Linear syntax for communicating elementary mathematics

C J Sangwin* P Ramsden†

Abstract

We consider computer aided assessment (CAA) of mathematics in which a student provides an answer in the form of a mathematical expression. A common approach is for CAA system implementors to adopt a linear syntax to allow students to communicate their answer to the machine. In this paper we consider the problems students encounter when mediating between (i) traditional mathematical notation and (ii) the requirements of a strict computer algebra system (CAS) syntax. We compare the linear syntaxes of five commonly used general purpose CAS, and report surprising variety even at the elementary levels.

1 Introduction

Mathematical notation is the product of a long tradition, and has evolved to take advantage of a rich set of special symbols, together with their relative size and position on a two dimensional page. Certain aspects reflect good notational design, and even arrangements that may be less happy in themselves have been embedded, perhaps irreversibly, by usage.

> Examples of the power of a well contrived notation to condense into small space, a meaning which would in ordinary language require several lines or even pages, can hardly have escaped the notice of most of my readers. [2, pg 330]

This quotation points to the *communicative* power of a good notation, but in addition to this a well-designed notation has the ability to *aid calculation and thought*. However, when typing a mathematical expression using a keyboard one has only a one-dimensional string of symbols taken from a limited alphabet. Translating mathematics into such a format is a fundamental problem. In this paper we consider the difficulties of mediating between (i) traditional mathematical notation and (ii) the requirements of a strict computer algebra system (CAS) syntax.

We were motivated to undertake this work while designing an input system for mathematical computer aided assessment (CAA). In particular, we are interested in CAA in which a student provides a mathematical expression and the system establishes properties automatically, rather than (say) selecting from among a list of suggested answers, as in a multiple choice question. Using a mainstream CAS to support this has become increasingly popular over the last five years. Perhaps the first was AiM, described by [13] which uses Maple. CalMath uses Mathematica, CABLE uses Axiom and the

*Maths, Stats & OR Network, School of Mathematics, University of Birmingham, Birmingham, B15 2TT, Email: C.J.Sangwin@bham.ac.uk

†Imperial College, London, SW7 2AZ Email: p.ramsden@imperial.ac.uk

STACK system (`http://www.stack.bham.ac.uk`) uses Maxima ([12]). From private correspondence, the authors are also aware of Derive being used in a similar way. In these cases the CAA system is layered on top of the CAS, and students must use the linear syntax of the CAS itself. It is not necessary to use a CAS to process student responses for CAA and many systems have their own input syntaxes. The Metric system of Ramsden and May (see [10]) is one example, and our discussion is just as relevant to these systems. In this paper we

1. review the syntaxes of Maple, Maxima, Axiom, Derive and Mathematica and compare these with mainstream mathematical usage;
2. examine students' ability to use one of these when communicating with the CAA system AiM, which uses Maple's syntax;
3. argue for, and discuss the characteristics of, an "informal syntax" designed for the purpose of mathematical CAA of school level algebra and calculus.

We draw a distinction between two *categories of user*: the professional and the student; we make the case that students, in particular, may experience problems connected with strict syntax. We also draw a distinction, within the 'student' category, between two *modes of use*: on one hand, problem-solving and calculation and, on the other, assessment. It is in the context of assessment that, we argue, these syntax issues become especially problematic. The following quotation is from a student's detailed comments, in course feedback, about the AiM system of [13], about which we shall say more below:

> *I feel the aim system is reasonably fair, however i have lost a lot of marks in quiz 3 for simple syntax errors.*

Strictly, this student had not made a *syntax error*: his or her interpretation of this string was simply different from that of AiM's underlying CAS; this resulted in repeated incorrect attempts and presumably frustration. This is despite the fact that he or she had answered the problem correctly at the level of mathematics: the 'error' was purely technical. Whatever the benefits or drawbacks of a non-standard, especially precise syntax in problem-solving and calculation, such a syntax has clear disadvantages if students taking tests are required to use it, precisely because of this risk of failing on a technicality. This, we feel, is unacceptable.

2 Methodology

2.1 Background to existing CAS

For the purposes of this paper we focus entirely on expressions entered in the CAS's 'basic' linear syntax, as strings of keyboard characters. The first section of results contains a review of Derive 5, Maple 9.03, Mathematica 5.0 and Maxima 5.10.0, and a version of Axiom compiled from the source code of November 24th 2004. All of these have been used in online CAA. The differences between the various CAS have been discussed elsewhere, for example [6], or [14]. However, these comparisons are from the point of view of the research mathematician. Furthermore, although both these works contain chapters on mathematics education, neither addresses the problems students experience when using the *linear*, typewritten syntax of the CAS concerned; instead, each concentrates on issues connected

with the use of the system's graphical Front End. Front End issues, while important, are not our concern here. Many CAS have well-developed user interfaces that support 2D typesetting. Such facilities are not available to an assessment system built using the CAS and so, for this reason, students are expected instead to type their answers into a text area, raising the mediation issues that we treat here. We examine the very basic linear syntax of each system, and pay particular attention to the syntax for arithmetic, algebra and the entry of functions encountered in elementary mathematics, such as $\sin$, $\log$ etc. We then consider briefly the syntax for entry of sets and lists.

In this paper we restrict comments to *input syntax* and refrain from commenting on evaluation and simplifications. Establishing properties of expressions, eg equivalence is central to CAA, but lies outside the scope of this paper: see [11].

