Iterating transducers*

Dennis Dams

Bell Laboratories, Lucent Technologies, Murray Hill NJ 07974, USA
On leave from Eindhoven University of Technology, The Netherlands
dennis@research.bell-labs.com

Yassine Lakhnech

VERIMAG, *Centre Equation*
2 av. de Vignate, 38610 Gières, France
lakhnech@imag.fr

Martin Steffen

Institut für angewandte Mathematik und Informatik
Christian-Albrechts-Universität, Preußerstraße 1–9, D-24105 Kiel, Deutschland
ms@informatik.uni-kiel.de

Abstract

Regular languages have proved useful for the symbolic state exploration of infinite state systems. They can be used to represent infinite sets of system configurations; the system's transitional semantics consequently can be modeled by finite-state transducers. A standard problem for infinite state systems is how to explore all states in a finite amount of time. When the initial states of a system are represented by a finite-state automaton $\mathcal{A}$ and the one-step transition relation by a finite-state transducer $\mathcal{T}$, this problem amounts to effectively computing an appropriate finite-state representation $\mathcal{T}^* \circ \mathcal{A}$ for the transduction of $\mathcal{A}$ under $\mathcal{T}$'s transitive closure.

In this paper we give a partial algorithm to compute a finite-state transducer $\mathcal{T}^*$ for a general class of transducers. The construction builds a quotient of an underlying infinite-state transducer $\mathcal{T}^{<\omega}$, using a novel behavioural equivalence based on *past* and *future* bisimulations on finite approximations of $\mathcal{T}^{<\omega}$. The extrapolation to $\mathcal{T}^{<\omega}$ of these finite bisimulations capitalizes on the structure of the states of $\mathcal{T}^{<\omega}$, which are strings of states of $\mathcal{T}$. We show how this extrapolation may be rephrased as a problem of detecting *confluence* properties of rewrite systems that represent the bisimulations. Thus, we can draw upon techniques from the area of rewriting.

A prototype implementation has been successfully applied to various examples.

Key words: model checking, regular model checking, transducer, rewrite systems

1 Introduction

Finite-state automata are omnipresent in computer science, providing a powerful tool for representing and reasoning about certain infinite phenomena. They are commonly used to capture dynamic behaviours, in which case an automaton's nodes model the states, and its edges the possible state transitions of a system. More recently, finite-state automata have also been applied to reason about infinite-state systems, in which case a single automaton is used to represent an infinite set of system states. In regular model-checking [15, 49, 1, 48, 3], regular sets of states of the system to be verified are represented by finite-state automata. For instance, consider a parameterized linear network of finite-state processes with the states of the processes modeled by the symbols of a finite alphabet. Then for every value of the parameter, i.e., for every fixed length of the network, a global configuration is represented by a word over the alphabet. A set of similar configurations corresponding to different values of the parameter, and hence to different network sizes, can then be modelled by a regular set. Or, in a system with data structures like unbounded message buffers, infinitely many buffer contents may be represented by an automaton. To reason about the dynamic behaviour of such a system, its transition relation is lifted to operate on such symbolically represented sets of states. A natural choice to represent the lifted transition relation are *finite-state transducers*.

Taking finite-state automata and transducers to describe infinite sets of states and their operational evolution is, in general, not sufficient when doing state exploration. To capture all reachable states, one needs to characterize the effect of applying a transducer an arbitrary number of times, in other words, one needs to compute $\mathcal{T}^* \circ \mathcal{A}$, where $\mathcal{A}$ characterizes the initial states and $\mathcal{T}^*$ the transitive closure of $\mathcal{T}$. In this paper, we consider the slightly more general problem of calculating the transducer $\mathcal{T}^*$ instead of the automaton representing $\mathcal{T}^* \circ \mathcal{A}$. In general, $\mathcal{T}^*$ is not finite-state anymore (and neither $\mathcal{T}^* \circ \mathcal{A}$). Indeed, it is undecidable whether the iteration of a given transducer is regular. This follows from Corollary 3.11 in [51], which shows (roughly) that iterated transducers are comparable to context-sensitive grammars. This is then to be combined with the fact that it is undecidable whether a given context-sensitive grammar is regular (see e.g. [44]).

Nonetheless, for length-preserving transducers, partial algorithms have been developed that, if they terminate, produce the closure in the form of a finite-state transducer [15, 48]. These algorithms can be explained in terms of the in general infinite-state transducer $\mathcal{T}^{<\omega} = \bigcup_{i \in \omega} \mathcal{T}^i$, the union of all finite

⋆ This work has been supported by the Esprit-LTR project *Vires*. A short version appeared in the Proceedings of the 13th Conference on Computer Aided Verification (CAV'01).

compositions of $\mathcal{T}$. Conceptually, they attempt to construct a finite quotient [1] of $\mathcal{T}^{<\omega}$ by identifying states that are equivalent in some way. For example, in [15, 48], the partial procedure for computing a finite quotient of $\mathcal{T}^{<\omega}$ is based on a subset construction applied to a transducer whose states are words over the states of $\mathcal{T}$. This infinite transducer realizes $\mathcal{T}^{<\omega}$. To make the subset construction converge in more cases than it would do, the constructed states, which are rational languages over the states of $\mathcal{T}$, are *saturated* according to an equivalence relation.

In this paper, we employ a different quotient construction to render $\mathcal{T}^{<\omega}$ finite, resulting in an algorithm whose application is not a-priori limited to length-preserving transducers. It works by computing successively the approximants $\mathcal{T}^{\leq n} = \bigcup_{0 \leq i \leq n} \mathcal{T}^i$ for $n = 0, 1, 2, 3, \ldots$, while attempting to *accelerate* the arrival at a fixpoint by collapsing states. This quotienting is based on a novel behavioural equivalence defined in terms of *past* and *future bisimulations*. This equivalence has to be infinite to collapse the infinite-state transducer $\mathcal{T}^{<\omega}$ to a finite one. Therefore, one is faced with the problem of effectively computing and representing a suitable infinite equivalence. To solve this problem, we first identify sufficient conditions on an approximant $\mathcal{T}^{\leq n}$ for its states (which are also states of $\mathcal{T}^{<\omega}$) to be equivalent as states of $\mathcal{T}^{<\omega}$. Then we show that the equivalence of two states of $\mathcal{T}^{\leq n}$ induces the equivalence of infinitely many states of $\mathcal{T}^{<\omega}$.

We illustrate the underlying intuition on a small example in which sets of unbounded natural numbers are represented as automata over the symbols 0 and *succ*. The transitions we consider are given by the function α, defined inductively by $\alpha(0) = \mathit{even}$ and $\alpha(\mathit{succ}(x)) = \neg(\alpha(x))$. It computes the parity *even* or *odd* of a number; $\neg$ is a function that toggles parities. Consider the transition relation $\rightarrow$ that corresponds to a single step in the evaluation of this recursive definition. Fig. 1(a) gives a transducer, $\mathcal{T}_\alpha$, that represents this transition relation. The slash (/) is used to separate the input symbol from the

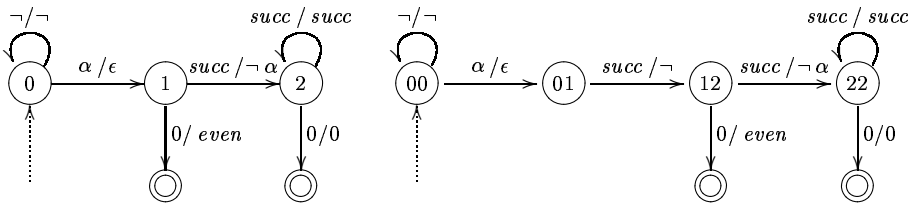


Fig. 1. Left (a): The transducer $\mathcal{T}_\alpha$. Right (b): Its product $\mathcal{T}_\alpha^2$.

output symbols; ϵ denotes the empty string. Note that by the self-loop on state 0, the transducer leaves any leading occurrences of the symbol $\neg$ unchanged, and similarly for the trailing occurrences of *succ* before the final 0.

[1] Note that we use the quotient construction in cases in which the underlying equivalence is not necessarily a bisimulation.

To start approximating $\mathcal{T}_\alpha^{<\omega}$, consider the product transducer $\mathcal{T}_\alpha^2$ shown in Fig. 1(b): It moves the symbol α over one more occurrence of *succ*, while turning it into a $\neg$, as reflected by the edge from state 01 to 12. In every next product transducer $\mathcal{T}_\alpha^3, \mathcal{T}_\alpha^4, \ldots$, an additional such *succ* /$\neg$-edge will appear. Clearly, the limit transducer $\mathcal{T}_\alpha^{<\omega}$, the union of all approximants, is going to have infinitely many states. On the other hand, the combined effect of the ever-growing sequence of *succ* /$\neg$-edges would be captured by a simple loop if states 01 and 12 were identified. Collapsing $\mathcal{T}_\alpha^{<\omega}$ in this way, we can hope for a finite quotient. To do so, we need to address the following questions: First, how can we justify equating pairs of states like 01 and 12, which are obviously semantically different in that they realize different transductions, i.e., what is the equivalence notion on $\mathcal{T}_\alpha^{<\omega}$ employed for quotienting. Secondly, how to compute the quotient without prior calculation of the infinite $\mathcal{T}_\alpha^{<\omega}$?

