A computer algebra user interface manifesto

David R. Stoutemyer[*]

Abstract

Many computer algebra systems have more than 1000 built-in functions, making expertise difficult. Using mock dialog boxes, this article describes a proposed interactive general-purpose wizard for organizing optional transformations and allowing easy fine grain control over the form of the result – even by amateurs. This wizard integrates ideas including:

- flexible subexpression selection;
- complete control over the ordering of variables and commutative operands, with well-chosen defaults;
- interleaving the choice of successively less main variables with applicable function choices to provide detailed control without incurring a combinatorial number of applicable alternatives at any one level;
- quick applicability tests to reduce the listing of inapplicable transformations;
- using an organizing principle to order the alternatives in a helpful manner;
- labeling quickly-computed alternatives in dialog boxes with a preview of their results, using ellipsis elisions if necessary or helpful;
- allowing the user to retreat from a sequence of choices to explore other branches of the tree of alternatives – or to return quickly to branches already visited;
- allowing the user to accumulate more than one of the alternative forms;
- integrating direct manipulation into the wizard; and
- supporting not only the usual input-result pair mode, but also the useful alternative derivational and *in situ* replacement modes in a unified window.

1 Introduction

> *"Before the Lisp machine interface to Macsyma,*
> *computer algebra was like doing mathematics encumbered by boxing gloves."*
> – Bill Gosper

I am sorry Bill, but that user interface from 1988 [18] disappeared with the Lisp machine, and its best features regrettably have not yet been implemented in any of the current most powerful computer algebra systems. Even the much earlier 1972 article [6] discusses desirable features that are still missing from modern systems.

Many computer algebra systems have more than 1000 built-in functions. Besides standard *mathematical functions* such as cos(...) and classic higher transcendental functions, built-in functions often include numerous *optional transformation functions* such as expand(...), factor(...),

---

[*]dstout at hawaii dot edu

trigExpand(...), convert(...), and simplify(...) that supplement default simplification with various transformations. Besides the expression being transformed, these transformational functions often accept optional extra arguments such as a list of variables, and/or various keyword arguments that control details such as the amount of factoring. Moreover, most computer algebra systems have numerous global *control variables* whose values help control transformations done by these functions and/or by default.

Unlike cos(...), the names and semantic details of these transformation functions and control variables are not part of any standard mathematics curriculum. Therefore it requires a long time to fully exploit such systems well, and most users never do. Moreover, these names and behaviors vary greatly between systems, making it challenging to become skilled with more than one system in order to exploit their differing capabilities. Consequently many users are frustrated because they don't know how to make any system transform an expression to a desired form.

Worse yet, often there is *no* composition of functions and/or or combination of control-variable settings capable of producing a desired form. For example, users often want expansion with respect to a certain proper subset of the variables, with polynomial coefficients that are factored with respect to the other variables – or want partial fractions with respect to a certain proper subset of some variables, with factored numerators and denominators.

This article address the usefulness of computer algebra systems as productivity tools to help amateur users accomplish their tasks without necessarily being aware of the underlying algorithms, transformation functions and their nomenclature. This article presents some ideas for enhancing the kind of wizard implemented on the Lisp Machine by:

1. more flexible subexpression selection;

2. complete control over the ordering of variables and commutative operands, with well-chosen defaults;

3. interleaving the choice of successively less main variables with applicable function choices to provide detailed control without incurring a combinatorial number of applicable alternatives at any one level;

4. quick applicability tests to reduce the listing of inapplicable transformations;

5. using an organizing principle to order the alternatives in a helpful manner;

6. labeling quickly-computed alternatives in dialog boxes with a preview of the result, using ellipsis elisions if necessary or helpful;

7. allowing the user to retreat from a sequence of choices to explore other branches of the tree of alternatives – or to return quickly to branches already visited;

8. allowing the user to accumulate more than one of the alternative forms;

9. integrating direct manipulation into the wizard; and

10. supporting not only the usual input-result mode, but also the useful alternative derivational and *in situ* replacement modes in a unified window.

Section 2 describes important features to combine in a user interface. Section 3 presents some mock examples of using the proposed wizard. Section 4 addresses design issues and their resolution, with

a summary in Section 5. The Appendix summarizes some important rational-expression transformations that should be included in addition to those discussed in subsection 4.3. The wizard must also be aware of all of the many transformations specific to irrational expressions that are built in or should be, but that is an open-ended topic too large for discussion here. However, the wizard should be implemented in an extensible way that allows users to add components easily at run time for new transformations that they implement.

This article often uses "Float" as an abbreviation for "floating-point number".

2 Important features to combine in a user interface

Many of the ideas in this section have been implemented to some extent in various computer algebra systems, but integrating them into a uniformly designed user interface could greatly enhance the user experience over that of any one current system.

Computer algebra often generates large expressions, and with current RAM sizes measured in gigabytes, screen area is now the most precious resource for most user's tasks.[1] Many of the ideas discussed here are concerned with attempting to make the best use of that limited resource to help amateur and expert users arrive quickly at the most comprehensible alternative output forms.

2.1 Enhancing the Lisp Machine Macsyma precedent

Among the most helpful features of Lisp Machine Macsyma was that as you moved the mouse over an expression, the minimal rectangle containing a syntactically complete subexpression surrounding the mouse pointer would automatically be framed, which is perhaps the entire expression. A right click would open a drop-down menu of common transformations such as factor and expand, or you could enter a function name of your own. The selected transformation is applied to the framed subexpression.

This feature could be regarded as a *wizard* that helped users quickly locate appropriate subexpressions for desired transformations, then apply them to those subexpressions. This is important because without subexpression selection and shortcuts for applying a desired transformations to selected subexpressions, it is painful for even expert users to force large expressions into anything near the form they would prefer – death by a thousand cuts and pastes.

Part of the pain is carefully reassembling a final result from independently transformed subexpressions, then carefully deleting the distracting debris of all the intermediate steps.

Even merely ordering commutative operands as desired is difficult or impractical in most computer algebra systems. For example, it is a constant irritant to be unable to transform a result such as $E = c^2 m$ to $E = mc^2$ or, better yet, to prearrange that it will automatically be ordered as desired.

2.2 Including direct manipulation

Direct manipulation provides a complementary way that major computer algebra systems could make their user interfaces more helpful: With Milo [2] or Theorist [7] you could use the mouse to select a term or a factor, then drag and drop it to

- reorder terms and factors,

---

[1]The maximum number of legible characters simultaneously legible on multiple high-resolution screens or sheets of paper is unimprovable.

- distribute a term over a factor,

- factor out a common factor from a sum of terms,

- transpose factors or terms from one side of an equation to the other.

The selected subexpression could also be dropped into a variable in another expression to substitute the subexpression for every instance of the variable in that other expression. Also, expressions could optionally be compressed by automatic or mouse-driven temporary replacement of subexpressions with ellipses.

Here is a temporal sequence of Milo snapshots for dragging x successively further right in an equation [16]:

$$\frac{\boxed{x}y}{2} + x = 1 \quad \rightarrow \quad \frac{y\boxed{x}}{2} + x = 1 \quad \rightarrow \quad \frac{y}{2}\boxed{x} + x = 1 \quad \rightarrow \quad \left(\frac{y}{2} + 1\right)\boxed{x} = 1 \quad \rightarrow \quad \frac{y}{2} + 1 = \frac{1}{x}$$

Milo evolved to The Plotting Calculator, which is still available and supported [3]. The most recent version of Theorist is named LiveMath$^{\mathrm{tm}}$, which is also still available and supported [19]. Both are oriented toward mathematics through calculus, but direct manipulation should also be implemented in other computer algebra systems, integrated with the transformation wizard proposed here: After selecting subexpressions, one of the transformation options, if applicable, should be "drag and drop".

2.3   Collecting multiple alternatives

The wizard generates and displays the results of alternative transformations as the user explores a tree of successive applicable transformations. The user can accumulate any number of these alternative results into a list that is returned as the result if the user wants more than one. For example, as users interactively view alternative factored and partial fraction forms for a rational expression, they can indicate which ones they want included in a returned list of alternatives. This is inspired by Wolfram|Alpha [38], which automatically returns multiple alternative results. The difference here is that the user can participate in a more thorough exploration of the possible alternatives and select only those of interest.
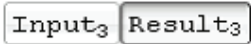
2.4   Input-result pairs *versus* derivation steps *versus* replacement

Most industrial-strength computer algebra systems use only the *input-result* mode. This mode is particularly appropriate when the goal is to obtain good final results in as few steps as is practical.

Some mathematics education programs such as Mathpert$^{\mathrm{tm}}$ and the SMG application for some TI computer algebra products use the *derivation* mode wherein the input is transformed to a result by selecting successive subexpressions and choosing transformations from menus, with the annotated result of each step displayed beginning on a separate line. This mode is also often used in theorem proving software [29, 37]. This mode is also good for expository use by professionals when they want to explain in a presentation or publication how a result is derived. For example, the multi-step derivational style is used several times in this article.

Some programs such as The Graphing Calculator offer the *replacement* mode wherein selected transformations replace the selected subexpressions *in situ*. This mode has the advantage of conserving screen space by minimizing the amount of debris – at the expense of not being able to view the input and result simultaneously.

An industrial-strength computer-algebra system should offer all three modes. The following observations can justify allowing mixtures of all three modes in a single session window and name-space context:

- An input-result pair can be regarded as a one-step instance of the derivation mode.

- A one step *in situ* replacement could be labeled with a two-button setter bar such as $\boxed{\texttt{Input}_3}$ $\boxed{\texttt{Result}_3}$ that toggles between the two.

- A multi-step *in situ* replacement could have a slider bar between these two endpoints, and perhaps also a Play button that does a slide show or an animation.

- A right click could offer the option of changing previous computations between these three modes, such as

  - collapsing a derivation sequence to an input-result pair or to an *in situ* replacement,
  - expanding an input-result pair to a refinable derivation sequence that was automatically used to create it.

The model-view-controller paradigm is a good way to achieve this multi-view software design [36].

2.5 {Undo^m, redo^n}

Anyone who has used software with a well designed essentially unlimited undo-redo capability knows how aggravating it is to return to software that offers only one step of undo – perhaps with no redo. With current RAM capacities measured in gigabytes there is no excuse for this. Internet browsing has familiarized users with using the "$\leftarrow$" and "$\rightarrow$" buttons together with a drop-down browse history list to revisit easily throughout the tree of past web page visitations. The wizard can use the same techniques and temporarily save all recent closed dialog boxes for quick regeneration.

2.6 Dynamically created dialog boxes specialized for the example

The variety of mathematics examples is so great that general-purpose dialog boxes created when a computer algebra system is built would be unpleasantly cumbersome to use:

- They would have numerous distracting grayed-out controls.

- They would entail numerous subsidiary dialog boxes to accommodate all of the inapplicable entries without making each dialog box unreasonably large.

- They would contain lengthy or awkward wording such as "variable or variables" or "variable(s)" to correctly accommodate both singular and plural cases without distracting grammatical errors.

Thus custom dialog boxes specialized to the framed subexpression must be created at run time.

2.7 Adapt to the user's level and goals

Computer algebra is being used by students from beginning secondary school algebra through graduate-level mathematics – and by professional mathematicians, scientists, engineers, economists etc. at many different levels of sophistication. The number of potential users declines rapidly with increasing mathematics level. However, most computer algebra systems are designed for the higher levels of this spectrum. Consequently the most powerful general-purpose systems are quite daunting to most potential users. For example, in many courses from secondary school algebra through university real-variable calculus:

1. Many students know nothing about hyperbolic functions, higher transcendental functions and hypergeometric functions.

2. Most students have not encountered many standard mathematics symbols and notations such as $\exists$, $\forall$, $\neg$, $\vee$, $\wedge$, $\Re$, $\Im$, $\mathbb{Z}$ and $\mathbb{Q}$.

3. Most students know nothing about terminology such as algebraic groups, rings, fields, ideals, varieties, and square-free factorization.

4. The expression $1/0$ is usually or always regarded as undefined rather than as $\pm\infty$ or a circle of infinite radius in the complex plane.

5. The expression $\sqrt{-1}$ is usually or always regarded as undefined rather than i.

6. The expression $(-1)^{1/3}$ is usually taken to mean -1 rather than $1/2 + i\sqrt{3}/2$.

The mathematically weakest students and professionals who could most benefit from computer algebra are most intimidated by the appearance of such unknown function names, symbols and nomenclature in their dialog boxes and results. Often this intimidation and the consequent loss of self esteem terminates receptivity to learning effective use of the computer algebra system.

In computer aided instruction there are efforts to automatically infer the level and overall goals of users, then adapt the interface accordingly. Those techniques are not explored in this article, and the termination of receptivity might occur before enough input occurs to make an accurate inference. However, one easy way to accomplish many of the benefits of such customization is for the first dialog box of a session to have a button labeled "Session preferences" that opens a dialog such as the following if pressed:
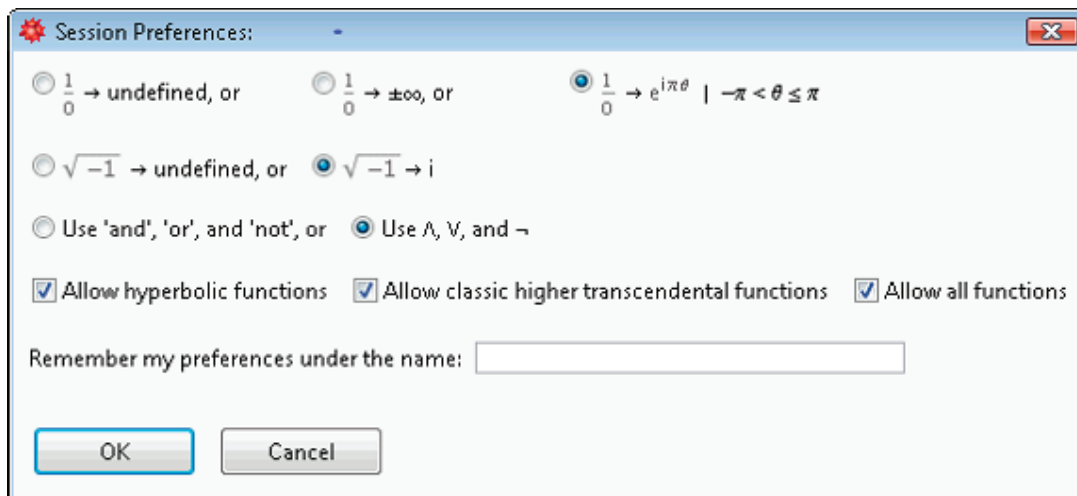
The defaults should be those of the computer algebra system, but the more elementary alternatives should appear first in each row of alternatives to minimize alarming elementary users.

