Reactive Links to Save Automata States

Maxime Crochemore and Dov M. Gabbay

King's College London

Abstract. The goal of the reactive automata model is to reduce the space required for the implementation of automata. A reactive automaton has extra links whose role is to change the behaviour of the whole automaton. These links do not increase their expressiveness. Typical examples of regular expressions associated with deterministic automata of exponential size according to the length of the expression show that reactive links provide an alternative representation of total linear size.
Keywords: automaton, language representation, reactivity.

1 Introduction and background

This note introduces the notion of reaction for automata and shows that using reactive links can reduce dramatically the number of states of an automaton. Within this framework some examples of state reduction are striking.

The basic use of automata is for testing if a word belongs to a regular language (membership testing). It can be done either on a regular describing the language or with an automaton, which is equivalent due to Kleene's Theorem (see for example [5]). When the automaton is deterministic the acceptance of a word of length m can be tested in linear time using $O(kn)$ space, where k is the size of the alphabet and n the number of states of the automaton. If the automaton is non deterministic the alternative is to simulate an equivalent deterministic automaton or to transform it into a deterministic automaton. The first option leads to $O(mn)$ membership time with space proportional to that of the non-deterministic automaton. The second option yields linear membership but at the cost of the determinisation which can be time and space exponential in the number of states. For related algorithms, see [1] or [6] and references therein. Both solutions are indeed implemented in the many variants of `grep` software aimed at locating regular motifs in texts. See also [7] for extra implementations and applications.

The notion of reactive automata was introduced in [3] and [4], as part of a general reactive methodology. The basic idea is that a reactive system is a system that dynamically changes during its execution as a reaction to the manner it is being utilised. A reactive system has to be distinguished from a time-dependent system as it is not dependent on an objective clock.

Reactive automaton Figure 1 displays an example of a reactive automaton. Consider the automaton (ii), which has one state i. The transition is indicated as above by the single-head arrow and reactivity by the double-head arrow. For simplicity we assume we have only one letter `a`. Assume i is the initial state as well as the terminal state. Upon receiving a letter `a` the machine stays at i, and can accept or continue. The status of i as a terminal state is then cancelled by the reactive arrow. Upon receiving a second letter `a` the machine cannot accept anymore the word `aa` as i is no longer a terminal state. Instead if the machine receives a third letter `a` state i is reactivated as a terminal state and the machine can accept the word $\mathtt{a}^3$. Obviously

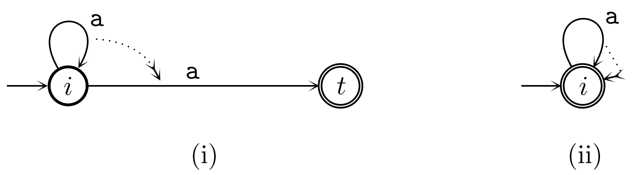

Figure 1. Two reactive automata accepting the language $\mathtt{a}^{2n+1}$. All arcs are initially active. (i) The automaton uses a reactive edge-to-edge link to cancel or activate the arc from the initial state i to the terminal state t. (ii) The automaton uses a reactive edge-to-state link to flip the status of state i alternately as terminal and non terminal.

this reactive automaton accepts words of the form $\mathtt{a}^{2n+1}$, $n \geq 0$, only. To implement such an acceptor without reactivity we would need more states (at least two for this example).

In the next section we define the notion of a reactive automaton and show in Section 3 that the expressive power of automata is unchanged by adding reactive links. In Section 4 we state the reduction power of reactive links. Examples of reactive automata given in Section 5 have only a linear number of reactive links while they are logarithmically smaller than the minimal automata associated with their accepted language. Some remarks and open questions are stated in the conclusion.

2 Reactive automata

We formally define reactive automata starting with the notion of reactive transformation assuming the reader has some knowledge of automata definition.

