

Combination of Domain-Specific Languages

Rafael Ugaz

Table of Contents

- 1 Context and problem
- 2 Background and Previous Work
- 3 Example
- 4 DSL Combination in AToMPM
- 5 Conclusion

Table of Contents

- 1 Context and problem
- 2 Background and Previous Work
- 3 Example
- 4 DSL Combination in AToMPM
- 5 Conclusion

Context and problem

- Engineering of DSLs
- Small and focused, intuitive notation for domain experts
- Lower the complexity of developing DSLs
- Avoid starting from scratch

Context and Problem

- Need for reuse of existing DSLs (similar benefits as reuse brought to software engineering)
- Need for libraries of reusable DSL building blocks or language patterns
- Most MM comp. techniques focus on the combination of one of the language modules (as, cs or sem)

Table of Contents

- 1 Context and problem
- 2 Background and Previous Work**
- 3 Example
- 4 DSL Combination in AToMPM
- 5 Conclusion

Research Internship 1 [7]

Literature review:

- Meyers [4] combines all three components in textual modeling environment MetaDepth.
- Pedro [5] uses *parametrization* to combine abstract syntax and semantics.
- De Lara [2] employs *hybrid concepts* to combine behavioral semantics.

Research Internship 2 [8]

Enabling technology:

- Code generation to Android (Java) of AS and OpSem
- Develop example DSLs
- Ad-hoc combination

Thesis

Contributions:

- Technique for combination of DSLs including their AS, CS and SEM
- Tool support in AToMPM
- Convert combined DSLs to Android (Java) code

AToMPM

- Multi-paradigm modeling environment
- Graphical modeling and textual commands
- Execution of model transformations
- Model everything explicitly
- Performance dependent on network communication

DSM environments synthesis

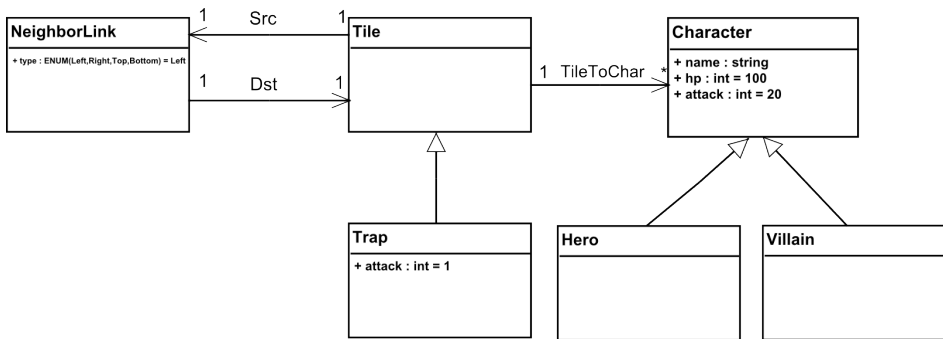
- SimpleClassDiagrams, based on UML class diagrams.
- TransformationRule, graph transformations.
- MoTif, rule scheduling.
- ConcreteSyntax, maps SVG \rightarrow MM element.

Table of Contents

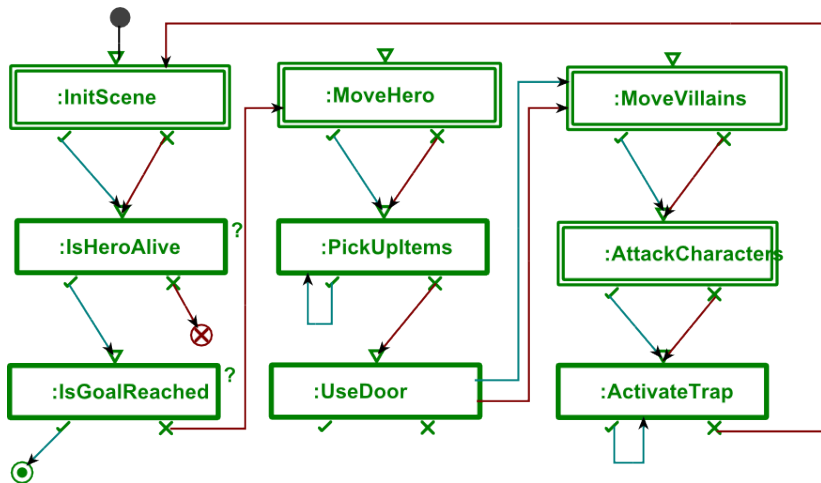
- 1 Context and problem
- 2 Background and Previous Work
- 3 Example**
- 4 DSL Combination in AToMPM
- 5 Conclusion

