Chomskian Hierarchies of Families of Sets of Piecewise Continuous Functions

Keijo Ruohonen

Department of Mathematics
Tampere University of Technology
33101 Tampere, Finland
keijo.ruohonen@tut.fi

Abstract

The venerable Chomsky hierarchy has long shown its value as a structural tool in formal languages and automata theory, and gained followers in various areas. We show here how very similar hierarchies can be obtained for families of sets of piecewise continuous functions. We use systems of ordinary differential equations as automata are used in establishing the traditional Chomsky hierarchy. A functional memory is provided by state-dependent delays which are used in a novel way, paired with certain state components, giving memory structures similar to push-down stores and Turing machine tapes. The resulting machine model may be viewed as a ”functional computing machine”, with functional input, functional memory, and, though this is not emphasized here, functional output.

1 Introduction

Ever since its introduction by Noam Chomsky in the 1950s (see [6, 7]) the hierarchy of families of languages named after him has played a prominent role in the theory of formal languages and computation. This can be seen immediately e.g. in [26, 17], two popular text-books in the area. The hierarchy can be formulated as a hierarchy of families of languages, or as a hierarchy of generating devices (grammars), or as a hierarchy of recognizing devices (automata), as summarized in the following table:

language family	grammar	automaton
regular languages	Type 3	finite automaton
context-free languages	Type 2	push-down automaton
context-sensitive languages	Type 1	linear-bounded automaton
computably enumerable languages	Type 0	Turing machine

Whatever angle it is viewed from, the Chomsky hierarchy appears as a rather natural structural backbone. It is therefore no wonder that similar natural structures have been sought after in various areas. While these extensions have been more complicated and not quite as natural as the original hierarchy, they have served a similar purpose. An example is the hierarchy of Lindenmayer systems (see e.g. [26]), and a more recent example can be found in [16].

Dynamical systems, working in continuous time and with a finite number of continuous states, have been intensively investigated lately from the point of view of computational power. Mathematically these systems are nonlinear systems of ordinary differential equations, with inputs given, say, as integral initial values. The ability of such systems to simulate universal Turing machines has been known for some time, see [23, 2, 5, 25]. Reviews of the earlier developments in this area can be found in [19] and [5], and [3] contains a more recent review in a somewhat different vein. If certain assumptions are made, essentially preventing embedding Turing-complete or more powerful oracles in the structure of the system, then the computational power is seen to be exactly the same as that of Turing machines (see e.g. [24, 25]). It would thus be possible to define the traditional Chomsky hierarchy using continuous dynamical systems. This, however, does not appear to produce anything new.

Ordinary differential equations form a traditional device for defining sets of functions. However, this is more in a generative sense than as recognizers. On the other hand, the recognizing aspect is present in systems and control theory, indeed, the similarities between control systems and sequential machines have been known long (see e.g. [15]). We use differential equations (provided with a special memory structure, see below) to recognize sets of functions. We restrict ourselves to piecewise continuous functions $\mathbb{R} \longrightarrow \mathbb{R}$ with bounded support. To simplify matters we allow only supports which are subsets of $[0, 1]$. As the reader may note when reading on, this is no real restriction, the theory is easily extended to arbitrary bounded supports, contained in given function-dependent finite intervals. To be quite specific, we define a *piecewise continuous function* in $\mathbb{R}$ as a function f such that, for all real numbers a, the limits $f(a-) = \lim_{x\uparrow a} f(x)$ and $f(a+) = \lim_{x\downarrow a} f(x)$ both always exist as finite numbers, and are equal $f(a)$ except possibly for a finite number of values of a. If in addition the equation

$$f(x) = \frac{1}{2}(f(x-) + f(x+))$$

is always satisfied we say that f is a *total piecewise continuous function.* These definitions are extended to arbitrary intervals in an obvious fashion. The set of all total piecewise continuous functions with support included in [0, 1] is denoted by F_{PC}.

Systems of ordinary differential equations have only state memory. A way to add memory is to allow delays. Delay-differential equations have a long history in applied mathematical modelling, especially in mathematical biology, see e.g. [9, 10]. We use a state-dependent delay but in a novel way: Certain dependent variables are used pairwise to define a piecewise continuous function which is used as a memory element, much as a Turing machine tape. There can be several such memory elements, or none. Posing certain natural restrictions, a memory element can be made push-down-like. The input is treated similarly, paired with a spesific dependent variable.

Since, from the point of view of automata, the Chomsky hierarchy is not so much about time or space complexity, but rather about the kind of memory available, it is possible to define Chomskian hierarchies of subsets of F_{PC}. We define and prove several such hierarchies. We also obtain several closure results for various levels of the hierarchies. A few open problems remain, especially concerning the (common) upper end of the hierarchies (the one corresponding to Turing machines and computably enumerable languages). While we investigate here only recognizers, corresponding systems with functional outputs could be easily defined, so in a sense we deal with ”functional computing machines”.

In the sequel we call our dynamical systems simply ”machines”. These machines are subject to certain restrictions pertaining to the type of ”computations” allowed. First, the machines are assumed to be ”deterministic”, i.e., they have only forward-unique solutions. Second, the machines are ”Zenoan”, meaning e.g. that no part of the memory or input is used infinitely often in any finite time interval. Third, as far as possible, solutions should depend continuously on parameters in the input. These properties are discussed in Section 2 where the detailed definition of the machines is given. It should be mentioned that the restrictions correspond roughly to what might be considered well-posedness for the kind of machines we investigate. A different theory would be obtained if any of the restrictions is lifted, e.g., it would be possible to obtain a similar and yet quite different theory for non-forward-unique solutions (allowing a kind of nondeterminism).

Certain classifications of real functions according to recursion or computation based criteria are known, notably those in [18] and [22]. It should also be mentioned that sets of n-tuples of reals can be defined by the well-known BSS-machines (see [4]). There are many excellent text-books in formal languages and automata theory. E.g. [26, 14, 11] are old classics and [17, 13] are popular modern books. Concerning ordinary differential equations we want to mention the comprehensive classical texts [8, 12], the nice concise presentation in [21], and [1], a veritable treasure trove of uniqueness results.

2 Basic Definitions

To define a *machine* M, we start by fixing its basic dimensions:

$$\begin{aligned} m_M &= \text{ dimension of state} \\ n_M &= \text{ dimension of functional memory} \end{aligned}$$

Both of these dimensions are assumed to be finite, $m_M > 0$ and $n_M \geq 0$. The following steps then lead us to the definition of M.

1. Input

The *input* is a total piecewise continuous function $f : (0, 1) \longrightarrow \mathbb{R}$ such that the limits $f(0+)$ and $f(1-)$ exist as finite numbers. Recall the definition of total piecewise continuity in Section 1. To define the way input is read by the machine M we first define

$$f^*(s) = \begin{cases} f(s), \text{ if } 0 < s < 1 \\ \frac{1}{2} f(0+), \text{ if } s = 0 \\ \frac{1}{2} f(1-), \text{ if } s = 1 \\ 0 \text{ elsewhere.} \end{cases}$$

Note that this simply means extending f to a total piecewise continuous function defined in $\mathbb{R}$ with support included in the interval [0, 1].

The input is then given as

$$\hat{f}(t) = f^*(s_0(t))$$

where $s_0(t)$ is the *position function* controlled by M (via a differential equation). Initially $s_0(0) = 0$, i.e., reading of input starts at $s = 0$. The value of $\hat{f}(t)$ is immediately available to M at time t. (In traditional automata-theoretic terms this could be called a ”read-only input tape”, $s_0(t)$ being the position of the ”read-head”.)

In the sequel we more or less identify f and f^*, and use F_{PC} to denote the set of all possible inputs.

2. State

The *state* of M at time t is a point $\mathbf{q}(t) \in \mathbb{R}^{m_M}$. The initial value is $\mathbf{q}(0) = \mathbf{0}$. The first state component q_1 is designated as the *acceptance indicator.* Dynamical evolution of the state is defined via a differential equation.

3. Functional memory

The machine may have a *functional memory.* A machine M without a functional memory, i.e., with $n_M = 0$, is called a *two-way state machine,* see Sections 3 and 4.

At any time t the contents of this memory is given via a function $\mathbf{x} : \mathbb{R} \longrightarrow \mathbb{R}^{n_M}$, as described below. This function is defined by a differential equation, initially $\mathbf{x}(0) = \mathbf{0}$. The corresponding *position function* (positions of the ”read-write-heads”) is denoted by $\mathbf{s}(t)$. Now, what we mean by the contents of the functional memory at time $t = T$, is the collection of the n_M functions $x^*_{i,T} : \mathbb{R} \longrightarrow \mathbb{R}$ $(i = 1, \dots, n_M)$, given by

$$x^*_{i,T}(s) = \begin{cases} 0, \text{ if } s_i(t) \neq s \text{ for } 0 \leq t \leq T \\ x_i(t^*_{i,T}(s)) \text{ otherwise} \end{cases}$$

where $t^*_{i,T}$ is the maximum inverse of s_i, i.e.,

$$t^*_{i,T}(s) = \max_{\substack{s_i(t)=s \\ 0 \leq t \leq T}} t.$$

Here s_i will be continuous in $[0, T]$, so $t^*_{i,T}(s)$ is defined if $s_i(t) = s$ for some t, $0 \leq t \leq T$. Thus, what needs to be stored of the function $\mathbf{x}$ is only what is needed to define the functions $x^*_{i,T}$, not the entire history $\mathbf{x}(t)$, $0 \leq t \leq T$. In Figure 1 an example of the curve $(s_i(t), x_i(t))$, $0 \leq t \leq T$, is given, the graph of $x^*_{i,T}(s_i)$ is in thick line.

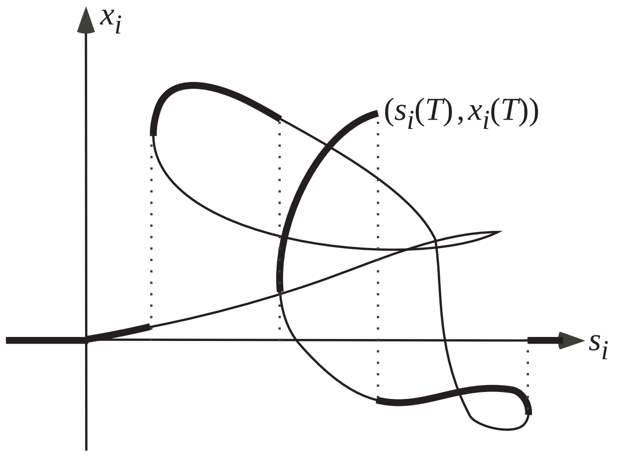

Figure 1.

