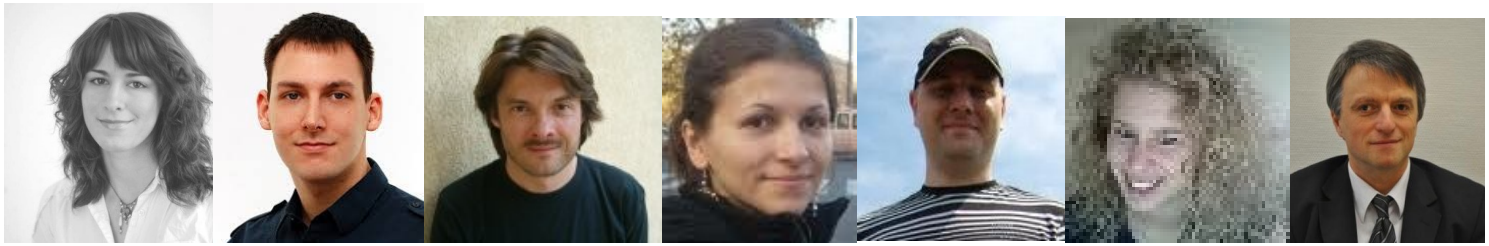
Parallelization by Refactoring - Labs -

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

Kozsik, T. et al.: Parallelization by Refactoring

Parallel Patterns for Adaptive Heterogeneous Multicore Systems

Kozsik, T. et al.: Parallelization by Refactoring

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

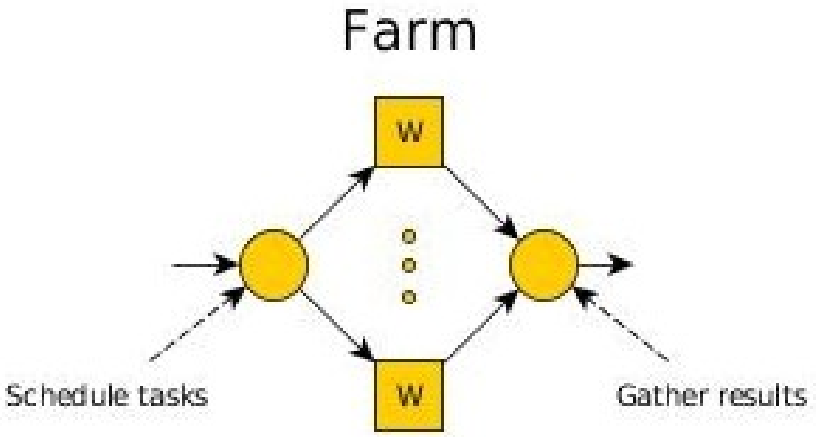
Kozsik, T. et al.: Parallelization by Refactoring

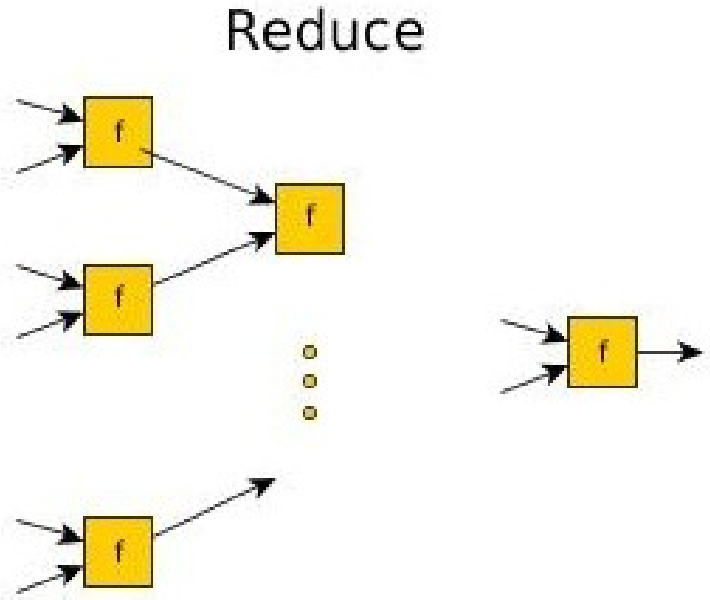
Pattern-based parallelism

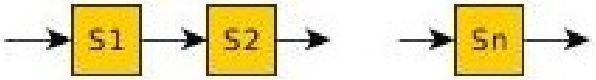
High-level approach to parallel programming

