MINIMIZATION OF REACTIVE PROBABILISTIC AUTOMATA

Olga Siedlecka

Abstract: The problem of finite automata minimization is important for software and hardware designing. Different types of automata are used for modeling systems or machines with finite number of states. The limitation of number of states gives savings in resources and time. In this article we show specific type of probabilistic automata: the reactive probabilistic finite automata with accepting states (in brief the reactive probabilistic automata), and definitions of languages accepted by it. We present definition of bisimulation relation for automata's states and define relation of indistinguishableness of automata states, on base of which we could effectuate automata minimization. Next we present detailed algorithm reactive probabilistic automata's minimization with determination of its complexity and analyse example solved with help of this algorithm.

Keywords: minimization algorithm, reactive probabilistic automata, equivalence of states of automata, bisimulation relation.

ACM Classification Keywords: F. Theory of Computation, F.1 Computation by Abstract Devices, F.1.1 Models of Computation, Automata; F.4 Mathematical logic and formal languages, F.4.3 Formal Languages

Conference: The paper is selected from XIVth International Conference "Knowledge-Dialogue-Solution" KDS 2008, Varna, Bulgaria, June-July 2008

Introduction

The problem of finite automata minimization appeared in the end of fifties of last century and its main point is to find automata with the minimum number of states accepting the same language as input automata. During last fifty years many algorithms for minimization of finite deterministic automata came into existence, most of which (except Brzozowsky algorithm which is based on derivatives [Brzozowski, 1962]), is based on equivalence of states. One of the most popular minimization algorithms is Hopcroft and Ullman's algorithm with running time $O(|\Sigma|n^2)$ (where $|\Sigma|$ is the number of symbols in the alphabet, n is the number of states) [Hopcroft, 2000]. Another algorithm with the same time complexity, but better memory complexity ($O(|\Sigma|n)$) is Aho-Sethi-Ullman's algorithm [Aho, 2006]. The most efficient deterministic finite automata minimization algorithm is Hopcroft's algorithm [Hopcroft, 1971] with time complexity $O(|\Sigma|n\log n)$.

In the same period of time scientists were searching for another models of computation. They developed probabilistic automata [Rabin, 1963], which are extensions of Markov chains with read symbols [Sokolova, 2004], models of finite automata over infinite words [Thomas, 1990], timed automata [Alur, 1994], hybrid automata [Henzinger, 1998] etc. We can find their ontological review in article: [Kryvyi, 2007]. It became important to find minimization algorithms for new types of automata. So far minimization of reactive probabilistic automata hasn't been described.

Probabilistic automata

It exists many types of probabilistic automata which differs with properties, applications or probability distributions (continuous or discrete). Their review we can find in article [Sokolova, 2004]. Hereunder we itemize few of probabilistic automata's types with discrete probability distribution: the reactive automata, the generative automata, the I\O automata, the Vardi automata, the alternating model of Hansson, the Segala automata, the bundle probabilistic automata, the Pnueli-Zuck automata and others.

The algorithm showed in article was formulated for the reactive probabilistic automata.

A reactive probabilistic automata is a triple $PA=(Q, \Sigma, \delta)$, where Q is the finite set of states, Σ is the finite set of input symbols (an alphabet), δ is the transition probability function given by $\delta: Q \times \Sigma \to D(Q)$ (where $D(Q)$ is the set of all discrete probability distribution on the set Q) [Sokolova, 2004] .

An initial reactive probabilistic automata with accepting states is a five $PA=(Q, \Sigma, \delta, q_0, F)$, in which we have additionally two elements: q_0 - a member of Q, is the start state, $F \subset Q$ is the set of final (accepting) states.

After reading given symbol automata is in state of superposition of states: $p_0q_0+p_1q_1+...+p_nq_n$, where $p_0+p_1+...+p_n=1$. Henceforth we will use shorter name of probabilistic automata within the meaning of initial reactive probabilistic automata with accepting states. An example of this type of automata we show on figure 1.

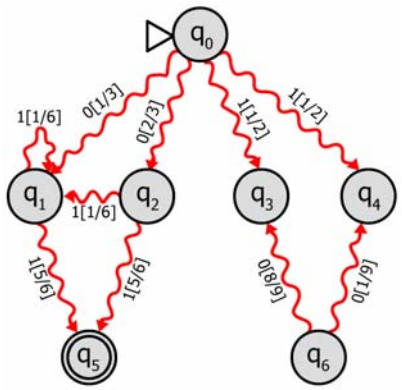

Fig.1. The initial reactive probabilistic automata with accepting states

The probability of going from state q_1 to state q_2 after reading symbol σ we denote as $\delta(q_1, \sigma)(q_2)=p$. An extended transition probability function, denoted by the same notation δ is given by:

$$\delta(q_1, w\sigma) = \sum_{q \in Q} \delta(q_1, w)(q) \cdot \delta(q, \sigma)$$ [Cao, 2006].

