a scientific problem in something close to the natural notation. We note that this notation is sufficent for the statement of many problems that use multivariate calculus and that this development, while expensive(from the perspective of an applications developer), is customizable and only needs to be carried out once. The next stage is to prove theorems concerning multivariate calculus satisfied by objects that use the display display forms defined here. Again while this may require extensive development it only needs to be performed once and then these results can be used in the construction of new justifications.

With these theorems proven it will be possible to use the NuPrl prover with a computer algebra system to prove the various theorems stated in [3] and hence obtain a running numerical algorithm using our environment. If such algebra systems are not considered sufficiently reliable then the relevant algorithms may be developed within the proof system. Users of an environment of the kind we are constructing have great flexibility with regard to the proof obligations they impose on themselves. At one extreme they can merely develop appropriate notations to ensure that they write their mathematics consistently. At the other extreme they can attempt to prove everything from first principles.

This work is part of an ongoing program of work the end result of which will be the demonstration of the construction of a running algorithm from the natural statement of a realistic theorem in applicable mathematics in a problem solving environment. Current work is concerned with testing the applicability of our notational mechanisms by writing all the theorems in [3] in this style and testing the notation on results in other applied mathematical domains. The development of theorems using the NuPrl system concerning differentiation, differentiability and Taylor series expansions is currently underway and will be used to justify the use of computer algerbra in proving results. Future work will be concerned with the justification of transformations for running algorithms constructed using these ideas.

5 Acknowledgements

The authors wish to thank Robert Constable for helpful comments on earlier drafts of this document and Richard Zippel for helpful discussions.

References

[1] *Implementing Mathematics with the NuPrl proof Development System* R.L. Constable, S.F. Allen, H.M. Bromley, W.R. Cleaveland, J.F. Cremer, R.W. Harper, D.J. Howe, T.B. Knoblock, N.P. Mendler, P.Panangaden, J.T. Saski, S.F. Smith. Prentice Hall(1986).

[2] *The NuPrl Proof Development System, Version 4.1 Reference Manual and User's Guide,* Cornell document, P.Jackson, 1994.

[3] *Scientific Computation, Symbolic Computation and Formal Inference* NuPrl Research Note, C.L.T. Mannion, 1993.

[4] *Mathematical notation* NuPrl Research Note, C.L.T. Mannion, S.F. Allen,1994.

[5] *Towards Integrated Systems for Symbolic Algebra and Constructive Formal Mathematics,* Cornell Tech. Report, R.L. Constable, P.B.Jackson,1993.

Expanded-Theorem 1

```
∀ n:ℕ. Set theIDXset ≡ 1..2×1..n,
           theIDXidFn
           ≡ λ<i,j>.λ<i',j'>.i=₂i'∈int∧₂j=₂j'∈int
           ∈ 𝔹id theIDXset.
       ∀ Gamma:{t:ℝ⟶}[0,1]⟶theIDXset⟶ℝ, H:((theIDXset⟶ℝ)⟶ℝ).
       H D'ble 3 times over theIDXset⟶ℝ⇒
       (∀ s:[0,1], p,q:{t:ℝ⟶}ℝn.
        (∀{t:ℝ}. Gamma{t}[s] = p{t},q{t})⇒
        ∀ r:1..n. infinitely D'ble {t:ℝ}. p{t}.r &
                  infinitely D'ble {t:ℝ}. q{t}.r &
                  ∀{t:ℝ}.
                  dq{t}.r/dt = Der{theIDXidFn,<1,r>} H[p{t},q{t}] &
                  dp{t}.r/dt = -Der{theIDXidFn,<2,r>} H[p{t},q{t}])
       ⇒
       Const t:ℝ. Path-Integral[Gamma[t]] ∈ ℝ
```

Notice that when the result is expanded we see the index identity function `theIDXidFn` , the abrreviation R∗ is expanded and all the suppressed time dependencies are made explicit. The unexpanded version is the appropriate version for further reasoning and yet, if at any point, details need clarification the full version is immediately available for reference. In fact selective expansion of any particular piece of the original theorem is possible.

