EQUIVALENCE RELATIONS OF MEALY AUTOMATA

Miklós Bartha
Memorial University of Newfoundland, St. John's, NL, Canada
Email: bartha@mun.ca

Abstract
Two independent equivalence relations are considered for Mealy automata. Simulation equivalence, on the one hand, arises from a new multi-step simulation concept under the constraint that the process of switching from one automaton to the other is reversible. Retiming equivalence, on the other hand, is the congruence induced by the so called sliding axiom in the monoidal category of automata. It is shown that these two equivalence relations coincide, and a characterization is given for simulation/retiming equivalence in terms of transition diagrams.

1. Introduction

Several definitions of simulation can be found in the literature, all of which are centered around the algebraic concept of homomorphism between automata. Works devoting an extensive coverage to simulation and related issues include [2, 11, 13, 18], to name but a few major ones. In this paper we take a new look at simulations between two Mealy automata in the light of some recent results on monoidal categories with trace, feedback, and iteration. [1, 14, 15, 9, 8].

Let (U, α) and (V, β) be two Mealy automata with states U and V, respectively, and assume, for simplicity, that the input A and output B are common for these automata. Then $\alpha : U \times A \to U \times B$ and $\beta : V \times A \to V \times B$ are functions or relations, depending on whether the automata are deterministic or not. In the deterministic case, a homomorphism from (U, α) to (V, β) is a function $h : U \to V$ such that $\alpha \circ (h \times id_B) = (h \times id_A) \circ \beta$. The "monoidal" scheme of this equation is shown in the diagram of Fig. 1a. The diagram is to be interpreted in the monoidal category $(\boldsymbol{Set}, \times)$, that is, sets and functions with the tensor operation being cartesian product. Interpreting the diagram in any symmetric monoidal category M yields the general notion of simulation (machine morphism) between M-automata [11]. For example, if $M = (\boldsymbol{Rel}, \times)$, that is, sets and relations with the tensor being product of sets and componentwise product of relations, then we are looking at simulations between nondeterministic (relational) automata [11]. For further examples of monoidal automata, see [11].

Simulations, as described above, are irreversible in nature. Once the switch has been made from (U, α) to (V, β) using a simulation h, it is not possible to retrieve the original operation of (U, α) by switching back from (V, β) using another simulation, unless h is an isomorphism. It is only the input-output behavior of automata that is preserved by simulations. To be able to define simulations that are reversible without being trivial, we introduce a look-back feature in the

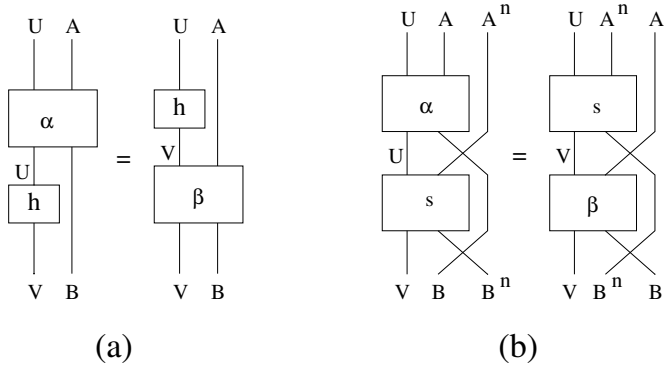

Figure 1: Homomorphism and simulation between Mealy automata

process of switching. By this we mean that the switch from (U, α) to (V, β) is carried out by a cross-transition function $s : U \times A^n \to V \times B^n$ in n steps. For the monoidal scheme of switching by s, see the diagram of Fig. 1b. Standing on this ground, one can already find simulations that are reversible without being direct isomorphisms as machine (homo-)morphisms. The process is as follows. Starting from (U, α), switch to (V, β) in n steps using simulation s. Then, after at least m steps, switch back to (U, α) using an m-step simulation t. After the second switch, run (U, α) for another fixed number of steps to allow full recovery. The operation of (U, α) should then be exactly the same as if the switching back and forth did not happen. Automata (U, α) and (V, β) are simulation equivalent if simulations s and t above exist, and the situation is symmetric in t and s.

The idea of simulation equivalence originates from [16], where a similar concept was introduced for synchronous systems. Synchronous systems are essentially structural Mealy automata [12], which come with a scheme-like syntactical description in terms of registers and functional elements, analogously to sequential circuits. Retiming a system means shifting its registers around within the graph of the system in a certain regular way. Retiming is a fundamental operation from the point of view of optimizing synchronous systems. If the number of registers in the system is sufficiently large, then they can be rearranged by the help of retiming to produce a systolic system, whereby at least one register can be found on the interconnection between any two functional elements. The obvious advantage of having a systolic system is that the length of the clock period for the system can be chosen minimum, i.e., as the maximal propagation delay of the functional elements. See [16] or [7] for more details. An important observation in [16] is that, whenever two systems are retiming equivalent, they can also simulate each other. The exact relationship between retiming and simulation, however, was not addressed in [16].

Synchronous systems have since been studied as single-sorted symmetric monoidal categories equipped with a feedback operation in a series of papers [3, 4, 5, 6, 7]. The main finding of [6] is that two synchronous systems can simulate each other under the free interpretation of their functional elements iff they are retiming equivalent. The present paper extends this result by showing that retiming equivalence coincides with simulation equivalence for Mealy automata in general, and provides a simple characterization of this equivalence in terms of transition diagrams.

On the basis of the synchronous system model, simulation equivalence was articulated in its present form in [8], with the underlying structure being an arbitrary symmetric monoidal category M. The structure of simulation equivalent M-automata, $Sim(M)$, has been constructed as a quotient of the category $Circ(M)$ [15] of isomorphism classes of M-automata in a suitable 2-categorical setting. We shall elaborate on this construction to some extent in Section 3 in the special case $M = (\boldsymbol{Set}, \times)$.

