4

Simplifying Square Roots of Square Roots by Denesting

David J. Jeffrey
The University of Western Ontario

Albert D. Rich
Soft Warehouse, Inc.

Abstract: We discuss why it is important to try to simplify the square root of an expression containing other square roots, and we give rules for doing this when it is possible. The square root of an expression containing nth roots is also considered briefly. This article, in addition to treating a specific mathematical topic, shows some of the steps that developers must take when writing computer algebra systems.

4.1 Introduction

Numbers such as $\sqrt{2}$ and $\sqrt[3]{5}$ are called *surds*[1] [Chrystal64, Hall88]. The more general terms *radical* and *algebraic number* are also used, but surd is the most specific. It is a venerable term that has been revived for use in computer systems[2]. Given positive integers n and k, the unique positive real root of the equation $x^n = k$ will be denoted $\sqrt[n]{k}$ and called a surd. Most books [Chrystal64, Hall88] allow k to be a positive rational number, and there is no great difference in principle if we accept that generalization; in this article, however, all the examples use integers. Another point of variation is the treatment of perfect-power factors, i.e., whether one writes $\sqrt{8}$ or $2\sqrt{2}$; here we use whichever form is more convenient.

The term *radical* may be familiar to some readers from the common phrase 'solvable in terms of radicals', which is used whenever a textbook discusses the roots of a polynomial [Dickson26]. A polynomial equation is solvable by radicals if '...its roots can be found by rational operations and extractions of a root...' [Dickson26]. Thus a quantity such as $\sqrt{1+\sqrt{2}}$ is a radical, but not a surd. Polynomials of degree less

---

[1] Please do not tell us jokes about these numbers being *ab*surd. We have heard them all before.
[2] The term *surd* is used by TEX as the name for the symbol $\surd$; Maple has a function called `surd` that is similar to the nth root defined here; like all good mathematical terms, the precise definition depends upon the context. In general, a mathematical term that does not have several conflicting definitions is not important enough to be worth learning.

From *Computer Algebra Systems: A Practical Guide*, M.J. Wester, Ed., Wiley 1999.

than or equal to 4 are always solvable by radicals, and higher-degree polynomials are sometimes solvable by radicals. An *algebraic number* is a root of some polynomial with integer coefficients, and in general will not be expressible as a radical, although all radicals are special cases of algebraic numbers. The positive real solution of the equation $x^n = r$, where r is a positive radical, will here be called the nth root of r, and will itself be a radical. The other familiar name for these types of numbers is $(1/n)$th power, but this term will not be used because fractional powers are regarded by many people as multivalued functions[3], and we want to work with uniquely defined (and named) quantities.

Our attention in this article is directed to radicals consisting of the square root of an expression containing surds, for example $\sqrt{\sqrt{2}+\sqrt{3}}$. Such expressions are often called *nested radicals*, although one could argue that strictly the word nested is redundant. Such expressions can sometimes be simplified, but the rules for such simplifications are not discussed in standard books. Here are some examples of simplifications. The first example shows that two square roots can sometimes be reduced to one:

$$\sqrt{3+2\sqrt{2}} = 1+\sqrt{2}\ . \tag{4.1}$$

Sometimes the number of square root operations remains unchanged, but people prefer the format of the new expression:

$$\sqrt{5+2\sqrt{6}} = \sqrt{2}+\sqrt{3}\ . \tag{4.2}$$

This rearrangement of the square-root operations is usually called *denesting*. Here are some more examples of square-root denesting.

$$\sqrt{5\sqrt{3}+6\sqrt{2}} = \sqrt[4]{27}+\sqrt[4]{12}\ , \tag{4.3}$$

$$\sqrt{12+2\sqrt{6}+2\sqrt{14}+2\sqrt{21}} = \sqrt{2}+\sqrt{3}+\sqrt{7}\ . \tag{4.4}$$

We do not restrict ourselves to taking the square root only of other square roots:

$$\sqrt{\sqrt[3]{9}+6\sqrt[3]{3}+9} = 3+\sqrt[3]{3}\ , \tag{4.5}$$

$$\sqrt{\sqrt[3]{5}-\sqrt[3]{4}} = \tfrac{1}{3}\left(\sqrt[3]{2}+\sqrt[3]{20}-\sqrt[3]{25}\right)\ . \tag{4.6}$$

