Using Mercury (for NLP), Manarchive

Ondřej Bojar
obo@cuni.cz

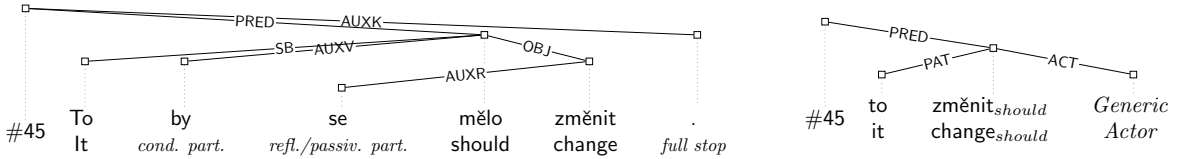
22 November 2007

Outline

- improved pickling (originally due to Peter Ross and Ian MacLarty)
- `cached` and `exclusive` calls
- Sun Grid Engine Interface
- first implementation of hugemap (need not fit in memory)
  - the underlying binding to tinycdb (~GDBM)
- SVG generation
- Manarchive goal and status
- various wishes
  - Perl binding
  - R binding, web-based promotion

Problems I Work With

- complex internal structure
  - dependency trees representing structure of sentences
  - rich structure of nodes (e.g. verb+tense+person+. . .)
  - linked (node to node or treelet to treelet) to other structures



- large amounts of data
  - at minimum 50,000 syntactic trees (aligned pairs of trees), but 1M wished
  - walk through, observe substructures, count observations.
- large (and complex) search spaces
  - machine translation: given a source tree, decompose, translate treelets, compose and return the most probable translation.

Improved Pickling

`pickle.m` and `byte_buffer.m` originally due to Peter Ross and Ian MacLarty

Improvements to `pickle`:

- string → sequence of bytes (not ints, i.e. not zero-padded to 32/64 bits)
- functor name and arity not pickled at all, just functor_lex
- no functor_lex if num_functors = 1 (so `unit` takes zero bytes)
- functor_lex → single byte if num_functors =< 256 (→ int otherwise)
- list(T) → length (int) + array of items (saves much space, faster)

Improvements to `byte_buffer`:

- can be "attached" to FILE* for reading or writing
- `preload(num_bytes::in, bb::di, bb::uo)` to `fread` instead of `fgetc`
- `memchunk_as_bb(dataptr::in, int::in) = (bb::uo)` to read from anywhere without extra copying

`cached` **and** `exclusive`

- `cached`: If the output of a predicate is ready in disk cache, just load it.

```
:- pred cached.call(cache_desc,    % which cache to use
          (pred(In, Out, io, io)), % the predicate to call
          (pred(io__output_stream, Out, io, io)), % serializer
          (pred(io__input_stream, Out, io, io)),  % deserializer
          In, Out,                 % the input and the output
          io, io).
```

- `exclusive`: Use file locks to prevent two processes run the same predicate simultaneously.

```
:- pred call(cache_desc,           % where to store lockfile
          (pred(In, Out, io, io)), % the predicate to call
          In, Out, % the input and the output
          io, io).
```

Sun Grid Engine Interface

- SGE is a simple queuing system (starts Unix processes at idling computers)
  - `qsub` to submit a non-interactive job
  - `qrsh` to run a job with bi-di pipe back to master

```
:- type host ---> localhost ; ssh(string) ; qrsh(sge_options).
:- type process(In, Out). % describes the job and bi-di pipes
:- type predlabel == string. % identifies the predicate to run

% Start a predicate on a host, send Input to it.
:- pred start(host, predlabel, external_pred_io(In, Out), In,
           process(In, Out), serializer(In), deserializer(Out), io, io).

% Wait for the predicate to finish, collect its Output.
:- pred finish(process(In, Out), Out, io, io).

% To be called at the very beginning of main/2.
:- pred slave_mode_io(predlabel, external_pred_io(In, Out),
           deserializer(In), serializer(Out), io, io).
```

GDBM / tinycdb Binding and hugemap

- External hashing libraries GDBM (or faster tiny constant db, `tinycdb`)
  - allow to store and retrieve key-value pairs not limited by RAM
  - pickling and unpickling Mercury data to byte `blob`.