A basic property of the functions $x^*_{i,T}$ is given by

Lemma 1. *If, in the interval* $0 \leq t \leq T$, x_i *and* s_i *are continuous, and* s_i' *and* $\operatorname{sgn}(s_i')$ *are piecewise continuous, then* $x^*_{i,T}$ *is piecewise continuous. (Piecewise continuity of* $\operatorname{sgn}(s_i')$ *means that* s_i' *changes sign only finitely often.)*

Proof. It suffices to show that $t^*_{i,T}$ is piecewise continuous. We take a partition of the interval $[0, T]$

$$0 = \tau_0 < \tau_1 < \cdots < \tau_{N-1} < \tau_N = T$$

such that, in each interval (τ_{j-1}, τ_j), either $s_i'(t)$ is identically zero or $s_i'(t)$ is $\neq 0$ and continuous. Now, a jump discontinuity of $t^*_{i,T}(s)$ can only take place at one of the points $s_i(\tau_j)$ $(j = 0, 1, \ldots, N)$. Between two consecutive points $s_i(\tau_{j_1})$ and $s_i(\tau_{j_2})$ then $t^*_{i,T}(s) = s_i^{-1}(s)$ is continuous and has finite limits at $s_i(\tau_{j_1})+$ and $s_i(\tau_{j_2})-$. $\square$

At time $t = T$, in addition to $\mathbf{q}(T)$ (the state) and $\hat{f}(T)$ ("input symbol under scan"), the machine M has available $\hat{\mathbf{x}}(T)$ ("symbols to be read on the tapes") where

$$\hat{x}_i(T) = \begin{cases} x^*_{i,T}(s_i(T)-), \text{ if } s_i'(T) < 0 \\ x_i(T), \text{ if } s_i'(T) = 0 \\ x^*_{i,T}(s_i(T)+), \text{ if } s_i'(T) > 0 \end{cases} \qquad (i = 1, \ldots, n_M).$$

Using these the dynamics of the machine is defined by differential equations. Note that $x^*_{i,T}(s_i(T)+) = x_i(T)$ for $s_i'(t) < 0$, since, for a $t < T$ sufficiently close to T, we have then $t^*_{i,T}(s_i(t)) = t$. Similarly $x^*_{i,T}(s_i(T)-) = x_i(T)$ for $s_i'(T) > 0$.

For *push-down machines* (to be treated in Sections 5 and 6) $n_M = 1$ and an alternative "one-sided" definition of $\hat{x}_1$ is needed. We define then

$$\hat{x}_1(T) = \begin{cases} x^*_{1,T}(s_1(T)-), \text{ if } s_1'(T) < 0 \\ x_1(T), \text{ if } s_1'(T) \geq 0. \end{cases}$$

(A similar situation exists of course in traditional automata theory.) For these machines it is in addition assumed that always $s_1(t) \geq 0$.

4. Differential equations

State transition, position changes (moving the ”read-head” and the ”read-write-heads”) and writing on the functional memory is controlled by a system of differential equations. We write the system in the form

$$\frac{d\mathbf{q}}{dt} = \mathbf{Q}(\mathbf{q}(t),\ \hat{f}(t), \hat{\mathbf{x}}(t)) \quad , \quad \frac{ds_0}{dt} = S_0(\mathbf{q}(t),\ \hat{f}(t), \hat{\mathbf{x}}(t)) \ ,$$
$$\frac{d\mathbf{s}}{dt} = \mathbf{S}(\mathbf{q}(t),\ \hat{f}(t), \hat{\mathbf{x}}(t)) \quad \text{and} \quad \frac{d\mathbf{x}}{dt} = \mathbf{X}(\mathbf{q}(t),\ \hat{f}(t), \hat{\mathbf{x}}(t))$$

where the functions

$$\mathbf{Q} : \mathbb{R}^{m_M+n_M+1} \longrightarrow \mathbb{R}^{m_M} \quad , \quad S_0 : \mathbb{R}^{m_M+n_M+1} \longrightarrow \mathbb{R} \ ,$$
$$\mathbf{S} : \mathbb{R}^{m_M+n_M+1} \longrightarrow \mathbb{R}^{n_M} \quad \text{and} \quad \mathbf{X} : \mathbb{R}^{m_M+n_M+1} \longrightarrow \mathbb{R}^{n_M}$$

are given. It will be assumed that these functions are continuous in $\mathbb{R}^{m_M+n_M+1}$. With zero initial values $\mathbf{q}(0) = \mathbf{0}$, $s_0(0) = 0$, $\mathbf{s}(0) = \mathbf{0}$ and $\mathbf{x}(0) = \mathbf{0}$ an initial value problem is then defined. The system of differential equations above is autonomous in that there is no explicit dependence on time t. As usual, time t may be included as a component of the state $\mathbf{q}$, if needed, as can be $\mathbf{x}(t)$, $s_0(t)$ and $\mathbf{s}(t)$.

We do not want our differential equations to be too badly behaved. Therefore we make the following assumptions which should hold for any input in F_{PC}.

1. We assume that $\mathbf{Q}$, S_0, $\mathbf{S}$ and $\mathbf{X}$ satisfy conditions guaranteeing existence and uniqueness of solutions in the forward direction for $t \geq 0$. We will not specify these conditions, however. Indeed, for the kind of controlled state-dependent-delay-differential equations that these equations are, few conditions of any generality seem to be known at the time of writing, at least as far as global behaviour is considered.

   On the other hand, locally $\hat{f}$ may be considered as being part of the external structure of the system, and ”ordinary” conditions apply (e.g. Carathèodory-type conditions). Similarly, $\hat{\mathbf{x}}$ may also be considered as part of the external structure, or sometimes as part of the state structure. We refer to [1, 12, 21].

2. Whenever possible, we will assume continuous dependence on parameters appearing in f. More specifically, we do not want the structure of the differential equations to be one allowing a discontinuous dependence, that is, wherever $f^*(s)$ and $x^*_{i,T}(s)$ are ’nice’, say Lipschitz-continuous, and $f^*(s)$ depends continuously on parameters in f, the solution of the differential equation depends continuously on those parameters as well. Note that, in a compact subset of the parameter space, this continuous dependence is uniform in t in any finite closed time interval.

3. The derivatives $s_0', s_1', \ldots, s_{n_M}'$ and their signs $\mathrm{sgn}(s_0'), \mathrm{sgn}(s_1'), \ldots, \mathrm{sgn}(s_{n_M}')$ are piecewise continuous in any finite time interval $0 \leq t \leq T$. By Lemma 1, this implies that the functions $x^*_{i,T}$ are piecewise continuous.

The reason for demanding existence of solution is obvious. On the other hand, in this paper we do not want to consider nonuniqueness in the forward direction. Nor do we allow any of the functions $\mathbf{Q}$, S_0, $\mathbf{S}$ and $\mathbf{X}$ to be undefined. (Again in traditional terms, we restrict ourselves to ”deterministic” machines.)

Condition 2 may be interpreted loosely as forcing computations to take only finitely many ”steps” in finite time intervals, i.e., we consider only ”Zenoan” computations. (A ”step” corresponds here to a maximal open time interval where each s_i' is continuous and either nonzero or identically zero.) Note that we do not state a similar condition for components of $\mathbf{q}(t)$ or $\mathbf{x}(t)$, however. (C.f. [25].)

Note. *We do not assume backward uniqueness (”reversibility” in traditional terms). On the other hand, we might want to restrict the way the functions* $\mathbf{Q}$*,* S_0*,* $\mathbf{S}$ *and* $\mathbf{X}$ *are given, say, explicitly (as in* [23, 25]*) or as computable functions in the sense of* [20] *(and* [24]*). As long as the functions are kept reasonably general this does not affect our results.*

We say that a machine is a *one-way machine* if $S_0(\mathbf{q}, \hat{f}, \hat{\mathbf{x}}) \geq 0$, and a *strictly one-way machine* if $S_0(\mathbf{q}, \hat{f}, \hat{\mathbf{x}}) > 0$. A *real-time machine* is a strictly one-way machine for which $S_0(\mathbf{q}, \hat{f}, \hat{\mathbf{x}}) = 1$, i.e., $s_0(t) = t$.

5. Acceptance

We define two kinds of acceptance mechanisms, closed acceptance and open acceptance. We say that the input is *accepted* if,

1. at some time $t = T$,

   (a) $q_1(T) \geq 1$ (*closed acceptance*)

   (b) $q_1(T) > 0$ (*open acceptance*)

   (recall that q_1 is the acceptance indicator and that initially $q_1(0) = 0$), and,

2. for one-way machines, additionally $s_0(T) = 1$, i.e., all of the input is ”read”.

6. Recognition

The set of functions *recognized* by M (within the set F_{PC}) consists of all inputs accepted by M, denoted by $F(M)$.

3 State Machines

By a *state machine* (*SM*) we mean a strictly one-way machine which does not have a functional memory. The corresponding differential equations are then

$$\frac{d\mathbf{q}}{dt} = \mathbf{Q}(\mathbf{q}(t), \hat{f}(t)) \quad \text{and} \quad \frac{ds_0}{dt} = S_0(\mathbf{q}(t), \hat{f}(t)).$$

Recall that for a strictly one-way state machine $S_0(\mathbf{q}, \hat{f}) > 0$. (In traditional automata theory this corresponds to the deterministic finite automaton.)

If open acceptance (resp. closed acceptance) is chosen, we use the acronym *OSM* (resp. *CSM*). The corresponding family of sets of piecewise continuous functions recognized by the machine is denoted by $\mathcal{F}(\mathrm{OSM})$ (resp. $\mathcal{F}(\mathrm{CSM})$). The family of complements of sets in $\mathcal{F}(\mathrm{OSM})$ (resp. $\mathcal{F}(\mathrm{CSM})$) is denoted by co–$\mathcal{F}(\mathrm{OSM})$ (resp. co–$\mathcal{F}(\mathrm{CSM})$). (Complements are naturally taken against F_{PC}.)

Theorem 2. *Every SM can be replaced by an equivalent SM with S_0 identically equal to* 1*, i.e.,* $s_0(t) = t$*. (In other words, every SM can be replaced by an equivalent real-time SM.)*

Proof. Take a state machine M recognizing the set $F(M)$. We use the notation above for the definition of M. We define another SM M' of the same dimension and with state $\tilde{\mathbf{q}}$ and position function $\tilde{s}_0$. The differential equations of M' are

$$\frac{d\tilde{\mathbf{q}}}{dt} = \frac{\mathbf{Q}(\tilde{\mathbf{q}}(t),\ \hat{\hat{f}}(t))}{S_0(\tilde{\mathbf{q}}(t),\ \hat{\hat{f}}(t))} \quad \text{and} \quad \frac{d\tilde{s}_0(t)}{dt} = 1.$$

Here the $\hat{\hat{f}}$ of M' is defined via $\tilde{s}_0$ and is not the same as the $\hat{f}$ of M. To see that $F(M) = F(M')$ we note first that $\hat{\hat{f}}(t) = f(t) = \hat{f}(s_0^{-1}(t))$. It is then a simple matter to verify that $\tilde{\mathbf{q}}(t) = \mathbf{q}(s_0^{-1}(t))$.

Thus the machine M is simulated by M', using the time $s_0^{-1}(t)$. □

In the sequel, we will assume that our state machines are real-time state machines. Note that then, for $0 < t < 1$,

$$\hat{f}(t) = f(t).$$

There is a close connection between the families $\mathcal{F}(\mathrm{OSM})$ and $\mathcal{F}(\mathrm{CSM})$.

Theorem 3. $\mathcal{F}(\mathrm{OSM}) = \text{co–}\mathcal{F}(\mathrm{CSM})$*, i.e., the families* $\mathcal{F}(\mathrm{OSM})$ *and* $\mathcal{F}(\mathrm{CSM})$ *are complementary. Moreover, it may be assumed that* $0 \le q_1(t) \le 1$.

