

Memory load as a predictor of parsing problems

Eric Auer <eric@coli.uni-sb.de>

March 29, 2002

In this text, I am applying a measure discussed by GLYN MORRILL in his text *Incremental Processing and Acceptability* (2000) on some typical test cases for predicting processing load. I repeat most examples from Morrill (2000), other test cases are collected from other courses which I have followed in the last 12 months. I will discuss the usability of the suggested measure and compare it to a quite generic alternative.

1 Performance measures

1.1 Introduction – Proof nets for CG

Competence and performance on language are often two quite different things: We know how language works (*competence*), but some sentences and constructions are very hard to understand or create – this is due to a lack of *performance* in the human language parser. Sentences that are hard for humans are often not sentences that are hard for processing by computers and vice versa. Psycholinguists are thus interested in predicting the processing load which some parsing task causes for humans. This helps to find out when texts *feel* ungrammatical or are hard to understand. If several ways of expressing one fact exist, we are interested in the *preferred* form both in production and understanding. With understanding we have the additional effect that there may be several ways to understand one and the same utterance, so we want to predict the preferred one (this does not need to be the most plausible one, as we will see below!).

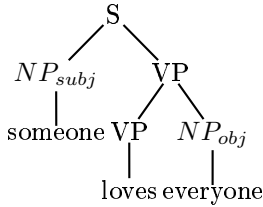
Morrill proposes a measure which can be easily applied in systems using *proof nets*: More information on proof nets can be found for example in *On the Semantic Readings of Proof-Nets* (1996) by DE GROOTE and RETORÉ. A proof net in this context is a graphical way of representing a sentence analysis, based on *Categorial Gram-*

mar and the *Lambek Calculus* (see *The mathematics of sentence structure* (1958) by LAMBEK). As common with CG, most of the proof net is determined by the lexical categories of the processed word: The category of each word is unfolded to a type tree. Parsing (proof search) is then equivalent to finding so-called *axiom linkings* that connect the terminals of the trees with each other. Once this is done in a way that satisfies the wellformedness constraints for proof nets, the semantic reading can be derived directly by following the edges of the proof net and combining the encountered node semantics (see again de Groote and Retoré 1996, or **the lecture notes and slides for this course** (*official name???*)). Both the wellformedness check and the semantics construction are defined on the graphical representation and thus often more appealing to human intuitions than other systems. The article by MORRILL itself gives a short overview of handling proof nets as well, it is quite self-contained.

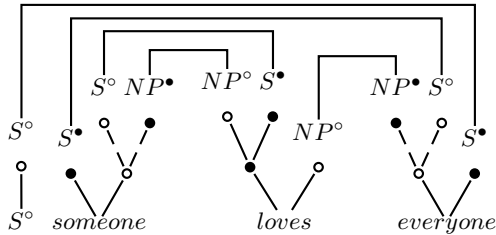
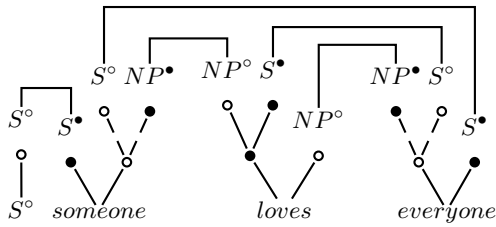
1.2 A first example – quantifier scope

Note that there can be several valid axiom linkings for one proof structure, each of the resulting proof nets then corresponds to one possible semantic reading. For example the sentence “*Someone loves everyone.*” has the two readings $\exists y \forall x.love(x, y)$ and $\forall x \exists y.love(x, y)$. The first (wide scope for “*someone*”) is the preferred

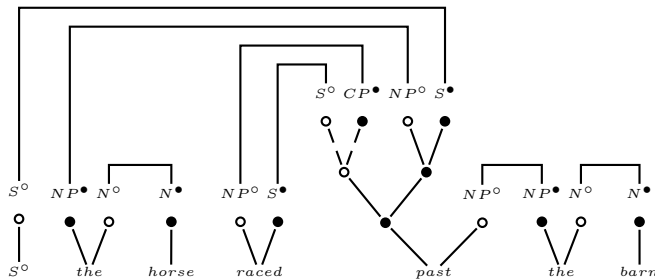
way for humans to interpret the sentence. Many systems using *Phrase Structure Trees* parse our sentence like shown in this tree:



While quantifier scope is a problem that needs to be treated by special mechanisms in many systems (e.g. QSTORE in HPSG), the two readings just drop out of the CG / proof-net approach for us in form of the two possible axiom linkings:



CG treats the two quantifiers as NPs with the side effect of sentence modification, and the two possible linkings correspond with the order in which the quantifiers contribute to the meaning of our assertion about love. Morrill notices that