Another complementary alternative to matching student mathematical level is to enable the easy creation of named shell programs and their icons that launch the computer algebra system then immediately set appropriate preferences automatically. An instructor can then create such shells named, for example, MapleForAlgebra1 or MathematicaForCalculus1.

3 Examples of using the wizard

This section contains examples of using the proposed wizard.

3.1 Simplifying an ugly expression

The *Mathematica* CreateDialog[...] function is a convenient way to create new dialog boxes at run time. Thus I used CreateDialog[...] to create dialog boxes that are appropriate for specific examples, without bothering to attach these boxes to each other or to any *Mathematica* transformation functions. The mock intermediate and final results are not that of any particular current computer algebra system, but rather what I wish they would produce – especially with regard to ordering of factors and terms. For example, suppose that *either an input or a result* of previous steps is

$$
\begin{aligned}
(p^2qr^3 &+ p^2qr^2s + p^2qr^2t + p^2qrst + p^2r^4 + p^2r^3s + p^2r^3t + p^2r^2st + pqr^4 + pqr^3s + pqr^3t + pqr^2st + \\
&2pqrs + 2pqrt + 4pqr + 4pqst + 2pqs + 2pqt + pr^5 + pr^4s + pr^4t + pr^3st + pr^2s + 2pqrs + \\
&2pqrt + 4pqr + 4pqst + 2pqs + 2pqt + pr^5 + pr^4s + pr^4t + pr^3st + pr^2s + pr^2t + 2pr^2 + \\
&2prst + 2prs + 2prt + 2pr + 2pst + ps + pt + qr^2s + qr^2t + 2qr^2 + 2qrst + 2qrs + 2qrt + \\
&2qr + 2qst + qs + qt + 2r^2s + 2r^2t + 4r^2 + 4rst + 2rs + 2rt) \big/ \\
(pqr^2 &+ pqrs + pqrt + pqst + pr^3 + pr^2s + pr^2t + prst + qr^3 + qr^2s + qr^2t + qrst + r^4 + r^3s + r^3t + r^2st).
\end{aligned}
\tag{1}
$$

I have no *particular* goal form in mind, but I would like a result that is more concise and comprehensible – and more efficient for substitution of numbers. I position the mouse pointer between the left margin and the expression, thus framing the entire expression, then right click and choose "transform".[2] This opens the following dialog box courteously positioned just above the subexpression if the subexpression is low on the screen, or just below the subexpression otherwise[3]:


---
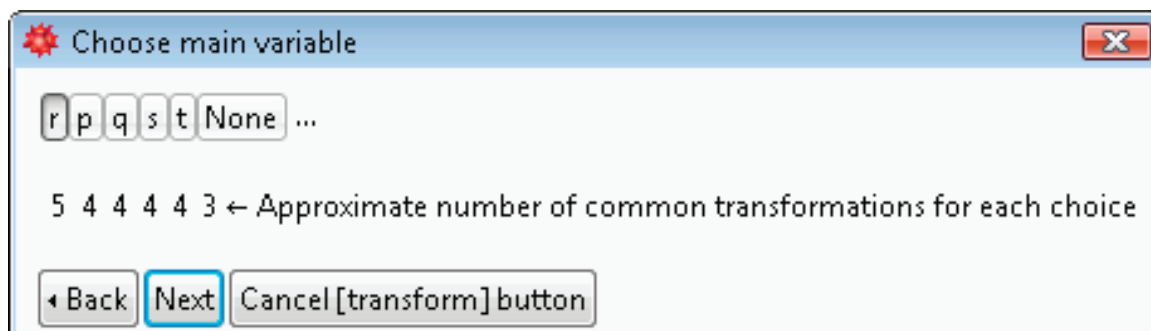
[2]Alternatively, I can choose "Transform ..." from the main menu bar or click ⎣Transform⎦ on the main toolbar.

[3]I hate it when a dialog box initially covers the information I need to respond to it!

Some transformations such as expansion of an improper ratio to a polynomial plus a proper ratio require a designated variable. The $\boxed{\text{None}}$ button considers only transformations that do not require such a variable, such as factoring or *polynomial* expansion with respect to *all* variables. With that choice, the variables would be ordered according to the analysis in [20].

The variables are listed in non-increasing order of an estimated number of applicable common transformations because the choice of main variable tends to have the greatest influence on the overall form of the result. Choosing a main variable for which there are few alternative forms tends to narrow the choices more than otherwise. For example, if we chose p, q, s or t as the main variable, then there would probably be fewer than 5 common alternatives for r thereafter. Thus button r was initialized to *pressed* to encourage lazy users such as me to accept it, which I do. This opens the following dialog box:
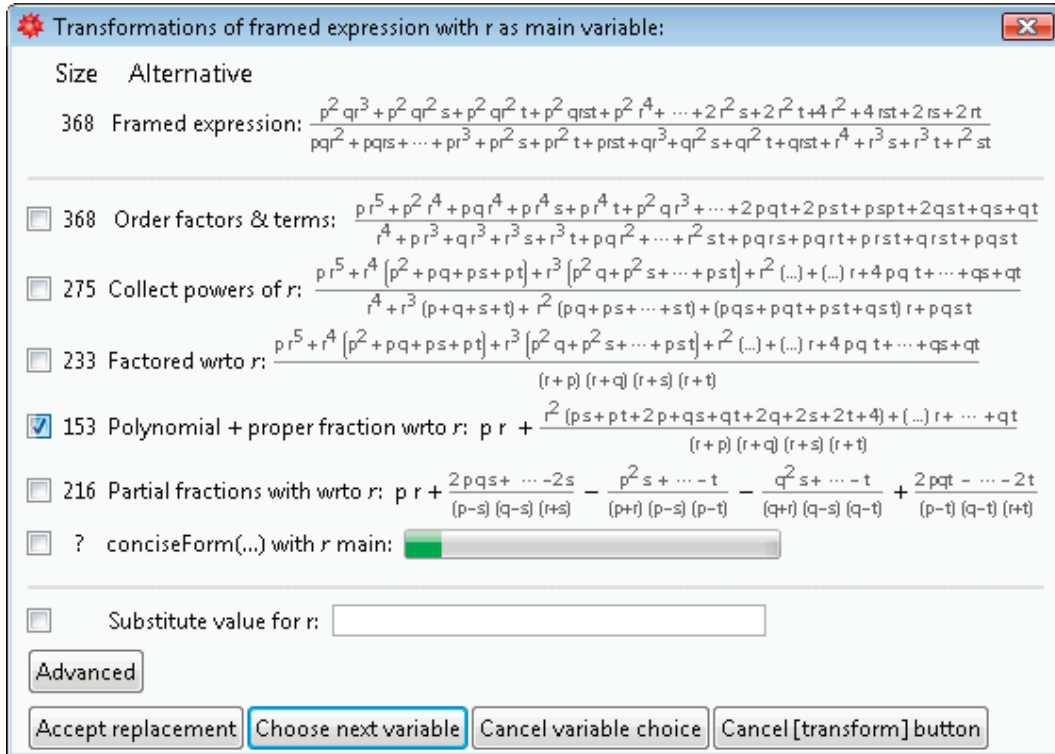
> "*A lot of times, people don't know what they want until you show it to them.*"
> – Steve Jobs

The $\boxed{\text{Advanced}}$ button lists alternatives that would interest most users only occasionally for this example – alternatives such as continued fractions, Hornerization, expression in terms of Chebyshev polynomials, or series approximations.

The displayed *sizes* are some easily computed measure that correlates approximately with the relative area that would be required to display the entire alternative results. The initially checked boxes are those having the smallest size.

The dialog box shows alternative results for all the alternatives that can be computed in a *total* of at most 0.1 seconds – with elisions if necessary to avoid scroll bars or using the entire screen. User interface designers [23] feel that maximum acceptable response times are:

- about 0.1 seconds for responses to a mouse click, key press, or anything involving hand-eye coordination;

- about 1 second for opening a progress indicator, closing a dialog box or reformatting a table;

- about 10 seconds for everything else, including displaying a graph or completing an understandably time-consuming task. This "mind begins to wander" threshold is not always achievable with computer algebra.

The *Mathematica* Collect[..., r], Factor[...], PolynomialQuotient[..., ..., r], PolynomialRemainder[..., ..., r] and Apart[..., r] functions require a total of only 0.04 seconds on a dual-core 1.6 gigahertz computer to compute *Mathematica*-ordered versions of the five initially-displayed results in this dialog box. Therefore this dialog box could be created and displayed in an acceptable amount of time.
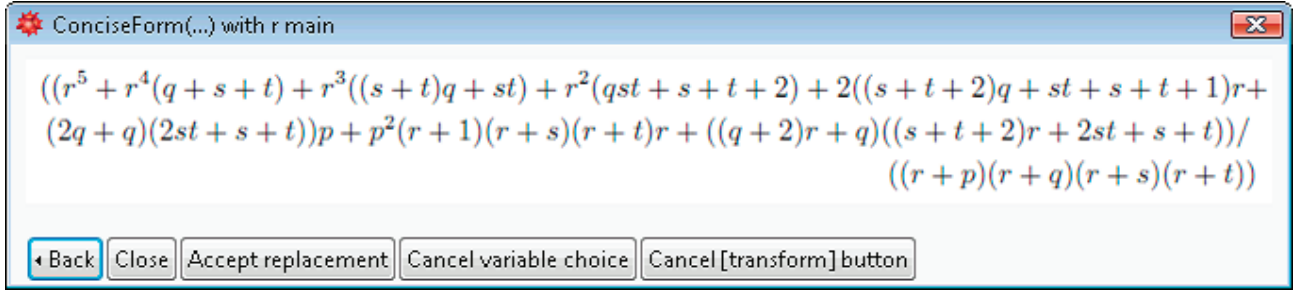
The progress bar for the conciseForm(...) alternative appears within one second, indicating that it is still being computed after the initial display of the dialog box. If that function doesn't post progress messages, then a slug cyclically moves from left to right to let the user know that the computer is working rather than merely awaiting user input.[4] The conciseForm(...) alternative represents the system's most powerful general-purpose simplification function such as the *Mathematica* FullSimplify[...] function, which requires 2.4 seconds for this example. This is way less than the time it requires for me to compare the five alternatives above conciseForm(...) with the Framed expression. As such continuing computations complete, their results are displayed in the dialog box – elided if necessary. They are listed below initially presented results to reduce displacement distraction when they complete. The conciseForm(...) alternative completes with a size of 125, which is the smallest, so that check box *also* becomes checked if I haven't already pushed a button.

Every time a check box is checked that has not been checked before, if the corresponding result is elided, then another dialog box opens that merely displays the un-elided alternative result, with scroll bars if necessary. For example, the initial "Polynomial + proper fraction wrt r" choice opens the dialog box
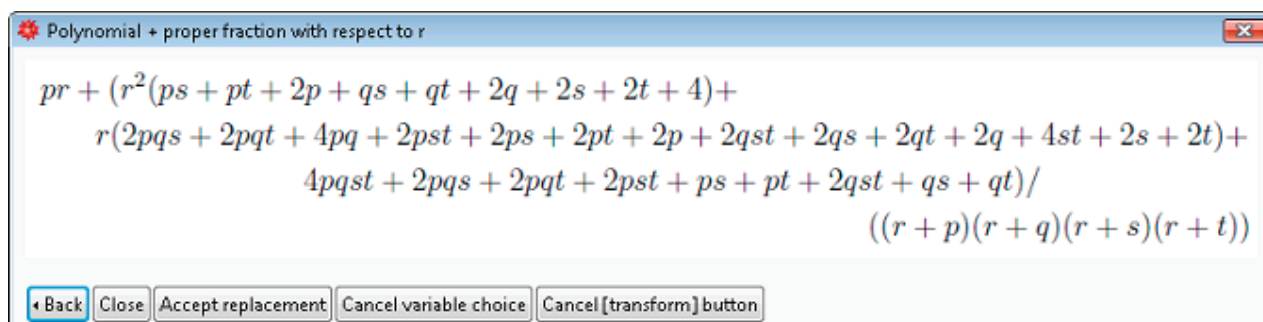
and when conciseForm(...) completes, it opens the dialog box

---

[4]It is not yet customary to have computer algebra functions post progress messages, but there are obvious candidate events for some algorithms. For example, many algorithms for degree n or for n variables, terms, factors, equations or columns process them one at a time, permitting messages of the form "$1/n$ % done", "$2/n$ % done", ... even if the time spent for each such step is likely to be rather uneven. People are comforted by progress bars even when they are inaccurate.

ConciseForm(...) with r main

$$((r^5 + r^4(q + s + t) + r^3((s + t)q + st) + r^2(qst + s + t + 2) + 2((s + t + 2)q + st + s + t + 1)r +$$
$$(2q + q)(2st + s + t))p + p^2(r + 1)(r + s)(r + t)r + ((q + 2)r + q)((s + t + 2)r + 2st + s + t))/$$
$$((r + p)(r + q)(r + s)(r + t))$$

‹ Back | Close | Accept replacement | Cancel variable choice | Cancel [transform] button

Both alternatives are significant improvements over the original framed expression (1), but both numerators are still lengthy with no easily discerned pattern. The reasons for the factored denominator in "Polynomial + proper fraction" are:

- The factored denominator was already computed for the alternative "Factored wrt r".