Proof. We fix a continuously differentiable function $u : \mathbb{R} \longrightarrow \mathbb{R}$ such that $u(x) = 0$ for $x \le 0$, $0 < u(x) < 1$ for $0 < x < 1$, and $u(x) = 1$ for $u \ge 1$. We may assume that time t is a state component of our machines, denoted simply by t.

Take a (real-time) SM M. We then specify another SM M' by adding to M a new state component p (the acceptance indicator), changing the mode of acceptance from open to closed or vice versa, and setting

$$\frac{dp}{dt} = u'(t - q_1(t))(1 - Q_1(\mathbf{q}(t),\ \hat{f}(t))).$$

Then $p(t) = u(t - q_1(t))$, and $F(M') = F_{\mathrm{PC}} - F(M)$. □

For the purpose of comparison between machines of various kinds, in this section and later, several sets of functions are defined. First, we say that a function f (an input) is

- a *palindrome* if $f(1 - x) = f(x)$ for $0 < x < 1$.
- a *square* if $f(x + 1/2) = f(x)$ for $0 < x < 1/2$.

These concepts have familiar connotations for words. We then define the sets

$$\begin{aligned} F_{\text{pal}} &= \{ f \mid f \text{ is a palindrome} \}, \\ F_{\text{sqr}} &= \{ f \mid f \text{ is a square} \}, \\ F_{\text{poly}} &= \{ f \mid f \text{ is a polynomial} \} \quad \text{and} \\ F_{\text{poly}-d} &= \{ f \mid f \text{ is a polynomial of degree at most } d \}. \end{aligned}$$

The complements of these sets (against F_{PC}) are denoted by $\overline{F}_{\text{pal}}$, etc.

We need the following classical result which gives us a kind of ”weak pumping”.

Dimension Theorem. *Let A be an open subset of $\mathbb{R}^l$ and $l > k$. If $\mathbf{f} : A \longrightarrow \mathbb{R}^k$ is a continuous mapping then it is not injective.* □

Theorem 4. *None of the sets F_{pal}, F_{sqr} and F_{poly} is in $\mathcal{F}(\text{OSM})$ nor in $\mathcal{F}(\text{CSM})$. The same is true for the complements $\overline{F}_{\text{pal}}$, $\overline{F}_{\text{sqr}}$ and $\overline{F}_{\text{poly}}$.*

Proof. We show that $F_{\text{pal}} \notin \mathcal{F}(\text{OSM}), \mathcal{F}(\text{CSM})$. The other nonmemberships are proved analogously. The result then follows for the complements by Theorem 3.

Assume first, contrary to what is claimed, that F_{pal} is recognized by the OSM M. But then, since $0 \in F_{\text{pal}}$, for a sufficiently small value of $\varepsilon > 0$ the function

$$g : g(x) = \varepsilon e^x$$

is accepted by M, a contradiction. (Recall that we assumed continuous dependence of solutions on parameters in inputs.)

Assume second, contrary to what is claimed, that F_{pal} is recognized by the (real-time) CSM M. Take then an open ball B in $\mathbb{R}^{m_M+2}$. For $\mathbf{b}$ in $\mathbb{R}^{m_M+2}$ we denote

$$P_{\mathbf{b}}(x) = b_1 + b_2 x + \cdots + b_{m_M+2} x^{m_M+1.}$$

By our assumptions, the mapping $\mathbf{h} : B \longrightarrow \mathbb{R}^{m_M+1}$, mapping the point $\mathbf{b} \in B$ to the point $(\mathbf{q}(1/2),\ f_{\mathbf{b}}(1/2))$ where $\mathbf{q}(t)$ is obtained from M on the palindrome input

$$f_{\mathbf{b}} : f_{\mathbf{b}}(x) = \begin{cases} P_{\mathbf{b}}(x) \text{ for } 0 < x \leq 1/2 \\ P_{\mathbf{b}}(1-x) \text{ for } 1/2 \leq x < 1, \end{cases}$$

is continuous. By the Dimension Theorem, there are points $\mathbf{b}, \mathbf{b}' \in B$ such that $\mathbf{b} \neq \mathbf{b}'$ and $\mathbf{h}(\mathbf{b}) = \mathbf{h}(\mathbf{b}')$. This means, however, that the function

$$g : g(x) = \begin{cases} f_{\mathbf{b}}(x) \text{ for } 0 < x \leq 1/2 \\ f_{\mathbf{b}'}(x) \text{ for } 1/2 \leq x < 1 \end{cases}$$

is accepted by M. This is a contradiction since g is not in F_{pal}. □

Theorem 5. $F_{\text{poly}-d} \in \mathcal{F}(\text{CSM})$ *and* $F_{\text{poly}-d} \notin \mathcal{F}(\text{OSM})$.

Proof. Nonmembership of $F_{\text{poly}-d}$ in $\mathcal{F}(\text{OSM})$ is shown as in the previous proof. We then show only that $F_{\text{poly}-2}$ is in $\mathcal{F}(\text{CSM})$, the general case is treated quite analogously. A real-time CSM M recognizing $F_{\text{poly}-2}$ is constructed as follows. We set $m_M = 6$ and

$$\frac{dq_1}{dt} = 1 - q_5(t) \quad , \quad \frac{dq_2}{dt} = \hat{f}(t) \quad , \quad \frac{dq_3}{dt} = q_2(t) \; ,$$
$$\frac{dq_4}{dt} = q_3(t) \quad , \quad \frac{dq_5}{dt} = Q_5(\mathbf{q}(t),\, \hat{f}(t)) \quad , \quad \frac{dq_6}{dt} = 1$$

where Q_5 will be given later. On input

$$f \,:\, f(x) = ax^2 + bx + c$$

we have then $q_6(t) = t$ and

$$\hat{f}(t) = at^2 + bt + c \quad , \quad q_2(t) = \frac{a}{3}t^3 + \frac{b}{2}t^2 + ct \; ,$$
$$q_3(t) = \frac{a}{12}t^4 + \frac{b}{6}t^3 + \frac{c}{2}t^2 \quad , \quad q_4(t) = \frac{a}{60}t^5 + \frac{b}{24}t^4 + \frac{c}{6}t^3.$$

We denote

$$\mathbf{S}(t) = \begin{pmatrix} t^2 & t & 1 \\ \frac{1}{3}t^3 & \frac{1}{2}t^2 & t \\ \frac{1}{12}t^4 & \frac{1}{6}t^3 & \frac{1}{2}t^2 \end{pmatrix} \quad \text{and} \quad \mathbf{R}(t) = \begin{pmatrix} \frac{1}{3}t^3 & \frac{1}{2}t^2 & t \\ \frac{1}{12}t^4 & \frac{1}{6}t^3 & \frac{1}{2}t^2 \\ \frac{1}{60}t^5 & \frac{1}{24}t^4 & \frac{1}{6}t^3 \end{pmatrix}.$$

Then we can write

$$\mathbf{S}(t) \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} \hat{f}(t) \\ q_2(t) \\ q_3(t) \end{pmatrix} \quad \text{and} \quad \mathbf{R}(t) \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} q_2(t) \\ q_3(t) \\ q_4(t) \end{pmatrix}.$$

We note the following facts:

- First, $\mathbf{S}(t)$ and $\mathbf{R}(t)$, as Wronskians of linearly independent monomials, are invertible for $t \neq 0$.

- Second, for $t \neq 0$,

$$\begin{pmatrix} \frac{1}{60}t^5 & \frac{1}{24}t^4 & \frac{1}{6}t^3 \end{pmatrix} = \begin{pmatrix} c_1t^3 & c_2t^2 & c_3t \end{pmatrix} \mathbf{S}(t)$$

  for some constants c_1, c_2 and c_3.

Finally we choose

$$Q_5(\mathbf{q}(t),\, \hat{f}(t)) = \left(\begin{pmatrix} c_1q_6(t)^3 & c_2q_6(t)^2 & c_3q_6(t) \end{pmatrix} \begin{pmatrix} \hat{f}(t) \\ q_2(t) \\ q_3(t) \end{pmatrix} - q_4(t) \right)^2.$$

For an input f in $F_{\text{poly}-2}$ we have $q_5(t) = 0$ for $0 \le t \le 1$, and $q_1(1) = 1$, leading to acceptance of f.

On the other hand, if an input f is accepted by M, then $Q_5(\mathbf{q}(t),\, \hat{f}(t)) = 0$ for $0 \le t \le 1$ and $q_4(t)$ satisfies the Euler final value problem

$$c_1 t^3 \frac{d^3 q_4}{dt^3} + c_2 t^2 \frac{d^2 q_4}{dt^2} + c_3 t \frac{dq_4}{dt} - q_4 = 0 \ ,$$
$$q_4(1) = A \quad , \quad q_4'(1) = B \quad , \quad q_4''(1) = C$$

in the interval $0 < t \le 1$, for some constants A, B and C. The same final value problem is satisfied by

$$q_4(t) = \frac{a}{60} t^5 + \frac{b}{24} t^4 + \frac{c}{6} t^3 \quad \text{where} \quad \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \mathbf{R}(1)^{-1} \begin{pmatrix} C \\ B \\ A \end{pmatrix} .$$

Thus f is a polynomial of degree at most 2. (Note that, if $c_1 \ne 0$, then $Q_5(\mathbf{q}(t),\, \hat{f}(t)) = 0$ implies that $\hat{f}(t) = q_4'''(t)$ is continuous in the interval $0 < t \le 1$.) $\square$

Note. *Similar results can be proved for other sets of functions continuously depending on a fixed number of parameters.*

Theorems 3 and 5 tell us that the families $\mathcal{F}(\text{OSM})$ and $\mathcal{F}(\text{CSM})$ are incomparable, and neither of them is closed under complement. This incomparability and lack of closure is largely compensated by the complementarity of the families. On the other hand, they are closed under other Boolean operations.

Theorem 6. *The families* $\mathcal{F}(\text{OSM})$ *and* $\mathcal{F}(\text{CSM})$ *are closed under union and intersection.*

Proof. It suffices to prove the closures for $\mathcal{F}(\text{OSM})$. For $\mathcal{F}(\text{CSM})$ the closures then follow by Theorem 3 and De Morgan's laws.

Take then two (real-time) OSMs M and N, with states $\mathbf{q}$ and $\mathbf{p}$, and position functions s_0 and r_0, respectively. Let the state differential equations of M and N, on input f, be

$$\frac{d\mathbf{q}}{dt} = \mathbf{Q}(\mathbf{q}(t),\, \hat{f}(t)) \quad \text{and} \quad \frac{d\mathbf{p}}{dt} = \mathbf{P}(\mathbf{p}(t),\, \hat{f}(t)).$$

By Theorem 3 we may assume that $0 \le q_1(t),\, p_1(t) \le 1$.