4.2 Why denest square roots?

Discovering the relations given in the examples above is an interesting mathematical challenge, but why should a computer algebra system (CAS) devote computing resources to denesting problems? The first reason is *simplicity*. Users of CAS, like all mathematicians, prefer to see their results expressed in 'simplest' form. Most people would regard the right-hand sides of our examples above as being 'simpler' than the left-hand sides, although there might be some who would disagree. For example, in

[3] The question "Are fractional powers multivalued?" is a standard basis for table assignments at conference banquets.

From *Computer Algebra Systems: A Practical Guide*, M.J. Wester, Ed., Wiley 1999.

[Landau92b], reasons are given for preferring the left-hand side of (4.2). However, we assume that in general users would want denesting to be discovered[4].

Another reason for denesting is *reliable simplification.* For both people and computers, there is a danger that the result of a mathematical simplification will depend upon the order in which rules are applied. For example, the simplification of $\sqrt{(1-\sqrt{2})^2}$ can proceed two ways. The first way is called 'top-down' by those who think of the expression as a tree, and 'outside to middle' by those who look at the printed form. Following this procedure, one applies first the rule $\sqrt{x^2} = |x|$ for any real x, and obtains $\sqrt{(1-\sqrt{2})^2} = \sqrt{2}-1$. The other way is 'bottom-up' or 'middle outwards', in which one expands the square and obtains $\sqrt{3-2\sqrt{2}}$. Without denesting, the simplification will not proceed any further. Thus two people using the same CAS to solve the same problem could arrive at apparently different answers, and this is what we wish to avoid[5].

A final reason is a small improvement in the *accuracy* of the numerical evaluation of some radical expressions. For example, the quantity

$$\sqrt{199999 - 600\sqrt{111110}} \approx 0.00158$$

approximates to zero if 10 decimal digits of precision are used, and only at 15 digits is the expression found to be nonzero; in contrast, the equivalent $100\sqrt{10} - 3\sqrt{11111}$ approximates to 0.00158 using just 7 digits of precision.

4.3 Where do nested radicals arise?

Older algebra textbooks contain explicit problems on nested radicals, for example, the books by Chrystal [Chrystal64] and Hall & Knight [Hall88], which are both 19th century books (the 1964 date on the Chrystal citation is not indicative of the age of the book, whose first edition was published in 1886). Clearly, users could challenge a computer system with problems from such books, but there are other sources of problems. Even if users do not deliberately pose denesting problems, the problems they *do* pose can generate subproblems that contain nested radicals.

Many problems in mathematics have solutions expressed as standard formulae. For example, all students learn quite early in their studies the formula for solving a quadratic equation; not surprisingly, it is programmed into most CAS. Consider what happens when that formula is used for the problem

$$x^2 + 6x - 4\sqrt{5} = 0\ .$$

The standard formula gives

$$x = -3 \pm \sqrt{9+4\sqrt{5}}\ ,$$

[4] Long after ordinary text was set mechanically, mathematical equations were still set by hand. Long horizontal lines, as in large fractions or large roots, were troublesome for the compositors, and authors of mathematics were encouraged to prefer forms that reduced the need for such lines. Perhaps now that we have TEX to set these expressions easily, our mathematical tastes with respect to simplicity will change also.

[5] Not to mention all the phone calls to technical support.

but in fact the quadratic can be factored as $(x+1-\sqrt{5})(x+5+\sqrt{5})$. Therefore the general formula is correct, but does not directly give the simplest result in this special case. The problem is an interesting one to try on humans, as well as computer systems.

