Iterating transducers*

Dennis Dams[1], Yassine Lakhnech[2], and Martin Steffen[3]

[1] Bell Labs, Lucent Technologies, Murray Hill NJ 07974, USA
On leave from Eindhoven University of Technology, The Netherlands
d.dams@tue.nl
[2] Verimag, Centre Equation
2 av. de Vignate, 38610 Gières, France
lakhnech@imag.fr
[3] 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 transitional semantics of the system consequently can be modeled by finite-state transducers. A standard problem encountered when doing symbolic state exploration for infinite state systems is how to explore all states in a finite amount of time. When representing the one-step transition relation of a system by a finite-state transducer $\mathcal{T}$, this problem boils down to finding an appropriate finite-state representation $\mathcal{T}^*$ for its 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 that is based *past* and *future* bisimulations computed 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 that have been developed in the area of rewriting.
A prototype implementation has been successfully applied to various examples.

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

* This work has been supported by the Esprit-LTR project Vires.

about infinite-state systems, in which case a single automaton is used to represent an infinite set of system states. In regular model-checking [3, 8, 1, 7], regular sets of states of the system to be verified are represented by finite-state automata. For instance, consider a parameterized 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 size 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 $\mathcal{T}$ an arbitrary number of times. In other words, one needs to compute the transitive closure of $\mathcal{T}$. In general, this closure is not finite-state anymore. 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 [3, 7]. These algorithms can be explained in terms of the, in generalinfinite-state, transducer $\mathcal{T}^{<\omega} = \bigcup_{i\in\omega} \mathcal{T}^i$, the union of all finite compositions of $\mathcal{T}$. Conceptually, they attempt to construct a finite quotient of $\mathcal{T}^{<\omega}$ by identifying states that are equivalent in some way. For example, in [3, 7], the underlying equivalence relation is induced by determinizing on-the-fly and then minimizing $\mathcal{T}^{<\omega}$. General transducers are not determinizable, but that paper considers length-preserving transducers, which are essentially standard automata over pairs of symbols and can be determinized accordingly. The minimal automation is then approximated using a technique called saturation to approximate the minimal automaton.

In this paper, we employ a different quotient construction, 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 $n = 0, 1, 2, \ldots,$ 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.* The largest such equivalence, being a relation over the infinite-state transducer $\mathcal{T}^{<\omega}$, may not be effectively computable. 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 $\alpha(0) = even$ and $\alpha(succ(x)) = \neg(\alpha(x))$. It computes the parity *even* or *odd* of a number; $\neg$ is a function that toggles parities. Consider the transition relation $\rightarrowtail$ 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

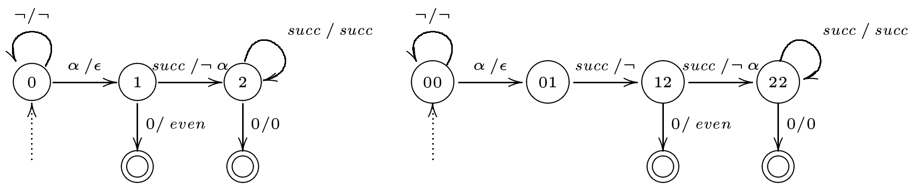

Fig. 1. Left (a): The transducer $\mathcal{T}_\alpha$. Right (b): Its product $\mathcal{T}_\alpha^2$.

(/) is used to separate the input symbol from the output symbols. 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.

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 (ϵ denotes the empty string). 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 (they are obviously semantically different in that the realize different transductions), i.e., what is the equivalence notion on $\mathcal{T}_\alpha^{<\omega}$ employed for quotienting, and secondly, how to compute the quotient without prior calculation of the infinite $\mathcal{T}_\alpha^{<\omega}$?

As for the first point, we must require 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 if 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 $\mathcal{T}_\alpha^1$. Another class of artificial runs 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 second question is how to know that two states in some approximant $\mathcal{T}_\alpha^{\leq n}$ are equivalent in the above sense, 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? For this 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, this means that we can conclude the existence of an appropriate state without actually having to construct $\mathcal{T}_\alpha^3$. Namely, by looking at $\mathcal{T}^{\leq 2}$ only, we see that 1 and 12 are future bisimilar, whence by congruence also 01 and 012. Similarly, past bisimilarity of 12 and 012 can be inferred by comparing 1 and 01. So we know that 012 is our candidate, without ever having constructed it in any approximant so far. In short, 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 forms the topic of Section 4, where we will also report on the results obtained from our prototype implementation. Section 5 concludes and discusses related and future work.

