CPoint's Mathematical User Interface

Andrea Kohlhase

School of Computer Science,
Carnegie Mellon University,
ako@cs.cmu.edu

Abstract. CPoint is a semantic, invasive editor for Microsoft PowerPoint. It enables the user to distinguish between form and content in a document by providing a user interface to the semantic XML data format OMDoc. Lately, CPoint introduced a mathematical user interface, which fully integrates mathematical symbols into PowerPoint presentations based on the semantics of the underlying objects rather than simply generating appropriate ink marks. Here, CPoint has to deal with two contrasting requirements: user-friendly creation and presentation of symbol objects (as building blocks of mathematical formula) in PowerPoint and their conversion into formal OpenMath expressions in the to be generated OMDoc document. Pragmatically, CPoint makes use of two already existent, open-source LaTex converters: TeXPoint and LaTeXML.

1 Introduction

CPoint is an invasive editor for *C*ontent in Microsoft Power*Point* (PPT) [KK04]. Its goal is to provide a PPT author with an interface to explicitly store seman-

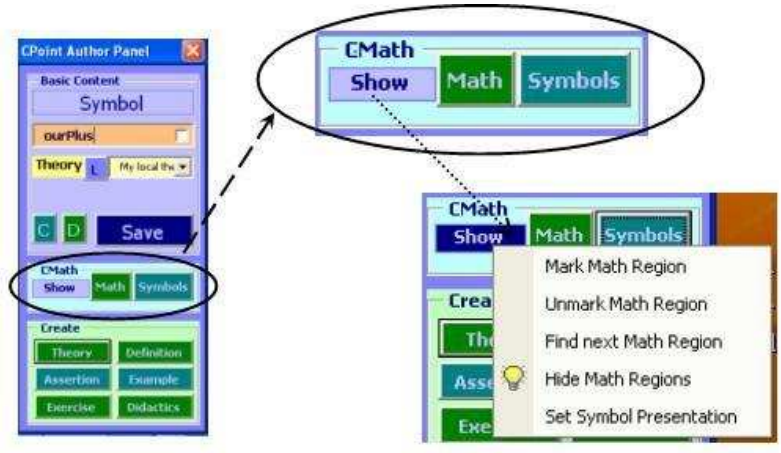

Fig. 1. The CPointAuthor User Interface Including CPoint's MUI

tic information (geared towards the XML data format OMDoc [Koh04b]) in the PPT slide show itself without destroying the presentational aspects of the PPT document. **C**POINT is written in Visual Basic for Applications (VBA) and is implemented as a PPT add-in, that makes its functionality available through a tool bar in the PPT menu where it is at an author's disposal whenever the PPT editor is running ([Koh04a], first release in 2000). PPT objects like text boxes or images are categorized as "Theory", "Symbol", "Example", "Motivation", etc. and additional semantic information can be supplemented. **C**POINT is an open-source application and can be obtained at `http://www.cs.cmu.edu/~ccaps`.

In this paper, we want to describe **C**POINT's *mathematical* user interface (MUI) for math symbols (Figure 1). As the term 'symbol' is somewhat overloaded, we distinguish between symbol objects (PPT objects that are categorized as symbol and carry their meaning), symbol occurrences (their usage, i.e. positions in text), and symbol presentations (their look, for instance the characters to represent them). Symbol objects are typically abstract, invisible objects, that a user can create from within a definition interface. Figure 2 presents **C**POINT's symbol declaration form.

