```
   *----*    MuPAD 2.0.0 -- The Open Computer Algebra System
  /|   /|
 *----* |    Copyright (c)  1997 - 2000  by SciFace Software
 | *--|-*                   All rights reserved.
 |/   |/
 *----*                     UNREGISTERED VERSION
               Please contact info@sciface.com to register.

>> # Elemi függvények #
>> exp(ln(2));

                                     2

>> exp(2*ln(x));

                                     2
                                    x
>> sin(arcsin(2));

                                     2
>> sin(-x);

                                  -sin(x)
>> sin(PI/2);

                                     1
>> cos(x+PI);

                                  -cos(x)
>> cos(x+2*PI);

                                  cos(x)
>> sin(arctan(2));

                                     1/2
                                  2 5
                                  ------
                                    5
>> DIGITS:=100: sin(0.1);

0.09983341664682815230681419841062202698991538801798225999276686156165174\
28329242760966244380406303627
>> DIGITS:=50: sin(1);

                                  sin(1)
>> DIGITS:=50: sin(1.0); DIGITS:=10:

          0.84147098480789650665250232163029899962256306079837
>> arccos(cos(3));

                                     3
>> log(10,100);

                                     2
>> sign(5);

                                     1
>> assume(x<0): sign(x*y)*sign(x);

                                  sign(y)
>> trunc(5.5);

                                     5
>> trunc(-5.5);

                                    -5
>> round(5.5);

                                     6
>> round(-5.5);

                                    -6
>> floor(5.5);

                                     5
```

```
>> floor(-5.5);

                                    -6
>> frac(1.234);

                                   0.234
>> frac(-1.234);

                                   0.766
>> sqrt(x);

                                    1/2
                                   x
>> binomial(20,10);

                                  184756
>> expand((a+b)^20);

 20     20           19         19           2   18           3   17           4   16
a    + b    + 20 a b    + 20 a    b + 190 a   b    + 1140 a   b    + 4845 a   b    +

          5  15            6  14             7  13              8  12
   15504 a  b    + 38760 a  b    + 77520 a  b    + 125970 a  b    +

            9  11            10  10             11  9             12  8
   167960 a  b    + 184756 a    b    + 167960 a    b   + 125970 a    b   +

          13  7            14  6            15  5            16  4
   77520 a    b   + 38760 a    b   + 15504 a    b   + 4845 a    b   +

         17  3          18  2
   1140 a    b   + 190 a    b
>> max(1,2,3);

                                     3
>> min(1,2,3);

                                     1
>>
>> # Határérték #
>> limit(sin(x)/x,x=0);

                                     1
>> limit((abs(x)-abs(0))/(x-0),x=0);

                                    -1
>> limit((abs(x)-abs(0))/(x-0),x=0,Left);

                                    -1
>> ?limit

limit -- compute a limit

Introduction

 limit(f, x = x0) computes the bidirectional limit lim(f(x), x = x0).

 limit(f, x = x0, Left) computes the one-sided limit lim(f(x), x = x0-).

 limit(f, x = x0, Right) computes the one-sided limit lim(f(x), x = x0+).

Call(s)

 limit(f, x <= x0> <, dir>)

Parameters

 f   -  an arithmetical expression representing a function in x
 x   -  an identifier
 x0  -  the limit point: an arithmetical expression, possibly infinity
        or -infinity

Options

 dir  -  either Left or Right. This controls the direction of the limit
         computation.
```

Returns

an arithmetical expression, an interval of type Dom::Interval, an expression of type "limit", or FAIL.

Side Effects

The function is sensitive to the environment variable ORDER, which determines the default number of terms in series computations (see series and example 6 below).

Properties of identifiers set by assume are taken into account.

Overloadable:

f

Related Functions

asympt, diff, discont, int, O, series, taylor

Details

- limit(f, x = x0) computes the bidirectional limit of f when x tends to x0 on the real axis. The limit point x0 may be omitted, in which case limit assumes x0 = 0.

  If the limit point x0 is infinity or -infinity, then the limit is taken from the left to infinity or from the right to -infinity, respectively.

  If the left and right limits are different, then undefined is returned; see example 2.

- limit(f, x = x0, Left) returns the limit when x tends to x0 from the left. limit(f, x = x0, Right) returns the limit when x tends to x0 from the right. See example 2.

- If the limit does not exist mathematically, but the system can assert that the function f is bounded when x approaches x0, then a bounding interval, of type Dom::Interval, for f(x) in a sufficiently small neighborhood of x0 is returned. This may happen, e.g., if f oscillates infinitesimally fast in the neighborhood of x0; see example 4. The boundaries are the limes inferior and the limes superior of f for x -> x0.

- If the limit cannot be computed, then the system returns a symbolic limit call (see example 3).

- If f contains parameters, then limit reacts to properties of those parameters set by assume; see example 5. If the limit cannot be computed without additional assumptions about the parameters, then limit indicates this by a warning.

- Internally, limit tries to determine the limit from a series expansion of f around x = x0 computed via series. If the number of terms in the series expansion is too small to compute the limit, then limit returns FAIL. In such a case, it may be necessary to increase the value of the environment variable ORDER in order to find the limit (see example 6).

Example 1

The following command computes lim((1-cos(x)/x^2),x=0):

```
>> limit((1 - cos(x))/x^2, x)

                                   1/2
```

A possible definition of e is given by the limit of the sequence (1+1/n)^n for n->infinity:

```
>> limit((1 + 1/n)^n, n = infinity)

                                 exp(1)
```

....

