Computation of Partial Automata Through Span Composition*

Karina Girardi Roggia, Marnes Augusto Hoff, and Paulo Blauth Menezes

Instituto de Informática - UFRGS,
Av. Bento Gonçalves, 9500, Campus do Vale, Bloco IV,
Porto Alegre, RS, Brazil – CEP 91501-970
{kaqui, marnes, blauth}@inf.ufrgs.br

Abstract. In this paper a way to have structures with partiality in its internal structure in a categorical approach is presented and, with this, a category of partial graphs $\mathcal{G}r_p$ is given and partial automata are constructed from $\mathcal{G}r_p$. With a simple categorical operation, computations of partial automata are given and can be seen as a part of the structure of partial automata.

Keyworks: computation, partial automata, Category Theory.

1 Introduction

In Computer Science, to express non-terminanting computations and to define partial recursive functions, it is common to use the notion of partiality. Actually, due to partiality, p.g., the class of partial recursive functions becomes equivalent to Turing Machines. Category Theory arrises as an useful tool to formalize abstract concepts making easy to construct proofs and investigate properties in many areas, specially in Semantics and Type Theory. The constructions about universal mappings like limits and adjunctions are getting useful interpretations in terms of compositionality of systems. Besides, in Category Theory there are tools to define structures more complex based in simple ones like Comma Categories, that allows to define a category based in another and to inherit properties. Categories of Graphs and Automata are usually defined by this structure.

Graphs are commonly used to model systems, either by simple graphs or by graph-based structures like Petri nets [1,2,3] and automata [4,5]. Automata is a common formalism used in many areas in Computer Science like compilers, microelectronics, etc. Most of the study about it is in the Formal Language area.

In this paper we define a different category of automata: a category of Partial Automata, named $\mathcal{A}ut_p$. This category is constructed over a category of partial graphs ($\mathcal{G}r_p$). The difference between a graph and a partial graph is in the definitions of the source and target functions that mapped an arc to a node: in graphs these functions are total while in partial graphs source and target functions are partial functions. Due to this difference, automata based in partial

* This work is partially supported by CAPES, CNPq and FAPERGS.

© Springer-Verlag Berlin Heidelberg 2005

graphs can have marks of initial and final states naturally. Beyond that, we can also define constructions like limits in $\mathcal{A}ut_p$ that allows composition of partial automata.

In [6], span Composition [7] is used to compose graphs given semantics of systems with dynamic topology, p.g.. This kind of composition could be used to define the computations of (partial) automata. Briefly, a span composition of two partial automata results in a partial automata where each edge represents a path of length up to two (between nodes), which first half is some edge of the first automaton and which second half is some edge of the second one. It is possible to compose the same automaton with itself several times which is the purpose of this paper. In the case of n successive span compositions, we can obtain all the words of its accepted language whose needs $n+1$ steps of computation in the arcs of the partial automaton that don't have source neither target.

2 Partial Graphs

To define partial automata, we'll first construct a way to define structures with partiality in its internal structure: this is done with the definition of *pComma* – a special kind of comma-category [8]. Then a category of partial graphs is defined. A partial graph is a directed graph where the functions source and target of arcs are **partial** functions. The definition of *pComma*, uses the notion of category of partial morphisms – named $p\mathcal{C}$ – defined in [9].