- The factored denominator is much more compact and informative than the fully expanded original denominator.

- There is nothing in the phrase "Polynomial + partial fraction" that promises displaying the denominator expanded with respect to r that was used to compute the polynomial part and the numerator.

- It is easy to frame the factored denominator then expand it if desired.

Although the ConciseForm result is slightly more compact, perhaps I could improve the numerator of the proper fraction result because I requested no more than a polynomial plus a proper fraction, and the quickest path to that goal was to expend no extra effort on the numerator beyond the collection with respect to the main variable r that was already done. Therefore in either the dialog box that contains the elided or the complete version of the proper fraction, I frame the entire numerator, right click, then choose "transform". This recursively opens up a new dialog box to choose a main variable, for which I again choose r for consistency. The resulting displayed alternatives include the factored numerator

$$(r(p + q + 2) + 2pq + p + q)(r(s + t + 2) + 2st + s + t).$$

This is much more compact, with insightful symmetries $p \leftrightarrow q$, $s \leftrightarrow t$ and $[p, q] \leftrightarrow [s, t]$ that are also true of the denominator. Therefore I accept this replacement in this *sub-problem* that is an alteration of the "Polynomial + proper fraction" alternative, thus transforming the expression in that dialog box to

$$pr + \frac{(r(p + q + 2) + 2pq + p + q)(r(s + t + 2) + 2st + s + t)}{(r + p)(r + q)(r + s)(r + t)}. \tag{2}$$

This is the nicest overall result so far, but before accepting it, I notice that although every factor contains r, the first numerator factor and the first two denominator factors are free of s and t, whereas the second numerator factor and the last two denominator factors are free of p and q. From experience I know that for two ratios having disjoint variable sets or nearly so, common denominators almost always increase bulk because there can be very little cancellation in the resulting numerator.

Thus conversely, partitioning the ratio in (2) into a ratio containing $\{r, p, q\}$ and a ratio containing $\{r, s, t\}$ then transforming each ratio to partial fractions *might* reduce bulk. Consequently, I drag the first numerator factor left of the ratio giving

$$pr + (r(p + q + 2) + 2pq + p + q)\frac{(r(s + t + 2) + 2st + s + t)}{(r + p)(r + q)(r + s)(r + t)}.$$

Then I drag the first two denominator factors under the former numerator factor giving

$$pr + \left(\frac{r(p + q + 2) + 2pq + p + q}{(r + p)(r + q)}\right)\left(\frac{r(s + t + 2) + 2st + s + t}{(r + s)(r + t)}\right). \tag{3}$$

(With the ability to select several non-adjacent subexpressions, I could instead select in (2) the first factor of the numerator and the first two factors of the denominator, then right click, then choose an action named something such as "collect", "group" or "isolate" – or perhaps directly choose partial fractions.)

Next I highlight the left ratio in (3), choose r as the main variable, then accept partial fraction expansion with respect to r, then do similarly for the right factor, giving

$$pr + \left(\frac{p + 1}{r + p} + \frac{q + 1}{r + q}\right)\left(\frac{s + 1}{r + s} + \frac{t + 1}{r + t}\right). \tag{4}$$

This is a very gratifying result compared to the equivalent input (1), and I am happy with the ordering of the terms and factors. Therefore I press the $\boxed{\text{Accept result}}$ button, which opens the dialog
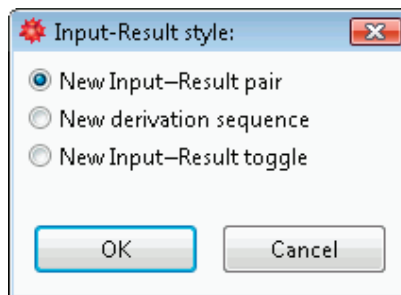
If I choose "New Input-Result pair", then the main computer algebra session window would be updated with a numbered input such as $Input_4$: Transform($Result_3, \ldots$) followed by expression (4) preceded by a label such as $Result_4$. The purpose of the Input line is to capture a programmatic way to transform $Result_3$ to $Result_4$ for purposes such as scripting. Most users will not want to view the ugly details, but if I click on the ellipsis in Transform($Result_3, \ldots$), then it expands to a procedure that generates $Result_4$, such as

```
Block ( Local (temp1 , temp2 , temp3 , temp4 ) ,
   temp1 := PolynomialPlusProperFraction ( Result3 , r ) ,
   temp2 := Factor ( Numerator ( Second ( temp1 )) , r ) ,
   temp3 := Factor ( Denominator ( Second ( temp1 )) , r ) ,
   First ( temp1 ) +
      PartialFractions ( First ( temp2 )/( First ( temp3 )* Second ( temp3 )) , r )*
      PartialFractions ( Second ( temp2 )/( Third ( temp3 )* Fourth ( temp3 )) , r ));
```

The "New derivation sequence" choice is similar, except it generates a collapsible sequence of such pairs using labels such as Intermediate$_{4,1}$, Intermediate$_{4,2}$, ...

The "New Input-Result toggle" is similar to the "New derivation sequence", except listing a single expression labeled with a two-button setter bar.

3.2 Transforming equations, inequalities and Boolean expressions

The primary activity that users want to do with equations, inequalities and systems thereof is to transform them into explicit solutions, so why force users to learn numerous different function names with different parameter semantics for solving different kinds of equations? If the user clicks the $\boxed{\text{Transform}}$ button when an entire equation, inequality or system thereof is framed, then the wizard tries to return solutions. Here is an example for a differential equation:
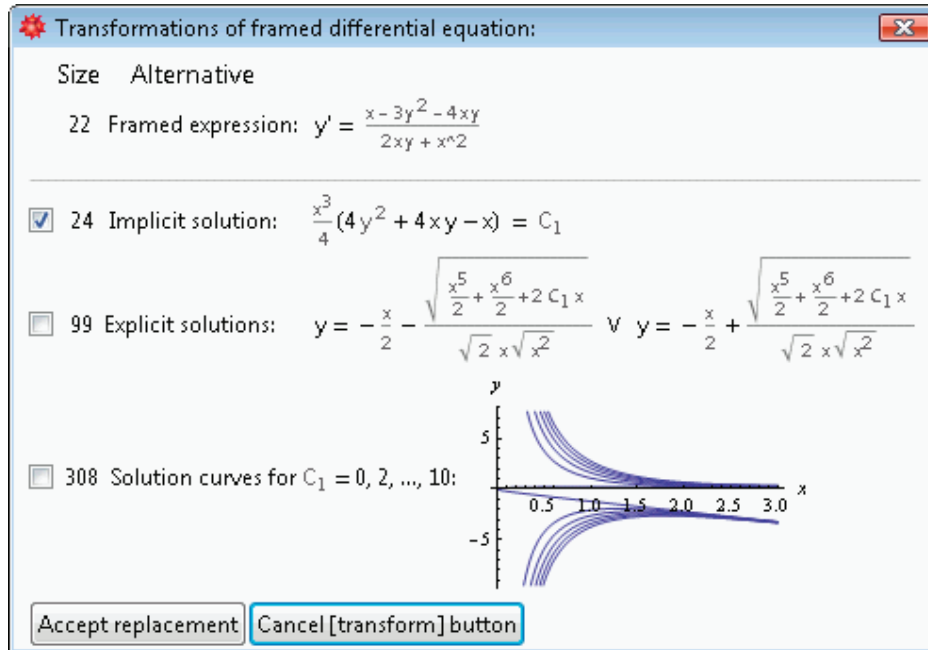
Notice the synergy of multiple views of the solutions. When the mouse pointer is over a curve, it is attached to a call-out displaying the corresponding explicit solution containing the associated numeric value for C_1. The plot range for x and the set of values for C_1 were chosen to insure that y is real. Checking this alternative opens a dialog with a magnified plot that gives the user control over the plot ranges for x and y together with the set of numeric values for C_1. The listed size of 308 is correlated with the area of the plot in the session window if accepted.

Here is an example for a system of two nonlinear algebraic equations:
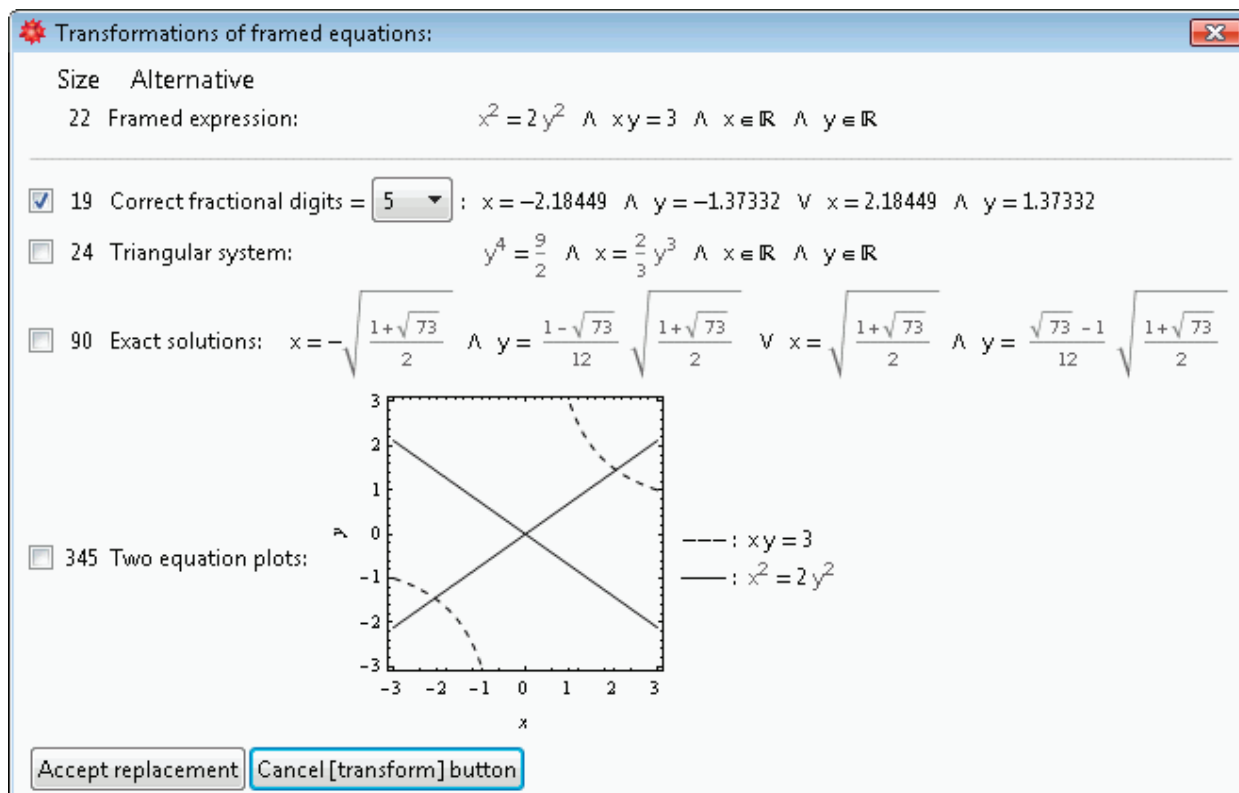
- The approximate solution was computed with *adaptive precision interval arithmetic* to deliver guaranteed requested accuracy initially set to correspond to six significant digits for nonzero components. If interval arithmetic is inapplicable or is taking too long for the specified accuracy, then the wizard tries *adaptive significance arithmetic*. If that is also taking too long, the wizard switches the popup digits setting to "IEEE" double and uses that. The corresponding displayed phrases are "Estimated correct fractional digits" or "Approximate solutions of unknown accuracy".

- The triangularized system is a reduced lexicographic Gröbner basis, which might be the preferred alternative for parametrized systems having exact explicit solutions that are messy or require unendurable time to complete. If the system included inequalities, then there would be a cylindrical algebraic decomposition instead.

- The x and y ranges in the plot were automatically set to include a margin around the convex hull of all the isolated finite solutions – a margin small enough to resolve detail near the solutions but large enough to provide useful context.

Here is an example of transforming a Boolean expression:
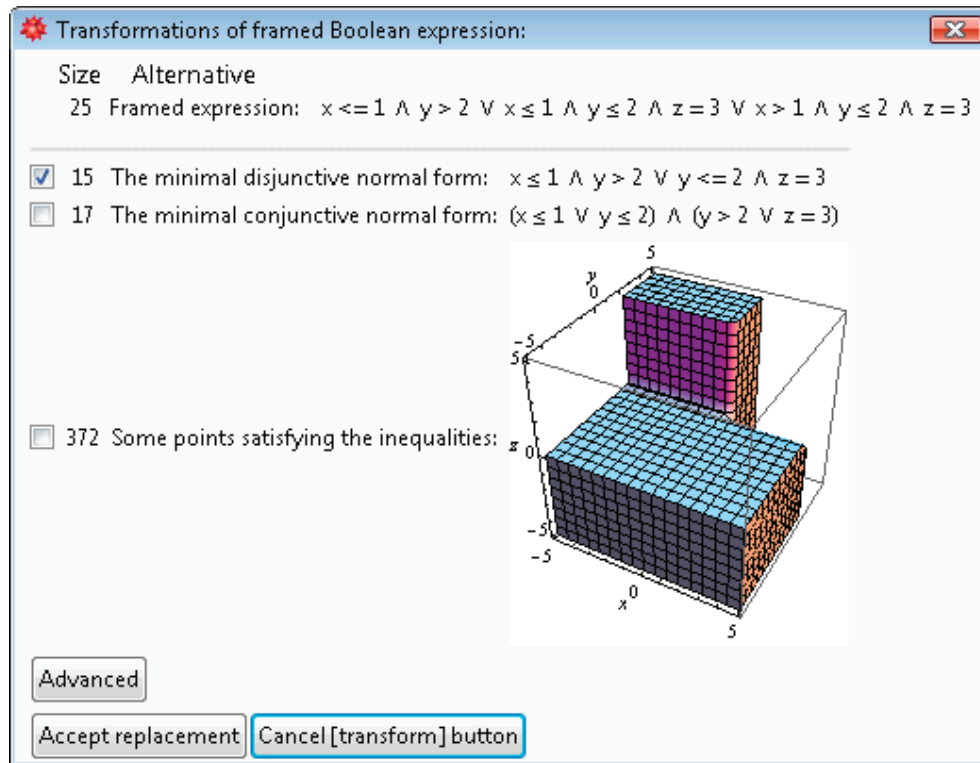
The ⟨Advanced⟩ button lists alternatives expressed in terms of nands, nors, etc.

