264863-2

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Diagonal Compositionality of Concurrent ,Finite State Systems

por
Paulo Blauth Menezes

RP 265 Maio/1996

UFRGS-II-CPGCC
Caixa Postal 15064 - CEP 91501-970
Porto Alegre RS BRASIL
Telefone: (051)316-6155
Fax: (051) 336-5576
Email: pgcc@inf.ufrgs

Submitted to the INFINITY - Verification of Infinite State Systems
Satellite Workshop of CONCUR '96

Diagonal Compositionality of Concurrent, Finite State Systems *

P. Blauth Menezes

Departamento de Informática Teórica, Instituto de Informática, Universidade Federal do Rio Grande do Sul
Caixa Postal 15064, 91501-970, Porto Alegre, Brazil - blauth@inf.ufrgs.br

Extended Abstract

In previous works together with J. Félix Costa and A. Sernadas, we construct a categorial semantic domain called Nonsequential Automata (first introduced in [10]), based on labeled transition systems [15] with full concurrency, inspired by [14], where synchronization and hiding are functorial defined using fibration and cofibration techniques based on [9] and inspired by [24]. Moreover, a class of morphisms stands for reification which satisfies the diagonal compositionality requirement, i.e., reifications compose (vertically), reflecting the stepwise description of systems, involving several levels of abstraction, and distributes through combinators (horizontally), meaning that the reification of a composite system is the composition of the reification of its parts. In [10] and [11] we show (through adjunctions) that nonsequential automata are more concrete then Petri nets extending the approach in [17] , where a formal framework for classification of models for concurrency is set. In fact, nonsequential automaton is the least concrete (or more abstract) level among related models which satisfies the diagonal compositionality requirement.

To experiment with the proposed semantic domain, a semantics for a concurrent, object-based specification language (using the terminology of [23]) is given in [13]. The language named Nautilus is based on the object-oriented language GNOME [18] which is a simplified and revised version of OBLOG [20], [21], [19]. Some features inspired by the semantic domain (and not present on GNOME) such as reification and aggregation are introduced. Also, the state-dependent calling of GNOME is extended for interaction, aggregation and reification in Nautilus. The diagonal compositionality requirement is essential to give semantics for Nautilus.

Since the techniques proposed for nonsequential automata such as the reification and synchronization could provide some interesting and elegant solutions for finite state systems, we derive a new categorial framework named Nonsequential Finite Automata which is outlined in this paper.

A nonsequential finite automaton is a kind of automaton with structured states and transitions. Structured states, inspired by [14] are "bags" of local states like tokens in Petri nets (as in [16]) and structured transitions are finite and specify a concurrency relationship between component transitions in the sense of [1] and [7]. Comparing the graphical representations of nonsequential finite automata and asynchronous transition systems, the independence relation in a nonsequential finite automaton is explicit. The labeling of transitions is not extensional in the sense that it allows the occurrence of more than one transition with the same label. Also, nonsequential finite automata are equipped with initial and final states, inspired by [8]. The resulting category is complete and cocomplete with products isomorphic to coproducts. The categorial product (or coproduct) stands for parallel composition.

The behavior of a nonsequential finite automaton is given by its transitive closure, i.e., the automaton is enriched with all conceivable sequential and nonsequential computations that can be split into permutations of original transitions, respecting source and target states, inspired by [14]. This transitive closure is induced by an adjunction between a category of categories and the

* This work was partially supported by CNPq - Conselho Nacional de Desenvolvimento Científico e Tecnológico in Brazil.

category of nonsequential finite automata, as illustrated in the Figure 1, where $tc = a \circ c$. The functor c has a requirement about concurrency which is $(t;u) \| (t';u') = (t \| t');(u \| u')$. That is, the computation determined by two independent composed transitions $t;u$ and $t';u'$ is equivalent to the computation whose steps are the independent transitions $t \| t'$ and $u \| u'$. The (regular) language accepted by a nonsequential finite automaton is induced by its transitive closure considering the label of each transition that has an initial and final state as source and target, respectively.

A reification morphism $\varphi: N_1 \rightarrow tcN_2$ maps transitions into transactions reflecting an implementation of an automaton on top of another, where the target object (more concrete) is enriched with its transitive closure. In fact, since a reification should to not preserve labeling, the mapping is restricted to the "shape" of component automata (and therefore, it is not an automaton morphism). However, we show that each reification induces a nonsequential finite automaton morphism and therefore, we may define a category whose morphisms are nonsequential finite automaton morphisms induced by reifications. In any case, the composition of morphisms is as in Kleisli categories (the right categorial setting to describe the composition of morphisms where the target object is enriched through an endofunctor). Both categories are isomorphic.

