Comorbidities Rule Builder

Documentation for the project
Under the guidance of
Professor I.V. Ramakrishnan

At
State University of New York at Stony Brook
Department of Computer Science
November 9, 2017

RULE BUILDER DOCUMENTATION

Introduction

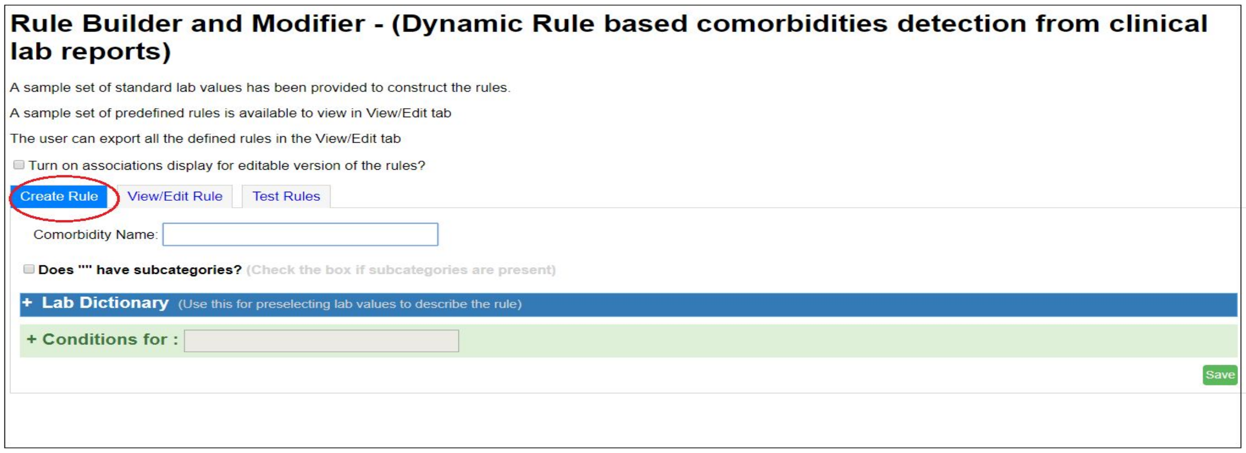
Comorbidities are diagnosed using a set of rules that compare certain lab dictionary values against assigned threshold limits.In the comorbidity diagnosis system, all these comparisons are done using static functions defined as a part of the CERNER system. Adding a new comorbidity requires changes in the backend javascript code, making the system rigid. The rule builder application specially allows users to create rules for newer comorbidities. These rules would then be used in the existing comparison based method to identify what comorbidities are present. This makes the system more user friendly as the intended user can write the rules using a GUI that makes creating the rules as simple as writing it in plain language. There is no need of any programming knowledge. Thus, the rule builder application makes the comorbidities' system flexible and more dynamic in nature.

Guide to the Rule Builder Application

1. Create Rule

This is the first tab of the home page of the rule builder application. This page allows the user to create new rules to be used for diagnosing comorbidities. To create a new rule, the user needs to populate all the lab dictionary values being used for comparisons against their threshold values. The tool also allows the user to add subcategory for each comorbidity and the set of rules for them individually. The conditions assigned for each rule are also interpreted in plain language which makes it easy for the user to validate the created rule with the intended rule functionality.
Following are the steps in which a rule can be created:

a. Open the rule builder link in the browser & click on 'Create Rule'


Rule Builder and Modifier - (Dynamic Rule based comorbidities detection from clinical lab reports)

A sample set of standard lab values has been provided to construct the rules.

A sample set of predefined rules is available to view in View/Edit tab

The user can export all the defined rules in the View/Edit tab

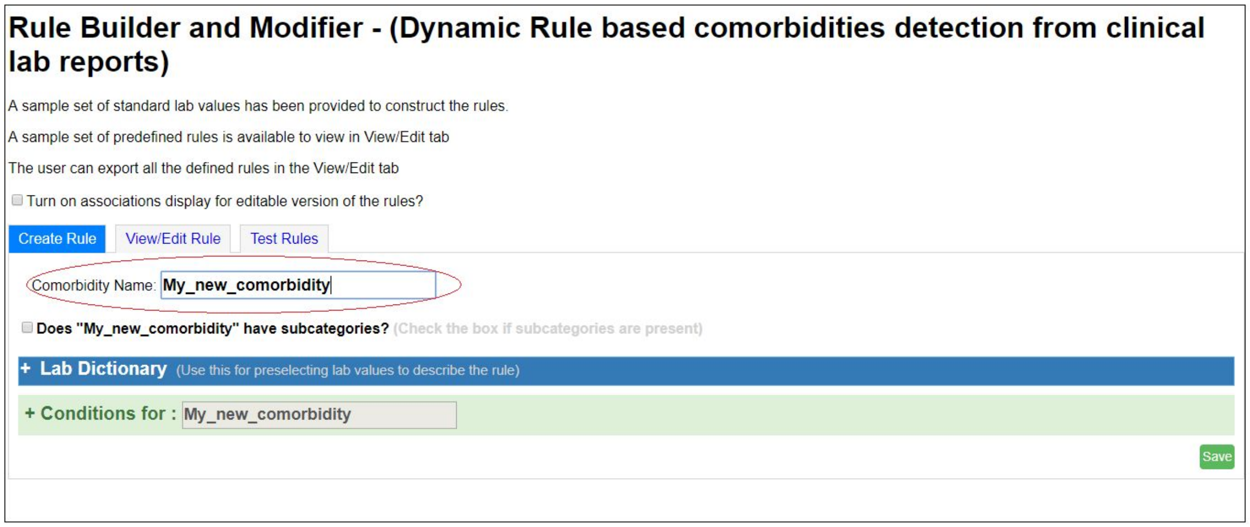
☐ Turn on associations display for editable version of the rules?

☐ **Does "" have subcategories?** (Check the box if subcategories are present)

+ Lab Dictionary (Use this for preselecting lab values to describe the rule)

+ Conditions for :

b. Type the name of the comorbidity for which the rule is to be created
(eg. Kidney Injury)



Rule Builder and Modifier - (Dynamic Rule based comorbidities detection from clinical lab reports)

A sample set of standard lab values has been provided to construct the rules.

A sample set of predefined rules is available to view in View/Edit tab

The user can export all the defined rules in the View/Edit tab

☐ Turn on associations display for editable version of the rules?