At the moment the above are just notational devices. In order to give these signs any mathematical meaning inside NuPrl the notation would have to be linked with theorems concerning derivatives and differentiability, continuity, etc based on accouts of functions of a real variable, based on an account of the real numbers, based on the foundational mechanisms inside NuPrl. Doing just this is the subject of current work using the Hamiltonian example discussed in [3].

4 Conclusion

What has been achieved? Using the NuPrl display mechanisms it has been possible to write mathematics in a fairly natural way. This work is part of an ongoing project developing systems for doing real mathematics. Refinement of the Hamiltonian example will allow the demonstration of the derivation of a running algorithm from the symbolic statement of a realistic applied problem . In [3] an outline of a scientific algorithm from the recent literature was given and a rough sketch of how to obtain a running algorithm by means of a formal proof was indicated. This document has taken things further by constructing a notation mechanism for this kind of mathematics in the NuPrl system and indicating how proofs might be constructed.

In addition the beginnings of a method for using these techniques can be inferred. The construction of abstraction and display forms relevant to a particular problem provide a means for checking that the formal statement of the problem is appropriately parameterized and notationally consistent just by writing it down within the system [4]. Questions as to the truth or falsity of the statement require the development of a formal proof.

There are some other observations to be made about this work.

The notational developments carried out here demonstrate the ways in which a logical apparatus can function as "glue" to link together the various stages in computational environments. In this case the NuPrl editor and display mechanism have been used to state

```
EDIT ABS differentiable_partial_iter

f D'ble n times over LB,UB:IDX⟶ℝ
== ℕind[n](λ.True
   ;n_D'ble. λf.f D'ble over LB,UB:IDX⟶ℝ &
             ∀ i:IDX, eq:(𝔹id IDX).
             n_D'ble[Der{eq,i} f])
   [f].
```

Labels have been left in these definitions to comment the results.

3.3 Formal Statement of a Theorem in Mechanics

The theorem in section 2 refers to the constancy of an integral around a closed path in phase space. In particular as this closed path develops in time, under the action of the equations of motion, a certain integral taken along the path remains constant. This can be described notationally by relating the dynamical variables $\mathbf{p}, \mathbf{q}$ to closed path in phase space. Intuitivly the phase space can be considered to be parameterized by two variables s, t. Varying s , for fixed t, moves the dynamical variables around a closed path.Varying t, for fixed s, gives the time development of $\mathbf{p}, \mathbf{q}$. With all this in hand the theorem in section two can be stated in the following way in NuPrl

Theorem 1

```
∀ n:ℕ. Set Index set: 1..2×1..n.
       ∀ Gamma:{t:ℝ⟶}[0,1]⟶ℝ*, H:(ℝ*⟶ℝ).
       H D'ble 3 times ⇒
       (∀ s:[0,1], p,q:{t:ℝ⟶}ℝn.
        (∀{t:ℝ}. Gamma[s] = p,q)⇒
        ∀ r:1..n. infinitely D'ble {t:ℝ}. p.r &
                  infinitely D'ble {t:ℝ}. q.r &
                  ∀{t:ℝ}.
                  dq.r/dt = Der{<1,r>} H[p,q] &
                  dp.r/dt = -Der{<2,r>} H[p,q])
       ⇒
       Const t:ℝ. Path-Integral[Gamma[t]] ∈ ℝ.
```

Here q.r just denotes the r th component of $\mathbf{q}$. The initial lines indicate the structure of the dynamical variables of the problem by means of the the strucure of the index set, the type structure of the closed path `Gamma` in phase space, the type structure of the Hamiltonian and its differentiability. Subsequent lines indicate the way in which the dynamical variables `p,q` are related to `gamma`, the differentiability requirements on the dynamical variables and the equations of motion. The final line gives the consequent of the theorem.

