DRAFT: Comments on Sets in Computer Algebra Systems, especially including Infinite Indexed Sets

Richard Fateman

December 23, 2012

1 Introduction

Computing with "sets" is well-explored in the programming-language and data-structure literature. Many languages have one or more set representations as well as operations for these sets. Unfortunately, the notion of set in mathematics is far more powerful than the notional support offered by ordinary programming languages. The programming languages' notations and operations work for explicit finite sets only, not (for example), the set of all odd primes.

Representing and manipulating *all* kinds of sets and set descriptions, including infinite set notations, or cases where the set is described algebraically, geometrically, logically, functionally or using some other "implicit" method is an open-ended challenge.

In this paper we address a subset of the problem prompted by an application of a computer algebra system (CAS). We show how we can "automatically" provide some simplification of set descriptions and other operations. One motivation comes from problems in which we obtained infinite sets such as $\{n\pi\}$ for integer n, and must compute with these further. These sets can be easily generated by a CAS asked to solve $\sin x = 0$ for x. To date, *simplification of operations* on such resulting sets are generally inadequate or unavailable from a CAS. Instead the result of some operations that produce or transform sets essentially require human intervention[1] before the computation can continue. Without explicit recognition the need to simplify sets, automated processing can quickly become impractical, and especially so if the designers of a CAS have systematically ignored the issue.

Another motivation comes from the representation and manipulation of algebraic numbers or functions. Thus one can refer to the set of algebraic numbers that are solutions to $x^3 - 1 = 0$, or perhaps more subtlely, the solutions in x for the equation $x^2 - y^2 = 0$. Some programs implement a simplification that $\sqrt{y^2} = |y|$, and even $-y$ if the system is told to assume $y < 0$.[2] There are other alternatives, including leaving $\sqrt{y^2}$ unchanged or returning a set $\{y, -y\}$. We advocate the latter.

It is possible that for distinct real numeric algebraic roots one can distinguish them by separating them into rational intervals, and so the roots can be ordered and indexed; this is not as plausible for multivariate polynomials.

Computer algebra systems often pretend that it is possible to somehow pick a particular algebraic branch to a square (or other) root, arguing that the program can choose the the distinguished principal value that is needed. This may work sometimes, and sometimes allows a computation to continue expeditiously. Yet not always: once one obtains a single contradiction from an incorrect or inconsistent choice, symbolic logic teaches us that any (including false) conclusion can be proven.

Even so, we find advocates for patching each "new error" as it is reported, rather than rooting out the source of the problem. The usual excuse is that it is too difficult to represent and compute with sets. It is especially difficult if you have committed substantial efforts to an inadequate design based on single-valued items, which then is not susceptible to a wide-spread extension to multi-valued items.

[1] or the assertion of some heuristic to choose one representative for the set

[2] It seems unlikely that the sign of y should influence the choice of branch cut of the square root, but it appears that $|y|$ is simplified to $-y$.

2 Set notation

There are many common notations for designating sets. Large sections of various font alphabets are consumed by special conventions for naming sets in algebra and analysis. We do not intend to close off future extension or further discussions, but to keep our discussion finite, we must make some choices to simplify our discussion. We hope to navigate a territory that contains some useful non-trivial results but does not require us to solve the unsolvable at every turn.

2.1 Basic named sets used for constructing others

The set of natural numbers $\{0, 1, 2, 3, \cdots\}$, will be denoted $\mathbb{N}$. We use it in the *index set* designation of other sets.

An alternative index set, of all integers, $\{\cdots, -2, -1, 0, 1, 2, \cdots\}$, will be denoted $\mathbb{Z}$. We will generally use identifiers n, m, k to mean variables restricted to $\mathbb{Z}$. Thus $n > -1$ means non-negative integers. (that is, $n > -1$ also means the same as $n \in \mathbb{N}$). Mathematica allows the following built-in names to be used for basic sets: `Algebraics, Booleans, Complexes, Integers, Primes, Rationals, Reals.`[3]

2.2 Finite Lists

While we focus on infinite sets, we must also allow for the "easy" cases.

