ELSEVIER

Theoretical Computer Science 194 (1998) 239–248

Theoretical Computer Science

Contents and Abstracts of the Electronic Notes in Theoretical Computer Science Vol. 2

Proceedings Joint COMPUGRAPH/SEMAGRAPH Workshop on Graph Rewriting and Computation (SEGRAGRA '95), Volterra, Pisa, Italy, 28 August–1 September 1995

Andrea Corradini and Ugo Montanari (Guest Editors)

Preface

This volume contains the *Proceedings* of SEGRAGRA '95, the Joint COMPUGRAPH/SEMAGRAPH Workshop on Graph Rewriting and Computation. The Workshop was held in the "Centro Studi Santa Maria Maddalena" of Volterra, near Pisa, Italy, from 28 August to 1 September 1995.

SEGRAGRA '95 addressed the area of Graph Rewriting and Term Graph Rewriting. It was the final workshop of two Esprit Basic Research Working Groups, COMPUGRAPH II and SEMAGRAPH II. Both working groups have been active in the three year period 1992–1995. They have decided to organize their final workshop as a common initiative, with the aim of presenting the achieved results and their view of the state of the art in the field.

Out of 39 submitted papers, the Program Committee selected 23 for presentation at the Workshop. These were grouped into sessions on Graph Rewriting I and II, Term Graph Rewriting I and II, Computing on Graphs, Concurrency & Parallelism, Modules, Software Engineering, Types and Proofs. The program included also invited talks by Eike Best (Universität Hildesheim), Thomas Johnsson (Chalmers University), Detlef Seese (Universität Karlsruhe) and Chris Wadsworth (Rutherford Appleton Laboratory); and tutorials by Henk Barendregt (Katholieke Universiteit Nijmegen), Andrea Corradini (Università di Pisa), Bruno Courcelle (Université Bordeaux I), Hartmut Ehrig (Technische Universität Berlin), Richard Kennaway (University of East Anglia, Norwich), Hans-Jörg Kreowski (Universität Bremen), Rinus Plasmeijer (Katholieke Universiteit Nijmegen) and Ronan Sleep (University of East Anglia, Norwich).

The Conference site was made available at no cost to SEGRAGRA '95 by the Cassa di Risparmio di Volterra. The Workshop was supported by Gruppo Nazionale Informatica Matematica (CNR) and Istituto di Elaborazione dell'Informazione (CNR).

We thank the anonymous referees, the Program Committee members:

Bruno Courcelle	(*Université de Bordeaux I*)
Hartmut Ehrig	(*Technische Universität Berlin*)
Chris Hankin	(*Imperial College, London*)
Dirk Janssens	(*University of Antwerp*)
Dirk Janssens	(*University of Antwerp*)
Jan Willem Klop	(*CWI, Amsterdam*)
Hans-Jörg Kreowski	(*Universität Bremen*)
Lone Leth	(*ECRC, München*)
Ugo Montanari **(Chair)**	(*Univerità di Pisa*)
Rinus Plasmeijer	(*Katholieke Universiteit Nijmegen*)
Jean-Claude Raoult	(*Université de Rennes*)
Grzegorz Rozenberg	(*University of Leiden*)
Ronan Sleep	(*University of East Anglia, Norwich*)

PII S0304-3975(96)00224-7

and the Organizing Committee members:

Andrea Corradini (**Chair**)	(*Università di Pisa*)
Stefania Gnesi	(*IEI-CNR, Pisa*)
Andrea Maggiolo-Schettini	(*Università de Pisa*)
Ugo Montanari	(*Università di Pisa*)
Ronan Sleep	(*University of East Anglia, Norwich*)

Fabio Gadducci and Marco Pistore from the University of Pisa gave us a valuable help in the local organization during the Workshop. We also thank the Managing Editors of the Electronic Notes in Computer Science series, Michael Mislove, Maurice Nivat, and Christos Papadimitriou, for giving us the opportunity of publishing the proceedings in this new electronic series, and Mike Mislove and Joost W.M. Kok (from Elsevier) for the constant support and the valuable suggestions they provided us during the preparation of this electronic volume.

