Neil Langmead
Konrad-Zuse-Zentrum für Informationstechnik (ZIB)
Takustrasse 7
D- 14195 Berlin Dahlem
Berlin Germany

January 1997

A new exp-log limits package for REDUCE

Contents

1 The Exp-Log Limits package

This package arises from the PhD thesis of Dominik Gruntz, of the ETH Zürich. He developed a new algorithm to compute limits of "exp-log" functions. Many of the examples he gave were unable to be computed by the present limits package in REDUCE,the simplest example being the following, whose limit is obviously 0:

```
load limits;

limit(x^7/e^x,x,infinity);

        7
       x
limit(----,x,infinity)
        x
       e
```

This particular problem arises, because L'Hopital's rule for the computation of indefinite forms (such as $0/0$, or $\frac{\infty}{\infty}$) can only be applied in a CAS a finite number of times, and in REDUCE [Red36], this number is 3. Applied 7 times to the above problem would have yielded the correct answer 0. The new algorithm solves this particular problem, and enables the computation of many more limit calculations in REDUCE. We first define the domain in which we work, and then give a statement of the main algorithm that is used in this package.

Definition:
Let $\Re[x]$ be the ring of polynomials in x with real coefficients, and let f be an element in this ring. The field which is obtained from $\Re[x]$ by closing it under the operations $f \to exp(f)$ and $f \to \log|f|$ is called the L- field (or logarithmico-exponential field, or field of exp-log functions for short).

Hardy proved that every L function is ultimately continuous, of constant sign, monotonic, and tends to $\pm\infty$ or to a finite real constant as $x \to +\infty$.

Here are some examples of exp-log functions, which the package is able to deal with:

$f(x) = e^x * \log(\log(x))$

$f(x) = \frac{\log(\log(x+e^{-x}))}{e^{x^2}+\log(\log(x))}$

$f(x) = \log(x)^{\log(x)}$

$f(x) = e^{x*\log(x)}$

2 The Algorithm

A complete statement of the algorithm now follows: Let f be a log-exp function in x, whose limit we wish to compute as $x \to x_0$. The main steps of the algorithm to do this are as follows:

- Determine the set Ω of the most rapidly varying subexpressions of $f(x)$. Limits may have to be computed recursively at this stage.
- Choose an expression ω such that $\omega > 0$, $\lim_{x\to\infty} \omega = 0$ and ω is in the same comparability class as any element of Ω. Rewrite the other expressions in Ω as $A(x)\omega^c$, where $A(x)$ only contains subexpressions in lower comparability classes than Ω.
- Let $f(\omega)$ be the function obtained from $f(x)$ by replacing all elements of Ω by their representation in terms of ω. Consider all expressions independent of ω as constants and compute the leading term of the power series of f(ω) around $\omega = 0^+$
- If the leading exponent $e_0 > 0$, then the limit is 0, and we stop. If the leading exponent $e_0 < 0$ then the limit is $\pm\infty$. The sign is defined by the sign of the leading coefficient c_0. If the leading exponent $e_0 = 0$ then the limit is the limit of the leading coeficient c_0. If $c_0 \notin C$, where $C = Const(L)$, the set of exp-log constants, we apply the same algorithm recursively on c_0.

The algorithm to compute the most rapidly varying subset (the mrv set) of a function f is given below:

procedure mrv(f)
if (not (depend(f,x))) → return ({})
else if $f = x$ → return($\{x\}$)
else if $f = gh$ → return(max(mrv(g),mrv(h)))
else if $f = g + h$ → return(max(mrv(g),mrv(h)))
else if $f = g^c$ and c $\in C$ → return(mrv(g))
else if $f = log(g)$ → return(mrv(g))
else if $f = e^g$ →
if $\lim_{x\to\infty} g = \pm\infty$ →
return(max($\{e^g\}$, mrv(g)))
else → return mrv(g)
end

The function max() computes the maximum of the two sets of expressions. Max() compares two elements of its argument sets and returns the set which is in the higher comparability class or the union of both if they have the same order of variation.

For further details, proofs and explanations of the algorithm, please consult [Grn96].

For example, we have

$mrv(e^x) = \{e^x\}$
$mrv(log(log(log(x + x^2 + x^3)))) = \{x\}$
$mrv(x) = \{x\}$
$mrv(e^x + e^{-x} + x^2 + x\log(x)) = \{e^x, e^{-x}\}$
$mrv(e^{e^{-x}}) = \{e^{-x}\}$

2.1 Mrv_limit Examples

Consider the following in REDUCE:

```
mrv_limit(e^x,x,infinity);

infinity

mrv_limit(1/log(x),x,infinity);

0

b:=e^x*(e^(1/x-e^-x)-e^(1/x));


           -1        - x
      x + x      - e
b := e       *(e          - 1)


mrv_limit(b,x,infinity);


-1


                                        -1
  ex:=  - log(log(log(log(x))) + log(x))  *log(x)

                       *(log(log(x)) - log(log(log(x)) + log(x)));


            - log(x)*(log(log(x)) - log(log(log(x)) + log(x)))
ex:=     ---------------------------------------------------
                    log(log(log(log(x))) + log(x))

off mcd;

mrv_limit(ex,x,infinity);

1
```

```
(log(x+e^-x)+log(1/x))/(log(x)*e^x);

  - x       -1       -1          - x
e    *log(x)  *(log(x  ) + log(e     + x));

mrv_limit(ws,x,infinity);

0

mrv_limit((log(x)*e^-x)/e^(log(x)+e^(x^2)),x,infinity);

0
```

3 The tracing facility

The package provides a means of tracing the *mrv_limit* function at its main steps, and is intended to help the user if he encounters problems. Messages are displayed informing the user which Taylor expansion is being computed, all recursive calls are listed, and the value returned by the *mrv* function is given. This information is displayed when a switch *tracelimit* is on. This is off by default, but can be switched on with the command

```
on tracelimit;
```

For a more complete examination of the workings of the algorithm, the user could also try the command

```
tr mrv_limit;
```

This is not recommended, as the amount of information returned is often huge and difficult to wade through. Here is a simple example in REDUCE:

```
Loading image file: /silo/cons/reduce35/Alpha/binary/redu37a.img
REDUCE Development Version,  4-Nov-96 ...

1: load mrvlimit;

2: on tracelimit;

3: mrv_limit(e^x,x,infinity);


                        -1
performing taylor on: ww

                       -1
series expansion is ww
```

```
            -1
series is ww

exponent list is {expt,-1}

leading exponent e0 is {expt,-1}

           x
mrv_f is {e }

mrv_f is {x}

                          -1
performing taylor on: ww

                        -1
series expansion is ww

            -1
series is ww

exponent list is {expt,-1}

leading exponent e0 is {expt,-1}

                          -1
performing taylor on: ww

infinity
```

Note that, due to the recursiveness of the functions *mrv* and *mrv_limit*, many calls to each function are made, and information is given on all calls when the *tracelimit* switch is on.

4 Comments, Bug reports and Suggestions

Thsi package was written when the author was a placement student at ZIB Berlin. Please address all comments, bugs and suggestions to Winfried Neun, ZIB, Takustrasse 7, D-14195 Berlin Dahlem, Germany, or e/mail neun@zib.de.

References

[Grn96] Gruntz, Dominik, *On Computing Limits in a Symbolik Manipulation System*, PhD Thesis, ETH Zürich

[Red36] Hearn, Anthony C. and Fitch, John F. *REDUCE User's Manual 3.6*, RAND Corporation, 1995