Example 5

limit is not able to compute the limit of x^n for x -> infinity without additional information about the parameter n:

```
>> delete n: limit(x^n, x = infinity)


      Warning: cannot determine sign of n [stdlib::limit::limitMRV]

                                n
                         limit(x , x = infinity)
```

However, for n > 0 the limit exists and equals infinity. We use assume to achieve this:

```
>> assume(n > 0): limit(x^n, x = infinity)

                                infinity
```

Similarly, the limit is zero for n < 0:

```
>> assume(n < 0): limit(x^n, x = infinity)

                                    0
```

Example 6

It may be necessary to increase the value of the environment variable ORDER in order to find the limit, as in the following example:

```
>> limit((sin(tan(x)) - tan(sin(x)))/x^7, x = 0)


      Warning: ORDER seems to be not big enough for series \
      computation [stdlib::limit::lterm]

                                  FAIL

>> ORDER := 8: limit((sin(tan(x)) - tan(sin(x)))/x^7, x)

                                  -1/30
```

Background

- o If a limit cannot be computed, then limit issues a warning with a possible reason, as shown in examples 5 and 6. You may want to suppress these warnings when you call limit from within your own procedures. You can control this by means of the procedure stdlib::limit::printWarnings.

  The calls stdlib::limit::printWarnings(TRUE) and stdlib::limit::printWarnings(FALSE) switch the warnings that limit issues on and off, respectively, and return the previous setting. The command stdlib::limit::printWarnings() returns the current setting, which is TRUE by default.

- o limit first tries a series computation to determine the limit. If this fails, then an algorithm based on the thesis of Dominik Gruntz: ``On Computing Limits in a Symbolic Manipulation System'', Swiss Federal Institute of Technology, Zurich, Switzerland, 1995, is used.

Changes

- o limit may return an interval of type Dom::Interval.

```
>>
>> # Szummáció #
>> sum(1/n,n=1..infinity);

                                 infinity
>> sum(1/(n^2),n=1..infinity);

                                     2
                                   PI
                                   ---
                                    6
>> sum(1/(n^3),n=1..infinity);

                                  zeta(3)
>> sum(1/(n^4),n=1..infinity);

                                     4
                                   PI
                                   ---
                                   90
>> float(_plus(1/(n^3),n=1..1000));
Error:  [_plus]
>> DIGITS:=200: float(PI); DIGITS:=10:

3.141592653589793238462643383279502884197169399375105820974944592307816406\
28620899862803482534211706798214808651328230664709384460955058223172535940\
81284811174502841027019385211055596446229489549303819
>>
>>
>> # Differenciálás #
>> diff(E^x+c,x);

                                  exp(x)
>> diff(x^n,x);

                                   n - 1
                                n x
>> diff(2^x,x);

                                  x
                                 2  ln(2)
>> diff(sin(2*x),x);

                               2 cos(2 x)
>> diff(%,x);

                               -4 sin(2 x)
>> diff(%,x);

                               -8 cos(2 x)
>> diff(%,x);

                               16 sin(2 x)
>> diff(%,x);

                               32 cos(2 x)
>> diff(x^100,x $ 30);

                                                                         70
      77910971370578048745872324992773214403583277006848000000000 x
>>
>> # Integrálás #
>> int(E^x,x);

                                  exp(x)
>> int(x^3-3*x^2+2,x);

                                          4
                                     3   x
                              2 x - x  + --
                                         4
>> int(1/(ln(x)*x),x);

                                ln(ln(x))
```

```
>> int(sin(x)/x,x=1..2);
Warning: While integrating, we will assume x has property [1, 2] of Type::\
Real instead of given property < 0. [intlib::defInt]

                              Si(2) - Si(1)
>> float(%);

                              0.6593299064
>> T:=series(sin(x)/x,x=0,10);

                        2    4      6       8
                       x    x      x       x         9
                   1 - -- + --- - ---- + ------ + O(x )
                       6    120   5040   362880
>> int(T,x);

                       3    5      7        9
                      x    x      x        x          10
                  x - -- + --- - ----- + ------- + O(x  )
                      18   600   35280   3265920
>> int(T,x=1..2);
Warning: While integrating, we will assume x has property [1, 2] of Type::\
Real instead of given property < 0. [intlib::defInt]
Warning: While integrating, we will assume x has property [1, 2] of Type::\
Real instead of given property < 0. [intlib::defInt]

                        9
                 int(O(x ), x = 1..2) + 75366677/114307200
>> float(%);

                           9
                    int(O(x ), x = 1..2) + 0.6593344689
>>
>> # Sorbafejtés #
>> taylor(cos(x)/x,x=0,10);
Error: does not have a Taylor series expansion, try 'series' [taylor]
>> taylor(exp(x),x=0,10);

              2    3    4    5     6      7      8        9
             x    x    x    x     x      x      x        x           10
     1 + x + -- + -- + -- + --- + --- + ---- + ----- + ------ + O(x  )
             2    6    24   120   720   5040   40320   362880
>>
>> # Gyökkeresés #
>> solve(x^4-x^2=1);

                    { --         1/2   1/2     1/2 -- }
                    { |         2    (5    + 1)     | }
                    { |  x = - ------------------   | }
                    { --                2          -- }
>> f:=x^5-x^4-3*x^3+2*x+5;

                                 3    4    5
                         2 x - 3 x  - x  + x  + 5
>> solve(f);

                                                  3      4      5
x in ]-infinity, 0[ intersect RootOf(2 X49 - 3 X49  - X49  + X49  + 5, X49)
>> numeric::solve(f);

{[x = -1.472991017], [x = 1.568770674], [x = 1.906476185],

   [x = - 0.5011279209 - 0.9401206818 I],

   [x = - 0.5011279209 + 0.9401206818 I]}
>> quit
```