When this result is stated in traditional mathematical notation all questions of indexing into variables is understood. Here our indexing mechanism is explicitly stated (as it must be for computational use). The above theorem could be made even closer to the conventional display by further suppression of variables and hiding the indexing mechanism inside some standard header for Hamiltonian systems if required.

This result is fairly close to the original but contains a considerable amout of implicit information. A user needing more information can see the suppressed detail by expanding the theorem (with a a mouse click) to obtain

a space with the strucure indicated by the index set. The predicate `eq` is a (curried) test for boolean identity, over the index set of the arguments to f, the purpose of which is to provide a computational means for identifying the argument position through which the partial derivative is being taken. Examination of the auxiliary definitions should hopefully make this clear. While this may seem a rather complex way to deal with the simple (from an application viewpoint) issue of indexing it is of some signifigance.

Indexing in this way allows the suppression of reference to the identity function in our main theorems. This adds another dimension to the display of mathematics in this style. If "natural" displays of mathematical objects are required it is necessary to be sensitive to the notations constructed at the early stages of a mathematical argument and their interactions with subsequent definitions.

Normally indexing is understood by users of a system (for example the mapping betweem array references and physical variables in some application). Since we wish to be able to manipulate mathematical objects with programs it is essential to be able to encode this information for computation. Note also that quite a lot is happening, mathematically, for such a simple idea.

The well formedness result refers to the differentiability of the functions with respect to each index position. Notation to capture this is defined in terms of the primitive notion of differentiability in the following way

```
f D'ble over LB,UB:IDX⟶ℝ
== ∀ i:IDX, eq:(𝔹id IDX)
   , X:(j:IDX⟶[LB[j],UB[j]]).
   D'ble x:[LB[i],UB[i]]. f[X{eq[i]:= λ.x}].
```

In the discussion of Hamiltonian systems it is necessary to consider functions that have finite and infinite degrees of differentiability. This occurs for each variable over some range and in the case of multivariate functions for each argument position. The following definitions capture these ideas notationally

```
infinitely D'ble x:ℝ. v(x)
== ∀n:ℕ.n times D'ble x:ℝ. v(x)

EDIT ABS differentiable_R_iter

n times D'ble x:ℝ. v(x)
== ∀ lb,ub:ℝ.
   lb≤ub⇒n times D'ble x:[lb,ub]. v(x)

EDIT ABS differentiable_iter

n times D'ble x:[lb,ub]. v(x)
== ℕind[n](λ.True
   ;n_D'ble. λf.(D'ble x:[lb,ub]. f[x]) &
             n_D'ble[λx.df[x]/dx])
   [λx.v(x)]

EDIT ABS differentiable_partial_iter_R

f D'ble n times over IDX⟶ℝ
== ∀ LB,UB:(IDX⟶ℝ).
   (∀i:IDX.LB.i≤UB.i)⇒
   f D'ble n times over LB,UB:IDX⟶ℝ
```

In the Hamiltonian example it is necessary to consider two kinds of differentiation; partial and total. Selecting the parameter $hint in the primitive definition as total or partial displays the apppropiate derivative. The associated theorem is a well formedness result that needs to be satisfied. Note that with notation for both kinds of differentiation the equations of motion in the theorem in section 2 can be stated in a natural way within the system. In the rest of this section we show how to use the NuPrl editor to build notations that allow the statement of the theorem concerning Hamiltonian dynamics given in section 2.

Referring back to that theorem we note that the idea of differentiability of a function with respect to a variable occurs. The notation in the theorem indicates that the Hamiltonian is three times differentiable with respect to each of the generalized momentum and coordinate variables and that each of these variables in infinitely differentiable with respect to the time. The basic idea of differentiability of a function over an interval is taken as primitive and denoted in the following way

```
PRIMITIVE: D'ble x:[lb,ub]. v(x)
{← This is supposed to mean that v(x) is
   differentiable on the interval [lb,ub]
   }.
```

Again this will be supported by appropriate theorems as this work develops.