Richard Banach
DPO rewriting and abstract semantics via opfibrations
http://www.elsevier.nl/locate/entcs/volume2.html

The classical DPO graph rewriting construction is re-expressed using the opfibration approach introduced originally for term graph rewriting. Using a skeleton category of graphs, a base of canonical graphs-in-context, with DPO rules as arrows, and with categories of redexes over each object in the base, yields a category of rewrites via the discrete Grothendieck construction. The various possible ways of combining rules and rewrites lead to a variety of functors amongst the various categories formed. Categories whose arrows are rewriting sequences have counterparts where the arrows are elementary event structures, and an event structure semantics for arbitrary graph grammers emerges naturally.

Erik Barendsen and Sjaak Smetsers
A derivation system for uniqueness typing
http://www.elsevier.nl/locate/entcs/volume2.html

Traditional functional programming languages are unable to deal with operations with side effects. In recent years, various proposals have been brought up as solutions to this shortcoming. One of these is the so-called uniqueness type system which has been incorporated in the functional language Clean. Originally, this type system was presented as an extension of a Curry style typing system for graph rewrite system. The present paper describes a restricted version of our original system, using an inductive syntax and natural deduction style type assignment system. This captures the core of syntax and natural deduction style type assignment system. This captures the core of uniqueness typing, and makes the relation with linear logic more visible. Apart from this, it makes the work on uniqueness more accessible, especially for people not familiar with graph rewriting. For the conventional as well as for the uniqueness type system, we prove preservance of typing during reduction and the existence of principal types.

Michel Bauderon
Parallel rewriting of graphs through the pullback approach
http://www.elsevier.nl/locate/entcs/volume2.html

In this paper, we continue the development of our description of the pullback approach to graph rewriting – already shown to encompass both the NCE and the double-pushout approach, by describing parallel application of rewriting rules. We show that this new framework provides a genuine definition of parallel rewriting: parallel application of several rewriting rules at several different places in the graph is actually expressed through the application of one single mathematical operation, and even further that a deterministic graph grammer can be described by a single rule which we call a P-grammar.

Eike Best and Maciej Koutny
Using net refinement to compute the fixpoint of a recursive expression
http://www.elsevier.nl/locate/entcs/volume2.html

The talk illustrates the general Petri net semantics of equations such as $X = term(X)$ using a basic CCS-like process algebra without restriction, synchronisation and relabelling.

Simon Brock and Gerald Ostheimer
A process semantics for functional programming
http://www.elsevier.nl/locate/entcs/volume2.html

The semantics of lazy functional programming languages is usually presented in two different ways: a semantics based on trees which is used to reason about a program, and a semantics based on graphs which is used to implement a program. The link between these semantics is often unclear. We present a process semantics for functional programming which has a number of interesting properties. First, it is structured in such a way that the relationship between the tree and graph semantics is clear. Second, it captures the main requirements of functional programming by incorporating laziness, cycles and strictness, Finally, there is a simple formal correspondence between this semantics and other operational presentations.

David Clark and Richard Kennaway
Some properties of non-orthogonal term graph rewriting systems
http://www.elsevier.nl/locate/entcs/volume2.html

This paper examines left-linear non-orthogonal term graph rewriting systems that allow asymmetric conflicts between redexes. Using a definition of compatibility of sequences based on Boudol's work on the semantics of term rewriting, it shows that two properties associated with functional languages are true of such graph rewriting systems. First, that a notation of standard computation can be defined and that an associated standardisation theorem can be proved. Second, that a set of events can be associated with a reduction sequence and hence event structures modelling all the possible reduction sequences from a given initial graph can be constructed.

