agree grammar engineering environment

status update and parser evaluation (preliminary)

`http://wiki.delph-in.net/moin/AgreeTop`

Glenn Slayden

DELPH-IN Summit, June 2011

Ma.Sci. research at the *University of Washington*

Supervised by Emily Bender

agree system overview

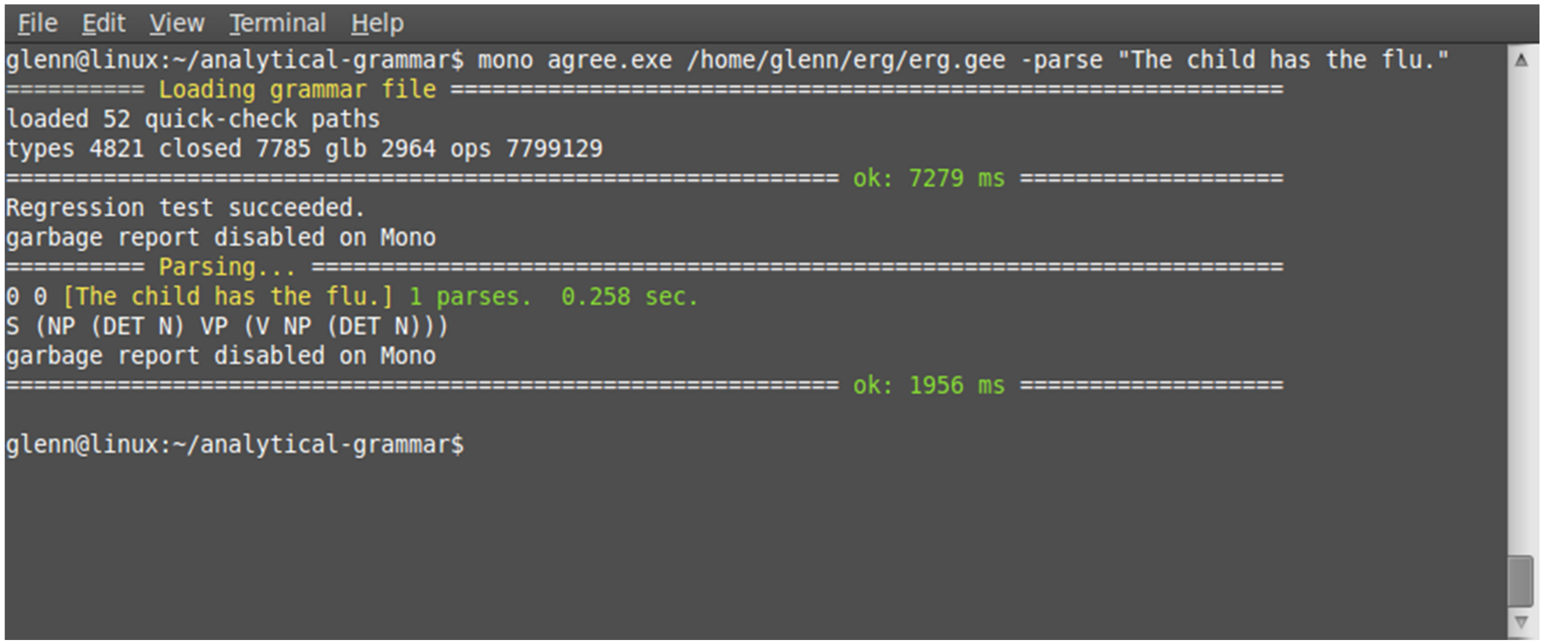
- TDL reader/parser
- Type hierarchy manager
- TFS storage
- Unifiers – incremental, n-way
- Lexicon manager
  – Weak GC references on lexical entries
- Concurrent chart parser
  – Morphological analyzer
  – DELPH-IN parser optimizations

agree system overview

- Sentence submitter
- Interactive command processor
- [incr tsdb()] database support (*preliminary*)
- Grammar configuration files
  - reads either LKB- or PET-format configuration files
  - custom tokenizer modules attached at runtime

agree Mono

- *agree* is primarily tested and developed on Windows (.NET runtime environment)
- Mac and Linux builds have also been tested:

