HARDWARE IMPLEMENTATION OF LANGUAGE RESOURCES FOR EMBEDDED SYSTEMS

Matej Rojc, Zdravko Kačič, Iztok Kramberger

Institute of Electronics, Faculty of Electrical Engineering and Computer Science, Maribor, Slovenia

Key words: finite-state machines, finite-state transducers, phonetic and morphological lexicons, spoken dialogue applications, Atmel flash memory, Atmel microcontroller

Abstract: A lot of external natural language resources are used in spoken dialogue systems. These resources present considerable problems because of the needed space and slow lookup-time. It is, therefore, very important that the presentation of external language resources is time and space efficient. It is also very important that new language resources are easily incorporated into the system, without modifying the common algorithms developed for multiple languages. This paper presents the method and results of compiling the large Slovenian phonetic and morphology lexicons (SIflex and SImlex) into corresponding finite-state transducers (FSTs). Representation of large lexicons using finite-state transducers is mainly motivated by considerations of space and time efficiency. In addition the approach of hardware implementation for both large (Slovenian) lexicons is described. We will demonstrate that the structure of the FST is very appropriate for storing in the Atmel AT49BV161 flash memory chip and the lookup algorithm for obtaining any desired information from the FST structure can be efficiently implemented using the Atmel AT90S8515 microcontroller. The described hardware implementation of both Slovenian lexicons can be connected directly to the PC using RS232 above all for development and test purposes and can be used especially in embedded systems which use speech technology.

Strojna implementacija jezikovnih virov za uporabo v vdelanih sistemih

Ključne besede: končni stroji, končni pretvorniki, fonetični in morfološki leksikoni, Atmel flash pomnilnik, Atmel mikrokrmilnik

Izvleček: V govornih sistemih dialoga se uporablja mnogo jezikovnih virov. Uporaba obsežnih jezikovnih virov predstavlja velik problem tako zaradi porabljenega pomnilniškega prostora, kot tudi zaradi počasnega dostopanja do željene informacije. Zato je zelo pomembno, da je predstavitev uporabljenih virov časovno in pomnilniško optimalna. Pri večjezičnih sistemih je pomembna tudi preprosta vključitev jezikovnih virov drugih jezikov v sam sistem, ne da bi bilo pri tem potrebno spreminjati skupne algoritme razvite za več jezikov. V članku predstavljamo metodo in rezultate prevajanja obsežnega slovenskega fonetičnega in morfološkega leksikona (SIflex in SImlex) v pripadajoča končna pretvornika (FST). Takšna predstavitev je izredno učinkovita tako glede porabe pomnilniškega prostora, kot tudi časa, potrebnega za dostop do informacije. Podrobneje bo predstavljen tudi pristop strojne implementacije obeh Slovenskih leksikonov. Struktura končnih pretvornikov je zelo primerna za njihov zapis v "flash" pomnilniško integrirano vezje (Atmel AT49BV161), algoritem pridobivanja informacije iz strukture končnega pretvornika pa lahko enostavno implementiramo z uporabo mikrokrmilnika (Atmel AT90S8515). Opisano strojno implementacijo obeh slovenskih leksikonov lahko priklopimo direktno na PC računalnik preko RS-232 serijskih vrat, predvsem za razvojne namene in testiranje. Posebej zanimiva pa je uporaba predstavitve leksikonov s končnimi stroji in njihove v članku predlagane strojne implementacije. Oba leksikona, predstavljena s končnimi stroji, lahko učinkovito uporabimo v poljubnih vdelanih sistemih, ki uporabljajo govorno tehnologijo.

1. Introduction

When using voice, at least speech recognition and text-to-speech synthesis technology should be integrated as significant constituent part of an embedded mobility suite in order to operate most of the common PDA (Personal digital assistance) functions such as e-mail, tasks, calendar, phone numbers and addresses. Both should allow natural way of communication using such devices. On the other hand the development of real models of human language that support research and technology development in language related fields requires a lot of linguistic data - lexicons containing thousands of words. In order to achieve the same language coverage, as in the case of e.g. the English language, such lexicons need to be up to ten times larger in the case of inflectional languages. Having a lot of Slovenian root forms can result in up-to 200 different inflectional forms. The use of such resources can, therefore, represent substantial computational load especially for embedded mobility systems, demanding low energy consumption and the smallest possible implementation. A given spoken language system, which uses fully inflected word forms, performs much worse with highly inflected languages (e.g. Slovenian) than with non or purely inflected languages (e.g. English), where the lexicons used can be much smaller. In general, external language resources (phonetic, morphology lexicons etc.) present a problem regarding memory usage and the time spent on lookup processes.