In the Hamiltonian case it is necessary to consider differentiation and differentiability with respect to every component of a vector variables. We denote partial differentiation with respect to a variable at an index position in the following way

```
EDIT ABS deriv_partial

Der{eq,i} f == λX.∂f[X{eq[i]:= λ.x}]/∂x|x=X[i]
{←
   THM: ∀ IDX:Type, LB,UB:(IDX→ℝ)
        , f:((j:IDX→[LB.j,UB.j])→ℝ).
        f D'ble over LB,UB:IDX→ℝ⇒
        ∀ eq:(𝔹id IDX), i:IDX.
        Der{eq,i} f∈(j:IDX→[LB.j,UB.j])→ℝ
   }.
```

The definition of `Der` uses the following auxiliary definitions.

```
EDIT ABS boolid

𝔹id A == {f:(A→A→𝔹)|∀ x,y:A. f[x;y] = true₂⇐⇒x = y}

EDIT ABS CALC_alter_fn

f{P:= NewVals}
== λx.if (P[x])→ NewVals[x] else f[x] fi.
```

The theorem following the definition of `Der` is a well formedness goal that needs to be satisfied by our definitions. The notational mechanism for `Der` requires further explanation. In the case of Hamiltonian dynamics the Hamiltonian depends on $2n$ independent variables divided into two collections of momentum and coordinate variables. We wish to respect this distinction in the functional dependence and provide a mechanism that can be extended to more than two collections of independent variables if needed. Notice that that `Der{eq,i}f` needs to be applied to something . Intuitively when `Der{eq,i}f` is used `f` is a map from

In NuPrl, using the current display mechanism, this equation can be written in the following way

```
Implicit x,y,z.
∂p/∂z = mu (∂(∂w/∂x)/∂x + ∂(∂w/∂y)/∂y).
```

Notice the form of the equation as it is presented here. At this stage p depends on z and w depends on x, y. In the above these variables are declared as implicit which permits them to elided from the display form. At some point in the development of a proof or program it might be necessary to indicate this dependence explicitly. Simply by removing the context declaring x, y, z to be implicit leads to the following form for the equation

```
∂p{z}/∂z = mu (∂(∂w{x}{y}/∂x)/∂x + ∂(∂w{x}{y}/∂y)/∂y).
```

The term w{x}{y} just indicates that suppression of arguments is permitted in appropriate contexts.

Contrast the three ways in which this equation has appeared in this note. It's first appearance was produced by means of some Latex commands. These commands have no mathematical significance and only allow the form of the equations to be produced. At best the Mathematica form of the equations is essentially an expression in a functional progamming language that cannot be further simplified by the evaluation rules of the language. To solve this in Mathematica a solution algorithm for this particular partial differential equation would have to be written. The NuPrl version has a rather different sense. It is the display form of a term in the NuPrl system. That term is an entity about which further results can be stated and theorems proved within the same system that allows the display, and moreover a term that preserves its display and meaning in the course of formal manipulation within the sytem. In principle an engineer or scientist can work with something very close to conventional notation and prove results and develop programs.

3.2 Notation and Particle Mechanics

The display form for differentiation that we have provided is the following

```
∂w{x}{y}/∂x
```

is the special form of

```
∂w{x}{y}/∂x|x=E
```

in which the variable x itself is used for E.

The notational definition as it appears in NuPrl is of the form shown below.

```
PRIMITIVE: Simple Derivative: $hint  da(x)/dx@x
{← The parameter is just used to suggest how it's
   used; mainly, total or partial.

   THM: ∀ a,b:ℝ, f:([a,b]→ℝ).
        (D'ble x:[a,b]. f[x])⇒
        ∀x:[a,b].(df[x]/dx)∈ℝ
   }.
```

puter algebra framework. Such systems were designed for the manipulation of the symbolic representation of mathematical objects (but not for proof with such objects). Mathematica, for instance, seems to provide some of what is required. Consider for instance the expression $f(x, y, z, t)$ where x, y, z depend on the parameter t. Using Mathematica to perform differentiation with respect to t and $x[t]$ yields the standard results in a form fairly close to that found in an engineering or physics textbook and is acceptable for realistic use. The explicit dependence on t is rather clumsy and it should be possible to suppress it when neccessary. Using these kind of ideas it is easy to write Mathematica programs to solve and plot the results for simple Hamiltonian systems.

