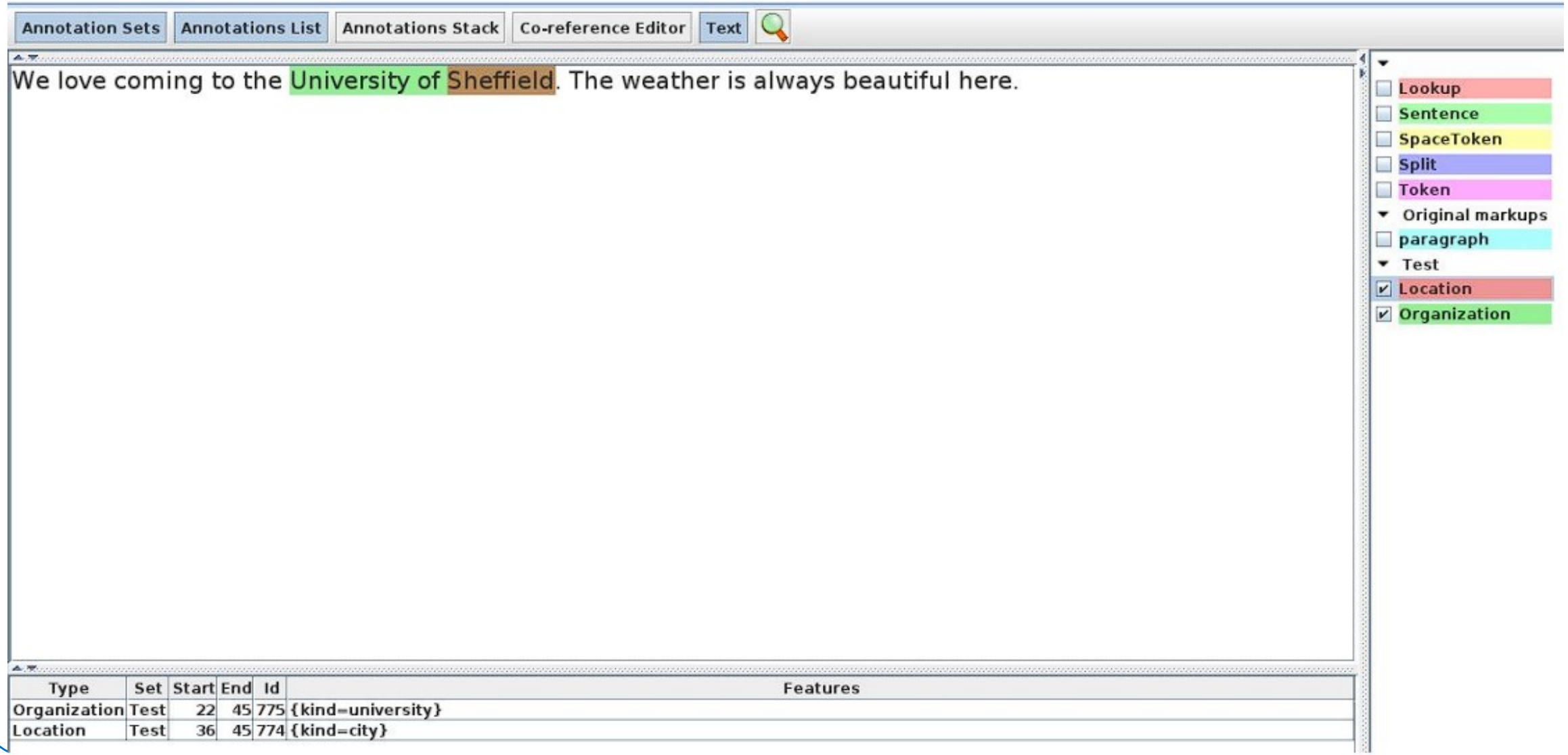
JAPE : Java Annotation Patterns Engine

- Moteur de patterns d'annotation Java
- Langage de filtrage spécialement développé pour GATE
- JAPE permet de reconnaître **les expressions régulières** dans les annotations sur les documents
- Les listes Gazetteer sont conçues pour l'annotation de caractéristiques simples et régulières
- La reconnaissance d'adresses électroniques à l'aide d'une simple liste de gazette serait 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