Finite-state machines are already used in many areas of natural language processing. Their use from the computational point of view is mainly motivated by considerations of space and time efficiency. Linguistically, finite-state machines allow an easier description of most of the relevant local phenomena in the language /1/. They also provide compact representation of the specific external language resources needed for knowledge representation in the automatic text-to-speech synthesis systems. These features of finite-state machines are of major importance especially, when dealing with spoken dialogue systems.

In the following sections an approach for compiling such lexicons into finite-state transducers is first presented that represent their time and space optimal representation. The effect of using finite-state transducers for the representation of external natural language resources means a greater reduction in the memory usage required by the lexicons, and an optimal access time (required for obtaining information) which is independent of the lexicons' sizes. In the following sections the whole compilation process into finite-state transducers is presented plus the results obtained for the described Slovenian lexicons (SIflex and SImlex). The lexicon representation appropriate for hardware implementation is then discussed in more detail. In conclusion the hardware implementation is presented of both lexicons (FST's) for use in embedded system.

2. Finite-state automata and finite-state transducers

2.1. Finite-state automata (FSA)

Finite-state automata (FSA) /2/ can be seen simply as an oriented graph with labels on each arc. Their fundamental theoretical properties make FSAs very flexible, powerful and efficient. FSAs can be seen as defining a class of graphs and also as defining languages.

2.1.1. Definition

A finite-state automaton A is a 5-tuple (Σ, Q, i, F, E) where Σ is a finite set called the alphabet, Q is a finite set of states, $i \in Q$ is the initial state, $F \subseteq Q$ is the set of final states, and $E \subseteq Qx(\Sigma \cup \{\epsilon\})xQ$ is the set of edges.

FSAs have been shown to be closed under union, Kleen star, concatenation, intersection and complementation, thus allowing for natural and flexible descriptions. In addition to their flexibility due to their closure properties, FSAs can also be turned into canonical forms that allow for optimal time and space efficiency /3/.

2.2. Finite-state transducer (FST)

FSTs can be interpreted as defining a class of graphs, a class of relations on strings, or a class of transductions on strings /1/. On the first interpretation, an FST can be seen as an FSA, in which each arc is labeled by a pair of symbols rather than by a single symbol.

Definition
A finite-state transducer T is a 6-tuple $(\Sigma_1, \Sigma_2, Q, i, F, E)$ such that:

- Σ_1 is a finite alphabet, namely the input alphabet
- Σ_2 is a finite alphabet, namely the output alphabet
- Q is a finite set of states
- $i \in Q$ is the initial state
- $F \subseteq Q$ is the set of final states
- $E \subseteq Qx\Sigma^*_1x\Sigma^*_2xQ$ is the set of edges

As with FSAs, FSTs are also powerful because of the various closure and algorithmic properties.

3. Use of FSMs for time and space optimal Lexicon representation

In general, when representing lexicons by automata, many entries share the same codes (strings, representing some piece of information). The number of codes is then small compared to the number of entries. Newly developed lexicons are more and more accurate and the number of codes can increase considerably. The increase in number of codes also increases the smallest possible size of such lexicons. During the construction of the automaton one needs to distinguish different codes, therefore space required for an efficient hashing of the codes can also become costly. Available lexicons that were used in this experiment suggest that the representation by automata would be less appropriate. Since morphological and phonetic lexicons can be viewed as a list of pairs of strings, their representation using finite-state transducers seems to be very appropriate. Representation of lexicons using finite-state transducers on the other hand also provides reverse look-up capability.

