

```

jul 31, 06 9:18          expression-tests.txt          Page 1/8
//
// Tests expressions.
//
// Follows this format (not implemented as grammar, need to learn
// what the C# String API can accomplish!):
//
// Test = Formulas '%' Results [ '%' Options]
// Formulas = Formula {'@' Formula}
// Results = Result {'@' Result}
// Options = Option {'@' Option}
//
// Formula = Cell '=' formula expression
// Result = Cell '=' Expected string value
//          | Cell '=' Exception(String)
//
// Option = "skip" '=' "Level0".."Level7"
//          | "cols" '=' Integer
//          | "rows" '=' Integer
//          | "numberofsheets" '=' Integer
//          | "startlevel" '=' "Level0".."Level7"
//
// NB All skip, cols, rows... options can be substituted with
// globalskip, globalcols, globalrows, which makes the options
// global from that point onwards.
//
// Comments can appear with // and # in this file
// Empty lines are allowed too
//
// Thomas S. Iversen, 2006.

// ***** Addition
A1=5%A1=5
A1=5@A2=6@A3=A1+A2%A3=11
A1=5+6%A1=11
A1=5@A2=6@A3=7@A4=A1+A2+A3%A4=18
A1=5+6+7%A1=18
A1=5.5%A1=5.5
A1=5.5@A2=6.6@A3=A1+A2%A3=12.1
A1=5.5+6.6%A1=12.1
A1=5.5@A2=6@A3=A1+A2%A3=11.5
A1=5.5+6%A1=11.5
A1=2+-2%A1=0
A1=2+IF()%A1=#ERR: ARGTYPE
A1=2+"Thomas"%A1=#ERR: ARGTYPE
A2=5@A3=6@A1=2+A2:A3%A1=#ERR: ARGTYPE

// ***** Multiplication
A1=5@A2=6@A3=A1*A2%A3=30
A1=5*6%A1=30
A1=5.6@A2=7.8@A3=A1*A2%A3=43.68
A1=5.6*7.8%A1=43.68
A1=5.6@A2=1@A3=A1*A2%A3=5.6
A1=5.6*1%A1=5.6
A1=5.6@A2=0@A3=A1*A2%A3=0
A1=5.6*0%A1=0
A1=2*IF()%A1=#ERR: ARGTYPE
A1=2*"Thomas"%A1=#ERR: ARGTYPE
A2=5@A3=6@A1=2*A2:A3%A1=#ERR: ARGTYPE

// ***** Subtraction
A1=5@A2=6@A3=A1-A2%A3=-1

```

```

jul 31, 06 9:18          expression-tests.txt          Page 2/8
A1=6@A2=5@A3=A1-A2%A3=1
A1=-5@A2=-6@A3=A1-A2%A3=1
A1=6@A2=-5@A3=A1-A2%A3=11
A1=-6@A2=5@A3=A1-A2%A3=-11
A1=6@A2=0@A3=A1-A2%A3=6
A1=6.5@A2=5.6@A3=A1-A2%A3=0.9
A1=2--2%A1=4
A1=2-IF()%A1=#ERR: ARGTYPE
A1=2-"Thomas"%A1=#ERR: ARGTYPE
A2=5@A3=6@A1=2-A2:A3%A1=#ERR: ARGTYPE

// ***** Division
A1=5@A2=1@A3=A1/A2%A3=5
A1=5@A2=5@A3=A1/A2%A3=1
A1=5@A2=0@A3=A1/A2%A3=Infinity
A1=5@A2=0.1@A3=A1/A2%A3=50
A1=5@A2=10@A3=A1/A2%A3=0.5
A1=5.5@A2=5.5@A3=A1/A2%A3=1
A1=5.5@A2=55@A3=A1/A2%A3=0.1
A1=5.5@A2=1@A3=A1/A2%A3=5.5
A1=5.5@A2=-1@A3=A1/A2%A3=-5.5
A1=-5.5@A2=1@A3=A1/A2%A3=-5.5
A1=-5.5@A2=10@A3=A1/A2%A3=-0.55
A1=-5.5@A2=0.1@A3=A1/A2%A3=-55
A1=-5.5@A2=0@A3=A1/A2%A3=-Infinity
A1=2/IF()%A1=#ERR: ARGTYPE
A1=2/"Thomas"%A1=#ERR: ARGTYPE
A2=5@A3=6@A1=2/A2:A3%A1=#ERR: ARGTYPE

// ***** String concatenation
A1="Thomas"@A2="Maibrutt"@A3=A1&A2%A3=ThomasMaibrutt
A1="Thomas"@A2=5@A3=A1&A2%A3=#ERR: ARGTYPE%skip=Level1
A1="Thomas"@A3=A1&A2%A3=#ERR: ARGTYPE%skip=Level7
A1="Thomas"@A2="Maibrutt"@A3="Mathias"@A4=A1&A2&A3%A4=ThomasMaibruttMathias
A1="Thomas"&"Maibrutt"%A1=ThomasMaibrutt
A1="Thomas"&5%A1=#ERR: ARGTYPE
A1=2&IF()%A1=#ERR: ARGTYPE
A2=5@A3=6@A1=2&A2:A3%A1=#ERR: ARGTYPE

// ***** Precedence
A1=5+6*7%A1=47
A1=5+6*7/7%A1=11
A1=5+6/7*7%A1=11
A1=5+6/6*7%A1=12
A1=5/6*7/8%A1=0.7291666666666667
A1=6/(6*7)/7%A1=0.0204081632653061
A1=6-6*8%A1=-42

// ***** Power
A1=2^3%A1=8
A1=2^3^4%A1=4096
A1=2^(3^4)%A1=2.41785163922926E+24
A1=2^3+4%A1=12
A1=2^3*4%A1=32
A1=2*3^4%A1=162
A1=2^6/3%A1=21.33333333333333
A1=9/3^4%A1=0.1111111111111111

A1=2--3^4%A1=-79
A1="Thomas"^2%A1=#ERR: ARGTYPE
A1=2^"Maibrutt"%A1=#ERR: ARGTYPE

