# How to Write a Good Paper
# and
# How to Give a Good Talk

(based on Simon Peyton Jones's how-to tutorials)


**Liang Huang**

Oregon State University


Simon Peyton Jones's original slides

Liang's interpretations and suggestions

# Why I am teaching writing...

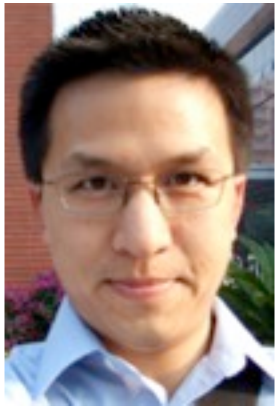
- not because I am a good writer (in fact I'm not)...
- but because I was a terrible writer!
- NLP studies sentence and discourse structures
- NLP has arguably the highest standards in writing

# How I learned writing

- ’03: knew nothing about writing (though I had several papers in China)
- ’04-5: all I wrote was crap; David turned them into beauty
  - e.g. I wrote in a draft (Huang & Chiang 2005) ~~*“Bikel (2002) was a hack...”*~~
- ’06-7: some progress by writing, writing, and writing...
  - one of the reviews for a submission with David (rejected)
    - *“in general this paper is written with admirable clarity, except for it doesn’t seem to be written by a single author or with the same level of discretion...”* (this made me not sad about the rejection... :P)
  - turns out David had revised all but one section (Huang & Chiang 2007)
- ’08 and on: all my submissions got 4 or 5 in “clarity”

# How I learned writing

- fallacy: students learn to write mainly from advisors
- truth: learn from anybody whom you can learn from
- I learned writing mainly from...
- and from writing seminars of...
- and from the slides by...


D. Chiang


K. Knight


L. Saul


B. Pierce


S. Peyton-Jones

the rest of the talk is largely based on Simon PJ's slides.


D. Gildea

# Why Study Writing/Presentation?

- research is all about communication
- communication involves writing, presenting, teaching
- the first and only principle in communication
  - always have your audience (the reader) in mind!
  - technical writing is NOT self-expression, but *infection*
  - you are to *teach* those who do not understand it
  - not those who already understand (what's the point?)
- explaining something deep in a clear way is an art
  - and involves a lot of creativity

# Writing is NOT about English

- writing is not about language, but about logic
  - writing is equally hard for both native and non-native speakers of English
  - a bad paper is bad in any language
- different levels of writing
  - high-level (paper): global shape, logic, argument, style
  - mid-level (discourse): coherence within a paragraph
  - low-level (sentences): ordering of words and phrases
  - lowest-level (words): word choice, grammar

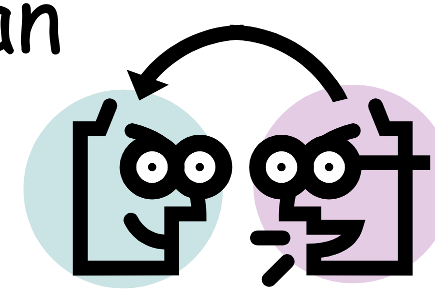
# First Principle: Audience-Centric

- always have your audience (the reader) in mind!
- writing is communication, NOT self-expression!
- reader-centric attitude, not self-centric

# The purpose of your paper

# Papers communicate ideas

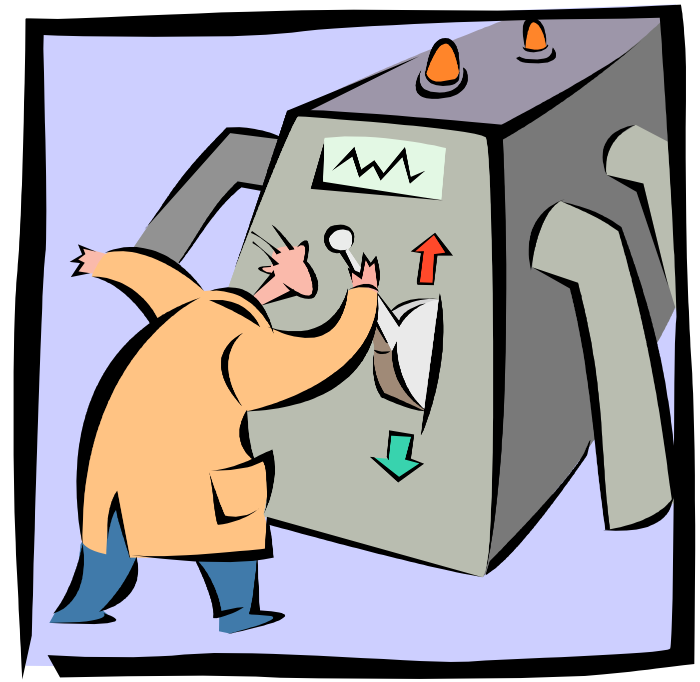
- Your goal: to infect the mind of your reader with **your idea**, like a virus
- Papers are far more durable than programs (think Mozart)



The greatest ideas are (literally) worthless if you keep them to yourself

# The purpose of your paper is **not**...

To describe the WizWoz system



- Your reader does not have a WizWoz
- She is primarily interested in re-usable brain-stuff, not executable artefacts

# Your narrative flow

- Here is a problem
- It's an interesting (important) problem
- It's an unsolved (hard) problem
- **Here is my idea**
- My idea works (details, data)
- Here's how my idea compares to other people's approaches

I wish I knew how to solve that!

I see how that works. Ingenious!

# Structure (conference paper)

- Title (1000 readers)
- Abstract (4 sentences, 100 readers)
- Introduction (1 page, 100 readers)
- The problem (1 page, 10 readers)
- My idea (2 pages, 10 readers)
- The details (5 pages, 3 readers)
- Related work (1-2 pages, 10 readers)
- Conclusions and further work (0.5 pages)

# The abstract

- I usually write the abstract **last**
- Used by program committee members to decide which papers to read
- Four sentences [Kent Beck]

*what* 1. State the problem 起

*why* 2. Say why it's an interesting problem 承

*how* 3. Say what your solution achieves 转

*results* 4. Say what follows from your solution 合

# Example

1. Many papers are badly written and hard to understand
2. This is a pity, because their good ideas may go unappreciated
3. Following simple guidelines can dramatically improve the quality of your papers
4. Your work will be used more, and the feedback you get from others will in turn improve your research

# Structure

- Abstract (4 sentences)
- **Introduction** (1 page)
- The problem (1 page)
- My idea (2 pages)
- The details (5 pages)
- Related work (1-2 pages)
- Conclusions and further work (0.5 pages)

# The introduction (1 page)

intro:

LH: this is the hardest part of writing!

1. Describe the problem
2. State your contributions

need to convey:
importance and hardness

...and that is all

ONE PAGE!

abstract:

1. State the problem
2. Say why it's an interesting problem
3. Say what your solution achieves
4. Say what follows from your solution

# LH Method for Stating the Problem

- intro = “your *slightly* biased view of the history” [N. Dinesh]
- need to convey: importance and depth

| | A | B |
|---|---|---|
| s1 | + | - |
| s2 | - | + |
| new | + | + |

- this is an *important* problem
- the dominant solution is good in A
- but bad in B (and B is important)
- the alternative solution is good in B
- but bad in A
- Q: how to combine their merits?? a *hard* problem!

example from Huang (2008)

# Forest Reranking: Discriminative Parsing with Non-Local Features*

**Liang Huang**
University of Pennsylvania

## Abstract

Conventional $n$-best reranking techniques often suffer from the limited scope of the $n$-best list, which rules out many potentially good alternatives. We instead propose *forest reranking*, a method that reranks a packed forest of exponentially many parses. Since exact inference is intractable with non-local features, we present an approximate algorithm inspired by forest rescoring that makes discriminative training practical over the whole Treebank. Our final result, an F-score of 91.7, outperforms both 50-best and 100-best reranking baselines, and is better than any previously reported systems trained on the Treebank.

## 1 Introduction

Discriminative reranking has become a popular technique for many NLP problems, in particular, parsing (Collins, 2000) and machine translation (Shen et al., 2005). Typically, this method first generates a list of top-$n$ candidates from a baseline system, and then reranks this $n$-best list with arbitrary features that are not computable or intractable to compute within the baseline system. But despite its apparent success, there remains a major drawback: this method suffers from the limited scope of the $n$-best list, which rules out many potentially good alternatives. For example 41% of the correct parses were not in the candidates of ~30-best parses in (Collins, 2000). This situation becomes worse with longer sentences because the number of possible in-

| | *local* | *non-local* |
|---|---|---|
| conventional reranking | only at the root | |
| DP-based discrim. parsing | exact | N/A |
| *this work*: forest-reranking | exact | *on-the-fly* |

Table 1: Comparison of various approaches for incorporating local and non-local features.

sentence length. As a result, we often see very few variations among the $n$-best trees, for example, 50-best trees typically just represent a combination of 5 to 6 binary ambiguities (since $2^5 < 50 < 2^6$).

Alternatively, discriminative parsing is tractable with exact and efficient search based on dynamic programming (DP) if all features are restricted to be *local*, that is, only looking at a local window within the factored search space (Taskar et al., 2004; McDonald et al., 2005). However, we miss the benefits of non-local features that are not representable here.

Ideally, we would wish to combine the merits of both approaches, where an efficient inference algorithm could integrate both local and non-local features. Unfortunately, exact search is intractable (at least in theory) for features with unbounded scope. So we propose *forest reranking*, a technique inspired by forest rescoring (Huang and Chiang, 2007) that approximately reranks the packed forest of exponentially many parses. The key idea is to compute non-local features incrementally from bottom up, so that we can rerank the $n$-best subtrees at all internal nodes, instead of only at the root node as in conventional reranking (see Table 1). This method can thus

# State your contributions

- Write the list of contributions first
- The list of contributions drives the entire paper: the paper substantiates the claims you have made
- Reader thinks "gosh, if they can really deliver this, that's be exciting; I'd better read on"

# State your contributions

Which of the two is best in practice? The trouble is that the evaluation model has a pervasive effect on the implementation, so it is too much work to implement both and pick the best. Historically, compilers for strict languages (using call-by-value) have tended to use eval/apply, while those for lazy languages (using call-by-need) have often used push/enter, but this is 90% historical accident — either approach will work in both settings. In practice, implementors choose one of the two approaches based on a qualitative assessment of the trade-offs. In this paper we put the choice on a firmer basis:

- We explain precisely what the two models are, in a common notational framework (Section 4). Surprisingly, this has not been done before.
- The choice of evaluation model affects many other design choices in subtle but pervasive ways. We identify and discuss these effects in Sections 5 and 6, and contrast them in Section 7. There are lots of nitty-gritty details here, for which we make no apology — they were far from obvious to us, and articulating these details is one of our main contributions.

  In terms of its impact on compiler and run-time system complexity, eval/apply seems decisively superior, principally because push/enter requires a stack like no other: stack-walking

Bulleted list of contributions

Do not leave the reader to guess what your contributions are!