The language accepted by the probabilistic automata is defined as function $L_{PA}: \Sigma^* \to [0,1]$, such that:

$$\forall w \in \Sigma^*, L_{PA}(w) = \sum_{q \in F} \delta(q_0, w)(w)$$ [Cao, 2006].

We say that language L is recognized with bounded error by an automata PA with interval (p_1,p_1), if $p_1<p_2$ and $p_1=sup\{P_w|w \notin L\}$, $p_1=inf\{P_w|w \in L\}$ [Golovkins, 2002].

We say that language L is recognized with probability p, if the language is recognized with interval $(1-p,p)$ [Golovkins, 2002].

We say that language L is recognized with probability $1-\varepsilon$, if for every $\varepsilon>0$ there exist an automata which recognizes the language with interval $(\varepsilon_1, 1-\varepsilon_2)$, where $\varepsilon_1, \varepsilon_2 \leq \varepsilon$ [Golovkins, 2002].

Bisimulation and indistinguishableness

Let R be an equivalence relation on the set S, and let $P_1, P_2 \in D(S)$ be discrete probability distributions. Then $P_1 \equiv_R P_2 \Leftrightarrow \forall C \in S/R: P_1[C]=P_2[C]$, where C is an equivalence class [Sokolova, 2004].

Let R be an equivalence relation on the set S, let A be a set, and $P_1, P_2 \in D(S)$ be discrete probability distributions. Then:

$P_1 \equiv_{R,A} P_2 \Leftrightarrow \forall C \in S/R, \ \forall a \in A: P_1[a,C]=P_2[a,C]$ [Sokolova, 2004].

Let $PA_1=(S, \Sigma, \delta)$ and $PA_2=(T, \Sigma, \delta)$ be two reactive probabilistic automatas. A bisimulation relation $R \subseteq S \times T$ exists if for all $(s,t) \in R$ and for all $\sigma \in \Sigma$:

- if $\delta(s, \sigma)=P_1$ then there exists a distribution P_2 with $t \in T$ such that $\delta(t, \sigma)=P_2$ and $P_1 \equiv_{R,\Sigma} P_2$ [Sokolova, 2004].

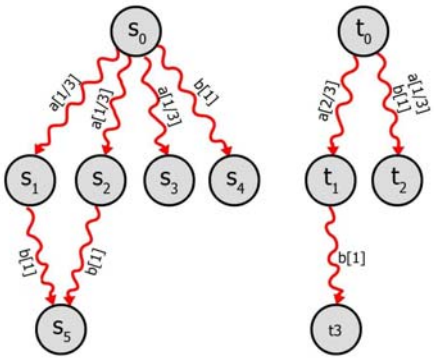

Fig.2. The bisimulation relation on PA

States $(s,t) \in R$ we call bisimilar, what is denoted by $s \approx t$.

Let $PA_1=(S, \Sigma, \delta, q_0, F_S)$ and $PA_2=(T, \Sigma, \delta, q_0, F_T)$ be two initial reactive probabilistic automata with accepting states. We can define indistinguishableness relation $N \subseteq S \times T$, if for all $(s,t) \in N$ and for all $\sigma \in \Sigma$:

- $(s,t) \in N^0$ if and only if $((s \in F_S \wedge t \in F_T) \vee (s \notin F_S \wedge t \notin F_T))$,

- $(s,t) \in N^k$ if and only if $(s,t) \in N^{k-1}$ and
  - if $\delta(s,\sigma)=P_1$ then exists the probability distribution P_2 with $t \in T$ such that $\delta(t,\sigma)=P_2$ and $P_1 \equiv_{R,\Sigma} P_2$.

For $n=|Q|$, we have $N \subseteq N^{n-2} \subseteq N^{n-3} \subseteq ... \subseteq N^1 \subseteq N^0$. States s,t we call indistinguishable, what is denoted by $s \equiv t$, if there exists indistinguishableness relation N, such that $(s,t) \in N$.