```

jul 31, 06 9:18	expression-tests.txt	Page 3/8
<pre> A1="Thomas"^^"Thomas"%A1=#ERR: ARGTYPE  // ***** PI A1=PI()%A1=3.14159265358979 A1=PI(5)%A1=#ERR: ARGCOUNT A1=PI(A2:A3)%A1=#ERR: ARGCOUNT A2=5@A1=PI(A2)%A1=#ERR: ARGCOUNT A1=PI("Thomas",5)%A1=#ERR: ARGCOUNT A2="Thomas"@A1=PI(A2)%A1=#ERR: ARGCOUNT  // ***** SIN A1=SIN()%A1=#ERR: ARGCOUNT A1=SIN("Thomas")%A1=#ERR: ARGTYPE  // speculative type deduction does not work at level7 // and will always assume that the types are correct. A2="Thomas"@A1=SIN(A2)%A1=#ERR: ARGTYPE%skip=Level7  A2=5@A1=SIN(A2)%A1=-0.958924274663138 A1=SIN(5)%A1=-0.958924274663138 A1=SIN(A2,5)%A1=#ERR: ARGCOUNT A1=SIN(1+4)%A1=-0.958924274663138 A1=SIN(1+"Thomas")%A1=#ERR: ARGTYPE A2="Thomas"@A1=SIN(1+A2)%A1=#ERR: ARGTYPE%skip=Level7 A1=SIN(1+"Thomas",7)%A1=#ERR: ARGCOUNT A1=SIN(A2:A2)%A1=#ERR: ARGTYPE A1=SIN(A2:A3)%A1=#ERR: ARGTYPE  // ***** Equal A1=5==5%A1=1 A1=6==5%A1=0 A1=-5==5%A1=0 A1=6==4+2%A1=1 A1="Thomas"==5%A1=#ERR: ARGTYPE A1="Thomas"+1==5%A1=#ERR: ARGTYPE A1=5@A2=5@A3=A1==A2%A3=1 A1=5@A2=6@A3=A1==A2%A3=0 A1="Thomas"@A2=6@A3=A1==A2%A3=#ERR: ARGTYPE%skip=Level7 A1=5@A2=6@A3=A1=="Thomas"+A2%A3=#ERR: ARGTYPE%skip=Level7 A1=5==A2:A3%A1=#ERR: ARGTYPE  // ***** LT A1=5&lt;5%A1=0 A1=6&lt;5%A1=0 A1=5&lt;6%A1=1 A1=-1&lt;6%A1=1 A1=-1&lt;-1%A1=0 A1=-1&lt;-2%A1=0 A1=-2&lt;-1%A1=1 A1="Thomas"&lt;-1%A1=#ERR: ARGTYPE A1=1&lt;"Thomas"%A1=#ERR: ARGTYPE A1="Maibritt"&lt;"Thomas"%A1=#ERR: ARGTYPE A1=1@A2=1@A3=A1&lt;A2%A3=0 A1=2@A2=1@A3=A1&lt;A2%A3=0 A1=1@A2=2@A3=A1&lt;A2%A3=1 // A2 == null, Level7 throws Exception A1=A2&lt;A2%A1=#ERR: ARGTYPE%skip=Level7 // A2,A3 == null, Level7 throws Exception A1=A2&lt;A3%A1=#ERR: ARGTYPE%skip=Level7 //Level7 returns 1 instead. A1=5&lt;"Thomas"@A2=A1&lt;5%A2=#ERR: ARGTYPE%skip=Level7 </pre>		

jul 31, 06 9:18	expression-tests.txt	Page 4/8
<pre> A1=5&lt;A2:A3%A1=#ERR: ARGTYPE  // ***** LTE A1=5&lt;=5%A1=1 A1=6&lt;=5%A1=0 A1=5&lt;=6%A1=1 A1=-1&lt;=6%A1=1 A1=-1&lt;=-1%A1=1 A1=-1&lt;=-2%A1=0 A1=-2&lt;=-1%A1=1 A1="Thomas"&lt;=-1%A1=#ERR: ARGTYPE A1=1&lt;="Thomas"%A1=#ERR: ARGTYPE A1="Maibritt"&lt;="Thomas"%A1=#ERR: ARGTYPE A1=1@A2=1@A3=A1&lt;=A2%A3=1 A1=2@A2=1@A3=A1&lt;=A2%A3=0 A1=1@A2=2@A3=A1&lt;=A2%A3=1 // A2 == null, Level7 throws Exception A1=A2&lt;=A2%A1=#ERR: ARGTYPE%skip=Level7 // A2,A3 == null, Level7 throws Exception A1=A2&lt;=A3%A1=#ERR: ARGTYPE%skip=Level7 //Level7 returns 1 instead. A1=5&lt;="Thomas"@A2=A1&lt;=5%A2=#ERR: ARGTYPE%skip=Level7 A1=5&lt;=A2:A3%A1=#ERR: ARGTYPE  // ***** GT A1=5&gt;5%A1=0 A1=6&gt;5%A1=1 A1=5&gt;6%A1=0 A1=-1&gt;6%A1=0 A1=-1&gt;-1%A1=0 A1=-1&gt;-2%A1=1 A1=-2&gt;-1%A1=0 A1="Thomas"&gt;-1%A1=#ERR: ARGTYPE A1=1&gt;"Thomas"%A1=#ERR: ARGTYPE A1="Maibritt"&gt;"Thomas"%A1=#ERR: ARGTYPE A1=1@A2=1@A3=A1&gt;A2%A3=0 A1=2@A2=1@A3=A1&gt;A2%A3=1 A1=1@A2=2@A3=A1&gt;A2%A3=0 // A2 == null, Level7 throws Exception A1=A2&gt;A2%A1=#ERR: ARGTYPE%skip=Level7 // A2,A3 == null, Level7 throws Exception A1=A2&gt;A3%A1=#ERR: ARGTYPE%skip=Level7 //Level7 returns 1 instead. A1=5&gt;"Thomas"@A2=A1&gt;5%A2=#ERR: ARGTYPE%skip=Level7 A1=5&gt;A2:A3%A1=#ERR: ARGTYPE  // ***** GTE A1=5&gt;=5%A1=1 A1=6&gt;=5%A1=1 A1=5&gt;=6%A1=0 A1=-1&gt;=6%A1=0 A1=-1&gt;=-1%A1=1 A1=-1&gt;=-2%A1=1 A1=-2&gt;=-1%A1=0 A1="Thomas"&gt;=-1%A1=#ERR: ARGTYPE A1=1&lt;="Thomas"%A1=#ERR: ARGTYPE A1="Maibritt"&gt;="Thomas"%A1=#ERR: ARGTYPE A1=1@A2=1@A3=A1&gt;=A2%A3=1 A1=2@A2=1@A3=A1&gt;=A2%A3=1 A1=1@A2=2@A3=A1&gt;=A2%A3=0 // A2 == null, Level7 throws Exception </pre>		