# Contributions should be **refutable**

| NO! | YES! |
|---|---|
| We describe the WizWoz system. It is really cool. | We give the syntax and semantics of a language that supports concurrent processes (Section 3). Its innovative features are... |
| We study its properties | We prove that the type system is sound, and that type checking is decidable (Section 4) |
| We have used WizWoz in practice | We have built a GUI toolkit in WizWoz, and used it to implement a text editor (Section 5). The result is half the length of the Java version. |

# What does "refutable" mean?

- refutable: falsifiable (可证伪的); easily verifiable

| | not refutable | refutable |
|---|---|---|
|  | ~~I'll devote myself to the American people.~~ | I'll reduce unemployment rate by 5% by 2010. |
|  | ~~政府将尽全力为人民服务.~~ | 政府将在两年之内把PM 2.5降低50%. |
| you | ~~our algorithm is really effective and efficient.~~ | our algorithm is faster than Jones's by a factor of $n^2 \log n$. |

# No "rest of this paper is…"

- Not:

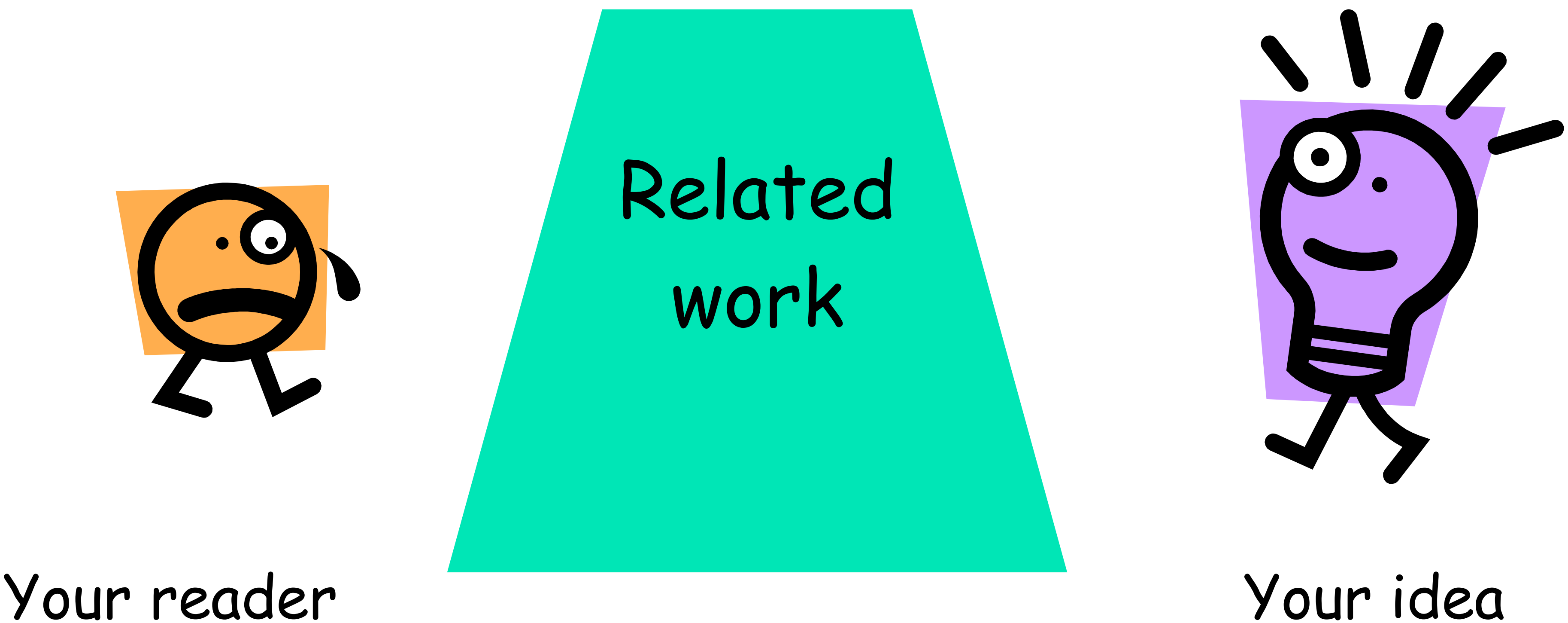
  "The rest of this paper is structured as follows. Section 2 introduces the problem. Section 3 … Finally, Section 8 concludes".

- Instead, **use forward references from the narrative in the introduction**. The introduction (including the contributions) should survey the whole paper, and therefore forward reference every important part.

# Structure

- Abstract (4 sentences)
- Introduction (1 page)
- ~~Related work~~
- The problem (1 page)
- My idea (2 pages)
- The details (5 pages)
- Related work (1-2 pages)
- Conclusions and further work (0.5 pages)

# No related work yet!




We adopt the notion of transaction from Brown [1], as modified for distributed systems by White [2], using the four-phase interpolation algorithm of Green [3]. Our work differs from White in our advanced revocation protocol, which deals with the case of priority inversion as described by Yellow [4].

# No related work yet

- Problem 1: the reader knows nothing about the problem yet; so your (carefully trimmed) description of various technical tradeoffs is absolutely incomprehensible


*LH: you haven't established the vocabulary after sec. 1*

- Problem 2: describing alternative approaches gets between the reader and your idea

*LH: but you will have all notations and vocabulary set up by sec. 6*

# LH: Two Types of Previous Work

- essential background
  - the previous work that your work builds upon
  - or improve upon ("shoulders of giants")
  - => intro (w/o which the readers can't understand your work)
  - or "Section 2: Preliminaries" (mathy background, e.g., Hiero/SCFG)
- related work: other previous work that is just related to yours
  - skipping them doesn't impede the understanding of your work
- simple criteria: can readers understand my work without A?

| your work (secs. 3-5) | related work (sec. 6) |
|---|---|
| essential background (sec. 2) | |

# Examples of Sec. 2: Preliminaries

- Sec. 2 should be tutorial-like

magnitude faster than full-integration with beam search, at the same level of search errors and translation accuracy as measured by BLEU.

## 2 Preliminaries

We establish in this section a unified framework for translation with an integrated $n$-gram language model in both phrase-based systems and syntax-based systems based on synchronous context-free grammars (SCFGs). An SCFG (Lewis and Stearns, 1968) is a context-free rewriting system for generating string pairs. Each rule $A \rightarrow \alpha, \beta$ rewrites a pair of nonterminals in both languages, where $\alpha$ and $\beta$ are the source and target side components, and there is a one-to-one correspondence between the nonterminal occurrences in $\alpha$ and the nonterminal occurrences in $\beta$. For example, the following rule

$$\mathrm{VP} \rightarrow \mathrm{PP}^{(1)}\ \mathrm{VP}^{(2)},\quad \mathrm{VP}^{(2)}\ \mathrm{PP}^{(1)}$$

captures the swapping of VP and PP between Chinese (source) and English (target).

### 2.1 Translation as Deduction

We will use the following example from Chinese to English for both systems described in this section:

*yǔ Shālóng jǔxíng le huìtán*
with Sharon hold [*past*] meeting
'held a meeting with Sharon'

A typical phrase-based decoder generates partial target-language outputs in left-to-right order in the form of *hypotheses* (Koehn, 2004). Each hypothesis has a *coverage vector* capturing the source-language words translated so far, and can be extended into a longer hypothesis by a phrase-pair translating an uncovered segment.

This process can be formalized as a deductive system. For example, the following deduction step grows a hypothesis by the phrase-pair ⟨*yǔ Shālóng*, with Sharon⟩:

$$\frac{(\_\bullet\bullet\bullet) : (w, \text{"held a talk"})}{(\bullet\bullet\bullet\bullet\bullet) : (w + c, \text{"held a talk with Sharon"})} \quad (1)$$

where a • in the coverage vector indicates the source word at this position is "covered" (for simplicity we omit here the ending position of the last phrase which is needed for distortion costs), and where $w$ and $w + c$ are the weights of the two hypotheses, respectively, with $c$ being the cost of the phrase-pair.

Similarly, the decoding problem with SCFGs can also be cast as a deductive (parsing) system (Shieber et al., 1995). Basically, we parse the input string using the source projection of the SCFG while building the corresponding subtranslations in parallel. A possible deduction of the above example is notated:

$$\frac{(\mathrm{PP}_{1,3}) : (w_1, t_1) \qquad (\mathrm{VP}_{3,6}) : (w_2, t_2)}{(\mathrm{VP}_{1,6}) : (w_1 + w_2 + c', t_2 t_1)} \quad (2)$$

where the subscripts denote indices in the input sentence just as in CKY parsing, $w_1$, $w_2$ are the scores of the two antecedent items, and $t_1$ and $t_2$ are the corresponding subtranslations. The resulting translation $t_2 t_1$ is the inverted concatenation as specified by the target-side of the SCFG rule with the additional cost $c'$ being the cost of this rule.

These two deductive systems represent the search space of decoding without a language model. When one is instantiated for a particular input string, it defines a set of derivations, called a *forest*, represented in a compact structure that has a structure of a graph in the phrase-based case, or more generally, a *hypergraph* in both cases. Accordingly we call items like $(\bullet\bullet\bullet\bullet\bullet)$ and $(\mathrm{VP}_{1,6})$ *nodes* in the forest, and instantiated deductions like

$$(\bullet\bullet\bullet\bullet\bullet) \rightarrow (\_\bullet\bullet\bullet) \text{ with Sharon},$$
$$(\mathrm{VP}_{1,6}) \rightarrow (\mathrm{VP}_{3,6})\ (\mathrm{PP}_{1,3})$$

we call *hyperedges* that connect one or more antecedent nodes to a consequent node.

### 2.2 Adding a Language Model

To integrate with a bigram language model, we can use the dynamic-programming algorithms of Och and Ney (2004) and Wu (1996) for phrase-based and SCFG-based systems, respectively, which we may think of as doing a finer-grained version of the deductions above. Each node $v$ in the forest will be split into a set of augmented items, which we call *+LM items*. For phrase-based decoding, a +LM item has the form $(v^{a})$ where $a$ is the last word of the hypothesis. Thus a +LM version of Deduction (1) might be:

$$\frac{(\_\bullet\bullet\bullet^{\text{talk}}) : (w, \text{"held a talk"})}{(\bullet\bullet\bullet\bullet\bullet^{\text{Sharon}}) : (w', \text{"held a talk with Sharon"})}$$

---

search is **not** required by perceptron convergence. All we need is that each update involves a "violation", i.e., the 1-best sequence has a higher model score than the correct sequence. Such an update is considered a "valid update", and any perceptron variant that maintains this is bound to converge. We call these variants "violation-fixing perceptrons" (Section 3.1).

- This theory explains why standard perceptron update may fail to work with inexact search, because violation is no longer guaranteed: the correct structure might indeed be preferred by the model, but was pruned during the search process (Sec. 3.2). Such an update is thus considered invalid, and experiments show that invalid updates lead to bad learning (Sec. 6.2).
- We show that the early update is always valid and is thus a special case in our framework; this is the first theoretical justification for early update (Section 4). We also show that (a variant of) LaSO (Daumé and Marcu, 2005) is another special case (Section 7).
- We then propose several other update methods within this framework (Section 5). Experiments in Section 6 confirm that among them, the max-violation method can learn equal or better models with dramatically reduced learning times (by 3 fold as compared to early update) on state-of-the-art part-of-speech tagging (Collins, 2002)[1] and incremental parsing (Huang and Sagae, 2010) systems. We also found strong correlation between search error and invalid updates, suggesting that the advantage of valid update methods is more pronounced with harder inference problems.

Our techniques are widely applicable to other structured prediction problems which require inexact search like machine translation and protein folding.

## 2 Structured Perceptron

We review the convergence properties of the standard structured perceptron (Collins, 2002) in our own notations that will be reused in later sections for non-exact search. We first define a new concept:

[1] Incidentally, we achieve the best POS tagging accuracy to date (97.35%) on English Treebank by early update (Sec. 6.1).

