Maple for Math Majors

Roger Kraft
Department of Mathematics, Computer Science, and Statistics
Purdue University Calumet
roger@calumet.purdue.edu

4. Mathematical Identities and Maple's Assume Facility

4.1. Introduction

In this worksheet we consider mathematical identities as another example of a use of an equal sign in mathematical notation. We show how to get Maple to demonstrate a number of mathematical identities. We look at the case of identities that are true for only some values of the variables in the identity. This leads us to introduce Maple's assume facility, which allows us to place assumptions on the values that an unassigned variable is supposed to represent.

```
>
```

4.2. Mathematical identities

In a previous worksheet we emphasized that equal signs are used in mathematical formulas in at least two different ways, to represent an assignment and as part of an equation. So for example, we interpret the equal sign in the formula $x = 0$ as an assignment and assume that the formula means that x is a name for zero. We translate this into Maple as the assignment statement `x:=0`. On the other hand we interpret the formula $x^2 - 2\,x - 1 = 0$ as an equation that should be solved for x. We translate this directly into Maple as the equation `x^2-2*x-1=0`, which can then be solved using the `solve` command.

```
> x^2-2*x-1=0;
> solve( %, x );
```

Now consider the mathematical formula $(1-x)^2 = 1 - 2\,x + x^2$. Here the equal sign is definitely not an assignment. And this is not really an equation either since this formula is not asking for which x is the equation true. The equation is true for all x and in fact the purpose of the formula is to tell us that. This formula is an example of an identity and the purpose of the equal sign in an identity is to tell us that the left and right hand sides are (under certain circumstances) interchangeable. So now we have a third use for an equal sign in a mathematical formula. How should this use of an equal sign be translated into Maple? We would want Maple to tell us that $1 - 2\,x + x^2$ is equivalent to $(1-x)^2$. The `expand` command in Maple will tell us exactly that.

```
> expand( (1-x)^2 );
```

So the mathematical identity $(1-x)^2 = 1 - 2\,x + x^2$ is represented in Maple by an application of one of Maple's commands.

```
>
```

Let us look at some other examples of identities and how Maple would demonstrate them. The `factor` command allows Maple to demonstrate the following algebraic identity.

$$3\,x^2-7\,x-20=(x-4)\,(3\,x+5)$$

```
> factor( 3*x^2-7*x-20 );
```

Here is an identity involving the exponential function.

$$\mathbf{e}^{x}\,\mathbf{e}^{y}=\mathbf{e}^{(x+y)}$$

For this identity we can use the `simplify` command.

```
> simplify( exp(x)*exp(y) );
```

Here is a trig identity.

$$\sin(x)\cos(y)=\frac{\sin(x+y)}{2}+\frac{\sin(x-y)}{2}$$

For this identity we need the `combine` command.

```
> combine( sin(x)*cos(y) );
```

Here is an identity about a rational function.

$$\frac{a-b}{(x-a)\,(x-b)}=\frac{1}{x-a}-\frac{1}{x-b}$$

For this identity we need the `convert` command with the special option `parfrac`.

```
> convert( (a-b)/((x-a)*(x-b)), parfrac, x );
```

The option `parfrac` is an abbreviation of "**par**tial **frac**tion". So this `convert` command converts a rational function into its partial fraction expansion.

```
>
```

Here is an identity involving the integers.

$$\sum_{k=1}^{n} k=\frac{n\,(n+1)}{2}$$

We translate this into an application of Maple's `sum` command.

```
> sum( k, k=1..n );
```

Well, we are not quite there yet. Try the `factor` command.

```
> factor( % );
```

We can combine these two steps into one Maple command.

```
> factor( sum(k, k=1..n) );
```

Suppose that we wanted Maple to actually display the whole identity in its output, not just the right hand side of the identity. We could do this using what is called the **inert form** of a Maple command. Many Maple commands have an inert form, which is the command with its initial letter capitalized. The inert form of `sum` is `Sum`. The inert form of a Maple command (if the command has an inert form) does not carry out the action of command. Instead, the inert form is a way to tell Maple to typeset the command in Standard Math Notation as the command's output. So for example, the inert form of `sum` does not do any summation, but it does display a nicely typeset summation formula.

```
> Sum( k, k=1..n );
```

