Feedback Can Reduce the Specification Complexity of Motor Programs

Magnus B. Egerstedt, *Member, IEEE,* and Roger W. Brockett, *Fellow, IEEE*

***Abstract*—In this paper, we show that when it is possible to use feedback in the specification of "motor programs," the length of the descriptions of the instruction sequences for carrying out a given task can be reduced by a factor that reflects the richness of the available feedback signals. The model on which this work is based is that of a finite automaton, modified in such a way that instruction processing is akin to the way in which difference or differential equations "process" piecewise constant inputs. In terms of such "free-running" automata, we show that when feedback is available the length of the shortest description can be reduced by a factor depending on the ratio of the size of the entire state space to the size of the set of states for which feedback is locally effective.**

***Index Terms*—Automata, complexity theory, feedback, motion control.**

I. Introduction

THE many visible and successful applications of feedback mechanisms at work testify to its effectiveness and over the years a variety of arguments have been advanced showing why, in particular settings, it is useful. The models commonly used bring to the fore considerations of sensitivity, uncertainty, etc. The existence of a variety of arguments should not be thought of as weakening the strength of any particular one but rather as a reflection of the multifaceted nature of feedback. Of course it would be desirable if the various arguments advanced for the use of feedback could be captured as special cases of an overarching argument, and a common element is the explicit or implied subdivision of the system into two parts, a forward path whose performance can only be characterized loosely and a feedback path whose behavior is known with greater certainty. Specific formalizations which start from this point include the following.

1) The Black argument for reducing the effect of drift in a high-gain amplifier by the use of a relatively constant, but low gain, feedback term [2].
2) The stochastic disturbance argument for using measurements to reduce the effect of probabilistic uncertainty. (See, for example, [8]).
3) The game theoretic argument in which a saddle point condition is enforced by feedback. (H_∞ control can be thought of in this way).

   To this list we now add a fourth item which we cast in terms of the effect feedback has on reducing the complexity of the implementation.
4) The complexity argument, proposed in this paper, showing that feedback can shorten motor programs if reliable sensory information is available.

In this paper, we consider the problem of describing a procedure that will assure that a system (robot, machine tool, etc.) will reach a certain goal state. We will give a quantitative analysis showing that the availability of feedback can reduce the length of the shortest description of such a procedure. In particular, Theorem 4.1 shows that the length of the description can be reduced by a factor that depends on the ratio of the size of the entire state space to the size of the set of states for which feedback is locally effective. This idea is made concrete by saying that a feedback control is locally effective if an observer-based feedback control can lead the system to the goal state without the need to supplement it with open-loop commands. In some natural settings, this argument can be repeated iteratively, leading to further reductions, as seen in Corollary 4.1.

We have found it necessary to use a few terms that are not (as yet) standard in the field of automatic control. Two in particular require clarification. The term "motor program" is taken from the field of biological motor control, where it is a standard idea referring to the mechanisms that support the definition and execution of commands that synchronize and coordinate the movement of muscle groups. These motor programs are sometimes described as being parameterized families, characterized by the speed and extent of the movements. (See, for example, [12] and [13]). From our point of view, a motor program is a symbolic string of instructions that specifies a particular movement, as suggested in [3], [4]. Second, rather than using differential equations as the starting point for our analysis we use a type of finite state models that we refer to as *free-running, feedback automata* (or FRF-automata for short). As will be shown, such models capture important aspects of time-invariant (but nonlinear) differential equations while retaining the "finitely describable" aspects of automata. Briefly stated, a FRF-automaton is a finite state machine that reads an input and then makes transitions repeatedly, using the same input value, until a particular output condition is realized. By measuring the length of the shortest description of an input sequences which accomplishes a given task, we get a complexity measure that allows us to compare open- and closed-loop control in a meaningful way.

Manuscript received May 15, 2001; revised June 18, 2002. Recommended by Associate Editor A. Bemporad. This work was supported in part by the U.S. Army Research Office under Grants DAAH 04 96 1 0445 and DAAG 55 97 1 0114, and by the National Science Foundation under Yale prime CCR 9980058 and the U.S. Army under Boston University prime GC169369 NGD.

M. B. Egerstedt is with the Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: magnus@ece.gatech.edu).

R. W. Brockett is with the Engineering and Applied Sciences, Harvard University, Cambridge, MA 02138 USA (e-mail: brockett@hrl.harvard.edu).

Digital Object Identifier 10.1109/TAC.2002.808466

II. The Model

A. Notation

The symbols that we use as inputs will be drawn from a finite set S, called the *input alphabet*, and finite strings of such input symbols are called *words*. We use $S^\star$ to denote the set of all such words, including the empty one. We let $s \in S$ denote an element in S, and use boldface $\mathbf{s} \in S^\star$ to denote elements in $S^\star$. If we define the associative operation of concatenation on $S^\star$, the empty word serves as an identity under this operation. Thus, $S^\star$ is the *free monoid* generated by S.

Now, consider the finite sets S, T, U, where $T \subset S$. We will, throughout this paper, let $S \backslash T$ denote the set $\{s \in S | s \notin T\}$, and let S^U denote the set of mappings from U to S. Given a subset $W \subset U$. If $\phi \in S^U$ then we let $\phi(W)$ denote the set $\{s \in S | s = \phi(w)$ for some $w \in W\}$. Finally, $\mathrm{card}(S)$ is the cardinality of S.

If we let X, V be finite sets, and let $\delta \in X^{X \times V}$, then we can identify (X, V, δ) with a *finite automaton* (see, for example, [1], [9], and [11]), whose operation is given by

$$x_{k+1} = \delta(x_k, v_k).$$

If we add another finite set Y and a mapping $\gamma \in Y^X$ to the definition, we get an *output* automaton $(X, Y, V, \delta, \gamma)$, where $x_{k+1} = \delta(x_k, v_k)$ and $y_k = \gamma(x_k)$.

Given a word $\mathbf{v} \in V^\star$, where $\mathbf{v} = v_1 \cdots v_p$, we use $\delta(x, \mathbf{v})$ as shorthand for

$$\delta(\delta(\cdots(\delta(x, v_1), v_2) \ldots, v_{p-1}), v_p)$$

and we let v^p denote the word obtained by concatenating v with itself $p-1$ times, i.e., $v^p = v \cdots v$.

B. Free-Running, Feedback Automata

We now introduce a dynamical system called a *free-running automaton*. The idea is to let such an automaton read an input from a given alphabet, and then advance the state of the automaton repeatedly without reading any new inputs until an interrupt is triggered.

Definition 2.1 (Free-Running Automaton): Let X, Y, V be finite sets, let $\mathcal{T} \subset \{0,1\}^Y$, and let $U = V \times \mathcal{T}$. Let furthermore $\delta : X \times V \to X$ and $\gamma : X \to Y$ be given. We say that $(X, U, Y, \delta, \gamma)$ defines a free-running automaton with the understanding that input strings from $U^\star$ cause x and l to evolve according to the rule

$$\begin{aligned} x_{k+1} &= \delta(x_k, v_{l_k}) \quad y_k = \gamma(x_k) \\ l_{k+1} &= l_k + \tau_{l_k}(y_k). \end{aligned}$$

A free-running automaton thus operates on a given input symbol v repeatedly until the interrupt is triggered, i.e., when $\tau(y)$ changes value from 0 to 1, and a new input pair (v', τ') is read.[1] Note the similarity between the triggered-based hybrid systems defined in [4] and Definition 2.1, where l takes on the role as a counter that marks the progression along the input string.