Now consider the formula, valid for real a, b with $b > a^2$,

$$\int \frac{1}{4x^2+4ax+b}\,\mathrm{d}x = \frac{1}{2\sqrt{b-a^2}} \arctan \frac{2x+a}{\sqrt{b-a^2}} \;. \tag{4.7}$$

If we substitute $a = 2$ and $b = 7 + 2\sqrt{2}$, we find that (4.7) becomes

$$\int \frac{1}{4x^2+8x+7+2\sqrt{2}}\,\mathrm{d}x = \frac{1}{2\sqrt{3+2\sqrt{2}}} \arctan \frac{2x+2}{\sqrt{3+2\sqrt{2}}} \;.$$

As example (4.1) shows, however, the nested radicals on the right-hand side can be simplified. Of course, for most values of a and b, the term $\sqrt{b-a^2}$ must remain unsimplified.

4.4 Developing algorithms

In the next section, we shall give algorithms for simplifying nested radicals, but it is worthwhile first to make a few general comments. Every description of an algorithm should specify the class of problems to which it applies, which in turn requires that we decide which problems we are going to tackle. As a rule of thumb, the more general our class of problems, the larger and slower our algorithm is likely to be. In general, system developers identify the most frequently occurring cases and concentrate first on them.

The simplification of radicals has been the subject of several recent studies [Landau92b, Landau92a, Zippel85], but these studies have taken general approaches to the problem, with the algorithms they derive being correspondingly lengthy. Here we consider algorithms of restricted applicability, handling only commonly occurring cases, and gain brevity at the price of generality. Specifically, the algorithms given here do not address example (4.6) above.

A second component in the specification of an algorithm is the form the answer will take. In this context, we observe that approximating a nested radical as a decimal fraction is certainly a type of simplification, but not one in which we are interested. The possibility that different mathematicians will define simplification differently was pointed out above, and that will certainly influence the form in which a simplification is sought. Here we shall take it that we aim to reduce the level of nesting, that we require exact results and that the result will also be expressed as a radical.

4.5 The algorithms

We begin with a short treatment of a simple case, and then proceed to the main case, which requires a longer study.

4.5.1 Square root of a three-term sum

Equation (4.5) can be generalized to the pattern

$$\sqrt{a^2 \pm 2ab + b^2} = |a \pm b| \ , \tag{4.8}$$

where either a or b is a surd other than a square root (i.e., so that the corresponding squared term is still a surd). The obvious starting point for detecting this pattern is the fact that there are 3 terms under the square root, although it must be realised that, in general, we shall not know the order in which the terms appear. For example, in the case of the problem $\sqrt{4+\sqrt[3]{81}+4\sqrt[3]{9}} = 2+\sqrt[3]{9}$, the first term under the square root corresponds to a^2. However, in the similar looking $\sqrt{6+\sqrt[3]{81}+\sqrt[3]{9}} = \sqrt[3]{3}+\sqrt[3]{9}$, the first term corresponds to $2ab$.

So the question is: given $\sqrt{X+Y+Z}$, do there exist a and b such that X, Y, Z correspond in some order to a^2, b^2 and $\pm 2ab$? We can suppose that the problem has been formulated in such a way that we know that $X+Y+Z>0$. One might think of analysing the structure of the 3 terms, but this could be a lengthy operation for a CAS. A quicker way to proceed is to notice that if $X=a^2$ and $Y=b^2$, then $4XY = 4a^2b^2 = (\pm 2ab)^2 = Z^2$. So the system can test $4XY - Z^2$ for zero, and then return $\left|\sqrt{X} + \operatorname{sgn} Z\sqrt{Y}\right|$. The signum function takes care of the $\pm$ possibility, and the absolute-value signs take care of the possibility that $Y>X$. In the same way, we can test $4XZ-Y^2$ and $4YZ-X^2$. Some final things to notice about this procedure are the simplicity of the failure condition — we give up if all test quantities are nonzero — and the fact that there is no obvious path to generalize it to encompass a wider class of problems.

4.5.2 Square root of square roots

Examples (4.1)—(4.3) show that an important pattern to consider is

$$\sqrt{X+Y} = \sqrt{A}+\sqrt{B} \ . \tag{4.9}$$

We need to find A, B in terms of X, Y, and we need the circumstances in which the right-hand side is simpler than the left. Squaring both sides gives us

$$X+Y = A+B+2\sqrt{AB} \ . \tag{4.10}$$

One way to satisfy this equation is to set $X=A+B$ and $Y^2=4AB$. Now if (4.9) is valid, then it should also be true that

$$\sqrt{X-Y} = \sqrt{A}-\sqrt{B} \ . \tag{4.11}$$

Multiplying (4.9) and (4.11) gives $\sqrt{X^2-Y^2} = A-B$. Having expressions for $A+B$ and $A-B$, we can derive expressions for A and B in terms of X and Y, and hence discover the following theorem.