jul 31, 06 9:18	expression-tests.txt	Page 5/8
<pre> A1=A2&gt;=A2%A1=#ERR: ARGTYPE%skip=Level7 // A2,A3 == null, Level7 throws Exception A1=A2&gt;=A3%A1=#ERR: ARGTYPE%skip=Level7 //Level7 returns 1 instead. A1=5&gt;="Thomas"@A2=A1&gt;=5%A2=#ERR: ARGTYPE%skip=Level7 A1=5&gt;=A2:A3%A1=#ERR: ARGTYPE  // ***** EQ A1=5==5%A1=1 A1=6==5%A1=0 A1=5==6%A1=0 A1=-1==6%A1=0 A1=-1==1%A1=1 A1=-1==2%A1=0 A1=-2==1%A1=0 A1="Thomas"===1%A1=#ERR: ARGTYPE A1=1=="Thomas"%A1=#ERR: ARGTYPE A1="Maibritt"=="Thomas"%A1=#ERR: ARGTYPE A1=1@A2=1@A3=A1==A2%A3=1 A1=2@A2=1@A3=A1==A2%A3=0 A1=1@A2=2@A3=A1==A2%A3=0 // A2 == null, Level7 throws Exception A1=A2==A2%A1=#ERR: ARGTYPE%skip=Level7 // A2,A3 == null, Level7 throws Exception A1=A2==A3%A1=#ERR: ARGTYPE%skip=Level7 //Level7 returns 1 instead. A1=5=="Thomas"@A2=A1==5%A2=#ERR: ARGTYPE%skip=Level7 A1=5==A2:A3%A1=#ERR: ARGTYPE  // ***** NEQ A1=5&lt;&gt;5%A1=0 A1=6&lt;&gt;5%A1=1 A1=5&lt;&gt;6%A1=1 A1=-1&lt;&gt;6%A1=1 A1=-1&lt;&gt;-1%A1=0 A1=-1&lt;&gt;-2%A1=1 A1=-2&lt;&gt;-1%A1=1 A1="Thomas"&lt;&gt;-1%A1=#ERR: ARGTYPE A1=1&lt;&gt;"Thomas"%A1=#ERR: ARGTYPE A1="Maibritt"&lt;&gt;"Thomas"%A1=#ERR: ARGTYPE A1=1@A2=1@A3=A1&lt;&gt;A2%A3=0 A1=2@A2=1@A3=A1&lt;&gt;A2%A3=1 A1=1@A2=2@A3=A1&lt;&gt;A2%A3=1 // A2 == null, Level7 throws Exception A1=A2&lt;&gt;A2%A1=#ERR: ARGTYPE%skip=Level7 // A2,A3 == null, Level7 throws Exception A1=A2&lt;&gt;A3%A1=#ERR: ARGTYPE%skip=Level7 //Level7 returns 1 instead. A1=5&lt;&gt;"Thomas"@A2=A1&lt;&gt;5%A2=#ERR: ARGTYPE%skip=Level7 A1=5&lt;&gt;A2:A3%A1=#ERR: ARGTYPE  // RAND // Rand only evalutated once. Could hypotetical give // a false positive! A1=RAND(@A2=A1@A3=A1@A4=A2==A3%A4=1  // Rand should accept zero parameters A1=RAND(5)%A1=#ERR: ARGCOUNT A1=RAND("Thomas")%A1=#ERR: ARGCOUNT A1=RAND(A2:A3)%A1=#ERR: ARGCOUNT </pre>		