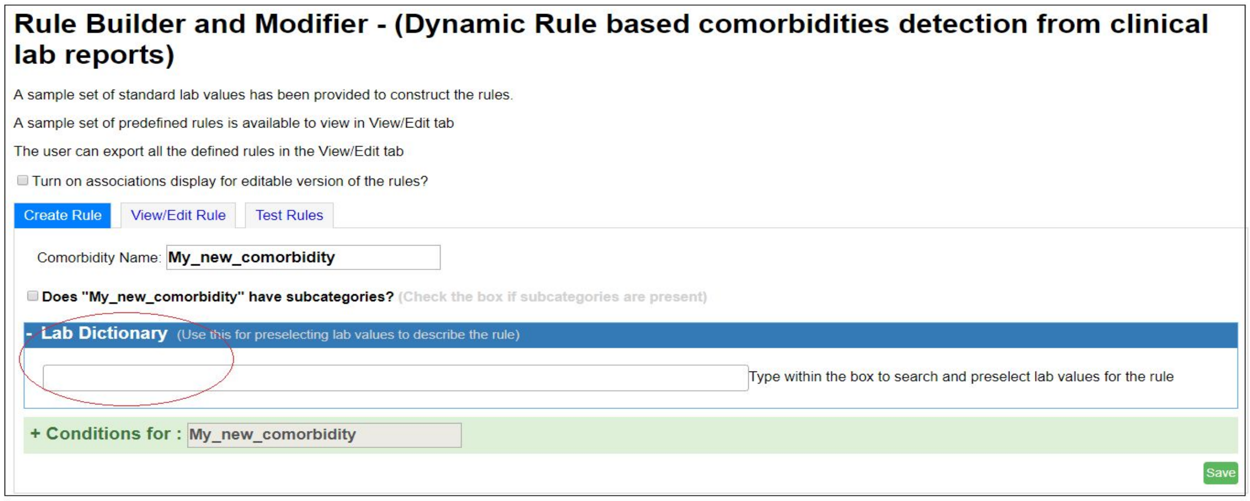
Comorbidity Name: My_new_comorbidity

☐ **Does "My_new_comorbidity" have subcategories?** (Check the box if subcategories are present)

+ Lab Dictionary (Use this for preselecting lab values to describe the rule)

+ Conditions for : My_new_comorbidity

c. Click on Lab Dictionary (+). The lab dictionary provides a list of all the terms needed to create a rule, such as Minimum Hemoglobin, Maximum Creatinine etc. The user can select one or more terms from the lab dictionary which would be used to create the rule.



Rule Builder and Modifier - (Dynamic Rule based comorbidities detection from clinical lab reports)

A sample set of standard lab values has been provided to construct the rules.

A sample set of predefined rules is available to view in View/Edit tab

The user can export all the defined rules in the View/Edit tab

☐ Turn on associations display for editable version of the rules?

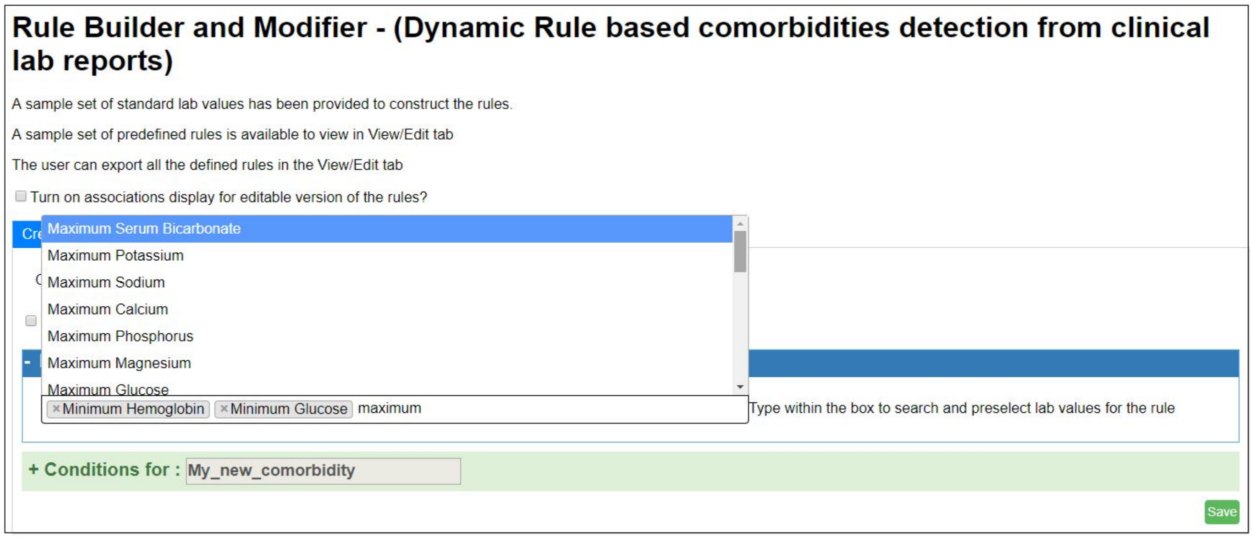
Comorbidity Name: My_new_comorbidity

☐ **Does "My_new_comorbidity" have subcategories?** (Check the box if subcategories are present)

- Lab Dictionary (Use this for preselecting lab values to describe the rule)

Type within the box to search and preselect lab values for the rule

+ Conditions for : My_new_comorbidity

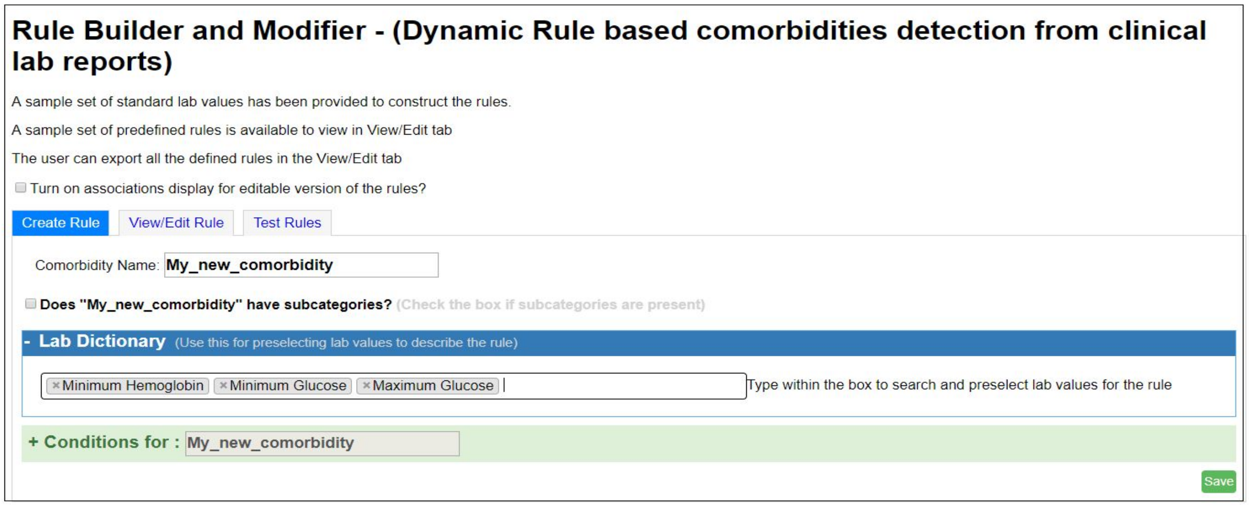

Rule Builder and Modifier - (Dynamic Rule based comorbidities detection from clinical lab reports)

A sample set of standard lab values has been provided to construct the rules.

A sample set of predefined rules is available to view in View/Edit tab

The user can export all the defined rules in the View/Edit tab

Turn on associations display for editable version of the rules?


Rule Builder and Modifier - (Dynamic Rule based comorbidities detection from clinical lab reports)

A sample set of standard lab values has been provided to construct the rules.