2.2 Students' errors with Maple's linear syntax

To examine students' use of a strict CAS syntax we consider the computer aided assessment system AiM ([13]) built upon Maple. AiM is an internet-based system that can be used for both formative and summative assessment, both of which are fully automated (indeed, it might more accurately be described as a system for computer assessment rather than computer-aided assessment, although the latter is the conventional term). Space precludes a detailed description of AiM here; see instead [13] or [11]. In the AiM system, students' answers are entered using Maple's linear syntax, and these are evaluated. AiM has been in regular use in a number of university mathematics departments in the United Kingdom and internationally since 1999. We consider the use of AiM by first year students at one such institution. Each week during the first year of study, approximately 200 students take a quiz consisting of between 10 and 20 short answer practice questions, usually randomly generated within carefully structured templates.

We are interested in the use of syntax for genuine tasks, not when learning the syntax itself. Hence, records from the first week of the first year are excluded. Only students' attempts at actual mathematical questions are considered. When answering such questions the students' attention should be focused on the mathematics and the task of using the CAS syntax is subsumed (in the sense of [8]).

AiM records every attempt of the student together with the actions requested and all outcomes, such as feedback. For each syntactically invalid expression the system applies various heuristics for common mistakes in an attempt to provide helpful feedback. These heuristics result in a "ValidationNote", which indicates what might be wrong with a particular response, and it is these which we use to help identify and group syntax errors. Unfortunately, these can be misleading as to the actual source of the error. What the validation notes do achieve is a grouping of errors into similar kinds. Really the only way to establish exactly what the student has done wrong is to examine each answer, which we do in subsequent sections in greater detail.

3 Results: existing CAS syntax

Many CAS terminate an expression with a punctuation character which also acts as a separator between adjacent expressions and controls whether the result is to be displayed. These are shown in Table 1. For Derive we have listed the terminating symbol = as execute and display. This performs "basic simplification", and other terminating symbols perform different simplifications.

CAS	Display	No display	π	e	$\sqrt{-1}$	∞
Axiom	[NONE]	`;`	`%pi`	`%e`	`%i`	`%plusInfinity`
Derive	`=`	[NONE]	`pi`	`#e`	`#i`	`inf`
Maple	`;`	`:`	`Pi`	`exp(1)`	`I`	`infty`
Mathematica	[NONE]	`;`	`Pi`	`E`	`I`	`Infinity`
Maxima	`;`	`$`	`%pi`	`%e`	`%i`	`Inf`

Table 1: Entry of basic commands

In all the CAS the default number base is ten, with rational numbers being expressed as a division using the symbol / (for example `2/3`). The decimals separator is a full stop, which is uncontroversial in the United Kingdom, Ireland and their former colonies. However, most of Europe uses a comma to denote the decimal separator instead. There are also differences in printed notation between the full stop, eg 3.1415, and a center dot, eg $3 \cdot 1415$, which cannot be captured using keyboard input. The extent to which digits are grouped to ease the reading of longer numbers varies from country to country, as do the symbols used to separate the groups. A full stop, comma, space or apostrophe are all used for this purpose. At variance with [5], none of the CAS parse numbers with digits grouped and separated with a space, eg `23 000`. Mathematica and Derive accept this as valid, with the interpretation of implied multiplication. All CAS reserve a comma to separate distinct data items, eg in a set or list.

Another ambiguity occurs with juxtaposition of integers and fractions, such as in $2\frac{3}{5}$. Only Mathematica accepts this as valid, but interprets it as $\frac{6}{5}$: an implied multiplication, rather than as an addition, ie as $\frac{13}{5}$. Yet another arises in combinatorics and probability, where multiplication of integers is sometimes denoted using a center dot, so that

$$\frac{4 \cdot 3}{2 \cdot 1} = 6,$$

rather than interpreting $\cdot$ as a decimal separator.

Differences in the ways very simple numbers are interpreted are illustrated in Table 2. In all systems except Derive and Mathematica, the string `2.5e-2` is interpreted as scientific notation for the floating point number 0.025. Mathematica requires either `2.5*10^-2`, `2.5 10^-2` or, unusually, `2.5*^-2`. Derive needs `2.5*10^-2`, which in the default settings is coerced into a rational $\frac{1}{40}$. No other system coerces floats into rational numbers by default, although this paper is not about simplification or evaluation settings.

When a unary minus precedes a number one might expect the unary minus to bind more tightly than any other prefix, postfix or binary operator, to signify that -4 is a single entity, the number minus four. This interpretation would result in `-4^2` $= (-4)^2 = 16$. However it does not, and all CAS interpret `-4^2` $= -(4^2) = -16$. Inspection of many contemporary text books reveals that mainstream usage adopts a similar approach to the unary minus as the CAS illustrated. However, there is inconsistency among the various systems when a unary minus follows directly from a binary arithmetic operation, as with the string `4^-2`. Here only Maple returns a syntax error. The rational number $\frac{1}{2}$ is entered as `1/2`. This is literally interpreted as the natural number 1 divided by 2, so that the division order of precedence is used, rather than treating $\frac{1}{2}$ as a single mathematical entity: a rational number[1].