jul 31, 06 9:18	expression-tests.txt	Page 6/8
<pre> // ***** SUM A1=SUM(A1)%A1=Exception(Cyclic)  // * NB * NB * NB: TinyCalc do as CoreCalc. // This is also how Gnumeric and OoCalc evaluates SUM(), // But not how Excel evaluate it. A1=SUM()%A1=0  A1=SUM(A2)%A1=#ERR: ARGTYPE%skip=Level7 A1=SUM("Thomas")%A1=#ERR: ARGTYPE A1=SUM("Thomas",5)%A1=#ERR: ARGTYPE A1=SUM(5,6)%A1=11 A1=SUM(6,-6)%A1=0 A1=SUM(5,-6)%A1=-1 A1=7@A2=8@A3=SUM(5,-6,A1:A2)%A3=14 A1=6@A3=7@A4=SUM(A1:A3)%A4=13 A1=6@A3=7@A4=SUM(SUM(A1:A3))%A4=13 // Backward areas A1=6@A3=7@A4=SUM(A3:A1)%A4=13 A1=6@A3=7@A4=SUM(SUM(A3:A1))%A4=13  // ***** FASTSUM A1=FASTSUM(A2)%A1=#ERR: ARGTYPE%skip=Level7 A1=FASTSUM("Thomas")%A1=#ERR: ARGTYPE A1=FASTSUM("Thomas",5)%A1=#ERR: ARGTYPE A1=FASTSUM(5,6)%A1=11 A1=FASTSUM(6,-6)%A1=0 A1=FASTSUM(5,-6)%A1=-1 A1=7@A2=8@A3=FASTSUM(5,-6,A1:A2)%A3=14 A1=6@A3=7@A4=FASTSUM(A1:A3)%A4=13 A1=6@A3=7@A4=FASTSUM(FASTSUM(A1:A3))%A4=13 // Backward areas A1=6@A3=7@A4=FASTSUM(A3:A1)%A4=13 A1=6@A3=7@A4=FASTSUM(FASTSUM(A3:A1))%A4=13  // ***** DOUBLESUM A1=DOUBLESUM(A2)%A1=#ERR: ARGTYPE%skip=Level7 A1=DOUBLESUM("Thomas")%A1=#ERR: ARGTYPE A1=DOUBLESUM("Thomas",5)%A1=#ERR: ARGTYPE A1=DOUBLESUM(5,6)%A1=11 A1=DOUBLESUM(6,-6)%A1=0 A1=DOUBLESUM(5,-6)%A1=-1 A1=7@A2=8@A3=DOUBLESUM(5,-6,A1:A2)%A3=14 A1=6@A3=7@A4=DOUBLESUM(A1:A3)%A4=13 A1=6@A3=7@A4=DOUBLESUM(DOUBLESUM(A1:A3))%A4=13 A1=6@A3=7@A4=DOUBLESUM(A3:A1)%A4=13 A1=6@A3=7@A4=DOUBLESUM(DOUBLESUM(A3:A1))%A4=13  // ***** references A1=6@A2=7@B1:B2={A1:A2}%B1=6 // Backward areas A1=6@A2=7@B1:B2={A2:A1}%B1=6 Sheet2!A1=5@Sheet2!A2=6@A1=Sheet2!A1+Sheet2!A2%A1=11  // Throws exception at level7 A1=Sheet2!A1+Sheet2!A2%A1=#ERR: ARGTYPE%skip=Level7 A1=5@Sheet2!A1="Thomas"@A2=A1+6%A2=11 </pre>		

jul 31, 06 9:18	expression-tests.txt	Page 7/8
<pre> A1="Thomas"@Sheet2!A1=5@A2=A1+6%A2=#ERR: ARGTYPE%skip=Level7 A1=5@Sheet2!A1="Thomas"@A2=6@Sheet2!A2="Mathias"@Sheet2!A3=A1+A2@A3=Sheet2!A3%A3=#ERR: ARGTYPE%skip=Level7 A1=5@Sheet2!A1="Thomas"@A2=6@Sheet2!A2="Mathias"@Sheet2!A3=Sheet1!A1+Sheet1!A2@A3=Sheet2!A3%A3=11  // ***** Areas A1=5@B1=6@A2=7@B2=8@C1:D2={TRANSPOSE(A1:B2)}%C1=5@D1=7@C2=6@D2=8%startlevel=0@endlevel=0 A1=5@A2:A3=A1*10%A2=50@A3=500 A1=0.5@A2:A10=A1*0.1%A10=5E-10  // ***** Cyclic A1=A2@A2=A1%A1=Exception(Cyclic) A1=RC%A1=Exception(Cyclic) A1=R[0]C[0]%A1=Exception(Cyclic) A1=R[-0]C[-0]%A1=Exception(Cyclic) A1=R[+0]C[+0]%A1=Exception(Cyclic) A1=Sheet1!A1%A1=Exception(Cyclic) A1=Sheet2!A1@Sheet2!A1=Sheet3!A1@Sheet3!A1=Sheet1!A1%A1=Exception(Cyclic) A1=IF(2;A1;3)%A1=Exception(Cyclic)  // ***** Unknown sheet A1=5@A2=Sheeeeeeeeeet1!A1%A2=Exception(SheetName)  // // ***** IF tests //  // test that if only evaluates second branch // if localtest == 0. All other values indicates // that the logical test is true!  A1=IF(0;2;3)%A1=3 A1=IF(1;2;3)%A1=2 A1=IF(2;3;4)%A1=3 A1=IF(-2;-1;0)%A1=-1  // nonstrict ifs are not allowed to throw exception. A1=IF(0;A1;1)%A1=1 A1=IF(1;3;A1)%A1=3  // perfectly legal to have two different types: A1="Thomas" &amp; IF(0;5;"Maibrutt")%A1=ThomasMaibrutt A1="Thomas" &amp; IF(0;"Mathias";5)%A1=#ERR: ARGTYPE A1=6 * IF(0;"Mathias";5)%A1=30  // Types A1=2@A2=IF(A1;"Mathias";5)%A2=Mathias A1=-2@A2=IF(A1;"Mathias";5)%A2=Mathias A1=0@A2=IF(A1;"Mathias";5)%A2=5 A1="Thomas"@A2=IF(A1;"Mathias";5)%A2=#ERR: ARGTYPE%skip=Level7 A1=5@A2=6@A3=IF(A1;A2;"Mathias";5)%A3=#ERR: ARGTYPE A1=5@A2=6@A3=IF("Thomas";"Mathias";5)%A3=#ERR: ARGTYPE A1=5@A2=6@A3:A4={A1:A2}@A5=IF(A3:A4;"Mathias";5)%A5=#ERR: ARGTYPE A1=IF(A2;A3;A4)%A1=#ERR: ARGTYPE%skip=Level7  // Parameter count A1=IF()%A1=#ERR: ARGCOUNT A1=IF(1)%A1=#ERR: ARGCOUNT A1=IF(1;2)%A1=#ERR: ARGCOUNT </pre>		