Retiming a synchronous system is analogous to the concept of sliding [14, 15] in monoidal categories with feedback. This fact was pointed out in [8, 10]. Furthermore, parallel to the results in [16, 6], retiming (sliding) equivalence always implies simulation equivalence, regardless of the underlying category M. It is therefore of a significant theoretical importance to see if simulation equivalence coincides with retiming equivalence for any particular category M. A general sufficient condition for this coincidence was worked out in [8], and the main result of the present paper could in fact be interpreted as a test for that condition in the category $(\boldsymbol{Set}, \times)$. To keep the paper self-contained, however, we shall not rely on the technically rather complicated Sim construction in [8]. We shall not even assume much expertise in category theory, other than familiarity with the basic definitions and understanding some simple monoidal diagrams like the ones in Fig. 1, which should be straightforward. The reader might, however, learn a lot about monoidal categories through the interpretation of these simple diagrams.

2. Simulation equivalence of Mealy automata

For the rest of the paper we shall assume that our Mealy automata are finite and deterministic. Even though the main result, namely that simulation equivalence coincides with retiming equivalence, holds true for all Mealy automata, the proof becomes much more complicated when the automata are infinite and/or nondeterministic. Therefore these cases will be dealt with in a separate paper.

We shall use the notation $(U, \alpha) : A \to B$ to identify an automaton with states U, input A, output B, and transition function (including the output component) $\alpha : U \times A \to U \times B$. Composition of functions, denoted by $\circ$, will follow the left-to-right application order in writing, that is, $(f \circ g)(x) = g(f(x))$. Identity on set A will be denoted by id_A, and $\pi_{A,B} : A \times B \to B \times A$ will stand for the symmetry [17] $(a, b) \mapsto (b, a)$.

Definition 2.1. For functions $s : X \times A \to Y \times C$ and $t : Y \times B \to Z \times D$, the *cascade product* of s and t is the function $cas_{X,Y,Z}(s, t) : X \times A \times B \to Z \times C \times D$ defined as:

$$cas_{X,Y,Z}(s, t) = (s \times id_B) \circ (id_Y \times \pi_{C,B}) \circ (t \times id_C) \circ (id_Z \times \pi_{D,C}).$$

See Fig. 2a. When X, Y and Z are clear from the context, the subscript X, Y, Z will be omitted from the notation. We shall also need the "twisted" version of *cas* defined by

$$sac(s, t) = (id_X \times \pi_{B,A}) \circ cas(s, t) \circ (id_Z \times \pi_{C,D}).$$

See Fig. 2b. It is easy to see that the binary operation of cascade product is associative. That is,

$$cas_{W,Y,Z}((cas_{W,X,Y}(r, s), t) = cas_{W,X,Z}(r, cas_{X,Y,Z}(s, t))$$

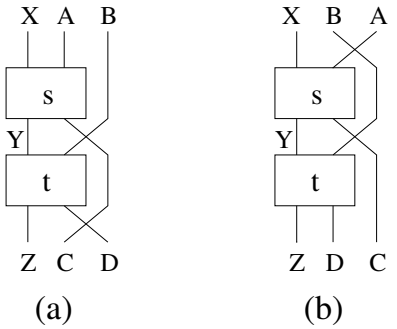

Figure 2: The operations *cas* and *sac*

for all appropriate functions r, s, t. See Fig. 3 for a monoidal proof. Therefore, for any automaton $(U, \alpha) : A \to B$ and integer $k \geq 1$, $\alpha^k = cas(\alpha, \ldots, \alpha) : U \times A^k \to U \times B^k$ is meaningful, and it identifies the k-step transition function of (U, α).

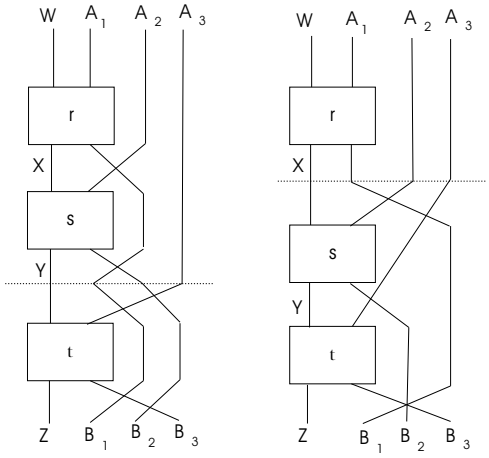

Figure 3: The associativity of *cas*

Definition 2.2. Let (U, α) and (V, β) be automata $A \to B$. A *simulation* from (U, α) to (V, β) is a function $s : U \times A^n \to V \times B^n$, $n \geq 0$, such that $cas_{U,U,V}(\alpha, s) = cas_{U,V,V}(s, \beta)$. If $n = 0$, then s is called *immediate*.

See again Fig. 1b. We shall write $s : (U, \alpha) \to (V, \beta)$ to underline the intuition that simulations are in fact 2-cells [17] in the category of automata. At this point the reader might wonder why the output is included in the cross-transition s by Definition 2.2. Indeed, since the definition implies $cas(\alpha^k, s) = cas(s, \beta^k)$ for any $k \geq 1$ (e.g. for $k = n$), the outputs of s must be the same as those of (U, α) in the first n steps. Thus, s is completely determined by its state transition component $U \times A^n \to V$. The reason for still keeping the output component in s is to be in line with [8, Definition 4.1], where $\times$ is just tensor, not necessarily product. This argument highlights a different philosophy, compared to [11], regarding the interpretation of $\times$ in the expression $U \times A \to U \times B$. While we treat both occurrences of $\times$ in this expression simply as tensor, Ehrig makes a distinction by separating the output function in the very definition of automata. The two approaches can be synchronized, so that they both become special cases

of machines in a general category by [18], but only if $\times$ is product in the underlying monoidal category. For the details of this argument, see [4]. Also notice that our simulation concept is quite different from weak simulation as defined in [2], whereby a move of the one automaton on input a is simulated by the move of the other on a string $f(a)$ determined by a fixed function $f : A \to A^*$.

