AAMAS 2012 Submission in LaTeX Format*

Paper XXX Kim Svensson†
University of Southampton
Southampton SO17 1BJ
United Kingdom
ks6g10@ecs.soton.ac.uk

ABSTRACT

This paper provides a sample of a LaTeX document which conforms, somewhat loosely, to the formatting guidelines for ACM SIG Proceedings. It is based on the *alternate* style which produces a *tighter-looking* paper and was designed in response to concerns expressed, by authors, over page-budgets. It complements the document *Author's (Alternate) Guide to Preparing ACM SIG Proceedings Using LaTeX2$_\epsilon$ and BibTeX*. This source file has been written with the intention of being compiled under LaTeX2$_\epsilon$ and BibTeX.

The developers have tried to include every imaginable sort of "bells and whistles", such as a subtitle, footnotes on title, subtitle and authors, as well as in the text, and every optional component (e.g. Acknowledgments, Additional Authors, Appendices), not to mention examples of equations, theorems, tables and figures.

To make best use of this sample document, run it through LaTeX and BibTeX, and compare this source code with the printed output produced by the dvi file. A compiled PDF version is available on the web page to help you with the 'look and feel'.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Delphi theory

Keywords

AAMAS proceedings, LaTeX, text tagging

1. INTRODUCTION

The *proceedings* are the records of a conference. IFAAMAS seeks to give these conference by-products a uniform, high-quality appearance. To do this, IFAAMAS follows the rigid requirements that ACM has for the format of the proceedings documents: there is a specified format (balanced double columns), a specified set of fonts (Arial or Helvetica and Times Roman) in certain specified sizes (for instance, 9 point for body copy), a specified live area (18 × 23.5 cm [7" × 9.25"]) centered on the page, specified size of margins (2.54cm [1"] top and bottom and 1.9cm [.75"] left and right; specified column width (8.45cm [3.33"]) and gutter size (.083cm [.33"]).

The good news is, with only a handful of manual settings[1], the LaTeX document class file handles all of this for you.

The remainder of this document is concerned with showing, in the context of an "actual" document, the LaTeX commands specifically available for denoting the structure of a proceedings paper, rather than with giving rigorous descriptions or explanations of such commands.

2. BACKGROUND

2.1 Coalition formation

2.2 The *DP* Algorithm

The DP algorithm as shown in algorithm 1 works by producing two output tables, O and f, where each table have one entry per coalition structure. An entry in f represent a value a certain coalition structure is given, while O represent which splitting, if any, maximised the coalition structure for the entry in f which it represent. More elaborated, given all coalitions of agents A, $C \subseteq A$, for each coalition in C, evaluate all pairwise disjoint subsets named splittings on their pairwise collective sum against the coalitions original value. Given one splitting is greater, update the value of the coalition $f(C) := f(C') + f(C \setminus C')$ and assign O on C to represent the new splitting, $O(C) := \{C', C''\}$. These steps are first carried out on all coalition structures with two agents, continuing until N agents. This means, given a coalition structure S with cardinality $|S| = n$, then all coalition structures for the sizes 1,2,...,n-1 have already been evaluated. Given As described later in section ?? the part of the algorithm that is parallilsed is the max function on line 4 which handles the evaluation of all splittings of a given coalition structure.

The table O may be discarded to reduce the memory requirement by half removing instant access to the splittings, the valid splittings is easily retrived however as outlined in

*For use with aamas2012 .cls
†Something

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), June, 4–8, 2012, Valencia, Spain.

[1]Two of these, the `\numberofauthors` and `\alignauthor` commands, you have already used; another, `\balancecolumns`, will be used in your very last run of LaTeX to ensure balanced column heights on the last page.

algorithm 2. Essentialy, all coalitions in $C \in CS^*$ which value in f is not equal to the initial bid in b, find the first splitting that is equal to the value in f. The overhead of this is insignificant as it needs to evaluate at most $n-1$ coalitions compared to the exponential numbers of evaluations carried out [**?**].