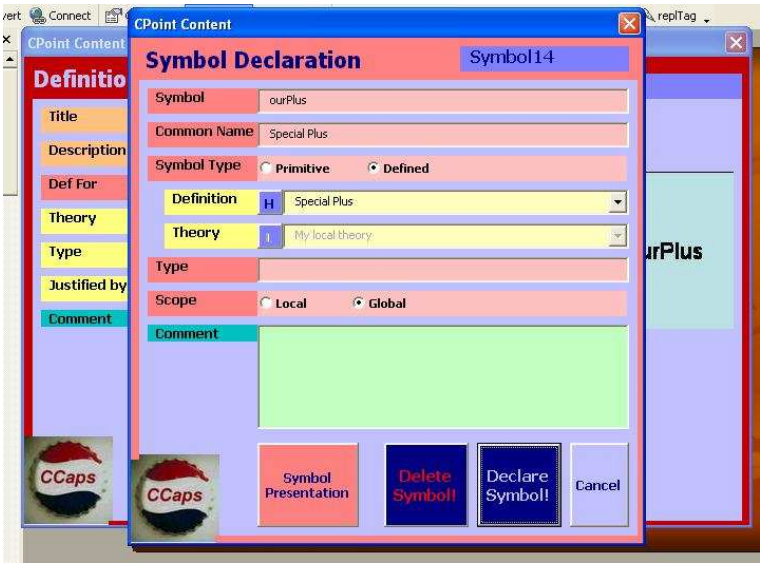

Fig. 2. Declaration of the Symbol Object `ourPlus`

CPOINT's MUI is concerned with two contrasting requirements: *user-friendly* creation and presentation of symbols (as building blocks of mathematical formula) in PPT and their conversion into the *formal* OPENMATH format [Cap03].

Unfortunately, the term "user-friendly" refers to several different kinds of users, so we want to specify our usage of the term. First, there are users, who (ardently) prefer typing to clicking or vice versa. This differentiation is basically

a matter of taste and can be handled in parallel via different, coexistent interfaces (graphic and command line). For now, we restrict ourselves to a command interface. In contrast, the discrimination into power users (expecting first and foremost power) and casual users (expecting convenience for the task at hand and immediate self-explanation) requires almost mutually exclusive interfaces. Here, we decided to orient **C**PoInt's MUI towards the power user group, since handling symbols is of a complex, dynamic nature. More specifically, symbol presentations have not only to be concise and at the same time distinctive — suggesting the usage of exceptional characters with possibly uncommon locations e.g. $\int_{\alpha}^{\beta}$, but they are also used very individually. Moreover, the underlying semantics have to be captured as well. From the power user's point of view, the flexibility and capability of LaTex [Lam94] for handling special glyphs and two-dimensional layouts can be hardly topped. Therefore, the LaTex interface yields our requirement of user-friendliness, hence **C**PoInt's MUI is modeled on it. Fortunately, the idea to build a powerful LaTex interface in PPT is not new: it is realized in the TeXPoint application as open-source software.

Using a LaTex interface has another advantage: LaTex code allows rigid formalism and extensions. Thus, it can be transformed into XML based formats like OpenMath, precisely which is implemented within the (open-source) LaTeXML program.

Pragmatically, **C**PoInt's MUI makes use of these already existent LaTex converters:

- by adapting and building upon TeXPoint's way with symbols and
- by providing special LaTex inlays in the respective generated OMDoc documents to be transformed by LaTeXML into OpenMath expressions.

1.1 TeXPoint: LaTex in PPT

TeXPoint [Nec] is an open-source PPT add-in that supports LaTex input in PPT. It was written by George Necula. It is widely accepted in campus communities as a math user interface in PPT.

TeXPoint distinguishes two working modes. In *inline mode* the user can type simple LaTex commands inside normal text like "Let `\alpha`, `\beta` be real numbers." where the LaTex macros are translated (i.e. looked up in a translation table) into the respective PPT characters α and β upon user request. In this mode, the user keeps PPT's facilities for optimizing the text's presentation. In contrast, in *display mode* all presentational aspects have to be dealt with in LaTex. In particular, the user creates a LaTex expression in TeXPoint's special LaTex editor (from within the PPT application), which is formatted by LaTex into a bitmap and displayed in the PPT document. The LaTex code is stored with the bitmap, so that the code can be subsequently manipulated e.g. in case of a simple typo. Furthermore — from **C**PoInt's point of view — the stored LaTex code is additional markup and can be output into OMDoc.

TeXPoint's strength consists in its input capabilities for mathematics (using well-known and powerful LaTex features) in the presentation-oriented PPT

environment. Nevertheless, there are two (in semantic respect major) drawbacks in inline mode: TEXPOINT is not extensible, i.e. LATEX macros cannot be created by the user, and it doesn't store the used LATEX code, so the markup symbol information is lost.