Theorem: *Let $X, Y \in \mathbb{R}$ with $X>Y>0$, then*

$$\sqrt{X\pm Y} = \sqrt{\tfrac{1}{2}X+\tfrac{1}{2}\sqrt{X^2-Y^2}} \pm \sqrt{\tfrac{1}{2}X-\tfrac{1}{2}\sqrt{X^2-Y^2}} \ . \tag{4.12}$$

From *Computer Algebra Systems: A Practical Guide*, M.J. Wester, Ed., Wiley 1999.

We call this the square-root-nesting equation. Some numerical experiments will show how it can be useful. If $X = 3$ and $Y = \sqrt{8}$, then (4.12) reproduces example (4.1), because in this case $\sqrt{X^2 - Y^2} = 1$. Example (4.2) is obtained similarly. Example (4.3) is a little different. Now, $X = \sqrt{75}$ and $Y = \sqrt{72}$, giving $\sqrt{X^2 - Y^2} = \sqrt{3}$. This is a rational multiple of X, and hence the first term on the right of (4.12) becomes

$$\sqrt{\tfrac{1}{2}X + \tfrac{1}{2}\sqrt{X^2 - Y^2}} = \sqrt{\tfrac{5}{2}\sqrt{3} + \tfrac{1}{2}\sqrt{3}} = \sqrt{3\sqrt{3}} = \sqrt[4]{27} \; .$$

The other term is similar and we still have a denesting.

The common feature in these examples is the reduction of each of the expressions $X \pm \sqrt{X^2 - Y^2}$ to a single term, or in other words to a non-sum. This, then, is a condition for (4.12) to be a simplification. An algorithm based on (4.12) works by assigning X and Y and then computing the terms on the right side of the equation, accepting them as a simplification if the square roots simplify to non-sums. We shall call this method 1.

We now turn to example (4.4). A little dexterity allows us to use (4.12) again. We group the terms so that $X = 12 + 2\sqrt{6}$ and $Y = 2\sqrt{14} + 2\sqrt{21}$. The terms in (4.12) now become

$$X \pm \sqrt{X^2 - Y^2} = 12 + 2\sqrt{6} \pm \sqrt{28 - 8\sqrt{6}} \; .$$

This may not seem to be leading to a simplification, but using method 1 above, we see $\sqrt{28 - 8\sqrt{6}} = 2\sqrt{6} - 2$. Therefore,

$$\begin{aligned} X + \sqrt{X^2 - Y^2} &= 10 + 4\sqrt{6} \; , \\ X - \sqrt{X^2 - Y^2} &= 14 \; . \end{aligned}$$

So with these simplifications, equation (4.12) becomes

$$\sqrt{12 + 2\sqrt{6} + 2\sqrt{14} + 2\sqrt{21}} = \sqrt{5 + 2\sqrt{6}} + \sqrt{7} \; .$$

This is already a simplification, but additionally the first term on the right-hand side is example (4.2) and can be simplified further. Therefore we finally reproduce (4.4). The repeated use of method 1 we shall call method 2.

Why stop there? Consider an even bigger problem:

$$\sqrt{65 - 6\sqrt{35} - 2\sqrt{22} - 6\sqrt{55} + 2\sqrt{77} - 2\sqrt{14} + 6\sqrt{10}} \; . \tag{4.13}$$

Assigning $X = 65 - 6\sqrt{35} - 2\sqrt{22}$ and $Y = -6\sqrt{55} + 2\sqrt{77} - 2\sqrt{14} + 6\sqrt{10}$, we can use method 2 to simplify the critical factor.

$$\begin{aligned} \sqrt{X^2 - Y^2} &= \sqrt{-468\sqrt{35} + 156\sqrt{22} - 24\sqrt{770} + 2869} \\ &= 39 - 6\sqrt{35} + 2\sqrt{22} \; . \end{aligned}$$

Continued use of (4.12) simplifies (4.13) to $3\sqrt{5} - \sqrt{7} - \sqrt{11} + \sqrt{2}$. This is method 3. Pursued further, this technique solves larger and larger problems.