JAPE

Chaque règle JAPE se compose de

- En-tête (Header)
- Nom de la règle
- LHS qui contient les patterns à faire correspondre
- RHS qui décrit les actions des patterns correspondants

```
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

Les entêtes de JAPE (Headers)

Chaque fichier JAPE doit contenir une série d'en-têtes en haut de page

- Ils contiennent le nom de la phase, l'ensemble des annotations d'entrées et d'autres options.
- Le nom de la phase fait partie du nom de la classe Java pour les actions RHS compilées, il doit donc contenir uniquement des caractères alphanumériques et des traits de soulignement et ne peut pas commencer par un chiffre.
- Les règles de phases distinctes s'exécutent indépendamment les unes des autres

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

Les entêtes de JAPE (Headers)

- La liste Annotations en entrée contient une liste de tous les types d'annotations que vous souhaitez utiliser pour la mise en correspondance par le LHS des règles.
- Si une annotation figure dans la liste Input mais n'est pas utilisée dans les règles, elle peut bloquer la mise en correspondance (par exemple Split).
- Si aucune entrée n'est incluse, toutes les annotations sont utilisées.

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

Les entêtes de JAPE (Headers)

Le style de correspondance contrôle

- Quelle règle est appliquée
- la quantité de contenu du document qui est ‘consommée’
- L'emplacement de la prochaine tentative de mise en correspondance

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

LHS de la règle

LHS est tout ce qui précède la flèche

- Il décrit les patterns à mettre en correspondance, en termes d'annotations et de (éventuellement) de leurs caractéristiques
- Chaque annotation est entourée de parenthèses.

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

LHS de la règle

Une règle LHS simple permet de faire ressortir les universités

- La règle recherche des mots spécifiques tels que « University of » suivi du nom d'une ville
- Pour faire correspondre une chaîne de texte, utiliser l'annotation "Token" et la fonction "string".
  {Token.string == "University"}
- Le PR gazzetter peut contenir le mot "Sheffield" dans la liste des villes.
- Ses correspondances seront affichées sous la forme d'une annotation "Lookup".
  {Lookup.minorType == city}

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


LHS de la règle

- La combinaison de patterns que vous souhaitez étiqueter est placée entre crochets, suivie de deux points et de l'étiquette
- Le nom de l'étiquette peut être n'importe quel nom légal : il n'est utilisé qu'à l'intérieur de la règle elle-même.

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

RHS de la règle

- L'étiquette RHS doit correspondre à une étiquette 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}
```

RHS de la règle

- L'étiquette sur le RHS doit correspondre à une étiquette sur le LHS.
- Générer une annotation de type « Location ».
- Ajouter une caractéristique à la nouvelle annotation générée.

```
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}
```

RHS de la règle

- Les caractéristiques et les valeurs sont facultatives, et vous pouvez en avoir autant que vous le souhaitez.
- Tous les éléments suivants sont valables :

  - :orgName.Organization = {}
  - :orgName.Organization = {kind=university}
  - :orgName.Organization = {kind=university, rule=University1}