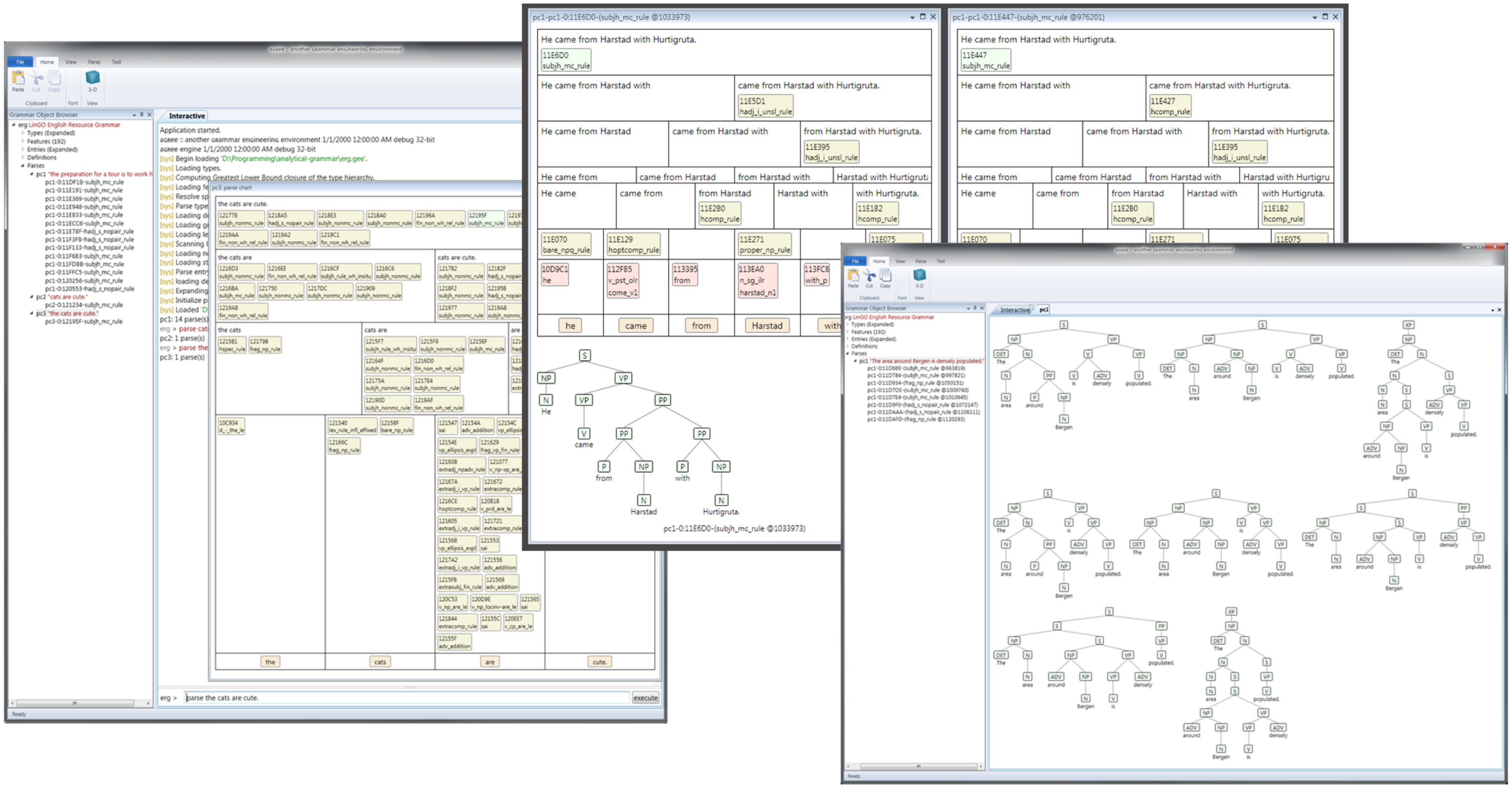


```
glenn@linux:~/analytical-grammar$ mono agree.exe /home/glenn/erg/erg.gee -parse "The child has the flu."
========== Loading grammar file ============================================================
loaded 52 quick-check paths
types 4821 closed 7785 glb 2964 ops 7799129
============================================================ ok: 7279 ms ===================
Regression test succeeded.
garbage report disabled on Mono
========== Parsing... ======================================================================
0 0 [The child has the flu.] 1 parses.  0.258 sec.
S (NP (DET N) VP (V NP (DET N)))
garbage report disabled on Mono
============================================================ ok: 1956 ms ===================

glenn@linux:~/analytical-grammar$
```