These successes show that an algorithm is possible, but we must ask several more questions before we attempt to implement it in a system. The most important one

From *Computer Algebra Systems: A Practical Guide*, M.J. Wester, Ed., Wiley 1999.

is how the method fails. After all, the majority of nested surds do not denest, and so any implementation must know when to give up. Answering this question turns out to be as difficult as finding the successful part of the algorithm[6]. It might seem frustrating having to spend a lot of time deciding when the system will not succeed, but this decidedly less glamorous activity is essential to the smooth operation of a CAS. Consider, therefore, an example slightly altered from (4.1): $\sqrt{4+2\sqrt{2}}$. Substituting $X=4$ and $Y=2\sqrt{2}$ into (4.12) gives

$$\sqrt{4+2\sqrt{2}} = \sqrt{2+\sqrt{2}} + \sqrt{2-\sqrt{2}}\ .$$

Observe that, on the right, there are two expressions, each as complicated as the one on the left, and therefore the process fails and no more denesting is possible.

Moving up one level of difficulty, we alter (4.4) and see what happens. Consider

$$\sqrt{12+2\sqrt{6}+2\sqrt{14}+2\sqrt{20}}\ .$$

We assign $X=12+2\sqrt{6}$ and $Y=2\sqrt{14}+2\sqrt{20}$, and find

$$\sqrt{X^2-Y^2} = \sqrt{32-16\sqrt{70}+48\sqrt{6}}\ .$$

Thus we have expressed a square root of 4 terms in terms of 2 square roots of 3 terms, and trying to simplify these leads to square roots of 2 terms, which do not simplify. So we have replaced one ugly expression with one ugly set of expressions, and so the procedure fails.

So we now have an idea of when there is no simplification, but we can also ask whether we would ever miss a simplification. For method 2, we must break a sum up into two groups, and it may be that the wrong grouping will miss a denesting. To check this possibility, we consider

$$(a+b+c)^2 = a^2+b^2+c^2+2ab+2bc+2ca\ .$$

If a,b,c are all square-root surds, then $a^2+b^2+c^2$ is an integer and we set $X = a^2+b^2+c^2+2ab$. Further, by the symmetry of the expression, it will not matter how we divide up the four terms, because one part of the division always contains the $a^2+b^2+c^2$ part together with one of the terms $2ab$, $2bc$, $2ca$, and the other part always contains the rest. Thus, we never miss.

For larger problems, however, the ordering *does* become important. For the case

$$(a+b+c+d)^2 = a^2+b^2+c^2+d^2+2ab+2ac+2ad+2bc+2bd+2cd\ ,$$

only the division $X=a^2+b^2+c^2+d^2+2ab+2cd$ leads to successfully discovering the denesting. In this case, the algorithm is guaranteed to succeed only if the system invests the resources to analyse the structure of the expression. If the expression is blindly split into two, then the method will sometimes succeed by the luck of the ordering, but otherwise fail.

---

[6] Since we used dexterity to get the algorithm to succeed, we need *sinisterity* to find examples for which it fails.

4.6 Types of implementation

The first aim of this article has been to describe the mathematical basis of some algorithms used in CAS, but there is a second aim, which is to give a flavour of the ways in which algorithms are implemented in CAS. To this end, we shall write out some implementations in a pseudocode. This code is not in the language of any system we know; it is just a way of setting down the steps one would follow.

Let us now consider different ways in which a developer might implement the rules we have seen above. As a first cut, we notice that each type of example has a different number of terms under the square root. We might therefore use this fact to select between methods 1—3 in section 4.5.2.