1.2 LATEXML: LATEX into XML-Based Formats

LATEXML [Mil] is an open-source Perl program that transforms LATEX documents into XML documents. In particular, LATEX math expressions are converted into OPENMATH code (presuming that respective constructors exist). It was written by Bruce Miller.

LATEXML's extension to being able to transform LATEX code inside of XML documents into XML expressions is implemented at the moment in the KWARC project at International University Bremen, Germany [Koh04c,Koh04d].

2 CPOINT's Math User Interface

In a nutshell, **C**POINT's math user interface fully integrates math symbols into PPT presentations by providing symbol presentations based on their underlying semantics and a satisfying symbol input and formal symbol output system (by supporting the input, storage, and output of LATEX code for symbol occurrences).

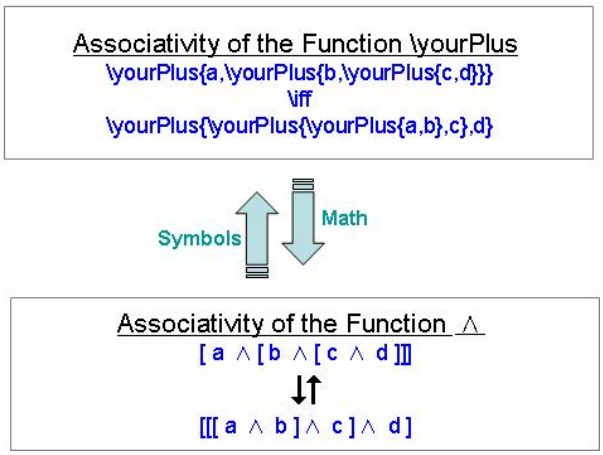

Fig. 3. **C**POINT's MUI in Action: Recursive Use of the Symbol Macro `yourPlus`

To address **C**POINT's MUI requirement of "user-friendly" creation and presentation of symbols, **C**POINT builds on TEXPOINT's LATEX input facilities,

but it extends its inline mode by addressing its shortcomings, particularly the inextensibility of macros (see "Symbol Macros" in 2.1) and the loss of markup information (see "Saving a Symbol's Occurrence in Inline Text" in 2.1) in inline mode.

Having the LaTeX markup available empowers **C**Point to output it in a converted OMDoc document. With the usage of a slight variation of LaTeXML, the user is able to convert the LaTeX expressions in the (XML) OMDoc document into OpenMath expressions (see 1.2).

Symbol macro management is supported, but on a rather basic level (see 2.2).

2.1 Extension of TeXPoint

On the one hand, **C**Point generalizes TeXPoint's handling of LaTeX macros (for symbols) to symbol macros, that are a special interpretation of **C**Point's symbols and their presentations. On the other hand, it extends TeXPoint's LaTeX storage from display mode to inline mode, so that this knowledge is not only available in the OMDoc conversion process but also in the user's authoring process of the PPT document.

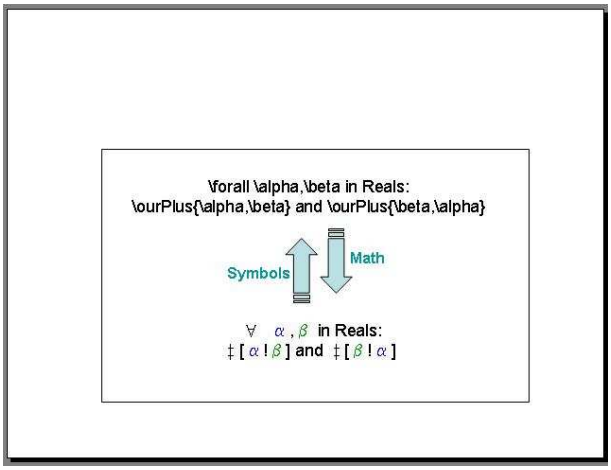

Fig. 4. CPoint's MUI in Action (Hide Mode): Symbol Macro `ourPlus`

Technically, **C**Point implements the additional functionality in its modules (e.g. **C**PointAuthor) calling TeXPoint functions whenever sensible. With respect to the MUI, **C**Point can be considered an add-on to TeXPoint. Unfortunately, an integration of the two wasn't possible for time reasons.