jul 31, 06 9:18	expression-tests.txt	Page 8/8
<pre> A1=IF(1;2;3)%A1=2 A1=IF(1;2;3;4)%A1=#ERR: ARGCOUNT A1=IF(1;2;3;4;5)%A1=#ERR: ARGCOUNT  // // ***** Inline tests // A1=IF(SIN(SUM(0));1;2)%A1=2 A1=SIN(IF(SUM(0;SUM(1));2;3))%A1=0.909297426825682 A1=IF(IF(1;2;3;4;5)%A1=4 A1=IF(IF(0;2;3;4;5)%A1=4 A1=IF(IF(1;0;3;4;5)%A1=5 A1=IF(IF(0;2;0;4;5)%A1=5 A1=IF(IF(0;2;IF(1;IF(2;3;4);3));6;5)%A1=6  A1=5@A2=6@A3=7@A4=SUM(FASTSUM(DOUBLESUM(A1:A3))%A4=18 A1=5@A2=6@A3=7@A4=SUM(DOUBLESUM(FASTSUM(A1:A3))%A4=18 A1=5@A2=6@A3=7@A4=DOUBLESUM(SUM(FASTSUM(A1:A3))%A4=18 A1=5@A2=6@A3=7@A4=DOUBLESUM(FASTSUM(SUM(A1:A3))%A4=18 A1=5@A2=6@A3=7@A4=FASTSUM(SUM(DOUBLESUM(A1:A3))%A4=18 A1=5@A2=6@A3=7@A4=FASTSUM(DOUBLESUM(SUM(A1:A3))%A4=18  A1=SIN(SIN(5))%A1=-0.818574144461719 A1=SIN(SIN(SIN(5))%A1=-0.73017233793675 A1=SUM(SUM(5))%A1=5 A1=SUM(SIN(SUM(A2:A3))%A1=0 A1=SUM(SIN(SUM(A2))%A1=#ERR: ARGTYPE%skip=Level7 </pre>		

jul 31, 06 9:18

performformulatests.cs

Page 1/10

```
// Perform expression tests

using System;
using System.Collections;
using System.Collections.Generic;
using System.IO;
using System.Text;

namespace TinyCalc
{
    class CellFormula {
        private String cell;
        private String formula;

        public CellFormula(String cell, String formula) {
            this.cell = cell;
            this.formula = formula;
        }

        public String Cell
        {
            get { return cell; }
        }

        public String Formula
        {
            get { return formula; }
        }
    }

    class Result {
    }

    class ExceptionResult : Result {
        private String e;
        public ExceptionResult(String e) {
            this.e = e;
        }
    }

    class CellResult : Result {
        private String cell;
        private String result;

        public CellResult(String cell, String result) {
            this.cell = cell;
            this.result = result;
        }

        public String Cell
        {
            get { return cell; }
        }

        public String Result
        {
            get { return result; }
        }
    }
}
```

mandag juli 31, 2006

../Regression-Tests/performformulatests.cs

jul 31, 06 9:18

performformulatests.cs

Page 2/10

```

    }

    abstract class TestExpression
    {
        public abstract Boolean Test(Options globaloptions);
        public abstract String TestDescription();
    }

    class SimpleFormulaTestExpression : TestExpression
    {
        private Boolean error;
        private String orig_input;
        public FormatOptions fo = new FormatOptions();
        private List<CellFormula> cellFormulas;
        private List<Result> Results;

        public String optionstring = null;

        public SimpleFormulaTestExpression(List<CellFormula> cellFormulas, List<
Result> Results)
        {
            this.cellFormulas = cellFormulas;
            this.Results = Results;
        }

        public SimpleFormulaTestExpression(CellFormula cellFormula, Result Result)
        {
            this.cellFormulas = new List<CellFormula>();
            this.Results = new List<Result>();
            cellFormulas.Add(cellFormula);
            Results.Add(Result);
        }

        public SimpleFormulaTestExpression(String cellresultstr)
        {
            this.cellFormulas = new List<CellFormula>();
            this.Results = new List<Result>();
            this.orig_input = cellresultstr;

            String[] splittedstrings = cellresultstr.Split("%".ToCharArray());

            if (splittedstrings.Length != 2 &&
                splittedstrings.Length != 3) {
                Console.WriteLine("Only one % is allowed to separate formulas from results");
                throw new NotSupportedException("Only one % is allowed to separate formulas from results");
            }

            // Options to this formula?
            if(splittedstrings.Length == 3) {
                optionstring = splittedstrings[2];
            }

            String[] formulas = splittedstrings[0].Split("@".ToCharArray());
            String[] results = splittedstrings[1].Split("@".ToCharArray());

            // Put formula into cells.

```

5/13

```

jul 31, 06 9:18      performformulatests.cs      Page 3/10
foreach(String formula in formulas) {
    String[] formulaparts = formula.Split("=".ToCharArray(),2);
    if(formulaparts.Length != 2) {
        Console.WriteLine("A testformula is allowed to contain o
nly one =");
        throw new NotSupportedException("A testformula is allowe
d to contain only one =");
    }
    cellFormulas.Add(new CellFormula(formulaparts[0],"=" + formulapa
rts[1]));
}

if(results.Length >= 1) {
    String[] formulaparts = results[0].Split("=".ToCharArray());
    if(formulaparts[1].StartsWith("Exception(") &&
        formulaparts[1].EndsWith(")")) {
        Results.Add(new ExceptionResult(formulaparts[1]));
    } else {
        foreach(String result in results) {
            formulaparts = result.Split("=".ToCharArray());
            if(formulaparts.Length != 2) {
                Console.WriteLine("A testresult is allowed to co
ntain only one =");
                throw new NotSupportedException("A testformula i
s allowed to contain only one =");
            }
            Results.Add(new CellResult(formulaparts[0],formulaparts[
1]));
        }
    }
}

public override String TestDescription() {
    return orig_input.Split("%".ToCharArray())[0];
}

public override Boolean Test(Options globaloptions)
{
    // Apply changes to the global options.
    globaloptions.ParseGlobal(optionstring);

    // local = global + local (for this test) options.
    Options localoptions = globaloptions.ParseLocal(optionstring);

    // No error seen so far
    error = false;

    // Each test get a new clean workbook
    Workbook wb = new Workbook();

    // Setup sheets
    for(int i=1; i<=localoptions.NumberOfSheets; i++)
    {
        new Sheet(wb, localoptions.Cols, localoptions.Rows);
    }

    // NB: This can throw an exception if the user specifies 0 number of
sheets!

