

# Java Annotation Patterns Engine


Xingyi Song and Mehmet Bakir  
13th GATE Course, Module 2

- Java Annotation Patterns Engine
  - Specially developed pattern matching language for GATE
  - JAPE allows you to recognise regular expressions in annotations on documents
  - Gazetteer lists are designed for annotating simple, regular features
    - recognising e-mail addresses using just a gazetteer would be impossible

```
Phase: University
Input: Token Lookup
Options: control = appelt
```

```
Rule: University2
(
  ({Token.string == "University"})
  {Token.string == "of"}
  ({Lookup.minorType == city}):cityName
) :orgName
-->
:cityName.Location = {kind = city},
:orgName.Organization = {kind = university}
```

# Annotations

Annotation Sets Annotations List Annotations Stack Co-reference Editor Text 

We love coming to the **University of Sheffield**. The weather is always beautiful here.

- Lookup
- Sentence
- SpaceToken
- Split
- Token
- ▼ Original markups
- paragraph
- ▼ Test
- Location
- Organization

Type	Set	Start	End	Id	Features
Organization	Test	22	45	775	{ kind=university}
Location	Test	36	45	774	{ kind=city}

# JAPE

- Each JAPE rule consists of
  - Header
  - Rule Name
  - LHS which contains patterns to match
  - RHS which describe the actions of matched patterns

```
Phase: University  
Input: Token Lookup  
Options: control = appelt
```

Header

```
Rule: University2
```

Rule Name

```
(  
  ({Token.string == "University"})  
  {Token.string == "of"}  
  ({Lookup.minorType == city}):cityName  
  ) :orgName
```

LHS Rule

```
-->  
:cityName.Location = {kind = city},  
:orgName.Organization = {kind = university}
```

RHS Rule

# JAPE Headers

- Each JAPE file must contain a set of headers at the top
- They contain Phase name, set of Input annotations and other Options
- Phase name makes up part of the Java class name for the compiled RHS actions, so it must contain alphanumeric characters and underscores only, and cannot start with a number
- Rules in the same phase compete for input
- Rules in separate phases run independently

```
Phase: University  
Input: Token Lookup  
Options: control = appelt
```

# JAPE Headers

- The Input Annotations list contains a list of all the annotation types you want to use for matching on the LHS of rules in that grammar phase
- If an annotation is listed in Input but not used in the rules, it can block the matching (e.g Split)
- If no input is included, then all annotations are used

```
Phase: University  
Input: Token Lookup  
Options: control = appelt
```

# JAPE Headers

- The matching style controls
  - Which rule gets applied
  - How much document content is 'consumed'
  - Which location to attempt matching next

```
Phase: University  
Input: Token Lookup  
Options: control = appelt
```

# LHS of the rule

- LHS is everything before the arrow
- It describes the pattern to be matched, in terms of annotations and (optionally) their features
- Each annotation is enclosed in curly brackets

```
(  
  ({Token.string == "University"})  
  {Token.string == "of"}  
  ({Lookup.minorType == city}):cityName  
) :orgName
```



# Universities

- A simple LHS rule matches Universities
  - The rule looks for specific words such as “University of” followed by the name of a city
    - To match a string of text, use the “Token” annotation and the “string” feature
      - `{Token.string == "University"}`
  - The gazetteer might contain the word “Sheffield” in the list of cities
    - The gazetteer matches will be output as “Lookup” annotation
      - `{Lookup.minorType == city}`

```
(  
  ({Token.string == "University"})  
  {Token.string == "of"}  
  ({Lookup.minorType == city}):cityName  
) :orgName
```

# LHS of the rule

- The pattern combination that you want to label is enclosed in round brackets, followed by a colon and the label
  - The label name can be any legal name you want: it's only used within the rule itself

```
(  
  ({Token.string == "University"})  
  {Token.string == "of"}  
  ({Lookup.minorType == city}):cityName  
) :orgName
```

# JAPE RHS

- The label on the RHS must match a label on the LHS

```
Phase: University  
Input: Token Lookup  
Options: control = appelt
```

```
Rule: University2  
(  
  ({Token.string == "University"})  
  {Token.string == "of"}  
  ({Lookup.minorType == city}):cityName  
) :orgName  
-->  
:cityName.Location = {kind = city},  
:orgName.Organization = {kind = university}
```

# JAPE RHS

- The label on the RHS must match a label on the LHS
- Generate an annotation with type “Location”

```
Phase: University  
Input: Token Lookup  
Options: control = appelt
```

```
Rule: University2  
(  
  ({Token.string == "University"})  
  {Token.string == "of"}  
  ({Lookup.minorType == city}):cityName  
) :orgName  
-->  
:cityName.Location = {kind = city},  
:orgName.Organization = {kind = university}
```