As for the first point, we must ensure that identifying states in the quotient does not introduce transductions not already present in $\mathcal{T}_\alpha^{<\omega}$. Equating 01 with 12 in the above example, consider the run through the "collapsed" transducer that goes from 00 to 01 (or rather to the new state obtained by collapsing 01 and 12) and then continues from this state as if continuing from 12. Exploiting the equation $01 = 12$, this run is introduced by the collapse. Even though the states 01 and 12 are semantically different, as observed above, identifying them does not change the *overall* semantics of $\mathcal{T}_\alpha^{<\omega}$, as there *exists* another state that "glues" together the past of 01 and the future of 12, namely state 1 of $\mathcal{T}_\alpha$. Another class of runs that are introduced by the collapse are those that go from 00 to 12 and then continue as if continuing from 01. But also in this case, there is a state in $\mathcal{T}_\alpha^{<\omega}$ that glues (this time) the past of 12 to the future of 01, although it has not been constructed when considering $\mathcal{T}_\alpha^{\leq 2}$. This state is 012 and would enter the scene as part of $\mathcal{T}_\alpha^3$, when constructing the next approximant. We formalize these ideas as follows: States q_1 and q_2 may be identified if there exists a past bisimulation P and a future bisimulation F such that the pair (q_1, q_2) is both in the composed relation $P;F$ and in $F;P$, thus ensuring the existence of both "gluing" states. Indeed, we will require that the bisimulations *swap*, i.e., $F;P{=}P;F$. So it will be enough to show that (q_1, q_2) is in either one of the composed relations. The situation is sketched in Fig. 2, containing $\mathcal{T}_\alpha$ and $\mathcal{T}_\alpha^2$ and where the state 012 of the not-yet-constructed $\mathcal{T}_\alpha^3$ is drawn by a dotted circle.

The second question is how to detect equivalent states in some approximant $\mathcal{T}_\alpha^{\leq n}$, i.e., how do we know that there exists a state *somewhere* in $\mathcal{T}_\alpha^{<\omega}$ that is past-bisimilar to one and future-bisimilar to the other? To this end we exploit the structure of $\mathcal{T}_\alpha^{<\omega}$'s states, namely that they are sequences of states from $\mathcal{T}_\alpha$. It is easily seen that bisimulations are *congruences* under juxtaposition of such sequences. In the example above, we can conclude the existence of an appropriate state without actually having to construct $\mathcal{T}_\alpha^3$: By looking at $\mathcal{T}^{\leq 2}$ only, we see that 1 and 12 are future bisimilar, whence by congruence also 01

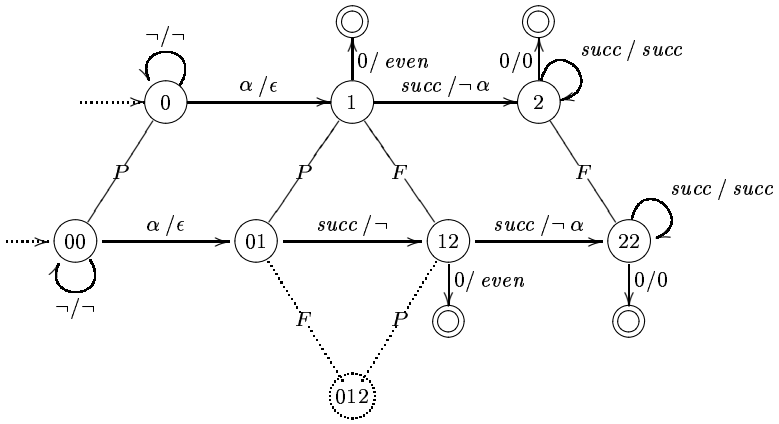


Fig. 2. $\mathcal{T}_\alpha$ and $\mathcal{T}_\alpha^2$ with past and future bisimulations

and 012. Similarly, past bisimilarity of 12 and 012 can be inferred from 1 and 01. Again, the inferred pairs $(01, 012) \in F$ and $(012, 12) \in P$ are indicated by dotted lines in Fig 2. Furthermore, exploiting the congruence property allows to extrapolate the quotienting relation found on a finite $\mathcal{T}_\alpha^{\leq n}$ to the whole $\mathcal{T}_\alpha^{<\omega}$, and thus to obtain a finite quotient of $\mathcal{T}_\alpha^{<\omega}$, without calculating the limit first.

The remainder of the paper is organized as follows. After introducing notation and the relevant preliminary definitions in the next section, Section 3 will formalize the criterion for a sound quotient. An algorithm based on this and profiting from results of rewriting theory is the topic of Section 4, where we will also report on the results obtained from our prototype implementation. Sections 5 and 6 conclude with related and future work.

2 Preliminaries

In this section we introduce word transducers and pieces of string rewrite theory pertinent for the later development.

2.1 Finite state word transducers

A *transducer* $\mathcal{T} = (Q, Q_i, Q_f, (\Sigma_1, \Sigma_2), R)$ consists of a set Q of states, sets $Q_i, Q_f \subseteq Q$ of initial resp. final states, a *signature* (Σ_1, Σ_2) specifying the input and output symbols, and a set R of rules. As usual, given a set Σ, Σ^* denotes the monoid of words over Σ, where we write word concatenation by juxtaposition and the pointwise lifting of concatenation to languages is written the same way. The empty word is written as ϵ. A rule of $\mathcal{T}$ has the form $qa \longrightarrow wq'$ with $q, q' \in Q$, $a \in \Sigma_1 \cup \{\epsilon\}$, and $w \in \Sigma_2^*$, specifying that when in state q and reading input symbol $a \in \Sigma_1$ (or reading no input in case

$a = \epsilon$), the transducer produces the output word w and assumes q' as its new state. A transducer whose sets of states and rules are both finite is also called *regular* or *rational*. The operation of a transducer is captured by the *reduction* relation $\rightarrow_R$ on strings consisting of symbols and a state (where ϵ has its usual meaning as neutral element of concatenation), defined as follows: For $t_1 \in \Sigma_1^*$ and $t_2 \in \Sigma_2^*$, $t_1 q a t_2 \rightarrow_R t_1 w q' t_2$ iff $qa \rightarrow wq' \in R$. For this and other arrows we use common notations like $\rightarrow^{-1}$ for inverse, $\rightarrow^*$ for reflexive-transitive closure, and $\leftrightarrow$ for symmetric closure. The transduction realised by a state q of $\mathcal{T}$, denoted $[\![q]\!]_\mathcal{T}$ or simply $[\![q]\!]$ if $\mathcal{T}$ is clear, is the function from Σ_1^* to $2^{(\Sigma_2^*)}$ defined by $t_2 \in [\![q]\!]_\mathcal{T}(t_1)$ iff there exists $q_f \in Q_f$ such that $qt_1 \rightarrow_R^* t_2 q_f$. The *semantics* of $\mathcal{T}$ is the function $[\![\mathcal{T}]\!] : \Sigma_1^* \rightarrow 2^{(\Sigma_2^*)}$ defined by $t_2 \in [\![\mathcal{T}]\!](t_1)$ iff there exist $q_i \in Q_i$ and $q_f \in Q_f$ such that $q_i t_1 \rightarrow_R^* t_2 q_f$. We will use the notation $\rightarrow_\mathcal{T}$ synonymously for the rewrite relation $\rightarrow_R$. We write $\mathcal{T} : \Sigma_1 \rightarrow \Sigma_2$, if $\mathcal{T}$ realizes a transduction from words over Σ_1 to words over Σ_2. In case $\Sigma_1 = \Sigma_2$, we write $\mathcal{T}$ shorter as (Q, Q_i, Q_f, Σ, R).

In the informal presention in Section 1, we represented the transducers graphically (cf. Fig. 1): States are given as nodes of a graph, the initial states are marked by an ingoing dotted arrow, the final states by using a double circle. The edges of the graph represent the rules are labelled by the pair of the input symbol (or ϵ) and the output word of the rule.

Transducers $\mathcal{T}_1 : \Sigma_1 \rightarrow \Sigma_2$ and $\mathcal{T}_2 : \Sigma_2 \rightarrow \Sigma_3$ can be composed into $\mathcal{T}_2 \circ \mathcal{T}_1$ by a standard product construction, where the rules R of the composition are defined by [2]

$$\frac{q_j a \rightarrow_{R_1} v q_j' \qquad q_i v \rightarrow_{R_2}^* w q_i'}{q_{ij} a \rightarrow w q_{ij}' \in R} \textsc{Comp}$$

In Comp, R_1 and R_2 are the rules of the two constituent transducers, and q_{ij} is a short-hand for the tuple of q_i and q_j. Note that multiple steps of $\mathcal{T}_2$ may be needed for q_i to "move through" v (or none, if $v = \epsilon$). This construction captures the semantical composition, i.e., $[\![\mathcal{T}_2]\!] \circ [\![\mathcal{T}_1]\!] = [\![\mathcal{T}_2 \circ \mathcal{T}_1]\!]$. An example for the composition of two transducers is shown in Fig. 1(b).

