Modelling Dialogue Systems by Finite Automata

by

Ivan Kopeček
Libor Škarvada

FI MU Report Series FIMU-RS-2003-01

March 2003

Modelling Dialogue Systems by Finite Automata

Ivan Kopeček and Libor Škarvada
Faculty of Informatics
Masaryk University
Brno, Czech Republic
email: {kopecek|libor}@fi.muni.cz

Abstract

Based on finite state formalization, this paper deals with the problem of modelling and automatic programming of dialogue systems. The approach presented here is based on finding an operator (a construction) that assigns a corresponding dialogue system to a dialogue corpus. It will be shown that this construction is universal, in the sense that each dialogue system can be obtained in this way, and some related issues (uniqueness, algorithmization and algorithmical complexity) will be briefly discussed. The theory is illustrated on a simple example. Some related problems are formulated.

1 Introduction

Research in dialogue systems involves a wide range of approaches. The applications demand solutions to many problems of heterogeneous types, often having interdisciplinary features. In this situation, there is a natural need to have general models that can be applied to a variety of related problems, both theoretical and practical.

Finite automata (transducers, Mealy automata) are simple algebraic structures relating internal states to input and output sequences, offering a general model of the participant of a dialogue (agent).

Choosing specific attributes of the model, finite automata allow modelling different instances of the dialogue communication. For example, dialogue automata (a slight modification of Mealy automata) may relate emotions (as described by attributes of the internal states) with prosody (described by the attributes of the dialogue utterances) and other parameters that are entering the model [7]. The approach offers systemization and generalization, standardization of terms and notions, and the applicability of general algebraic methods.

A typical way of developing a dialogue system is based on simulating the system by Wizard of Oz simulation (see eg [3, 4]), developing the dialogue strategy based on the simulation and then programming it and testing it. A natural question arises in this connection: would it be possible to use the dialogue corpus that is created in the phase of system simulation for automatization of the dialogue system programming?

Based on the finite state formalization, we will, in what follows, analyse this question: Is it possible to construct a dialogue system from a dialogue corpus? Can we find effective algorithms to do this?

To explain the approach and the ideas behind it, we first briefly outline the finite state formulation of the dialogue system and present a simple example, on which the theory is demonstrated.

2 Preliminaries and notation

In what follows, we use the standard terms and notation used in the algebraic theory of formal languages and automata (see [8]). If M is a set, then M^* will denote the free monoid over the set M, ie the set of all strings consisting of the elements of the set M (including the empty string, denoted ε). An alphabet is a finite nonempty set. A language over an alphabet Σ is a subset of Σ^*; by corpus we usually mean a finite language. If U and V are sets, $U \times V$ denotes the Cartesian product of U and V.

For languages U, V we write $U{\cdot}V = \{uv \mid u \in U, v \in V\}$. If A is a finite set, $|A|$ means the number of elements in A; if u is a word (string of symbols), $|u|$ means its length; if A is an automaton, $|A|$ means the number of its states.

Deterministic finite automaton is a device which, starting in an initial state, reads an input word symbol by symbol and, based on its transition function, switches its internal state. Formally, it is a quintuple $A = (Q, \Sigma, \delta, s_0, F)$ of finite set Q of states, finite alphabet Σ, transition function $\delta : Q \times \Sigma \to Q$, initial state s_0, and set $F \subseteq Q$ of final states. We speak of partial deterministic finite automaton if its transition function is partial. In the paper deal with partial deterministic finite automata for which the sets of states and final states coincide, ie $F = Q$.

If A is an automaton, then $L(A)$ denotes the language generated (recognized) by A. If θ is an equivalence relation on the set of states of A then we construct a new automaton whose states are equivalence classes of θ, and which preserves the transition function and initiality/finality of states, lifted to equivalence-class level. The resulting finite automaton is called *quotient automaton* of A with respect to θ.

3 Finite state approach to dialogues

Let us recall that Mealy-type automaton, or finite transducer, (see eg [2]) is an ordered sextuple $A = (Q, X, Y, \delta, \lambda, s_0)$ where Q, X, Y are finite non-empty sets, $\delta : Q \times X \to Q$ and $\lambda : Q \times X \to Y$ are functions called transition function and output function, respectively, and $s_0 \in Q$ is a distinguished state, called *initial state*. Sets Q, X and Y are the sets of states, input symbols and output symbols, respectively. Being in state $s \in Q$, the automaton detects the actual input symbol, changes its state according to the transition function δ and outputs a symbol according to the output function λ. Hence, the Mealy-type automaton converts an input string (consisting of input symbols) into an output string (consisting of output symbols).

