Writing Technical Articles

The notes below apply to technical papers in computer science and electrical engineering, with emphasis on papers in systems and networks.

Read Strunk and White, *Elements of Style*. Again.

Give the paper to somebody else to read. If you can, find two people: one person familiar with the technical matter, another only generally familiar with the area.

Papers can be divided roughly into two categories, namely original research papers and survey papers. There are papers that combine the two elements, but most publication venues either only accept one or the other type or require the author to identify whether the paper should be evaluated as a research contribution or a survey paper. (Most research papers contain a "related work" section that can be considered a survey, but it is usually brief compared to the rest of the paper and only addresses a much narrower slice of the field.)

Research Papers

A good research paper has a clear statement of the problem the paper is addressing, the proposed solution(s), and results achieved. It describes clearly what has been done before on the problem, and what is new.

The goal of a paper is to describe novel technical results. There are four types of technical results:

1. **An algorithm;**
2. **A system construct: such as hardware design, software system, protocol, etc.;**

   > **One goal of the paper is to ensure that the next person who designs a system like yours doesn't make the same mistakes and takes advantage of some of your best solutions. So make sure that the hard problems (and their solutions) are discussed and the non-obvious mistakes (and how to avoid them) are discussed. (Craig Partridge)**

3. **A performance evaluation: obtained through analyses, simulation or measurements;**
4. **A theory: consisting of a collection of theorems.**

A paper should focus on

- **describing the results in sufficient details to establish their validity;**
- **identifying the novel aspects of the results, i.e., what new knowledge is reported and what makes it non-obvious;**
- **identifying the significance of the results: what improvements and impact do they suggest.**

Paper Structure

- **Typical outline of a paper is:**
  - **Abstract, typically not more than 100-150 words;**
  - **Introduction (brief!): introduce problem, outline solution; the statement of the problem should include a clear statement why the problem is important (or interesting).**
  - **Related Work (or before summary). Hint: In the case of a conference, make sure to cite the work of the PC co-chairs and as many other PC members as are remotely plausible, as well as from anything relevant from the previous two proceedings. In the case of a journal or magazine, cite anything relevant from last 2-3 years or so volumes.**

- **Outline of the rest of the paper: "The remainder of the paper is organized as follows. In Section 2, we introduce ..Section 3 describes ... Finally, we describe future work in Section 5." [Note that Section is capitalized. Also, vary your expression between "section" being the subject of the sentence, as in "Section 2 discusses ..." and "In Section, we discuss ...".]**
- **Body of paper**
  - **problem**
  - **approach, architecture**
  - **results**

  The body should contain sufficient motivation, with at least one example scenario, preferably two, with illustrating figures, followed by a crisp generic problem statement model, i.e., functionality, particularly emphasizing "new" functionality. The paper may or may not include formalisms. General evaluations of your algorithm or architecture, e.g., material proving that the algorithm is O(log N), go here, not in the evaluation section.

  Architecture **of proposed system(s) to achieve this model should be more generic than your own peculiar implementation. Always include at least one figure.**

  Realization: **contains actual implementation details when implementing architecture isn't totally straightforward. Mention briefly implementation language, platform, location, dependencies on other packages and minimum resource usage if pertinent.**

  Evaluation: **How does it really work in practice? Provide real or simulated performance metrics, end-user studies, mention external technology adoptors, if any, etc.**
- **Related work, if not done at the beginning**
- **Summary and Future Work**
  - **often repeats the main result**
- **Acknowledgements**
- **Bibliography**
- **Appendix (to be cut first if forced to):**
  - **detailed protocol descriptions**
  - **proofs with more than two lines**
  - **other low-level but important details**

It is recommended that you write the approach and results sections first, which go together. Then problem section, if it is separate from the introduction. Then the conclusions, then the intro. Write the intro last since it glosses the conclusions in one of the last paragraphs. Finally, write the abstract. Last, give your paper a title.

