Central European Functional Programming School, 2015

Parallelization by Refactoring - Labs -

Dept. Programming Languages and Compilers
Eötvös Loránd University, Hungary

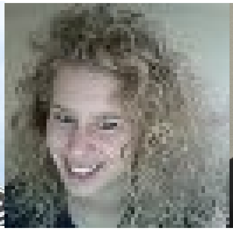
Judit Kőszegi

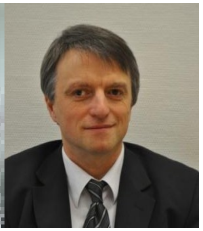
Dániel Horpácsi

Tamás Kozsik

Melinda Tóth

István Bozó


Viktória Fördős


Zoltán Horváth

Let's PaRTE!

map-like function speedup prediction

pattern candidate discovery

static analysis task farm

refactoring pipeline parallel patterns

algorithmic skeletons divide and conquer

ParaPhrase Refactoring Tool for Erlang

RefactorErl Wrangler

Parallel Patterns for Adaptive Heterogeneous Multicore Systems

University of St Andrews

ROBERT GORDON UNIVERSITY • ABERDEEN

National College of Ireland

Cloud Competency Center

SEVENTH FRAMEWORK PROGRAMME

Queen's University Belfast

AGH

AGH University PL

Erlang SOLUTIONS

UNIVERSITÀ DEGLI STUDI DI TORINO

ALMA UNIVERSITAS TAURINENSIS

UNIVERSITÀ DI PISA

Universität Stuttgart

HLRS

Mellanox TECHNOLOGIES

scch

software competence center hagenberg

ParaPhrase approach

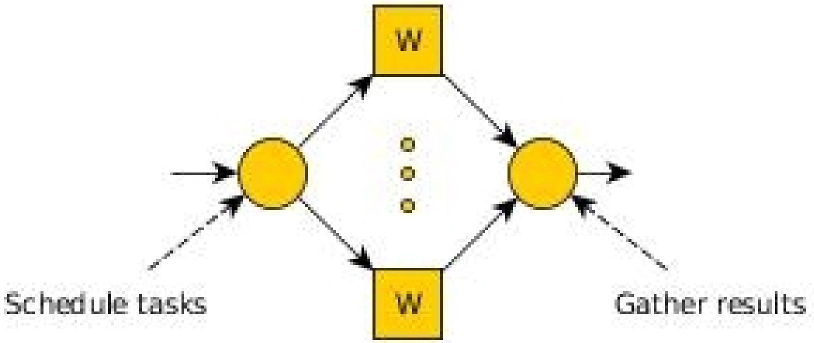
- Identify (strongly hygienic) components
- Discover patterns of parallelism
- Structure the components into a parallel program
  - Turn the patterns into concrete code (skeletons)
  - Take performance, energy etc. into account
- Restructure if necessary
- Use a refactoring tool

Pattern-based parallelism

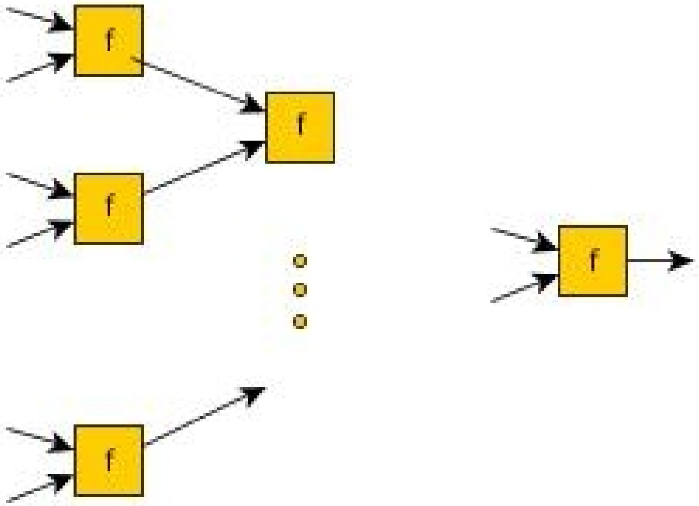
High-level approach to parallel programming