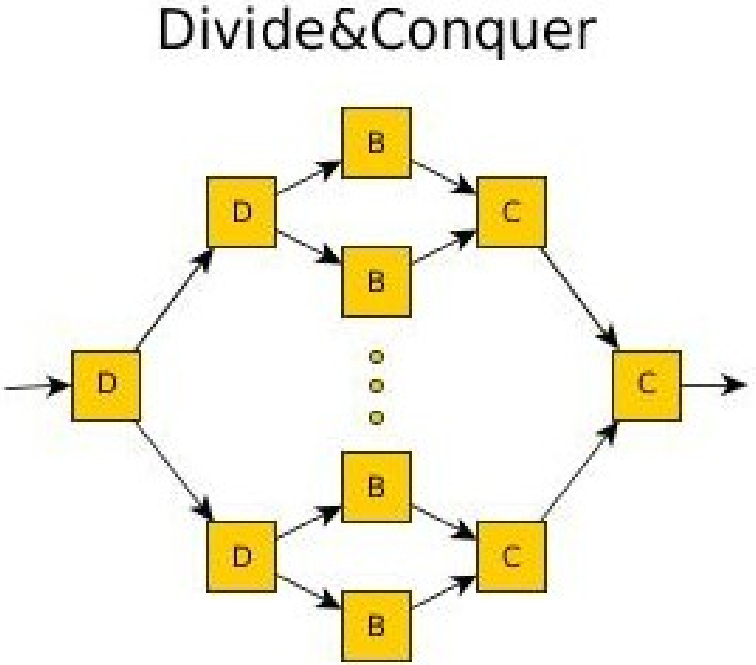
- Rely on a library of algorithmic skeletons
- Easier to develop code
- Easier to modify / maintain
- Better utilization of resources
  - Static resource management
  - Dynamic resource management

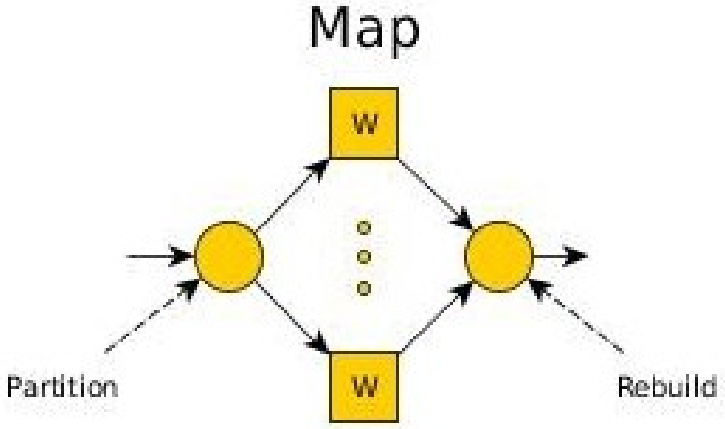
Kozsik, T. et al.: Parallelization by Refactoring