agree WPF

- For Windows, there is a graphical client application
- This will not be available on Mono



erg LinGO English Resource Grammar
Types (Expanded)
Features (192)
Entries (Expanded)
Definitions
Parses
pc1 "the preparation for a tour is to work h
pc1-0:11DF1B-subjh_mc_rule
pc1-0:11E191-subjh_mc_rule
pc1-0:11E369-subjh_mc_rule
pc1-0:11E948-subjh_mc_rule
pc1-0:11EB33-subjh_mc_rule
pc1-0:11ECC6-subjh_mc_rule
pc1-0:11E78F-hadj_s_nopair_rule
pc1-0:11F3FB-hadj_s_nopair_rule
pc1-0:11F133-hadj_s_nopair_rule
pc1-0:11F6B3-subjh_mc_rule
pc1-0:11FDBB-subjh_mc_rule
pc1-0:11FFC5-subjh_mc_rule
pc1-0:120256-subjh_mc_rule
pc1-0:120553-hadj_s_nopair_rule
pc2 "cats are cute."
pc2-0:121234-subjh_mc_rule
pc3 "the cats are cute."
pc3-0:12195F-subjh_mc_rule

```
Application started.
agree :: another grammar engineering environment 1/1/2000 12:00:00 AM debug 32-bit
agree engine 1/1/2000 12:00:00 AM debug 32-bit
[sys] Begin loading 'D:\Programming\analytical-grammar\erg.gee'.
[sys] Loading types.
[sys] Computing Greatest Lower Bound closure of the type hierarchy.
[sys] Loading fe
[sys] Resolve sp
[sys] Parse type
[sys] Loading de
[sys] Loading gr
[sys] Loading le
[sys] Scanning l
[sys] Loading n
[sys] Loading st
[sys] Parse entr
[sys] loading de
[sys] Expanding
[sys] Initialize p
[sys] Loaded 'D
pc1: 14 parse(s)
erg > parse cat
pc2: 1 parse(s)
erg > parse the
pc3: 1 parse(s)
```

erg > parse the cats are cute.

pc1-pc1-0:11E6D0-(subjh_mc_rule @1033973)

He came from Harstad with Hurtigruta.

pc1-0:11E6D0-(subjh_mc_rule @1033973)

pc1-pc1-0:11E447-(subjh_mc_rule @976201)

pc1 "The area around Bergen is densely populated."

agree concurrent unification

- All published TFSes are immutable
- Both “unifiers” are thread-safe
  - To be precise, there is no “unifier”
  - Rather, being passive algorithms, thread safety means that they are agnostic about concurrent use by the parser
  - Any number of top-level unifications can be underway at once
  - There is no way to disable thread-safety; single-threaded operation is configured in the parser

agree: parsing sequence

- Each parse chart is thread-safe, specifically, lock-free
  - This differs from van Louhizen, where each thread had its own chart, and these charts later had to be coordinated
  - Task agenda is not explicitly scheduled
- Parser respects a configurable task concurrency limit
  - The default is unlimited
  - For single-threaded tests, this is set to 1.
- Parser introduces one task per morphology stack
  - Parallel morphological analysis
- Chart dependencies: always single-threaded

lock-free parse chart

interlocked global sequence

- When a new passive edge is generated, it is given an atomic sequence stamp
- The parser queues two tasks:

1. Generate new active edges for the passive edge
    - New active edges also get atomically stamped
        - ❖ They gather their retroactive passive edges
        - ❖ That is, *passive edges* with a *lower* sequence stamp
2. Send the passive edge to subscribed active edges
    - That is, *active edges* with a *lower* sequence stamp

agree: packing

- Ambiguity packing
  - Parallel implementation of Oepen and Carroll 2000
  - Proactive/retroactive packing; subsumption and equivalence
- The packing/unpacking code is new in the past two weeks; there may be some bugs remaining

agree: parser optimizations

- Quick-check
- Key-driven (bidirectional)
- Chart dependencies
- Ambiguity packing
- Rule pre-check filter
- Spanning only rules

Morphology features:

- Stand-off input tokenization
- Arbitrarily overlapping input token hypotheses
- Inflection rule RegEx
- But not yet:
  - TMR, CM, REPP, etc.