Algorithm 1 Dynamic Programming algorithm

INPUT: $b(C)$ the bids for all coalitions $C \subseteq A$ where A is the set of agents.
VARIABLES: f a function that maps from a subset $C \subseteq A$ to a value
O a function that maps from a subset $C \subseteq A$ to the set that maximize the value for set C.

1: **for** all $x \in A$, **do**$f(\{x\}) := b(\{x\}), O\{x\} := \{x\}$ **end for**
2: **for** $i := 2$ to n **do**
3: **for** all $C \subseteq A : |C| == i$ **do**
4: $f(C) := max\{f(C \backslash C') + f(C') : C' \subseteq C \wedge 1 \leq |C'| \leq \frac{|C|}{2}\}$
5: **if** $f(C) \geq b(C)$ **then** $O(C) := C^*$ Where C^* maximizes right hand side of line 4 **end if**
6: **if** $f(C) < b(C)$ **then** $f(C) := b(C) \wedge O(C) := C$ **end if**
7: **end for**
8: **end for**
9: Set $CS^* := \{A\}$
10: **for** all $C \in CS^*$ **do**
11: **if** $O(C) \neq C$ **then**
12: Set $CS^* := (CS^* \setminus \{C\}) \cup \{O(C), C \setminus O(C)\}$
13: Goto 10 and start with a new CS^*
14: **end if**
15: **end for**
16: **return** CS^*

Algorithm 2 Recursive enumeration of most optimal splittings

INPUT: $b(C)$ the bids for all coalitions $C \subseteq A$. $f(C)$ the final values for all coalitions $C \subseteq A$.

1: Set $CS^* := \{A\}$
2: **for** all $C \in CS^*$ **do**
3: **if** $f(C) \neq b(C)$ **then**
4: find first C^* where $f(C) = f(C \backslash C^*) + f(C^*) : C^* \subseteq C \wedge 1 \leq |C^*| \leq \frac{|C|}{2}$
5: Set $CS^* := (CS^* \setminus \{C\}) \cup \{C^*, C \setminus C^*\}$
6: Goto 2 and start with a new CS^*
7: **end if**
8: **end for**
9: **return** CS^*

2.3 The CUDA Architecture

Graphics Processing Units(GPU) from Nvidia and AMD is highly multithreaded, many-core architectures primarily aimed at highly parallel image processing and rendering, however it have in the last years moved more towards supporting general purpose computing. It does so by devoting a larger amount of transitors towards many computational units rather than data caching and advanced flow controll more often seen in CPU architectures. Nvidia describes their general purpose GPU CUDA architecture as a Single Instruction Multiple Threads (SIMT) architecture, meaning groups of multiple threads excecute the same instructions concurrently and is proportional to SIMD architectures. This enables their GPUs to be highly advantageous when performing data-independant and non-divergent tasks.

Table 1: Memory scope, lifetime, and speed

Type	Scope	Lifetime	Relative Speed
Register	Thread & Warp	Thread	Fastest
Shared	Block	Block	Fast
Global	Kernel & Host	Program	Slow

To understand this further the grouping of the threads need to be explained. A kernel which is a device specific CUDA function that is called by the sequential host code, will request a specified number of blocks in a grid of blocks. Each block may to this date consist of up to 1024 threads depending on the compatability of the card, with a maximum grid size of $2^{31}-1$ blocks subjected to compatability. When run, the blocks will be distributed onto available multiprocessors, which then independantly schedule the runtime of the block. Note that blocks may be excecuted concurrently or sequential depending on the current workload and the number of available multiprocessors. The block is split into smaller units of 32 threads called warps, all threads within the same warp are always scheduled the same instruction to be run and this is what embodies the SIMT paradigm. Therefore, branching threads causing inter-warp divergence means a warp will have inactive threads not excecuting any instructions, which may lead to poor efficiency with worst case of sequential performance. Further, warps are scheduled independantly of eachother meaning possible concurent excecution of warps.

