LKB: Porting, Algorithm and Feature Updates

John Carroll
Department of Informatics, University of Sussex, UK

DELPH-IN Summit, Cambridge, July 2019

Outline

Maintainability

Scalability

Usability

Compatibility

Maintainability

Aim to make LKB code more understandable and easier to extend

Critical code reviews and execution profiling

- code: obscure / inefficient / buggy
- comments: missing / misleading

Concentrated on: basic feature structure operations; type unification; parser task filtering and ordering, ambiguity packing, batch processing

Data structures remain simplistic (easier debugging); external behaviour unchanged

Cross-checked surprises against other systems, e.g.

- interaction between `*chart-packing*` and `*first-only-p*` (PET)
- cyclic check when testing feature structure unifiability with a start symbol (ACE)

Building

LKB-FOS is getting to be stable; repository http://svn.delph-in.net/lkb/branches/fos/ kept in sync with releases of pre-built binaries

3 releases in past year, via http://moin.delph-in.net/LkbFos : Oct 2018, Mar/Jul 2019

A few issues with build process:

- not fully automated
- no automated tests
- a few patches to McCLIM to improve usability are not yet in the repository
- need to use older versions of some libraries, due to regressions

(Can go into more detail in a SIG discussion)

Scalability

Aim to support grammar development at all scales (student grammars → resource grammars)

Type hierarchy: previous work reported at last year's Summit addressed browsing type hierarchies and processing large hierarchies (e.g. Norsyg)

Lexicons: the 'constant database' module performed poorly when database hashes collided (frequent with large lexicons, e.g. Zhong/zhs); fix greatly improves speed of lexicon reading, batch check, generator indexing

MRS construction: was slow dealing with thousands of parses – added memoisation to deal efficiently with repeated sub-structure and repeated computation (e.g. type of an MRS variable, canonical case of a type name, etc.)

Storage management

In many applications, long-lived data is unchanging

- therefore requires little garbage collector attention ⇒ hierarchy of 'generations', older generations being GCed less frequently

This is **exactly wrong** for our (quasi-destructive) DAG processing

- the (long-lived) grammar and lexicon DAGs are full of pointers into new objects
- GCing a new generation requires adjusting all old→new generation pointers – expensive!

Solution is to use only a single generation, and also don't GC frequently so data has more of a chance to become garbage. In SBCL:

```
(setf (sb-ext:generation-number-of-gcs-before-promotion 0) 1000000)
(setf (sb-ext:bytes-consed-between-gcs) (* 500 (expt 2 20))) ; =500MB
```

Reduces GC overhead in parsing from around 25% to 10%

Comparative evaluation (1)

- based on Glenn Slayden's evaluation in 2011
- ERG rev 8962, no token mapping, exhaustive unpacking, single thread
- 287 items from 'hike': those that do not exceed resource limit in LKB when packing turned off (265,610 total derivations vs. 277,946 in original)
- Xeon 5460, 3.17GHz, 32GB vs. i5 (2400S), 2.5Ghz, 8GB

System	*Total CPU time (hh:mm:ss)*
'classic' LKB (no packing)	2:41:25
agree	3:04
PET	1:47
LKB-FOS	2:52

Glenn Slayden, *agree grammar engineering environment*, DELPH-IN Summit, June 2011

Glenn Slayden, *Array TFS storage for unification grammars*, MS dissertation, UW, 2012

Comparative evaluation (2)

- couldn't get ACE to compile ERG rev 8962, so updated evaluation
- ACE 0.9.30 with precompiled ERG (2018)
- 171 items from 'hike': those items for which LKB and ACE give same numbers of derivations (total of 153,706)
- same hardware as before: i5 (2400S), 2.5Ghz, 8GB
- measure total CPU time and memory allocated

System	*First parse only*		*All parses*	
	CPU (sec)	*Space (GB)*	*CPU (sec)*	*Space (GB)*
ACE	18.5	3.4	96.3	9.9
LKB-FOS	24.6	2.8	126.3	25.6

Usability

Aim to make using the LKB as pleasant and productive as possible, for all kinds of user