An immediate consequence of Definition 2.2 is the fact that, for every $l \geq 1$, $cas(\alpha^l, s)$ is also a simulation $(U, \alpha) \to (V, \beta)$. Intuitively, the only difference between s and $cas(\alpha^l, s)$ is that the latter "realizes" the same simulation l steps later than s. We say that s and $cas(\alpha^l, s)$ are indistinguishable. In particular, the identity simulation $1_{(U,\alpha)} = id_U$ is indistinguishable from α^l for every $l \geq 0$ (assuming the natural definition $\alpha^0 = id_U$).

Definition 2.3. Simulations $s, s' : (U, \alpha) \to (V, \beta)$ are *indistinguishable*, in notation $s \equiv s'$, if there exist integers $k, l \geq 0$ such that $cas(\alpha^k, s) = cas(\alpha^l, s')$.

Simulations can be composed using the cascade product. It is easy to see that, if $s : (U, \alpha) \to (V, \beta)$ and $t : (V, \beta) \to (W, \gamma)$ are simulations between automata $A \to B$ (with A and B fixed), then $cas(s, t)$ is a simulation $(U, \alpha) \to (W, \gamma)$. Indeed:

$$cas(\alpha, cas(s, t)) = cas(cas(\alpha, s), t) = cas(cas(s, \beta), t) =$$

$$= cas(s, cas(\beta, t)) = cas(s, cas(t, \gamma)) = cas(cas(s, t), \gamma).$$

Composition of simulations is therefore associative. This composition, together with the identity simulations $1_{(U,\alpha)}$, defines the so called vertical category of Mealy automata. We shall specify the horizontal structure of automata as a monoidal category in Section 3.

Definition 2.4. Automata (U, α) and (V, β) are *simulation equivalent*, in notation $(U, \alpha) \sim (V, \beta)$, if there exist simulations $s : (U, \alpha) \to (V, \beta)$ and $t : (V, \beta) \to (U, \alpha)$ such that $cas(s, t) \equiv 1_{(U,\alpha)}$ and $cas(t, s) \equiv 1_{(V,\beta)}$.

Notice that the connection $cas(s, t) \equiv 1_{(U,\alpha)}$ also allows for a recovery period in the operation of the automaton (U, α) after the two subsequent switches s and t, as explained informally in the Introduction.

Example Let $A = \{0, 1\}$, and consider the unit delay automaton $\nabla_A = (A, \pi_{A,A}) : A \to A$ with one of its variants $3_A = (A', \alpha) : A \to A$ in Fig. 4, where $A' = \{0, 0', 1\}$. Since ∇_A is a homomorphic image of 3_A, there exists an immediate simulation $s : 3_A \to \nabla_A$ determined by $s(0) = s(0') = 0$ and $s(1) = 1$. Now let $t(0, i) = (i, 0)$ for $i = 0, 1$, $t(1, 0) = (0', 1)$ and $t(1, 1) = (1, 1)$. It is easy to check that t is a simulation $\nabla_A \to 3_A$ such that $cas(s, t) = \alpha$ and $cas(t, s) = \pi_{A,A}$. Thus, ∇_A and 3_A are simulation equivalent.

Heuristically, when switching from ∇_A to 3_A by simulation t, one looks back on the input. In state 0, if the last input symbol was 0, then switch to state 0, otherwise switch to $0'$. In state 1 always switch to state 1.

Example With $I = \{\emptyset\}$, consider the trivial one-state automaton $\mathbf{1}_I = (I, id_I) : I \to I$

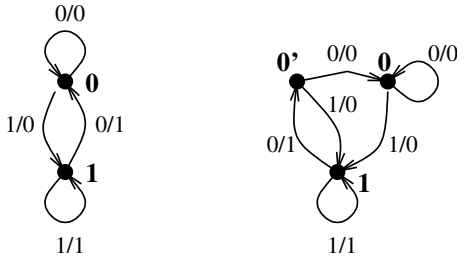

Figure 4: The automaton ∇_A and its variant 3_A

and the two-state automaton $2_I = (A, id_A) : I \to I$, where $A = \{0, 1\}$. Even though these two automata can trivially simulate each other, they are not simulation equivalent. Indeed, Definition 2.4 would require an isomorphism between I and A, which is impossible.

Notice that $\mathbf{1}_I$ and 2_I are not simulation equivalent as nondeterministic automata either, where the simulations s and t in Definition 2.2 would come as relations, rather than functions. This is rightfully so, because it is not possible to switch from 2_I to $\mathbf{1}_I$ and back in any number of steps without altering the original operation of 2_I in an irreversible way.

3. Retiming equivalence of Mealy automata

In order to introduce the concept of retiming, we make a short digression towards the general theory of monoidal categories with feedback, using Mealy automata as an illustrative example. The reader is referred to [15, 8] for the precise definitions.

The collection of Mealy automata can be given the structure of a category *Aut* equipped with a tensor operation $\otimes$ as follows. Objects are sets, and tensor of them is cartesian product of sets. Composition and tensor of morphisms in *Aut* is the well-known serial and parallel composition of automata [13]:

$$(U, \alpha) \cdot (V, \beta) = (U \times V, (\pi_{U,V} \times id_A) \circ (id_V \times \alpha) \circ (\pi_{V,U} \times id_B) \circ (id_U \times \beta));$$

$$(U, \alpha) \otimes (V, \beta) = (U \times V, (id_U \times \pi_{V,A} \times id_C) \circ (\alpha \times \beta) \circ (id_U \times \pi_{B,V} \times id_D)).$$

See Fig. 5. Identity on object A is the automaton $\mathbf{1}_A = (I, id_A)$. The category ***Set*** is embedded

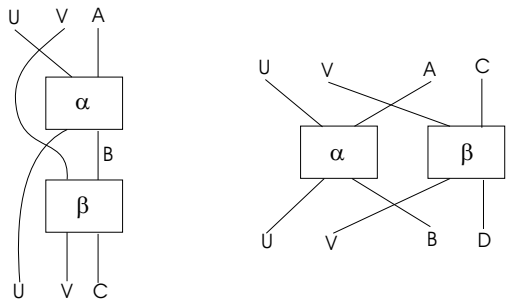

Figure 5: Composition and tensor in *Aut*

into *Aut* by the functor $A \mapsto A$, $\alpha \mapsto (I, \alpha)$. (Recall that $I = \{\emptyset\}$ is the unit object of the monoidal category $(\boldsymbol{Set}, \times)$). Feedback is defined in *Aut* as follows:

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

Isomorphism classes of automata are called circuits in [15], and $Circ$ is the quotient of Aut by isomorphism. As it was proved in [15], tensor (of isomorphism classes of automata) makes $Circ$ a symmetric monoidal category with the symmetry adopted from $\boldsymbol{Set}$. We shall denote isomorphism of automata by $\cong$.