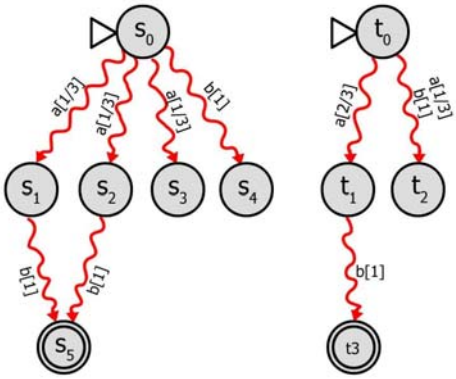

Fig.3. The indistinguishableness relation on PA

Minimization of reactive probabilistic automata

A probabilistic automata $PA=(Q, \Sigma, \delta, q_0, F)$ recognizing language L with probability p we call minimal, if there doesn't exist automata with smaller number of states recognizing language L with not smaller probability.

A minimization of probabilistic automata parts on two steps:

- elimination of unreachable states (probability to reach those states is 0),
- joining of indistinguishable states (using indistinguishableness relation).

First we show on below code elimination of unreachable states:

Alg.1. Algorithm of elimination of unreachable states:

```
INPUT: PA=(Q,Σ,δ,q0,F)- reactive probabilistic automata.
OUTPUT: PA'=(Q',Σ,δ',q0,F') - reactive probabilistic automata without unreachable
states, recognizing the same language as PA.
1.    FOR ALL {q∈Q} DO
2.      markedStates[q]←0;
3.    END FOR
4.      S.push(q0); markedStates[q]←1; pr←0;
5.    WHILE {S≠∅} DO
6.      p←S.first();
7.      S.pop();
8.      FOR ALL {σ∈Σ} DO
9.        FOR ALL {q∈Q} DO
10.         pr←δ(p,σ)(q);
11.         IF {pr≠0 ∧ markedStates[q0]=0} THEN
12.           S.push(q);
13.           markedStates[q]←1;
14.         END IF
15.       END FOR
16.     END FOR
17.   END WHILE
18.   FOR ALL {q∈Q} DO
19.     IF {markedStates[q]=1} THEN
20.       Q'.push(q);
21.     END IF
22.   END FOR
23.   F'←F∩Q;
24.   FOR ALL {q∈Q} DO
25.     IF {markedStates[q]=1} THEN
26.       FOR ALL {p∈Q} DO
27.         IF {markedStates[p]=1} THEN
28.           FOR ALL {σ∈Σ} DO
29.             δ'(q,σ)(p)←δ(q,σ)(p);
30.           END FOR
31.         END IF
32.       END FOR
33.     END IF
34.   END FOR
```

In this algorithm *S* is auxiliary stack, on which we put states, which we can reach with non-zero probability going out from the start state q_0. The transition probability function $\delta(p,\sigma)(q)$ gives probability *pr* of reaching state *q*, going out from state *p*, reading symbol σ. The running time of the algorithm time is bounded by:

$$T(n,|\Sigma|) \leq a(7+9n+2|\Sigma|n+2n^2+6|\Sigma|n^2)+c(4+8n+2|\Sigma|n+3n^2+5|\Sigma|n^2),$$

where *a* is time of an assignment and *c* is time of comparison, clearly $O(|\Sigma|n^2)$ is the time complexity of this algorithm.

In the algorithm of joining indistinguishable states we use already defined indistinguishableness relation. In one word, states to be indistinguishable, have to be in the same equivalence class, and must have the same probability distribution for symbols and equivalence classes, which can be reach from this states. Inspired by Hopcroft-Ullman's algorithm [Hopcroft, 2000], first we assume that all pairs of states are indistinguishable, above that, that first element of pair is member of final states' set and second isn't. Next analysing all pair of states and all symbols we find distinguishable states, until the moment that any change is made. Algorithm analyses probability distributions of reaching state from state.