[1] Again in evaluation, there are very subtle differences. Maple evaluates `-9^1/2` as $\frac{-9}{2}$, whereas all other systems return

CAS	`23 000`	`2.6e-2`	`4^-2`	`-9^1/2`	`x+-2`	`x*-2`
Axiom	ERROR	0.026	$\frac{1}{16}$	$-\frac{9}{2}$	$x-2$	$-2x$
Derive	0	$\frac{13e}{5}-2$	$\frac{1}{16}$	$-\frac{9}{2}$	$x-2$	$-2x$
Maple	ERROR	0.026	ERROR	$\frac{-9}{2}$	ERROR	ERROR
Mathematica	0	$-2+2.6e$	$\frac{1}{16}$	$-\frac{9}{2}$	$-2+x$	$-2x$
Maxima	ERROR	0.026	$\frac{1}{16}$	$-\frac{9}{2}$	$x-2$	$-2x$

Table 2: Entry of numbers

CAS	**Assignment**	**Equation**	**Function definition**	**Boolean infix**
Axiom	`:=`	`=`	`==`	`=`
Derive	`:=`	`=`	`:=`	`=`
Maple	`:=`	`=`	`:=`	`=`
Mathematica	`=` (or `:=`)	`==`	`:=` (or `=`)	`==`
Maxima	`:`	`=`	`:=`	`=`

Table 3: Expressing different forms of equality

Entry of constants π, e, $\sqrt{-1}$ and ∞ are shown in Table 1, with particular attention being paid to capitalization. Here ∞ is the positive real infinity, CAS usually differentiate between this, negative real infinity and complex infinity. None of the systems used j in place of i to denote $\sqrt{-1}$. For Mathematica, the table above gives the InputForm versions which are supported in the other input formats.

In all the CAS, both `i` and `e` are interpreted as arbitrary variables, not mathematical constants. Hence in `e^x` the e is simply an arbitrary and undefined variable. Systems do allow the constant defined in Table 1 to be raised to a power using the exponential notation `^`. For example, in Maxima `%e^x` is legitimate. The extent to which simplification is necessary to establish equivalence of these two forms varies between the different CAS.

Note that in Maple, `pi` and `Pi` are different, with the former simply being the variable denoted by the Greek letter. However, both are displayed exactly as 'π'. This is not the case with the Greek letter `gamma` which is interpreted by Maple, not as a variable, but as Euler's constant, $\gamma \approx 0.5772$.

Robert Recorde originally used his parallel lines $=$ *"to auoide the tedious repetition of these woordes: is equalle to"*. Their usage has evolved since 1557 to stand for four quite different operations which are (i) assignment of a value to a variable; (ii) to denote an equation yet to be solved; (iii) definition of a function; and finally (iv) as a Boolean infix operator, returning either TRUE or FALSE. The choices made by various CAS are shown in Table 3. In Mathematica, though not in some other CAS implementations such as Maple, senses (ii) and (iv) are essentially equivalent.

Note that Mathematica distinguishes between *immediate assignment* which happens when the variable or function is defined, and *delayed assignment* which happens when the variable or function is called. The symbol `=` is used for the former, and `:=` for the latter, so that the appropriate assignment for

$-\frac{9}{2}$. In the former the first operation is division, in the second it is unary minus.

variables is usually immediate and functions delayed. This is a powerful and useful distinction but experience with students indicates that this is apt to confuse.

There are differences in the Boolean functions (*verb* forms) for logical NOT, AND, OR etc, and corresponding *noun* forms used as connectives, which we do not detail here.

Inequalities use their keyboard symbols, with non-strict inequalities using `<=` or `>=`. All systems reject minor variants of this syntax, including `< =` (ie with a space between), or `=<`. Only Mathematica accepts chains of inequalities, such as $1 < x < 2$, interpreting this as a list of inequalities, all of which must hold.

3.1 Arithmetic and basic algebraic expressions

All the CAS considered use a linear direct algebraic logic syntax, with infix binary operators for arithmetic, rather than Polish Notation. This corresponds closely to traditional written mathematics, eg "two times x" is expressed as `2*x`, rather than `* 2 x`. All used parentheses for grouping terms.

The arithmetic binary operations $+$, $-$ and $/$ were identical in all systems. All but Derive and Mathematica were strict in requiring an explicit multiplication sign `*`. Derive allows implied multiplication under some circumstances. That is to say it interprets `2x` as `2*x`. Mathematica's InputForm exploits the fact that parentheses are never used to enclose function arguments, see Section 3.2, and therefore allows a multiplication sign to be omitted wherever this would not be ambiguous. For example `2x`, `x(x-1)` and `(x+1)(x-1)` are all interpreted multiplicatively, but `xy` is interpreted as a single variable with a two letter name. However, in Mathematica's TraditionalForm, parentheses may be used to enclose function arguments, creating fresh potential for ambiguities. In TraditionalForm `2x` and `(x+1)(x-1)` are interpreted as multiplication, whereas `x(x-1)` is treated as a function. To force multiplication, the user must inert either an asterisk or a space character. Note, however, that TraditionalForm is not a linear syntax but a 2D formatted one, dependent on Mathematica's Front End functionality, and therefore not in general available to a custom CAA application.

Exponentiation was denoted using the symbol `^` in all systems, with Axiom also using `**`. Factorials are obtained using a postfix operator, eg `n!` in all systems except Axiom, which used the function `factorial(n)`. Prefix notation is also available in some systems for the arithmetic operators. For example, `x + y` could be entered in Mathematica as `Plus[x, y]` or in Maxima using `"+"(x,y)`.

