Simulation equivalence of automata and circuits

Miklós Bartha *
Department of Computer Science, Memorial University of Newfoundland

Abstract

Automata over a symmetric monoidal category M are introduced, and a multi-step simulation is defined among such automata. The collection of M-automata is given the structure of a 2-category on the same objects as M, in which the vertical structure is determined by groups of indistinguishable simulations. Two M-automata are called simulation equivalent if they are connected by an isomorphism of 2-cells in this 2-category. It is shown that the category of simulation equivalent M-automata is monoidal, and it satisfies all the axioms of traced monoidal categories, except the one that explicitly kills the delay.

1 Introduction

Simulation equivalence is a fundamental relationship between two synchronous systems. As introduced in [18], a synchronous system is a deterministic Mealy automaton in which states are so called configurations – determined by the values stored in the memory elements (registers) of the system – and the transition function is "wired in" through certain functional elements (e.g. logical gates). By definition, system (automaton) G can simulate system F if the following condition is satisfied.

> To every sufficiently old state u of F there corresponds a state v of G such that F and G exhibit the exact same input-output behavior when started from states u and v, respectively.

Systems F and G are simulation equivalent if they can simulate each other. The term "sufficiently old" in the definition above means that, starting from an arbitrary initial state u_0, F might be run for a number of steps (until state u) before setting the simulating state v of G. The state v then depends on u_0 and also on the inputs that have arrived at F during the delay period.

As pointed out in [18], the relevance of simulation equivalence in synchronous systems manifests itself in the ability to transform an arbitrary synchronous system into a systolic one, in which at least one register stops data on each interconnection between the functional elements. If the number of registers present in the system is sufficiently large, then the systolic transformation is achieved by shifting the registers around within the graph of the system in a suitable way. This process is called retiming in [18]. It is then proved that retiming in synchronous systems respects simulation equivalence. Thus, every synchronous system having a sufficiently large number of registers can be transformed into a simulation equivalent systolic one. The numerous advantages of dealing with a systolic, rather than an ordinary synchronous system are well-known in computer architecture design. (See again [18] as a source of reference.)

*Partially supported by Natural Science and Engineering Research Council of Canada, Discovery Grant #170493-03

It was later proved in [7] with a much more sophisticated formalism that retiming equivalence essentially coincides with simulation equivalence, at least when the functional elements are interpreted in the free algebra determined by them. Thus, the semantical concept simulation equivalence can be characterized in syntactical terms by retiming. Obviously, this result has important consequences regarding the decidability of simulation equivalence in synchronous systems [8].

The present paper takes a much more general look at simulation equivalence. Starting from an arbitrary monoidal category M, we first consider the category $Circ(M)$ constructed in [17]. The morphisms of $Circ(M)$ represent synchronous systems (automata, circuits, etc.). Then we define a quotient category $Sim(M)$ of $Circ(M)$, which reflects simulation equivalence of systems. Finally, we prove that $Sim(M)$ is a category with feedback satisfying the so called circular feedback axiom, which corresponds directly to retiming in synchronous systems.

As a motivating example, consider the monoidal category $(\boldsymbol{Set}, \times)$, where $\times$ is product of sets. A $\boldsymbol{Set}$-automaton $A \to B$ is a pair (U, α), where U is a set and $\alpha : U \times A \to U \times B$ is a function. Clearly, (U, α) is a standard deterministic Mealy automaton with states U, input alphabet A, and output alphabet B. The automaton has no final states, and it lacks a specified initial state, too. The intuitive notion of simulation between automata (U, α) and (V, β) has been explained above. The formal definition of simulation is a bit more complex and requires more than the intuitive concept. Nevertheless, this simple heuristics as an underlying idea can easily be retrieved in Definition 4.1 below.

Unfortunately, due to space constraints, we can only include the above example in our presentation here, and ask the reader to follow the steps of the Sim construction on this particular model. A number of relating examples can, however, be found in [17] in connection with the $Circ$ construction, which the reader may want to analyze from the simulation point of view. Also for space restrictions, we must assume familiarity with basic category theory, automata theory, and some elements of computer hardware design.

2 Feedback vs. trace and iteration

Feedback, trace, and iteration in monoidal categories are strongly related concepts, which have been studied intensively through various models in mathematics and computer science. While iteration [9, 13] is primarily used as a fixed-point operation in (single-sorted) algebraic theories, feedback and trace [1, 5, 15, 17] have a more general scope covering all symmetric (or braided) monoidal categories. If, however, the underlying monoidal category is a theory, then iteration, feedback, and trace have the same expressive power and they are interchangeable using simple syntactic rules. The only significant difference between iteration theories and traced monoidal categories over an algebraic theory is the presence of the commutative axiom [9, 10], which is required in iteration theories, but not in traced monoidal categories. Thus, a traced monoidal category over an algebraic theory is a Conway theory in the sense of [9], and vice versa. This fact was essentially proved back in 1987, cf. [4, Theorem 2], where the traced monoidal category axioms were introduced on magmoids in an algebraic context. These axioms were called scheme identities in [4], and they were used to axiomatize flowchart schemes in the sense of Elgot [12]. The same identities,

with the exception of yanking (called X in [4]), were used in [5] to axiomatize synchronous schemes. See also [14] for the connection of traced monoidal categories and theories with a dagger (fixed-point, iteration). Connections with other areas of mathematics (algebra, geometry) and physics can be found in [15]. Further recent papers on this topic include [1, 2, 16, 20].

In [17], categories with feedback were defined by relaxing the traced monoidal category axioms, and the free category with feedback $Circ(M)$ generated by an arbitrary symmetric strict monoidal category M was constructed from so called circuits in M. Our concern in this paper is with monoidal categories having a feedback operation that satisfies all of the traced monoidal category axioms, except for yanking. This axiom is the "milestone" which separates feedback (with delay) from trace, i.e., feedback with no delay, therefore it must not hold in our delay-oriented models.