Symbol Macros **C**Point offers an user interface for the presentation of a previously defined symbol, where the presentations can be assigned in several

formats. Figure 5 and Figure 9 show the symbol presentations for the (fictive) symbols `ourPlus` and `yourPlus`. The former determines the PPT character "‡" as **symbol** PPT **presentation**, i.e. the character with character code 122 and the respective defining PPT character properties (where the definitional quality depends on the mark in the adjoining check-box) is the format of any occurrence of the symbol `ourPlus`. In contrast, the symbol presentation for `yourPlus` is given by a **symbol** LaTeX **presentation** with the LaTeX expression "`\and`" (Figure 9). The symbol LaTeX presentation "`\ddagger`" for the symbol `ourPlus` is not used for its symbol occurrence, but it is output into the generated OMDoc document.

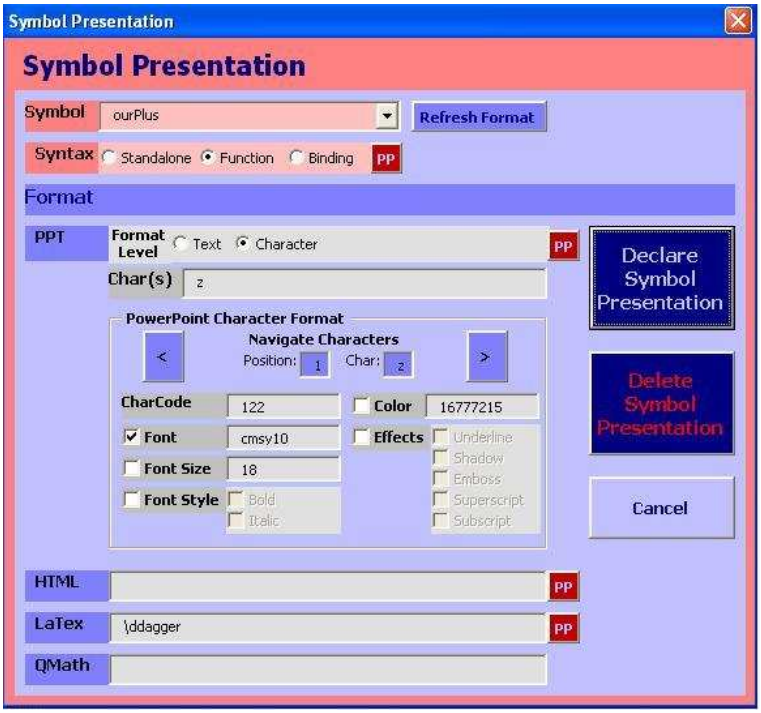

Fig. 5. Symbol Presentation for the Symbol `ourPlus`

As `ourPlus` is determined to be a function, there are several symbol presentation properties to assign like fixity, brackets, and the separator (see Figure 6). These properties (available for functions and bindings) depend on each format and therefore, they can be set via the "PP" button for each format separately. For `ourPlus` there is a default for any format set. In particular, the function `ourPlus` has a prefix notation with square brackets and "!" as a separator (resulting in ‡ [a ! b] for arguments a,b).

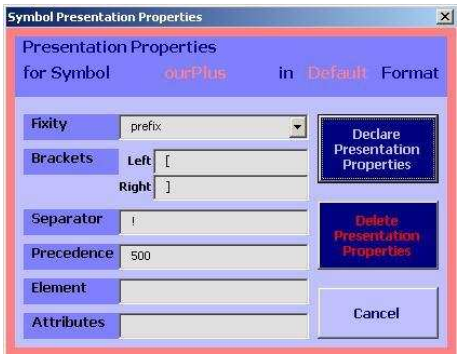

Fig. 6. Symbol Presentation Properties for the Function `ourPlus`

By enabling the user to define an individual presentation for each annotated symbol, **C**PPoint can interpret the symbol's name as a macro name, giving rise to a **symbol macro**. The input of a backslash followed by a symbol's name is treated analogously to a LaTex command in TEXPoint's inline mode, i.e. on demand (see Figure 1) such a symbol macro command is replaced by its previously defined presentation. TEXPoint's facilities are offered as a fallback option if no individual symbol presentation exists. In particular, when using **C**PPoint's symbol macro mechanism, a user accustomed to LaTex or TEXPoint can input math into a PPT text as usual, he does not have to do overhead work by annotating symbols and their presentations unless he wants to.