- Rely on a library of algorithmic skeletons
- Easier to develop code
- Easier to modify / maintain
- Better utilization of resources
  - Static resource management
  - Dynamic resource management

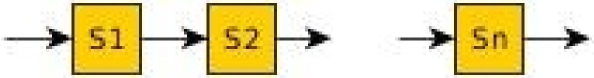
Farm



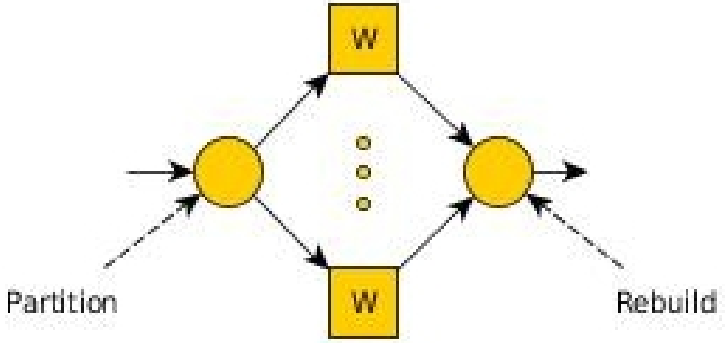
Reduce



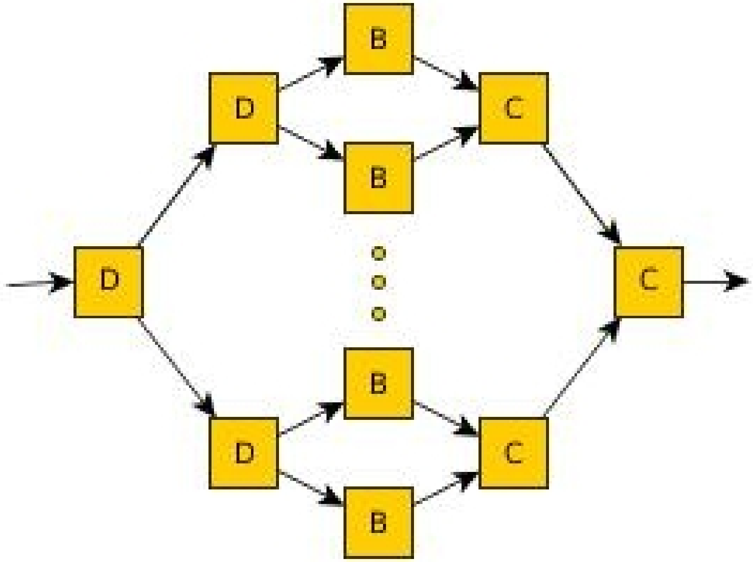
Pipeline



Map

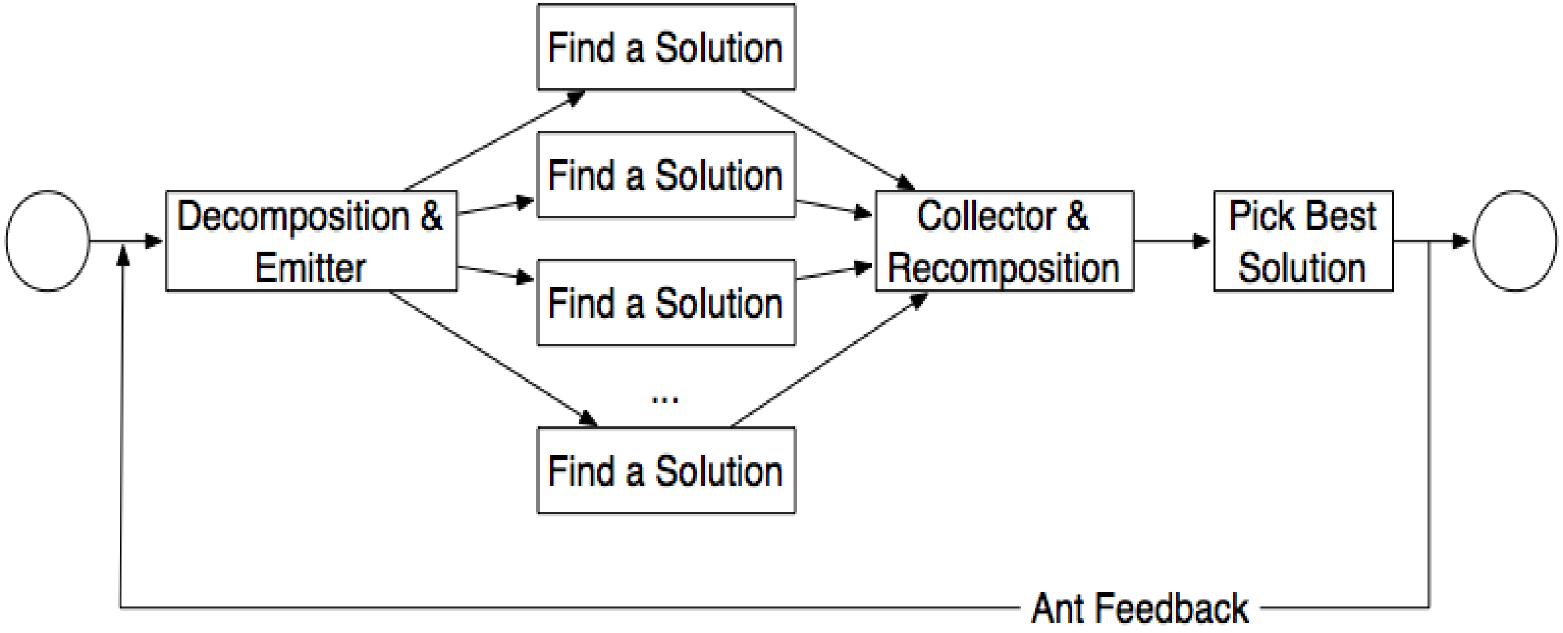


Divide&Conquer



Kozsik, T. et al.: Parallelization by Refactoring

Pool pattern



Ant Colony Optimization

Kozsik, T. et al.: Parallelization by Refactoring

Parallel skeletons

http://paraphrase-ict.eu/Deliverables/deliverable-2.6

The **skel** library

- Basic algorithmic skeletons
  farm, pipe, map, reduce, ord, feedback etc.
- High-level patterns: skel hlp
  dc, evolutionPool etc.
- Heterogeneous skeletons: Lapedo
  OpenCL kernels for CPU and GPU

http://paraphrase-ict.eu/Deliverables/d27prototype.tar.gz

Example: parsing modules

```
[ parse ( scan ( read ( Module ) ) )
                    || Module <- Modules ]
```

```
skel:do([
  { farm, [{ pipe, [ { seq, fun read/1  },
                     { seq, fun scan/1  },
                     { seq, fun parse/1 }
                   ] }
          ], 5 }
  ], Modules )
```

Example: radix sort

```
sort( List ) -> sk_hlp:dc(

     fun({Lst,Level}) -> length(Lst) < 2 end,

     fun({Lst,Level}) -> Lst end,

     fun({Lst,Level}) ->
           [ {Bucket,Level+1}
             || Bucket <- divide(Lst, Level)
           ]
        end,

     fun lists:append/1

  )  ({List,0}).
```

How shall I parallelize?

- Refactor
  - Use a tool!
  - Guided, semi-automatic transformations
- Experiment
  - Measure, validate
- Repeat
- Applicable for legacy code as well