In a problem solving environment that is the long term goal of this work, it would be desirable to use something like the Mathematica notations for differentiation and functional dependence as the display forms for terms that can be manipulated in proofs.

In the next section we discuss how the various mathematical objects mentioned in the theorem discussed earlier can be rendered into NuPrl display forms and the theorem stated in that system. A subsequent paper will link these display forms to the formal representation of ordinary and partial derivatives in a proof system allowing a formal proof of these results.

3 A notation for analysis

In this section the issue of providing a notation that can deal with the notational issues arising in Hamiltonian dynamics is discussed. The specific problems that we wish to deal with are

- The provision of a notational mechanism for the description of total and partial derivatives as terms in a formal system of mathematics.
- The application of this notation in the multivariate case associated with the Hamiltonian mechanics. Notice that in this case the independent variables have a particular structure that has to be dealt with.
- The statement of differentiability requirements on the functions occuring in the problem.

The notational devices that have been developed are fairly general and allow us to consider cases other than particle mechanics.

3.1 Notation and Partial differential Equations

To illustrate the notation for partial differentiation, in a simple example consider the following differential equation as it would be written in ordinary mathematical notation

$$\frac{\partial p}{\partial z} = \mu \left(\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} \right).$$

It describes the laminar, viscous flow in a rectangular duct. The axial velocity $w(x, y)$ is determined by the pressure gradient and the viscosity μ. Note that it is not usual to indicate the explicit functional dependence in the differential equation.

Although it not particulary useful we can write the above differential equation in Mathematica in the following way

$$D[p[z], z] == mu(D[w[x, y], \{x, 2\}] + D[w[x, y], \{y, 2\}]).$$

The fact that the explicit time dependence of the coordinates and momenta is not explicity indicated is just a small part of the problem that occurs throughout applications. The conventional practice is to infer from context many functional dependencies (e.g. in plasma physics that the distribution function depends on particle positions and velocities that are themselves functions of time, in the above case that $\mathbf{p}, \mathbf{q}$ are to be regarded as functions of the time). It appears that the quantities $\mathbf{p}, \mathbf{q}$ are to be regarded in different ways in different contexts.

The differential equations that occur in the theorem use two distinct notions of differentiation. It should be noted that differentiation symbols are complex signs that need to handled carefully. For instance in the expression $\frac{df(x)}{dx}$ it would not be sensible to substitute 3 for x in numerator and denominator. For many real applications notation is extended to handle more complex examples with layers of functional dependencies, iterated differentiations and application of parameterized operators. Computational support intended to preserve mathematical meaning and allow high level proof requires coherent explanations of the notational mechanisms rather than a succession of ad-hoc devices.

2.2 Computational Approach

What are the possibilities for rewriting the previous theorem in a more computational style?

One way to proceed is to to write everything in a programming language notation. There are several difficulties with this approach. Consider the example of the Hamiltonian above. An experienced (even a tyro) dynamicist immediately understands that the Hamiltonian is a map from the space of coordinates and tangent vectors to R. Sometimes it is convenient to use this interpretation whilst on other occasions it is desirable to understand the Hamiltonian as a map from paths in phase space to R. The "slurring" of meaning is essential to practice. This issue of parameterised paths in state spaces of systems is basic to scientific and engineering practice yet seems to be difficult to handle in a natural way in the context of programming languages. One way to deal with this is just to define several different Hamiltonians. This would just be unacceptable for practical use. Zippel has suggested that the problem can be dealt with by regarding entities such as the Hamiltonian as generic functions in an object oriented programming sense. This seems to be a more agreeable solution but raises the issue of typing for object languages.