As it was proved in [8], simulation equivalence is compatible with the operations composition, tensor, and feedback. Consequently, since $\cong \subseteq \sim$, the quotient of Aut by $\sim$ is in fact a quotient of $Circ$, so that it is a monoidal category (with feedback). This category is denoted by Sim. Building on the vertical structure of simulations described in Section 2, Sim can be enriched to a 2-category [17]. In order for this, one needs a suitable horizontal composition and tensor of simulations. Let $s : (U, \alpha) \to (V, \beta)$ be a simulation between automata $A \to B$ and $t : (U', \alpha') \to (V', \beta')$ be a simulation between automata $B \to C$, so that s and t take the same n number of steps. Define

$$s \bullet t = (id_U \times \pi_{U', A^n}) \circ (s \times id_{U'}) \circ (id_V \times \pi_{B^n, U'} \circ t).$$

See Fig. 6. According to [8, Proposition 3.8], $s \bullet t : (U, \alpha) \cdot (U', \alpha') \to (V, \beta) \cdot (V', \beta')$ is a simulation, which is called the *horizontal composite* of s and t. If s and t take different number of steps, then one must choose indistinguishable representations for them that take the same number of steps before composing these representations horizontally. It is easy to see that in this way horizontal composition is well-defined on classes of indistinguishable simulations, and it is associative. The identities for horizontal composition are determined by the trivial function id_I as an immediate simulation $\mathbf{1}_A \to \mathbf{1}_A$ for each set A. (Recall that $\mathbf{1}_A = (I, id_A)$.) Moreover, the interchange law

$$cas(s, s') \bullet cas(t, t') \equiv cas(s \bullet t, s' \bullet t')$$

holds for all appropriate simulations s, t, s', t'.

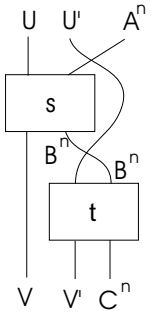

Figure 6: Horizontal composition of simulations

Tensor of simulations can be worked out in the same way, and it comes with an analogous interchange law with respect to the vertical composition *cas* of simulations. See [8] for the details. Thus, groups of indistinguishable simulations as 2-cells extend Sim to a monoidal 2-category.

The *sliding axiom* [14] in a monoidal category with feedback is the following identity:

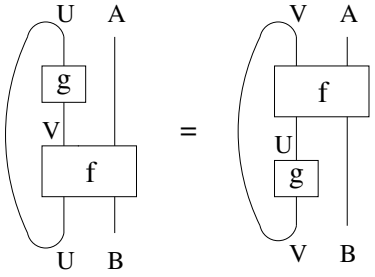

Figure 7: The sliding axiom

$\uparrow^U ((g \otimes \mathbf{1}_A) \cdot f) = \uparrow^V (f \cdot (g \otimes \mathbf{1}_B))$, where $f : V \otimes A \to U \otimes B$, $g : U \to V$.

See Fig. 7. We wish to emphasize that, unlike in our earlier monoidal diagrams, the boxes in Fig. 7 represent automata rather than just functions. The sliding axiom does not hold in $Circ$, but the equivalence that it stipulates is what we call *retiming equivalence*. More explicitly, retiming equivalence is the quotient of $Circ$ induced by the sliding axiom (identity) in the usual algebraic sense. To explain the terminology, imagine that there is a register (delay element) on the two round feedback lines in Fig. 7. In case the boxes are just functions, it is this assumption that renders a sequential behavior to the cycles in the graphs at hand, turning functions into real automata. These registers are de facto present in the graphs (schemes) of synchronous systems as weights on the interconnections between functional elements, eliminating the need to distinguish between "feedback" and "composition" lines. An elementary retiming step in a synchronous system is shifting a layer of registers from the input side of a functional element (box) to the output side, or vice versa. Shifting these registers is topologically the same as sliding the box from one side of the registers to the other. Therefore the phenomenon sliding is the exact replica of retiming. This is one of the major observations in [10], where the sliding axiom was first used to model retiming of synchronous systems. The maneuver of shifting registers through boxes is yet more conspicuous in Lemma 3.2 below.

Definition 3.1. Let (U, α) and (V, β) be automata $A \to B$, and assume that there are functions $\delta : V \times A \to U \times B$ and $\eta : U \to V$ such that $\alpha = cas_{U,V,U}(\eta, \delta)$ and $\beta = cas_{V,U,V}(\delta, \eta)$. Then η is a *retiming* from (U, α) to (V, β), in notation, $(U, \alpha) \to_\eta (V, \beta)$.

See again Fig. 7, interpreting the boxes f and g as functions δ and η. It is routine to check that, if automaton (V, β) results form (U, α) by general sliding (i.e., by applying the sliding axiom in a left-to-right manner), then (V, β) can also be obtained from (U, α) by an appropriate retiming η. Therefore no generality is lost with respect to retiming when using Definition 3.1 instead of the sliding axiom. It is also easy to see that retiming is compatible with the operations serial and parallel composition, and feedback. More precisely, if (U, α), (U', α'), (V, β), and (V', β') are appropriate automata such that $(U, \alpha) \to_\eta (V, \beta)$ and $(U', \alpha') \to_{\eta'} (V', \beta')$, then

$$(U, \alpha) \cdot (U', \alpha') \to_{\eta \times \eta'} (V, \beta) \cdot (V', \beta'),$$
$$(U, \alpha) \otimes (U', \alpha') \to_{\eta \times \eta'} (V, \beta) \otimes (V', \beta'),$$

and $\uparrow^W (U, \alpha) \to_{\eta \times id_W} \uparrow^W (V, \beta)$. Thus, the smallest equivalence relation containing retiming is the congruence induced by the sliding axiom in $Circ$, that is, retiming equivalence. For this

reason we shall also refer to the sliding axiom as the *retiming axiom*. Retiming equivalence will be denoted by $\sim_r$.