Assume that our system has a function that counts the number of terms in a sum, called `TermsInSum`. Some code for a function that returns a simplification, or else fails, is given below using this approach.

```
Simplify_Square_Root(Expression)
Check that Expression is made up of surds.
Let T=TermsInSum(Expression)
If      T=1 then
          Extract the root if possible, else FAIL
else if T=3 then
          Follow method for square-root of 3 terms;
          X=first term; Y=second term; Z=third term;
               is any of 4XY-Z^2, 4XZ-Y^2, 4YZ-X^2 zero?
               Yes: compute appropriate expression else FAIL.
else if T=2 then
          Follow method 1.
          X=first term; Y=second term;
          is X^2-Y^2 a non-sum?
          Yes: compute square root, else FAIL.
          Is X+sqrt(X^2-Y^2) a non-sum, and
             X-sqrt(X^2-Y^2) a non-sum?
          Yes: compute the two square roots, else FAIL.
else if T=4 then
          Follow method 2.
          X=first 2 terms; Y=remaining terms;
          Compute TEMP=X^2-Y^2.
          Does TermsInSum(TEMP)=2 ?
          Yes: Let U=first term in TEMP; V=remaining term.
          Is U^2-V^2 a non-sum? Yes: complete method else FAIL.
else if T=7 then
          Follow method 3.
          Order terms; select 2 corresponding to 2ab and 2cd;
          Assign X, Y; continue as described in the text.
else FAIL.
```

This very explicit code would not be attractive to many current computer algebra system developers. In particular, it is very specific and tied to the examples that we

From *Computer Algebra Systems: A Practical Guide*, M.J. Wester, Ed., Wiley 1999.

have so far explored. Therefore, adding new cases must be done explicitly, and the code has little hope of working for examples slightly different from those it was designed for. For example, if this code were implemented and a user challenged the system with an example the developer had not considered, like

$$\sqrt{5+2\sqrt{6}+5\sqrt{7}+2\sqrt[4]{700}+2\sqrt[4]{1575}} = \sqrt{2}+\sqrt{3}+\sqrt[4]{175} \;,$$

then there is no hope that the system could surprise the programmer by rising to the occasion, because the square root contains 5 terms, and that case is not treated[7]. As users report problems that the system 'cannot do', the developer is faced with constantly revisiting the code. This will always happen to some extent, of course, but it is particularly inevitable with this style of programming.

The above code also contains coding that repeats itself. This suggests that a more flexible approach will be a recursive one, meaning one in which the routine will be structured to call itself. The difficulty with a recursive approach is that we must be very careful to have a way of stopping it[8]. We want to abandon the process whenever it looks as though we are no longer making progress. The easiest way to do this is to have a numerical measure of the degree of nesting of a radical, and stop the recursion whenever this measure increases.

4.7 A measure of the degree of nesting of a radical

We now describe the function that is used to control the recursive denesting of radicals. One measure of the nesting of a radical is given in [Landau92b]; the one given here is similar in spirit. Suppose we have a radical x. We wish to associate with it an integer $\mathcal{N}(x)$ that is its nesting level. Our rules are:

4.7.1 If x is a number

A number here means an integer, but more generally it includes rational numbers also. (More abstractly, a member of the base number field.) For a number x, set $\mathcal{N}(x)=1$. Some measures of nesting start counting at 0, but the starting point is arbitrary. Any undefined symbols are also given $\mathcal{N}=1$.

4.7.2 If x is an nth root of something

If x has the form $\sqrt[n]{y}$, with $n>1$, then we assign $\mathcal{N}(\sqrt[n]{y}) = 1+\mathcal{N}(y)$. This is the fundamental feature that we wish to capture with our measure, so if we think of the radical being built up from other radicals, then every time we take another root we increase the measure. Thus $\mathcal{N}(\sqrt{2}) = 1+\mathcal{N}(2) = 2$. Notice that the size of n is not used. Therefore the simplification $\sqrt{4}=2$ is visible to this measure ($\mathcal{N}$ decreases from 2 to 1), but the simplification $\sqrt[4]{4}=\sqrt{2}$ is not, because only the strength of the root is

[7] As a case in point, we were agreeably surprised ourselves when Derive succeeded on this one.

[8] A system experiencing uncontrolled recursion is said to suffer from *recussion* — a special form of concussion. Its name reminds us that the developer often starts cussin' all over again.