agree concurrency: pipeline submitter

- The lock-free parse chart and concurrent unification capabilities are inherent to the design of the system
- These features are relevant for applications requiring the fastest possible results for a single parse
- For batch processing, *agree* also has an integrated concurrent sentence submitter
  - The submitter operates within the same process, referencing the same grammar
  - Avoids the overhead of multiple OS processes
  - Avoids redundant grammar loading which is wasteful of space and time

agree concurrency: grammars and parsing

- *agree* also supports loading and parsing processing multiple grammars within the same process
- This ability is designed for single-process machine translation scenarios (future work)
- Parse charts are independent and can be individually retained
  - The system can simultaneously hold, display, and report on multiple independent parse charts for interactive comparison, etc.

agree concurrency: summary

In agree,

- Multiple concurrent unifications can operate on…
- …multiple concurrent, diverse parsing tasks, within…
- …multiple parse charts, referencing…
- …one or more grammars…
- …per OS process…
- (…of which you could run more than one)

Evaluation methodology

- ERG rev. 8962
- Hike corpus subset (287/330)

  Subset was originally based on LKB's ability to exhaustively unpack and also minus sentences containing numerals

  http://www.agree-grammar.com/corpora/hike/hike-input-PET.txt
- Identical derivations from all parsers for all tests
- Exhaustive unpacking
  - *agree* currently does not support parse selection
- OS (on pluggable hard drive; swapped into the same machine)
  - LKB, PET: Linux x64

    ```
    $ cheap packing=7 -cm english.grm < hike-input-PET.txt
    ```
  - *agree*: Windows Server 2008 x64, .NET 4.0, gcServer
- Hardware: 8-way (2 × Xeon 5460), 3.17GHz, 32GB