[1]Although not essential to the development in this paper, in Appendix A it is shown that the language recognized by a free-running automaton can always be recognized by a finite automaton as well.

Since we are interested in formalizing how feedback control affects the evolution of the automata, we need to allow the input sets to have additional structure, e.g., $A \times B$ or $\{0,1\}^P$. When the input alphabet is a set of the form $V \times K$, with K being a subset of $V^{Y \times V}$, we can interpret an input letter as providing a pair (v, κ), with $v \in V$ being an open-loop signal and $\kappa : Y \times V \to V$ being a feedback control law. Given this special structure, transitions are generated according to the rule

$$\delta(x, (v, \kappa)) = \delta(x, \kappa(\gamma(x), v)).$$

If the input alphabet W admits this structure, and δ takes this form, then we will say that a finite automaton (X, W, δ) admits the structure of a *variable feedback automaton*,[2] or feedback automaton for short.

We now combine the ideas of free-running and feedback automata to get our primary object of study: the *free-running, feedback automaton* (FRF-automaton). It is a free-running automaton whose input alphabet admits the structure $\Sigma = V \times K \times \mathcal{T}$, where V is a finite set, $K \subset V^{Y \times V}$, and $\mathcal{T} \subset \{0,1\}^Y$. Thus, the input to a FRF-automaton is a triple (v, κ, τ), where $v \in V$ (open-loop control), $\kappa : Y \times V \to V$ (closed-loop control), and $\tau : Y \to \{0,1\}$ (interrupt).

Definition 2.2 (Free-Running, Feedback Automaton): Let $X, Y, V, \mathcal{T}, \delta, \gamma$ be as in Definition 2.1 and let $\Sigma = V \times K \times \mathcal{T}$, where $K \subset V^{Y \times V}$. We say that $(X, \Sigma, Y, \delta, \gamma)$ defines a free-running, feedback automaton with the understanding that input strings from $\Sigma^\star$ cause x and l to evolve according to the rule

$$\begin{aligned} x_{k+1} &= \delta(x_k, \kappa_{l_k}(\gamma(x_k), v_{l_k})) \quad y_k = \gamma(x_k) \\ l_{k+1} &= l_k + \tau_{l_k}(y_k). \end{aligned}$$

The interpretation here is that the FRF-automaton operates on the pair (v, κ) repeatedly, as a feedback automaton, until the interrupt $\tau(y)$ changes from 0 to 1, in which case a new input triple is read.

Two observations about the FRF-automata can be made already at this point: Consider the FRF-automaton $(X, Y, \Sigma, \delta, \gamma)$, where $\Sigma = V \times V^{Y \times V} \times \{0,1\}^Y$, and the finite automaton (X, V, δ). For every input sequence $w_1 \cdots w_s \in V^\star$ that drives the finite automaton through the states $x_1, \ldots, x_{s+1}$, the same sequence can be visited by the FRF-automaton by simply letting (v_i, κ_i, τ_i), $i = 1, \ldots, s$ be

$$\begin{cases} v_i & \text{arbitrary} \\ \kappa_i(\gamma(x), v) = w_i & \forall x \in X, v \in V \\ \tau_i(y) = 1 & \forall y \in Y. \end{cases}$$

This input string simply leads the FRF-automaton along the same path using the same open-loop instructions. Note, however, that by Definition 2.2, the FRF-automaton takes an additional step after x_{s+1} has been reached. If x_{s+1} is thought of as a terminal state, x_f, this can be remedied by letting $\delta(x_f, v) = x_f, \forall\, v \in V$, which will be implicitly assumed throughout this paper.

[2]Our notion of a feedback automaton is quite different from that in [15]. There the output is fed directly to the automaton as the next input. In contrast to this, we let the feedback mappings be explicitly specified as inputs to the automaton.

The other key observation is that if $Y = X$, $\gamma(x) = x, \forall\, x \in X$, then the sequence of states $x_1, \ldots, x_{s+1}$ can be traversed by the FRF-automaton using only *one* input provided that no state appears twice in the sequence. The input (v, κ, τ) that realizes this is

$$\begin{cases} v & \text{arbitrary} \\ \kappa(x_i, u) = w_i & \forall u \in V,\ i = 1, \ldots, s \\ \tau(x) = 1 & \text{if and only if } x = x_{s+1}. \end{cases}$$

This input leads the FRF-automaton along the same path, but since each state is visited at most once, a state feedback policy can achieve this in a straight forward manner.

This last observation indicates that the free-running property of the FRF-automata implies that they can, in general, be guided along a path using fewer instructions than the classical finite automata. However, since the input set to a finite automaton is the finite set V, while the input set to the corresponding FRF-automaton is of the form $V \times K \times \mathcal{T}$, where $K \subset V^{Y \times V}, \mathcal{T} \subset \{0,1\}^Y$, the input set has a higher cardinality in the latter of these cases. Any reasonable measure of the complexity of a control procedure must take the size of the input space into account since the number of bits required to code a word over a given alphabet typically depends logarithmically on the size of the alphabet. (See, for example, [7]). This dependency is captured in a natural way if we define the complexity of a control procedure as the description length of the input sequence, i.e., as the number of bits needed for specifying the strategy.

C. Specification Complexity

Definition 2.3 (Description Length): Consider a finite set S. We say that a word $\mathbf{s} \in S^\star$ has description length

$$\mathcal{L}(\mathbf{s}, S) = |\mathbf{s}| \log_2(\mathrm{card}(S)).$$

Definition 2.4 (Specification Complexity): Consider a FRF-automaton, A, with state–space X and input set Σ. Let $\boldsymbol{\sigma}$ be the word of minimal description length over Σ that drives the automaton from x_0 to x_f. We then say that the task of driving A between x_0 and x_f has specification complexity $\mathcal{C}(A, x_0, x_f) = \mathcal{L}(\boldsymbol{\sigma}, \Sigma)$.

D. Example

As an example, we explore the applicability of the FRF-automaton model in the context of a particular travel direction generating program called *MapQuest* [14]. This program generates directions in "people readable form" for traveling by car between two addresses.

Our first problem when interpreting the instructions provided by this program is that of deciding how to split a given instruction into an open-loop and a closed-loop part. An instruction such as `Turn RIGHT onto QUINCY ST`[3] is, in fact, shorthand for the composition of the instructions "Drive until you see a sign indicating Quincy Street," followed by the open-loop instruction "Turn right." If we associate each intersection with a state in a FRF-automaton, there are three open-loop instructions that can be read by the automaton, namely `Left`, `Right`, and `Straight`. Hence, the finite set V in the input alphabet $\Sigma = V \times K \times \mathcal{T}$ is

$$V = \{L, R, S\}.$$

[3]All the examples in this section are taken directly from *MapQuest*.