2 Preliminaries

A *transducer* $\mathcal{T} = (Q, Q_i, Q_f, \Sigma, R)$ consists of a set Q of states, sets $Q_i, Q_f \subseteq Q$ of initial resp. final states, a set Σ of symbols, and a set R of rules. A rule has the form $qa \twoheadrightarrow wq'$ with $q, q' \in Q$, $a \in \Sigma \cup \{\epsilon\}$, and $w \in \Sigma^*$, specifying that when in state q and reading input symbol $a \in \Sigma$ (or reading no input in case $a = \epsilon$), the transducer produces output w and assumes q' as its new state. A finite-state transducer is also called *regular*. The operation of a transducer is captured by the *reduction* relation $\twoheadrightarrow_R$ on strings consisting of symbols and a state (where ϵ has its usual meaning as neutral element of concatenation), defined as follows: For $t_1, t_2 \in \Sigma^*$, $t_1 qa t_2 \twoheadrightarrow_R t_1 wq' t_2$ iff $qa \twoheadrightarrow wq' \in R$. For this and other arrows we use common notations like $\twoheadrightarrow^{-1}$ for inverse, $\twoheadrightarrow^*$ for reflexive-transitive closure, and $\twoheadleftarrow\!\!\twoheadrightarrow$ for symmetric closure. $\mathcal{T}$'s *semantics* $[\mathcal{T}] : \Sigma^* \to 2^{(\Sigma^*)}$ is defined as follows: $t_2 \in [\mathcal{T}](t_1)$ iff there exist $q_i \in Q_i$ and $q_f \in Q_f$ such that $q_i t_1 \twoheadrightarrow_R^* t_2 q_f$. We will use the notation $\twoheadrightarrow_{\mathcal{T}}$ synonymously for the rewrite relation $\twoheadrightarrow_R$.

Transducers $\mathcal{T}_1$ and $\mathcal{T}_2$ over the same symbol set can be composed into $\mathcal{T}_2 \circ \mathcal{T}_1$, so that the output of $\mathcal{T}_1$ is input to $\mathcal{T}_2$. This is a standard product construction where the rules R of the composition are defined by

$$\frac{q_j a \twoheadrightarrow_{R_1} v q_j' \qquad q_i v \twoheadrightarrow^*_{R_2} w q_i'}{q_{ij} a \twoheadrightarrow w q_{ij}' \in R}$$

where R_1 and R_2 are the rules of the two constituent transducers, and where we write q_{ij} as short-hand for the tuple (q_i, 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., $[\![\mathcal{T}_2]\!] \circ [\![\mathcal{T}_1]\!] = [\![\mathcal{T}_2 \circ \mathcal{T}_1]\!]$. The n-fold composition of a transducer $\mathcal{T}$ with itself is written as$\mathcal{T}^n$, with $\mathcal{T}^0$ being defined such that it realizes the neutral element wrt. transduction composition, i.e., $[\![\mathcal{T}^0]\!] = Id_{\Sigma^*}$. 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 $\mathcal{T}$. Finally we will need the *union* of transducers: given two transducers $\mathcal{T}_1$ and $\mathcal{T}_2$ over the same signature, $\mathcal{T}_1 \cup \mathcal{T}_2$ denotes the transducer over the same signature, given by the union of states, of initial states, of final states, and of rules, respectively, where we assume that the sets of states are disjoint. Union can be easily extended to the union of countably many transducers. Note that finite union preserves finiteness.

To obtain a finite-state transducer out of an a-priori infinite $\mathcal{T}^{<\omega}$, we will have to identify "equivalent" states. The notion of equivalence used to this end will be based on *bisimulation* equivalences [12, 10] on states. Besides the standard future bisimulation we need the *past* variant as well.