Andrea Corradini and Reiko Heckel
A compositional approach to structuring and refinement of typed graph grammars
http://www.elsevier.nl/locate/entcs/volume2.html

Based on a categorical semantics that has been developed for typed graph grammars, we use colimits (pushouts) to model composition and (reverse) graph grammar morphisms to describe refinements of typed graph grammars. Composition of graph grammars w.r.t common subgrammars is shown to be compatible with the semantics, i.e. the semantics of the composed grammar is obtained as the composition of the semantics of the component grammars. Moreover, the structure of a composed grammar is preserved during a refinement step in the sense that compatible refinements of the components induce a refinement of the composition. The concepts and results are illustrated by an example.

Andrea Corradini
Concurrent computing: from Petri nets to graph grammars
http://www.elsevier.nl/locate/entcs/volume2.html

Petri nets are widely accepted as a specification formalism for concurrent and distributed systems. One of the reasons of their success is the fact that they are equipped with a rich theory, including well-understood concurrent semantics; they also provide an interesting benchmark for tools and techniques for the description of concurrent systems. Graph grammers can be regarded as a proper generalization of Petri

nets, where the current state of a system is described by a graph instead as by a collection of tokens. In this tutorial paper I will review some basic definitions and constructions concerning the concurrent semantics of nets, and I will show to what extent corresponding notions have been developed for graph grammars. Most of such results come out from a joint research by the Berlin and Pisa COMPUGRAPH groups.

Bruno Courcelle
Logic and graphs
http://www.elsevier.nl/locate/entcs/volume2.html

The lecture is based on a chapter of a forthcoming handbook of graph grammars, edited by G. Rozenberg. This chapter is entitled: "The expression of graph properties and graph transformations in monadic second-order logic".

Anke Drappa and Ralf Melchisedech
The use of graph grammar in a software engineering education tool
http://www.elsevier.nl/locate/entcs/volume2.html

In the past 20 years there was significant effort devoted to the development of different graph grammar approaches. Graph grammars have been applied in a variety of fields, both in the computer science area and other sciences, like biology. In this paper we want to present the software engineering application system SESAM, which is based on the very powerful combination of a graph grammar approach and system dynamics. SESAM (Software Engineering Simulation by Animated Models) is a simulation system for software projects. Students as well as practitioners should learn how to lead software projects effectively without spending (or even wasting) time and money in real projects. SESAM consists of two components: the model building component which allows users ("process engineers") to build models of software projects and the simulation component which allows users ("project leaders") to animate those models and to simulate software projects as an interactive game. To reflect the real world in the simulation model, we use objects and relationships between objects which are characterized by attributes. These objects and relationships form a graph structure which is manipulated during the simulation by a set of graph rewriting rules. These graph rewriting rules basically model effects which can be observed during real software projects, like hiring new project members or increased productivity after having successfully introduced a new tool. In contrast to conventional graph rewriting systems, SESAM rules out only reflect the changes concerning the graph structure, but also the changes of attribute values with (simulation) time. To achieve the highest possible flexibility in computing the attribute values, we have extended the notation for the graph rewritting rules by system dynamics elements.

Frank Drewes
Semirings and tree-to-graph-to-tree transductions
http://www.elsevier.nl/locate/entcs/volume2.html

Tree-to-graph-to-tree transductions without sharing (tgt transductions) that compute functions from an arbitrary domain into a semiring are studied. Tgt transductions build the output using hyperedge-replacement graph operations. In spite of the fact that tgt transductions are more powerful than top-down tree transductions, we show that compositions of top-down tree transductions can compute the same functions into semirings as tgt transductions. This has a number of consequences, in particular for the case where the input algebra is a graph algebra over hyperedge-replacement operations.

Hartmut Ehrig
Introduction to COMPUGRAPH
http://www.elsevier.nl/locate/entcs/volume2.html

This is a survey of the main aims and results of the ESPRIT Basic Research Working Group COMPUTING BY GRAPH TRANSFORMATION with main emphasize on COMPUGRAPH II,

1992–1996, following up the first phase of COMPUGRAPH, 1989–1992. Starting with an overview of the history of graph grammars and graph transformations the research goals and main results are presented within the following three research areas: Foundations, Concurrency, and Graph Transformations for Specification and Programming.