Abstract

- **The abstract must not contain references, as it may be used without the main article. It is acceptable, although not common, to identify work by author, abbreviation or RFC number. (For example, "Our algorithm is based upon the work by Smith and Wesson.")**
- **Avoid use of "in this paper" in the abstract. What other paper would you be talking about here?**
- **Avoid general motivation in the abstract. You do not have to justify the importance of the Internet or explain what QoS is.**
- **Highlight not just the problem, but also the principal results. Many people read abstracts and then decide whether to bother with the rest of the paper.**
- **Since the abstract will be used by search engines, be sure that terms that identify your work are found there. In particular, the name of any protocol or system developed and the general area ("quality of service", "protocol verification", "service creation environment") should be contained in the abstract.**
- **Avoid equations and math. Exceptions: Your paper proposes** $E = m c^2$**.**

Introduction

- **Avoid stock and cliche phrases such as "recent advances in XYZ" or anything alluding to the growth of the Internet.**
- **Be sure that the introduction lets the reader know what this paper is about, not just how important your general area of research is. Readers won't stick with you for three pages to find out what you are talking about.**
- **The introduction must motivate your work by pinpointing the problem you are addressing and then give an overview of your approach and/or contributions (and perhaps even a general description of your results). In this way, the intro sets up my expectations for the rest of your paper -- it provides the context, and a preview.**
- **Repeating the abstract in the introduction is a waste of space.**
- **Example bad introduction:**

  > **Here at the institute for computer research, me and my colleagues have created the SUPERGP system and have applied it to several toy problems. We had previously fumbled with earlier versions of SUPERGPSYSTEM for a while. This system allows the programmer to easily try lots of parameters, and problems, but incorporates a special constraint system for parameter settings and LISP S-expression parenthesis counting.**
  >
  > **The search space of GP is large and many things we are thinking about putting into the supergpsystem will make this space much more colorful.**

- **A pretty good introduction, drawn from Eric Siegel's class:**

  Many new domains for genetic programming require evolved programs to be executed for longer amounts of time. For example, it is beneficial to give evolved programs direct access to low-level data arrays, as in some approaches to signal processing \cite{teller5}, and protein segment classification \cite{handley,koza6}. This type of system automatically performs more problem-specific engineering than a system that accesses highly preprocessed data. However, evolved programs may require more time to execute, since they are solving a harder task.

  Previous or obvious approach:

  (Note that you can also have a related work section that gives more details about previous work.)) One way to control the execution time of evolved programs is to impose an absolute time limit. However, this is too constraining if some test cases require more processing time than others. To use computation time efficiently, evolved programs must take extra time when it is necessary to perform well, but also spend less time whenever possible.

  Approach/solution/contribution:

  The first sentence of a paragraph like this should say what the contribution is. Also gloss the results.)

  In this chapter, we introduce a method that gives evolved programs the incentive to strategically allocate computation time among fitness cases. Specifically, with an *aggregate computation time ceiling* **imposed over a series of fitness cases, evolved programs dynamically choose when to stop processing each fitness case. We present experiments that show that programs evolved using this form of fitness take less time per test case on average, with minimal damage to domain performance. We also discuss the implications of such a time constraint, as well as its differences from other approaches to {\it multiobjective problems}. The dynamic use of resources other than computation time, e.g. memory or fuel, may also result from placing an aggregate limit over a series of fitness cases.**

  Overview:

  The following section surveys related work in both optimizing the execution time of evolved programs and evolution over Turing-complete representations.

Next we introduce the game Tetris as a test problem. This is followed by a description of the aggregate computation time ceiling, and its application to Tetris in particular. We then present experimental results, discuss other current efforts with Tetris, and end with conclusions and future work.

Body of Paper

- **Avoid use of passive tense if at all possible. Example: "In each reservation request message, a refresh interval used by the sender is included." reads better and shorter as "Each ... message includes ..."**
- **Use strong verbs instead of lots of nouns. Examples:**

verbose, weak verbs, bad	short, strong, good
make assumption	assume
is a function of	depends on
is an illustration	illustrates
is a requirement	requires, need to

- **Check for missing articles, particularly if your native tongue doesn't have them. Roughly, concepts and classes of things don't, most everything else more specific does. ("Routers route packets. The router architecture we consider uses small rodents.") [NEED POINTER HERE]**
- **Use consistent tense (present, usually, unless reporting results achieved in earlier papers).**
- **Use hyphens for concatenated words: "end-to-end architecture", "real-time operating system" (but "the computer may analyze the results in real time"), "per-flow queueing", "flow-enabled", ...**

