Size reduction of multitape automata

Hellis Tamm

Literature:

Tamm, H. On minimality and size reduction of one-tape and multitape finite automata. PhD thesis, Department of Computer Science, University of Helsinki, Finland, 2004.

Tamm, H., Nykänen, M., and Ukkonen, E. Size reduction of multitape automata. Tenth Int. Conf. on Implementation and Application of Automata (CIAA 2005). To appear in: LNCS 3845, Springer-Verlag, 2006.

Motivation

- To develop a string handling and manipulating database system
- Expressing string predicates in the Alignment Declaration language
- String declarations are converted into an executable form via an intermediate form – two-way multitape automata
- Size reduction of multitape automata

Size reduction of multitape automata

We present:

- multitape automata size reduction algorithm
- NFA reduction algorithm (based on [Kameda and Weiner])

We combine these two algorithms to get an algorithm for reducing the size of our two-way multitape automata.

Alignment Declaration Language

Grahne, G., Hakli, R., Nykänen, M., Tamm, H., and Ukkonen, E. Design and implementation of a string database query language. *Inform. Syst.* **28**, (2003), 311-337.

To describe string comparison and manipulation operations over several strings that are manipulated together.

Strings are denoted by variables $x, y, \ldots$. Each string is surrounded by left and right endmarkers [and]. Initially, the current position for each string is its left endmarker. To scan a string, the current position can be moved either to the next or previous symbol. A basic statement is an *on*-statement, for example, like

```
scan x on x='a'
```

