reg - creates an automaton based on a specified regular expression

Usages: reg <name> <number system> … <number system> <regular expression>
reg <name> <alphabet> … <alphabet> <regular expression>

Results saved in: Result, Automata Library

In previous versions of Walnut, the regular expression consisted of regular expression operations such as OR (|), concatenation, Kleene star (*) and others, on an alphabet of 0-9. This has been extended to arbitrary tuples of integers. Now one can specify, for example, [1, -1, 1][0, 1, -1]* to mean 10* in the first coordinate, -11* in the second, and 1-1* in the third. In particular, one can now specify numbers above 9 and below 0, provided brackets are used (eg. 1[10]* is 1 followed by an arbitrary number of 10's and similar for 1[-1]*). Brackets may also be used for numbers 0-9 (eg. [1][10]*) but are not mandatory in this case. A number system or alphabet must be supplied for each coordinate in the expression's tuples.

combine - produces a DFAO whose output on a given input corresponds to the highest index automaton in the list supplied that accepts said input

Usage: combine <name> <automaton exp> … <automaton exp>

Results saved in: Result, Word Automata Library

An automaton expression is either the name of an automaton on its own, (eg. myAutomaton) or the name with a value assigned by an equals sign (eg. myAutomaton=3). Walnut assigns a default value equal to the index of the automaton in the list, beginning with 1. For example, "combine A A1 A2=10 A3" produces the same output as "combine A A1=1 A2=10 A3=3". This output is a DFAO called A that outputs 0 if none of A1, A2, or A3 accepts an input, 1 if A1 accepts but A2 and A3 do not, 10 if A2 accepts but A3 does not, and 3 if A3 accepts.

morphism - produces a morphism object from its functional definition

Usage: morphism <name> "letter -> string, … , letter -> string"

Results saved in: Result, Morphism Library

Walnut will define a morphism that can be used in other operations. For example, "morphism thue "0->01 1->10"" defines the Thue-Morse morphism which maps 0 to 01 and 1 to 10.

promote - converts a morphism to its equivalent DFAO

Usage: promote <name> <morphism>

Results saved in: Result, Word Automata Library

For example, "promote thueAutomaton thue" will create a DFAO called thueAutomaton equivalent to the morphism from the previous example. This automaton consists of two states, 0 and 1, such that 0 outputs 0, 1 outputs 1, and 0 goes to 0 on 0 and 1 on 1, while 1 goes to 1 on 0 and 0 on 1. Any morphism used must not have negative values in the domain or range, since automata have states numbered from 0.

image - applies a uniform morphism to a DFAO to produce a new DFAO

Usage: image <name> <morphism> <DFAO>

Results saved in: Result, Word Automata Library

This application proceeds as Cobham outlined in his 1972 paper. If the morphism supplied is not uniform, an error will be produced.

inf - Determines whether or not an automaton accepts infinitely many inputs. If it does, returns a regular expression defining an infinite family of accepted inputs.

Usage: inf <name>

Results saved in: N/A

Leading/trailing zeroes are removed (depending on whether the automaton is msd or lsd). The function searches for a cycle, and if one is found, constructs a prefix from q0 to it, and from it to an accepting state, such that the prefix and suffix are minimal for the particular cycle. This works with multiple arity as well.

test - Outputs the first n values accepted by an automaton, in lexicographic order, where n is a number supplied by the user.

Usage: test <name> <number>

Results saved in: N/A

If fewer than n values are accepted, all accepted values are printed in lexicographic order.