(The fact that the summation symbol is black, instead of blue, is meant to show that it is the result of an inert command.) So here is how we get Maple to display the entire identity. Use the inert form of `sum` on the left hand side of an equal sign and the regular `sum` on the right hand side.

```
> Sum(k, k=1..n) = factor( sum(k, k=1..n) );
```

Here is an integral identity, displayed by using both the regular and inert forms of the `int` command.

```
> Int( sqrt(x^2+a^2), x ) = int( sqrt(x^2+a^2), x );
```

We can get Maple to display all of our previous identities. These identities do not even need the use of an inert command.

```
> (1-x)^2 = expand( (1-x)^2 );
> 3*x^2-7*x-20 = factor( 3*x^2-7*x-20 );
> exp(x)*exp(y) = simplify( exp(x)*exp(y) );
> sin(x)*cos(y) = combine( sin(x)*cos(y) );
> (a-b)/((x-a)*(x-b)) = convert( (a-b)/((x-a)*(x-b)), parfrac, x
  );
> 
```

Here is how we would do each of these identities in its other direction.

```
> 1-2*x+x^2 = factor( 1-2*x+x^2 );
> (3*x+5)*(x-4) = expand( (3*x+5)*(x-4) );
> exp(x+y) = expand( exp(x+y) );
> sin(x+y)/2 + sin(x-y)/2 = expand( sin(x+y)/2 + sin(x-y)/2 );
> 1/(x-a) - 1/(x-b) = simplify( 1/(x-a) - 1/(x-b) );
> 
```

Notice that it took quite a few Maple commands to demonstrate these identities and it is not always obvious which Maple command might be useful for demonstrating a particular identity. In the next worksheet we will go over some of the details of using these commands and we will try to give some guidelines on which command should be used when. In particular, the commands that are covered in detail in the next worksheet are `simplify`, `factor`, `expand`, `combine`, and `convert`.

All of the identities demonstrated above have the property that they are true for all the possible values of the variables in the identity. Not all identities have this property. For some identities, the identity is true for only some values of the variables involved. For example, the trigonometric identity $\arcsin(\sin(x)) = x$ is true only if x is between $-\pi/2$ and $\pi/2$. Because of this, Maple will not readily demonstrate this identity for us.

```
> arcsin( sin(x) );
> simplify( % );
```
But we can tell the `simplify` command to make an assumption about the possible values that the variable `x` can represent.
```
> simplify( arcsin( sin(x) ), assume=RealRange(-Pi/2, Pi/2) );
```
In the next section we look at more examples of identities that are true for only some values of the involved variables. And in the section after that we show how to make use of Maple's assume facility to place assumptions on what values a variable can represent.
```
>

>
```

4.3. A closer look at mathematical identities

An important idea to remember about mathematical identities is that they need not be true for all values of the variables involved. An identity may be true only when the variables in the identity have their values restricted to a certain set of numbers. To put it another way, an identity may only be true if we make an assumption about what values the variables in the identity may represent. For example, here are three identities, each with a set of numbers for which the identity is true.

$$\sqrt{x^2} = x \text{ is true for all positive real numbers,}$$

$$\sqrt{x^2} = |x| \text{ is true for all real numbers,}$$

$$\sqrt{x^2} = \operatorname{csgn}(x)\, x \text{ is true for all complex numbers.}$$

Let us see how we get Maple to demonstrate each of these identities and also some similar identities.
```
>
```