In macOS

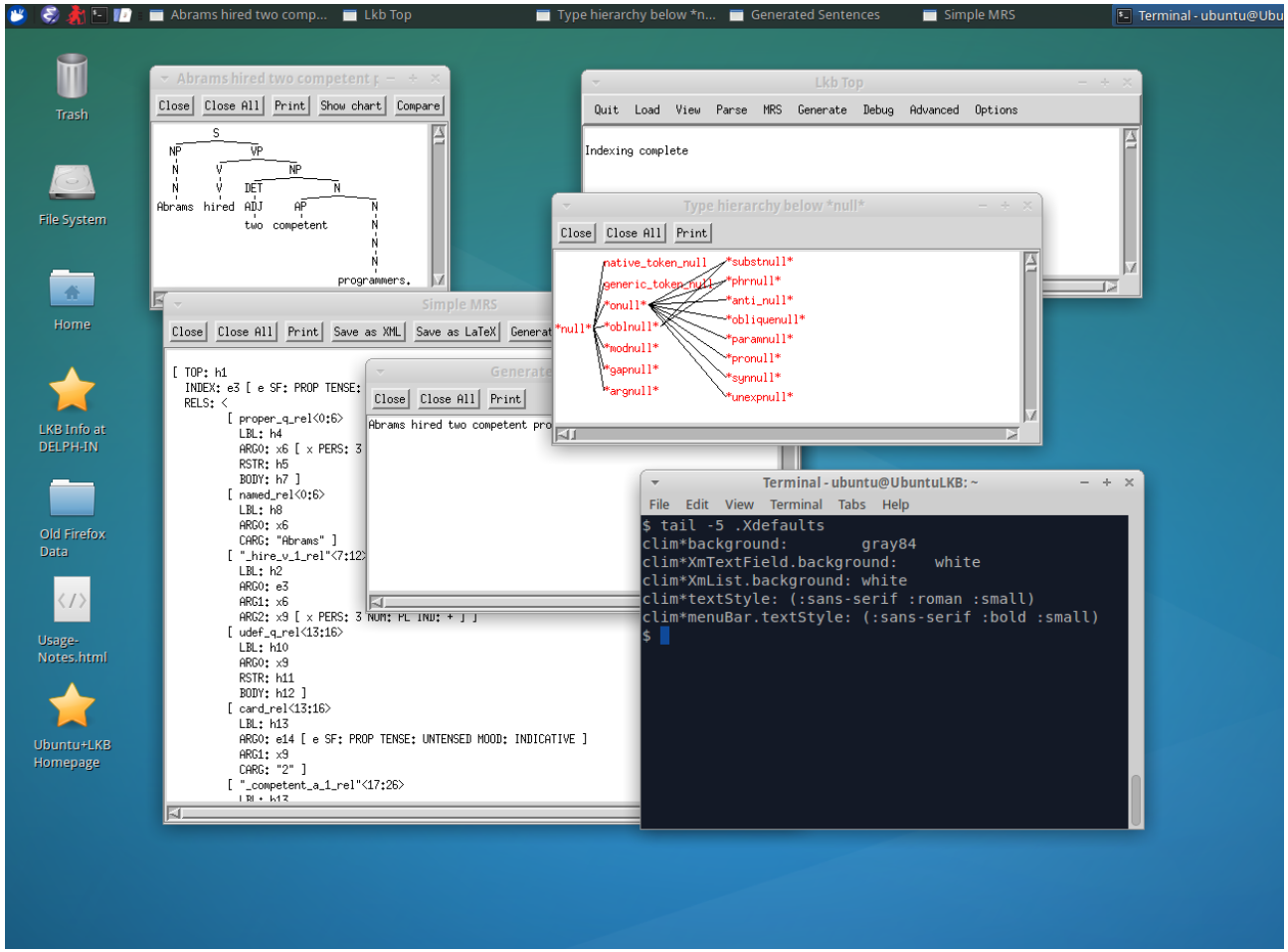
- familiar option key combos and keyboard layout switching for non-ASCII input
- `LKB.app` is a double-clickable application, bypassing Terminal / xterm

Customising the interface

- `*dialog-font-size*` controls font size in dialog box text fields, parse history menu, and Lkb Top window
- `clim:*default-text-style*` controls font style of all static text in interface
- experimental support for specifying a different font family for viewing the grammar and processing results

Digression: can customise **'classic' LKB** interface via X resources – by specifying attribute/ value pairs for application `clim` in .Xdefaults (or by using xrdb)

E.g. for a GTK-ish look



University of Sussex

Compatibility

Aim to make it easier to switch between DELPH-IN processors and platforms

TDL syntax standardisation:

- recent TdlRfc specification
- error messages include line/column number; error recovery more reliable