We then take a new state component v (the new acceptance indicator). The real-time machine with state $(\mathbf{q}, \mathbf{p}, v)$ and state differential equations

$$\frac{d\mathbf{q}}{dt} = \mathbf{Q}(\mathbf{q}(t),\, \hat{f}(t)) \quad , \quad \frac{d\mathbf{p}}{dt} = \mathbf{P}(\mathbf{p}(t),\, \hat{f}(t)) \quad \text{and}$$
$$\frac{dv}{dt} = Q_1(\mathbf{q}(t),\, \hat{f}(t)) + P_1(\mathbf{p}(t),\, \hat{f}(t))$$

recognizes $F(M) \cup F(N)$. Note that here $v(t) = q_1(t) + p_1(t)$. Changing the differential equation of v to

$$\frac{dv}{dt} = Q_1(\mathbf{q}(t),\, \hat{f}(t)) p_1(t) + q_1(t) P_1(\mathbf{p}(t),\, \hat{f}(t))$$

we get a machine recognizing $F(M) \cap F(N)$, and $v(t) = q_1(t) p_1(t)$. $\square$

We close this section by an observation on the ”discrete computation power” CSMs.

Theorem 7. *Let L be the complement of a computably enumerable set in $\mathbb{N}$ and denote*

$$F = \{ f \mid f \text{ is constant and } f(x) \in L \text{ for } 0 < x < 1 \}.$$

Then $F \in \mathcal{F}(\mathrm{CSM})$.

Proof. We use ”pure state machines” or machines with an integer input, see [25]. There exists such a machine M recognizing $\mathbb{N} - L$ in the time interval $[0, 1]$. Note that acceptance of an input in $\mathbb{N}$ means here that the acceptance indicator q_1 grows from zero to a positive value; if the input is rejected then $q_1(t) = 0$ for $0 \le t \le 1$. M receives its input a as an initial value of certain state components. It is, however, easy to see that it might as well receive it as a parameter value (simply let the computation start by a copying of the parameter value a to the necessary state components). The differential equation of M is then of the form

$$\frac{d\mathbf{q}}{dt} = \mathbf{Q}(\mathbf{q}(t), a),$$

with zero initial values. (Actually, in [25], it is assumed that $\mathbf{Q}$ is defined in a finite interval of $\mathbf{q}$-values. It is easily seen that this interval can be replaced by the whole $\mathbb{R}^{m_M}$.)

We first take the real-time OSM M' with state $\mathbf{q}$ and state differential equation

$$\frac{d\mathbf{q}}{dt} = \mathbf{Q}(\mathbf{q}(t), \hat{f}(t)).$$

When restricted to constant inputs in $\mathbb{N}$, M' accepts exactly all numbers in $\mathbb{N} - L$. By Theorem 3, there is a CSM M_1 recognizing the complement of $F(M')$. Restricted to constant inputs in $\mathbb{N}$, M_1 accepts exactly all numbers in L.

Let M_2 be a CSM recognizing $F_{\mathrm{poly}-0}$ (i.e., constants), cf. Theorem 5. Finally we fix a continuous function $u : \mathbb{R} \longrightarrow \mathbb{R}$ such that $u(x) < 0$ for $x < 0$ and $u(x) = 0$ for $x \ge 0$, and take the real-time CSM M_3 with state q and state differential equation

$$\frac{dq}{dt} = \cos^2(\pi \hat{f}(t)) + u(\hat{f}(t)).$$

Restricted to constant inputs, M_3 accepts exactly all numbers in $\mathbb{N}$. Then $F = F(M_1) \cap F(M_2) \cap F(M_3)$ is in $\mathcal{F}(\mathrm{CSM})$ by Theorem 6. □

4 Two-Way State Machines

A *two-way state machine* (*2-SM*) is a machine with no functional memory. As indicated in Section 1, the state structure of such a machine has a very strong controlling capability, indeed, it has all the power of a Turing machine operating on integers. Very little of this capability can be used in computations of SMs. In order to utilize results computed by the state structure the machine needs to stop or indefinitely slow down reading its input. A two-way state machine can do this, and it can also re-read its input or read it in reverse.

Another property of 2-SMs, not possessed by SMs, is the ability to integrate over time intervals of arbitrary (finite) length. There is indeed no bound on the time accepting an input may take. This makes it possible for 2-SMs to amplify minute effects, either by a back-and-forth movement of the read-head, or by letting the read-head move forward or backward, or by letting it stand still for an unspecified time. It should be remembered, however, that the read-head cannot change direction infinitely many times in a finite time interval.

Note. *Apparently the "intermediate" possibility of allowing a state machine to be a (non-strict) one-way machine is of interest, too. (In traditional automata-theoretic terms, this would allow "empty moves".) The behaviour of such state machines is rather different from that of the SMs in Section 3—then e.g.* $\mathcal{F}(\mathrm{OSM}) \subset \mathcal{F}(\mathrm{CSM})$*—and is not dealt with in this paper.*

We use the notations 2-OSM, 2-CSM, $\mathcal{F}(\text{2-OSM})$ and $\mathcal{F}(\text{2-CSM})$ in an obvious fashion. We first prove some inclusions.

Theorem 8. *(i)* $\mathcal{F}(\text{2-OSM}) \subset \mathcal{F}(\text{2-CSM})$

(ii) $\mathcal{F}(\mathrm{OSM}) \subset \mathcal{F}(\text{2-OSM})$

(iii) $\mathcal{F}(\mathrm{CSM}) \subset \mathcal{F}(\text{2-CSM})$

Proof. Strictness of these inclusions is a consequence of Theorem 9. To prove the inclusions, we fix continuous functions $u_1, u_2 : \mathbb{R} \longrightarrow \mathbb{R}$ such that $u_1(x) = 0$ for $x \leq 0$ and $u_1(x) > 0$ for $x > 0$, $u_2(x) > 0$ for $0 < x < 1$, $u_2(x) = 0$ for $x \leq 0$ and $x \geq 1$, and $\int_0^1 u_2(x)\,dx = 1$. With the machine M (2-OSM, OSM or CSM) we associate the differential equations

$$\frac{d\mathbf{q}}{dt} = \mathbf{Q}(\mathbf{q}(t),\, \hat{f}(t)) \quad \text{and} \quad \frac{ds_0}{dt} = S_0(\mathbf{q}(t),\, \hat{f}(t)).$$

If M is strictly one-way, then we assume that $S_0 = 1$ and $0 \leq q_1(t) \leq 1$.

(i) Take a 2-OSM M. We then take a new state component p (the new acceptance indicator). The differential equations for our 2-CSM are those of M and

$$\frac{dp}{dt} = u_1(q_1(t)) + p(t).$$

If, at some time t, $q_1(t) > 0$ (indicating acceptance), then p starts growing and will eventually reach the acceptance treshold 1. This does not happen in any other situation.

(ii) Consider then an OSM M. We again take a new state component p and the differential equations

$$\frac{d\tilde{\mathbf{q}}}{dt} = \mathbf{Q}(\tilde{\mathbf{q}}(t),\, \hat{\tilde{f}}(t))u_2(t) \quad , \quad \frac{d\tilde{s}_0}{dt} = u_2(t) \quad \text{and}$$
$$\frac{dp}{dt} = \tilde{q}_1(t)u_1(t-1).$$

(We may assume that time t is a state component.) Here p grows above 0 if and only if $q_1(1) > 0$. Note that the initial value problem

$$\frac{dg}{dt} = u_2(t) \quad , \quad g(0) = 0,$$

defines a bijective time transformation $t' = g(t)$ in the interval $0 \le t \le 1$, and $\tilde{\mathbf{q}}(t) = \mathbf{q}(t')$, $\tilde{s}_0(t) = t'$ and $\hat{\tilde{f}}(t) = f(\tilde{s}_0(t)) = f(t')$. Thus the 2-OSM defined by the above differential equations simulates M in time t', stopping the simulation at time $t = t' = 1$.

(iii) Take finally a CSM M. An equivalent 2-CSM is then defined as in part (ii) except that

$$\frac{dp}{dt} = \tilde{q}_1(t)u_2(t-1).$$

□

Thus, while in traditional automata theory all kinds of finite automata (whether deterministic, nondeterministic, two-way, or with or without empty moves) are equivalent, this is not the case for our state machines.

The state structure may be given time to compute, while other parts do not evolve, by the following construct. The time axis is divided into *odd intervals* $[2i, 2i+1)$ and *even intervals* $[2i+1, 2i+2)$ $(i = 0, 1, \dots)$. Multiplying the right hand side of the differential equation of a state component by

$$\sigma_{\text{odd}}(t) = \max\left(0, \frac{\pi}{2}\sin\pi t\right) \quad (\text{resp. } \sigma_{\text{even}}(t) = \max\left(0, -\frac{\pi}{2}\sin\pi t\right))$$

forces it to evolve only during odd (resp. even) time intervals. Of course, if needed, the time axis may be divided into intervals modulo any K $(K = 2, 3, \dots)$ by a similar construct, resulting in the time-division functions $\sigma_k(t)$ $(k = 1, 2, \dots, K)$. Each part of the machine may then be given a time slot corresponding to positive values of some $\sigma_k(t)$. This is called *time-division modulo K*.

End of a computation and other signalling information can be communicated between parts of the machine using certain state components q_i as 0-1-flags and multiplying right hand sides of differential equations of the pertinent other state components by $q_i(t)$ or $1 - q_i(t)$. Thus one part of the machine may compute keeping the flag value in $q_i(t) = 0$ while other parts wait deactivated, and then signal end of its computation by raising the flag value to $q_i(t) = 1$, activating then certain other parts to continue their particular actions. Results of the computation may be communicated to the other parts of the machine via certain deactivated state components.

Obviously, exact description of such synchronization and control—not to mention simulation of universal Turing machines—via complete sets of differential equations leads to very complicated expressions. Therefore, only the basic ideas of such constructs are given in proofs here and in subsequent sections. For more details on such constructs see [23, 25].

Theorem 9. *(i) The set $F_{\text{poly}-d}$ is not in $\mathcal{F}$(2-OSM).*

(ii) None of the sets F_{pal}, F_{sqr} and F_{poly} is in $\mathcal{F}$(2-CSM).

(iii) The sets $\overline{F}_{\text{pal}}$, $\overline{F}_{\text{sqr}}$ and $\overline{F}_{\text{poly}-d}$ are in $\mathcal{F}$(2-OSM).

Proof. (i) See the proofs of Theorems 4 and 5. Note, however, that to use here the assumed continuous dependence of solutions on parameters in the input we must assume that f^* is, say, Lipschitz-continuous in $\mathbb{R}$. So, since $0 \in F_{\text{poly}-d}$, the assumption

$F_{\text{poly}-d} \in \mathcal{F}(\text{2-OSM})$ implies that, for a sufficiently small value of $\varepsilon > 0$, the function $g : g(x) = \varepsilon \sin \pi x$ is also in $\mathcal{F}(\text{2-OSM})$, a contradiction. ($g^*$ is Lipschitz-continuous in $\mathbb{R}$.)

(ii) We show that $F_{\text{pal}} \notin \mathcal{F}(\text{2-CSM})$ and refer to the proof of Theorem 4. The other nonmemberships are proved analogously. Assume, contrary to the claim, that F_{pal} is recognized by the 2-CSM M.

The constant function 0 is in F_{pal}. We consider first the computation of M accepting 0. Take a ξ, $0 < \xi < 1$, such that whenever $s_0(t) = \xi$ then $S_0(\mathbf{q}(t), \hat{f}(t)) \neq 0$, i.e., the read-head never stops at ξ. (It is easy to see that such ξ must exist.) Let $t_1, \dots, t_p$ be exactly all times when s_0 has the value ξ. (Note that, by our assumptions, these must be finite in number.)