  In general, hyphens are used
  - **adding prefixes that would result in double vowels (except for co-, de-, pre-, pro-), e.g., supra-auditory**
  - **all-: all-around, all-embracing;**
  - **half-: half-asleep, half-dollar (but halfhearted, halfway);**
  - **quasi-: quasi-public**
  - **self-: self-conscious, self-seeking (but selfhood, selfless)**
  - **to distinguish from a solid homograph, e.g., re-act vs. react, re-pose vs. repose, re-sign vs. resign, re-solve vs. resolve, re-lease vs. release**
  - **A compound adjective made up of an adjective and a noun in combination should usually be hyphenated. (WiT, p. 230) Examples: cold-storage vault, hot-air heating, short-term loan, real-time operating system, application-specific integrated circuit, Internet-based.**
  - **words ending in -like when the preceding word ends in 'l', e.g., shell-like**
- **Numbers ten or less are spelled out: "It consists of three fields", not "3 fields".**
- **Avoid in-line enumeration like: "Packets can be (a) lost, (b) stolen, (c) get wet." The enumeration only interrupts the flow of thought.**
- **Avoid itemization (bullets), as they take up extra space and make the paper read like PowerPoint slides. Bullets can be used effectively for emphasis of key points. If you want to describe components or algorithms, often the description environment works better, as it highlights the term, providing a low-level section delineation.**
- **Instead of "[1] shows" use "Smith [1] showed" or "Smith and Jones [1] showed" or "Smith *et al* [1] showed" (if more than two authors). Or, alternatively, "the foobar protocol [1] is an example ...". This keeps the reader from having to flip back to the references, as they'll recognize many citations by either author name or project name. No need to refer to RFC numbers in the text (except in RFCs and Internet Drafts). Exception for very low-level presentation: "RFC822-style addresses".**
- **Use normal capitalization in captions ("This is a caption", not "This is a Caption").**
- **Avoid excessive parenthesized remarks as they make the text hard to read; fold into the main sentence. Check whether the publication allows footnotes - some magazines frown upon them. More than two footnotes per page is a bad sign. You probably should have applied to law school instead.**

- **There is no space between the text and the superscript for the footnote. I.e., in LaTeX, it's** `text\footnote{}` **rather than** `text \footnote{}`**.**
- **Check that abbreviations are always explained before use. Exceptions, when addressed to the appropriate networking audience: ATM, BGP, ftp, HTTP, IP, IPv6, RSVP, TCP, UDP, RTP, RIP, OSPF, BGP, SS7. Be particularly aware of the net-head, bell-head perspective. Even basic terms like PSTN and POTS aren't taught to CS students... For other audiences, even terms like ATM are worth expanding, as your reader might wonder why ATM has anything to do with cells rather than little green pieces of paper.**
- **Never start a sentence with "and".**
- **Avoid capitalization of terms. Your paper is not the U.S. Constitution or Declaration of Independence. Technical terms are in lower-case, although some people use upper case when explaining an acronym, as in "Asynchronous Transfer Mode (ATM)".**
- **Each paragraph should have a lead sentence summarizing its content. If this doesn't work naturally, the paragraph is probably too short. Try reading just the first lines of each paragraph - the paper should still make sense. For example,**

  > **There are two service models, integrated and differentiated service. Integrated service follows the German approach that anything that isn't explicitly allowed is verboten. It strictly regulates traffic, but also makes the trains run on time. Differentiated service follows the Animal Farm appraoch, where some traffic is more equal than others. It seems simpler, until one has to worry about proletariat traffic dressing up as the aristocracy.**

- `$i$th`**, not** `$i-th$`**.**
- **Units are always in roman font, never** *italics* **or LaTeX math mode. Units are set off by one (thin) space from the number. In LaTeX, use ~ to avoid splitting number and units across two lines.** `\;` **or** `\,` **produces a thin space.**
- **Use "kb/s" or "Mb/s", not "kbps" or "Mbps" - the latter are not scientific units. Be careful to distinguish "Mb" (Megabit) and "MB" (Megabytes), in particular "kb" (1,000 bits) and "KB" (1,024 bytes). Units and Measurements, Taligent style guide**
- **Use "ms", not "msec", for milliseconds.**
- **Use "0.5" instead of ".5", i.e., do not omit the zero in front of the decimal point. (***Words into Type* **recommends that "for quantities less than one, a zero should be set before the decimal point except for quantities that never exceed one.")**
- **Avoid "etc."; use "for example", "such as", "among others" or, better yet, try to give a complete list (unless citing, for example, a list of products known to be incomplete), even if abstract. See also Strunk and White:**

  > **Etc.: Not to be used of persons. Equivalent to and the rest, and so forth, and hence not to be used if one of these would be insufficient, that is, if the reader would be left in doubt as to any important particulars. Least open to objection when it represents the last terms of a list already given in full, or immaterial words at the end of a quotation. At the end of a list introduced by such as, for example, or any similar expression, etc. is incorrect.**

