Part VI

THE ATL ECOSYSTEM

Some history

- 2003: Not yet ATL…
  - Jean Bézivin, et al. "The ATL transformation-based model management framework." Technical report (2003).

Some history

- 2003: First mention in the literature
  - Implemented by **Frédéric Jouault**
  - Jean Bézivin, et al. "First experiments with the ATL model transformation language: Transforming XSLT into XQuery." 2nd OOPSLA Workshop on Generative Techniques in the context of Model Driven Architecture. Vol. 37. 2003.
  - Description of the algorithm resembles current's, with some (nice) considerations:
    - Mentions checking against source/target meta-models
    - Runtime errors if binding are not resolved
    - Specific construct for resolving non-default target elements

Some history

- 2005: “the paper” with > 1000 citations
  - Frédéric Jouault, and Ivan Kurtev. "Transforming models with ATL." International Conference on Model Driven Engineering Languages and Systems. Springer Berlin Heidelberg, 2005..
  - Beware, there is a short and a long version
  - Description of the ATL algorithm
    - Mostly matches current ATL
  - ATL available as Eclipse plug-in
    - (perhaps even in 2004)

Some history

- 2005: Rapid growth
  - AMMA, KM3, AM3, AMW
  - MoDISCO
  - See: An introduction to the ATLAS Model Management Architecture

Some history

- 2006: AMMA
  - Ivan Kurtev, et al. "Model-based DSL frameworks." Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications. ACM, 2006.

