Geco 3.0 – Parser model

Grammatica

in formato EBNF (*per cui parentesi **()**, punto di domanda **?**, asterisco ***** e croce **+** sono sempre metasimboli)*; in minuscolo gli elementi non-terminali, in **grassetto MAIUSCOLO** gli elementi terminali; commenti dopo il doppio slash **//**; **SC** è il terminale per il *punto e virgola*; **LP** e **RP** i terminali per le parentesi; **COMMA** è il terminale per la virgola; **ID** è il terminale per il generico identificatore (in formato non puntato); **FIELD_NAME** è il terminale per il singolo *fieldReference* (i.e. un *identificatore preceduto da un punto*)

Il linguaggio, brevemente, è una serie, eventualmente vuota, di istruzioni:

start:

```
( getCollection
| setIntermediateAs
| saveAs
| spatialJoin
| joinOfCollections
| filter
| group
| expand
| mergeCollections
| intersectCollections
| subtractCollections
| useDb
| trajectoryMatching
| createFuzzyOperator
| createJavaScriptFunction
)*   EOF
```

Per il resto della grammatica, fare riferimento al file "**Geco 3.0 - Syntactic Grammar.pdf**"

Modello delle classi

Il parser è composto è contenuto in 3 package

- **geco.parser**: contiene il compilatore Geco e l'oggetto con il quale risponde
  - **geco.parser.GecoParser**: il compilatore;
  - **geco.parser.Environment**: oggetto di risposta del compilatore

- **geco.model**: contiene i descrittori alle *istruzioni* del linguaggio Geco
  - ***abstract*** **geco.model.Instruction:** classe astratta che descrive l'istruzione generica e da cui derivano tutti gli altri descrittori (di seguito);
  - **geco.model.GetCollection** ***extends Instruction***
  - **geco.model.SetIntermediateAs** ***extends Instruction***
  - **geco.model.SaveAs** ***extends Instruction***
  - **geco.model.IntersectCollections** ***extends Instruction***
  - **geco.model.SpatialJoin** ***extends Instruction***
  - **geco.model.SubtractCollections** ***extends Instruction***
  - **geco.model.MergeCollections** ***extends Instruction***
  - **geco.model.Filter** ***extends Instruction***
  - **geco.model.Group** ***extends Instruction***
  - **geco.model.Expand** ***extends Instruction***
  - **geco.model.JoinCollections** ***extends Instruction***
  - **geco.model.UseDb** ***extends Instruction***
  - **geco.model.TrajectoryMatching** ***extends Instruction***
  - **geco.model. FuzzyOperator** ***extends Instruction***: introdotta nella ver. 3.0
  - **geco.model. JavascriptFunction** ***extends Instruction***: introdotta nella ver. 3.0

- **geco.model.util**: contiene tutti gli oggetti che compongono le istruzioni del linguaggio Geco
  - **geco.model.util.Condition:** questa e le classi che da questa derivano descrivono il modello delle condizioni. L'elemento terminale di una condizione è il predicato descritto dalla classe Predicate. Per ulteriori dettagli fare riferimento al modello descritto più avanti.
  - **geco.model.util.ConditionOr** ***extends Condition***
  - **geco.model.util.ConditionAnd** ***extends Condition***
  - **geco.model.util.ConditionNot** ***extends Condition***
  - **geco.model.util.Predicate**
  - **geco.model.util.WithPredicate**
  - **geco.model.util.WithoutPredicate**
  - **geco.model.util.Expression**
  - **geco.model.util.ExpressionTerm**
  - **geco.model.util.LogicalCondition:** questa classe introdotta nella versione 3.0 descrive in maniera alternativa il modello logico delle condizioni ed è parametrica rispetto ai predicati

  - **geco.model.util.SpatialJoinCondition**: componente dello SPATIAL JOIN
  - **geco.model.util.Partition**: componente della GROUP
  - **geco.model.util.Unpack**: componente della EXPAND
  - **geco.model.util.CaseClause**: descrittore di CASE

- **geco.model.util.WhereCase**: componente della CASE
- **geco.model.util.TrajectoryPartition**: componente del TRAJECTORY MATCHING
- **geco.model.util. PartitionMatching**: componente del Trajectory Partition