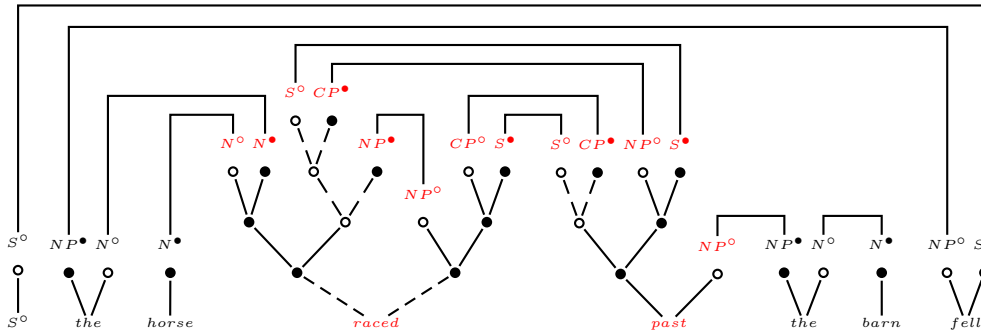


the dispreferred reading has both the point with the *most* correlations (axiom links) to be held in memory and the highest consumption of memory \times time. The idea is to sum up the number of axiom links spanning over each space between two words. Note that we start our parse with the pseudo-word S° (which means that we expect a sentence). In our example, the values are [1, 2, 3] for the first and [1, 4, 3] for the second reading. The idea is that the axiom links in a proof nets give a close approximation of the actual number of items a human sentence processor has to keep in short term memory. This extends to incremental processing as well: At each time (word), Morrill counts the links coming from or passing by from that point and to the right (future).

2 Evaluation of the memory load measure

2.1 Comments on a classic

I do not consider this measure to be completely *“psychologically real”* but it is an interesting approximation of human memory load. Consider for example the classical case of *Garden Path* sentences: *“The horse raced past the barn fell.”* Without the *“fell”*, the *“past the barn”* modifies *the way of racing* – in other words, the *“past”* is a word which modifies an action by mentioning a place. **With** the *“fell”*, the reader has to re-interpret the first part of the sentence as being a big NP. In that case, *“raced past a place”* acts as a modifier on *“horse”*.



This can be seen as “the horse (that was) *raced past the barn*” or as “the horse (that) *raced past the barn*”. The effect of the “that” being optional shows up quite often in English, and I suggest that the pending decision on whether a phonologically empty “that” has occurred also involves some processing load: In this CG framework, almost any case of the effect has to be treated by means of selecting another one of the lexical entries for a word (note that NP(“the horse (that was) *raced*”) can be handled by the same type of “*raced*” as NP(“the horse *raced past the barn*”): In the former case, most terminal nodes of “*raced*” are linked with each other, rendering the whole word into a simple *adjective* (with special word order!), but with lots of *bells and whistles* that do not leave the local tree.

The measure of Morrill would thus consider “The horse *raced fell*.” to be relatively easy to parse. A **computer** trying to parse the sentence (such as the quite user-friendly system *Grail* – see *Grail – An Automated Proof Assistant for Categorical Grammar Logics* (1998) by MOOT) still are likely to use lots of computing resources for trying other, not *word-internal* linkings. I think that this is not similar to the workings of a human parser. Rather, I suggest that there is a further lexical entry declaring “*raced*” as a plain *special word order adjective*.

There is a second problem in the same sentence: Replacing mixed-polarity pairs of **S** and **NP** by a **new atomic type**, we suddenly end up with a sharp decrease of the predicted memory load. One could argue that the mixed-polarity nature of our new type implies greater memory load, but the reduction of direct links between “*raced*” and “*past*” by a factor of two in this case certainly suggests a closer look.

One way to treat this problem would be to assume an infinite number of atomic types, each

associated with a certain memory load (which could also depend on the **direction** – with or against the flow of time – which is completely ignored by Morrill). That way, “*raced*” and “*past*” might be analyzed as having very few terminal nodes (e.g. 3 and 2) but with some of the nodes requiring quite *memory-heavy* links. Of course, the ramifications of that suggestion involve lots of mathematical questions like: Will it still be possible to do the semantics construction and the wellformedness check properly with such lexicon entries? A further possibility would be to *bundle* nodes (if related to common links) only for the memory load measure. Using the bundling idea in some heuristics for deciding which linkings to try first could on the other hand improve the efficiency of software like Grail!

2.2 Success stories

Although there are lots of points to make about the psychological reality of the measure proposed by Morrill, it was successful both in predicting the preferred reading of “*Someone loves everyone*.” and in predicting a Garden Path effect in “The horse *raced past the barn fell*.” – even though the latter is achieved in a way that involves a lexical entry for “*raced*” that is likely to have heavy impact on memory load whenever it is used. The point which still is made by Morrill is that ending the sentence after “*barn*” gives a very tame proof net in terms of memory load, and adding the “*fell*” increases memory load a lot.

Looking at the two proof nets, the fact that we need both to revise a *lexicon* lookup and to *re-link* axiom links that were seen as *closed* as early as at the point when “*past*” is encountered is something that should influence our processing load measure as well. But if we stick with predicting the parsing complexity *looking back* from

when the proof net got completed, it has to be said that checking some overall memory load over time is quite a fine grained measure.