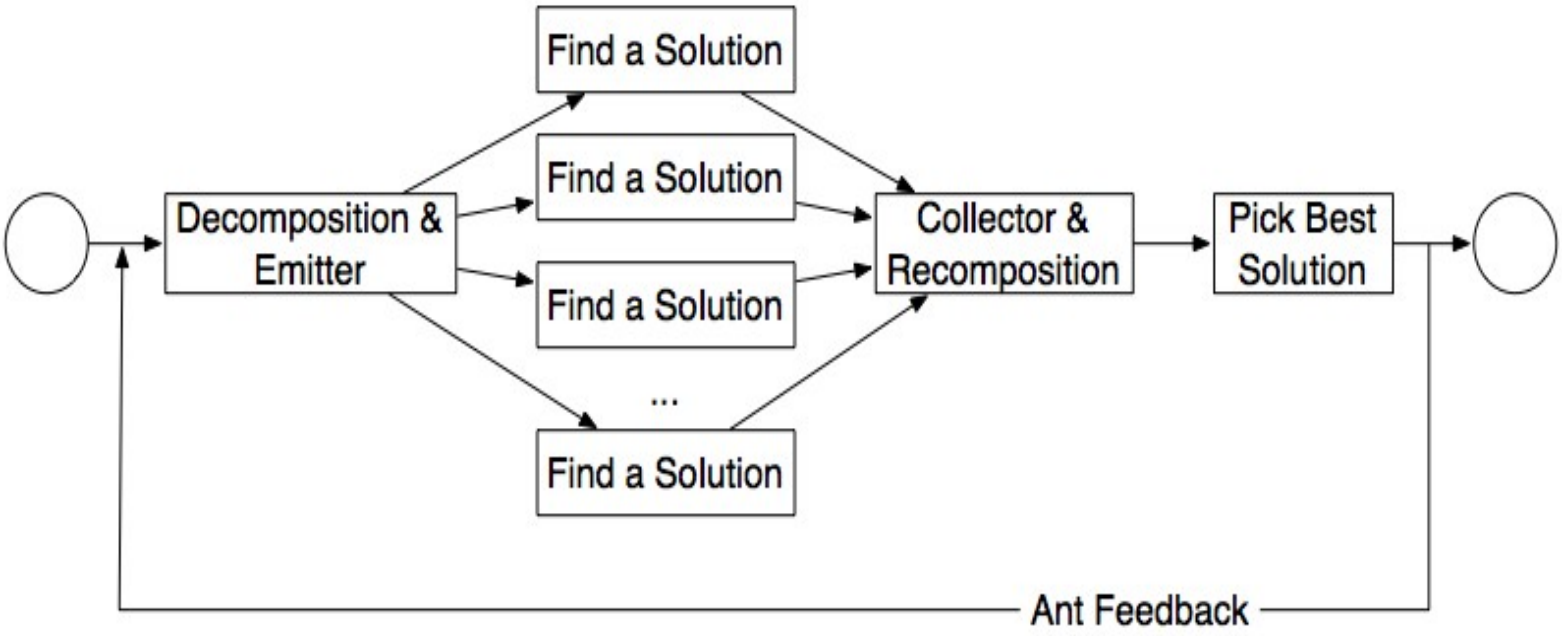






Kozsik, T. et al.: Parallelization by Refactoring

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

Kozsik, T. et al.: Parallelization by Refactoring



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 )
```

Kozsik, T. et al.: Parallelization by Refactoring

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}).
```

Kozsik, T. et al.: Parallelization by Refactoring

How shall I parallelize?

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

Kozsik, T. et al.: Parallelization by Refactoring

Where shall I parallelize?

