Progress Corticon

Challenge July-2025: Rules with Regular Expressions

Seth Meldon

Prompt

A university is developing a system that helps a student to select certain courses to receive a desired degree. Each course has a unique code, such as “MA523” or “CS38251”. Help a university create a simple decision service that evaluates a set of all courses to split it into two lists of allowed and not allowed courses, following these rules:

RULE 1. If the Degree Code is “MA Single” and the Course Code is similar to MA52# or MA62#, where # stands for any single character, add this course to the list of allowed courses.

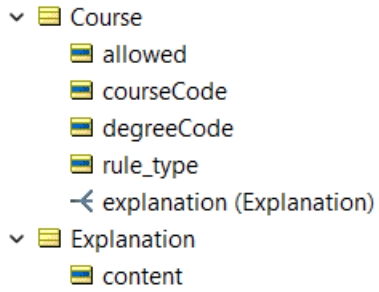
RULE 2. If the Degree Code is not “CS%” and the Course Code is similar to CS%288% or CS%289%, where % stands for any combination of characters, add this course to the list of not allowed courses.

Rule Implementation in Corticon.js Studio

Corticon provides many approaches to designing rules, so two rule authors’ implementations of the same rules can look quite different while achieving the same result. Here, let’s show three ways to implement the problem, using the same rule vocabulary.

Rule Vocabulary

The course itself is represented by the entity `Course`, which has a boolean attribute, `allowed`, and string attributes for `degreeCode` and `courseCode`. The attribute `rule_type` will define which of the three rule evaluation types to execute. A second entity, `Explanation` will be created and appended to the applicable `Course` entity, containing the rationale for why a rule is triggered in its attribute `content`.



Rule Implementation with Regular Expressions

Here we use regular expressions to define complex patterns in a concise way, able to a wide range of conditions with minimal rules.

Here's how it implements the rules:

- **Allowed Courses**: A course is marked as `allowed` if the `degreeCode` is "`MA Single`" and the `courseCode` matches the pattern `MA52.` or `MA62.`, where '`.`' represents any single character.
- **Disallowed Courses**: A course is `disallowed` if the `degreeCode` does not match `CS.*` (anything starting with "CS") and the `courseCode` matches the patterns `CS.*288.*` or `CS.*289.*`.

	Conditions	0	1	2
a	Course.degreeCode='MA Single'		T	-
b	Course.courseCode.matches('MA52.|MA62.')		T	-
c	Course.degreeCode.matches('CS.*')		-	F
d	Course.courseCode.matches('CS.*288.*|CS.*289.*')		-	T
e				
f				
g				
	Actions			
	Post Message(s)			
A	Course.allowed		T	F

When the rule in column 1 fires (having met the conditions in row 'a' and 'b'), in addition to sending back new value '`T`' for `Course.allowed`, it will send back the explanations in Action rows 'C' and 'D'. Similarly, rule 2 will trigger 'E' and 'F':

C	Course.explanation+=Explanation.new [content = 'Condition met for Degree Code "' + Course.degreeCode + '": Course.degreeCode=MA Single']
D	Course.explanation+=Explanation.new [content = 'Condition met for Course Code "' + Course.courseCode + '": Course.courseCode starts with "MA52#" or "MA62#", where # stands for any single character']
E	Course.explanation+=Explanation.new [content = 'Condition met for Degree Code "' + Course.degreeCode + '": Course.courseCode does not start with the string "CS"']
F	Course.explanation+=Explanation.new [content = 'Condition met for Course Code "' + Course.courseCode + '": string meets the pattern "CS%288%" or "CS%289%", where % stands for any combination of characters']