The threads communicate with eachother through writes to various types of memory outlined in table 1. There are three types of thread writable memory in the architecture; registers and local memory are each threads coupled memory which is not volatile and may be shared with other threads inside the same warp as described in section 2.4.2. Shared memory is by its name shared between all threads within the same block, as it may be written to by any thread within the block it should be treated as volatile, thus syncronization inside the block have to be consider whilts dealing with shared memory. Finally, global memory is the only persistant memory which will persist between each kernel call, it may be manipulated by the host, but also by any thread, and is the only means of communication inbetween kernels, blocks, and the host.

2.4 Model and Cuda Implementation

Given the three entities of data that is needed to be represented for each coalition structure; the coalition structure itself, its value and the most beneficial splitting. Memory constraints will be imposed given a large amount of agents as a result of its exponential growth of coalition structures. In order to minimize memory usage several technices were imposed. Firstly, given N agents, the number of values for each coalition structure possible is $2^{|N|}$, which was repre-

Table 2: Splittings of $C = \{f_1, f_3, f_4\}$ Binary $C = 1101$

Set system	$\{f_1\}\{f_3, f_4\}$		$\{f_3\}, \{f_1, f_4\}$		$\{f_4\}, \{f_1, f_3\}$	
Binary system	0001	1010	0010	1001	1000	0011

Table 3: Splittings of $C = \{f_1, f_3, f_4\}$ Binary $C = 1101$

Set system	$\{f_1\}\{f_3, f_4\}$		$\{f_3\}, \{f_1, f_4\}$		$\{f_4\}, \{f_1, f_3\}$	
Binary system	0001	1100	0100	1001	1000	0101

sented as a simple value array. The coalition structure itself may be represented as an integer where the n'th agent of the coalition structure is represented by setting the n'th bit in an binary integer. Given four agents $A = f_1, f_2, f_3, f_4$, coalition $C = f_1, f_3, f_4$ would be represented as $C = 1101$ in the binary system and 11 in the decimal system. Therefore, if the coalition structure is represented as an integer it can implicitly be stored as an index to its coalition value and most beneficial splitting.

2.4.1 Coalition Structure Splittings

Algorithm 3 ***initShift*** input $Coalition : C$

1: $t := C$
2: $count := 0$
3: **while** $t > 0$ **do**
4: $\quad index := FindFirstSet(t)$
5: $\quad shift_{count} := index$
6: $\quad nullBit(t, index)$
7: $\quad count++$
8: **end while**
9: **return** shift

Splittings as mentioned are pairwise disjoint subsets of a coalition structure, given the coalition structure $C = \{f_1, f_3, f_4\}$ the splittings are shown in table 3. In order to generate the splitting there is essentially two methods used, the, initShift, initialSplit and nextSplit methods. The function initShift as detailed in algorithm 3 is necessary to setup the environment for all calls to initialSplit, what it does is using the bit operation $findfirstset$ to find the indexes of all bits set in from the integer coalition input. This will give each entry in the $shift$ array an unique number. This unique numbers will be used by initialSplit to distribute the bits of the count to fit the configurations bits. It does so by taking a count as input representing which n'th splitting should be created, finds the index of its set bits, and finaly left shifts each bit with the value in the shift array its index reference to.

nextSplit works through a recurence relation which means in order to have concurent threads independant of eachother, an initial splitting for each thread have to be calculated using initialSplit. initialSplit works by first genereting an packed index array of which bits are set in the coalition structure using initShift. Given which n'th splitting it should generate, it distributes the bits of n to the corresponding bits of coalition C. Thereafter nextSplit will be used to generate the next splitting.

Algorithm 4 ***nextSplit*** input $Coalition : C$ $Splitting : S$

1: $C' := twosComplement(C)$
2: $S' := bitwiseAND((C' + S), C)$
3: **return** S'

Algorithm 5 ***initialSplit*** input $Count : n, Coalition : C$

1: $t := n$
2: $S := 0$
3: **while** $t > 0$ **do**
4: $\quad index := FindFirstSet(t)$
5: $\quad S := S + leftShift(1, shift_{index,C})$
6: $\quad nullBit(t, index)$??
7: **end while**
8: **return** S

2.4.2 Reduction

As the evaluation of each coalition structure is to find the splitting of the coalition structure which maximises the value of the coalition structure, it is simply needed to compare the values of all splittings with eachother to find the most valued one.

