Syntax
Abstract vs. Concrete

F. Mallet

Frederic.Mallet@univ-cotedazur.fr

UNIVERSITÉ CÔTE D'AZUR 

UNIVERSITÉ CÔTE D'AZUR

Syntax

- Abstract
  - see. Abstract Data Types
  - Meta Model
- Concrete
  - Java has one textual concrete syntax
  - Java can have several graphical concrete syntax (scratch?)

UNIVERSITÉ CÔTE D'AZUR

XText

- Usage
  - Generate a concrete textual syntax from EMF
  - Generate Eclipse features to build IDE for DSL
- Definitions
  - EMF : Eclipse Modeling Framework
  - IDE : Integrated Development Environment
  - DSL : Domain-Specific Language

UNIVERSITÉ CÔTE D'AZUR

Xtext

- Install: on top of Eclipse Modeling
  - Help/Install new software…
  - Work with: "--All Available Sites"
  - Modeling / Xtext Complete SDK
- New Project
  - Xtext / Xtext Project From Existing Ecore Models
  - EPackages: add your EMF packages
    - (e.g., stateMachine.genmodel)
  - Pick Entry rule (e.g. Automata)
  - Pick project name, Language name, extensions
- Configure your EMF project (convert into Xtext)

UNIVERSITÉ CÔTE D'AZUR

Xtext: Important files

- Five plugins are generated
  - Modify only the main plugin / src
  - Look at src-gen and xtend-gen
- Textual syntax grammar
  - MyXXX.xtext
- Generation flow
  - GenerateMyXXX.mwe2

UNIVERSITÉ CÔTE D'AZUR

Grammar: ANTLR

- Xtext generates an ANTLR file from metamodel
  - ANTLR: https://www.antlr.org/
    - ANother Tool for Language Recognition
- Generate text parsers: rule-based
  - Some predefined terminals : STRING, ID
  - Similar to regular expressions
    - | choice ()? : optional ()* : 0 or many ()+ : one or many
- Examples

Rule name Type

```
EString returns ecore::EString:
    STRING | ID;
```

Text to recognize

UNIVERSITÉ CÔTE D'AZUR

Grammar: ANTLR

- Xtext generates an ANTLR file from metamodel
  - ANTLR: https://www.antlr.org/
    - ANother Tool for Language Recognition
- Generate text parsers: rule-based
  - Some predefined terminals : STRING, ID
  - Similar to regular expressions
    - | choice ()? : optional ()* : 0 or many ()+ : one or many
- Examples

```
State returns State:
    {State}
    'State' terminal
    name=EString; Assign field 'name' from class 'State'
                  according to rule EString
```

UNIVERSITÉ CÔTE D'AZUR

Grammar: ANTLR

- Xtext generates an ANTLR file from metamodel
- Generate text parsers: rule-based
- Examples

```
Transition returns Transition:
    'Transition'
    '{'
        'source' source=[State|EString]
        'target' target=[State|EString]
    '}';
```

Should be a reference to state
Use the id of the state !

```
State returns State:
    {State}
    'State'
    name=EString;
```

UNIVERSITÉ CÔTE D'AZUR

Grammar: ANTLR

- Xtext generates an ANTLR file from metamodel
- Generate text parsers: rule-based
  - May modify the rules manually
- Examples

```
Transition returns Transition:
        source=[State|EString]
        '->'
        target=[State|EString]
    ;
```