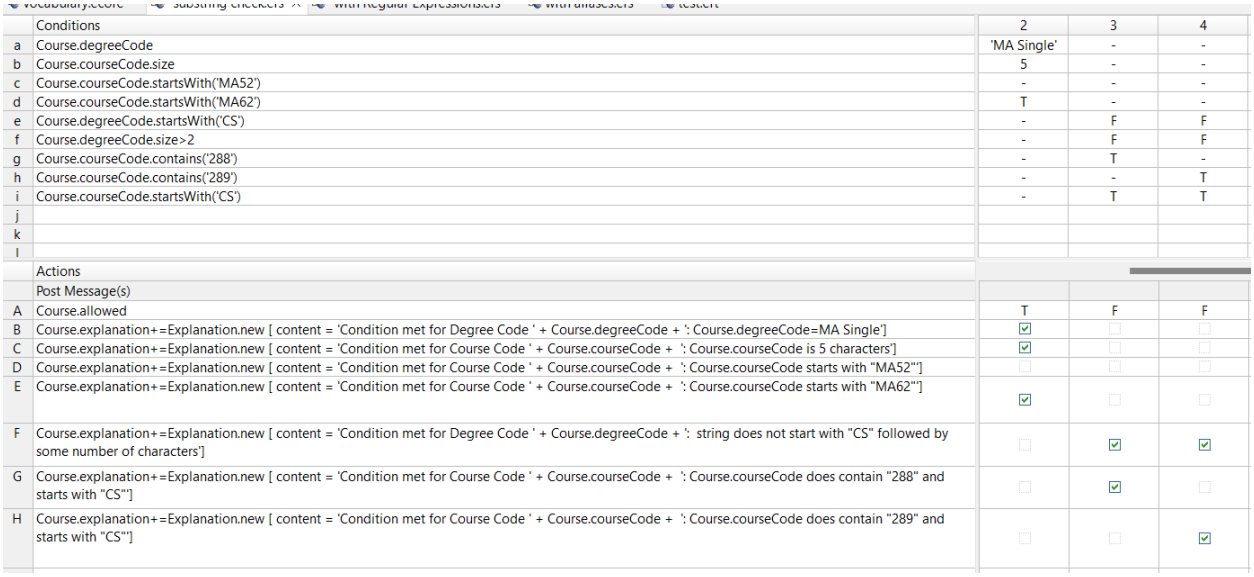
Substring Check

This rulesheet uses basic string manipulation functions to check for specific patterns and conditions. It relies on a combination of startsWith, contains, and checking the string size to validate the course and degree codes. This approach is straightforward and easy to understand but can be more verbose if the conditions are complex.

Here are the conditions for this rulesheet:

- **Allowed Courses**: A course is `allowed` if the `degreeCode` is exactly "`MA Single`", the `courseCode` is 5 characters long, and it starts with either "`MA52`" or "`MA62`".

- **Disallowed Courses**: A course is `disallowed` if the `degreeCode` does not start with "`CS`" and the `courseCode` contains either "`288`" or "`289`".