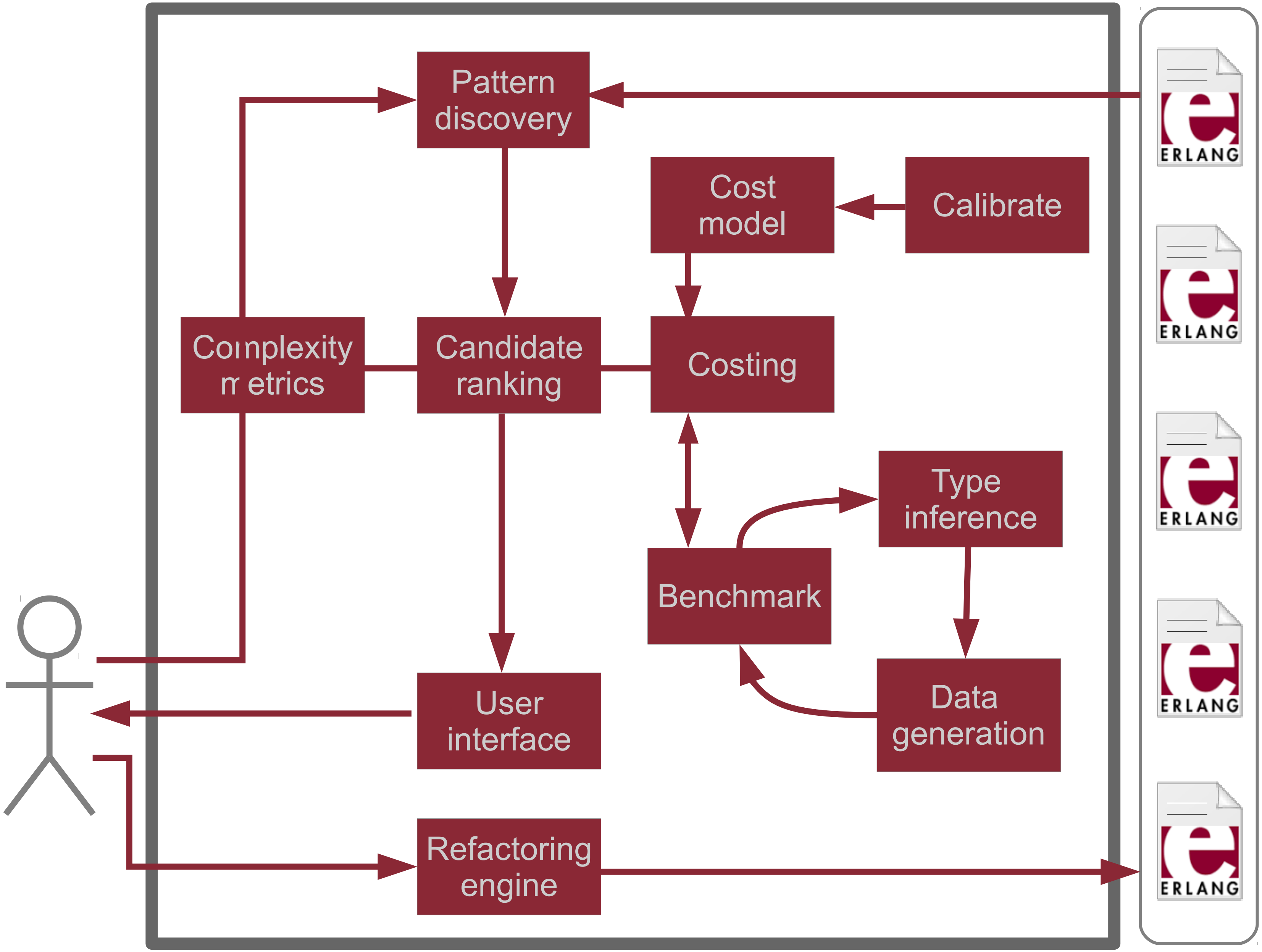
Where shall I parallelize?

- Independent computations
- Good potential for speedup
  - Complex computation?
  - Low sequential overhead?
- Find candidates automatically
  - Use a tool!
  - Static program analysis

PaRTE

ParaPhrase Refactoring Tool for Erlang

- Locate parallel pattern candidates
- Estimate speedup for different configurations
- Advise programmer
- Assist with refactoring
- Enforce preservation of functionality


Pattern discovery
Cost model
Calibrate
Complexity metrics
Candidate ranking
Costing
Type inference
Benchmark
Data generation
User interface
Refactoring engine
ERLANG
ERLANG
ERLANG
ERLANG
ERLANG

Expectations

PaRTE...