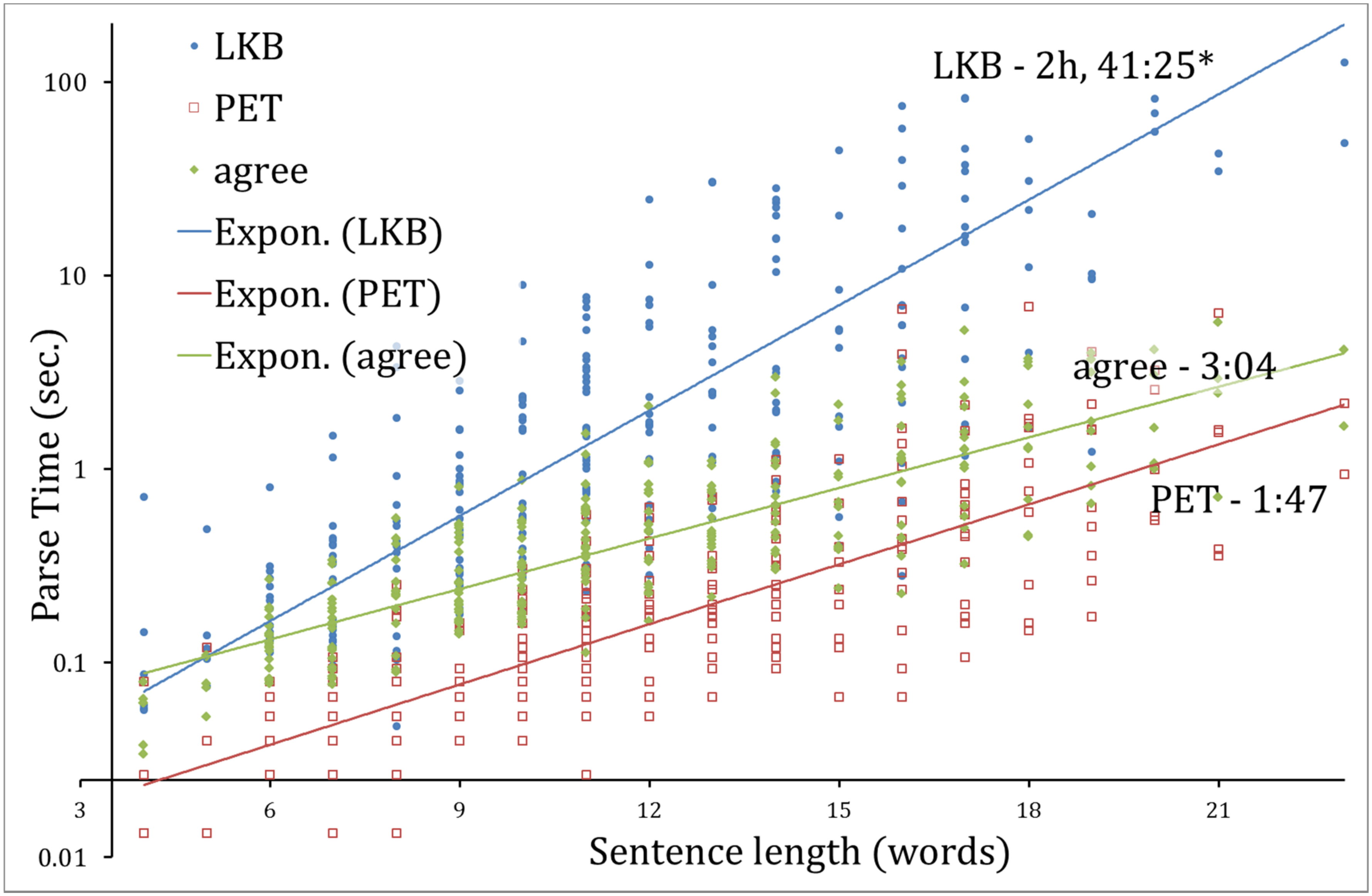
Test models

- Model 1: batch processing
  - Since any parser can be configured sentence pipelining, eliminate this variation
  - *agree's* intrinsic multi-threading was judged essentially similar in effect, so for these tests, parser task concurrency is limited to 1
- Model 2: real-time requirement