Lab 1 : Créer la première application JAPE

- ❑ Charger "ANNIE"
- ❑ Cliquer avec le bouton droit de la souris sur "Processing Resources" (ressources de traitement)
  - ■ Nouveau --> "JAPE Transducer"
  - ■ Dans grammarURL, sélectionner le fichier JAPE situé à "JAPE/grammar/university1.jape".
- ❑ Cliquer avec le bouton droit de la souris sur ‘Language Resources’
  - ■ Nouveau --> "Document GATE"
  - ■ Sélectionner **"JAPE/corpus/university1.txt"**
- ❑ Cliquer avec le bouton droit de la souris sur le document nouvellement créé
  - ■ Sélectionner "Nouveau corpus avec ce document«
- ❑ Double-cliquer sur ANNIE sous Applications
  - ■ Supprimer ANNIE NE Transducer (en utilisant les flèches de gauche)
  - ■ Supprimer ANNIE OrthoMatcher
  - ■ Déplacer le PR JAPE (en utilisant les flèches du côté droit)
- ❑ Définir le corpus et exécuter
- ❑ Analyser les résultats

Opérateurs du LHS

- Les opérateurs comprennent
  - | OR
  - * zéro ou plusieurs occurrences
  - ? zéro ou une occurrence
  - + une ou plusieurs occurrences

- Utiliser des parenthèses pour délimiter la plage des opérateurs

({Lookup.minorType == city} | {Lookup.minorType == country})
({Lookup.minorType == city} | {Lookup.minorType == country})+

- **({Lookup.minorType == city} | {Lookup.minorType == country})+**
  → Une ou plusieurs villes ou pays dans n'importe quel ordre et n'importe quelle combinaison

- **({Lookup.minorType == city} | ({Lookup.minorType == country})+)**
  → Une ville OU un ou plusieurs pays

Lab 2 : Opérateurs du LHS

- **Comment trouver des universités avec leur nom**
  - **University of Sheffield**
  - **University of Sheffield UK**
  - **University of UK**
  - **University of US**
  - **University of Sheffield US**

Lab 2 : Opérateurs du LHS -Solution

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

Résultat


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

Déclarations multi-contraintes (Multi-constraint statements)

- Il suffit de séparer les contraintes par une virgule → Égal à et

{Lookup.minorType == city, Lookup.minorType == country}

- Veiller à ce que toutes les contraintes soient enfermées dans une seule accolade.
  - **{Lookup.minorType == city, Lookup.minorType == country} --** and
  - **{Lookup.minorType == city} {Lookup.minorType == country}** -- sequence
  - **({Lookup.minorType == city} | {Lookup.minorType == country})** -- or

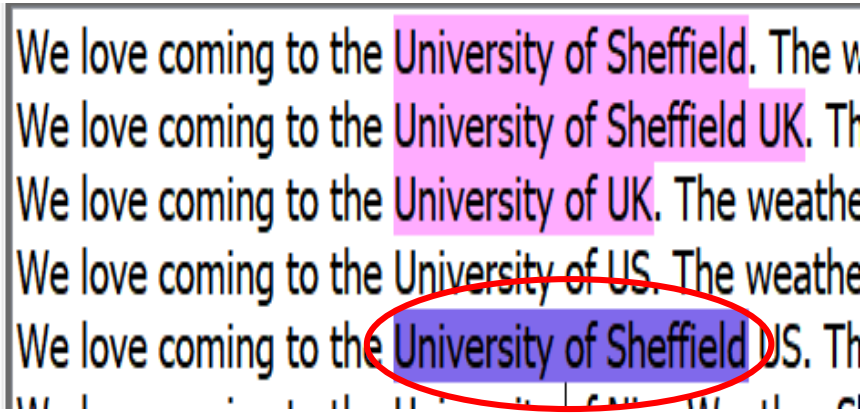
Lab 3 : Déclarations multi-contraintes (Multi-constraint statements)

- **Exclure les universités dont le pays n'est pas le Royaume-Uni.**
  - **University of US**
  - **University of Sheffield US**

Lab 3 : Déclarations multi-contraintes - Solution

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

Résultat




```
Rule: University2and
(
 ({Token.string == "University"})
 {Token.string == "of"}
 ({Lookup.minorType == city} | {Lookup.minorType == country,
Lookup.name == "The United Kingdom of Great Britain and Northern
Ireland"})+:cityName
) :orgName
-->
:cityName.Location = {kind = city},
:orgName.Organization = {kind = university}
```

Contraintes négatives

Utiliser l'opérateur ! pour indiquer la négation

- {Lookup.minorType == city, Lookup.minorType != country}
  - 'Lookup.minorType' est une ville, mais 'Lookup.minorType' n'est pas un pays.