3.3   The wizard is helpful even for mere numbers

Common useful internal representations for exact numbers are rational numbers and irrational constant expressions such as $\sqrt{\pi} + \ln 2$. Common representations for approximate numbers are:

1. software variable-precision Floats – preferably with adaptive significance tracking,

2. IEEE double Floats to take advantage of fast hardware instructions;

3. intervals whose endpoints are each independently an exact rational number or a Float – preferably adaptive precision for floating-point endpoints, and allowing a disjoint union of such intervals with either open or closed endpoints.

Before commencing a computation a user might want to transform from an exact to an approximate representation to make the computation faster – or from an approximate to an exact representation to avoid rounding errors. *After* a computation a user might want to convert from an approximate to an exact representation to attempt recovering an exact result – or from an exact to an approximate representation to make lengthy exact numbers more comprehensible or faster for purposes such as plotting. Also, at the end of a computation a user might want to simply alter the *display* of a number, such as displaying an exact rational number factored or as an integer plus a proper fraction or as a decimal fraction or in scientific form with a particular number of fractional or significant digits. The wizard makes it easy to do these transformations.

3.3.1 Alternate forms for rational numbers

If the framed subexpression is

$$\boxed{\frac{13717421001 37174210}{10973936901}},$$

then the wizard could offer the dialog box
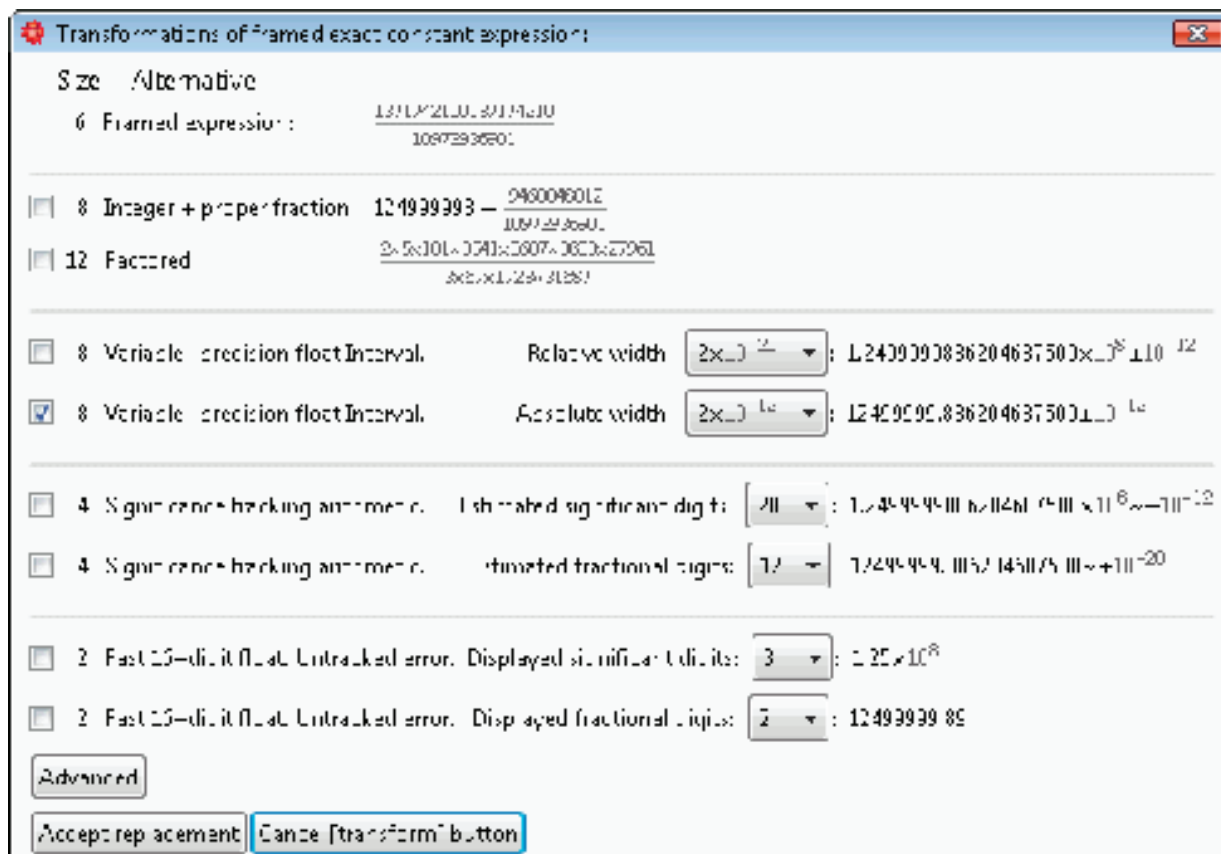
In accordance with the recommendations of [27], the approximate alternatives are ordered from intervals through bare <u>IEEE Floats</u> to encourage more use of arithmetic that is closer in spirit to exact arithmetic. The $\boxed{\text{Advanced}}$ button could offer p-adic, continued fraction, and different radix representations.[5]

Notice that this dialog provides useful supplementary information about the framed number even if the user never intended to replace the framed number: The user now has a good estimate for its magnitude, can see that it is well approximated by 1.25×10^8, and that both the numerator and denominator are composite but square free.

---

[5]In most systems, default simplification would immediately transform a factored or continued fraction or integer plus proper fraction form of a rational number back to a reduced ratio. Thus special passivity is required to make such volatile forms appear in results, and such non-idempotent forms are quite likely to disappear when such results are used in subsequent inputs. For this reason, many systems return a factored rational number as a list of pairs of bases and exponents, etc. The wizard must correct for such impediments to direct substitution of transformed subexpressions into the expression from which it came, suppressing default simplification where necessary to preserve the displayed overall result in standard mathematical form.

3.3.2   Alternate forms for intervals and Floats

If the framed subexpression is an interval, an IEEE double or a variable-precision Float, then the wizard could offer alternatives including approximating the number with an exact rational or irrational constant. For example, if the framed number was the IEEE double $\boxed{7.024814731040727}$ or a reasonably close approximation to it, albeit perhaps *displayed* with fewer significant digits, then the wizard could offer
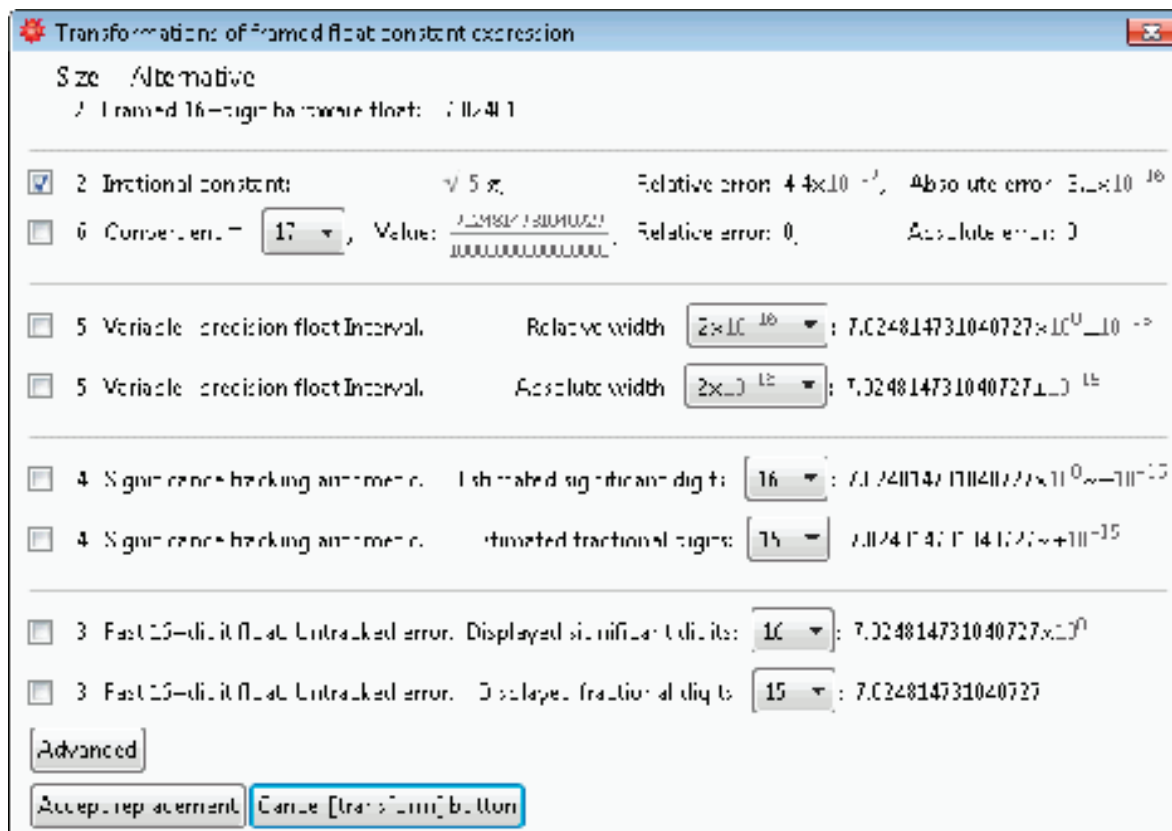
Details matter. For example:

- The alternate expressions are aligned, if practical, to make it easier to compare them.

- If the framed Float displayed few digits, the initial displayed digits for alternatives displays all or many digits – and *vice versa*.

The delightful alternative $\sqrt{5}\,\pi$ was computed quickly by the Maple identify(...) function, for which there is a more powerful free stand-alone version on the internet [5].

3.3.3   Alternate forms for non-real numbers

For non-real numbers, which of *rectangular*, *unit polar* and *exponential polar* form is most attractive depends on the particular number. For example, compare

$$
\begin{array}{ccccc}
\textit{Rectangular} & & \textit{Unit polar} & & \textit{Exponential polar} \\[2mm]
\boldsymbol{7 + 5i} & = & (-1)^{\arctan(5/7)/\pi} & = & \sqrt{74}\,e^{\arctan(5/7)\,i}, \\[3mm]
-2\sin\left(\dfrac{3\pi}{14}\right) + 2\cos\left(\dfrac{3\pi}{14}\right)i & = & \boldsymbol{2(-1)^{5/7}} & = & 2e^{\,5\pi i/7}, \\[3mm]
2\cos(1) + 2\sin(1)\,i & = & 2\,(-1)^{1/\pi} & = & \boldsymbol{2e^{i}}.
\end{array}
$$

Default simplification would ideally *display* the form that is most concise for each non-real number in a result even if a different form is used internally. However, optional transformations can conveniently offer all three alternatives.

4   Design issues and their resolutions

Challenging design issues include deciding:

1. What set of transformations should the wizard consider?

2. How can the wizard quickly estimate the number of applicable transformations without knowing the next variable choice?

3. When the next variable choice is known or $\boxed{\text{None}}$, how can the wizard quickly determine what subset of transformations are applicable and schedule them so that a worthwhile number are completed quickly?

This section addresses these issues and a few others. However, there are so many transformations that the wizard should know about that this section concentrates on those that help resolve issues 2 and 3. For more completeness, the Appendix discusses additional transformations that are relevant to the rational aspects of expressions. Transformations of the *irrational* aspects of expressions is too large a topic for treatment in this article.

First, three simple definitions:

Definition. *Default simplification* is the result of pressing the $\boxed{\text{Enter}}$ key or else perhaps $\boxed{\text{Shift}}$ $\boxed{\text{Enter}}$ with the factory-default mode settings and no transformational or simplification functions anywhere in the input expression.

Definition. A *functional form* is an expression of the form

$$f\left(\textit{expression}_1,\ \textit{expression}_2,\ \ldots\right)$$

where f is any function name.

Definition. *Generalized variables* are the smallest subtrees of an expression tree that are not a sum, difference, product, ratio, rational number, Float, or reasonably regardable as an integer power.

For example, z, π, i, $\cos(x + f(2))$, $z^{1/5}$, and $3^{1/5}$ are generalized variables. In contrast, $3/4$, x/y, and $x - 3$ are not. Also, $z^{2/5}$ and $3^{2/5}$ are not generalized variables, because they can be regarded as $(z^{1/5})^2$ and $(3^{1/5})^2$.

The importance of generalized variables is that transformations that are applicable with respect to variables in an expression can also be applicable with respect to generalized variables in an expression. For example, a user might want ordering, expansion or factoring with respect to to π and or $\cos(x + f(2))$. Additional transformations might be applicable with respect to generalized variables that are not merely indeterminates, such as $\cos(x + f(2)) \rightarrow \sin(\pi/2 - x - f(2))$, $\cos(x + f(2))^2 \rightarrow 1 - \sin(x + f(2))^2$, or $\pi \rightarrow 3.14159$.

As a prerequisite to discussing the wizard, it is helpful to organize the most important optional transformations offered by most computer algebra systems into categories of related transformations. For simplicity, the discussion addresses only constant ground domains that are common scalar numeric domains of characteristic 0. However, much of the discussion is relevant to other ground domains such as $\mathbb{Z}_m$ or {true, false}.

4.1 Different transformations for different generalized variables

> *"To each his own."*
> – Cicero

> *"I got different strokes for different folks."*
> – Muhammad Ali

Often users want certain transformations such as expansion or factoring only with respect to certain variables. For example:

- To compute the integral of $\left(x^5 + (c + 1)^{999}\, x + 1\right)^2$ with respect to x, it is helpful to expand with respect to x, but foolish to expand with respect to c.

