Partial Evaluation of Interaction Nets.

Denis BECHET*

* Laboratoire d'Informatique de l'École Normale Supérieure
75230 PARIS CEDEX 05
URA 1327 du CNRS
e-mail `bechet@ens.ens.fr`

1.   Introduction

In recent years, partial evaluation has once again become the object of considerable interest. Its general principle consists in specializing a process when one or more of its inputs are given. This extended abstract describes how this concept can be applied to interaction nets. After a brief introduction to interaction nets and box nets, we present the three main mechanisms of specialization: pre-evaluation, abbreviations and the use of laws.

2.   Interaction nets

Interaction nets introduced by Yves Lafont in [Laf90]. For a full study, see [Gir87, Laf90, Laf91].

They come in the form of undirected labelled graphs coupled up with rewriting rules. Each vertex which we call an *agent* belongs to a class which consists of a name, a *principal port* and *auxiliary ports*. A net is constructed by connecting the ports of the agents in twos using *links*. The principle of interaction requires that two agents and only two interact when they are linked by their principal ports. Such a couple is called an *active pair*. It follows that the left member of a rule consists always of two agents connected by their principal ports. The right member can be any net with the same free variables as the left one (linearity) and with no alive pair (optimization). The reduction mechanism replaces any alive pair by the right part of the rule defining the interaction between those two agents.
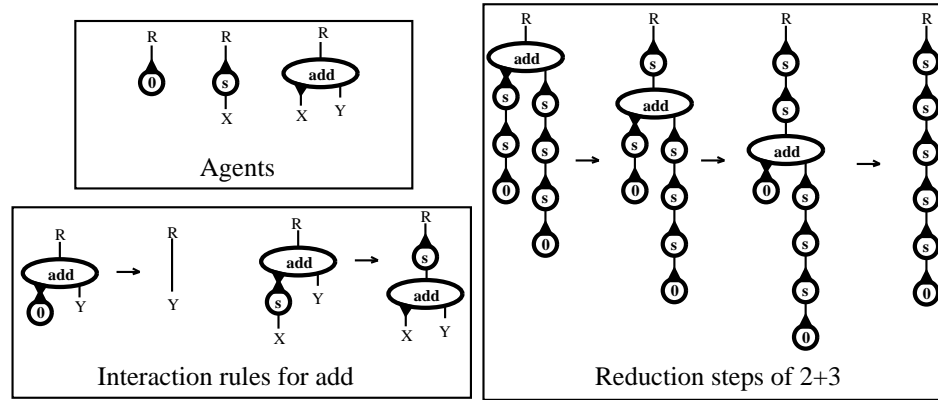
Figure 1. Interaction nets: unary arithmetics.

For instance, unary integers are constructed from the two agents S and 0 which have respectively one and no auxiliary ports. The addition operator has three ports, two for the arguments and one for the result. Figure 1 gives samples of interaction nets.

3.   Box nets

From a more denotational point of view, agents, rules, and the distinction between principal and auxiliary ports are not relevant because they reflect a choice of implementation. By replacing each agent of an interaction net by a box, we obtain an equivalent box net. Each agent corresponds to a box with the same name and the same list of ports.

The only significant notion in box nets is the definition of new boxes using more basic ones. A box can be associated either to an agent or to a box net. The two operations which can be applied to a box net are the replacement of a box by its definition (unfolding) and the inverse (folding). Of course, to each box net there corresponds an interaction net obtained by successive unfoldings of the boxes (in this abstract, we assume non-recursive definitions).