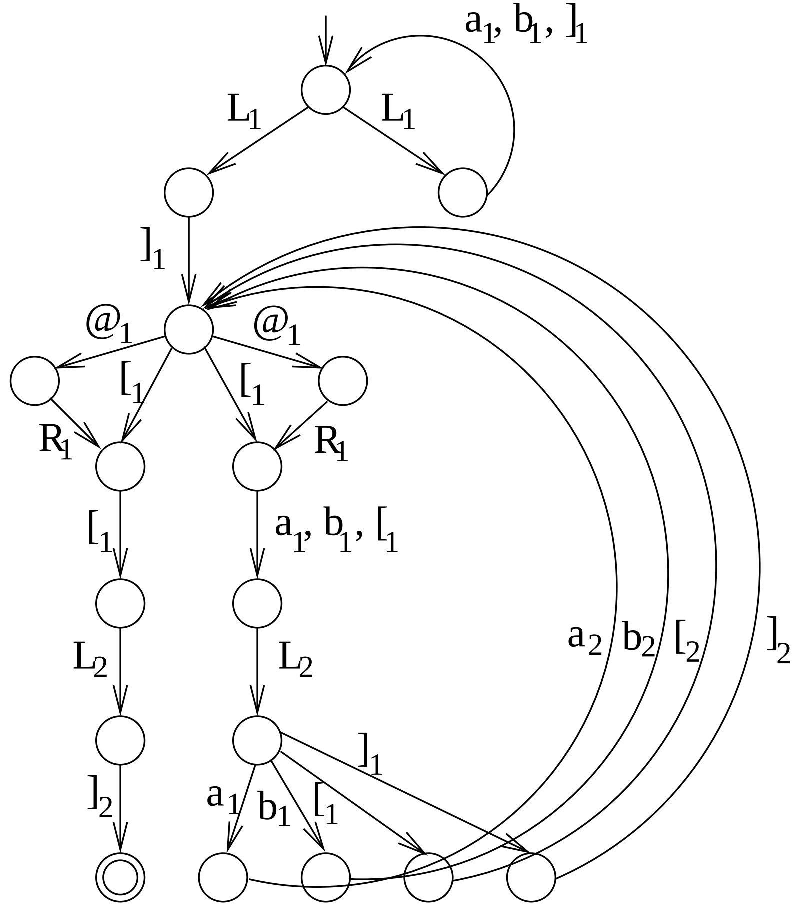
```
rightscan x,y on x=y
```

An example

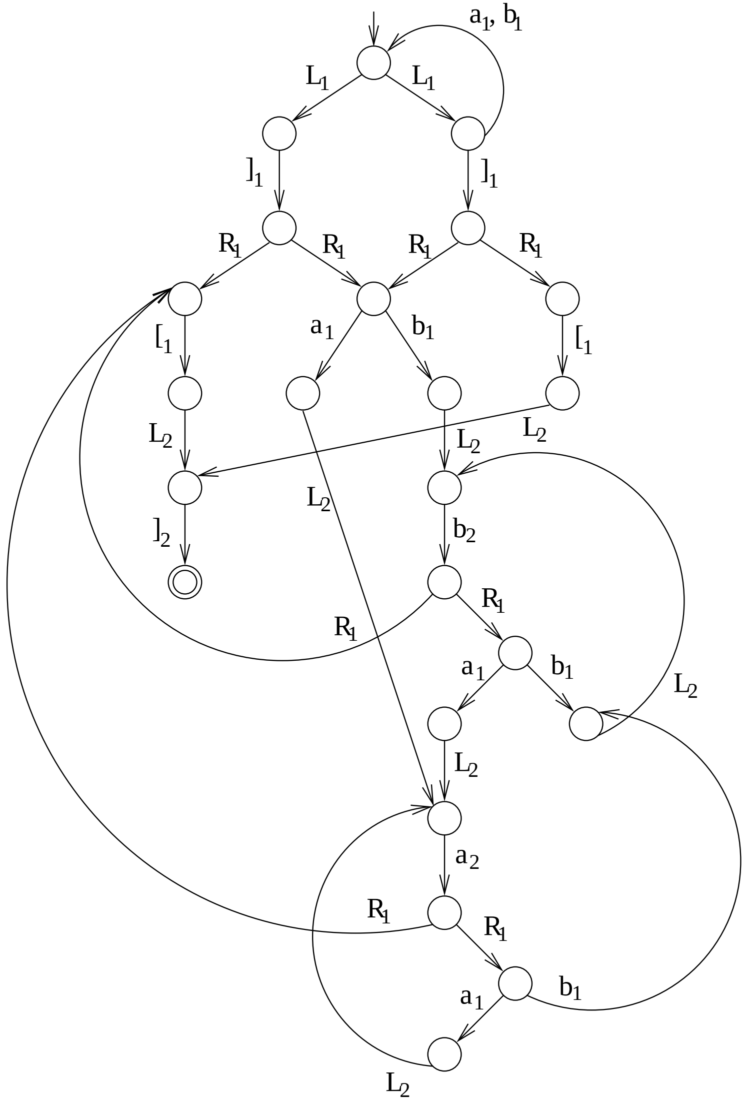
```
reversal(x, y)
    keep x in 'a', 'b'
    keep y in 'a', 'b'
       repeat * times
          scan x on
       end
       scan x on x=]
       repeat * times
          rightscan x on
          scan y on x=y
       end
       rightscan x on x=[
       scan y on y=]
    end
    end
```

Example: multitape automaton

```
reversal(x, y)
    keep x in 'a', 'b'
    keep y in 'a', 'b'
        repeat * times
            scan x on
        end
        scan x on x=]
        repeat * times
            rightscan x on
            scan y on x=y
        end
        rightscan x on x=[
        scan y on y=]
    end
    end
```



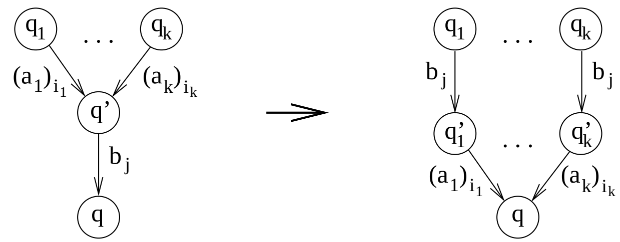
Example: expanded automaton



Automaton transformations

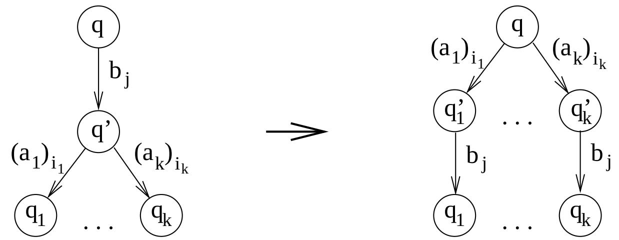
Swap Upwards

(a)