Now, the output associated with each state corresponds to the observation of a street sign. Thus, it seems plausible to take the output Y to be the set of streets in the town, state, or country of interest. If we let the number of such signs be N, then the description length of the instruction Turn RIGHT onto QUINCY ST thus becomes $\log_2(\mathrm{card}(V)\mathrm{card}(K)) = \log_2(3 \cdot 3^{3N})$, or, if we include the contribution from the interrupt in the calculation, $\log_2(3 \cdot 3^{3N} \cdot 2^N)$, since the interrupt is a mapping $\tau : Y \to \{0,1\}$.

What about the instruction `Drive for 11.6 km and take the exit heading NORTH`? When thinking about the difference between open-loop and closed-loop control, a sharp distinction is usually made between time and all other variables. Functions of time are said to be open-loop, which is standard in the control field, because time, or more precisely time relative to some initial time, is assumed to be universally available. However, in some situations it is also natural to think that relative or even absolute positions might be available. The same can be said for temperature, air pressure, humidity, the Dow Jones average, and the conversion between the Yen and the dollar. It can be useful to refine the model proposed in this paper in such a way that an instruction is declared to be open-loop relative to a list of such "universally" available variables. In this way, we can extend the special status of time to other variables.

If we have an odometer we can think of the instruction above as an open-loop instruction by declaring the universally available variables to include relative distance. This instruction would then correspond to a series of `Straight` commands that direct the traveler through the encountered intersections, followed by a turn command.

If we now formalize these observations, the FRF-automaton that we use for interpreting and executing instructions from MapQuest is $(X, Y, \Sigma, \delta, \gamma)$, where

$$\begin{aligned} X &= \{\text{intersections}\} \\ Y &= \{\text{street signs}\} \\ \Sigma &= V \times V^{Y \times V} \times \{0,1\}^Y, \text{ where } V = \{L, R, S\} \\ \delta : X \times V &\to X \text{ gives the next intersection encountered} \\ \gamma : X &\to Y. \end{aligned}$$

An example of the type of instructions that MapQuest provides is listed later.[4] There, the instructions and the corresponding inputs to the FRF-automaton are shown. Open-loop inputs are described using values in $V = \{L, R, S\}$, and the closed-loop inputs are represented as mappings from street signs to the set V.

1) Start out going SOUTH on OXFORD STR: R.
2) Turn RIGHT onto BEACON ST: $\mathrm{BEACON} \to R$.
3) Turn LEFT onto BROADWAY: $\mathrm{BROADWAY} \to L$.
4) Turn RIGHT onto FULKERSON ST: $\mathrm{FULKERSON} \to R$.
5) Turn LEFT onto MA-2A: $\mathrm{MA-2A} \to L$.
6) Turn LEFT onto MELNEA CASS BLVD: $\mathrm{MELNEA} \to L$.
7) Turn RIGHT after 0.3 km: S, R .

[4]The example describes how to drive from the Maxwell–Dworkin building at Harvard University, Cambridge, MA, to the Applied Mathematics building at Brown University, Providence, RI.

8) Turn RIGHT onto FRONTAGE RD: $\mathrm{FRONTAGE} \rightarrow R$.
9) Take the I-93 ramp SOUTH: R.
10) Stay on I-93 (22.7 km): S, S, S, S.
11) Take I-95S exit: $I-95S \rightarrow R$.
12) Stay on I-95 (49.7 km): S, S, S, S, S, S, S.
13) Take BRANCH AVE exit: $\mathrm{BRANCH} \rightarrow R$.
14) Keep LEFT in ramp: L.
15) Turn RIGHT: R.
16) Turn RIGHT onto NMAIN: $\mathrm{NMAIN} \rightarrow R$.
17) Stay STRAIGHT: S.
18) Turn LEFT onto OLNEY ST: $\mathrm{OLNEY} \rightarrow L$.
19) Turn RIGHT onto HOPE ST: $\mathrm{HOPE} \rightarrow R$.
20) Turn RIGHT onto GEORGE ST: $\mathrm{GEORGE} \rightarrow R$.

Thus the FRF-automaton model is rich enough to capture this real-life use of open-loop and closed-loop instructions. We will return to this example to illustrate points and to provide motivation later in this paper.

III. Observer Automata

In this section, we elaborate further on the properties of the FRF-automata. For the reminder of the paper, let the FRF-automaton be given by $(X, Y, \Sigma, \delta, \gamma)$, where $\Sigma = V \times V^{Y \times V} \times \{0,1\}^Y$, unless stated otherwise.

A. Observability

Consider the finite automaton $(X, Y, V, \delta, \gamma)$. We define the *output sequence map* $\mathcal{O} : \mathbb{Z}^+ \times X \times V^Y \rightarrow Y^\star$ as the string of outputs

$$\mathcal{O}(p, x, w) = \gamma(x_1) \cdot \gamma(x_2) \cdots \gamma(x_p)$$

where $w : Y \rightarrow V$, and $x_1 = x$, $x_2 = \delta(x_1, w(\gamma(x_1))), \ldots, x_p = \delta(x_{p-1}, w(\gamma(x_{p-1})))$. Here, $y_1 \cdot y_2$ denotes the concatenation of the letters y_1 and y_2 from the finite alphabet Y, and $\mathcal{O}(p, x, w) \in Y^p \subset Y^\star$, where Y^p is the set of words of length p over Y.

What we want to do is to characterize when output feedback is effective. We know that if we can construct an observer, i.e., reconstruct the state of the system, then feedback would be of use. The state of the system can be reconstructed if, for some given feedback policy, the output sequence map is injective in its second argument, i.e., the system produces different output strings for different initial states.

Definition 3.1 (Observability): A finite automaton $(X, Y, V, \delta, \gamma)$ is observable if there exist a positive integer p_{obs} and an output-to-input mapping $w_{obs} : Y \rightarrow V$ that satisfies

$$\mathcal{O}(p_{obs}, x_1, w_{obs}) \neq \mathcal{O}(p_{obs}, x_2, w_{obs})$$

for all $x_1, x_2 \in X,\ x_1 \neq x_2$.[5]

We also need to define observer automata for recovering the states of the original systems in order for observer-based, output feedback to be useful:

[5]This definition of observability is somewhat different from the definitions encountered in the literature on discrete event systems. (See, for example, [10].). There, it is assumed that certain events can be detected, while others are nondetectable. The state space is thus partitioned into one trivially observable and one unobservable part.

Definition 3.2 (Observer Automaton): Consider the observable finite automaton $A = (X, Y, V, \delta, \gamma)$. Let Z, O be finite sets, $\Omega = V \times V^{O \times V}$, $g : Z \times Y \times \Omega \rightarrow Z$, and $h : Z \times Y \rightarrow O$. Then, (Z, O, Ω, g, h) is said to be an observer to A if there exists a $\omega = (v, w) \in \Omega$ such that

$$\begin{aligned} x_{k+1} &= \delta(x_k, w(o_k, v)) \quad y_k = \gamma(x_k) \\ z_{k+1} &= g(z_k, y_k, w(o_k, v)) \quad o_k = h(z_k, y_k) \end{aligned}$$

implies that the current state of X can be uniquely determined from the current state of Z, provided that sufficiently many iterations have been made. We say that the number of iterations necessary for achieving this is the settling time of the observer.