Maple assumes, by default, that variables represent complex numbers. So Maple will, by default, only express identities that are true for all complex numbers. Of the three identities from the previous paragraph, the last one is true for all complex numbers, and so that is how Maple will, by default, simplify the expression `sqrt(x^2)`.
```
> simplify( sqrt(x^2) );
```
On the other hand, if we tell the `simplify` command to make an assumption about `x`, then it will simplify the expression `sqrt(x^2)` differently. Here is how we use the `assume` option to the `simplify` command to express the first of the three identities.
```
> simplify( sqrt(x^2), assume=positive );
```
The second identity from the previous paragraph turns out to be not as straight forward as the other two. If we tell `simplify` to assume that `x` represents only real numbers, then we get an expression that is equivalent to, but not exactly the same as, what we want.
```
> simplify( sqrt(x^2), assume=real );
```
We can convert this result into a form using absolute values, but again we get a result that is equivalent to, but not exactly the same as, what we want.
```
> convert( %, abs );
```
If we simplify the last result and once again tell `simplify` that `x` represents a real number, then we get exactly what we want.

```
> simplify( %, assume=real );
```
The last three commands can be abbreviated a bit. It turns out that we can skip the middle `convert` command.
```
> simplify( sqrt(x^2), assume=real );
> simplify( %, assume=real );
```
In fact, the last two commands can be abbreviated even more. We can drop the assumption on `x` in the first `simplify` command.
```
> simplify( sqrt(x^2) );
> simplify( %, assume=real );
```
We can combine the last two commands together into one command that demonstrates the identity.
```
> sqrt(x^2) = simplify(simplify( sqrt(x^2) ), assume=real);
```
This example shows that even when telling `simplify` to use an appropriate assumption, it may not always be obvious how to go about demonstrating a particular identity.
```
>
```

Here is an identity that is a slight variation on the above three examples.

$\sqrt{(x-10)^2} = x - 10$ is true for all real numbers greater than or equal to 10.

To get the `simplify` command to demonstrate this identity we need to be able to tell `simplify` to assume that `x` represents a real number that is greater than 10. Here is how we do this.
```
> sqrt((x-10)^2);
> simplify( %, assume=RealRange(10,infinity) );
>
```

Here is a slight variation on the last example.

$\sqrt{(x+10)^2} = -(x+10)$ is true for all real numbers less than or equal to -10.

In order to demonstrate this identity we need to tell `simplify` to assume that `x` represents a real number that is less than or equal to -10.
```
> sqrt((x+10)^2);
> simplify( %, assume=RealRange(-infinity,-10) );
>
```

Now let us consider the identity $\sqrt{(x^2-1)^2} = x^2 - 1$. This is true if x is less than -1 or if x is greater than 1. In other words, we need to assume that `x` is contained in one of two disjoint intervals.
```
> sqrt((x^2-1)^2);
> simplify(%, assume=OrProp(RealRange(-infinity,-1),
>                           RealRange(1,infinity)) );
```
Here is an interesting example. Modify the last identity by factoring $x^2 - 1$ to get the following equivalent identity $\sqrt{((x+1)(x-1))^2} = (x+1)(x-1)$. This is still true if x is less than -1 or if x is greater than 1. But to demonstrate this identity we need to use `simplify` three times in a row. This shows, once again, that even when we make an appropriate assumption about a variable, it may

not be obvious how to get the desired result.

```
> sqrt(((x+1)*(x-1))^2);
> simplify( %, assume=OrProp(RealRange(-infinity,-1),
>                            RealRange(1,infinity)) );
> simplify( %, assume=OrProp(RealRange(-infinity,-1),
>                            RealRange(1,infinity)) );
> simplify( %, assume=OrProp(RealRange(-infinity,-1),
>                            RealRange(1,infinity)) );
>
```

Now consider the identity $\sqrt{(x^2-1)^2} = 1 - x^2$. This is true if x is greater than -1 and less than 1.

```
> sqrt((x^2-1)^2);
> simplify( %, assume=RealRange(-1,1) );
>
```

Maple tries hard to not to apply an identity to an expression unless the identity is provably correct under the assumptions made on the variables in the expression. But if Maple should use an identity incorrectly, this could lead to incorrect results from a calculation. Here is an example of how Maple can get tripped up and return an incorrect identity. Recall that the identity $\arctan(\tan(x)) = x$ is true only if x is between $-\pi/2$ and $\pi/2$.

```
> arctan( tan(x) );
> simplify( % );
> simplify( %%, assume=RealRange(-Pi/2,Pi/2) );
```