Some history

- 2009: Industrialisation
  - Obeo took control
  - The language became stable

* “ATL 3.1 – Industrialization improvements”, William Piers

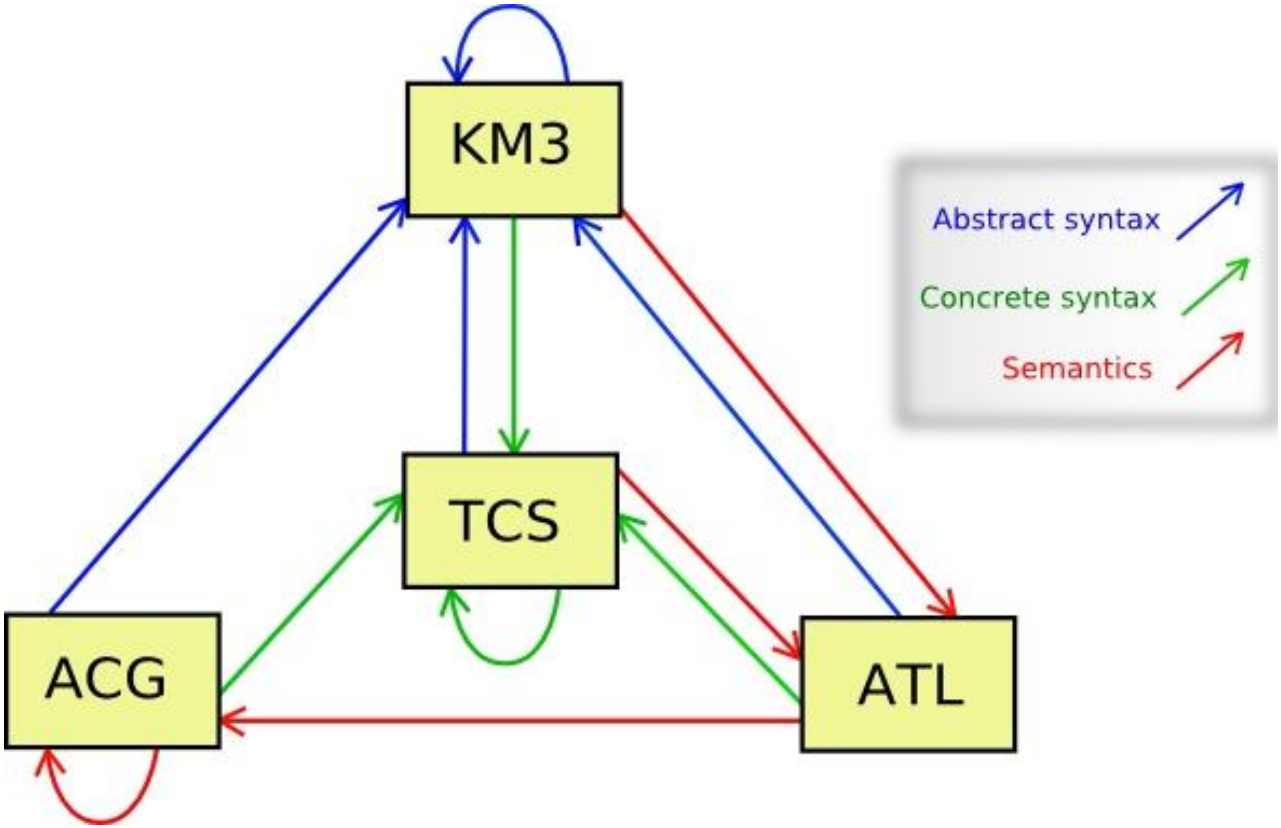
Tools around ATL

- EMFTVM
- AnATLyzer
- EMFMigrate
- MoDISCO
- AMW, AM3, KM3 seem to be dead
- Wires
- …

The ATL ecosystem

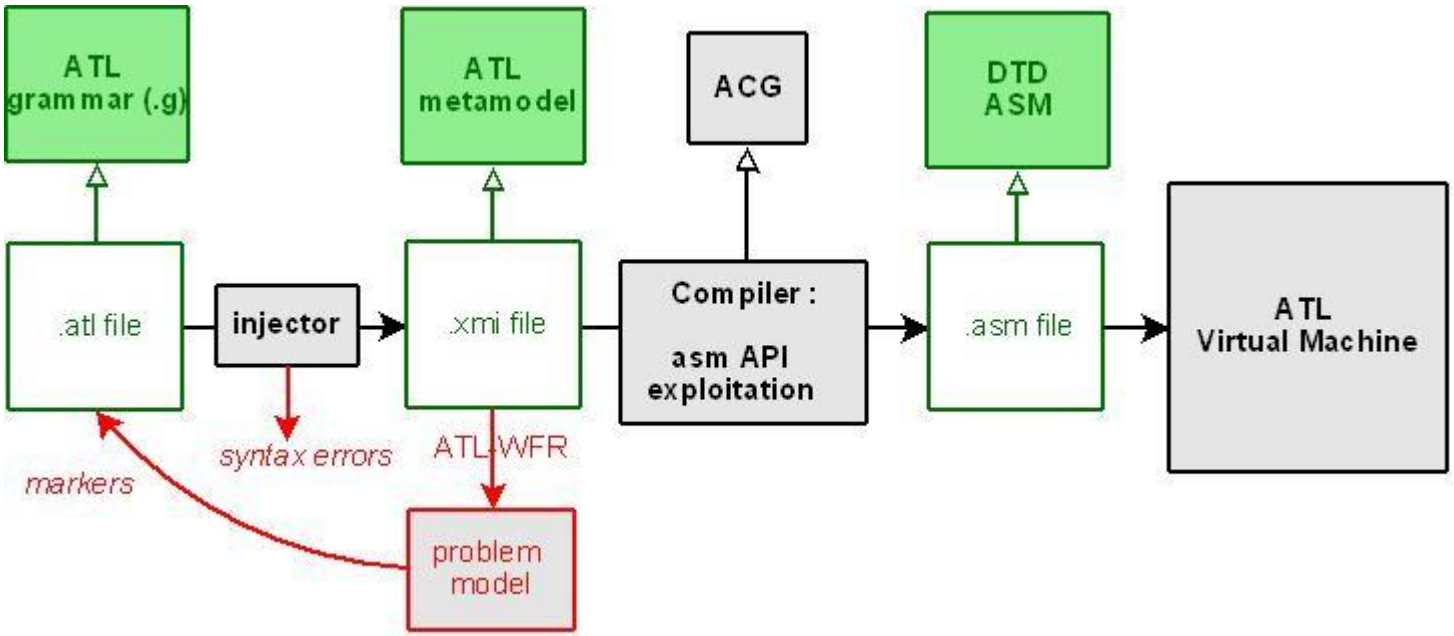
ATL architecture

Architecture



* From: https://wiki.eclipse.org/ATL/Developer_Guide

Architecture



* From: https://wiki.eclipse.org/ATL/Developer_Guide

Source code

```
git clone git://git.eclipse.org/gitroot/mmt/org.eclipse.atl.git
```