The following generalization of the retiming axiom will be crucial in Section 4.

Lemma 3.2. *For any functions $\alpha : U \times A \to V \times C$ and $\beta : V \times B \to U \times D$,*

$$(\nabla_A \otimes \mathbf{1}_B) \cdot (U, cas(\alpha, \beta)) \to_{\pi_{A,U} \circ \alpha} (V, sac(\beta, \alpha)) \cdot (\nabla_C \otimes \mathbf{1}_D).$$

Proof. Monoidal computation, sketched in Fig. 8. Recall that $\nabla_A = \uparrow^A \pi_{A,A} = (A, \pi_{A,A})$ is the unit delay automaton (register) $A \to A$, and *sac* is the "twisted" version of *cas*. The proof stands in all monoidal categories with feedback. □

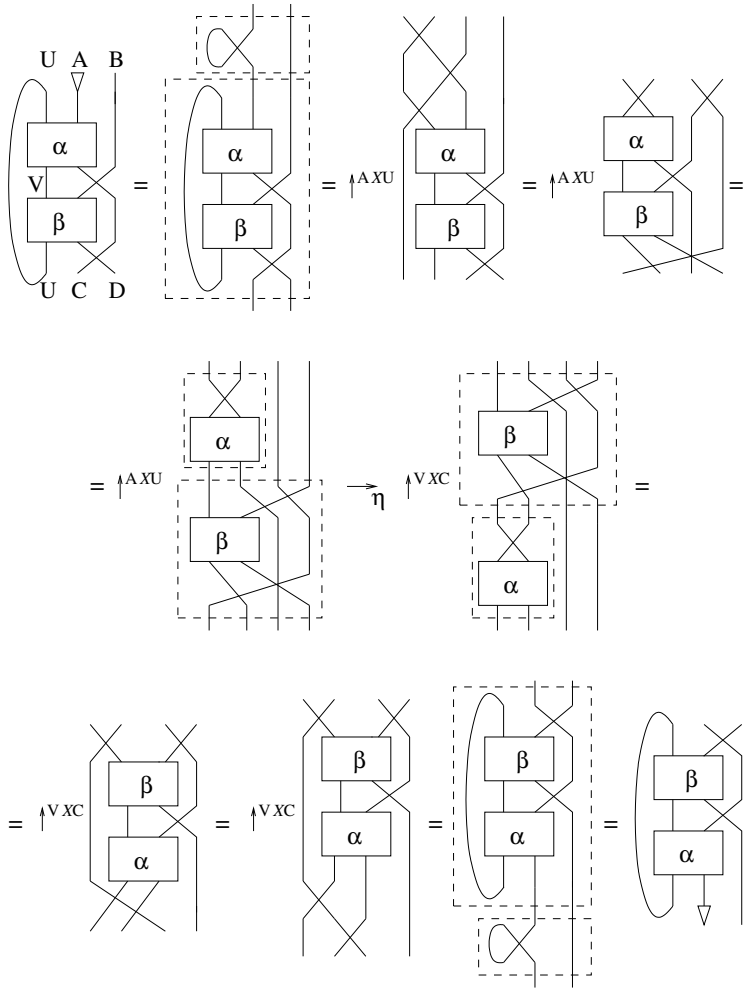

Figure 8: The proof of Lemma 3.2

Evidently, if $(U, \alpha) \to_\eta (V, \beta)$, then η is a homomorphism, or, in our language, an immediate simulation. One disturbing property of such retiming homomorphisms is that they are not composable. The composite of two retiming homomorphisms is just an ordinary homomorphism.

Retiming homomorphisms, on the other hand, are trivially decomposable. If $(U, \alpha) \to_\eta (V, \beta)$, and $\eta = \eta_1 \circ \eta_2$ with $\eta_1 : U \to W$ and $\eta_2 : W \to V$, then there exists an automaton (W, γ) such that $(U, \alpha) \to_{\eta_1} (W, \gamma) \to_{\eta_2} (V, \beta)$. Clearly, $\gamma = cas(\eta_2, \delta, \eta_1)$, where $\delta : V \times A \to U \times B$ is the function specified in Definition 3.1. In particular, retiming homomorphisms have an epi-mono factorization. It is therefore possible to study the impact of surjective and injective retimings separately, and then synthesize the results.

The connection between the automata 3_A and ∇_A in Fig. 4 is a typical example of a surjective retiming $\eta : A' \to A$, where η is the homomorphism s defined there. The effect of η is called state fusion, described as follow. Two states u_1 and u_2 of automaton (U, α) can be picked for fusion if $\alpha(u_1, a) = \alpha(u_2, a)$ for all $a \in A$. In this case, u_1 and u_2 are merged into one state u, so that all transitions arriving at u_1 or u_2 are redirected to u. The inverse operation of state fusion is called state splitting. Now the effect of a general surjective retiming η is merging all states of U that are mapped to the same element in V into one state by state fusion.

The following example illustrates the effect of an injective retiming.

Example Consider the automata $\mathbf{1}_A = (I, id_A)$, $\epsilon_A = (A, \epsilon)$, and $\epsilon'_A = (A, \epsilon')$ $A \to A$, where $A = \{0, 1\}$, $\epsilon(0, i) = \epsilon(1, i) = (0, i)$, $\epsilon'(0, i) = (0, i)$, and $\epsilon'(1, i) = (0, 0)$ for $i = 0, 1$. Clearly, ϵ_A can be obtained from $\mathbf{1}_A$ by state splitting. At the same time, $\mathbf{1}_A$ can be retimed into both ϵ_A and ϵ'_A by the injective retiming $\emptyset \mapsto 0 : I \to A$.