We now defend our choice of language by showing that we can associate an observer automaton to any observable automaton. The constructions are as follows: Let p_{obs} be the positive integer in Definition 3.1, and let

$$Z = \{e\} \cup Y \cup Y^2 \cup \cdots \cup Y^{p_{obs}-1} \cup X$$

where e is any symbol distinguishable from the rest of the states in Z.

Now, consider $w_{obs} : Y \rightarrow V$ from Definition 3.1, and define the mapping $\Xi : Y^{p_{obs}} \times V \rightarrow X$ as follows: For any given $x_1 \in X$ and $v \in V$, we let

$$\Xi(\mathcal{O}(p_{obs}, x_1, w_{obs}), v) = \delta(x_{p_{obs}}, v)$$

where $x_{p_{obs}} = \delta(x_{p_{obs}-1}, w_{obs}(\gamma(x_{p_{obs}-1}))), \ldots, x_2 = \delta(x_1, w_{obs}(\gamma(x_1)))$. For every $\mathbf{y} \in Y^{p_{obs}}$ that does not satisfy $\mathbf{y} = \mathcal{O}(p_{obs}, x, w_{obs})$ for some $x \in X$, we let $\Xi(\mathbf{y}, v)$ be assigned an arbitrary value. Ξ is thus to be thought of as a mapping that reconstructs the current state of the original automaton using the injective property of the output sequence map, and evolving that state one step further using the feedback policy w_{obs}.

Now, let $h : Z \times Y \rightarrow O$ be given by

$$h(z, y) = \begin{cases} y & \text{if } z \notin X \\ z & \text{otherwise} \end{cases}$$

and we thus have $O = Y \cup X$. This is an important fact since we have already seen that the cardinality of the input set appears in the definition of the specification complexity. If we were to use observer-based feedback control, the size of the output set of the observer would affect the input set to the system. Our current construction thus gives that $\mathrm{card}(O) = \mathrm{card}(X) + \mathrm{card}(Y)$. However, it is not yet clear that this construction does in fact produce an observer automaton since the evolution of the observer automaton, i.e., g, is not yet specified.

Let $z_0 = e$, and let $g : Z \times Y \times \Omega \rightarrow Z$ be given by the equation shown at the bottom of the next page.

Given $w_{obs} : Y \rightarrow V$ from Definition 3.1 we define $\bar{w}_{obs} : O \times V \rightarrow V$ as $\bar{w}_{obs}(o, v) = w_{obs}(o)$ if $o \in Y$, and $w_{obs}(\gamma(o))$ if $o \in X$. If we apply this input, together with any $v \in V$, to both the original automaton and the observer automaton it is straight forward to check that the observer automaton has settling time p_{obs}. We call this observer automaton the *standard observer automaton* since it simply uses p_{obs} steps to let the observer settle, and then copies the operation of the original automaton by using the injectivity of the output sequence map.

B. Attractors

We now characterize the situations under which a desired, final state is in fact reachable. To do so we derive conditions under which a desired state can be reached using observer-based feedback control. For this, we need the notion of an attractor. We say that $x_f \in X$ is a *global attractor* for the difference equation $x_{k+1} = \lambda(x_k)$ if, for all $x_0 \in X$, it holds that

$$\lim_{k\to\infty} x_k = x_f.$$

In other words, x_f is a global attractor if it is reachable from every point $x \in X$ and the system remains at x_f once it has reached x_f.[6]

Theorem 3.1 (Creating Global Attractors): Consider the finite automaton (X, V, δ). If $x_f \in X$ can be reached from every initial state, and there is a $v_f \in V$ such that $\delta(x_f, v_f) = x_f$, then there is a mapping $w_{attr} : X \to V$ such that

$$x_{k+1} = \delta(x_k, w_{attr}(x_k))$$

has x_f as a global attractor.

Proof: Choose an arbitrary $x_1 \in X$. Let $\mathbf{v}_1$ denote a (not necessarily unique) shortest input sequence that drives the automaton from x_1 to x_f. Decompose $\mathbf{v}_1$ as $\mathbf{v}_1 = v_1 \cdot \hat{\mathbf{v}}_1$, where $v_1 \in V$ and $\hat{\mathbf{v}}_1 \in V^\star$. Let the candidate for the controller that makes x_f a global attractor satisfy $w_{attr}(x_1) = v_1$.

Now, let x_2 denote the state $\delta(x_1, v_1)$ and repeat the argument until the automaton reaches x_f. By letting the initial state vary over all of X, a control that drives the automaton between an arbitrary initial state and x_f is obtained. Furthermore, let $w_{attr}(x_f) = v_f$, which implies that x_f is a global attractor. ■

This theorem is useful since it allows us to use state feedback for driving FRF-automata to desired states, and we state this fact as a corollary:

Corollary 3.1: Given the FRF-automaton $(X, Y, \Sigma, \delta, \gamma)$, where $\Sigma = V \times V^{Y\times V} \times \{0,1\}^Y$, $Y = X$, and $\gamma(x) = x$, $\forall x \in X$. This automaton can reach any given state $x_f \in X$ using only one instruction if x_f can be reached from any initial state under the operation of the finite automaton (X, V, δ).

Proof: Choose the input (v, κ, τ) as

$$\begin{cases} v & \text{arbitrary} \\ \kappa(x, w) = w_{attr}(x) & \text{as in Theorem 3.1 } \forall x \in X, w \in V \\ \tau(x) = 1 & \text{if and only if } x = x_f. \end{cases}$$

This input drives the automaton to x_f, and the corollary follows. ■

[6]This implies that if x_f is a global attractor, then there exists a finite, positive integer N such that $x_{N+p} = x_f, \forall\, p \in \mathbb{Z}^+$.

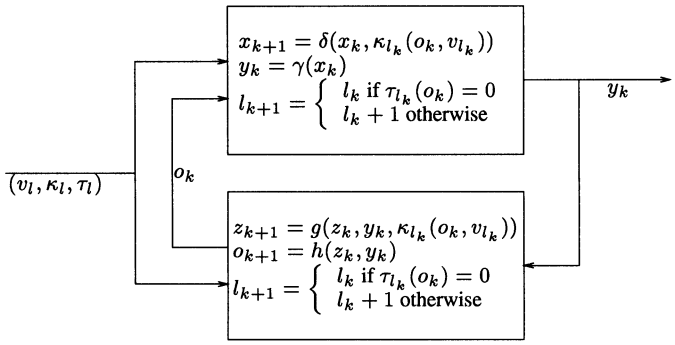

Fig. 1. FRF-automaton together with an observer automaton.

We can now combine this result with the notion of observability in order to get a characterization of when observer-based closed-loop control is useful.