**Algorithm 1** Structured Perceptron (Collins, 2002).

Input: data $D = \{(x^{(t)}, y^{(t)})\}_{t=1}^{n}$ and feature map $\mathbf{\Phi}$
Output: weight vector $\mathbf{w}$
Let: $\mathrm{EXACT}(x, \mathbf{w}) \triangleq \mathrm{argmax}_{s \in \mathcal{Y}(x)}\ \mathbf{w} \cdot \mathbf{\Phi}(x, s)$
Let: $\Delta\mathbf{\Phi}(x, y, z) \triangleq \mathbf{\Phi}(x, y) - \mathbf{\Phi}(x, z)$
1: **repeat**
2: **for each** example $(x, y)$ in $D$ **do**
3: $z \leftarrow \mathrm{EXACT}(x, \mathbf{w})$
4: **if** $z \neq y$ **then**
5: $\mathbf{w} \leftarrow \mathbf{w} + \Delta\mathbf{\Phi}(x, y, z)$
6: **until** converged

**Definition 1.** The **standard confusion set** $C_s(D)$ for training data $D = \{(x^{(t)}, y^{(t)})\}_{t=1}^{n}$ is the set of triples $(x, y, z)$ where $z$ is a wrong label for input $x$:

$$C_s(D) \triangleq \{(x, y, z) \mid (x, y) \in D, z \in \mathcal{Y}(x) - \{y\}\}.$$

The rest of the theory, including separation and violation, all builds upon this concept. We call such a triple $S = \langle D, \mathbf{\Phi}, C\rangle$ a **training scenario**, and in the remainder of this section, we assume $C = C_s(D)$, though later we will define other confusion sets to accommodate other update methods.

**Definition 2.** The training scenario $S = \langle D, \mathbf{\Phi}, C\rangle$ is said to be **linearly separable** (i.e., dataset $D$ is linearly separable in $C$ by representation $\mathbf{\Phi}$) with **margin** $\delta > 0$ if there exists an *oracle vector* $\mathbf{u}$ with $\|\mathbf{u}\| = 1$ such that it can correctly classify all examples in $D$ (with a gap of at least $\delta$), i.e., $\forall (x, y, z) \in C, \mathbf{u} \cdot \Delta\mathbf{\Phi}(x, y, z) \geq \delta$. We define the *maximal margin* $\delta(S)$ to be the maximal such margin over all unit oracle vectors:

$$\delta(S) \triangleq \max_{\|\mathbf{u}\|=1} \min_{(x,y,z) \in C} \mathbf{u} \cdot \Delta\mathbf{\Phi}(x, y, z).$$

**Definition 3.** A triple $(x, y, z)$ is said to be a **violation** in training scenario $S = \langle D, \mathbf{\Phi}, C\rangle$ with respect to weight vector $\mathbf{w}$ if $(x, y, z) \in C$ **and** $\mathbf{w} \cdot \Delta\mathbf{\Phi}(x, y, z) \leq 0$.

Intuitively, this means model $\mathbf{w}$ is possible to mislabel example $x$ (though not necessarily to $z$) since $y$ is not its single highest scoring label under $\mathbf{w}$.

**Lemma 1.** *Each update triple $(x, y, z)$ in Algorithm 1 (line 5) is a violation in $S = \langle D, \mathbf{\Phi}, C_s(D)\rangle$.*

# Structure

- Abstract (4 sentences)
- Introduction (1 page)
- The problem (1 page)
- My idea (2 pages)
- The details (5 pages)
- Related work (1-2 pages)
- Conclusions and further work (0.5 pages)

# Presenting the idea

**3. The idea**

Consider a bifurcated semi-lattice D, over a hyper-modulated signature S. Suppose $p_i$ is an element of D. Then we know for every such $p_i$ there is an epi-modulus j, such that $p_j < p_i$.

- Sounds impressive...but
- Sends readers to sleep
- In a paper you MUST provide the details, but FIRST convey the idea

# Presenting the idea

- Explain it as if you were speaking to someone using a whiteboard
- **Conveying the intuition is primary**, not secondary
- Once your reader has the intuition, she can follow the details (but not vice versa)
- Even if she skips the details, she still takes away something valuable

# Putting the reader first

- **Do not** recapitulate your personal journey of discovery.  This route may be soaked with your blood, but that is not interesting to the reader.

- Instead, choose the most direct route to the idea.

# The payload of your paper

Introduce the problem, and your idea, using

**EXAMPLES**

and only then present the general case

# Using examples

The Simon PJ question: is there any typewriter font?

## 2 Background

To set the scene for this paper, we begin with a brief overview of the *Scrap your boilerplate* approach to generic programming. Suppose that we want to write a function that computes the size of an arbitrary data structure. The basic algorithm is "for each node, add the sizes of the children, and add 1 for the node itself". Here is the entire code for `gsize`:

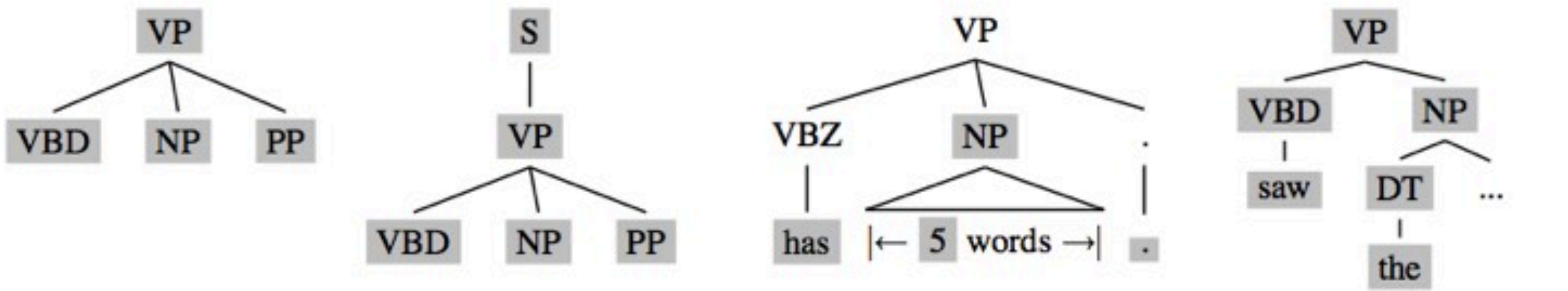
```
gsize :: Data a => a -> Int
gsize t = 1 + sum (gmapQ gsize t)
```

Example right away

The type for `gsize` says that it works over any type `a`, provided `a` is a *data* type — that is, that it is an instance of the class `Data`[1] The definition of `gsize` refers to the operation `gmapQ`, which is a method of the `Data` class:

```
class Typeable a => Data a where
  ...other methods of class Data...
  gmapQ :: (forall b. Data b => b -> r) -> a -> [r]
```

# Examples and Illustrations




(a) **Rule** (local) ⟨ VP → VBD NP PP ⟩

(b) **ParentRule** (non-local) ⟨ VP → VBD NP PP | S ⟩

(c) **WordEdges** (local) ⟨ NP 5 has . ⟩

(d) **NGramTree** (non-local) ⟨ VP (VBD saw) (NP (DT the)) ⟩

Figure 2: Illustration of some example features. Shaded nodes denote information included in the feature.

tags, which are generated dynamically.

More formally, we split the feature extractor $\mathbf{f} = (f_1, \ldots, f_d)$ into $\mathbf{f} = (\mathbf{f}_L; \mathbf{f}_N)$ where $\mathbf{f}_L$ and $\mathbf{f}_N$ are the local and non-local features, respectively. For the former, we extend their domains from parses to hyperedges, where $f(e)$ returns the value of a local feature $f \in \mathbf{f}_L$ on hyperedge $e$, and its value on a parse $y$ factors across the hyperedges (local productions),

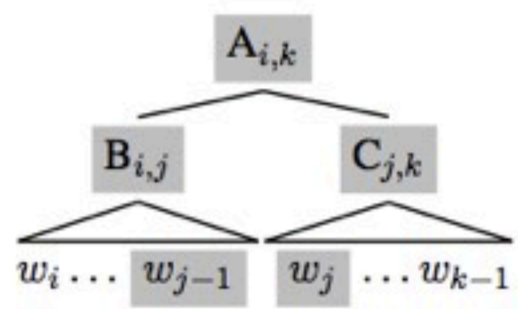



Figure 3: Example of the unit **NGramTree** feature at node $A_{i,k}$: ⟨ A (B $\ldots w_{j-1}$) (C $\ldots w_j$) ⟩.

# Examples and Illustrations

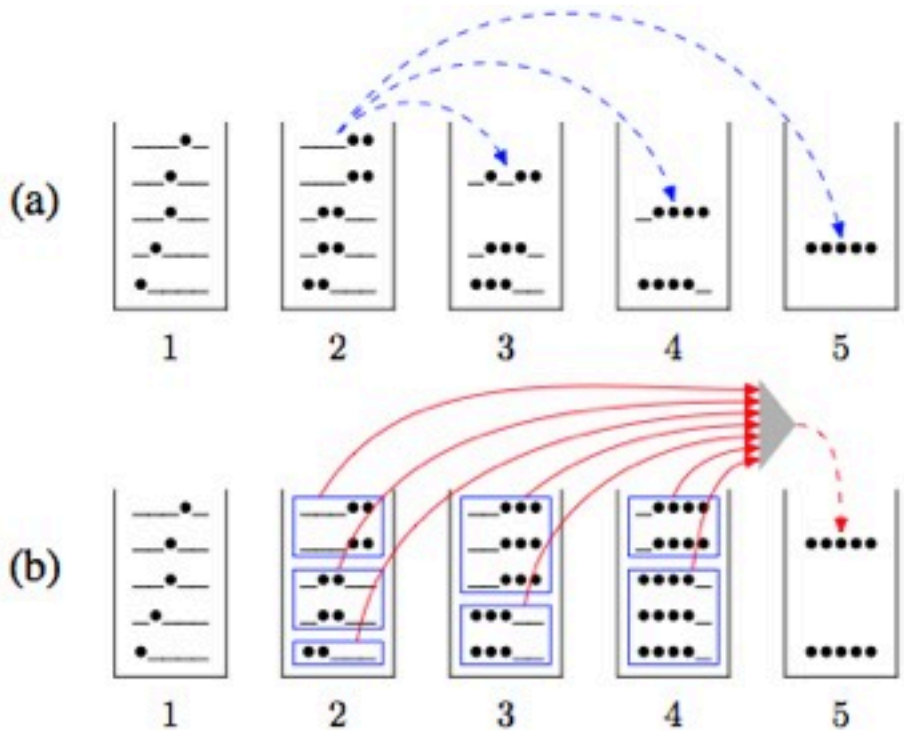



Figure 5: (a) Pharaoh expands the hypotheses in the current bin (#2) into longer ones. (b) In Cubit, hypotheses in previous bins are fed via hyperedge bundles (solid arrows) into a priority queue (shaded triangle), which empties into the current bin (#5).

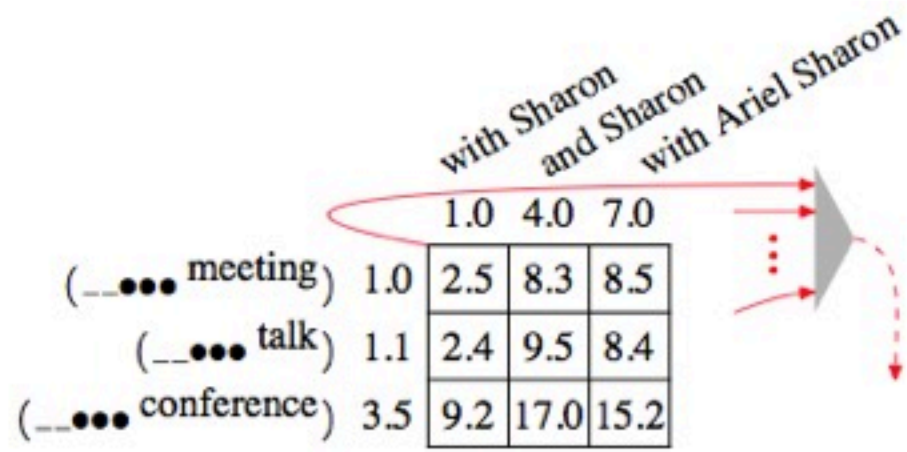



Figure 6: A hyperedge bundle represents all +LM deductions that derives an item in the current bin from the same coverage vector (see Figure 5). The phrases on the top denote the target-sides of applicable phrase-pairs sharing the same source-side.