So, the standard evaluation of a box net consists in first obtaining the corresponding interaction net, then reducing it. The last operation tries to reconstruct the boxes from the result. Figure 2 gives some examples of
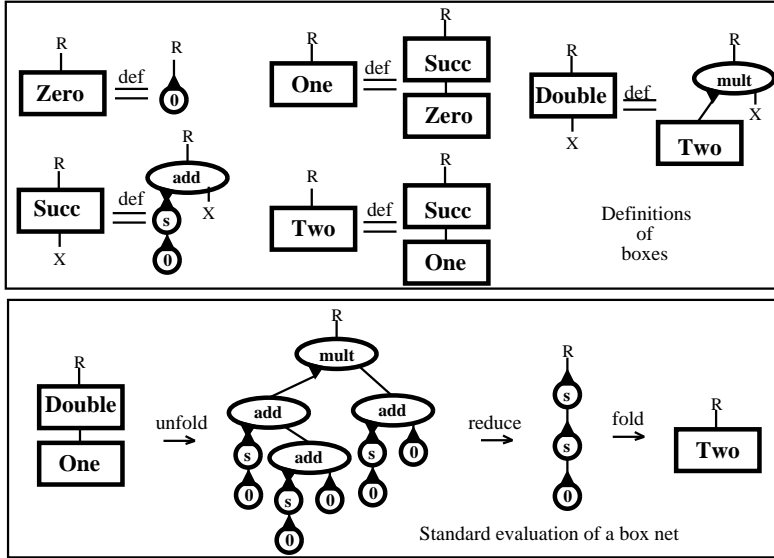
Figure 2. Box nets.

definitions and a standard evaluation.

4. Pre-evaluation

We have seen that boxes come without any notion of calculus. A box can represent a constant (which is reduced like `Zero` in Figure 2), a way to express a constant using combinators (for instance, constructing the integers from the boxes `Zero` and `Succ` even if their real implementation is not in unary arithmetics — see the definition of `One`) or a process (the box `Double` is defined as the multiplication of the argument by two).

So, the translation of a box into an interaction net can be reduced. Boxes `Succ`, `One`, `Two` and `Double` in Figure 2 are examples of this. The first idea of partial evaluation consists in replacing the definition of a box by the net obtained after reduction. Figure 3 shows the pre-evaluation of box `One`.

This technique of pre-evaluation seems attractive and gives good results, but is not able to modify the inherent effectiveness of algorithms. It merely eliminates a constant number of reductions per box. To improve the mechanism of evaluation further, we need to introduce new agents and new
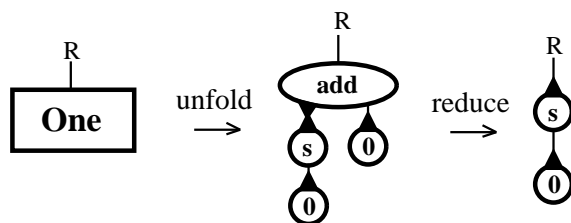
Figure 3. Pre-evaluation of the box `One`.

rules.

5.   Abbreviations

This mechanism attempts to identify a box with an agent and to introduce rules for it. Since in general a box cannot be seen as an agent because the latter only has one principal port, the interaction net associated to a box is split into sub-nets which can be abbreviated.

For each abbreviable net, a new agent is defined. It will interact with the agents that interact with the principal agent of the abbreviation (the agent whose principal port is free). For each pair, a new rule is computed. For instance, if we set `incr` the abbreviation for the addition of the argument and one (see Figure 4), the agent `incr` will interact with `S` and `0` because the principal agent is `add` and it interacts with `S` and `0`. So, `incr` will be defined by two rules. This mechanism is similar to the definition/unfold/fold transformation of Burstall and Darlington [BD77] or Sato et Tamaki [ST84]. This specialization is especially efficient when the boxes represent the emulation of a process by a low level machine: evaluation of a program by an interpretor, the use of primitive data structures to introduce complex ones, emulation of a process by pre-defined functions. In consequence, abbreviations give good results for the specialization of an interpretor, for pattern-matching, and for the production of modules and new kinds of data. The improvements are linear for the number of reduction steps and in particular, it is not possible to transform an exponential algorithm into a polynomial one.