In our approach we shall be working with monoidal automata and simulations among them. These concepts have their origins in [6], where automata over a single-sorted algebraic theory T, called T-automata, and simulations of them with surjective and injective mappings were used to construct the free strong feedback theory $F_s(T)$ generated by T. The very same idea underlies the $Circ$ construction of [17], which is more general in its scope, but very restricted in the sense that simulations in it are isomorphisms only. Reasoning in terms of Mealy automata, the $Circ$ construction does not go beyond the observation of isomorphism between automata, so that it carries very little semantical information. Our Sim construction, on the other hand, does have a significant semantical contents, which will be pointed out in Sections 4 and 5.

Another novelty of our Sim construction compared to the $Circ$ construction is that it captures the stepwise behavior of monoidal automata and introduces a multi-step simulation between them in the way it was originally done for synchronous systems in [18]. The result is a monoidal 2-category, which leads to a category with circular feedback in a natural way.

3 Monoidal categories with trace and feedback

Recall from [19] that a *strict monoidal category* is a category M equipped with an associative bifunctor, called tensor, which has a unit element. In the spirit of the author's earlier works on the subject, composition (left-to-right) and tensor will be denoted by $\cdot$ and $+$, respectively, and I will stand for the tensor unit object. A *symmetry* in M is a family of isomorphisms $\pi_{A,B} : A + B \to B + A$, natural in A and B, such that

$$\pi_{A,B} \cdot \pi_{B,A} = 1_{A+B} \quad \text{and} \quad (\pi_{A,B} + 1_C) \cdot (1_B + \pi_{A,C}) = \pi_{A,B+C}.$$

The notation $\pi_{A,B}$ for symmetries is adopted from [17]. Morphisms built up solely from symmetries using composition and tensor in a regular fashion are called *permutations*.

We say that a monoidal category M is single-sorted if the set of its objects is generated by a single object 1. Such structures are also known as magmoids [3]. In the sequel, by monoidal category we shall always mean a strict symmetric monoidal category. This will simplify the presentation of the Sim construction a great deal, although the construction itself works for non-strict symmetric monoidal categories

as well with obvious modifications. In particular, our example category $(\boldsymbol{Set}, \times)$ is not strict, but it is perhaps the best suited to help the reader's understanding of simulation equivalence.

The following definition of traced monoidal categories originates from [15]. Trace in a category M will be introduced below as an operation $M(U + A, U + B) \to M(A, B)$, rather than $M(A + U, B + U) \to M(A, B)$ as it appears in [15], to be in accordance with the author's earlier works. In addition, the fact that M is strict and symmetric will also be taken into account to simplify the definition.

Definition 3.1 A *trace* for a monoidal category M is a natural family of functions

$$Tr^U_{A,B} : M(U + A, U + B) \to M(A, B)$$

satisfying the following three axioms:
vanishing:

$$Tr^I_{A,B}(f) = f \ , \quad Tr^{U+V}_{A,B}(g) = Tr^V_{A,B}(Tr^U_{V+A,V+B}(g));$$

superposing:

$$Tr^U_{A,B}(f) + g = Tr^U_{A+C,B+D}(f + g), \text{ where } g : C \to D;$$

yanking:

$$Tr^U_{U,U}(\pi_{U,U}) = 1_U.$$

Naturality of trace is meant in all three "variables" A, B, U. While naturality in A and B is fairly obvious, we wish to spell out the meaning of naturality in U, because the resulting axiom is our main concern in this paper.
sliding:

$$Tr^U_{A,B}((g + 1_A) \cdot f) = Tr^V_{A,B}(f \cdot (g + 1_B)), \text{ where } f : V + A \to U + B,\ g : U \to V.$$

The objects A, B in $Tr^U_{A,B}$ will usually be clear from the context, in which case they will be omitted from the notation.

According to [17], a *category with feedback* is a monoidal category M equipped with a feedback operation $\uparrow^U_{A,B}$: $M(U + A, U + B) \to M(A, B)$, which satisfies all the requirements for $Tr^U_{A,B}$ in Definition 3.1, except for the yanking axiom. Furthermore, naturality of $\uparrow^U_{A,B}$ in U is only required in a weaker form, when the morphism $g : U \to V$ in the sliding axiom is an isomorphism. In the context of categories with feedback we shall refer to sliding as the axiom of *circular feedback*, by the name it appears first in [21] regarding the single-sorted case. A *category with circular feedback* is then a category with feedback which satisfies the circular feedback axiom.

Let M be a monoidal category, fixed for the rest of the paper. An *M-automaton* $A \to B$ is a pair (U, α), where U is an object and $\alpha : U + A \to U + B$ is a morphism in M. Reflecting a digital circuit interpretation, the object U is called the state component, while α is the *combinational logic* of (U, α). The collection of M-automata can be given the structure of a category Aut_M equipped with a tensor as follows. Objects, and tensor of them are as in M. Furthermore:

$$1_A = (I, 1_A);$$

$$(U, \alpha) \cdot (V, \beta) = (U + V, (\pi_{U,V} + 1_A) \cdot (1_V + \alpha) \cdot (\pi_{V,U} + 1_B) \cdot (1_U + \beta));$$

$$(U,\alpha)+(V,\beta)=(U+V,(1_U+\pi_{V,A}+1_C)\cdot(\alpha+\beta)\cdot(1_U+\pi_{B,V}+1_D)).$$

See Fig. 1. The category M is embedded into Aut_M by the functor $A\mapsto A$, $\alpha\mapsto(I,\alpha)$. Feedback is defined in Aut_M as follows:

$$\uparrow^V_{A,B}(U,\alpha)=(U+V,\alpha),\quad \text{where } \alpha: U+V+A\to U+V+B.$$

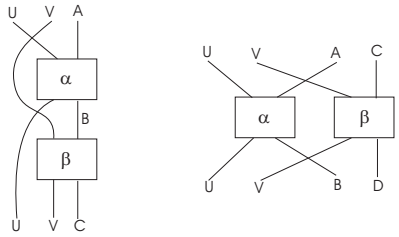

Figure 1: Composition and tensor in Aut_M

Notice that tensor is not a bifunctor in Aut_M, therefore this category is not monoidal. Thus, in a strict sense, Aut_M – with our $\uparrow$ in it – does not qualify as a category with feedback. Of course, this does not mean that feedback is ill-defined in Aut_M, rather, the name "category with feedback" is ambiguously chosen.