3.2 Functions

The syntax for functions, eg $\sin(x)$, shows significant systematic differences. Mathematica uses the function name followed by the argument enclosed in brackets (for example f[x]), and parentheses are reserved for grouping terms. Other CAS use parentheses instead (for example f(x)). Both Derive and Axiom accept a space to signify function application. For example, `sin x`, is interpreted as $\sin(x)$ with parentheses required only to group terms, eg `cos (n*pi)`. Here the space is optional. Using a space in this way reduces the number of symbols. Further, `sin sin x` is interpreted as a composition, $\sin(\sin(x))$, which can be written naturally as $\sin^2 x$. This approach led Babbage to his "calculus of notations", [1], with the natural consequence that the inverse is now written as $\sin^{-1}$. So one could write

$$\sin^{-1}\ \sin\ x = \sin^{(1-1)}\ x = \sin^0\ x = x.$$

(We do not comment here whether this is mathematically legitimate.) It is interesting to note the inconsistency with modern usage, in which $\sin^2(x)$ is generally used to denote $(\sin(x))^2$, while $\sin^{-1}(x)$ is, in the English-speaking tradition, taken as the inverse, rather than the reciprocal of $\sin(x)$. Derive was unique in accepting `sin^2(x)` as $(\sin\ x)^2$, and was consistent in interpreting `sin^-1(x)` as $\frac{1}{\sin\ x}$. However, this differs from current usage for the inverse. Mathematica accepts `Sin^2 [x]` and `Sin^(-1) [x]`, but does not interpret them. One quirk unique to Maple is the interpretation of `2(x+1)` as the application of the constant function 2 to the argument $(x+1)$, which results in the value 2. Hence `sin^2(x)` is the sine function raised to the power of `2(x)`, which is the constant function 2 and argument x. This is interpreted as $\sin^2$, which is a function, rather than the result of applying a function to an argument[2]. The expression $\sin^2$ when applied to x, is interpreted as $\sin(x)^2$, not as a composition. As with numbers above, `sin^-1` is rejected, but `sin^(-1)` is interpreted as $\frac{1}{\sin}$, not the inverse. Axiom and Maxima rejected all attempts to exponentiate functions without arguments.

For the trigonometrical functions all CAS use `sin`, `cos`, `tan`, with Mathematica using a capital initial letter. In all but Derive the system enforces *radian angular measure*. Derive offers the user a choice, with radians the default setting. `sec`, `csc`, and `cot` denote the secant, cosecant and cotangent respectively. Hyperbolic functions were denoted by `sinh` etc. For inverse trigonometrical functions Maple used the pattern following `arcsin`, with `Arcsin` treated as inert. Mathematica uses `ArcSin`, Axiom, Maxima and Derive use `asin` and so on.

Maxima is case sensitive and in Derive the case sensitivity can be altered. Mathematica capitalizes the initial letter of all inbuilt keywords in InputForm and StandardForm, though capitalization of common mathematical functions is optional in TraditionalForm. Maple is case sensitive and uses an initial capital letter to denote a noun (inert) form which is not simplified. In Derive the case sensitivity can be altered. With this in mind, square roots are obtained as follows. In Axiom, Derive and Maple, `sqrt(x)` is used. The form `Sqrt(x)` is inert in Maple. Mathematica uses `Sqrt[x]`.

All systems implement the exponential function as `exp(x)`, with Mathematica using its consistent variation `Exp[x]`. In all the CAS implementations `log(x)` refers to the natural logarithm. Derive also used `LN` to denote the natural logarithm.

All systems use `abs` to denote the modulus function, rather than using the traditional written form $|x|$, where as usual Mathematica uses `Abs[x]`. Although the symbol `|` does appear on a keyboard there would be problems in identifying matching |'s in an expression containing more than one application of the modulus function, particularly when combined with implied multiplication: eg `|a|b-c|d|`.

3.3 Sets and lists

In Maple and Derive, sets are a core part of the CAS and are defined using curly braces, for example a, b, c. In Mathematica, sets as such are not implemented as a distinct data structure; instead, lists (see below) support certain set theoretic operations. Axiom uses a construction `set[a,b,c]`, Maxima uses both `set(a,b,c)` and `{a, b, c}`.

A list is expressed using square brackets, such as `[1,2,2,3]`, in all systems except Mathematica,

[2]Some CAS allow the manipulation of symbols representing functions as objects. Maple's `D` operator acts on a function, for example `D(sin)` simplifies to cos. At the stages of learning we consider here students are not expected to have encapsulated the notion of function to that of a single object and we may omit consideration of such notations: a substantial simplification.

which uses curly braces for both sets and lists. Derive and Maxima treat lists in a very similar way to matrices with one row, or as row vectors. Maple does not take this view having a rather stronger system for data types.

The use of square brackets to denote lists precludes their use to also denote closed real intervals. Parentheses, which CAS use to denote grouping of terms, are also used in mathematics to denote both open real intervals and tuples of numbers, for example co-ordinates. Axiom interprets `(1,2)` as a tuple. Maple treats `(1,2)` as an expression sequence, but strips off the parentheses. Maxima uses parentheses containing comma separated expressions as a programming construct in which it evaluates the expressions in sequence and returns the value of the last expression. In all the CAS, an expression such as `(1,2]` was rejected as syntactically invalid.