# JAPE RHS

- The label on the RHS must match a label on the LHS
- Generate an annotation with type “Location”
- Add feature to the new generated annotation

```
Phase: University
Input: Token Lookup
Options: control = appelt
```

```
Rule: University2
(
  ({Token.string == "University"})
  {Token.string == "of"}
  ({Lookup.minorType == city}):cityName
) :orgName
-->
:cityName.Location = {kind = city},
:orgName.Organization = {kind = university}
```

# JAPE RHS

- Features and values are optional, and you can have as many as you like
- All the following are valid:
  - :orgName.Organization = {}
  - :orgName.Organization = {kind=university}
  - :orgName.Organization = {kind=university, rule=University1}

# Hands On

- Create first JAPE application
  - Load “ANNIE” for some basic annotations
  - Right Click “Processing Resources”
    - New --> “JAPE Transducer”
    - In grammarURL select the jape file located at “JAPE/grammar/university1.jape”
  - Right Click Language Resources
    - New --> “GATE Document”
    - Select “JAPE/corpus/university1.txt”
  - Right Click the newly created document
    - Select “New corpus with this document”
  - Double click ANNIE under Applications
    - Remove ANNIE NE Transducer (by using the left hand side arrows)
    - Remove ANNIE OrthoMatcher
    - Move the JAPE PR (by using the right hand side arrows)
  - Set the corpus and run

# Hands On

- Short cut
  - Right click the “Applications”
  - Select “Restore Application from File”
  - Select “JAPE/main.xgapp”



# Operators on the LHS

- Operators include
  - | OR
  - \* zero or more occurrences
  - ? zero or one occurrence
  - + one or more occurrences
- Use round brackets to delimit the range of the operators
  - `{Lookup.minorType == city} | {Lookup.minorType == country}`
  - `{Lookup.minorType == city} | {Lookup.minorType == country}+`

# Operators on the LHS

- $(\{\text{Lookup.minorType} == \text{city}\} \mid \{\text{Lookup.minorType} == \text{country}\})^+$ 
  - One or more cities or countries in any order and combination
- $(\{\text{Lookup.minorType} == \text{city}\} \mid (\{\text{Lookup.minorType} == \text{country}\})^+)^+$ 
  - One city OR one or more countries

# Hands On

- How to find universities with name
  - University of Sheffield
  - University of Sheffield UK
  - University of UK
  - University of US
  - University of Sheffield US
  
- Short cut:
  - In JAPE\_Examples
  - Disable uni1
  - Enable uni2

# Multi-constraint statements

- Just separate the constraints with a comma
  - Equal to and
  - `{Lookup.minorType == city, Lookup.minorType1 == country}`
- Make sure that all constraints are enclosed within one pair of curly brackets
  - `{Lookup.minorType == city, Lookup.minorType1 == country} -- and`
  - `{Lookup.minorType == city} {Lookup.minorType1 == country} -- sequence`
  - `({Lookup.minorType == city} | {Lookup.minorType1 == country}) -- or`

# Hands On

- Exclude universities with country that is not from United Kingdom
  - University of US
  - University of Sheffield US
  
- Short cut (half solution):
  - In JAPE\_Examples
  - Disable uni2
  - Enable uni2and

# Negative constraints

- You can use the ! operator to indicate negation
  - {Lookup.minorType == city, Lookup.minorType1 != country}
    - Lookup.minorType is city expect Lookup.minorType1 is not country
  - {Token.orth == upperInitial, !Lookup}
    - Token.orth is upperInitial but not a Lookup annotation
  - {Lookup.minorType != country}
    - This matches ANY annotation except a Lookup whose minorType is “country”

# Hands On

- Exclude universities with country that is not from United Kingdom
  - University of US
  - University of Sheffield US
- “uni2and” solution create a false positive annotation
  - How to solve it?
- Short cut:
  - In JAPE\_Examples
  - Disable uni2and
  - Enable uni2not

# Hands On

- How to find universities with name
  - University of Sheffield
  - University of Sheffield UK
  - University of UK
  - University of Nice Weather Sheffield UK
  
- Short cut (half solution):
  - In JAPE\_Examples
  - Disable uni2not
  - Enable uni3



# Hands On

- How to find universities with name
  - University of Sheffield
  - University of Sheffield UK
  - University of UK
  - University of Nice Weather Sheffield UK
- Use “Input” to make rule aware sentence split
- Short cut (half solution):
  - In JAPE\_Examples
  - Disable uni3
  - Enable uni3Split