Definition 1 (Switch Reactive Transformation). *Let $R \subseteq S \times \Sigma \times S$ be the transition relation of an (ordinary) automaton $\mathcal{A}$. Let $\mathbf{T}^+$, $\mathbf{T}^-$ be two subsets of $(S \times \Sigma \times S) \times (S \times \Sigma \times S)$; they are composed of pairs of the form $((p,\sigma,q),(r,\tau,s))$ where $\sigma,\tau \in \Sigma$, $p,q,r,s \in S$, and $(p,\sigma,q) \in R$.*

We define a transformation $(p,\sigma,q) \longrightarrow R^{(p,\sigma,q)}$ for $(p,\sigma,q) \in R$ using the sets $\mathbf{T}^+$ and $\mathbf{T}^-$ as follows:

$$R^{(p,\sigma,q)} = (R \setminus \{(r,\tau,s) \mid (r,\tau,s) \in R \text{ and } ((p,\sigma,q),(r,\tau,s)) \in \mathbf{T}^-\}) \cup \{(r,\tau,s) \mid (r,\tau,s) \in R \text{ and } ((p,\sigma,q),(r,\tau,s)) \in \mathbf{T}^+\}$$

Definition 2 (Switch Reactive Automaton).

1. *A reactive automaton is an ordinary non-deterministic automaton with a switch reactive transformation, i.e. a triple $\mathcal{R} = (\mathcal{A}, \mathbf{T}^+, \mathbf{T}^-)$ which defines the switch reactive transformation above.*
2. *Let $\sigma_1\sigma_2\cdots\sigma_n$ be a word on the alphabet Σ. We define the notion of a (non-deterministic) run of $\mathcal{R}$ over $\sigma_1\sigma_2\cdots\sigma_n$. The run is a sequence of pairs (p_k, R_k), $k = 0,\ldots,n$, defined as follows:*

   Step 0. *We start with the pair $(p_0, R_0) = (i, R)$ from the automaton $\mathcal{A} = (S, i, \Sigma, F, R)$.*

   Step $k > 0$. *Assume pairs (p_0,R_0), (p_1,R_1), …, (p_{k-1},R_{k-1}) have been defined as the result of a run over $\sigma_1\sigma_2\cdots\sigma_{k-1}$. Then, state p_k is such that $(p_{k-1},\sigma_k,p_k) \in R_{k-1}$ and $R_k = R_{k-1}^{(p_{k-1},\sigma_k,p_k)}$.*

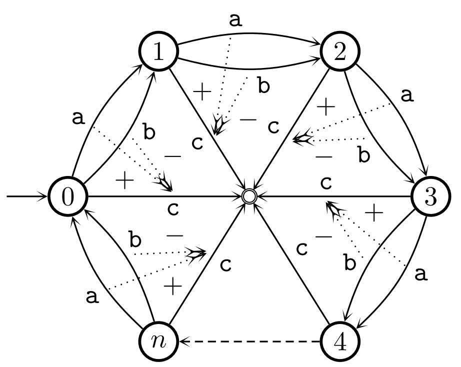

Figure 2. Reactive automaton on the alphabet $A = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ accepting the language $A^*\mathtt{a}A^n\mathtt{c}$. The automaton is deterministic. Any a-transition activates a c-arc from its origin state to the centre, which is the only terminal state. Any b-transition switches off such an arc. If a run over a word stops in the terminal state (the last step is a c-transition), the $(n+1)$th letter before the end of the word must be a, so the word belongs to the language, and conversely.

3. *We say that the reactive automaton $\mathcal{R}$ accepts the word $\sigma_1\sigma_2\cdots\sigma_n$ if there is a run of the automaton over this word that ends with $p_n \in F$.*

Figure 2 shows a reactive automaton that changes its arcs going to the unique terminal state. The automaton is deterministic and has linear size, $O(n)$. The automaton accepts the language $A^*\mathtt{a}A^n\mathtt{c}$ over the alphabet $A = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$. It is known that the minimal deterministic (ordinary) automaton accepting the same language has $2^{n+1} + 1$ states (see [5]).