Another measure would be checking a *maximal cut*, that is, the maximum number of axiom links that have to be held open at any point during processing. In our example, we have a value of four for the *“The horse . . . barn”* and a value of six for the complete sentence. There are lots of other occasions where both measures give the same prediction, but in the end the memory load over time is better. As Morrill points out, it can correctly predict the preference for

a) *“The chance (that the nurse (who the doctor supervised) lost the reports) bothered the intern.”*

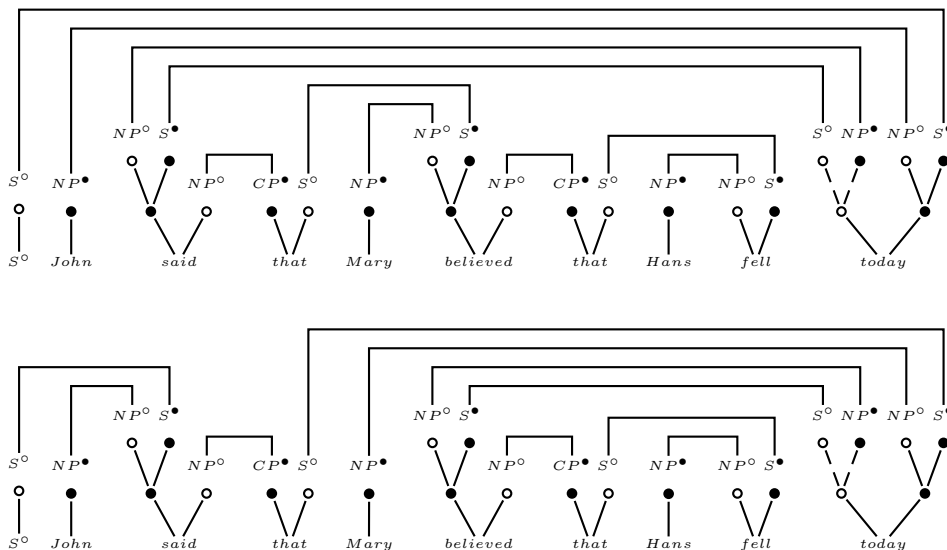
over b) *“? The intern (who the chance (that the doctor lost the reports)) bothered supervised the nurse.”*¹

which cannot be done using maximal cut alone. So Morrill sees the memory load measure as superior to maximal cut: The latter is for example employed in *Proof nets and the complexity of processing center embedded constructions* (1998) by

JOHNSON. Morrill explicitly refers to Johnson for comparison and shows memory load curves, but no proof nets for the two sentences in question. The peak is equivalent to the maximal cut (and equal for both sentences), but the area (memory load over time) differs, although not very much. Other measures could for example predict a larger difference in complexity based on penalties for certain constructions, using methods like pattern matching on Phrase Structure Trees²

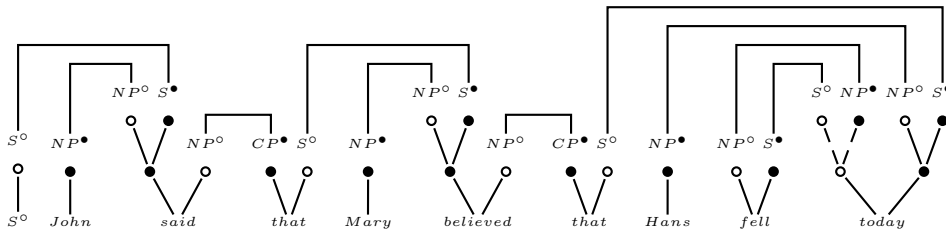
2.3 Preferred readings: Adverb attachment

The memory load measure also fits nicely with comparing the preferred readings for sentences with multiple ways to attach adverbs, like *“John said that Mary believed that Hans fell today.”*: Maximal cut does predict correctly that it is the *falling* that is most likely to be associated with *“today”*, but it does not predict differences between the two other readings. Memory load, on the other hand, increases significantly if the links which propagate the effect of *“today”* are stretched to attach to *“Joe said”*.



¹As a side note I must say that I do not have – or lost, due to too much exposure to strange linguistic effects – an intuition about the reasons of the extreme differences in parsing complexity.

²It would be nice to have a reference to something in this style actually done for example with TAG, or a more conservative implementation using weighted cost-per-use on the CFG rules used to construct parse trees, but I cannot provide one here and now!



While Phrase Structure Trees had a problem with quantifier scope in general and were not suitable to make predictions about readings of “*Someone loves everyone.*”, they *do* offer some insight in preferred readings of the mentioned adverb attachment example: Imagine attaching to “*Joe said*”: In order to connect the adverb there, we must predict the use of or actually use the $VP \rightarrow V (NP) ADV$ rule (variants would be $VP \rightarrow VP ADV$ or $S \rightarrow SADV$) if we want to stick with binary trees) very early / high in the Phrase Structure Tree, which is equivalent to *keeping* the unsaturated *ADV* branch open for a long time.