From *Computer Algebra Systems: A Practical Guide*, M.J. Wester, Ed., Wiley 1999.

reduced. This is acceptable, since nesting is what we are trying to measure, but other applications might require a measure in which n is included somehow.

4.7.3 If x is a product

From the point of view of nesting, $\sqrt{2}\sqrt{3}$ is no more complicated than $\sqrt{6}$. More generally, given radicals $\sqrt[m]{x}$ and $\sqrt[n]{y}$, there are integers a, p such that $\sqrt[m]{x}\,\sqrt[n]{y} = \sqrt[p]{a}$. Clearly the right-hand side counts as a single nesting, so it would be inconsistent to consider the left side as being more nested. Now consider $\sqrt{2}\left(\sqrt{2+\sqrt{2}}\right)$. The second factor is the one that will attract our attention and dictate the degree of nestedness of the whole expression, and so the rule is $\mathcal{N}(xy) = \max(\mathcal{N}(x), \mathcal{N}(y))$.

4.7.4 If x is a sum

Consider the example $\sqrt{2}+\sqrt{8} = 3\sqrt{2}$. The right-hand side is preferable, and so any measure should give a bonus for a reduction in the number of terms. Now consider $x = \sqrt{4} + \sqrt{2+\sqrt{2}} + \sqrt{2+\sqrt{2+\sqrt{2}}}$. To simplify this expression, we must give separate attention to each term; if in addition we can combine terms, so much the better. Therefore our measure adds together the degrees of nestedness of the separate terms. This can be written as follows. A sum of radicals is split into two groups, a and b. Then $\mathcal{N}(a+b) = \mathcal{N}(a) + \mathcal{N}(b)$.

4.7.5 Examples

Let us apply these rules to our examples above. For example (4.1), the left side is

$$\mathcal{N}\left(\sqrt{3+2\sqrt{2}}\right) = 1 + \mathcal{N}\left(3+2\sqrt{2}\right) = 1 + \mathcal{N}(3) + \mathcal{N}\left(\sqrt{2}\right) = 1+1+2 = 4\ .$$

The right side has $\mathcal{N} = 3$, so there is a definite simplification by this measure. For example (4.2), both sides have $\mathcal{N} = 4$, and as mentioned, most users would then choose the denested expression. In (4.3), the left side has $\mathcal{N} = 5$, while the right has $\mathcal{N} = 4$ because the 4th roots do not change the value of $\mathcal{N}$. This result supports the choices made in designing $\mathcal{N}$, because if 4th roots had been penalised relative to square roots, then the right side could easily have obtained the higher score.

4.8 Recursive simplification

Once we have the function $\mathcal{N}$, we can use it in applying the square-root-nesting equation (4.12). Again resorting to a pseudocode for the implementation, we write

```
Recursive-Sqrt-Simplify(E)
Compute n=N(E)
Split E^2 into X and Y.
Compute T=X^2-Y^2
If N(T) > n then FAIL.
Simplify P=X+SQRT(T)
```

```
If N(P) > n then FAIL.
Simplify Q=X-SQRT(T)
If N(Q) > n then FAIL.
Return SQRT(P/2)+SIGN(Y)*SQRT(Q/2)
```

Notice the advantage of recursion: this code covers all cases. However, it does not include any attempt to sort the expression before splitting, and so will miss some cases.

Now let us step through the code using example (4.4). Since we shall make several trips through the `Recursive-Sqrt-Simplify` routine, we must distinguish the same variable names in different calls. We do this by adding subscripts to each variable name. Thus we start with the expression

$$E_1 = \sqrt{12 + 2\sqrt{6} + 2\sqrt{14} + 2\sqrt{21}}\ .$$

The first step is to compute $n_1 = \mathcal{N}(E_1) = 8$. Then we split: $X_1 = 12 + 2\sqrt{6}$ and $Y_1 = 2\sqrt{14} + 2\sqrt{21}$. Now compute

$$T_1 = 28 - 8\sqrt{6} \quad \text{and} \quad \mathcal{N}(T_1) = 3 < n_1\ .$$

The reduction in nesting allows the calculation to continue to the computation of P_1.

$$P_1 = 12 + 2\sqrt{6} + \sqrt{28 - 8\sqrt{6}}\ .$$

This expression must now be simplified. When P_1 is passed to the simplifier, it will see the nested root contained in P_1 and call `Recursive-Sqrt-Simplify`. The following computation will now take place.