- **Avoid bulleted lists of one-sentence paragraphs. They make your paper look like a slide presentation and interfere with smooth reading.**
- **Avoid excessive use of "i.e.". Vary your expression: "such as", "this means that", "because", ... "I.e." is not the universal conjunction!**
- **Remember that "i.e." and "e.g." are** *always* **followed by a comma.**
- **Do not use ampersands (&) or slash-abbreviations (such as s/w or h/w) in formal writing; they are acceptable for slides.**
- **"respectively" is preceded by a comma, as in "The light bulbs lasted 10 and 100 days, respectively."**
- **Never use "related works" unless you are talking about works of art. It's "related work".**
- **Use "in Figure 1" instead of "following figure" since figures may get moved during the publication**

or typesetting process. Don't assume that the LaTeX figure stays where you put it.
- Section, Figure and Table are capitalized, as in "As discussed in Section 3". Figure can be abbreviated, but the others are not usually, but that's a matter of taste - just be consistent.
- Do not use GIF images for figures, as GIFs produce horrible print quality and are huge. Export into PostScript. At that stage, you'll learn to "appreciate" Microsoft products. `xfig` and `gnuplot` generally produce PostScript that can be included without difficulties.
- Figures show, depict, indicate, illustrate. Avoid "(refer to Fig. 17)". Often, it is enough to simply put the figure reference in parenthesis: "Packet droppers (Fig. 17) have a pipe to the bit bucket, which is emptied every night."

Bibliography

- Avoid use of *et al.* in a bibliography unless list is very long (five or more authors). The author subsumed into *et al.* may be your advisor or the reviewer... Note punctuation of *et al.*.
- If writing about networks or multimedia, use the network bibliography. All entries not found there should be sent to me. A listing of frequently-used references for networks is available.
- Internet drafts must be marked ``work in progress".
- Book citations include publication years, but no ISBN number.
- It is now acceptable to include URLs to material, but it is probably bad form to include a URL pointing to the author's web page for papers published in IEEE and ACM publications, given the copyright situation. Use it for software and other non-library material. Avoid long URLs; it may be sufficient to point to the general page and let the reader find the material. General URLs are also less likely to change.
- Leave a space between first names and last name, i.e., "J. P. Doe", not "J.P.Doe".

Acknowledgements

- Generally, anonymous reviewers don't get acknowledged, unless they really provided an exceptional level of feedback or insight. Rather than "We thank X for helping us with Y", you might vary this as "X helped with Y.".

Reporting Numerical Results and Simulations

In all but extended abstracts, numerical results and simulations should be reported in enough detail that the reader can duplicate the results. This should include all parameters used, indications of the number of samples that contributed to the analysis and any initial conditions, if relevant.

When presenting simulation results, provide insight into the statistical confidence. If at all possible, provide confidence intervals. If there's a "strange" behavior in the graph (e.g., a dip, peak or change in slope), this behavior either needs to be explained or reasons must be given why this is simply due to statistical aberration. In the latter case, gathering more samples is probably advised.

Figures should be chosen wisely. You can never lay out the whole parameter space, so provide insight into which parameters are significant over what range and which ones are less important. It's not very entertaining to present lots of flat or linear lines.

The description of the graph should not just repeat the graphically obvious such as "the delay rises with the load", but explain, for example, how this increase relates to the load increase. Is it linear? Does it follow some well-known other system behaviors such as standard queueing systems?

LaTeX Considerations

- There's no need to enclose numbers in $$ (math mode).
- Use `\cite{a,b,c}`, not `\cite{a} \cite{b} \cite{c}`.

- **Use the** `\usepackage{times}` **option for LaTeX2e - it comes out much nicer on printers with different resolutions. Plus, compared to cmr, it probably squeezes an extra 10% of text out of your conference allotment.**
- **Multi-letter subscripts are set in roman, not italics. For example,**

```
x_{\mathrm max}
```

- **For uniformity, use the LaTeX2e graphics set, not the earlier psfigure set:**

```
\usepackage{graphics}
...
\begin{figure}
\resizebox{!}{0.5\textheight}{\includegraphics{foo.eps}}
\caption{Some figure}
\label{fig:figure}
\end{figure}
```