A sample set of predefined rules is available to view in View/Edit tab

The user can export all the defined rules in the View/Edit tab

Turn on associations display for editable version of the rules?

Comorbidity Name: My_new_comorbidity

Does "My_new_comorbidity" have subcategories? (Check the box if subcategories are present)

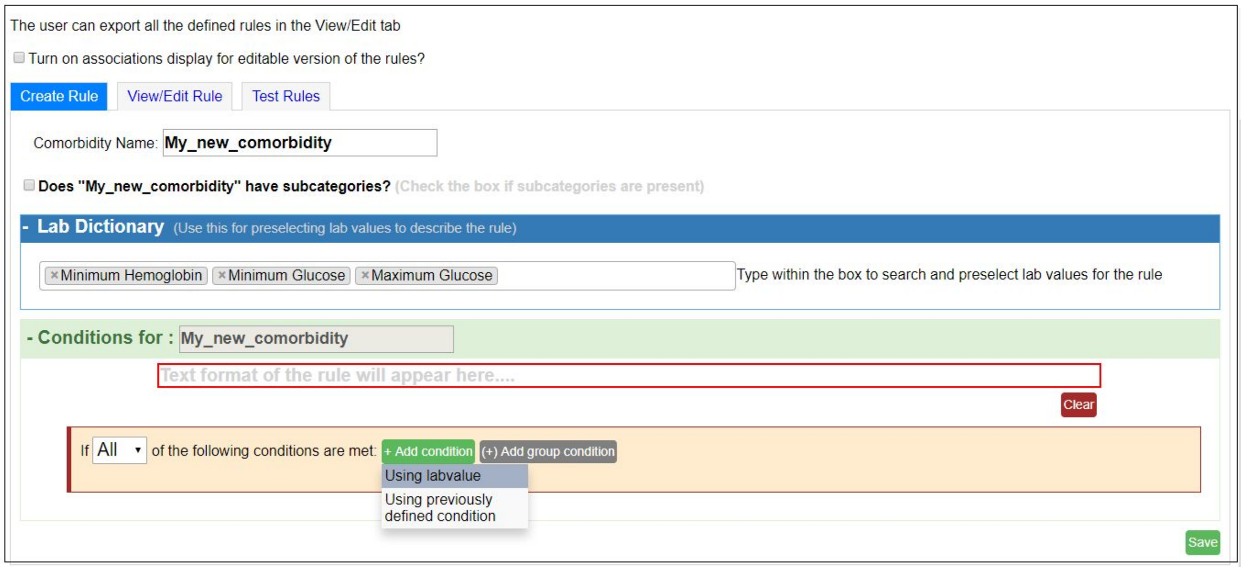
- Lab Dictionary (Use this for preselecting lab values to describe the rule)
×Minimum Hemoglobin
×Minimum Glucose
×Maximum Glucose
Type within the box to search and preselect lab values for the rule
+ Conditions for :
My_new_comorbidity
Save

d. After selecting all the required terms from the Lab Dictionary, the conditions of the rule need to be specified. Click on the 'Conditions for' (+) tab present below the Lab Dictionary terms. Here, the user has to select 1 each from three options:

i. All / any: A rule might be true if 'all' the mentioned conditions are true
A rule might be true even if just 'one' condition out of all is true
So accordingly, the user needs to select either 'all' or 'any'

ii. Add Condition: Every new condition in the rule can be added in one of 2 ways:
1.Using lab value
2. Using previously defined condition
User needs to select either one of these two

iii. For nested rules, i.e those which involve a group condition, the user can select 'add group condition'.


The user can export all the defined rules in the View/Edit tab

Turn on associations display for editable version of the rules?

Comorbidity Name: My_new_comorbidity

Does "My_new_comorbidity" have subcategories? (Check the box if subcategories are present)

Lab Dictionary (Use this for preselecting lab values to describe the rule)

×Minimum Hemoglobin ×Minimum Glucose ×Maximum Glucose

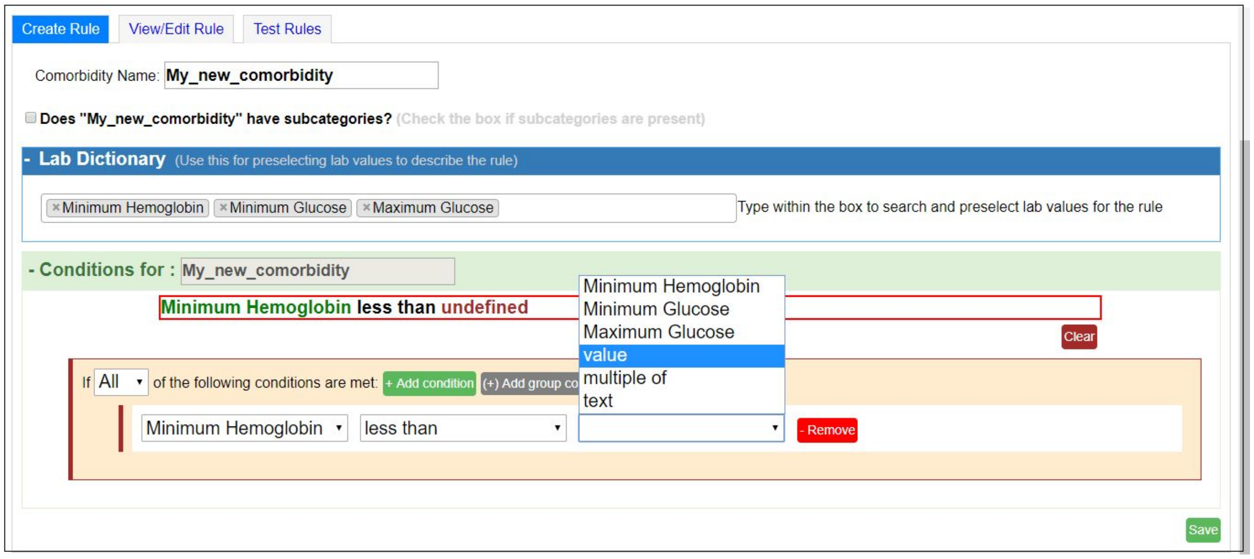
Type within the box to search and preselect lab values for the rule

Conditions for : My_new_comorbidity

Text format of the rule will appear here....