Even if we propose *ADV* adjuncts to be always expected for *VPs*, the low attachment preference could still be explained: Attaching “*today*” to the lowest *VP* also means attaching it to the most recent constituent, which can be expected to be most *salient* if we treat short-term memory as some place where especially unused information fades away over time. This seems quite plausible: The other extreme would be to assume that information is put on some stack where information can spill over as new elements come in.

To be exact, the *stack* comparison is not to be taken too strict: At least some effects – for example cross-serial dependencies in Dutch and Swiss German – cannot be handled with plain Context Free Grammars and thus not with stack machines in their classic form. Either pipe-like memory handling is needed in addition to the stack handling, or we must use other ways of addressing short term memory contents after all. One example would be addressing by function (such as agent / theme / ...) or content.

A classic among the predictors for preferred readings is *Late Closure*, as described in *On comprehending sentences: Syntactic parsing strategies* (1978) by FRAZIER. This is defined as follows:

Late Closure: *When possible, attach incoming lexical items into the clause or phrase currently being processed (i.e. the lowest possible nonter-*

minimal note dominating the last item analyzed).

This explains Garden Path effects such as in “*Since John always jogs a mile seems light work*”, but as GIBSON, PEARLMUTTER, CANSECO-GONZALES and HICKOK point out in *Cross-linguistic Attachment Preferences: Evidence from English and Spanish* (1994), there must be a second principle in balance to Late Closure or some other mechanism to explain different preferences across languages. Gibson et al. suggest using *Predicate Proximity*:

Predicate Proximity: *Attach as close as possible to the head of a predicate phrase*

They use this to account for the different attachment preferences in

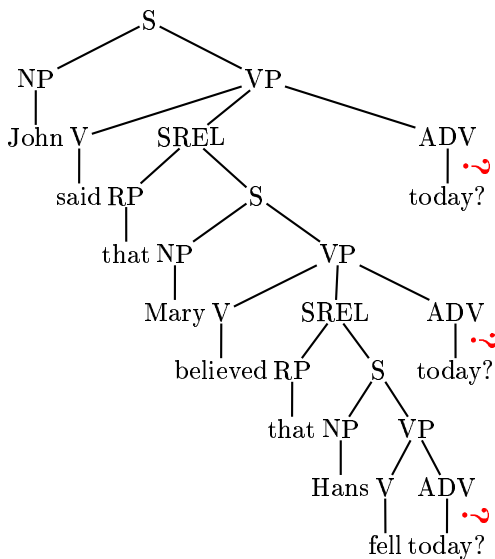
“*El periodista entrevistado a la hija del colonel que tuvo el accidente.*”

as opposed to “*The journalist interviewed the daughter of the colonel who had had the accident.*”:

Spanish readers assume the daughter to have had an accident, English readers assume the colonel to be the one to which the RC applies. However, if there are *three* possible attachments, the two mechanisms Late Closure and Predicate Proximity together never cause the *middle* attachment site to be preferred (this is obvious: both predict that extreme – high or low – attachments are preferred, and putting both together still does not give middle attachments the preference). I will discuss this in more detail using another example (NP(“*The apple by the tosti with the cheese that shined*”)) below. For some overview on further predictors and how to use them cross-linguistically, I recommend the article by Gibson et al. mentioned above.

2.4 Pondering an approach based on Phrase Structure Trees

To pursue some direct (?) PST approach of predicting processing load a bit further, I list some issues which could be used in such a measure: The most obvious would be keeping track how many branches (or even nodes) are left open or dangling. In the case of intermediate nodes, we would call them open or *unsaturated* if they have open (not leading to terminal nodes) successors. The first example of this measure in action can be easily seen if we draw the PST for the adverb attachment case mentioned above:



It can be seen that higher attachment causes one VP node to stay less saturated for longer – assuming it would be possible to retract the expectation for an ADV for the other VPs at the end of the sentence, otherwise we must assume that the parser would be most happy with *three* instances of “*today*” which seems to be quite implausible (among others due to the nesting pattern involved). So another approach is to assume an oracle that knows the final tree in advance. It would be similar to follow all combinations of VPs with and without ADV in parallel: Both models would predict completion of the higher attachment parses to be more complex.

A linear order oriented approach is using Late Closure as described above: New words are

preferably combined with the currently active subtree. In most cases this means combination of words that are close to each other in terms of linear order. Combined measures could for example model Late Closure and tree-un-saturated-ness together taking into account which branches are outdated or fading away (due to increasing distance) or even abandoned. For example if we attach “*today*” to “*Mary believed*”, then “*Hans fell*” can no longer be the target of ADV attachment.

2.5 Three-site relative clause attachment

In a recent course we³ were doing research on the possibility to attach a relative clause to the middle one of three NPs in fragments such as: NP(“*the apple by the tosti with the cheese that shined*”).