In order to analyse dialogue communication using formal models, we have to use formal definitions of some basic related notions. This means, of course, that our formal model is essentialy simplified in comparison to reality.

Let U be an alphabet consisting of possible utterances of the dialogue class we analyse. Then any string consisting of the elements of U may be considered as dialogue (over the alphabet U). In this description, the set of all possible dialogues over the alphabet U is identical to the free monoid over the alphabet U. In what follows, we will assume that each dialogue has an even number of elements (ie even length) or it is the empty string. This means, simply put, that to any utterance in a dialogue there is a reply. This slight modification does not affect the generality, because we can always add to our alphabet a symbol that means "no reply", covering the above-mentioned "no reply" situation. Let $u = (u_1, u_2, \ldots, u_n)$ be a dialogue. Then the odd indexed elements of u correspond to the utterances of the first participant of the dialogue and the even indexed elements correspond to the utterances of the second participant of the dialogue. The case when u is an empty string is interpreted as the empty dialogue. As mentioned above, it is assumed that n is an even number. The pairs $(u_1, u_2), (u_3, u_4), \ldots$ will be called steps of the dialogue.

The participants of the dialogue can be modelled by Mealy automata in a very natural way; the transition and output functions define how the corresponding participant of the dialogue responds to the previous utterance of the other participant (the participant will change its internal state according to the transition function) and how the dialogue will continue (this is given by the output function). Hence, in this model, the alphabet of the dialogue is the union of the alphabets of the participants of the dialogue. This model assumes that the first utterance of the dialogue is given. However, if we model computer dialogue systems in human-computer interaction, we can suppose that the initial utterance is given by human.

Indeed, in our paper we will use our abstract model for modelling computer dialogue systems. The finite state approach is supported by the fact that currently the most widely used dialogue programming standard VoiceXML can be seen as an abstract finite-state automaton (see eg [1, 9]). The second argument is that we can utilize the finite-state automata theory to solve the related problems.

Our assumption that the length of the dialogues is always even (or null) gives us the opportunity to consider dialogue as a strings of its steps. This view leads us to the simplification, which is described in Section 5. Instead of Mealy automata, we can take as a model for a computer dialogue system a similar, but somewhat simpler structure, prefix automaton.

In order the resulting Mealy automaton be deterministic its convenient to assume that the languages under consideration contain only "uniquely evolving" dialogs, ie those in which the second participant's replies are detrmined by the dialog history:

Definition 1 *A dialogue language $L \subseteq (\Sigma_1 \cup \Sigma_2)^*$ is* deterministic *if for every odd-length word $w \in (\Sigma_1 \cup \Sigma_2)^*$ there is at most one symbol $b \in \Sigma_2$ such that wb is a prefix of some dialog in L.*

In the following sections we will deal only with deterministic dialogue languages.

4 Example

In what follows we present a simple example of our approach. The example illustrates the basic notions presented above, and is used also in the following sections.

Example 1 Let us consider a very simplified situation that can appear when developing a dialogue library system. Suppose that the user specifies the author and the name of a book and the system returns the information about the presence of the book in the library. The user is not restricted in the form in which he or she will input the specification. It might be a speech utterance in a natural language, a freely formulated text sentence, author and name in any order, etc. (Of course, for such a simple system we could prompt the user sequentially for the data, but the effect of free conversation with computer appears in more complex situations, which can not be presented in such a brief example.) The free input format as well as user's errors may cause the system not to understand some part(s) of the input information.

We will see the input information from the "point of view" of the system, ie if the system does not understand a part of the input information, it will be considered as not understandable and denoted by X (regardless of whether it was really understandable or not).

For the sake of simplicity, denote *A* the author of the requested book and *T* the title of the book. Then, the input alphabet of our model can be expressed as the following set Σ_1:

$$\Sigma_1 = \{\mathit{AT}, \mathit{AX}, \mathit{XT}, A, T, X\}$$

(The symbol *AT* represents inputting both the author and the title by the user, symbols *AX*, *XT* represent the same situation but only one part is understandable.)