Being interested in equality only up-to isomorphism, associativity of composition justifies the notation $\mathcal{T}^n : \Sigma \rightarrow \Sigma$ for an n-fold composition of a transducer $\mathcal{T} : \Sigma \rightarrow \Sigma$ with itself. By the same token we will use Q^n as the set of states of $\mathcal{T}^n$, when Q is the set of states of $\mathcal{T}$. Typical elements from Q^n — and later from Q^*, as well — will be denoted q_α, q_β, $q_{\gamma'}$ ..., and sometimes even α, β, γ' ..., if the meaning is clear from the context. Also we identify

[2] It should go without saying that in the definition — and for all comparable situations for the rest of the paper — we will silently assume that the sets of states of the different transducers are disjoint, so as to obviate accidental confusion.

notationally the state $q_\alpha q_\beta$ with $q_{\alpha\beta}$. Besides composition, we will later need the *union* of two transducers. By $\mathcal{T}_1 \cup \mathcal{T}_2$ for two transducers over the same signature we simply mean the transducer over the same signature, given by the union of states, of initial states, of final states, and the union of rules, respectively. Note that finite union preserves finiteness. For $n < \omega$ we define $\mathcal{T}^{\leq n} = \bigcup_{0 \leq i \leq n} \mathcal{T}^i$. Union can be easily extended to the union of countably many transducers; we define $\mathcal{T}^{<\omega} = \bigcup_{i \in \omega} \mathcal{T}^i$. Using Q^* for the set of states of $\mathcal{T}^{<\omega}$, we denote the empty sequence of states by q_ϵ, which represents the neutral element wrt. concatenation, i.e., $q_\epsilon q_\alpha = q_\alpha = q_\alpha q_\epsilon$. With this convention, we can define the zeroth iteration $\mathcal{T}^0$ of a transducer $\mathcal{T} : \Sigma \to \Sigma$ as $(\{q_\epsilon\}, \{q_\epsilon\}, \{q_\epsilon\}, \Sigma, \emptyset)$. Under the conventions for q_ϵ, the transducer $\mathcal{T}^0$ clearly realizes the neutral element wrt. transduction composition, i.e., $[\![\mathcal{T}^0]\!] = Id_{\Sigma^*}$.

2.2 Bisimulations and quotienting

To obtain a finite-state transducer from an a priori infinite $\mathcal{T}^{<\omega}$, we will have to identify certain states. The notion of equivalence used to this end will be based on *bisimulation* equivalences [63, 57] on states. Besides the standard future bisimulation we need the *past* variant as well. Note that we require bisimulations to be equivalences. This is because we need them to quotient transducers.

Definition 1 (Bisimulation) *Let $\mathcal{T} = (Q, Q_i, Q_f, (\Sigma_1, \Sigma_2), R)$ be a transducer. An equivalence relation $F \subseteq Q \times Q$ is a* future bisimulation *on $\mathcal{T}$ if for all pairs (q_1, q_2) of states, $q_1\ F\ q_2$ implies:*

> *If $q_1 \in Q_f$, then $q_2 \in Q_f$, and for every a, w, q_1' such that $q_1 a \rightarrow_{\mathcal{T}} w q_1'$, there exists q_2' such that $q_2 a \rightarrow_{\mathcal{T}} w q_2'$ and $q_1'\ F\ q_2'$.*

An equivalence relation $P \subseteq Q \times Q$ is a past bisimulation *on $\mathcal{T}$, if for all pairs (q_1', q_2') of states, $q_1'\ P\ q_2'$ implies:*

> *If $q_1' \in Q_i$, then $q_2' \in Q_i$, and for every a, w, q_1 such that $q_1 a \rightarrow_{\mathcal{T}} w q_1'$, there exists q_2 such that $q_2 a \rightarrow_{\mathcal{T}} w q_2'$ and $q_1\ P\ q_2$.*

We call q_1 and q_2 (future) bisimilar*, written $q_1 \sim_f q_2$, if there exists a future bisimulation F with $q_1\ F\ q_2$; and $q_1 \sim_p q_2$ denotes two* past bisimilar *states, defined analogously. For transducers, we write $\mathcal{T}_1 \sim_f \mathcal{T}_2$, if there exists a future bisimulation $F \subseteq Q \times Q$ such that for all $q_1 \in Q_i$, there exists $q_2 \in Q_i$ such that $q_1\ F\ q_2$, and conversely for all $q_2 \in Q_i$ there exists $q_1 \in Q_i$ such that $q_1\ F\ q_2$.*

In correspondence with the rules of the transducer, bisimulation is defined to preserve the relationship after consuming one symbol of the input alphabet.

Later we will need the generalization to the case where the transducer does more than just one basic step. In the following we denote by $\hat{R}$ the homomorphic lifting of a relation R on states Q to a relation on words from $\Sigma_1^* Q \Sigma_2^*$.

Lemma 2 *Let $\mathcal{T} = (Q, Q_i, Q_f, (\Sigma_1, \Sigma_2), R)$ be a transducer, and further $F \subseteq Q \times Q$ and $P \subseteq Q \times Q$ a future and a past bisimulation.*

(1) If $t_1 \hat{F} t_2$ and $t_1 \rightarrow_{\mathcal{T}}^* t_1'$, then $t_2 \rightarrow_{\mathcal{T}}^* t_2'$ and $t_1' \hat{F} t_2'$, for some t_2'.
(2) If $t_1' \hat{P} t_2'$ and $t_1 \rightarrow_{\mathcal{T}}^* t_1'$, then $t_2 \rightarrow_{\mathcal{T}}^* t_2'$ and $t_1 \hat{P} t_2$, for some t_2.

Above, the words t_1, t_2, t_1', and t_2' are from $\Sigma_2^* Q \Sigma_1^*$.

The bisimulation relations enjoy the following properties ([57]):

Lemma 3 *Let $\mathcal{T}_1 : \Sigma_1 \to \Sigma_2$ and $\mathcal{T}_2 : \Sigma_2 \to \Sigma_3$ be two transducers with state sets Q_1 and Q_2 respectively.*

(1) The identity relation $Id \subseteq Q_1 \times Q_1$ is a future bisimulation.
(2) If $F_1 \subseteq Q_1 \times Q_1$ and $F_2 \subseteq Q_1 \times Q_1$ are future bisimulations, then so is $(F_1; F_2 \cup F_2; F_1)^*$.
(3) If $F_1 \subseteq Q_1 \times Q_1$ and $F_2 \subseteq Q_2 \times Q_2$ are future bisimulations on $\mathcal{T}_1$ and $\mathcal{T}_2$ resp., then $F \subseteq (Q_2 \times Q_1) \times (Q_2 \times Q_1)$ defined by $q_2 q_1 \; F \; \tilde{q}_2 \tilde{q}_1$ iff $q_1 \; F_1 \; \tilde{q}_1$ and $q_2 \; F_2 \; \tilde{q}_2$ is a future bisimulation on $\mathcal{T}_2 \circ \mathcal{T}_1$.

The same holds correspondingly for past bisimulations.

The second point basically states that bisimulations are closed under relational composition, but because we require bisimulations to be equivalence relations, we need to take the symmetric-transitive closure. For the same reason, we need to take the transitive closures of the unions in the following.

Lemma 4 *Given a transducer $\mathcal{T} = (Q, Q_i, Q_f, (\Sigma_1, \Sigma_2), R)$.*

(1) If $F_1, F_2 \subseteq Q \times Q$ are future bisimulations, then so is $(F_1 \cup F_2)^*$.
(2) If $P_1, P_2 \subseteq Q \times Q$ are past bisimulations, then so is $(P_1 \cup P_2)^*$.

PROOF. We show only the forward case, the past one is analogous. Let F abbreviate $(F_1 \cup F_2)^*$. Since F_1 and F_2 are symmetric, so is their union, and hence F is an equivalence.

For the condition on initial states, assume $q_1 F q_2$ and $q_1 \in Q_i$. By induction on the length of $q_1 (F_1 \cup F_2)^* q_2$, also $q_2 \in Q_i$. For preservation under steps, assume $q_1 F q_2$ and $q_1 a \rightarrow_{\mathcal{T}} w q_1'$ for some $a \in \Sigma_1 \cup \{\epsilon\}$ and some word $w \in \Sigma_2^*$. Proceed by induction on $q_1 (F_1 \cup F_2)^k q_2$. The base case, $k = 0$, is immediate. For $k > 0$, there exists a state $q_3 \in Q$ with $q_1 (F_1 \cup F_2) \; q_3 \; (F_1 \cup F_2)^{k-1} q_2$. We distinguish, whether $q_1 F_1 q_3$ or $q_1 F_2 q_3$. Since F_1 is a future bisimulation, we get

in the first case that there exists a state $q_3' \in Q$ with $q_1' F_1 q_3'$ and $q_3 a \to_{\mathcal{T}} w q_3'$. By induction, we get a state q_2' with $q_2 a \to_{\mathcal{T}} w q_2'$ and $q_3'\ F\ q_2'$. With $F_1 \subseteq F$ and transitivity of F, also $q_1'\ F\ q_2'$. The case where $q_1\ F_2\ q_3$ is symmetric. □

Lemma 5 *Let $\mathcal{T}$ be a transducer with state set Q. The relation $\sim_f$ on Q is a future bisimulation, $\sim_p$ is a past bisimulation.*

It is standard to show that future bisimilarity implies semantical equality, i.e., $\mathcal{T}_1 \sim_f \mathcal{T}_2$ implies $[\![\mathcal{T}_1]\!] = [\![\mathcal{T}_2]\!]$, and that the two relations $\sim_p$ and $\sim_f$ are *congruences* on Q^*, the free monoid over of $\mathcal{T}$'s set Q of states: If $\alpha \sim_f \alpha'$ and $\beta \sim_f \beta'$, then $\alpha\beta \sim_f \alpha'\beta'$, for all $\alpha, \alpha', \beta, \beta' \in Q^*$, and similarly for $\sim_p$. We will exploit this property in Section 4.

Lemma 6 *Given a transducer $\mathcal{T}$, the relations $\sim_p$ and $\sim_f$ are congruences on Q^*, the free monoid over the set Q of states of $\mathcal{T}$.*

The definition of a quotient is fairly standard: Its states are given as equivalence classes wrt. the quotienting relation, the initial resp. final states are the classes containing an initial resp. a final state of the original transducer, and a rule connects a left-hand and a right-hand side in the quotient if there exist two corresponding terms forming a rule of the original transducer.

