"According to Abramowitz and Stegun"
or arccoth needn't be uncouth

Robert M. Corless

David J. Jeffrey
Stephen M. Watt
Ontario Research Centre
for Computer Algebra

.

for x < 0 the limit is In x - 2~ri. Related to this is the fact that

In ~ # In z (3)
on the branch cut: instead ln~ = ln z + 21ri on the cut. Similarly,

In 1 ~ --In z (4)
Z

on the branch cut: instead In ½ = -ln z + 2z-i on the cut. Two families of solutions have been mooted to these problems.

• [10] points out that the concept of a "signed zero "4 [9] (for clarity, we write the positive zero as 0 + and
the negative one as 0-) can be used to solve prob-lems such as the above, if we say that, for x < 0, ln(x+0+i) = ln(x) +~ri whereas ln(x+0- 4) = ln(x)-~ri. Equation (2) then becomes an equality for all x, inter-preting the x on the right as x+0-i. Similarly, (3) and
(4) become equalities throughout. Attractive though this proposal is (and OpenMath should probably not inhibit systems that do have signed zeros), it does not answer the fundamental question: what to do if the user types ln(-1).
A similar idea is proposed in [12], who make the func-tions two-valued on the branch cuts, so that In(-1) = ±~ri. This has the drawback of not fitting readily with numerical evaluation.

More importantly, neither scheme addresses problems off the branch cuts.

• [4] points out that most 'equalities' do not hold for the complex logarithm, e.g. ln(z 2) ~ 21nz (try z = -1), and its generalisation

ln(zlz2) ~ lnzl + lnz2. (5)

The most fundamental of all non-equalities is z = In expz, with an obvious violation at z = 2~ri. They therefore propose to introduce the unwinding number K, defined 5 by

/C(z) = z- lnexpz [~z- ~r]
21ri = |~[ e Z (6)

(note that the apparently equivalent definition |-Y-~---~-~/ L 2~r J differs precisely on the branch cut for In as applied to exp z). (5) can then be rescued as

ln(zl z2) = In zl + In z2 - 27ri/C(ln zl + In z2). (7)

Similarly (3) can be rescued as

In ~ = In z -- 21ri/C(ln z). (8)

Note that, as part of the algebra of /C, /C(l~z) =

Ic(- lnz) ¢ ~C(ln ~).
These problems translate into difficulties with the powering operation, even in cases as apparently simple as square roots. In fact,
= V~'X/~ (--1) ~:(ln~l+ln,~).

axis, passing through infinity if necessary: arctan is a good

Lemma 2 An alternative formulation is

arccos(z) = -iln (z + i l~-- zz) . (19)

This is trivially equal to:

-- iln (z -I-iV/(1- zi(1 + z))

which may be numerically more stable near z = ::1=1. This issue, and the choice between (18) and (19), which may de-pend on the availability of hardware square-root capability, are beyond the scope of this paper. The proof is in Appendix B.

4.3 Definition of arctan z

[2] gives the following branch cuts: (-ice,-i] and [i, ic~). [10] suggests, and we will adopt, the following definition:

~/(in(1 + iz) - In(1 -- iz)). (20)

This satisfies

arctan(-z) = - arctan(z), (21)

and is now the definition adopted by Common Lisp [13, p. 309], replacing

- iln ((1 + i=)Jl/(1 + z2)),
which does not satisfy (21).
With this definition, we have the following relationship
(proved in Appendix D) between arcsin and arctan:

Z arcsin z = arctan ~+r/C(- ln(l+z))-Tr/C(- ln(1-z))

(22)

4.4 Definition of arccot z f5133. tc 5.916 0 Tpag55 0 Td (and )Tj 0.03737 Tc 2.799z2)),

5.1 Definition of arcsinh z
[2] gives the branch cuts: (-ioo, -i) and (i, ioo). We follow [10], which gives the principal expression
arcsinh(z) ---- in (z -t- V/1-~), (28)
and this satisfies the symmetry rule:
arcsinh(z) --- - arcsinh(-z)
5.2 Definition of arccosh z
[2] gives the branch cut (-oo, 1): in addition there is a branch point at z = -1. We follow [10],

6.4 Couthness of arccot / arccoth

In this case, equation(13) translates into

arccoth(-iz) ~ i arccot(z). (39)

Substituting in equations (25) and (31) gives

5In =i In xz -- i//

Since the inputs x° to the logarithms are equal, this is trivial.

6.5 Couthness of arcsec / arcsech

In this case, equation(13) translates into

arcsech(z) ~ i arcsec(z). (40)

Substituting in equations (26) and (34) gives

21n \V -~-- ( z~ + ~z z) a in (1/z +iv~-~-lT~ )

(41)
At z = ½, this becomes

a palpable falsehood. In fact, one side is the negation of the other. Now, lemma 2 shows that

Replacing z by 1/z gives

In ¼+i =21n v V--W/"

Substituting this into equation (41) gives

Since lna = lnb if and only if a = b, this reduces to

~ iv/ST. These are equal when arg ~ < 0,
equivalent to arg(1 - 1/z) < 0 This is true precisely when ~z < 0 or ~z = 0 and ~z _> 1. Hence the pair arcsec/arcsech is not couth.

6.6 Couthness of arccsc / arccsch

In this case, equation(13) translates into

arccsch(-iz) ~ i arccsc(z). (42)

Substituting in equations (27) and (35) gives

In _---~z + 1+ ~ = In +

These are patently equal.
1°It is tempting to a computer scientist to write "arguments of", but that way lles linguistic confusion.

7 Implications for Phrase-book Writers

An OpenMath phrase-book is actually a piece of software that translates between the semantics of OpenMath (as de-fined in the Content Dictionaries) and the semantics of a particular application, as well as simply translating names. Of course, life for the phrase-book writer is simplest if the semantics of the application are the same as those of Open-Math, but this will not always be the case.
A classic example of this would be the translation of arccot between OpenMath, whose semantics we propose to define by (25), and a system, such as Maple or Axiom, where the semantics of arccot are defined by (24). Application--~OpenMath arccot z ~ ~ - arctan z OpenMath-+Application arccot z ~ arctan (l/z)
Derive has a different definition of arctan to eliminate the unwinding numbers from (22), so that, for Derive, arcsin(z) = arctan ~ This definition can be related

Derive

to that of OpenMath either via unwinding numbers or via ,~..~(z) = arctan~. It is often possible to deal with such

Derive

differences on branch cuts by such a 'double conjugate' rep-resentation. This representation has the advantage that the (always legal) rule ~ ~ z means that the transformation is self-inverse.

8 Proposals

This section lists the concrete suggestions that the authors have for OpenMath.

1. OpenMath should define a b (in the case of non-integer b) via
a b = exp(b in a), (43)

rather than the current weasel words: "When the sec-ond argument is not an integer care should be taken as
to the meaning of this function; however it is here to represent general powering" (axithl. ocd [11]).

2. OpenMath should base all its single-valued definitions of the naturally multi-valued elementary functions on the logarithm function. This means that it would be reasonable to define

arcsin (16) arcsinh (28) couth
arccos (18) arccosh (29) uncouth
arctan (20) arctanh (30) couth
arccot (25) arccoth (31) couth
arcsec (26) arcsech (34) uncouth
arccsc (27) arccsch (35) couth

4.