- {Token.orth == upperInitial, !Lookup}
  - Token.orth est upperInitial mais n'est pas une annotation Lookup

- {Lookup.minorType != country}
  - Cela correspond à N'IMPORTE QUELLE annotation, à l'exception d'une recherche dont le type mineur est "country"

Lab 4 : Contraintes négatives

Exclure les universités dont le pays n'est pas le Royaume-Uni

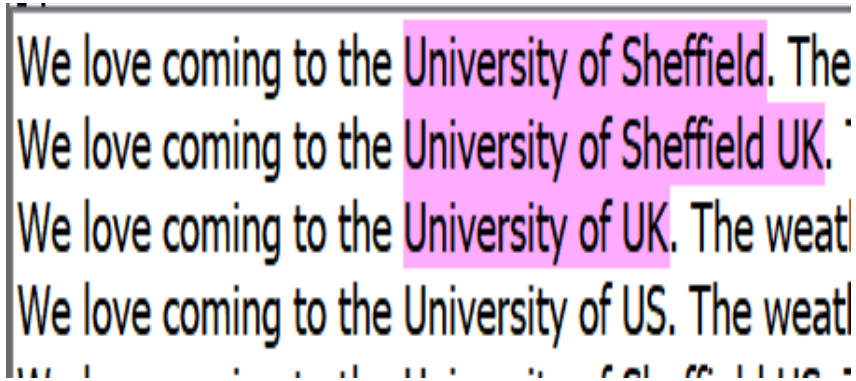
- **University of US**
- **University of Sheffield US**

- **La solution « Lab3" crée une annotation faussement positive. Comment résoudre ce problème ?**

Lab 4 : Contraintes négatives - Solution

```
Rule: University2not
(
 ({Token.string == "University"})
 {Token.string == "of"}
 ({Lookup.minorType == city} | {Lookup.minorType == country,
Lookup.name == "The United Kingdom of Great Britain and Northern
Ireland"})+:cityName
) :orgName
(
 ({!Lookup.minorType == country})
)
-->
:cityName.Location = {kind = city},
:orgName.Organization = {kind = university}
```

Résultat



Lab 5 : Contraintes négatives

- **Comment trouver des universités avec les noms :**
  - **University of Sheffield**
  - **University of Sheffield UK**
  - **University of UK**
  - **University of Nice Weather Sheffield UK**

Lab 5 : Contraintes négatives – Solution 1

```
Rule: University3
(
 ({Token.string == "University"})
 {Token.string == "of"}
 ({Token})*
 ({Lookup.minorType == city} | {Lookup.minorType == country, Lookup.name == "The United Kingdom of Great Britain and Northern Ireland"})+:cityName
) :orgName
(
 ({!Lookup.minorType == country})
)
```

Résultat

We love coming to the University of Sheffield. The weather is always beautiful here.
We love coming to the University of Sheffield UK. The weather is always beautiful here.
We love coming to the University of UK. The weather is always beautiful here.
We love coming to the University of US. The weather is always beautiful here.
We love coming to the University of Sheffield US. The weather is always beautiful here.
We love coming to the University of Nice Weather Sheffield UK. The weather is always beautiful here.
Sheffield University of cause always have a beautiful Sheffield weather.
We love coming to the Universidade de Coimbra. The weather is always beautiful here.