Definition 1 (*pComma*)**.** *Consider the finitely complete category $\mathcal{C}$ and the functors $\mathbf{inc_p} : \mathcal{C} \to p\mathcal{C}$ (the canonical inclusion functor), $\mathbf{f} : \mathcal{F} \to \mathcal{C}$ and $\mathbf{g} : \mathcal{G} \to \mathcal{C}$. Therefore, pComma$(\mathbf{f}, \mathbf{g})$ is such that the objects are triples $S = \langle F, s, G\rangle$, where F is a $\mathcal{F}$-object, G is a $\mathcal{G}$-object and $s : \mathbf{inc_p} \circ \mathbf{f}F \to \mathbf{inc_p} \circ \mathbf{g}G$ is a $p\mathcal{C}$-morphism; a morphism $h : S_1 \to S_2$ where $S_1 = \langle F_1, s_1, G_1\rangle$, $S_2 = \langle F_2, s_2, G_2\rangle$ is a pair $h = \langle h_F : F_1 \to F_2, h_G : G_1 \to G_2\rangle$ where h_F and h_G are morphisms in $\mathcal{F}$ and $\mathcal{G}$ respectively, and are such that in $p\mathcal{C}$ (see figure 1) $(\mathbf{inc_p} \circ \mathbf{g}h_G) \circ s_1 = s_2 \circ (\mathbf{inc_p} \circ \mathbf{f}h_F)$; the identity morphism of an object $S = \langle F, s, G\rangle$ is $\iota_S = \langle \iota_F : F \to F, \iota_G : G \to G\rangle$; and the composition of $u = \langle u_F, u_G\rangle : S_1 \to S_2$, $v = \langle v_F, v_G\rangle : S_2 \to S_3$ is $v \circ u = \langle v_F \circ u_F, v_G \circ u_G\rangle : S_1 \to S_3$.*

$$\begin{tikzcd}
\mathbf{inc_p} \circ \mathbf{f}F_1 \arrow[r, "s_1"] \arrow[d, "\mathbf{inc_p}\circ\mathbf{f}h_F"'] & \mathbf{inc_p} \circ \mathbf{g}G_1 \arrow[d, "\mathbf{inc_p}\circ\mathbf{g}h_G"] \\
\mathbf{inc_p} \circ \mathbf{f}F_2 \arrow[r, "s_2"] & \mathbf{inc_p} \circ \mathbf{g}G_2
\end{tikzcd}$$

Fig. 1. Diagram of Partial Comma Category

Definition 2 (Category of Partial Graphs). *The category of partial graphs with total homomorphisms, named $\mathcal{G}r_p$, is the partial comma category pComma$(\mathbf{\Delta}, \mathbf{\Delta})$ (beeing $\mathbf{\Delta} : \mathcal{S}et \to \mathcal{S}et^2$ the diagonal functor).*

Thus, a partial graph is $\langle V, T, \partial_0, \partial_1\rangle$ respectively set of nodes, set of arcs and source and target *partial* functions. Seeing a graph as a system, arcs with target function defined only can be seen as entry-points, arcs with source function defided only as end-points. And arcs with neither defined can be seen as transactions. We can divide the set of arcs of a given partial graph respecting the type of the arc.

Definition 3 (Division of T). *Let $G = \langle V, T, \partial_0, \partial_1\rangle$ a partial graph, $\varnothing : T \to \{*\}$ the empty partial function, $tot_T : T \to \{*\}, tot_V : V \to \{*\}$ both total functions and $\partial_0^* = tot_V \circ \partial_0$, $\partial_1^* = tot_V \circ \partial_1$. The following subobjects are given by the equalizers in $p\mathcal{S}et$ like in figure 2:$\langle K_0, \neg\partial_0\rangle$ equalizer of ∂_0^* and $\varnothing$ – arcs of G with source undefined; $\langle K_1, \neg\partial_1\rangle$ equalizer of ∂_1^* and $\varnothing$ – arcs of G with target undefined; $\langle E_0, `\partial_0'\rangle$ equalizer of $\partial_0^*$ and tot – arcs of $G$ with source defined; and $\langle E_1, `\partial_1'\rangle$ equalizer of ∂_1^* and tot – arcs of G with target defined. The pullbacks of figure 3 give the division of T in four classes, where:*

$$\begin{tikzcd}
K_0 \arrow[r, tail, "\neg\partial_0"] & T \arrow[r, shift left, "\partial_0^*"] \arrow[r, shift right, "\varnothing"'] & \{*\} & K_1 \arrow[r, tail, "\neg\partial_1"] & T \arrow[r, shift left, "\partial_1^*"] \arrow[r, shift right, "\varnothing"'] & \{*\} \\
E_0 \arrow[r, tail, "`\partial_0'"] & T \arrow[r, shift left, "\partial_0^*"] \arrow[r, shift right, "tot_T"'] & \{*\} & E_1 \arrow[r, tail, "`\partial_1'"] & T \arrow[r, shift left, "\partial_1^*"] \arrow[r, shift right, "tot_T"'] & \{*\}
\end{tikzcd}$$