Rather than discuss this approach here we will consider what can be done within the type theory paradigm. Note also that in order to develop an enviroment for scientific programming it will be neccessary to be able to develop symbolic mathematical representations of mathematical objects at general positions in various spaces. Providing the capability to carry out mathematical operations on such objects within programming languages is a challenge for the future suggested by this kind of work (i.e. making general mathematical objects first class objects in a programming language).

Additional issues arise in relation to the simplification of expressions. It seems much harder for human readers to rearrange complicated programs than to see the simplifications in mathematical formulae. (Imagine reasoning about the terms of a complicated perturbation expansion written in a programming language.) The work outlined here is concerned with human directed manipulation of symbolic expressions. Ease of simplification and manipulation would seem to require the use display forms that can be used in a particular context overlaying programming notations.

An alternative approach is to consider what kinds of support are available within the com-

in an informal way the types of quantities e.g p for generalised momentum) to the more difficult (not stating explicitly the functional dependencies of various quantities). In order to use theorem proving tools on mathematical and engineering systems having descriptions of this kind it is necessary to be able to indicate free and bound variables in the statement of the theorem and all dependencies of functions. On the other hand a scientific user would find such a system intolerable to use. Practical reasoning makes essential use of computationlly ambiguous notation.

In order to deal with this problem the NuPrl system provides display forms that can hide a lot of the detail required by the theorem prover and yet preserve their form in intermediate steps. Clearly it will be neccessary to give the full details of all notations used at least once in a discussion.

2.1 Ambiguities

In this section the ambiguities associated with the theorems stated in [3] will be indicated and some suggestions for dealing with them outlined. This is a general problem for anyone interested in providing computer support for mathematics. How can a notation be provided that is acceptable to users and yet gives enough information for machine support?

Consider the following theorem from [3]

Theorem 1 $*$
Given a Hamiltonian function

$$H(\mathbf{p},\mathbf{q}) \in \mathrm{C}^3(\mathrm{R}^n \times \mathrm{R}^n \longrightarrow \mathrm{R}),$$
$$\mathbf{p},\mathbf{q} \in \mathrm{C}^\infty(\mathrm{R} \longrightarrow \mathrm{R}^n)$$

such that

$$\begin{aligned} \frac{dq_r}{dt} &= \frac{\partial H}{\partial p_r}, \\ \frac{dp_r}{dt} &= -\frac{\partial H}{\partial q_r} \end{aligned}$$

then

$$\oint_\Gamma \mathbf{p} \cdot \mathrm{d}\mathbf{q}$$

is an invariant of the motion where Γ *is a closed path in phase space.*

What are the conceptual and notational difficulties that would be faced using an algebra system and a theorem proving tool to demonstrate this result?

A user familiar with the area coming across such a theorem in a text book would immediately recognise it as a theorem in Hamiltonian dynamics and that the quantities of interest are the time evolution of the vectors of dynamical variables, the coordinates and momenta denoted by $\mathbf{p},\mathbf{q}$. The specific variable names indicate the Hamiltonian , H, and the coordinates and momenta. The differentiability conditions on the Hamiltonian H with repect to the phase space variables $\mathbf{p},\mathbf{q}$ and their differentiability conditions with respect to the time parameter are indicated. The dependency of these functions on the time is not explicitly indicated except in the type definition which would not normally be explicitly stated. The differential equations use two distinct notions of differentiation. Finally the use of the decorated line integrals needs to be explained.

years of practice and experience, suggests and guides the development of arguments in new applications. A simple example from Hamiltonian dynamics concerns the use of the variables p, q. These variables are used in Hamiltonian dynamics to stand for position and momentum. Their implicit dependence on time is immediately understood by workers in the field.

The desire to provide computer support for the construction of mathematical arguments and the development of programs requires fresh consideration of mathematical notations. If such systems are to be useful to working mathematicians and engineers, a notation that closely matches current working practice has to be provided. On the other hand the use of machines to support this activity using the tools of type theory, computer algebra and theorem proving means that the notation used must be completely unambiguous.