We choose the output alphabet of our model as the following set Σ_2:

$$\Sigma_2 = \{I, \mathit{RA}, \mathit{RT}, \mathit{R\text{-}all}\}$$

With the following meaning:
I = tell the user the requested information;
RA = ask the user to repeat the name of the author;
RT = ask the user to repeat the title of the book;
R-all = ask the user to repeat both the name of the author and the title of the book.

Now, our dialogue corpus C may look as follows: $C = \{d_1, d_2, \ldots, d_9\}$

$$\begin{aligned}
d_1 &= (\mathit{AT}, I)\\
d_2 &= (\mathit{AX}, \mathit{RT}, T, I)\\
d_3 &= (\mathit{AX}, \mathit{RT}, X, \mathit{RT})\\
d_4 &= (\mathit{XT}, \mathit{RA}, A, I)\\
d_5 &= (\mathit{XT}, \mathit{RA}, X, \mathit{RA})\\
d_6 &= (X, \mathit{R\text{-}all}, \mathit{AT}, I)\\
d_7 &= (X, \mathit{R\text{-}all}, \mathit{AX}, \mathit{AR})\\
d_8 &= (X, \mathit{R\text{-}all}, \mathit{XT}, \mathit{RA})\\
d_9 &= (X, \mathit{R\text{-}all}, X, \mathit{R\text{-}all})
\end{aligned}$$

For this simple situation, it can be shown that a minimal Mealy automaton that covers the corpus C is the following automaton A:

$$A = (Q, \Sigma_1, \Sigma_2, f, g, s_0)$$

where $Q = \{s_0, s_1, s_2\}$ and the transition function f and output function g are defined as follows:

$$\begin{array}{lcllcl}
f(s_0, \mathsf{AT}) & = & s_0, & g(s_0, \mathsf{AT}) & = & \mathsf{I}, \\
f(s_0, \mathsf{AX}) & = & s_1, & g(s_0, \mathsf{AX}) & = & \mathsf{RT}, \\
f(s_0, \mathsf{XT}) & = & s_2, & g(s_0, \mathsf{XT}) & = & \mathsf{RA}, \\
f(s_0, \mathsf{X}) & = & s_0, & g(s_0, \mathsf{X}) & = & \mathsf{R\text{-}all}, \\
f(s_1, \mathsf{X}) & = & s_1, & g(s_1, \mathsf{X}) & = & \mathsf{RT}, \\
f(s_1, \mathsf{T}) & = & s_0, & g(s_1, \mathsf{T}) & = & \mathsf{I}, \\
f(s_2, \mathsf{X}) & = & s_2, & g(s_2, \mathsf{X}) & = & \mathsf{RA}, \\
f(s_2, \mathsf{A}) & = & s_0, & g(s_2, \mathsf{A}) & = & \mathsf{I}
\end{array}$$

5 Prefix automata and formulation of the problem

Let Σ_1 and Σ_2 be alphabets of dialogue utterances of the first and the second participant respectively. Let us have a language L over $\Sigma_1 \cup \Sigma_2$ of dialogues and its finite sublanguage—corpus C. It is often the case that the dialogues not covered by C are represented in C by their prefixes, in other words, the dialogues in $L - C$ are extensions of those in C. Or the language L may contain prefixes of dialogues in C.

We aim to constructing a Mealy automaton whose behaviour covers the corpus. By "covering" we mean that the automaton generates a language which is *prefix similar* to the corpus in the sense of Definition 6.

Definition 2 *Let L be a language over alphabet Σ. The language $L{\cdot}\Sigma^* = \{uv \mid u \in L, v \in \Sigma^*\}$ is called* postfix-closure *of L.*

Definition 3 *Let L, L' be languages over the alphabet Σ. L' will be called* postfix extension *of L, if L' contains L and L' is a subset of the postfix-closure of L.*

Definition 4 *Let L be a language over the alphabet Σ. The language $\{w \in \Sigma^* \mid \exists u{\in}\Sigma^*.\ wu \in L\}$ is called* prefix-closure *of L and is denoted $\bar{L}$.*
A language L for which $L = \bar{L}$ is called prefix-closed.

Definition 5 *Let L be a language over the alphabet Σ. If there are no two words $u, v \in L$ such that u is a prefix of v then the language L is called* prefixless.