If All of the following conditions are met:

e. After selecting the above conditions, a text box containing a default lab value is displayed. The user can now actually create the rule by specifying all the necessary values. As the user creates the rule, a text version of the rule is displayed in the red-bordered text box present below the name of the comorbidity. The user can read the textual rule displayed here, to cross check if it is correctly created.
An example is as shown below:


Comorbidity Name: My_new_comorbidity

Does "My_new_comorbidity" have subcategories? (Check the box if subcategories are present)

Lab Dictionary (Use this for preselecting lab values to describe the rule)

×Minimum Hemoglobin ×Minimum Glucose ×Maximum Glucose

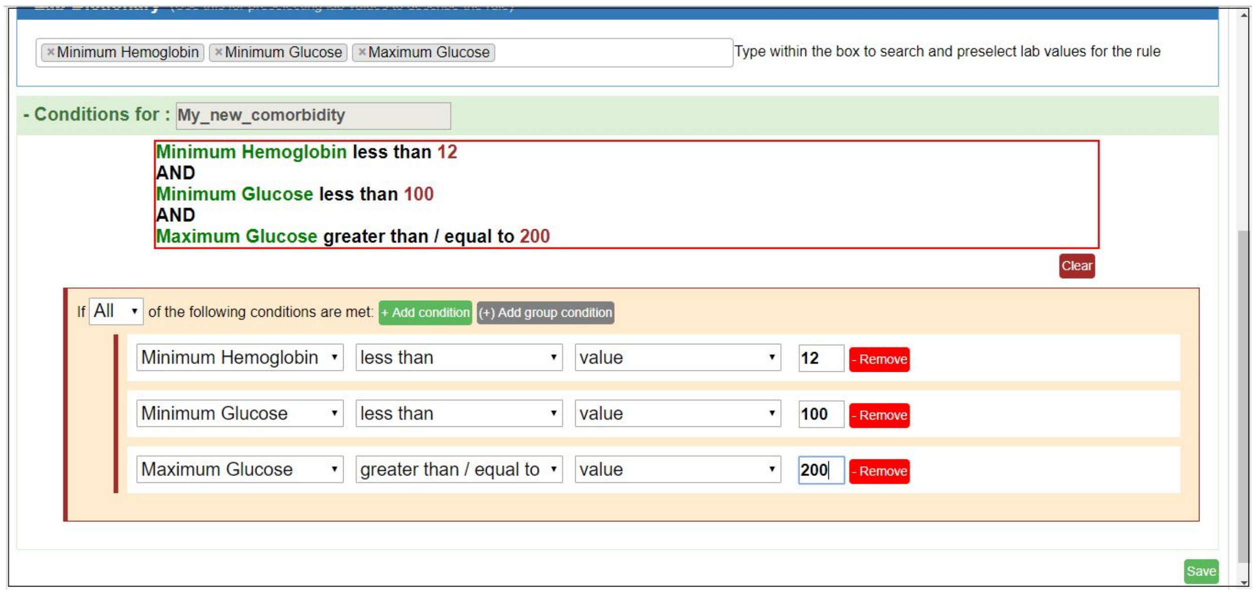
Type within the box to search and preselect lab values for the rule

Conditions for : My_new_comorbidity

Minimum Hemoglobin less than undefined

If All of the following conditions are met:

Minimum Hemoglobin | less than


×Minimum Hemoglobin ×Minimum Glucose ×Maximum Glucose | Type within the box to search and preselect lab values for the rule

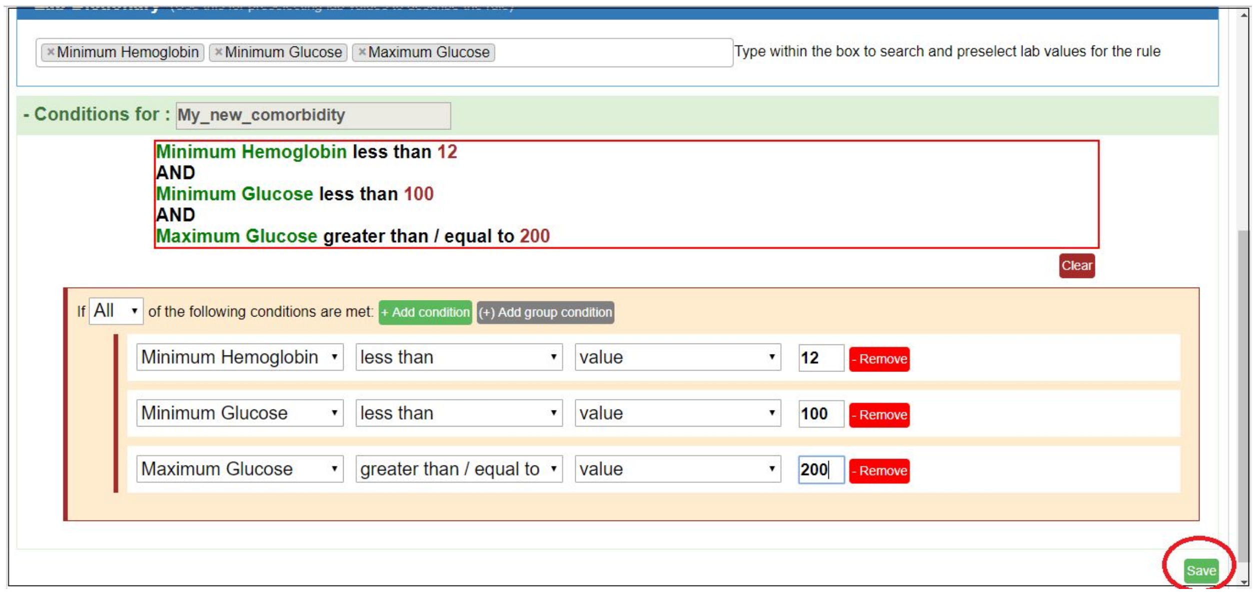
- Conditions for : My_new_comorbidity

Minimum Hemoglobin less than 12
AND
Minimum Glucose less than 100
AND
Maximum Glucose greater than / equal to 200

If All of the following conditions are met:

Minimum Hemoglobin	less than	value	12
Minimum Glucose	less than	value	100
Maximum Glucose	greater than / equal to	value	200

f. After the rule is satisfactorily created, there is tab at the bottom-right corner of the page to save the rule. Once the user clicks on ‘Save’, the newly created rule gets added to the list of previous rules.

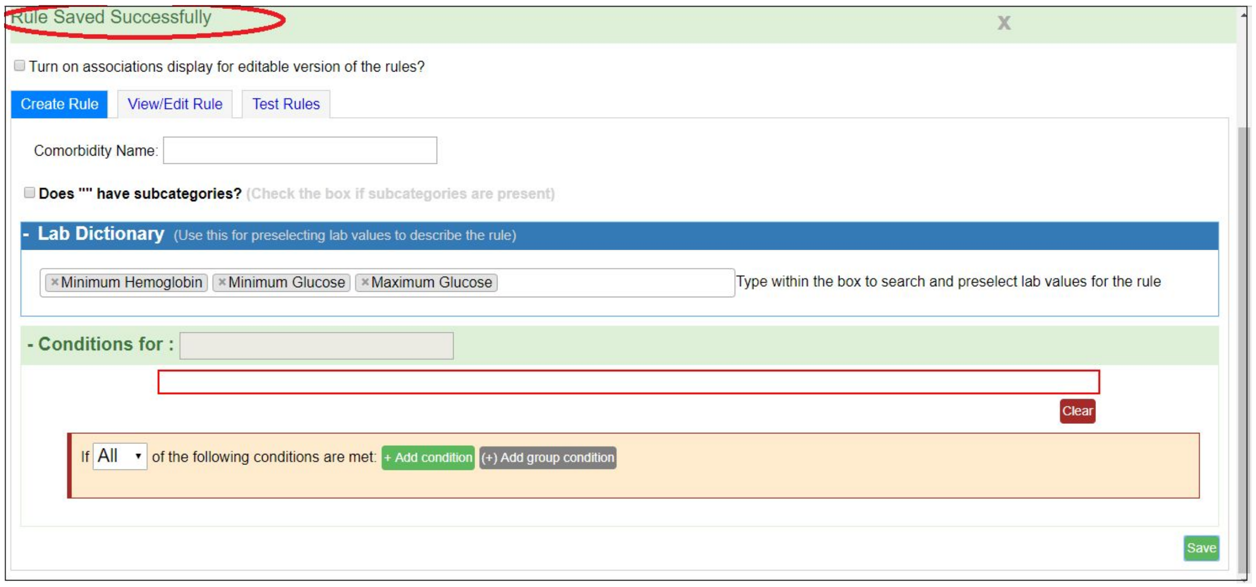

×Minimum Hemoglobin ×Minimum Glucose ×Maximum Glucose | Type within the box to search and preselect lab values for the rule