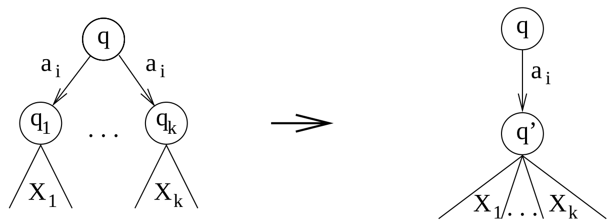
Swap Downwards

(b)



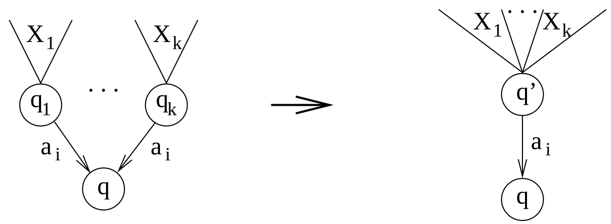
Sink Combine

(c)



Source Combine

(d)



Towards the reduction algorithm

Let $A = (Q, \Sigma, \delta, q_I, F)$ be an n-tape automaton.
Let $\Sigma' = \Sigma \cup \{\mathtt{[}, \mathtt{]}, \mathtt{@}\} \cup \{L, R\}$. Let $a \in \Sigma'$, $i \in \{1, ..., n\}$, and $q_1, q_2, q \in Q$.

procedure MoveTransitionUp($A, (q_1, a_i, q_2), q$**)**

1. **if** transition (q_1, a_i, q_2) exists in A **then**
2. use the Sink Combine transformation to merge all such states that are reachable from q_1 by a transition labelled by a_i and suitable for this transformation;
3. **if** $q \neq q_1$ and $outdegree(q_1) = 1$ **then**
4. use the Swap Upwards transformation on the outgoing transition of q_1 and let T be the set of transitions with the label a_i created by this transformation;
5. **for all** $(q'_1, a_i, q'_2) \in T$ where $q'_1, q'_2 \in Q$ **do**
6. `MoveTransitionUp`$((q'_1, a_i, q'_2), q)$;

Towards the reduction algorithm

Def. A transition is called a *future transition* for the state q and tape i if it is the first transition involving this tape on some path in A that starts from q.

Let us fix some $q \in Q$, $a \in \Sigma'$ and $i \in \{1, ..., n\}$. We want to find a set of future transitions for q and i, with the label a_i, such that by calling the procedure `MoveTransitionUp()` for each of these transitions and the state q, we can reduce the number of states of A by a certain amount.

Towards the reduction algorithm

Let $FT_{q,i,a}$ be a maximal set of future transitions for q and i, with the same label a_i such that the following three conditions hold for the set $P_{FT_{q,i,a}}$ of all paths in A which start from q and end by any transition $(q', a_i, q'') \in FT_{q,i,a}$. Let p be any path in $P_{FT_{q,i,a}}$. Let the two last states on p be q' and q''.

Assume the following:
(i) there are no loops in p, except that q'' may be equal to q;

(ii) every state on p that appears after q and before q'' is non-initial and non-final, all of its incoming and outgoing transitions are traversed by some path in $P_{FT_{q,i,a}}$, and all of its incoming transitions involve a tape that is different from i;
(iii) if q' has more than one outgoing transition then q'' is non-initial and has only one incoming transition.

Towards the reduction algorithm

Proposition P1. The set $FT_{q,i,a}$ is uniquely defined.

Proposition P2. The series of calls to the procedure `MoveTransitionUp()` where it is invoked with every transition in $FT_{q,i,a}$ and q, results in size reduction of A by $|FT_{q,i,a}| - 1$ states.

Also, for another $b \in \Sigma'$ with the set $FT_{q,i,b}$, the application of transformations of (P2) for the set $FT_{q,i,a}$ does not affect the application of transformations of (P2) for the set $FT_{q,i,b}$.

The proofs can be found in my PhD thesis.

Similarly to the conditions (i)–(iii), symmetric conditions can be specified that allow to eliminate states from the automaton by a symmetric procedure `MoveTransitionDown()` that uses the Source Combine and Swap Downwards transformations.