Gregor Engels and Andy Schürr
Encapsulated hierarchical graphs, graph types, and meta types
http://www.elsevier.nl/locate/entcs/volume2.html

Currently existing graph grammar-based specification languages have serious problems with supporting any kind of "specification-in-the-large" activities. More precisely, they have deficiences with respect to modeling hierarchical data structures or specifying meta activities like manipulation of graph schemata. Furthermore, already proposed graph grammar module concepts are still too abstract to be useful in practice. Our contribution addresses these problems by introducing a new hierarchical graph data model with an infinite number of schema, meta-schema, etc., layers. It forms the base for a forthcoming concrete modular graph grammar specification language where in addition information hiding aspects like explicit export and import interfaces are expressible.

Annegret Habel and Detlef Plump
Unification, rewriting, and narrowing on term graphs
http://www.elsevier.nl/locate/entcs/volume2.html

The concept of graph substitution recently introduced by the authors is applied to term graphs, yielding a uniform framework for unification, rewriting, and narrowing on term graphs. The notion of substitution allows definitions of these concepts that are close to the corresponding definitions in the term world. The rewriting model obtained in this way is equivalent to "collapsed tree rewriting" and hence is complete for equational deduction. For term graph narrowing, a completeness result is established which corresponds to Hullot's classical result for term narrowing. The general motivation for using term graphs instead of terms is to improve efficiency: sharing common subterms saves space and avoids the repetition of computations.

Reiko Heckel and Annika Wagner
Ensuring consistency of conditional graph rewriting – a constructive approach –
http://www.elsevier.nl/locate/entcs/volume2.html

Algebraic graph transformations of the single pushout type are extended by conditional application conditions and consistency conditions. Both are specified in a categorical (i.e. graphical) way by means of total graph morphisms. Conditional application conditions may express positive and negative context conditions as well as combinations of them consisting of a premise and a disjunction of conclusions (similar to Gentzen formulas). Consistency conditions describe uniqueness and existence of elements in a graph generalizing conditional equations. As the main result a construction is shown that, given a rule r and a consistency condition CC, generates a (conditional) application condition $A_{CC}(r)$ for r such that $A_{CC}(r)$ is satisfied by a match $L \xrightarrow{m} G$ if and only if the derived graph H is consistent, for all consistent graphs G.

Peter Heimann, Gregor Joeris, Carl-Arndt Krapp and Bernhard Westfechtel
A programmed graph rewriting system for software process management
http://www.elsevier.nl/locate/entcs/volume2.html

Managing the software development and maintenance process has been identified as a great challenge for several years. Software processes are highly dynamic and can only rarely be planned completely in advance. We present an approach to software process management which is based on hierarchical nets of processes

connected by data and control flow relations. Editing and execution of process nets are highly intertwined. Dynamic process nets are formally defined in PROGRES, a specification language which is based on programmed graph rewriting systems. Graph rewriting systems are a natural choice for several reasons. In particular, process nets are complicated graph structures, and editing as well as execution operations may be specified in a uniform way by graph rewrite rules. The graph rewriting system will form the foundation of a sophisticated process management system.

Dirk Janssens
Process languages for ESM systems
http://www.elsevier.nl/locate/entcs/volume2.html

In most approaches to graph grammars the formal description of their behaviour is based on a derivation relation between graphs. Graph grammar processes may then be constructed from derivation sequences. However, one may also follow an alternative approach, considering a graph grammar directly as the generator of a language of process objects, as was done in earlier work about ESM graph rewriting. In this paper the relation between the two approaches is clarified, using results about the composition and decomposition of ESM processes. It is shown that processes in the usual sense correspond to a special case of general ESM process objects. The use of these general process objects is illustrated by an application of ESM graph rewriting to the modeling of parallel object-oriented systems.

Thomas Johnsson
Graph reduction, and how to avoid it
http://www.elsevier.nl/locate/entcs/volume2.html

In the first part of this talk, I will review the thought that led to the G-machine. In the second part, I will describe some recent work on formalising the "avoiding graph reduction" bit, by doing fold/unfold transformation on a basic inefficient graph constructing interpreter.