The methods used in the compilation of large scale lexicons into finite-state transducers (FST) assume that the lexicons are given as large list of strings and not as a set of rules as considered by Kaplan and Kay for instance /1/. In Fig. 1 some items from Slovenian phonetic (SIflex) and morphology lexicon (SImlex) are shown. Both lexicons were compiled into corresponding finite-state transducers, using proprietary toolkit *fsm*HAL. It consists of a large set of various algorithms and tools for FSM manipulation and is written is C++ program lanugage. During the compilation process the following algorithms were used: *union, determinization*, and *minimization* (Aho, 1974; Watson, 1995), (Mohri,1995).

mod/el	model
m O - d /e: l	mod/el.N:cmsn:cmsa
mod/ela	modela
m O - d /e: - l a	mod/el.N:cmsg:cmdn:cmda
mod/elu	modelu
m O - d /e: - l u	mod/el.N:cmsd:cmsl
mod/elom	modelom
m O - d /e: - l O m	mod/el.N:cmsi:cmpd
mod/eloma	modeloma
m O - d /e: - l O - m a	mod/el.N:cmdd:cmdi
mod/elih	modelih
m O - d /e: - l i x	mod/el.N:cmdl:cmpl
mod/eli	modeli
m O - d /e: - l i	mod/el.N:cmpn:cmpi
mod/ele	modele
m O - d /e: - l E	mod/el.N:cmpa
a)	b)

Figure 1: Slovenian phonetic (a) and morphology lexicons (b). Slovenian morphology lexicon (SImlex) is coded according to Sampa /5/ and Multext specifications /6/.

Figure 2: Part of Slovenian phonetic lexicon (SIflex) represented as FST.

The representation using finite-state transducers was performed for the SIflex and SImlex Slovenian lexicons. The starting size for SIflex was 1.8 MB (60.000 items) and 1.4 MB for SImlex (40.000 items). The final size achieved using the presented algorithms was 352 kB for SIflex and 662 kB for SImlex (Table 1) /4/. Representation of large lexicons using finite-state transducers is mainly motivated by considerations of space and time efficiency. For both lexicons a great reduction in size and optimal access time was achieved. Using such representation the look-up time is optimal, since it depends only on the length of the input word and not on the size of the lexicon.

	FST_1	FST_2
Number of states	*69.498*	*90.613*
Number of transitions	*90.801*	*130.839*
Size of bin file	*252 kB*	*662 kB*

Table 1: The final finite-state transducers representing Slovenian phonetic (FST_1) (60.000 items) and Slovenian morphology lexicon (FST_2) (40.000 items).

4. Lexicons FST byte representation

Finite-state transducers representing lexicons are actually finite-state automata that have transitions labeled with two symbols. One of the symbols represents input, the other output. Therefore they translate strings. Since FST's of large-scale lexicons can be quite huge (lots of states and transitions) their implementation is not trivial. It is very important to use 'every bit' in their binary representation. The information in the final FST binary file is organized into sequences of 6 bytes. Every such byte sequence describes information of the one state transition (Fig. 3).

The information representing one transition is coded using 6 bytes. The choice depends on the maximum value of a particular data type and final FST size of the lexicons. The FST input and output alphabets for Slovenian lexicons (SIflex and SImlex) (ortographic characters, SAMPA phonetic symbols, some punctuation symbols) can be coded using eight bits (one octet) (first byte for input alphabet symbol and the second byte for output alphabet symbol). The third byte serves for flags (final bit, stop bit, next bit) and other three bytes are used for calculation of the next state

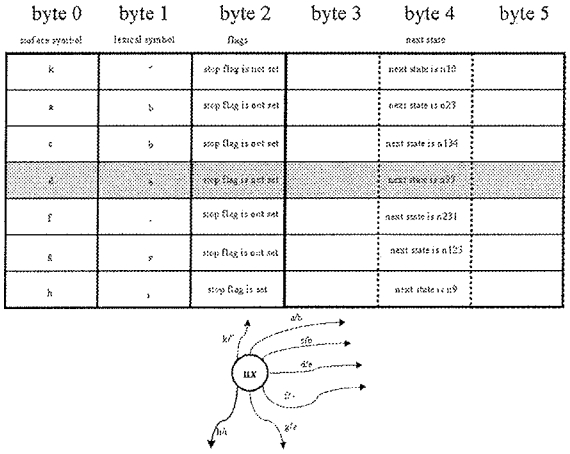

Figure 3: Lexicons FST byte representation.