Two M-automata $(U,\alpha),(V,\beta):A\to B$ are said to be *isomorphic* if there exists an isomorphism $\gamma:U\to V$ in M such that $(\gamma+1_A)\cdot\beta=\alpha\cdot(\gamma+1_B)$. Isomorphism classes of M-automata are called circuits in [17], and $Circ(M)$ is the quotient of Aut_M by this isomorphism. As it was proved in [17], tensor (of isomorphism classes of M-automata) already makes $Circ(M)$ a monoidal category with the symmetry adopted from M. Moreover, $Circ(M)$ is a category with feedback.

4 The *Sim* construction

In this section we present a construction which, from a given monoidal category M, produces a category $Sim(M)$ with circular feedback. In order to make the construction feasible, we need to make the following assumption on the nature of composition in M.

Virtual trace:

For $\alpha_i: A\to U+C_i$, $\beta_i: U+C_i\to B$, $i=1,2$:

$(1_U+\alpha_1)\cdot(\pi_{U,U}+1_{C_1})\cdot(1_U+\beta_1)=(1_U+\alpha_2)\cdot(\pi_{U,U}+1_{C_2})\cdot(1_U+\beta_2)$

implies $\alpha_1\cdot\beta_1=\alpha_2\cdot\beta_2$. See Fig. 2.

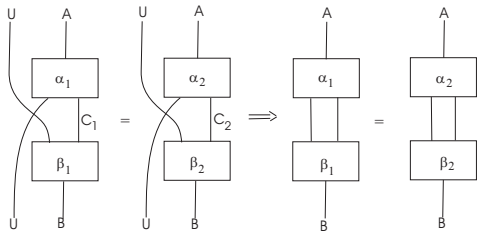

Figure 2: The virtual trace implication

Intuitively, the virtual trace implication enables the use of an imaginary trace operation when the result is expressible in terms of composition. Indeed,

$$Tr^U((1_U+\alpha_i)\cdot(\pi_{U,U}+1_{C_i})\cdot(1_U+\beta_i))=\alpha_i\cdot\beta_i$$

is trivially satisfied in all traced monoidal categories. Therefore the virtual trace implication simply ensures that this imaginary partial trace operation is well-defined. Notice that virtual trace is a necessary condition for any monoidal category to have an extension to a traced monoidal category.

The virtual trace implication is quite natural and holds in surprisingly many monoidal categories. For example, if tensor is product or coproduct in M, then this implication is almost straightforward. It also holds in all scheme algebras [4, 5]. In general, it holds true whenever the morphisms of M express a kind of data-flow semantics, either explicitly or implicitly. From this point on we shall assume that the virtual trace implication holds in our category M.

For objects A, B in M, a *transition chain* (chain, for short) $A \to B$ is a non-empty sequence

$$s = ((s_i, X_i, X_{i+1})|\ 0 \leq i < n),\ (n \geq 1), \quad \text{where} \quad s_i : X_i + A \to X_{i+1} + B.$$

Define the *cascade product* of s recursively as follows. (See Fig. 3.)

$cas_0(s) = s_0$;
$cas_i(s) = (cas_{i-1}(s) + 1_A) \cdot (1_{X_i} + \pi_{B,A}) \cdot (s_i + 1_{iB})$;
$cas(s) = cas_{n-1}(s) \cdot (1_{X_n} + \pi) : X_0 + nA \to X_n + nB$,

where for any natural number m, mB stands for the object $B + \ldots + B$, and $\pi : nB \to nB$ is the permutation that reverses the sequence of n B-blocks.

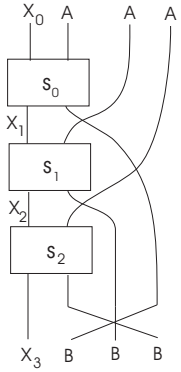

Figure 3: Cascade product of a chain with 3 links

Chains $s = ((s_i, X_i, X_{i+1})|\ 0 \leq i < n)$ and $t = ((t_j, Y_j, Y_{j+1})|\ 0 \leq j < m)$ $A \to B$ are said to be *linkable* if $X_n = Y_0$. Clearly, in this case, the concatenation $s \parallel t$ of the two chains is also a chain $A \to B$. With a slight ambiguity, if the sequence s or t in $s \parallel t$ has a single item, that item will be identified with the sequence itself.

Definition 4.1 An *n-step simulation* $(n \geq 1)$ $(U, \alpha) \to (V, \beta)$ in $Aut_M(A, B)$ is a chain $s = ((s_i, X_i, X_{i+1})|\ 0 \leq i < n)$ $A \to B$ such that:

(i) $X_0 = U$ and $X_n = V$;

(ii) $cas(s \parallel \beta) = cas(\alpha \parallel s)$.

An *immediate (0-step)* simulation $(U, \alpha) \to (V, \beta)$ is just a morphism $\delta : U \to V$ such that $(\delta + 1_A) \cdot \beta = \alpha \cdot (\delta + 1_B)$.