# Other operators

- We can also use the comparison operators `>`, `>=`, `<` and `<=`
  - `{Token.length > 3}` matches a Token annotation whose length is an integer greater than 3
- You can specify ranges when you don't know the exact number of occurrences of something
  - `({Token})[2,5]` will find between 2 and 5 consecutive Tokens

# Hands On

- How to find universities with name
  - University of Sheffield
  - University of Sheffield UK
  - University of UK
  - University of Nice Weather Sheffield UK
- 
- Short cut (full solution):
  - In JAPE\_Examples
  - Disable uni3Split
    - (uni4toklen) Solution1: use `Token.length > 3`
    - (uni4numtok) Solution2: use `Token[0, 2]`
    - (uni4) Solution 3: `Token.orth == upperInitial`

# Other operators

- You can also use `=~` and `==~` to match regular expressions
  - `{Token.string ==~ "[Dd]ogs"}` matches a Token whose string feature value is (exactly) either “dogs” or “Dogs”
  - `{Token.string =~ "[Dd]ogs"}` is the same but matches a Token whose string feature contains either “dogs” or “Dogs” within it
  - Similarly, you can use `!=~` and `!~`

# Hands On

- How to find universities with name
  - University of Sheffield
  - University of Sheffield UK
  - University of UK
  - University of Nice Weather Sheffield UK
  - Universidade de Coimbra
- “Universidade” and “University” share “Universi”
- Short cut (half solution):
  - In JAPE\_Examples
  - Disable uni4
  - Enable regex

# Contextual operators

- The contextual operators “contains” and “within” match annotations within the context of other annotations
- {Organization contains Lookup} matches if an Organization annotation completely contains a Lookup annotation.
- {Lookup within Organization} matches if a Lookup annotation lies completely within an Organization annotation
- The difference between the two is that the first annotation specified is the one matched
- In the first example, Organization is matched
- In the second example, Lookup is matched

# Combining operators

- You can combine operators of different types, e.g.
  - {Person within {Lookup.majorType == organization}}
  - {!Person within {Lookup.majorType == organization}}
  - {Person within {Lookup.majorType != organization}}
  - {Person contains {!Lookup}, Person within {Organization}}
- But be sure you know what you're doing, as it can get quite complicated!

# Hands On

- Based on regex,
  - How to find city within university?
  - How to find university contains city?
- Short cut :
  - In JAPE\_Examples
  - Keep regex
  - Enable Annotation Set Transfer
    - We need it to transfer Annotations from output to input set
  - Enable citywithin
  - Enable unicontain



# Copying Feature Values to the RHS

- JAPE provides simple support for copying feature values from the LHS to the RHS

(

```
{Lookup.majorType == location}
```

):loc

-->

```
:loc.Location = { type = :loc.Lookup.minorType}
```

- if more than one Lookup annotation is covered by the label, then one of them is chosen at random to copy the feature value from

# Copying String and length to the RHS

- JAPE provides simple support for copying feature values from the LHS to the RHS

(

{Lookup.majorType == location}

):loc

-->

:loc.Location = { string = :loc.Lookup@string, size = :loc@length}

- @string : Copy the text string of the annotation
- @length: Copy the text string length of the annotation

# Hands On

- Check regex for how to copy features

# JAPE

- Each JAPE rule consists of
  - Header
  - Rule Name
  - LHS which contains patterns to match
  - RHS which describe the actions of matched patterns

```
Phase: University
Input: Token Lookup
Options: control = appelt
```