The first mechanism of partial evaluation can be seen as a pre-compilation. This second one finds its motivation in the notion of modularity
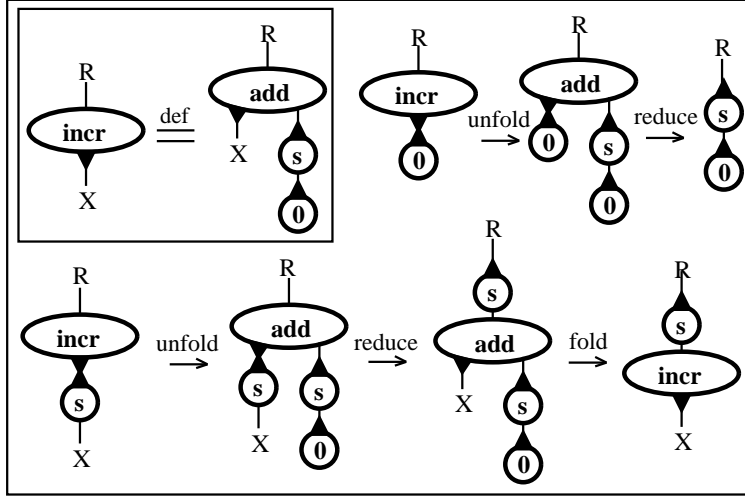
Figure 4. Introduction of a new agent and of its rules.

and the elimination of abstract levels. The program tries to stay very close to the objects that were defined by the programmer, without modifying the algorithm complexity. So, to improve the programs even further, we will use equivalences of interaction nets and laws which enable us to modify the process, to eliminate redundant or duplicated computation.

6. Behavioural equivalence and laws

Some agents have the same behaviour as the identity. They can be eliminated. Others are similar and can be identified. Certain algebraic properties are sometimes useful — for example the associativity and commutativity of the addition. Finally, explicit duplication and erasing due to the linearity of interaction nets sometimes creates redundant copies which are erased later.

In the light of all these considerations, a set of local transformations has been fixed. They are based on a behavioural equivalence. Two nets are equivalent towards one or several ports if, when we connect them to the same net, the results on the indicated ports are coherent (see [Laf88, GLT89] for coherence spaces). In general, this kind of equivalence is partially correct (the termination property can be not respected). However, we have been

able to find constraints which guarantee total correctness.

The results of this mechanism are sometimes surprising even for simple transformations such as the elimination of the couple duplication/erasing, the merging of two identical nets whose arguments are copies of the same variable (for example, the transformation of a recursive definition of Fibonacci's function to a linear calculus), the use of laws (the transformation of a recursive definition of the reverse function on lists to a definition with accumulator. See Figure 5), etc. Moreover, those transformations allow us to justify the laws and, consequently, to see more clearly the limits of their use.

A comparison with other systems of partial evaluation has convinced us that interaction nets bring the "eurekas" that are needed by specialization programs. The fact that the rules are local, the absence of a distinction between constructors and destructors, the linearity of interaction nets and their multiple results enabled us to circumscribe the search for definitions.

7.   Conclusion

Thus, interaction nets seem well suited to the mechanisms of partial evaluation. In addition to the simple reduction that occurs when several nets are juxtaposed (which is a consequence of the transformation of a box into an interaction net) and in addition to a system that introduces new agents and new rules, we have been able to define a set of transformations based on a behavioural equivalence. A comparison with other specialization systems shows the advantage of using interaction nets in partial evaluation. This advantage is specifically due to the features mentioned above: local aspect, linearity, the explicit duplications/erasures, and the non-differentiation between constructors and destructors.

References

[Bec90]   Bechet, D., 'Les Abréviations dans les réseaux d'interactions', *Rapport de D.E.A.*, Université Paris 7, 1990.

[Bec91]   Bechet, D., 'Problèmes d'évaluation partielle dans les réseaux d'interactions', *Rapport interne du laboratoire d'informatique de l'école normale supérieure*, 1991.