The intermediate quantities in this trip through `Recursive-Sqrt-Simplify` will be subscripted with 2.

$$E_2 = \sqrt{28 - 8\sqrt{6}} \quad \text{and} \quad n_2 = \mathcal{N}(E_2) = 4\ .$$

Now splitting, we get $X_2 = 28$ and $Y_2 = -8\sqrt{6}$, giving $T_2 = 400$ and $\mathcal{N}(T_2) = 1$. Since the nesting is reducing, the process continues.

$$\begin{aligned} P_2 = 28 + \sqrt{400} &= 48 \quad \text{and} \quad \mathcal{N}(P_2) = 1 < n_2\ , \\ Q_2 = 28 - \sqrt{400} &= 8 \quad \text{and} \quad \mathcal{N}(Q_2) = 1 < n_2\ . \end{aligned}$$

So this second run through `Recursive-Sqrt-Simplify` returns $\sqrt{24} - 2$.

That was all to simplify P_1 at the first level, so now we return to that level and continue.

$$P_1 = 12 + 2\sqrt{6} + \sqrt{24} - 2 = 10 + 4\sqrt{6} \quad \text{and} \quad \mathcal{N}(P_1) = 3 \leq n_1 = 8\ .$$

Using the simplification of $\sqrt{T_1}$ already found, we can compute $Q_1 = 14$ and hence the procedure returns $\sqrt{5 + 2\sqrt{6}} + \sqrt{7}$. Since this expression is different from the starting one, the system will return this to the main simplification routine, which will restart from the top. This time, the denesting routine will receive $E = \sqrt{5 + 2\sqrt{6}}$ and return $\sqrt{2} + \sqrt{3}$. Together with the $\sqrt{7}$ already found, the simplify routine obtains $\sqrt{2} + \sqrt{3} + \sqrt{7}$. Again the final expression has changed, but this time when the process restarts, there is nothing to simplify, so it stops.

From *Computer Algebra Systems: A Practical Guide*, M.J. Wester, Ed., Wiley 1999.

4.9 Testing

All CAS developers have test suites that they run their systems on. These suites must contain problems for which the system is expected to obtain the correct simplification, and problems for which the system should correctly find no simplification. Derive has one such suite specifically for denesting problems. It contains all the examples given in this chapter and many others. We challenge readers to use their dexterity and sinisterity to invent some interesting examples to add to our suite. As a starting point, a specific case for which we have not given an example is a square root in which the ordering of the terms is important to the denesting.

An ideal test suite will have at least one example to activate each path through the algorithms of the system. As this simple denesting code illustrates, there are always many places where quantities are tested and execution paths switched. Compiling a thorough test suite even for this small part of a complete system is lengthy and difficult, so it is no wonder that over many years of use, users find examples that activate previously unexercised paths in the code. The test suites of all the CAS contain many examples contributed, often inadvertently, by (we hope temporarily) disgruntled users.

From *Computer Algebra Systems: A Practical Guide*, M.J. Wester, Ed., Wiley 1999.

References

[Chrystal64] G. Chrystal (1964) *Algebra, volumes I and II.* Chelsea Publishing Company, New York, 7th edition.
[Dickson26] L. E. Dickson (1926) *Algebraic theories.* Dover, New York.
[Hall88] H. S. Hall and S. R. Knight (1888) *Higher Algebra.* MacMillan, London, 2nd edition.
[Landau92a] Susan Landau (1992) A Note on Zippel Denesting. *Journal of Symbolic Computation* **13**: 41–45.
[Landau92b] Susan Landau (1992) Simplification of Nested Radicals. *SIAM Journal on Computing* **21**: 85–110.
[Zippel85] R. Zippel (1985) Simplification of Expressions involving Radicals. *Journal of Symbolic Computation* **1**: 189–210.