Fig. 2. Equalizers in $p\mathcal{S}et$

$\langle VV, vv\rangle$, being $vv = `\partial_0' \circ vv_0 = `\partial_1' \circ vv_1$, arcs with ∂_0 and ∂_1 defined; $\langle VF, vf\rangle$, being $vf = `\partial_0' \circ vf_0 = \neg\partial_1 \circ vf_1$, arcs with $\partial_0$ defined only; $\langle FV, fv\rangle$, being $fv = \neg\partial_0 \circ fv_0 = `\partial_1' \circ fv_1$, arcs with ∂_1 defined only; and $\langle FF, ff\rangle$, being $ff = \neg\partial_0 \circ ff_0 = \neg\partial_1 \circ ff_1$, arcs with ∂_0 and ∂_1 undefined.

$$\begin{tikzcd}
& T & \\
K_0 \arrow[ur, tail, "\neg\partial_0"] & pb & K_1 \arrow[ul, tail, "\neg\partial_1"'] \\
& FF \arrow[ul, tail, "ff_0"] \arrow[ur, tail, "ff_1"'] &
\end{tikzcd}
\qquad
\begin{tikzcd}
& T & \\
K_0 \arrow[ur, tail, "\neg\partial_0"] & pb & E_1 \arrow[ul, tail, "`\partial_1'"'] \\
& FV \arrow[ul, tail, "fv_0"] \arrow[ur, tail, "fv_1"'] &
\end{tikzcd}$$

$$\begin{tikzcd}
& T & \\
E_0 \arrow[ur, tail, "`\partial_0'"] & pb & K_1 \arrow[ul, tail, "\neg\partial_1"'] \\
& VF \arrow[ul, tail, "vf_0"] \arrow[ur, tail, "vf_1"'] &
\end{tikzcd}
\qquad
\begin{tikzcd}
& T & \\
E_0 \arrow[ur, tail, "`\partial_0'"] & pb & E_1 \arrow[ul, tail, "`\partial_1'"'] \\
& VV \arrow[ul, tail, "vv_0"] \arrow[ur, tail, "vv_1"'] &
\end{tikzcd}$$

Fig. 3. Division of Arcs

3 Partial Automata

The term "partial automaton" had been used before to define an algebraic structure based in the definition of automaton. One of the most frequent reference of this term is given by [10]. This kind of partial automata accepts a different type of language in comparison with the languages in [4] that are one of the subjects of this work. Despite, the term *partial automata* in this paper is different from Rutten. Here, a partial automaton is an automaton (possible non-deterministic) where transitions can occur without the assumption of any state before and/or after them. In other words: is an automaton defined from a partial graph.

To define a partial automata category of partial automata we first define two functors: $\mathbf{arcs_p}$ and $\mathbf{coprod_4}$. The forgetful functor $\mathbf{arcs_p}$ takes a partial graph to its set of arcs T in $\mathcal{S}et$ with a function $m : T \to \Omega^2$ that classifies each arc of T in its type by the definition 3, where $\Omega^2 = \{vv, vf, fv, ff\}$.

Definition 4 (Functor $\mathbf{arcs_p}$). *The functor* $\mathbf{arcs_p} : \mathcal{G}r_p \to \mathcal{S}et/\Omega^2$ *is such that, taking* $\langle V, T, \partial_0, \partial_1\rangle$ *any partial graph,* $\mathbf{arcs_p}(\langle V, T, \partial_0, \partial_1\rangle) = \langle T, m\rangle$ *where* $m : T \to \Omega^2$ *is such that (given* $t \in T$*)* $m(t) = \langle v, v\rangle$ *if* $t \in VV, m(t) = \langle v, f\rangle$ *if* $t \in VF, m(t) = \langle f, v\rangle$ *if* $t \in FV$ *or* $m(t) = \langle f, f\rangle$ *if* $t \in FF$*; and given* $h = \langle h_V, h_T\rangle : \langle V_1, T_1, \partial_0^1, \partial_1^1\rangle \to \langle V_2, T_2, \partial_0^2, \partial_1^2\rangle$ *a total homomorphism of partial graphs,* $\mathbf{arcs_p}(h) = h_T$.