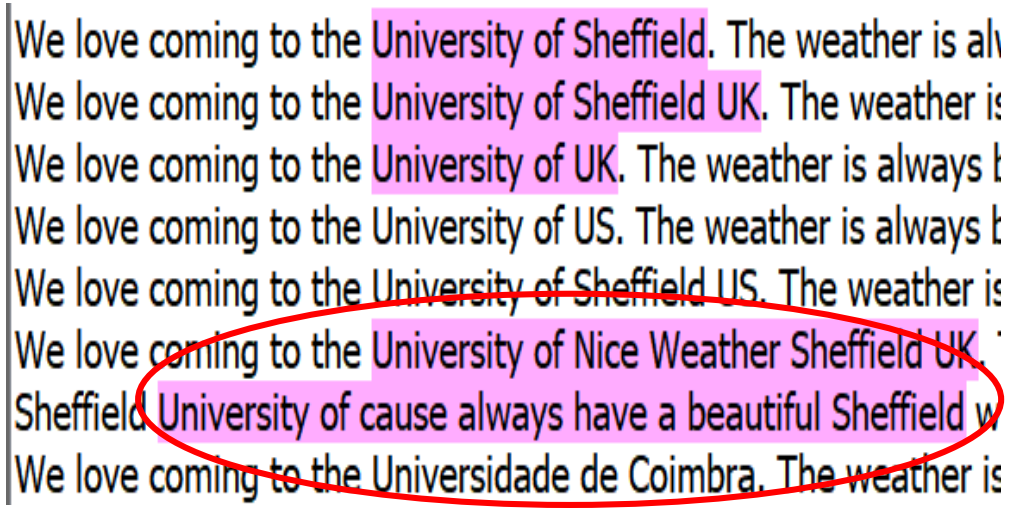
Lab 6 : Contraintes négatives – Solution 2

Utiliser 'Input' pour faire un découpage de phrase. → Si une annotation figure dans la liste Input mais n'est pas utilisée dans les règles, elle peut bloquer la mise en correspondance (par exemple Split).

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

```
Rule: University3
(
 ({Token.string == "University"})
 {Token.string == "of"}
 ({Token})*
 ({Lookup.minorType == city} | {Lookup.minorType == country, Lookup.name == "The United Kingdom of Great Britain and Northern Ireland"})+:cityName
) :orgName
(
 ({!Lookup.minorType == country})
)
```

Résultat



Autres opérateurs

- Nous pouvons également utiliser les opérateurs de comparaison >, >=. < et <=
  - {Token.length > 3} correspond à une annotation Token dont la longueur est un entier supérieur à 3.

- Nous pouvons spécifier des plages lorsque nous ne connaissons pas le nombre exact d'occurrences de quelque chose.
  - ({Token})[2,5] trouvera entre 2 et 5 tokens consécutifs

Lab 7 : Autres opérateurs

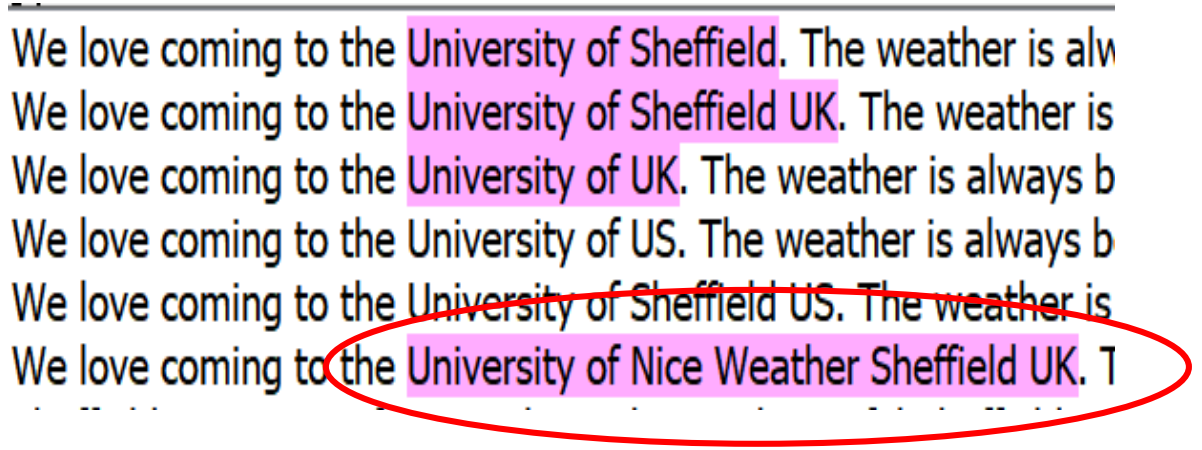
- **Comment trouver des universités avec un nom**
  - **University of Sheffield**
  - **University of Sheffield UK**
  - **University of UK**
  - **University of Nice Weather Sheffield UK**