1. The empty set $\{\}$, sometimes written $\emptyset$. One can usually identify a consistent role for how this interacts with other elements in a CAS. Usually any "arithmetic" operation involving $\emptyset$ produces $\emptyset$. Set operations (such as union) have well-known meanings with $\emptyset$, though set complement requires some kind of universe for context.

2. Singletons: sets with only a single value. Conventional single-valued results of today's typical CAS could be considered singleton sets. For example $1 + 1 = 2$ is simply identical to $\{1\} + \{1\} \to \{2\}$. We can introduce sets as $1 + \sqrt{9} = 1 + \sqrt{9} = \{1\} + \{-3, 3\} = \{1 - 3, 1 + 3\} = \{-2, 4\}$.

   Here is another example, starting from single-valued constants:

   $$1^{1/3} = \{1,\ \ -1/2 + 1/2\, i\sqrt{3},\ \ -1/2 - 1/2\, i\sqrt{3}\}$$

   .

   An astute (or argumentative) observer might say that if $\sqrt{9}$ has two values, so does $\sqrt{3}$ in that last expression. Even further, how do we know that it is the same value of $\sqrt{3}$ in the two places in which it occurs? So what is meant by that triple of expressions? In fact, in a purely algebraic setting in which the triple of expressions is used, we can choose either the popular positive interpretation approximated by 1.732... or its negation, as long as we use it consistently. In other settings we may need to distinguish (say) the positive root. We can commit unintentional errors If that set of roots is separated and the three values interpreted in different contexts (and with inconsistent signs for $\sqrt{3}$). A safer statement may be a "single-use" expression that uses only one root expression (which root is arbitrary): $1^{1/3} = \{1,\ \ -1/2 + 1/2\, i\mathrm{rootof}(\mathrm{r}^2 - 3, \mathrm{r})\}$ or an expansion of the previous expression which picks root 1 and root 2, knowing that they are distinct: $1^{1/3} = \{1,\ \ -1/2 + 1/2\, i\mathrm{rootof}(\mathrm{r}^2 - 3, \mathrm{r}, 1),\ \ -1/2 + 1/2\, \mathrm{irootof}(\mathrm{r}^2 - 3, \mathrm{r}, 2)\}$.

3. Other explicit lists: all elements are provided: E.g. {a,b,c}.

4. Multisets: sets including possible multiple occurrences of an element. E.g. $\{a, b, a, c\}$. The most obvious application of multisets in regard to computer algebra is the representation of zeros of a polynomial, where the representation for a set of pairs (*value, multiplicity*), or two (indexed) lists, of roots and multiplicities, or an unordered set of roots and a mapping from roots to multiplicities.

[3]The ability of the program to exploit properties associated with these names may depend on the version of the system being used. It does make some information available to a built-in or user-written program should that program inquire. It would be a mistake for the user to assume that all programs make use of all the properties.

Either would be easier for a program to handle. We will generally try to ignore extra duplicated explicit elements and will try not to distinguish $\{a, b, a, c\}$ from $\{a, b, c\}$. Indeed, a set such as $\{x^2 \mid x \in \mathbb{R}\}$ will have each positive element twice, and we might not notice it.

2.3 Plus-minus Expressions

This notation (as input or output) is not commonly accepted in computer algebra systems, but is widely used in applied mathematics texts.

$-b/2 \pm \sqrt{d}$ is a set of two expressions.

$a \pm b \pm c$ is a set of 4 expressions, though $a \pm b \mp c$ is a set of 2 expressions.

2.4 Indexed Sets

The common notation is $\{f(n) \mid n \in \mathbb{N}\}$

A better notation would make clear the relationship between the two occurrences of n. In particular, only when n occurs "free" on the left, is it bound to the n on the right. (A non-free, i.e. bound n occurs in the integrand "n^2dn" here: $\{n + \int_0^n n^2dn \ \mid \ n \in \mathbb{N}\}$) and only the "external" n ranges over $\mathbb{N}$.

A notation which uses explicit ideas available from the lambda calculus, which is simple and familiar to anyone who has learned Lisp or logic: We suggest something like $\{\lambda(n)f(n) \ \mid \mathbb{N}\}$ if appropriate, as being correct and more formal if there is only one parameter, here n. Mathematica uses the notation `Function[{n},f[n]]` or the shorthand `f[#]&` for this.