We have seen significant differences in CAS syntaxes, even at an elementary level. Such differences become compounded: choices at one level affect those at the next. Since computer systems are unforgiving to even minor syntax errors, these apparently minute variations really do matter. At this stage it is well to recall again [2, pg 326] who's comments on *"a profusion of notations (when we regard the whole science) which threaten, if not duly corrected, to multiply our difficulties instead of promoting our progress"* apply equally well to the variety of syntaxes for existing CAS.

4 Results: students' use of the AiM system

These results illustrate students' use of Maple's syntax within the AiM CAA system, and data is taken from one year comprising 201 students. We must consider the process used by students to enter their answers: eventually students are able to express their answer in a syntactically valid. Hence, for each invalid expression a subsequent edit and re-validation will provide valid form. The student will probably then ask for this to be marked, generating a third data point. These are all included in an average and so for each invalid expression there will be one or two extra subsequent valid expressions. Hence the percentage success rates reported here mask a *much more serious problem*.

We compare the beginning of the first semester, with the end of the second semester. Data set 1 is drawn from three tests in weeks 2 and 3, containing a total of 11 questions. Data set 2, with 12 similar questions was from the last two weeks of the second semester, some five months later. We consider only those questions for which the response was a linear algebraic/trigonometric expression. Other responses, eg lists or matrices, were also interspersed with these questions.

Data set	Responses	% Syntax error	% `*`'s	%)'s	Indet	bad −
1	3550	16.51	7.81	3.58	1.83	0.65
2	2951	6.57	1.80	2.17	0.34	0.98

The headings `*`'s and)'s respectively indicate the percentage of all responses with missing multiplication, or mismatched parentheses. The "Indet" error indicates a "strange indeterminant". For example typing `exp^(5*x)` would generate this error, since `exp` is being interpreted as a variable name. Two questions from data set 1 were of this type and accounted for 1.44% of all syntax errors. These were precisely the problem of using a *function* for the exponential e^x, rather than having the letter e assumed to represent the base of the natural logarithm and using the `^` symbol. In the second data set one question accounted for 0.41% of all syntax errors as "bad minus sign". This is caused by an expression such as `x^-4`, whereas the system expects parentheses such as `x^(-4)`.

The error rate drops significantly to 6.57% by the end of the second semester. This is still high, with $1/3$ of all errors being missing parentheses. Note that typical responses from data set 1 were expressions such as `9*exp(-1)`, and `8/11*x^(11/8)+c` whereas later in their first year more complex expressions are required such as
`-ln(2*x^7+x^2-4)+c` or `(1/8*t+3/2)*x^2+(-t-31/2)*x+35+15/8*t`. Despite this added complexity, the competency had significantly improved, as one might expect. Missing parentheses become the most significant concern, however these are still necessary in an informal linear syntax.

4.1 Algebraic expressions: detailed errors

A more detailed analysis is obtained by looking at results to a single question. For example, in week 2 students were asked to integrate $x^{\frac{p}{q}}$ where p and q were small integers. Students' raw answers will be of the form `7/12*x^(12/7)+c`, involving multiplication, fractional numbers, exponentiation and parentheses.

The system recorded 267 responses from the 158 students who attempted this question and 72.8% of these students gave the correct answer on their first attempt, 16.5% on their second attempt. Here 5.1% of students missed parentheses entering one of either `7/12*x^12/7` $= \frac{x^{12}}{12}$, or `x^(12/7)/12/7` $= \frac{x^{\frac{12}{7}}}{84}$. All these students had the correct answer in mind, but these are essentially syntax problems: the student has entered an expression which is interpreted in a way other than expected and so had failed to communicate their answer effectively. Other syntactically valid expressions entered by 2 students were `7/12(x^(12/7))`, and `(7/12)(x^(12/7))`. Maple interpreted these as function application. We found 19 with a missing multiplication sign and 5 with mismatched parentheses. Three responses contained floating point numbers which are rejected as "invalid", although this is for pedagogic not syntactical reasons.

The question "Find $\int a\sin(bx)\,dx$" was also set during week 2. Here we took a and b as small integers excluding -1, 0 or 1, and the trig function was randomly selected from either $\sin$ or $\cos$. The system recorded 314 responses from the 157 students who attempted this question. Again, the mathematics was unproblematic: 71.3% of students gave the correct answer on their first attempt, and 21.2% on their second.

Again, missing multiplication signs and unmatched parentheses predominated the syntax errors, with 26.4% of first responses containing one of these mistakes. Additional mistakes specific to this question included responses such as the following `(-1/4)*sin4x+k`. Notice the unnecessary parentheses surrounding the fraction, but those to denote function application are missing, as is a multiplication sign. Perhaps surprisingly, only one student used white space instead of parentheses, typing `-2*sin 2*x + c`.

It should be noted that all the preceding examples are taken from early in the first semester, when students are still learning the syntax, and using it perhaps for the first time in a real application. Hence, it is not surprising to find this variety or frequency of errors.

4.2 Inequalities

To illustrate inequalities we considered all responses to the question "Solve $x^2 - 14x + 48 \leq 0$". Random versions were given in which the quadratic had roots $a \in \{2, \cdots, 7\}$, and a selection from $a + \{2, \cdots, 5\}$. This was set during the second week of the first semester, during which students had comparatively little experience. Hence, students were given the following syntax hint. *" Give your answer as a collection of inequalities such as* `x < -7 and x >=5`*. Don't use quote marks, and you can replace* `and` *with* `or` *if you need to."* 159 students attempted the question, and 57.9% obtained full marks; 87.4% made fewer than four incorrect attempts, eventually giving a correct answer.