On the analogy of the above example, observe that the effect of an injective retiming is that of an isomorphism, together with the introduction of a number of inaccessible states (i.e., states having no transitions arriving at them), which come with arbitrary transitions not in contradiction with these states being inaccessible. We shall say that a Mealy automaton is *accessible* if it does not have inaccessible states. In general, a state u of a Mealy automaton (U, α) is called *run-out* if it cannot be reached by any sufficiently long input string starting from any state; otherwise u is *permanent*. Since our automata are finite, (U, α) is accessible iff U does not contain run-out states. The restriction of (U, α) to its permanent states will be denoted by (U_p, α_p).

Two states $u, u' \in U$ are said to be *retiming equivalent* (in notation $u \sim_r u'$) if u and u' are equivalent in the usual sense, and, furthermore, u and u' are taken to the same state by (U, α) on every sufficiently long input string w. (Recall e.g. from [13] that state-equivalence of automaton (U, α) is the largest equivalence $\approx$ on U such that whenever $u \approx u'$, $\delta(u, a) \approx \delta(u', a)$ and $\lambda(u, a) = \lambda(u', a)$ hold for every input $a \in A$, where $\alpha = (\delta, \lambda)$ is the decomposition of the transition function to state transition and output.) Equivalently, using the n-step transition function α^n, $\alpha^n(u, w) = \alpha^n(u', w)$ for every sufficiently large n and input $w \in A^n$. Automaton (U, α) is called *reduced* if $u \sim_r u'$ implies $u = u'$ for all states $u, u' \in U$. A *minimal* automaton is one that is both accessible and reduced.

According to the characterization of retiming homomorphisms above, every automaton can be transformed into a reduced one by a sequence of surjective retimings. Also, every automaton can be made accessible by a sequence of inverse injective retimings. Moreover, a straightforward

induction shows that if (V, β) can be obtained from (U, α) by a sequence of direct or inverse retimings, then (U_p, α_p) and (V_p, β_p) reduce to the same minimal automaton, which is unique up to isomorphism. In this way we have proved the following characterization theorem.

Theorem 3.3. *Two Mealy automata (U, α) and (V, β) are retiming equivalent iff the automata (U_p, α_p) and (V_p, β_p) reduce to the same minimal automaton.*

For $(U, \alpha) : A \to B$, consider the automaton $(U, \alpha^k) : A^k \to B^k$. The following statement will be used in Section 4. Its proof should be evident.

Proposition 3.4. *Automaton (U, α) is accessible (reduced, minimal) iff (U, α^k) is such for all $k \geq 1$.*

4. Simulation equivalence coincides with retiming equivalence

We begin with the simple statement that retiming equivalence implies simulation equivalence.

Proposition 4.1. *Let (U, α) and (V, β) be Mealy automata $A \to B$. If $(U, \alpha) \sim_r (V, \beta)$, then $(U, \alpha) \sim (V, \beta)$.*

Proof. We can assume, without loss of generality, that $(U, \alpha) \to_\eta (V, \beta)$ for some retiming $\eta : U \to V$. Then $\alpha = cas(\eta, \delta)$ and $\beta = cas(\delta, \eta)$ for an appropriate function $\delta : V \times A \to U \times B$. Consequently, $cas(\alpha, \eta) = cas(\eta, \beta)$ and $cas(\beta, \delta) = cas(\delta, \alpha)$, so that η and δ are simulations of 0 and 1 step, respectively. Moreover, $cas(\eta, \delta) = \alpha \equiv 1_{(U,\alpha)}$ and $cas(\delta, \eta) = \beta \equiv 1_{(V,\beta)}$, showing that $(U, \alpha) \sim (V, \beta)$. □

Now we turn to the more challenging proof of $\sim \subseteq \sim_r$. The proof is preceded by two lemmas. In these lemmas, the state and output components of the transition function α are denoted by α_{state} and α_{out}, respectively.

Lemma 4.2. *Let (U, α) and (V, β) be accessible automata $A \to B$. If $(U, \alpha^k) \cong (V, \beta^k)$ for all sufficiently large k by the same isomorphism $\xi : U \to V$, then ξ is also an isomorphism from (U, α) to (V, β).*

Proof. We need to show that, for every $u \in U$ and $a \in A$, $\xi(\alpha_{state}(u, a)) = \beta_{state}(\xi(u), a)$ and $\alpha_{out}(u, a) = \beta_{out}(\xi(u), a)$. The latter equation being trivial, we deal with state transitions only. Let $v = \xi(u)$. Since (U, α^k) and (V, β^k) are accessible and isomorphic for any sufficiently large k, there exist states $u_k \in U$, $v_k \in V$ and input $w \in A^k$ such that $\xi(u_k) = v_k$, $\alpha^k_{state}(u_k, w) = u$ and $\beta^k_{state}(v_k, w) = v$. Thus, for the input $wa \in A^{k+1}$,

$$\xi(\alpha_{state}(u, a)) = \xi(\alpha^{k+1}_{state}(u_k, wa)) = \beta^{k+1}_{state}(v_k, wa) = \beta_{state}(v, a),$$

for ξ is an isomorphism between (U, α^{k+1}) and (V, β^{k+1}) as well. □

Lemma 4.3. *Let (U, α) and (V, β) be minimal automata $A \to B$, and assume that there exists an automaton (W, γ) with mappings $\eta : A \times U \to W$ and $\chi : A \times V \to W$ such that*

$$\nabla_A \cdot (U, \alpha) \to_\eta (W, \gamma) \leftarrow_\chi \nabla_A \cdot (V, \beta).$$

Then $(U, \alpha) \cong (V, \beta)$ via an isomorphism ξ that is completely determined by the mappings η and χ.

