Adaptive Web Courses: a Categorial Framework *

P. Blauth Menezes & Júlio P. Machado

II/UFRGS - CP 15064, 91501-970, Porto Alegre/RS, Brazil
Fax: +55 51 319-1576, e-mail: {blauth, juliopm}@inf.ufrgs.br

Abstract
Adaptive hypermedia are suitable to offer flexible Web-based courses and can be useful to solve problems associated with the use of educational hypermedia as the inability to satisfy heterogeneous needs, match a web course to students and build different courses from a set of hypermedia materials. In this context, a framework based on Automata Theory and Category Theory named Hyper-Automaton is defined in order to create a semi-automated system for developing Web Courses. Courses are (nondeterministic finite) automata with output and links between pages are automata transitions (not HTML source) and thus, reusing the instructional material is straightforward. The categorial constructions of products, coproducts, limits, colimits, restriction, relabeling and reification morphisms have meaningful interpretation as constructors to define (possible complex) hypertext documents for web courses. Then we show how adaptive web courses (i.e., courses that adapt itself according to the behavior of the student) can be defined in this framework using categorial operations for adaptive presentation (or content-level adaptation) and adaptive navigation (or link-level adaptation).

Keywords: web courses, adaptive courses, composition, automata theory, category theory.

1 Introduction

Web-based application systems, as well as other complex hypermedia systems with a large variety of users, suffer from an inability to satisfy heterogeneous needs. A remedy for the negative effects of the traditional "one-size-fits-all" approach is to enhance a system's ability to adapt its own behavior to the goals, tasks, interests, and other features of individual users. In particular, adaptive hypermedia may be applied to computer-based learning environments and are suitable to offer flexible Web-based courses.

With the movement of hypermedia systems towards the web, some authors realized the new environment didn't have the necessary features for integrated network based learning environments. As pointed out by (Maurer, 1997) a more powerful WWW system is needed to be able to structure the material into modules (Halasz & Schwartz, 1994) that can be reused and updated without the need to change any links to and from them: "It is of extreme importance that we understand that in WWW we have to replace links by structure. ... embedded links used for structuring purposes must not be used in sophisticated multimedia networked educational environments...".

This paper discusses how to augment the WWW with a hypermedia service (named Hyper-Automaton (Machado *et al*, 2000)) based on the application of concepts inherent in Computing Science, specially Automata Theory (Hopcroft & Ullman, 1979), Category Theory (Mac Lane, 1971) and Hyperdocuments (Nielsen, 1990), that provides a basic framework for building Adaptive Web Courses (Brusilovsky, 1998).

With categories of automata we can use tools from Category Theory in order to formally define and precisely understand some operations over hyperbase components (Menezes & Machado, 1999, 2000). A category of automata is used in three levels during the construction of a web course environment: finite automata with output structures the

hyperdocuments, categorial operations (limits, colimits, restriction, relabeling) are automata composition and adaptation schemes in order to build new hyperbases, and reification introduces a notion of transaction.

The categorial operations (mainly relabeling and restriction) in the Hyper-Automaton framework are able to provide Adaptive Presentation and Adaptive Navigation (Brusilovsky, 1999). The goal of adaptive presentation is to adapt the contents of a hypermedia page, and adaptive navigation is aimed at supporting the user in hyperspace orientation and navigation by manipulating the link structure.

To experiment the proposed model, the Hyper-Automaton System is aimed at supporting Web-based courses at the Computing Institute of the Federal University of Rio Grande do Sul (http://teia.inf.ufrgs.br). The presentation of the instructional Web pages is accomplished through the composition of several documents, chunks of HTML texts, images, Java simulators and other hypermedia resources.

With respect to previous works as in (Menezes & Machado, 2000), in this paper the categorial framework is analyzed in terms of adaptive web courses while the results are still applicable to hypertext systems as a whole. The framework of duo-internal graphs (Menezes & Machado, 1999; Menezes, 2000) is replaced by a simpler one know as internal graphs as in (Corradini, 1990) and (Asperti & Longo, 1991). Also we review and complement some results about the inheritance of limits and colimits properties of internal graphs presented in (Menezes & Machado, 2000) with the purpose of keeping this paper self-contained.

Initially, the concept of hyperdocuments structured as automata is presented and we discuss the use of anticipation in adaptive hypermedia systems. Next, we introduce an example depicting the use of categorial operations as course composition and adaptation schema in the Hyper-Automaton framework. Then, we define a category of internal graphs and present the results about limits and colimits. Also we present, in detail, a category of automata from which we define the restriction, relabeling and reification operations used in the hypertext framework. With respect to the system, a representative part is implemented and tested (see (Machado *et al*, 2000)).

2 Hypertext are Automata

The formal model of hypertext we propose is based on a Finite Automaton with Output (Hopcroft & Ullman, 1979; Menezes, 2000b) representation of a hyperbase structure. We take advantage of the fact that automata not only capture the descriptive power of Directed Graphs, known to be a useful abstraction in hypertext systems (Conklin, 1987), but provide as well a mathematically precise abstract machine for control and analysis of hypertext execution or browsing and is also an universally known formalism.

For completeness of our discussion we first provide a short set of definitions for the Hyper-Automaton. Following this, we give the definitions for the categorial framework of the hypertext model. The notation style is that commonly used in Automata Theory.

The visual interface of the browser environment provides the user with a tangible interpretation of the Mealy/Moore Machines. The output alphabet Δ is annotated with units of information (hypermedia HTML pages) and, in that case, the result of the next-

output function δ_S: $Q \rightarrow \Delta^*$ (Moore) or the next-state function δ: $Q \times \Sigma \rightarrow Q \times \Delta^*$ (Mealy) is the display of the HTML documents concatenated in one browser window. The input alphabet Σ that labels the transitions between states Q in the automaton are displayed as HTML links that a user can select. The link itself is the projection of the next-state function δ in the hypertext environment. If a link is followed, then the current displayed contents are deactivated and the contents mapped to the output nodes (Moore) or transitions (Mealy) are activated, in accordance to the transition executed.

Although the essence of the model is its machine-supported linking, the way the output-alphabet (hypertext contents) is constructed has an important role in the system. The sizing of the HTML files are small and are the result of the modularization of a document into information fragments within a syntactic unit, such as a definition, an example, an explanation paragraph, etc. Thus, keeping the notion of output-word, a hypertext node is composed by several other small fragments (some kind of composite node).

With the use of Finite Automaton the links are implemented as transition functions and are stored in a matrix representing the source state and destination state, and they are not embedded in the HTML code. Such structure constitutes what is defined as external links (Halasz & Schwartz, 1994), and has the following advantages: we can define any number of automata over a hyperbase and the linked files themselves are not changed by creating hypertext references between them; any HTML file can be edited without altering the automaton structure; in terms of reuse of hypermedia materials, once there is no hard-coded links in the pages it is a straightforward procedure.

3 Adaptive Hypermedia and Anticipation

An important topic of work on using this automaton model involves the problem of matching a web course automaton to an audience. Because readers have different interests, authors tend to create automata with branching paths and side trips. The issue then becomes how to determine which branch to take at the various choice points. Another problem is how to build different web courses starting from the same set of hypermedia instructional materials (for instance, see Fig. 1).

To make the scope of this paper more clear, we use the following definition for adaptive hypermedia: "by adaptive hypermedia systems we mean all hypertext and hypermedia systems which reflect some features of the user in the user model and apply this model to adapt various visible aspects of the system to the user" (Brusilovsky, 1996). The problem of user modeling, i.e. building and updating the user model in adaptive systems is not a focus of this paper.

According to (De Bra, 1999; Brusilovsky, 1996), an adaptive hypermedia system tries to guide the user manipulating the link structure or the link presentation. The adaptive system also provides additional or alternative information to ensure that the most relevant information is shown and that the user can understand the information as it is presented (some technical terms may need to be explained or avoided for instance). In other words, the system tries to use knowledge about a particular user, represented in the user model, to adapt the information and links being presented to that user. Adaptation can also protect the user from getting lost in hyperspace.

In the sequence of works (De Bra & Calvi, 1998; De Bra, 1999), adaptive hypermedia concept is related to systems that try to anticipate (Dubois, 1998) the needs and desires of the user. Depending on the user's knowledge, state information on a given subject may need to be presented in different ways. Students who are first reading about hypertext for instance may be confused when they see the term "node" whereas the word "page", used in the same context, would be meaningful to them.

(Brusilovsky, 1996, 1997, 1998, 1999) distinguishes several types of adaptation technologies, mainly Adaptive Presentation (adaptive multimedia presentation, adaptive text presentation) and Adaptive Navigation (direct guidance, adaptive sorting of links, adaptive hiding of links, adaptive annotation of links, map adaptation).