Definition 1 (Bisimulation). *Let $\mathcal{T} = (Q, Q_i, Q_f, \Sigma, R)$ be a transducer. An equivalence relation $F \subseteq Q \times Q$ is a* future bisimulation *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 $q_1 a \twoheadrightarrow_{\mathcal{T}} w q_1'$, there exists q_2' such that $q_2 a \twoheadrightarrow_{\mathcal{T}} w q_2'$ and $q_1'\ F\ q_2'$.*

An equivalence relation $P \subseteq Q \times Q$ is a past bisimulation, *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 $q_1 a \twoheadrightarrow_{\mathcal{T}} w q_1'$, there exists q_2 such that $q_2 a \twoheadrightarrow_{\mathcal{T}} w q_2'$ 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.*

The bisimulation relations enjoy the expected properties ([10]): For both the future and the past case, the identity relation is a bisimulation, the inverse of a bisimulation is one, as well, and the notion of bisimulation is closed under relational composition. Furthermore, the notions of bisimulation are closed under union, more precisely, given two future bisimulation relations F_1 and F_2, then $(F_1 \cup F_2)^*$ is a future bisimulation, and analogously for the past case. The extra Kleene closure is needed since we require a bisimulation relation to be an equivalence. 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 of states Q. We will exploit this property in Section 4.

The definition of a quotient is fairly standard:

Definition 2 (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 $\hat{q}a \twoheadrightarrow w\hat{q}' \in R_{/\cong}$ iff there exist q, q' such that $\hat{q} = [q]_{\cong}$, $\hat{q}' = [q']_{\cong}$, and $q'a \twoheadrightarrow wq' \in R$.*

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 on the example of 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 (past)" will be captured by *being future (past) bisimilar*. To ensure the existence of both required states, we will restrict our attention to *swapping* future and past bisimulations:

Definition 3 (Swapping). *Two relations R and S over the same set* swap *(or:* are swapping*), if $R; S = S; R$ (where ; denotes relational composition).*

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. Note that the theorem covers collapsing $\mathcal{T}^{<\omega}$ with respect to $\sim_f$ or 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. The full proof appears in [4].

Theorem 4. *Let $\mathcal{T}$ be a transducer, and F and P a swapping pair of a future and 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 sketch. With $F; P$ being a congruence, we will write $\equiv_{F;P}$ for that relation in the rest of the proof.

The important direction to show is that $t' \in [\mathcal{T}^{<\omega}_{/\equiv_{F;P}}](t)$ implies $t' \in [\mathcal{T}^{<\omega}](t)$ (the reverse implication is straightforward: Collapsing states never yields fewer transductions). To show this implication requires a characterization of the reductions realized by a quotient: Since for any congruence relation $\cong$, $\mathcal{T}^{<\omega}_{/\cong}$ is given by identifying states of $\mathcal{T}^{<\omega}$ while retaining the reduction relation of $\mathcal{T}^{<\omega}$ (modulo the collapsing of

$$\begin{tikzcd}
t_1 \arrow[r, "\mathcal{T}^{<\omega}"'] \arrow[d, equal, "P"'] & t_3 \arrow[rr, "*", "\mathcal{T}^{<\omega} \cup \equiv_{F;P}"'] \arrow[d, equal, "P"'] & & t_2 \arrow[ddl, equal, "F;P"] \\
t_1' \arrow[r, "\mathcal{T}^{<\omega}"'] & t_3'' \arrow[r, "*", "\mathcal{T}^{<\omega}"'] \arrow[d, equal, "F"'] & t_2'' \arrow[d, equal, "F"'] & \\
 & t_3' \arrow[r, "*", "\mathcal{T}^{<\omega}"'] & t_2'. &
\end{tikzcd}$$

states), the possible reduction steps of $\mathcal{T}^{<\omega}_{/\cong}$ are either reduction steps from $\mathcal{T}^{<\omega}$ or steps replacing a word by a congruent one, i.e., $[t_1]_{\cong} \twoheadrightarrow^*_{\mathcal{T}^{<\omega}_{/\cong}} [t_2]_{\cong}$ iff $t_1 \ (\twoheadrightarrow_{\mathcal{T}^{<\omega}} \cup \cong)^* \ t_2$. Using this characterization for the congruence $\equiv_{F;P}$, the above implication can be phrased as (and generalized, for sake of induction, to) the following requirement:

> If $t_1 (\twoheadrightarrow_{\mathcal{T}^{<\omega}} \cup \equiv_{F;P})^* \ t_2$, then there exist words t'_1 and t'_2 such that $t'_1 \twoheadrightarrow^*_{\mathcal{T}^{<\omega}} t'_2$, and furthermore $t_1 \equiv_{F;P} t'_1$ and $t_2 \equiv_{F;P} t'_2$.

This property is shown by induction on the length of the reduction sequence. Distinguishing in the induction step $t_1 \equiv_{F;P} t_3$ and $t_1 \twoheadrightarrow_{\mathcal{T}^{<\omega}} t_3$, both cases are solved by straightforward induction, where the second one (cf. the above diagram) exploits the assumption that $\equiv_{F;P}$ is swapping.