Things to Avoid

- **Too much motivational material: 3 reasons are enough -- and they should be described very briefly.**
- **Describing the obvious parts of the result: "Obvious" is defined as any result that a graduate of our program would suggest as a solution if you pose the problem that the result solves.**
- **Describing unnecessary details: a detail is unnecessary, if its omission will not harm the reader's ability to understand the important novel aspects of the result.**
- **Spelling errors: With the availability of spell checkers, there is no reason to have spelling errors in a manuscript. If you as the author didn't take the time to spell-check your paper, why should the editor or reviewer take the time to read it or trust that your diligence in technical matters is any higher than your diligence in presentation? Note, however, that spell checkers don't catch all common errors, in particular word duplication ("the the"). If in doubt, consult a dictionary such as the (on line) Merriam Webster.**

Guidelines for Experimental Papers

"Guidelines for Experimental Papers" set forth for researchers submitting articles to the journal, *Machine Learning***.**

1. **Papers that introduce a new learning "setting" or type of application should justify the relevance and importance of this setting, for example, based on its utility in applications, its appropriateness as a model of human or animal learning, or its importance in addressing fundamental questions in machine learning.**
2. **Papers describing a new algorithm should be clear, precise, and written in a way that allows the reader to compare the algorithm to other algorithms. For example, most learning algorithms can be viewed as optimizing (at least approximately) some measure of performance. A good way to describe a new algorithm is to make this performance measure explicit. Another useful way of describing an algorithm is to define the space of hypotheses that it searches when optimizing the performance measure.**
3. **Papers introducing a new algorithm should conduct experiments comparing it to state-of-the-art algorithms for the same or similar problems. Where possible, performance should also be compared against an absolute standard of ideal performance. Performance should also be compared against a naive standard (e.g., random guessing, guessing the most common class, etc.) as well. Unusual performance criteria should be carefully defined and justified.**
4. **All experiments must include measures of uncertainty of the conclusions. These typically take the form of confidence intervals, statistical tests, or estimates of standard error. Proper experimental methodology should be employed. For example, if "test sets" are used to measure generalization performance, no information from the test set should be available to the learning process.**
5. **Descriptions of the software and data sufficient to replicate the experiments must be included in**

the paper. Once the paper has appeared in Machine Learning, authors are strongly urged to make the data used in experiments available to other scientists wishing to replicate the experiments. An excellent way to achieve this is to deposit the data sets at the Irvine Repository of Machine Learning Databases. Another good option is to add your data sets to the DELVE benchmark collection at the University of Toronto. For proprietary data sets, authors are encouraged to develop synthetic data sets having the same statistical properties. These synthetic data sets can then be made freely available.

6. Conclusions drawn from a series of experimental runs should be clearly stated. Graphical display of experimental data can be very effective. Supporting tables of exact numerical results from experiments should be provided in an appendix.
7. Limitations of the algorithm should be described in detail. Interesting cases where an algorithm fails are important in clarifying the range of applicability of an algorithm.

The Conference Review Process

It is hard to generalize the review process for conferences, but most reputable conferences operate according to these basic rules:

1. The paper is submitted to the technical program chair(s). Many current conferences require electronic submission, in either PostScript or PDF formats, occasionally in Word.
2. The technical program chair assigns the paper to one or more technical program committee members, hopefully experts in their field. The identity of this TPC member is kept secret.
3. The TPC member usually provides a review, but may also be asked to find between one and three reviewers who are not members of the TPC. They may be colleagues of the reviewer at the same institution, his or her graduate students or somebody listed in the references. The graduate student reviews can be quite helpful, since these reviewers often provide more detailed criticism rather than blanket dismissal. Any good conference will strive to provide at least three reviews, however, since conferences operate under tight deadlines and not all reviewers deliver as promised, it is not uncommon that you receive only two reviews.
4. The technical program chair then collects the reviews and sorts the papers according to their average review scores.
5. The TPC, or usuaally a subset that can make the meeting, then meets Usually, the bottom third and the top third are rejected and accepted without (much) further discussion. The papers discussed are those in the middle of the range, where a TPC member feels strongly that the paper ended up in the wrong bin, or where the review scores differ significantly, in particular if there are only two reviews.

Other References

- Berkeley Information Systems and Technology Publications style guide
- Cisco style guide
- Oded Goldreich wrote an essay entitled "How not to write a paper", with recommendations on style and organization.
- Don Knuth has online the TeX source of a book on "Mathematical Writing" (also useful for Computer Science).
- *The structure of paper/report in Systems*, by Michalis Faloutsos, U.C. Riverside
- The Elements of Style. William Strunk Jr. and E.B. White. Macmillan Publishing Co., New York, 1979.

  This is an **amazing** little book that you can read in a few hours. Your writing style will never be the same afterwards. This $8 book is the best investment you can ever make.

- Bugs in Writing. Lyn Dupre', (2nd ed.)