Reduction algorithm for automaton A

```
m := 0; reduced := true; A_1 := CopyOf(A);
while reduced = true do
    reduced := false;
    for tape := 1 to n do
        m_up := Upwards(A, tape);
        m_down := Downwards(A_1, tape);
        if m_up > 0 or m_down > 0 then
            if m_up ≥ m_down then
                A_1 := CopyOf(A);
                m := m + m_up;
            else
                A := CopyOf(A_1);
                m := m + m_down;
            reduced := true;
return A, m;
```

procedure Upwards$(A, tape)$

$m := 0;$

$reduced := true;$

while $reduced = true$ **do**

  $reduced := false;$

  for all $q \in Q$ as long as $reduced = false$ **do**

    find the set $FT_{q,tape} = \bigcup_{a \in \Sigma'} FT_{q,tape,a};$

    for all $a \in \Sigma'$ where $|FT_{q,tape,a}| > 1$ **do**

      find a state q' such that $FT_{q',tape,a} = FT_{q,tape,a}$ and the longest path from q' to the originating state of any transition in $FT_{q,tape,a}$ is of minimal length;

      for all $t \in FT_{q',tape,a}$ **do**

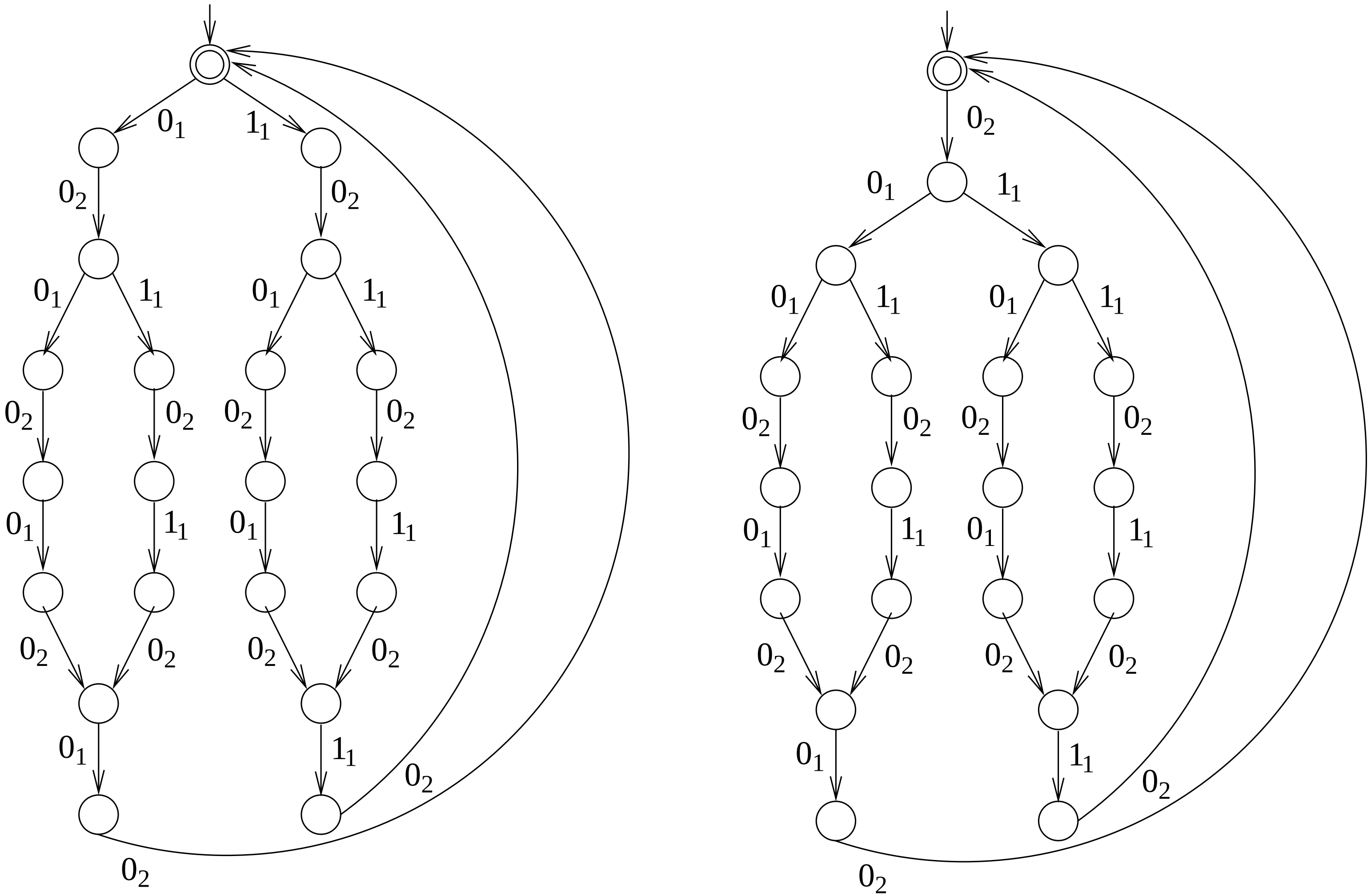
        `MoveTransitionUp`$(A, t, q');$

      $m := m + |FT_{q',tape,a}| - 1;$

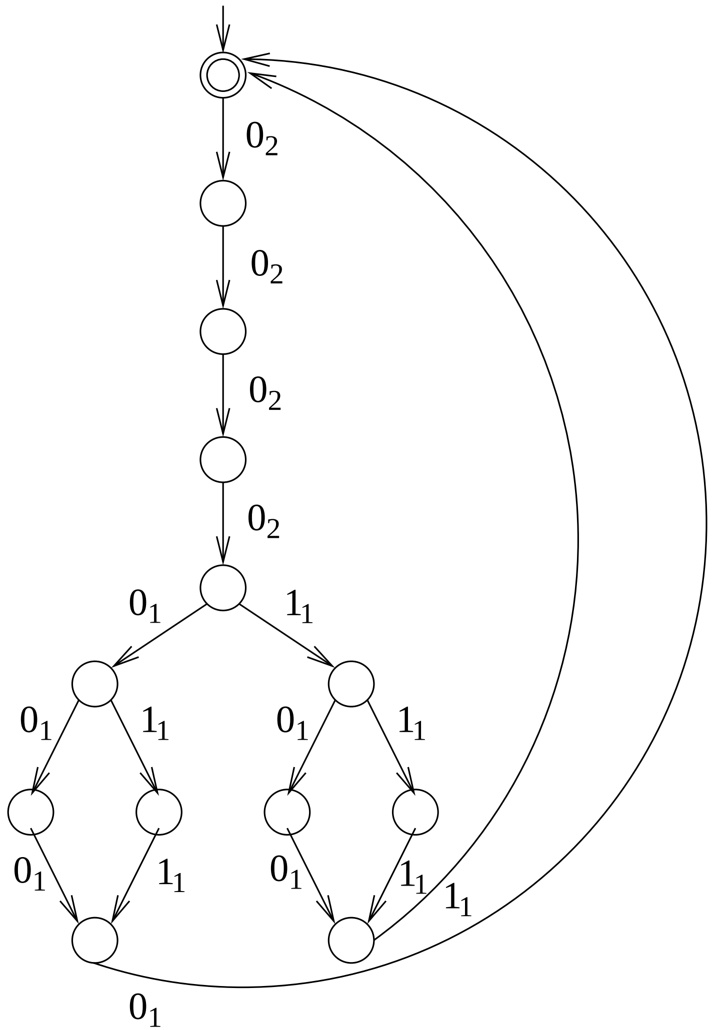
      $reduced := true;$

return $m;$

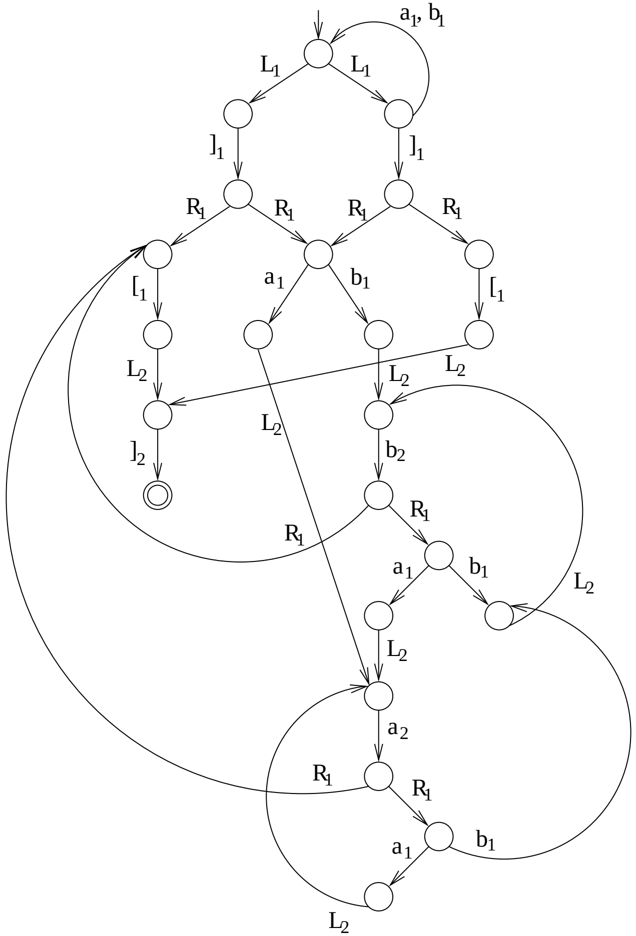
Example: reduction algorithm in work

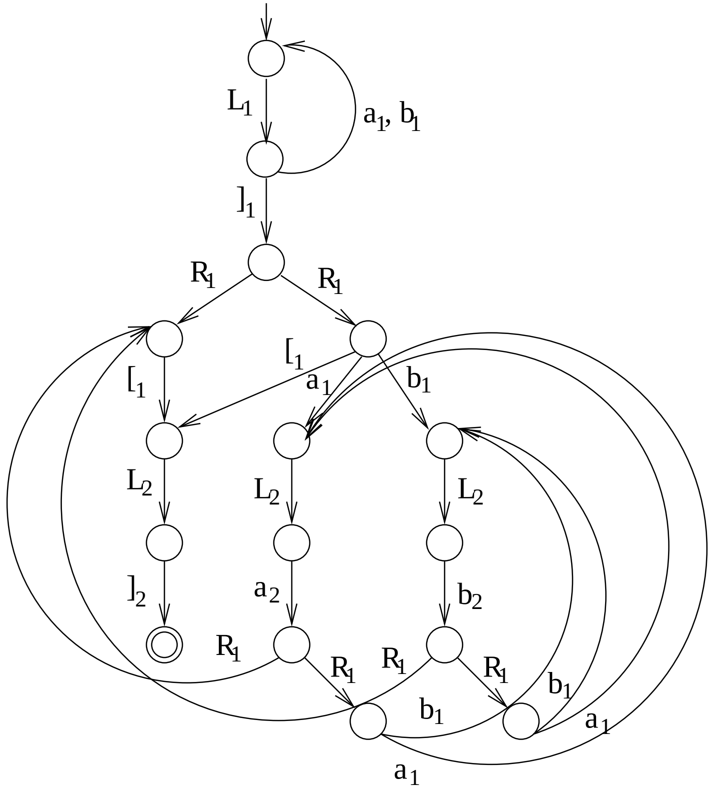


Example: the resulting automaton



Example: applying the reduction algorithm to the automaton for `reversal(x, y)` predicate





Another approach: using NFA reduction algorithms

Our multitape automata can be viewed as (one-tape) NFAs over the alphabet $\{a_i \mid a \in \Sigma', i \in \{1, ..., n\}\}$. Therefore, we can apply NFA size reduction methods as well.

We consider NFA reduction based on [Kameda and Weiner, 1970].

Let A be an NFA and let $C = \mathit{subset\_construction}(A^R)$.

Kameda and Weiner: two states of A are equivalent if and only if they appear exactly in the same states of C. This is useful for DFA minimization – by merging the equivalent states one can find a minimal DFA. But this method can be used for NFA reduction, too.

Similarly, we can find the equivalent states of A^R, and by appropriate merging of states, use this to reduce A.

Merging the equivalent states in NFA can produce useless states which can be eliminated.

NFA reduction

Similarly to [Ilie and Yu], we can possibly get a smaller NFA by combining the reductions corresponding to the two equivalences.

We propose the following method for NFA reduction.