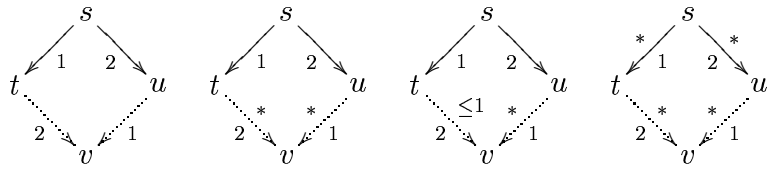
Definition 7 (Quotient) *Let $\mathcal{T} = (Q, Q_i, Q_f, \Sigma, R)$ be a transducer and $\cong \subseteq Q \times Q$ an equivalence relation. $\mathcal{T}_{/\cong}$ is defined as the transducer $(Q_{/\cong}, \{[q]_\cong \mid q \in Q_i\}, \{[q]_\cong \mid q \in Q_f\}, \Sigma, R_{/\cong})$, where $Q_{/\cong}$ is the set of $\cong$-equivalence classes of Q and $[q]_\cong$ the $\cong$-equivalence class of q. The rules of $\mathcal{T}_{/\cong}$ are given by $\hat{q}a \to w\hat{q}' \in R_{/\cong}$ iff there exist q and q' such that $\hat{q} = [q]_\cong$, $\hat{q}' = [q']_\cong$, and $qa \to wq' \in R$.*

2.3 Rewrite systems

Here, we briefly recall some notions and results from (string) rewriting, which we will need in Section 4. A more thorough treatment of the field can be found e.g. in [7, 26] or, for string rewrite systems, in [13]. Given a finite alphabet Σ, a *string rewriting system R on Σ*, sometimes also called semi-Thue system, is a subset of $\Sigma^* \times \Sigma^*$. Each pair $(\alpha, \beta) \in R$ is called a *rule* and states that, in any context, the string α may be replaced by β. This is formally captured by the *rewrite relation generated by R*, $\to_R$, defined by $\alpha_l \beta \alpha_r \to_R \alpha_l \beta' \alpha_r$ iff $(\beta, \beta') \in R$ and $\alpha_l, \alpha_r \in \Sigma^*$. The relation $\to_R$ is said to be *terminating* (also called *strongly normalizing* or *Noetherian*) if it allows no infinite sequences of rewrite steps. The *congruence closure* of R, also known as Thue congruence, over the monoid Σ^* of strings over Σ is $\leftrightarrow_R^*$. Note in passing that the semantics of a word transducer with signature (Σ_1, Σ_2) is given by a string rewriting system where rules are drawn specifically from $Q(\Sigma_1 \cup \{\epsilon\}) \times \Sigma_2^* Q$.

The rewrite relation given by R is a priori non-deterministic in that $\to_R$ neither prescribes an order among the rules nor to which substring a rule is applied. That the non-determinism of $\to_R$ is inessential is captured by *confluence* [3], a key concept of reduction systems in general and rewrite systems specifically: At any point, the outcomes of two different choices can be reconciled by subsequent reduction. We will also need the corresponding notion for pairs of relations, known as commutation.

Definition 8 (Confluence and commutation) *Let $\to_1$ and $\to_2$ be two relations. The relations $\to_1$ and $\to_2$ are said to enjoy the* commuting diamond property, *if, for all s, t, and u, $s \to_1 t$ and $s \to_2 u$ implies $t \to_2 v$ and $u \to_1 v$ for some v. The two relations are said to* locally commute, *if $s \to_1 t$ and $s \to_2 u$ implies $t \to_2^* v$ and $u \to_1^* v$ for some v. They are* strongly commuting, *if $s \to_1 t$ and $s \to_2 u$ implies $t \to_2^{\leq 1} v$ and $u \to_1^* v$ for some v. The two relations are said to* commute, *if $s \to_1^* t$ and $s \to_2^* u$ implies $t \to_2^* v$ and $u \to_1^* v$. The definitions are summarized in the diagrams below.*



In case $\to_1 = \to_2$, the notion of (local) commutation is called (local) confluence*; instead of satisfying the commuting diamond property, the relation is said to satisfy the diamond property.*

A standard topic in rewrite theory is reducing properties of a many-step rewrite relation to properties of the one-step relation, which are simpler to establish. We will need the following standard result, reducing commutation to the commuting-diamond property, respectively the property of strong commutation.

Lemma 9 *Let $\to_1$ and $\to_2$ be two relations.*

(1) If $\to_1$ and $\to_2$ have the commuting-diamond property, then they commute.
(2) If $\to_1$ and $\to_2$ strongly commute, then they commute.

To check the commuting-diamond property or strong commutation, still "rejoinability" of infinitely many pairs of elements — the elements t and u in the above diagrams — needs to be checked. Rejoining u and t is obviously possible in case they were derived from s by replacement of non-overlapping substrings

[3] Confluence is sometimes phrased equivalently as *Church-Rosser*-property [22, 59].

of s. This observation gives rise to the definition of *critical pairs* [50] (we show the variant for checking commutation in case of string rewriting):

Definition 10 (Critical pair) *Let R and S be string rewriting systems on Σ. Consider rewrite rules $(\alpha_R, \beta_R) \in R$ and $(\alpha_S, \beta_S) \in S$ such that α_R overlaps with α_S in the following way: either $\gamma_1\alpha_R = \alpha_S\gamma_2$ with $|\gamma_1| < |\alpha_S|$, or $\alpha_R = \gamma_1\alpha_S\gamma_2$, for some $\gamma_1, \gamma_2 \in \Sigma^*$. Then the corresponding* critical pair *is defined as $(\gamma_1\beta_R, \beta_S\gamma_2)$ in the first case and $(\beta_R, \gamma_1\beta_S\gamma_2)$ in the second.*

Now, in order to check whether $\rightarrow_R$ and $\rightarrow_S$ have the commuting-diamond property, it suffices to check, for every critical pair (δ_R, δ_S), whether there exists a δ such that $\delta_R S \delta$, and $\delta_S R \delta$. There is a similar condition in case strong commutation is used. So, in case that R and S are finite, there are also only finitely many critical pairs to check.

Rewrite theory offers several answers to the question whether strings s and t are *congruent* under some system R, i.e. whether $s \leftrightarrow_R^* t$. The first answer is: If this system is confluent and terminating, then strings are congruent iff they rewrite to the same normal form. This obviously gives a procedure to determine congruence. Being a special case of commutation, confluence of R can be checked using Lemma 9, by inspecting critical pairs. In case R turns out to be not confluent, still not all hope is lost. The next, more advanced technique offered by rewrite theory is to try to turn the rewrite system R into an equivalent rewrite system that is confluent, using so-called Knuth-Bendix *completion* [50]; we refer to [7] for details.

3 Sound Quotienting of $\mathcal{T}^{<\omega}$

Next we formalize the equivalence relation used to quotient $\mathcal{T}^{<\omega}$ and show the correctness of the construction. As illustrated in Section 1, the key intuition behind a sound quotient is that, whenever identifying states q_1 and q_2, there must *exist* a state realizing q_1's future and q_2's past, and a state realizing q_1's past and q_2's future. "Having the same future resp. past" will be captured by *being future resp. past bisimilar*. To ensure the existence of both required states, we will restrict our attention to *swapping* future and past bisimulations:

Definition 11 (Swapping) *Two relations R and S over the same set* swap *(or:* are swapping*), if $R; S = S; R$, where ";" denotes relational composition.*

In order to prove soundness of the construction, i.e., preservation of the transduction semantics, we need to characterize the reductions realized by a quotient of $\mathcal{T}^{<\omega}$. As $\mathcal{T}^{<\omega}_{/\cong}$ is given by identifying states of $\mathcal{T}^{<\omega}$ while retaining the reduction relation of $\mathcal{T}^{<\omega}$ (modulo the collapsing of states), the possible

reductions steps of $\mathcal{T}^{<\omega}_{/\cong}$ are either reduction steps from $\mathcal{T}^{<\omega}$ or steps replacing a word by a $\cong$-congruent one.

Lemma 12 *Let $\mathcal{T} = (Q, Q_i, Q_f, \Sigma, R) : \Sigma \to \Sigma$ be a finite-state transducer and furthermore, $\cong$ a congruence on Q^*. In abuse of notation we use the same symbol for the homomorphic lifting of $\cong$ to words over $\Sigma^* Q^* \Sigma^*$. Then $[t_1]_{\cong} \to^*_{\mathcal{T}^{<\omega}_{/\cong}} [t_2]_{\cong}$ iff $t_1 \, (\to_{\mathcal{T}^{<\omega}} \cup \cong)^* \, t_2$, for all words $t_1, t_2 \in \Sigma^* Q^* \Sigma^*$.*

Note that in the definition of $\mathcal{T}^{\leq n}$ as $\bigcup_{0 \leq i \leq n} \mathcal{T}^i$, the unions are all disjoint, if $Q \neq \emptyset$. In this case, it is straightforward to see that the construction is indeed sound (without further mention, we will assume $Q \neq \emptyset$ for the rest of the paper):

Lemma 13 *Let $\mathcal{T} = (Q, Q_i, Q_f, \Sigma, R)$ be a finite-state transducer. Then for all $t_1, t_2 \in \Sigma^*$,*

(1) $t_2 \in [\![\mathcal{T}^{\leq n}]\!](t_1)$ iff there exists a $k \leq n$ such that $t_2 \in [\![\mathcal{T}^k]\!](t_1)$, and
(2) $t_2 \in [\![\mathcal{T}^{<\omega}]\!](t_1)$ iff there exists a $k \in \omega$ such that $t_2 \in [\![\mathcal{T}^k]\!](t_1)$.