Richard Kennaway
Infinitary rewriting and cyclic graphs
http://www.elsevier.nl/locate/entcs/volume2.html

Infinitary rewriting allows infinitely large terms and infinitely long reduction sequences. There are two computational motivations for studying these: the infinite data structures implicit in lazy functional programming, and the use of rewriting of possibly cyclic graphs as an implementation technique for functional languages. We survey the fundamental properties of infinitary rewriting in orthogonal term rewrite systems, and its relation to cyclic graph rewriting.

Zurab Khasidashvili and Vincent van Oostrom
Context-sensitive conditional expression reduction systems
http://www.elsevier.nl/locate/entcs/volume2.html

We introduce context-sensitive conditional expression reduction systems (CERSs) by generalizing the notion of conditional TRSs to the higher order case. We define orthogonality for CERSs and prove confluence for orthogonal CERSs. The existing confluence results for orthogonal conditional systems are covered by our confluence theorem. Further, it can be used to infer confluence from the subject reduction property in several typed λ-calculi, and to establish confluence for rewrite systems with pattern-matching definitions, since the latter are conditional rewrite systems. To justify further our framework, we show how easily can one express several proof and transition systems as CERSs. In particular, we give encodings of Hilbert-style proof systems, Gentzen-style sequent-calculi, rewrite systems with rule priorities, and the π-calculus into CERSs; the latter is an important example of real context-sensitive systems.

Martin Korff and Leila Ribeiro
Concurrent derivations as single pushout graph grammar processes
http://www.elsevier.nl/locate/entcs/volume2.html

Algebraic graph transformations visually support intuition, have a strong theoretical basis, and provide a formal, implementation independent basis for the description of discretely evolving computational systems and their formal and tractable analysis. Graph grammar models of concurrent systems (Petri nets, actor systems) have inspired corresponding semantics developments. Recently, this led to the introduction of partial orders of concurrent derivations (concurrent computations). A concurrent derivation (CDer) abstracts from the (sequential) order of rule applications in the sequential derivation and thus can be considered as a concurrent process. Complementary, a morphism between two concurrent derivations expresses that the first is a computational approximation of the second. In this paper we newly introduce non-deterministic concurrent derivations (CTrees) as classes of concurrently equivalent sequential derivation trees. Due to the fact that also infinite computations are represented by CTrees, the category of all CTrees of a given graph grammar has a final object (the concurrent counterpart of the whole sequential tree of the given grammar) which is approximated by all other CTrees. We show that (syntactical) morphisms between two graph grammars induce corresponding adjunction between the corresponding (semantic) categories of CDers and CTrees, respectively.

Hans-Jörg Kreowski
Specification and programming (by graph transformation)
http://www.elsevier.nl/locate/entcs/volume2.html

In this tutorial, some basic ideas will be outlined and exemplified how graph transformation can be employed in specification and programming. In particular, some of the respective achievements of the COMPUGRAPH project will be pointed out.

Sabine Kuske
Implementing β-reduction by hypergraph rewriting
http://www.elsevier.nl/locate/entcs/volume2.html

The λ-calculus can be considered as the computational basis for functional programming. Graph reduction for the λ-calculus was first studied by Wadsworth (1971) contributing to a higher performance concerning the implementation of functional languages. One main advantage of the representing λ-terms by graphs is that common subterms can be shared such that several redexes can be reduced in parallel. In the last 25 years, the theory of graph rewriting has been well developed; in particular, Habel/Kreowski/Plump (1991) and Corradini/Rossi (1993) showed hypergraph rewriting to be a suitable formalism for the implementation of term rewriting systems and logic programming. In this paper, it is shown how to implement the β-reduction in the λ-calculus with a hypergraph rewriting mechanism called collapsed λ-tree rewriting, Collapsed λ-tree rewriting consists of collapsed λ-tree reduction on the one had and a splitting mechanism on the other hand. Both kinds of manipulating collapsed λ-trees are based on hypergraph rewriting and can be performed in arbitrary order. Each reduction step is sound in the sense that it corresponds to a sequence of β-reduction steps is sound in the sense that it corresponds to a sequence of β-reduction steps in the represented λ-term. With the additional splitting mechanism completeness with respect to the Gross–Knuth strategy is achieved. As a consequence, there exists a normal form for a collapsed λ-tree if and only if there exists a normal form for the represented λ-term.

