Černý's conjecture and the road coloring problem

Jarkko Kari[1], *Mikhail Volkov*[2]

[1]Department of Mathematics
FI-20014 University of Turku
Turku, Finland

[2]Department of Mathematics and Mechanics
620083 Ural State University
Ekaterinburg, Russia
email: jkari@utu.fi, Mikhail.Volkov@usu.ru

March 1, 2010   23 h 25

chapterKV

Contents

1 Synchronizing automata, their origins and importance

A complete deterministic finite automaton (DFA) $\mathcal{A}$ with input alphabet A and state set Q is called *synchronizing* if there exists a word $w \in A^*$ whose action resets $\mathcal{A}$, that is, w leaves the automaton in one particular state no matter at which state in Q it is applied: $q \cdot w = q' \cdot w$ for all $q, q' \in Q$. Any word w with this property is said to be a *reset* word for the automaton.
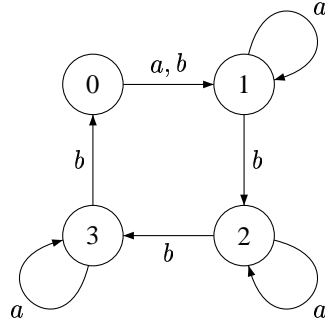
Figure 1. A synchronizing automaton

Figure 1 shows an example of a synchronizing automaton with 4 states. The reader can easily verify that the word ab^3ab^3a resets the automaton leaving it in the state 1. With somewhat more effort one can also check that ab^3ab^3a is the shortest reset word for this automaton. The example in Figure 1 is due to Černý, a Slovak computer scientist, in whose pioneering paper [16] the notion of a synchronizing automaton explicitly appeared for the first time. (Černý called such automata *directable*. The word *synchronizing* in this context was probably introduced by Hennie [11].) Implicitly, however, this concept has been around since the earliest days of automata theory. The very first synchronizing automaton that we were able to trace back in the literature appeared in Ashby's classic book [1, pp. 60–61].

In [16] the notion of a synchronizing automaton arose within the classic framework of Moore's "Gedanken-experiments" [12]. For Moore and his followers finite automata served as a mathematical model of devices working in discrete mode, such as computers or relay control systems. This leads to the following natural problem: how can we restore control over such a device if we do not know its current state but can observe outputs produced by the device under various actions? Moore [12] has shown that under certain conditions one can uniquely determine the state at which the automaton arrives after a suitable sequence of actions (called an *experiment*). Moore's experiments were adaptive, that is, each next action was selected on the basis of the outputs caused by the previous actions. Ginsburg [9] considered more restricted experiments that he called *uniform*. A uniform experiment[1] is just a fixed sequence of actions, that is, a word over the input alphabet; thus, in Ginsburg's experiments outputs were only used for calculating the resulting state at the end of an experiment. From this, just one further step was needed to come to the setting in which outputs were not used at all. It should be noted that this setting is by no means artificial—there exist many practical situations when it is technically impossible to observe output signals. (Think of a satellite which loops around the Moon and cannot be controlled from the Earth while "behind" the Moon.)

The original "Gedanken-experiments" motivation for studying synchronizing automata is still of importance, and reset words are frequently applied in model-based testing of reactive systems. See [6, 3] as typical samples of technical contributions to the area and

---

[1]After [8], the name *homing sequence* has become standard for the notion.

Needs double-checking!!

51 [15] for a recent survey.