Lemma 3.1 (Observers Make Single Instruction Goal Achievement Possible): Let the observable finite automaton $(X, Y, V, \delta, \gamma)$ be such that x_f can be reached from any initial state. Then, by using the standard observer automaton, it is possible to drive the state of the FRF-automaton $(X, Y, \Sigma_O, \delta, \gamma)$, where $\Sigma_O = V \times V^{O\times V} \times \{0,1\}^O$, between any initial state and x_f using only one instruction.

Proof: Construct the standard observer automaton. (See Fig. 1). Pick $w_{obs} : Y \to V$ as in Definition 3.1, and choose the input sequence to the FRF-automaton as

$$\begin{cases} v = & \text{arbitrary} \\ \kappa(o, v) = w_{obs}(o), & \text{if } o \in Y \\ \kappa(o, v) = w_{attr}(o), & \text{if } o \in X \\ \tau(o) = 1 & \text{if and only if } o = x_f \end{cases}$$

where $w_{attr}(x)$ is defined in Theorem 3.1.

By using this input, the FRF-automaton traverses its states until the observer has converged, i.e., k advances p_{obs} steps. Then, it drives its state to x_f as in Corollary 3.1, which concludes the proof. ■

C. Observable Subsets

Recall that the specification complexity is proportional to the logarithm of the cardinality of the input space. By using observer-based feedback we see that this complexity that depends directly on the size of output set associated with the observer automaton. In the standard observer construction we saw that this set had cardinality $\mathrm{card}(X) + \mathrm{card}(Y)$. In this section we investigate if it is possible to reduce the size of this set by only defining the observer locally, i.e., on a subset of the state space.

Definition 3.3 (Observable Subset): Consider the finite automaton $(X, Y, V, \delta, \gamma)$. A subset $X_g \subset X$ such that $\gamma(X_g) \cap \gamma(X \backslash X_g) = \emptyset$ is said to be observable if there exist a positive

$$z_{k+1} = g(z_k, y_k, w_k(h(z_k, y_k), v_k)) = \begin{cases} y_k & \text{if } z_k = e \\ z_k \cdot y_k & \text{if } z_k \in Y^q,\ q < p_{obs} - 1 \\ \Xi(z_k \cdot y_k, w_k(h(z_k, y_k), v_k)) & \text{if } z_k \in Y^{p_{obs}-1} \\ \delta(z_k, v_k) & \text{if } z_k \in X. \end{cases}$$

integer p_{obs} and a $w_{obs} : Y \to V$ that satisfies the following conditions:

- $\mathcal{O}(p_{obs}, x_1, w_{obs}) \neq \mathcal{O}(p_{obs}, x_2, w_{obs})$, $\forall x_1, x_2 \in X_g, x_1 \neq x_2$;
- for all $x_1 \in X_g$ it follows that $x_q \in X_g$, $q = 1, \ldots, p_{obs}$, where $x_2 = \delta(x_1, w_{obs}(\gamma(x_1)))$, $x_3 = \delta(x_2, w_{obs}(\gamma(x_2))), \ldots$.

What this means is simply that a subset, X_g, is an observable subset to a given automaton if the state can be recovered on X_g without the state leaving X_g.

In a manner analogous to the previously defined observer automata, we can now define observer automata on subsets as well.

Definition 3.4 (Subset-Observer Automaton): Consider the finite automaton $A = (X, Y, V, \delta, \gamma)$, where $X_g \subset X$ is an observable subset. (Z, O, Ω, g, h), where Z, O are finite sets, $\Omega = V \times V^{O \times V}$, $g : Z \times Y \times \Omega \to Z$, and $h : Z \times Y \to O$ is a subset-observer automaton to A if there exists a $\omega = (v, w) \in \Omega$ such that the following conditions hold:

$$x_{k+1} = \delta(x_k, w(o_k, v)) \quad y_k = \gamma(x_k)$$
$$z_{k+1} = g(z_k, y_k, w(o_k, v)) \quad o_k = h(z_k, y_k)$$

gives that the current state in Z can be mapped uniquely to the current state in X after sufficiently many iterations. Also, for all $x_1 \in X_g$ it holds that $x_q \in X_g$, $q = 1, \ldots, p_{obs}$, where $x_2 = \delta(x_1, w(o_1, v))$, $x_3 = \delta(x_2, w(o_2, v))$, and so on.

Lemma 3.2 (Subset-Observers Exist): Let $X_g \subset X$ be an observable subset to the finite automaton $A = (X, Y, V, \delta, \gamma)$. Then, a subset-observer automaton (Z, O, Ω, g, h) to A can always be constructed.

Proof: Let Z, Ω be given by the standard observer automaton. Let $O = \{e\} \cup \gamma(X_g) \cup X_g$, and let

$$h(z, y) = \begin{cases} y, & \text{if } z \notin X \cup \{e\} \\ z, & \text{otherwise.} \end{cases}$$

Furthermore, let $g(z_k, y_k, w_k(h(z_k, y_k), v_k))$ be defined by the equation shown at the bottom of the page, where the mapping $\Xi(\cdot)$ is defined in the standard observer automaton.

If we now use $\bar{w}_{obs}(o, v) = v$ if $o = e$, $\bar{w}_{obs}(o, v) = w_{obs}(o)$ if $o \in \gamma(X_g)$, and $\bar{w}_{obs}(o, v) = w_{obs}(\gamma(o))$ if $o \in X_g$, then a repetition of the argument in the derivation of the standard observer automaton shows that it is in fact a subset-observer automaton, with settling time p_{obs}. ■

It should be noted that the subset-observer's output set has a lower cardinality than the standard observer automaton as long as

$$\text{card}(X_g) < \text{card}(X) - 1.$$

This observation will be used in Section IV where the main complexity theorems are presented. Since the cardinality of the input set depends on the size of the domain of the feedback mapping, a smaller domain can be expected to reduce the specification complexity. Thus, by using open-loop control on $X \backslash X_g$, and using observer-based feedback on X_g, we can expect a decrease in specification complexity. In order to make this observation quantitative, we introduce the notions of ballistic reachability and control-invariant reachability: A set $X_s \subset X$ is *ballistically reachable from* x if there exists a $v \in V$ such that $\delta(x, v^q) \in X_s$ for some $q \in \mathbb{Z}^+$. What this means is that X_s is ballistically reachable from x if it is possible to drive the state of the automaton from x to X_s using one open-loop input repeatedly until the trajectory reaches X_s. Furthermore, X_s is ballistically reachable from $X_t \subset X$ if there exists a $v \in V$ such that for all $x \in X_t$ it holds that $\delta(x, v^{q(x)}) \in X_s$ for some $q(x) \in \mathbb{Z}^+$. An element $x \in X_s \subset X$ is said to be *control-invariantly reachable in* X_s if it can be reached from all states in X_s without the trajectory leaving X_s. These concepts are illustrated in Fig. 2.

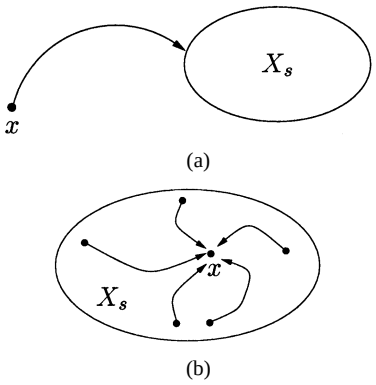

Fig. 2. (a) X_s is ballistically reachable from x. (b) x is control-invariantly reachable in X_s.