The goal of adaptive presentation (or content-level adaptation) technology is to adapt the content of a hypermedia page to the user's goals, knowledge and other information stored in the user model. In a system with adaptive presentation, the pages are not static, but adaptively generated or assembled from pieces for each user. For example, with several adaptive presentation techniques, expert users receive more detailed and deep information, while novices receive more additional explanation.

The goal of the adaptive navigation (or link-level adaptation) is to support the student in hyperspace orientation and navigation by changing the appearance of visible links. In particular, the system can adaptively sort, annotate, or partly hide the links of the current page to make easier the choice of the next link to proceed. The goal is to help students find an "optimal path" through the learning material.

In section 4, we show how the Hyper-Automaton framework copes with the task of providing both content and link-level adaptation.

In general, the adaptation procedures are implemented as a function of the knowledge the system has about the user's prior actions (e.g. the navigation path over a hyperbase) or the system tries to anticipate the user's goal based on a predefined user's model (e.g. goal driven navigation). An adaptive web test using the Hyper-Automaton - a Mealy Machine in which the sequence of questions (easy, medium and hard levels) is determined by the answer to the previous question - is an example of adaptation based on past events. Previewing the future browsing path a user will follow based on the categorization of classes of user - obtained from user interaction logs over a hyperbase - is an example of anticipative behavior on adaptive hypertext.

4 Categorial Operations as Hypertext Documents Composition and Adaptation

In what follows, we show how the categorial constructions of limits, colimits, restriction, relabeling and reification morphisms can be interpreted to define (possible complex) hypertext documents for Web courses.

A path-control mechanism can be implemented using restriction rules applied over the Hyper-Automaton. Basically, we have 3 different restrictions: *transitions* - the author may select specific transitions which will not be available in the automaton anymore, thus removing possible links between content pages; *labels* - one can choose transition labels to be removed from an entire automaton, restricting the available browsing paths; *states* - the author may want to remove certain contents from the hyperdocument, this operation can be

accomplished by removing states from the automaton and corresponding labeled transitions. In the adaptive context, the restriction morphisms are related to the adaptive navigation process. In particular, the adaptation procedure called link removal results from the restriction of transitions and labels. A restriction applied to states corresponds to adaptive presentation of contents where selected states are removed and the corresponding hyperdocuments become unavailable to the user.

Another operation is built over the input alphabet. The author can relabel in order to alter the link's keyword anchors inside a set of hyperdocuments in a fast and safe procedure, without the need to alter the HTML code of the pages. The relabeling procedure also represents the adaptive presentation of hyperdocuments. The relabel morphism allows the system to adapt the content of a page (name of link anchor or the output hyperdocument) according to the user's goal.

In some cases, a restriction can be simulated by a relabeling with encapsulation, i.e., instead of "erasing" a transition, in many cases it is possible to encapsulate it (an encapsulated transition still exists but can not be selected by the user).

During the development phase of a hyperdocument, the author may be interested in gradually adding new contents to the hyperbase. Usually these materials are not completely finished and the author may want to specify in the automaton structure which states are in "beta form", so the users will not be allowed to browse them. Such procedure illustrates an application of the categorial construction of encapsulating (hiding or windowing) components of an automaton. The procedure known as link disabling (an adaptive navigation process) can be obtained with a special kind of relabeling of the input alphabet (see encapsulation in section 7), which means that the link is not made invisible but its link functionality is removed.

In the context of web-based courses, reification morphisms are related to the concept of teaching lessons or lectures. In other words, a lesson is defined as a specific sequence (transition mapped into transactions in a hyperbase automaton) of selected hypermedia elements that must be fully browsed by the students. As the reification process implies a kind of "atomic transaction" in the sense that it is (a possible complex) task that must be fully done, it also supports the concept of tests or formative assessment (a set of questions must be completed in order to the students accomplish the task).

In terms of anticipation (Dubois, 1998), a set of relabel and restriction morphisms represents the models of possible adaptations that can be applied to the presentation and navigation of the current hyper-automaton representing the user's browsing session (see Fig. 1), and it is quite practical to think of generating even more models as the user gets involved in a certain navigation. During the navigation over the hyperbase, several different hyper-automata for that set of hyperdocuments may be created by the system and the adaptation mechanisms (restrictions, relabelings, and reifications) represent the instantiation of one possible future automaton in the current automaton for the hyperbase. The dynamic creation of hyper-automaton is not the goal of this article. However, all features to achieve this purpose are presented.

Example 1 The example depicted in Fig. 2 (Menezes & Machado, 2000)shows how categorial operations are used as course composition scheme in order to build new courses

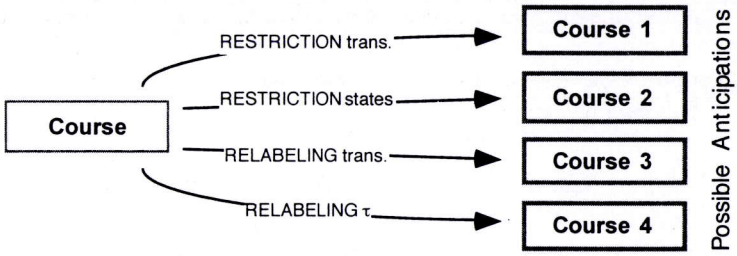

Fig. 1: Relabeling and Restriction as Adaptive Navigation (Linking)

from a set of existing courses. In Fig. 2, the single boxes stand for the original course automata for a given hyperbase, the highlighted boxes are the courses resulted from the categorial operations, shown as labeled arrows. Course represents the hypertext version of the book "Formal Language and Automata" (Menezes, 2000b), consisting of texts and illustrative pictures; Simulator is the set of HTML pages designed to frame a Java applet that simulates automata and related concepts, for example; Exercises 1...n represents a database of formative questions; Full Course is the desired resulting course and Courses w/ Lectures and Tests is the full course organized in lessons (or lectures) and tests. In this context, we have that:

a) Colimit. The Professor wants to improve the course and decides to include an existing database of questions, so the students can access related exercises. The Course with Exercises is the resulting object of a colimit between Course and Exercises i;
b) Product. In the next step, the Professor builds a course containing links attached to the content pages pointing to a simulator that helps the students learn the concepts introduced in the text pages. A resulting object of the product operation between Course w/Exercises and Simulator is such that students can access the Java applet simulator from every page;
c) Restriction and Relabeling. In order to attend requests from more advanced students, the Professor builds a course with an alternative path or trail as an "expert" version of the book contents without simulators or exercises. An Expert Course is a resulting object of a restriction and relabeling operations in order to remove undesirable labeled transitions and relabel transitions for the new purpose;
d) Coproduct. The Full Course is a resulting object of the coproduct operation between the "enhanced" Easy Course and the "plain" Expert Course;
e) Reification. The Full Course is organized in terms of lessons and tests which are a kind of "atomic transaction" in the sense that they are (a possible complex) task that must be fully done, resulting in a Course w/ Lectures and Tests;
d) Encapsulation. In this example, Subject on Development represents web course pages that are not yet finished and the Professor wants to "hide" or "encapsulate" the new subjects so the students will not be allowed to browse them. The Encapsulation operation (or relabeling to τ) results in Course w/ Encapsulated Subject. ❑

Remark 2 In the universe of categories, all the original components involved in operations remain intact or are easily recuperated. For instance, in the example depicted on Fig. 2, the creation of Course w/Exercises is such that the original modules Course and Exercises 1...n remain unchanged. ❑

Remark 3 It is important to highlight that the categorial operations provide a support for the definition of a high-level language for the creation and the organization of web courses. ❑

