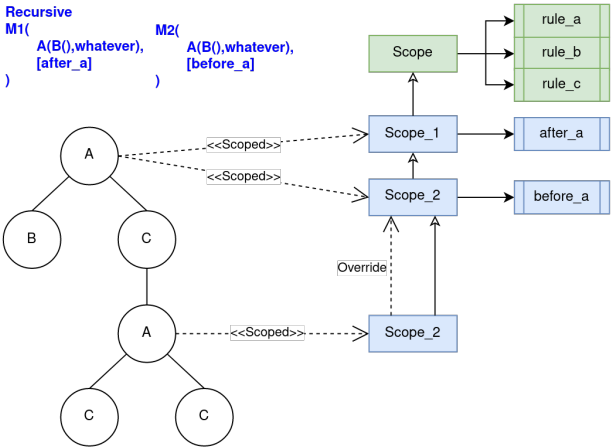
- Original semantics rules defined in the global scope

Scopes of semantics rules



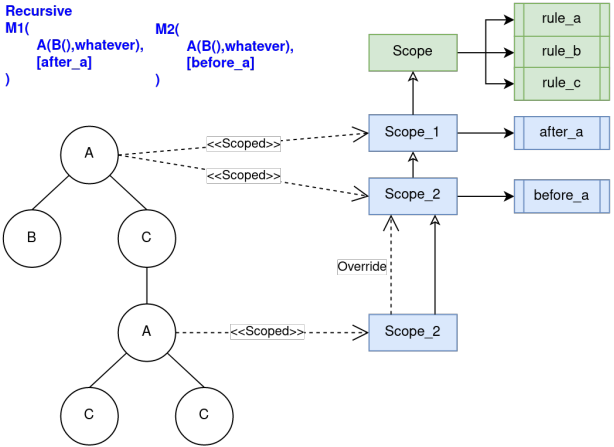
- Original semantics rules defined in the global scope
- Adaptation introduce new rules through new scopes

Scopes of semantics rules



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

Scopes of semantics rules



- 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

Meta-language for Original Semantics

```
1  model minijava.ecore
2  import helper.java as help
3
4  rule divide_lhs,
5      Div(a1, a2) -> Div(a1', a2)
6  where
7      a1 -> a1'
8
9  rule divide_rhs,
10     Div(Number(n1), a2) -> Div(Number(n1), a2')
11 where
12     a2 -> a2'
13
14 rule divide_result,
15     Div(Number(n1), Number(n2)) -> n3
16 where
17     n2 != 0
18     n3 : help.div(Number(n1), Number(n2))
```

- Inspired by Spoofax DynSem

Meta-language for Original Semantics

```
1  model minijava.ecore
2  import helper.java as help
3
4  rule divide_lhs,
5      Div(a1, a2) -> Div(a1', a2)
6  where
7      a1 -> a1'
8
9  rule divide_rhs,
10     Div(Number(n1), a2) -> Div(Number(n1), a2')
11 where
12     a2 -> a2'
13
14 rule divide_result,
15     Div(Number(n1), Number(n2)) -> n3
16 where
17     n2 != 0
18     n3 : help.div(Number(n1), Number(n2))
```

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

Meta-language for Original Semantics

```
1  model minijava.ecore
2  import helper.java as help
3
4  rule divide_lhs,
5      Div(a1, a2) -> Div(a1', a2)
6  where
7      a1 -> a1'
8
9  rule divide_rhs,
10     Div(Number(n1), a2) -> Div(Number(n1), a2')
11 where
12     a2 -> a2'
13
14 rule divide_result,
15     Div(Number(n1), Number(n2)) -> n3
16 where
17     n2 != 0
18     n3 : help.div(Number(n1), Number(n2))
```