- To solve
$$\left(c^{999} - 1\right)\left(z^2 - 1\right) = 0 \quad | \quad c^{999} \neq 1$$
it is helpful to factor with respect to z, but foolish to factor with respect to c.

In these cases we would prefer either concise or mere default simplification with respect to c.

When the user requests successive transformation for successive variables, we do not want to destroy transformations done for prior variables. Consequently, requested transformations are automatically mapped into the largest subexpressions that do not contain variables that have already been treated. For example:

1. If the alternative for *expanding* a framed expression with respect to the chosen main variable x is
$$\left(y^2 + 2y + 1\right)x^2 - y^2 + 2y - 1,$$
then *factoring* this alternative with respect to y gives $(y + 1)^2 x^2 - (y - 1)^2$ rather than

$$((y + 1)x + y - 1)\left((y + 1)x - y + 1\right).$$

2. If the alternative for *factoring* a framed expression with respect to the chosen main variable x is
$$(y-1)(y+1)\left((y-2)(y+2)x+(z+1)^2\right)(x+(2y+1)(2y-1)),$$
then *expanding* this alternative with respect to y gives
$$(y^2-1)\left((y^2-4)x+(z+1)^2\right)\left(x+4y^2-1\right).$$

3. If the alternative for *factoring* a framed expression with respect to the chosen main variable x is
$$\left(y^2-1\right)\left((y^2-4)x+(z+1)^2\right)\left(x+4y^2-1\right),$$
then *factoring* this alternative with respect to y gives
$$(y-1)(y+1)\left((y-2)(y+2)x+(z+1)^2\right)(x+(2y-1)(2y+1)).$$

(Notice that the wizard factored not only the *coefficients* of powers of x with respect to y, including the coefficient of the zeroth power of x, but also the top-level *content* y^2-1, because none of these contain x.)

4. If the alternative for *expanding* a framed expression with respect to the chosen main variable x is
$$\left((z+1)^9y^2+y+3\right)x^2+(z+1)^9x+(y+1)(y-1),$$
then *expanding* this alternative with respect to y gives two distinct alternatives: *distributed* form
$$(z+1)^9x^2y^2+yx^2+3x^2+(z+1)^9x+y^2-1,$$
and the often more concise *recursive* form
$$\left((z+1)^9y^2+y+3\right)x^2+(z+1)^9x+y^2-1,$$
both of which are offered to the user. Expansion of $(z+1)^9$ will be offered if the user proceeds to that last remaining variable rather than balking or accepting an alternative already displayed.

Now consider the input $\sin(x)\left(\cos(2y)+1\right)\cos(x)$. If the user is allowed to choose trigonometric expansion of multiple angles for the generalized variable $\cos(2y)$ but choose the opposite transformation for $\sin(x)$ and $\cos(x)$, then this product can transform to the particularly concise equivalent $\sin(2x)\cos(y)^2$ because $\cos(2y)\equiv 2(\cos(y)^2-1)$ and $\sin(x)\cos(x)\to\sin(2x)/2$.

Thus for maximum flexibility:

1. The user should be able to choose the order of generalized variables.

2. The user should be able to choose separate transformations for each generalized variable.

3. The choices for each variable should include conciseForm(...) and mere reordering with any associated default simplification when such results differ from the framed subexpression.

4. Where there is expansion with respect to two or more successive variables, both distributed and recursive forms should be offered if they are not identical.

4.2 Control over the order of generalized variables

> *"Order is the shape upon which beauty depends."*
> – Pearl S. Buck

Subsection 4.1 discussed how collection of similar powers of a generalized variable can be recursively applied to the resulting collected coefficients to perform transformations for successive generalized variables in any order. However, current computer algebra systems would nonetheless impose their built-in ordering rules to the resulting factors and terms. Therefore the displayed ordering of factors in terms and of terms in multinomials might not correspond to the recursive most main to least main order in which the user has treated successive variables. For example after requesting expansion with respect to y with coefficients that are factored with respect to all other variables a user might obtain a result such as

$$x^3 + y^3(z+1)^2 + y^2 z$$

rather than the more appropriate result

$$(z+1)^2 y^3 + zy^2 + x^3.$$

Also, regardless of the requested order, Newton's definition of force might be displayed as

$$f = am \tag{5}$$

which is visually quite disturbing despite its compliance with the usual alphabetical ordering convention for variables in a monomial. Consequently:

> Optional transformations should include control over the displayed ordering of factors and terms.

The few systems that give such control tend to do so indirectly and incompletely via control over the ordering of generalized variables. For example:

- In the Maxima computer algebra system the desired traditional order displayed in result equation (5) can be accomplished by a declaration such as ordergreat(a, m) or orderless(m, a). If the user has issued an ordergreat(...) and a non-conflicting orderless(...) declaration, then all other variables order between the least of the great and the greatest of the least, alphabetically. The effect is global from the time of a declaration until all declared orders are deleted with an unorder() declaration, which *must* be used between any two ordergreat(...) declarations or between any two orderless(...) declarations. Maxima also provides another mechanism for overriding default alphabetical ordering:

  $$\text{declare}\,(variable_1,\ property_1,\ variable_2,\ property_2,\ \ldots)$$

  gives each variable the corresponding property. Possible properties include *constant*, *scalar*, and *mainvar*. The command remprop(*variable*, *property*) can be used to remove such a declaration. Using $\prec$ to represent "less main",

  $$constants \prec scalars \prec undeclared \prec mainvars.$$

  By default, alphabetical order is used within each of these categories.

149

- The Reduce computer algebra system has an order declaration that is similar to ordergreat(...), except that more than one cumulative order declaration is allowed before a declaration

$$\text{order nil;}$$

which clears all such ordering declarations. The Reduce order declaration also accepts functional forms and built-in literal constants such as π, which is important.

- These and some other systems provide some control over the ordering of special distributed polynomials for Gröbner bases, but that is not very helpful for controlling the ordering of factors and terms in *general* expressions.

4.3   Common alternate forms for the *rational* aspect of expressions

Many computer algebra systems have separate functions for common denominators, various factorization levels, and various levels of polynomial or partial fractions expansions. This subsection describes how, for any particular ordering of generalized variables, these traditionally disparate concepts can be organized into a single topologically sorted list of partially-ordered alternatives varying from the most complete commonly-named factorization through the most complete commonly-named expansion offered by many computer algebra systems. This *organizing principle* greatly simplifies the transformation wizard by preventing selection of a set of contradictory transformations and by making the trade-off consequences in this list more obvious.

This subsection concerns only addition, subtraction, multiplication, division and integer powers, but most of the ideas also apply recursively to rational compositions of generalized variables and to fractional powers. Moreover, this subsection discusses only factoring, common denominators and expansion because they are most relevant to estimating quickly how many transformations are applicable for each variable and for quickly determining exactly which common transformations are applicable for a particular variable. The Appendix discusses additional rational transformations.

4.3.1   Reasons for common denominators, factoring and expansion

Definition. A **candid expression** is one that is not equivalent to an expression that visibly manifests a simpler expression class [24] .

As counterexamples:

- The expression $x(y+1) - xy$ is *not* candid because it contains the superfluous variable y.

- The expression $(x+1)^2 - x^2$ is *not* candid because it appears to be quadratic but is actually linear.

- The expression $(x+1)/(x^2 + 2x + 1)$ is *not* candid because it is equivalent to $1/(x+1)$, which has lower numerator and denominator degrees.

For most computer algebra systems, any amount of factoring *includes* reduction over a common denominator. Reduction over a common denominator yields a candid form for rational expressions, because the resulting form has no superfluous variables and has maximum possible cancellation of poles with coincident zeros.

4.3.2 A *univariate* partially-ordered set of factorization and expansion levels

1. For univariate factoring there are names for certain amounts of exact or approximate factoring based on multiplicities and the desired numeric coefficient domain of the factors:

   (a) *term primitive*,[6]

   (b) *square free*,

   (c) *over the integers* $\mathbb{Z}$,

   (d) *over the Gaussian integers* $\mathbb{Z}[i]$,

   (e) *over particular algebraic extensions*[7]

   (f) *exact reasonably absolute*,[8]

   (g) *exact absolute*,[9]

   (h) *approximate absolute over the floating-point real numbers* $\tilde{\mathbb{R}}$,

   (i) *approximate absolute over the floating-point complex numbers* $\tilde{\mathbb{C}}$.[10]

2. For systems that support variable precision Floats, users can choose the precision level for alternatives (h) and (i). For systems that offer significance and/or interval arithmetic, those alternatives immediately before (h) for real numbers and before (i) for non-real numbers.

---

[6]The *term content* of a univariate polynomial is the gcd of the numeric coefficients times the smallest occurring power of the variable. Factorization into the term content times the *term primitive part* forces a common denominator if any coefficient has a denominator, because with polynomials A, B, C and D,

$$\frac{A}{B} + \frac{C}{D} \equiv AB^{-1} + CD^{-1} \quad \to \quad (AD + BC)B^{-1}D^{-1} \equiv \frac{AD + BC}{BD} \quad \to \quad \frac{(AD + BC)/G}{(BD)/G}$$

where $G \leftarrow \gcd(AD + BC, BD)$.

[7]As a convenience in *Mathematica*, Factor[*expression*, Extension $\to$ Automatic] automatically uses extensions implied by the complex unit i and/or any radicals *present* in *expression*. For example,

$$\text{Factor}[x^2 + x - 2 + \sqrt{2}, \text{Extension} \to \text{Automatic}] \quad \to \quad -\left(-x - 1 + \sqrt{2}\right) \cdot \left(x + \sqrt{2}\right)$$

However, Factor[$x^2 + 2 \cdot \sqrt{2}{\cdot}x - 1$, Extension $\to$ Automatic] $\not\to$ $(x + \sqrt{2} - \sqrt{3}) \cdot (x + \sqrt{2} + \sqrt{3})$ because $\sqrt{3}$ is not in the given polynomial. We must instead use Factor[$x^2 + 2 \cdot \sqrt{2}{\cdot}x - 1$, Extension $\to \{\sqrt{2}, \sqrt{3}\}$] to obtain this factorization, but how many users would know to include $\sqrt{3}$?

[8]This is what is usually expected of algebra through calculus students for purposes such as solving equations or integrating rational functions: Algebraic extensions implied by radicals in the input together with use of the quadratic formula and n^{th} roots to factor binomials. *Derive* offers this option but also includes cubic and quartic formulas, which tends to generate unreasonably messy factorizations.

[9]This means whatever algebraic extension is necessary to factor the polynomial as much as possible, without the extension being provided by the user. Reference [9] discusses some algorithms for this. Some systems appear to use absolute factorization in their functions that solve systems of polynomial equations and integrate, but unfortunately appear not to offer it as a built-in factorization option. Therefore many computer algebra systems cannot directly factor $x^2 + 2x - 1$ into $(x + 1 + \sqrt{2})(x + 1 - \sqrt{2})$ without assistance, which any beginning algebra student can do!.

Attempted exact absolute factorization might consume an intolerable amount of computing time, or resulting factors might entail intolerably messy nested radicals or intolerably messy subexpressions containing functional forms named something such as Root. This is why I list the *exact reasonably absolute* level of factorization.

[10]Alternatives (h) and (i) are *approximations* rather than equivalence transformations. Some methods for exact absolute factorization begin from an approximate absolute factorization that is often preferable to the resulting messy exact factorization!

3. With $F_1 \succeq F_2$ denoting the fact that for a given example, a factorization at level F_1 is either identical to a factorization at level F_2 or is a further splitting of the factorization at level F_2, we have

$$\mathbb{Z}[i] \ \succeq \ \mathbb{Z} \ \succeq \ \text{square free} \ \succeq \ \text{term primitive},$$

$$\text{exact absolute} \ \succeq \ \text{specific algebraic extensions} \ \succeq \ \mathbb{Z},$$

$$\text{exact absolute} \ \succeq \ \text{reasonably exact absolute} \ \succeq \ \mathbb{Z},$$

$$\tilde{\mathbb{C}} \ \succeq \ \tilde{\mathbb{R}} \ \succeq \ \mathbb{Z},$$

$$\tilde{\mathbb{C}} \ \succeq \ \mathbb{Z}[i].$$

Thus these factorization levels form a *directed acyclic graph.* that we can topologically sort into one of several alternative lists, such as order 1(a) through 1(i) above.[11]

4. If we fully expand the product of the numerator factors and the product of the denominator factors of a reduced ratio, then we have the reduced ratio of two fully expanded polynomials. Despite the common denominator, the result is an expanded polynomial when this reduced denominator is 1 or when both the numerator and denominator are numeric. Therefore this form is on the borderline between factored and expanded.

5. The computer algebra built-into Texas Instruments hand held, Windows and Macintosh products has a function propFrac(*expression*, *variable*) that expands *expresssion* into a expanded polynomial with respect to *variable* plus a reduced ratio of two polynomials that is *proper* with respect to *variable*. The propFrac(...) function can easily be implemented using a polynomial quotient and remainder function, and the resulting form is an appropriate next node in our partial ordering from most factored to most expanded. This form is often more concise than either a common denominator or a partial fraction expansion. For example, this form was a key intermediate step in the example of subsection 3.1. For canonicality:

   (a) The coefficients of the resulting expanded univariate polynomial part that are not complex Floats can be normalized to Gaussian rationals $\mathbb{Q}[i]$ or rationalized algebraic numbers.

   (b) The denominator of the proper ratio can be made unit normal as described in [25].

   (c) The numeric coefficients in the resulting proper ratio that are not complex Floats can be normalized to Gaussian integers or algebraic integers.

   (d) If all of the denominator numeric coefficients are complex Floats, then we can normalize their magnitudes – such as making the largest of the real and imaginary magnitudes in the denominator coefficients be 1.0.

6. Polynomial expansion can be regarded as a special case of propFrac(...) for when the denominator of the given reduced ratio is numeric – perhaps 1.

7. If the reduced ratio of two polynomials has a non-numeric denominator, then the relevant adjective phrases 1(a) through 1(i) above can be used to label successive nodes corresponding to the amount of denominator factorization for corresponding partial fraction expansions.