- **geco.model.util.GenerateAction**: descrittore di GENERATE ACTION
- **geco.model.util.GeometricOption**: descrittore di GEOMETRIC OPTION
- **geco.model.util.ObjectStructure**: descrittore di OBJECT STRUCTURE
- **geco.model.util.OutputFieldSpec**

- **geco.model.util.DbCollection**
- **geco.model.util.DbName**
- **geco.model.util.Field**
- **geco.model.util.SortField**: classe FIELD estesa con il campo VERSUS
- **geco.model.util.Value:** classe che impacchetta tutti i tipi di valori degli attributi (numerici, booleani, tra apici o quote)
- **geco.model.util.Parameter:** classe introdotta nella ver. 3.0 che descrive i parametri per CreateJavascriptFunction e CreateFuzzyOperator

- **geco.model.fuzzy:** package introdotto nella ver. 3.0 che contiene le contiene le classi aggiuntive per le estensioni fuzzy
  - **geco.model.fuzzy.AlphaCut:** descrittore della parte AlphaCut nell'estensione fuzzy della clausola WhereCase
  - **geco.model.fuzzy.FuzzyGenerate:** descrittore della parte Fuzzy Generate nell'estensione fuzzy della clausola WhereCase
  - **geco.model.fuzzy.FuzzyPoint:** descrittore dei punti della clausola Polyline del CreateFuzzyOperator
  - **geco.model.fuzzy.FuzzyRange:** descrittore della clausola Range del CreateFuzzyOperator
  - **geco.model.fuzzy.FuzzySetReference:** descrittore dei FuzzySet elencati nella clausola SetFuzzySets
  - **geco.model.fuzzy.IfFails:** descrittore del predicato IF-FAILS. Può contenere un'espressione logica che lega diversi IffOperators secondo il modello descritto dalla classe LogicalCondition.
  - **geco.model.fuzzy.IffOperator:** descrittore degli operatori che possono essere contenuti nei predicati IF-FAILS.
  - **geco.model.fuzzy.KeepingDroppingFuzzySets:** descrittore della parte
  - **geco.model.fuzzy.SetFuzzySets:** descrittore della parte KEEPING/DROPPING dell'estensione Fuzzy della WhereCase

Note sulle classi – fare riferimento al class diagram

Ogni classe descrive una parte più o meno complessa del linguaggio J-CO. Tutte le classi hanno quindi il metodo ***toString()*** che formattata in maniera standard la parte descritta. I metodi per i quali è definito il metodo ***toMultilineString()*** ritornano la parte descritta su più linee.

In generale, per comodità di utilizzo, si sono lasciati gli attributi notevoli delle classi ***public*** evitando di creare i dovuti metodi di accesso in lettura e scrittura.

Classe geco.parser.GecoParser

- Una volta istanziato il parser, l'analisi si attiva dal metodo ***start ()***
- La risposta del parser è contenuta nell'oggetto ***Environment***

Class geco.parser.Environment

- Il metodo ***getInstructionList ()*** restituisce la lista delle istruzioni riconosciute
- Il metodo ***getErrorList()*** restituisce la lista degli errori rilevati
- In **caso di errori** il parser restituisce comunque una lista delle istruzioni, ma bisogna fare attenzione che alcune componenti potrebbero essere **NULL** laddove ci si aspetta che non lo siano

Class *abstract* geco.model.Instruction

- Classe da cui derivano tutti I descrittori delle istruzioni GECO
- Per ogni istruzione (sono 11 - vedere grammatica) c'è una classe descrittrice
- Ogni Istruzione ha un ID, un numero di sequenza con la quale rilevata dal parser, un descrittore testuale dell'istruzione
- Ogni istruzione ha un metodo ***toString()*** che ricompone il testo descrittore che ha generato l'istruzione stessa in un'unica linea
- Ogni istruzione ha un metodo ***toMultilineString()*** che opera come ***toString()*** ma il testo generato è su più linee
- Se un'istruzione ha delle parti opzionali (es la *SpatialJoinCondition* e la *Case* della *SPATIAL JOIN*), allora ci sono dei metodi booleani del tipo ***hasCondition()*** per indicare se l'oggetto associato è istaziato o **NULL**

package model.util

- Le classi contenute in questo package contengono i componenti dei descrittori delle istruzioni contenute nel package geco.model
- Se un componente ha delle parti opzionali (es l'*ObjectStructure* e la *GeometricOption* all'interno della *GenerateAction*) allora c'è un metodo booleano del tipo ***hasGeometricOption()*** per indicare se l'oggetto associato è istanziato o **NULL**
- Se un componente può avere diversi aspetti mutualmente esclusivi (es. un oggetto Predicate può contenere in maniera alternativa un oggetto *Expression* oppure *WithPredicate* oppure *WithoutPredicate*) allora sono definite delle costanti che descrivono tutti i casi è c'è un attributo ***int type*** che descrive il caso specifico, è tutti gli attributi non necessari per descrivere il caso in questione sono a **NULL**.