- **(uni4toklen) Solution1: utiliser Token.length > 3**
- **(uni4numtok) Solution2: utiliser Token[0, 2]**
- **(uni4) Solution 3: Token.orth == upperInitial**

Lab7 : Autres opérateurs- Solution

```
Rule: University4
(
 ({Token.string == "University"})
 {Token.string == "of"}
 ({Token.orth == upperInitial})*
 ({Lookup.minorType == city} | {Lookup.minorType == country, Lookup.name == "The United Kingdom of Great Britain and Northern Ireland"})+:cityName
) :orgName
(
 ({!Lookup.minorType == country})
)
-->
```

Résultat



Autres opérateurs

- C’est possible d’utiliser =~ et ==~ pour faire correspondre des expressions régulières.

  - {Token.string ==~ "[Dd]ogs"} correspond à un token dont la valeur de l'élément de chaîne est (exactement) soit "dogs" ou ‘Dogs’

  - {Token.string =~ "[Dd]ogs"} est identique mais correspond à un token dont la valeur de la chaîne contient soit "dogs" soit "Dogs"

  - De même, on peut utiliser !=~ et !~

Autres opérateurs : Copie des valeurs des caractéristiques dans le RHS

● JAPE fournit un support simple pour copier les valeurs des caractéristiques de LHS à RHS.

```
(
 {Lookup.majorType == location}
):loc
-->
:loc.Location = { type = :loc.Lookup.minorType}

(
{Lookup.majorType == location}
):loc
-->
:loc.Location = { string = :loc.Lookup@string, size = :loc@length}
```

● Si plus d'une annotation Lookup est couverte par l'étiquette, alors l'une d'entre elles est choisie au hasard pour copier la valeur de la caractéristique.

Lab 8 : Autres opérateurs

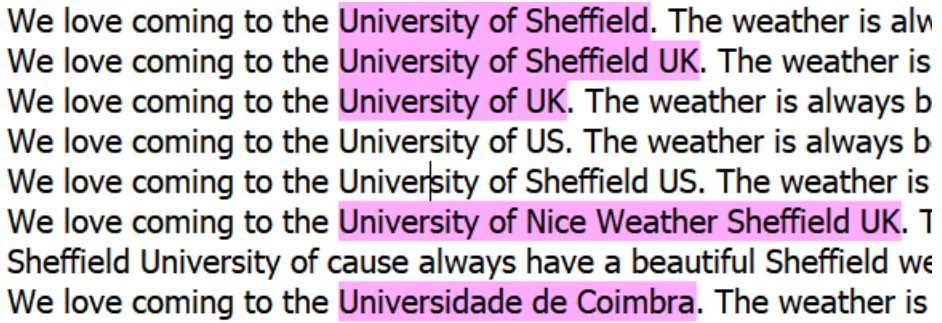
- **Comment trouver des universités avec un nom**
  - **University of Sheffield**
  - **University of Sheffield UK**
  - **University of UK**
  - **University of Nice Weather Sheffield UK**
  - **Universidade de Coimbra**

“Universidade” et “University” partagent “Universi”