Our research aimed to verify whether it is possible to *communicate* each of the possible attachments using only *prosody*. The fragments that we used are controlled for equal plausibility of the three cases (the NP above is a simplified variant which may void this property). In an additional experiment, we found that it is indeed possible to stress the middle NP in a string of the pattern NP PP PP (NP P NP P NP) in a way communicating this selection (middle NP is most important) to a listener. In fact, the problems to stress the third NP were greater.

The effects in the main experiment (fragments *with* relative clause) were quite interesting: It was very hard for the speakers to express anything else than “*The apple (...) shined*” while sticking to the given wording: One speaker could also convey attachment middle, another to the last NP reliably, and some speakers were unable to make the listeners understand at all. In this context, understanding is indicated by knowing the intended attachments by listening better than by guessing.

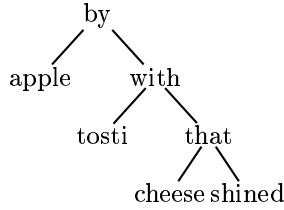
Finally, we did some research on the collected audio data to find the prosodic patterns that the speakers employed while trying to fulfil their part of the communication task. The whole ex-

³I should add a proper reference as soon as the bibliographical data has settled to a constant value...

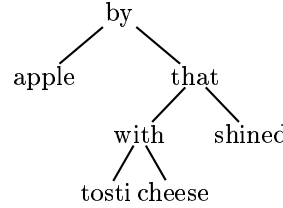
periment can be seen as an addition to research by WIJNEN and QUENÉ (*Prosodic Phrasing and Relative Clause Attachment in a Three-Site Context* (1998-2000): Our experiment makes no assumptions on the prosodic patterns but uses real listeners to check the possibility of communicating middle attachments at all.

Ignoring the determiners, we get the following trees from our fragment. The trees are shown in the same order as the proof nets below, the first version is predicted to be the preferred reading, the last two versions are both predicted to be not preferred at all. Notice that this feels a lot like Late Closure but *only* if we assume that we get a fresh NP after a P. If we assume that that the P leaves the current NP (“*the apple*”) active and only adds further specification to the type of the apple, the proof net memory load predictions are opposite to the Late Closure predictions... On the other hand, *adding further specification* is already a lot what Predicate Proximity is about. Maybe I am getting something wrong here, but I am sure that the proof net memory load measure *cannot capture both* at the same time. I think this will show up in unwanted effects when trying to extend the measure used by Morrill to cross-linguistical frameworks (see the Spanish vs. English case above).

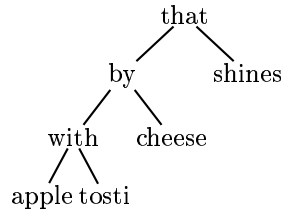
1. CHEESE shines



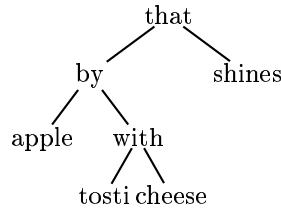
2. TOSTI-with-cheese shines:



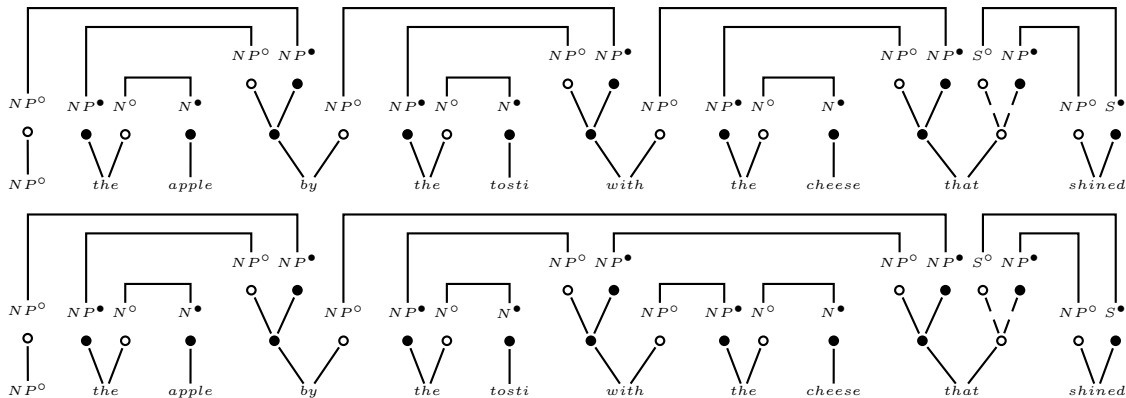
3. (APPLE-by-t)-with-c) shines:

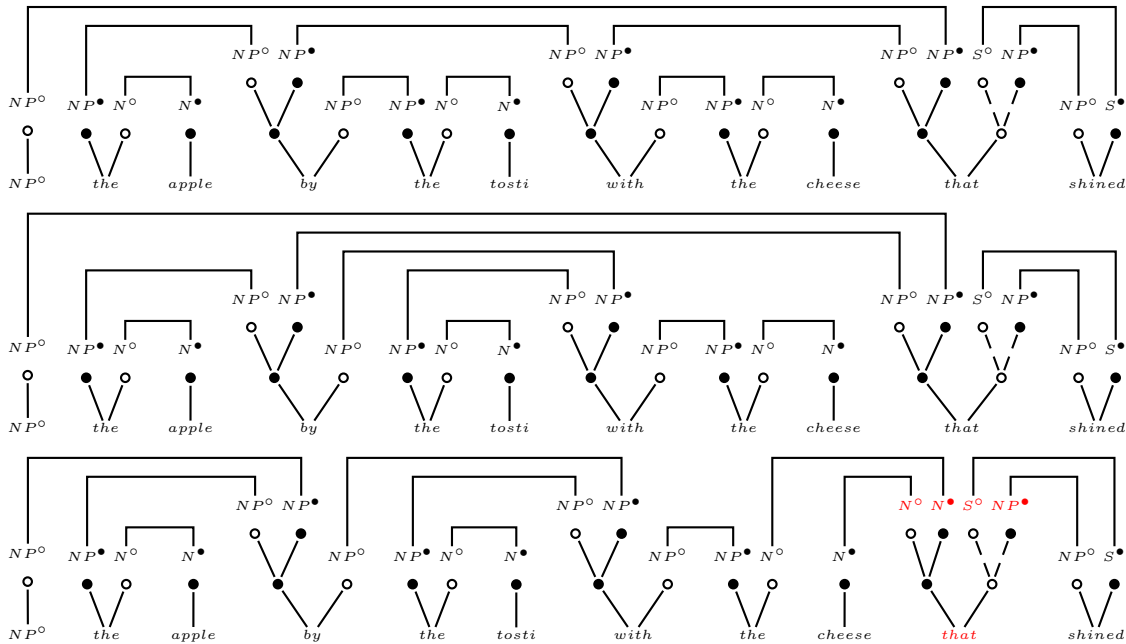


4. APPLE-by-(t-with-c) shines:



Warning: I use a “*that*” that modifies NPs instead of Ns – but using a “*that*” that modifies Ns as usual gives only the first reading, as is shown in the last of the following five figures.





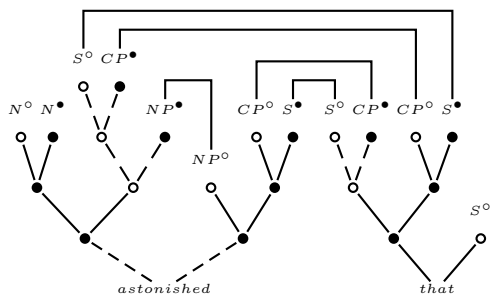
2.6 Other viewpoints from Morrill (2000)

Morrill has an interesting point to make about

“That that two plus two equals four surprised John astonished Mary.”

vs. *“Mary was astonished that John was surprised that two plus two equals four.”*,

where the latter uses lexical entries for *“astonished”* and *“that”* like this:



Although we have quite heavy linking between the two words, the whole parse uses not that much memory over time: After a *“that”*, the expectations are always reduced to *“S”* again. The other parse, on the other hand, uses a lot of memory: We assume *“that”* to be just of type CP/S (and the verbs to consume a subject of type CP instead of NP) for that parse, but still we get a

very high overall load due to stretching the semantic linkings of the verbs from their *“that”* to their quite late positions. Each such stretch causes the load to increase by two for that time, the peak is as high as *seven*. The proof nets can be seen in Morrill (2000).

I was planning to present the proof nets for

“John gave the book that Mary hated to Hans”

and *“John gave Hans the book that Mary hated”*

as discussed in Morrill (2000) here, but there is no special point about them: The prediction is that of those two semantically equal sentences, the latter is the preferred one, mainly because the *“gave...to”* stretches an expectation over some time which does not happen with the latter form of the sentence. Other measures would have predicted the same with the same ease.

2.7 Yet more examples

The first example in this section is about center embedding and taken from Morrill (2000):