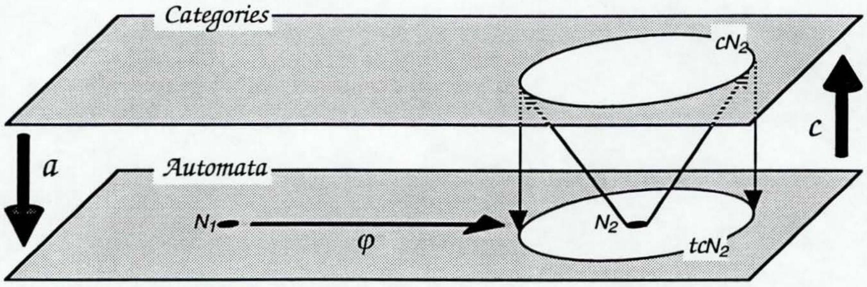

Figure 1. Transitive closure and transitions mapped into transactions

Therefore, reifications compose, i.e., the vertical compositionality requirement is achieved. Moreover we find a general theory of reification of finite state systems which also satisfies the horizontal compositionality requirement. i.e., for reifications $\varphi: N_1 \rightarrow tcM_1$, $\psi: N_2 \rightarrow tc\, M_2$, we have that:

$$\varphi N_1 \times \psi N_2 = \varphi \times \psi(N_1 \times N_2)$$

where $\varphi N_1 \times \psi N_2$ and $N_1 \times N_2$ are parallel compositions and the reification $\varphi \times \psi$ is (uniquely) induced by φ and ψ.

Synchronization and hiding are functorial operations defined using fibration and cofibration techniques based on [9], [12] and inspired by [24]. Both functors are induced by morphisms at the label level. A synchronization restricts a parallel composition according to some table of synchronizations (at label level), as illustrated in the Figure 2, where $\times N_i$ and $\|_{sync} N_i$ denote the parallel composition and the synchronized automaton, respectively.

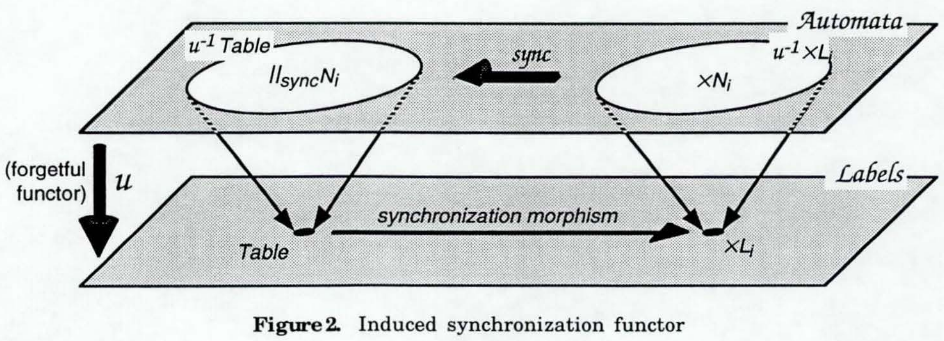

Figure 2. Induced synchronization functor

Also, we show a categorial way to construct tables of synchronizations for calling and sharing and the corresponding synchronization morphism. The table of synchronizations is given by a colimit whose resulting diagram has a shape illustrated in the Figure 3 where the central arrow has as source an object named channel and as target the table of synchronizations.

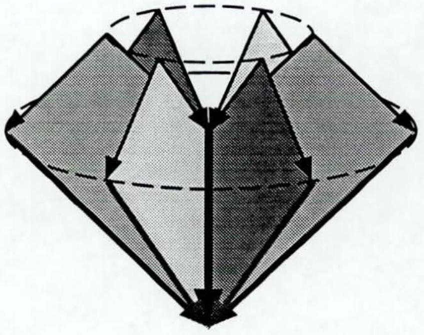

Figure 3. Table of synchronizations given by a colimit

The synchronization of reified automata is the synchronization of the source automata whose reification is induced by component reifications (in this construction, the horizontal compositionality is an essential property). An important result is that the horizontal compositionality requirement is extended for synchronization.

A view of an automaton is obtained through hiding of transitions introducing an internal nondeterminism as ε-moves in the sense of [5]. A hidden transition cannot be used for synchronization. The hiding of a reification is induced by the hiding of the source automaton.

The clarification of the relationship between nonsequential finite automata and (sequential) finite automata is done through translation functors, inspired by [17]. From the steps of abstractions that are involved, we can infer (as expected) that nonsequential finite automata are more concrete than finite automata (in the sense of [17]).

At this moment we are extending the framework for nonsequential finite automata with output, developing tools and working on some specific problems related to language recognition. Also, following the same approach, we are starting some works toward nonsequential pushdown automata.

References

[1] M. A. Bednarczyk, *Categories of Asynchronous Systems*, Ph.D. thesis, technical report 1/88, University of Sussex, 1988.

