# Parallel Evaluation of Interaction Nets with MPINE

Jorge Sousa Pinto*

Departamento de Informática
Universidade do Minho
Campus de Gualtar, 4710-057 Braga, Portugal
jsp@di.uminho.pt

**Abstract.** We describe the MPINE tool, a multi-threaded evaluator for Interaction Nets. The evaluator is an implementation of the present author's Abstract Machine for Interaction Nets [5] and uses POSIX threads to achieve concurrent execution. When running on a multi-processor machine (say an SMP architecture), parallel execution is achieved effortlessly, allowing for desktop parallelism on commonly available machines.

## Interaction Nets

Interaction Nets [3] are a graph-rewriting formalism where the rewriting rules are such that only pairs of nodes, connected in a specific way, may be rewritten. Because of this restriction, the formalism enjoys strong local confluence. Although the system has been introduced as a visual, simple, and inherently parallel programming language, translations have been given of other formalisms into Interaction Nets, specifically term-rewriting systems [1] and the $\lambda$-calculus [2, 4]. When used as an intermediate implementation language for these systems, Interaction Nets allow to keep a close control on the sharing of reductions.

Interaction Nets have always seemed to be particularly adequate for being implemented in parallel, since there can never be interference between the reduction of two distinct redexes.

Moreover there are no global time or synchronization constraints. A parallel reducer for Interaction Nets provides a reducer for any of the formalisms that can be translated into these nets, without additional effort.

We present here MPINE (for Multi-Processing Interaction Net Evaluator), a parallel reducer for Interaction Nets which runs on generic shared-memory multiprocessors, based on the **POSIX** threads library. The system runs notably on the widely available SMP architecture machines running the Unix operating system.

## A Concurrent Abstract Machine for Interaction Nets

In [5] the author has proposed an abstract machine for Interaction Net reduction, providing a decomposition of interaction steps into finer-grained operations. The

* Research done whilst staying at Laboratoire d'Informatique (CNRS UMR 7650), École Polytechnique. Partially supported by PRAXIS XXI grant BD/11261/97.

multi-threaded version of this machine is a device for the concurrent implementation of Interaction Nets on shared-memory architectures, based on a generalized version of the producer-consumers model: basic machine tasks are kept on a shared queue, from which a number of threads take tasks for processing. While doing so, new tasks may be generated, which will be enqueued.

Besides allowing for finer-grained parallelism than Interaction Nets themselves, the decomposition of interaction also allows for improvements concerning the synchronization requirements of the implementation. In particular, it solves a basic deadlock situation which arises when one naively implements Interaction Nets in shared-memory architectures, and it lightens the overheads of synchronization – the number of mutexes required by each parallel operation is smaller than that required by a full interaction step, which may be quite significant.

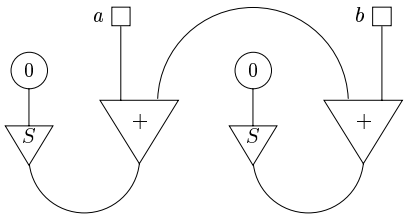
We direct the reader to [6] for details on these issues.

## MPINE

The abstract machine may be implemented on any platform offering support for multi-threaded computation. MPINE, which uses **POSIX** threads, is, to the best of our knowledge, the first available parallel reducer for Interaction Nets.

The program has a text-based interface. The user provides as input an interaction net written in a language similar to that of [3], and the number of threads to be launched (which should in general be equal to the number of available processors in the target machine). The output of the program (if the input net has a normal form) is a description of the reduced net in the same language.

We give a very simple example in which we declare three agents (Zero, Successor, Addition) that will allow us to implement the sum of Natural numbers by rewriting with Interaction Nets, using the usual inductive definition.