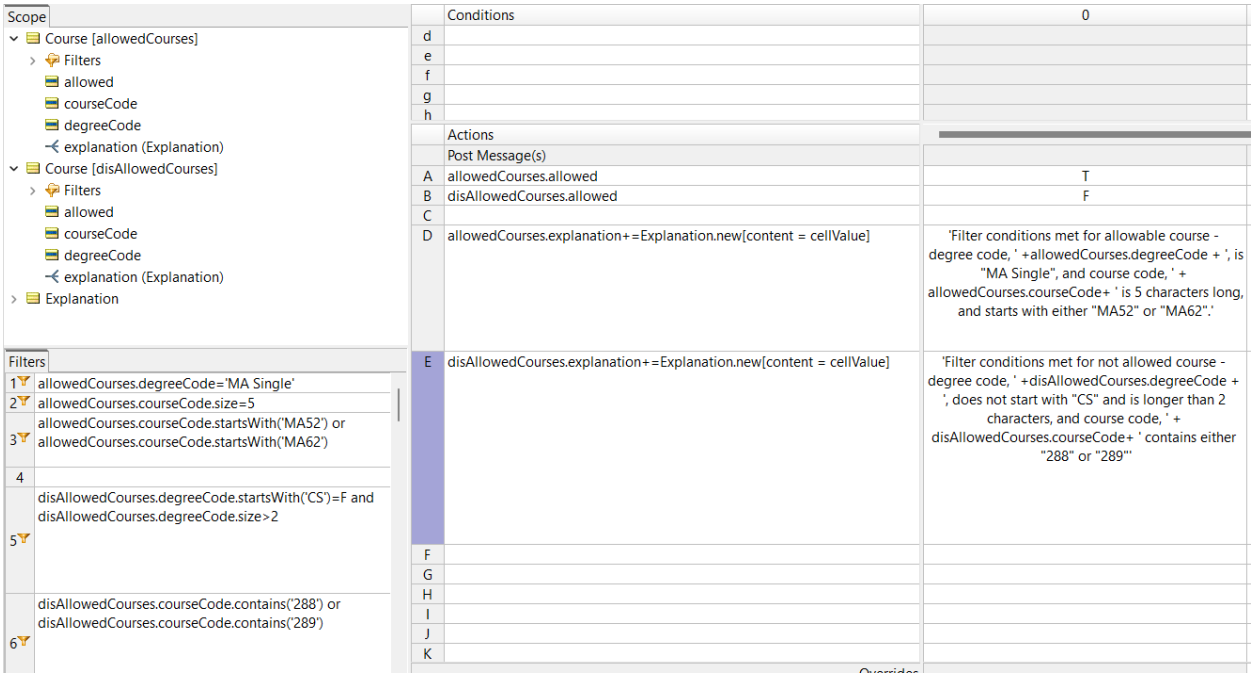
	Conditions	2	3	4
a	Course.degreeCode	'MA Single'	-	-
b	Course.courseCode.size	5	-	-
c	Course.courseCode.startsWith('MA52')	-	-	-
d	Course.courseCode.startsWith('MA62')	T	-	-
e	Course.degreeCode.startsWith('CS')	-	F	F
f	Course.degreeCode.size>2	-	F	F
g	Course.courseCode.contains('288')	-	T	-
h	Course.courseCode.contains('289')	-	-	T
i	Course.courseCode.startsWith('CS')	-	T	T
j				
k				
l				
	Actions			
	Post Message(s)			
A	Course.allowed	T	F	F
B	Course.explanation+=Explanation.new [content = 'Condition met for Degree Code ' + Course.degreeCode + ': Course.degreeCode=MA Single']	☑	☐	☐
C	Course.explanation+=Explanation.new [content = 'Condition met for Course Code ' + Course.courseCode + ': Course.courseCode is 5 characters']	☑	☐	☐
D	Course.explanation+=Explanation.new [content = 'Condition met for Course Code ' + Course.courseCode + ': Course.courseCode starts with "MA52"']	☐	☐	☐
E	Course.explanation+=Explanation.new [content = 'Condition met for Course Code ' + Course.courseCode + ': Course.courseCode starts with "MA62"']	☑	☐	☐
F	Course.explanation+=Explanation.new [content = 'Condition met for Degree Code ' + Course.degreeCode + ': string does not start with "CS" followed by some number of characters']	☐	☑	☑
G	Course.explanation+=Explanation.new [content = 'Condition met for Course Code ' + Course.courseCode + ': Course.courseCode does contain "288" and starts with "CS"']	☐	☑	☐
H	Course.explanation+=Explanation.new [content = 'Condition met for Course Code ' + Course.courseCode + ': Course.courseCode does contain "289" and starts with "CS"']	☐	☐	☑

With Aliases

This rulesheet uses filters and aliases to break down the conditions into smaller, more manageable parts. Each filter checks a specific condition, and the results are then combined to make a final decision. While this approach can be more organized and easier to read for very complex rules, it can also be less direct than the other two methods.

The logic is as follows:

- **Allowed Courses**:
    - A filter first checks if the `degreeCode` is "`MA Single`".
    - Another filter verifies that the `courseCode` is 5 characters long.
    - A third filter ensures the `courseCode` starts with "`MA52`" or "`MA62`".
    - If all three filters pass, the course is `allowed`.
- **Disallowed Courses**:
    - One filter checks if the `degreeCode` does not start with "`CS`" and is longer than 2 characters.
    - A second filter checks if the `courseCode` contains "`288`" or "`289`".
    - If both conditions are met, the course is `disallowed`.