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

Meta-language for Original Semantics

```
1  model minijava.ecore
2  import helper.java as help
3
4  rule divide_lhs,
5      Div(a1, a2) -> Div(a1', a2)
6  where
7      a1 -> a1'
8
9  rule divide_rhs,
10     Div(Number(n1), a2) -> Div(Number(n1), a2')
11 where
12     a2 -> a2'
13
14 rule divide_result,
15     Div(Number(n1), Number(n2)) -> n3
16 where
17     n2 != 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

Meta-language for Original Semantics

```
1  model minijava.ecore
2  import helper.java as help
3
4  rule assign_expr,
5      Assign(var, expr) -> Assign(var, expr')
6  where
7      expr -> 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) -> def.value
```

Accessing dynamic information

Meta-language for Semantics Adaptation

```
1  model minijava.ecore
2  semantics minijava.sem
3
4  recursive ApproximateDouble{
5
6      match Assignement(VarRef(def), expr)
7      where def.type = Float
8
9  Before binop_rhs rule binop_rhs_float,
10     Binop(Double(n1), a2)
11     ->
12     Binop(Float(n1), a2)
13
14 Before binop_result rule binop_result_float,
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

Conclusion

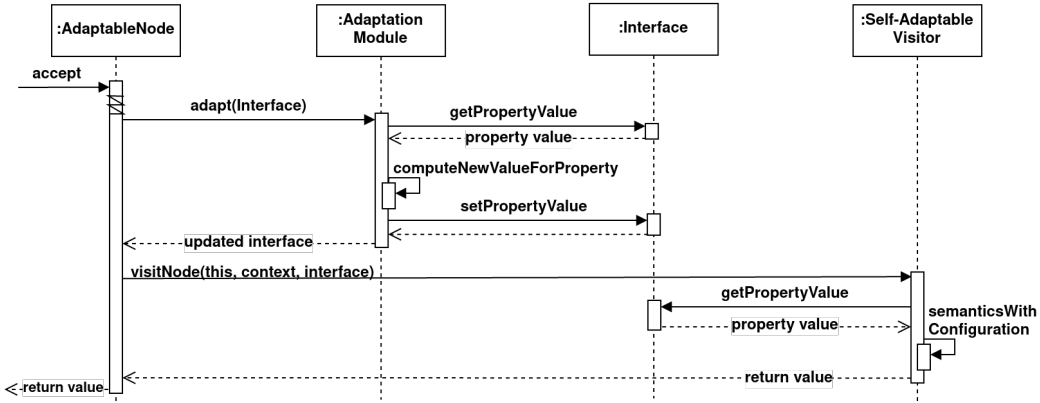
- The concept of Self-Adaptable Language and its conceptual framework
- A framework to implement Self-Adaptable Virtual Machines
- Ongoing work on specification of adaptive semantics

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

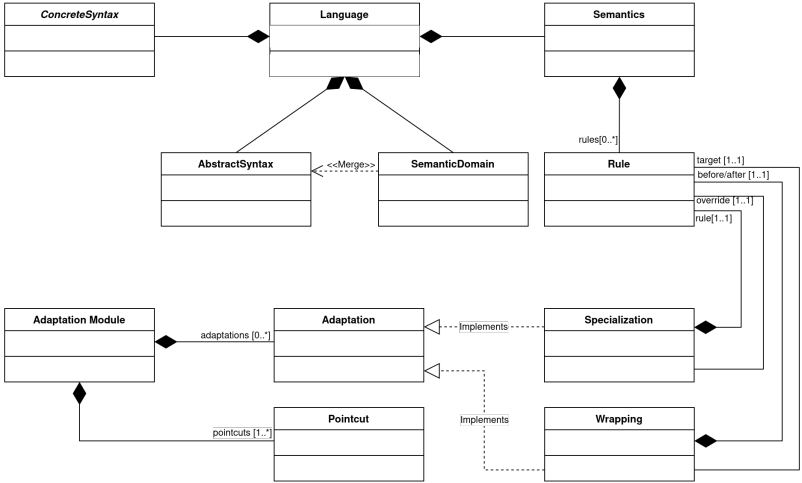
Thanks for your attention !

Correctness envelope implementation



Example of use of the correctness envelope

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.