To see that the result follows from the above implication, use the soundness observation for the unquotiented transducer, that $t' \in [\mathcal{T}^{<\omega}](t)$ iff $t' \in [\mathcal{T}^k](t)$ for some $k \in \omega$, and specialize t_1 resp. t_2 to $q_i\tilde{t}_1$ resp. to $\tilde{t}_2 q_f$, where $\tilde{t}_1, \tilde{t}_2 \in \Sigma^*$, and furthermore $q_i \in Q_i$ and $q_f \in Q_f$. □

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

To make algorithmic use of the quotienting result, we must be able to effectively compute (and represent) swapping bisimulation relations on $\mathcal{T}^{<\omega}$. 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 4 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 will draw upon various standard notions and results from rewrite theory, briefly recalling them as they occur. A detailed treatment of the field can be found in e.g. [2]. The basic notions can be illustrated on our running example. The fact that states 1 and 12 are future bisimilar is rephrased by assuming two rewrite rules, one saying that 1 may be rewritten into 12, and another saying that 12 may be rewritten into 1. We use $\twoheadrightarrow_F$ to denote the *rewrite relation generated by* F, i.e., for $\alpha, \alpha' \in Q^*$, we have $\alpha \twoheadrightarrow_F \alpha'$ if $\alpha = \alpha_l \beta \alpha_r$, $\alpha' = \alpha_l \beta' \alpha_r$, and $(\beta, \beta') \in F$ for some $\alpha_l, \alpha_r, \beta, \beta' \in Q^*$; similarly for $\twoheadrightarrow_P$. The relations $\twoheadleftarrow\!\!\twoheadrightarrow^*_F$ and $\twoheadleftarrow\!\!\twoheadrightarrow^*_P$ denote the *congruence closures*[1] of F and P over the monoid Q^* of strings over Q.

We first address question 1) from above. As mentioned in Section 2, the future and past bisimilarity relations are congruences over the monoid Q^*, i.e., 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

[1] As F and P are symmetric, taking the symmetric closure has no effect, but we still prefer to write $\twoheadleftarrow\!\!\twoheadrightarrow_F$ and $\twoheadleftarrow\!\!\twoheadrightarrow_P$ instead of $\twoheadrightarrow_F$ and $\twoheadrightarrow_P$ in order to stress that they are symmetric.

for $\sim_p$. Based on the congruence property, the following lemma expresses the required extrapolation of bisimulation relations from a finite approximant to $\mathcal{T}^{<\omega}$.

Lemma 5. *Let $\mathcal{T}$ be a finite-state transducer with states Q and, for some $n \geq 0$, let F and $P \subseteq Q^{\leq n} \times Q^{\leq n}$ be future and a past bisimulation on $\mathcal{T}^{\leq n}$. Then the relation $\leftrightarrow^*_F$, resp. $\leftrightarrow^*_P$, is a future, resp. a past, bisimulation on $\mathcal{T}^{<\omega}$.*

After having extended finite bisimulations F and P, question 2) is whether $\leftrightarrow^*_F$ and $\leftrightarrow^*_P$ additionally enjoy the swapping requirement. Now, reducing properties of a many-step rewrite relation to properties of the one-step relation is a standard topic in rewrite theory. First note that swapping of relations is closely related to the notion of *commutation*: R and S commute if $(R^{-1})^*; S^* \subseteq S^*; (R^{-1})^*$ (note the transitive closures). Now for *symmetric* relations, clearly R and S commute iff R^* and S^* swap. The following lemma (see e.g. [2]) reduces commutation to the *commuting-diamond property*: R and S have the commuting-diamond property if $R^{-1}; S \subseteq S; R^{-1}$.

