Language Specification for SUGAR

Health Informatics 3

May 11, 2015

1 Introduction

This document will describe the language used to describe analyses in our program. The language should be simple and expressive. This will ensure the best usability and improve the user experience.

We've opted to name her SUGAR, which stands for Simple Usable Great Analysis Reasoner.

2 Basic Idea

We like the BASH approach to piping input from one process to another, so we'll be borrowing the pipe | command. Don't worry, we'll give it back. Parentheses are great to use for grouping things like parameters and the ; is great for closing things.

3 Examples

Let's start with something like:

$$from(statSensor)|constraints(idCheck)|is(person1)|save(\text{"}person1.txt\text{"});$$

Here we run into a couple of things, first of all we have our commands (from, constraints etc.). These are straight forward and should pose no issues. They always have their parameters between parentheses (if they have no parameters parentheses are still required).

Next we have our pipes, which we have described in the previous section.

We also have some identifiers: statSensor, idCheck etc. these should pose any problems either as long as they don't contain symbols such as parentheses or spaces. We do find the need to describe the contraints for the constraints analysis, so will be declaring macros later on.

Furthermore we find the string literal, which should be trivial.

Last of all we have a closing ;.

4 Macros

Macros will be useful to describe variables and other things. This will simplify the life of the user, since things like constraints, which could take a couple of

lines to declare, won't be mixed with the processing chain.

Let's begin with another example:

$def\ idCheck(measurement) : Constraint = measurement.id = 1\ AND\ (measurement.level > 5)\ OR\ NOT(true)\ AND\ (measurement.level > measurement.value)$;

Let's say all macros start with def. This will simplify all our lives.

Next we declare the identifier, which in this case is idCheck.

Then we declare the type of macro, in this case it is Constraint, although we expect things like Process, Chunk, what have you. This is seperated by a =.

Next we have our constraint, which is a subject for another section.

5 Constraints

Constraints can be somewhat complicated and rightfully so, therefore the declaration should be powerfull, but not overly complicated.

In the previous example we saw a constraint, let's take a closer look.

In the $measurement.id = 1$ we declare that for the given table the column id should be 1, simple enough.

In the $measurement.level > 5$ we declare that level should be larger than 5.

With $table.column <= 2$ and $table.column >= 2$ we declare that the table's column should be less/larger or equal to 2.

Next we have the AND, OR and NOT statement. With AND we declare that both constrains should hold. For example when we have $table.id > 2 AND Dtable.id < 4$, than the id should be larger than 2 and less than 4. With OR we declare that one or more constrains should hold. So when we have $table.name = "Matthijs" ORtable.name = "Bob"$, than name must be "Matthijs" or the name must be "Bob".

6 Computation

We should support basic operations such as + - * / pow() sqrt().

We should also feature some way to do time operations, such as time intervals etc. + - days min hours etc.

6.1 Functions

For aggregates we should support MAX, MIN, AVG, COUNT, SUM, MEDIAN, STDDEV

7 Chunking

Chunking is just a special kind of constraint. We should support the same operation $><>=<= ANDORNOT$

8 Connections

Connections should be directed and should be capable of being many to many. This means implementing such operators:

(First,) Results, Triggered by, Constraints operands.

The first means that we only connect one row to another. The results and triggered by are to be used when checking if there are any connections. The

This results in something like this:
$def\ measurementInput(measurement, input) : Connection = FIRST\ input\ WHERE\ input.time < measurement.time\ AND\ input.date >= measurement.date\ ORDER\ input.date, input.time\ ASC$

$connection(measurementInput, statSensor, website)|is(connected)$

9 Coding

Codings are quite complicated. We want to cover all possible combinations of events. Let's say something like this: $def\ \ input(input, measurement)\ :\ Coding = IF\ input\ WHERE\ measurement.time < (input.time - 5min)\ AND\ measurement.date = input.date\ STORE\ "MI", input.time, input.date$

$coding(input, website, statSensor)|append(coded)$

To code the taking of a measurement and immediately entering it in the website.

So you specify IF there is a certain row, for which there is another row or multiple other rows and then you output a code and certain other values. There should be a way to connect rows and the code back together afterwards.

10 Comparison

TBD

11 Conversion

TBD