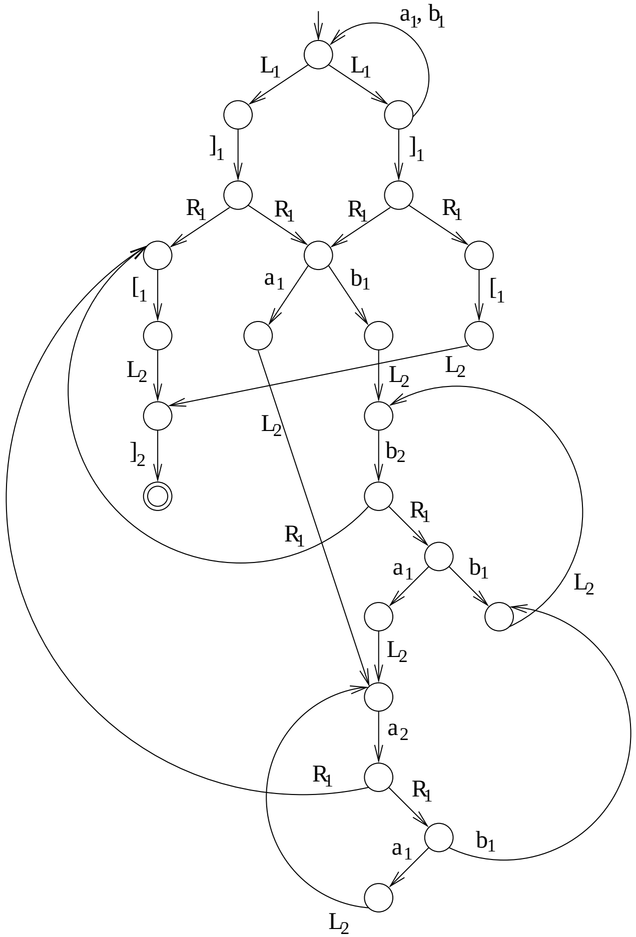
First, find and merge the equivalent states of an NFA, and eliminate the useless states from the automaton.

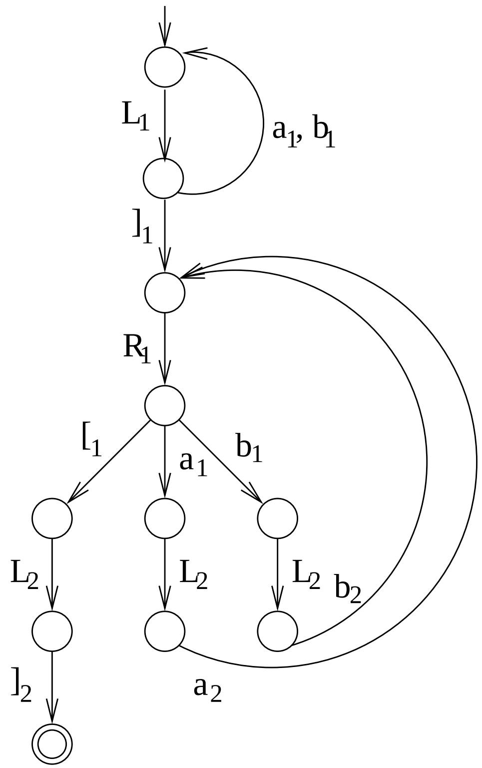
Second, find and merge the equivalent states of the reversal of the resulting automaton, eliminating the useless states as well.

If the automaton size was reduced by the second method, then again, apply the first method, etc.

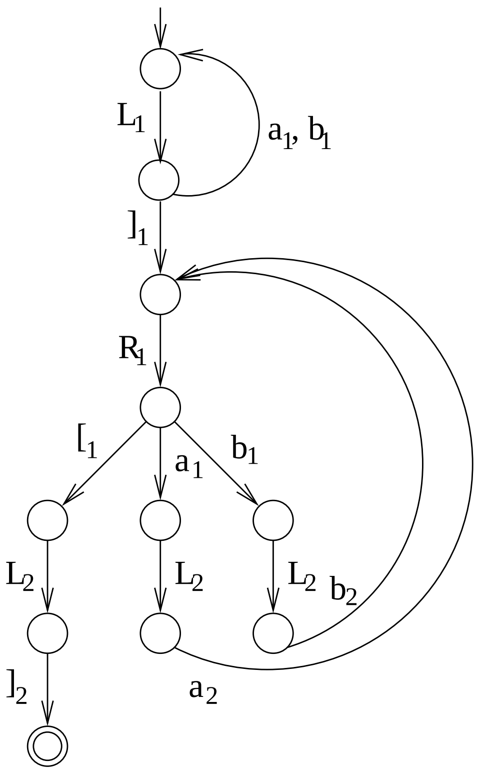
That is, alternatingly apply two reduction methods (with the elimination of useless states), until no more reduction of the automaton occurs.