We are now ready to formulate the section's central result, which allows to collapse the infinite $\mathcal{T}^{<\omega}$ to a possibly finite transducer without changing its semantics. In general, identifying states allows more derivations. In order to obtain a finite quotient, we need to collapse states with different transduction semantics, or here more loosely states not being future bisimulation equivalent. Note again that identifying two states q_1 and q_2 realizing a different transduction semantics, will *individually* lead to more transductions, i.e., the equivalence class into which two different states are collapsed shows in general more derivations than the two states individually. In Fig. 2, for instance, the states 01 are 12 realize different transductions and the state obtained by identifying them obviously shows more behaviour. The crucial point is that *overall* the quotient of $\mathcal{T}^{<\omega}$ does not show more bevaviour, since requiring 01 $F;P$ 12 and 01 $P;F$ 12 assures that there exists states that realize the behaviour of the collapsed state.

Theorem 14 *Let $\mathcal{T}$ be a transducer, and F and P a swapping pair of a future and a past bisimulation on $\mathcal{T}^{<\omega}$. Then the quotient $\mathcal{T}^{<\omega}_{/_{F;P}}$ of $\mathcal{T}^{<\omega}$ under $F;P$ is well-defined and preserves the transduction relation, i.e., $[\![\mathcal{T}^{<\omega}_{/_{F;P}}]\!] = [\![\mathcal{T}^{<\omega}]\!]$.*

PROOF. First note that $F;P$ is a congruence, thus the quotient $\mathcal{T}^{<\omega}_{/_{F;P}}$ is well-defined. In the following we will write more suggestively $\equiv_{F;P}$ for this congruence; likewise we will use $\equiv_F$ and $\equiv_P$ for F and P.

For semantical equality, we need to show that $t' \in [\![\mathcal{T}^{<\omega}]\!](t)$ iff $t' \in [\![\mathcal{T}^{<\omega}_{/_{\equiv_{F;P}}}]\!](t)$, for all words t and t' from Σ^*. The "only-if"-direction is immediate: by identi-

fying states, the quotient $\mathcal{T}^{<\omega}_{/\equiv_{F;P}}$ can at least derive everything $\mathcal{T}^{<\omega}$ can. For the "if"-direction, assume $t' \in [\![\mathcal{T}^{<\omega}_{/\equiv_{F;P}}]\!](t)$, i.e., $q_{0^k} t \rightarrow^*_{\mathcal{T}^{<\omega}_{/\equiv_{F;P}}} t'$ for some words t and t' and a natural number k. By Lemma 12, this means $q_{0^k} t \ (\rightarrow_{\mathcal{T}^{<\omega}} \cup \equiv_{F;P})^* \ t'$. We generalize the proof obligation to:

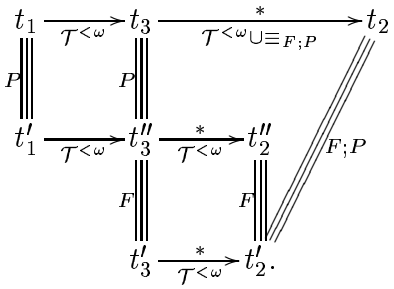
> for all $t_1, t_2 \in \Sigma^* Q^* \Sigma^*$: if $t_1 (\rightarrow_{\mathcal{T}^{<\omega}} \cup \equiv_{F;P})^* \ t_2$, then there exist words t'_1 and t'_2 such that $t'_1 \rightarrow^*_{\mathcal{T}^{<\omega}} t'_2$, and furthermore $t_1 \equiv_{F;P} t'_1$ and $t_2 \equiv_{F;P} t'_2$,

and proceed by induction on the length of the reduction $t_1 \ (\rightarrow_{\mathcal{T}^{<\omega}} \cup \equiv_{F;P})^* \ t_2$.

Base step: $t_1 = t_2$
Immediately by $t_1 \rightarrow^0_{\mathcal{T}^{<\omega}} t_1 = t_2$ and using reflexivity of $\equiv_{F;P}$.

Induction step: $t_1 (\rightarrow_{\mathcal{T}^{<\omega}} \cup \equiv_{F;P}) \ t_3 \ (\rightarrow_{\mathcal{T}^{<\omega}} \cup \equiv_{F;P})^* \ t_2$
By induction, there exist words t'_3 and t'_2 such that $t'_3 \rightarrow^*_{\mathcal{T}^{<\omega}} t'_2$, and where furthermore $t_3 \equiv_{F;P} t'_3$ and $t_2 \equiv_{F;P} t'_2$. Depending on the first step from t_1 to t_3, we distinguish two subcases. The first one where $t_1 \equiv_{F;P} t_3$ follows straightforwardly with transitivity of $\equiv_{F;P}$.

Subcase: $t_1 \rightarrow_{\mathcal{T}^{<\omega}} t_3$
In the following, we will silently use the fact that the properties of the relations $\equiv_F$, $\equiv_P$, and $\equiv_{F;P}$ carry over when lifted onto words. By $t_3 \equiv_{F;P} t'_3$ there exists a word t''_3 such that $t_3 \equiv_P t''_3 \equiv_F t'_3$ (remember that $\equiv_F$ and $\equiv_P$ are assumed to be swappable). Since $\equiv_P$ is a past bisimulation, there exists a word t'_1 such that $t'_1 \rightarrow_{\mathcal{T}^{<\omega}} t''_3$ and $t_1 \equiv_P t'_1$. From $t''_3 \equiv_F t'_3$, we obtain with Lemma 2 that $t''_3 \rightarrow^*_{\mathcal{T}^{<\omega}} t''_2$ for some word t''_2 with $t''_2 \equiv_F t'_2$. By reflexivity of $\equiv_F$ and $\equiv_P$ on words, we further get $t_1 \equiv_{F;P} t'_1$ and $t''_2 \equiv_{F;P} t'_2$, which finally implies with transitivity of $\equiv_{F;P}$ on words that $t_2 \equiv_{F;P} t''_2$:



To see that the theorem follows from the property proven, use Lemma 13 and specialize t_1 resp. t_2 to $q_0 \tilde{t}_1$ resp. $\tilde{t}_2 q_f$ where $\tilde{t}_1, \tilde{t}_2 \in \Sigma^*$ and where furthermore $q_0 \in Q_i$ and $q_f \in Q_f$. For t'_1 we know $t_1 = q_0 \tilde{t}_1 \equiv_{F;P} t'_1$, which means $t'_1 = q_\alpha \tilde{t}'_1$ for some state $q_\alpha \in Q^*$ and $\tilde{t}'_1 \in \Sigma^*$. Since $q_0^k \equiv_{F;P} q_\alpha$, the equivalence class $[q_\alpha]_{\equiv_{F;P}}$ is an initial state of $\mathcal{T}^{<\omega}_{/\equiv_{F;P}}$. The argument for $t_2 = \tilde{t}_2 q_f \equiv_{F;P} t'_2$ is analogous. □

Note that the theorem covers collapsing $\mathcal{T}^{<\omega}$ with respect to $\sim_f$ or else with respect to $\sim_p$ as special cases, since the identity relation on Q^* is a past as well as a future bisimulation and moreover, as neutral element of relational composition, swaps with every relation. Using both future and past bisimulation in combination is more general and allows for a smaller quotient than using any of the relations in isolation.

4 An On-the-fly Algorithm for Quotienting $\mathcal{T}^{<\omega}$

The previous result explains under which conditions quotienting $\mathcal{T}^{<\omega}$ is sound, but does not give guidance how to find the bisimulations nor how to algorithmically determine the quotient $\mathcal{T}^{<\omega}_{/\equiv_{F;P}}$. To make algorithmic use of the quotienting result, we must be able to effectively compute and represent swapping bisimulation relations on the infinite $\mathcal{T}^{<\omega}$. In this section, we show how to obtain these by extrapolating from information established on a finite approximant $\mathcal{T}^{\leq n}$, and exploiting the structure of $\mathcal{T}^{<\omega} = \mathcal{T}^0 \cup \mathcal{T}(\mathcal{T}^0) \cup \mathcal{T}(\mathcal{T}(\mathcal{T}^0)) \cup \ldots$.

To apply Theorem 14 we must extrapolate two properties: 1) the (future or past) bisimulation requirement, and 2) the property of swapping. In order to do the extrapolation, we will view the relations F and P on $Q^{\leq n}$ as rewriting systems on Q^*, indeed a restricted form of ground (i.e., without variables) rewriting systems on strings.

We start with the first question from above. As mentioned in Section 2.1, the future and past bisimulations are congruences over the monoid Q^*. This allows to extend bisimulations F and P from a finite approximant to $\mathcal{T}^{<\omega}$.

Lemma 15 *Let $\mathcal{T}$ be a finite-state transducer with states Q, and F and $P \subseteq Q^{\leq n} \times Q^{\leq n}$ be a future and a past bisimulation on $\mathcal{T}^{\leq n}$, with $n \geq 0$. Then the relation $\leftrightarrow\!\!\!\!\rightarrow^*_F$, resp. $\leftrightarrow\!\!\!\!\rightarrow^*_P$, is a future, resp. a past, bisimulation on $\mathcal{T}^{<\omega}$.*

Indeed, we can tighten Lemma 15 by loosening the requirements for the relations F and P on $\mathcal{T}^{\leq n}$: It suffices that their congruence-closure, projected on the finite approximant $\mathcal{T}^{\leq n}$, are bisimulations. This allows for smaller representations for the bisimulation relations in the algorithm.