- can find many places to introduce parallelism;
- provides good hints where to parallelize
- gives fair speedup predictions;
- works effectively with a smart programmer;
- offers performance gains with small effort.

How to put your hands on it

- Download from wiki:

  http://pnyf.inf.elte.hu/trac/refactorerl/wiki/parte

- Requirements:

Linux or OSX	Erlang OTP 17
Emacs >=23	g++ >= 4.5

- Install: `./install_parte -build parte`
- Start:

  ```
  cd referl/tool
  bin/referl -db nif
  ```

- Ask for help: erlang@plc.inf.elte.hu

At CEFP

http://pnyf.inf.elte.hu/trac/refactorerl/wiki/parte/cefp

- Boot in Linux, install PaRTE
- Use our VirtualBox virtual machine

Demo

Evolutionary Multi-Agent Simulation Framework

- Developed at AGH University (Poland)
- Computationally intensive MAS
- Meta-heuristic used in optimization and problem solving

http://paraphrase-enlarged.elte.hu/downloads/euc.zip

Demo

What can we do with this?

- Load into PaRTE:

  ri:add(“path_to_mas_sequential.erl”).

- Pattern discovery (farm, pipe, pool)

  refpp_api:run(mas_sequential).

  refpp_skeleton:find_candidates().

  refpp_skeleton:find().

- Transformation

How to use Emacs?

- Look for .emacs in home
- “Meta key” is “Alt”
  - Alt-X refactorerl-mode
  - Alt-X erlang-parte-on

Refactoring

- Program Shaping
- Introduction of Skeletons
- Cleanup Transformations

Either directly or after PC discovery

Your task

- Discover patterns in a matrix-multiplication module
- Access the code from here:

http://paraphrase-enlarged.elte.hu/downloads/euc.zip

Example: mymath.erl

```
-module(mymath).

-export([fib/1,prime/1,pi/0]).

-define(PI,3.14).

pi() -> ?PI.

fib(N) when N<2 -> N;
fib(N) -> fib(N-1) + fib(N-2).

prime(1) -> false;
prime(N) when N > 1 -> prime(N,2).

prime(N,M) when M*M > N -> true;
prime(N,M) when N rem M == 0 -> false;
prime(N,M) -> prime(N,M+1).
```

Compiling and running

```
$ ls mymath.erl
mymath.erl
$ erl
Erlang R17 (erts-5.10.1) [source] [smp:4:4]
[async-threads:10] [hipe] [kernel-poll:false]

Eshell V5.10.3 (abort with ^G)
1> c(mymath).
{ok,mymath}
2> mymath:prime(1987).
true
3> q().
ok
4> $ ls mymath*
mymath.beam mymath.erl
```

Bonus: Discover patterns in Mnesia

```
reverse([]) -> [];

reverse([ H=#commit{ ram_copies      = Ram,
                     disc_copies     = DC,
                     disc_only_copies= DOC,
                     snmp            = Snmp }
        | R ]) ->

  [ H#commit{
     ram_copies       = lists:reverse(Ram),
     disc_copies      = lists:reverse(DC),
     disc_only_copies = lists:reverse(DOC),
     snmp             = lists:reverse(Snmp)
    }
  | reverse(R) ].
```