- Conditions for : My_new_comorbidity

Minimum Hemoglobin less than 12
AND
Minimum Glucose less than 100
AND
Maximum Glucose greater than / equal to 200

If All of the following conditions are met:

Minimum Hemoglobin	less than	value	12
Minimum Glucose	less than	value	100
Maximum Glucose	greater than / equal to	value	200


Rule Saved Successfully

Turn on associations display for editable version of the rules?

Comorbidity Name:

Does "" have subcategories? (Check the box if subcategories are present)

Lab Dictionary (Use this for preselecting lab values to describe the rule)

×Minimum Hemoglobin ×Minimum Glucose ×Maximum Glucose

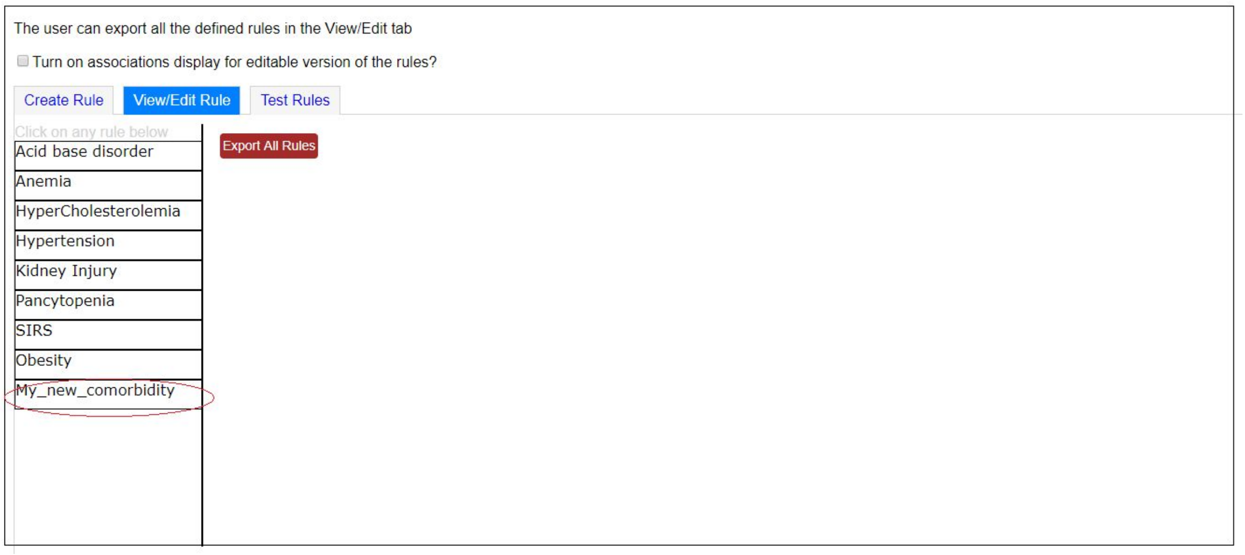
Type within the box to search and preselect lab values for the rule

Conditions for :

If All of the following conditions are met:

2. View/Edit Rule

This is the second tab of the home page of the rule builder application.This page displays all the rules created thus far. It also allows the user to modify an existing rule with functionality like adding/ removing comparisons, changing the threshold values used for comparison with lab dictionary values. The changes made from here are reflected in the globally persisted rules across the entire application.

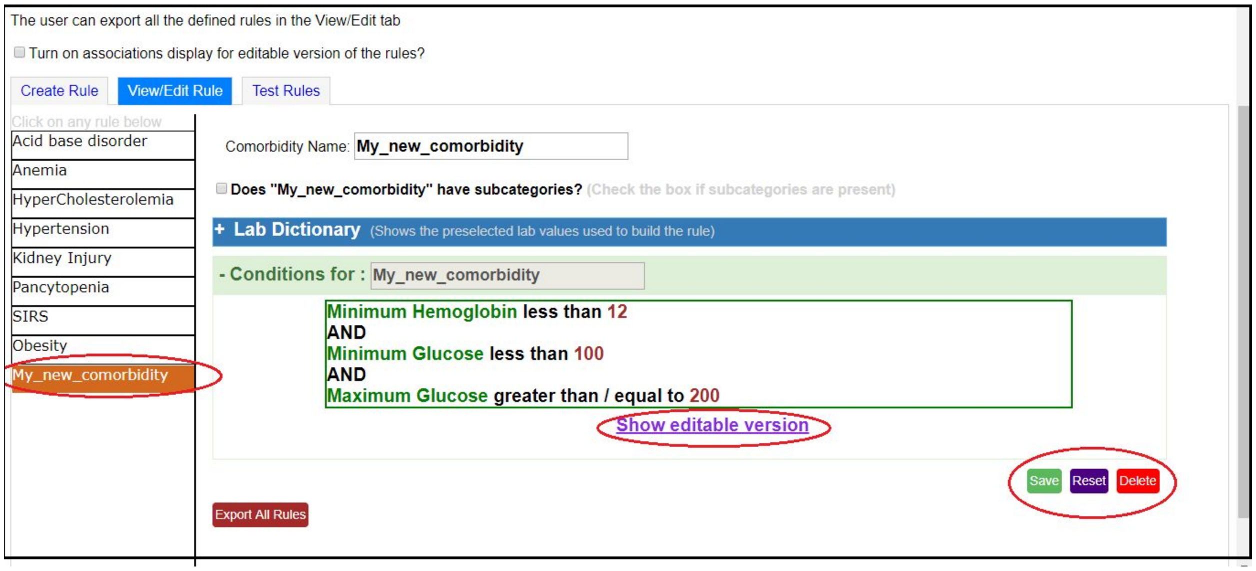

The user can export all the defined rules in the View/Edit tab

Turn on associations display for editable version of the rules?

Click on any rule below

Acid base disorder
Anemia
HyperCholesterolemia
Hypertension
Kidney Injury
Pancytopenia
SIRS
Obesity
My_new_comorbidity

There is an ‘export rules’ tab provided on this page. By clicking on this tab, the user can download a copy of all rules in json format, i.e. rules.json
On selecting a particular rule from the list of rules displayed on the left, the user can view the text-based rule. If it needs to be edited, there is a link at the bottom of the rules pane - ‘show editable version’. Clicking on this link, the user can change any values/ terms of the rule. Also, after editing the rule, there are 3 options provided - to reset, save or delete the rule which is selected.