Lemma 6. *Let F and P be two relations on $Q^{\leq n} \times Q^{\leq n}$. If $\leftrightarrow_F$ and $\leftrightarrow_P$ have the commuting-diamond property[2], then they commute.*

To effectively identify cases where the (infinite) relations $\leftrightarrow_F$ and $\leftrightarrow_P$ have the commuting-diamond property, one can restrict attention to the so-called *critical pairs.* Consider rewrite rules (α_F, β_F) and (α_P, β_P) from F and P respectively, such that α_F overlaps with α_P in the following way: either $\gamma_1\alpha_F = \alpha_P\gamma_2$ with $|\gamma_1| < |\alpha_P|$, or $\alpha_F = \gamma_1\alpha_P\gamma_2$, for some $\gamma_1, \gamma_2 \in Q^*$. Then the corresponding critical pair is defined as $(\gamma_1\beta_F, \beta_P\gamma_2)$ in the first case and $(\beta_F, \gamma_1\beta_P\gamma_2)$ in the second. Now, in order to check whether $\leftrightarrow_F$ and $\leftrightarrow_P$ have the commuting-diamond property, it suffices to check, for every such critical pair (δ_F, δ_P), whether there exists δ such that $\delta_F\ P\ \delta$, and $\delta_P\ F\ \delta$[3]. As the rewrite systems F and P are finite, there are also only finitely many critical pairs to check. Note that this technique offers only a *sufficient* condition for the commuting-diamond property.

Lemma 5 and Lemma 6 together allow now to apply the quotienting Theorem 4 and do the desired extrapolation.

Corollary 7 (Soundness). *Let $\mathcal{T}$ be a transducer with states from Q and, for some n, let $F \subseteq Q^{\leq n} \times Q^{\leq n}$ and $P \subseteq Q^{\leq n} \times Q^{\leq n}$ a future resp. a past bisimulation on $\mathcal{T}^{\leq n}$. If $\leftrightarrow_F$ and $\leftrightarrow_P$ have the commuting-diamond property, 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 unions of the transducers in parts (a) and (b) of

[2] In fact, a weaker property suffices, called *strong commutation*: $R^{-1}; S \subseteq (S \cup Id); (R^{-1})^*$.

[3] There is a similar condition in case strong commutation is used.

```
input   T = (Q, Q_0, Σ, R)
X := T_id ;
repeat
   X := (T ∘ X) ∪ T_id ;
   determine bisimulations F and P on X s.t.
     ↔_F and ↔_P swap and each possess the diamond property ;
until  X_/≡ ∼_f (T ∘ X_/≡) ∪ T_id
```

Fig. 3. Calculating $\mathcal{T}^*$

Fig. 1), one pair of a future and a past bisimulation (represented as rewriting systems) is $F = \{(12,1),(1,12),(22,2),(2,22)\} \cup Id_{\{0,1,\dots,22\}}$ and $P = \{(00,0),(0,00),(01,1),(1,01)\} \cup Id_{\{0,1,\dots,22\}}$, where Id_S denotes the "identity rewrite system" on S. Indeed, these bisimulations are the largest 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. 2, where only the relevant part is shown. It can be checked that the construction 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.

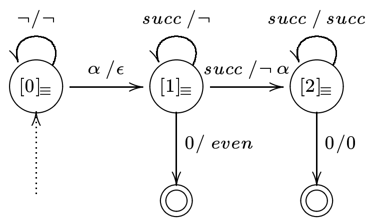

Fig. 2. Transducer $\mathcal{T}_\alpha^*$

The algorithm based on these ideas is sketched in pseudo-code in Fig. 3. Given a transducer $\mathcal{T} = (Q, Q_0, \Sigma, R)$, 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 [11, 5]. Note that in the termination condition, the approximant transducer $\mathcal{X}$ is quotiented using the whole equivalence $\equiv = \leftrightarrow_F^*;\leftrightarrow_P^*$, 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 $\leftrightarrow_F^*;\leftrightarrow_P^* = (\leftrightarrow_F \cup \leftrightarrow_P)^* = \leftrightarrow_{F\cup P}^*$. So, the question is when strings are congruent under the rewrite system $F \cup P$. The first answer of rewrite theory is: If this system is *confluent*, i.e., commutes with itself, and *terminating*, i.e., allows no infinite sequences of rewrite steps, then strings are congruent iff they rewrite to the same normal forms. This obviously gives a procedure to determine congruence. Being a special case of commutation, confluence of $F \cup P$ can be checked using Lemma 6, by inspecting critical pairs. In practice, we can avoid duplicating work by the following standard result.

Lemma 8. *If $\leftrightarrow_F$ and $\leftrightarrow_P$ commute, then $\leftrightarrow_F \cup \leftrightarrow_P$ is confluent if each of $\leftrightarrow_F$ and $\leftrightarrow_P$ in separation is confluent.*

So, if commutation of $\leftrightarrow_F$ and $\leftrightarrow_P$ has already been checked when determining whether $\leftrightarrow_F^*$ and $\leftrightarrow_P^*$ swap, then it suffices to check confluence of the individual relations. In case $\leftrightarrow_F \cup \leftrightarrow_P$ 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 $F \cup P$ into an equivalent rewrite system that is confluent, using so-called Knuth-Bendix *completion* [9]; we refer to [2] for details.