(jump to the next 6-byte sequence). Using such approach, states are only indirectly marked and are actually defined with transition sequences. All the transitions not having set the stop bit belong to the same state. Next state transition start from the byte sequence, that has a stop bit set.

5. Hardware implementation of the FST Lexicons

The described FST representation of the lexicons is very appropriate also for implementation using the flash memory chip (Atmel AT49BV161T). AT49BV161T is a 16-megabit (1Mx16/2Mx8) 3 Volt Flash Memory and is organised as 1.048.576 words of 16 bits each, or 2.097.152 bytes of 8 bits each. The x16 data appears on I/O0-I/15, the x8 data appears on I/O0-I/O7. This device can be read or reprogrammed using a single 2.65V power supply, making it ideally for in-system programming.

In the AT49BV/LV161(T) configuration, the BYTE pin controls wheather the device data I/O pins operate in the byte or word configuration. In our approach the BYTE pin is set

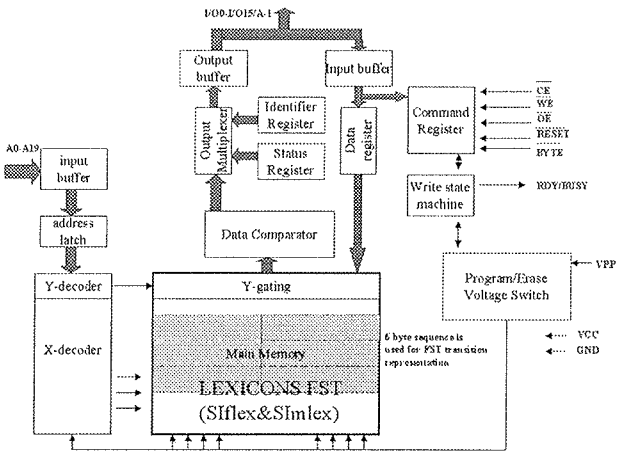

Figure 4: Lexicons FST byte representation in flash memory chip AT49BV161T.

at logic "0", and the device is in byte configuration. The data I/O pins I/O0-I/O7 are active and controlled by CE and OE. The data I/O pins I/O8 - I/O14 are tri-stated, and the I/O15 pin is used as an input for the LSB (A-1) adress function. All together with other address pins A0 - A19, we are able to address (2^{21} bytes) 2097152 bytes, what is enough for Slovenian FST lexicons (Fig. 4).

6. Using FST representing both lexicons

As transducers translate a string into another string (string-to-string transducers), the lookup algorithm is straightforward - it consists of following labels from one level in a lexicon transducer.

procedure lookup (state, word, index, output)
if (state ∈ F)
print(output)
for each a ∈ $\Sigma \cup \{\varepsilon\}$ such that $\exists_{t \in Q}\delta$ (state, ,a) = t
lookup(t, word, index+1, output . a)
for each a ∈ $\Sigma \cup \{\varepsilon\}$ such that $\exists_{t \in Q}\delta$
(state,word[index],a) = t
lookup(t,word,index+1, output . a)

The dot operator represents concatenation. The result of concatenating string with an empty string is the same string: word . ε = word. The use of the empty string in transition labels is necessary, as the lengths of the strings may not match. It is also useful for the alignment of segments of strings that represent the same features. Such alignment may reduce the size of the transducer. The presented algorithm was implemented on the Atmel Microcontroller AT AT90S8515. Comparison of input symbols of the FST transitions with word characters and calculation of the next state (next byte position) is performed using bit operations (Fig. 5).