Take then $f_{\mathbf{b}} \in F_{\text{pal}}$ where $\mathbf{b}$ is in a small ball $B \subset \mathbb{R}^{pm_M+p+2}$ centered in the origin, see the proof of Theorem 4, and $g_{\mathbf{b}} : g_{\mathbf{b}}(x) = f_{\mathbf{b}}(x)x(1-x)$. (Note that then $g_{\mathbf{b}} \in F_{\text{pal}}$ and $g^*_{\mathbf{b}}$ is Lipschitz-continuous in $\mathbb{R}$.) We consider now the computation of M accepting $g_{\mathbf{b}}$. Assuming B is small enough, the times when the read-head of M visits ξ are close to $t_1, \dots, t_p$ and their number is the same, and the read-head never stops at ξ. (Continuous dependence of $t_1, \dots, t_p$ on parameters in the input follows because $s_0^{-1}(\xi)$ has this property by the Implicit Function Theorem.) Let us denote these times by $t'_1, \dots, t'_p$. Let $\mathbf{h} : B \longrightarrow \mathbb{R}^{pm_M+p+1}$ be the continuous mapping defined by

$$\mathbf{h}(\mathbf{b}) = (\mathbf{q}(t'_1), \dots, \mathbf{q}(t'_p), t'_1, \dots, t'_p, g_{\mathbf{b}}(\xi)).$$

By the Dimension Theorem, there are points $\mathbf{b}, \mathbf{b}' \in B$ such that $\mathbf{b} \neq \mathbf{b}'$ and $\mathbf{h}(\mathbf{b}) = \mathbf{h}(\mathbf{b}')$. This is a contradiction since then M also accepts the non-palindrome

$$h : h(x) = \begin{cases} g_{\mathbf{b}}(x) \text{ for } 0 < x \leq \xi \\ g_{\mathbf{b}'}(x) \text{ for } \xi \leq x < 1. \end{cases}$$

(iii) We only sketch the proof of $\overline{F}_{\text{pal}} \in \mathcal{F}(\text{2-OSM})$. (The membership $\overline{F}_{\text{sqr}} \in \mathcal{F}(\text{2-OSM})$ is proved analogously, and $\overline{F}_{\text{poly}-d} \in \mathcal{F}(\text{2-OSM})$ follows from Theorems 3, 5 and 8.)

Using the control offered by the state structure of a 2-OSM the following procedure is carried out. For the successive values $l = 2, 3, \dots$ the 2-OSM M compares the input values

$$f\left(\frac{i}{2l}\right) \quad \text{and} \quad f\left(\frac{2l-i}{2l}\right) \quad (i = 1, 2, \dots, l-1).$$

For this purpose these values are copied to state components, say to q_2 and q_3, which are then deactivated. (The copying requires temporarily stopping movement of the position function.) M then continues by activating the differential equation

$$\frac{dq_1}{dt} = (q_2(t) - q_3(t))^2 + q_1(t)$$

for a while before raising the value of l. Note that, before moving to the next value of l, the components q_2 and q_3 must be reset to zero, again reading the input. If, for some l and i,

$$f\left(\frac{i}{2l}\right) \neq f\left(\frac{2l-i}{2l}\right)$$

then the acceptance indicator q_1 starts growing from its initial value 0, leading to acceptance of input. $\square$

Note that, by Theorems 4 and 5, this implies strictness of all inclusions in Theorem 8. The following corollary is also immediate.

Corollary 10. *The families* $\mathcal{F}(\text{CSM})$ *and* $\mathcal{F}(\text{2-OSM})$ *are incomparable.* $\square$

Note. *We assumed that the structure of our differential equations implies continuous dependence of solution on parameters in the input. For state machines this guarantees the continuity for all inputs sufficiently well-behaved in the interval* $(0,1)$. *Such is not the case any more for two-way state machines. The input may have jump discontinuities at* $x = 0$ *or* $x = 1$ *or elsewhere, and these can be utilized as jump discontinuities of the right hand side of our differential equations. Thus the solutions need not depend continuously on parameters in the input, even if the input is, say, Lipschitz-continuous in* $(0,1)$.

On the other hand, if the input f *has the property that* f^* *is, say, Lipschitz-continuous in* $\mathbb{R}$ *then continuous dependence on parameters in* f *is valid (as a consequence of our assumptions). This fact is utilized in the proof of Theorem 9.*

Theorem 9 shows that the families $\mathcal{F}(\text{2-OSM})$ and $\mathcal{F}(\text{2-CSM})$ are not closed under complement, indeed, complements of certain families in $\mathcal{F}(\text{2-OSM})$ are not even in $\mathcal{F}(\text{2-CSM})$. (The situation is thus quite different from that for state machines.) For Boolean operations we have

Theorem 11. *(i) The family* $\mathcal{F}(\text{2-OSM})$ *is closed under union and intersection.*

(ii) The intersection of a set in $\mathcal{F}(\text{2-OSM})$ *and a set in* $\mathcal{F}(\text{2-CSM})$ *is in* $\mathcal{F}(\text{2-CSM})$.

(iii) The family $\mathcal{F}(\text{2-CSM})$ *is closed under union.*

Proof. First, it should be noted that the construct of the proof of Theorem 6 is not applicable here.

Take then two 2-SMs M and N, with states $\mathbf{q}$ and $\mathbf{p}$, and position functions s_0 and r_0, respectively. For our new machine M', we use time-division modulo 4. The following sequence of four operations is then carried out cyclically by M':

1. Simulate M for time 1 and then deactivate this simulation.

2. Using s_0 and r_0 (initially zero) move the position function to the value of r_0 and then deactivate it.

3. Simulate N for time 1 and then deactivate this simulation.

4. Using r_0 and s_0 move the position function to the value of s_0 and then deactivate it.

We denote the state components of M', corresponding to $\mathbf{q}$, $\mathbf{p}$ and r_0, by $\tilde{\mathbf{q}}$, $\tilde{\mathbf{p}}$ and $\tilde{r}_0$, respectively. We also fix a continuous function $u : \mathbb{R} \longrightarrow \mathbb{R}$ such that $u(x) = 0$ for $x \leq 0$ and $u(x) > 0$ for $x > 0$.

(i) Assuming M and N are both 2-OSMs, it remains to define the acceptance mechanism of M'. We take new state components v_1, v_2 and v_3 (the new acceptance indicator), and define

$$\frac{dv_1}{dt} = u(\tilde{q}_1(t)) \quad \text{and} \quad \frac{dv_2}{dt} = u(\tilde{p}_1(t)).$$

To get $F(M') = F(M) \cup F(N)$ (resp. $F(M') = F(M) \cap F(N)$) we define simply

$$\frac{dv_3}{dt} = v_1(t) + v_2(t) \quad \text{(resp. } \frac{dv_3}{dt} = v_1(t)v_2(t)\text{)}.$$

(ii) Assume then that M is a 2-OSM and N a 2-CSM. We take a new state component v with the differential equation

$$\frac{dv}{dt} = u(\tilde{q}_1(t)) + v(t).$$

and then modify the differential equations of $\tilde{\mathbf{p}}$ and $\tilde{r}_0$ as follows:

$$\frac{d\tilde{\mathbf{p}}}{dt} = \tilde{\mathbf{P}}(\tilde{\mathbf{p}}(t),\ \hat{f}(t))v(t) \quad \text{and} \quad \frac{d\tilde{r}_0}{dt} = \tilde{R}_0(\tilde{\mathbf{p}}(t),\ \hat{f}(t))v(t).$$

If M does not accept the input, then $v(t) = 0$ for all $t \geq 0$ and the simulation of N by M' never starts. If, on the other hand, the input is accepted by M then, for some t, $v(t) > 0$ and the simulation of N by M' starts. Since then $\lim_{t\to\infty} v(t) = \infty$, the simulation is carried out to the end. Using the acceptance indicator $\tilde{p}_1$, the input is thus accepted by M' if only if it is accepted by both M and N.

(iii) Assume M and N are both 2-CSMs. Only slight modifications of the above basic construct of M' are needed here. First, a new acceptance indicator v is chosen for M'. Whenever M (resp. N) is simulated by M', then $v(t)$ is equal to $\tilde{q}_1(t)$ (resp. $\tilde{p}_1(t)$). The value of v is updated during steps 2. and 4., exactly as is the value of the position function. □

Closure of the family $\mathcal{F}$(2-CSM) under intersection remains an open problem.

5 Push-Down Machines

We recall that for a *push-down machine* (*PDM*) M we have $n_M = 1$ and $\hat{x}_1$ is defined by

$$\hat{x}_1(T) = \begin{cases} x^*_{1,T}(s_1(T)-), \text{ if } s'_1(T) < 0 \\ x_1(T), \text{ if } s'_1(T) \geq 0. \end{cases}$$

The corresponding differential equations are then

$$\frac{d\mathbf{q}}{dt} = \mathbf{Q}(\mathbf{q}(t),\ \hat{f}(t), \hat{x}_1(t)) \quad , \quad \frac{ds_0}{dt} = S_0(\mathbf{q}(t),\ \hat{f}(t), \hat{x}_1(t)) \ ,$$
$$\frac{ds_1}{dt} = S_1(\mathbf{q}(t),\ \hat{f}(t), \hat{x}_1(t)) \quad \text{and} \quad \frac{dx_1}{dt} = X_1(\mathbf{q}(t),\ \hat{f}(t), \hat{x}_1(t)).$$

By definition, PDM is a strictly one-way machine, i.e. $S_0(\mathbf{q},\ \hat{f},\hat{x}_1) > 0$. It is also assumed that always $s_1(t) \geq 0$. We use the notations OPDM, CPDM, $\mathcal{F}$(OPDM), $\mathcal{F}$(CPDM), etc. in an obvious fashion.

A PDM can only read its functional memory when the "read-write-head" is moving left. (Indeed, when the "read-write-head" is moving right or is stopped, i.e., $S_1(\mathbf{q}(t)) \geq 0$, we have $\hat{x}_1(t) = x_1(t)$, and x_1 could be included among the state components.) In this respect it is quite like its namesake in traditional automata theory, the push-down automaton. Our machines have, however, forward-unique solutions, and thus the corresponding type of automaton is actually the deterministic push-down automaton. Deterministic context-free languages have rather poor closure properties, and so appear to do the sets $\mathcal{F}$(OPDM) and $\mathcal{F}$(CPDM).

Note. *The traditional DPDA has empty moves, corresponding to zero-values of our $S_0(\mathbf{q})$. Our definition above thus corresponds to DPDAs without empty moves. It may be noted, however, that deterministic context-free languages can be recognized by DPDAs in linear time. One-way PDMs certainly appear to be an interesting subclass of machines, but they are not investigated any further here.*

It is immediate that $\mathcal{F}(\text{OSM}) \subseteq \mathcal{F}(\text{OPDM})$ and $\mathcal{F}(\text{CSM}) \subseteq \mathcal{F}(\text{CPDM})$. Both inclusions are strict as a consequence of Theorems 4 and 14.