  - Sentence pipelining is outside the model definition and is therefore disallowed
  - Any intrinsic parser advantages which are helpful (native code, multi-threading) are permitted

Cross-parser evaluation, log t

Model 1: batch processing requirement

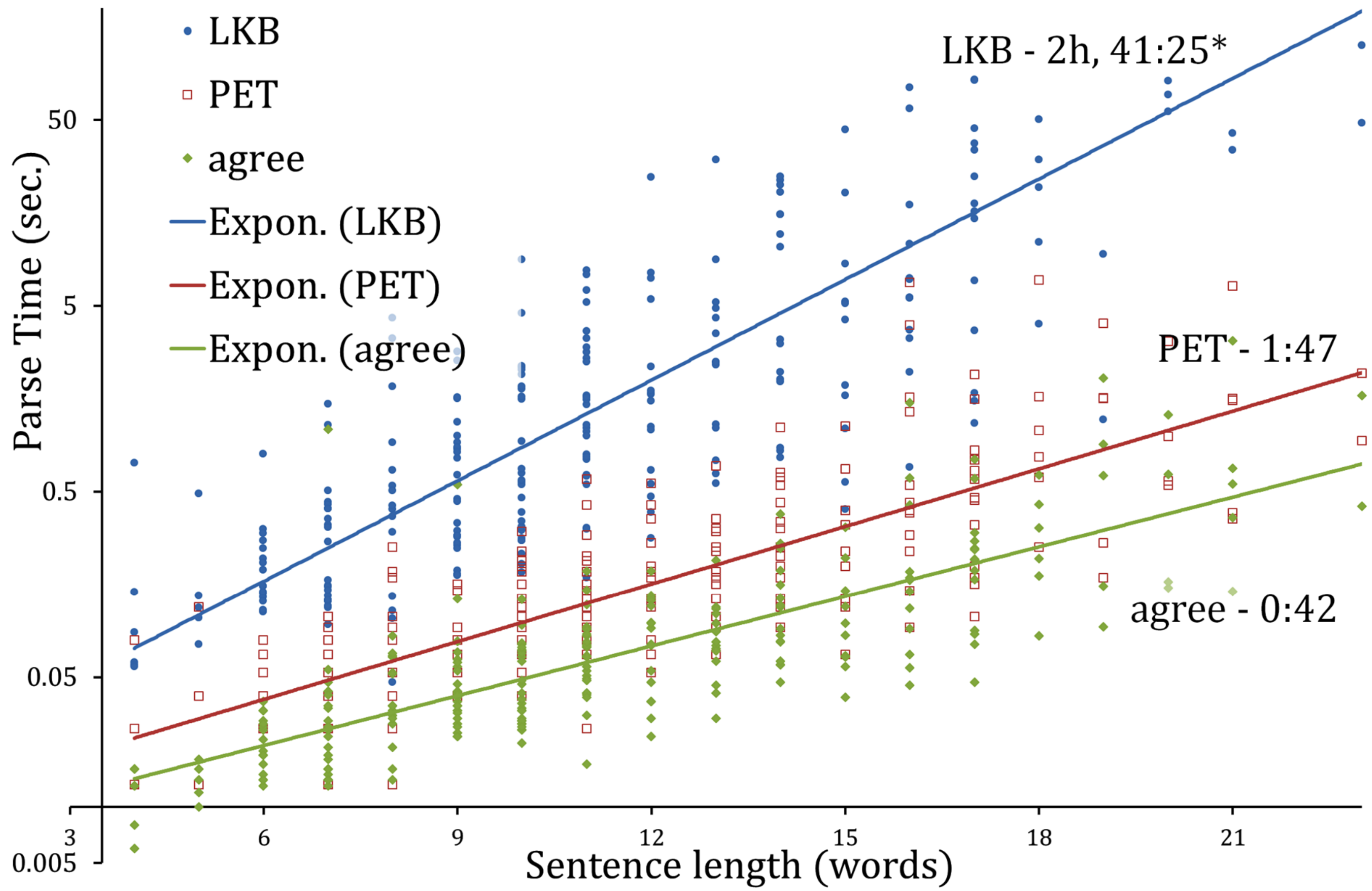
agree constrained to single-threaded, pipeline 1