Lab 8 : Autres opérateurs- Solution

```
Rule: University5
(
 ({Token.string =~ "Universi"})
 ({Token.string == "of"} | {Token.string == "de"})
 ({Token.orth == upperInitial})*
 ({Lookup.minorType == city} | {Lookup.minorType == country, Lookup.name == "The United Kingdom of
Great Britain and Northern Ireland"} | {Lookup.minorType == province}):loc
):orgName
(
 ({!Lookup.minorType == country})
)
-->
:loc.Location = {kind = :loc.Lookup.minorType},
:orgName.Organization = {kind = university, uniName=:orgName@string, uniNameLen=:orgName@length}
```

Résultat



Opérateurs contextuels

- Les opérateurs contextuels "contains" et "within" font correspondre les annotations dans le contexte d'autres annotations.

- **{Organization contains Lookup}** correspond si une annotation Organization contient complètement une annotation Lookup.

- **{Lookup within Organization}** correspond si une annotation Lookup se trouve complètement dans une annotation Organization.

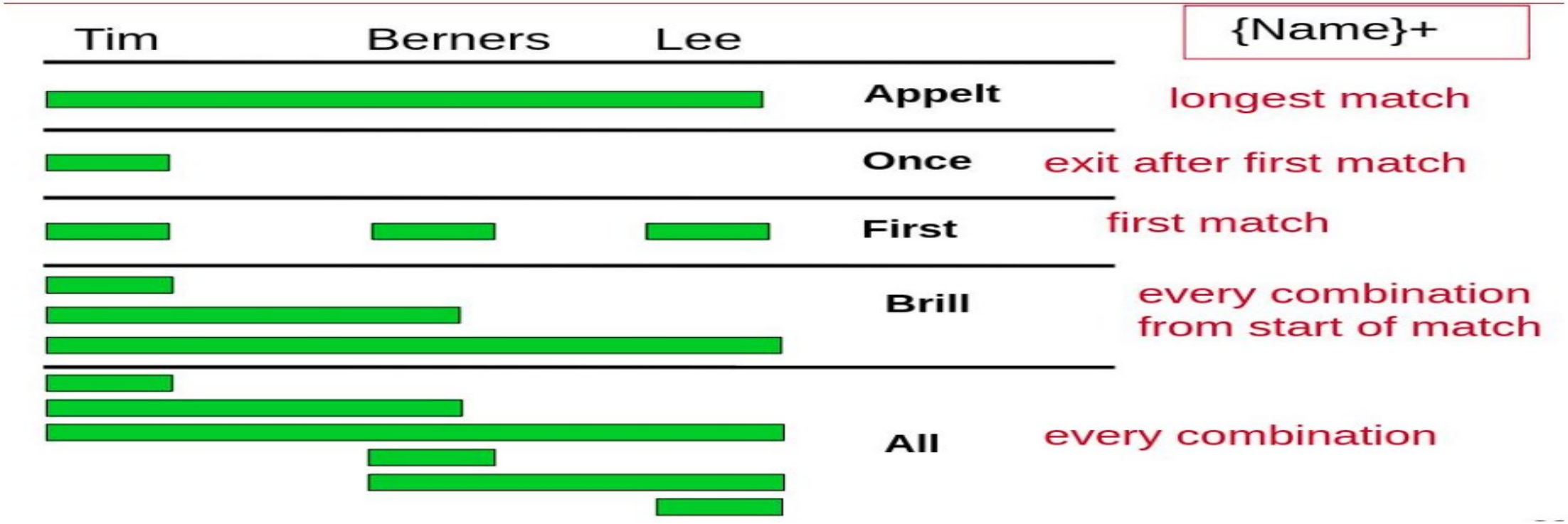
- La différence entre les deux est que la première annotation spécifiée est celle qui correspond (the first annotation specified is the one matched).
- Dans le premier exemple, c'est l'annotation 'organization' qui est prise en compte.
- Dans le deuxième exemple, c'est l'annotation lookup qui est prise en compte.