Lemma 16 *Let $\mathcal{T}$ be a finite-state transducer with states Q, and F and $P \subseteq Q^{\leq n} \times Q^{\leq n}$ two string rewriting systems such that $\leftrightarrow\!\!\!\!\rightarrow^*_F \cap (Q^{\leq n} \times Q^{\leq n})$ and $\leftrightarrow\!\!\!\!\rightarrow^*_P \cap (Q^{\leq n} \times Q^{\leq n})$ is a future resp. a past bisimulation on $\mathcal{T}^{\leq n}$. Then the relation $\leftrightarrow\!\!\!\!\rightarrow^*_F$ (resp. $\leftrightarrow\!\!\!\!\rightarrow^*_P$) is a future (resp. a past) bisimulation on $\mathcal{T}^{<\omega}$.*

Having extended the finite bisimulations F and P to $\mathcal{T}^{<\omega}$ by congruence closure, the second question is whether $\leftrightarrow\!\!\!\!\rightarrow^*_F$ and $\leftrightarrow\!\!\!\!\rightarrow^*_P$ additionally enjoy the

swapping requirement. Note that the notion of swapping of relations from Definition 11 is closely related to that of commutation and clearly two *symmetric* relations R and S commute iff R^* and S^* swap. This together with Lemma 9 implies:

Lemma 17 *Let F and P be two relations on $Q^{\leq n} \times Q^{\leq n}$.*

(1) If $\leftrightarrow_F$ and $\leftrightarrow_P$ have the commuting diamond property, then $\leftrightarrow_F^*$ and $\leftrightarrow_P^*$ swap.
(2) If $\leftrightarrow_F$ and $\leftrightarrow_P$ strongly commute, then $\leftrightarrow_F^*$ and $\leftrightarrow_P^*$ swap.

To effectively identify cases where the (infinite) relations $\leftrightarrow_F$ and $\leftrightarrow_P$ have the commuting-diamond property, one can restrict attention to the critical pairs, of which there are only finitely many as the rewrite systems F and P are finite.

Lemma 15 and Lemma 17 together allow now to apply the quotienting Theorem 14 and do the desired extrapolation.

Corollary 18 (Soundness) *Let $\mathcal{T}$ be a transducer with states Q, and F and $P \subseteq Q^{\leq n} \times Q^{\leq n}$ a future resp. a past bisimulation on $\mathcal{T}^{\leq n}$, with $n \geq 0$. If $\leftrightarrow_F$ and $\leftrightarrow_P$ have the commuting-diamond property (or they strongly commute), then $[\![\mathcal{T}^{<\omega}]\!] = [\![\mathcal{T}^{<\omega}_{/\leftrightarrow_P^*;\leftrightarrow_F^*}]\!]$.*

To make notation a little less heavy-weight, we will for the rest use $\equiv$ to abbreviate the congruence relation $\leftrightarrow_F^*;\leftrightarrow_P^*$.

Let us illustrate the ideas so far on the transducer from Fig. 1. On the approximant $\mathcal{T}_\alpha^{\leq 2}$ (i.e. the union of the transducers in parts (a) and (b) of Fig. 1), one pair of a future and a past bisimulation (represented as rewriting systems) is $F = \{(12,1),(1,12),(22,2),(2,22)\} \cup Id_{\{0,1,\ldots,22\}}$ and $P = \{(00,0),(0,00),(01,1),(1,01)\} \cup Id_{\{0,1,\ldots,22\}}$, where Id_S denotes the "identity rewrite system" on S. Indeed, these bisimulations are the largest possible choices. It can be easily checked that the corresponding rewrite relations $\leftrightarrow_F$ and $\leftrightarrow_P$ have the commuting-diamond property. For example, the overlapping pair consisting of state 1 from the rule $(1,12)$ of F and state 1 from the rule $(1,01)$ of P opens a diamond that may be closed again by rewriting both 12 and 01 to 012 (using the same rules).

Now, without actually attempting to fully compute the relation $\equiv$, we can already detect several equivalences between states. Most importantly, the states 1, 01, and 12 belong to the same equivalence class. Furthermore, we have $00 \equiv 0$ and $22 \equiv 2$. Quotienting $\mathcal{T}_\alpha^{\leq 2}$ by this equivalence gives the transducer of Fig. 3, where only the relevant part is shown (i.e., the equivalence classes $[10]_\equiv$, $[11]_\equiv$, $[20]_\equiv$, and $[21]_\equiv$ are left away). It can be checked that the construc-

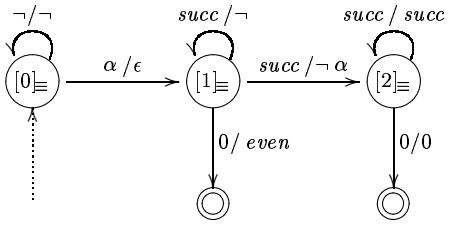


Fig. 3. Transducer $\mathcal{T}_\alpha^*$

tion stabilizes at this point, so we have arrived at $\mathcal{T}_\alpha^*$. Note that quotienting $\mathcal{T}_\alpha^{<\omega}$ using $\sim_p$ or $\sim_f$ in isolation does not give a finite quotient.

The algorithm based on these ideas is sketched in pseudo-code in Fig. 4. Given a transducer $\mathcal{T} = (Q, Q_i, Q_f, \Sigma, R)$, the *until*-loop iteratively calculates, in variable $\mathcal{X}$, the approximations $\mathcal{T}^{\leq n}$. On each approximation, bisimulations F and P are computed by a partition refinement algorithm [62, 34].

---

input $\mathcal{T} = (Q, Q_i, Q_f, \Sigma, R)$
$\mathcal{X} := \mathcal{T}_{id}$;
repeat
  $\mathcal{X} := (\mathcal{T} \circ \mathcal{X}) \cup \mathcal{T}_{id}$;
  determine bisimulations F and P on $\mathcal{X}$ s.t.
    $\nrightarrow_F$ and $\nrightarrow_P$ swap and each possess the diamond property;
until $\mathcal{X}_{/\equiv} \sim_f (\mathcal{T} \circ \mathcal{X}_{/\equiv}) \cup \mathcal{T}_{id}$

Fig. 4. Calculating $\mathcal{T}^*$

Note that in the termination condition, the approximant transducer $\mathcal{X}$ is quotiented using the whole equivalence $\equiv \; = \; \nrightarrow_F^*; \nrightarrow_P^*$, and not just by those identifications that happen to be directly detectable on $\mathcal{X}$, as suggested in the example above. The ability to do so relies again on techniques from rewrite theory. First, it can be shown that $\nrightarrow_F^*; \nrightarrow_P^* = (\nrightarrow_F \cup \nrightarrow_P)^* = \nrightarrow_{F \cup P}^*$. So, the question is when strings are congruent under the rewrite system $F \cup P$. For this, we would like to have a confluent and terminating system. Confluence of $F \cup P$ can be checked by inspecting critical pairs. In practice, we can avoid duplicating work by the following standard result.

Lemma 19 *If $\nrightarrow_F$ and $\nrightarrow_P$ commute, then $\nrightarrow_F \cup \nrightarrow_P$ is confluent if each of $\nrightarrow_F$ and $\nrightarrow_P$ in separation is confluent.*

So, if commutation of $\nrightarrow_F$ and $\nrightarrow_P$ has already been checked when determining whether $\nrightarrow_F^*$ and $\nrightarrow_P^*$ swap, then it suffices to check confluence of the individual relations. In case $\nrightarrow_F \cup \nrightarrow_P$ turns out to be not confluent, Knuth-Bendix completion might be needed.

As for checking termination — it is clear that the relations F and P in sepa-

ration are already non-terminating, as they are reflexive and symmetric. But also in this case, there is the possibility of turning $F \cup P$ into an equivalent system that does terminate. Because of the very simple form of this rewriting system —ground rewriting on strings— it is easy to capture $\leftrightarrow^*_{F \cup P}$ by a terminating one: Just order pairs *lexicographically* and remove the "reflexive" part $Id_{Q^{\leq n}}$. In our example, the quotienting relation $\equiv$ can in this way be represented by the four rules $\{(00, 0), (01, 1), (21, 1), (22, 2)\}$, where the right-hand side of each rule is strictly smaller than the corresponding left-hand side in lexicographic order.

A few points concerning the implementation deserve mention. For once, the naive iteration as sketched in the pseudo-code can be optimized in a number of ways, especially by reusing information collected from the lower approximants when treating $\mathcal{T}^{\leq n+1}$. For instance, in case one knows already that $(00, 0)$ are past bisimilar after investigating the first two levels, as in our example, there is no need to check $(000, 00)$ for past-bisimilarity at the third (if at all it would be needed to construct that level). Another, more tricky point is that the search for bisimulations F and P under the additional requirements of swapping and confluence, adds an element of *non-determinism* to the process. Namely, it may be that bisimulations as they are found do not swap or are not confluent, but that smaller bisimulations would in fact satisfy these requirements. In such a case we would have to *choose* which pairs of states to delete from the bisimulation relations. However, in the examples we tested, the largest bisimulations $\sim_f$ and $\sim_p$, as given by the partition refinement, always worked.

We tested our implementation on various examples, for instance the one of Fig. 1 and the token array example of [48]. In all but one case, the transitive closure was computed in a short time on a standard desktop workstation. In the remaining case, a *ring* configuration of the token array, the computation took too long. We expect that by implementing some additional optimizations (see below), larger transducers can be successfully handled.

5 Related Work

5.1 Transductions and their iteration

Regular or rational transductions were introduced by Elgot and Mezei in [31] as a natural extension of regular languages and used to represent and reason about computations. Much of the theory of rational transductions has been developed by Schützenberger ([66]), Eilenberg ([30]), and Nivat ([61]); overviews can be found in [30, 8]. [30] contains results, presented in an al-

gebraic setting, that relate transducers to concepts such as *rational relations* and the *(generalized) sequential machines (GSM)* and *GSM mappings* of [39]. Other closely related devices are *Mealy machines* ([56]) and *Moore machines* ([58]). Closure properties of classes of languages under GSM mappings have been studied, see e.g. [44]. [30] also contains several properties of the class of *length-preserving* rational relations. One result worth mentioning is that any length-preserving rational relation can be realized by a "letter-to-letter" transducer or nondeterministic Mealy machine.