### 5.1 Phrase-based Decoding

We implemented *Cubit*, a Python clone of the Pharaoh decoder (Koehn, 2004),[3] and adapted cube pruning to it as follows. As in Pharaoh, each bin

# Visual structure

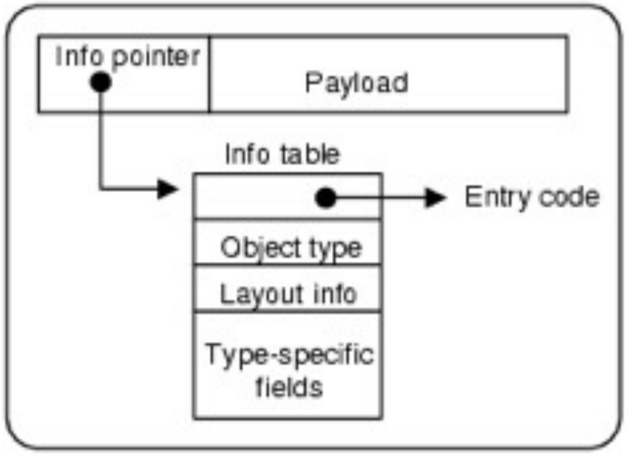



**Figure 3. A heap object**

The three cases above do not exhaust the possible forms of $f$. It might also be a *THUNK*, but we have already dealt with that case (rule THUNK). It might be a *CON*, in which case there cannot be any pending arguments on the stack, and rules UPDATE or RET apply.

### 4.3 The eval/apply model

The last block of Figure 2 shows how the eval/apply model deals with function application. The first three rules all deal with the case of a *FUN* applied to some arguments:

- If there are exactly the right number of arguments, we behave exactly like rule KNOWNCALL, by tail-calling the function. Rule EXACT is still necessary — and indeed has a direct counterpart in the implementation — because the function might not be statically known.
- If there are too many arguments, rule CALLK pushes a *call*

remainder of the object is called the *payload*, and may consist of a mixture of pointers and non-pointers. For example, the object $CON(C\ a_1 \ldots a_n)$ would be represented by an object whose info pointer represented the constructor $C$ and whose payload is the arguments $a_1 \ldots a_n$.

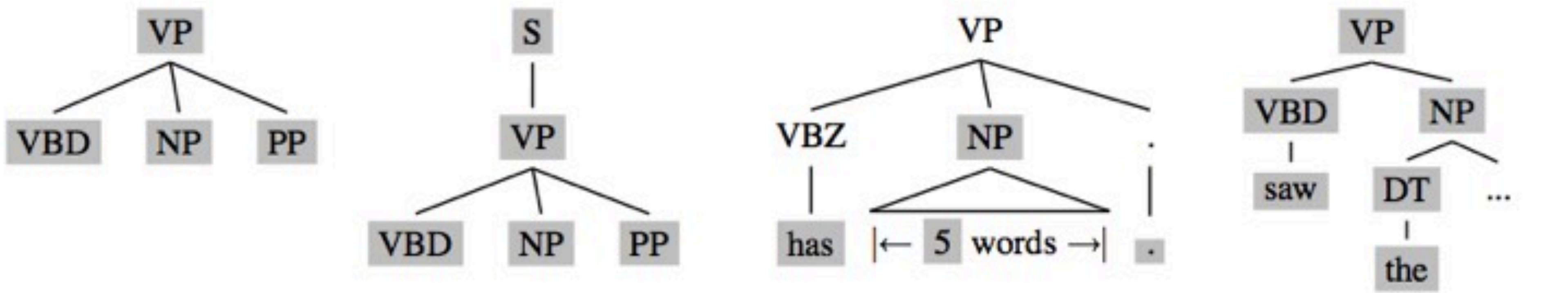
The info table contains:

- Executable code for the object. For example, a *FUN* object has code for the function body.
- An object-type field, which distinguishes the various kinds of objects (*FUN*, *PAP*, *CON* etc) from each other.
- Layout information for garbage collection purposes, which describes the size and layout of the payload. By "layout" we mean which fields contain pointers and which contain non-pointers, information that is essential for accurate garbage collection.
- Type-specific information, which varies depending on the object type. For example, a *FUN* object contains its arity; a *CON* object contains its constructor tag, a small integer that distinguishes the different constructors of a data type; and so on.

In the case of a PAP, the size of the object is not fixed by its info table; instead, its size is stored in the object itself. The layout of its fields (e.g. which are pointers) is described by the (initial segment of) an argument-descriptor field in the info table of the FUN object which is always the first field of a PAP. The other kinds of heap object all have a size that is statically fixed by their info table.

A very common operation is to jump to the entry code for the object, so GHC uses a slightly-optimised version of the representation in Figure 3. GHC places the info table at the addresses *immediately*

# Visual Structure -- Breathe!




(a) **Rule** (local) ⟨ VP → VBD NP PP ⟩

(b) **ParentRule** (non-local) ⟨ VP → VBD NP PP | S ⟩

(c) **WordEdges** (local) ⟨ NP 5 has . ⟩

(d) **NGramTree** (non-local) ⟨ VP (VBD saw) (NP (DT the)) ⟩

Figure 2: Illustration of some example features. Shaded nodes denote information included in the feature.

tags, which are generated dynamically.

More formally, we split the feature extractor $\mathbf{f} = (f_1, \ldots, f_d)$ into $\mathbf{f} = (\mathbf{f}_L; \mathbf{f}_N)$ where $\mathbf{f}_L$ and $\mathbf{f}_N$ are the local and non-local features, respectively. For the former, we extend their domains from parses to hyperedges, where $f(e)$ returns the value of a local feature $f \in \mathbf{f}_L$ on hyperedge $e$, and its value on a parse $y$ factors across the hyperedges (local productions),

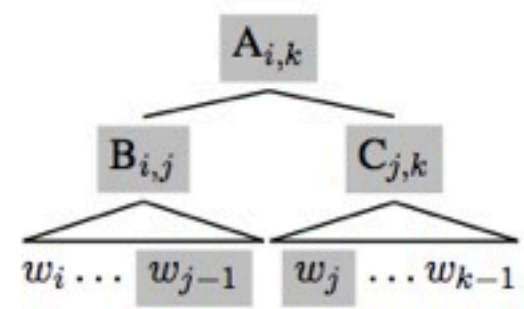



Figure 3: Example of the unit **NGramTree** feature at node $A_{i,k}$: ⟨ A (B $\ldots w_{j-1}$) (C $\ldots w_j$) ⟩.

# Visual Structure -- Breathe!

**Abstract**

Most current parameter tuning methods for machine translation (such as MERT and PRO) are *agnostic* about search, while search errors are well-known to adversely affect translation quality. We propose to promote potentially accurate partial translations and prevent them from being pruned, and develop two metrics to evaluate partial derivations. Our method can be applied to all of the three most popular tuning algorithms: MERT, PRO, and MIRA, where extensive experiments on Chinese-to-English and English-to-Chinese translation show up to +2.6 BLEU gains with each of the three algorithms.

**1 Introduction**

Parameter tuning has been an active area of research in machine translation. However, most of the existing tuning algorithms only compare complete translations (Och, 2003; Hopkins and May, 2011; Chiang, 2012), while many potentially "promising" partial translations are pruned by the search algorithm in the prohibitively large search space. For example, the popular beam-search decoding algorithm for phrase-based MT (Koehn, 2004) only explores $O(nb)$ items for a sentence of $n$ words (with a beam width of $b$), while the full search space is $O(2^n n^2)$ or worse (Knight, 1999).

As one of the few exceptions to the "search-agnostic" majority, Yu et al. (2013) and Zhao et al. (2014) propose a variant of the perceptron algorithm that learns to keep the reference translations in the beam or chart. However, there are several obstacles that prevent their method from becoming popular: First of all, they rely on "forced decoding" to track

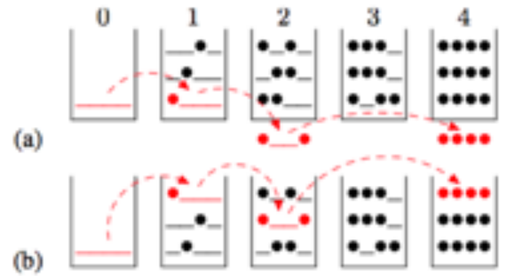



Figure 1: (a) Some potentially promising partial translations (in red) fall out of the beam (bin 2); (b) We identify such partial translations and assign them higher model scores so that they are more likely to survive the search.

gold derivations that lead to the reference translation, but in practice only a small portion (around 30% in their experiments) of (mostly very short) sentence pairs have at least one such derivation. Secondly, they learn the model on the training set, and while this does enable a huge feature set, it is much too slow compared to MERT and PRO (by at least an order of magnitude).

We instead propose a very simple framework, **search-aware tuning**, which does not depend on forced decoding, and thus can be trained on all sentence pairs of any dataset. The key idea of our new approach is that, besides caring about the rankings of the complete translations, we also promote potentially promising partial translations so that they are more likely to survive throughout the search, see **Figure 1** for illustration. We make the following contributions:

- Our idea of search-aware tuning can be applied (as a patch) to all the three most popular tuning methods (MERT, PRO, and MIRA) by defining a modified objective function (Section 4).
- To measure the "promise" or "potential" of a

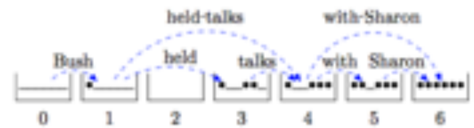



Figure 2: Forced decoding and $y$-good derivation lattice.

where the score of applying each rule now also includes a *combination cost* due to the bigrams formed when applying the phrase-pair, e.g.

$$s'_3 = s'_2 + s(r_3) + dc(|6-3|) - \log P_{\text{lm}}(\text{with} \mid \text{talk})$$

To make this exponential-time algorithm practical, beam search is the standard approximate search method (Koehn, 2004). Here we group +LM states into $n$ bins, with each bin $B_i$ hosting at most $b$ states that cover exactly $i$ Chinese words (see Figure 1).

**2.2 Forced Decoding**

The idea of forced decoding is to consider only those (partial) derivations that can produce (a prefix of) the exact reference translation (assuming single reference). We call these partial derivations "$y$-good" derivations (Daumé, III and Marcu, 2005), and those that deviate from the reference translation "$y$-bad" derivations. The forced decoding algorithm is very similar to +LM decoding introduced above, with the new "forced decoding LM" to be defined as only accepting two consecutive words on the reference translation, ruling out any $y$-bad hypothesis:

$$P_{\text{forced}}(b \mid a) = \begin{cases} 1 & \text{if } \exists j, \text{ s.t. } a = y_j \text{ and } b = y_{j+1} \\ 0 & \text{otherwise} \end{cases}$$

In the +LM state, we can simply replace the boundary word by the index on the reference translation:

$$\cfrac{\cfrac{\cfrac{(\_\_\_\_\_\_\_, 0) : (0, \text{"<s>"})}{(\bullet_1\_\_\_\_\_\_, 1) : (w'_1, \text{"<s> Bush"})}\, r_1}{(\bullet\_\_\bullet\bullet\bullet_6, 3) : (w'_2, \text{"<s> Bush held talks"})}\, r_2}{(\bullet\bullet\bullet_3\bullet\bullet\bullet, 5) : (w'_3, \text{"<s> Bush held talks with Sharon"})}\, r_3$$

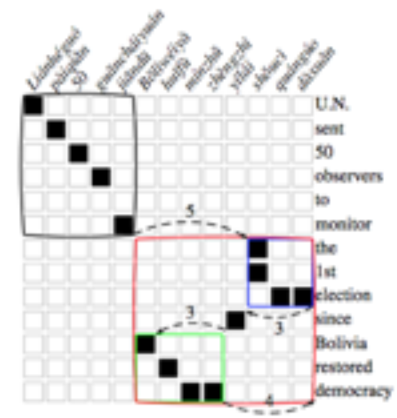



Figure 3: Example of unreachable sentence pair and reachable prefix-pair. The first big jump is disallowed for a distortion limit of 4, but we can still extract the top-left box as a reachable prefix-pair. Note that this example is perfectly reachable in syntax-based MT.