Lemma 3.3 (One Instruction Suffices When Using Subset-Observers): Let X_f be an observable subset to the finite automaton $(X, Y, V, \delta, \gamma)$, and let $x_0 \notin X_f$, $x_f \in X_f$. If X_f is ballistically reachable from x_0, and x_f is control-invariantly reachable in X_f, then there exists a FRF-automaton $(X, Y, \Sigma', \delta, \gamma)$ that can reach x_f from x_0 using only one instruction.

Proof: Construct the subset-observer from Lemma 3.2 and let $v_{ol} \in V$ be an open-loop control that drives the automaton from x_0 to X_f. (The existence of such a control follows since X_f is ballistically reachable from x_0). Let the input to the FRF-automaton, $\sigma = (v, \kappa, \tau)$, $v \in V$, $\kappa : O \times V \to V$, $\tau : O \to \{0, 1\}$ be given by

$$\begin{cases} v = v_{ol} & \\ \kappa(o, v) = v, & \text{if } o = e \\ \kappa(o, v) = w_{obs}(o), & \text{if } o \in \gamma(X_f) \\ \kappa(o, v) = w_{attr}(o), & \text{if } o \in X_f \\ \tau(o) = 1, & \text{if and only if } o = x_f. \end{cases}$$

$$(z_k, y_k, w_k(h(z_k, y_k), v_k)) = \begin{cases} e & \text{if } y_k \notin \gamma(X_g) \\ y_k & \text{if } z_k = e \text{ and } y_k \in \gamma(X_g) \\ z_k \cdot y_k & \text{if } z_k \in \gamma(X_g)^q,\ q < p_{obs} - 1 \\ \Xi(z_k \cdot y_k, w_k(h(z_k, y_k), v_k)) & \text{if } z_k \in \gamma(X_g)^{p_{obs}-1} \\ \delta(z_k, v_k) & \text{if } z_k \in X_g \end{cases}$$

In other words, $\Sigma' = \{v_{ol}\} \times V^O \times \{0,1\}^O$, and the previous input drives the state of the automaton from x_0 to X_f using the open-loop input v_{ol}. It then executes the observer-based motion from Lemma 3.1 on the subset X_f. ■

D. Example, Continued

Thus far, we have produced an FRF-automaton that captures important aspects of the way travel directions are given and processed when driving between different locations. Given a list of streets that we can expect to encounter during a particular journey, we can now construct a subset-observer for reducing the size of the input alphabet by only considering relevant streets. By those we understand the streets encountered during the trip where a left or right turn is called for. The observations corresponding to all other streets are, as in Lemma 3.2, denoted by the single symbol e in the subset-observer. The new input alphabet thus becomes $\Sigma' = V \times V^O \times \{0,1\}^O$, where $O = \{e, \text{relevant streets}\}$. In the example in Section II-D, O (with $\text{card}(O) = 15$) becomes

$$\{e, \text{OXFORD}, \text{BEACON}, \text{BROADWAY}, \text{FULKERSON}$$
$$\text{MA-2A}, \text{MELNEA}, \text{FRONTAGE}, \text{I-93}, \text{I-95}, \text{BRANCH}$$
$$\text{MAIN}, \text{OLNEY}, \text{HOPE}, \text{GEORGE}\}.$$

As illustrated in Fig. 3, single instructions suffice for driving the automaton to a desired state if an input can be constructed that generates a ballistic, open-loop movement that traverses a large part of the state–space, followed by an observer-based, closed-loop movement. It is interesting to investigate whether the instructions provided by MapQuest have a similar structure. We examine this in a probabilistic setting, and for this we need to estimate the number of bits of information that comes from the open-loop and the closed-loop part, respectively. To this end, it seems reasonable to adopt the "choice complexity" model. In this model, the number of bits associated with the open-loop command "turn left" is $\log_2(3)$, since $V = \{L, R, S\}$, and the number of bits associated with the observation of the street sign for Quincy street is $\log_2(\text{card}(O))$, where O is as defined above. A statement like `Turn LEFT onto BROADWAY` in the previous example would then have a ratio between the "closed-loop bits" and the total number of bits as $\log_2(15)/\log_2(3 \cdot 15) \approx 0.7$. We denote this ratio by r_{cl}.

We now let d_s denote the distance traveled from the starting address, and let d_f denote the distance remaining to travel to the final address. We can then define d as

$$d = \min\{d_s, d_f\}$$

and store each instruction as the data pair (d, r_{cl}). We consider an instruction to be closed-loop if $r_{cl} \geq 0.5$ and open-loop if $r_{cl} < 0.5$, thus generating a threshold based partition of the instructions into two types.

We will now generate some statistical results based on an analysis of a sample of the directions when driving between the Maxwell–Dworkin building at Harvard University to 20 other universities around the U.S. For this, we fit empirical probability densities to the two sets of instruction types, i.e., to the set of open-loop and closed-loop instructions respectively, as functions of d. After extracting and classifying the data from the 20 travel directions (totaling 724 instructions) we choose to fit Gaussian probability densities to these two collections of data points

$$p(d|\text{open-loop}) = \frac{1}{\sqrt{2\pi}\hat{\sigma}_{ol}} e^{-(d-\hat{\mu}_{ol})^2/(2\hat{\sigma}_{ol}^2)}$$
$$p(d|\text{closed-loop}) = \frac{1}{\sqrt{2\pi}\hat{\sigma}_{cl}} e^{-(d-\hat{\mu}_{cl})^2/(2\hat{\sigma}_{cl}^2)}$$

as seen in Fig. 4. In that figure, the sample means and covariances were found to be

$$\hat{\mu}_{ol} = 38.9\hat{\sigma}_{ol} = 32.1$$
$$\hat{\mu}_{cl} = 10.7\hat{\sigma}_{cl} = 23.0$$

where the subscripts ol and cl denote open-loop and closed-loop, respectively.

We now turn our attention to Bayes' decision rule as a mean of probabilistically classifying the instructions as open-loop or closed-loop based on the variable d. The reason for doing this is that we wish to get some feeling for when open-loop and closed-loop instructions are most likely to be used. This result is then to be used as guidance when we produce instructions with short description lengths.

If we let $P(\cdot)$ denote a probability distribution and $p(\cdot)$ denote a probability density, we can classify an instruction as open-loop or closed-loop using Bayes' decision rule, i.e., by choosing the classification with the greatest conditional

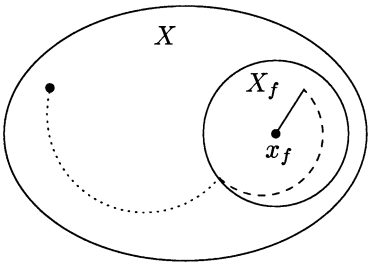

Fig. 3. Observer-based, single instruction, goal-finding procedure in Lemma 3.3. The dotted line is the open-loop part of the evolution, the dashed line defines the part where the observer is converging, and the solid line is the last part of the evolution.