Resources

ParaPhrase project (FP7 contract no. 288570)
http://paraphrase-ict.eu/

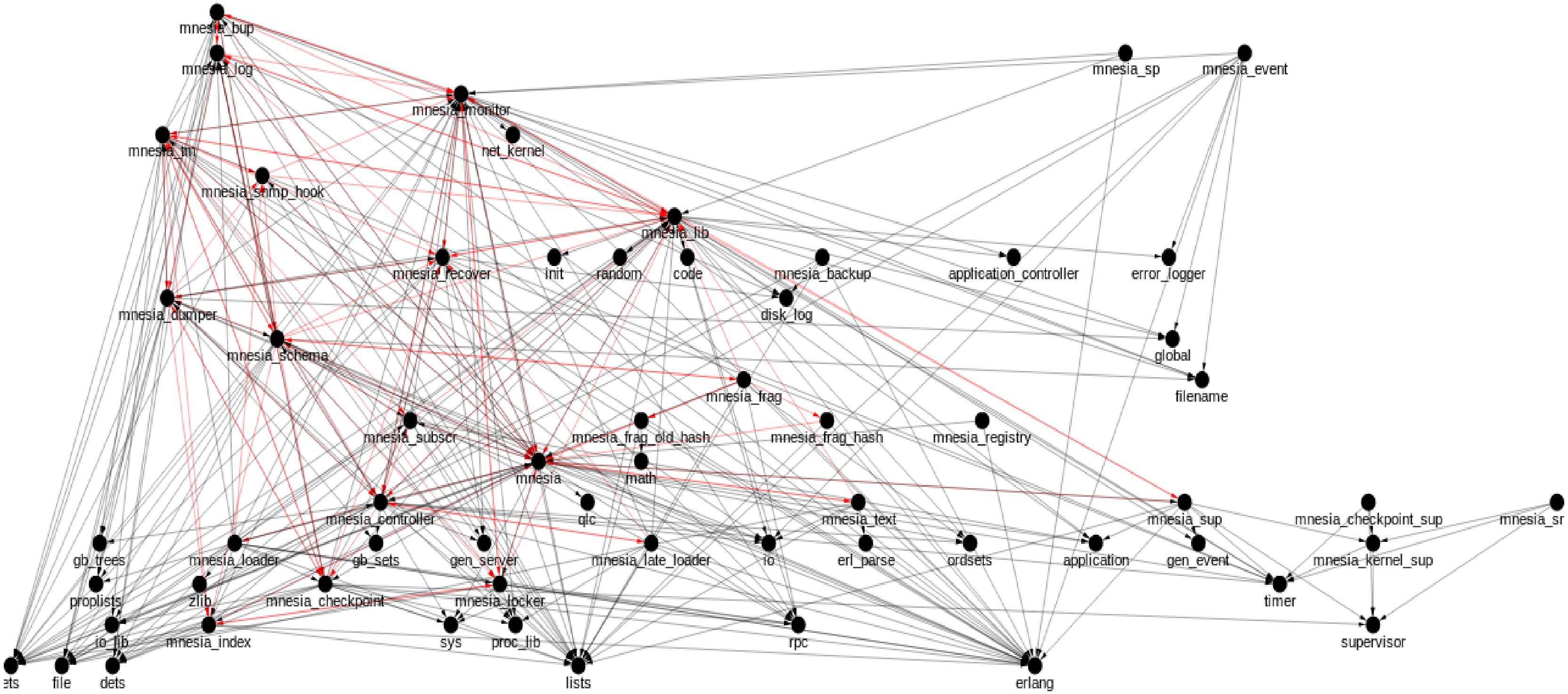
ParaPhrase @ ELTE
http://paraphrase-enlarged.elte.hu/

PaRTE docs & download:
http://pnyf.inf.elte.hu/trac/refactorerl/wiki/parte

CEFP 2015 instructions:
http://pnyf.inf.elte.hu/trac/refactorerl/wiki/parte/cefp

RefactorErl

Static source code analyzer and transformer
http://plc.inf.elte.hu/erlang


mnesia_bup
mnesia_log
mnesia_sp
mnesia_event
mnesia_monitor
net_kernel
mnesia_tm
mnesia_snmp_hook
mnesia_lib
mnesia_recover
init
random
code
mnesia_backup
application_controller
error_logger
disk_log
mnesia_dumper
mnesia_schema
global
filename
mnesia_frag
mnesia_subscr
mnesia_frag_old_hash
mnesia_frag_hash
mnesia_registry
mnesia
math
mnesia_controller
qlc
mnesia_text
mnesia_sup
mnesia_checkpoint_sup
mnesia_sr
gb_trees
mnesia_loader
gb_sets
gen_server
mnesia_late_loader
io
erl_parse
ordsets
application
gen_event
mnesia_kernel_sup
proplists
zlib
mnesia_checkpoint
mnesia_locker
timer
io_lib
mnesia_index
sys
proc_lib
rpc
supervisor
ets
file
dets
lists
erlang