Definition 6 *A language L such that $\bar{C} \subseteq L \subseteq \bar{C} \cup C{\cdot}\Sigma^*$ is called* prefix-similar *to C.*

Thus our desired automaton should generate all dialogues from the corpus, all their prefixes, and some of their extensions.

We can make a simplification of the problem. Notice that all the considered dialogues are of even length and the utterances from Σ_1 alternate with those from Σ_2 (cf Section 3). Let $\Sigma = \Sigma_1 \times \Sigma_2$ be a new alphabet—alphabet of dialogue steps. Let C be a corpus, same as the original one, but with adjacent utterances of the dialogues paired into dialogue steps. For example the dialogue $(\mathsf{AX}, \mathsf{RT}, \mathsf{T}, \mathsf{I})$ from Example 1 becomes $((\mathsf{AX},\mathsf{RT}), (\mathsf{T},\mathsf{I}))$ etc.

Now we can switch from Mealy automata to a more convenient notion of deterministic finite automata. Finite automata differ from Mealy automata in that they do not produce any output. They just consume an input string, switching their internal state according to their transition function. Some of their states are distinguished as *final*. When the automaton stops, after consuming the whole input, in a final state, the input word is called *accepted*. Otherwise it is *rejected*. The set of accepted words is called the language *generated* or *recognized* by the automaton. For details see [8].

Among variants of finite automata we deal with partial deterministic finite automata with all states final. Only the last condition, viz finality of all states, restricts the class of generated languages—namely to prefix-closed regular languages. (Neither the partiality nor the determinism does.) The reason why we do not need nonfinal states is that it is sufficient to deal with prefix-closed languages. If a language L is prefix-similar to corpus C, then its prefix closure $\bar{L}$ is prefix-similar to C as well. Additionally, the finite automata without nonfinal states cohere better with Mealy automata in the sense that we can more easily switch from them back to Mealy automata.

Definition 7 *By* prefix automaton *we understand a partial deterministic finite automaton whose set of states and set of final states are equal. Instead of quintuple* $(Q, \Sigma, \delta, s_0, Q)$ *we use only quadruple* (Q, Σ, δ, s_0) *for its description.*

The problem we aim to solve in this paper is:

Problem 1 *Given a corpus C, as a finite subset of some deterministic dialogue language, find a minimal prefix automaton generating a language which is prefix-similar to C.*

In Section 7 we show how to construct a prefix automaton from a given corpus and how to transform this prefix automaton to Mealy automaton.

6 Automata assigned to the corpus and solvability of the problem

Notation We introduce the following notation:
$F(C) = \{A \mid A$ generates a language prefix-similar to $C\}$;
$f(C) = \{A \in F(C) \mid$ if $A' \in F(C)$ then $|A| \leq |A'|\}$.

It is well known that that there may be many equivalent finite automata generating the same language L. But among them the one with minimum number of states is determined uniquely (up to renaming of states). If the condition that the automaton should generate the given language is relaxed to that it should generate a languge prefix-similar to L, then there is essentially more ambiguity.

Example 2 Let $\Sigma = \{a, b\}$, $C = \{a, ab, abb\}$, $L_1 = \{a^i b^j \mid i > 0, j \geq 0\}$, $L_2 = \{aw \mid w \in \Sigma^*\}$. Both L_1, L_2 are prefix-similar to C, and the corresponding prefix automata A_1, A_2 have both two states. None of L_1, L_2 can be generated by a one-state automaton. Furthermore, A_1 and A_2 differ because $L_1 \neq L_2$. Hence $A_1 \in f(C)$, $A_2 \in f(C)$, and thus $|f(C)| \geq 2$.

Moreover, Example 2 shows that even if we restrict the set of prefix automata to prefix automata with a minimum number of states, the nonuniqueness is still there.

On the oher hand, the following results in this section show that if we have a corpus C with $|f(C)| > 1$, we can always modify corpus C to C' so that $|f(C')| = 1$.