**2.3 Reachable Prefix-Pairs**

In practice, many sentence pairs in the parallel text fail in forced decoding due to two reasons:

1. **distortion limit**: long-distance reorderings are disallowed but are very common between languages with very different word orders such as English and Chinese.
2. **noisy alignment and phrase limit**: the word-alignment quality (typically from GIZA++) are usually very noisy, which leads to unnecessarily big chunks of rules beyond the phrase limit.

If we only rely on the reachable whole sentence pairs, we will not be able to use much of the training set for Chinese-English. So we propose to augment the set of reachable examples by considering reachable prefix-pairs (see Figure 3 for an example).

**3 Violation-Fixing Perceptron for MT**

Huang et al. (2012) establish a theoretical framework called "violation-fixing perceptron" which is tailored for structured learning with inexact search

# Visual Structure -- non-text > text

- "Remember to think of the paper as a collection of experimental results, summarized as clearly and economically as possible in figures, tables, equations, and schemes. The text in the paper serves just to explain the data, and is secondary. The more information can be compressed into tables, equations, etc., the shorter and more readable the paper will be." -- George Whitesides
- Much of CS is *not* an experimental science, but you can still think of a paper as a collection of ideas, examples, algorithms and pseudocode, diagrams, definitions, theorems, proofs, plots, and tables.
- focus on the non-text parts and write text just to explain them

# Visual Structure -- Paper Gestalt

- scientific evidences from CVPR 2010 - "paper gestalt"
  - a paper's fate (acceptance/rejection) can largely be determined by its visual features (layout) alone!

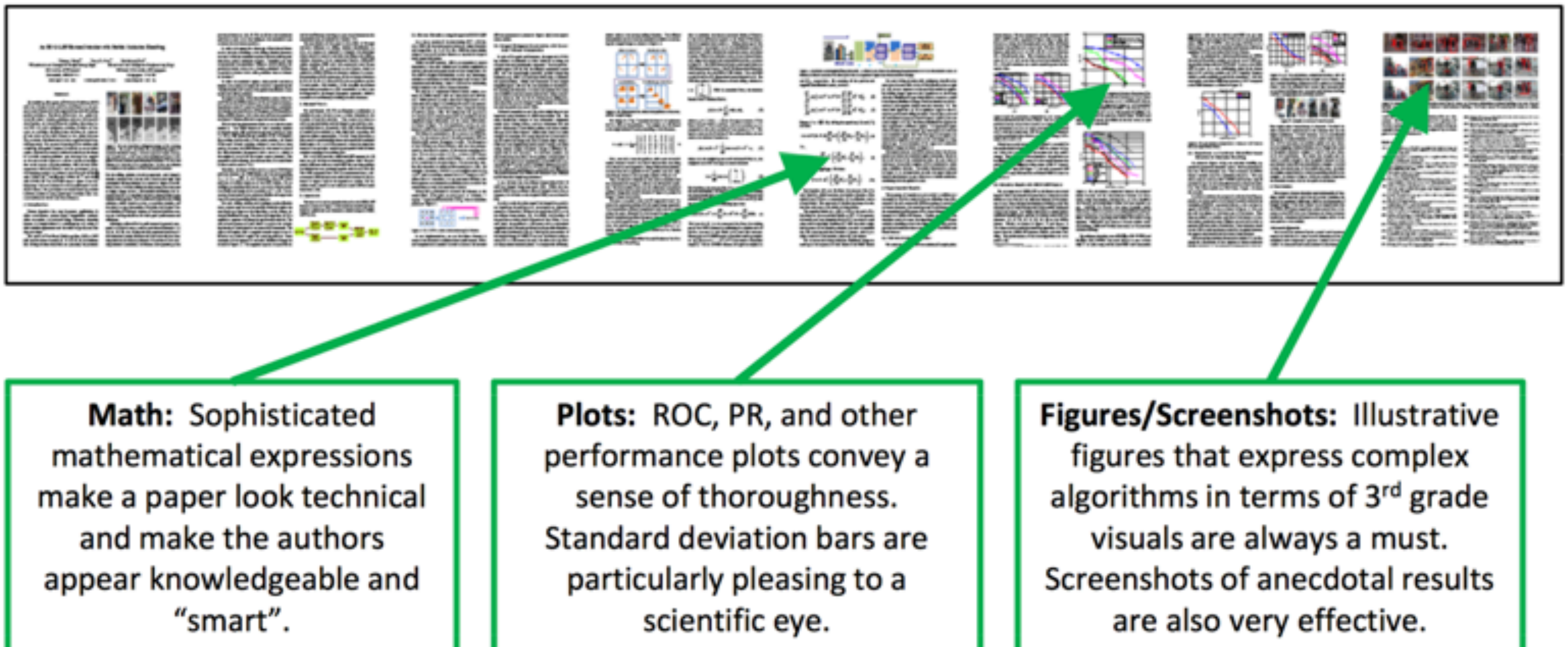



Figure 6. **Characteristics of a "Good" paper.**

Thanks to Jian Cheng and Junliang Xing for suggesting "paper gestalt".

# Visual Structure -- Paper Gestalt

- scientific evidences from CVPR 2010 - "paper gestalt"
  - a paper's fate (acceptance/rejection) can largely be determined by its visual features (layout) alone!

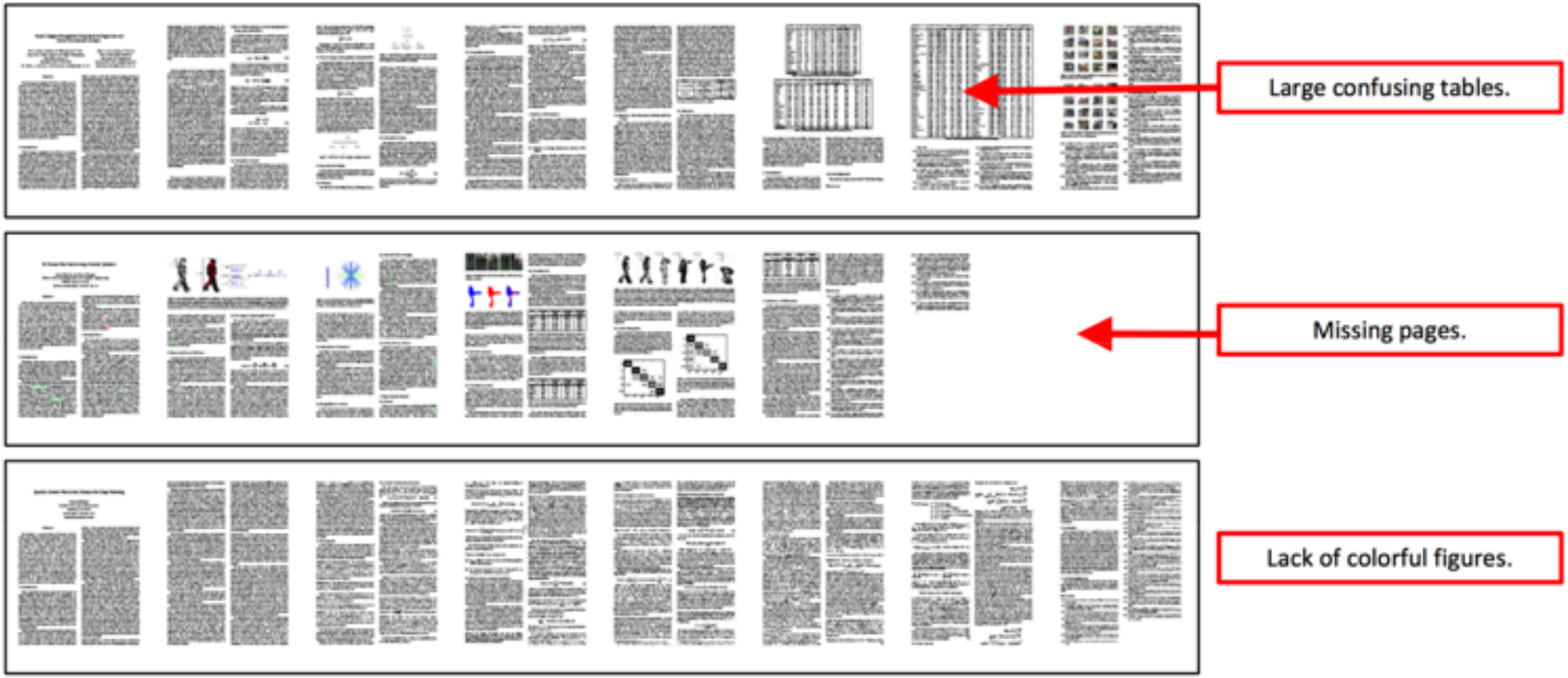



Figure 7. **Characteristics of a "Bad" paper.**

Thanks to Jian Cheng and Junliang Xing for suggesting "paper gestalt".

# Visual Structure -- Paper Gestalt

- scientific evidences from CVPR 2010 - “paper gestalt”
    - a paper’s fate (acceptance/rejection) can largely be determined by its visual features (layout) alone!

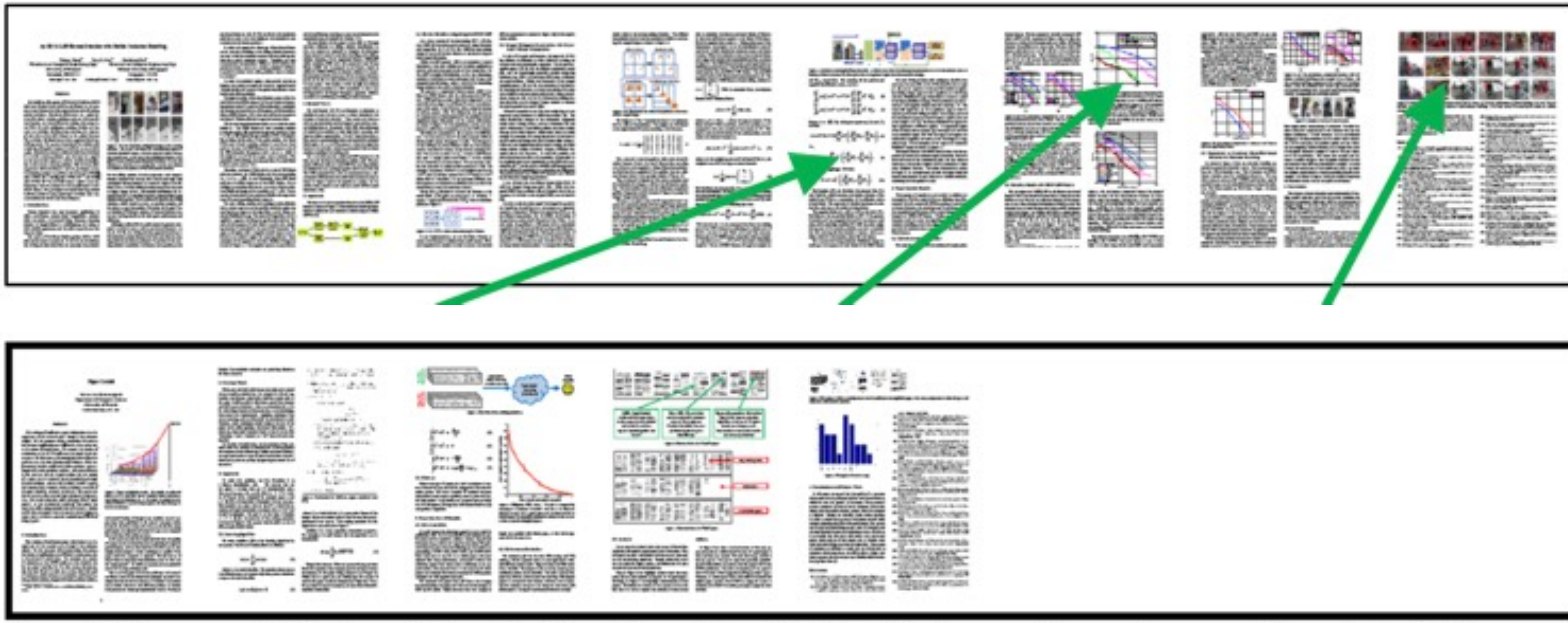


Figure 8. **Our paper**. While it certainly suffers from the problem of missing/blank pages, it has a nice composition of colorful figures and impressive mathematical equations.