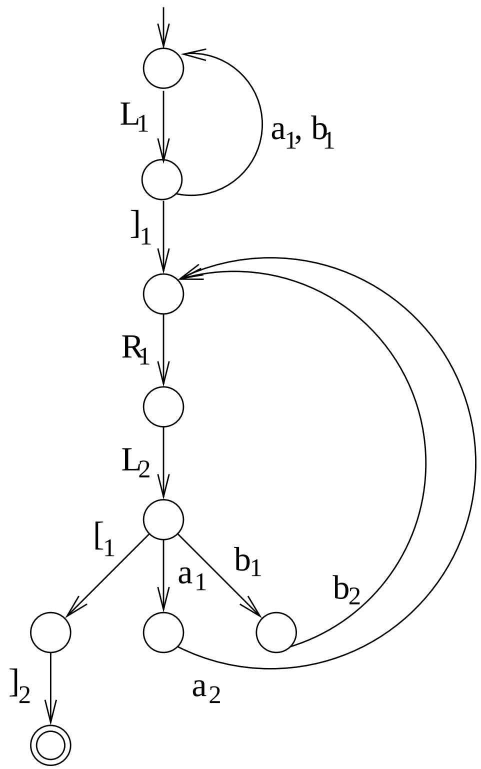
Example: applying the NFA reduction algorithm to the automaton for `reversal(x, y)` predicate





Example: applying the multitape automata reduction algorithm after NFA reduction





A more general multitape automata reduction algorithm

Apply two sequences of algorithms consisting of the NFA reduction procedure and the multitape automata reduction algorithm by turn on A, at one time starting with the NFA reduction algorithm and the other time starting with the multitape automata reduction algorithm, and stopping when no more size reduction occurs to A. Output the smaller of the resulting two automata.

Experimental results

String predicate	n	$\|\Sigma\|$	$\|A_{orig}\|$	$\|A_{exp}\|$	Automaton size during the reduction process			
reversal	2	2	17	23	Red_{NFA} 11	Red_{Multi} 9	Red_{NFA} 9	
					Red_{multi} 16	Red_{NFA} 11	Red_{multi} 9	Red_{NFA} 9
substring	2	2	11	18	Red_{NFA} 9	Red_{multi} 9		
					Red_{multi} 17	Red_{NFA} 9	Red_{multi} 9	
subsequence	2	2	11	17	Red_{NFA} 7	Red_{multi} 7		
					Red_{multi} 16	Red_{NFA} 7	Red_{multi} 7	
prefix	2	2	9	16	Red_{NFA} 7	Red_{multi} 7		
					Red_{multi} 15	Red_{NFA} 7	Red_{multi} 7	

String predicate	n	$\|\Sigma\|$	$\|A_{orig}\|$	$\|A_{exp}\|$	Automaton size during the reduction process			
concatenation	3	2	21	20	Red_{NFA} 13	Red_{multi} 12	Red_{NFA} 12	
					Red_{multi} 19	Red_{NFA} 13	Red_{multi} 12	Red_{NFA} 12
shuffle	3	2	21	51	Red_{NFA} 12	Red_{multi} 10	Red_{NFA} 10	
					Red_{multi} 45	Red_{NFA} 12	Red_{multi} 10	Red_{NFA} 10
overlap	3	2	15	48	Red_{NFA} 21	Red_{multi} 20	Red_{NFA} 20	
					Red_{multi} 44	Red_{NFA} 20	Red_{multi} 19	Red_{NFA} 19
edit distance	3	4	24	168	Red_{NFA} 28	Red_{multi} 27	Red_{NFA} 27	
					Red_{multi} 143	Red_{NFA} 28	Red_{multi} 27	Red_{NFA} 27