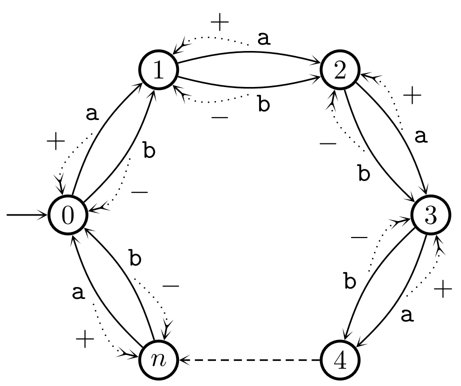

Figure 3. Reactive deterministic automaton on the alphabet $A = \{\mathtt{a}, \mathtt{b}\}$ accepting the language $A^*\mathtt{a}A^n$ similar to the language of the automaton of Figure 2. Initially, the automaton has no terminal state and it uses edge-to-state reaction. Any a-transition makes its starting state a terminal state. Any b-transition transforms its starting state into a non-terminal state. If the run over a word stops in a terminal state, since the state has been set as a terminal state by an a-transition on the nth letter before, the word belongs to the language, and conversely.

The definitions extend to accommodate edge-to-state reaction. For example, Figure 3 shows another reactive automaton which changes its terminal states only and accepts a language similar to that of Figure 2. The definition of a reactive automaton changing its terminal states is a simple adaptation of the above definition. Changing terminal states can be simulated in the model of switch reactive automata as follows without altering the possible determinism: all potential terminal states are made non terminal and linked by an ε-arc to a unique terminal node; reactive links to states are redirected to the new ε-arcs.

3 Reactivity and non-reactivity

We focus on switch reactive automata and show that their expressive power is identical to the one of ordinary automata. The proof can be adapted to automata with the various types of reactive links described in previous sections.

Theorem 3. *Any switch reactive deterministic or non-deterministic automaton is equivalent to an (ordinary) deterministic or non-deterministic automaton, respectively.*

Proof. Let $\mathcal{R} = (\mathcal{A}, \mathbf{T}^+, \mathbf{T}^-)$. We define the automaton $\mathcal{B} = (\hat{S}, (i, R), \Sigma, F, \hat{R})$ whose set of states $\hat{S}$ is composed of pairs of the form (x, R') where $x \in S$ and R' is related to R via the switch reactive transformation using $\mathbf{T}^+$ and $\mathbf{T}^-$. The transition relation $\hat{R}$ of $\mathcal{B}$ is defined using $\mathbf{T}^+$ and $\mathbf{T}^-$ as follows: $((x_1, R_1), \sigma, (x_2, R_2)) \in \hat{R}$ iff $(x_1, \sigma, x_2) \in R_1$ and $R_2 = R_1^{(x_1,\sigma,x_2)}$. Then $\mathcal{B} = (\hat{S}, i, \Sigma, F, \hat{R})$.

It is straightforward to see that a run of $\mathcal{R}$ on $\sigma_1\sigma_2\cdots\sigma_n$ corresponds to a run of $\mathcal{B}$ on the same word and vice versa. □

Remark We saw that a switch reactive automaton actually starts as an ordinary automaton $\mathcal{A} = (S, i, \Sigma, F, R)$ and then changes into different automata while receiving the input. The proof extends to Reactive Automata with state-to-state or edge-to-state reactive links. Therefore, changes can either affect the transition relation or the terminal states as shown on the previous examples.

4 Saving states of automata using reactive links

Reactive links can be used to reduce the number of states of (ordinary) automata. We state the fundamental results for deterministic and non-deterministic automata.

Theorem 4. *Any automaton $\mathcal{A}$ with kn states admits an equivalent reactive automaton $\mathcal{R}(\mathcal{A})$ with $k + n$ states. If $\mathcal{A}$ is deterministic, so is $\mathcal{R}(\mathcal{A})$.*

The construction in the proof of the above theorem cannot be iterated to reduce further the number of states.

Using higher levels of reactive links (e.g. reactive links from arc to reactive links) the next statement holds. As above the construction in the proof cannot be iterated.

Theorem 5. *If $\mathcal{A}$ is deterministic automaton with k^n states, it has an equivalent reactive automaton $\mathcal{R}(\mathcal{A})$ with $k \cdot n$ states.*

5 Examples in size reduction

The use of reactive links in automata can dramatically reduce the size of a deterministic automaton accepting a given regular language. The two previous sections show that this is possible for deterministic as well as non-deterministic automata but with a large number of reactive arcs. Instead, in the next examples, reactivity keeps the total size of automata as small as the size of their non-deterministic equivalent but without loosing determinism. In these examples determinism without reactivity leads to an exponential blow up of the number of states and then of the total size of automata.

