Cympha™

Language that Yields Medical Process HAndling

SYNTAX

May 22, 2017

Rickard Verner Hultgren *rihu0003@student.umu.se*

1 Introduction

The possibilities of treating diseases increase in a rapid speed. Consequently both diagnostics and treatment become more complex. The purpose of LYMPHA is to give clarity to those complex reasonings. LYMPHA is a logical language for formulating medical algorithms. The language can be applied as *clinical decision support system*.

The clinical work flow is dealing with series of events effecting the patients condition and evaluations of the condition. In LYMPHA this is put as *statements* devided by -> building up a *serie*. A serie ends with semicolon as follows:

statement -> *statement* -> *statement* ;

CODE (REGEX)

```
[^;]*(?=;)
```

CODE (PYTHON)

```
[x.strip() for x in serie.split("->")]
```

A statement has either value of true (1) if it is executed, or false (0). A statemtent is further devided into these parts:

variable name (*specification*) = *content*

CODE (REGEX)

```
(\w*[a-zA-Z]\w*\.)\(^[^\)]*?(?=\))?\s*?=\s*?(^[^;]*?(?=;))
```

Events effecting the condition includes ethymology, diagnostics and treatment. E.g. a bone frature that is examined and then treated. Some medical procedures are both diagnostical and treatment. Nevertheless these are events, that are semanticly separated from the evaluation of data from an event. The evaluation does oftne consist of sub-evaluation that will become factors in the main-evaluation: Therefore there are these two *datatypes* in LYMPHA: *events* and *factors*.

2 Variable name

Variable names are one word with at least one letter. The last character of the name indicates what datatype the variable has.

- *Events*; Always end with a full stop (.).
- *Factors*; Always end with a questionmark (?).

Even though both datatypes have a structure of an element as explained above, the parts looks a little bit different.

3 Specifications

A specification is a sub-variable. It is built as follow:

datatype [*sub-variable name*] = *content*

CODE (REGEX)

```
(\w*[a-zA-Z]\w*\)\[^[^\]]*?(?=\])?\s*?=\s*?(\w*[a-zA-Z]\w*\)
```

Here follows how structurize the parts:

3.1 Property

These properties are used:

Dataype	*Meaning*
L	length
L2	surface area
L3	volume
M	mass
N	mole
-T	time elapsed
%s	string
R	other real numbers

3.2 Sub-variable name

These is the name or unit of the data.

3.3 Sub-variable content

This is the value. All strings has quoationmarks at the beginning and at the end.

4 Content

These is the part of a statement that separates factors from events. The goal of a factor content is to evaluate data. The goal of event content is to descripe sub-events.

4.1 Event content

This is one serie of events and factors separted with commas ,:

{ *event* , *factor* , *event* }

CODE (PYTHON)

```
[x.strip() for x in content.split(',')]
```

4.2 Factor content

This is an evaluation of other sub-factors. Sub events are not allowed to be included in factor content.

tipping point *relational operator* |{ *sub-factor* , *sub-factor* }|

CODE (REGEX)

```
(\d)\s*?(==|>|<|>=|<=)\s*?{(\w*[a-zA-Z]\w*\)}
```

Valid rational operators

relational operator	*read as*
==	if and only if (≡)
>	greater than
>=	greater than or equal (≥)
<	lesser than
<=	less than or equal (≤)
!=	not equal to (≠)

5 Datatypes

The goal of the syntax is to build asignments for the two datatypes. Here are the type members for each datatype:

- Events; Always end with a full stop (.).
  - *string name*
  - *list specifications*
  - *list content*
- Factors; Always end with a questionmark (?).
  - *string name*
  - *list specifications*
  - *list content*
  - *int tipping point*
  - *int relational operator*