We will not necessarily use this notation for the simple examples of this paper.

2.5 Compound Sets

Given any of the representations for sets A, B, C, it may be infeasible to simplify expressions like $A \cap B$ or $A \cup B$ or the complement, $\bar{A}$. Our representation is too general to always permit explicit representation of unions or intersections. The typical explicit sets provided in "set" packages are different in this respect.

Set simplification can use various identity laws, distribution, deMorgan's laws. While useful around the margins and dealing with left-over pieces of sets, we suspect that for our applications these are not going to be very effective.

2.6 Operations

These may not be algorithmically computable *in general*. In most cases this does not prevent us from writing a program to perform these operations *sometimes*, and in other cases leave the results unresolved. Do we need to address the Closed World Assumption CWA (meaning: Everything not known to be true is false) vs. the Open World Assumption OWA (meaning: Something not known to be true might be true or false)

1. Test for empty set. Some sets might be empty but we might not be able to figure it out. Thus $\{[x, y, z, n] \mid (x^n + y^n = z^n) andx, y, z, n \in \mathbb{N} + andn > 2\}$

2. Choose an element. (and remove it from a set.) Generate the next element, etc. If the choice is deterministic, then this imposes an order on the set − but this then could be used to distinguish otherwise "equal" sets. While this may be done in an indexed set (or tuple) by choosing the element with (say) the least index, this could be, in the general case, not computable.

3. Membership. Is $x \in S$? e.g. is $0 \in \{\lambda(n) \ 2 \times n \ \mid \mathbb{N}\}$? This is problematical with regard to negation/complement and the CWA. Is $x \in \bar{S}$?

4. Intersection: Here the problem is to find an expression simpler than $A \cap B$ for an intersection. This is not possible in general, but it is helpful to be able to compute, for example that $\{\lambda(n)(2 \times n + 1) \ \mid \mathbb{N}\} \cap \{\lambda(n)2 \times n \ \mid \mathbb{N}\}$ can be expressed more simply, namely as the empty set $\emptyset$.

5. Union: Find a simple expression than $A \cup B$ for a union. For $\{\lambda(n)2 \times (n+1) \ \mid \mathbb{N}\} \cup \{\lambda(n)2 \times n \ \mid \mathbb{N}\}$ we can use the set $\mathbb{N}$.

6. Cardinality: the number of elements, which may be infinite. It would be helpful to actually oompute unique cardinality (counting repeated elements once). For some descriptions the cardinality may not be computable, or our program may not be clever enough to compute it. Sets defined by characteristic functions or sequences computed online (e.g. "pipes") typically fall in this category. It is not always possible to tell whether we have a set or a multiset (with repeated elements). When such a distinction is required in a given application, it would be an additional (and again perhaps not computable) burden to assure that a set does not have repeated elements.

7. Cross product: $A \times B$, the set of ordered pairs from sets A and B.

8. Arithmetic: Arguably, $A + B$ could be the pairwise summing of elements from A and B, that is $\{a + b | a \in A, \quad b \in B\}$. If (say) A is not a set but a scalar quantity s, the sum is presumably $\{a + s | a \in A\}$. An apparent conundrum follows for $A \times B$, if A is the scalar 0 and B is the empty set. A commonly applied rule of thumb in algebra systems is that $0 \times$ *anything* is 0, and yet by this definition, we get not 0, but the empty set. However, if the scalar 0 is considered to be the singleton set $\{0\}$ then a point-wise product or sum with the empty set is still the empty set. The use or generation of the empty set in arithmetic has to defined consistent with other data and operations since the empty set may not be immediately recognized, and rules should not be violated even if $\emptyset$ is disguised.

9. Many other operations (single argument) may be added to this list simply by moving the operation inside the set descriptor.

3 Techniques

1. Reduction to canonical index names. e.g. operations on sets should *not* be able to distinguish $\{\lambda(n)(2 \times n + 1) \quad | \mathbb{N}\}$ from $\{\lambda(m)(2 \times m + 1) \quad | \mathbb{N}\}$, each of which is the set of non-negative odd integers.