Lemma 1 *Let C be a corpus and C' a postfix extension of C. If L' is prefix-similar to C', then L' is prefix-similar to C as well.*

Proof: Let $w \in L'$. According to the definition of prefix-similarity, the following cases may occur:
(1) $w \in C'$; then there are $u \in C$ and v such that $w = uv$, ie w is an extension of a word belonging to C. Hence, $w \in C{\cdot}\Sigma^*$.
(2) $w \in C'{\cdot}\Sigma^*$; then there are $u \in C$ and v, s such that $w = uvs$, ie even in this case w is an extension of a word belonging to C. Hence, $w \in C{\cdot}\Sigma^*$.
(3) $w \in \bar{C}'$; then w is a prefix of uv where $u \in C$. If w is a prefix of u, then $w \in \bar{C}$. If w is not a prefix of u, then it is a postfix extension of u, ie $w \in C{\cdot}\Sigma^*$.
In all cases either $w \in C{\cdot}\Sigma^*$, or $w \in \bar{C}$. Further, L' contains C', and C' contains C. Hence, according to the definition of prefix-similarity, L' is prefix-similar to C.

Lemma 2 *Let C be a prefix closed corpus and C' a postfix extension of C. Then $f(C') \subseteq f(C)$.*

Proof: The assertion easily follows from Lemma 1.

Lemma 3 *For any corpus C it holds $f(C) = f(\bar{C})$.*

Proof: It can be easily seen that a language is prefix similar to C if and only if it is prefix similar to $\bar{C}$. Hence, $F(C) = F(\bar{C})$, which implies the assertion.

Proposition 1 *For any corpus C there is a corpus C' such that C' is a postfix extension of C and $|f(C')| = 1$.*

Proof: Let A be a prefix automaton from $f(C)$. According to Lemma 3 $f(C) = f(\bar{C})$, hence $A \in f(\bar{C})$. Let us denote $L(A, n) = \{u \in L(A) \mid |u| \leq n\}$. As $L(A, n)$ is a postfix extension of $\bar{C}$ for every $n \geq \max\{|u| \mid u \in A\}$, we have (according to Lemma 2) $f(L(A, n)) \subseteq f(\bar{C}) = f(C)$. Let m be an integer such that all the sets $L(X, m)$, where $X \in f(C)$, differ. Since the set $f(C)$ is finite, such a number exists (otherwise at least two prefix automata belonging to $f(C)$ would produce the same language, which would contradict our assumptions about $f(C)$). Now, the corpus $L(A, m)$ has the following properties:
(1) $L(A, m)$ is a postfix extension of C;
(2) $f(L(A, m)) \subseteq f(C)$;
(3) if $X \in f(C)$ and $X \neq A$, then$X \notin f(L(A, m))$.
These properties of $L(A, m)$ imply that there is only one minimal prefix automaton that generates a corpus prefix-similar to $L(A, m)$. As $L(A, m)$ is a postfix extension of $\bar{C}$, we obtain the assertion.

Proposition 2 *For any minimal prefix automaton A there is a corpus C such that $A \in f(C)$.*

Proof: Let A be a minimal prefix automaton and let us denote $L(A, n) = \{u \in L(A) \mid |u| \leq n\}$. Further, let S be the set of all languages that are generated by prefix automata that have less or equal number of states as the automaton A. Of course, the set S is finite. Let m be a number such that all the sets $L(X, m)$, where $X \in S \cup L(A)$, differ. Because the set $S \cup L(A)$ is finite, such a number exists. Then $A \in F(L(A, m))$, A is minimal, and any other prefix automaton from $F(L(A, m)$ has greater or equal number of states. Hence, if we take $C = L(A, m)$ we get $A \in f(C)$.

Corollary 1 *For any minimal prefix automaton A there is a corpus C such that there is exactly one prefix automaton satisfying $A \in f(C)$.*

Proof: follows directly from Propositions 1 and 2.

Proposition 2 ensures that by means of generating dialogue systems from corpora we can get any dialogue system. >From Corollary 1 then follows that it can be done uniquely.

An interesting problem (not solved here) is

Problem 2 *Characterize the corpora for which there is exactly one minimal prefix automaton.*

7 Construction

We aim to construct a prefix automaton which generates a language prefix-similar to corpus C, and has minimum number of states among such automata.

The construction of a prefix automaton generating a language prefix-similar to corpus C will be as follows:

First, we construct a deterministic finite automaton T generating exactly C. For this we may choose any automaton of the class of equivalent finite automata, but the most straightforward construction leads to the *trie automaton*, see Definition 8.