Now consider the following combination of the `convert` and `simplify` commands.

```
> arctan( tan(x) );
> convert( %, exp );
> simplify( % );
```

These two commands managed to simplify `arctan(tan(x))` to `x` without any assumption on the value of `x`. So if `x` should in fact be representing a number greater than $\pi/2$ for example, then the above calculation would be wrong. And if the expression `arctan(tan(x))` appeared in a long calculation along with this combination of `convert` and `simplify`, then the long calculation would be incorrect. This shows that Maple can sometimes make incorrect simplifications, and it also shows that the results of complicated calculations always need to be double checked for correctness.

```
>
```

Exercise: Try to determine if the error in the last two commands was caused by the `convert` command or by the `simplify` command.

```
>
```

So far, all of the identities we have done in this section have involved intervals of real numbers. But other kinds of sets of numbers can be used in identities. Here are a few that use integers. For example, $\sin(n\,\pi) = 0$ for any integer *n*.

```
> sin(n*Pi);
> simplify(%, assume=integer);
```

And $\cos(n\,\pi) = (-1)^n$ for any integer *n*.

```
> cos(n*Pi);
> simplify(%, assume=integer);
```

And of course, $(-1)^n$ is either 1 or −1, depending on whether *n* is even or odd.

```
> simplify(%, assume=even);
> simplify(%%, assume=odd);
```

Here are two more identities that depend on even and odd integers.

```
> cos(n*Pi/2);
> simplify(%, assume=even);
> sin(n*Pi/2);
> simplify(%, assume=odd);
>
```

The last several examples have shown how to demonstrate some identities that are true for various kinds of sets of real numbers. There are many other kinds of identities and many different kinds of sets of numbers that an identity can be true for. But the `assume` option of the `simplify` command is not versatile enough to handle most of them. In the next section we will look at Maple's `assume` command, which can be used in far more situations than the `assume` option for `simplify`.

```
>

>
```

4.4. Using Maple's assume facility

In the last section we used the `assume` option for the `simplify` command to demonstrate the following identity.

$$\sqrt{x^2} = x \text{ is true for all positive real numbers.}$$

We can also use Maple's `assume` command to demonstrate this identity. The `assume` command allows us to "permanently" attach an assumption to a variable. We can use `assume` to tell Maple that the unassigned variable `x` is meant to represent only real, positive numbers.

```
> assume( x, positive );
```

The `assume` command did not return anything. But we can use the `about` command to find out that Maple is now making an assumption about the values represented by `x`.

```
> about( x );
```

Many Maple commands can make use of this information about the values that `x` is supposed to represent. In fact, Maple itself will make use of this information and automatically simplify the expression `sqrt(x^2)` and thereby demonstrate the identity that we want.

```
> sqrt(x^2);
```

Notice that the output has a tilde (~) after the variable `x`. This is to remind us that there is an assumption being made about the values represented by `x`. Notice that the `about` command started its out by mentioning that `x` has been renamed `x~`. The name `x~` is used by Maple for `x` in the output of commands, but we should continue to use the name `x` in the inputs to commands.

```
> x;
```

Using `assume` to make an assumption about a variable's values leaves the variable unassigned in the sense that it does not have a specific value assigned to it. But the `assume` command does "assign" a property to the variable, the property being the set of numbers that the variable is supposed to take its values from. We remove an assumption from a variable the same way that we would remove a specific value from the variable, by unassigning the variable.

```
> x := 'x';
> about( x );
```

Here are two other ways to use the `assume` command to make the same assumption about the posible values of the variable `x`.

```
> assume( x, RealRange(Open(0),infinity) );
> about( x );
```

And we can also use the following, much more convenient, notation.

```
> assume( x>0 );
> about( x );
```