In Figure 4 we see an example for the usage of the symbol macro for `ourPlus` and LaTex commands like `\forall`. Moreover, we overwrote the original LaTex translation for α and β by introducing new symbols `alpha` and `beta` together with a symbol PPT presentation fixing the characters and the color.

If the user just determines a symbol LaTex presentation for a certain symbol, then this replaces the symbol command and is in turn interpreted by TEXPoint, see for instance `yourPlus`' symbol presentation in Figure 9 and LaTex presentation properties in Figure 10 resulting for instance in the associativity statement of `yourPlus` in Figure 3. For functions with an associative fixity like `yourPlus` there maybe n arguments, each of which can be simple text, LaTex commands, or symbol macro commands themselves. The PPT input of the character "“" directly behind a symbol macro call implies the beginning of an argument list, the "," is fixed as an argument separator and "" ends the list.

Saving a Symbol's Occurrence in Inline Text The main difficulty in saving symbol macro commands in inline text is obvious: there may be more than one symbol occurrence, so the exact position of the symbol macro command has to be remembered as well as the command itself. **C**PPoint provides the concept of a **math region** that marks a selected text area by setting math brackets which are (almost) invisible in the final PPT talk. By adding a math region id, this concept enables us to align math regions with stored commands, i.e. to save the

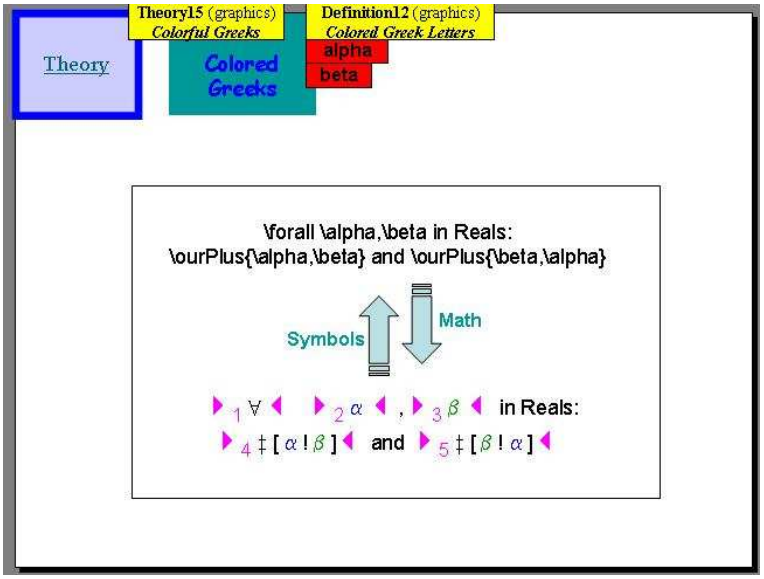

Fig. 7. CPoint's MUI in Action (Visualize Mode): `ourPlus`

position of each symbol's occurrence and the symbol macro command with the respective PPT object. To actually see hidden objects like math regions, we can use CPoint's "Visualize Mode": Figure 4 and Figure 7 show a slide view in Hide Mode and in Visualize Mode.

Naturally, the position information implies an object's **Math and Symbol Macro View**. In the former, the symbol macro commands (marked by a backslash) are textually replaced by the most exact symbol presentation (ranging from a symbol PPT presentation over a symbol LaTeX presentation to a TeXPoint translation) and a framing math region. In the latter, symbol presentations and the accompanying math regions are replaced by the respective symbol macro commands and may get corrected by the user. The "Math" button executes the symbol commands (resulting in math presentation), whereas the "Symbols" button replaces the symbol presentations with the symbol macro commands (see Figure 1 for locating the buttons and Figure 3 to envision their actions).