The user can export all the defined rules in the View/Edit tab

☐ Turn on associations display for editable version of the rules?

Create Rule | View/Edit Rule | Test Rules

Click on any rule below
- Acid base disorder
- Anemia
- HyperCholesterolemia
- Hypertension
- Kidney Injury
- Pancytopenia
- SIRS
- Obesity
- My_new_comorbidity

Comorbidity Name: My_new_comorbidity

☐ **Does "My_new_comorbidity" have subcategories?** (Check the box if subcategories are present)

+ Lab Dictionary (Shows the preselected lab values used to build the rule)

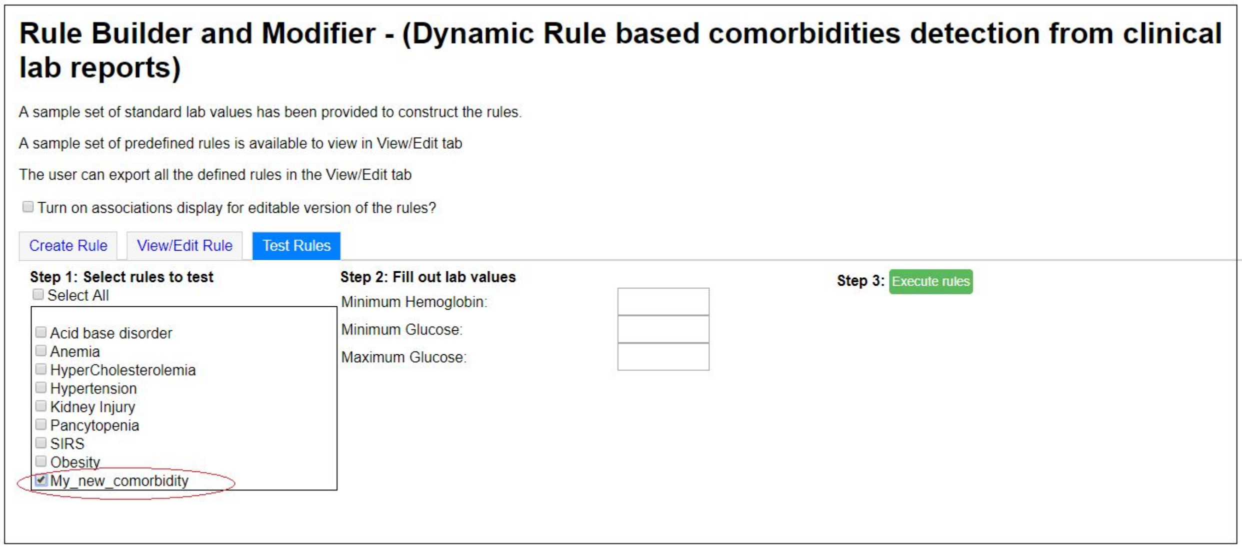
- Conditions for : My_new_comorbidity

Minimum Hemoglobin less than 12
AND
Minimum Glucose less than 100
AND
Maximum Glucose greater than / equal to 200

Show editable version

3. Test Rules

This is a part of the rule builder to test its working in a local set up environment. This tab displays all the Comorbidities for which rules are created thus far. Selecting a comorbidity displays the lab dictionary values on the right and allows the user to provide their corresponding values as input. It then invokes the rule.


Rule Builder and Modifier - (Dynamic Rule based comorbidities detection from clinical lab reports)

A sample set of standard lab values has been provided to construct the rules.

A sample set of predefined rules is available to view in View/Edit tab

The user can export all the defined rules in the View/Edit tab

☐ Turn on associations display for editable version of the rules?

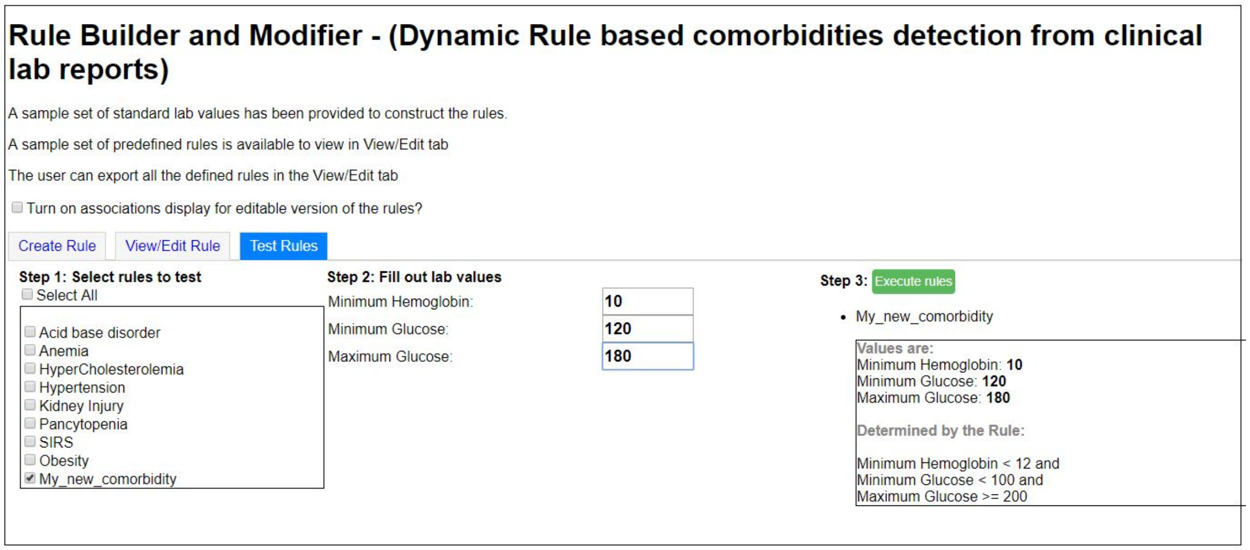
Create Rule | View/Edit Rule | Test Rules

Step 1: Select rules to test
☐ Select All

☐ Acid base disorder
☐ Anemia
☐ HyperCholesterolemia
☐ Hypertension
☐ Kidney Injury
☐ Pancytopenia
☐ SIRS
☐ Obesity
☑ My_new_comorbidity

Step 2: Fill out lab values
Minimum Hemoglobin:
Minimum Glucose:
Maximum Glucose:

Step 3: Execute rules


Rule Builder and Modifier - (Dynamic Rule based comorbidities detection from clinical lab reports)

A sample set of standard lab values has been provided to construct the rules.

A sample set of predefined rules is available to view in View/Edit tab

The user can export all the defined rules in the View/Edit tab

☐ Turn on associations display for editable version of the rules?