The reduction is done on four levels of scope as seen on lines ?? to ?? in algorithm ??. On thread level, each thread evaluate a number of splittings to determine their most valued splitting, then all threads inside the same warp concurrently exchange their largest register values to find the most valued splitting among the warp. This is done by utilizing a function called $\_\_shfl\_xor$ which allows for an exchange of register values between threads within the same warp, allowing for a substantial reduction in shared memory use. On thread block level, the threads are split up into two groups, active and inactive threads. The active threads will compare its value against a corresponding inactive thread, then half the number of active thread. Iterate until the maximum value have converged into one single thread holding the maximum value for the whole thread block. Finally, the single thread will attempt to update the global memory using atomic functions given that the value of the coalition in table f is less.

We have already seen several typeface changes in this sample. You can indicate italicized words or phrases in your text with the command `\textit`; emboldening with the command `\textbf` and typewriter-style (for instance, for computer code) with `\texttt`. But remember, you do not have to indicate typestyle changes when such changes are part of the *structural* elements of your article; for instance, the heading of this subsection will be in a sans serif[2] typeface, but that is handled by the document class file. Take care with the use of[3] the curly braces in typeface changes; they mark the beginning and end of the text that is to be in the different typeface.

You can use whatever symbols, accented characters, or non-English characters you need anywhere in your document; you can find a complete list of what is available in the *LaTeX User's Guide*[5].

[2]A third footnote, here. Let's make this a rather short one to see how it looks.
[3]A fourth, and last, footnote.

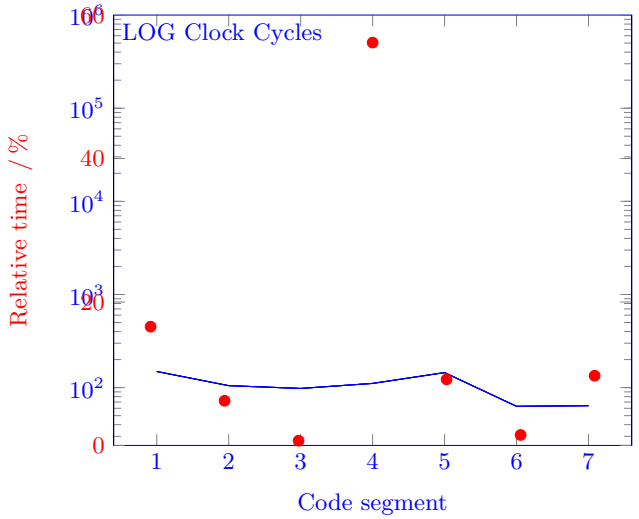
2.5 Algorithm

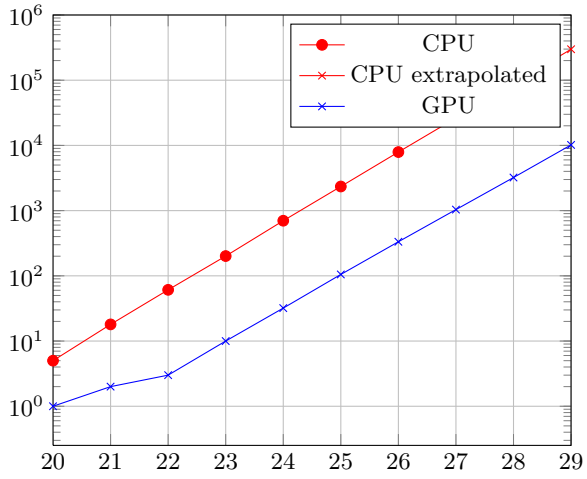
2.6 Experimental setup

The GPU instance of the algoritm was run on a linux desktop computer using CUDA version 5.0 containing 12GiB DDR3 RAM, 3.2GHz AMD Phenom II X4 CPU and a consumer grade NVIDIA GeForce GTX 660 Ti with a GPU clock of 915MHz and 6008MHz effective clock on the memory.