In research on the *iteration* of transducers (transductions, GSMs), two directions can be distinguished. One class of papers studies the results of closing classes like free monoids and regular languages under the iterated application of transducers. For example, [71] shows that every recursively enumerable language can be obtained by iterating a so-called GSM relation (corresponding to some nondeterministic machine) on a singleton language — thus demonstrating the power of iteration. In [51], the restriction to length-preserving transductions is considered, and it is shown (among other things) that the iteration of length-preserving rational transductions on a free monoid yields essentially a context-sensitive language. [54] shows that the number of states of a transducer used to iterate in this way can be limited to 4: Additional states do not add extra expressiveness of the iteration, while each number smaller than 4 gives rise to known subclasses. An interesting related development suggested by DNA computing is "computing by carving" ([54]).

5.2 Regular model checking

Often interested in language-theoretic expressiveness results and clarifying the connections between various classes and their computational models, the papers mentioned above focus on characterizing the smallest classes of languages closed under iterations of transducers or similar devices. Partly inspired by using regular languages and their transductions for symbolic model checking, another class of papers aims to identify restricted forms of transducers whose iteration on a regular language is still regular. The corresponding symbolic form of state exploration, sometimes called *regular model checking,* has recently attracted attention in the context of verification of various classes of infinite state systems. Examples are [49, 2, 60, 48, 15] and this paper. Other related papers include [64, 37, 29, 21, 67]. The work in [36] goes beyond the framework of regular languages and transductions by considering context-free grammars. See also [14] for a recent survey of the use of regular languages and rewriting for model checking.

Closest to our work is [15] which presents an algorithm for the iterated effect of a length-preserving transducer using standard determinization and mini-

mization techniques from automata theory. As in our work, the key of the construction is to collapse $\mathcal{T}^{<\omega}$, called column transducer in [15], into a hopefully finite quotient. Minimizing (and determinizing) this infinite transducer gives rise to a smaller one whose states are languages of states of $\mathcal{T}$. More specifically, states are regular languages and since $\mathcal{T}$'s edges are labeled by $\Sigma \times \Sigma$, its transitions transform regular languages into regular languages. The additional identifications can be seen as assuming a static, predefined set of equations on the words of Q^*. Starting from the initial state of $\mathcal{T}^{<\omega}$, which can be represented by the regular language q_0^*, the algorithm tries to compute the states of $\mathcal{T}^{<\omega}$ performing a forward symbolic reachability analysis (this is the determinization) while relaxing the condition stating when a state has already been visited. This relaxation (called saturation in their work) assumes a fixed set of equivalences between states of $\mathcal{T}^{<\omega}$. In contrast, our algorithm tries do discover such equations dynamically as it proceeds to higher n's when exploring $\mathcal{T}^{\leq n}$. The restriction to length-preserving transductions and the static nature of their saturation process allow Bouajjani et al. to provide sufficient conditions on the form of the transducer to guarantee termination of the construction. The work has recently be extended to deal with regular tree languages in [3].

In [16], the notion of *alphabetic pattern constraints* is identified as a subclass of regular languages. The class is effectively closed under permutation rewriting, a restricted form of rewriting or transduction. [38] considers the closure of regular tree languages under general term rewriting. Also related is the work of [52], where the closure of a set of process-algebra terms under iterated predecessor and successor operations is considered and where the sets of terms are viewed as a regular tree language (also see [33]).

A common source of infinity in the state space are various forms of data types like stacks [35], message queues [12, 10, 11], or integers [9]. The regularity of these data structures and the restricted form of transitions can often be exploited by symbolic methods based on regular languages and their transformation. See [70] for a survey of various approaches in this direction. To explore the symbolically represented infinity of states in a finite amount of steps, often the effect of single steps or cycles needs to be *accelerated,* i.e., all iterations of a loop are explored at once in a so-called *meta-transition.* In contrast, our procedure approximating $\mathcal{T}^{<\omega}$ achieves an accelerating affect by identifying states in a finite prefix $\mathcal{T}^{\leq n}$. Through the cycles introduced by identifying states which are semantically not equivalent within $\mathcal{T}^{\leq n}$, the procedure (if it halts) allows to explore the effect of the infinite limit $\mathcal{T}^{<\omega}$ in a finite amount of steps.

Future and past simulations including their combinations have been extensively studied in [53]. We are not aware of any work that considers the composition of swapping such relations as is done in this paper.

6 Conclusions and Future Work

We have presented a partial algorithm for computing the transitive closure of regular word transducers. This algorithm allows to reason about the effect of iterating transduction relations an unbounded number of times. Such relations are used, for instance, in regular model checking where they represent the transition relation of an infinite-state system. Given a transducer $\mathcal{T}$, our algorithm is based on quotienting, w.r.t. the composition of a future and a past bisimulation, the possibly infinite-state transducer $\mathcal{T}^{<\omega}$, the union of all finite compositions of $\mathcal{T}$. To be able to develop our algorithm, we presented sufficient conditions that allow to exploit bisimulations discovered on a finite approximant $\mathcal{T}^{\leq n}$, and hence, to avoid constructing $\mathcal{T}^{<\omega}$. Though our prototype implementation can be improved in several ways, we obtained encouraging results on the examples we have considered.

In order to compute $\mathcal{T}^*(S)$ for a given regular set S, our results specialize to automata, allowing to accelerate the computation of $\mathcal{T}^{\leq 0}(S)$, $\mathcal{T}^{\leq 1}(S)$, $\mathcal{T}^{\leq 2}(S)$, ... This problem, where the set of initial configurations is also a parameter of the algorithm, can be solved in more cases than the general case.

A pair of swapping bisimulations played a crucial role in our development. As the concept is a rather general, we work on applying it in other contexts than transducers, as well. Furthermore, properties of swapping bisimulations deserve a study in their own right.

A natural question is whether our heuristic constitutes a semi-algorithm: In those cases that $[\![\mathcal{T}^{<\omega}]\!]$ is representable as a finite-state transducer, will our algorithm find such a transducer in a finite amount of time? This is not the case, since the equivalence used for the quotient construction, being based on bisimulations, is finer than the language equivalence (or, stricly speaking, transduction equivalence) that is ultimately needed for semantic equivalence.

Besides the improvements mentioned in Section 4 and implementation improvements like using BDDs [18, 55, 19] to represent transducers, we believe that there are variations of our algorithm that are worth studying. One such variation consists in computing at each iteration of the algorithm the composition of $\mathcal{T}$ with the quotiented transducer obtained up to that iteration. This would reduce the number of states of the transducers that occur as intermediate results of the algorithm. A similar idea underlies what is called compositional model-checking, e.g. [41]. The difficulty in our context lies in the generalization of the computed bisimulations to $\mathcal{T}^{<\omega}$.

The construction presented in this paper and the notion of bisimulation is rather general. Hence it is worthwhile to study, whether and how the results can be transferred to more general classes of transducers, for instance prefix

recognizable transducers [20], or even to other models of computation, for instance communicating processes. We are currently extending our results to the case of tree transducers. Here, in the general case, one is confronted with negative results from tree transducer theory, the main one being that regular tree transducers are not closed under composition. To avoid this problem, we restrict ourselves to linear tree transducers. A preliminary account, which also provides the full proofs for the word case, can be found in [24].

Acknowledgements

We thank the anonymous referees for their careful work and insightful remarks. Furthermore thanks to Kedar Namjoshi, Kai Baukus, and Karsten Stahl for inspiring discssions and suggestions, and careful reading of earlier versions of this document.

References

[1] P. A. Abdulla, A. Bouajjani, and B. Jonsson. On-the-fly analysis of systems with unbounded lossy Fifo-channels. In Hu and Vardi [45], pages 305–318.

[2] P. A. Abdulla, A. Bouajjani, B. Jonsson, and M. Nilsson. Handling global conditions in parameterized system verification. In Halbwachs and Peled [43], pages 134–145.

[3] P. A. Abdulla, P. Mahata, and J. d'Orso. Regular tree model checking. unpublished, 2001.

[4] ACM. *27th Annual Symposium on Principles of Programming Languages (POPL) (Boston, MA)*, Jan. 2000.

[5] R. Alur, editor. *CAV '96, Proceedings of the 8th International Conference on Computer-Aided Verification, New Brunswick, NJ*, volume 1102 of *Lecture Notes in Computer Science*, 1996.

[6] A. Arnold, editor. *Trees in Algebra and Programming (CAAP '90)*, volume 431 of *Lecture Notes in Computer Science*. Springer-Verlag, 1990.

[7] F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.

[8] J. Berstel. *Transductions and Context-Free Languages*. B. G. Teubner, 1979.

[9] B. Boigelot. *Symbolic Methods for Exploring Infinite State Spaces*. PhD thesis, Université de Liège, 1998.

[10] B. Boigelot and P. Godefroid. Symbolic verification of communication protocols with infinite state spaces using QDDs. In Alur [5], pages 1–12.

[11] B. Boigelot, P. Godefroid, B. Willems, and P. Wolper. The power of QDDs. In van Hentenryck [68], pages 172–186.

[12] B. Boigelot and P. Wolper. Symbolic verification with periodic sets. In Dill [28], pages 55–67.

[13] R. Book and F. Otto. *String Rewriting Systems*. Monographs in Computer Science. Springer-Verlag, 1993.

[14] A. Bouajjani. Languages, rewriting systems, and verification of infinite-state systems. In ICALP2001 [46], pages 24–39.