Role-Playing Game

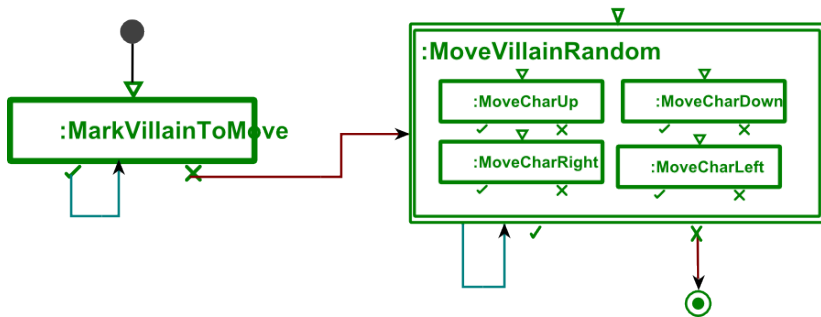
Abstract Syntax (fragment)



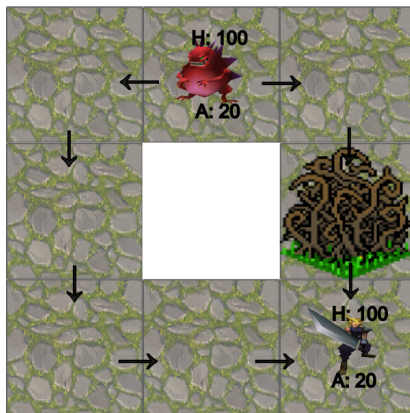
Operational Semantics



:MoveVillains



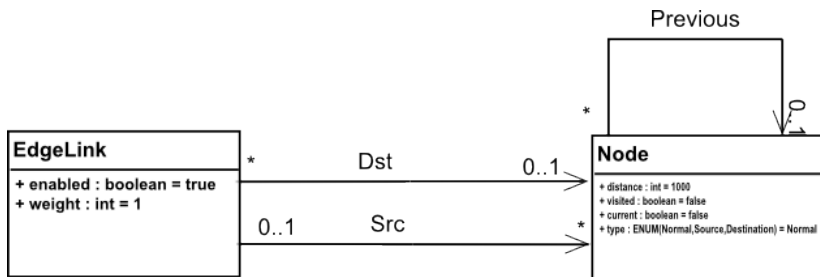
Instance model



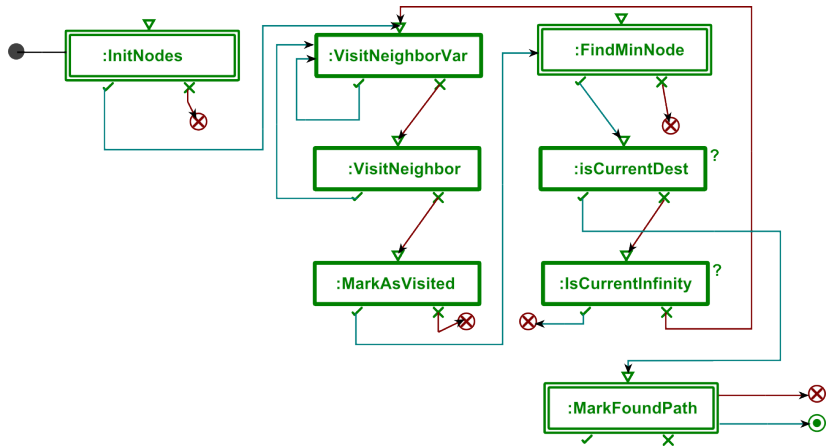
Make game *less easy*: move villain towards hero