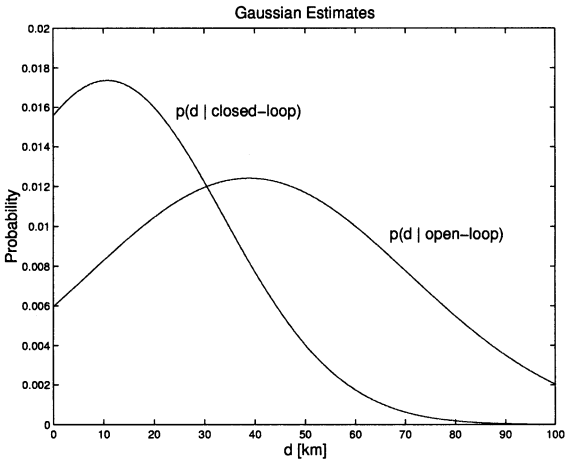

Fig. 4. Fitted Gaussian densities associated with the closed-loop ($r_{cl} \geq 0.5$) and the open-loop ($r_{cl} < 0.5$) instructions are shown as functions of the distance to the closest endpoint of the trip.

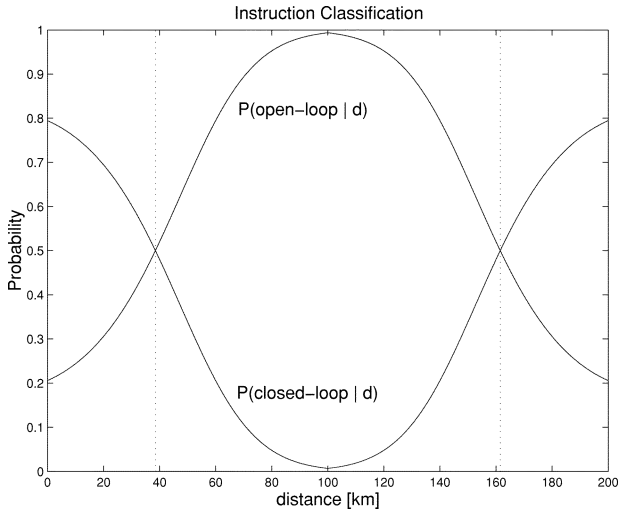

Fig. 5. Conditional probabilities that indicate how likely it is that a given instruction is closed-loop or open-loop are shown. We have assumed (supported by the empirical data) that the probabilities are symmetrical around the start and goal point. From the figure it can be seen that Bayes' decision boundary is located at $d = 38.5$ km.

probability $P(\omega|d) = p(d|\omega)P(\omega)/p(d)$, where $\omega \in \Omega = \{\text{open-loop}, \text{closed-loop}\}$. The priors are

$$P(\text{closed-loop}) = \frac{\text{total number of closed-loop instructions}}{\text{total number of instructions}} = \frac{290}{724} = 0.401$$

$$P(\text{open-loop}) = \frac{\text{total number of open-loop instructions}}{\text{total number of instructions}} = \frac{434}{724} = 0.599$$

and the probability densities $p(d|\text{open-loop})$ and $p(d|\text{closed-loop})$ are given in Fig. 5. The conditional probabilities are plotted in Fig. 5, and it can be seen that the Bayes' decision boundary occurs at $d = 38.5$ km. What this means is that close to the goal and to the starting point, closed-loop instructions are more likely, while open-loop instructions are likely far away from those points. In Section IV, we derive a suite of complexity theorems that capture this effect in a natural way, based on Lemma 3.3.

IV. Instructions Which Lead to the Goal

The reason for studying the situation in Lemma 3.3 is that it captures the idea that it is possible to successfully combine uncertain feed-forward control and high-precision feedback control on different parts of the state–space. Since the size of the input set is dependent on the size of the output space of the observer automaton when feedback is used, the description lengths of the inputs should be reduced if we only use feedback where it is locally effective, i.e., on reduced parts of the state–space.

However, in order to compare purely open-loop control, i.e., control when no observations are made, with a situation where sensory information is available we must be able to generate open-loop motions on the FRF-automata. It is clear that the input sequence $\boldsymbol{\sigma}_{ol} = (v_1, \kappa_{ol}, \tau_{ol}) \cdots (v_q, \kappa_{ol}, \tau_{ol}) \in \Sigma^\star$, where $\kappa_{ol}(v, y) = v \; \forall v \in V, \; y \in Y$, $\tau_{ol}(y) = 1 \; \forall y \in Y$ achieves this. However, this word has length q, and it is drawn from the input alphabet $\Sigma = V \times V^{Y \times V} \times \{0,1\}^Y$, and thus the description length is $\mathcal{L}(\boldsymbol{\sigma}_{ol}, \Sigma) = q \log_2(\text{card}(\Sigma))$. But, this is clearly not a very meaningful result. Instead we can restrict the input alphabet to be $\Sigma_{ol} = V \times \{\kappa_{ol}\} \times \{\tau_{ol}\}$, which has cardinality $\text{card}(V)$. The description length of $\boldsymbol{\sigma}_{ol}$ is now $\mathcal{L}(\boldsymbol{\sigma}_{ol}, \Sigma_{ol}) = q \log_2(\text{card}(V))$, relative to the smaller input set Σ_{ol}, which is the description length we should expect in the purely open-loop case. (We do not want the complexity to depend on $\text{card}(Y)$ since we do not rely on the outputs for specifying the evolution of the automaton).

Now, consider a connected, classical, finite automaton $A = (X, V, \delta)$. We recall that the *backward eccentricity* of a state, $ecc(A, x)$, is the minimum number of instructions necessary for driving the automaton from any other state to x. (See, for example, [6]). We define the *radius* of A to be

$$\text{radius}(A) = \min_{x \in X} ecc(A, x).$$

Consider the FRF-automaton $\tilde{A}$. If we let

$$\mathcal{C}(\tilde{A}, x) = \max_{x_0 \in X} \mathcal{C}(\tilde{A}, x_0, x)$$

then we directly get that

$$\begin{aligned} \mathcal{C}(A_{ol}, x) &= ecc(A, x) \log_2(\text{card}(V)) \\ &\geq \text{radius}(A) \log_2(\text{card}(V)) \end{aligned}$$

where A_{ol} is the FRF-automaton $(X, Y, \Sigma_{ol}, \delta, \gamma)$.

A. Main Theorem

The previous definitions enable us to state the following theorem.