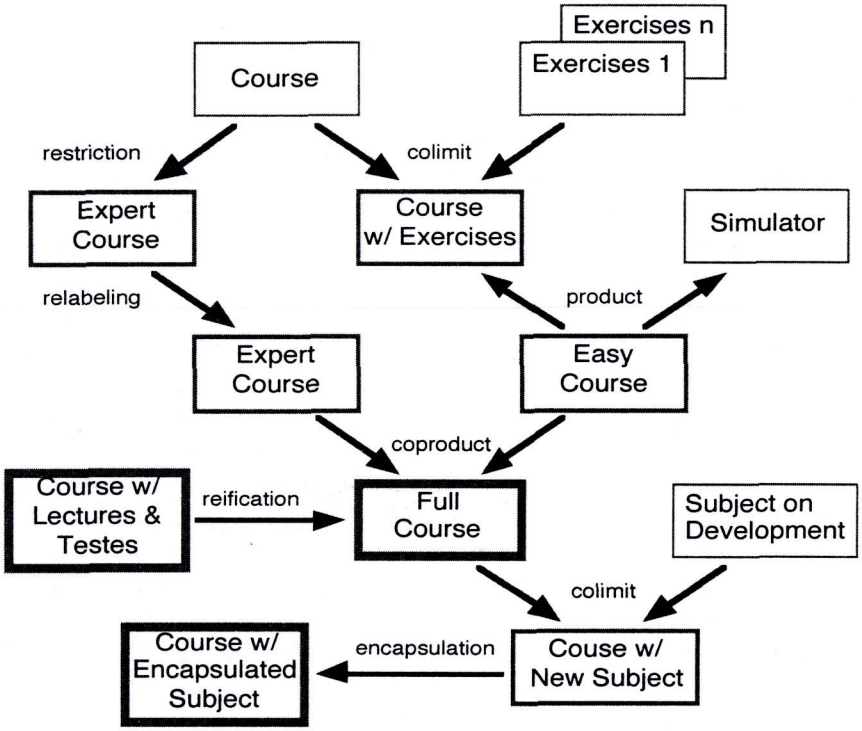

Fig. 2: Construction of a Composed Course

Example 4 The example depicted in Fig. 3 shows how categorial operations (mainly relabeling and restriction morphisms) are used to define adaptations in adaptive web courses. In this example, a small portion of a course is composed of content pages, an example and a related exercise test. The Course1 (for beginners) represents the adaptation procedure called link removal in which a specific transition is removed from the course; in other words, a restriction on transition results in a course where the exercise is only available after the user sees an example. Link removal can also be obtained by restricting states, in this case the Course2 (for advanced users) had the example and exercise completely removed. Course3 (for intermediate level students) is obtained by relabeling transitions to τ, which is equivalent to a special kind of adaptive navigation known as link hiding; thus, the user will have access only to the example pages. ❑

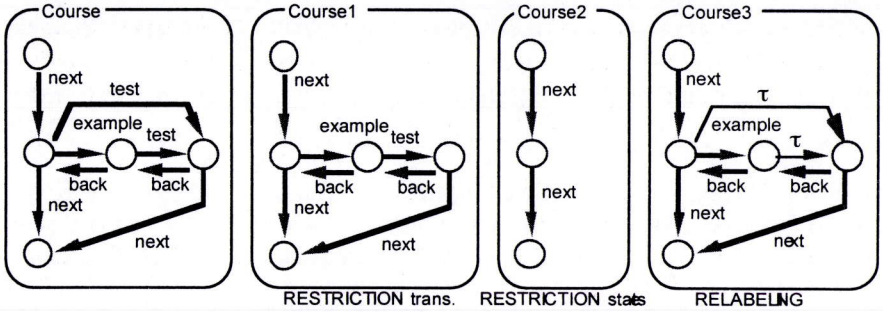

Fig. 3: Adaptations in Web Course

5 Internal (Labeled) Graphs

As stated in (Corradini, 1990) and (Asperti & Longo, 1991) a graph can be considered as a diagram in the category *Set* where nodes and arcs are sets and the source and target operations are (total) functions. Moreover, graph morphisms are commutative diagrams in *Set*. This means that *Set* plays the role of "universe of discourse" of the category of graphs: it is defined internally to the category *Set*. This suggests a generalization of graphs as diagrams in an arbitrary universe (base) category. This approach is known as *internalization* and can be extended for reflexive graphs and categories. In this context, limits and colimits of categories of internal graphs are inherited from the categories of nodes and arcs. The results about inheritance of limits and colimits properties from the base category are, for our knowledge, new.

Definition 5 Let C be a (base) category. The *Category of Internal Graphs* denoted by *Gr(C)*, is the comma category $\Delta_C \downarrow \Delta_C$ where Δ_C: $C \rightarrow C^2$ is the diagonal functor such that sends each C-object A to the C^2-object $\langle A, A\rangle$ and sends each C-morphism f: $A \rightarrow B$ to the C^2-morphism $\langle f, f\rangle$: $\langle A, A\rangle \rightarrow \langle B, B\rangle$. ❑

Therefore, a *Gr(C)*-object G is a quadruple $G=\langle V, T, \partial_0, \partial_1\rangle$ where V, T are C-objects, ∂_0, ∂_1: $T \rightarrow V$ are C-morphisms and a *Gr(C)*-morphism $h=\langle h_V, h_T\rangle$: $\langle V_1, T, \partial_{01}, \partial_{11}\rangle \rightarrow \langle V_2, T, \partial_{02}, \partial_{12}\rangle$ where h_V: $V_1 \rightarrow V_2$ and h_T: $T_1 \rightarrow T_2$ are C-morphisms, is such that $h_V \circ \partial_{i1} = \partial_{i2} \circ h_T$, for $i \in \{0, 1\}$.

Proposition 6 If C is complete (cocomplete) then *Gr(C)* is complete (cocomplete).

Proof: Since *Gr(C)* is the comma category $\Delta_C \downarrow \Delta_C$, the proof is a direct corollary (see, for instance, (Casley, 1991)). Remember that Δ_C preserves colimits and limits (since it has right and left adjoints - see (Mac Lane, 1971)). ❑

Definition 7 Let C be a (base) category. An internal reflexive graph is a quadruple $G = \langle V, T, \partial_0, \partial_1, \iota\rangle$ where $\langle V, T, \partial_0, \partial_1\rangle$ is a *Gr(C)*-object and ι: $V \rightarrow T$ is a C-morphism such that $\partial_i \circ \iota = id_V$, for $i \in \{0, 1\}$. A morphism between internal reflexive graphs $h = \langle h_V, h_T\rangle$: $\langle V_1, T, \partial_{01}, \partial_{11}, \iota_1\rangle \rightarrow \langle V_2, T, \partial_{02}, \partial_{12}, \iota_2\rangle$ is a *Gr(C)*-morphism such that $\iota_2 \circ h_V = h_T \circ \iota_1$. Internal reflexive graphs and the corresponding morphisms constitute the *Category of Internal Reflexive Graphs*, denoted by *RGr(C)*. ❑

Proposition 8 If C is complete (cocomplete) then $RGr(C)$ is complete (cocomplete).

Proof: Let $u: RGr(C) \to Gr(C)$ be a forgetful functor such that, for each $RGr(C)$-object $G = \langle V, T, \partial_0, \partial_1, \iota \rangle$, $u\,G = \langle V, T, \partial_0, \partial_1 \rangle$ and for each $RGr(C)$-morphism $h = \langle h_V, h_T \rangle$: $G_1 \to G_2$, $u\,h = \langle h_V, h_T \rangle$: $u\,G_1 \to u\,G_2$. Since u is a faithful functor, $\langle RGr(C), u \rangle$ is concrete category over $Gr(C)$ (Adámek *et al*, 1990). Suppose that C is complete (cocomplete). Then $Gr(C)$ is complete (cocomplete). Therefore, to prove that $RGr(C)$ is complete (cocomplete) we have just to prove that for each $RGr(C)$-diagram D, the limit (colimit) of D in $Gr(C)$ can be lifted as a initial source (final sink) in $RGr(C)$. Suppose I a family of indexes, $i \in I$ and $k \in \{0, 1\}$. For simplicity, in what follows, we omit that $i \in I$ and $k \in \{0, 1\}$. *Lifting products*. Let $\{G_i = \langle V_i, T_i, \partial_{0i}, \partial_{1i}, \iota_i \rangle\}$ be an indexed family of $RGr(C)$-objects and $\times u\,G_i = \langle \times V_i, \times T_i, \times \partial_{0i}, \times \partial_{1i} \rangle$ the corresponding $Gr(C)$-product together with $\{\pi_i: \times u\,G_i \to u\,G_i\}$. Then, $\times G_i = \langle \times V_i, \times T_i, \times \partial_{0i}, \times \partial_{1i}, \times \iota_i \rangle$, together with $\{\pi_i: \times G_i \to G_i\}$ is an initial source of $\{G_i\}$ where $\times \iota_i$ is uniquely induced by the product construction as illustrated in Fig. 4 (left). *Lifting equalizers*. Consider the $RGr(C)$-objects $G_1 = \langle V_1, T_1, \partial_{01}, \partial_{11}, \iota_1 \rangle$, $G_2 = \langle V_2, T_2, \partial_{02}, \partial_{12}, \iota_2 \rangle$, the $RGr(C)$-morphisms $f_1 = \langle v_1, t_1 \rangle$, $f_2 = \langle v_2, t_2 \rangle$: $G_1 \to G_2$ and the corresponding $Gr(C)$-equalizer $\langle e_V, e_T \rangle$: $\langle V, T, \partial_0, \partial_1 \rangle \to u\,G_1$. Then $G = \langle V, T, \partial_0, \partial_1, \iota \rangle$ together with $e = \langle e_V, e_T \rangle$: $G \to G_1$ is an initial source in $RGr(C)$, where ι is uniquely induced by the C-equalizers, as illustrated in Fig. 4 (right). The lifting of coproducts and coequalizers are analogous. ❑