  This is a great book that expands on Strunk&White. It has more examples, and was

written by an author who edited numerous technical publications, many of which were in computer science.

- **The Chicago Manual of Style, Univ. of Chicago Press.**

  This is the bible for American academic style. It's long and heavy, but has everything you ever want to know about style. When in doubt, or if you get conflicting stylistic advice, following The Chicago Manual of Style is your best choice.

- **A Handbook for Scholars by Mary Claire van Leunen; Alfred Knopf, Publisher.**

  This is another useful book written for publishing (computer) scientists.

- **The UIST Guide for Authors is geared towards a specific conference, but the general process and guidelines are similar to many other conferences.**
- *The Science of Scientific Writing*, **George D. Gopen and Judith A. Swan, In** *American Scientist*, **Vol. 78, No. 6 (Nov-Dec, 1990), pp. 550-558.**

  This is a useful article that teaches scientists how to write single sentences and paragraphs.

- **Roy Levin and David D. Redell, An evaluation of the ninth SOSP submissions -or- How (and how not) to write a good systems paper,** *ACM SIGOPS Operating Systems Review* **17(3):35-40 (July, 1983).**
- **Alan Snyder, How to get your paper accepted at OOPSLA,** *OOPSLA '91 Proceedings*, **pp. 359-363.**
- **Ralph Johnson** *et al*, **How to get a paper accepted at OOPSLA,** *Panel at OOPSLA'93*, **pp 429-436.**
- **Craig Partridge, How to Increase the Chances Your Paper is Accepted at ACM SIGCOMML.**

  Generally useful advice that also applies to other networking conferences.

- **What kinds of papers does USENIX publish?**
- **Alan Jay Smith,** *The Task of the Referee*, *IEEE Computer* **23(4):65-71 (April 1990).**

Talks

- **"The Short Talk" (Charles Van Loan)**
- **"Pointers on giving a talk" (D. Messerschmitt)**
- **"Tips for Preparing Scientific Presentation" (ONR)**

Contributors

This page contains material provided by Gail Kaiser, Craig Partridge, Sumit Roy, Eric Siegel, Sal Stolfo, Luca Trevisan, Yechiam Yemini, Erez Zadok.

Last updated Monday, March 05, 2001 14:17:17 by Henning Schulzrinne

HOW TO HAVE YOUR ABSTRACT REJECTED

Mary-Claire van Leunen and Richard Lipton

If your ideas are bad enough all on their own, you needn't worry about this advice, Banality, irrelevance, plagiarism, and plain old madness will get any abstract rejected, no matter how good it is. Similarly, if your ideas are brilliant, pointed, original, and sane, you have a hard road ahead of you. Even the worst abstract may not suffice for rejection Program committees differ in their standards. If, however, you are like most of us, neither a genius nor an idiot, neither Newton nor Simple Simon, you will have to put some effort into making your abstract suitable for rejection. Here are a few tips we can offer.

Submit late. This is the basic rule in having your abstract rejected. Don't even start writing it until the deadline for submission is long past. Keep the program committee informed of your progress. ``Seems to be a little hole in the proof somewhere.'' ``Don't sit on the edge of your chairs.'' ``Almost ready.'' ``It's a-comin'.'' ``Any minute now.'' Everyone on the committee is sure to remember your name when your abstract finally arrives.

Submit incorrectly. The device of sending abstracts to the local arrangements chairman is overused. Try something fresher. Send your abstract to last year's program chairman. Send it to this year's in care of the school where he did his undergraduate work or, better yet, to the school that turned him down for tenure. Send it to someone whose name sounds a little like his. Under any circumstances, be sure to send it postage due.

Grossly exceed the maximum length requirements. Most extended abstracts should be eight to twelve pages long, or between 1,500 and 3,500 words. Your aim, then, should be for at least 10,000 words. (Read symbols aloud to count how many words they are -- don't count characters.) There are several interesting variations on this ploy.

The Godzilla: Submit a seventy-page paper with instructions to - the program committee to read the first twelve pages. Be sure page 12 ends mid-section, mid-paragraph, mid-sentence.

The Monster from the Black Lagoon: Submit a twelve-page abstract with thirty pages of appendices. Be sure there is no way anyone can understand the body of the abstract without reading all of the appendices. By-far the easiest way to accomplish this is to introduce your own utterly idiosyncratic notation. 1+1 = 2, for instance, will in your notation be written

```
    b
     b| <
    1 2 1*.
   z| > ^
```

The King Kong: Submit an eight-page abstract of 20,000 words. You may need special typographic equipment for this one, but don't worry; it exists. With an IBM composer, six-point type, no margins, and no displays, you can write 20,000 words on the head of a pin.