Pathfinding

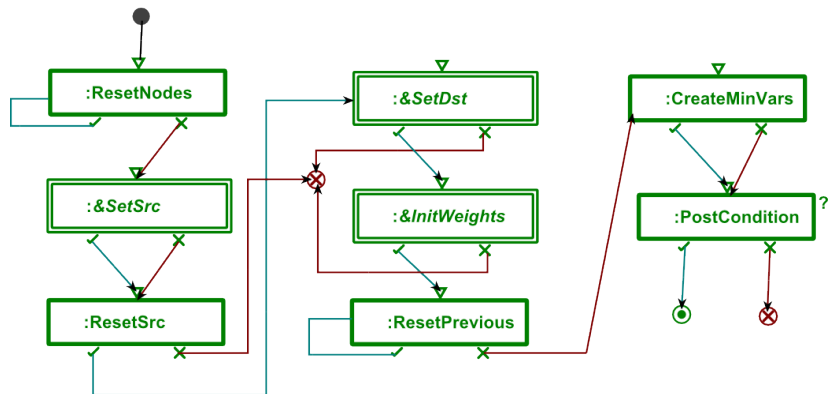
Abstract Syntax



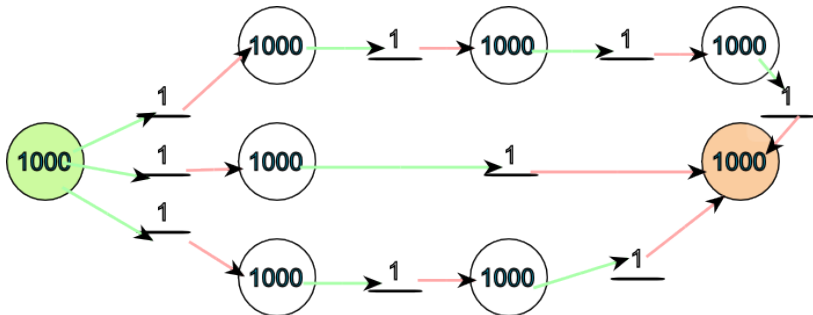
Operational Semantics



:InitNodes



Instance model



Instance model

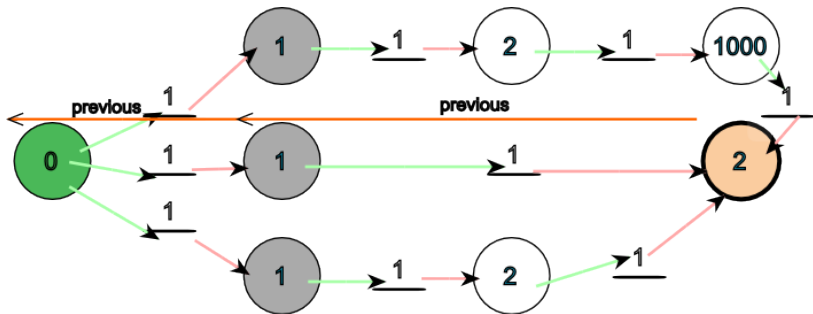


Table of Contents

- 1 Context and problem
- 2 Background and Previous Work
- 3 Example
- 4 DSL Combination in AToMPM**
- 5 Conclusion