Thanks to Jian Cheng and Junliang Xing for suggesting “paper gestalt”.

# The details: evidence

- Your introduction makes claims
- The body of the paper provides **evidence to support each claim**
- Check each claim in the introduction, identify the evidence, and forward-reference it from the claim
- Evidence can be: analysis and comparison, theorems, measurements, case studies

# Structure

- Abstract (4 sentences)
- Introduction (1 page)
- The problem (1 page)
- My idea (2 pages)
- The details (5 pages)
- **Related work** (1-2 pages)
- Conclusions and further work (0.5 pages)

# Related work

Fallacy

To make my work look good, I have to make other people's work look bad

# The truth: credit is not like money

Giving credit to others does not diminish the credit you get from your paper

- Warmly acknowledge people who have helped you
- Be generous to the competition. "In his inspiring paper [Foo98] Foogle shows.... We develop his foundation in the following ways..."
- Acknowledge weaknesses in your approach

# Credit is not like money

Failing to give credit to others can kill your paper

If you imply that an idea is yours, and the referee knows it is not, then either

- You don't know that it's an old idea (bad)
- You do know, but are pretending it's yours (very bad)

# Structure

- Abstract (4 sentences)
- Introduction (1 page)
- The problem (1 page)
- My idea (2 pages)
- The details (5 pages)
- Related work (1-2 pages)
- Conclusions and further work (0.5 pages)

# Conclusions and further work

- Be brief.

# The process of writing

# Writing papers: model 1

Idea → Do research → Write paper

# Writing papers: model 2

J. Eisner

~~Idea → Do research → Write paper~~

Idea → Write paper → Do research

*LH: proposal*

- Forces us to be clear, focused
- Crystallises what we don't understand
- Opens the way to dialogue with others: reality check, critique, and collaboration

# Do not be intimidated

**Fallacy** You need to have a fantastic idea before you can write a paper. (Everyone else seems to.)

Write a paper,
and give a talk, about
**any idea**,
no matter how weedy and insignificant it may seem to you

# Do not be intimidated

Write a paper, and give a talk, about any idea, no matter how insignificant it may seem to you

- **Writing the paper is how you develop the idea in the first place**
- It usually turns out to be more interesting and challenging that it seemed at first

LH: talk, write as early as you can;
don't wait until you feel ready;
it doesn't mean you have to publish it.

# The process

- Start early.  Very early.
  - Hastily-written papers get rejected.
  - Papers are like wine: they need time to mature
- Collaborate
- Use CVS to support collaboration

# Getting help

Get your paper read by as many friendly guinea pigs as possible

- Experts are good
- Non-experts are also very good
- Each reader can only read your paper for the first time once! So use them carefully
- Explain carefully what you want ("I got lost here" is much more important than "Jarva is mis-spelt".)

# Listening to your reviewers

**Treat every review like gold dust**

Be (truly) grateful for criticism as well as praise

This is really, really, really hard

But it's

really, really, really, really, really, really, really, really, really, really

important

# Listening to your reviewers

- Read every criticism as a positive suggestion for something you could explain more clearly
- DO NOT respond “you stupid person, I meant X”. Fix the paper so that X is apparent even to the stupidest reader.
- Thank them warmly. They have given up their time for you.

# Language and style

# Use the active voice

The passive voice is "respectable" but it DEADENS your paper. Avoid it at all costs.

| NO | YES |
|---|---|
| It can be seen that... | We can see that... |
| 34 tests were run | We ran 34 tests |
| These properties were thought desirable | We wanted to retain these properties |
| It might be thought that this would be a type error | You might think this would be a type error |

"We" = you and the reader

"We" = the authors

"You" = the reader

# Even Newton used the active voice!

- I held the Prism.
- I looked through the Prism
- I stopt the Prism
- I observed the length of its refracted Image
- I removed the Prism out of the Sun's Light and looked

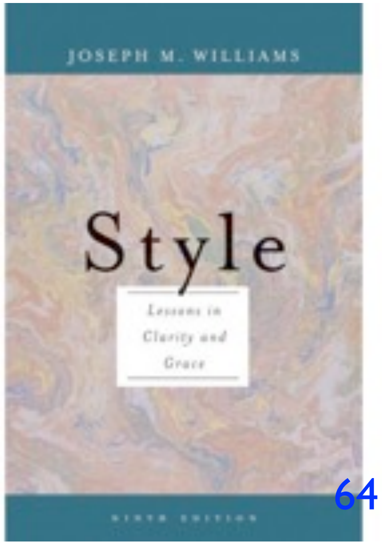
Isaac Newton (1704), Optics.

# Use simple, direct language

| NO | YES |
|---|---|
| The object under study was displaced horizontally | The ball moved sideways |
| On an annual basis | Yearly |
| Endeavour to ascertain | Find out |
| It could be considered that the speed of storage reclamation left something to be desired | The garbage collector was really slow |

# Resources for the Writing Part

- writing resources: http://www.cis.upenn.edu/~lhuang3/writing/
- high-level (language-independent)
  - Simon Peyton-Jones: *How to Write a Research Paper*
  - Mark-Jan Nederhof: *Common Pitfalls in Academic Writing*
- low-level (language-specific -- use NLP!)
  - Gopen & Swan: *The Science of Scientific Writing*
  - Williams: *STYLE: Clarity and Grace* series
  - Strunk and White: *The Elements of Style*
  - Cook: *Line by Line*



# How to give a good research talk

Simon Peyton Jones

Microsoft Research, Cambridge

1993 paper joint with
John Hughes (Chalmers),
John Launchbury (Oregon Graduate Institute)

# Your ideal audience...

The audie… you would lik…

- Have read … your ea…er papers
- Thoroughly un…and all the relevant theory of cartesian … endomorphic bifunctors
- Are all … to hear … out the latest develo…ents in your w…
- A… fresh, alert, and read… …r action

# Your actual audience…

The audience you get

- Have never heard of you
- Have heard of bifunctors, but wish they hadn't
- Have just had lunch and are ready

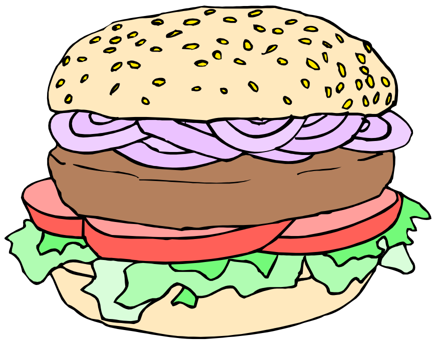
Your mission is to

WAKE THEM UP

And make them glad they did

# What your talk is for

Your paper = **The beef**



Your talk = **The beef advertisement**

**Do not confuse the two**

# The purpose of your talk…

- To give your audience an intuitive feel for your idea
- To make them foam at the mouth with eagerness to read your paper
- To engage, excite, provoke them

“I think the first duty of all art, including fiction of any kind, is to entertain. That is to say, to hold interest. No matter how worthy the message of something, if it's dull, you're just not communicating.” --Poul Anderson

# What to put in

# What to put in

1. Motivation (20%)
2. Your key idea (80%)
3. There is no 3

# Motivation

You have 2 minutes to engage your audience before they start to doze

- Why should I tune into this talk?
- What is the problem?
- Why is it an interesting problem?

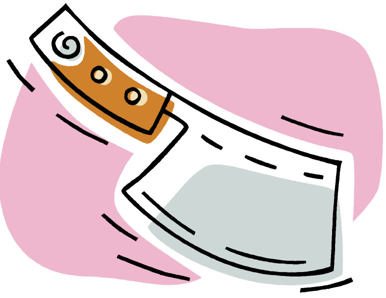
Example: Java class files are large (brief figures), and get sent over the network. Can we use language-aware compression to shrink them?

Example: synchronisation errors in concurrent programs are a nightmare to find. I'm going to show you a type system that finds many such errors at compile time.

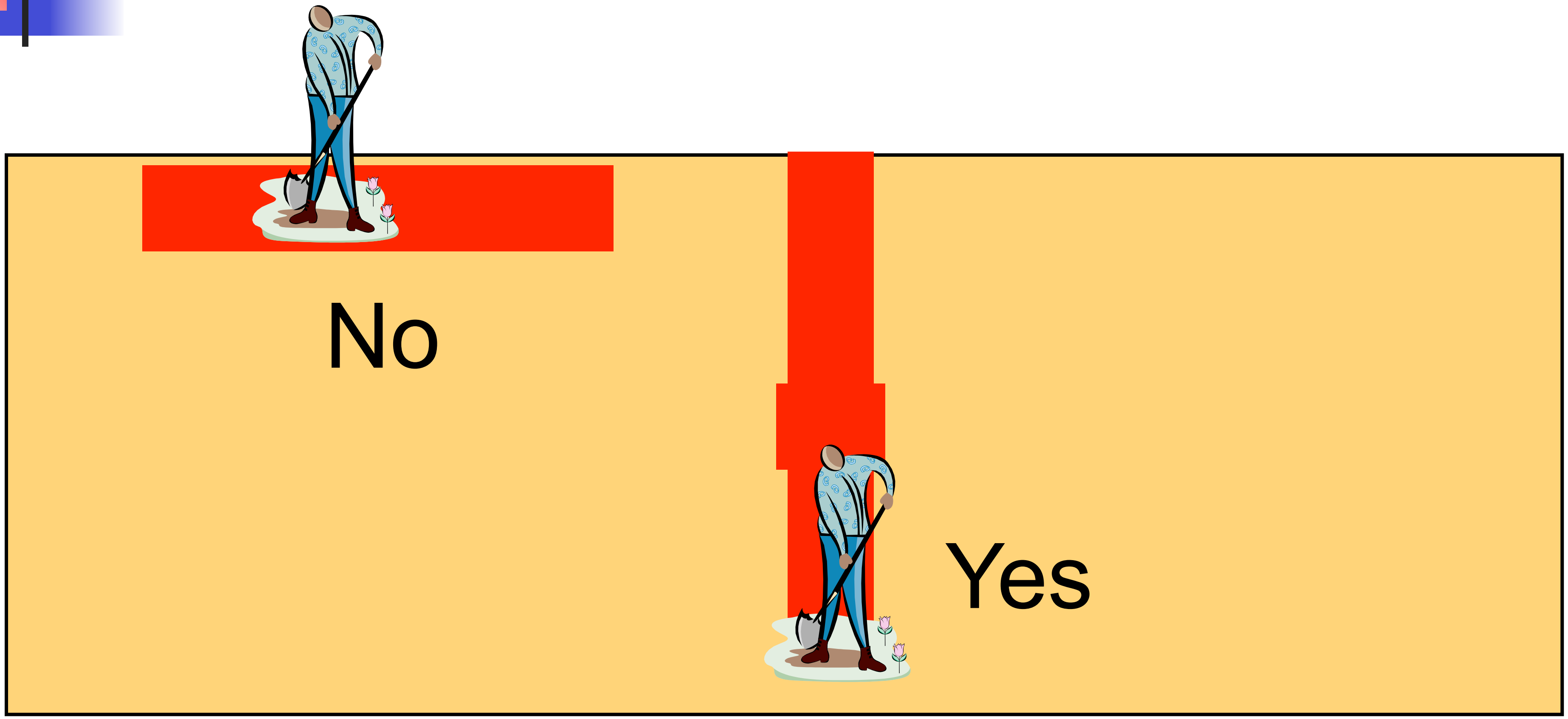
# Your key idea

If the audience remembers only one thing from your talk, what should it be?