---

[11]Actually, different specific algebraic extensions generally form a directed acyclic subgraph because, for example, we could have any one, two or all three of the extensions $\sqrt{2}$, $\sqrt{3}$ and $\sqrt{5}$, giving more than one path to $\{\sqrt{2}, \sqrt{3}, \sqrt{5}\}$. These directed acyclic subgraphs are the field extension lattices of Galois theory.

8. However, the square-free aspect of partial fraction expansion has two variants in the partial ordering. In non-decreasing order of the amount of expansion, adjective phrases applicable to the square free aspect are:

   (a) *incomplete*, meaning multiples of all powers of the same square-free denominator factor are combined over a common denominator, and

   (b) *complete*, meaning instead that for each resulting *very proper* ratio $N(x)/D(x)^m$ with expansion variable x, $\deg_x(N(x)) < \deg_x(D(x))$.[12]

Whenever a resulting numerator has more than one term, we can distribute an associated denominator over the numerator terms. It is generally unwise to distribute a multinomial denominator over the numerator terms for purposes such as integration, and it almost always increases bulk. However, it is helpful to do such a distribution for purposes such as fragmenting a ratio into the greatest number of simplest possible pieces for angle sum expansion.[13] The wizard can display both alternatives when they are not identical.

Table 1 shows the named alternative forms for a univariate example of the reduced ratio of two expanded polynomials.

1. The first two rows and last two rows are approximations to all of the other rows, which are equivalent to each other.

2. The double line separating "ratio of expanded polynomials" and "polynomial + proper ratio" separates factored from expanded forms.

3. Factors that differ from those of the preceding row are boldface.

4. Wherever there is a sum in a numerator, the corresponding denominator can optionally be distributed over the terms of the numerator.

5. For each of these named levels an example can be constructed where it is more concise than all of the other named levels. Therefore all of the levels are important.[14]

6. If a system doesn't offer built-in support for all of these named levels and a wizard implementer is not inclined to add such support, then:

   (a) Missing intermediate factorization levels can be provided by over-factoring then expanding appropriate subsets of factors.

   (b) Missing expansion levels can be provided by over expanding then combining appropriate subsets of summands.

7. The input could be any of these expressions or any rational expression that is equivalent to one of these expressions. If an input contains Floats, then float-free alternatives can be obtained by using a function such as the Maple identify($\ldots$) function to determine close rational or irrational constants [5].

---

[12]In contrast, for the *incomplete* square-free partial fraction expansion we can guarantee only that $\deg_x(N(x)) < \deg_x(D(x)^m)$. Many systems offer only complete expansions, but incomplete expansions are adequate for most purposes and are often more concise!

[13]If you must distribute multinomial denominators over numerator terms, it is most efficient to wait until after the expansion is complete in other respects.

[14]There are also unnamed intermediate levels such as splitting some but not all of the square-free factors over $\mathbb{Z}$.

Table 1: A univariate expression partially ordered from most factored to expanded

Amount of factor or expand	Boldface parts are different from the alternative above them
$\tilde{\mathbb{C}}$	$\dfrac{1.5(z-1.37)(z+5.7)(z-1.07+0.76i)(z-1.07+0.76i)\cdots(z+1.09+0.18i)(z+1.09-0.18i)}{z(z-1)^2(z+i)(z-i)(z+1.414)(z-1.414)(z+1.13)(z-1.04+0.82i)(z-1.04-0.82i)\cdots}$
$\tilde{\mathbb{R}}$	$\dfrac{1.5\cdots(\boldsymbol{z^2-2.13z+1.71})(\boldsymbol{z^2-1.13z+0.9})(\boldsymbol{z^2-0.05z+0.24})\cdots(\boldsymbol{z^2+2.19z+1.23})}{z(z-1)^2(\boldsymbol{z^2+1})(z+1.414)(z-1.414)(z+1.13)(\boldsymbol{z^2-2.08z+1.76})(\boldsymbol{z^2+0.95z+1.5})}$
reasonably absolute $\left(\mathbb{Z}\left[i,\sqrt{2}\,\right]\right)$	$\dfrac{\boldsymbol{3(z^{12}+4z^{11}-9z^{10}+4z^9+z^8+13z^6-29z^5+7z^4+13z^3-25z^2+6z-6)}}{\boldsymbol{2}z(z-1)^2(z+i)(z-i)(\boldsymbol{z+\sqrt{2}})(\boldsymbol{z-\sqrt{2}})(z^5+z+3)}$
$\mathbb{Z}[i]$	$\dfrac{3(z^{12}+4z^{11}-9z^{10}+4z^9+z^8+13z^6-29z^5+7z^4+13z^3-25z^2+6z-6)}{2z(z-1)^2(z+i)(z-i)(\boldsymbol{z^2-2})(z^5+z+3)}$
$\mathbb{Z}$	$\dfrac{3(z^{12}+4z^{11}-9z^{10}+4z^9+z^8+13z^6-29z^5+7z^4+13z^3-25z^2+6z-6)}{2z(z-1)^2(\boldsymbol{z^2+1})(z^2-2)(z^5+z+3)}$
square free	$\dfrac{3(z^{12}+4z^{11}-9z^{10}+4z^9+z^8+13z^6-29z^5+7z^4+13z^3-25z^2+6z-6)}{2z(z-1)^2(\boldsymbol{z^9-z^7-z^5+3z^4-z^3-3z^2-2z-6})}$
term primitive	$\dfrac{3(z^{12}+4z^{11}-9z^{10}+4z^9+z^8+13z^6-29z^5+7z^4+13z^3-25z^2+6z-6)}{2z(\boldsymbol{z^{11}-2z^{10}+2z^8-2z^7+5z^6-8z^5+2z^4+3z^3-5z^2+10z-6})}$
ratio of expanded polynomials	$\dfrac{\boldsymbol{3z^{12}+12z^{11}-27z^{10}+12z^9+3z^8+39z^6-87z^5+21z^4+39z^3-75z^2+18z-18}}{\boldsymbol{2z^{12}-4z^{11}+4z^9-4z^8+10z^7-16z^6+4z^5+6z^4-10z^3+20z^2-12z}}$
polynomial + proper ratio	$\dfrac{3}{2}+\dfrac{\boldsymbol{18z^{11}-27z^{10}+6z^9+9z^8-15z^7+63z^6-93z^5+12z^4+54z^3-105z^2+36z-18}}{2z^{12}-4z^{11}+4z^9-4z^8+10z^7-16z^6+4z^5+6z^4-10z^3+20z^2-12z}$
term primitive partial fraction	$\dfrac{3}{2}+\dfrac{\boldsymbol{3}}{\boldsymbol{2z}}+\dfrac{\boldsymbol{15z^{10}-21z^9+6z^8+3z^7-9z^6+48z^5-69z^4+6z^3+45z^2-90z+6}}{\boldsymbol{2z^{11}-4z^{10}+4z^8-4z^7+10z^6-16z^5+44^5+6z^3-10z^2+20z-12}}$
incomplete square free part frac	$\dfrac{3}{2}+\dfrac{3}{2z}+\dfrac{\boldsymbol{3z+3}}{\boldsymbol{(z-1)^2}}+\dfrac{\boldsymbol{12z^8-3z^6-3z^4+36z^3-18z+24}}{\boldsymbol{2z^9-2z^7-2z^5+6z^4-2z^3-6z^2-4z-12}}$
complete square free part frac	$\dfrac{3}{2}+\dfrac{3}{2z}+\dfrac{\boldsymbol{3}}{\boldsymbol{(z-1)^2}}+\dfrac{\boldsymbol{3}}{\boldsymbol{2z-2}}+\dfrac{12z^8-3z^6-3z^4+36z^3-18z+24}{2z^9-2z^7-2z^5+6z^4-2z^3-6z^2-4z-12}$
partial fractions over $\mathbb{Z}$	$\dfrac{3}{2}+\dfrac{3}{2z}+\dfrac{3}{(z-1)^2}+\dfrac{3}{2z-2}+\dfrac{\boldsymbol{3z}}{\boldsymbol{z^2-2}}+\dfrac{\boldsymbol{3z}}{\boldsymbol{z^2+1}}+\dfrac{\boldsymbol{3z^2-12}}{\boldsymbol{2z^5+2z+6}}$
partial fractions over $\mathbb{Z}[i]$	$\dfrac{3}{2}+\dfrac{3}{2z}+\dfrac{3}{(z-1)^2}+\dfrac{3}{2z-2}+\dfrac{3z}{z^2-2}+\dfrac{\boldsymbol{3z}}{\boldsymbol{2z+2i}}+\dfrac{\boldsymbol{3}}{\boldsymbol{2z-2i}}+\dfrac{3z^2-12}{2z^5+2z+6}$
absolute partial fractions	$\dfrac{3}{2}+\dfrac{3}{2z}+\dfrac{3}{(z-1)^2}+\dfrac{3}{2z-2}+\dfrac{\boldsymbol{3}}{\boldsymbol{2z+2\sqrt{2}}}+\dfrac{\boldsymbol{3}}{\boldsymbol{2z-2\sqrt{2}}}+\dfrac{3z}{2z+2i}+\dfrac{3}{2z-2i}+\dfrac{3z^2-12}{2z^5+2z+6}$
partial fraction over $\tilde{\mathbb{R}}$	$\cdots+\dfrac{\boldsymbol{1.5}}{\boldsymbol{z+1.41}}+\dfrac{\boldsymbol{1.5}}{\boldsymbol{z-1.41}}+\dfrac{\boldsymbol{3.0z}}{\boldsymbol{z^2+1}}+\dfrac{\boldsymbol{0.162}}{\boldsymbol{z+1.1}}-\dfrac{\boldsymbol{0.18z-0.27}}{\boldsymbol{z^2-2.1z+1.8}}+\dfrac{\boldsymbol{0.018z+0.31}}{\boldsymbol{z^2+0.95z+1.5}}$
partial fraction over $\tilde{\mathbb{C}}$	$1.5+\cdots-\dfrac{\boldsymbol{0.089+0.049i}}{\boldsymbol{z-1.04+0.82i}}-\dfrac{\boldsymbol{0.089-0.049i}}{\boldsymbol{z-1.04-0.82i}}+\dfrac{\boldsymbol{0.0088+0.13i}}{\boldsymbol{z+0.48+1.13i}}+\dfrac{\boldsymbol{0.0088-0.13i}}{\boldsymbol{z+0.48-1.13i}}$

4.3.3 *Multivariate* partially-ordered sets of factorization and expansion levels

Reference [26] describes how to do multivariate partial fraction expansion with respect to two or more successive generalized variables. As a degenerate case, the expansion is *polynomial* expansion with respect to variables that don't occur in the reduced common denominator of the given expression.

As shown that article, the number of terms in a multivariate partial fraction expansion can depend on the ordering of the expansion variables. Table 2 shows eight alternatives for factoring and or expanding a bivariate example over $\mathbb{Z}$. Notice how the partial fraction expansion with respect

to y in the last two rows introduces poles at $x = \pm 1$ into some individual ratios, forcing the use of a piecewise result to avoid contracting the domain of definition. Making a ratio proper can also cause this. (Unfortunately, most current computer algebra systems quietly do such domain reductions.) Common denominators remove these additively removable singularities.

Table 2: Some alternative recursive form factorizations and expansions over $\mathbb{Z}$.
f = **f**actored. e = **e**xpanded to incomplete partial fractions.

1st	2nd	Result
f_x	f_y	$\dfrac{(y-1)(y+1)(y^2+3)x^3 - y(y+1)(y^4-y^3+y^2-3y-4)x - 2y^2(y-1)(y^2+2)}{(x-y)(x+y)(y-1)(y+1)(y^2+2)}$
f_x	e_y	$\dfrac{(y^4+2y^2-3)x^3 - (y^6-2y^3-7y^2-4y)x - (2y^5-2y^4+4y^3-4y^2)}{(x-y)(x+y)(y^4+y^2-2)}$
f_y	f_x	$\dfrac{xy^6+2y^5-(x^3+2)y^4-2(x-2)y^3-(2x^3+7x+4)y^2-4xy+3x^2}{(y-x)(y+x)(y-1)(y+1)(y^2+2)}$
f_y	e_x	$\dfrac{xy^6+2y^5-(x^3+2)y^4-(2x-4)y^3-(2x^3+7x+4)y^2-4xy+3x^2}{(y-x)(y+x)(y-1)(y+1)(y^2+2)}$
e_x	f_y	$\dfrac{y^3+3}{y^2+2}x + \dfrac{2y^2}{(x+y)(y-1)(y+1)} + \dfrac{2y}{(x-y)(y-1)(y+1)}$
e_x	e_y	$x + \dfrac{x}{y^2+2} + \dfrac{2}{x+y} + \dfrac{1}{xy-x+y^2-y} + \dfrac{1}{xy+x+y^2+y} + \dfrac{1}{xy-x-y^2+y} + \dfrac{1}{xy+x-y^2-y}$
e_y	f_x	$\begin{cases} 1 - \dfrac{y+8}{6(y-1)^2} - \dfrac{11y+8}{6(y+1)^2} + \dfrac{1}{3(y^2+2)}, & \text{if } x=-1, \\ -1 + \dfrac{1}{(y-1)^2} - \dfrac{2y+1}{(y+1)^2} - \dfrac{1}{y^2+2}, & \text{if } x=1, \\ x + \dfrac{x}{y^2+2} + \dfrac{2x}{(x-1)(x+1)(x-y)} - \dfrac{2x^2}{(x-1)(x+1)(y+x)} + \dfrac{2x}{(x-1)(x+1)(y-1)} - \dfrac{2}{(x-1)(x+1)(y+1)}, & \text{otherwise} \end{cases}$
e_y	e_x	$\begin{cases} 1 - \dfrac{y+8}{6(y-1)^2} - \dfrac{11y+8}{6(y+1)^2} + \dfrac{1}{3(y^2+2)}, & \text{if } x=-1, \\ -1 + \dfrac{1}{(y-1)^2} - \dfrac{2y+1}{(y+1)^2} - \dfrac{1}{y^2+2}, & \text{if } x=1, \\ \cdots + \dfrac{2}{y+x} - \dfrac{1}{xy+y+x^2+x} + \dfrac{1}{xy-y+x^2-x} - \dfrac{1}{xy+y-x^2-x} - \dfrac{1}{xy-y-x^2+x} + \cdots - \dfrac{1}{xy-y+x-1} + \dfrac{1}{xy-y-x+1}, & \text{otherwise} \end{cases}$

4.4 Series and other approximations

The discussion so far has been about transformations to alternatives that are *equivalent* to the input wherever the input is defined – except perhaps approximating exact numbers in the input with approximate Floats or approximating Floats in the input with nearby rational numbers. This subsection instead addresses the equally important alternatives of transformations that *approximate* the input with *simpler expressions*.