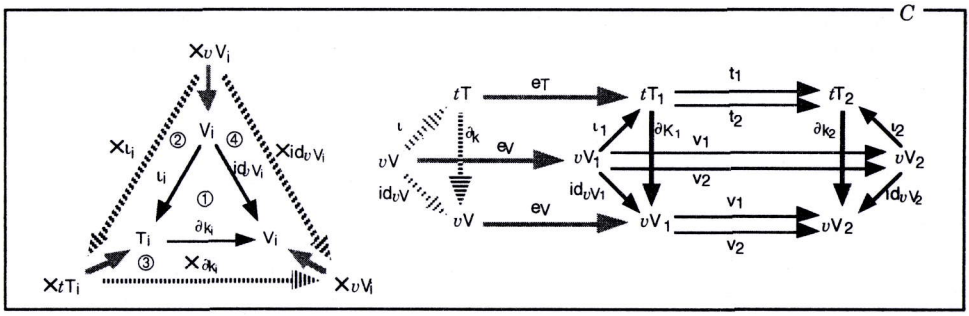

Fig. 4: Morphisms uniquely induced

As the Hyper-Automaton is build over the concept of automata, we need to identify transitions (hypertext links) and states (hypertext nodes). We do that by labeling arcs and nodes in a graph.

Usually, the labeling of graphs is restricted to arcs. However, it may be the case that labeling must be both, on arcs and nodes and, in this case, it is usual that labeling for each arc preserves the labeling of corresponding source and target nodes. In this context, labeling can be seen as a graph-morphism where the source graph represents the "shape" and the target one represents the "labels" (of arcs and nodes). The approach is analogous for all internal graphs introduced. Therefore, $G(C)$ denotes $Gr(C)$ and $RGr(C)$.

Definition 9 The *Category of Internal Labeled Graphs* is the comma category $LG(shape, lab) = shape \downarrow lab$ where $shape: G(S) \to G(C)$, $lab: Lab \to G(C)$ are functors, $G(S)$ is the category of "shapes" and Lab is the category of "labels". ❑

Therefore, a *LG(shape, lab)*-object is a triple $A = \langle G, lab, L \rangle$, where the "shape" G is a *G(S)*-graph, the "labels" L is a *Lab*-object and lab: $shape\, G \rightarrow lab\, L$ is a graph morphism in *G(C)* corresponding to the labeling. A *LG(shape, lab)*-morphism is $\langle h_G, h_L \rangle: \langle G_1, lab_1, L_1 \rangle \rightarrow \langle G_2, lab_2, L_2 \rangle$ such that $lab\, h_L \cdot lab_1 = lab_2 \cdot shape\, h_G$.

Proposition 10 Let *shape: G(S)* $\rightarrow$ *G(C)*, *lab: Lab* $\rightarrow$ *G(C)* be functors. If *G(S)*, *Lab* are complete (cocomplete) and *lab* preserves limits (*shape* preserves colimits) then *LG(shape, lab)* is complete (cocomplete).

Proof: Since $LG(shape, lab) = shape \downarrow lab$, the proof is a direct corollary. ❑

6 Finite Automata

A finite automata is basically a finite graph with labeling on arcs. The corresponding category of internal graphs is finitely bicomplete. However, while the coproduct construction can be interpreted as a choice between component systems, the product construction defines a kind of "total synchronization" with little practical applications. A more useful category can be obtained using reflexive graphs. Since labeling is restricted to arcs, the target object of a labeling morphism is a reflexive one-node graph. In this case, a notion of "encapsulation" of transitions can be defined. In what follows, *FinSet* denotes the category of finite sets and finite functions which is finitely bicomplete. For simplicity, in this paper we do not deal with outputs, which is a simple extension.

Definition 11 Consider the category of finite reflexive graphs *RGr(FinSet)*, the identity functor *shape: RGr(FinSet)* $\rightarrow$ *RGr(FinSet)* and the functor *lab: FinSet* $\rightarrow$ *RGr(FinSet)* where each finite set L is taken into the corresponding one node reflexive graph $\langle \{\bullet\}, L_\tau, !, !, \iota_\tau \rangle$ such that $L_\tau = L +_{FinSet} \{\tau\}$, $!: L_\tau \rightarrow \{\bullet\}$ is the unique function and $\iota_\tau: \{\bullet\} \rightarrow L_\tau$ takes the unique element of $\{\bullet\}$ into τ. Then, the *Category of Finite Automata* is the category of internal graphs $Aut = LRGr(shape, lab)$. ❑

Thus, the shape of a sequential automata is a reflexive finite graph and labeling is over an one node reflexive graph, i.e., a *Aut*-object is a triple $A = \langle G, lab, L \rangle$ where $G = \langle V, T, \partial_0, \partial_1, \iota \rangle$ is a *RGr(FinSet)*-object, L is a *FinSet*-object and lab: $\langle V, T, \partial_0, \partial_1, \iota \rangle \rightarrow \langle \{\bullet\}, L_\tau, !, !, \iota_\tau \rangle$ is a *RGr(FinSet)*-morphism. A transition labeled by τ (the identity transition of the one node reflexive graph) can be considered as an encapsulated transition. Note that all identity transitions of the shape are labeled by τ (usually an identity transition means "no operation" and they are encapsulated). The proof of the following proposition is a direct corollary of previous results.

Proposition 12 The Category of Finite Automata *Aut* is finitely bicomplete. ❑

Example 13 In *Aut*, the coproduct and product constructions can be interpreted as choice and parallel composition (all possible combination of component transitions) as illustrated in Fig. 5. For simplicity, the label τ of the identity transitions is omitted. ❑

7 Restriction and Relabeling: Functorial Operations

Restriction and relabeling are functorial operations defined using fibration and cofibration techniques based on (Menezes & Costa, 1996) and inspired by (Winskel, 1987). Both

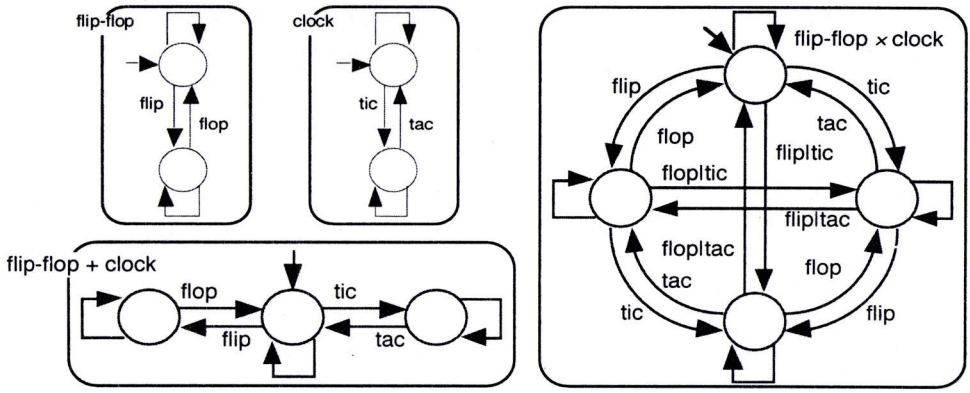

Fig. 5: Automata - coproduct and product

functors are induced by morphisms at the label level. A restriction restricts the transitions of an automaton according to some table of restrictions (of labels). A similar approach can be done for restrictions on transitions and states. A relabeling relabels the transitions of an automaton according to some morphism of labels. Encapsulation and synchronization are special cases of relabeling and restriction (synchronization restricts a parallel composition "erasing" all those transitions which do not reflect some given table of restrictions).

Proposition 14 The forgetful functor u: *Aut* $\rightarrow$ *FinSet* that takes each automaton $A = \langle G, lab, L \rangle$ onto its underlying set of labels L_τ is a fibration and a cofibration.