Second, we define relation ρ on set of states of T with the following property: Any equivalence relation $\theta \subseteq \rho$ determines the corresponding quotient automaton T/θ whose language $L(T/\theta)$ is a postfix extension of C.

Third, modifying the quotient automaton by making all its states final we get a prefix automaton that generates a language prefix-similar to C.

By *trie automaton* $T(C)$ we understand a partial deterministic finite automaton generating exactly the corpus C, whose diagram is a tree.

Definition 8 *Let C be a corpus over alphabet Σ. Define $T(C) = (\bar{C}, \Sigma, \delta, \varepsilon, C)$ where, for every $w \in \bar{C}$ and every $a \in \Sigma$, if $wa \in \bar{C}$ then $\delta(w, a) = wa$ else $\delta(w, a)$ is undefined.*

For the definition and applications of tries see [5].

Example 3 Recall the corpus from Example 1. The diagram of its trie automaton is depicted on Fig 1.

If the corpus C is prefixless (recall Definition 5), the final states of $T(C)$ are exactly the leaves of its diagram. For the rest of this section we assume that C is prefixless.

The relation ρ, on which the construction is based, is defined similarly to bisimulation equivalence [8], but is larger.

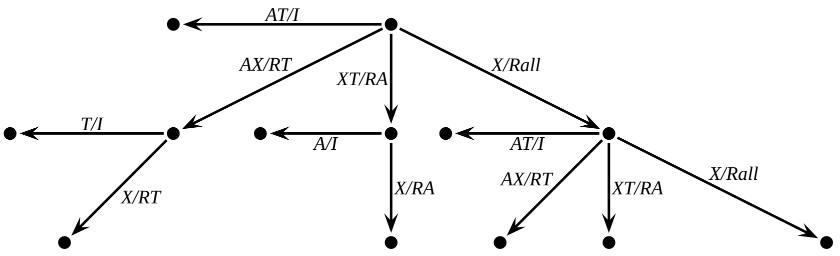

Figure 1: Trie automaton from Example 3

Definition 9 *Let $T = T(C)$ be our trie automaton, denote $T = (Q, \Sigma, q_0, \delta, F)$. Define relation $\rho \subseteq Q \times Q$ as the greatest relation satisfying the following condition: For every two states, $p, q \in Q$, $(p, q) \in \rho$ if and only if*
(i) $p \in F$, or
(ii) $q \in F$, or
(iii) for every $a \in \Sigma$, if $\delta(p, a) = p'$ then there exists a state $q' \in Q$ such that $(p', q') \in \rho$, and for every $a \in \Sigma$, if $\delta(q, a) = q'$ then there exists a state $p' \in Q$ such that $(p', q') \in \rho$.

Notice that ρ is not an equivalence relation because it is not transitive.

Proposition 3 *Let C is a prefixless corpus, $T = T(C)$, and let θ be any equivalence relation under the relation ρ (from Definition 9), $\theta \subseteq \rho$. Then*
(1) every word $w \in C$ is accepted by T/θ, and
(2) for every word w accepted by T/θ, either $w \in \bar{C}$ or $w \in C{\cdot}\Sigma^*$ holds.

Proof: The proof of part (1) is straightforward and follows immediately from the construction of T/θ.

Let $w = a_1 \dots a_n \in L(T/\theta)$ be a nonempty word (for the empty word the proposition is trivial). We show that w is either a prefix or an extension of some word in C. Let q_0 be an initial state of T, $q_1, \dots, q_n$ some states of T representing states of T/θ, and let $[q_0]a_1[q_1]a_2 \dots a_n[q_n]$ be the accepting run of T/θ for w. There exist some states r_0, r_1 in T, $r_0 \in [q_0]$, $r_1 \in [q_1]$ such that $\delta(r_0, a_1) = r_1$ and $(r_0, q_0) \in \theta$, $(r_1, q_1) \in \theta$. Hence $(r_0, q_0) \in \rho$, $(r_1, q_1) \in \rho$. By Definition 9, either $r_0 \in F$, or $q_0 \in F$, or there is an a_1-transition from q_0 to some state p_1, $(p_1, r_1) \in \rho$. Case $r_0 \in F$ is impossible since there are no transitions from final states in T. If $q_0 \in F$ then $\varepsilon \in C$ and w is an extension of ε. Let $p_1 = \delta(q_0, a_1)$, $(p_1, r_1) \in \rho$. Again, by Definition 9, either $p_1 \in F$, of $r_1 \in F$, or there is an a_2-transition from p_1 to some state p_2, $(p_2, \delta(r_1, a_2)) \in \rho$. If $p_1 \in F$ then $a_1 \in C$, and w is equal to a_1 or is its extension. If $r_1 \in F$ then $w = a_1$ and we are done. If the third case holds then $[p_1]a_2[q_2]a_3[q_3]a_4 \dots a_n[q_n]$ is a run of automaton T/θ starting in state $[p_1]$ and accepting word $a_2 \dots a_n$. Following the same steps (by induction on the length of w) we show that there is a run $q_0a_1p_1a_2p_2a_3 \ldots a_np_n$ of automaton T accepting the word w or its extension. As the word w may end sooner than T reaches its final state, w can also be a prefix of a word of $L(T) = C$.