As for checking termination — it is clear that the relations F and P in separation 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. 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 or the token array example of [7]. 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), this and other, larger transducers can be successfully handled.

5 Conclusions, Related Work, and Future Work

We 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 $\mathcal{T}^{\leq 0}(S)$, $\mathcal{T}^{\leq 1}(S)$, $\mathcal{T}^{\leq 2}(S)$, $\ldots$. 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[4].

Closest to our work is [3, 7], which presents an algorithm using standard subset-construction and minimization techniques from automata theory. Sufficient conditions for termination of the algorithm are identified. Roughly speaking, our algorithm and the one from [3, 7] start from opposite extremes. Our algorithm starts from $\mathcal{T}$ and tries to compute a finite quotient of $\mathcal{T}^{<\omega}$. Their algorithm starts from the initial state of $\mathcal{T}^{<\omega}$, which can be represented by the regular language q_0^*, and 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 $\mathcal{T}^{<\omega}$. On the contrary, our algorithm tries to discover such equivalences dynamically, i.e., during execution. Now, an important assumption in their approach is that the set of pairs $(a, w) \in \Sigma \times \Sigma^*$ that occur along the edges of $\mathcal{T}^{<\omega}$ is finite and known in advance (or at least a finite super-set must satisfy these conditions). In case $\mathcal{T}$ is a "letter-to-letter" transducer, only pairs from $\Sigma \times \Sigma$ may occur in $\mathcal{T}^{<\omega}$, and hence, the assumption is satisfied. However, for non-length-preserving transducers the assumption is in general not satisfied.

Besides the improvements mentioned in Section 4 and implementation improvements like using BDDs 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 upto 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. [6]. The difficulty in our context lies in the generalization of the computed bisimulations to $\mathcal{T}^{<\omega}$.

[4] This suggestion was made both by Kedar Namjoshi and an anonynous referee.

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 [4].

Acknowledgements We like to thank the anonymous referees for their careful work and their valuable and insightful comments and suggestions.

References

1. P. A. Abdulla, A. Bouajjani, and B. Jonsson. On-the-fly analysis of systems with unbounded lossy Fifo-channels. In A. J. Hu and M. Y. Vardi, editors, *Proceedings of CAV '98*, volume 1427 of *Lecture Notes in Computer Science*, pages 305–318. Springer-Verlag, 1998.
2. F. Baader and T. Nipkow. *Term Rewriting and All That.* Cambridge University Press, 1998.
3. A. Bouajjani, B. Jonsson, M. Nilsson, and T. Touili. Regular model checking. In E. A. Emerson and A. P. Sistla, editors, *Proceedings of CAV '00*, volume 1855 of *Lecture Notes in Computer Science*, pages 135–145. Springer-Verlag, 2000.
4. 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).
5. J. Fernandez. An implementation of an efficient algorithm for bisimulation equivalence. *Science of Computer Programming*, 13:219–236, 1989.
6. S. Graf and B. Steffen. Compositional minimization of finite state systems. In E. M. Clarke and R. P. Kurshan, editors, *Computer Aided Verification 1990*, volume 531 of *Lecture Notes in Computer Science*. Springer-Verlag, 1991.
7. B. Jonsson and M. Nilsson. Transitive closures for regular relations for verifying infinite-state systems. In S. Graf and M. Schwartzbach, editors, *TACAS 2000*, volume 1785 of *Lecture Notes in Computer Science*. Springer-Verlag, 2000.
8. Y. Kesten, O. Maler, M. Marcus, A. Pnueli, and E. Shahar. Symbolic model checking with rich assertional languages. In 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.
9. 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.
10. R. Milner. *Communication and Concurrency.* Prentice Hall, 1989.
11. R. Paige and R. E. Tarjan. Three partition refinement algorithms. *SIAM Journal on Computing*, 16(6):973–989, 1987.
12. D. Park. Concurrency and automata on infinite sequences. In P. Deussen, editor, *Fifth GI Conference on Theoretical Computer Science*, volume 104 of *Lecture Notes in Computer Science*, pages 167–183. Springer-Verlag, 1981.