Proof: *Fibration*. Let $f: L_{\tau 1} \rightarrow L_{\tau 2}$ be a *FinSet*-morphism and $A_2 = \langle G_2, lab_2, L_2 \rangle$ be an automaton where $G_2 = \langle V_2, T_2, \partial_{02}, \partial_{12}, \iota_2 \rangle$ is a *RGr(C)*-object. Let the object G_1 together with lab_1: *shape* $G_1 \rightarrow$ *lab* $L_{\tau 1}$ and u_G: $G_1 \rightarrow G_2$ be the pullback of *lab* f: *lab* $L_{\tau 1} \rightarrow$ *lab* $L_{\tau 2}$ and lab_2: *shape* $G_2 \rightarrow$ *lab* $L_{\tau 2}$. Define $A_1 = \langle G_1, lab_1, L_1 \rangle$ which is an automaton by construction. Then $u = \langle u_G, f \rangle$: $A_1 \rightarrow A_2$ is cartesian with respect to f and A_2. *Cofibration*. Let $f: L_{\tau 1} \rightarrow L_{\tau 2}$ be a *FinSet*-morphism and $A_1 = \langle G_1, lab_1, L_1 \rangle$ be an automaton where $G_1 = \langle V_1, T_1, \partial_{01}, \partial_{11}, \iota_1 \rangle$. Define $A_2 = \langle G_2, lab_2, L_2 \rangle$ where $G_2 = G_1$ and lab_2 = *lab* f lab_1. Then $u = \langle id_{V1}, id_{T1}, f \rangle$: $A_1 \rightarrow A_2$ is cocartesian with respect to f and A_1. ❑

Definition 15 Consider the bifibration u: *Aut* $\rightarrow$ *FinSet*. Let $A = \langle G, lab, L \rangle$ be an automaton and restr: $Table_\tau \rightarrow L_\tau$ a restriction *FinSet*-morphism. The *Restriction* of A is *restr* A where the functor *restr*: $u^{-1} L_\tau \rightarrow u^{-1} Table_\tau$ is induced by u and restr. Let $A = \langle G, lab, L_1 \rangle$ be an automaton and relab: $L_{\tau 1} \rightarrow L_{\tau 2}$ be a relabeling morphism. The *Relabeling* of A is *relab* A where the functor *relab*: $u^{-1} L_{\tau 1} \rightarrow u^{-1} L_{\tau 2}$ is induced by u and relab. ❑

Example 16 Consider the automaton in Fig. 6 (left). Suppose that we want a joint behavior sharing the transitions flip and tic. Then, Table = { flop, tac, flop | tac, flip | tic }. The resulting restricted automaton is illustrated in the Fig. 6 (center). Suppose that relab: $L_{\tau 1} \rightarrow L_{\tau 2}$ is such that relabels flop to τ (all other labels are not relabeled). The resulting relabeled automaton is illustrated in the Fig. 6 (right). If we consider that transitions

labeled by τ are encapsulated, then the resulting automaton has two transitions encapsulated by the relabeling operation, represented by a different texture in the figure (as usual, the label τ is omitted). ❑

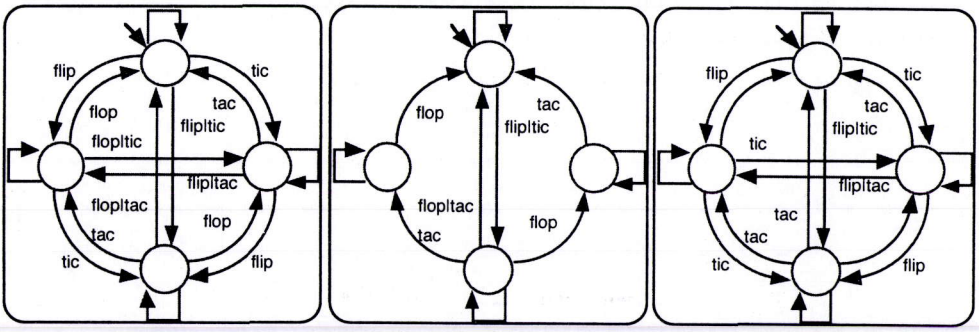

Fig. 6: Automata - Original (left), restricted (center) and encapsulated (right)

8 Internal (Labeled) Categories

The internalization approach for categories has to consider that the composition operation of morphisms is partial, i.e., for a category $B = \langle V, T, \partial_0, \partial_1, \iota, \circ \rangle$, the operation $\circ$ is defined for $\{(f, g) \mid \partial_1(f) = \partial_0(g)\}$ which is the resulting object $T \times_V T$ of the pullback of ∂_0 and ∂_1. Thus, we can generalize the internal category concept to an arbitrary universe category provided that it is finitely complete.

Definition 17 Let C be a finitely complete category. The *Category of Internal Categories* denoted by *Cat(C)* is constituted by internal categories and functors to C where:

a) An *Internal Category* is $B = \langle V, T, \partial_0, \partial_1, \iota, \circ \rangle$ where $\langle V, T, \partial_0, \partial_1, \iota \rangle$ is an *RGr(C)*-object and $\circ : T \times_V T \to T$ is a C-morphism such that the diagrams in Fig. 7 commute (the morphisms $\iota \times_V id_T$, $id_T \times_V \iota$, $\circ \times_V id_T$ and $id_T \times_V \circ$ are uniquely induced by the pullback construction);

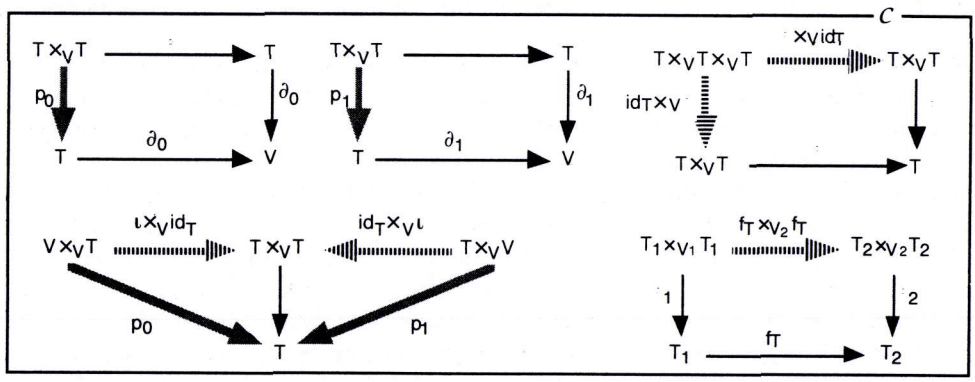

Fig. 7: Domain, codomain, associativity (top) and identity (left bottom), with respect to composition; internal functors (bottom, right)

b) An *Internal Functor* $f: B_1 \rightarrow B_2$ where $B_1 = \langle V_1, T_1, \partial_{01}, \partial_{11}, \iota_1, \circ_1 \rangle$ and $B_2 = \langle V_2, T_2, \partial_{02}, \partial_{12}, \iota_2, \circ_2 \rangle$ is a $RGr(C)$-morphisms $f = \langle f_V: V_1 \rightarrow V_2, f_T: T_1 \rightarrow T_2 \rangle$ such that diagram in Fig. 7 (bottom, right) commutes (the morphism $f_T \times_{V2} f_T$ is uniquely induced by the pullback construction). ❑

The approach for labeling of internal categories is analogous to the one for graphs.

Definition 18 The *Category of Labeled Internal Categories* is the comma category $LCat(shape, lab) = shape \downarrow lab$ where $shape: Cat(S) \rightarrow Cat(C)$, $lab: Lab \rightarrow Cat(C)$ are functors, $Cat(S)$ is the category of "shapes" and Lab is the category of "labels". ❑

Therefore, a $LCat(shape, lab)$-object is a triple $D = \langle B, lab, L \rangle$, where B is a $Cat(S)$-category representing the "shape", L is a Lab-object representing the "labels" and $lab: shape\, B \rightarrow lab\, L$ is a functor in $Cat(C)$ corresponding to the labeling.

9 Finite Computations

The Category of (Sequential) Finite Computations is the Category of Finite Automata extended with the operation of composition on transitions. Therefore, the Category of Finite Computation is just the category of all finite categories with labeling on morphisms.

Definition 19 Consider the internal category $Cat(FinSet)$, the internal subcategory of one object $Cat_1(FinSet)$, the identity functor $id: Cat(FinSet) \rightarrow Cat(FinSet)$ and the inclusion functor $inc: Cat_1(FinSet) \rightarrow Cat(FinSet)$. Then, the *Category of Finite Computation* is the category of internal labeled categories $Comp = LCat(id, inc)$. ❑

Thus, the shape of a sequential computation is a finite category and labeling is over an one object finite category. A transition labeled by τ (the identity transition of the single object) can be considered as an encapsulated transition. All identity transitions of the shape are labeled by τ.