[2] J. F. Costa, A. Sernadas, C. Sernadas & H. D. Ehrich, *Object Interaction*, Mathematical Foundations of Computer Science '92 (I. Havel, V. Koubek, Eds.), pp. 200-208, LNCS 629, Springer-Verlag, 1992.

[3] H. D. Ehrich & A. Sernadas, *Algebraic Implementation of Objects over Objects*, Stepwise Reification of Distributed Systems: Models, Formalisms, Correctness (J. W. de Bakker, W. -P. de Roever, G. Rozenberg, Eds.), pp. 239-266, Springer-Verlag, 1990.

[4] R. Gorrieri, *Reification, Atomicity and Transactions for Process Description Language*, Ph.D. thesis, Università di Pisa, 1990.

[5] J. E. Hopcroft & J. D. Ullman, *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley, 1979.

[6] S. Mac Lane, *Categories for the Working Mathematician*, Springer-Verlag, 1971.

[7] A. Mazurkiewicz, *Basic Notion of Trace Theory*, REX 88: Linear Time, Branching Time and Partial Orders in Logic and Models for Concurrency (J. W. de Bakker, W. -P. de Roever, G. Rozenberg, Eds.), pp. 285-363, LNCS 354, Springer-Verlag, 1988.

[8] P. B. Menezes, *Marked Petri Nets Within a Categorial Framework*, RITA: Revista de Informática Teórica e Aplicada, No. 2, Vol. 2, pp. 71-92, UFRGS, Brazil, 1995.

[9] P. B. Menezes & J. F. Costa, *Synchronization in Petri Nets*, to appear in Fundamenta Informaticae Vol. 26.1, Annales Societatis Mathematicae Polonae, IOS Press.

[10] P. B. Menezes & J. F. Costa, *Compositional Reification of Concurrent Systems*, Journal of the Brazilian Computer Society - Special Issue on Parallel Computation, No. 1, Vol. 2, pp. 50-67, 1995.

[11] P. B. Menezes & J. F. Costa, *Systems for System Implementation*, in Proceedings of the 14th International Congress on Cybernetics (1995), accepted for publication in the Journal of Cybernetics.

[12] P. B. Menezes, J. F. Costa & A. Sernadas, *Reification Mapping for (Discrete Event) System Theory*, Proceedings of the Fifth International Conference on Computer Aided System Technology, EUROCAST 95, pp. 103-116, LNCS 1030, Springer-Verlag, 1996.

[13] P. B. Menezes, A. Sernadas & J. F. Costa, *Nonsequential Automata Semantics for a Concurrent Object-Based Language*, preprint IST/DM/21/95, Universidade Técnica de Lisboa, Instituto Superior Técnico, Lisbon, 1995. Submitted.

[14] J. Meseguer & U. Montanari, *Petri Nets are Monoids*, Information and Computation 88, pp. 105-155, Academic Press, 1990.

[15] R. Milner, *Communication and Concurrency*, Prentice Hall, 1989.

[16] W. Reisig, *Petri Nets: An Introduction*, EATCS Monographs on Theoretical Computer Science 4, Springer-Verlag, 1985.

[17] V. Sassone, M. Nielsen & G. Winskel, *A Classification of Models for Concurrency*, CONCUR 93: 4th International Conference of Concurrency (E. Best, Ed.), pp. 82-96, LNCS 715, Springer-Verlag, 1993.

[18] A. Sernadas & J. Ramos, *A Linguagem GNOME: Sintaxe, Semântica e Cálculo*, technical report, Universidade Técnica de Lisboa, Instituto Superior Técnico, Lisbon, 1994.

[19] C. Sernadas, P. Resende, P. Gouveia & A. Sernadas, *In-the-Large Object-Oriented Design of Information Systems*, The Object-Oriented Approach in Information Systems (F. van Assche, B. Moulin, C. Rolland, Eds.), pp. 209-232, North-Holland, 1991.

[20] C. Sernadas, P. Gouveia & A. Sernadas, *OBLOG: Object-Oriented, Logic-Based Conceptual Modeling*, technical report, Universidade Técnica de Lisboa, Instituto Superior Técnico, Lisbon, 1992.

[21] C. Sernadas, P. Gouveia, J. Gouveia & P. Resende, *The Reification Dimension in Object-Oriented Database Design*, Specification of Data Base Systems (D. Harper, M. Norrie, Eds.), pp. 275-299, Springer-Verlag, 1992.

[22] M. E. Szabo, *Algebra of Proofs*, Studies in Logic and the Foundations of Mathematics, vol. 88, North-Holland, 1978.

[23] P. Wegner, Concepts and Paradigms of Object-Oriented Programming, OOPS Messenger, Vol. 1, No. 1, ACM Press, 1990.

[24] G. Winskel, *Petri Nets, Algebras, Morphisms and Compositionality*, Information and Computation 72, pp. 197-238, Academic Press, 1987.