Compiler

- Written in ACG
  - Documentation
    - https://wiki.eclipse.org/ACG
    - http://www.eclipse.org/atl/documentation/old/ATL_VM_Presentation_[1.0].pdf
    - http://www.eclipse.org/atl/documentation/old/ATL_VMSpecification[v00.01].pdf

Compiler

- ATL.acg
  - This is the operational semantics of ATL
  - Complemented with some runtime libraries
- Available in “dsls/” folder of the source code

Compiler

- Semantics of bindings (excerpt)

```
code Binding {
        dup
        if(self.isAssignment) {
                push self.propertyName
                call 'J.refUnsetValue(S):J'
        }
        getasm
        analyze self.value
        call 'A.__resolve__(J):J'
        set self.propertyName
}
```

Plug-ins

- org.eclipse.m2m.atl.core
  - Generic interfaces for launching and model handling
- org.eclipse.m2m.atl.core.emf:
  - EMF implementation of the Core API
- org.eclipse.m2m.atl.dsls
  - Definition of ATL and ACG
- org.eclipse.m2m.atl.engine
  - Provides ATL compiling and parsing utilities
- org.eclipse.m2m.atl.engine.emfvm
  - Implementation of the EMFVM

Exploring Eclipse plug-ins

- File -> Import -> Import Plug-ins and Fragments
- Select “Import As” -> Projects with source folders
- Filter your plug-in ID using wildcards (*name*)
  - Use your imagination to finally determine the plug-in(s) that you need

* See: www.vogella.com/tutorials/EclipseCodeAccess/article.html
(to learn how to browse Eclipse source code)

ATL plug-ins

- org.eclipse.m2m.atl.engine.emfvm
  - Data type implementations
    - Package org.eclipse.m2m.atl.engine.emfvm.lib
  - Built-in operations
    - Class org.eclipse.m2m.atl.engine.emfvm.lib.ExecEnv
  - Bytecode processor
    - Class org.eclipse.m2m.atl.engine.emfvm.ASMOperation

ATL Compiler

- New compilers can be created
  - Extension point org.eclipse.m2m.atl.engine.atlcompiler
  - See:
    - http://git.eclipse.org/c/mmt/org.eclipse.atl.git/tree/plugins/org.eclipse.m2m.atl.engine/plugin.xml
  - Useful to create new back-ends
  - Can you implement the extension point with an existing name to have two backends??
    - E.g. to create a new output e.g., for verification purposes

The ATL ecosystem

MoDisco & EMFTVM

MoDisco

- URL
  - https://eclipse.org/MoDisco/
- Paper:
  - Bruneliere, Hugo, et al. "Modisco: A model driven reverse engineering framework." Information and Software Technology 56.8 (2014): 1012-1032.

MoDisco

- Provides facilities to extract models from Java code:
  - Abstract syntax of Java
  - UML models
  - KDM models
- Extensible