52   Another stong motivation comes from the coding theory. We refer to [2, Chapters 3
53 and 10] for a detailed account of profound connections between codes and automata; here
54 we restrict ourselves to a brief introduction into a special (but still very important) case of
55 maximal prefix codes. Recall that a *prefix code* over a finite alphabet A is a set X of words
56 in A^* such that no word of X is a prefix of another word of X. A prefix code is *maximal*
57 if it is not contained in another prefix code over the same alphabet. A maximal prefix
58 code X over A is *synchronized* if there is a word $x \in X^*$ such that for any word $w \in A^*$,
59 one has $wx \in X^*$. Such a word x is called a *synchronizing word* for X. The advantage of
60 synchronized codes is that they are able to recover after a loss of synchronization between
61 the decoder and the coder caused by channel errors: in the case of such a loss, it suffices
62 to transmit a synchronizing word and the following symbols will be decoded correctly.
63 Moreover, since the probability that a word $v \in A^*$ contains a fixed word x as a factor
64 tends to 1 when the length of v increases, synchronized codes eventually resynchronize by
65 themselves, after sufficiently many symbols being sent. (As shown in [4], the latter prop-
66 erty in fact characterizes synchronized codes.) The following simple example illustrates
67 these ideas: let $A = \{0, 1\}$ and $X = \{000, 0010, 0011, 010, 0110, 0111, 10, 110, 111\}$.
68 Then X is a maximal prefix code and one can easily check that each of the words 010,
69 011110, 011111110, ... is a synchronizing word for X. For instance, if the code word
70 000 has been sent but, due to a channel error, the word 100 has been received, the de-
71 coder interprets 10 as a code word, and thus, loses synchronization. However, with a
72 high probability this synchronization loss only propagates for a short while; in particu-
73 lar, the decoder definitely resynchronizes as soon as it encounters one of the segments
74 010, 011110, 011111110, ... in the received stream of symbols. A few samples of such
75 streams are shown in Figure 2 in which vertical lines show the partition of each stream
76 into code words and the boldfaced code words indicate the position at which the decoder
resynchronizes.

| Sent | 0 0 0 \| 0 0 1 0 \| **0 1 1 1** \| ... |
Received	1 0 | 0 0 0 | 1 0 | **0 1 1 1** | ...
Sent	0 0 0 | 0 1 1 1 | 1 1 0 | 0 0 1 1 | 0 0 0 | 1 0 | **1 1 0** | ...
Received	1 0 | 0 0 1 1 | 1 1 1 | 0 0 0 | 1 1 0 | 0 0 1 0 | **1 1 0** | ...
Sent	0 0 0 | 0 0 0 | 1 1 1 | **1 0** | ...
Received	1 0 | 0 0 0 | 0 1 1 1 | **1 0** | ...

Figure 2. Restoring synchronization

77
78   If X is a finite prefix code over a finite alphabet A, then its decoding can be imple-
79 mented by a deterministic automaton that is defined as follows. Let Q be the set of all
80 proper prefixes of the words in X (including the empty word ε). For $q \in Q$ and $a \in A$,
81 define

$$q \cdot a = \begin{cases} qa & \text{if } qa \text{ is a proper prefix of a word of } X, \\ \varepsilon & \text{if } qa \in X. \end{cases}$$

82 The resulting automaton $\mathcal{A}_X$ is complete whenever the code X is maximal and it is
83 easy to see that $\mathcal{A}_X$ is a synchronizing automaton if and only if X is a synchronized

code. Moreover, a word x is synchronizing for X if and only if x is a reset word for $\mathcal{A}_X$ and sends all states in Q to the state ε. Figure 3 illustrates this construction for the code $X = \{000, 0010, 0011, 010, 0110, 0111, 10, 110, 111\}$ considered above. The solid/dashed lines correspond to (the action of) 0/1.
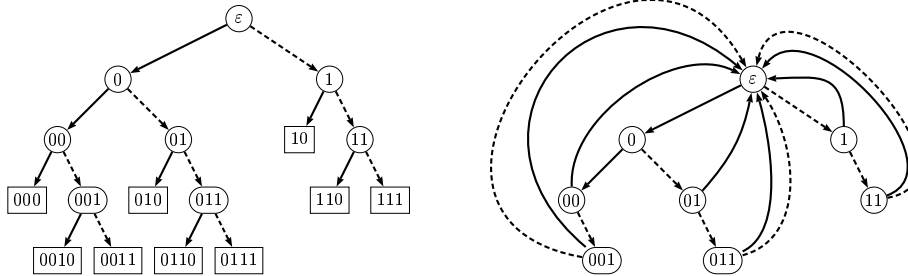
Figure 3. A synchronized code (on the left) and its automaton (on the right)