Proof. By definition, $\nabla_A \cdot (U, \alpha) = (A \times U, \hat{\alpha})$, where for all $a, a' \in A$ and $u \in U$,

$$\hat{\alpha}((a', u), a) = ((a, \alpha_{state}(u, a')), \alpha_{out}(u, a')).$$

Similarly, $\nabla_A \cdot (V, \beta) = (A \times V, \hat{\beta})$. It follows that the automata $\nabla_A \cdot (U, \alpha)$ and $\nabla_A \cdot (V, \beta)$ are accessible, even though they are not minimal in general. Without loss of generality we can assume that the mappings η and χ are onto. Indeed, accessible states are mapped into accessible ones by any homomorphism. Furthermore, if $w \in W$ is missed by either η or χ, then w is inaccessible in (W, γ). Thus, we can concentrate on the surjective part of the epi-mono factorizations of η and χ, which will have the same range.

We claim that for every $u \in U$ there exists a unique $v \in V$ such that

$$\forall a \in A : \eta(a, u) = \chi(a, v).$$

For, let $u = \alpha_{state}(u_1, a_u)$ for some $u_1 \in U$ and $a_u \in A$. (Remember that (U, α) is accessible.) Then there exists a pair $(a_v, v_1) \in A \times V$ such that $w = \eta(a_u, u_1) = \chi(a_v, v_1)$. Set $v = \beta_{state}(v_1, a_v)$. Now,

$$\eta(a, u) = \eta(\hat{\alpha}_{state}((a_u, u_1), a)) = \gamma_{state}(w, a) = \chi(\hat{\beta}_{state}((a_v, v_1), a)) = \chi(a, v).$$

With regard to the uniqueness of v, assume that $\forall a \in A : \eta(a, u) = \chi(a, \bar{v})$ holds for some $\bar{v} \neq v$. Then $\forall a \in A : \chi(a, v) = \chi(a, \bar{v})$, so that for any $a' \in A$, $\hat{\beta}_{state}((a, v), a') = \hat{\beta}_{state}((a, \bar{v}), a')$. (Remember that χ is a retiming.) This last equation implies that $\forall a \in A : \beta_{state}(v, a) = \beta_{state}(\bar{v}, a)$, which contradicts (V, β) being reduced. The correspondence $u \mapsto v$ established above is therefore a mapping $\xi : U \to V$. A symmetric argument shows that ξ is a bijection. Moreover, if u and v are corresponding states, then for all $a, a' \in A$:

$$\alpha_{out}(u, a) = \hat{\alpha}_{out}((a, u), a') = \hat{\beta}_{out}((a, v), a') = \beta_{out}(v, a), \text{ and}$$

$$\eta(a', \alpha_{state}(u, a)) = \eta(\hat{\alpha}_{state}((a, u), a')) = \chi(\hat{\beta}_{state}((a, v), a')) = \chi(a', \beta_{state}(v, a)),$$

proving that ξ is an isomorphism. This isomorphism is completely determined by η and χ, as required. □

Corollary 4.4. *Let (U, α) and (V, β) be minimal automata $A \times C \to B$, and assume that there exists an automaton (W, γ) with mappings $\eta : A \times U \to W$ and $\chi : A \times V \to W$ such that*

$$(\nabla_A \otimes \mathbf{1}_C) \cdot (U, \alpha) \to_\eta (W, \gamma) \leftarrow_\chi (\nabla_A \otimes \mathbf{1}_C) \cdot (V, \beta).$$

Then $(U, \alpha) \cong (V, \beta)$ via an isomorphism ξ that is completely determined by the mappings η and χ.

Proof. Consider the automata $f \cdot (U, \alpha)$ and $f \cdot (V, \beta)$, where

$$f = (\mathbf{1}_A \otimes \nabla_C) \cdot (\nabla_A \otimes \mathbf{1}_C) : A \times C \to A \times C.$$

Observe that $f \cong \nabla_{A \times C}$ by the isomorphism $\pi_{A,C}$. (The category $Circ$ is monoidal.) Furthermore, since retiming is compatible with the operations in $Circ$,

$$f \cdot (U, \alpha) \rightarrow_{\eta'} (\mathbf{1}_A \otimes \nabla_C) \cdot (W, \gamma) \leftarrow_{\chi'} f \cdot (V, \beta)$$

holds with the retimings $\eta' = id_C \times \eta$ and $\chi' = id_C \times \chi$. Consequently,

$$\nabla_{A\times C} \cdot (U, \alpha) \rightarrow_{\eta''} (\mathbf{1}_A \otimes \nabla_C) \cdot (W, \gamma) \leftarrow_{\chi''} \nabla_{A\times C} \cdot (V, \beta),$$

where $\eta'' = (\pi_{A,C} \times id_U) \circ \eta'$ and $\chi'' = (\pi_{A,C} \times id_V) \circ \chi'$. The result now follows from Lemma 4.3, because the condition

$$\forall a \in A \forall c \in C : \eta''((a, c), u) = \chi''((a, c), v)$$

for all states u and v is equivalent to $\forall a \in A : \eta(a, u) = \chi(a, v)$. □

We are now ready to prove the main result of the paper.

Theorem 4.5. *Simulation equivalence of finite state Mealy automata coincides with retiming equivalence.*

Proof. Let (U, α) and (V, β) be simulation equivalent automata $A \rightarrow B$. By Proposition 4.1 we need only show that $(U, \alpha) \sim_r (V, \beta)$. In the light of Proposition 4.1 and Theorem 3.3 we can assume, without loss of generality, that (U, α) and (V, β) are minimal. Consequently, the automata (U, α^k) and (V, β^k) are also minimal for all $k \geq 1$ by Proposition 3.4. According to Lemma 4.2, it is therefore sufficient to prove that $(U, \alpha^k) \cong (V, \beta^k)$ for all sufficiently large k via a fixed isomorphism ξ.