Step 1: Select rules to test

☐ Select All

☐ Acid base disorder
☐ Anemia
☐ HyperCholesterolemia
☐ Hypertension
☐ Kidney Injury
☐ Pancytopenia
☐ SIRS
☐ Obesity
☑ My_new_comorbidity

Step 2: Fill out lab values

Minimum Hemoglobin:	10
Minimum Glucose:	120
Maximum Glucose:	180

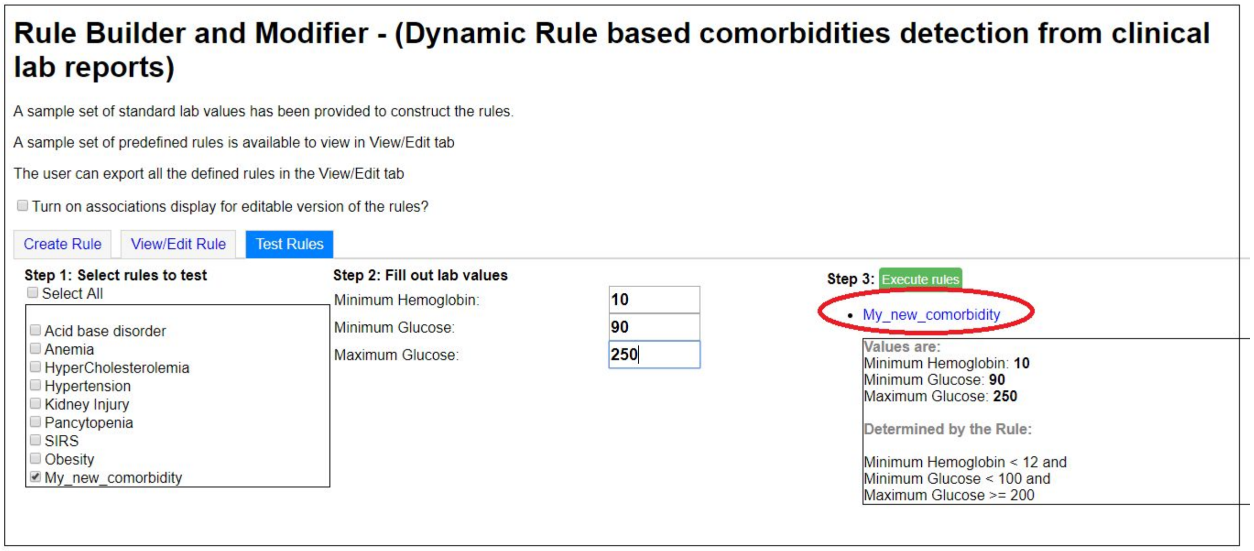
Step 3: Execute rules

- My_new_comorbidity

Values are:
Minimum Hemoglobin: **10**
Minimum Glucose: **120**
Maximum Glucose: **180**

Determined by the Rule:

Minimum Hemoglobin < 12 and
Minimum Glucose < 100 and
Maximum Glucose >= 200


Rule Builder and Modifier - (Dynamic Rule based comorbidities detection from clinical lab reports)

A sample set of standard lab values has been provided to construct the rules.

A sample set of predefined rules is available to view in View/Edit tab

The user can export all the defined rules in the View/Edit tab

☐ Turn on associations display for editable version of the rules?

Step 1: Select rules to test

☐ Select All

☐ Acid base disorder
☐ Anemia
☐ HyperCholesterolemia
☐ Hypertension
☐ Kidney Injury
☐ Pancytopenia
☐ SIRS
☐ Obesity
☑ My_new_comorbidity

Step 2: Fill out lab values

Minimum Hemoglobin:	10
Minimum Glucose:	90
Maximum Glucose:	250

Step 3: Execute rules

- My_new_comorbidity

Values are:
Minimum Hemoglobin: **10**
Minimum Glucose: **90**
Maximum Glucose: **250**

Determined by the Rule:

Minimum Hemoglobin < 12 and
Minimum Glucose < 100 and
Maximum Glucose >= 200

As seen from the above examples, if a comorbidity is detected on the basis of the rule being tested, it is displayed in blue colour. Upon hovering on the comorbidity, the rule and current readings are displayed. If it is not a comorbidity, it is simply displayed in black colour.

Thus the newly created rules can be tested.

Implementation Details

Rule Interpreter

The rule interpreter is a complete workflow tool that processes the rules and reports the comorbidities present. A 'rule' here is a set of individual or complex comparison based expressions of lab dictionary values with certain threshold values. Comorbidities can be diagnosed based on the result of the evaluations of the rules. This section encompasses all the functionality related to interpretation and evaluation of comorbidities rules. The design of the interpreter has been influenced by Douglas Crockford's Top Down Operator Precedence.

1. Interpreter: rule.interpreter.interpreter.js

Processes all the defined rules and invokes lexer, parser, evaluator and related functions. The functionality is mainly provided by the following functions:-

➔ processRulesJson():- Processes all the defined rules and invokes the parseRule() function. It takes as its input - rule.JSON, which is an array of all the rules in JSON (key,value) format where the key is the name of the comorbidity. The corresponding value is dependent on whether the comorbidity contains any subcategories:

- If the comorbidity does not contain subcategories, then the value for the key is the rule defined for that comorbidity and parseRule() is called on that rule. The result gets stored in the 'result' variable. Theresult variable is another JSON object with the key as the comorbidity name and the value as True or False depending upon the result of parseRule().

- If the comorbidity contains subcategories, then the value is an array of JSON objects. These nested JSON objects have the subcategory of the comorbidities as the key and the corresponding rules as the values. The function iterates over each subcategory and parseRule() is invoked for each sub-category's rule. The result of the parseRule() is stored in the result variable with the primary key as the comorbidity name and the secondary key as the subcategory name. The value here is True or False depending on whether the subcategory of the comorbidity is present or not.

➔ parseRule():- Invokes the lexer, parser and evaluator related functionality on the rule being evaluated. The parseRule function takes as input a rule, the variables containing the values of all the lab dictionary parameters and processed Result, which contains the rules for previously defined comorbidities. The function calls the lexer with the rule as the input and extract tokens from the rule. It then creates an Abstract Syntax Tree (AST) by passing the tokens as input to the parse() function. The AST is then passed on as input, along with variables and processed Result to the evaluate() function, which analyzes and evaluates the conditions in the rule. It returns as the output - True or False depending on whether comorbidity was evaluated as being present or not respectively.

2. **Lexer: rule.interpreter.lexer.js**

   The lexer splits the incoming rules into tokens for further processing. The main functionality can be found in the following function/s:-

   ➔ lexer():- Generate tokens from the rule. This function initially declares all the tokens that the rule could possibly get divided into. The tokens are defined using Regular Expressions to standardize the format of each token. The incoming rule is split based on the'\s' (,i.e.,space character) to get all the tokens. It defines several utility functions to classify the token type as number or identifier, rule etc. All the tokens generated by splitting the rule are then iterated over and passed to the classifying functions to create a list called tokens, where each element is a JSON object of key - type of token, and value - the token itself. The lexer also adds appropriate tokens for the any/or conditions. The list is returned to the calling function (parseRule(), in our case) for further processing.