Abstract Syntax

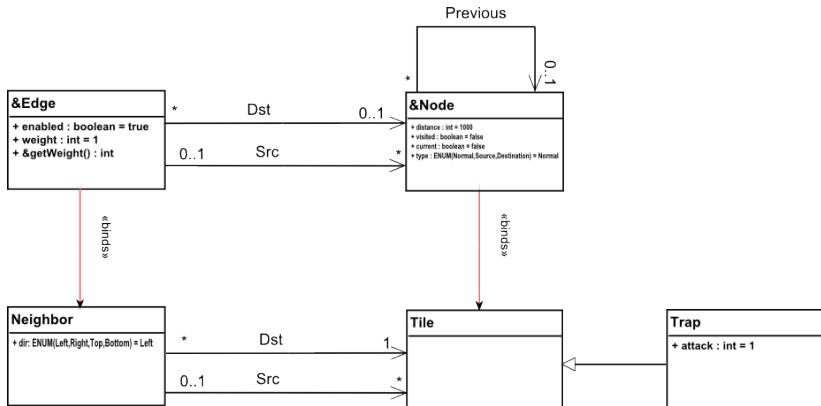
Template Instantiation

Template Instantiation¹

- Delay details of model structure
- Instantiated by different parameters
- Mechanism for reuse of patterns

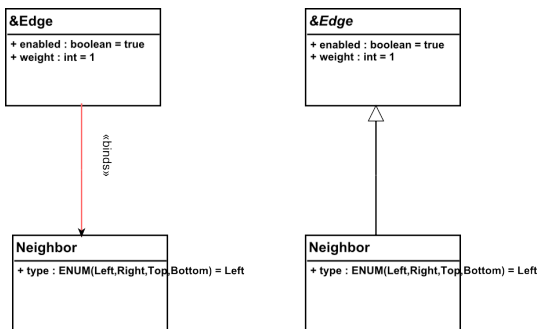
¹Matthew Emerson and Janos Sztipanovits. “Techniques for metamodel composition”. In: *OOPSLA–6th Workshop on Domain Specific Modeling*. 2006, pp. 123–139.

Binding



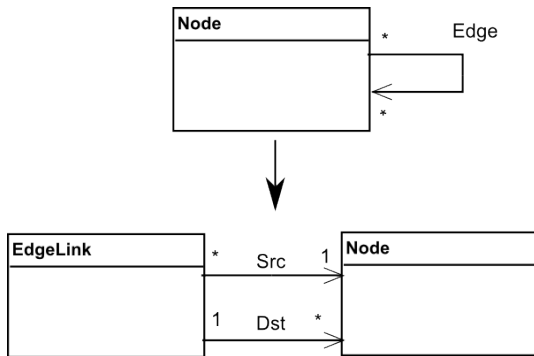
Bind Class

- TransformationRule flag *match_subtypes*
- Can't be used for its normal purpose in template Tr



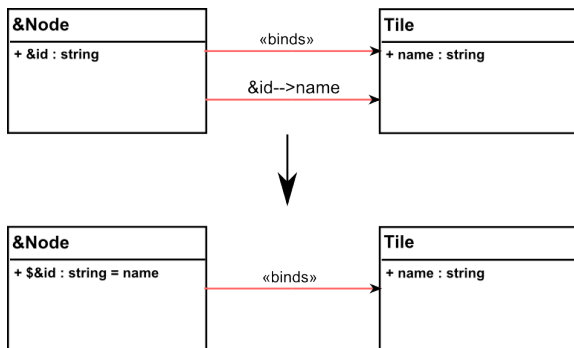
Bind Associations

- No association inheritance in AToMPM
- Convert associations to classes

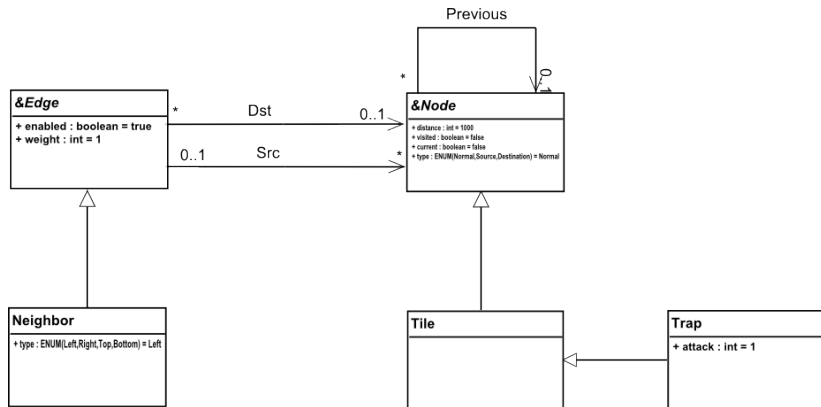


Bind Fields

- AToMPM hidden attributes \$attr
- Store name of bound field in hidden attribute
- Retrieve with API functions getAttr() from rule code