Modello delle CONDIZIONI *di geco.model.util.Condition*

- Il modello è ricorsivo (vedere grammatica… dalla produzione *orCondtion* in poi)
- La classe principale è la ***geco.model.util.Condition*** che descrive una *condizione generica* da cui
- Una condizione generica può essere un' ***espressione booleana*** oppure un ***predicato***
- La classe *Condition* ha definite delle costanti e un attributo ***type*** che indica il caso dello specifico oggetto istanziato
- Un'espressione booleana è composta da condizioni legate dagli operatori OR, AND e NOT (indicati con precedenza crescente)
- La classe **ConditionOr** descrive una serie di condizioni in OR tra loro
- La classe **ConditionAnd** descrive una serie di condizioni in AND tra loro

- La classe **ConditionNot** descrive un condizioni preceduta da un operatore NOT
- Un predicato può essere di 4 tipi: *With*, *Without*, *Expression* o *Comparison* (i.e. un confronto tra due espressioni es: *e1 <= e2*)
- La classe **Predicate** ha un attributo ***type*** che descrive il caso dello specifico oggetto istanziato. In base al caso specifico, i descrittori degli altri casi sono a **NULL**
  La classe **Predicate**, nel caso l'oggetto sia di tipo *Expression* ha solo il l'attributo ***expression1*** valorizzato, nel caso sia di tipo *Compare (o Comparison)* anche l'attributo ***expression2*** è valorizzato (oltre all'attributo ***comparator***)
- I predicati *WITH* e *WITHOUT* sono descritti nella grammatica e sono foglie non ricorsive del modello delle condizioni
- Le *espressioni* sono le classiche espressioni con le 4 operazioni **+**, **-**, *****, **/** (indicate con precedenza crescente)
- Un'espressione (class ***Expression***) è una *sommatoria* (albebrica) di diversi *termini* (class ***ExpressionTerm***). Per ogni *termine* c'è un operatore che lo precede (il primo elemento in caso può essere vuoto, oppure no… pensare ai casi: **a+b** e **+a+b**)
- Un termine è una *produttoria* (algebrica) di *fattori* (class **ExpressionFactor**). Per ogni fattore c'è un operatore che lo precede (**NB.** *in questo caso il 1° elemento è sempre vuoto*)
- Un fattore è un elemento atomico dell'espressione. Può essere una costante numerica, un *fieldReference*, un identificatore generico, una funzione con eventuali parametri (che eventualmente possono essere delle ulteriori espressioni complesse) tra parentesi, *oppure una nuova* **condizione tra parentesi** (e qui si *attiva la ricorsione* del modello).
- Consultare il documento "Geco 2.6 - ObjectDiagram.pdf" per avere degli esempi di generazione degli oggetti che descrivono una condizione

Nuovo Modello delle CONDIZIONI *di geco.model.util.LogicalCondition<P>*

- Questo modello è stato introdotto nella versione 3.0 per gestire le relazioni tra operatori IffOperator all'interno di IF-FAILS
- Il modello descrive una generica espressione booleana di predicati, dove il tipo/classe ***P*** del predicato è parametrica. Nel caso specifico di IF-FAILS, i singoli predicati sono gli operatori IffOperator.
- Una LogicalCondition può avere uno di questi aspetti/tipi
  - PREDICATE : descrive il semplice predicato
  - NOT_CONDITION: descrive una condizione negata
  - AND_CONDITION: descrive una lista di condizioni in AND
  - OR_CONDITION: descrive una lista di condizioni in OR
  - SUB_CONDITION: descrive una sub-condizione (condizione tra parentesi)
- In base al tipo di condizione solo l'attributo relativo è valorizzato, mentre gli altri sono tutti a NULL.
- Per dettagli consultare il documento **Geco 3.0 - ClassModel.pdf**