2.2 Symbol Macro Management

User-friendliness of a MUI requires to minimize the user's actions, especially redundant ones. So far, we have described CPoint's math user interface for exactly one PPT document. As the definition of a symbol presentation is local to that document, the user can't reuse his work in another PPT document which is clearly unfavorable. Therefore, CPoint provides a symbol macro management

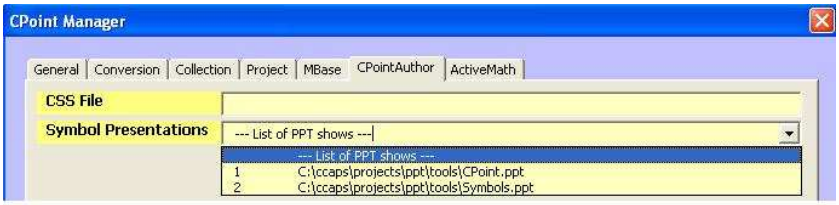

Fig. 8. CPoint Symbol Macro Management

tool in the CPoint manager[1]. Here, the user can specify certain PPT shows (sometimes ambiguously called symbol presentations), whose symbols and their presentations are read in when CPointAuthor is started (see Figure 8). Subsequently, they are available in the MUI. Unfortunately, the dependency on the local setting is still an unsolved problem.

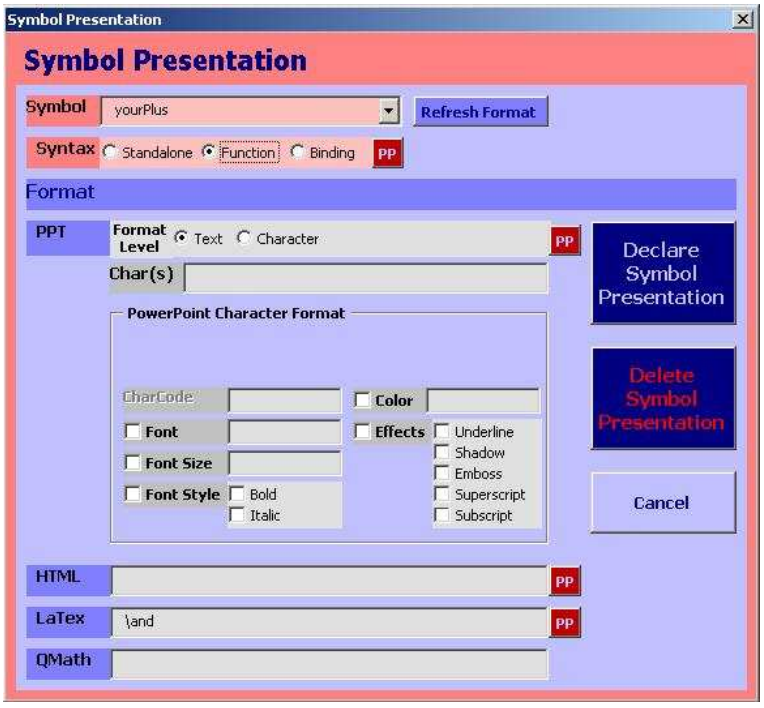

Fig. 9. Symbol Presentation for the Symbol `yourPlus`

[1] With the CPoint manager global parameters can be set and managed. They are stored in a local CPoint initialization file.

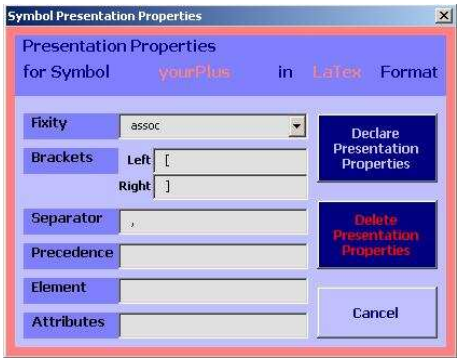

Fig. 10. Symbol Presentation Properties for the Function `yourPlus`

3 Conclusion

Combining the features of the open-source software packages **C**PoINT, LaTeXML, and TEXPoINT, math formulas (built on symbols) can be created and updated in LaTeX fashion inside Microsoft PowerPoint (**C**PoINT, TEXPoINT) and converted into respective LaTeX code in an OMDoc file (**C**PoINT), which in turn gets translated into OpenMath expressions (LaTeXML).