[BD77]   Burstall, R.M. and Darlington, J., 'A transformation system for developing recursive programs', *Journal of the Association for Computing Machinery*,
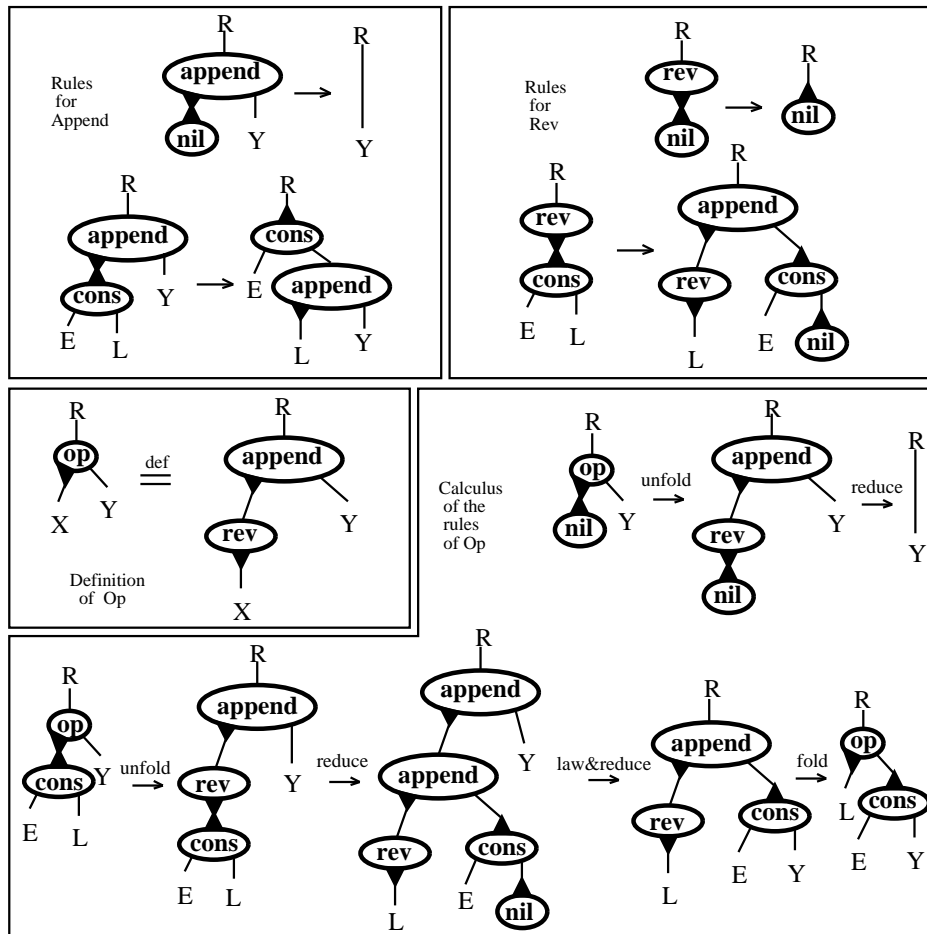
Figure 5. Inversion of a list.

vol. 24, No. 1, January 1977, pp. 44–67.

[Gir87]   Girard, J.Y., 'Linear Logic', *Theoretical Computer Science*, No. 50, 1987, pp. 1–102.

[GLT89]   Girard, J.Y., Lafont, Y. and Taylor, P., *Proofs and Types*, Cambridge University Press, 1989.

[Laf88]   Lafont, Y., 'Introduction to Linear Logic', *Lectures notes for the Summer School on Constructive Logis and Category Theory*, Isle of Thorns, August 1988.

[Laf90]   Lafont, Y., 'Interaction Nets', *Proceedings of Seventeenth ACM Symposium on Principes of Programming Languages*, 1990, pp. 95–108.

[Laf91]   Lafont, Y., 'The Paradigm of Interaction', *Proceedings of the Workshop on Concurrency*, Programming Methodology Group, Göteborg, April 1992, pp. 276–293.

[ST84]   Sato, T. and Tamaki, H., 'Unfold/Fold Transformation of Logic Programs', *Proceedings of the Second International Logic Programming Conference*, Uppsala, 1984, pp. 127–138.