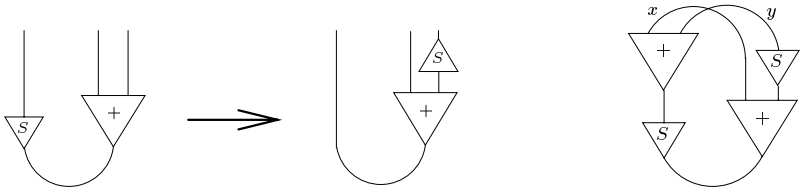



This net represents the two equations $S(0) + a = x$ and $S(0) + x = b$, or simply $S(0) + (S(0) + a) = b$. Each cell has a unique principal port. For + cells this represents the first argument of the sum. The file `example.net` contains a description of the net together with the interaction system in which it is defined:

```
agents
        Z       0;
        S       1;
        A       2;
rules
```

```
        A(x,x) >< Z;
        A(x,S(y)) >< S(A(x,y));
net
        S(Z) = A(a,x);
        S(Z) = A(x,b);
interface
        a;
        b;
end
```

The `agents` section contains declarations of agents, with their arity; then the `rules` and the interaction `net` are given as sequences of active pairs, written as equations. The file ends with the `interface` of the net, a sequence of terms. Interaction rules rewrite *active pairs* of cells connected through their principal ports. An example is given below, corresponding to the (term-rewriting) rule $S(y)+x \longrightarrow S(y+x)$. Interaction rules are written as pairs of terms by connecting together the corresponding free ports in both sides of each rule, as shown:




The following is an invocation of the reducer with 4 threads, with the above file as input, followed by the result produced:

```
> mpine -4 example.net
==============Initial Net===============
Displaying net equalities...
        S(Z) = A(x0,x1)
        S(Z) = A(x2,x0)
Displaying observable interface...
        x2
        x1
===============Reduced Net==============
Displaying observable interface...
        x0
        S(S(x0))
```

The input net is printed followed by the reduced net. Since this is a net with no cycles, there are no equations left (if there were they would be displayed). The system may also be instructed to print some statistics, including the number of tasks performed by each individual thread. The reader is referred to the user's guide for more information on additional features of the system.

### Some Benchmark Results

We show a set of benchmark results for the reduction of nets obtained from $\lambda$-terms using the YALE translation of [4]. The terms are Church numerals, which we have selected simply because they generate abundant computations.

| term | N.int | seq.red | $par_2$red | $par_2$/seq % |
|---|---|---|---|---|
| **2232II** | 37272 | 5.02 | 5.54 | 110 |
| **423II** | 105911 | 29.8 | 15.7 | 52.7 |
| **333II** | 473034 | 552 | 310 | 56.2 |
| **2233II** | 1417653 | 7096 | 5381 | 75.8 |

For each term we show the number of interactions and the time taken to reduce the corresponding net, both by a sequential reducer (free of the synchronization overheads) and by MPINE running on a 2 processor machine. We also show the ratio of the two. These preliminary results are promising: in two of the above nets, the ideal goal of reducing by half the execution time is practically attained.

### Availability

MPINE is written in C. The distribution contains a user's guide and some example files. It is available as a statically linked binary for Linux i386 ELF from the author's homepage. A sequential reducer is also available.

### Future Work

Further optimizations for MPINE are proposed in [6], which have yet to be incorporated in the implementation. It also remains to test the implementation with a large set of terms, notably in machines with more than two processors.

## References


1. Maribel Fernández and Ian Mackie. Interaction nets and term rewriting systems. *Theoretical Computer Science*, 190(1):3–39, January 1998.
2. Georges Gonthier, Martín Abadi, and Jean-Jacques Lévy. The geometry of optimal lambda reduction. In *Proceedings of the 19th ACM Symposium on Principles of Programming Languages (POPL'92)*, pages 15–26. ACM Press, January 1992.
3. Yves Lafont. Interaction nets. In *Proceedings of the 17th ACM Symposium on Principles of Programming Languages (POPL'90)*, pages 95–108. ACM Press, January 1990.
4. Ian Mackie. YALE: Yet another lambda evaluator based on interaction nets. In *Proceedings of the 3rd ACM SIGPLAN International Conference on Functional Programming (ICFP'98)*, pages 117–128. ACM Press, September 1998.
5. Jorge Sousa Pinto. Sequential and concurrent abstract machines for interaction nets. In Jerzy Tiuryn, editor, *Proceedings of Foundations of Software Science and Computation Structures (FOSSACS)*, number 1784 in Lecture Notes in Computer Science, pages 267–282. Springer-Verlag, 2000.
6. Jorge Sousa Pinto. *Parallel Implementation with Linear Logic (Applications of Interaction Nets and of the Geometry of Interaction)*. PhD thesis, École Polytechnique, 2001.