Combined abstract syntax



Concrete Syntax

Metamodel merge²

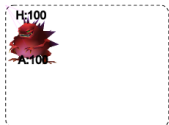
- Union operation (set theory)
- AToMPM allows duplicate names
- Template language contains abstract or auxiliary entities

²Rachel A Pottinger and Philip A Bernstein. “Merging models based on given correspondences”. In: *Proceedings of the 29th international conference on Very large data bases-Volume 29*. VLDB Endowment, 2003, pp. 862–873.

Combined concrete syntax



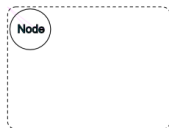
Heroicon



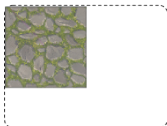
Villainicon



Trapicon



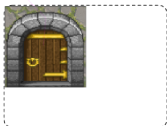
&NodeIcon



Tileicon



Obstacleicon



Dooricon



&EdgeIcon



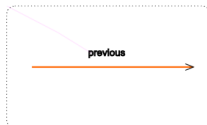
Goalicon



Weaponicon



Keyicon



PreviousLink

Semantics

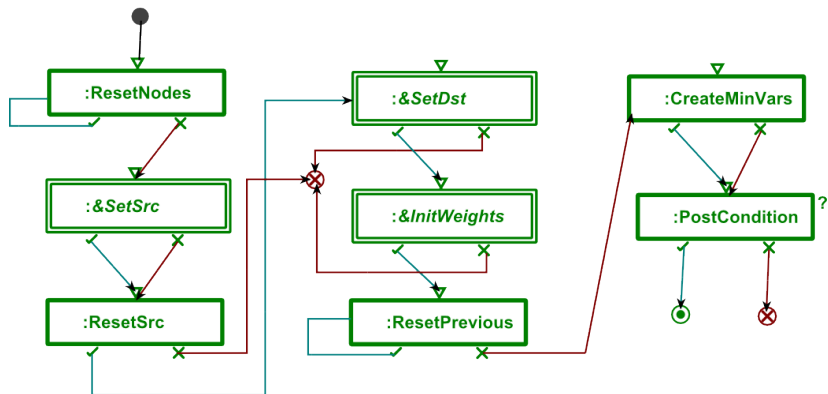
Metamodel Interfacing

- Metamodel Interfacing: model new behavior that does not belong to either language
- Hybrid Concepts use *operations* to hide structural requirements

Abstract Rules

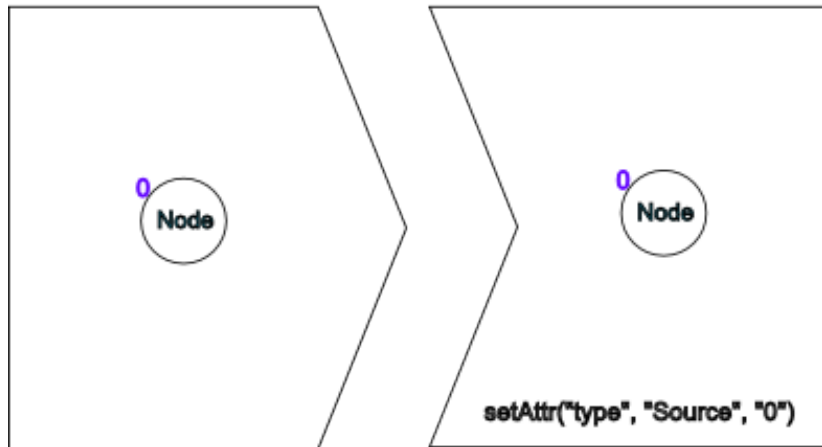
- Conform to TransformationRule language
- Default or no implementation at template design time
- Provided by the user at combination time using combined pattern MM