2. Black box generators. In the sense of pipes or streams, consider a set which allows you to examine one of its elements and then, if necessary can run a program generating the next element in the set. This may ordinarily require a pipe that generates elements in order. An implementation in Lisp might look like

```
(define (integers-from n)(make-pipe n (integers-from (1+ n 1))))
(define non-neg-integers (integers-from 0))
```

   From this can be created, using other programs working as filters or mapping functions, almost any computational series. For example, pipes that contain only primes are illustrated in a well-known computing text [1].

3. Compare to well-known sets: $\mathbb{Z}$, $\mathbb{N}$. $\mathbb{Q}$, (rationals) $\emptyset = \{\}$ to see if a relationship (subset, equality?) holds.

4. Simplifications include all operations (distributivity, deMorgan's laws) defined for sets generally in the same CAS.

4 Requirements, benchmarks

We would like to be able to consider solving the problems in this (growing) list.

1. $\{2n \mid n \in \mathbb{N}\} = \{2 * m \mid m \in \mathbb{N}\} = \{2 * k + 4 \mid k > -3\}$

   To notice this equality, the usual approach in a CAS is to find a *canonical form* to which each of these can be simplified. (See lambda-binding above).

2. $\{2n + 1 \mid n \in \mathbb{N}\} \cup \{2 * n \mid n \in \mathbb{N}\} = \mathbb{N}$

   Even and odd numbers are all there can be in $\mathbb{N}$.

3. $\{2n+1 \mid n \in \mathbb{N}\} \cap \{2n \mid n \in \mathbb{N}\} = \{\}$

   Even and odd numbers don't overlap.

4. $\{n-1 \mid n \geq 0\} \cap \{m \mid m \leq 0\} = \{-1,\ 0\}$

   Finite sets can result from intersections of infinite sets.

5. $\{2n \mid n \in \mathbb{N}\} \cup \{3m \mid m \in \mathbb{N}\} = \{n \mid ((n \bmod 3 = 0) \vee (n \bmod 2 = 0))\ \wedge\ n \in \mathbb{N}\}$. This right-hand expression is probably not more attractive than the left-side expression. Arbitrary manipulation of the index set is clearly going to lead to problems.

   Let $P = \{n \mid \text{prime}(n)\}$ and $R = \{(p+q) | (p \in P) \wedge (q \in P)\}$ If we can prove the set of even numbers greater than 4 is a subset of R we have proved Goldbach's conjecture[4].

6. $\{2n \mid n \in \mathbb{N}\} \cap \{3m \mid m >\in \mathbb{N}\} = \{n \mid n \bmod 6 = 0) \mid n \in \mathbb{N}\}$

   How can we deal with this?

   $\{n^2 \mid n \in \mathbb{N}\} = \{m^2 - 2m + 1 \mid m \in \mathbb{N}\}$ Each includes all integers that are squared.

5 Applications

5.1 Symmetry, periodicity in integration

Our initial motivation for this exploration was an application that requires detecting periodicity of a function. In this application [2], in the process of computing a definite integral, we first check for symmetry or antisymmetry of an integrand around some point (to be found). Our method depends on solving a related shifted equation whose solution can be an infinite set of the nature required above. Continuing the computation in a CAS unattended, that is, with the human taken *out of the loop*, requires further operations to be done sight unseen. In particular, the intersection of two sets, each an infinite set of points, might be finite (even empty) or infinite. An algorithmic approach is required, even if it is clear that all problems will not be solvable. Similar questions arise about set unions.

5.2 Other related problems

The notational and simplification issues here have parallels in dealing with other mathematical objects. For example, consider $\sum_{n=1}^{\infty} f(n) + \sum_{m=1}^{\infty} g(m)$. Simplification of this expression requires renaming of one (or both) of the index sets; the same lambda-binding idea holds. It requires aligning the index sets in $\sum_{n=1}^{\infty} f(n)x^n + \sum_{m=0}^{\infty} g(m)x^m$.

6 Plan of attack

As stated initially, we must make some judicious choices and fix some of the many variabilities.