Closed-form exact results aren't always obtainable. Even when they are, the results might be too bulky to convey needed insight or to permit fast enough well-conditioned evaluation for numerous floating-point values of the variables therein. Therefore, various kinds of approximation are useful transformations. Also it is important to realize that the ultimate destiny of many exact expressions is to substitute Floats into them, in which case the resulting rounding errors might exceed those caused by an approximate expression. Here are some examples of appropriate optional approximate transformations for a wizard to offer:

- Quadrature can often be used to determine a single approximate number for a definite integral.

- Approximate equation solving is often preferable even when compact explicit exact solutions are obtainable.

- Generalized infinite or truncated Laurent-Puiseux series (allowing, for example, logarithmic factors) can concisely approximate lengthy expressions. The wizard can initialize expansion points to ones that are most likely of interest, such as 0, infinities, and poles. If selected, the user can adjust the expansion points and the requested order.

- Padé approximations often have a larger region of convergence and greater computational efficiency than power series.

- Truncated Fourier or wavelet series are often more appropriate than expansions about a point.

A well-chosen approximation can be simpler than any obtainable exact result and yet retain all of the important characteristics of an exact result.

4.5   Generating the list of generalized variables

If a variable v in the framed subexpression has an assigned value, then it would be misleading to list that irrelevant v. However, we do want to consider listing some or all of the generalized variables in the *assigned value*, if any. We can use default simplification for this purpose, because it replaces all assigned variables with their values.

On many systems *default* simplification can easily produce results containing generalized variables that candid simplification would eliminate. For example, the default simplification of most systems merely reorders the factors and/or terms in the input $((x + 1)x - x^2 - x + 2) / (y^2 - 1)$, but all of the factoring and expansion transformations described in sub-subsections 4.3.2 and 4.3.3 transform the expression to $2/(y^2 - 1)^2$ or $2/((y - 1)(y + 1))$ or $1/(y - 1) + 1/(y + 1)$, which all depend on y alone.

It is helpful for the wizard to recognize such *superfluous* generalized variables. For polynomial expressions, expansion to recursive form always eliminates superfluous variables. For other rational expressions, reduction over a common denominator always does so.[15] Thus the first thing that the wizard can do is compute such a form to identify generalized variables that thereby disappear. One of these transformations can be initially checked to help encourage the user to eliminate superfluous generalized variables. However, the user might prefer to eliminate them in a way that alters cherished structure less drastically. To do this, the wizard could also offer the alternative, for example, "merely eliminate superfluous x and $\ln(y)$". This can be accomplished by substituting simple exact constants such as 0, 1 or -1 for those generalized variables, then applying default simplification. If the substitution value is at a removable singularity and thereby causes division by 0, then the wizard can backtrack and try another simple constant.

A generalized variable wouldn't be offered if doesn't affect ordering and no optional transformation is applicable to it. For example, it is pointless to list x if it only occurs as the argument in $\sin(x)$. However, it might be worthwhile to list the generalized variable $\sin(x)$, depending on how it occurs in the default-simplified result. As another example, *none* of the transformations or ordering choices being discussed here are applicable to the subexpression $3x^2$.

There will usually be an initial "Choose main variable" dialog if there is more than one applicable generalized variable. If so, and after choosing the transformation with respect to the selected main

---

[15]If this reduction produces $0/0$, then the expression is undefined for all values of its variables, so the one and only offered alternative can be the result of "$0/0$".

variable there is still more than one applicable generalized variable for some maximal subexpression that doesn't contain the main variable, then there will be another dialog labeled instead "Choose next most main variable", and so on until the user aborts the investigation or accepts replacements.

4.6 Estimating the number of common transformations for each generalized variable choice

Quick syntactic checks can identify some opportunities for transforming a framed subexpression with respect to a generalized variable. For example,

- Expansion is applicable to any variable that occurs in a multinomial raised to an integer power or a multinomial multiplied by any subexpression.

- A common denominator is applicable to any variable that occurs to a negative power in a sum.

- Angle sum expansion is applicable if the argument of a sinusoid is a sum.

- The transformations $\sin(u)^2 \to 1 - \cos(u)^2$ or $\sin(u)^n \to (\sin(nu) + \cdots)/2^n$ are applicable if $\sin(u)$ is raised to any integer power exceeding 1.

Quick syntactic checks can also *preclude* some opportunities for transforming a framed subexpression with respect to a generalized variable. For example, expansion is precluded if the framed subexpression is monomial or linear in the variable.

Another ordering heuristic is that the number of applicable transformations is likely to increase with the degree of a variable in a numerator or denominator, and more transformations are associated with high degree denominators than high degree numerators, because denominator factorizations can also enable partial fraction expansions.

If time remains in the 0.1 second acceptable delay for generating and displaying the first dialog box, then we can compute more accurate estimates for the number of common applicable transformations for each variable: For the rational aspects of a framed subexpression, minimal-effort reduction over a common denominator reveals the numerator and denominator degrees of every top-level generalized variable, which are all of those that can be affected by top-level factoring or expansion. Minimal effort here means using partially factored form – preferably recursive – as described in [24]. If this reduced partially factored form differs from the framed subexpression, then it is already one applicable transformation. Moreover, expansion to a polynomial plus a proper fraction is applicable to every such generalized variable whose degree in the resulting numerator is at least as large as the degree in the denominator; and it is easy to compute these degrees for partially factored forms.

If time still remains in the 0.1 second acceptable delay, then we can compute more accurate estimates by factoring the common denominator with respect to all of its generalized variables. From this it is easy to determine which successively lesser factorizations would combine two or more factors containing that main variable and thus determine the number of distinct named partial fraction expansions with respect to that variable and a lower bound on the number of distinct named factorizations over a common denominator.

If time still remains, then the wizard can factor the numerator to better estimate the number of named factorizations over a common denominator.

To reduce the chance of exceeding 0.1 second before *any* factorization is achieved, it could be done in levels, such as term content with respect to every variable, then square free, etc. through, say, exact reasonably absolute factorization followed by absolute factoring over the complex Floats. If the allotted 0.1 seconds runs out before completing this agenda, then we simply use the best estimates that we have at that time. Moreover, we can continue to refine and update the estimates after the initial display, without changing the initial order of the generalized variable buttons.

4.7   Recognizing applicable transformations

The exact and approximate factorizations with respect to all variables is a useful point of departure for computing alternative results regardless of what variable the user chooses. Therefore it is helpful to proceed computing those factorizations in the background while users ponder their choice of variable.

After a user chooses a variable, the wizard can complete the distinct factorization levels by appropriately expanding pairs of factors, which is fast. To convey the strongest possible information, when labeling the displayed alternatives or elided versions thereof, they are labeled with the most thorough applicable factoring level. For example, if the only distinct factorizations are factorizations over the Gaussian integers and over the integers, then the later would not be labeled "square-free" even though it is that as well. If there is only one factored alternative, then it could be labeled merely "factored" for brevity, but with a more detailed phrase displayed upon mouse-over. This provides a learning opportunity for mathematically unsophisticated users.

If there is a denominator and it contains the chosen variable, the wizard can then proceed to compute the polynomial part and proper fraction, followed by any distinct partial fraction levels with respect to that variable.

5   If I want this interface, why haven't I implemented it?

Good question. The reasons are:

- I am not an interface programmer.

- It is best done by a team.

- Some aspects will require access to proprietary internals that are inaccessible to outsiders for systems that aren't open source.

- It will require testing feedback from numerous users. Surely some of the ideas presented here won't work out well in practice, and better ideas will occur.

- If it isn't the default interface distributed with a computer algebra system, then it is unlikely to be known to most users, and the system is likely to evolve in a way so that the alternative interface no longer works.

- For various reasons, corporations are often unwilling to adopt and maintain packages written by outsiders as first class parts of their system – thoroughly and seamlessly integrated into the system and the documentation with no need for building or loading.

Therefore, although I would be delighted to help, the purpose of this manifesto is to stimulate computer algebra users to request better interfaces and stimulate decision makers to build them.

> *"If you build it, they will come."*
> – an apt misquote from *Field of Dreams*

Computer algebra users of the world: the squeaky wheels get the grease!

Acknowledgments

Appendix: More transformations for rational expressions

It is worthwhile to list in one place most of the many known transformations that might be of general interest. Subsection 4.3 discussed factoring, common denominators and expansion. Here are some additional transformations for the rational aspect of expressions:

A.1: Basic – of interest to most users

The wizard should automatically try the following transformations in the background because they can dramatically decrease the bulk of an expression and reveal important structure:

Polynomial shifts

Most of the factorization and expansion levels leave polynomial sub-expressions that often have more than two terms. Sometimes merely re-expressing such a multinomial in terms of optimally shifted variables can greatly reduce the number of terms and/or the size of coefficients. For example,

$$\left(y^2 - 4y + 4\right) x^3 + \left(3y^2 - 12y + 12\right) x^2 + \left(3y^2 - 12y + 12\right) x + y^2 - 4y + 1 \;\;\rightarrow\;\; (y-2)^2(x+1)^3 + 8,$$

$$531441\, x^6 + 2834352\, x^5 + 6298560\, x^4 + 7464960\, x^3 + 4976640\, x^2 + 1769472\, x + 262151 \rightarrow (9x+8)^6 + 7\,.$$

Articles [13, 14], give algorithms for computing optimal shifts. As a quick preclusion test, it is not worth shifting a polynomial with respect to a variable if the polynomial is monomial or binomial with respect to that variable.

Polynomial decomposition

Complementary to such shift decompositions, Kozen and Landau [17] describe an efficient algorithm for completely decomposing a univariate polynomial $p(x)$ into nested polynomials

$$p_1\left(p_2\left(\ldots\left(p_m\left(x\right)\right)\ldots\right)\right)$$

with each $p_k\left(t\right)$ of degree at least 2 in t. For example, the irreducible polynomial

$$\begin{aligned} P(x) \;\; &:= \;\; x^{12} + 4x^{10} + x^9 + 6x^8 + 3x^7 + 4x^6 + 3x^5 + x^4 + x^2 + 7 \\ &\rightarrow \;\; \left(x^3 + x\right)^4 + \left(x^3 + x\right)^3 + 7. \end{aligned}$$

Their algorithm also applies recursively to multivariate polynomials represented recursively. As a quick preclusion test, such decompositions are inapplicable with respect to a variable of degree less

than 4 or having few terms. There are also algorithms for other kinds of univariate and multivariate polynomial decompositions, as described, for example, in [12, 34, 33, 32, 35].

Polynomials rewritten in these ways can reveal significant structure, help precondition an expression for efficient repeated numeric evaluation or reduced rounding error, and facilitate solutions of higher-degree polynomial equations or systems of equations. For example, with the above decomposition, masochists could apply the quartic formula then the cubic formula to express the zeros of $P(x)$ in terms of radicals.

Linear combinations of powers

At least since Pythagoras, people have been interested in representing numbers and non-numeric expressions as sums, differences, or general linear combinations of powers of other expressions. For example, if an expression can be rewritten as a sum of even powers of real subexpressions, then the expression is thereby proven to be nonnegative for all real values of these subexpressions. Algorithms for such transformations can be found in, for example, [21, 22, 31].

A.2: Advanced – of interest only to experts

Here are some transformations that should be tried only in response to the Advanced button because they would probably intimidate and distract most users without any compensating benefit to them.

Expression in terms of orthogonal polynomials

Important optional transformations include a change of basis from monomials to orthogonal polynomials, which can

- yield more concise results,

- yield results less subject to magnified rounding errors,

- reveal patterns that otherwise wouldn't be apparent, or

- suggest efficient accurate min-max truncated approximations.

Many computer algebra systems provide functions that return one of various classic orthogonal polynomials of a specified degree in a specified variable using the monomial basis. However, most of the systems provide little or no automation for:

- converting ordinary polynomials to linear combinations of orthogonal polynomials specified by symbols such as the Chebyshev polynomials of the first kind $T_0(z)$, $T_1(z)$, ... ;

- propagating linear combinations of such polynomials into other linear combinations thereof *exactly* through rules such as

$$
\begin{aligned}
T_m(z)\,T_n(z) &\rightarrow \frac{1}{2}\,T_{|n-m|}(z) + \frac{1}{2}\,T_{m+n}(z), \\
T_m\left(T_n(z)\right) &\rightarrow T_{mn}(z), \\
\int T_n(z)\,dz &\rightarrow
\begin{cases}
\frac{1}{4}\left(T_0(x) + T_2(x)\right), & \text{if } n = 1, \\
\frac{1}{2-2n}\,T_{n-1}(z) + \frac{1}{2+2n}\,T_{n+1}(z), & \text{otherwise,}
\end{cases}
\end{aligned}
$$

and *approximately* through infinite or truncated series expansions;

- efficiently and accurately substituting Floats into expressions involving combinations of such functions: Rather than substituting a number z_0 for z in the monomial-basis representations of the symbols $T_0(z)$, $T_1(z)$, ... in a combination thereof, it is much faster and more accurate to compute the successive $T_k(z_0)$ from the recurrence $T_n(z_0) \leftarrow 2z_0 \, T_{n-1}(z_0) - T_{n-2}(z_0)$.