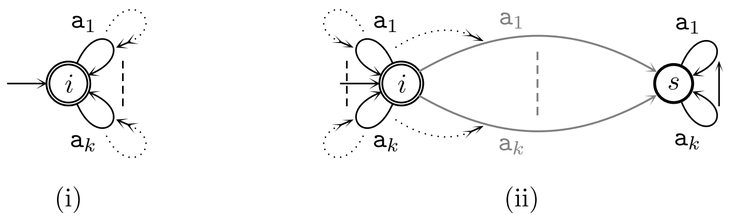

Figure 4. Two deterministic reactive automata accepting the set of strings in which each letter of the alphabet $\{\mathtt{a}_1, \mathtt{a}_2, \ldots, \mathtt{a}_k\}$ appears at most once. All loop arcs are initially active. Loops on state i are made inactive after their first use. (i) Incomplete version. (ii) Complete version: only arcs from i to s are initially inactive and become active after the first use of their corresponding loop on state i.

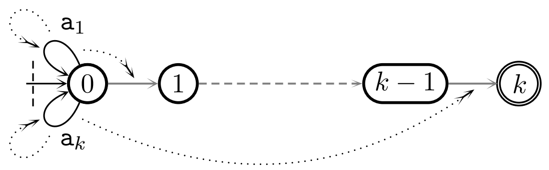

Figure 5. A deterministic reactive automaton accepting the set of strings that are permutations of the letters $\mathtt{a}_1, \mathtt{a}_2, \ldots, \mathtt{a}_k$. All loops on the initial state are initially active and other ε-arcs are inactive. One reactive link for letter $\mathtt{a}_i$ cancels its respective loop while the second activates its associated ε-arc.

The first example corresponds to the finite language of words in which each letter of the alphabet appears at most once (see Figure 4). Its principle is that loops on the initial state are cancelled by a reactive link immediately after being used. States of a deterministic automaton for the language have to store the set of letters already treated and therefore the minimal automaton has at least an exponential number of states. Instead the total size of the reactive automaton accepting the language is $O(k)$ on a k-letter alphabet.

The automaton of Figure 5 accepts all the $k!$ permutations of letters. To do that we add a path of length k from the initial state to the unique terminal state. Compared with the automaton of the previous example, the aim is to count the number of letters treated on the initial state. Each loop on the initial state has an additional reactive link that activates its associated ε-arc on the path. So, the terminal state can be reached only if all ε-arc are activated. We view the automaton as deterministic because its light non-determinism due to ε-arcs can be remove by considering a special symbol marking the end of words. The size of the reactive automaton is $O(k)$, which contrasts with the $O(2^k)$ size of the minimal automaton accepting the language.

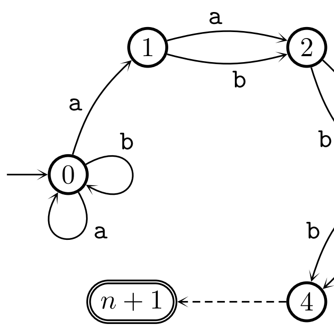

Figure 6. Non-deterministic automaton on the alphabet $A = \{\mathtt{a}, \mathtt{b}\}$ accepting the language $A^*\mathtt{a}A^n$. Its equivalent reactive automata of Figures 2 and 3 have sizes of the same order.

The language $A^*\mathtt{a}A^n$ is accepted by the non-deterministic automaton of Figure 6 that has $n+2$ states and $O(n)$ total size. The non-determinism appears on the initial state only. It is known that the minimal deterministic automaton accepting the same language has 2^{n+1} states (see [5]) and then also $O(2^n)$ total size, while the equivalent reactive automaton of Figure 2 has only also $n+2$ states. It is noticeable that it has $O(n)$ total size despite the addition of reactive links.

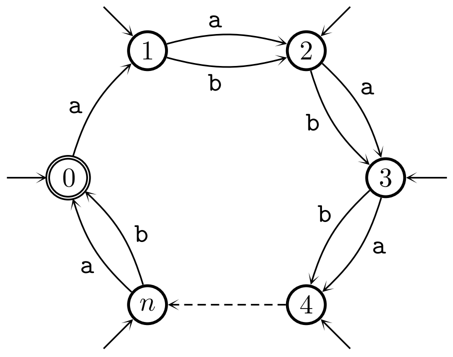

Figure 7. Non-deterministic automaton on the alphabet $A = \{\mathtt{a}, \mathtt{b}\}$ accepting the language $A^{\leq n}(\mathtt{a}A^n)^*$. All states are initial states.