We are not interested in discussing simplification of (other kinds of) sets that may already be included in a CAS. Thus we assume $A \cup A = A \cap A \rightarrow A$ and similar rules are known and applied when possible. Since our motivation was to be able to compute further with cases as returned from CAS solve programs, let us look at them again.

Consider the set we got from solving $\sin x = 0$ for x. The expression arcsin(0) is actually a set, $n\pi$ meaning $\{n\pi \mid n \in \mathbb{N}\}$[5]

Here is the kind of computations might we wish to do with this set:

[4]Namely all positive even integers $n \geq 4$ can be expressed as the sum of two primes. A reward of \$1,000,000 for such a proof was offered by a publisher (Faber and Faber), although it is now expired.

[5]A possible quibble: if $n\pi$ is an angle, then can we say that—as angles—$0 = 2\pi$ in which case the set of unique solutions is the set of only two: $\{0,\ \pi\}$. We reject this because there are other ways of looking at arcsin where the angle continues to evolve indefinitely to higher numbers.

What solutions does the equation $\sin(x) = 0$ have in common with $\sin(x/2) = 0$? The two solution sets are $\{n\pi \mid n \in \mathbb{N}\}$ and $\{2m\pi \mid m \in \mathbb{N}\}$. Their intersection is found by (a) assuring that the index range is identical, and (b) solving $n\pi = 2m\pi$ for (say) n, resulting in $n = 2m$. The intersection result is thus $\{2m\pi \mid m \in \mathbb{N}\}$. It is not appropriate to solve for m to get $m = n/2$ because m must be in $\mathbb{N}$, and half-integer values are not permitted. We must compute this explicitly, as shown below.

6.1 What is the union of the two sets?

This can be rather complicated since we are searching for a simplified form. Here is a heuristic that sometimes helps, given that A and B are simplified set notations. If $A \subseteq B$ then $A \cup B = B$. Constructively can we show $x \in A \;\Rightarrow\; x \in B$? In order to show that for each value $2m\pi$ there is a value $n\pi$, set them equal and solve for n. We must conclude $(n = 2m \;\wedge\; m \in \mathbb{N}) \Rightarrow n \in \mathbb{N}$ by noting that multiplication by 2 maps $\mathbb{N}$ into $\mathbb{N}$.

6.2 When are two sets equal?

Consider a problem that will turn out to be too hard. We encountered the question `solve(sin(x+c)=sin(x),c);` Macsyma says:

$$\left\{c = 2\,\arctan\left(\frac{\cos x}{\sin x}\right) + 2\,\pi\,\mathrm{n}_1, c = 2\,\pi\,\mathrm{n}_2\right\}.$$

This is kind of hard to fathom, so substituting $y - c/2$ for x and then expanding the sines of sums, we get:

$$2\,\sin\left(\frac{c}{2}\right)\,\cos y \;=\; 0$$

The first factor leads to solutions of the form $c = 2\arcsin 0$. The second factor leads to $x + c/2 = \arccos 0$ or $c = 2 \cdot (\arccos 0 - x)$. Combining these we can claim the solution is

$$\{n\pi\} \cup \{(2m+1)\pi - 2x\}.$$

Here is another approach to the same question. Let us convert to exponential form first, then use the `radcan` simplification command. In this case Macsyma gives:

$$\{-2\,x + 2\,\pi\,n + \pi\} \cup \{0\}.$$

Actually, that $c = 0$ solution is defective; it is claiming that $e^{ic} = 1$ has the (sole) solution $c = 0$ instead of $c = 2k\pi$. A corrected solution would then be

$$\{-2\,x + 2\,\pi\,n + \pi\} \cup \{\;2k\pi\}.$$

The current Mathematica version 8.0 says $c = 0$, but an old version of Mathematica (4.0?) says

$$\{-x + \arcsin(\sin(x))\}.$$

This expression is a piecewise-continuous function of x that is periodically constant. When $-\pi/2 \le x \le \pi/2$ then $c = 0$
When $\pi/2 \le x \le 3\pi/2$ then $c = -2x + \pi$
When $3\pi/2 \le x \le 5\pi/2$ then $c = -2\pi$. etc.