Lemma 4.2 *Let* $s = ((s_i, X_i, X_{i+1})|\ 0 \leq i < n)$ *and* $t = ((t_j, Y_j, Y_{j+1})|\ 0 \leq j < m)$ *be linkable chains* $A \to B$ *with* $V = X_n = Y_0$. *Then*

$$cas(s \parallel t) \cdot \pi_{nB,mB} = (cas(s) + 1_{mA}) \cdot (1_V + \pi_{nB,mA}) \cdot (cas(t) + 1_{nB}).$$

Corollary 4.3 *If $s : (U, \alpha) \to (V, \beta)$ and $t : (V, \beta) \to (W, \gamma)$ are non-immediate simulations in $Aut_M(A, B)$, then $s \parallel t$ is a simulation $(U, \alpha) \to (W, \gamma)$.*

Corollary 4.4 *If $s : (U, \alpha) \to (V, \beta)$ is a non-immediate simulation, then so is $s \parallel \beta$.*

Both Corollaries 4.3 and 4.4 can be generalized to all simulations by adopting the following natural rules regarding the operation $\parallel$. Let δ and δ' be immediate simulations and s be a non-immediate one between appropriate M-automata.

(i) $\delta \parallel s$ and $s \parallel \delta$ are formed by melting δ into the first link (respectively, last link) of s using composition in M, i.e., by replacing $s_0 : V + A \to X_1 + B$ with $(\delta + 1_A) \cdot s_0$ (respectively, $s_{n-1} : X_{n-1} + A \to V + B$ with $s_{n-1} \cdot (\delta + 1_B)$).

(ii) $\delta \parallel \delta' = \delta \cdot \delta'$.

Concerning cascade products, define $cas(\delta) = \delta$.

Proposition 4.5 *Simulations as morphisms define a category on $Aut_M(A, B)$ as objects.*

Proof. Identities in $Aut_M(A, B)$ are the identity morphisms 1_U in M as immediate simulations. Composition of simulations is $\parallel$. $\square$

Intuitively, Corollary 4.4 says that simulations s and $s \parallel \beta$ are essentially the same between the automata (U, α) and (V, β). For the very same reason, s and $\alpha \parallel s$ are the same, too. The only difference between s and $s \parallel \beta$ is that $s \parallel \beta$ "realizes" the simulation one step later, even though it has already been established by s. We shall say that s and $s \parallel \beta$ are indistinguishable. In order to put this relation as a proper equivalence between simulations, we introduce the following notation. For an n-step simulation $s : (U, \alpha) \to (V, \beta)$ and integer $m \geq n$, $\phi_{n,m}(s)$ is the simulation $s \parallel \beta^{m-n}$ (i.e., $s \parallel (\beta, \ldots, \beta)$).

Definition 4.6 Simulations $s, s' : (U, \alpha) \to (V, \beta)$, where s and s' are n-step and n'-step, respectively, are *indistinguishable* if there exists $l \geq max(n, n')$ such that $cas(\phi_{n,l}(s)) = cas(\phi_{n',l}(s'))$.

We shall use the symbol $\equiv$ to denote indistinguishability of simulations.

Proposition 4.7 *If $s \equiv s' : (U, \alpha) \to (V, \beta)$ and $t \equiv t' : (V, \beta) \to (W, \gamma)$, then $s \parallel t \equiv s' \parallel t' : (U, \alpha) \to (W, \gamma)$.*

By virtue of Proposition 4.7 we redefine our vertical categories $Aut_M(A, B)$, so that its morphisms be groups of indistinguishable simulations between automata. For simplicity, however, we shall keep working with representatives rather than equivalence groups of simulations.

We now turn to defining a horizontal composition on simulations of M-automata. Let $s = ((s_i, X_i, X_{i+1}) |\ 0 \leq i < n)$ and $t = ((t_i, Y_i, Y_{i+1}) |\ 0 \leq i < n)$ be chains $A \to B$ and $B \to C$, respectively. (See Fig. 4.) Define $s \bullet t = ((u_i, Z_i, Z_{i+1}) |\ 0 \leq i < n)$ to be the chain $A \to C$ in which $Z_i = X_i + Y_i$, and

$$u_i = (1_{X_i} + \pi_{Y_i,A}) \cdot (s_i + 1_{Y_i}) \cdot (1_{X_{i+1}} + \pi_{B,Y_i} \cdot t_i).$$

Equivalently, by the help of the (postfix) virtual trace operation $\Uparrow$ in M:

$$u_i = ((1_{X_i} + \pi_{Y_i,A} + 1_B) \cdot (s_i + t_i) \cdot (1_{X_{i+1}} + \pi_{B,Y_{i+1}+C})) \Uparrow^B .$$

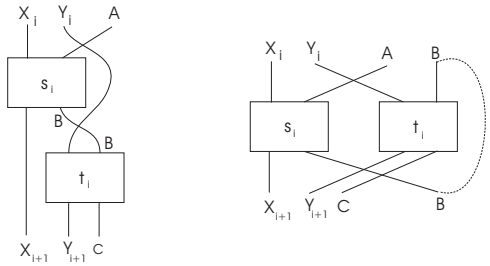

Figure 4: Horizontal composition of transition chains

Proposition 4.8 *For simulations $s : (U, \alpha) \to (V, \beta)$ and $t : (U', \alpha') \to (V', \beta')$ between M-automata $A \to B$ and $B \to C$, where s and t are chains as above with $X_0 = U$, $X_n = V$, $Y_0 = U'$, $Y_n = V'$, $s \bullet t : (U, \alpha) \cdot (U', \alpha') \to (V, \beta) \cdot (V', \beta')$ is a simulation. Moreover, for every $m \geq n$, $\phi_{n,m}(s) \bullet \phi_{n,m}(t) = \phi_{n,m}(s \bullet t)$.*

Proof. For simplicity, we only show a diagram that proves the first statement for 1-step simulations. The proof itself is highlighted by dotted lines in the diagram of Fig. 5. Notice that the virtual trace implication need not be used in this proof, although it clarifies the situation a great deal. □