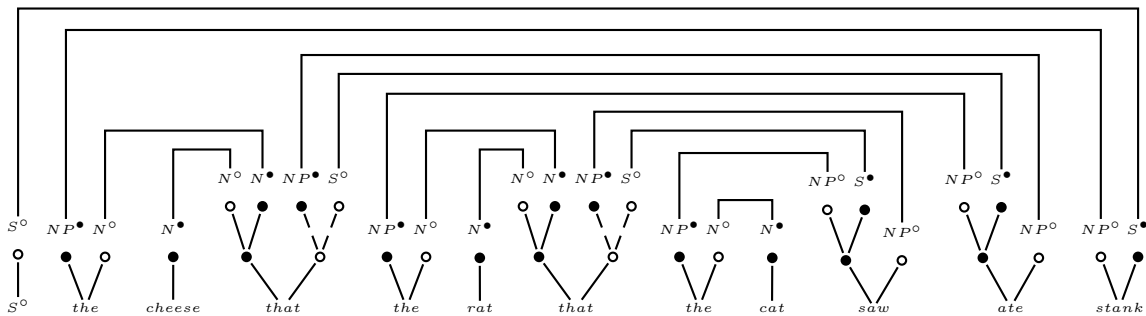
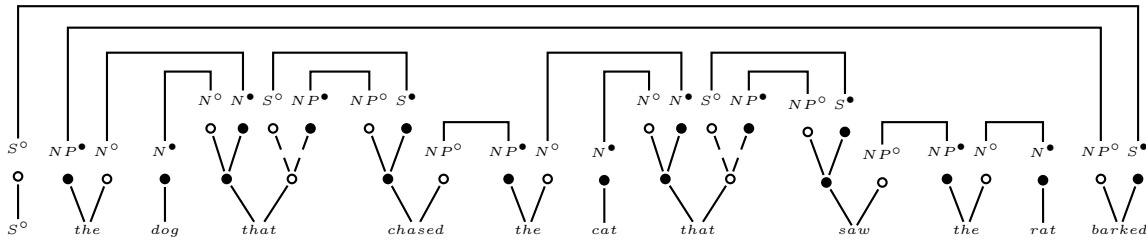
“The dog that chased the cat that saw the rat barked.”

vs. *“The cheese that the rat that the cat*

saw ate stank.”)

Yet again, the memory load measure correctly predicts the first version as being easier to understand. This success is a bit relativized by the fact that the use (lexicon entry) of “*that*” in the second version is a quite infrequent one, so a prediction based on probabilities of grammar rules and/or lexical selection would work just as well. Another notable point

is that the parse for the better to parse sentence is a tree like “*S(NP(the,(dog, (that, VP(chased, NP(the,(cat, (that, VP(saw, NP(the,rat)))))))), barked) ”*, while the other parse (if we use traces for the NP complements of the VPs) has to keep *two VP'* open for quite a while. Such a parse would use $N \rightarrow N \text{ that } S$ and $S \rightarrow NP VP'$ and finally $VP' \rightarrow V_{tr} \text{ (trace)}$.



The second example is derived from a paper about frequency-based (roughly: lexical entry probability) prediction of Garden Path effects from a course at my home university:

“The athlete realized her potential/exercise/shoes were-expensive/made-her-a-winner/Ø.”

and *“As the man sailed the boat/isle he-smiled/appeared”.*

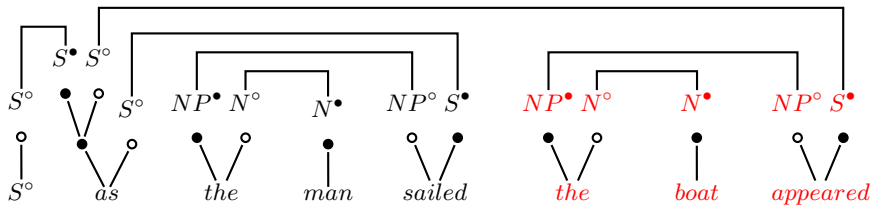
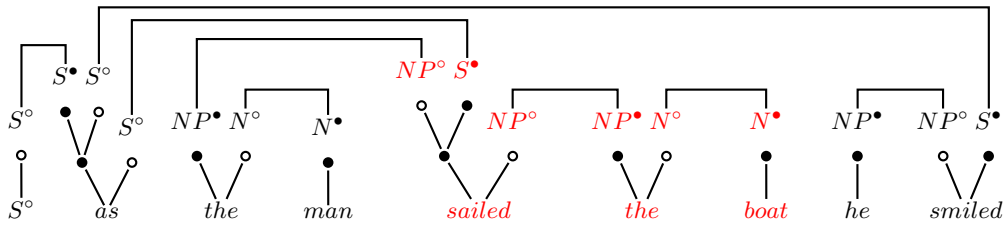
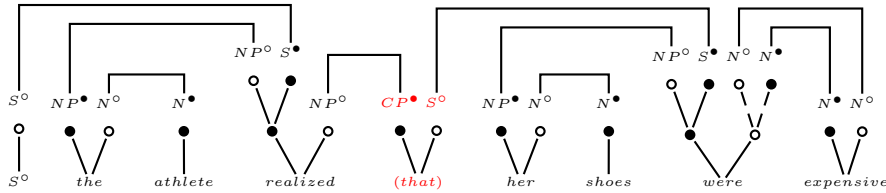
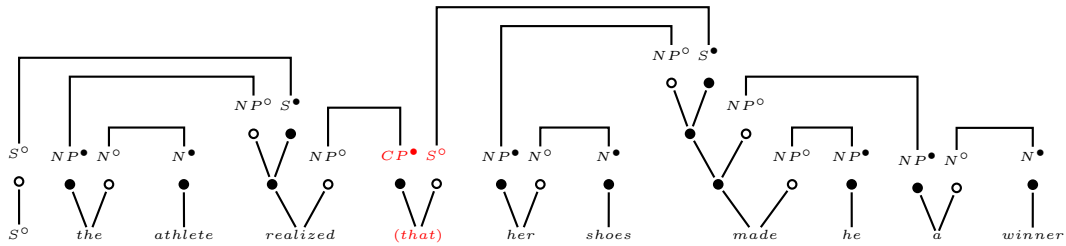
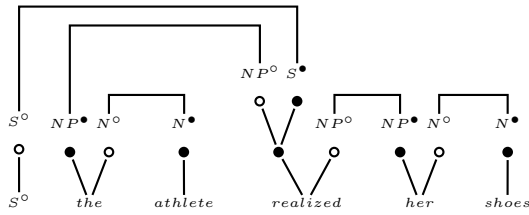
The Garden Path effect is correctly predicted by the memory load measure: The easier reading also involves less memory load. Note that the original paper was mainly researching the effect of varying the “*potential/exercises/shoes*” and “*boat/isle*” selection: Predictors based on plausibility of word combinations would for ex-