3. **Parser: rule.interpreter.parser.js**

   Generate the Abstract Syntax Tree (AST) from the tokens for evaluation and interpretation. The main functionality is as follows:-

   ➔ parse():- The parser() function takes as input the generated tokens from the lexer function call and creates a list of expressions of the format of <Left Hand Side> <Operator> <Right Hand Side>. To note, each of LHS and RHS can be individual expressions. The parser utilizes the concepts of Null Denotation (nud), Left Denotation (led) and Left Binding Power(lbp). Every operator token gets associated with an lbp attribute based on the preference according to standard arithmetic rules, so that the parsing can happen and the expression is created as intended by the rule creator. The nud and led are functions associated with the tokens. The nud method gets associated with tokens of type - Number and prefix operators. The led function gets associated with the infix and suffix operator tokens. The parser invokes the nud function on the first token. The lbp of the next token is then checked. If it is as at least as large as the given binding power, then the led function for that token gets called. Operator tokens have higher binding power and led methods invoke the parses again to get the result of subexpressions. These are used with the left value (if it was of type Number) to compute a final left. The parsing of the expression continues till the end is reached. The ending of the token list is represented here as the (end) token.The parse tree generated in this manner is returned as output.

4. **Evaluator: rule.interpreter.evaluator.js**

   ➔ evaluate():- The evaluate function takes the list of expressions generated from the parse() function. It iterates over each expression and resolves the operation based on the type of the operator in the expression by using the parseNode() function.

➔ parseNode():- The parseNode() function takes in the node as input, and based on the node type, evaluates the value of the node. To evaluate an expression, it recursively calls parseNode() function on the left node and the right node to get the respective values. It applies the arithmetic operation on the corresponding values to get the output. It should be noted that the left and the right node of an expression can be a complex expression in itself, but because of the recursive nature of the calls, the parseNode() function is able to successfully evaluate the outcome of the expression.

Moreover, the parseNode() function also takes an input parameter 'anyKLevel', which is specific to OR(||) operations. This is to be used in cases where the user specifies 'any 2/3/4/N of the following conditions are true' in case of comorbidity rules. The evaluation for this is slightly different and was done by maintaining a count of the number of children expressions that are evaluated as true. Every time a child expression is evaluated to be true, the count is decremented by 1. Finally, the count is compared against 0 and i t the parent expression is True only if the final count is less than or equal to 0.

5. **Display**

The earlier version of the rule builder just displayed the list of comorbidities highlighting the comorbidities shown by the patient in blue font. The new version displays the values and the rule by which the comorbidity was determined when the user hovers over the comorbidity. When the rules are processed and the patient is diagnosed for comorbidities, in the return value along with the main category and subcategory of comorbidities detected the rule is also returned. Later, the rule is parsed and the lab variables are identified thereby displaying the values of the lab variables and the actual rule itself.

➔ formatRule():- The formatRule function gets called from the processRule function after the lexer, scanner and evaluator have completed their phases and the presence or absence of comorbidity has been detected. The rule is then formatted to identify the lab variables used in the rule and the corresponding lab values are appended to the lab variables thereby creating the "Values" section of the display. The generated values part of the display and the actual rule is appended to the return value of the processRules function for each comorbidity.

<u>Cerner Integration</u>

Comorbidity.js : This file links the Cerner application to the Rule Builder tool.

Adding a new rule to the integrated system is described as follows:-

➢ The rules are specified in JSON format at the beginning of the file. Care must be taken while writing the rules, any incorrect syntax or extra spaces or missing brackets can render an error. Before testing for any new rule, the rule must be added to this JSON variable.

➢ All the variables that are used to verify the rules are populated with the values received from the reports.

➔ buildCCBody() In the function buildCCBody, we declare a dictionary variables that stores the values of the rules' variables in one place. This is used at a later time to check all rules and return the rules that satisfy the conditions. This dictionary must be populated with the variables present in the rule you want to add.

➔ processRules() The buildCCbody function invokes the processRules function. This function lies within the Interpreter code and is used to update the variables used in our rule verification. The update includes checking for minimum or maximum constraints, calculating a third value from two given values and checking for null values. Error handling for any new rules must be taken care of here. It returns the updated 'variables' dictionary.

At the end of the processRules function call, we get a result dictionary that contains the disease name, its subcategories (if present) and the result of the evaluation. Next, we go through each rule one by one, and check for the variables that are present in this rule.

- If a rule has a subcategory, its result type will be an object and not a Boolean. In that case, we iterate through the subcategories and see if any of them are true; if yes, we display them in blue, otherwise in black.

- If a rule doesn't have a subcategory, we simply check the Boolean value; if it is true, we display the disease in the final display in blue, otherwise we don't display it at all. The display code is common and similar for all the disease types. We basically add rows to a table body and add checkboxes to denote if the disease is present.

Testing

The testing of the rule builder functionality was done in a controlled environment (MOCK server) separate from a production server. A sample set of 70 patients were considered. These patients were diagnosed with a large number of comorbidities, combinations of multiple comorbidities included as well. The output of the rule builder for the list of comorbidities was cross validated against the existing comorbidity diagnosis system that is in use. The format of the output generated by this rule builder was also matched against the existing format to maintain consistency across the application. Other details of each comorbidity such as the lab dictionary values, date recorded and the rule used to diagnose the comorbidity are also displayed as a mouse hover over functionality.

Conclusion

So far in this project, the rule builder application has been implemented in completion. It allows users to create new rules, with a user friendly interface, providing facilities to read the rules while they get created, to select lab values from an existing dictionary, as well as to export the rules in JSON format. Additionally, the user can view as well as edit an existing rule, using the View/Edit tab. The tool has been tested locally on all rules.
Also, the rule builder code has been integrated with the Cerner system's code. When selecting a patient's data, one can view the rule builder results by clicking the 'Rule Builder' tab on the left navigation pane. Results are displayed in the exact same format as Cerner displays results, so that there are no variations. Testing for this has also been done on 70 patients, as described in section 3.

Future Work

Currently, the rules JSON for the comorbidities generated from the rule builder is read statically by the code of the CERNER system. We need to come up with a mechanism that allows the rules JSON created from the rule builder to persist globally in the system. Additionally, as a part of the CERNER system, we need to implement a mechanism for the system to read these persistent rules JSON on the fly.

Code Repository Link

➔ The code for the rule builder integrated with the CERNER system can be found at the path: https://bitbucket.org/jugudannie/cernerrulebuilder

➔ The code for the standalone version of the rule builder (to run on a local setup) can be found at the path :https://bitbucket.org/ankitmit/rulebuilder

➔ The web version of the rule builder with the display can be found at the path https://smohan24.github.io/rule-builder/

➔ The code for the rule builder with changes for display can be found at the path https://bitbucket.org/hbusireddy/sbcomorbidity/src/951ccd5d3ba0500124ccd6dd54d3b87569015320/rulebuilder_display/?at=master

References

[1] "Top Down Operator Precedence - Douglas Crockford's Javascript." 21 Feb. 2007, http://javascript.crockford.com/tdop/tdop.html. Accessed 23 Dec. 2016.

[2] Simple Top-Down Parsing
http://effbot.org/zone/simple-top-down-parsing.htm

There are two other documents for setting up a VPN to access Cerner. They are present in the same repository.

- Comorbidities - Documentation - Sriganesh Navaneethakrishnan
- Comorbidities Project Documentation