3. RESULTS

LOG Clock Cycles and relative time for each code segment





3.1 Math Equations

You may want to display math equations in three distinct styles: inline, numbered or non-numbered display. Each of the three are discussed in the next sections.

3.1.1 Inline (In-text) Equations

A formula that appears in the running text is called an inline or in-text formula. It is produced by the **math** environment, which can be invoked with the usual `\begin.`

Input

f The array which holds the bids

Φ A reference to an maximum value bucket unique for each kernel

C_0 The first coalition struction to do evaluation on

Ψ The maximum number of splittings

Constants

λ How many bids should be evaluated per thread

$confpkernel$

$nparallelconf$

Variables

Υ A shared array containing warps maximum bid values

υ A local array containing one of the threads bid value

$bid = blockIdx.x$ Which block the threads belong to

$conf$

$bdim = blockdim.x$ How many threads inside the block

$tid = threadIdx.x$ The thread index inside the block

$\psi := \lambda * (tid + bdim * bid)$ Initial subset construction index

Start of algorithm

1: **if** $tid = 0$ **then**
2: $conf_0 := C_0$
3: **for** $i := 1$ to $confpkernel$ **do**
4: $conf_i := nextCoalitoin(conf_{i-1})$
5: **end for**
6: **end if**
7: **if** $tid < confpkernel$ **then**
8: $initShift(conf_{tid})$
9: **end if**
10: **for** $x := 0$ to $confpkernel$ **do**
11: Set all values in υ to 0
12: **if** $\psi \geq \Psi$ **then**
13: goto postfetch
14: **end if**
15: **for** $z := 0$ to $nparallelconf$ **do**
16: **if** $conf_{z+x} >= maxval$ **then**
17: break
18: **end if**
19: $C := initialSplit(\psi, conf_{z+x}$
20: $\upsilon_{0,z} := f(conf_{z+x} \backslash C) + f(C)$
21: $C := nextSplit(C)$
22: $\upsilon_{1,z} := f(conf_{z+x} \backslash C) + f(C)$
23: **end for**
24: **for** $z := 0tonparallelconf$ **do**
25: **if** $\upsilon_{1,z} > \upsilon_{0,z}$ **then**
26: $\upsilon_{0,z} := \upsilon_{1,z}$
27: **end if**
28: $warpReduction(\upsilon_{0,z})$
29: **if** $tid\%32 = 0$ **then**
30: $i := tid/32$
31: $\Upsilon_{i,z} := \upsilon_{0,z}$
32: **end if**
33: **end for**
34: $blockReduction()$
35: **if** $tid = 0$ **then**
36: $atomicUpdate()$
37: **end if**
38: $x := x + nparallelconf$
39: **end for**

. .\end construction or with the short form $. . .$. You can use any of the symbols and structures, from α to ω, available in LaTeX[5]; this section will simply show a few examples of in-text equations in context. Notice how this equation: $\lim_{n\to\infty} x = 0$, set here in in-line math style, looks slightly different when set in display style. (See next section).

3.1.2 Display Equations

A numbered display equation – one set off by vertical space from the text and centered horizontally – is produced by the **equation** environment. An unnumbered display equation is produced by the **displaymath** environment.

Again, in either environment, you can use any of the symbols and structures available in LaTeX; this section will just give a couple of examples of display equations in context. First, consider the equation, shown as an inline equation above:

$$\lim_{n\to\infty} x = 0 \tag{1}$$

Notice how it is formatted somewhat differently in the **displaymath** environment. Now, we'll enter an unnumbered equation:

$$\sum_{i=0}^{\infty} x + 1$$

and follow it with another numbered equation:

$$\sum_{i=0}^{\infty} x_i = \int_0^{\pi+2} f \tag{2}$$

just to demonstrate LaTeX's able handling of numbering.

3.2 Citations

Citations to articles [1, 3, 2, 4], conference proceedings [3] or books [6, 5] listed in the Bibliography section of your article will occur throughout the text of your article. You should use BibTeX to automatically produce this bibliography; you simply need to insert one of several citation commands with a key of the item cited in the proper location in the `.tex` file [5]. The key is a short reference you invent to uniquely identify each work; in this sample document, the key is the first author's surname and a word from the title. This identifying key is included with each item in the `.bib` file for your article.