Theorem 12. *Every PDM can be replaced by an equivalent PDM with S_0 identically equal to* 1*, i.e., $s_0(t) = t$. (In other words, every PDM can be replaced by an equivalent real-time PDM.)*

Proof. We refer to the proof of Theorem 2. The differential equations above are replaced by

$$\frac{d\tilde{\mathbf{q}}}{dt} = \frac{\mathbf{Q}(\tilde{\mathbf{q}}(t),\ \hat{\tilde{f}}(t),\hat{\tilde{x}}_1(t))}{S_0(\tilde{\mathbf{q}}(t),\ \hat{\tilde{f}}(t),\hat{\tilde{x}}_1(t))} \quad , \qquad \frac{d\tilde{s}_0}{dt} = 1\ ,$$

$$\frac{d\tilde{s}_1}{dt} = \frac{S_1(\tilde{\mathbf{q}}(t),\ \hat{\tilde{f}}(t),\hat{\tilde{x}}_1(t))}{S_0(\tilde{\mathbf{q}}(t),\ \hat{\tilde{f}}(t),\hat{\tilde{x}}_1(t))} \quad \text{and} \quad \frac{d\tilde{x}_1}{dt} = \frac{X_1(\tilde{\mathbf{q}}(t),\ \hat{\tilde{f}}(t),\hat{\tilde{x}}_1(t))}{S_0(\tilde{\mathbf{q}}(t),\ \hat{\tilde{f}}(t),\hat{\tilde{x}}_1(t))}.$$

As before, we have $\hat{\tilde{f}}(t) = \hat{f}(s_0^{-1}(t))$. Now, assuming $\hat{\tilde{x}}_1(t) = \hat{x}_1(s_0^{-1}(t))$, a simple calculation shows that

$$\tilde{\mathbf{q}}(t) = \mathbf{q}(s_0^{-1}(t)) \quad , \quad \tilde{s}_1(t) = s_1(s_0^{-1}(t)) \quad \text{and} \quad \tilde{x}_1(t) = x_1(s_0^{-1}(t)).$$

We have taken here simply $\tilde{\hat{x}}_1(t) = \hat{x}_1(s_0^{-1}(t))$, so it remains to show that $\hat{\tilde{x}}_1(t) = \tilde{\hat{x}}_1(t)$. Since s_0 is strictly increasing,

$$\tilde{t}^*_{1,T}(s) = \max_{\tilde{s}_1(t)=s} t = s_0\left(\max_{s_1(s_0^{-1}(t))=s} s_0^{-1}(t)\right) = s_0(t^*_{1,s_0^{-1}(T)}(s)).$$

It is then easily verified that $\tilde{x}^*_{1,T}(s) = x^*_{1,s_0^{-1}(T)}(s)$ and finally $\hat{\tilde{x}}_1(t) = \hat{x}_1(s_0^{-1}(t))$. $\square$

As was the case for state machines, the families $\mathcal{F}(\mathrm{OPDM})$ and $\mathcal{F}(\mathrm{CPDM})$ are complementary. (In many ways this result corresponds to the closure under complementation of traditional deterministic context-free languages.)

Theorem 13. $\mathcal{F}(\mathrm{OPDM}) = \text{co–}\mathcal{F}(\mathrm{CPDM})$, *i.e., the families* $\mathcal{F}(\mathrm{OPDM})$ *and* $\mathcal{F}(\mathrm{CPDM})$ *are complementary. Moreover, it may be assumed that* $0 \leq q_1(t) \leq 1$.

Proof. See the proof of Theorem 3. □

Theorem 14. *(i) The set* F_{pal} *is in* $\mathcal{F}(\mathrm{CPDM})$ *but not in* $\mathcal{F}(\mathrm{OPDM})$.

(ii) The sets F_{sqr}, $F_{\mathrm{poly}-d}$ *and* F_{poly} *are not in* $\mathcal{F}(\mathrm{OPDM})$.

(Corresponding statements for the complements of the sets are obtained in an obvious way by Theorem 13.)

Proof. The nonmembership of F_{pal}, F_{sqr}, F_{poly} and $F_{\mathrm{poly}-d}$ in $\mathcal{F}(\mathrm{OPDM})$ is proved similarly in all cases . We prove here the claim $F_{\mathrm{pal}} \notin \mathcal{F}(\mathrm{OPDM})$. Assume the contrary. There is then a (real-time) OPDM M which recognizes F_{pal}. Let M accept zero input (which is in F_{pal}), and, considering this accepting computation, denote the maximum value of $|x_1(t)|$ by U, and

$$K = \{(\mathbf{q}(t), 0, x) \mid 0 \leq t \leq 1 \text{ and } |x| \leq U\} \subset \mathbb{R}^{m_M+2}.$$

Furthermore, for some $\alpha > 0$ and $\delta > 0$, we have then $q_1(t) > \alpha$ for $1 - \delta \leq t \leq 1$.

Choose next an R such that in $\mathbb{R}^{m_M+2}$ the distance of any point $\mathbf{r}$ with $\|\mathbf{r}\| = R$ from K is at least 1. Then, because of continuity, $\|(\mathbf{Q}, 1, X_1)(\mathbf{r})\|$ is bounded in the ball $\|\mathbf{r}\| \leq R$, say by C.

Define then

$$g_\varepsilon : g_\varepsilon(x) = \begin{cases} 0 \text{ for } 0 \leq x \leq 1 - \varepsilon \\ x + \varepsilon - 1 \text{ for } 1 - \varepsilon \leq x \leq 1. \end{cases}$$

Obviously g_ε is not in F_{pal} for small $\varepsilon > 0$, but on input g_ε the solution curve

$$\mathcal{C}_\varepsilon : \mathbf{r} = (\mathbf{q}(t), g_\varepsilon(t), x_1(t)) \quad (0 \leq t \leq 1 - \varepsilon)$$

is then in K. Since g_ε is rejected by M, for all sufficiently small ε, $0 < \varepsilon < \delta$, the solution curve $\mathcal{C}_\varepsilon$ must escape the ball $\|\mathbf{r}\| \leq R$ within the time interval $1 - \delta \leq t \leq 1$, in order for $|Q_1|$ to achieve arbitrarily large values. This is impossible if $\delta < 1/C$ since escaping the ball starting from K takes at least time $1/C$.

We show next that $F_{\mathrm{pal}} \in \mathcal{F}(\mathrm{CPDM})$ by a direct construction, which of course resembles very much that for the traditional PDA. For this purpose we fix a continuous function $u : \mathbb{R} \longrightarrow \mathbb{R}$ such that $u(x) = 0$ for $x \leq 0$ and $x \geq 1/2$, and $u(x) > 0$ for $0 < x < 1/2$. We then set $m_M = 4$ and

$$\frac{dq_1}{dt} = 1 - (\hat{x}_1(t) - q_2(t) + q_3(t))^2 \quad , \quad \frac{dq_2}{dt} = u(q_4(t))\,\hat{f}(t)\ ,$$
$$\frac{dq_3}{dt} = u(1 - q_4(t))\,\hat{f}(t) \quad , \quad \frac{dq_4}{dt} = 1 \quad , \quad \frac{ds_0}{dt} = 1\ ,$$
$$\frac{ds_1}{dt} = \frac{\pi}{2}\sin 2\pi q_4(t) \quad \text{and} \quad \frac{dx_1}{dt} = u(q_4(t))\,\hat{f}(t).$$

(For $q_4 < 0$ and $q_4 > 1$ we define $S_1(\mathbf{q}, \hat{f}, \hat{x}_1) = 0$.) For $0 \le t \le 1/2$ we have then

$$q_1(t) = q_4(t) = s_0(t) = t \quad , \quad q_2(t) = x_1(t) = \hat{x}_1(t) = \int_0^t u(x)\, f(x)\, dx \ ,$$
$$q_3(t) = 0 \quad \text{and} \quad s_1(t) = \frac{1}{2}\sin^2 \pi t.$$

For $1/2 \le t \le 1$ we have $t = 1/2 + \tau$ for some $\tau \ge 0$,

$$q_2(t) = \int_0^{1/2} u(x)\, f(x)\, dx \quad , \quad q_3(t) = \int_{1/2}^{1/2+\tau} u(1-x)\, f(x)\, dx = \int_{1/2-\tau}^{1/2} u(x)\, f(1-x)\, dx \ ,$$
$$q_4(t) = s_0(t) = t \quad , \quad s_1(t) = \frac{1}{2}\sin^2 \pi(1/2+\tau) = \frac{1}{2}\sin^2 \pi(1/2-\tau) \quad \text{and}$$
$$\hat{x}_1(t) = \int_0^{1/2-\tau} u(x)\, f(x)\, dx.$$

If $f \in F_{\text{pal}}$ then $f(1-x) = f(x)$, and for $t \ge 1/2$ we have $q_2(t) - q_3(t) = \hat{x}_1(t)$. Thus f is accepted. If $f \notin F_{\text{pal}}$, then for some t_1, $1/2 < t_1 < 1$, we have $q_3(t_1) \neq q_2(t_1) - \hat{x}_1(t_1)$. It follows that $q_1(1) < 1$ and f is rejected. □

The proof of Theorem 6 is readily applicable here and so we have

Theorem 15. *The family* $\mathcal{F}$(CPDM) *(resp.* $\mathcal{F}$(OPDM)*) is closed under intersection and union with sets in* $\mathcal{F}$(CSM) *(resp.* $\mathcal{F}$(OSM)*).* □

6 Two-Way Push-Down Machines

As for a PDM, for a *two-way push-down machine* (*2-PDM*) M we have $n_M = 1$ and $\hat{x}_1$ is defined by

$$\hat{x}_1(T) = \begin{cases} x^*_{1,T}(s_1(T)-), \text{ if } s_1'(T) < 0 \\ x_1(T), \text{ if } s_1'(T) \ge 0. \end{cases}$$

The difference is that a 2-PDM is not a one-way machine, i.e. $S_0(\mathbf{q}, \hat{f}, \hat{x}_1)$ may have both positive and negative values, and zero values. It is still assumed that always $s_1(t) \ge 0$. We use the notations 2-OPDM, 2-CPDM, $\mathcal{F}$(2-OPDM), $\mathcal{F}$(2-CPDM), etc. in an obvious fashion.

Obviously $\mathcal{F}$(2-OSM) $\subseteq$ $\mathcal{F}$(2-OPDM) and $\mathcal{F}$(2-CSM) $\subseteq$ $\mathcal{F}$(2-CPDM). The latter inclusion is strict since $F_{\text{pal}} \in \mathcal{F}$(2-CPDM) $-$ $\mathcal{F}$(2-CSM) (see Theorems 9 and 14 and Theorem 16 below). The former inclusion is strict because $F_{\text{poly}-d} \in \mathcal{F}$(2-OPDM) $-$ $\mathcal{F}$(2-OSM) (see Theorem 9 and Theorem 19 below). In fact, as will be noted in Section 8, $\mathcal{F}$(2-CSM) $\subset$ $\mathcal{F}$(2-OPDM).