UNIVERSITÉ CÔTE D'AZUR

Grammar: ANTLR

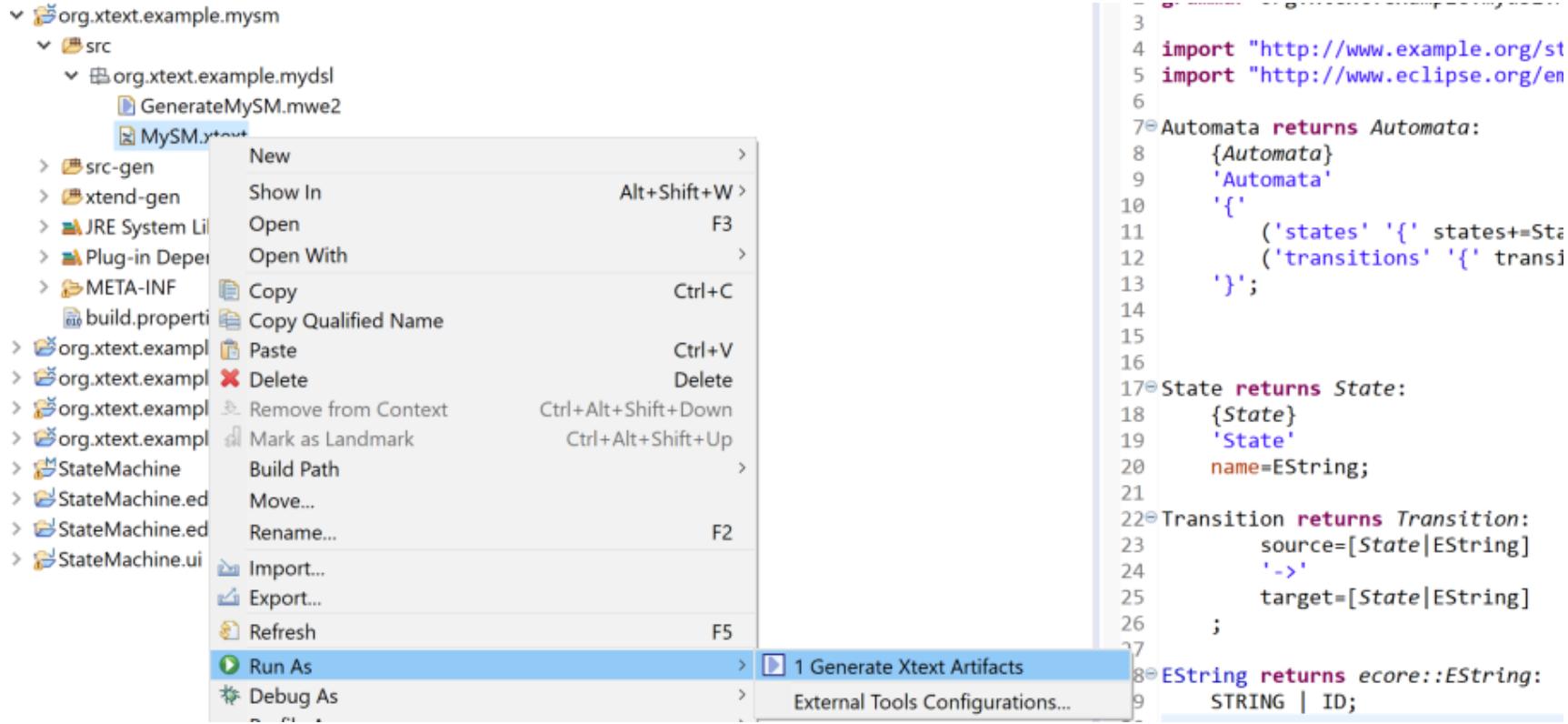
- Xtext generates an ANTLR file from metamodel
- Generate text parsers: rule-based
  - Beware ambiguous expressions: **LALR(1)**
- Examples

```
Automata returns Automata:
    {Automata}
    'Automata'
    '{'
        ('states' '{' states+=State ( "," states+=State)* '}' )?
        ('transitions' '{' transitions+=Transition ( "," transitions+=Transition)* '}' )?
    '}';
```

UNIVERSITÉ CÔTE D'AZUR

Grammar: ANTLR

- Xtext generates an ANTLR file from metamodel
- Generate text parsers: rule-based
  - Beware ambiguous expressions: **LALR(1)**
- Generate the ANTLR Parser



UNIVERSITÉ CÔTE D'AZUR

Grammar: ANTLR

- Xtext generates an ANTLR file from metamodel
- Generate text parsers: rule-based
  - Beware ambiguous expressions: **LALR(1)**
- Generate the ANTLR Parser
  - Important files (in src-gen / …..parser.antlr)
    - MyXXParser.java
  - Internal files (in src-gen / …..parser.antlr.internal)
    - InternalMyXXLexer.java
    - InternalMyXXParser.java
    - InternalMyXX.g
  - Others: (in src / ……generator)

UNIVERSITÉ CÔTE D'AZUR

Grammar: ANTLR

- Xtext generates an ANTLR file from metamodel
- Generate text parsers: rule-based
  - Beware ambiguous expressions: **LALR(1)**
- Generate the ANTLR Parser
  - Xtend generator: (in src / …..generator)
    - https://www.eclipse.org/xtend

```
override void doGenerate(Resource resource, IFileSystemAccess2 fsa, IGeneratorContext context) {
    val aut = resource.contents.get(0) as Automata;
    fsa.generateFile('essai.txt', '''
    // From My automata
    Automata {
    «FOR s : aut.states»
        State «s.name»
    «ENDFOR»
    }
    ''');
}
```

UNIVERSITÉ CÔTE D'AZUR

xTend

- See small video on https://www.eclipse.org/xtend
- Hello, world

```
public class Greeter {
    def public void greetPeople(List<String> people) {
        for (String name : people) {
            System.out.println(hello(name));
        }
    }
    def public String hello(String name) {
        return "Hello " + name + "!";
    }
}
```

UNIVERSITÉ CÔTE D'AZUR

xTend

- See small video on https://www.eclipse.org/xtend
- Hello, world

```
public class Greeter {
    def public void greetPeople(List<String> people) {
        for (String name : people) {
            System.out.println(hello(name));
        }
    }
    def public String hello(String name) {
        return "Hello " + name + "!";
    }
}
```

UNIVERSITÉ CÔTE D'AZUR

xTend

- See small video on https://www.eclipse.org/xtend
- Hello, world

```
public class Greeter {
    def greetPeople(List<String> people) {
        for (name : people) {
            println(hello(name))
        }
    }
    def hello(String name) {
        "Hello " + name + "!"
    }
}
```

UNIVERSITÉ CÔTE D'AZUR

xTend

- See small video on https://www.eclipse.org/xtend
- Hello, world

```
public class Greeter {
    def greetPeople(List<String> people) {
        people.forEach [
            println(hello)
        ]
    }
    def hello(String name) '''
        Hello «name»!
    '''
}
```