However, 29.2% of students' answer sequences contained an invalid response, despite the clear syntax hint. Of these, 11.3% of students entered their first answer as one would perhaps write it with an implied logical connective: `6<=x<=8`, and a further 6.4% of students made this syntax mistake somewhere else in their answer history. So almost 18% of students made this kind of error somewhere, despite the explicit syntax instructions in the question itself: traditional written mathematics exerts a powerful force on the mind. In addition to this, 8.9% of students used either `=>` or `=<` in their answer. A significant number had an answer history consisting of a sequence such as (i) `6<=x<=10`, (ii) `6<=x=<10`, (iii) `x>=6 and x<=10`, where the final answer is both syntactically and mathematically correct. Notice here that the student has the right idea, but cannot express it.

4.3 Sets

The following question requiring entry of sets occurred in the second semester.

> If $f(x) = e^x(x^2 - 8) + 8$, find all stationary points of $f(x)$.

The randomly generated quadratic was guaranteed to have small integer roots. Students were given the following explicit syntax hint. *"Enter the x-values as a set, e.g.* $\{5, 7\}$*"*. The system recorded 173 responses to this problem, from 138 students, and 89.3% obtained the correct answer on the first attempt and a further 7.3% on the second. However, 19 responses (11.0%) contained a syntax error. Ten students tried to enter the *coordinates* of the points, eg "`(2,5.92) and (-4,222.40)`" or "`(2,5.92),(-4,222.40)`", effectively ignoring the syntax hint. 7 students used the wrong kind of brackets, choosing either parentheses or square brackets instead of the curly brackets indicated. These students may simply not have been able to discern which brackets to use, or they may have ignored the hint.

We conclude that a strict syntax for inequalities is inappropriate for assessment, even for university mathematics students, particularly with `=<` and `< =` (ie unnecessary space between symbols) which are unambiguous. The second, perhaps more important, conclusion is that clear syntax hints do not necessarily ameliorate the problem. Students solve the problem and then expect to type in the answer using a syntax which closely mirrors their written work. Their mental absorption in the task in hand results in their temporarily forgetting the syntax hint.

We note that the students in this study are taking either single honours or joint honours mathematics degree courses. These are some of the highest achieving mathematics students in their generation. We conjecture that if these students struggle with any aspect of the input syntax, their peers on other degree programmes or younger school students are even more likely to encounter these problems. More worryingly, we might also expect others to take longer to learn the syntax, and hence for this to

constitute an even more significant barrier to expressing their mathematical ideas. These conclusions have implications for both formative and high stakes assessment.

5 Discussion

Students make a significant number of mistakes, even when prompted with explicit syntax instructions. Although these may be apparently trivial to the experienced mathematician our experiences of using CAA strongly suggests they are quite significant to the student. To address this we consider whether an "informal syntax" might be developed. Both the authors have experience of designing and implementing such an informal syntax in CAA which rely on various heuristics. The designers of other CAA systems have taken a similar approach and such heuristics are also the basis of related applications: character recognition systems, and the mathematical pen-based entry systems. Another example is the ambiguous grammar for mathematics implemented in the Tables of Integrals Look Up (TILU) system, `http://torte.cs.berkeley.edu:8010/tilu`. In this section we discuss these issues, beginning with the *principles for mathematical notations* set out by [3].

(B1) All notation should be as simple as the nature of the operations to be indicated will admit.

(B2) We must adhere to one notation for one thing.

(B3) Not to multiply the number of signs without necessity.

(B4) When it is required to express new relations that are analogous to others for which signs are already contrived, we should employ a notation as nearly allied to those signs as we conveniently can.

(B5) Whenever we wish to denote the inverse of any operation, we must use the same characteristic with the index -1.

(B11) Parentheses may be omitted, if it can be done without introducing ambiguity.

Principles (B6)–(B10) inclusive refer to operations in higher mathematics or to fonts and formatting on the printed page and so are not relevant. We note that these principles are by no means universally accepted; for example, (B5) is used for trigonometric functions in the English speaking world but not in other European traditions, and is nowhere used for exponential functions. Instead we propose to augment this list with the following.

(P1) *Informal linear syntax should correspond with printed text and written mathematics.*
Students can rightly expect us to be consistent in the way mathematics is expressed, as far as is reasonable given the constraints of a one dimensional input mechanism.

(P2) *Informal linear syntax should not obstruct learning the strict syntax of a CAS.*
There should be nothing to un-learn at a later stage.

Our research and development work in CAA suggests that the conflicts are so serious that it is *impossible* to implement an informal *syntax*. Instead a more protracted process is necessary during which the student gradually refines their input in the light of feedback from the system. For example, the

one dimensional string typed might be displayed in a two dimensional format so that explicit grouping of expressions can be more easily perceived in the traditional way. Multiplication might be made explicit with a $\times$ or $\cdot$. While this mirrors very closely the mechanism in AiM and Metric we still feel it necessary to allow a more liberal interpretation of the symbols than is currently in place.

It follows that there might be some 'unlearning' to be done at the transition from written mathematics to online assessment and then again when a strict CAS syntax is encountered. The ability to communicate mathematics with a machine is a skill which is likely to become increasingly important. We feel there is a difference between introducing a new notation to express an established idea, and that when the meaning of an established notation is changed. We have strenuously avoided the second of these. Hence, we opt to weaken (B2) so that forms close to written mathematics and strict CAS syntax should coexist.