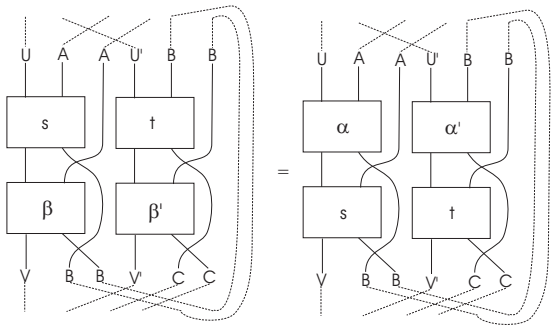

Figure 5: The proof of Proposition 4.8

The simulation $s \bullet t$ is called the *horizontal composite* of s and t. Using Corollary 4.4 it becomes possible to define the horizontal composite of two simulations even when they take different number of steps. Thus, by the second statement of Proposition 4.8, horizontal composition is properly established on groups of indistinguishable simulations. It is routine to check that $\bullet$ is associative, and the identities for this composition are determined by the morphism 1_I of M as an immediate simulation $(I, 1_A) \to (I, 1_A)$ for each object A. Furthermore, the horizontal composite of two vertical identities is itself a vertical identity, which is obvious.

Theorem 4.9 *Groups of indistinguishable simulations as 2-cells extend Aut_M to a 2-category.*

Proof. All components of this 2-category have been checked earlier, hence we need only show the interchange law

$$(s \parallel s') \bullet (t \parallel t') \equiv (s \bullet t) \parallel (s' \bullet t')$$

for all appropriate simulations s, t, s', t'. We leave this as an easy exercise. □

The next step in our construction is to show that in Aut_M, simulations are compatible with tensor and feedback. Let $s = ((s_i, X_i, X_{i+1}) |\ 0 \leq i < n)$ and

$t = ((t_i, Y_i, Y_{i+1})|\ 0 \leq i < n)$ be chains $A_1 \to B_1$ and $A_2 \to B_2$, respectively, and define $s \Box t = ((u_i, Z_i, Z_{i+1})|\ 0 \leq i < n)$ to be the chain $A_1 + A_2 \to B_1 + B_2$ in which $Z_i = X_i + Y_i$ and

$$u_i = (1_{X_i} + \pi_{Y_i,A_1} + 1_{A_2}) \cdot (s_i + t_i) \cdot (1_{X_{i+1}} + \pi_{B_1,Y_{i+1}} + 1_{B_2}).$$

Proposition 4.10 *For simulations $s : (U_1, \alpha_1) \to (V_1, \beta_1)$ and $t : (U_2, \alpha_2) \to (V_2, \beta_2)$ between M-automata $A_1 \to B_1$ and $A_2 \to B_2$, where s and t are as above with $X_0 = U_1$, $X_n = V_1$, $Y_0 = U_2$, and $Y_n = V_2$, $s \Box t$ is a simulation. For every $m \geq n$, $\phi_{n,m}(s) \Box \phi_{n,m}(t) = \phi_{n,m}(s \Box t)$. Moreover, the interchange law $(s \parallel s') \Box (t \parallel t') \equiv (s \Box t) \parallel (s' \Box t')$ holds for all appropriate simulations s, t, s', t'.*

Now let $s = ((s_i, X_i, X_{i+1})|\ 0 \leq i < n)$ be a chain $U + A \to U + B$, and define $\uparrow^U s = ((s_i, X_i + U, X_{i+1} + U)|\ 0 \leq i < n)$ as a chain $A \to B$.

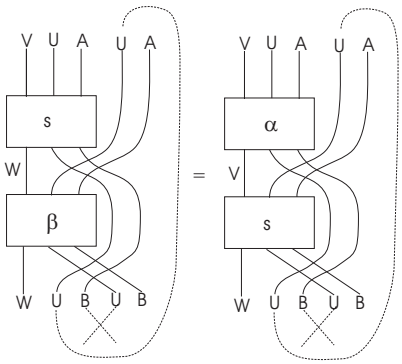

Figure 6: The proof of Proposition 4.11

Proposition 4.11 *If $s : (V, \alpha) \to (W, \beta)$ is a simulation between M-automata, where s is a chain $U + A \to U + B$ as above with $X_0 = V$ and $X_n = W$, then $\uparrow^U s : (V + U, \alpha) \to (W + U, \beta)$ is a simulation. Moreover, $s \equiv s'$ implies $\uparrow^U s \equiv \uparrow^U s'$, and the interchange law $\uparrow^U (s \parallel t) = (\uparrow^U s) \parallel (\uparrow^U t)$ holds for all appropriate simulations s, t.*

Proof. Again, we restrict the proof to the case $n = 1$, and provide a diagram for justification. The dotted feedback line in the digram of Fig. 6 indicates the main point of the proof. Observe that the virtual trace implication does play a crucial role in this argument. Indeed, the U-to-U cross-connection in the diagram, although expressible in terms of composition in M, cannot be made legal without this implication, for another V-to-V (or W-to-W) connection already exists between the same two boxes, which connection has been established by composition. □

Definition 4.12 *Automata (U, α) and (V, β) in $Aut_M(A, B)$ are simulation equivalent, in notation $(U, \alpha) \sim (V, \beta)$, if they are isomorphic according to the vertical structure of the 2-category Aut_M. The category of simulation equivalent M-automata is defined as $Sim(M) = Aut_M / \sim$.*

Example. Let $A = \{0, 1\}$, and consider the ***Set***-automaton $\nabla_A = \uparrow^A \pi_{A,A}$ together with one of its variants over the set of states $A' = \{0, 0', 1\}$ in Fig. 7. As we shall see in Section 5, these two automata are simulation equivalent. On the other

hand, the trivial one-state automaton 1_I is not simulation equivalent with the two-state automaton the transition diagram of which consists of two identical copies of 1_I. These automata can simulate each other, but Definition 4.12 would require an isomorphism between I and a two-element set in ***Set***.