The details of the construction of the `.bib` file are beyond the scope of this sample document, but more information can be found in the *Author's Guide*, and exhaustive details in the *LaTeX User's Guide*[5].

This article shows only the plainest form of the citation command, using `\cite`. This is what is stipulated in the SIGS style specifications. No other citation format is endorsed or supported.

3.3 Tables

Because tables cannot be split across pages, the best placement for them is typically the top of the page nearest their initial cite. To ensure this proper "floating" placement of tables, use the environment **table** to enclose the table's contents and the table caption. The contents of the table itself must go in the **tabular** environment, to be aligned properly in rows and columns, with the desired horizontal and vertical rules. Again, detailed instructions on **tabular** material is found in the *LaTeX User's Guide*.

Table 4: Frequency of Special Characters

Non-English or Math	Frequency	Comments
Ø	1 in 1,000	For Swedish names
π	1 in 5	Common in math
$	4 in 5	Used in business
Ψ_1^2	1 in 40,000	Unexplained usage

Figure 1: A sample black and white graphic.

Immediately following this sentence is the point at which Table 1 is included in the input file; compare the placement of the table here with the table in the printed dvi output of this document.

To set a wider table, which takes up the whole width of the page's live area, use the environment **table*** to enclose the table's contents and the table caption. As with a single-column table, this wide table will "float" to a location deemed more desirable. Immediately following this sentence is the point at which Table 2 is included in the input file; again, it is instructive to compare the placement of the table here with the table in the printed dvi output of this document.

3.4 Figures

Like tables, figures cannot be split across pages; the best placement for them is typically the top or the bottom of the page nearest their initial cite. To ensure this proper "floating" placement of figures, use the environment **figure** to enclose the figure and its caption.

This sample document contains examples of **.ps**, **.eps** and **.pdf** files to be displayable with LaTeX. Note that if you are using **pdflatex** to typeset your paper, you may only include **.pdf**, **.png**, **.jpeg** and **.gif** files in your paper. If, instead, you choose to use **latex** and **dvipdf** to typeset your paper, you may only include **.ps**, **.eps** files. More details on each of these is found in the *Author's Guide*.

As was the case with tables, you may want a figure that spans two columns. To do this, and still to ensure proper "floating" placement of tables, use the environment **figure*** to enclose the figure and its caption. and don't forget to end the environment with figure*, not figure!

3.5 Theorem-like Constructs

Other common constructs that may occur in your article are the forms for logical constructs like theorems, axioms, corollaries and proofs. There are two forms, one produced by the command `\newtheorem` and the other by the command `\newdef`; perhaps the clearest and easiest way to distinguish them is to compare the two in the output of this sample document:

This uses the **theorem** environment, created by the `\newtheorem` command:

THEOREM 1. *Let f be continuous on $[a, b]$. If G is an*

Figure 2: A sample black and white graphic that has been resized with the `\includegraphics` command.

Table 5: Some Typical Commands

Command	A Number	Comments
`\alignauthor`	100	Author alignment
`\numberofauthors`	200	Author enumeration
`\table`	300	For tables
`\table*`	400	For wider tables

Figure 3: A sample black and white graphic that needs to span two columns of text.

Figure 4: A sample black and white graphic that has been resized with the `\includegraphics` command.

antiderivative for f on $[a, b]$, then

$$\int_a^b f(t)dt = G(b) - G(a).$$

The other uses the **definition** environment, created by the `\newdef` command:

Definition 1. If z is irrational, then by e^z we mean the unique number which has logarithm z:

$$\log e^z = z$$

Two lists of constructs that use one of these forms is given in the *Author's Guidelines*.