The largest source of students' errors is a missing `*` and given that juxtaposition traditionally denotes multiplication this is not particularly surprising. Work such as [9] also found that *"for some students the surface features of ordinary notation provide a necessary cue to successful syntax decisions"*. No contextual information is available to a CAA system and so ambiguity can arise. In some situations there is no ambiguity, for example: numbers with letters, eg $3x$ interpreted as `3*x`; numbers with parentheses, $2(x-1)$ interpreted as `2*(x-1)`; back-to-back parentheses, eg $(x-1)(x+1)$ interpreted as `(x-1)*(x+1)`; a known function, eg $\cos(x)$ interpreted as `cos(x)`.

For our application students do not define their own functions with arbitrary names as they might in a CAS worksheet. Furthermore, the range of named functions used is comparatively small and all (except perhaps $\ln$) have at least three letters, ie $\sin$, $\cos$ and so on. We also suggest giving single letters the following implicit meanings, which accords closely with current common usage in mathematics learning and teaching, ie (P1). a–d, real numbers; e is the base of the natural logarithms; f–h are unknown functions; and i, j are both $\sqrt{-1}$. We assume k–n are integers, o is unused, p, q are polynomials, or functions. r–z are real, or perhaps complex, numbers.

A potentially serious problem arises with expressions such as `x(t+1)`. If the symbols x and t are given implicit meanings the system will always interpret each as a real number, and will therefore assume that this expression represents a *multiplication*. However if x is a time-dependent displacement then this string represents a function. It is not clear to us how asking the student for further clarification might work in practice. Indeed, there must be a danger that such a query would only serve to confuse: there is no ready, and generally accepted, notational distinction between the two interpretations, and attempting to choose on the basis of a verbal distinction might in itself present students (especially inexperienced ones) with a perplexing challenge. If the idea of giving symbols implicit meanings is to work, another solution to this class of difficulty needs to be found. The authors' view, after much consideration, is that *either* the idea of implicit symbol interpretations must be abandoned *or* authoring systems for CAA need to allow authors to override the default syntax to provide a context for CAA. Even if implicit symbol interpretations are abandoned the question of the correct interpretation of expressions like $x(t+1)$ still arises, and the need may still arise for default interpretations to be overridden.

These issues illustrate clearly, in the authors' opinion, that not all problems associated with the interpretation of mathematical expressions in CAA can be solved at the level of the underlying syntax. We argue that some problems can be solved only by ensuring that input systems contain the facility for clarification dialogs, or that during CAA authoring it is possible to override default interpretations, or both. Currently such features are absent.

It might also be objected that symbols should remain abstract, as is the case with existing CAS implementations. In mainstream usage certain letters *do* fulfill traditional roles.

> The advantage of selecting in our signs, those which have some resemblance to, or which from some circumstance are associated in the mind with the thing signified, has scarcely been stated with sufficient force: the fatigue, from which such an arrangement saves the reader, is very advantageous to the more complete devotion of his attention to the subject examined. [2, pg 370]

For example, Babbage suggests the use of t to denote a temporal variable. He also recommends signs where the meaning is closely associated with the shape such as $=$, $<$ and $\leq$. Contemporary thought agrees, for example. [7]

> In choosing infix symbols, there is a simple principle that really helps our ability to calculate: we should choose symmetric symbols for symmetric operators, and asymmetric symbols for asymmetric operators, and choose the reverse of an asymmetric symbol for the reverse operator. The benefit is that a lot of laws become visual: we can write an expression backwards and get an equivalent expression. For example, $x+y<z$ is equivalent to $z>y+x$. By this principle, the arithmetic symbols $+\times<>=$ are well chosen but $-$ and $\neq$ are not.

In accordance with (B11) and (B1) we propose to accept a space to signify function application. So that `sin x`, `f x` be permitted. Parentheses are then used only to indicate grouping of terms. Function application then becomes `sin (2x)`, or `f (x+1)`. We propose to make a space optional when terms are grouped in function application, to permit `sin(2x)`, `f(x+1)`. This closely corresponds to (P1), and (P2), although not with Mathematica's unique use of brackets, in InputForm.

We wish to record some specific decisions: since elementary algebra usually assumes the real domain we use $\sqrt{x}$ to refer to the positive square root when it exists, and we regard $x^{\frac{1}{2}}$ as synonymous with $\sqrt{x}$. In all CAS, inverse trig functions are denoted using `asin` or `arcsin`, so that (B5) conflicts with (P2). We propose, to agree with (B5) and allow `sin^-1(x)` for the inverse. In keeping with (B2), we interpret `sin^2(x)` as composition, perhaps with a parser warning of this interpretation. If function application binds more tightly than exponentiation then `sin x^2` will be interpreted as $(\sin\ x)^2$, and `sin(x^2)` is unambiguous. To maintain (P2), other forms for inverse operations are permitted and there is probably no harm in accepting a variety of forms including `asin`, `arcsin`, etc.