Alg.2. Algorithm of joining indistinguishable states:

```
INPUT: PA=(Q,Σ,δ,q0,F) - reactive probabilistic automata.
OUTPUT: PA'=(Q',Σ,δ',q0',F') - minimal reactive probabilistic automata recognizing
language L_PA.
1.    FOR {i←0; i<|Q|; i←i+1} DO
2.      FOR {j←0; j≤i; j←j+1} DO
3.        IF {(qi∈F ∧ qj∉F) ∨ (qi∉F ∧ qj∈F)} THEN
4.          Dqi,qj←1;
5.        ELSE
6.          Dqi,qj←0;
7.        END IF
8.      END FOR
9.    END FOR
10.   FOR {i←1; i<|Q|; i←i+1} DO
11.     FOR {j←0; j<i; j←j+1} DO
12.       IF {Dqi,qj=0} THEN
13.         FOR ALL {σ∈Σ} DO
14.           E1←0, E2←0, N1←0, N2←0;
15.           FOR ALL {p∈Q} DO
16.             IF {Dqi,p=0} THEN
17.               E1←E1+δ(qi,σ)(p);
18.             ELSE
19.               N1←N1+δ(qi,σ)(p);
20.             END IF
21.             IF {Dqj,p=0} THEN
22.               E2←E2+δ(qj,σ)(p);
23.             ELSE
24.               N2←N2+δ(qj,σ)(p);
25.             END IF
26.           END FOR
27.           IF {E1≠E2 ∨ N1≠N2} THEN
28.             Dqi,qj←1; break;
29.           END IF
30.         END FOR
31.       END IF
32.     END FOR
33.   END FOR
34.   Q'←Q, F'←F, q0'←q0;
35.   FOR {i←1; i<|Q|; i←i+1} DO
36.     FOR {j←0; j<i; j←j+1} DO
37.       IF {Dqi,qj=0} THEN
38.         Q'←Q'\{qi,qj}, Q'←Q'∪{qij};
39.         IF{qi∈F} THEN
40.           F'←F'\{qi,qj}, F'←F'∪{qij};
```

```
41.          END IF
42.          IF {j=0} THEN
43.            q₀←qⱼ;
44.          END IF
45.        END IF
46.      END FOR
47.    END FOR
48.    FOR ALL {q₁×q₂×σ ∈ Q′×Q′×Σ} DO
49.      IF {q₁∉Q ∧ q₁=p₁p₂ : p₁p₂∈Q} THEN
50.        δ′(q₁,σ)(q₂)← δ(p₁,σ)(q₂);
51.      ELSIF {q₂∉Q ∧ q₂=p₁p₂ : p₁p₂∈Q } THEN
52.        δ′(q₁,σ)(q₂)← δ(q₁,σ)(p₁)+δ(q₂,σ)(p₂);
53.      ELSE
54.        δ′(q₁,σ)(q₂)← δ(q₁,σ)(q₂);
55.      END IF
56.    END FOR
```

Analyzing algorithm in details: on input we have reactive probabilistic automata; on output we get minimal automata that accept the same language as input automata. In lines 1 to 9 we tentatively fill structure *D*, which is lower triangular matrix of all combination of automata's states. In place where one of the states is final and second isn't, we set value 1, because states are distinguishable. In other case we set 0, providing that all other pairs of states are indistinguishable. In lines 10 to 33 is the main part of algorithm, which decides if states are equal or not, comparing probability distributions. First (line 12) we verify if pair of states is indistinguishable $D_{qi,qj}=0$ (otherwise it makes no sense in analyzing them). For every symbol from alphabet Σ we reset value of auxiliary variables *E1, E2, N1, N2*, in which we will sum probabilities of reaching distinguishable states *N* or indistinguishable states *E*. States will be generally recognized as indistinguishable if values of *E1, E2* and *N1, N2* will be respectively equal. If for two analyzed states, for any symbol of alphabet, we get different values of those variable, loop is interrupted (line 28), because states are distinguishable and we go to next iteration. In the last part of algorithm (from line 34) we create output automata, so we replace indistinguishable states by single states, and calculate values for transition probability function (from line 48). Depending, if we analyze reaching state or going out from new state, values of probability will be summed or copied. The running time of the algorithm is bounded by:

$$T(n,|\Sigma|) \leq a(5 + 4.5n - 3.5|\Sigma|n + 7.5n^2 + 2|\Sigma|n^2 + 3n^3 + 1.5|\Sigma|n^3) +$$
$$c(2 + 7n - 2.5|\Sigma|\, n + 7n^2 + |\Sigma|\, n^2 + 7n^3 + 1.5|\Sigma|n^3),$$

so complexity will be $O(|\Sigma|n^3)$.

Lets analyze steps of both algorithms on example from figure 1. First we reset table *markedStates[q_i]*, which size is 7 (automata has 7 states). We push on stack start state. Next we mark with *1* field for this state in table *markedStates[q_0]*. We pop from the stack start state and push those, which we can reach from start state reading symbol *0*, with nonzero probability (those will be q_1, q_2) and for symbol *1*, respectively q_3, q_4, in every case marking them with *1* in table *markedStates[q_i]*. In next iteration we search for states we can reach from states put on the stack. Finally, the only state, which wasn't marked is q_6. In next steps we exclude it from the set of states of automata.