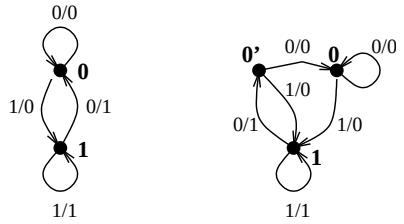

Figure 7: The automaton $\nabla_{\{0,1\}}$ and its variant

Theorem 4.13 *$Sim(M)$ is a category with circular feedback.*

Proof. By the interchange laws (Theorem 4.9 and Propositions 4.10, 4.11) simulation equivalence of M-automata is compatible with composition, tensor, and feedback. Also, $Sim(M)$ is a quotient of $Circ(M)$. Indeed, what makes two M-automata equivalent as circuits in $Circ(M)$ is, in our language, an immediate simulation that is also an isomorphism in M. Thus, $Sim(M)$ is a category with feedback, so $\sim$ can also be considered as a congruence in $Circ(M)$. As to the circular feedback axiom, let $f = (W_f, \alpha) : V+A \to U+B$ and $g = (W_g, \beta) : U \to V$ be M-automata. We need to show that the morphisms $\uparrow^V (f \cdot (g + 1_B))$ and $\uparrow^U ((g + 1_A) \cdot f)$ are simulation equivalent in $Aut_M(A, B)$. An immediate simulation $s : \uparrow^U ((g + 1_A) \cdot f) \to \uparrow^V (f \cdot (g + 1_B))$ is easy to find:

$$s = (\pi_{W_g,W_f} + 1_U) \cdot (1_{W_f} + \beta).$$

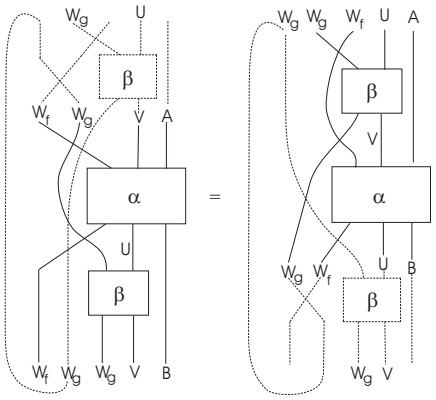

Figure 8: Circularity of feedback

For a justification, see the diagram of Fig. 8. A similar diagram shows that

$$t = (\pi_{W_f,W_g} + 1_V + 1_A) \cdot (1_{W_g} + \alpha)$$

is a one-step simulation $\uparrow^V (f \cdot (g + 1_B)) \to \uparrow^U ((g + 1_A) \cdot f)$. A simple computation shows that $s \bullet t$ is exactly the combinational logic of $\uparrow^U ((g + 1_A) \cdot f)$ and $t \bullet s$ is that of $\uparrow^V (f \cdot (g + 1_B))$. The result now follows from Corollary 4.4 by the definition of indistinguishability. $\square$

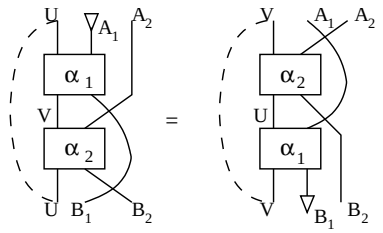

Figure 9: The retiming identity

5 Universality of the *Sim* construction

We start out by a generalization of the circular feedback axiom. This generalization, called the retiming identity, was first considered in [11] for synchronous systems. The axiom, shown in Fig. 9, is the equation appearing in Lemma 5.1 below. The notation $\nabla_A = \uparrow^A \pi_{A,A}$ is adopted from [5], and it identifies a *register* of sort A. For convenience, we also generalize our cascade product *cas* so that it applies for sequences of morphisms $s_i : X_i + A_i \to X_{i+1} + B_i$ without the restriction that $A_i = A$ and $B_i = B$ for fixed objects A, B.

Lemma 5.1 *For any morphisms $\alpha_1 : U + A_1 \to V + B_1$ and $\alpha_2 : V + A_2 \to U + B_2$ in a category with circular feedback,*

$$(\nabla_{A_1} + 1_{A_2}) \cdot \uparrow^U cas(\alpha_1 \parallel \alpha_2) = \pi_{A_1,A_2} \cdot (\uparrow^V cas(\alpha_2 \parallel \alpha_1)) \cdot \pi_{B_2,B_1} \cdot (\nabla_{B_1} + 1_{B_2}).$$

Proof. We present a proof covering the special case $U = V = I$ and $\alpha_2 = 1_I$ only, which already features the trick of pulling a register through a box. The reader can follow the steps of this simple proof on Fig. 10. □

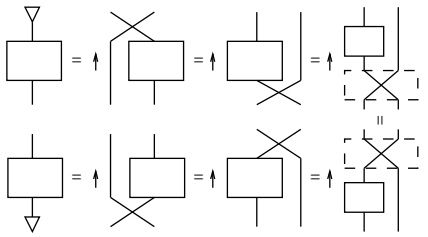

Figure 10: The proof of Lemma 5.1.

The point of retiming in general is indeed shifting a layer of registers from one side of a "box" to the other, as introduced originally in [18] for synchronous systems. In Fig. 9, register ∇_{A_1} is shifted explicitly from the input side of box α_1 to the output side, whereas the register ∇_U is part of the U-feedback connection coming from box α_2 and is turned into a register ∇_V on the V-feedback connection from α_1 to α_2. Lemma 5.1 simply says that the retiming identity is a consequence of the circular feedback identity in monoidal categories with feedback. For this reason, the congruence relation induced by the circular feedback identity in any monoidal category with feedback will be called *retiming equivalence*; denoted $\sim_r$.

