Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 204 (2008) 181–198
www.elsevier.com/locate/entcs

Token-passing Nets for Functional Languages

José Bacelar Almeida, Jorge Sousa Pinto, Miguel Vilaça [1]

CCTC / Departamento de Informática
Universidade do Minho
4710-057 Braga, Portugal

Abstract

Token-passing nets were proposed by Sinot as a simple mechanism for encoding evaluation strategies for the λ-calculus in interaction nets. This work extends token-passing nets to cover a typed functional language equipped with structured types and unrestricted recursion. The resulting interaction system is derived systematically from the chosen big-step operational semantics. Along the way, we actually characterize and discuss several design decisions of token-passing nets and extend them in order to achieve simpler interaction net systems with a higher degree of embedded parallelism.

Keywords: Interaction nets, reduction strategies, λ-calculus, recursion.

1 Introduction

Interaction nets [7] constitute a Turing-complete computational paradigm, where computation is purely local, and thus (strong) confluence holds.

The linear λ-calculus can be very naturally encoded in interaction nets with just two symbols. When one drops the linearity restriction however, things become more subtle: since variable substitution is implemented within the formalism, and not as an external meta-operation, copying and erasure of terms must be dealt with explicitly. Many encodings have been studied (e.g. [9,10,8]), some of which allow for a great degree of sharing of computations.

Token-passing nets [11] were proposed as a simple mechanism for encoding the most common evaluation strategies for the λ-calculus in the interaction net framework. One of the most attractive features is their simplicity, allowing computations to be easily traced in the term syntax. This makes them particularly well-suited for debugging or educational purposes.

The purpose of this paper is two-fold:

[1] Emails: {jba,jsp,jmvilaca}@di.uminho.pt. The work of the 3rd. author was funded by FCT grant SFRH / BD / 18874 / 2004.

doi:10.1016/j.entcs.2008.03.061

(i) To explicitly characterize token-passing nets as a class of interaction nets;

(ii) to extend the token-passing encoding of the λ-calculus to a typed functional language with structured types. In particular, we propose a novel way of encoding arbitrary recursion with a fixpoint construct.

Sinot imposes a linearity restriction on the evaluation token, which forces evaluation to proceed sequentially. A novelty of our approach to token-passing is that sequentiality is not taken to be a defining attribute, since we believe that relaxing the sequentiality constraint gives rise to more natural (and considerably simpler) encodings for the strategies considered. This allows us to encode parallel call-by-value for our functional language.

For the sake of generality, we use the framework of Combinatory Reduction Systems to specify the syntax of our terms with binding, and we give a generic encoding of CRS terms into a class of syntactical nets. CRS syntax gives us a generic way of combining first-order term rewriting with λ-style bindings, which is very appropriate for our development in this paper.

The paper is structured as follows: Section 2 reviews basic notions of interaction nets and encodings of the λ-calculus, including the token-passing encoding. Section 3 introduces a notion of nets capable of representing the syntax of a large class of term languages with binding. Section 4 is devoted to the characterization of a general class of token-passing nets and systems, and to the description of the token-passing system for the λ-calculus. We proceed to give in Section 5 an encoding of a functional language with recursion. Section 6 concludes with some remarks, extensions, and pointers to further work.

2 Interaction Nets and λ-Calculus Encodings

An interaction net system [7] is specified by giving a set Σ of symbols, and a set $\mathcal{R}$ of interaction rules. Each symbol $\alpha \in \Sigma$ has an associated (fixed) *arity*. An occurrence of a symbol $\alpha \in \Sigma$ will be called an *agent*. If the arity of α is n, then the symbol has $n+1$ *ports*: a distinguished one called the *principal port*, and n *auxiliary ports* labelled $x_1, \ldots, x_n$. A net built on Σ is a graph (not necessarily connected) where the nodes are agents. The edges between nodes of the graph are connected to ports in the agents, such that there is at most one edge connected to every port in the net. Edges may be connected to two ports of the same agent. Principal ports of agents are depicted by an arrow. The ports where there is no edge connected are called the *free ports* of the net. The set of free ports define the *interface* of the net.

There are two special instances of a net: a wiring (a net containing no agents, only edges between free ports), and the empty net (containing no agents and no edges). The dynamics of Interaction Nets are based on the notion of *active pair*: any pair of agents (α, β) in a net, with an edge connecting together their principal ports. An *interaction rule* $((\alpha, \beta) \to N) \in \mathcal{R}$ replaces an occurrence of the active pair (α, β) by the net N. Rules must satisfy two conditions: the interfaces of the left-hand side and right-hand side are equal (this implies that the free ports are preserved during reduction), and there is at most one rule for each pair of symbols,