The operation of composition on transitions gives us, in the Hyper-Automaton universe, all the possible navigation paths over the hyperbase structured as an automaton.

10 Reification of Finite Automata and Vertical Compositionality

A reification maps transitions into transactions reflecting an implementation of an automaton on top of another, based on (Menezes *et al*, 1998), but in a different framework. It is defined as a special morphism of automata where the target one (more concrete) is enriched with its computational closure that can be split into permutations of original transitions, respecting source and target states, inspired by (Meseguer & Montanari, 1990). The computational closure is easily obtained in a categorial context: there is an obvious forgetful functor from the category *Comp* to *Aut* that forgets about the composition operation; this functor has as left adjoint a functor that freely generates computations from an automaton. The envisaged computational closure is obtained by composing these two functors, i.e., $tc = ca \cdot ac$. The composition of two reification morphisms $\varphi: A_1 \rightarrow tc\, A_2$ and $\psi: A_2 \rightarrow tc\, A_3$ (where the target of φ is different from the source of ψ) is inspired by Kleisli categories. Automaton and reification constitute a new category and therefore, the *vertical compositionality is achieved*, i.e., reification of

automata compose. However, the construction does not satisfy the horizontal compositionality. For instance, the computational closure of the product of automata is in general different from the product of the computational closure of given automata.

Definition 20 Consider the categories $Comp = LCat(id, inc)$ and $Aut = LRGr(shape, lab)$. The forgetful functor $ca: Comp \to Aut$ is such that:

a) Takes each $Comp$-object $A = \langle G, lab, L \rangle$ where $G = \langle V, T, \partial_0, \partial_1, \iota, \ \rangle$ is a $Cat(FinSet)$-object, $L = \langle \{\bullet\}, L_\tau, !, !, \iota_\tau, \ L \rangle$ is a $Cat_1(FinSet)$-object and $lab: G \to inc\, L$ is a functor into the Aut-object $\mathsf{A} = \langle \mathsf{G}, \mathsf{lab}, \mathsf{L} \rangle$ where $\mathsf{G} = \langle V, T, \partial_0, \partial_1, \iota \rangle$, $\mathsf{L} = L_\tau\text{-}\{\tau\}$ and $\mathsf{lab}: \mathsf{G} \to \langle \{\bullet\}, L_\tau, !, !, \iota_\tau \rangle$ is induced by lab;

b) Takes each $Comp$-morphism $h = \langle h_G, h_L \rangle: \langle G_1, lab_1, L_1 \rangle \to \langle G_2, lab_2, L_2 \rangle$ into the Aut-morphism induced $\mathsf{h} = \langle h_G, h_L \rangle: \langle \mathsf{G}_1, \mathsf{lab}_1, \mathsf{L}_1 \rangle \to \langle \mathsf{G}_2, \mathsf{lab}_2, \mathsf{L}_2 \rangle$. ❑

Definition 21 Consider the categories $Comp = LCat(id, inc)$ and $Aut = LRGr(shape, lab)$. The functor $ac: Aut \to Comp$ is such that:

a) Takes each Aut-object $\mathsf{A} = \langle \mathsf{G}, \mathsf{lab}, \mathsf{L} \rangle$ where $\mathsf{G} = \langle V, T, \partial_0, \partial_1, \iota \rangle$ and $\mathsf{lab}: \mathsf{G} \to \langle \{\bullet\}, L_\tau, !, !, \iota_\tau \rangle$ into the $Comp$-object $A = \langle G, lab, L \rangle$ such that:

a.1) $G = \langle V, T^c, \partial_0{}^c, \partial_1{}^c, \iota^c, \ \rangle$ is a $Cat(FinSet)$-object where $\iota^c: V \to T^c$ is induced by $\iota: V \to T$ and T^c, $\partial_0{}^c$, $\partial_1{}^c$ and $\ $ are defined by the following rules of inference:

$$\frac{t: A \to B \in T}{t: A \to B \in T^c} \qquad \frac{t: A \to B \in T^c \quad u: B \to C \in T^c}{t;u: A \to C \in T^c}$$

subject to the following equational rules:

$$\frac{t: A \to B \in T^c}{\iota_A;t = t \ \ \& \ \ t;\iota_A = t} \qquad \frac{t: A \to B \in T^c \quad u: B \to C \in T^c \quad v: C \to D \in T^c}{t;(u;v) = (t;u);v}$$

a.2) $L = \langle \{\bullet\}, L_\tau{}^c, !, !, \iota_\tau, \ L \rangle$ where $L_\tau{}^c$ is defined as above such that $\iota_\tau: \{\bullet\} \to L_\tau$ takes the unique element of $\{\bullet\}$ into τ and $lab: G \to inc\, L$ is induced by the free generation above;

b) Takes each $Comp$-morphism $\mathsf{h} = \langle h_G, h_L \rangle: \langle \mathsf{G}_1, \mathsf{lab}_1, \mathsf{L}_1 \rangle \to \langle \mathsf{G}_2, \mathsf{lab}_2, \mathsf{L}_2 \rangle$ into the Aut-morphism $h = \langle h_G, h_L \rangle: \langle G_1, lab_1, L_1 \rangle \to \langle G_2, lab_2, L_2 \rangle$ induced by the free generation. ❑

Proposition 22 The functor $ac: Aut \to Comp$ is left adjoint to $ca: Comp \to Aut$.

Proof: Consider $\eta = \langle \eta_G, \eta_L \rangle: id_{Aut} \to ca\ ac$ a natural transformation which is an embedding on arcs of each component graph. Thus, for each automaton $\mathsf{A} = \langle \mathsf{G}, \mathsf{lab}, \mathsf{L} \rangle$, computation $A = \langle G, lab, L \rangle$ and Aut-morphism $\mathsf{f}: \mathsf{A} \to ca\,A$, there is only one $Comp$-morphism $\mathsf{g}: ac\,\mathsf{A} \to A$ such that $\mathsf{f} = ca\,\mathsf{g} \ \eta\mathsf{A}$. In fact g is $ac\,\mathsf{f}$ restricted to the target object A. By duality, $\varepsilon = \langle \varepsilon_M, \varepsilon_E \rangle: ac\ ca \to id_{Comp}$ is a natural transformation which takes each freely composed arc $\langle t \ \ u \rangle$ into the arc $\langle t \ u \rangle$ of each component category. ❑

Definition 23 The *Computational Closure Functor* is $tc = ca\ ac: Aut \to Aut$. ❑

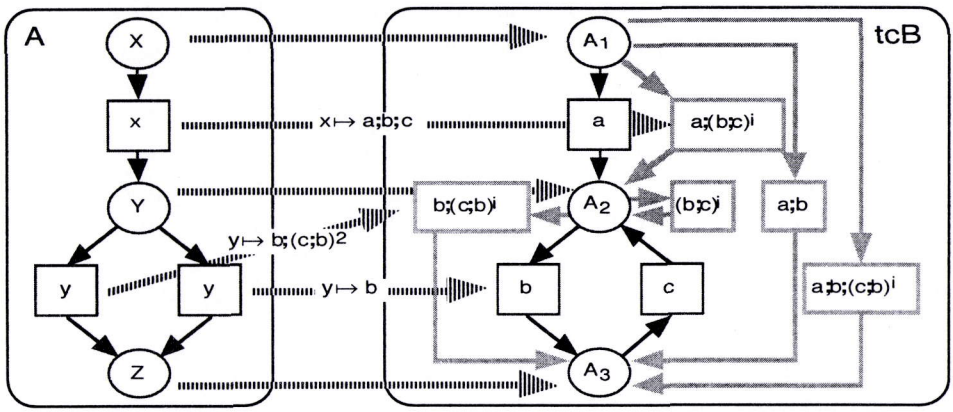

Fig. 8: Reification morphism from the automaton A into the computations of B

Example 24 In the Fig. 8 the automaton B is represented with its computational closure (the structures added are represented using a different line). Since there is a cycle, the resulting automaton has an infinite number of transitions, represented by $i \in \{1, 2, 3, ...\}$ ❑

Let $\langle ac, ca, \eta, \varepsilon \rangle$: $Aut \rightarrow Comp$ be the adjunction above. Then, $T = \langle tc, \eta, \mu \rangle$ is a monad on *Aut* (see, for instance, (Asperti & Longo, 1991)) such that $\mu = ca\,\varepsilon\,ac$: $tc^2 \rightarrow tc$ where ca: $ca \rightarrow ca$ and ac: $ac \rightarrow ac$ are the identity natural transformations and $ca\,\varepsilon\,ac$ is the horizontal composition of natural transformations. A monad is useful to understand the computations, i.e.:

- $tc\,A$ is the automaton A with all its possible computations
- η_A: $A \rightarrow tc\,A$ includes A into its computations
- μ_A: $tc^2 A \rightarrow tc\,A$ flattens computations of computations into computations.