MoDisco

- Example: A transformation that extracts the ATL library from the source code

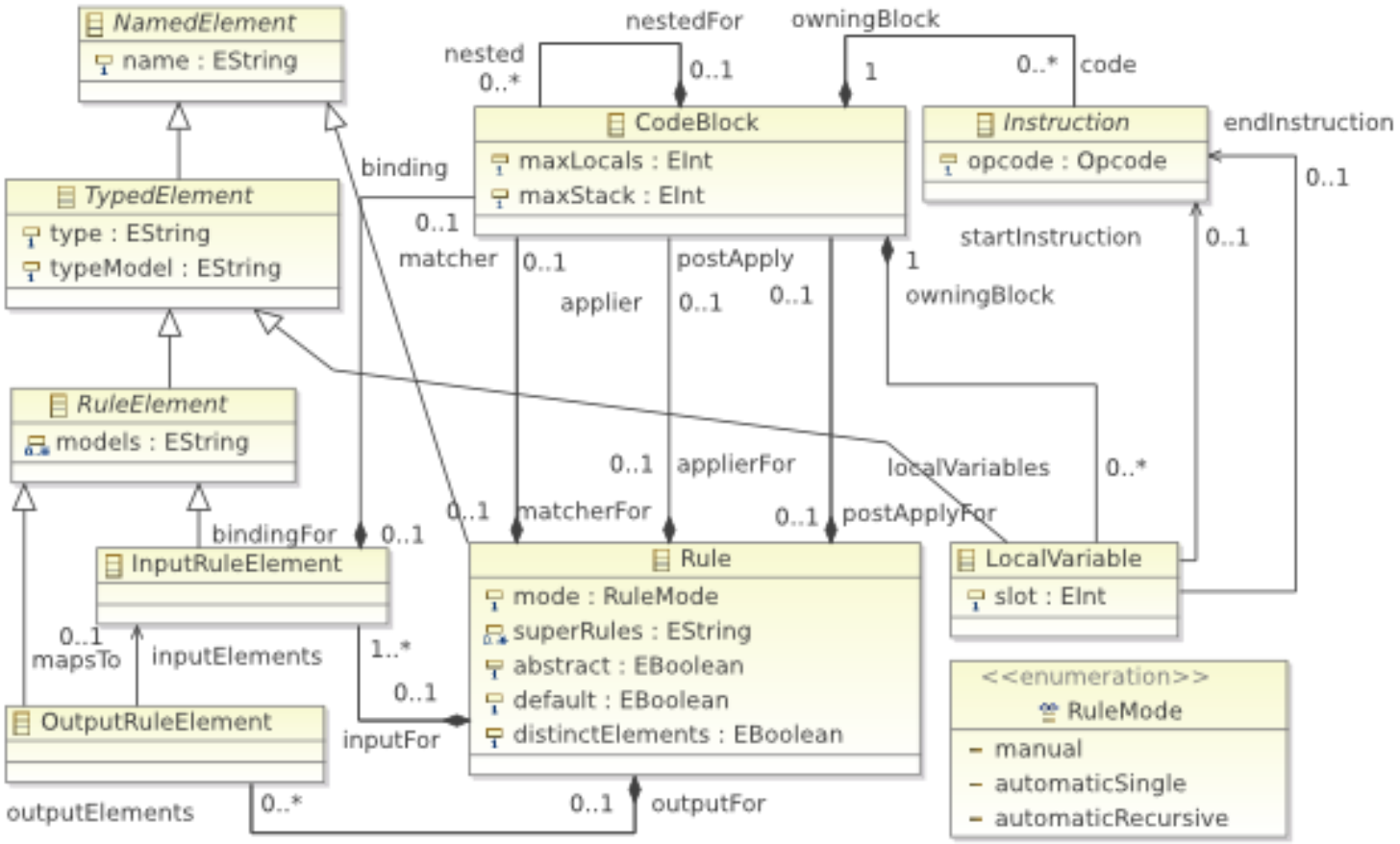
EMFTVM

- Why?
  - Created by Dennis Wagelaar to solve limitations in ATL
- Bytecode as an EMF model
  - Native notion of rule
  - Code blocks (i.e., closures)
- Not only for ATL:
  - https://github.com/dwagelaar/simplegt
  - https://github.com/dwagelaar/simpleocl

EMFTVM

- Papers:
  - Wagelaar, Dennis, et al. "Towards a general composition semantics for rule-based model transformation." International Conference on Model Driven Engineering Languages and Systems. Springer Berlin Heidelberg, 2011.
- Documentation
  - https://wiki.eclipse.org/ATL/EMFTVM

High-level structure



How to use EMFTVM

- Annotation: -- @atlcompiler emftvm
- Specific runner and a different programmatic API, and specific Ant tasks
  - emftvm.loadModel
  - emftvm.loadMetamodel
  - emftvm.run
  - emftvm.saveModel

Some differences

	ATL	EMFTVM
Module	@atl2006, @atl2010	@emftvm
Module name	Can be any	Same as the file
Helpers	No overloading, single dispatch	Overloading, multiple dispatch
Lazy rules with filter	Ignored	Pattern matched
Endpoint rules	Supported	Not supported
Refining mode	Deletion with drop	Deletion with no target pattern
Rule inheritance	Single	Multiple, Load time
Superimposition	Load time	Import, inheritance
Helper dispatch	Single	Multiple
VM	Custom	Optional JIT to JVM

Exercise

- Determine the meaning of <:= looking at the bytecode generated by EMFTVM