`hugemap` based on `tinycdb`:

- **add** values to keys (journalling to disk if RAM cache full)
- **join** all values of a key, compile the map to `tinycdb`
  join has to be associative and commutative
- **search** for (joined) value given a key

Future:

- Google-like parallel map-reduce (Dean and Ghemawat, 2004) over hugemap.

hugemap **Interface**

```
:- type hugemap_growing(K, V).  % the map expects to get set/add requests
:- type hugemap(K, V).          % the map expects to get search requests
:- type hugemap_saved(K, V) ---> hugemap_saved( filename :: string ).

:- typeclass hugemapable(K, V) <= (serializable(K), serializable(V)) where [
     func join(K, V, V) = V % given a key, an old value and a new value, join the values
   ].

:- pred init(config::in, hugemap_growing(K, V)::out, io::di, io::uo) is det.

    % merge with an old value (if any exists) or set the value
:- pred add(K::in, V::in, hugemap_growing(K, V)::in, hugemap_growing(K, V)::out,
          io::di, io::uo) is det <= hugemapable(K, V).

:- pred stop_growing(hugemap_growing(K, V)::in, hugemap(K, V)::out,
          io::di, io::uo) is det <= hugemapable(K, V).

:- pred search(hugemap(K, V)::in, K::in, V::out) is semidet <= hugemapable(K, V).
```

Drawing SVG

- an intuitive API fro drawing Scalable Vector Graphics:
  augment picture by **placing** a subdrawing at a location, bbox grows automatically

```
:- type drawing.
:- type bless_subdrawing ---> no_bless ; bless(drawing_name).
:- type pointspec ---> v(vector) % a location within a box
                     ; sub(drawing_name, pointspec)
                     ; tl; tc; tr ; ml; mc; mr ; bl; bc; br.

:- pred place(pointspec::in, drawing::in,
           pointspec::in, bless_subdrawing::in,
           drawing::in, drawing::out) is det.
:- func text(string) = drawing.   % create a box containing text
:- func to_svg(drawing) = svg.    % Final output to SVG XML text

% eg:   place(tl, text("HELLO"), v(origin), bless("HELLO"), !D),
%       place(tl, text("WORLD"), sub("HELLO", br), !D)
```

Manarchive: the Goal

- http://manarchive.sf.net/
- a multi-purpose collection of libraries (can be used separately).
- Manarchive provides a common development and testing platform.

```
# Download current release of Manarchive:
svn co https://manarchive.svn.sourceforge.net/svnroot/manarchive manarchive
# Compile the 'manager', Manarchive package manager:
cd manarchive/manager; make

# Compile against package 'lib_tools' from Manarchive:
mmc --make \
  ‘ $MANARCHIVE/manager/manager -g $GRADE \
        --mmc-flags-to-link-against-noninstalled \
        -p lib_tools ‘ \
  target_to_compile
```

Manarchive: Current Status

- `lib_tools` the only package so far:
  - routines augmenting standard modules, eg. `map_tools`, `io_tools`
- `manager` still lacks package dependencies and support for custom compilation flags (eg. '`-lgz`')
- `pickle`, `cached` etc. slowly on their way to Manarchive

Idea of gradual migration of routines:

my projects → obomerclib → Manarchive → Mercury library

Wished: Mercury-Perl Binding

Perl might be the most distant language to Mercury design objectives.

But Perl has its good sides:

- excellent regular expressions
- full support of Unicode (and beyond)
- huge community and modules for everything (CPAN)
- Perl is written in C after all

Weak Perl Binding Option

```
:- type perl.
:- type perl_code == string.
```

and then

```
:- func new = (perl::uo) is det.
:- pred run(perl_code::in, perl::di, perl::uo) is det.
:- pred eval_to_int(perl_code::in, int::out, perl::di, perl::uo) is det
```

or

```
:- pred new(perl::out, io::di, io::uo) is det.
:- pred run(perl_code::in, perl::in, io::di, io::uo) is det.
% ...
```

Full-fledged Perl Binding Option

```
:- type maybe_undef(T) ---> undef; defined(T).
:- func first_match(perlre, string) = maybe_undef(string).
:- pragma foreign_proc(perl,
first_match(RE::in, HayStack::in) = (Needle::out),
        [will_not_call_mercury, thread_safe, promise_pure], "
  if ($HayStack =~ /$RE/) {
    $Needle = $1;
  }
").
```

Requirements:

- launch perl interpreter before main, if any perl proc called
- possibly export Mercury predicates to Perl, too

Various other wishes

- improve binding to R, the statistical package (works but problems with garbage collectors)

- add Mercury "interactive screenshots" to Mercury web page
  $\Rightarrow$ prospective users might see how exact and useful error messages are
  $\Rightarrow$ users might add further comments and explanations to error messages for newbies

- more collaboration on Manarchive (more beta-testers of my modules)

Summary of Long-Term Mercury Exposure

+ type and mode declarations
+ compiler and its error checking excellent
+ excellent support from Mercury developers

⇒ changed the way how I think.

− current limitations of typeclasses, monadic style problematic

⇒ I cannot expect another intellectual leap.

Practical issues:

+ tools: mprof great, but I'm still not using mdprof :-(
  Maybe if mdprof had built-in micro HTTP server and dp grades by default. . .
± I've no motivation for (decl) debugging.
− little community support, limited libraries

References

Dean, Jeffrey and Sanjay Ghemawat. 2004. Mapreduce: Simplified data processing on large clusters. In *OSDI*, pages 137–150.