Maple finds only the solution is $c = 0$. True, but not very complete.

Comparing the above sets for all $x \in D$ (D might be real or complex numbers) is complicated, and not something we have programmed. In fact, the Mathematica set does not include solutions like $c = -2x + 5\pi$ which are provided in the Macsyma solution, so the sets are *not* equal.

The sets differ on a grosser level. The Macsyma solutions comprise an infinite set of lines with slope -2 spaced 2π apart. The Mathematica solution is single-valued (one member for each real x).

7 RAW NOTES

7.1 How to represent sets

We propose that a *basic set* is a pair: The first element is $\lambda(n)f(n)$. e.g. $\lambda(n)n^2$ for a set of squares. The second element is a domain for n, default is $n \in \mathbb{N}$. Other possibilities include $n > a$, $n < a$. The expression a must be integer-valued, although not necessarily an explicit integer. As will be evident shortly, we do not guarantee uniqueness of representation, and there may be sets that cannot be represented (conveniently) this way.

Other set representations are possible: explicit enumeration, `rootof` expressions for algebraic varieties, geometric subsets of n-dimensional sets. Not all operations make sense in combining sets, and there is presumably a large range of possible combinations that could be specified for different types of sets that have no obviously useful meaning. This ordinarily does not prevent a CAS from representing such items "unsimplified" until perhaps some semantics can be imposed upon them.

7.2 More Operations

- Change of index

```
 >  (lambda(n)f(n); n>a)
  if a is -1,  change to (lambda(n)f(n); n in N)
  if a is not -1, change  to  ((lambda(n)(lambda(q) f(q))(n+a)) n in N)
   ; q is not free in f.

 change of index, <  (lambda(n)f(n); n<a)
  if a is 1,  change to (lambda(n)f(-n); n in N)
  if a is not 1,  change to ((lambda(n)(lambda(q) f(q))(-n+a)) n in N)
 ; q is not free in f.

(lambda(n)(lambda(q) f(q))(n+q)) can ordinarily be rewritten as (lambda(n)(f(n+q))).
 [assuming purely functional expression f.]
```

- Membership test e is in $(lambda(n)f(n), n \in \mathbb{N})$ if there is a solution `n=k` to $f(n) = e$ and $n \in \mathbb{N}$. Requires commands solve, and natnump.

```
Natnump(x):= if x is a singleton set then
             if x is an explicit natural number
               or x is declared to be a natural number
               or (x= y+z and natnump(y) and natnump(z))
               or (x= y*z and natnump(y) and natnump(z))
               or (x= y-z and natnump(y) and natnump(z))
               or (x= y^z and natnump(y) and natnump(z))
       else (* x is not a singleton *)
           is every element of x a Natnump.
```

may fail to identify some natnums.

fails on $(n+1)*n/2$ which is a natural number. A human might notice that either n or$n+1$ is even and thus the 2 always divides one or the other. How can this be automated?

heuristic: for each indeterminate $\{v1, v2, ..\}$ substitute a random integer. If the result is an integer, suggest that the expression might be natnump. Test repeatedly to be more confident.. haha.

- Every test (Used above in natnum). `Every(f,S)` returns true if f(e) is true for every element e in the set S. Presumably f has no side effects on S. If S cannot be enumerated, this results in an error (or perhaps false).
- There_Exists test

  `There_Exists(f,S)` returns true if f(e) is true for at least one element e in the set S. Returns [true, e] perhaps? Presumably f has no side effects on S. `There_Exists (lambda([r],is (equal(r,q))), S) [1]` is the same as membership of q in S if S can be enumerated. If S cannot be enumerated, this results in an error (or perhaps false).
- subset test is $(\lambda(n)f(n), n \in \mathbb{N})$ a subset of $(lambda(m)g(m), m \in \mathbb{N})$?