Header

```
Rule: University2
```

Rule Name

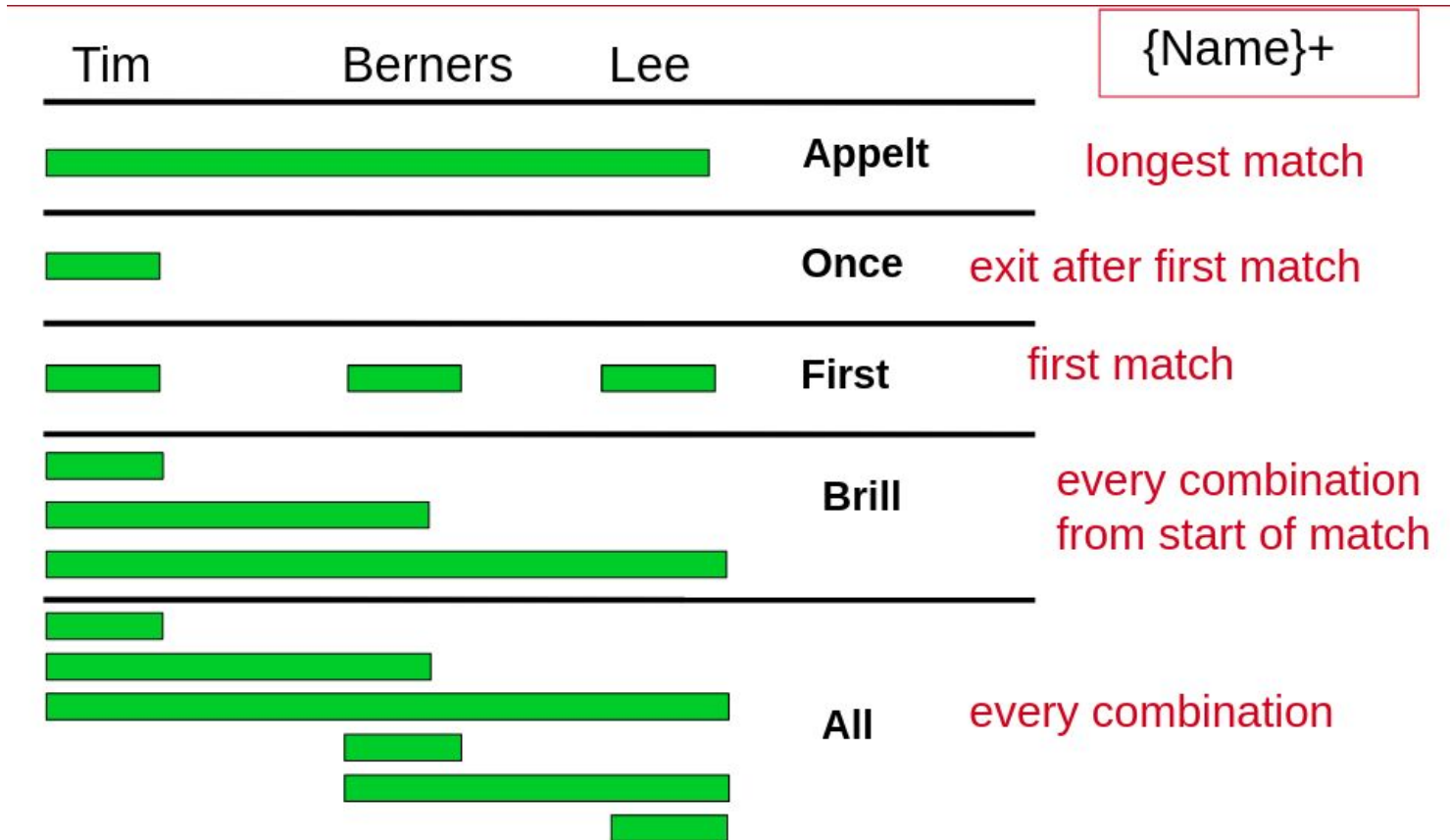
```
(
  ({Token.string == "University"})
  {Token.string == "of"}
  ({Lookup.minorType == city}):cityName
) :orgName
```

LHS Rule

```
-->
:cityName.Location = {kind = city},
:orgName.Organization = {kind = university}
```

RHS Rule

# Control



# Hands On

- Based on regex,
  - How to find city within university?
  - How to find university contains city?
  -
- What if we put `citywithin` and `unicontain` into one file?
- Short cut :
  - In JAPE\_Examples
  - Disable citywithin
  - Disable unicontain
  - Enable control\_appelt

# Hands On

- No city annotation anymore
  - Appelt find the longest match and stop
- Recommend put rules for different tasks into different files
- We can change control to all
- Short cut :
  - Disable control\_appelt
  - Enable control\_all

# Appelt style

- In the appelt style, which rule to apply is selected in the following order:
  - longest match
  - explicit priority
    - Higher numbers have greater priority
    - If no explicit priority parameter, default value is -1
  - rule defined first
  - Once a match has fired, matching continues from the next offset following the end of the match

```
Rule:    Location1  
Priority: 25
```



# Hands On

- Example of rule defined first matches
- Short cut :
  - In JAPE\_Examples
  - Disable control\_all
  - Enable control\_first
  - (Create University1)
- Example of rule with priority
- Short cut :
  - In JAPE\_Examples
  - Disable control\_first
  - Enable control\_priority
  - (Create University2)