7. Hardware implementation of the lookup algorithm

Atmel 8-bit microcontroller with 8kB downloadable flash memory was used for the implementation of the lookup algorithm. This microcontroller provides a highly flexible and cost effective solution to many embedded control applications. Raw Instruction Set architecture of used microcontroller features execution of powerful instructions in a single cycle that achieves enough throughput for high performance functionality /8/. The standard asynchronous serial interface UART of the microcontroller is mainly used for the testing purposes and it's not intended to be used for any data transmission in real-mode functionality as the data transfer rates are too low. The presented hardware architecture features in-system programming of both program Flash memory of the microcontroller and the external Flash memory used for FST Siflex and Simlex data storage. For real-mode functionality another peripheral exten-

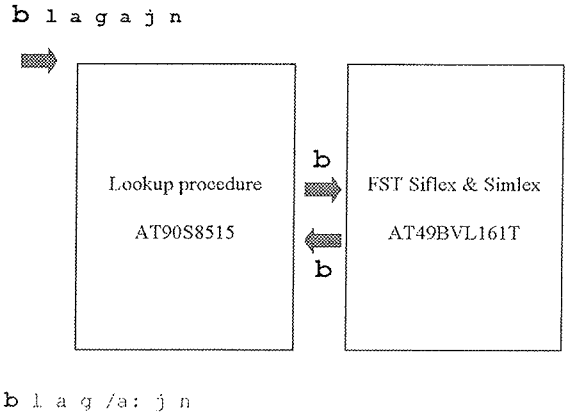

Figure 5: Lookup algorithm implementation using microcontroller AT90C8515.

sion of the microcontroller called serial peripheral interface SPI is used. SPI features high speed data transfers up to 5 Mbit/s that gives the hardware structure enough data bandwidth for efficient activity. In advance the SPI interface is already present and used in most common PDAs today as those are build around Intel's StrongARM SA-1110 microprocessor /7/. This appliance simplifies interconnections between FSA hardware and present or future embedded microcomputers.

Port A is 8-bit bidirectional I/O port. It is connected directly to flash memory over the data bus and is used as an input/output. Output represents word character (input alphabet), and the input comes from FST in the flash memory as output symbol of the FST transition.

Data addressing of the external Flash memory is done by the microcontroller lookup procedure. Due to the lack of microcontroller programmable pins and used address multiplexing it's suitable to partially compute the complete address pointer and update the separate address latches in-time between computations of the next lookup addresses. As an address output serves port C. Since we need 21 address lines to be able to address any byte in the flash memory (2.057.152), the port C pins are connected to three latches 74HC573. Thus the microcontroller methodically computes the exact address pointer in three steps as each one of them updates appropriate address latch after its computation. In this manner there is no execution speed breakdown because of used addressing architecture.

Port D is used for driving flash memory chip and Port B for driving all three latches 74HC573. Port D is also 8 bit bidirectional I/O port and is connected with MAX3232 level shifter. The firmware of the microcontroller features two separate fully operational program modes. First one is intended for external flash programming and functionality testing (port B). In this mode the firmware is featuring low speed data transfers between personal computer and the FST. The firmware build-in Flash programming procedure re-