```

```

jul 31, 06 9:18      performformulatests.cs      Page 4/10
Sheet firstsheet = wb["Sheet1"];

// Add formulas
foreach (CellFormula cellformula in cellFormulas)
{
    String[] cellparts = cellformula.Cell.Split("!".ToCharArray(),2)
;

    // If the cell containing the formula'
    // Contains an ! a sheet reference is assumed.

    Sheet actualsheet;
    String cell;
    if(cellparts.Length == 1) {
        actualsheet = firstsheet;
        cell = cellformula.Cell;
    } else {
        // two parts, sheetname and formula
        // Lookup sheet.
        actualsheet = wb[cellparts[0]];
        cell = cellparts[1];
    }

    try {

        String[] cellareaparts = cell.Split(":".ToCharArray(),2);
        if(cellareaparts.Length >= 1 &&
            cellareaparts.Length <=2 ) {
            if(cellareaparts.Length == 1) {
                // Cell has format A1 and denotes single cell
                actualsheet.AddCell(cellformula.Formula, cell);
            } else {
                // Cell has format A1:B2 and denotes area
                // Test if formula is sourrounded by curly brackets
                // In which case it is a matrix formula.
                // NB! Different syntax than real spreadsheets
                // Real spreadssheet will accept {=TRANSPPOSE(A1:B2)}
                // This testsystem will accept C1:D2 = {TRANSPPOSE(A1
:B2)}

                if(cellformula.Formula.StartsWith("=") &&
                    cellformula.Formula.EndsWith("}")) {

                    String trimmedmatrixformula = cellformula.Formula.Tri
m("={".ToCharArray());

                    actualsheet.AddMatrixFormula("="+trimmedmatrixformula
,cellareaparts[0],cellareaparts[1]);

                } else {
                    // Not matrix formula. Just insert same formula (shar
ed)

                    // Into all cells denoted by the area
                    actualsheet.AddCellArea(cellformula.Formula,cellareap
arts[0],cellareaparts[1]);
                }
            }
        } else {
            Console.WriteLine("A cell area contains only one :");

```

```

jul 31, 06 9:18          performformulatests.cs          Page 5/10
        throw new NotSupportedException("A cell area contains on
ly one :");
    }
} catch(Exception e) {
    // An exception here happens during parsing
    // an is most likely lookup of sheetname

    if(Results.Count == 1 &&
        Results[0] is ExceptionResult) {
        return false;
    } else {
        return true;
    }
}

// Recalculate at all levels and test for results
for (GeneratorLevel level = localoptions.StartLevel; level <= localo
ptions.EndLevel; level++)
{
    if (localoptions.SkipLevel(level))
        continue;

    GeneratorOptions.Level = level;
    GeneratorOptions.QuietRecalc = true;

    if(Results.Count == 1 &&
        Results[0] is ExceptionResult) {
        try {
            wb.Recompute();
            if(!error)
                Console.WriteLine();
            Console.WriteLine(" {0}: Exception expected", l
evel.ToString());
        } catch(Exception e) {
            // FIXME: should tjeck that we catches the RIGHT
exception

            // This is not done as of now :-/ Minor detail.
            error = false;
        }
    } else {
        try {
            wb.Recompute();
            foreach (Result result in Results)
            {
                if(result is CellResult) {
                    Boolean equal = String.Equals(fi
rstsheet.ShowValue((result as CellResult).Cell), (result as CellResult).Result);

```

```

jul 31, 06 9:18          performformulatests.cs          Page 6/10
        if(!equal) {
            if(!error)
                Console.WriteLine
e();
            Console.WriteLine(" {0
}: expected {1} but got {2}", level.ToString(), (result as CellResult).Result, f
irstsheet.ShowValue((result as CellResult).Cell));
        }
        error |= !equal;
    }
} catch(Exception e) {
    if(!error)
        Console.WriteLine();
    Console.WriteLine(" {0}: Unexpected exception:
{1}", level.ToString(), e.Message);
    error = true;
}
}
// return error status
return error;
}

// Class used to control the test options.
public class Options {
    private GeneratorLevel startlevel;
    private GeneratorLevel endlevel;
    private int numberofsheets;
    private int cols;
    private int rows;
    private IDictionary<GeneratorLevel, Boolean> skip;

    public Options() {
        startlevel = GeneratorLevel.Level0;
        endlevel = GeneratorLevel.Level7;
        numberofsheets = 3;
        cols = 10;
        rows = 10;
        skip = new Dictionary<GeneratorLevel, Boolean>();
        skip.Add(GeneratorLevel.Level1,true);
    }

    public Options(GeneratorLevel startlevel,
        GeneratorLevel endlevel,
        int numberofsheets,
        int cols,
        int rows) {
        this.startlevel = startlevel;
        this.endlevel = endlevel;
        this.numberofsheets = numberofsheets;
        this.cols = cols;
        this.rows = rows;
        skip = new Dictionary<GeneratorLevel, Boolean>();
    }

    public GeneratorLevel Skip {
        // FIXME: Can throw exception if key is already defined
        set { skip.Add(value,true); }
    }
}

```

jul 31, 06 9:18

performformulatests.cs

Page 7/10

```

    }

    private GeneratorLevel ParseLevel(String s) {
        String n = s.Replace("Level", "");
        switch(int.Parse(n)) {
            case(0): return GeneratorLevel.Level0;
            case(1): return GeneratorLevel.Level1;
            case(2): return GeneratorLevel.Level2;
            case(3): return GeneratorLevel.Level3;
            case(4): return GeneratorLevel.Level4;
            case(5): return GeneratorLevel.Level5;
            case(6): return GeneratorLevel.Level6;
            case(7): return GeneratorLevel.Level7;
        }
        throw new NotImplementedException();
    }

    private void ParseSkip(String part) {
        Skip = ParseLevel(part);
    }

    private void ParseCols(String part) {
        cols = int.Parse(part);
    }

    private void ParseRows(String part) {
        rows = int.Parse(part);
    }

    private void ParseStartLevel(String part) {
        startlevel = ParseLevel(part);
    }

    private void ParseEndLevel(String part) {
        endlevel = ParseLevel(part);
    }

    private void ParseNumberOfSheets(String part) {
        numberofsheets = int.Parse(part);
    }

    private void ParseOption(String prefix, String optionstring) {
        String[] optionparts = optionstring.Split("=".ToCharArray(), 2);

        if(String.Equals(prefix + "skip", optionparts[0]))
            ParseSkip(optionparts[1]);
        else
            if(String.Equals(prefix + "cols", optionparts[0]))
                ParseCols(optionparts[1]);
            else
                if(String.Equals(prefix + "rows", optionparts[0]))
                    ParseRows(optionparts[1]);
                else
                    if(String.Equals(prefix + "startlevel", optionparts[0]))
                        ParseStartLevel(optionparts[1]);
                    else
                        if(String.Equals(prefix + "endlevel", optionparts[0]))
                            ParseEndLevel(optionparts[1]);

