Domain-Level Debugging for Compiled DSLs with the GEMOC Studio
(Tool Demo)

Erwan Bousse  Tanja Mayerhofer  Manuel Wimmer

TU Wien (Austria)

September 17, 2017
1. **Context**

2. **Domain-level feedback**

3. **Demo**

4. **Conclusion**
(Domain-level) interactive debugging

- Many existing approaches to define **Domain Specific Languages (DSLs)** with execution semantics.
- Enable the use of **interactive debugging** (i.e., breakpoints, step into, etc.) to understand and investigate defects.

- Two main kinds of DSLs to consider:
  - interpreted DSLs (with operational semantics)
  - compiled DSLs (with translational semantics)
(Domain-level) interactive debugging

- Many existing approaches to define **Domain Specific Languages (DSLs)** with execution semantics.
- Enable the use of **interactive debugging** (ie. breakpoints, step into, etc.) to understand and investigate defects.

- Two main kinds of DSLs to consider:
  - interpreted DSLs (with operational semantics)
  - compiled DSLs (with translational semantics)
Interpreted VS compiled DSLs

Runtime services
Provides services at runtime by observing occurring steps and the changing state
Runtime services
Provides services at runtime by observing occurring steps and the changing state
Interpreted VS compiled DSLs

Runtime services
Provides services at runtime by observing occurring *steps* and the changing *state*
Interpreted VS compiled DSLs

Runtime services
Provides services at runtime by observing occurring steps and the changing state
Interpreted VS compiled DSLs

Runtime services
Provides services at runtime by observing occurring steps and the changing state
Interpreted VS compiled DSLs

Runtime services
Provides services at runtime by observing occurring steps and the changing state
Interpreted VS compiled DSLs

Runtime services
Provides services at runtime by observing occurring steps and the changing state
Interpreted VS compiled DSLs

Runtime services
Provides services at runtime by observing occurring steps and the changing state
Interpreted VS compiled DSLs

Runtime services
Provides services at runtime by observing occurring steps and the changing state

Erwan Bousse, Tanja Mayerhofer, Manuel Wimmer
Domain-Level Debugging for Compiled DSLs
Problem

How to provide domain-level interactive debugging when executing models conforming to compiled DLSs?

Idea

At runtime, automatically translate on-the-fly \textit{target} states and steps into \textit{source} states and steps, which can then be observed by domain-level tools.
Problem

How to provide domain-level interactive debugging when executing models conforming to compiled DLSs?

Idea

At runtime, automatically translate on-the-fly target states and steps into source states and steps, which can then be observed by domain-level tools.
Overview

Result

The same runtime services can be (re)used for both interpreted and compiled DSLs!
Overview

Result
The same runtime services can be (re)used for both interpreted and compiled DSLs!
Overview

The same runtime services can be (re)used for both interpreted and compiled DSLs!
Overview

Result
The same runtime services can be (re)used for both interpreted and compiled DSLs!
Overview

Result
The same runtime services can be (re)used for both interpreted and compiled DSLs!
Overview

Result
The same runtime services can be (re)used for both interpreted and compiled DSLs!
Demo

Demo...
Conclusion

- Providing interactive debugging for compiled DSLs is not trivial
- **Idea:** a *feedback manager* to translate target steps back to the source domain
- **Evaluation:** median execution slowdown of 1.8 times due to feedback
- **Future work:** manage non-GEMOC-ready compilers (e.g., *code generators*)

- **Eclipse Research Consortium GEMOC:** sustains the GEMOC studio as a research platform to experiment on the globalization of, possibly executable and heterogeneous, modeling languages
- Contributors are welcome!

http://gemoc.org/
https://github.com/eclipse/gemoc-studio-modeldebugging
Conclusion

- Providing interactive debugging for compiled DSLs is not trivial
- **Idea:** a *feedback manager* to translate target steps back to the source domain
- **Evaluation:** median execution slowdown of 1.8 times due to feedback
- **Future work:** manage non GEMOC-ready compilers (eg. *code generators*)

- **Eclipse Research Consortium GEMOC:** sustains the GEMOC studio as a research platform to experiment on the globalization of, possibly executable and heterogeneous, modeling languages
- Contributors are welcome!

http://gemoc.org/
https://github.com/eclipse/gemoc-studio-modeldebugging