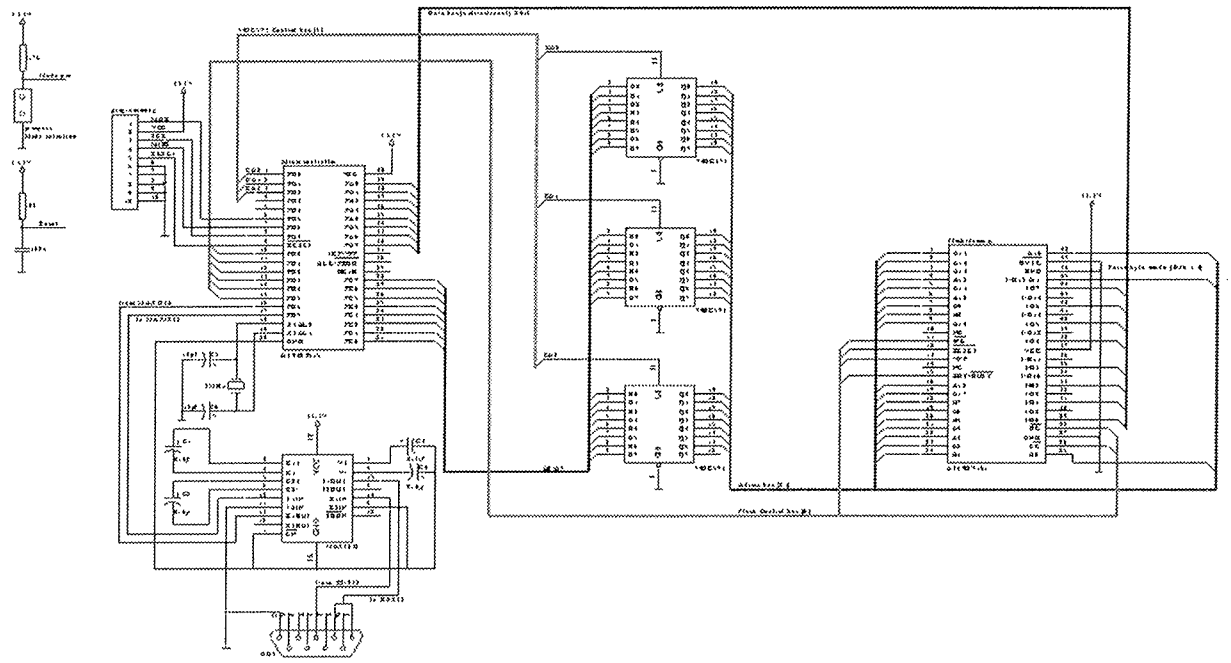

Figure 6: Hardware implementation of the system.

ceives or transmits data through asynchronous serial interface and reads or writes them in external Flash memory. In that fashion the complete FST algorithm can be tested through the personal computer. In real-mode functionality of FST the second part of the firmware is executed. In this part of the firmware the high speed serial peripheral interface is used for data transmissions. Choosing between those two modes is done simply with defining the logical state on the MODE pin of the microcontroller. As startup or as a reset condition has been applied to the microcontroller, the firmware checks the logical state on this pin and executes the appropriate functionality mode. In appliance with existing PDAs there is no need to implement the testing functionality of FST. The whole hardware implementation of the system is shown in Fig. 6.

8. Conclusion

Being able to operate most of the common PDA (Personal Digital Assistance) functions such as e-mail, voice, calendar, phone numbers and addresses, by using voice, the speech recognition and text-to-speech synthesis technology must be integrated into any embedded mobility suite. Development of real models of human language requires a lot of linguistic data. Finite-state machines were used for Slovenian large-scale lexicons, since they are time and space optimal solution. The effect of using finite-state transducers is great reduction of the memory usage required and the optimal access time, independent from the size of the lexicons. As showed the FST structure enable easy, flexible and efficient hardware implementation that can be used in embedded systems as significant part of the speech embedded mobility devices.

9. References

/1/ Mehryar Mohri., (1995) On Some Applications of Finite-State Automata Theory to Natural Language Processing, Natural Language Engineering 1, Cambridge University Press.

/2/ Bruce William Watson, (1995) Taxonomies and Toolkits of Regular Language Algorithms, PhD Thesis, Eindhoven University of Technology and Computing Science.

/3/ Aho, Alfred V., John E. Hopcroft, and Jeffrey D. Ullman, (1974) The design and analysis of computer algorithms. Addison Wesley: Reading, MA.

/4/ Matej Rojc, Zdravko Kačič, (2001) Representation of Large Lexica Using Finite-State Transducers for the Multilingual Text-to-Speech Synthesis System, Eurospeech 2001, Scandinavia.

/5/ SAMPA for Slovenian, (1998) http://www.phon.ucl.ac.uk/home/sampa/slovenian.html

/6/ MULTEXT project lexical specifications, (1996)
http://www.lpl.univaix.fr/projects/multext/LEX/LEX.Specifications.html

/7/ Intel. Intel StrongARM SA-1110 Microprocessor: Developer's Manual. Intel, June 2000.

/8/ Atmel. 8-Bit AVR Microcontroller with 8K bytes Downloadable Flash. Atmel, 1997.

mag. Matej Rojc
izr. prof. dr. Zdravko Kačič
mag. Iztok Kramberger
Inštitut za elektroniko,
Fakulteta za elektrotehniko, računalništvo in informatiko, Maribor, Slovenija

Institute of Electronics,
Faculty of Electrical Engineering and Computer Science,
Smetanova 17, 2000 Maribor, Slovenia
Tel. +386 02 220 7000, Fax. +386 02 251 1178
e-mail: dsplab@uni-mb.si

Prispelo (Arrived): 10.05.2002 *Sprejeto (Accepted): 28.06.2002*