- You must identify a key idea. "What I did this summer" is No Good.
- Be specific. Don't leave your audience to figure it out for themselves.
- Be absolutely specific. Say "If you remember nothing else, remember this."
- Organise your talk around this specific goal. Ruthlessly prune material that is irrelevant to this goal.



# Narrow, deep beats wide, shallow




Avoid shallow overviews at all costs

Cut to the chase: the technical "meat"

# Your main weapon 1

## Examples are your main weapon

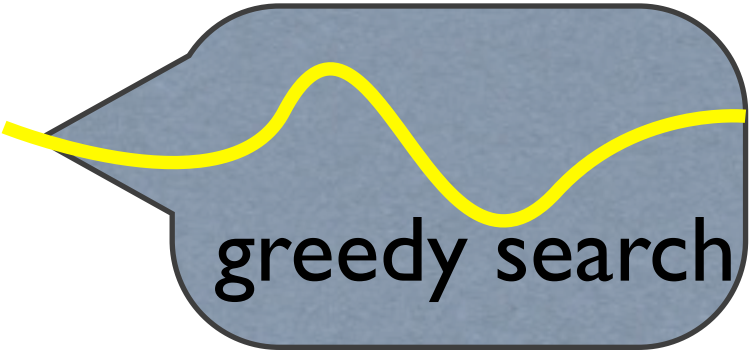
- To motivate the work
- To convey the basic intuition
- To illustrate The Idea in action
- To show extreme cases
- To highlight shortcomings

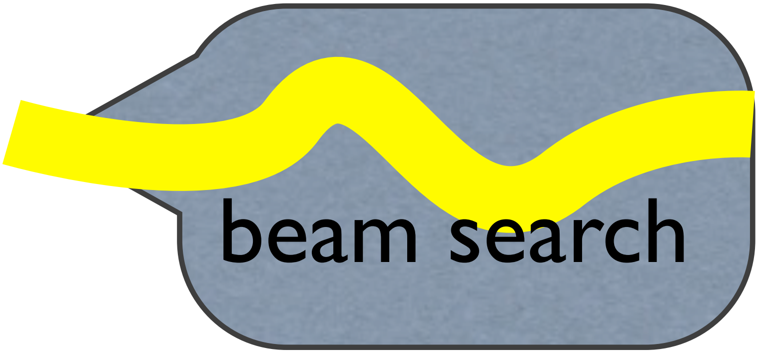
When time is short, omit the general case, not the example

# LH: Your main weapon #2

Visualization!

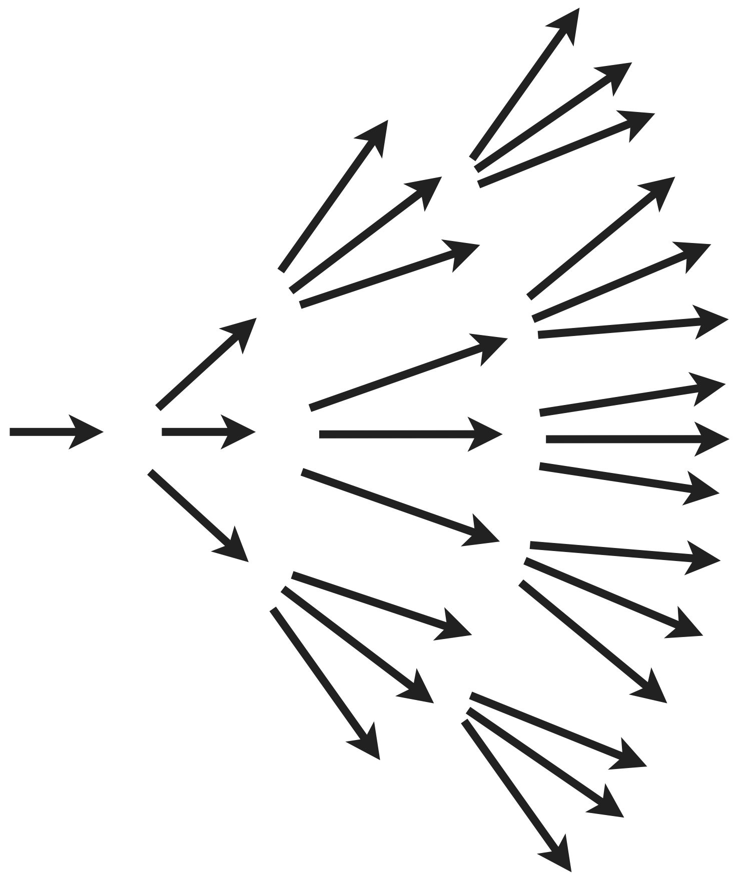
a picture is worth a thousand words!

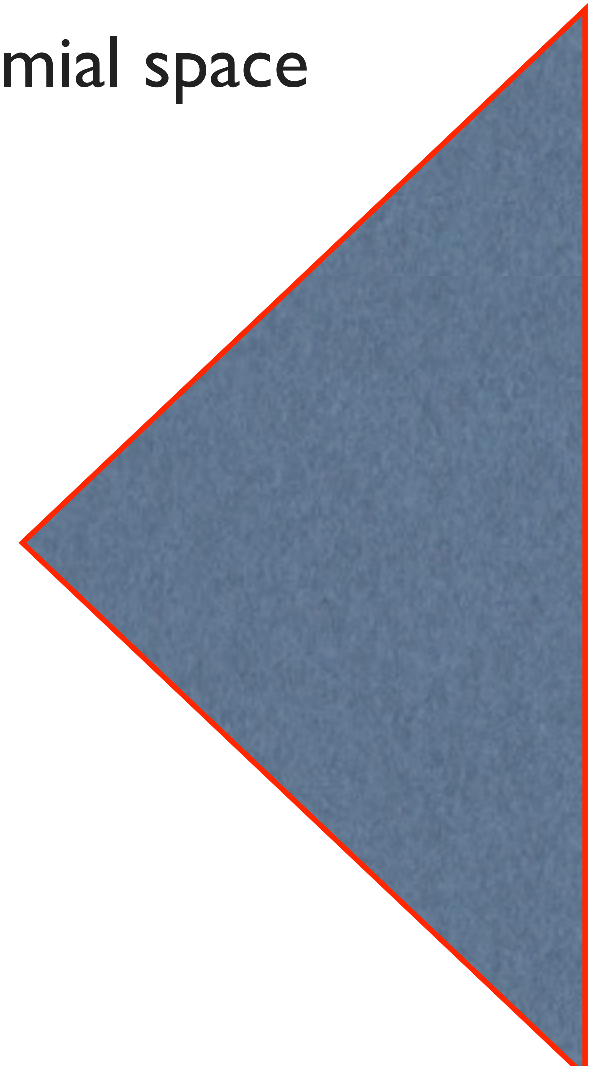


# Example: Dynamic Programming

- each state => three new states (shift, l-reduce, r-reduce)
- key idea of DP: share common subproblems
  - merge equivalent states => polynomial space



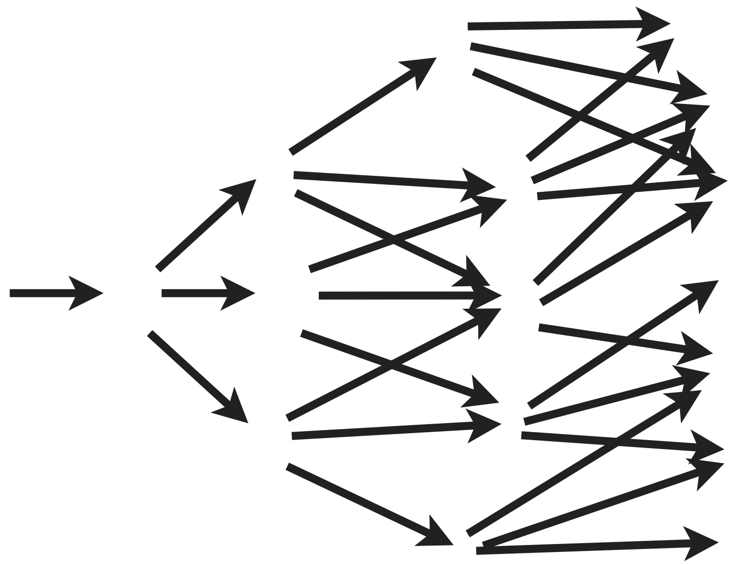


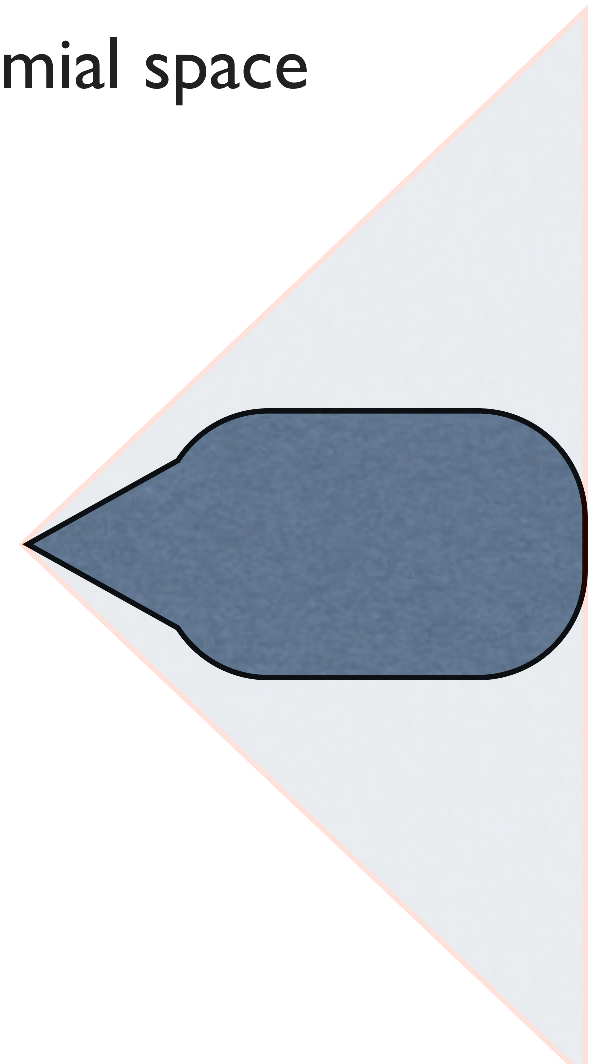


(Huang and Sagae, 2010)

# Example: Dynamic Programming

- each state => three new states (shift, l-reduce, r-reduce)
- key idea of DP: share common subproblems
  - merge equivalent states => polynomial space