Notice that the assumption does not prevent us from assigning any value to the variable, since the process of assigning a value causes the assumption to be removed.

```
> x := -1;
>
```

There is a larger set of numbers for which the identity $\sqrt{x^2} = x$ is true. Instead of assuming that x is a positive real number, we can assume that x is a complex number with positive real part. In other words, the following identity is also true.

$$\sqrt{x^2} = x \text{ is true for all complex numbers } x \text{ with } 0 < \Re(x).$$

The `assume` option for `simplify` is not versatile enough to demonstrate this identity. We have to use the `assume` command. Here is how we use the `assume` and `simplify` commands to demonstrate this identity. First we use `assume` to tell Maple that the unassigned variable `x` is meant to represent only complex numbers with positive real part (but first we need to unassign `x` to remove the constant value that we just gave to it).

```
> x := 'x':
> assume( Re(x)>0 );
```

We can use `about` to check the assumption about the values represented by `x`.

```
> about( x );
```

The `simplify` command, without any options, will make use of this information about the values that `x` can represent and demonstrate the identity that we want. (Notice that automatic simplification does not work this time.)

```
> sqrt(x^2);
> simplify( % );
>
```

Here is still another version of the previous identity.

$$\sqrt{x^2} = x \text{ if } x \text{ is purely imaginary with positive imaginary part.}$$

Here is how we can use **assume** to make this assumption about **x**. First, we make the assumption that **x** is a purely imaginary complex number.

```
> assume( Re(x)=0 );
```

Now we make the additional assumption that the imaginary part of **x** is positive.

```
> additionally( Im(x)>0 );
```

The **about** command shows us that the last two commands have a cumulative affect on the assumptions that Maple now makes about what numbers **x** can represent.

```
> about( x );
```

And now the **simplify** command will use these assumptions to demonstrate the identity for us.

```
> sqrt(x^2);
> simplify( % );
>
```

The **assume** command has many advantages over the **assume** option to **simplify**. As we have just seen, it is more powerful in that it allows us to make assumptions that cannot be made from within the **simplify** command. Also, once we use the **assume** command to make an assumption, that assumption can be used by many other commands, not just the **simplify** command. In fact, the assume facility is useful for more than just demonstrating identities. For example, below we show how it can also be used for doing calculations with quantities that depend on a parameter. But there are some limitations on the use of **assume**. For example, if we combine the previous two examples, we get the following identity.

$$\sqrt{x^2} = x \text{ if } x \text{ has positive real part, or if } x \text{ has zero real part and positive imaginary part.}$$

I do not know of any easy way to use the **assume** command to make the assumptions needed to demonstrate this identity.

```
>
```

Not all commands are aware of the assumptions that **assume** can place on a variable. For example, the **evalb** command is not aware of these assumptions. Let us assume that **x** is between -1 and 1.

```
> assume( x, RealRange(-1,1) );
```

If **x** is assumed to be between -1 and 1, then each of the following three inequalities should be true. But **evalb** leaves the inequalities unevaluated.

```
> evalb( x^2<=1 );
> evalb( x<=2 );
> evalb( (x-1)*(x+1)<=0 );
```

Fortunately, Maple has a separate command that is aware of assumptions and lets us ask questions about a variable in an equation or inequality. We can use the **is** command to evaluate all three of the above inequalities.

```
> is( x^2<=1 );
> is ( x<=2 );
> is( (x-1)*(x+1)<=0 );
```

`is` will return `true` if an equation or inequality is true for every possible value of every variable in the relation, and `is` returns `false` if the equation or inequality is false for some possible value of a variable. The following `is` command returns `false` since `x>0` is not true for all of the values the `x` represents.

```
> is( x>0 );
```

Notice that the inequalities `x>0` and `x>2` are different in an important way. While `x>0` is false for some of the values that `x` represents, `x>2` is false for *all* of the values that `x` represents. Maple's assume facility has as interesting (undocumented) command, `coulditbe`, that lets us distinguish between these two possible reasons for `is` to return `false`.

```
> coulditbe( x>0 );
> coulditbe( x>2 );
```