We expect **C**PoINT's acceptance in the TEXPoINT user community, as it extends TEXPoINT's inline mode in several useful ways (and doesn't hinder its usage at all). It supplements real added-value for a PPT content creator [KK04], as math symbols are not mere ink marks in the final talk, but are semantically specified as symbol occurrence. Furthermore, these "ink marks" can be set to be non-standard characters with specified properties (like color). In particular, it is possible to overwrite and/or extend TEXPoINT's choice of characters and define new symbol macros, manipulated in turn in TEXPoINT's (inline mode) command and (display mode) update fashion.

The advantage for an OMDoc author is more than obvious, since writing math formulas can be done now in PPT with such a structure, that OpenMath expressions can be deducted.

We can think of several improvements of **C**PoINT's math user interface. In particular, almost all the information for direct OpenMath expression creation (with arguments) exists in the **C**PoINT data, this could be directly exploited. Furthermore, the symbol macro management can be greatly improved, e.g. a special symbol search interface. As with content objects in general, the navigation of math objects is lacking the necessary finesse, therefore more sophisticated ways of navigation have to be invented and implemented. Finally, the input of characters in the Symbol Presentation Properties Form (like brackets and separators) could be extended from mere ASCII input to PPT character input (like the one for the symbol presentation characters themselves).

Acknowledgments The development of the **C**PoINT system has been funded by the National Science Foundation under grant CCF-0113919 and a grant from

Carnegie Mellon's Office for Technology in Education in the context of the CCaps project [CC00]. The work presented here has benefitted from Michael Kohlhase's visions and his cooperation was highly appreciated. The author would also like to thank the anonymous referees for valuable remarks, hints, and reusable formulations.

References

[CC00] CCaps (*Course Capsules*). Carnegie Mellon University, 2000. Home Page at `http://www.cs.cmu.edu/~ccaps`.

[Cap03] *The OpenMath standard, version 2.0.* Technical report, 2003. The OpenMath Society, `http://www.openmath.org/standard/om20`.

[KK04] Andrea Kohlhase and Michael Kohlhase. *CPoint: Dissolving the Author's Dilemma.* In Andrea Asperti, Grzegorz Bancerek, and Andrej Trybulec, editors, Mathematical Knowledge Management, MKM'04, number 3119 in LNAI. Springer Verlag, 2004.

[Koh04a] Andrea Kohlhase. **C**Point *Documentation.* Carnegie Mellon University, 2004. Technical Manual `http://www.faculty.iu-bremen.de/mkohlhase/kwarc/software/CPoint.html`.

[Koh04b] Michael Kohlhase. OMDoc *An Open Markup Format for Mathematical Documents (Version 1.2).* 2004. Manuscript, `http://www.mathweb.org/omdoc/omdoc1.2.ps`.

[Koh04c] Michael Kohlhase. *The KWARC Project.* International University Bremen, 2004. Program Home Page at `http://www.faculty.iu-bremen.de/mkohlhase/kwarc/index.htm`.

[Koh04d] Michael Kohlhase. *Semantic Markup for LaTeX* 2004. Workshop Mathematical User-Interfaces (MathUI) at MKM04.

[Lam94] Leslie Lamport. *LaTeX: A Document Preparation System, 2/e.* 1994. Addison Wesley.

[Mil] Bruce Miller. *LaTeXML: A* LaTex *to* XML *Converter.* Web Manual at `http://dlmf.nist.gov/LaTeXML/`.

[Nec] George Necula. TeXPoint. Program Home Page at `http://raw.cs.berkeley.edu/texpoint/index.htm`.