As an immediate consequence of Proposition 3 we get

Corollary 2 *For any equivalence relation $\theta \subseteq \rho$ the language $L(T(C)/\theta)$ is prefix-similar to C.*

Given an equivalence $\theta \subseteq \rho$ we take the automaton T/θ, make all its states final, getting automaton B, which generates a language prefix-similar to C. Then the automaton B accepts all words in $\bar{C}$ plus some extensions of words in C.

Finally we may return from the prefix automaton back to Mealy automaton. Let $A = (Q, \Sigma, \delta, s_0)$ be a prefix automaton over alphabet $\Sigma = \Sigma_1 \times \Sigma_2$. Then the corresponding Mealy automaton $A' = (Q', \Sigma_1, \Sigma_2, f, g, s'_0)$ can be constructed in the following way: $Q' = Q$, $s'_0 = s_0$, and for each $s \in Q'$, $a \in \Sigma_1$, both $f(s,a) = t$, $g(s,a) = b$, whenever there is some $b \in \Sigma_2$ such that $\delta(s,(a,b)) = t$. A necessary condition for uniqueness of this step is that there is as most one such symbol b. But this is guaranteed by the determinism of our dialogue language.

8 Conclusions and future work

We have shown that abstract models can effectively simulate behaviour of the dialogue systems in human-computer interaction and that using this model we can get interesting simple formulated problems in which the technique of finite-state automata theory can be exploited. In the paper, we have presented basic results related to the concept of automatically building computer dialogue systems from dialogue corpora.

Some problems, however, remain open. They are related to the question of effective algorithmization of the proposed method.

We think that for a given corpus *some* prefix automaton can be found effectively in polynomial time. However, we conjecture that finding a *minimal* automaton is NP hard.

The next important issue is implementation of the method for real dialogue systems. Besides, the approach provokes some interesting theoretical problems that are related to automata theory, program verification, bisimulation etc.

References

[1] Cenek, P.: Dialogue Interfaces for Library Systems, *FI MU Report Series*, FIMU-RS-2001-04, June 2000.

[2] Gecseg, F., Peak, I.: *Algebraic Theory of Automata*, Akademiai Kiado, Budapest, 1972

[3] Fraser, N.M., Gilbert, G.N.: Simulating Speech Systems. *Computer Speech and Language*, 5, 1991, pp. 81–89

[4] Good, M.D., Whiteside. J.A., Wixon, R.R., Jones, S.J.: Building a User-derived Interface, *Communications of the ACM*, 27(10), 1984, 1032–1043

[5] Knuth, D.E.: *The art of computer programming, Sorting and searching*, Addison-Wesley, 1998

[6] Kopeček, I.: Modeling of the Information Retrieval Dialogue Systems. In *Proceedings of the Workshop on Text, Speech and Dialogue – TSD'99*, Lectures Notes in Artificial Intelligence 1692, Springer-Verlag, 1999, pp. 302–307

[7] Kopeček, I.: Emotions and Prosody in Dialogues: An Algebraic Approach Based on User Modelling. In *Proceedings of the ISCA Workshop on Speech and Emotions*, Belfast, 2000, pp. 184–189

[8] Kozen, Dexter C.: *Automata and Computability*, New York, Springer, 1997

[9] *VoiceXML Forum—Voice extensible Markup Language VoiceXML*, available at `http://www.voicexml.org/specs/VoiceXML-100.pdf`