We propose to accept `e^x` as exponentiation and to retain a function `exp` to maintain (P2). Written mathematics and many hand-held calculators use $\log$ for logarithms to base 10, and uses $\ln$ for the natural logarithm, which is an irreconcilable conflict between (P1) and (P2). A radical solution is that of [4] who proposed the logarithm $\log_a(b)$ should be written as $a{\downarrow}b$ by analogy with `a^b`. This accords with (B4), and finesses the problem of a conflict with (B5): `e^-1` is the reciprocal of e, not the inverse. The syntax `a_b` as $\log_a(b)$ is compact (agreeing with (B1)) and also removes the conflict between (P1) and (P2), but such a radical departure from (P1) may be unacceptable. If not, `log` should be retained as the natural logarithm, perhaps with a warning, as should `ln`. Logarithms to the base 10 should be entered as `10_x` or `log10 x`. To be consistent with (B11), we propose to accept, `exp^-1 x`, `ln^-1 x` and `log^-1 x` for $\ln(x)$, e^x and e^x respectively, even thought this does not correspond to (P1).

White space is sometimes problematic: eg typing `< =` rather than `<=`. We cannot ignore all white space, eg function application or implied multiplication. Examples include `2 x`, `x y` (rather than a variable `xy`) or `x sin(x)`. We propose not to permit spaces within numbers, even if these aid readability. Also CAS accept scientific notation, eg `1.1e-49`. We propose taking e to be the base of the natural logarithm and so this scientific notation should be dropped. We rather feel that requiring a verbatim expression such as `1.1*10^-49` reinforces the meaning to the student, which `1.1e-49` might obscure.

The absolute value function should be written as either `abs` or using matching `|`'s when only two are present in an expression.

Another significant source of error occurs with chained inequalities such as expressions such as $-1 \leq x \leq 5$. We propose to adopt this convention. Furthermore, `<=` and `=<` should by synonymous, and white space between the symbols $=$, $<$ and $>$ should be quietly removed.

We propose that lists be entered as a comma separated list between square brackets. Sets be entered as a comma separated list between curly braces. There is no harm in allowing `set(a,b,c)` as a constructor function.

6 Conclusion

What extent should a general increased use of linear syntax affect written mathematics at an earlier stage? We have worked *from existing written elementary mathematics* towards *existing CAS syntax*. Any increase in the use of technology perhaps should affect teaching at an earlier stage. Equally we might question the design decisions of those implementing CAS and ameliorate the impact of (P2). Any change has severe implications for current CAS users, eg legacy code. History has consistently demonstrated that changes to mathematical notation are exceedingly difficult to achieve.

The decision to opt for an input mechanism with an informal syntax depends on the particular student group and their needs. For this reason we have reported in our results and discussions only the issues raised. We do not propose a fully formed "informal syntax". For students early in their studies of mathematics, such as school students or non-specialists learning mathematics at universities, such a mechanism may be entirely sufficient. Of course, there will be students who need to learn exactly the syntax of a CAS in order to communicate with it as a useful tool. It may be argued by some that *all* students need to learn a strict CAS syntax, and so should do so early. We disagree. A more liberal approach will allow students to concentrate on the mathematics, not the notation. What is the point of stressing that $x(t+1)$ could be a function to students who have yet to encounter the notion of an abstract function? There will come a point where students perceive or even demand the *need* for a more strict syntax as their maturity develops. This should be a natural maturation process, which if correctly designed, the CAA system and its choice of syntax can support.

Given the variety of teaching contexts within the United Kingdom and internationally we do not feel in a position to conclude this paper with specific proposals. While it is tempting to do so, history has demonstrated that such an attempt would be futile. While both authors have implemented much of the proposed liberal syntax of their own in the Metric and STACK CAA systems, it is the responsibility of teachers to make choices for their own students. This paper is an attempt to highlight what we consider to be the issues for further debate and discussion.

References

[1] Babbage, C., 1821. Observations on the notation employed in the calculus of functions. Transactions of the Cambridge Philosophical Society.

[2] Babbage, C., 1827. On the influence of signs in mathematical reasoning. Transactions of the Cambridge Philosophical Society II, 325–377.

[3] Babbage, C., 1830. On notations. Edinburgh Encyclopaedia 15, 394–9.

[4] Brown, M., June 1974. Some thoughts on the use of computer symbols in mathematics. The Mathematical Gazette 58 (404), 78–79.

[5] BS 6727, 1987. Specification for representation of numerical values in character strings for information interchange. British Standards Institute, BS 6727:1987.

[6] Grabmeier, J., Kaltofen, E., Weispfenning, V., 2003. Computer Algebra Handbook. Springer.

[7] Hehner, C. R., 2004. from Boolean algebra to unified algebra. The Mathematical Intelligencer 26 (2), 3–9.

[8] Hewitt, D., 1996. Mathematical fluency: the nature of practice and the role of subordination. For the learning of mathematics 16 (2), 28–35.

[9] Kirshner, D., 1989. The visual syntax of algebra. Journal for Research in Mathematics Education 20 (3), 274–287.

[10] Ramsden, P., June 2004. Fresh Questions, Free Expressions: METRICs Web-based Self-test Exercises. Maths Stats and OR Network online CAA series `http://ltsn.mathstore.ac.uk/articles/maths-caa-series/`.

[11] Sangwin, C. J., 2004. Assessing mathematics automatically using computer algebra and the internet. Teaching Mathematics and its Applications 23 (1), 1–14.

[12] Sangwin, C. J., Grove, M. J., 2006. STACK: addressing the needs of the “neglected learners”. In: Proceedings of the WebAlt Conference, Eindhoven.

[13] Strickland, N., 2002. Alice interactive mathematics. MSOR Connections 2 (1), 27–30.

[14] Wester, M., 1999. Computer Algebra Systems: a Practical Guide. Wiley.