Igor Litovsky, Yves Métivier and Eric Sopena
Checking global graph properties by means of local computations: the majority problem
http://www.elsevier.nl/locate/entcs/volume2.html

One of the main characteristics of distributed systems is the local nature of the computation. A set of processors, connected in some specific way, try to reach a common goal (e.g. computing some function) after

a finite number of elementary steps, each involving solely a subset of "near" processors. In this framework, the main question is to characterize those functions, that is those *global* properties of the network, that can be computed by means of local transformations in the network. In this paper we investigate that question by using a computational model based on graph relabelling systems. We are mostly interested in the following paradigm, called the *majority* problem}, which is an abstraction that underlies a variety of distributed computation problems on graphs: let A and B be any two labels, G be a graph whose vertices are labelled on $\{A, B\}$, $|G|_A$ (resp. $|G|_B$) be the number of vertices of G labelled with A (resp. with B). To what extent are we able to compare the quantities labelled with A (resp. with B). To what extent are we able to compare the quantities $|G|_A$ and $|G|_B$? We prove that using such graph recognizers we can decide whether $|G|_A > |G|_B$, $|G|_B > |G|_A$ or $|G|_A = |G|_B$. Then, using the notion of *k-covering*, we prove that it is not possible to decide whether $|G|_A = |G|_B + m$ for any $m > 0$.

M. Monserrat, F. Rosselló, J. Torrens and G. Valiente
Hypergraph rewriting using conformisms
http://www.elsevier.nl/locate/entcs/volume2.html

In this paper we study single-pushout transformation in a category of *spans*, a generalization of the notion of partial morphism in, for instance, Kennaway (1991), and Robinson and Rosolini (1988). As an application, single-pushout transformation in a category of hypergraphs with a special type of partial morphisms, the *conformisms*, is presented. In particular, we show the existence of the pushout of any pair of conformisms of hypergraphs with the same source hypergraph, and how to construct one such a pushout. Finally, hypergraph rewriting using conformisms is compared to single-pushout hypergraph rewriting by means of a detailed example.

M.J. Plasmeijer
CLEAN: a programming environment based on term graph rewritting
http://www.elsevier.nl/locate/entcs/volume2.html

The main features of the lazy functional language Concurrent Clean and of its semantics based on Term Graph Rewriting are presented.

Yves-Marie Quemener and Thierry Jéron
Model-checking of infinite Kripke structures defined by simple graph grammars
http://www.elsevier.nl/locate/entcs/volume2.html

We present an algorithm for checking whether an infinite transition system, defined by a graph grammar of a restricted kind, is a model of a formula of the temporal logic CTL. We first present the syntax and the semantics of CTL, that are defined with respect to transition systems, whose states are labelled with atomic propositions. Then, we show how to adapt the formalism of graph grammars, for expressing such infinite transition systems. We consider simple graph grammars, which are composed of one rule, whose right member includes one hyperarc which is the left member of the rule. This enables to finitely represent an infinite transition system, formed by one initial graph, on which a given pattern can be infinitely added. Our algorithm treats such a finite representation, and modify it, ensuring that the labelling for formulas remains coherent with the truth values of the different states of the infinite transition system.

Georg Schied and Klaus Barthelmann
Linear types for higher order processes with first class directed channels
http://www.elsevier.nl/locate/entcs/volume2.html

Distributed higher-order processes (DHOP) is a small programming language for distributed systems based on the behavioural paradigm of process calculi. It has modern features like static typing, first class

processes, and a dynamically reconfigurable network topology. Unlike most of similar languages which also rely on message passing, DHOP enforces directed one-to-one communication channels between processes. A typing system using the idea of linear types allows statically to check this one-to-one condition. The soundness of the typing system can be shown based on the operational semantics that is defined by means of graph grammars.