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

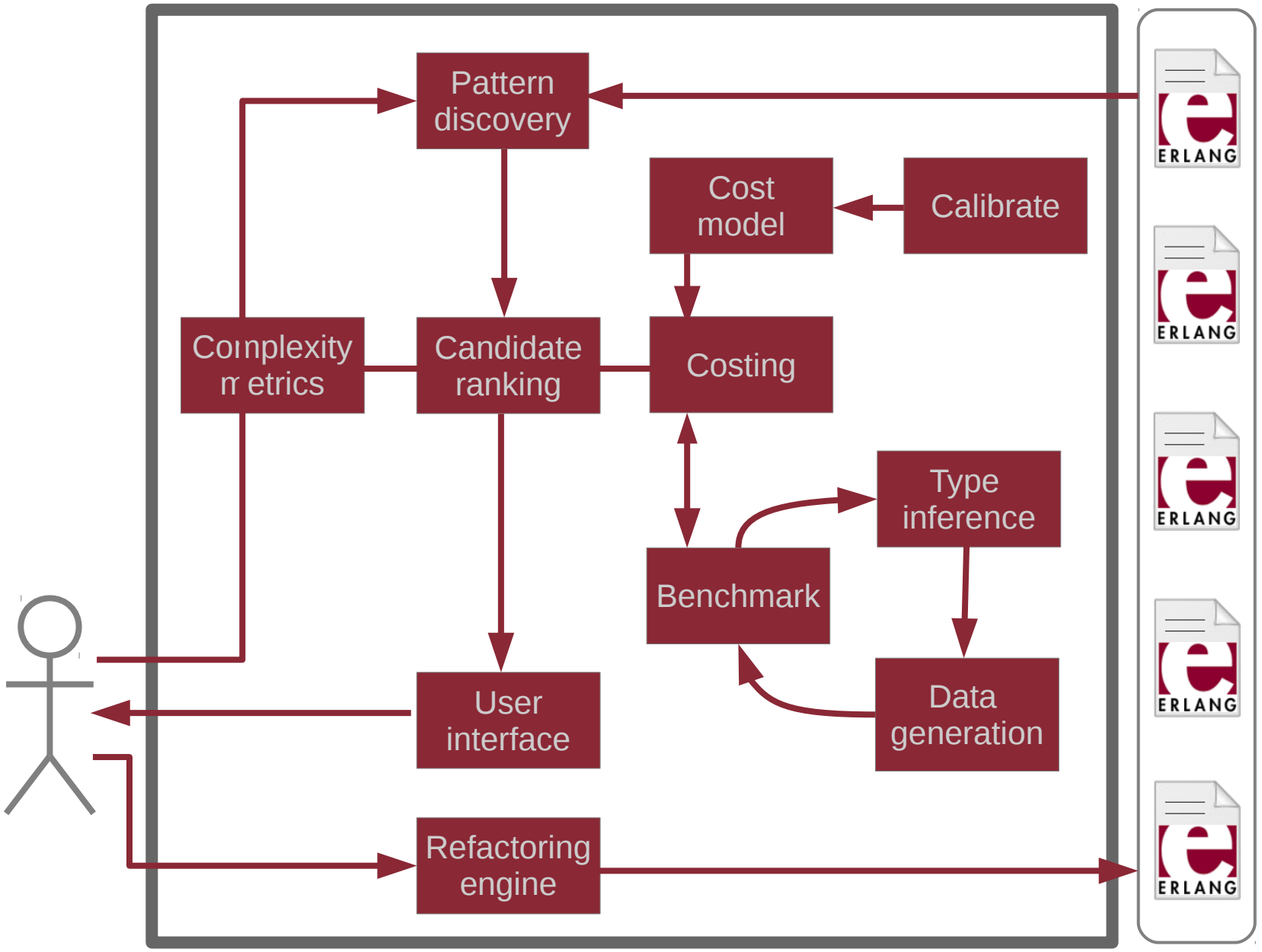
Kozsik, T. et al.: Parallelization by Refactoring

PaRTE

ParaPhrase Refactoring Tool for Erlang

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

Kozsik, T. et al.: Parallelization by Refactoring



Kozsik, T. et al.: Parallelization by Refactoring

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.

Kozsik, T. et al.: Parallelization by Refactoring

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

Kozsik, T. et al.: Parallelization by Refactoring

At CEFP

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

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

Kozsik, T. et al.: Parallelization by Refactoring

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/mas.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().

  refpp_api:run([{cand_types, [farm]}]).

  refpp_skeleton:find().
- Transformation

Kozsik, T. et al.: Parallelization by Refactoring

How to use Emacs?

