COUNT$numbers

Give a regular expression for all binary numbers with an even number of "1"
ISEVEN(COUNT(STRING(1)))
additions -**##**- [['regular', 'ISEVEN', 1.0], ['give', 'COUNT', 1.0]]

finite automata which accept sets of strings composed of zeros and ones which possess runs of even numbers of "0" and odd numbers of "1"
AND(ISEVEN(COUNT(STRING(0))), ISODD(COUNT(STRING(1))))
additions -**##**- [['odd', 'ISODD', 1.0], ['even', 'ISEVEN', 1.0], ['accept', 'COUNT', 1.0], ['and', 'AND', 1.0]]

--
EQ$same

The set of all strings over RegX that have the same number of occurrences of the substring "01" as of the substring "10".
BoolCondition(MATCHFORMAT(EXPRESSION(RegX)),
EQ(COUNTBOTH(ANDSTR(STRING(01), STRING(10)))))
additions -**##**- [['over', 'MATCHFORMAT', 1.01], ['regx', 'EXPRESSION', 1.01], ['as', 'EQ', 1.01], ['number', 'COUNTBOTH', 1.01], ['as', 'ANDSTR', 1.01]]

Strings that contain the same number of "01" as "10".
EQ(COUNTBOTH(ANDSTR(STRING(01), STRING(10))))
additions -**##**- [['as', 'EQ', 1.01], ['contain', 'COUNTBOTH', 1.01], ['as', 'ANDSTR', 1.01]]

--
INTEQUALS$same

w has the same number of occurrences of "10" and "01"
INTEQUALS(COUNT(STRING(10)), COUNT(STRING(01)))
additions -**##**- [['has', 'COUNT', 1.0]]

--
SYMBOLATP$positions

The set of strings of "0" and "1" such that at least one of the last #10 positions is a "1"
EXISTSINT(LASTFEW(INTEGER(10)), STREQUALS(SYMBOLATP(), STRING(1)))
additions -**##**- [['set', 'SYMBOLATP', 1.0], ['is', 'STREQUALS', 1.0], ['last', 'LASTFEW', 1.0], ['least', 'EXISTSINT', 1.0]]

--
MATCHFORMAT$form

A language with words of form RegX
MATCHFORMAT(EXPRESSION(RegX))
additions -**##**- [['language', 'MATCHFORMAT', 0.5], ['regx', 'EXPRESSION', 1.0]]

A language with words of form RegX
MATCHFORMAT(EXPRESSION(RegX))
additions -**##**- [['language', 'MATCHFORMAT', 0.5], ['regx', 'EXPRESSION', 1.0]]

A language with words of form RegX
MATCHFORMAT(EXPRESSION(RegX))
additions -**##**- [['language', 'MATCHFORMAT', 0.5], ['regx', 'EXPRESSION', 1.0]]

A language with words of the form RegX
MATCHFORMAT(EXPRESSION(RegX))
additions -**##**- [['language', 'MATCHFORMAT', 0.5], ['regx', 'EXPRESSION', 1.0]]

A language with words of the form RegX
MATCHFORMAT(EXPRESSION(RegX))
additions -**##**- [['language', 'MATCHFORMAT', 0.5], ['regx', 'EXPRESSION', 1.0]]

A language with words of the form RegX
MATCHFORMAT(EXPRESSION(RegX))
additions -**##**- [['language', 'MATCHFORMAT', 0.5], ['regx', 'EXPRESSION', 1.0]]

Design a linear bounded automaton which accepts strings of the form RegX.
MATCHFORMAT(EXPRESSION(RegX))
additions -**##**- [['a', 'MATCHFORMAT', 0.25]]

--
CONTAINSP$substrings

w is a binary string containing both substrings "010" and "101"
CONTAINSP(ANDSTRINGS(STRING(010), STRING(101)))
additions -**##**- [['containing', 'CONTAINSP', 1.0], ['both', 'ANDSTRINGS', 1.0]]

--
CONTAINSP$has

w has neither "aa" nor "bb" as a substring.
NOT(CONTAINSP(ANDSTRINGS(STRING(aa), STRING(bb))))
additions -**##**- [['neither', 'NOT', 1.0], ['w', 'CONTAINSP', 0.3333333333333333], ['nor', 'ANDSTRINGS', 1.0]]

Give a NFA that only accepts strings such that x either has the substring "01" or has the substring "021".
OR(CONTAINSP(STRING(01)), CONTAINSP(STRING(021)))
additions -**##**- [['or', 'OR', 1.0], ['substring', 'CONTAINSP', 1.0]]

--
CONTAINSP$including

Give a regular expression for all binary numbers including the substring "101"
CONTAINSP(STRING(101))
additions -**##**- [['substring', 'CONTAINSP', 1.0]]

--
CONTAINSP$having

Words over a, b having either #1 "b" or #2 consecutive "b"
CONTAINSP(ORSTRINGS(STRING(b), REPEAT(STRING(b), INTEGER(2), REQ())))

additions -**##**- [['consecutive', 'REPEAT', 1.0], ['or', 'ORSTRINGS', 1.0]]

The set of all binary strings having a substring "00" and ending with "01".
AND(CONTAINSP(STRING(00)), ENDSWITHP(STRING(01)))
additions -**##**- [['ending', 'ENDSWITHP', 1.0], ['substring', 'CONTAINSP', 1.0], ['and', 'AND', 1.0]]

The set of all binary strings having a substring "00" but not ending with "01".
AND(CONTAINSP(STRING(00)), NOT(ENDSWITHP(STRING(01))))
additions -**##**- [['not', 'NOT', 1.0], ['ending', 'ENDSWITHP', 1.0], ['substring', 'CONTAINSP', 1.0], ['but', 'AND', 1.0]]

--
CONTAINSP$have

w does not have "001" as a substring.
NOT(CONTAINSP(STRING(001)))
additions -**##**- [['not', 'NOT', 1.0], ['w', 'CONTAINSP', 0.333333333333]]

Consider the DFA that accepts all strings which have "01" as a substring.
CONTAINSP(STRING(01))
additions -**##**- [['strings', 'CONTAINSP', 0.5]]

Give a DFA such that it contains all strings that have "aba" as a substring
CONTAINSP(STRING(aba))
additions -**##**- [['contains', 'CONTAINSP', 1.0]]

--
CONTAINSP$includes

accepts any string that includes the sequence "abcba" within it.
CONTAINSP(STRING(abcba))
additions -**##**- [['string', 'CONTAINSP', 0.5]]