The functor $\mathbf{coprod_4}$ does the 4-ary disjoint union of a set and associates a function where each element of a given set A goes to each element of Ω^2.

Definition 5 (Functor $\mathbf{coprod_4}$). *The functor* $\mathbf{coprod_4} : \mathcal{S}et \to \mathcal{S}et/\Omega$ *is such that, given any set* A*,* $\mathbf{coprod_4}(A) = \langle \amalg_A^4, \alpha\rangle$ *where* $\alpha : \amalg_A^4 \to \Omega^2$ *is such that (suppose* $a_i \in \amalg_A^4$ *where* $i \in \{1, 2, 3, 4\}$ *indicate the source of the element in the coproduct – first, second, third or fourth immersion)* $\alpha(a_i) = \langle v, v\rangle$ *if* $i = 1, \alpha(a_i) = \langle v, f\rangle$ *if* $i = 2, \alpha(a_i) = \langle f, v\rangle$ *if* $i = 3$ *or* $\alpha(a_i) = \langle f, f\rangle$ *if* $i = 4$*; and taking* $f : A \to B$ *a function,* $\mathbf{coprod_4}(f) = f^* : \langle \amalg_A^4, \alpha\rangle \to \langle \amalg_B^4, \beta\rangle$ *where (for* $p, q \in \{v, f\}$*)* $f^*(\langle a, \langle p, q\rangle\rangle) = \langle f(a), \langle p, q\rangle\rangle$.

Definition 6 (Category $\mathcal{A}ut_p$). *The category of Partial Automata, called* $\mathcal{A}ut_p$*, is the comma-category* $\mathbf{arcs_p} \downarrow \mathbf{coprod_4}$.

4 Computation of Partial Automata

We use an extension of span composition to define computations of partial automata in the sense used in [6], where span composition was used to compose graphs as dynamic systems. To have definitions and proprieties of span and span composition see [7,11,6].

Definition 7 (Partial Automata Composition of Transitions). *Given the partial automata* $A_1 = \langle V, T_1, \partial_0^1, \partial_1^1, \Sigma_1, etiq_1\rangle$ *and* $A_2 = \langle V, T_2, \partial_0^2, \partial_1^2, \Sigma_2, etiq_2\rangle$*. The* Binary Composition of Transitions*, or just Composition of Transitions, of* A_1 *and* A_2 *is the partial automaton* $A_1 \rhd A_2 = \langle V, T, \partial_0, \partial_1, \Sigma, etiq\rangle$ *where* T *is the object from the pullback,* $\Sigma = \Sigma_1 \times \Sigma_2$ *and* $etiq$ *is the function induced by the product in* $p\mathcal{S}et$ *illustrated in figure 4 and* $\partial_0 = \partial_0^1 \circ p_0$ *and* $\partial_1 = \partial_1^2 \circ p_1$.

$$\begin{tikzcd}
V & & V & & V \\
& T_1 \arrow[ul, "\partial_0^1"] \arrow[ur, "\partial_1^1"'] \arrow[dd, "etiq_1"'] & & T_2 \arrow[ul, "\partial_0^2"] \arrow[ur, "\partial_1^2"'] \arrow[dd, "etiq_2"] & \\
& & T \arrow[ul, "p_0"] \arrow[ur, "p_1"'] \arrow[dd, dashed, "etiq"'] & & \\
& \Sigma_1 & & \Sigma_2 & \\
& & \Sigma_1 \times \Sigma_2 \arrow[ul, "\pi_{\Sigma_1}"] \arrow[ur, "\pi_{\Sigma_2}"'] & &
\end{tikzcd}$$

Fig. 4. Partial Automata Composition of Transitions

Example 1 (Partial Automata and Composition of Transitions). Let the partial automaton $A = \langle \{A, B\}, \{t, u, v,\ w, x, y, z\}, \partial_0^A, \partial_1^A, \{0, 1\}, etiq_B \rangle$ from figure 5 (left) where the functions ∂_0^A, ∂_1^A and $etiq_A$ are represented in there. The resulted Composition of Transitions $A \rhd A$ is in figure 5 (right).