[15] A. Bouajjani, B. Jonsson, M. Nilsson, and T. Touili. Regular model checking. In Emerson and Sistla [32], pages 135–145.

[16] A. Bouajjani, A. Muscholl, and T. Touili. Permutation rewriting and algorithmic verification. In LICS'01 [47].

[17] L. Brim, J. Gruska, and J. Zlatuska, editors. *International Symposium on Mathmatical Foundations of Computer Science (MFCS)*, volume 1450 of *Lecture Notes in Computer Science*. Springer-Verlag, 1998.

[18] R. E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, 40(2), 1986.

[19] J. R. Burch, E. M. Clarke, D. E. Long, K. L. McMillan, and D. L. Dill. Symbolic model checking: 10^{20} states and beyond. *Information and Computation*, 98(2):142–170, 1992.

[20] O. Burkhart. Model checking rationally restricted right closures of recognizable graphs. In *Methods and Tools for the Verification of Infinite State Systems, Grenoble*, March 1997.

[21] D. Caucal. On the regular structures of prefix rewriting. In Arnold [6], pages 87–102.

[22] A. Church and J. B. Rosser. Some properties of conversion. *Transactions of the American Mathematical Society*, 39:472–482, 1936.

[23] E. M. Clarke and R. P. Kurshan, editors. *Computer Aided Verification 1990*, volume 531 of *Lecture Notes in Computer Science*. Springer-Verlag, 1991.

[24] D. Dams, Y. Lakhnech, and M. Steffen. Iterating transducers. Technical Report TR-ST-01-03, Lehrstuhl für Software-Technologie, Institut für Informatik und praktische Mathematik, Christian-Albrechts-Universität Kiel, Jan. 2001. A preliminary version is available on-line at `http://radon.ics.ele.tue.nl/~vires/public/results.htm` (reports section).

[25] J. W. de Bakker, K. Huizing, and W.-P. de Roever, editors. *Real-Time: Theory in Practice (REX Workshop)*, volume 600 of *Lecture Notes in Computer Science*. Springer-Verlag, 1991.

[26] N. Dershowitz and J.-P. Jouannaud. Rewrite systems. In van Leeuwen [69], pages 244–320. Also as research report 478, LRI.

[27] P. Deussen, editor. *Fifth GI Conference on Theoretical Computer Science*, volume 104 of *Lecture Notes in Computer Science*. Springer-Verlag, 1981.

[28] D. Dill, editor. *Computer Aided Verification (Proc. of CAV'94)*, volume 818 of *Lecture Notes in Computer Science*. Springer-Verlag, 1994.

[29] M. Duflot, L. Fribourg, and U. Nilsson. Unavoidable configurations of parametrized rings of processes. Research Report LSV-00-10, Laboratoire Spécification et Vérification, Ecole Normale Supérieure de Cachan, Nov. 2000.

[30] S. Eilenberg. *Automata, Languages and Machines*, volume A. Academic Press, New York, 1974.

[31] C. C. Elgot and J. E. Mezei. On relations defined by generalized finite automata. *Journal of Research and Development*, 9:47–68, 1965.

[32] E. A. Emerson and A. P. Sistla, editors. *CAV '00, Proceedings of the 12th International Conference on Computer-Aided Verification, Chicago IL*, volume

1855 of *Lecture Notes in Computer Science*. Springer-Verlag, 2000.

[33] J. Esparza and A. Podelski. Efficient algorithms for pre* and post* on interprocedural parallel flow graphs. In POPL'00 [4], pages 1–11.

[34] J. Fernandez. An implementation of an efficient algorithm for bisimulation equivalence. *Science of Computer Programming*, 13:219–236, 1989.

[35] A. Finkel, B. Willems, and P. Wolper. A direct symbolic approach to model checking pushdown systems. *Electronic Notes in Theoretical Computer Science*, Aug. 1997. Presented at Infinity '97.

[36] D. Fisman and A. Pnueli. Beyond regular model checking. In *Procedings of the 21th Conference on the Foundations of Software Technology and Theoretical Computer Science*, Lecture Notes in Computer Science, 2001. to appear.

[37] L. Fribourg and H. Olsén. Reachability sets of parametrized rings as regular languages. *Electronic Notes in Theoretical Computer Science*, 9, 2000. Also published as part of the Proceedings of the 2nd International Workshop on Verification of Infinite State Systems (INFINITY'97), Bologna, Italy, July 1997.

[38] R. Gilleron and S. Tison. Regular tree languages and rewrite systems. *Fundamenta Informaticae*, 24:157–175, 1995.

[39] S. Ginsburg. *The Mathematical Theory of Context-Free Languages*. McGraw-Hill, 1966.

[40] S. Graf and M. Schwartzbach, editors. *Proceedings of the Sixth International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2000)*, volume 1785 of *Lecture Notes in Computer Science*. Springer-Verlag, 2000.

[41] S. Graf and B. Steffen. Compositional minimization of finite state systems. In Clarke and Kurshan [23].

[42] O. Grumberg, editor. *CAV '97, Proceedings of the 9th International Conference on Computer-Aided Verification, Haifa. Israel*, volume 1254 of *Lecture Notes in Computer Science*. Springer, June 1997.

[43] N. Halbwachs and D. Peled, editors. *CAV '99: Computer Aided Verification*, volume 1633 of *Lecture Notes in Computer Science*. Springer-Verlag, 1999.

[44] J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.

[45] A. J. Hu and M. Y. Vardi, editors. *Computer-Aided Verification, CAV '98, 10th International Conference, Vancouver, BC, Canada, Proceedings*, volume 1427 of *Lecture Notes in Computer Science*. Springer-Verlag, 1998.

[46] *28th Colloquium on Automata, Languages and Programming (ICALP)*, volume 2076 of *Lecture Notes in Computer Science*. Springer-Verlag, July 2001.

[47] IEEE. *Sixteenth Annual Symposium on Logic in Computer Science (LICS) (Boston, MA)*. Computer Society Press, June 2001.

[48] B. Jonsson and M. Nilsson. Transitive closures of regular relations for verifying infinite-state systems. In Graf and Schwartzbach [40].

[49] Y. Kesten, O. Maler, M. Marcus, A. Pnueli, and E. Shahar. Symbolic model checking with rich assertional languages. *Theoretical Computer Science*, 256:93–112, 2001. A conference version appeared in [42].

[50] D. E. Knuth and P. B. Bendix. Simple word problems in universal algebra. In J. Leech, editor, *Computational Problems in Abstract Algebra*, pages 263–297. Pergamon Press, 1970.

[51] M. Latteux, D. Simplot, and A. Terlutte. Iterated length-preserving rational transductions. In Brim et al. [17], pages 286–295.

[52] D. Lugiez and P. Schnoebelen. The regular viewpoint on PA-processes. In Sangiorgi and de Simone [65], pages 50–66. A longer version appeared as INRIA Rapport de Recherche, Nr. 3403, April 1998.

[53] N. Lynch and F. Vaandrager. Forward and backward simulations for timed-based systems. In de Bakker et al. [25], pages 397–446.

[54] V. Manca, C. Martin-Víde, and G. Păun. Iterated GSM mappings: A collapsing hierarchy. TUCS Report 206, Turku Centre for Computer Science, 1998.

[55] K. L. McMillan. *Symbolic Model Checking: An Approach to the State Space Explosion Problem.* PhD thesis, Carnegie Mellon University, 1992.

[56] G. H. Mealy. A method for synthesizing sequential circuits. *Bell System Technical Journal*, 34(5):1045–1079, 1955.

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

[58] E. F. Moore. Gedanken experiments on sequential machines. *Automata Studies*, pages 129–153, 1956.

[59] M. H. A. Newman. On theories with a combinatorial definition of "equivalence". *Annals of Mathematics*, 43(2):223–243, 1942.

[60] M. Nilsson. Regular model checking, 2000. Licenciate Thesis of Uppsala University, Department of Information Technology.

[61] M. Nivat. Transductions des languages de Chomsky. *Ann. de l'Inst. Fourier*, 18:339–456, 1968.

[62] R. Paige and R. E. Tarjan. Three partition refinement algorithms. *SIAM Journal on Computing*, 16(6):973–989, 1987.

[63] D. Park. Concurrency and automata on infinite sequences. In Deussen [27], pages 167–183.

[64] A. Pnueli and E. Shahar. Liveness amd acceleration in parameterized verification. In Emerson and Sistla [32], pages 328–343.

[65] D. Sangiorgi and R. de Simone, editors. *CONCUR '98: Concurrency Theory (9th International Conference, Nice, France)*, volume 1466 of *Lecture Notes in Computer Science.* Springer-Verlag, Sept. 1998.

[66] M. P. Schützenberger. Sur les relations rationelles entre monoïdes libres. *Theoretical Computer Science*, pages 234–259, 1976.

[67] T. Touili. Regular model checking using widening techniques. In *Proceedings on the Workshop on Verification of Parametrized Systems (VEPAS'01), Crete*, volume 50 of *Electronic Notes in Theoretical Computer Science.* Elsevier Science Publishers, July 2001.

[68] P. van Hentenryck, editor. *Proceedings of the Fourth International Static Analysis Symposium (SAS '97, Paris, France)*, volume 1302 of *Lecture Notes in Computer Science.* Springer-Verlag, 1997.

[69] J. van Leeuwen, editor. *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantics. Elsevier, 1990.

[70] P. Wolper and B. Boigelot. Verifying systems with infinite but regular state spaces. In Hu and Vardi [45], pages 88–97.

[71] D. Wood. Iterated a-NGSM maps and Γ systems. *Information and Control*, 32(1):1–26, 1976.