If we define a reification morphism as an *Aut*-morphism φ: $A \rightarrow tc\,B$, then the composition of reifications can be as in Kleisli categories (each adjunction induces a monad, which defines a Kleisli category). However, for several applications, reifications should not preserve labeling (and thus, they are not *Aut*-morphisms). As we show below, each reification induces a *Aut*-morphism resulting in a category whose morphisms are *Aut*-morphisms induced by reifications. Both categories are isomorphic.

Definition 25 Let $T = \langle tc, \eta, \mu \rangle$ where $\eta = \langle \eta_G, \eta_L \rangle$, $\mu = \langle \mu_G, \mu_L \rangle$ be the monad induced by the adjunction $\langle ac, ca, \eta, \varepsilon \rangle$: $Aut \rightarrow Comp$. The *Category of Finite Automata and Reifications ReifAut*, is such that (suppose the *Aut*-objects $A_k = \langle G_k, lab_k, L_k \rangle$, for $k \in \{1, 2, 3\}$): *Objects*. *ReifAut*-objects are the *Aut*-objects; *Morphisms*. $\varphi = \varphi_G$: $A_1 \rightarrow A_2$ is a *ReifAut*-morphism where φ_G: $G_1 \rightarrow tc\,G_2$ is a *RGr(FinSet)*-morphism and for each *Aut*-object A, $\varphi = \eta_G$: $A \rightarrow A$ is the identity morphism of A in *ReifAut*; *Composition*. Let φ: $A_1 \rightarrow A_2$, ψ: $A_2 \rightarrow A_3$ be *ReifAut*-morphisms. The composition $\psi \circ \varphi$ is a morphism $\psi_G \; K\varphi_G$: $A_1 \rightarrow A_3$ where $\psi_G \; K\varphi_G$ is as illustrated in the Fig. 9. ❑

In what follows, an automaton $\langle G, lab, L \rangle$ may be denoted as a morphism lab: $G \rightarrow lab\,L$ or just by lab: $G \rightarrow L$.

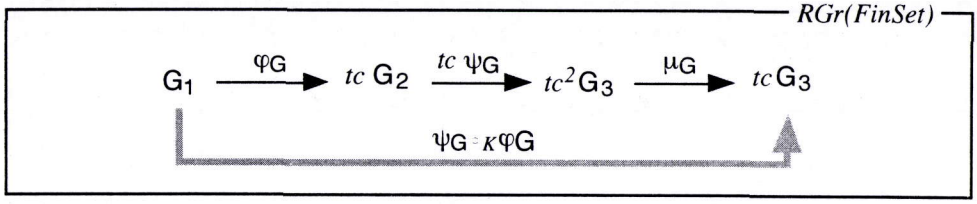

Fig. 9: Composition of reifications: composition in the Kleisli category forgetting about the labeling

Definition 26 Let $T = \langle tc, \eta, \mu \rangle$ where $\eta = \langle \eta_G, \eta_L \rangle$, $\mu = \langle \mu_G, \mu_L \rangle$ be the monad induced by the adjunction $\langle ac, ca, \eta, \varepsilon \rangle$: *Aut*$\rightarrow$*Comp)*. The *Category of Finite Automata and Reifications with Induced Labeling* $ReifAut_L$, is such that (suppose the *Aut*-objects $A_k = \langle G_k, lab_k, L_k \rangle$, for $k \in \{1, 2, 3\}$): *Objects.* $ReifAut_L$-objects are the *Aut*-objects; *Morphisms.* Let φ_G: $G_1 \rightarrow tc\,G_2$ be a *RGr(FinSet)*-morphism. Then $\varphi = \langle \varphi_G, \varphi_L \rangle$: $A_1 \rightarrow A_2$ is a $ReifAut_L$-morphism where φ_L is given by the pushout illustrated in the Fig. 10 (left). For each *Aut*-object A, $\varphi = \langle \eta_G$: $G \rightarrow tc\,G$, φ_L: $L \rightarrow L_\eta \rangle$: $A \rightarrow A$ is the identity morphism of A in $ReifAut_L$ where φ_L is as above; *Composition.* Let φ: $A_1 \rightarrow A_2$, ψ: $A_2 \rightarrow A_3$ be $ReifAut_L$-morphisms. The composition $\psi \circ \varphi$ is a morphism $\langle \psi_G \circ_K \varphi_G, \psi_L \circ_L \varphi_L \rangle$: $A_1 \rightarrow A_3$ where $\psi_G \circ_K \varphi_G$ e $\psi_L \circ_L \varphi_L$ is as illustrated in the Fig. 10 (right). ❑

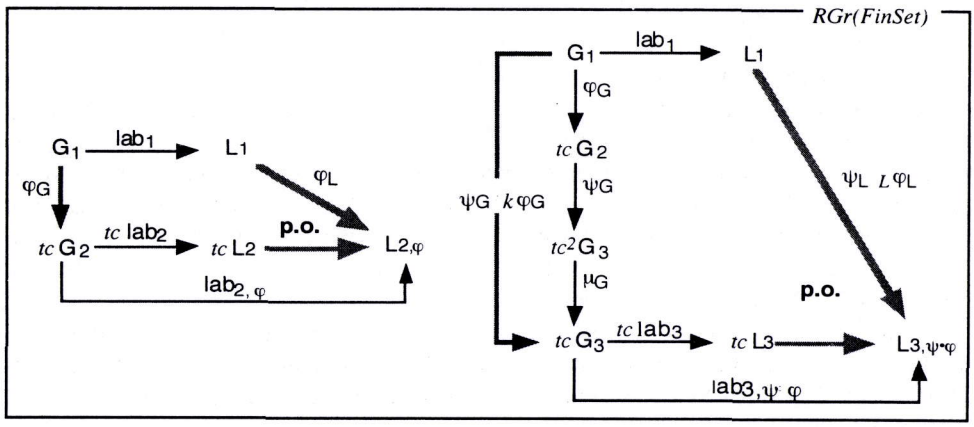

Fig. 10: Reification with induced labeling

It is easy to prove that *ReifAut* and $ReifAut_L$ are isomorphic (we identify both categories by *ReifAut*). Thus, every reification morphism can be viewed as a *Aut*-morphism. For a *ReifAut*-morphism φ: $A \rightarrow B$, the corresponding *Aut*-morphism is denoted by φ: $A \rightarrow tc\,B$.

Since reifications constitute a category, the *vertical compositionality is achieved.*

Proposition 27 The endofunctor *tc: Aut*$\rightarrow$*Aut* does not preserve limits.

Proof: Consider the automata A_1, A_2 and the resulting object of the product $A_1 \times A_2$ as in Fig. 11. They also represent the automata $tc\,A_1$, $tc\,A_2$ e $tc\,A_1 \times tc\,A_2$, respectively. However, $tc\,A_1 \times tc\,A_2$ is different from $tc\,(A_1 \times A_2)$. ❑

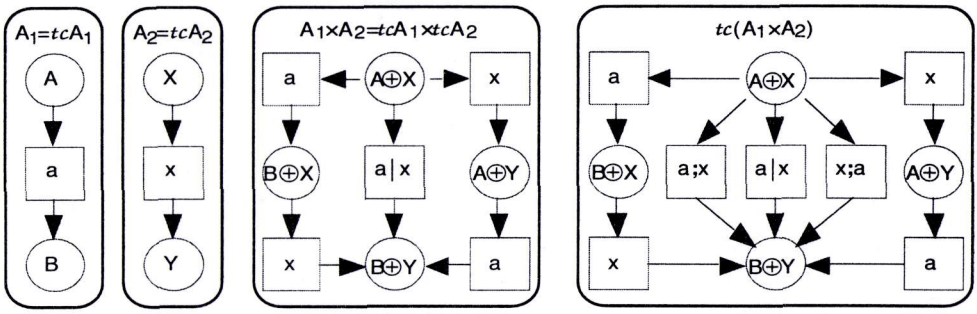

Fig. 11: The functor computational closure does not preserve limits

Therefore, the proposed model does not satisfy the horizontal compositionality. For instance, the product of two reification morphisms may not be a reification morphism. This means that reifications should be done after (or mixed with) operations based on limits and colimits. If the horizontal compositionality is required, then we may construct and adjunction between the categories of automata and nonsequential automata. In fact, we show in (Menezes & Costa, 1995) that nonsequential automata is the least concrete model (among the compared models) to satisfy the horizontal compositionality, extending the approach in (Sassone *et al*, 1993).