Now we introduce two new axioms, which are aimed at capturing the sequential behavior of the feedback operation (with delay). Let $\alpha : U + A \to U + B$ and $\beta : V + A \to V + B$ be morphisms in M, and for every $n \geq 1$ construct the chains $\alpha^n = (\alpha, \ldots, \alpha)$ and $\beta^n = (\beta, \ldots, \beta)$ $A \to B$. The *sequential feedback* axioms are the following two implications in $Circ(M)$.

Speed-up:
If $\uparrow^U cas(\alpha^n) \sim_r \uparrow^V cas(\beta^n)$ for every sufficiently large n, then $\uparrow^U \alpha \sim_r \uparrow^V \beta$.
Delay:
If $\nabla_A \cdot \uparrow^U \alpha \sim_r \nabla_A \cdot \uparrow^V \beta$, then $\uparrow^U \alpha \sim_r \uparrow^V \beta$.

The rationale for the speed-up axiom is the following. Let $f = (U, \alpha)$ and $g = (V, \beta)$ be the M-automata associated with α and β, respectively. Then $f^n = (U, cas(\alpha^n))$ is essentially the same as f, except that it performs n steps of f in one clock cycle. In other words, f^n is the n-speed-up of f. Hence, if f^n and g^n are retiming equivalent for all sufficiently large n, then it is natural to expect that f and g be retiming equivalent as well.

Concerning the delay axiom, our philosophy is as follows. We know that the input-output channels are not retimable in synchronous systems (see [7, 8]). Therefore it does not matter if we put an extra register (morphism ∇_A) on the interconnections starting from the input channels, retimability of f to g should not be affected.

Theorem 5.2 *If $Circ(M)$ satisfies the sequential feedback axioms, then $Sim(M)$ is the free monoidal category with circular feedback generated by M.*

Proof. We show that $\sim \subseteq \sim_r$ holds in $Circ(M)$, provided that feedback is sequential in that category. This statement, together with Theorem 4.13, implies that $\sim = \sim_r$. Given that $Circ(M)$ is freely generated by M as a category with feedback, $Sim(M) = Circ(M)/\sim$ will also be free as a category with circular feedback.

Let $f = (U, \alpha)$ and $g = (V, \beta)$ be M-automata $A \to B$ such that $f \sim g$. By definition, there exist simulations $s : f \to g$ and $t : g \to f$ such that $s \cdot t \equiv 1_f$ and $t \cdot s \equiv 1_g$. Spelling this out, we have:

(i) $cas(s \parallel \beta) = cas(\alpha \parallel s)$ and $cas(t \parallel \alpha) = cas(\beta \parallel t)$;
(ii) $cas(\alpha^k \parallel s \parallel t) = cas(\alpha^{k+n+m})$ and $cas(t \parallel s \parallel \beta^k) = cas(\beta^{m+n+k})$,

where n and m denote the length of s and t, respectively, and k is any sufficiently large integer. Using Lemma 5.1, a short computation shows that

$(\nabla_{mA} + 1_{(n+k)A}) \uparrow^V cas(\beta^{m+n+k}) = \pi_1 \uparrow^U cas(\alpha^{k+n+m}) \cdot \pi_2 \cdot (\nabla_{mB} + 1_{(n+k)B})$,

where $\pi_1 = \pi_{mA,(n+k)A}$ and $\pi_2 = \pi_{(n+k)B,mB}$. See Fig. 11 for the case $n = m = k = 1$. On the other hand, directly by Lemma 5.1,

$(\nabla_{mA} + 1_{(n+k)A}) \uparrow^U cas(\alpha^{m+n+k}) \sim_r \pi_1 \uparrow^U cas(\alpha^{n+k+m}) \cdot \pi_2 \cdot (\nabla_{mB} + 1_{(n+k)B})$.

Thus,

$(\nabla_{mA} + 1_{(n+k)A}) \uparrow^V cas(\beta^{m+n+k}) \sim_r (\nabla_{mA} + 1_{(n+k)A}) \uparrow^U cas(\alpha^{m+n+k})$, so that

$\nabla_{(m+n+k)A} \uparrow^V cas(\beta^{m+n+k}) \sim_r \nabla_{(m+n+k)A} \uparrow^U cas(\alpha^{m+n+k})$.

The statement now follows from the sequential feedback axioms. □

Returning to our Example, we show that the two automata appearing in Fig. 7 are retiming equivalent, and therefore they are simulation equivalent as well. Let $\delta : A \times A \to A \times A$ be the transition function (with the output component) of automaton ∇_A on the left, and $\delta' : A' \times A \to A' \times A$ be that of its variant on the right. (Remember that $A = \{0, 1\}$ and $A' = \{0, 0', 1\}$.) Define $\alpha : A \times A \to A' \times A$ and $\kappa : A' \to A$ by $\alpha(i, j) = \delta'(i, j)$, $\kappa(i) = i$, and $\kappa(0') = 0$ for every $i, j \in \{0, 1\}$. Then, clearly, $\delta = \alpha \cdot (\kappa \times 1_A)$ and $\delta' = (\kappa \times 1_A) \cdot \alpha$.