T_initNodes



:&SetSrc

Selects the Source node for the upcoming shortest path calculation



Implementation of `&SetSrc`

Selects the Source node for the upcoming shortest path calculation

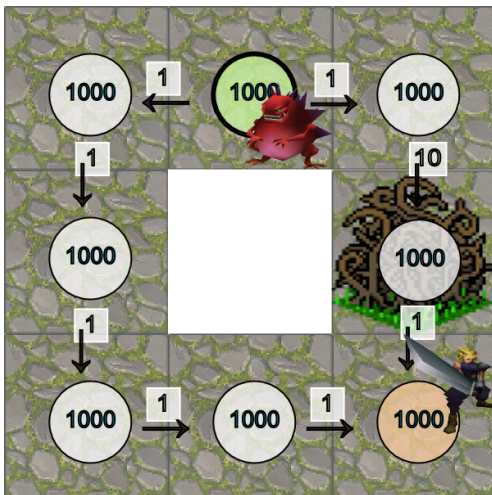


```
getAttr("name", "3") == 'Move'
```



```
setAttr("type", "Source", "1")
```


Combined instance



Transformed combined instance

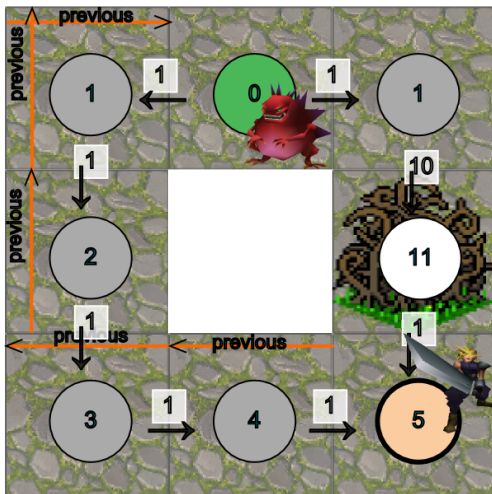


Table of Contents

- 1 Context and problem
- 2 Background and Previous Work
- 3 Example
- 4 DSL Combination in AToMPM
- 5 Conclusion**

Conclusion





- Provided tool support for combination of DSLs
- Combined all three components of a language
- Generation of Android application
- User deals with graphical models
- Stays true to maxim: *model everything*

Future Work


- More complex examples (User Input, Character AI, ...)
- Explore concrete syntax combination
- Combine static semantics (constraints)
- Composition requirements
- Generate code from concrete syntax

Demo and questions . . .

References I

-  Matthew Emerson and Janos Sztipanovits. “Techniques for metamodel composition”. In: *OOPSLA–6th Workshop on Domain Specific Modeling*. 2006, pp. 123–139.
-  Juan de Lara and Esther Guerra. “From types to type requirements: genericity for model-driven engineering”. In: *Software & Systems Modeling* 12.3 (2013), pp. 453–474.
-  Levi Lucio et al. “The Formalism Transformation Graph as a Guide to Model Driven Engineering”. In: (2012).
-  Bart Meyers et al. “Composing Textual Modelling Languages in Practice”. In: *Procs. of the Intl. Workshop on Multi-Paradigm Modeling (MPM’12)*. 2012.

References II

-  Luis Pedro, Didier Buchs, and Vasco Amaral. “Foundations for a Domain Specific Modeling Language Prototyping Environment”. In: (2008).
-  Rachel A Pottinger and Philip A Bernstein. “Merging models based on given correspondences”. In: *Proceedings of the 29th international conference on Very large data bases-Volume 29*. VLDB Endowment. 2003, pp. 862–873.
-  Rafael Ugaz. “Weaving of Domain-Specific Languages: A literature review”. In: (2014).
-  Rafael Ugaz. “Weaving of Domain-Specific Languages: Enabling technology”. In: (2014).