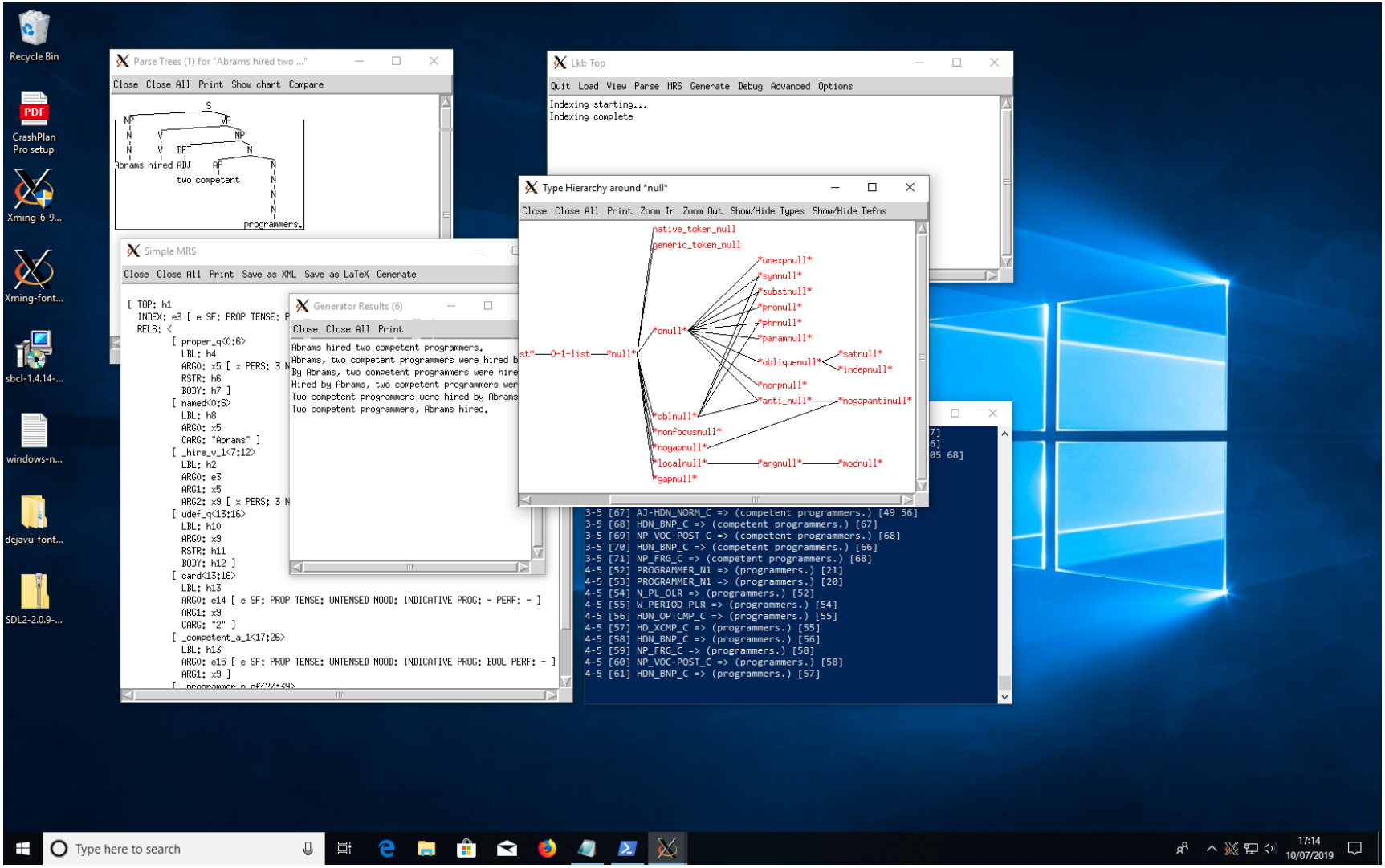
```
morph.tdl:1275:30: Error: In ST-DECL, found a single quote, which is
no longer a valid notation for strings
```

Spanning rules:

- prototype implementation gives average 5% reduction in parse time
- ... but in testing with ERG 2018, specifying spanning rules changes number of parses for some sentences

Token mapping: still not implemented

Platforms



University of Sussex

Code runs natively in Windows 10, but requires X server, e.g. Xming

Unfortunately, a few serious bugs:

- CLIM graphics do not start up from a saved image
- picks up a very basic X Windows font (not DejaVu Sans)
- window contents sometimes do not redraw properly after being obscured
- trying to display a large FS provokes an error

```
The value 32771 is not of type (SIGNED-BYTE 16)
```

  Caused by an X Windows limitation which McCLIM manages to program around on other platforms (by 'coordinate swizzling')

No LUI; PostgreSQL lexicon doubtful

Summary

Have worked on maintainability, scalability, usability, compatibility

Highlights

- many performance issues fixed
- Windows port looks promising

In progress / still to do

- user-friendly font selection
- token mapping
- LkbWishlist
- updated version of 'classic' LKB

Thanks! Any questions?