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

Kozsik, T. et al.: Parallelization by Refactoring

Refactoring

- Program Shaping
- Introduction of Skeletons
- Cleanup Transformations

Either directly or after PC discovery

Kozsik, T. et al.: Parallelization by Refactoring

Your task

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

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

Kozsik, T. et al.: Parallelization by Refactoring

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).
```

Kozsik, T. et al.: Parallelization by Refactoring

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
```

Kozsik, T. et al.: Parallelization by Refactoring

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) ].
```

Kozsik, T. et al.: Parallelization by Refactoring

Resources

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

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

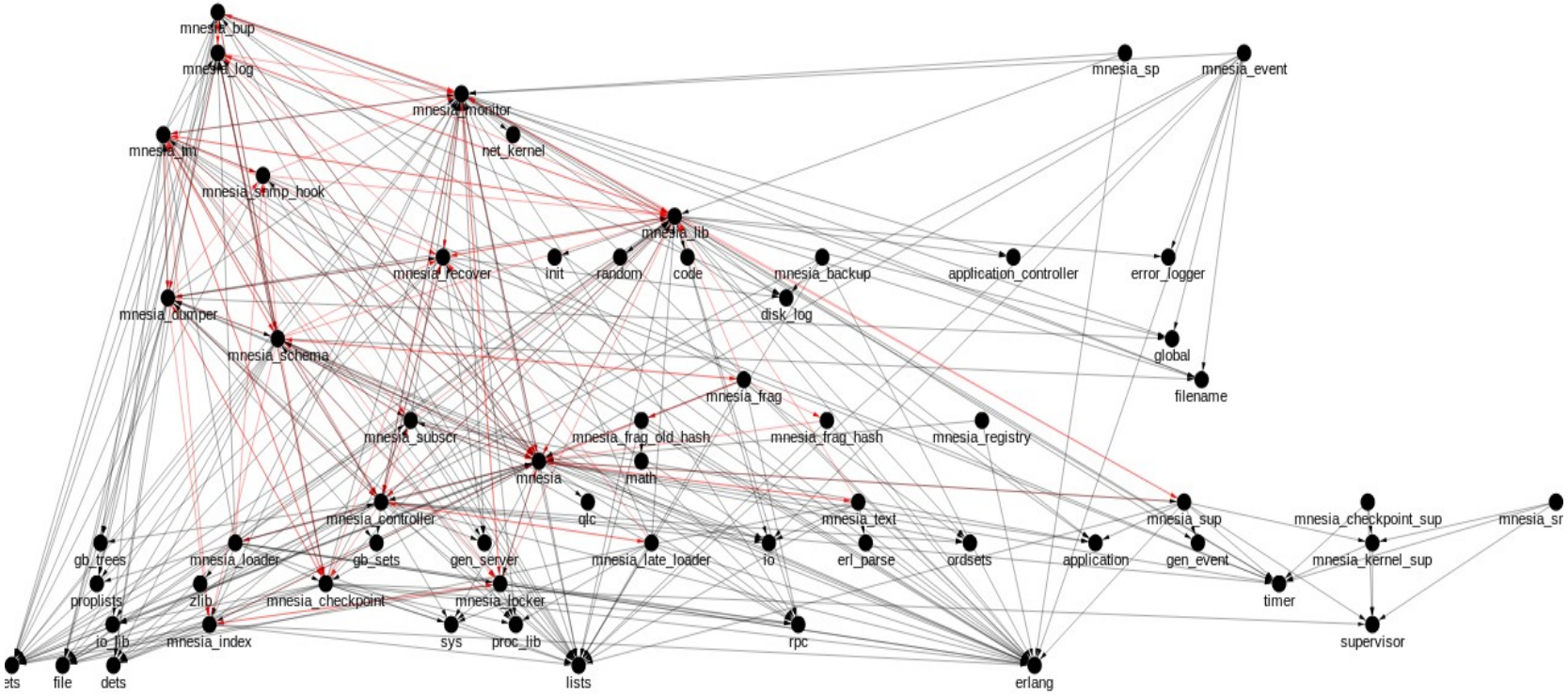
PaRTE docs & download:
http://pnyf.inf.elte.hu/trac/refactorerl/wiki/parte

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



Kozsik, T. et al.: Parallelization by Refactoring