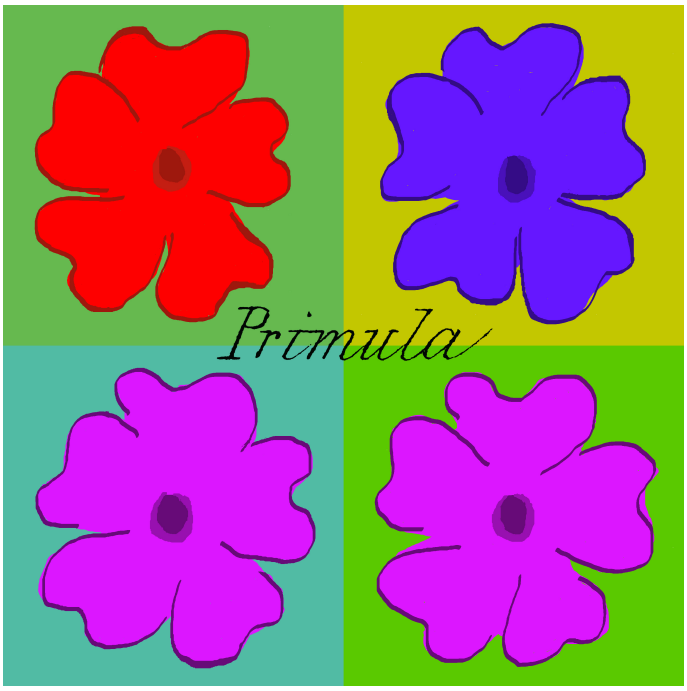
The *Primula* System: user's guide
Version 3.0
RBN syntax

Manfred Jaeger
Institut for Datalogi, Aalborg Universitet, Selma Lagerløfs Vej 300, 9220 Aalborg Ø
jaeger@cs.aau.dk

Primula homepage: www.cs.aau.dk/∼jaeger/Primula

September 15, 2021



Syntax

Names

Throughout, a *name* is an alphanumeric string that starts with a letter, and that may also contain hyphens '-' and underscores '_'. Angles $<, >$ are meta-symbols used to specify general syntactic components.

Definitions

A `.rbn` file consists of a sequence of *macro definitions* and *probabilistic relations definitions*. Both have essentially the same form of an equation between a relational atom and a probability formula, terminated by a semicolon. The only difference is that a macro definition is preceded by '@':

Macro definition:

`@` $<$ *TypedAtom* $>$ = $<$ *ProbabilityFormula* $>$`;`

Probabilistic relation definition:

$<$ *TypedAtom* $>$ = $<$ *ProbabilityFormula* $>$`;`

An *TypedAtom* is a relation name followed by a list of variables. The variables can be preceded by type specifications. These type specifications must correspond to pre-defined unary relations in the input structure to which the rbn is applied. If no type specification is given, then the relation specified by this atom is applied to all objects in a domain. Examples:

```
male(v)
male([person]v)
edge([node]v,[node]w)
likes([user]u,[movie]m)
test()
```

The type specifications only have an effect in the atoms of probabilistic relations definitions. They can also be used in macro definitions to improve readability. Atoms can have any number ≥ 0 of arguments. A relation of arity 0 represents a global property of the input structure. For example, the global `test` relation could represent that an input structure is an element of a test set.

Probability Formulas

Probability formulas are formed using macro calls and 4 different constructs.

Macro call

If the `.rbn` contains a macro definition

`@my_macro(x,y)` = < *ProbabilityFormula* >;

then a macro-call is just an expression of the form `@my_macro(u,v)`. It represents the probability formula obtained by substituting the arguments $(\mathtt{u}, \mathtt{v})$ in the macro call for the arguments `(x,y)` in the probability formula on the right-hand side of the macro definition.

Constants

A constant is a fixed numerical constant, or a parameter identifier. Parameter identifiers are names prefixed with either '#' or '$'. The '#' prefix denotes a parameter that is constrained to represent a probability value in the interval $[0, 1]$, whereas '$' prefixes parameters that stand for arbitrary real values. Examples for constants:

```
0.5
-2.6
#success_probability
$weight
```

Atoms

Atoms have the same syntax as the typed atoms above, but without type constraints. The arguments of atoms can only be strings representing variables, no constants representing concrete domain objects.

Convex Combinations

A convex combination, alternatively *wif-then-else* construct, has the form

`WIF` < *ProbabilityFormula* >
`THEN` < *ProbabilityFormula* >
`ELSE` < *ProbabilityFormula* >

Combination Functions

The central construct is the combination function formula:

`COMBINE` < *ListOfProbabilityFormula* >
`WITH` < *CombinationFunction* >
`FORALL` < *ListOfVariableNames* >
`WHERE` < *BooleanFormula* >

A < *ListOfProbabilityFormula* > is a comma separated list of probability formulas.

A < *CombinationFunction* > is one of the strings listed in the syntax column in the following table. Each combination function is a function that maps arbitrary multi-sets $\boldsymbol{x} = \{x_1, \ldots, x_n\}$ of reals to a real number. In the "Signature" column of the following table R stands for the reals, and I for the interval $[0, 1]$. The expression $I^* \to I$ then states that the function maps multisets of probabilities to a probability value. $R^* \to I$ thus is the signature of a "squashing" function, that maps multisets of arbitrary numbers to a probability.

Syntax	Name	Signature	Definition	For $\boldsymbol{x} = \emptyset$
`n-or`	noisy-or	$I^* \to I, R^* \to R$	$1 - \prod_i (1 - x_i)$	0
`mean`	mean	$I^* \to I, R^* \to R$	$(\sum_i x_i)/n$	0
`esum`	exponential sum	$I^* \to I, R^* \to R$	$exp(-\sum_i x_i)$	1
`invsum`	inverse sum	$R^* \to I$	$1/max\{1, \sum_i x_i\}$	1
`l-reg`	logistic regression	$R^* \to I$	$1/(1 + exp(-\sum_i x_i))$	0.5
`sum`	sum	$R^* \to R$	$\sum_i x_i$	0
`prod`	product	$I^* \to I, R^* \to R$	$\prod_i x_i$	1

A < *ListOfVariableNames* > is a comma separated list of names for variables.

A < *BooleanFormula* > can be either

- the constants `true` or `false`
- an atom in one of the predefined relations (no references to probabilistic atoms are possible in the `WHERE` clause of a combination function)
- an equality of the from
  < *VariableName* > = < *VariableName* >
- a conjunction of the form
  `(` < *BooleanFormula* > `&` < *BooleanFormula* >`&` ... `&` < *BooleanFormula* > `)`
- a disjunction of the form
  `(` < *BooleanFormula* > | < *BooleanFormula* > |... | < *BooleanFormula* > `)`
- a negation of the form
  ∼ < *BooleanFormula* >

An input file may contain comment-lines starting with %.