```

jul 31, 06 9:18

performformulatests.cs

Page 8/10

```

        else
            if(String.Equals(prefix + "numberofsheets", optionparts[0]))
                ParseNumberOfSheets(optionparts[1]);
    }

    private void ParseOptions(String prefix, String optionstring) {
        String[] options = optionstring.Split("@".ToCharArray());
        foreach (String option in options) {
            ParseOption(prefix, option);
        }
    }

    public void ParseGlobal(String optionstring) {
        if(optionstring != null)
            ParseOptions("global", optionstring);
    }

    private Options Clone() {
        Options newo = new Options(
            this.startlevel,
            this.endlevel,
            this.numberofsheets,
            this.cols,
            this.rows);

        foreach (KeyValuePair<GeneratorLevel, Boolean> kvp in this.skip)
        {
            newo.Skip = kvp.Key;
        }
        return newo;
    }

    public Options ParseLocal(String optionstring) {
        if(optionstring != null) {
            Options localoptions = this.Clone();
            localoptions.ParseOptions("", optionstring);
            return localoptions;
        }

        return this;
    }

    public Boolean SkipLevel(GeneratorLevel level) {
        try {
            return skip[level];
        }
        catch (KeyNotFoundException) {
            return false;
        }
    }

    public int Cols
    {
        get { return cols; }
        set { cols = value; }
    }

    public int Rows
    {

```

jul 31, 06 9:18

performformulatests.cs

Page 9/10

```

        get { return rows; }
        set { rows = value; }
    }

    public GeneratorLevel StartLevel
    {
        get { return startlevel; }
        set { startlevel = value; }
    }

    public GeneratorLevel EndLevel
    {
        get { return endlevel; }
        set { endlevel = value; }
    }

    public int NumberOfSheets
    {
        get { return numberofsheets; }
        set { numberofsheets = value; }
    }
}

// Main entry point
class Program
{
    // Perform all the tests.
    static void DoTests(List<TestExpression> ExprList, Options globaloptions
) {
    Console.WriteLine("Conducting {0} regression tests", ExprList.Co
unt);
    Console.WriteLine();
    int errorcount = 0;
    foreach(TestExpression testexpr in ExprList) {
        Console.Write("Testing: {0}", testexpr.TestDescription()
);
        if(testexpr.Test(globaloptions)) {
            Console.WriteLine("  Error");
            errorcount ++;
        } else {
            Console.WriteLine("  OK");
        }
    }

    Console.WriteLine("-----");
    if(errorcount > 0) {
        Console.WriteLine("{0} failed tests among {1} tests", er
rorcount, ExprList.Count);
    } else {
        Console.WriteLine("ALL {0} tests succeeded", ExprList.Co
unt);
    }
}

    static void Main(string[] args)
    {
        Options globaloptions = new Options();

```

jul 31, 06 9:18

performformulatests.cs

Page 10/10

```

        List<TestExpression> ExprList = new List<TestExpression>();

        String filename = null;

        // Require an argument (filename of tests)

        if(args == null) {
            Console.WriteLine("One argument is required (filename fo
r the tests)");
            return;
        }

        if (args != null && args.Length > 0)
        {
            if (args.Length > 1)
                Console.WriteLine("Only one argument is allowed
(filename for the tests)");
                return;
            else
            {
                filename = args[0];
            }
        }

        // Read tests
        StreamReader sr = File.OpenText(filename);

        if(sr == null) {
            Console.WriteLine("Could not open file {0}", filename);
            return;
        }

        String line = null;

        // Read tests
        while ((line = sr.ReadLine()) != null)
        {
            // Remove leading and trailing spaces
            line.Trim();
            if(line.Length == 0 ||
                line.StartsWith("//") ||
                line.StartsWith("#")) {
                continue;
            }
            ExprList.Add(new SimpleFormulaTestExpression(line));
        }
        sr.Close();

        // Perform tests
        DoTests(ExprList, globaloptions);
    }
}