Of the various named orthogonal polynomials, $T_0(z)$, $T_1(z)$, ... are probably most important. Therefore, Trefethen and others [30] developed a powerful MATLAB package that automates effective use of these polynomials for many applications, using IEEE double Floats to represent the coefficients. Fateman [11] implemented some of these capabilities in Maxima to take advantage of its variable precision floating point and exact rational arithmetic. Such transformations and analogous ones for other orthogonal polynomials would be a welcome addition to many computer algebra systems.

Expression in terms of symmetric polynomials

Analogous transformations for the most common kinds of *symmetric polynomials* would be another welcome addition. Such polynomials can help reduce the curse of dimensionality for problems that exhibit symmetries when pairs of variables are interchanged. As a start toward this, the *Mathematica* function SymmetricReduction $[expression, \{v_1, v_2, \ldots v_n\}, \{s_1, s_2, \ldots, s_n\}]$ returns the pair $\{p, r\}$ where *expression* depends on variables $\{v_1, v_2, \ldots v_n\}$, p is the symmetric component of *expression* in terms of symbols $\{s_1, s_2, \ldots, s_n\}$ representing the *elementary symmetric polynomials* through degree n, and r is the residual of *expression* that can't be so represented. To convert a symmetric result back to the original variables, function SymmetricPolynomial $[m, \{v_1, v_2, \ldots v_n\}]$, returns the mth elementary symmetric polynomial using variables $\{v_1, v_2, \ldots v_n\}$. Sturmfels [28] contains algorithms for transforming to and from symmetric polynomials.

Rational Decomposition

We can attempt separate polynomial decompositions on every numerator and denominator polynomial in an expression. However, there are also algorithms for decomposing *ratios* of polynomials into *nested ratios*, as discussed in [4, 15, 39]. As an example from the first of these articles, but using recursive form and primitive normalization, the ratio

$$\frac{(y^2 + 2z^2 y + z^4 - 81)\, x^2 - 2y \cdot (y^5 + z^2 y^4 + 225\, z)\, x + y^2\, (y^8 - 625\, z^2)}{(y^2 + 2z^2 y + z^4 - 162)\, x^2 - 2y \cdot (y^5 + z^2 y^4 + 450\, z)\, x + y^2\, (y^8 - 1250\, z^2)}$$

$$\rightarrow \begin{cases} 1, & \text{if } 9x + 25zy = 0, \\[2ex] \dfrac{\left(\dfrac{(y + z^2)\, x - y^5}{9x + 25\, zy}\right)^2 - 1}{\left(\dfrac{(y + z^2)\, x - y^5}{9x + 25\, zy}\right)^2 - 2}, & \text{otherwise.} \end{cases}$$

This example also illustrates that rational decomposition can introduce new removable singularities in the nested form, such as on the manifold $9x + 25zy = 0$ for this example. We can avoid this by clearing the nested denominators but preserving the nested polynomial components thereof to obtain *correlated* polynomial decompositions of the numerator and denominator:

$$\frac{\left(\dfrac{(y + z^2)\, x - y^5}{9x + 25zy}\right)^2 - 1}{\left(\dfrac{(y + z^2)\, x - y^5}{9x + 25zy}\right)^2 - 2} \quad \rightarrow \quad \frac{\left((y + z^2)\, x - y^5\right)^2 - (9x + 25\, zy)^2}{\left((y + z^2)\, x - y^5\right)^2 - 2\, (9x + 25\, zy)^2}.$$

If desired, for this example we can then further factor the difference in two squares in the numerator and in the denominator to obtain the factorization over $Z[\sqrt{2}\,]$.

$$\frac{\left(\left(y + z^2 + 9\right)x - y^5 + 25\,zy\right)\left(\left(y + z^2 - 9\right)x - y^5 - 25\,zy\right)}{\left(\left(y + z^2 + 9\sqrt{2}\right)x - y^5 + 25\sqrt{2}\,zy\right)\left(\left(y + z^2 - 9\sqrt{2}\right)x - y^5 - 25\sqrt{2}\,zy\right)}.$$

The numerator factorization could easily have been computed from the original numerator. However, the required $\sqrt{2}$ algebraic extension necessary to factor the multivariate denominator would be more difficult to determine without the intervening rational decomposition.

Continued fractions

Continued fractions are another type of compound-fraction representation for rational expressions. Acton [1] lists three different variants of continued fractions together with algorithms for converting between them and a reduced ratio. Cuyt and Verdonk [10] review methods for multivariate continued fractions. As an example of a continued fraction expansion that reveals a simple pattern:

$$\frac{\left(z^4 - 105\,z^2 + 945\right)z}{15\left(z^4 - 28\,z^2 + 63\right)} \mid z^2 \notin \left\{63, \frac{45}{2}, \frac{3}{2}\left(35 \pm \sqrt{805}\right)\right\} \to \cfrac{z}{1 - \cfrac{z^2}{3 - \cfrac{z^2}{5 - \cfrac{z^2}{7 - \cfrac{z^2}{9}}}}}.$$

Here a constraint was appended to the input to avoid the *appearance* of contracting the domain of definition because of removable singularities introduced by the continued fraction. If instead we used a piecewise result, then it would have 9 pieces, 8 of which are constants that can be determined by substituting the two square roots of each element in the constraint set into the original expression.

Actually, if the computer algebra system automatically handles unsigned zeros and infinities correctly, then with exact computation the continued fraction form evaluates to the correct finite values even at these removable singularities. However, unlike the reduced ratio, the continued fraction form *might* be less accurate due to catastrophic cancellation near those introduced singularities. Therefore it is worth alerting the user to these singularities by either appending an input constraint or producing a piecewise result.[16]

Hornerized forms

In its simplest form, Horner's rule is the factoring out of the least power of a variable from successively lower-degree terms. For example,

$$- 1234321\,x^7 - 1234321\,x^5 + 2468642\,x^4 + 7x^3 + 14x - 21$$
$$\to \left(\left(\left(\left(-1234321\,x^2 - 1234321\right)x + 2458642\right)x + 7\right)x^2 + 14\right)x - 21.$$

---

[16]With correct handling of infinities, a continued fraction can be defined at infinity where a reduced common denominator is *not*. For example, $1/(1 + 1/z) \to 1$ at $z = \infty$, where the corresponding reduced ratio $z/(z+1) \to \infty/\infty$ is indeterminate. Most people would prefer having a result defined at all *finite* values of variables to being defined at *infinite* values, but a mere division can turn 0 into an infinite value.

It is also worth partially factoring out units and numeric content to the extent that it reduces bulk or the number of operations. For example,

$$- 1234321\, x^7 - 1234321\, x^5 + 2468642\, x^4 + 7x^3 + 14x - 21$$
$$\rightarrow ((-1234321((x^2 + 1)\, x - 2)\, x + 7)\, x^2 + 14)\, x - 21.$$

Horner's rule often leads to faster evaluation when substituting numbers for variables, which is particularly important in situations such as plotting, where substitution is done many times for different values. Horner's rule also often improves accuracy for approximate arithmetic, because the operands of a catastrophic cancellation are closer to the input numbers, hence less contaminated with rounding error.

Horner's rule can be viewed as factoring out term content term by term, starting with the highest-degree terms at each level. With this viewpoint, we can apply it to multinomials throughout an expression in all of the above special forms. Ceberio and Kreinovich [8] discuss greedy algorithms for computing efficient multivariate Hornerized forms.

Another transformation that can enable faster evaluation when substituting numbers for variables is to factor out an integer common divisor of the exponents from a product of powers. For example, if the powers are done with the help of repeated squaring, then

$$(y + 3)^6 x^4 \rightarrow \left((y + 3)^3\, x^2 \right)^2$$

uses only five multiplications rather than six.[17]

References

[1] Acton, F.S., *Numerical Methods that Work*, Chapter 11, Harper and Row, 1970 or The Mathematical Association of America, 1990.

[2] Avitzur, R., Milo (a Macintosh computer program), Paracomp Inc., 1988.

[3] Avitzur, R., The Graphing Calculator Story, `http://www.pacifict.com/Story/`

[4] Ayad, M. and Fleischmann, P., On the decomposition of rational functions, *Journal of Symbolic Computation* 43 (4), pp. 259-274, 2008.

[5] Bailey, D. and Borwein, J., Inverse Symbolic Calculator,
`http://isc.carma.newcastle.edu.au/advanced`

[6] Barton, D. and Fitch, J.P., A review of algebraic manipulative programs and their application, *The Computer Journal* 15(4), pp. 362-381, 1972.

[7] Bonadio, A., Theorist (a computer program), Prescience Corporation, 1989.

[8] Ceberio, M. and Kreinovich, V., Greedy algorithms for optimizing multivariate Horner schemes, *ACM SIGSAM Bulletin* 38(1), pp. 8-15, 2004.

---

[17]However, for most systems default simplification automatically distributes the outer exponent 2 over the two factors unless something special is done to prevent it.

[9] Cheze, C. and Galligo, A., Four lectures on polynomial absolute factorization, *Algorithms and Computation in Mathematics* 14, pp. 339-392, 2005.

[10] Cuyt, A.A.M. and Verdonk, B.M., A review of branched continued fraction theory for the construction of multivariate rational approximants, *Applied Numerical Mathematics* 4 (2-4), pp. 263-271, 2005.

[11] Fateman, R.J., Notes on Chebyshev series and computer algebra, 2011,
`http://www.cs.berkeley.edu/~fateman/papers/cheby.pdf`

[12] Faugère, J.C. and Perret, L.. High order derivatives and decomposition of multivariate polynomials. *Proceedings of ISSAC 2009*, pp. 207-214.

[13] Giesbrecht, M., Kaltofen, E. and Lee, Wen-shin., Algorithms for computing the sparsest shifts of polynomials via the Berlekamp/Massey algorithm, *Proceedings of ISSAC 2002*, pp. 101-108.

[14] Grigoriev, D.Y., Lakshman, Y.N., Algorithms for computing sparse shifts for multivariate polynomials, *Proceedings of ISSAC 1995*, pp. 96-103.

[15] Gutierrez, J., Rubio, R. and Sevella, D., On multivariate rational function decomposition, *Journal of Symbolic Computation* 33 (5), pp. 545-562, 2002.

[16] Kajler, N. and Soiffer, N., A survey of user interfaces for computer algebra systems, *Journal of Symbolic Computation* 25 (2), pp. 127-159, 1998.

[17] Kozen, D. and Landau, S., Polynomial decomposition algorithms, *Journal of Symbolic Computation* 7 (5), pp. 445-456, 1989.

[18] Krausz, F., A better user interface for Symbolics Lisp Machine Macsyma, Macsyma Newsletter 5(3), pp. 3-5, 1988.

[19] MathMonkeys, LLC, LiveMath, formerly known as Theorist, MathView, MathPlus, and Live Math Maker.,
`http://www.livemath.com/`

[20] Moses, J, Algebraic simplification: a guide for the perplexed. *Proceedings of the second ACM symposium on symbolic and algebraic manipulation*, pp. 282-304, 1971.

[21] Powers, V. and Wörmann, T., An algorithm for sums of squares of real polynomials,
`http://www.mathcs.emory.edu/~vicki/pub/sos.pdf`

[22] Reznick, B.E., *Sums of Even Powers of Real Linear Forms*, Memoirs of the American Mathematical Society, 1992, and `http://www.math.uiuc.edu/~reznick/memoir.html`

[23] Shneiderman, B. and Plaisant, C., *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, 4th edition, Pearson Addison Wesley, p. 367, 2004.

[24] Stoutemyer, D.R., Ten commandments for good default expression simplification, *Journal of Symbolic Computation*, 46(7), pp. 859-887, 2011.

[25] Stoutemyer, D.R., Unit normalization of multinomials over Gaussian integers, *ACM Communications in Computer Algebra* 43 (3/4), pp. 73-76, 2009.

[26] Stoutemyer, D.R, Multivariate partial fraction expansion. *ACM Communications in Computer Algebra* 42 (4), pp. 206-210, 2008.

[27] Stoutemyer, D.R., Useful computations need useful numbers, *ACM Communications in Computer Algebra* 41 (3), pp. 75-99, 2007.

[28] Sturmfels, B., *Algorithms in invariant theory*, 2nd edition, Springer 2008.

[29] Théry, L., Bertot, Y. and Kahn, G., Real theorem provers deserve real user-interfaces, *SDE 5 Proceedings of the fifth ACM SIGSOFT symposium on Software development environments* 17(5), pp. 120-129, 1992.
Preprint at `http://hal.inria.fr/docs/00/07/69/07/PDF/RR-1684.pdf`

[30] Trefethen, L.N. and others, Chebfun Version 4.2, The Chebfun Development Team, 2011, `http://www.maths.ox.ac.uk/chebfun/`

[31] Vandenberghe, L. and Boyd, S., "Semidefinite Programming", *SIAM Review* 38, pp. 49-95, March 1996.

[32] von zur Gathen, J., Functional Decomposition of Polynomials: The Wild Case., *Journal of Symbolic Computation* 10(5): pp. 437-452, 1990.

[33] von zur Gathen, J., Gutierrez, J., Rubio, R., Multivariate Polynomial Decomposition, *Applicable Algebra in Engineering, Communication and Computing* 14(1), pp. 11-31, 2003.

[34] von zer Gathen, J. and Weiss, J., Homogeneous bivariate decompositions, *Journal of Symbolic Computation* 19, pp. 409-434, 1992.

[35] Watt, S.M., Functional Decomposition of Symbolic Polynomials, *Proceedings of the International Conference on Computational Sciences and its Applications*, IEEE Computer Society, pp. 353-362, 2008.

[36] Wikipedia, Model-view-controller,
`http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller#References`

[37] UITP, User Interfaces for Theorem Provers, `http://www.informatik.uni-bremen.de/uitp/`

[38] Wolfram|alpha, `http://www.wolframalpha.com/`

[39] Zippel, R., Rational function decomposition, *Proceedings of ISSAC-91*, pp. 1-6, 1991.