Rule specification grammar

```
<specification> :== (<globalFilter>?) "obligations:\n" (<rule>*)
<globalFilter> :== <filter>
<rule> ::= " - " <name> <type> (<filter>?) (<higherFunctions>?) <trigger> <requirement>
 <deadline> <domain>


<name> ::= "name: " <string> "\n" /*<string> is any valid java string, must be unique
 across all rules*/
<type> ::= "type: " ("achievement" | "maintenance") "\n"
<filter> ::= "filter: " <attributeCondition> "\n" /*<attributeCondition> is any first-
 order logic condition where all variables satisfy <attributeIdentifier>*/
<higherFunctions> ::= "higherFunctions: \n" (<higherFunctionDef>+)
<higherFunctionDef> ::= "\"" <functionIdentifier> "\" : \ndomain: " <domainDef>
 ",\nfunction: " <updateFunction> "\n"


<trigger> ::= "trigger: " <functionCondition> "\n" /*<attributeCondition> is any first-
 order logic condition where each variable satisfies either <updateFunction> or
 <functionIdentifier>*/
<requirement> ::= "requirement: " <functionCondition> "\n"
<deadline> ::= "deadline: " <functionCondition> "\n"
<domain> ::= "domain: [" ((<attributeIdentifier> (", " <attributeIdentifier>)*)?) "]\n"


<attributeIdentifier> ::= <identifier> /*<identifier> can be any legal java identifier
 matching a field in the event log. Note the value type of the attribute is specified in
 the log's metadata, and the use of an attributeIdentifier variable within any
 mathematical expression must be valid for its type*/
<functionIdentifier> ::= "%" <identifier> /*<identifier> can be any legal java
 identifier, must be unique across all higher function definitions*/
<updateFunction> ::= <MATCH> | <LAST> | <COUNTIF> | <SUMIF> | <ADDSUB> | <LAST> |
 <DAYSBETWEEN> | <END>

/*The inputs to update functions are mathematical expressions (including first-order
 conditions) where each variable in the expression must satisfy one of
 <attributeIdentifier>, <updateFunction> or <functionIdentifier>. Note for certain
 function inputs, the return value-type of the expression is limited*/
<MATCH> = "MATCH(" <boolExpression> ")" /*function output type is bool*/
<COUNTIF> = "COUNTIF(" <boolExpression> ")" /*function output type is numeric*/
<SUMIF> = "SUMIF(" <numericExpression> ", " <boolExpression> ")" /*function output type
 is numeric*/
<ADDSUB> = "ADDSUB("<numericExpression> ", " <boolExpression> ", " <boolExpression> ")"
 /*function output type is numeric*/
<LAST> = "LAST(" <expression>", " <boolExpression> ")" /*function output type matches
 input expression type*/
<DAYSBETWEEN> = "DAYS_BETWEEN(" <boolExpression> ", " <boolExpression> ")" /*function
 output type is numeric*/
<END> = "END()" /*function output type is bool*/
```

Event-log metadata

```
<metadata> ::= <attributeMap> <parserOptions>

<attributeMap> ::= "attributeMap:\n" (<attribute>*)
<attribute> ::= " - label: " <attributeIdentifier> "\ntype: " <attributeType> "\n"
  (<attributeProperties>?)
<attributeType> ::= "int" | "integer" | "bool" | "boolean" | "double" | "str" | "string"
  | "timestamp"
<attributeProperties> ::= "properties:\n" (<format>?)
<format> :== "format: " <parsingFormat> "\n" /*<parsingFormat> is the specific parsing
  format for the attribute, eg. the timestamp format*/

<parserOptions> ::= <timestampAttribute> (<relabelMap>?) (<rekeyMap>?)
<timestampAttribute> ::= "timestampAttribute: " <attributeIdentifier> "\n"
<relabelMap> ::= "relabel:\n" (<relabelEntry>+)
<relabelEntry> ::= <originalAttributeIdentifier> ":" <newAttributeIdentifier> "\n"
  /*<originalAttributeIdentifier> and <newAttributeIdentifier> have the same grammar as
  <attributeIdentifier>*/

<rekeyMap> ::= "rekey:\n" (<rekeyEntry>+)
<rekeyEntry> ::= " - " <function> <lookup> <return> <domain>
<function> ::= "function: " <attribteCondition> "\n"
<lookup> ::= "lookup: " <originalAttributeIdentifier> "\n"
<return> ::= "return: " <newAttributeIdentifier> "\n"
```