```

jul 31, 06 9:18

grammar-tests.txt

Page 1/1

```
// ***** GRAMMAR extension tests
// Test briefly that scientific notation works
A1=1.0+1.0%A1=2
A1=1.0+1.1%A1=2.1
A1=1.0e-1+1.1%A1=1.2
A1=1.0e-2+1.1%A1=1.11
A1=1.0e+2+1.1%A1=101.1
A1=1.0e2+1.1%A1=101.1

A1=1.0+1.0%A1=2
A1=1.0+1.1%A1=2.1
A1=1.0E-1+1.1%A1=1.2
A1=1.0E-2+1.1%A1=1.11
A1=1.0E+2+1.1%A1=101.1
A1=1.0E2+1.1%A1=101.1

A1=1E-1+1.1%A1=1.2
A1=1E-2+1.1%A1=1.11
A1=1E+2+1.1%A1=101.1
A1=1E2+1.1%A1=101.1

E1=1.1@A1=1E-1+E1%A1=1.2
E1=1.1@A1=1E-2+E1%A1=1.11
E1=1.1@A1=1E+2+E1%A1=101.1
E1=1.1@A1=1E2+E1%A1=101.1

A1=5@A2=R[-1]C%A2=5
A1=5@A2=6@A3=R[-1]C%A3=6
A1=5@A2=6@A3=R[-2]C%A3=5
A1=5@A2=6@A3:A4=R[-2]C%A3=5@A4=6
A1=5@B2:E5=R[-1]C[-1]+1%E5=9@E4=#ERR: ARGTYPE@D5=#ERR: ARGTYPE%skip=Level7
A1=1@B1=2@C1=3@D1=4@A2=5@D2=6@A3=7@D3=8@A4=9@B4=10@C4=11@D4=12@B2=R[-1]C[-1]+RC[-1]+R[-1]C@C2=R[-1]C[+1]+RC[+1]+R[-1]C@B3=R[+1]C[-1]+RC[-1]+R[+1]C@C3=R[+1]C[+1]+RC[+1]+R[+1]C%B2=8@C2=13@B3=26@C3=31
A1=0.5@A2:A10=R[-1]C*0.1%A10=5E-10

// Test bounds.
// Failed at first due to problem of OutOfBound exception not being caught
// and the recomputation flags therefor not being reset.
//
// Has Benn Fixed
A1=0.5@A2=R[-2]C%A2=Exception(IndexOutOfBounds)
A1=0.5@A2=R[-2]C[-1]%A2=Exception(IndexOutOfBounds)
A1=0.5@A2=RC[-1]%A2=Exception(IndexOutOfBounds)
A1=0.5@B2=R[-1]C[-1]%B2=0.5

A1=0.5@A2=0.6@A3:A5=SUM(R[-2]C:R[-1]C)%A3=1.1@A4=1.7@A5=2.8

// *****
// For Sheet reference tests see expression-tests.txt
//
// For power function tests see expression-tests.txt
// *****
```

jul 31, 06 9:18

**runtests.bat**

Page 1/1

```
echo "Performing expression tests"  
TinyCalc -f="performformulatests.cs" -a="expression-tests.txt"  
TinyCalc -f="iotests.cs"  
TinyCalc -f="performformulatests.cs" -a="grammar-tests.txt"
```

```

jul 31, 06 9:18      IOTests.cs      Page 1/3
// Perform IO tests
//
// Read XMLSS workbook
// Read GNUMERIC workbook.
// tjeck values that TinyCalc supports.
//     if exceptions are thrown => errors. Maybee implement a handler.
// Write XMLSS or GNUMERIC workbook in XMLSS format and reread and test the save
d workbook.
//
// Copyright Thomas S. Iversen, 2005-2006.

using System;
using System.Collections;
using System.Collections.Generic;
using System.IO;
using System.Text;

namespace TinyCalc
{
    class Program
    {
        static Boolean Test(Sheet sheet, String cell, String value) {
            Boolean ok =String.Equals(sheet.ShowValue(cell), value);
            if(!ok) {
                Console.WriteLine("Problem. Got {0} but expected {1} in
cell {2}", sheet.ShowValue(cell), value, cell);
            }
            return ok;
        }

        static Boolean WorkbookCorrect(Workbook wb) {
            // Test if the workbook appears to be correct.
            Boolean ok = true;
            Sheet sheet = wb["Ark1"];

            ok &= Test(sheet, "B3", "5");
            ok &= Test(sheet, "B4", "6.1");
            ok &= Test(sheet, "B5", "7.1");
            ok &= Test(sheet, "B6", "7.1");
            ok &= Test(sheet, "B7", "82");
            ok &= Test(sheet, "B8", "82");
            ok &= Test(sheet, "B9", "teststring1");
            ok &= Test(sheet, "B10", "\"teststring2\"");
            ok &= Test(sheet, "B11", "\"teststring3\"");
            ok &= Test(sheet, "B12", "teststring1\\teststring2\\");
            ok &= Test(sheet, "B13", "\"teststring2\\\"teststring3\\");
            ok &= Test(sheet, "B14", "11.1");
            ok &= Test(sheet, "B15", "11.1");
            ok &= Test(sheet, "B16", "11.1");
            ok &= Test(sheet, "B17", "13.2");
            ok &= Test(sheet, "B18", "11.1");
            ok &= Test(sheet, "B19", "12.1");
            ok &= Test(sheet, "B20", "206.1");
            ok &= Test(sheet, "B22", "206.1");

            ok &= Test(sheet, "B24", "5");

```

```

jul 31, 06 9:18      IOTests.cs      Page 2/3
        ok &= Test(sheet, "C24", "6");
        ok &= Test(sheet, "D24", "7");

        ok &= Test(sheet, "B28", "18");

        ok &= Test(sheet, "B26", "9");
        ok &= Test(sheet, "B27", "10");

        ok &= Test(sheet, "B28", "18");
        ok &= Test(sheet, "B29", "164");

        ok &= Test(sheet, "B31", "2");
        ok &= Test(sheet, "C31", "3");
        ok &= Test(sheet, "B32", "4");
        ok &= Test(sheet, "C32", "5");

        ok &= Test(sheet, "B33", "6");
        ok &= Test(sheet, "C33", "7");
        ok &= Test(sheet, "B34", "8");
        ok &= Test(sheet, "C34", "9");

        ok &= Test(sheet, "B35", "10");
        ok &= Test(sheet, "C35", "11");

        ok &= Test(sheet, "B36", "12");
        ok &= Test(sheet, "B37", "13");

        ok &= Test(sheet, "B38", "36");
        ok &= Test(sheet, "C38", "41");
        ok &= Test(sheet, "B39", "64");
        ok &= Test(sheet, "C39", "73");

        ok &= Test(sheet, "B41", "1064");
        ok &= Test(sheet, "C41", "1213");

        ok &= Test(sheet, "B42", "28537");
        ok &= Test(sheet, "B43", "28537");

        ok &= Test(sheet, "B47", "57074");

        return ok;
    }

    static void Main(string[] args)
    {
        WorkbookIO workbookio = new WorkbookIO();
        GeneratorOptions.QuietRecalc = true;

        Workbook wbl = workbookio.Read("../TinyCalc/TinyCalc/XML Analysi
s/excel2002-10.6501.6735.sp3/xmlss-xmltest.xml");

        Boolean xmlssok;
        Boolean gnumericok;

        wbl.Recompute();
        xmlssok= WorkbookCorrect(wbl);
        if(!xmlssok) {

```