Combiner différents operateurs

- C’est possible de combiner des opérateurs de types différents, par exemple
  - {Person within {Lookup.majorType== organization}}
  - {!Person within {Lookup.majorType == organization}}
  - {Person within {Lookup.majorType != organization}}
  - {Person contains {!Lookup}, Person within {Organization}}

Contrôles

- Chaque grammaire possède l'un des cinq styles de contrôle possibles : "brill", "all", "first", "once" et "appelt". Ce style est spécifié au début de la grammaire. Si aucun style de contrôle n'est spécifié, la valeur par défaut est brill, mais il est recommandé de toujours spécifier un style de contrôle par souci de clarté.



Lab 9: Combiner plusieurs règles

- **Comment trouver les universités qui contiennent un nom de ville ?**
- **Comment trouver les villes dans les noms des universités ?**

Lab 9: Combiner plusieurs règles – Solution (seul fichier JAPE deux règles)

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

Rule: CityWithin
(
 {Lookup.minorType == city, Lookup within {Organization.kind == university}}
):cityName
-->
:cityName.City = {kind = uniCity}

Rule: Unicontain
(
  {Organization.kind == university, Organization contains {Lookup.minorType == city}}
):orgName
-->
:orgName.University = {kind = cityUni}
```

Lab 9: Combiner plusieurs règles – Problème

Un seul fichier JAPE avec control de type appelt

→ Problème :Plus d'annotation 'City'

→ Appelt a trouvé le plus long match puis s'est arrêté

→ Comment peut-on résoudre ce problème?

Lab 9: Combiner plusieurs règles – Solution

→ **Solution :**

- → **Mettre chaque règle dans un fichier**
- → **Changer de type de contrôle ver ALL**
- → **Multiphase**

Lab 9: Controll=all

Phase: University
Input: Token Lookup Split Organization
Options: **control = all**

Lab 9: Combiner plusieurs règles - Multiphase

MultiPhase: Example
Phases:
unicontain
Citywithin

```
Phase: University
Input: Token Lookup Split Organization
Options: control = appelt
Rule: CityWithin
(
 {Lookup.minorType == city, Lookup within
{Organization.kind == university}}
):cityName
-->
:cityName.City = {kind = uniCity}
```

```
Phase: University
Input: Token Lookup Split Organization
Options: control = appelt
Rule: Unicontain
(
  {Organization.kind == university, Organization
contains {Lookup.minorType == city}}
):orgName
-->
:orgName.University = {kind = cityUni}
```

Utiliser la priorité (Le contrôle Applet)

Dans le style applet, la règle à appliquer est sélectionnée dans l'ordre suivant :

- correspondance la plus longue

- priorité explicite
  - Les nombres les plus élevés ont une plus grande priorité
  - En l'absence de paramètre de priorité explicite, la valeur par défaut est -1

- règle définie en premier

- Une fois qu'une correspondance a été trouvée, la correspondance se poursuit à partir du décalage suivant la fin de la correspondance.

Lab 10: Utiliser la priorité (Le contrôle Applet)

- Déclarer deux règles identique a la règle ci-dessous en générant a chaque fois une annotation univeeisty1 pour la regle1 et university2 pour la regle2
- Comment peut-on prioriser l'une sur l'autre en utilisant le contrôle applet

```
Rule: Unicontain
(
  {Organization.kind == university, Organization contains {Lookup.minorType == city}}
):orgName
-->
:orgName.University = {kind = cityUni}
```

Lab 10 : Utiliser la priorité (Le contrôle First)

- Comment peut-on prioriser l'une des règles en utilisant un autre type de contrôle?

```
Rule: Unicontain
(
  {Organization.kind == university, Organization contains {Lookup.minorType == city}}
):orgName
-->
:orgName.University = {kind = cityUni}
```