You might think that the opposite strategy would work equally well --- submitting an abstract that falls far short of the minimum requirement. Not so. Look at it from the program committee's point of view. They must read a hundred or more abstracts in the midst of their other duties. Mere brevity after all those monster abstracts is going to look good to them.

Vacuity, however, is an excellent technique, and it may be allied with the shortness strategy for a Run Spot Run abstract. Here is a good example:

> We worked in complexity. We proved some theorems. We proved some big theorems and some little theorems. Some proofs were big, some were small. We tried to match up the proofs with the theorems, but we couldn't always do it. Then we were sleepy and went to bed. Good night.

This is a good example but not a perfect one. Substitute ``computer science'' for ``complexity''" and you will see how much room for improvement there is in any abstract, no matter how vapid it may seem at first glance. Only by constant, careful revision can you insure the rejection of every single abstract you prepare.

A new tactic we would like to commend is the Grocery List. For this you must give at least forty theorems. The North American record stands at 97, but there are allegations that the author was under the influence of chemical stimulants and the judges are currently reserving the title. We confidently expect to see in next year's competition exciting new combinations of the Grocery List with the Run Spot Run and other devices.

Give no motivation. Present your results in a vacuum. Strip your ideas of any hint they might offer as to their origin, direction, or relevance. Say nothing about practical applications unless you are submitting to a theory conference, in which case you should be sure to call them ``pragmatics''.

Be sure also to give no background. Even the novice will know enough to leave off all acknowledgements and references. But the master will go further. He will give the appearance of citation without any substance. He will enclose a reference 1ist on which every item is submitted, in preparation, or a private communication. He will call obscure results by pet names he has invented himself. And he will describe as ``well known'' results published only in Old Serbian --- preferably false ones.

Prove trivial results in exhaustive detail, breaking your proofs into as many lemmas as ou can and disrupting the line of reasoning with notes, remarks, and asides. On the other hand, assert difficult proofs. Assert them badly, with a sneer, if you can manage it. The judicious typographical error in the statement of your theorem adds a note of drollery to this device.

Never under any circumstances provide a cogent verbal sketch of a proof that stresses its provocative turns while leaving the obvious unstated. Never. This alone may get an abstract accepted even if you have faithfully followed all the rest of our advice.

As for your language, it should be pompous, impersonal, drab, and bleary. Work hard at your grammar. There is no excuse for agreement between subject and predicate in any sentence of more than ten words. Indefinite referents combined with false parallels will leave the unwary program committee member clutching his head and wheeling about the room in confusion. Any temptation he had to accept your abstract will disappear instantly.

Mind the appearance of your paper, too. An ancient, faded grey typewriter ribbon is good, but why not try a fading red one for that extra spark of individuality? Sixteen-pound paper seems flimsy enough until you realize that you can find twelve and even less. Why not type on the back of used second sheets and enclose a sanctimonious note about recycling paper products? Why not submit a handwritten manuscript --- written in pencil? Why not have your four-year-old type your paper? And what ever became of the old-fashioned muddy footprint, the coffee ring, and the grease smear? You cannot give too much attention to these small details. They can make the difference between an abstract that is marginally, reluctantly acceptable and one that will be firmly rejected year after year.

(You should understand that once you have a soundly accept-proof manuscript you should resubmit it every year. You will become part of the mythology of your field. As program committee succeeds program committee, the question will be asked, ``Did you get Old Whosit's paper again? What's he calling it this year?'')

A final word: If after adopting all these strategies and developing several of your own you have a paper accepted anyway, do not despair. Do not take it personally. The program committee has certain quotas it must fill. A certain number of papers must be accepted regardless of merit. Your friends and colleagues will understand, and no one will hold it against you. Just don't let it happen too often.

How to Write an Abstract

(Draft 10/20/97)

Phil Koopman, Carnegie Mellon University

Abstract

Because on-line search databases typically contain only abstracts, it is vital to write a complete but concise description of your work to entice potential readers into obtaining a copy of the full paper. This article describes how to write a good computer architecture abstract for both conference and journal papers. Writers should follow a checklist consisting of: motivation, problem statement, approach, results, and conclusions. Following this checklist should increase the chance of people taking the time to obtain and read your complete paper.