One way to deal with this difficulty is to have a precise working notation for the development environment and to have reasonably conventional display forms that can be expanded or contracted as required by the user. This ability to hide or display information, in context, during the course of an argument is something that human reasoners do easily. The NuPrl system [5] provides some tools to support the construction and editing of display forms that can be used in this flexible manner. By developing display forms for specific problem domains it is possible to provide an environment for writing mathematics that supports elementary checking that mathematics has, at least, been written correctly[4].

The next section discusses some of the computational difficulties associated with the normal working notation of physical and engineering mathematics using the theorems in [3] for illustration. The deficiencies of some of the obvious computational approaches are also indicated. Necessarily kinematics (the description of motion) precedes dynamics. The mathematical description of the time evolution of discrete and continuum systems is so basic an aspect of mathematical modeling that it is often overlooked. An environment intended to support the development of scientific and engineering applications programs from symbolic descriptions should provide natural, computationally unambiguous descriptions of the motion of particles and continua. Subsequent mathematical arguments are crucially based on these descriptions.

In the subsequent section the development of a notation for the discussion of ordinary and partial derivatives using the NuPrl system is discussed. These are illustrated by considering a partial differential equation occurring in fluid dynamics. Using these display forms it is shown how one of the theorems occuring in [3] can be displayed in the current NuPrl proof system. In a subsequent paper the display forms will be linked to formal terms in a proof systems allowing the derivation of formal proofs in dynamics. The whole point of the display forms is that they will be preserved in the course of the proof and constitute the working notation of a user of the system. The discussion of ordinary and partial derivatives that is given here will be useful, more generally, for the discussion of the kinematics of continua. The final section contains some conclusions and indications of current work.

2 Notational Ambiguities

In a previous paper [3] an indication of one way in which a realistic scientific algorithm could be developed was outlined. It indicated how a computer algebra system (such as Weyl) and a type system (such as NuPrl) could be used to produce an algorithm from a formal argument. The theorems stated used the normal notation of mathematical physics. The practical use of mathematics in physics and engineering makes use of various conventions that are specific to the problem domain. These vary from the trivial (using specific variable names to indicate

A Notation for Computer Aided Mathematics

C.L.T. Mannion* S.F. Allen†

January 20, 1995

Abstract

The NuPrl4 term structure and editor display mechanism are used to provide unambiguous notations for use in the development of computer supported mathematical arguments. These notations are used to provide a natural statement of a theorem in Hamiltonian dynamics, anchored in a computationally unambiguous representation, that can be made explicit if required.

1 Introduction

This note uses the NuPrl [1, 2] term structure and editor display mechanism to provide an unambiguous notation that can be used in the computer assisted development of arguments in mathematics. The example used to demonstrate this application is a theorem in Hamiltonian dynamics. These results are part of a larger project that is developing environments for scientific computation that provide support for the construction of working programs by transformation from their natural mathematical statement. The common justification for the development of such problem solving environments is that they will raise the semantic level at which program construction occurs and hence improve the programming process. Here we argue that a prerequisite for raising the semantic level for scientific and engineering applications is raising the notational level.

From a computational perspective the typical working notations of mathematics are ambiguous in that the arguments to functions are often implicit, free and bound variables are not explicitly indicated, scoping of definitions and abstractions is elastic, etc. For instance in the mathematical discussion of the motion of particles, the position of a particle is usually described by coordinates that are parameterized by time such as $x(t)$. The time development the particle is then described by some equations involving the variable x with the time dependence suppressed.

When working in a high level computational environment, how should such variables be regarded? Typically this kind of suppression of information occurs in the discussion of complicated physical situations in many dimensions with many different parameterizations (as can be easily verified by perusal of standard texts, lecture notes and research papers in various subject domains). Such notations are, however, tremendously useful and are the working notations of science and engineering. The working notation, after typically many

*partially supported by NSF grant CCR-9108-062, ONR grant N00014-92-J-1989 and Cornell Theory Center

†supported by ONR grant N00014-91-J-4123