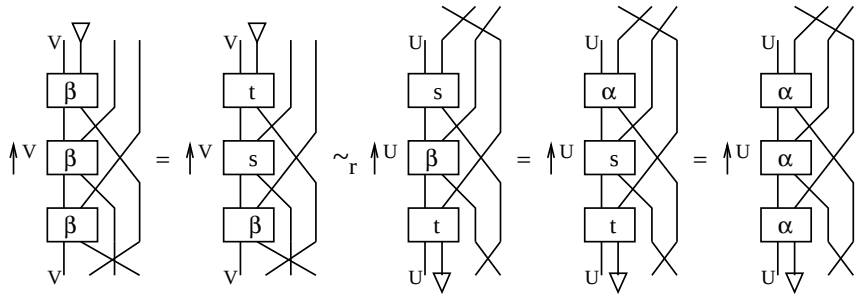

Figure 11: The proof of Theorem 5.2

In general, a state s of a finite state Mealy automaton f is called *run-out* if no sufficiently long input can take f to s; otherwise s is called *permanent*. Denote by $P(f)$ the restriction of f to its permanent states, and let δ be the transition function of $P(f)$. Two states s and s' of $P(f)$ are said to be *simulation equivalent* (in notation $s \sim s'$) if s and s' are bisimulation equivalent in the usual sense, and, furthermore, for every sufficiently long input string w, $\delta(s, w) = \delta(s', w)$. (See again the transition diagram on the right-hand side of Fig. 7 to verify that states 0 and $0'$ are simulation equivalent.) The equivalence $\sim$ gives rise to a minimal automaton $P(f)/\sim$, in terms of which retiming equivalence of finite state Mealy automata can be characterized in the following way.

Theorem 5.3 *Two finite state Mealy automata f and g are retiming equivalent iff the automata $P(f)/\sim$ and $P(g)/\sim$ are isomorphic.*

The proof of Theorem 5.3 will be presented in a forthcoming paper.

Using the above characterization, one can prove that feedback is sequential in $Circ(\boldsymbol{Set}_f)$, where $\boldsymbol{Set}_f$ is the restriction of $\boldsymbol{Set}$ to finite sets and functions. Thus, by Theorem 5.2, we have the following important result.

Corollary 5.4 *$Sim(\boldsymbol{Set}_f)$ is freely generated by $\boldsymbol{Set}_f$ as a category with circular feedback.*

At present we do not know if Corollary 5.4 holds for infinite state Mealy automata or not. Nor do we know if, in general, the sequential feedback axioms are necessary for $\sim_r$ to coincide with $\sim$, or if any condition at all is needed to ensure this coincidence. We conjecture, however, that $\sim_r = \sim$ holds in $Circ(M)$ for all monoidal categories M in which tensor is product or coproduct.

References

[1] S. Abramsky, Retracing some paths in process algebras, CONCUR'96, U. Montanari and V. Sassone, eds., Springer-Verlag, *Lecture Notes in Comput. Sci.* **1119** (1996) 1–17.

[2] S. Abramsky, Abstract scalars, loops, and free traced and strongly compact closed categories, CALCO 2005, J. Fiadeiro, N. Harman, M. Roggenbach, and J. Rutten, eds., Springer-Verlag, *Lecture Notes in Comp. Sci.* **3629** (2005) 1–29.

[3] A. Arnold, M. Dauchet, Théorie des magmoïdes, *RAIRO Inform. Théor.* **12** (1978), 235–257 and **13** (1979), 135–154.

[4] M. Bartha, A finite axiomatization of flowchart schemes, *Acta Cybernet.* **2** (1987), 203–217.

[5] M. Bartha, An equational axiomatization of systolic systems, *Theoret. Comput. Sci.* **55** (1987), 265–289.

[6] M. Bartha, An algebraic model of synchronous systems, *Information and Computation* **97** (1992), 97–131.

[7] M. Bartha, B. Čirovič, On some equivalence notions of synchronous systems, Proceedings, 11th International Conference on Automata and Formal Languages, Dogogókő, Hungary (Z. Ésik and Z. Fülöp, eds.), 2005.

[8] M. Bartha Strong retiming equivalence of synchronous systems, Proceedings of the International Conference CIAA'05, Sophia Antipolis, France, *Lecture Notes in Computer Science* Vol. 3845/2006, pp. 66–77.

[9] S. L. Bloom, Z. Ésik, *Iteration Theories: The Equational Logic of Iterative Processes*, Springer Verlag, Berlin, 1993.

[10] S. L. Bloom, Z. Ésik, Axiomatizing schemes and their behaviors, *J. Comput. System Sci.* **31** (1985), 375–393.

[11] B. Čirovič, Equivalence relations of synchronous systems, Ph.D. Dissertation, Memorial University of Newfoundland, 2000.

[12] C. C. Elgot, Monadic computations and iterative algebraic theories, in: H.E. Rose, ed.,*Logic Colloquium 73* (North-Holland, Amsterdam, 1975) 175–230.

[13] C. C. Elgot, *Selected Papers* (S. L. Bloom, ed.), Springer Verlag, New York, 1982.

[14] M. Hasegawa, *Models of Sharing Graphs: A categorical semantics of let and letrec,* Ph.D. Thesis, Edinburgh (1997), Springer (1999).

[15] A. Joyal, R. Street, and D. Verity, Traced monoidal categories, *Math. Proc. Camb. Phil. Soc.* **119** (1996), 447–468.

[16] P. Katis, N. Sabadini, and R. F. C. Walters, Bicategories of processes, *J. Pure Appl. Algebra* **115** (1997) 141–178.

[17] P. Katis, N. Sabadini, and R. F. C. Walters, Feedback, trace, and fixed-point semantics, *Theoret. Informatics Appl.* **36** (2002), 181–194.

[18] C. E. Leiserson, J. B. Saxe, Optimizing synchronous systems, *J. VLSI Comput. Systems* **1** (1983), 41–67.

[19] S. MacLane, *Categories for the Working Mathematician*, Springer Verlag, Berlin, 1971.

[20] A. Simpson and G. Plotkin, Complete axioms for categorical fixed-point operators, in *Proc. 15th LICS* (2000) 30–41.

[21] Gh. Ştefănescu, "Feedback Theories (A Calculus for Isomorphism Classes of Flowchart Schemes)," Research Report 24, National Institute for Scientific and Technical Creation, Bucharest, 1986.