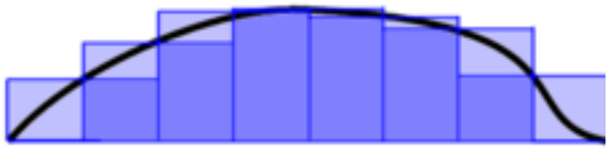


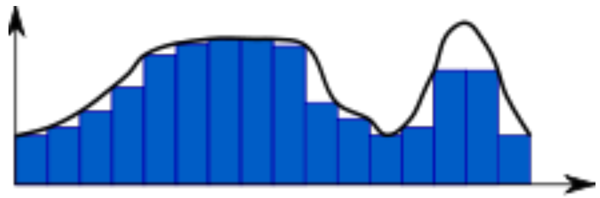


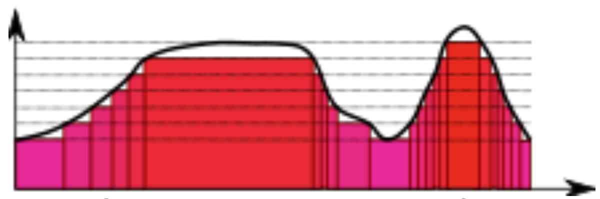
(Huang and Sagae, 2010)

# Real Life Analogy: Lebesgue Integral

- Riemann Integral (Newton-Leibniz Style)
    - intuitive, but left many important functions unintegrable







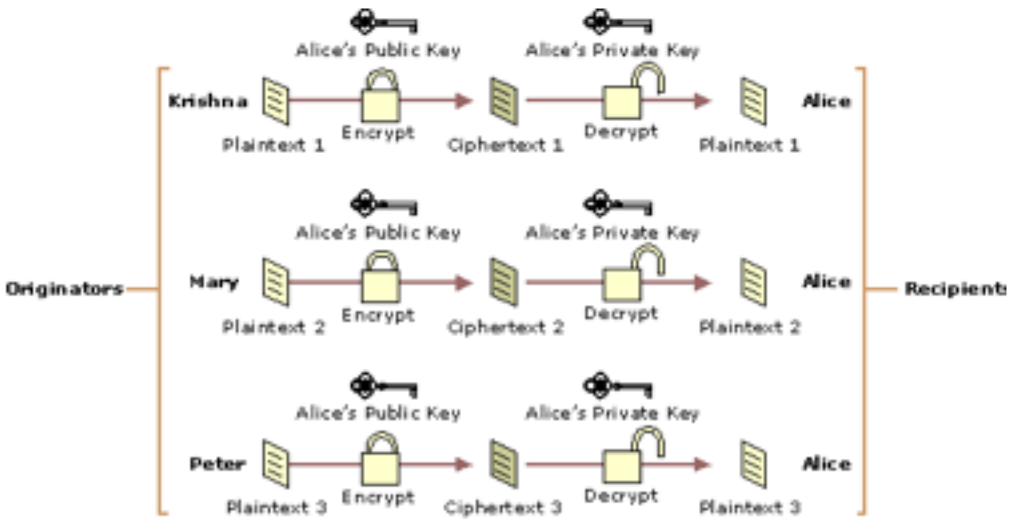
- Lebesgue Integral
    - greatly extended the domain of integrable functions

- Real Life Analogy

Lebesgue to Paul Montel:

I have to pay a certain sum, which I have collected in my pocket. I take the bills and coins out of my pocket and give them to the creditor in the order I find them until I have reached the total sum. This is the Riemann integral. But I can proceed differently. After I have taken all the money out of my pocket I order the bills and coins according to identical values and then I pay the several heaps one after the other to the creditor. This is my integral.—*Source*: (Siegmund-Schultze 2008)

# Real Life Analogy: Public-Key







public key

private key

# What to leave out

# ~~Outline of my talk~~

- Background
- The FLUGOL system
- Shortcomings of FLUGOL
- Overview of synthetic epimorphisms
- $\pi$-reducible decidability of the pseudo-curried fragment under the Snezkovwski invariant in FLUGOL
- Benchmark results
- Related work
- Conclusions and further work

# No outline!

"Outline of my talk": conveys near zero information at the start of your talk

- But you can put up an outline for orientation **after your motivation**
- …and signposts at pause points

# ~~Related work~~

| | |
|---|---|
| [PMW83] | The seminal paper |
| [SPZ88] | First use of epimorphisms |
| [PN93] | Application of epimorphisms to wibblification |
| [BXX98] | Lacks full abstraction |
| [XXB99] | Only runs on Sparc, no integration with GUI |

# Do not present related work

But

- You absolutely must know the related work; respond readily to questions
- Acknowledge co-authors (title slide), and pre-cursors (as you go along)
- Do not disparage the opposition
  - X's very interesting work does Y; I have extended it to do Z

# ~~Technical detail~~

$$\frac{}{\Gamma \vdash k : \tau_k} \qquad \frac{\Gamma \cup \{x : \tau\} \vdash e : \tau'}{\Gamma \vdash \lambda x.e : \tau \to \tau'} \qquad \frac{\Gamma \vdash e_1 : \mathtt{ST}\ \tau^\circ\ \tau \qquad \Gamma \vdash e_2 : \tau \to \mathtt{ST}\ \tau^\circ\ \tau'}{\Gamma \vdash e_1\ \mathtt{>>=}\ e_2 : \mathtt{ST}\ \tau^\circ\ \tau'}$$

$$\frac{\Gamma \vdash e : \tau}{\Gamma \vdash \mathtt{returnST}\ e : \mathtt{ST}\ \tau^\circ\ \tau} \qquad \frac{\Gamma \vdash e : \tau}{\Gamma \vdash \mathtt{newVar}\ e : \mathtt{ST}\ \tau^\circ\ (\mathtt{MutVar}\ \tau^\circ\ \tau)} \qquad \frac{\Gamma \vdash e : \mathtt{MutVar}\ \tau^\circ\ \tau}{\Gamma \vdash \mathtt{readVar}\ e : \mathtt{ST}\ \tau^\circ\ \tau}$$

$$\frac{\Gamma \vdash e_1 : \mathtt{MutVar}\ \tau^\circ\ \tau \qquad \Gamma \vdash e_2 : \tau}{\Gamma \vdash \mathtt{writeVar}\ e_1\ e_2 : \mathtt{ST}\ \tau^\circ\ \mathtt{Unit}} \qquad \frac{}{\Gamma \cup \{x : \forall \alpha_i . \tau\} \vdash x : \tau[\tau_i / \alpha_i]}$$

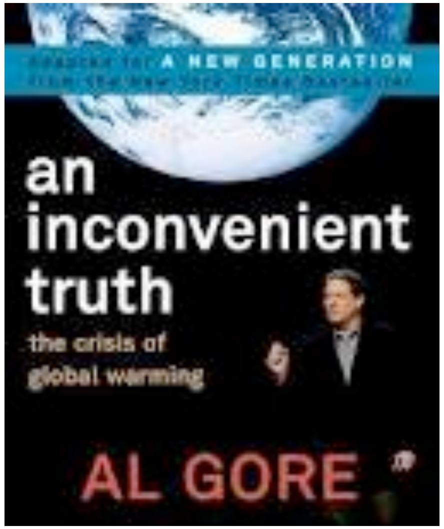
$$\frac{\Gamma \vdash e : \tau' \to \tau \qquad \Gamma \vdash e' : \tau'}{\Gamma \vdash e\ e' : \tau} \qquad \frac{\Gamma \vdash e : \mathtt{ST}\ \alpha^\circ\ \tau}{\Gamma \vdash \mathtt{runST}\ e : \tau}\ \alpha^\circ \notin FV(\Gamma, \tau)$$

$$\frac{\forall j . \Gamma \cup \{x_i : \tau_i\}_i \vdash e_j : \tau_j \qquad \Gamma \cup \{x_i : \forall \alpha_{j_i} . \tau_i\}_i \vdash e' : \tau'}{\Gamma \vdash \mathbf{let}\ \{x_i = e_i\}_i\ \mathbf{in}\ e' : \tau'}\ \alpha_{j_i} \in FV(\tau_i) - FV(\Gamma)$$

**Figure 1.** Typing Rules

# Omit technical details

- Even though every line is drenched in your blood and sweat, dense clouds of notation will send your audience to sleep




- Present specific aspects only; refer to the paper for the details
- By all means have backup slides to use in response to questions

# Presenting your talk

# Polish your slides the night before

(…or at least, polish it then)

Your talk absolutely must be fresh in your mind

- Ideas will occur to you during the conference, as you obsess on your talk during other people's presentations
- Do not use typeset slides, unless you have a laptop too
- Handwritten slides are fine
    - Use permanent ink
    - Get an eraser: toothpaste does not work

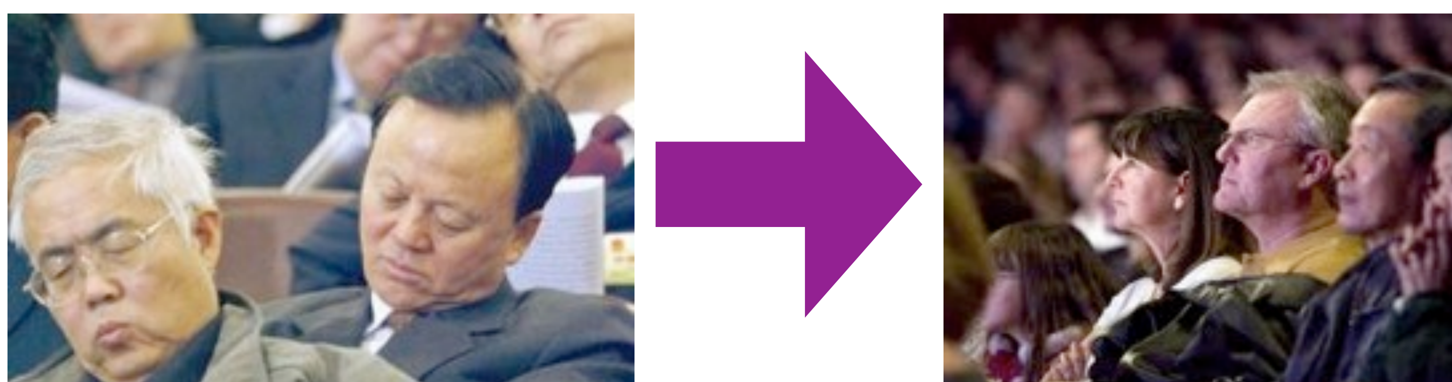
# How to present your talk

By far the most important thing is to

**be enthusiastic**

# Enthusiasm

- If you do not seem excited by your idea, why should the audience be?
- It wakes 'em up
- Enthusiasm makes people dramatically more receptive
- It gets you loosened up, breathing, moving around



# LH: Be Fun -- Three Jokes Rule

- Include as many **relevant** jokes as possible
- three jokes rule
  - one at the beginning (motivation)
  - one at the middle (to wake people up)
  - and one at the end (take-home point)
- especially important in job talks!

# Relevant Jokes: Translation Errors



liang's rule: if you see "X carefully" in China, just don't do it.


# Relevant Jokes: Translation Errors

# Relevant Jokes: Translation Errors


clear evidence that MT is used in real life.

# LH: Use a wireless presenter

- wireless click + laser pointer + [USB disk]
- smoothes your transition!


# Face the audience and Be Active!

- avoid looking and pointing at your laptop
- look at the audience (70%), screen (25%), and laptop (5%)
- do NOT stand still behind the lectern; move around!


# Use Keynote instead of Powerpoint

- PPT sucks (although it's the best software from MSFT)
- Keynote is much more elegant
- LaTeX is OK only for very mathematical talks (eg PL)
  - even there I think Keynote might be better

# Finishing

## Absolutely without fail, finish on time

- Audiences get restive and essentially stop listening when your time is up.  Continuing is very counter productive
- Simply truncate and conclude
- Do not say “would you like me to go on?” (it’s hard to say “no thanks”)

# There is hope

The general standard is so low that you don't have to be outstanding to stand out

You will attend 50x as many talks as you give. Watch other people's talks intelligently, and pick up ideas for what to do and what to avoid.

# Conclusion: Technical Communication

| | | | |
|---|---|---|---|
| interaction | almost zero | a little or a lot | a whole lot |
| technical details | needed | not needed | needed |
| difficulty | hardest | easiest | depends |