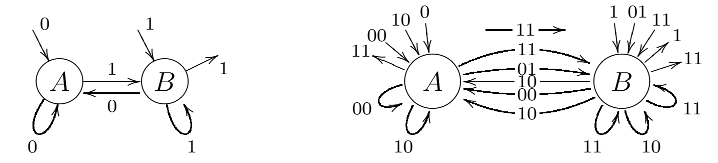

Fig. 5. Partial Automaton A (left) and its Composition of Transitions (right)

Definition 8 (Finite Computation of Partial Automata). *Given a partial automaton $A = \langle V, T, \partial_0, \partial_1, \Sigma, etiq \rangle$, the finite computations of length up to $n{+}1$ of A is the class of arcs FF (as in the definition 3) of the resulting automaton from the Transitions Composition with itself n times.*

Let a partial automaton that computes the language L (the automaton from example 1 computes the language $L = \{w | w \in \{0, 1\}^* \wedge w$ finishes in $11\}$). Composing itself *ad infinitum* by composition od transistions, the resulting arcs without source neither target nodes can be seen as the transitive closure L^+. If we compose itself n times, this kind of arcs will be the subset of L^+ whose word's length is limited to $n + 1$, i.e., the computations of the automaton with $n + 1$ steps.

5 Concluding Remarks

In this paper we presented a way to define computations of a different type of automata: the partial automata. One of the advantages of this kind of automaton is that initial and final actions are natural in the structure, that is constructed in a categorical approach. Generally, to construct structures like graphs and automata in a categorical way, initial and/or final states are not natural. Besides, we can computes the language of a partial automaton with a simple operation of composition seen in each resulting automaton the steps of the computation.

From this work is possible to research and to develop extensions to more complex structures like, for instance, Petri Nets; to explore partial automata that evolve (by a graph-grammar like approach) using the composition of transitions and to study proprieties of formal languages using this approach.

References

1. Peterson, J.L.: Petri Net Theory and the Modelling of Systems. Prentice-Hall, Englewoods Cliffs, New Jersey (1981)
2. Meseguer, J., Montanari, U.: Petri nets are monoids. Information and Computation **88** (1990) 105–155
3. Menezes, P.B.: Diagonal compositionality of partial petri nets. In: 2nd US-Brazil Joint Workshops on the Formal Foundations of Software Systems, Eletronic Notes in Theoretical Computer Science v.14 (1998)
4. Hopcroft, J.E.: Introduction to automata theory, languages and computation. Addison-Wesley (1979)
5. Adamek, J., Trnkova, V.: Automata and Algebras in Categories. 1 edn. Kluwer, Dordrecht (1990)
6. Hoff, M.A., Roggia, K.G., Menezes, P.B.: Composition of Transformations: A Framework for Systems with Dynamic Topology. International Journal of Computing Anticipatory Systems **14** (2004) 259–270
7. Bénabou, J.: Introduction to bicategories. In: Reports of the Midwest Category Seminar. Number 47 in Springer Lecture Notes in Mathematics. Springer-Verlag (1967) 1–77
8. Borceux, F.: Handbook of Categorical Algebra 1: Basic Category Theory. Volume 50 of Encyclopedia of Mathematics and Its Applications. Cambridge University Press, Cambridge (1994)
9. Asperti, A., Longo, G.: Categories, Types, and Structures - An Introduction to Category Theory for the Working Computer Scientist. MIT Press, Cambridge, USA (1991)
10. Rutten, J.J.M.M.: Automata and coinduction (an exercise in coalgebra). In Sangiorgi, D., de Simone, R., eds.: International Conference on Concurrency Theory, CONCUR, n.9. Volume 1466 of Lecture Notes in Computer Science., Berlin, Springer-Verlag (1998) 194–218
11. Bruni, R., Gadducci, F.: Some algebraic laws for spans. In W. Kahl, D.P., Schmidt, G., eds.: Proceedings of RelMiS 2001, Workshop on Relational Methods in Software. Electronic Notes in Theoretical Computer Science, n.44 v.3, Elsevier Science (2001)