11 Concluding Remarks

Recently, adaptive hypermedia as a direction of research has received special attention in the WWW context. We argue that adaptive hypermedia is one of the ways to increase the functionality of WWW. Adaptive technologies can contribute to several directions of research and development on Web-based Educational Systems. Adaptive presentation can improve the usability of course material presentation. Adaptive navigation support and adaptive sequencing can be used for overall course control and for helping the student in selecting most relevant tests and assignments.

In this paper we have developed a categorial framework named Hyper-Automaton, based on Automata Theory, Category Theory and Hyperdocuments technology for the definition of Adaptive Web Courses. In fact, the results are applicable to hypertext systems as a whole.

To experiment the proposed model, we defined the Hyper-Automaton System where courses are automata with output, and categorial operations play the role of hyperdocuments composition and adaptation (possibly representing anticipations). Hyper-Automaton leads to a high level modularization of the hypertext materials, with the following advantages: easy reusing pages in several documents, thus avoiding redundancy; hyperdocuments are independent from the automaton structure which can be modified without interfering in page design and vice versa; allows users to create links from and to any document; easy to implement and to maintain; it has a simple and direct graphic interface; permits the elaboration of different studying paths with specific goals, capable of providing self-study; categorial operations are used as composition and adaptation scheme in order to build new documents from a set of existing ones through high level

constructions; it defines constructors for both adaptive presentation and adaptive navigation.

The proposed framework is based on internal (reflexive) graphs and categories, making changes and proofs very easy. For the near future we plan to extend the framework possibly using category-based Graph Transformations as proposed in (Menezes, 1999) for the inclusion of new operations as expansion, abstraction, addition, deletion, etc. We are also working on the definition of relabeling and restriction morphisms over the output function of the Mealy and Moore Machines.

Acknowledgements

This work partially supported by CNPq (HoVer-CAM, MEFIA), FAPERGS (QaP-For) and UFRGS in Brazil.

References

Adámek J., Herrlich H., & Strecker G. (1990). *Abstract and Concrete Categories*, Wiley.

Asperti A. & Longo G. (1991). *Categories, Types and Structures - An Introduction to the Working Computer Science*, Foundations of Computing (M. Garey, A. Meyer, Eds.), MIT Press.

Brusilovsky P. (1996). *Methods and Techniques of Adaptive Hypermedia*, User Modeling and User-Adapted Interaction, v.6, n.2-3, p.87-129.

Brusilovsky P. (1997). *Efficient Techniques for Adaptive Hypermedia*, Intelligent Hypertext: Advanced Techniques for the World Wide Web (Nicholas C. & Mayfield J. eds.), LNCS 1326, Springer-Verlag, p.12-30.

Brusilovsky P. (1998). *Adaptive Educational Systems on the World-Wide-Web: A Review of Available Technologies*, Workshop WWW-Based Tutoring (Stern M.K., Woolf B. & Murray T. eds.), 4th International Conference on Intelligent Tutoring Systems, San Antonio.

Brusilovsky P. (1999). *Adaptive and Intelligent Technologies for Web-based Education*, Special Issue on Intelligent Systems and Teleteaching (Rollinger C. & Peylo C. eds.), Künstliche Intelligenz, v.4, p.19-25.

Casley R. T. (1991). *On the Specification of Concurrent Systems*, Ph.D. thesis, Univ. of Stanford.

Conklin J. (1987). *Hypertext: An Introduction and Survey*. IEEE Computer, v.20, n.9, p.17-41.

Corradini A. (1990). *An Algebraic Semantics for Transition Systems and Logic Programming*, Ph.D. thesis, technical report TD-8/90, Università di Pisa.

De Bra P. & Calvi L. (1998). *AHA: a Generic Adaptive Hypermedia System*, II Workshop on Adaptive Hypertext and Hypermedia (Brusilovsky P. & De Bra P. eds.), HYPERTEXT'98, ACM, Pittsburgh.

De Bra P. (1999). *Design Issues in Adaptive Web-Site Development*, II Workshop on Adaptive Systems and User Modeling on the WWW (Brusilovsky P. & De Bra P. eds.), 8th International World Wide Web Conference, TUE Computing Science Report 99-07, Eindhoven University of Technology, Toronto.

Dubois D. (1998). *Introduction to Computing Anticipatory Systems*, International Journal of Computing Anticipatory Systems (Dubois D. ed.), CHAOS, Liège, v.2, p.3-23.

Halasz F. & Schwartz M. (1994). *The Dexter Hypertext Reference Model*, Communications of the ACM, 1994, v.37, n.2, p.30-39.

Hopcroft J. E. & Ullman J. D. (1979). *Introduction to Automata Theory, Languages and*

Computation, Addison-Wesley.
Mac Lane S. (1971). *Categories for the Working Mathematician*, Springer-Verlag.
Machado J. P., de Morais C. T. Q., Menezes P. B. & Reis R. (2000). *Structuring Web Course Pages as Automata: revising concepts*, Recherche d'Informations Assistée par Ordinateur 2000, 6 Conference on Content-based Multimedia Information Access (Mariani J. & Harman D. eds.), CID-Centre de Hautes Etudes Internationales d'Informatique Documentaire, Paris, v.1, p.150-159.
Maurer H. (1997). *Necessary Ingredients of Integrated Network Based Learning Environments*, Proceedings of the ED-MEDIA/ED-TELECOM Educational Multimedia/Hypermedia and Telecommunications (Müldner T. & Reeves T.C. eds.), AACE-Association for the Advancement of Computing in Education, Charlottesville, v.1, p.619-624.
Menezes P. B. (1999). *A Categorial Framework for Concurrent, Anticipatory Systems*, CASYS'98 - Second International Conference on Computing Anticipatory Systems, AIP Conference Proceedings 465 (Dubois D. ed.), American Institute of Physics, Woodbury, p.185-199.
Menezes P. B. (2000). *Duo-Internal Labeled Graphs with Distinguished Nodes: a Categorial Framework for Graph Based Anticipatory Systems*, CASYS'99 - Third International Conference on Computing Anticipatory Systems, International Journal of Computing Anticipatory Systems (Dubois D. ed.), CHAOS, Liège, v.6, p.75-93.
Menezes P. B. (2000b). *Formal Languages and Automata* (in Portuguese, Linguagens Formais e Autômatos), 3 edition, Sagra-Luzzatto, Brazil.
Menezes P. B. & Costa J. F. (1995). *Compositional Refinement of Concurrent Systems*, Journal of the Brazilian Computer Society - Special Issue on Parallel Computation (Medeiros C.B. ed.), n.1, v.2, p.50-67.
Menezes P. B. & Costa J. F. (1996). *Synchronization in Petri Nets*, Fundamenta Informaticae v.26.1, p.11-22, Annales Societatis Mathematicae Polonae, IOS Press.
Menezes P. B. & Machado J. P. (1999). *Web Courses are Automata: a Categorial Framework*, II Workshop on Formal Methods (Mello A.C.V. de & Moreira A.M. eds.), Universidade Federal de Santa Catarina, Florianópolis, p.79-88.
Menezes P. B. & Machado J. P. (2000). *Hyper-Automaton: Hypertext Framework with Categorial Operations*, IV Brazilian Symposium on Programming Languages (in Portuguese, IV Simpósio Brasileiro de Linguagens de Programação)(Borba P., Moura H.P. de & Santos A.L. de M. eds.), Universidade Federal de Pernambuco, Recife, p.29-47.
Menezes P. B., Sernadas A. & Costa J. F. (1998). *Nonsequential Automata Semantics for a Concurrent Object-Based Language*, First US-Brazil Workshop on Formal Foundations of Software Systems (Cleaveland R., Mislove M. & Mulry P. eds.), Electronic Notes in Theoretical Computer Science, v.14, Elsevier. URL http://www.elsevier.nl/locate/entcs/volume14.html.
Meseguer J. & Montanari U. (1990). *Petri Nets are Monoids*, Information and Computation 88, Academic Press, p.105-155.
Nielsen J. (1990). *Hypertext and Hypermedia*, Academic Press.
Sassone V., Nielsen M. & Winskel G. (1993). *A Classification of Models for Concurrency*, CONCUR 93 (E. Best, Ed.), LNCS 715, Springer-Verlag , p.82-96.
Winskel G. (1987). *Petri Nets, Algebras, Morphisms and Compositionality*, Information and Computation 72, Academic Press, p.197-238.