There is one other similar construct environment, which is already set up for you; i.e. you must *not* use a `\newdef` command to create it: the **proof** environment. Here is a example of its use:

Proof. Suppose on the contrary there exists a real number L such that

$$\lim_{x\to\infty} \frac{f(x)}{g(x)} = L.$$

Then

$$l = \lim_{x\to c} f(x) = \lim_{x\to c} \left[gx \cdot \frac{f(x)}{g(x)} \right] = \lim_{x\to c} g(x) \cdot \lim_{x\to c} \frac{f(x)}{g(x)} = 0 \cdot L = 0,$$

which contradicts our assumption that $l \neq 0$. □

Complete rules about using these environments and using the two different creation commands are in the *Author's Guide*; please consult it for more detailed instructions. If you need to use another construct, not listed therein, which you want to have the same formatting as the Theorem or the Definition[6] shown above, use the `\newtheorem` or the `\newdef` command, respectively, to create it.

A *Caveat* for the TEX Expert

Because you have just been given permission to use the `\newdef` command to create a new form, you might think you can use TEX's `\def` to create a new command: *Please refrain from doing this!* Remember that your LATEX source code is primarily intended to create camera-ready copy, but may be converted to other forms – e.g. HTML. If you inadvertently omit some or all of the `\def`s recompilation will be, to say the least, problematic.

4. CONCLUSIONS

This paragraph will end the body of this sample document. Remember that you might still have Acknowledgments or Appendices; brief samples of these follow. There is still the Bibliography to deal with; and we will make a disclaimer about that here: with the exception of the reference to the LATEX book, the citations in this paper are to articles which have nothing to do with the present subject and are used as examples only.

5. ACKNOWLEDGMENTS

This section is optional; it is a location for you to acknowledge grants, funding, editing assistance and what have you. In the present case, for example, the authors would like to thank Gerald Murray of ACM for his help in codifying this *Author's Guide* and the **.cls** and **.tex** files that it describes.

6. REFERENCES

[1] M. Bowman, S. K. Debray, and L. L. Peterson. Reasoning about naming systems. *ACM Trans. Program. Lang. Syst.*, 15(5):795–825, November 1993.

[2] J. Braams. Babel, a multilingual style-option system for use with latex's standard document styles. *TUGboat*, 12(2):291–301, June 1991.

[3] M. Clark. Post congress tristesse. In *TeX90 Conference Proceedings*, pages 84–89. TeX Users Group, March 1991.

[4] M. Herlihy. A methodology for implementing highly concurrent data objects. *ACM Trans. Program. Lang. Syst.*, 15(5):745–770, November 1993.

[5] L. Lamport. *LaTeX User's Guide and Document Reference Manual*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1986.

[6] S. Salas and E. Hille. *Calculus: One and Several Variable*. John Wiley and Sons, New York, 1978.

APPENDIX

A. HEADINGS IN APPENDICES

The rules about hierarchical headings discussed above for the body of the article are different in the appendices. In the **appendix** environment, the command **section** is used to indicate the start of each Appendix, with alphabetic order designation (i.e. the first is A, the second B, etc.) and a title (if you include one). So, if you need hierarchical structure *within* an Appendix, start with **subsection** as the highest level. Here is an outline of the body of this document in Appendix-appropriate form:

A.1 Introduction

A.2 The Body of the Paper

A.2.1 Type Changes and Special Characters

A.2.2 Math Equations

Inline (In-text) Equations.

Display Equations.

A.2.3 Citations

A.2.4 Tables

A.2.5 Figures

A.2.6 Theorem-like Constructs

A Caveat for the TEX Expert

A.3 Conclusions

A.4 Acknowledgments

A.5 Additional Authors

This section is inserted by LaTeX; you do not insert it. You just add the names and information in the `\additionalauthors` command at the start of the document.

A.6 References

Generated by bibtex from your .bib file. Run latex, then bibtex, then latex twice (to resolve references) to create the .bbl file. Insert that .bbl file into the .tex source file and comment out the command `\thebibliography`.

B. MORE HELP FOR THE HARDY

The aamas2012 .cls file is based on the sig-alternate.cls file that is itself chock-full of succinct and helpful comments. If you consider yourself a moderately experienced to expert user of LaTeX, you may find reading it useful but please remember not to change it.