By definition, there exist simulations $s : (U, \alpha) \rightarrow (V, \beta)$ and $t : (V, \beta) \rightarrow (U, \alpha)$ such that $cas(s, t) \equiv 1_{(U,\alpha)}$ and $cas(t, s) \equiv 1_{(V,\beta)}$. Spelling this out, we have:

(i) $cas(\alpha, s) = cas(s, \beta)$ and $cas(\beta, t) = cas(t, \alpha)$;

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

where $s : U \times A^n \rightarrow V \times B^n$, $t : V \times A^m \rightarrow U \times B^m$, and k is any sufficiently large integer. Furthermore, $cas(\alpha^k, s, t) = cas(s, t, \alpha^k)$ follows from a repeated application of (i) above. Using Lemma 3.2, a short computation yields:

$$(\nabla_{A^n} \otimes \mathbf{1}_{A^{m+k}}) \cdot (U, \alpha^{n+m+k}) \rightarrow_\eta (V, sac(\beta^{k+m}, \beta^n)) \cdot (\nabla_{B^n} \otimes \mathbf{1}_{B^{k+m}}),$$

where $\eta = \pi_{A^n,U} \circ s$. See Fig. 9 for the case $n = m = k = 1$. On the other hand, directly from Lemma 3.2:

$$(\nabla_{A^n} \otimes \mathbf{1}_{A^{m+k}}) \cdot (V, \beta^{n+m+k}) \rightarrow_\chi (V, sac(\beta^{m+k}, \beta^n)) \cdot (\nabla_{B^n} \otimes \mathbf{1}_{B^{m+k}}),$$

where $\chi = \pi_{A^n,U} \circ \alpha^n$. Observe that η and χ do not depend on k. Therefore, according to Corollary 4.4, $(U, \alpha^k) \cong (V, \beta^k)$ holds for all sufficiently large k via a fixed isomorphism ξ, which completes the proof. □

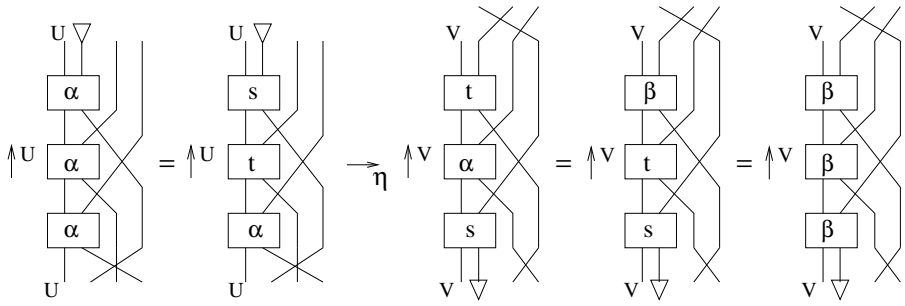

Figure 9: The proof of Theorem 4.5

5. Conclusions

We have given two different characterizations of the congruence relation induced by the sliding/retiming axiom in the monoidal category $Circ$ of Mealy automata. First we introduced simulation as an additional vertical structure in $Circ$, and defined simulation equivalence to be a pair of isomorphisms in this vertical category. Operating with surjective and injective retiming homomorphisms, we then showed that two automata are retiming equivalent iff their restriction to permanent states can be reduced to the same minimal automaton. On the basis of this result we proved that retiming equivalence coincides with simulation equivalence. In this way we have also provided a semantics to the sliding axiom in certain monoidal categories with feedback.

References

[1] ABRAMSKY, S., Retracing some paths in process algebras, in: U. Montanari, V. Sassone (Eds.), Proc. CONCUR'96, Pisa, 1996, Lecture Notes in Computer Science 1119 (1996), 1–17.

[2] ARBIB, M.A., Theories of Abstract Automata, Prentice-Hall, Englewood Cliffs, N.J. 1969.

[3] BARTHA, M., An equational axiomatization of systolic systems, Theoretical Computer Science 55 (1987), 265–289.

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

[5] BARTHA, M., Foundations of a theory of synchronous systems, Theoretical Computer Science 100 (1992), 325–346.

[6] BARTHA, M., ČIROVIČ, B., On some equivalence notions of synchronous systems, in: Z. Ésik, Z. Fülöp (Eds.), Proc. 11th International Conference on Automata and Formal Languages, Dogogókő, Hungary, 2005, 69–82.

[7] BARTHA, M., Strong retiming equivalence of synchronous schemes, in: J. Farré, I. Litovsky, S. Schmitz (Eds.), Proc. 10th International Conference, CIAA, Sophia Antipolis, 2005, Lecture Notes in Computer Science 3845 (2006), 66–77.

[8] BARTHA, M., Simulation equivalence of automata and circuits, in: E. Csuhaj-Varjú, Z. Ésik (Eds.), Proc. 12th International Conference on Automata and Formal Languages, Balatonfüred, Hungary, 2008, 86–99.

[9] BLOOM, S.L., ÉSIK, Z., Iteration Theories: The Equational Logic of Iterative Processes, Springer-Verlag, Berlin 1993.

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

[11] EHRIG, H., Universal Theory of Automata, B.G. Teubner, Stuttgart 1974.

[12] GÉCSEG, F., PEÁK, I., Algebraic Theory of Automata, Akadémiai Kiadó, Budapest 1972.

[13] HARTMANIS, J., STEARNS, R.E., Algebraic Structure Theory of Sequential Machines, Prentice-Hall, Englewood Cliffs, N.J. 1966.

[14] JOYAL, A., STREET, R., VERITY, D., Traced monoidal categories, Mathematical Proceedings of the Cambridge Philosophical Society 119 (1996), 447–468.

[15] KATIS, P., SABADINI, N., WALTERS, R.F.C., Feedback, trace, and fixed-point semantics, Theoretical Informatics and Applications 36 (2002), 181–194.

[16] LEISERSON, C.E., SAXE, J.B., Optimizing synchronous systems, Journal of VLSI and Computer Systems 1 (1983), 41–67.

[17] MACLANE, S., Categories for the Working Mathematician, Springer-Verlag, Berlin 1971.

[18] MANES, E.G., Algebraic Theories, Springer-Verlag, Berlin 1976.