ABSTRACT  Actor systems are a model of massively parallel systems based on asynchronous message passing. This paper presents a formalism for actor systems in the framework of graph grammars. To this aim actor grammars are introduced, motivated, and illustrated by examples. Some of the basic properties pertinent to derivations in actor grammars are discussed.

Key words: massive parallelism, actor systems, neighbourhood controlled embedding, handle rewriting, actor grammars.

CONTENTS

Introduction
1. Preliminaries
2. Actor systems
3. Actor vocabularies and configuration graphs
4. Actor grammars
5. Discussion

INTRODUCTION

In the last few years there has been a growing interest in various models of concurrent computation. One of these is the Actor paradigm, which is based on asynchronous message passing. Actor systems have been introduced in [H77] and [HB77]. Most of the research on actor systems presented in the literature is done within a rather informal framework (see, e.g., [H77], [HB77], [L81] and [T83]), although some more formal approaches exist (see, e.g., [C81] and [A86]). We refer
to Section 2 for a short introduction to actor systems.

The aim of this paper is to present a formal framework for (a version of) actor systems, which is based on the use of graph grammars. One of the basic decisions to be made in setting up such a framework is the choice of a suitable representation of a configuration (a "snapshot") of an actor system. It seems rather natural to represent a configuration by a graph. Hence to each configuration corresponds a graph and to a computation in an actor system corresponds a sequence of graph transformations. Since graph grammars provide an elegant way to describe such transformations, we use them to model the dynamic behaviour of actor systems. It turns out that the graph grammars we need, called actor grammars, are based on features which are well-known in graph grammar theory: they are handle-rewriting systems, where both the rewriting and the embedding are based on (node and edge) labels, and the embedding process is confined to the direct neighborhood of the rewritten handles (see e.g., [ENR83], [JR83] and [HK83]).

1. PRELIMINARIES

In this section we recall some basic notions and terminology concerning graphs and sets. This allows us to set up a notation suitable for this paper.

(1) Sets and relations
(1.a) For a set X, \( I_X \) denotes the identity relation on X; for sets X, Y, X - Y denotes the difference of X and Y.
(1.b) Let A, B be sets and let \( R \subseteq A \times B \). Then \( R^{-1} \) denotes the inverse of R, and for a subset C of A, \( R(C) = \{ y \in B \mid \text{there exists an } x \in C \text{ such that } (x,y) \in R \} \). If \( C = \{ x \} \), then we write \( R(x) \) instead of \( R(C) \). A function is considered to be a set of pairs, and the inverse of a function \( \xi \) is denoted by \( \xi^{-1} \).
(1.c) Let A, B and C be sets, let \( R \subseteq A \times B \) and let \( S \subseteq B \times C \). Then \( S \circ R \) denotes the composition of \( R \) and \( S \) (first \( R \), then \( S \)).

(2) Graphs
(2.a) Let \( \Sigma \) and \( \Delta \) be finite nonempty sets. A \((\Sigma,\Delta)\)-labeled graph is a system \( g = (V,E,\phi) \) where V is a finite set (called the set of nodes of g), E is a subset of \( V \times \Delta \times V \) (called the set of edges of g) and \( \phi \) is a function from V into \( \Sigma \) (called the node-labeling function of g).
(2.b) Let \( g = (V,E,\phi) \) be a \((\Sigma,\Delta)\)-labeled graph. The source function of g is the function \( s \) from E into V defined by \( s(a,\delta,b) = a \), for each \( (a,\delta,b) \in E \). The target function of g is the function \( t \) from E into V defined by \( t(a,\delta,b) = b \), for each \( (a,\delta,b) \in E \). The edge-labeling function of g is the function \( \psi \) from E into \( \Delta \), defined by \( \psi(a,\delta,b) = \delta \), for each \( (a,\delta,b) \in E \).