Cross-parser evaluation, log t

Model 2: single-sentence real-time requirement

all systems: no sentence pipelining



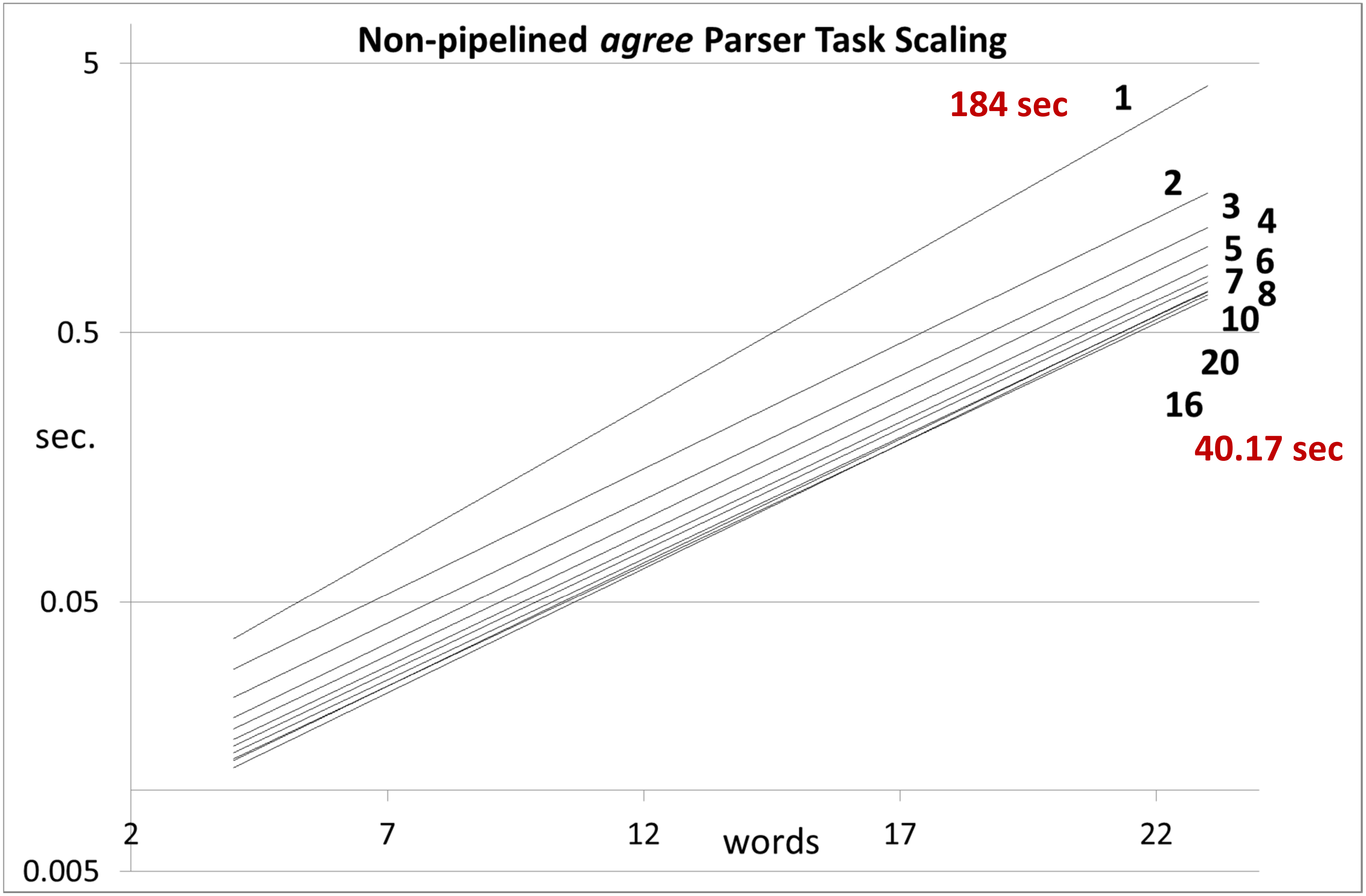
Model 2: long sentence result

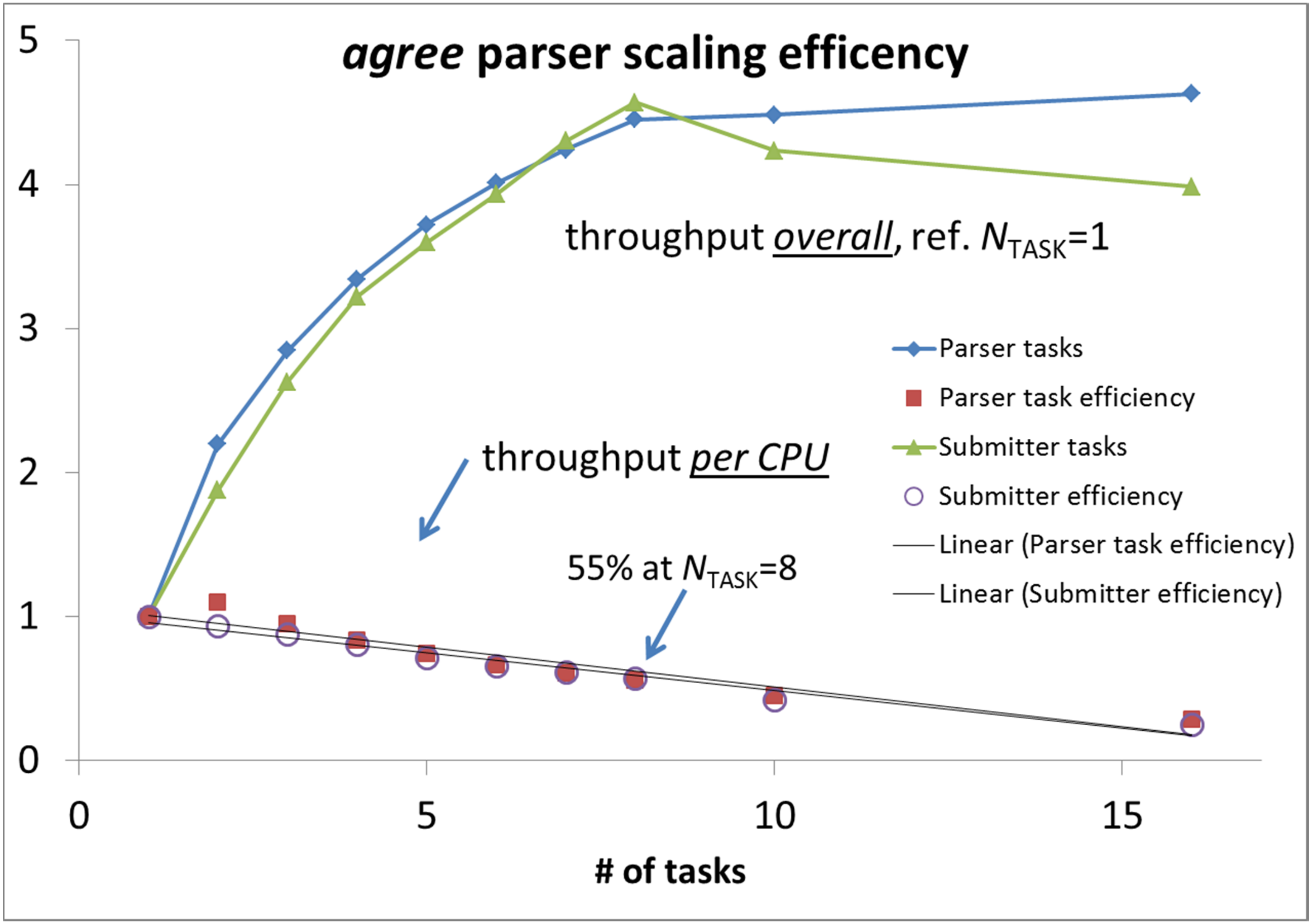
- Parse 10 times, with full packing and exhaustive unpacking:

Contact lenses come in small, light packages that are easy to ship through the mail and they require frequent replacement by the user.

parser	derivations	job total sec.	per parse, sec.
PET	67,716	54.46	5.45
agree, n-way	67,716	36.09	3.61

- Concurrency allows *agree* to overcome the penalty of operating in a managed runtime environment and parse this long sentence 34% faster than a native-code implementation


Non-pipelined *agree* Parser Task Scaling
5
0.5
0.05
0.005
sec.
2
7
12
words
17
22
184 sec
1
2
3
4
5
6
7
8
10
20
16
40.17 sec


agree parser scaling efficency
throughput *overall*, ref. N_{TASK}=1
throughput *per CPU*
55% at N_{TASK}=8
Parser tasks
Parser task efficiency
Submitter tasks
Submitter efficiency
Linear (Parser task efficiency)
Linear (Submitter efficiency)
# of tasks
0
1
2
3
4
5
0
5
10
15

agree best batch configuration

- The parsing sequence has single-threading choke points (i.e. chart dependencies)
- Therefore, the best configuration I found so far for batch processing with *agree* is: *12* parser tasks (limit), *pipeline 2*
- In accordance with the goal-driven test definitions, this configuration is not represented in either of the two parser test models
- With these settings, *agree* parses and exhaustively unpacks ‘Hike’ in 35.62 sec on the reference machine

Future work

- Characterize parser memory use
- Restore 43 Hike sentences and re-test
- *agree* parser features
  - Token mapping rules
  - Generic LE instantiation
  - parse selection
- Generation
- Further development of GUI

  …effort thus far limited by the desire to invest time in tools that also work on Linux
- DELPH-IN feedback…
  - Mike has volunteered to take a look at the Mono build

References

Ulrich Callmeier. 2001. *Efficient Parsing with Large-Scale Unification Grammars*. MA Thesis, Universität des Saarlandes - Fachrichtung Informatik.

Ulrich Callmeier. 2000. PET: a platform for experimentation with efficient HPSG processing techniques. *Natural Language Engineering* 6(1): 99-107.

Rebecca Dridan. 2010. Using Lexical statistics to improve HPSG Parsing. PhD Thesis, Universität des Saarlandes.

Dan Flickinger (2000). English Resource Grammar. In *Flickinger, Oepen, Tsujii, Uszkoreit, eds*.

Bernd Kiefer, Hans-Ulrich Krieger, John Carroll, and Rob Malouf. 1999. A Bag of Useful Techniques for Efficient and robust Parsing. In *Proc. of the 37th annual meeting of the Association for Computational Linguistics*. 473-480

References

Robert Malouf, John Carroll, and Ann Copestake. 2000. Robert Malouf, John Carroll, and Ann Copestake. 2000. Effcient feature structure operations witout compilation. *Natural Language Engineering*, 1(1):1-18.

Marcel P. van Lohuizen. 2001. A generic approach to parallel chart parsing with an application to LinGO. *Proc. of the 39th Meeting of the Association for Computational Linguistics*.

Marcel P. van Lohuizen. 1999. Parallel processing of natural language parsers. In *PARCO '99*.

Marcel P. van Lohuizen. 2000. Exploiting parallelism in unification-based parsing. In *Proc. of the Sixth International Workshop on Parsing Technologies (IWPT 2000)*, Trento, Italy.

Stephan Oepen and John Carroll. 2000. Ambiguity packing in constraint-based parsing - practical results. In *Proceedings of the 1st Conf. NAACL*, pages 162–169, Seattle, WA.