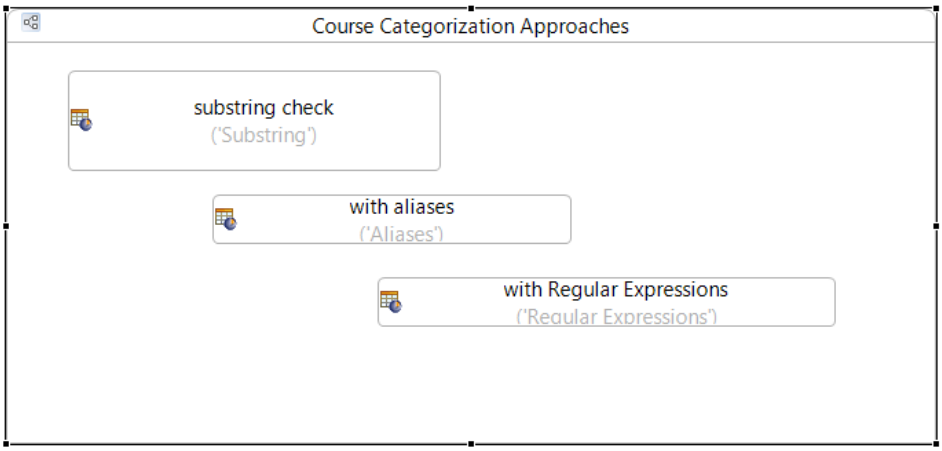


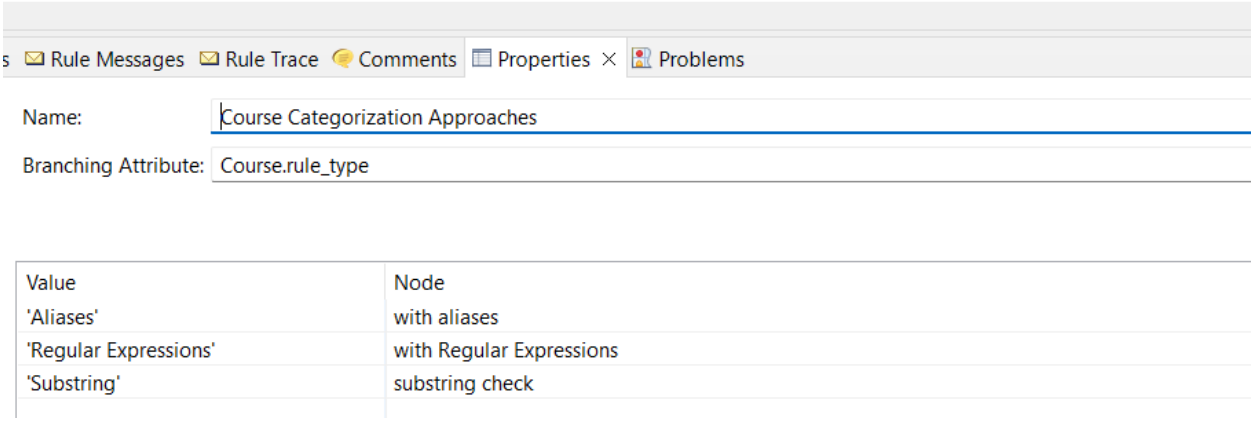
Routing to the Respective Rulesheets

The ruleflow is designed to process Course entities by dynamically choosing one of three different rulesheets based on the value of the `rule_type` attribute. This is controlled by a central **Branch Container**.

The logic is as follows:

- **Initial Step**: The process begins at the Branch Container.
- **Branching Condition**: The value of the `Course.rule_type` attribute is evaluated.
- **Execution Paths**:
    - If `Course.rule_type` is **'Substring'**, the ruleflow executes the **substring check.ers** rulesheet.
    - If `Course.rule_type` is **'Regular Expressions'**, the ruleflow executes the **with Regular Expressions.ers** rulesheet.
    - If `Course.rule_type` is **'Aliases'**, the ruleflow executes the **with aliases.ers** rulesheet.





Value	Node
'Aliases'	with aliases
'Regular Expressions'	with Regular Expressions
'Substring'	substring check

Try it

Use the end result decision service via a web front end to pass in Course test cases here.

Download the full rule project zip file to import into Corticon Studio or Corticon.js Studio here.