Introduction

Now that the use of on-line publication databases is prevalent, writing a really good abstract has become even more important than it was a decade ago. Abstracts have always served the function of "selling" your work. But now, instead of merely convincing the reader to keep reading the rest of the attached paper, an abstract must convince the reader to leave the comfort of an office and go hunt down a copy of the article from a library (or worse, obtain one after a long wait through inter-library loan). In a business context, an "executive summary" is often the *only* piece of a report read by the people who matter; and it should be similar in content if not tone to a journal paper abstract.

Checklist: Parts of an Abstract

Despite the fact that an abstract is quite brief, it must do almost as much work as the multi-page paper that follows it. In a computer architecture paper, this means that it should in most cases include the following sections. Each section is typically a single sentence, although there is room for creativity. In particular, the parts may be merged or spread among a set of sentences. Use the following as a checklist for your next abstract:

- **Motivation:**
  Why do we care **about the problem and the results? If the problem isn't obviously "interesting" it might be better to put motivation first; but if your work is incremental progress on a problem that is widely recognized as important, then it is probably better to put the problem statement first to indicate which piece of the larger problem you are breaking off to work on. This section should include the importance of your work, the difficulty of the area, and the impact it might have if successful.**
- **Problem statement:**
  What *problem* **are you trying to solve? What is the** *scope* **of your work (a generalized approach, or for a specific situation)? Be careful not to use too much jargon. In some cases it is appropriate to put the problem statement before the motivation, but usually this only works if most readers already understand why the problem is important.**
- **Approach:**
  How did you go about solving **or making progress on the problem? Did you use simulation, analytic models, prototype construction, or analysis of field data for an actual product? What was the** *extent* **of your work (did you look at one application program or a hundred programs in twenty different programming languages?) What important** *variables* **did you control, ignore, or measure?**
- **Results:**
  What's the answer? **Specifically, most good computer architecture papers conclude that something**

is so many percent faster, cheaper, smaller, or otherwise better than something else. Put the result there, in numbers. Avoid vague, hand-waving results such as "very", "small", or "significant." If you must be vague, you are only given license to do so when you can talk about orders-of-magnitude improvement. There is a tension here in that you should not provide numbers that can be easily misinterpreted, but on the other hand you don't have room for all the caveats.

- **Conclusions:**
  What are the implications **of your answer? Is it going to change the world (unlikely), be a significant "win", be a nice hack, or simply serve as a road sign indicating that this path is a waste of time (all of the previous results are useful). Are your results** *general*, **potentially generalizable, or specific to a particular case?**

Other Considerations

An abstract must be a fully self-contained, capsule description of the paper. It can't assume (or attempt to provoke) the reader into flipping through looking for an explanation of what is meant by some vague statement. It must make sense all by itself. Some points to consider include:

- **Meet the word count limitation. If your abstract runs too long, either it will be rejected or someone will take a chainsaw to it to get it down to size. Your purposes will be better served by doing the difficult task of cutting yourself, rather than leaving it to someone else who might be more interested in meeting size restrictions than in representing your efforts in the best possible manner. An abstract word limit of 150 to 200 words is common.**
- **Any major restrictions or limitations on the results should be stated, if only by using "weasel-words" such as "might", "could", "may", and "seem".**
- **Think of a half-dozen search phrases and keywords that people looking for your work might use. Be sure that those exact phrases appear in your abstract, so that they will turn up at the top of a search result listing.**
- **Usually the context of a paper is set by the publication it appears in (for example,** *IEEE Computer* **magazine's articles are generally about computer technology). But, if your paper appears in a somewhat un-traditional venue, be sure to include in the problem statement the domain or topic area that it is really applicable to.**
- **Some publications request "keywords". These have two purposes. They are used to facilitate keyword index searches, which are greatly reduced in importance now that on-line abstract text searching is commonly used. However, they are also used to assign papers to review committees or editors, which can be extremely important to your fate. So make sure that the keywords you pick make assigning your paper to a review category obvious (for example, if there is a list of conference topics, use your chosen topic area as one of the keyword tuples).**

Conclusion

Writing an efficient abstract is hard work, but will repay you with increased impact on the world by enticing people to read your publications. Make sure that all the components of a good abstract are included in the next one you write.

Further Reading

Michaelson, Herbert, *How to Write & Publish Engineering Papers and Reports*, **Oryx Press, 1990. Chapter 6 discusses abstracts.**

Cremmins, Edward, *The Art of Abstracting 2nd Edition*, **Info Resources Press, April 1996. This is an entire book about abstracting, written primarily for professional abstractors.**

Phil Koopman 10/97