Hans Jürgen Schneider
A note on outward and inward productions in the categorical graph-grammar approach and Δ-grammars
http://www.elsevier.nl/locate/entcs/volume2.html

The double-pushout approach to graph grammars uses the pushout construction to generalize the notion of concatenation: The left-hand (right-hand) side of a production and the context are glued together along a special gluing graph. Recently, Banach has presented a new version with arrows going inwards. By proving the correspondence between the usual double-pushout approach and Banach's inward version in a purely categorical setting, we can extend Banach's approach. We show that both approaches have the same generative power as long as one side of the production is an injection and that Banach's approach is more powerful if both sides are not injective. Furthermore, considering the inward approach makes evident a close relationship between the categorical approach and Kaplan's Δ-grammars making the latter an operational description of the first. This relationship allows slightly generalizing Δ-grammars as well as making the categorical approach easily implementable.

Detlef Seese
Linear time computable problems and logical descriptions
http://www.elsevier.nl/locate/entcs/volume2.html

The paper is devoted to the general problem of the tradeoff between the complexity of algorithmic problems, the structure of the input objects and the expressive power of problem description languages. The article concentrates on linear time algorithms and on first-order logic as problem description language. One of the main results is a proof that each first-order problem can be solved in linear time for arbitrary relational structures of universally bounded degree. The basic idea of the proof is a localization technique which is based on a method which was originally developed by Hanf to show that two infinite structures agree on all first-order sentences. Fagin, Stockmeyer and Vardi developed a variant of this technique which is applicable in descriptive complexity theory to finite relational structures of universally bounded degree. The paper is organized as follows: after a short introduction into the area of research, the basic terminology and the notion BIORAM are introduced, which is a special RAM serving as basis of the linear time computability used in this article. Section 3 introduces local r-types and handles the case of structures of bounded degree by reducing the general problem to a local investigation of r-types. Some open problems and remarks conclude the paper in Section 4.

Duncan Shand and Simon Brock
Proofs as graphs
http://www.elsevier.nl/locate/entcs/volume2.html

This note introduces a method of representing and reasoning about the actions of a class of proof procedures. A graph-like structure, called a proof diagram, is introduced in which conclusions of inferences can be shared. A version of Kruskal's Tree Theorem is developed for these structures and from there a notion of minimal proof is introduced. The notion of minimal proof allows us to make a link between standard treatments of proof and proofs generated by mechanical theorem provers. We discuss various uses for the proof diagrams including the development of tactics by graph reduction and propose a graph based meta-language for theorem provers.

Ronan Sleep
SEMAGRAPH: the theory and practice of term graph rewriting
http://www.elsevier.nl/locate/entcs/volume2.html

Semagraph II is a working group (WG 6345) supported by the CEE basic research directorate. The main aim of Semagraph II has been to mature and promulgate the research results and prototypes developed during the Semagraph I Basic Research Action (BRA 3074). This note outlines the aims and achievements of Semagraph, and provides references to key results.

Gabriele Taentzer and Andy Schürr
DIEGO, another step towards a module concept for graph transformation systems
http://www.elsevier.nl/locate/entcs/volume2.html

DIEGO graph transformation systems offer means for DIstributed programming with Encapsulated Graph Objects. Their operational semantics definition follows the lines of distributed graph transformation, thereby permitting the specification of distributed systems with concurrently interacting objects. Adapting previously made proposals for a graph grammar module concept and for graph models with information hiding to this setting, the new approach supports structuring of large specifications into small reusable modules with well-defined interfaces between them. Use relationships between export interfaces and import interfaces are our means to construct system architectures and to allow reuse of modules in different environments.

Chris Wadsworth
Graph reduction: a retrospective
http://www.elsevier.nl/locate/entcs/volume2.html

Graph reduction is 25 years old. This invited presentation will reflect on the development and impact of research in the field starting with the work of Wadsworth in 1970.