An additional source of problems related to synchronizing automata has come from *robotics* or, more precisely, from part handling problems in industrial automation such as part feeding, fixturing, loading, assembly and packing. Within this framework, the concept of a synchronizing automaton was again rediscovered in the mid-1980s by Natarajan [13, 14] who showed how synchronizing automata can be used to design sensor-free orienters for polygonal parts, see [17, Section 1] for a transparent example illustrating Natarajan's approach in a nutshell. Since the 1990s synchronizing automata usage in the area of robotic manipulation has grown into a prolific research direction but it is fair to say that publications in this area deal mostly with implementation technicalities. However, amongst them there are papers of significant theoretical importance such as [7, 10, 5].

2 Algorithmic and complexity issues

3 The Černý conjecture

4 The Road Coloring Problem

5 Generalizations

References

[1] W. R. Ashby. *An introduction to cybernetics.* Chapman & Hall, 1956. 2

[2] J. Berstel, D. Perrin, and C. Reutenauer. *Codes and automata.* Number 129 in Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2009. 3

[3] V. Boppana, S. Rajan, K. Takayama, and M. Fujita. Model checking based on sequential atpg. In *Computer Aided Verification, Proc. 11th International Conference*, Lecture Notes in Comput. Sci., pages 418–430. Springer-Verlag, 1999. 2

[4] R. M. Capocelli, L. Gargano, and U. Vaccaro. On the characterization of statistically synchronizable variable-length codes. *IEEE Transactions on Information Theory*, 34(4):817–825, 1988. 3

[5] Y.-B. Chen and D. J. Ierardi. The complexity of oblivious plans for orienting and distinguishing polygonal parts. *Algoritmica*, 14:367–397, 1995. 4

[6] H. Cho, S.-W. Jeong, F. Somenzi, and C. Pixley. Synchronizing sequences and symbolic traversal techniques in test generation. *J. Electronic Testing*, 4:19–31, 1993. 2

[7] D. Eppstein. Reset sequences for monotonic automata. *SIAM J. Comput.*, 19:500–510, 1990. 4

[8] A. Gill. State-identification experiments in finite automata. *Inform. Control*, 4(2-3):132–154, 1961. 2

[9] S. Ginsburg. On the length of the smallest uniform experiment which distinguishes the terminal states of a machine. *J. Assoc. Comput. Mach.*, 5:266–280, 1958. 2

[10] K. Goldberg. Orienting polygonal parts without sensors. *Algorithmica*, 10:201–225, 1993. 4

[11] F. C. Hennie. Fault detecting experiments for sequential circuits. In *Switching Circuit Theory and Logical Design, Proceedings of the Fifth Annual Symposium*, pages 95–110. IEEE Press, 1964. 2

[12] E. F. Moore. Gedanken experiments on sequential machines. In C. E. Shannon and J. McCarthy, editors, *Automata Studies*, pages 129–153. Princeton Universty Press, 1956. 2

[13] B. K. Natarajan. An algorithmic approach to the automated design of parts orienters. In *Proc. 27th Annual Symp. Foundations Comput. Sci.*, pages 132–142. IEEE Press, 1986. 4

[14] B. K. Natarajan. Some paradigms for the automated design of parts feeders. *Internat. J. Robotics Research*, 8(6):89–109, 1989. 4

[15] S. Sandberg. Homing and synchronizing sequences. In M. Broy, B. Jonsson, J.-P. Katoen, M. Leucker, and A. Pretschner, editors, *Model-Based Testing of Reactive Systems*, volume 3472 of *Lecture Notes in Comput. Sci.*, pages 5–33. Springer-Verlag, 2005. 3

[16] J. Černý. Poznámka k homogénnym eksperimentom s konečnými automatami. *Matematicko-fyzikalny Časopis Slovenskej Akadémie Vied*, 14(3):208–216, 1964. in Slovak. 2

[17] M. Volkov. Synchronizing automata and the Černý conjecture. In C. Martín-Vide, F. Otto, and H. Fernau, editors, *Language and Automata Theory and Applications*, volume 5196 of *Lecture Notes in Comput. Sci.*, pages 11–27. Springer-Verlag, 2008. 4

Index