The last remarkable example concerns the language $A^{\leq n}(\mathtt{a}A^n)^*$ on the alphabet $A = \{\mathtt{a}, \mathtt{b}\}$. Figure 7 displays a non-deterministic automaton accepting it. It is non-deterministic because all its $n+1$ states are initial states. Figure 8 shows an equivalent reactive automaton, which, as above, may be considered as deterministic since ε-arcs are useful only at the end of the input word. Reactive links are from $\mathtt{b}$-transitions and cancel their associated ε-arc to the terminal state. The number of states is $n+2$ and the total size is $O(n)$. This is to be compared with the result of Béal et al. [2], which shows that the minimal deterministic automaton for this language has an exponential number of states.

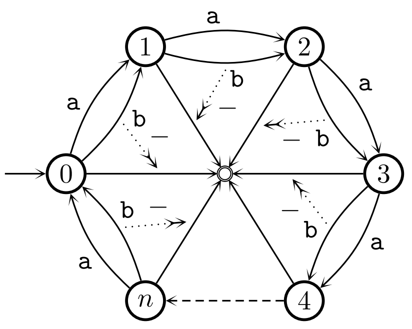

Figure 8. Reactive deterministic automaton on the alphabet $A = \{\mathtt{a}, \mathtt{b}\}$ accepting the same language $A^{\le n}(\mathtt{a}A^n)^*$ as the automaton of Figure 7. It has only one terminal state in the center. During a run, at least one ε-arc remains if some positions of letter a in the input word form a non extendible arithmetic progression of period n. The automaton has only one more state and twice as many arcs as the automaton of Figure 7.

6 Conclusion

The strength of reactive links in automata comes essentially from the reduction of the size of their implementation. Designing a non-deterministic automaton to solve a Pattern Matching question leads to slow algorithms requiring extra work to be implemented. Instead, the use of reactive links can turn a non-deterministic automaton into a deterministic Pattern Matching machine. This has two simultaneous advantages: little effort at implementation and efficient running time since the basic operation required when parsing a stream of data with an automaton is table lookup to change state, which avoids any other more time demanding low or high level instruction.

Open question In some sense, reactivity competes with non-determinism to get small automata accepting a given language, although they are not antagonist concepts. Because despite results of Section 4 showing that reactivity reduces significantly the number of states of automata, the solution requires in general a large number of extra links. But the significant examples of Section 5 raise our hope that this number can indeed be fairly small.

This note leaves open the question of whether, given a language described by a regular expression of size r, there is a reactive deterministic automaton of size $O(r)$ accepting it.

References

1. A. V. Aho: *Algorithms for finding patterns in strings*, in Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity (A), MIT Press, 1990, pp. 255–300.
2. M.-P. Béal, M. Crochemore, F. Mignosi, A. Restivo, and M. Sciortino: *Computing forbidden words of regular languages.* Fundamenta Informaticae, 56(1,2) 2003, pp. 121–135.
3. D. M. Gabbay: *Reactive Kripke semantics and arc accessibility*, in Combinatorial Logic, W. Carnielli, F. M. Dionesio, and P. Mateus, eds., Centre of Logic and Computation, University of Lisbon, 2004, pp. 7–20.
4. D. M. Gabbay: *Reactive Kripke semantics and arc accessibility*, in Pillars of Computer Science: Essays dedicated to Boris (Boaz) Trakhtenbrot on the occasion of his 85th birthday, A. Avron, N. Dershowitz, and A. Rabinovitch, eds., vol. 4800 of LNCS, Springer-Verlag, Berlin, 2008, pp. 292–341.

5. J. E. Hopcroft, R. Motwani, and J. D. Ullman: *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, third edition ed., 2006.
6. M. Lothaire, ed., *Applied Combinatorics on Words*, Cambridge University Press, 2005.
7. G. Navarro and M. Raffinot: *Flexible Pattern Matching in Strings—Practical on-line search algorithms for texts and biological sequences*, Cambridge University Press, 2002.