Theorem 16. *(i)* $\mathcal{F}$(2-OPDM) $\subseteq$ $\mathcal{F}$(2-CPDM) *(ii)* $\mathcal{F}(\text{OPDM}) \subset \mathcal{F}(\text{2-OPDM})$

(iii) $\mathcal{F}(\text{CPDM}) \subset \mathcal{F}(\text{2-CPDM})$

Proof. See the proof of Theorem 8. Strictness of the inclusion (iii) follows because $\overline{F}_{\text{pal}} \in \mathcal{F}(\text{2-OPDM}) - \mathcal{F}(\text{CPDM})$ (see Theorems 9 and 14). Strictness of the inclusion (ii) in turn follows because $F_{\text{pal}} \in \mathcal{F}(\text{2-OPDM}) - \mathcal{F}(\text{OPDM})$ (see Theorem 14 and Corollary 20 below). □

Strictness of the inclusion (i) in the above theorem remains open. Indeed, 2-PDMs are already quite powerful machines and we lack techniques for proving nonmembership.

Theorem 17. *The set F_{sqr} is in* $\mathcal{F}(\text{2-CPDM})$.

Proof. The construct is almost the same as that of Theorem 14(i). The only difference is that the input segment $f(x)$, $1/2 < x < 1$, is read from right to left. □

2-PDMs have the ability to perform operations of numerical analysis. More spefically, we define a *numerical analysis machine* as follows:

(A) The input is a piecewise continuous function f in the interval [0, 1]. The machine has a black box which, given a finite floating point representation x, $0 \leq x \leq 1$, returns the corresponding finite floating point representation of $f(x)$, using a given scheme of rounding numbers.

(B) The floating point presentation is given, as usual, by the sign, the mantissa and the exponent, in the integer interval $1, 2, \ldots, N$. Here N is a precision parameter which can be increased by the machine. Any number of integers, and thus any number of floating point numbers, can be stored by the machine.

(C) The machine has universal computing power on integers.

For 2-PDMs, $f(x)$ is readily available at any time and the state control structure has a universal computing power on integers. It is the rounding of reals to integers that needs to be explained. Such a rounding is not possible without losing stability, and thus must be performed using the push-down memory. One way of doing it is the following. First, a unit-step is created in the push-down memory:

1. Stop the read-write-head at some position y_0 (i.e., take the value of s_1' to 0 and hold it there).
2. Reset the values of x_1 and x_1' to 0.
3. Move the read-write-head from y_0 to $y_0 + 1$, stopping it there and keeping $x_1 = 0$.
4. Raise the value of x_1 from 0 to 1, resetting x_1' to 0 and keeping $s_1 = y_0 + 1$.
5. Move the read-write-head from $y_0 + 1$ to $y_0 + 2$, stopping it there and keeping $x_1 = 1$.
6. Lower the value of x_1 from 1 to 0, resetting x_1' to 0 and keeping $s_1 = y_0 + 2$.

Rounding a real number, say z, to integer may be accomplished by comparing z to i for $i = 0, \pm 1, \pm 2, \ldots$, until an integer j is found such that $j \leq z < j + 1$. The comparison is done using the created unit-step at $y_0 + 1$. We denote $w = \frac{2}{\pi} \arctan(z - i)$, and continue the process as follows:

7. Move the read-write-head from $y_0 + 2$ to $y_0 + 1 + w$ and immediately switch on the controlling differential equation

$$\frac{ds_1}{dt} = -s_1(t) + y_0 + 1$$

(s_1 may be assumed to be among the state components), all the time keeping $x_1 = 0$ and $x_1' = 0$. While this differential equation controls s_1 we have

$$s_1(t) = y_0 + 1 + we^{-t+t_0},$$

where t_0 is the time of switching on the differential equation, and hence either $z \leq i$ and $\hat{x}_1(t) = 0$ or $z > i$ and $\hat{x}_1(t) = 1$.

We have thus

Theorem 18. *Any numerical analysis machine can be simulated by a 2-PDM.* □

It may be noted that the theorem holds true already for 2-SMs, if their inputs are confined to ones containing the needed step structure. We also have

Theorem 19. *Let the set $F \in \mathcal{F}$(2-CPDM) be recognized by a 2-CPDM M such that for some constant $T > 0$ all inputs in F are accepted in time T, i.e., for any input in F, $q_1(T) \geq 1$. Then $F \in \mathcal{F}$(2-OPDM).*

Proof. The 2-OPDM recognizing F simulates M for time T, deactivates the differential equation of q_1 and then rounds $q_1(T)$ to $\lfloor q_1(T) \rfloor$. □

All of $\mathcal{F}$(CPDM) and many sets known to be in $\mathcal{F}$(2-CPDM) are seen to be already in $\mathcal{F}$(2-OPDM) by this theorem (but remember that equality of the latter two families is open):

Corollary 20. $\mathcal{F}$(CPDM) $\subset \mathcal{F}$(2-OPDM) *and* $F_{\text{sqr}} \in \mathcal{F}$(2-OPDM). □

(Strictness of the inclusion follows because $\overline{F}_{\text{pal}} \in \mathcal{F}$(2-OPDM) $- \mathcal{F}$(CPDM).)

Note. *Actually, a stronger form of Theorem 19 holds true: If for an input of M the time T of possible acceptance, which may vary among inputs, can be numerically computed, then the conclusion of the theorem holds true. This indicates that even if $\mathcal{F}$(2-OPDM) is a proper subfamily of $\mathcal{F}$(2-CPDM), it may be quite difficult to prove it.*

7 Linear-Bounded Machines?

The family of context-sensitive languages played initially a prominent role in the traditional Chomsky hierarchy. Nowadays it is mainly thought of as a subset of the space complexity class $\mathcal{PSPACE}$. (Indeed, some modern text-books on formal languages pass over CS-languages, see e.g. [13].) The corresponding automata type is the linear-bounded automaton (LBA). Finding a counterpart for LBAs—or other space/time-bounded Turing machines—among our machines is problematic. The simple reason for this is discussed below.

Maximum length of the interval where the position functions take their values is not a proper space measure for our machines. Indeed, replacing s_i by $\tilde{s}_i = \arctan s_i$ and writing

$$\frac{d\tilde{s}_i}{dt} = \frac{S_i(\mathbf{q}(t),\ \hat{f}(t), \hat{\mathbf{x}}(t))}{1 + \tan^2 \tilde{s}_i(t)}$$

restricts the position to the interval $(-\pi/2, \pi/2)$. (We may assume that $\tilde{s}_i$ appears as a state component.) Similarly we may replace x_i by $\tilde{x}_i = \arctan x_i$ $(i = 1, \dots, n_M)$ and write

$$\frac{d\mathbf{q}}{dt} = \mathbf{Q}(\mathbf{q}(t),\ \hat{f}(t), \tan \hat{\tilde{x}}_1(t), \dots, \tan \hat{\tilde{x}}_{n_M}(t)),$$

etc., which restricts the range of $\tilde{\mathbf{x}}$ to the interval $(-\pi/2, \pi/2)^{n_M}$. Size of range of $\mathbf{x}$ thus does not appear to be a meaningful measure of space either. Note that our assumption on continuity of right hand sides of the differential equations is then violated, but, less drastic compression of $\mathbf{s}$ and $\mathbf{x}$ is of course possible while retaining the continuity.

Time is equally malleable. Multiplying the right hand sides of the differential equations by $1 + \tan^2 t$ replaces the semi-infinite time interval $[0, \infty)$ by $[0, \pi/2)$. (Again, we consider t as a state component.) This, however, might violate our assumption of Zenoan computations, e.g. not allowing the derivatives $s_i'(t)$ to change sign infinitely many times in any finite time interval, and continuity of right-hand sides as well. Of course, finite accelerations of computations are possible, too, which do not lead to violation of our initial assumptions.

Note. *The above time transformation creates a singularity at $t = \pi/2$. This probably is not as serious as it looks: The main result of* [25] *shows that for "pure state machines" this singularity can be removed.*

Restriction of machine type to (strictly) one-way machines or/and fixing or bounding the dimension of the functional memory certainly are ways to limit use of resources, but these do not seem to lead machines resembling LBAs in any particular way.

8 General Machines

The largest family in the Chomsky hierarchy is the family of computably enumerable languages (aka recursively enumerable languages), and the corresponding automata type is the (deterministic) Turing machine. The most general machine in our case is obtained when no restrictions are placed on the dimension or type of the functional memory or the moves of the read-head. We call it simply the *general machine* (*GM*), and use the corresponding notations OGM, CGM, $\mathcal{F}$(OGM), $\mathcal{F}$(CGM), etc., as before.

It is immediate that $\mathcal{F}$(2-OPDM) $\subseteq$ $\mathcal{F}$(OGM) and $\mathcal{F}$(2-CPDM) $\subseteq$ $\mathcal{F}$(CGM), the strictness of these inclusions, however, remains open. Obviously, GMs have all the power of 2-PDMs (e.g. the ability to do all numerical analysis computations, see Theorem 18). There is no difference between open and closed acceptance for GMs:

Theorem 21. $\mathcal{F}$(OGM) $=$ $\mathcal{F}$(CGM).

Proof. Proof of the inclusion $\mathcal{F}$(OGM) $\subseteq$ $\mathcal{F}$(CGM) is similar to the proof of Theorem 8. To show the reverse inclusion take a CGM M. We construct an OGM M' recognizing $F(M)$. We add a new state component p and a new functional memory component y, and the corresponding position function r. The computation of M' begins by initializing y^*_T to

$$y^*_T(s) = \begin{cases} 1 \text{ for } 0 < s < 2 \\ 0 \text{ for } s < 0 \text{ and } s > 2 \end{cases}$$

within a certain time interval, and then setting both $y(t)$ and $r(t)$ to 0. Simulation of the computation of M then begins (for this purpose M is embedded in M') and the evolution of the new components is given by

$$\frac{dy}{dt} = 0 \quad , \quad \frac{dr}{dt} = Q_1(\mathbf{q}(t),\ \hat{f}(t), \hat{\mathbf{x}}(t)) \quad \text{and}$$
$$\frac{dp}{dt} = \hat{y}(t) Q_1(\mathbf{q}(t),\ \hat{f}(t), \hat{\mathbf{x}}(t)).$$

During this stage at time $t = T$ we have thus

$$y(T) = 0 \quad , \quad y^*_T(s) = \begin{cases} 1 \text{ for } p(T) < s < 2 \\ 0 \text{ for } s \le p(T) \text{ and } s > 2 \end{cases} ,$$

$$r(T) = q_1(T) \quad \text{and} \quad p(T) = \min\left(2, \max_{0 \le t \le T} q_1(t)\right).$$

Note how p retains information about whether or not q_1 has reached the value 1. This might happen for one single moment of time only, and thus could not be used to directly trigger open acceptance.

The construction of M' is finished using time-division modulo 2 (see Section 4) where during odd time intervals the above process is carried out, and in each even time interval a unit-step construct (see Section 6) is used to compute $\lfloor p(t) \rfloor$. If $\lfloor p(t) \rfloor \ge 1$ then the value of the (new) acceptance indicator state is raised from 0 to a positive value. $\square$

We now drop the letters indicating type of acceptance, and use the notations GM, $\mathcal{F}$(GM), etc.

