RockIt: Numerical Extension

Documentation

written by
Jakob Huber
jakob @informatik.uni-mannheim.de

Data & Web Science Group
University of Mannheim

February 2015

Chapter 1

Numerical Extension

The branch `n-ext`[1] extends ROCKIT with features that are useful in the concrete domain of numbers. In particular, the extensions provides support for defining predicates whose truth values depends on a boolean expression that is composed of algebraic expressions. It also allows to replace simple variables with an algebraic expression that is used to compute the actual ground value of a variable.

We use EVALEX[2] in order to compute the values of Boolean expressions and algebraic expressions. Hence, all operators and functions of EVALEX can be used to state an expression. However, all expressions need to be bracketed:

[*expression* **]**

Note that the extension is only tested with MAP inference and MLNs containing hard- and soft formulas, but no cardinality formulas.

1.1 Numerical Predicates

The truth value of *numerical predicates* depend on boolean expressions that are composed of algebraic expressions. Hence, all variables of such predicates need to be numerical (e.g. integer or double). A Boolean expression is associated with predicate using the character sequence `:=`. The predicate definition for *numerical predicates* does not state the type of the different variables as they are implicitly considered as numerical predicates (e.g. double). The predicates can be viewed as evidence as their truth values is observed.

[1] `https://rockit.googlecode.com/svn/branches/n-ext`
[2] `https://github.com/uklimaschewski/EvalEx`

We provide an example that uses *numerical predicates* in Listing 1.1, 1.2 and 1.3. The third variable of the predicate `sumP` respectively `sum` should be the sum of the first two variables in order that the grounded predicate is true. Hence, we associate the respective constraint with `[x+y == z]` with the predicates `sum`. The evidence contains two groundings of the predicate `sumP` which is the weighted version of the predicate `sum`. The MAP state contains only one grounding as the other one does not fulfill the constraint of `sum`.

Listing 1.1: Example for numerical predicates: prog.mln

```
*sumP(n, n, n, float_)
sum(x,y,z) := [x+y == z]

conf: !sumP(x,y,z,conf) v sum(x,y,z)
```

Listing 1.2: Example for numerical predicates: evidence.db

```
sumP("1.0", "1.0", "2.0", 1.0)
sumP("1.0", "1.0", "1.0", 1.0)
```

Listing 1.3: Example for numerical predicates: out.db (MAP state)

```
sum("1.0", "1.0", "2.0")
```

1.2 Algebraic Expressions for Variables

The second part of the extension allows to replace a variable with an *algebraic expressions* which allows to infer new numbers based on numerical values occurring as ground values of other variables in the formula.

We provide an example that uses *algebraic expressions* in Listing 1.4, 1.5 and 1.6. The example computes all sums of the numerical values (predicate `val`) given as evidence. The predicate `sum` states the sum of the first variables at its third position.

Listing 1.4: Example for algebraic expressions: prog.mln

```
*val(int_)
sum(int_,int_,int_)

!val(x) v !val(y) v sum(x, y, [x+y]).
```

Listing 1.5: Example using algebraic expressions: evidence.mln

```
val("1.0")
val("2.0")
```

Listing 1.6: Example for algebraic expressions: out.db (MAP state)

```
sum("1.0", "1.0", "2.0")
sum("1.0", "2.0", "3.0")
sum("2.0", "1.0", "3.0")
sum("2.0", "2.0", "4.0")
```