iRODS Rule Language Cheat Sheet
iRODS Version 4.0.3
Author: Samuel Lampa, BILS

Numeric Literals

```
1 # integer
1.0 # double
```

Strings

Concatenation:

```
'A \'string\', ' ++ "another \"string\"
```

Some valid escape characters:

```
\n, \r, \t, \\, \', \", \$, \*
```

Boolean constants

```
true # True
false # False
```

Boolean comparison

```
! # Not
&& # And
|| # Or
%% # Or used in the "##" syntax
```

Arithmetic operators

```
- # Negation
^ # Power
* # Multiplication
/ # Division
% # Modulors
- # Subtraction
+ # Addition
```

Arithmetic comparison

```
> # Greater than
< # Less than
>= # Greater than or equal
<= # Less than or equal
```

Arithmetic functions

```
exp(<num>)
log(<num>)
abs(<num>)
floor(<num>) # always returns integer
ceiling(<num>) # always returns integer
average(<num>, <num>, ...)
max(<num>, <num>, ...)
min(<num>, <num> , ...)
```

String functions

```
writeLine("stdout", "Hi!");
```
... prints out "Hi!."

```
"This "++"is"++" a string."
```
... equals to "This is a string."

```
"This is a string." like "This is*"
```
... equals to `true`

```
"This is." like regex "Th.*is[.]"
```
... equals to `true`

```
substr("This is a string.", 0, 4)
```
... Output: This

```
strlen("This is a string.")
```
... Output: 17

```
split("This is a string.", " ")
```
... equals to: `[This,is,a,string.]`

```
writeLine("stdout", triml("This is a string.", " "));
```
... equals to: is a string.

```
trimr("This is a string.", " ")
```
... equals to: This is a

List functions

```
list(<elem>, <elem>, ...)
```
... creates a new list. Example:

```
list("This","is","a","list")
```

```
elem(<list>, <index>)
```
... retrieves elements from a list (0-indexed). Ex:

```
elem(list("This","is","a","list"),0)
```
... returns "This"

```
setelem(<list>, <index>, <value>)
```
... updates an item in a list. Ex:

```
setelem(list("A","list"),0,"My")
```
... evaluates to `list("My","list")`.

```
size(<list>)
```
... gives the size of a list. Ex:

```
size(list("This","is","a","list"))
```
... evaluates to 4.

```
hd(<list>)
```
... gives the head of a list, Ex:

```
hd(list("This","is","a","list"))
```
... evaluates to "This"

```
tl(<list>)
```
... gives the tail of a list. Ex:

```
tl(list("This","is","a","list"))
```
... evaluates to list("is","a","list")

```
cons(<element>, <list>)
```
... adds elements to a list. Ex:

```
cons("My",list("list"))
```
... evaluates to list("My","list").

Tuples

Tuples are created like so:

```
( <component>, ..., <component> )
```

If statements

Logical if:

```
if <expr> then { <actions> } else { <actions> }
```

Example:

```
if (*A==1) then { true; } else { false; }
```

Functional if (returning value of any type):

```
if <expr> then <expr> else <expr>
```

Example:

```
if true then 1 else 0
if *A==1 then true else false
```

The following abbreviation are allowed (the red striked part can be abbreviated) in functional ifs:

```
if (...) then { ... } else { ... }
if (...) then { ... } else { if (...) then {...} else {...} }
```

Multiple abbreviations can be combined for example:

```
if (*X==1) { *A = "Mon"; }
else if (*X==2) {*A = "Tue"; }
else if (*X==3) {*A = "Wed"; }
```

Foreach loops

Without iterator:

```
foreach(*C) {
    writeLine("stdout", *C);
}
```

With the iterator variable (*E in this case):

```
foreach(*E in *C) {
    writeLine("stdout", *E);
}
```