ample reject VP(“*sailed the isle*”) and thus involve less effect of the Garden Path on reaching “*... appeared.*”; PICKERING, TRAXLER and CROCKER discuss this in detail (using an eye-tracking experiment as source of some empirical data) and suggest a new measure called *informativity* in their paper *Ambiguity Resolution in Sentence Processing: Evidence against Frequency-Based Accounts* (2000).

Informativity is a measure that takes into account that some Garden Paths are easier to recover from than others. It also takes into account the probability of knowing after few further words whether the parse is following a Garden Path. Those two factors sometimes override the predictions made by plain plausibility based accounts. Of course this is also more fine-grained than the predictions made by the mem-

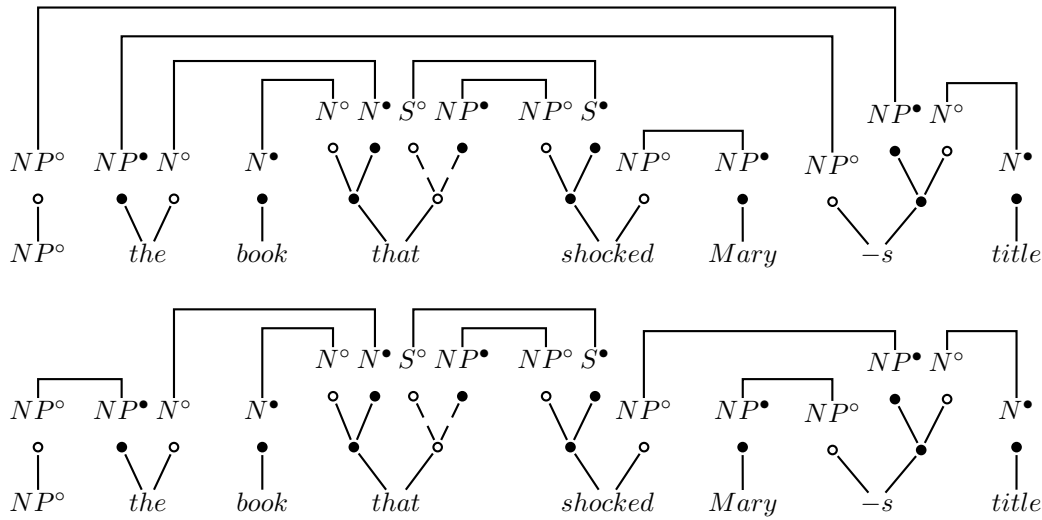
ory load measure, which completely ignore plausibility. Memory load treats all words of one cat-

egory the same, so everybody may sail an isle...



The last example is about some (almost) morphological effect described by Morrill: For NP (“*The book that shocked Mary -s title*”), the nonsensical (implausible) reading (Mary-s title got shocked) is the preferred one. The memory load measure correctly predicts this: The

first proof-net represents the plausible but dis-preferred reading. The second proof-net, representing the implausible reading, and it can be seen that this reading involves far less memory load as measured in the Morrill style.



3 Conclusion

Although the *psychological reality* of the memory load measure as described in Morril (2000) is questionable in some aspects, the predictions on processing complexity are quite good for the discussed effects. The seemingly simpler to compute related measure of open branches and nodes in PST is far less versatile, and if the general framework uses CG proof-net based parsing, we

get the memory load measure at *minimal cost* anyway. Of course, lots of other measures and finetuning mechanisms for them will be seen, and lots of experiments have to be designed and conducted to check the predictions. For the scope of this course, however, the discussed memory load measure can be seen as a quite useful addition to the learned techniques.

◇⁴

⁴Layout: L^AT_EXwith pstricks – thanks go to the authors of this impressive typesetting software! Editor: joe / Linux. This document online: <http://www.coli.uni-sb.de/~eric/> ...