  Solve the equation $f(n) = g(m)$ for $n = k$ test `natnump(k)` assuming $m \in \mathbb{N}$?
- equality $A \subset$ B and $B \subset A$.
- Intersection

  ```
     Let A=  ( lambda(n)f(n), n in N)
         B= ( lambda(m)g(m), m in N).
  ```

  $A \cap B$ is the set of all e such that there is an n in N and an m in N such that e=f(n)=g(m).

  example, A = set of n^2, B = set of $2m$. $\{0, 1, 4, 9, 16, 25, \cdots\} \cap \{0, 4, 8, 16, \cdots\} = \{0, 4, 16, \cdots\}$

  the common elements can be found by setting $n^2 = 2 * m$ and solving the polynomial diophantine equation for n,m in N.

  Since we cannot expect to solve arbitrary polynomial equations, how about restricting the functions f, g, to linear functions of n?

  Example: what can we do with $\{2n + 1\} \cap \{2n\}$ or $\{2n + 5\} \cap \{2m\}$?

  for x in intersection, x = 2n+1 for some $n \in \mathbb{N}$ and also $x = 2m$ for some $m \in \mathbb{N}$.

  Thus $2n + 1 = 2m$, and `solve(2*n+1=2*m,m)` gives $m = (2n + 1)/2$ which cannot be shown to be a natnum by our previous method; substitution of random integers also shows, at least some of the time, it is *not* natnum.

  `solve(2*n+5=2*m,m)` gives $m = (2n + 5)/2$, same argument.

  How about $12kn$ and $20km$ intersection, where `solve` gives $m = 12/20n$, certainly not a natnum. Caution is required here: assuming there is no solution because we can't find it with this weak test leads to an incorrect conclusion.

  ```
  Find Least Common Multiple (LCM) of 60.
  when n=0, m=0,    0
       n=5, m=3,   60
       n=10, m=6, 120
      etc

  In general, if we want to find the intersection, for variables
  n and m, with constants a,b,c,d
  for a*n+b =  c*m+d  do this..

  a*n=c*m+ (d-b)  ==> all converted to mod c.

   e.g. if d-b is negative, add multiples of c

  so we need to find out if there are
  ```

```
solutions for n:   a*n mod c = d-b
.....
Example
         12kn+3       20km+1

when k=1, n=0,1,...  gives 3, 15, 27, 40, 51, 63, 75, 87, 99, 111, 123
          m=0,1,...  gives 1, 21, 41, 61, 81, 101, 121, 141, 161,181,

The question then is to solve for n: (12n+3) mod 20 = 1,
or rewrite as 12n mod 20 = 18
```

Mathematica allows this:

```
Solve[{12n ==18,Modulus==20},n,Mode-->Modular]

Solve[12n ==8,n,Modulus->20]
```

gives the following set notation as a rule where `C[1]` is the first "constant generated in representing the results of various symbolic computations"

```
{{n->4+5 C[1]}}
```

This result is arguably false without the condition (which Mathematica is capable of uttering in other circumstances), that `C[1]` is an `Integer`. One could argue that since the computation is done "modulo 20" that any numbers including `n` and `C[1]` are in the finite field of integers mod 20. Some thought should lead you to conclude that this does not reflect the likely meaning of the question. That is, $n = 50004$ is a solution.

ROOTSUM: the result of integration of a rational function may be expressed as a summation whose index ranges over a `rootsof` operation in Maxima, by setting `integrate_use_rootsum:true.` Then we see the result of

```
integrate(1/(x^4-a*x+1),x);
```

$$\sum_{r \in \mathrm{ootsof}(x^4 - a\,x + 1)} \frac{\log(x - r)}{4\,r^3 - a}$$

We can compute with this, sometimes. For example, computing the derivative of this expression and simplifying, should return the integrand. The `rootsof` expression here implicitly uses the main variable x, and denotes a set. We could also denote the individual roots by indexing them, and we could in some circumstances indicate the particular choice of each separable root. We have written more about this topic separately (see rootsum.tex paper).

References

[1] H. Abelson and G. Sussman, *Structure and Interpretation of Computer Programs,* MIT Press/ McGraw Hill.

[2] R. Fateman "Methods for integration of (anti)-symmetric functions" Draft.

[3] R. Fateman and W. Kahan, "Improving Exact Integrals from Symbolic Computation Systems," Tech. Rept. Ctr. for Pure and Appl. Math. PAM 386, Univ. Calif. Berkeley. 1986. (poster session at ISSAC 2000.)