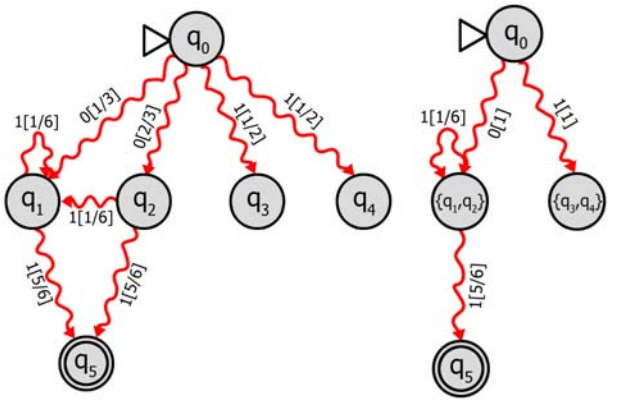

Fig.4. a) Elimination of unreachable states b) Joining of indistinguishable states

The algorithm of joining indistinguishable states in first part fill structure $D_{qi,qj}$ with *1* in those places where one of states is final, and second isn't – for all combinations of other states with state q_5. Next we check successively all

combinations of states and sum probabilities of going out from this states in variables *E1, E2, N1, N2*, for example for states q_1, q_0, values for this variables are *E1=0, E2=1, N1=0, N2=0*, so this pair of states is distinguishable and $D_{qi,qj}=1$. Finally structure $D_{qi,qj}$ has value *1* only for pairs: q_1, q_2 and q_3, q_4, which will be replaced by new single states q_{12}, q_{34}. Probabilities for reaching those states will be summed, and for going out from them will be copied.

Conclusion

In article we define indistinguishableness relation for reactive probabilistic automata, what give us opportunity to build minimization algorithm, with complexity $O(|\Sigma|n^3)$. Algorithms will terminate, because number of states or symbols in alphabet is always limitation for iterations (and we work on finite sets). The probability for accepting words doesn't change because it is respectively summed or copied.

The definition of indistinguishableness relation and minimization algorithm is the base for further work on adequate algorithm for quantum automata.

Bibliography

[Aho, 2006] A.V. Aho, M.S. Lam, R. Sethi, J.D. Ullman, Compilers: Principles, Techniques, and Tools (2nd Edition). Addison Wesley, 2006.

[Alur, 1994] R. Alur, D.L. Dill, A theory of timed automata. Theoretical Computer Science 126, 2 (1994), pp. 183 - 235.

[Brzozowski, 1962] J. A. Brzozowski, Canonical regular expressions and minimal state graphs for definite events. In Proceedings of the Symposium on Mathematical Theory of Automata (1962), vol. 12 of MRI Symposia Series, Polytechnic Press of the Polytechnic Institute of Brooklyn, pp. 529 - 561.

[Cao, 2006] Y. Cao, L. Xia, M. Ying, Probabilistic automata for computing with words. ArXiv Computer Science e-prints (2006).

[Golovkins, 2002] M. Golovkins, M. Kravtsev, Probabilistic reversible automata and quantum automata. Lecture Notes In Computer Science 2387 (2002), p. 574.

[Henzinger, 1998] T.A. Henzinger, P.W. Kopke, A. Puri P.Varaiya, What s decidable about hybrid automata? Journal of Computer and System Sciences 57 (1998), pp. 94 - 124.

[Hopcroft, 1971] J.E. Hopcroft, An n log n algorithm for minimizing the states in a finite automaton. In The Theory of Machines and Computations, Z. Kohavi, Ed. Academic Press, 1971, pp. 189 - 196.

[Hopcroft, 2000] J.E. Hopcroft, R. Motwani, J.D. Ullman, Introduction to Automata Theory, Languages, and Computation (2nd Edition). Addison Wesley, 2000.

[Kryvyi, 2007] S. Kryvyi, L. Matveeva, E. Lukianova, O. Siedlecka, The Ontology-based view on automata theory. In Proceedings of 13-th International Conference KDS-2007 (Knowledge-Dialog-Solution) (Sofia, 2007), ITHEA, Ed., vol. 2, pp. 427-436.

[Rabin, 1963] M.O. Rabin, Probabilistic automata. Information and Controle 6 (1963), pp. 230 - 245.

[Sokolova, 2004] A. Sokolova, E. de Vink, Probabilistic automata: System types, parallel composition and comparison. In Validation of Stochastic Systems: A Guide to Current Research (2004), LNCS 2925, pp. 1 – 43.

[Thomas, 1990] W. Thomas, Automata on infinite objects. Handbook of theoretical computer science: formal models and semantics B (1990), pp. 133 - 191.

Authors' Information

Siedlecka Olga – Institute of Computer and Information Sciences, Czestochowa University of Technology, ul. Dabrowskiego 73 42-200 Czestochowa, Poland; e-mail: olga.siedlecka@icis.pcz.pl