`coulditbe` will return `true` if an equation or inequality is true for some possible value of the variables in the relation, and `coulditbe` returns `false` if the equation or inequality is false for every possible value of every variable. So given an equation or inequality, if `is` returns `true` then the relation is true for all possible values of all the variables, if `coulditbe` returns `false` then the relation is false for all possible values of all the variables, and if `is` returns `false` and `coulditbe` returns `true` then the relation is true for some possible values and false for some possible values. (Could it be that `is` returns `true` and `coulditbe` returns `false`?)

```
>
```

If there are no assumptions on a variable in an inequality or equation, then `is` returns the special value `FAIL`, which means that `is` cannot determine if the inequality or equation is true for all values or false for some value of the variable.

```
> x := 'x';
> is( x>0 );
>
```

Here is an example of `is` and `coulditbe` working with more than one variable.

```
> assume(x>1);
> assume(y<0);
> is( x-y>1 );
> is( x+y<1 );
> coulditbe( x+y<1 );
```

Unfortunately, `coulditbe` can sometimes have trouble producing the correct result. The following command should return `true`.

```
> coulditbe( x+y>1 );
```

Here is an example where `is` has trouble producing the correct result.

```
> assume( x>1 );
> assume( y>1 );
> is( (x-1)*(y-1)>0 );
```
But if we reverse the direction of the last inequality, then `is` has no problem, which makes the previous result all the more surprising.
```
> is( (x-1)*(y-1)<0 );
>
```

Here is a typical example of how `assume` can be used while doing a calculation. Let us define a function `f` with a parameter `a`.
```
> a := 'a': x := 'x':
> f := x->exp(a*x);
```
Unless we know something about the parameter `a`, we cannot say much about the limit of `f` as `x` goes to infinity.
```
> limit( f(x), x=infinity );
```
The previous command returned unevaluated, reflecting Maple's lack of knowledge about `a`. Now let us assume that `a` is positive.
```
> assume( a>0 );
> Limit( f(x), x=infinity ) = limit( f(x), x=infinity );
```
And now assume that `a` is negative.
```
> assume( a<0 );
> Limit( f(x), x=infinity ) = limit( f(x), x=infinity );
>
```

Here is an interesting example of what `assume` can do with a piecewise defined function.
```
> f := x -> piecewise(x<=0, x^2, x<1, x^3, x>=2, x^4);
```
If we evaluate this piecewise defined function at an unassigned variable, then we just get the function displayed as an expression.
```
> f(y);
```
Assume that `y` is a negative real number.
```
> assume(y<0);
```
Now if we evaluate `f` at `y`, we get the appropriate sub expression from the definition of `f`.
```
> f(y);
```
However, if we assume that `y` is a positive real number and then evaluate `f` at `y`, Maple does not return just the two appropriate sub expressions. Instead, Maple returns the whole definition of `f`.
```
> assume(y>0);
> f(y);
```
If we add another assumption to `y` and then re-evaluate `f` at `y`, then Maple can return an appropriate sub expression from the definition of `f`.
```
> additionally(y<1);
> f(y);
>
```

```
>
```

4.5. Online information on assume

Maple's assume facility is pretty complex and the online documentation is not very helpful. Here is the help page for the `assume` command. It is also the help page for the `about`, `additionally`, and `is` commands.

```
> ?assume
```

The `assume` command assigns properties to names. The following two help pages give some information about properties.

```
> ?property
> ?asspar
```

One way to get a sense of how properties are defined is to look at the definitions of several predefined properties. Here are some examples.

```
> about(complex);
> about(real);
> about(positive);
> about(fraction);
> about(integer);
> about(posint);
> about(odd);
```

We mentioned the use of inert functions for displaying nicely typeset identities. The following help page has some general information about inert functions.

```
> ?value
```

There are other ways to demonstrate an identity besides deriving it algebraically. The `testeq` command provides a probabilistic way to determine if an equation is an identity or not.

```
> ?testeq
```

```
>
```