Note. *A construct similar to the one in the previous proof may be used to show that a 2-CSM can be simulated by a 2-OPDM. Since only one memory component is available for a 2-OPDM, the contents of the two memories (or positions of unit-steps) must be stored and retrieved in an alternating fashion as in the proof of Theorem 11.*

Since there is no restriction on dimension of the functional memory, a GM can simulate two (or more) GMs, much as a 2-SM can simulate two 2-SMs (see Theorem 11 and its proof). We have thus

Theorem 22. *The family $\mathcal{F}(\mathrm{GM})$ is closed under union and intersection.* $\square$

We do not know whether or not F_{poly} is in $\mathcal{F}(\mathrm{CPDM})$ or even in $\mathcal{F}(2\text{-CPDM})$, but we have

Theorem 23. *The family F_{poly} is in $\mathcal{F}(\mathrm{GM})$.*

Proof. We give a somewhat sketchy proof here, and refer to the proof of Theorem 5. The general idea is to check through degrees $d = 0, 1, \ldots$ whether or not the input f is a polynomial of degree d.

First, using two functional memory components repeatedly in an alternating fashion, the cumulative integrals

$$c_i = \int_0^1 \int_0^{t_i} \cdots \int_0^{t_2} \int_0^{t_1} f(x)\,dx\,dt_1\,dt_2 \cdots dt_i \quad (i = 0, 1, \ldots, d)$$

are computed and their values are stored. A piecewise constant functional memory component is used for the storing since d can be arbitrarily high. (Writing into and reading from such a memory is similar to the use of unit-steps in the previous sections.) From these numbers c_i the coefficients $a_0, a_1, \ldots, a_d$ of the candidate polynomial are computed as in the proof of Theorem 5, and stored in another piecewise constant memory component. Note that these coefficients are linear combinations of the numbers $c_0, c_1, \ldots, c_d$ with rational coeffients which can be computed by the state control.

The candidate polynomial $P_d(x)$ is then constructed by repeated integration using the recursion

$$\begin{aligned} P_0(x) &= d!a_d\ , \\ P_i(x) &= (d-i)!a_{d-i} + \int_0^x P_{i-1}(y)\,dy \quad (i = 1, 2, \ldots, d), \end{aligned}$$

and stored. Finally the integral

$$I = \int_0^1 (f(x) - P_d(x))^2\,dx$$

is computed. A unit-step construct (see Section 6) is used to check whether or not $I = 0$. In the positive case the input is accepted. In the negative case d is replaced by $d + 1$, and the search for the polynomial continues. $\square$

The cardinality of the set of all GMs is the cardinality of the continuum and hence lower than that of $2^{F_{\mathrm{PC}}}$. There are thus subsets of F_{PC} not recognized by any GM. An explicit example of such a set can be obtained by diagonalization and the celebrated Kolmogorov Superposition Theorem. The theorem has recently found use in neural network theory, we quote in full a refinement obtained by David Sprecher:

Kolmogorov Superposition Theorem. ([27], Theorem 1) *Let* $\{\lambda_k\}$ *be a sequence of positive integrally independent numbers. There exists a continuous monotonically increasing function* $\psi : [0, 1/5!] \longrightarrow [0, 1/5!]$ *having the following property: For every real-valued continuous function* $f : [0, 1]^n \longrightarrow \mathbb{R}$ *with* $n \geq 2$ *there are continuous functions* Φ_q *such that*

$$f(x_1, \ldots, x_n) = \sum_{q=0}^{2n} \Phi_q \left(\sum_{p=1}^{n} \lambda_q \psi(x_p + q a_n) \right),$$

for a suitable constant a_n. $\square$

It is not difficult to see that this theorem can be used to give a coding of any GM as a function in F_{PC}. We fix one such coding scheme, and denote the code of a GM M by γ_M. We define then

$$D = \{\gamma_M \mid M \text{ is a GM and } \gamma_M \notin F(M)\}.$$

It is immediate that D cannot be recognized by any GM.

Note. *Despite the Gödel-number-like properties of* γ_M *it appears that it cannot be used to construct a universal GM. It may be noted, though, that* γ_M *is effectively obtainable, if not explicitly then at least numerically, see* [28, 29] *and the references therein.*

The family of computable languages (aka recursive languages) may be defined as the family of languages L such that both L and $\overline{L}$ are computably enumerable. Using GMs, the corresponding family is

$$\mathcal{C} = \mathcal{F}(\mathrm{GM}) \cap \text{co–}\mathcal{F}(\mathrm{GM}).$$

Sets of functions in $\mathcal{C}$ are in a sense "decidable" by our machines. It is immediate that $\mathcal{C}$ contains $\mathcal{F}(\mathrm{OPDM})$ and $\mathcal{F}(\mathrm{CPDM})$. On the other hand, D is an example of an "undecidable" set.

An interesting open problem is whether $\mathcal{C}$ is proper subset of $\mathcal{F}(\mathrm{GM})$. (In traditional formal language theory a central result states that not all computably enumerable languages are computable.) By Theorem 23, F_{poly} is in $\mathcal{F}(\mathrm{GM})$, but it is not known whether $\overline{F}_{\mathrm{poly}} \in \mathcal{F}(\mathrm{GM})$. Thus F_{poly} might resolve the problem.

9 The Hierarchies

We collect here in Figure 2 in a graphical form the Chomsky-like hierarchies obtained in the previous sections. We have basically four hierarchies, depending on whether or not two-way machines are used and which of the two types of acceptance is adopted. Each of these four hierarchies resembles the traditional Chomsky hierarchy, except that the third family (corresponding to context-sensitive languages) is missing. It may be noted that we could use the family of sets recognized by somehow restricted machines (say, machines M with $n_M = 1$) as the third family, but we feel that this is somewhat arbitrary.

In the diagram of Figure 2 an arrow means strict inclusion, an arrow with a question mark means inclusion (only strictness is open), and absence of a directed path means incomparability. An open but conjectured incomparability is marked with a dashed line.

Complementary families are connected with a dotted line. Actually, as the reader may verify by our previous theorems, the sets $F_{\text{poly}-d}$, F_{pal}, $\overline{F}_{\text{poly}-d}$ and $\overline{F}_{\text{pal}}$ suffice to show all known noninclusions in the diagram. (And it is known in addition that $F_{\text{poly}-d} \in \mathcal{F}(\text{2-CSM}) - \mathcal{F}(\text{OPDM})$.)

$$\begin{tikzcd}
 & \mathcal{F}(\text{GM}) & & \\
 & \mathcal{F}(\text{2-CPDM}) \arrow[u, "?"] & & \\
 & \mathcal{F}(\text{2-OPDM}) \arrow[u, "?"] & & \\
 & & & \mathcal{F}(\text{2-CSM}) \arrow[uull] \\
\mathcal{F}(\text{OPDM}) \arrow[uur] \arrow[rr, dotted, no head] \arrow[uurrr, dashed, no head] \arrow[drrr, dashed, no head] & & \mathcal{F}(\text{CPDM}) \arrow[uul] & \\
 & & & \mathcal{F}(\text{2-OSM}) \arrow[uu] \\
\mathcal{F}(\text{OSM}) \arrow[uu] \arrow[rr, dotted, no head] \arrow[urrr] & & \mathcal{F}(\text{CSM}) \arrow[uu] \arrow[uuur] &
\end{tikzcd}$$

Figure 2.

References

[1] AGARWAL, R.P. & LAKSHMIKANTHAM, V.: *Uniqueness and Nonuniqueness Criteria for Ordinary Differential Equations.* World Scientific (1993)

[2] ASARIN, E. & MALER, O.: On Some Relations Between Dynamical Systems and Transition Systems. In *Proceedings of ICALP '94* (S. Abiteboul & E. Shamir, Eds.). *Lecture Notes in Computer Science* **820**. Springer–Verlag (1994), 59–72

[3] BLONDELL, V.D. & TSITSIKLIS, J.N.: A Survey of Computational Complexity Results in Systems and Control. *Automatica* **36** (2000), 1249–1274

[4] BLUM, L. & CUCKER, F. & SHUB, M. & SMALE, S.: *Complexity and Real Computation.* Springer–Verlag (1998)

[5] BRANICKY, M.: Universal Computation and Other Capabilities of Hybrid and Continuous Dynamical Systems. *Theoretical Computer Science* **138** (1995), 67–100

[6] CHOMSKY, N.: Three Models for the Description of Language. *IRE Transactions on Information Theory* **2** (1956), 113–124

[7] CHOMSKY, N.: On Certain Properties of Grammars. *Information and Control* **2** (1959), 137–167

[8] CODDINGTON, E. & LEVINSON, N.: *Theory of Ordinary Differential Equations.* McGraw–Hill (1984)

[9] EL'SGOL'TS, L.E. & NORKIN, S.B.: *Introduction to the Theory and Application of Differential Equations with Deviating Arguments.* Academic Press (1973)

[10] HALE, J.K.: *Theory of Functional Differential Equations.* Springer–Verlag (1977)

[11] HARRISON, M.A.: *Introduction to Formal Language Theory.* Addison–Wesley (1978)

[12] HARTMAN, P.: *Ordinary Differential Equations.* Birkhäuser (1982)

[13] HOPCROFT, J.E. & MOTWANI, R. & ULLMAN, J.D.: *Introduction to Automata Theory, Languages and Computation.* Addison–Wesley (2001)

[14] HOPCROFT, J.E. & ULLMAN, J.D.: *Introduction to Automata Theory, Languages and Computation.* Addison–Wesley (1979)

[15] KALMAN, R.E. & FALB, P.L. & ARBIB, M.A.: *Topics in Mathematical System Theory.* Tata McGraw–Hill (1974)

[16] LINDGREN, K. & MOORE, C. & NORDAHL, M.: Complexity of Two-Dimensional Patterns. *Journal of Statistical Physics* **91** (1998), 909–951

[17] MARTIN, J.C.: *Introduction to Languages and the Theory of Computation.* McGraw–Hill (1996)

[18] MOORE, C.: Recursion Theory on the Reals and Continuous-Time Computation. *Theoretical Computer Science* **162** (1996), 23–44

[19] ORPONEN, P.: A Survey of Continuous-Time Computation Theory. In *Advances in Algorithms, Languages, and Complexity* (D.-Z. Du & K.-I. Ko, Eds.). Kluwer (1997), 209–224

[20] POUR-EL, M.B. & RICHARDS, I.: *Computability in Analysis and Physics.* Springer–Verlag (1989)

[21] ROXIN, E.O.: *Ordinary Differential Equations.* Wadsworth (1972)

[22] RUBEL, L.A.: The Extended Analog Computer. *Advances in Applied Mathematics* **9** (1993), 39–50

[23] RUOHONEN, K.: Undecidability of Event Detection for ODEs. *Journal of Information Processing and Cybernetics* **29** (1993), 101–113

[24] RUOHONEN, K.: An Effective Cauchy–Peano Existence Theorem for Unique Solutions. *International Journal on Foundations of Computer Science* **7** (1996), 151–160

[25] RUOHONEN, K.: Decidability and Complexity of Event Detection Problems for ODEs. *Complexity* **2** (1997) No. 6, 41–53

[26] SALOMAA, A.: *Formal Languages.* Academic Press (1973)

[27] SPRECHER, D.A.: A Universal Mapping for Kolmogorov's Superposition Theorem. *Neural Networks* **6** (1993), 1089–1094

[28] SPRECHER, D.A.: A Numerical Implementation of Kolmogorov's Superpositions. *Neural Networks* **9** (1996), 765–772

[29] SPRECHER, D.A.: A Numerical Implementation of Kolmogorov's Superpositions II. *Neural Networks* **10** (1997), 447–457