# Multiphase

- The multiphase transducer lists the other JAPE to be loaded
  - all you need to load is this file
- For example if we want load both unicontain and citywithin
  - Short cut example can be found in “multiphase”

**MultiPhase:** Example

**Phases:**

unicontain

citywithin

# Using a macro in a rule

- Macros provide an easy way to reuse long or complex patterns
- The macro is specified once at the beginning of the grammar, and can then be reused by simply referring to its name, in all future rules

```
Macro: ofENDE  
(  
  {Token.string == "of"} | {Token.string == "de"}  
)
```

- Short cut example can be found in “macro”

# Using a macro in a rule

Phase: University  
Input: Token Lookup Split  
Options: control = appelt

```
Macro: ofENDE  
{  
  {Token.string == "of"} | {Token.string == "de"}  
}
```

Rule: UniversityMacro

```
{  
  {Token.string == "Universi"}  
  (ofENDE)  
  {{Token.orth == upperInitial}}*  
  {{Lookup.minorType == city} | {Lookup.minorType == country, Lookup.normalized == "United Kingdom"} | {Lookup.minorType == province}}:loc  
}:orgName  
{  
  {!Lookup.minorType == country}  
}  
-->  
:loc.Location = {kind = :loc.Lookup.minorType},  
:orgName.Organization = {kind = university, uniName=:orgName@string, uniNameLen=:orgName@length}
```

# Advanced JAPE

- Include Java in RHS JAPE
  - Delete annotations from the input
  - More complex feature
  - Collect statistics
  
- Don't worry if you are not a (Java) developer
  - This section will show you a number of 'recipes' which you can edit slightly for specific task

# Advanced JAPE

- The pre-defined variables available to Java RHS blocks are:
  - `doc` The document currently being processed (`String`).
  - `inputAS` The `AnnotationSet` specified by the `inputASName` runtime parameter to the JAPE
  - `outputAS` The `AnnotationSet` specified by the `outputASName` runtime parameter to the JAPE
  - `bindings` The bindings map `AnnotationSet`.

# Advanced JAPE

- Main Document API Calls
  - Obtain the document content
    - `DocumentContent getContent();`
  - Get the default annotation set
    - `AnnotationSet getAnnotations();`
  - Get the names for the annotation sets
    - `AnnotationSet getAnnotations(String name);`

# Hands On

- Print document content to Message tab
  - `String docContent = doc.getContent().toString();`
  - `System.out.println(docContent);`
  
- Short cut:
  - Disable regex
  - Disable Annotation Set Transfer
  - Disable multiphase and macro
  - Enable java1



# Advanced JAPE

- Main Document API Calls
  - Obtain the document content
    - `DocumentContent getContent();`
  - Get the default annotation set
    - `AnnotationSet getAnnotations();`
  - Get the names for the annotation sets
    - `AnnotationSet getAnnotations(String name);`

# Advanced JAPE

- Main AnnotationSet API Calls
  - Get annotation by ID
    - Annotation get(Integer id);
  - Get all annotations of one type
    - AnnotationSet getAnnotations(String type)
  - Get all annotations starting at a given location, or right after.
    - AnnotationSet getAnnotations(Long offset)
  - Get all annotations that overlap an interval
    - AnnotationSet getAnnotations(Long startOffset, Long endOffset)
- Combined get methods
  - Get by type and feature constraints
    - AnnotationSet getAnnotations(String type, FeatureMap constraints)

# GATE Annotations

- Have a start and an end Node (gate.Node);
- Main Annotation methods:
  - Get its type
    - `String getType();`
  - Get start node
    - `Node getStartNode();`
  - Get end node
    - `Node getEndNode();`
  - Get Feature
    - `FeatureMap getFeatures();`
- Main Node methods
  - Get offset
    - `Long getOffset();`

# Hands On

- Print Annotations sets
  - `AnnotationSet TestAnnoInDoc = doc.getAnnotations("Test");`
  - `System.out.println(TestAnnoInDoc);`
  - `System.out.println(outputAS);`
  
- Short cut:
  - Disable java1
  - Enable java2

# Hands On

- Remove city and country Lookup annotation from input set if they are overlap with “new” university annotation
- **and** Create new Lookup annotation with
  - Original features
  - Put a new feature “haveUni” with value “name of the university”
- Short cut:
  - Disable java2
  - Enable removeuni

# More About JAPE JAVA

- If you want know more about GATE JAVA API
  - Module 8: GATE Embedded
  - JAPE Document: <https://gate.ac.uk/userguide/chap:jape>
  - JAVA Document: <https://jenkins.gate.ac.uk/job/gate-core/javadoc/index.html>