Theorem 4.1 (Main Theorem): Assume that $\text{card}(V) \geq 2$. Suppose that $x_f \in X_f$, where X_f is an observable subset for the finite automaton A. Assume that $\text{card}(\gamma(X_f)) < \text{card}(X_f)$ and $\gamma(X_f) \cap \gamma(X \backslash X_f) = \emptyset$. If X_f is ballistically reachable from $X \backslash X_f$, and x_f is control-invariantly reachable in X_f, then there exists a FRF-automaton $A_{FRF} = (X, Y, \Sigma', \delta, \gamma)$ such that

$$\frac{\mathcal{C}(A_{FRF}, x_f)}{\mathcal{C}(A_{ol}, x_f)} \leq \frac{4\text{card}(X_f)}{\text{radius}(A)}.$$

Proof: The proof is found by investigating the size of the input alphabet necessary for generating the input in Lemma 3.3. We can let $\Sigma' = \{v_{ol}\} \times V^O \times \{0,1\}^O$, and let the input, $\sigma = (v, \kappa, \tau)$, be given by

$$\begin{cases} v = & \text{arbitrary} \\ \kappa(o, v) = v_{ol}, & \text{if } o = e \\ \kappa(o, v) = w_{obs}(o), & \text{if } o \in \gamma(X_f) \\ \kappa(o, v) = w_{attr}(o), & \text{if } o \in X_f \\ \tau(o) = 1, & \text{if and only if } o = x_f. \end{cases}$$

The size of the input space is thus

$$\begin{aligned} \text{card}(\Sigma') &= (2\text{card}(V))^{(1+\text{card}(\gamma(X_f))+\text{card}(X_f))} \\ &\leq (2\text{card}(V))^{2\text{card}(X_f)} \\ &\leq \text{card}(V)^{4\text{card}(X_f)}. \end{aligned}$$

The description length $\mathcal{L}(\sigma, \Sigma')$ is, thus, given by

$$4\text{card}(X_f)\log_2(\text{card}(V)).$$

Now, since $\mathcal{C}(A_{ol}, x_f) \geq \text{radius}(A)\log_2(\text{card}(V))$, the theorem follows. ∎

B. Chained Version of the Main Theorem

Goals are seldom final goals. More often they tend to be intermediate goals in a grander scheme. This is for instance the case when mobile robots are navigating using landmarks. The theory that we have developed so far does not acknowledge this fact, and in this subsection we modify it so that we can take into account the situation where a number of goal states are visited by the automaton.

It is clear that the premises on which the previous theorem is based are too restrictive to capture the *chained* structure that intermediary goals give rise to. Instead, we need to extend the trajectories from the main theorem (Theorem 4.2) through a chain of goals states. This can be achieved by assuming that we work with an automaton where subset-observers can be designed around different states, i.e., the intermediate goals. We also assume that the sets on which the observers are defined are ballistically reachable from each other. We could then use open-loop control for driving the system between these sets on the parts of the state space where the lack of sensory information prevents effective use of feedback. We compliment this with feedback controllers on the subsets where subset-observers can be constructed, as seen in Fig. 6. For the sake of completeness, we explicitly state the chained extension of the main theorem as a corollary.

Corollary 4.1: (Chained Version of the Main Theorem) Assume that $\text{card}(V) \geq 2$. Let the sets $X_1, \ldots, X_n$ be disjoint, observable subsets with cardinality less than or equal to C, where $\gamma(X_i) \cap \gamma(X \backslash X_i) = \emptyset$, $\gamma(X_i) \cap \gamma(X_j) = \emptyset$, $i \neq j$. Let $x_f \in X_n$ be control-invariantly reachable in X_n and let X_1 be ballistically reachable from x_0. Assume that there exists intermediary goals $x_i \in X_i$, $i = 1, \ldots, n-1$ such that x_i is control-invariantly reachable in X_i and X_{i+1} is ballistically reachable from x_i. Then there exists a FRF-automaton $A_{FRF} = (X, Y, \Sigma', \delta, \gamma)$ such that

$$\frac{\mathcal{C}(A_{FRF}, x_f)}{\mathcal{C}(A_{ol}, x_f)} \leq \frac{4nC}{\text{radius}(A)}.$$

Proof: In order to prove this corollary, we need to construct an observer that makes it possible to reach x_f using as few instructions as possible. In fact, we will show that one instruction is enough, and for this we set the state space of the observer to be

$$Z = \{e_1, \ldots, e_n\} \cup Y \cup \cdots \cup Y^{\bar{p}_{obs}-1} \cup X$$

where $\bar{p}_{obs}$ is the largest of all the positive integers p_{obs_i} associated with each observable subset X_i, $i = 1, \ldots, n$. The idea here is to let e_i denote the state of the system when open-loop paths are followed between the different subsets; elements in $Y^\star$ are to be used when the observer automaton is settling in a particular subset, and a copy of the state of the original automaton should be used once the observer has settled. With this in mind, we let the outputs of the observer automaton be y when the observer is settling, and let it be z during the ballistic motions between the different subsets. We also assign the value z when the observer has settled on those subsets, i.e.,

$$h(z, y) = \begin{cases} y, & \text{if } z \notin X \cup \{e_1, \ldots, e_n\} \\ z, & \text{otherwise.} \end{cases}$$

Our output from the observer automaton, thus, becomes

$$O = \{e_1, \ldots, e_n\} \cup \gamma(X_1) \cup \cdots \cup \gamma(X_n) \cup X_1 \cup \cdots \cup X_n.$$

In order to get an evolution of the observer automaton that is consistent with these choices, we let the transitions be generated as follows: $g(z_k, y_k, w_k(h(z_k, y_k), v_k))$ is given by the equation shown at the bottom of the page.

If we let $\Sigma' = \{\hat{v}\} \times V^O \times \{0, 1\}^O$, for some arbitrary $\hat{v} \in V$, we can use $\sigma = (v, \kappa, \tau)$ as the input to the FRF-automaton, where

$$\begin{cases} v = \hat{v} & \\ \kappa(o, v) = v_{ol_i}, & \text{if } o = e_i \\ \kappa(o, v) = w_{obs_i}(o), & \text{if } o \in \gamma(X_i) \\ \kappa(o, v) = w_{attr_i}(o), & \text{if } o \in X_i \\ \tau(o) = 1, & \text{if and only if } o = x_f. \end{cases}$$

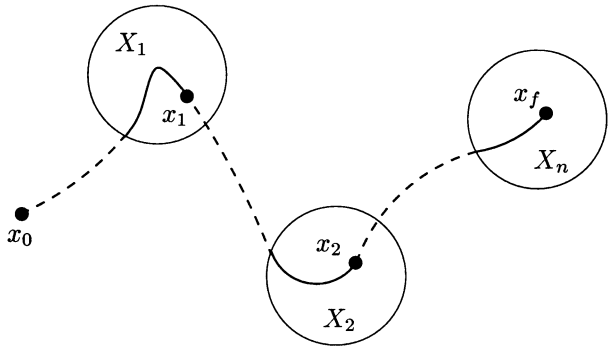

Fig. 6. Chained version of the main theorem. The dashed lines correspond to open-loop trajectories, while feedback is used for generating the solid-line trajectories.

$$\begin{cases} e_i & \text{if } z_k = x_{k-1} \\ e_i & \text{if } z_k = e_i \text{ and } y_k \notin \gamma(X_i) \\ y_k & \text{if } z_k = e_i \text{ and } y_k \in \gamma(X_i) \\ z_k \cdot y_k & \text{if } z_k \in Y^q,\ q < p_{obs_i} - 1 \\ \Xi(z_k \cdot y_k, w_k(h(z_k, y_k), v_k)) & \text{if } z_k \in Y^{p_{obs_i}-1} \\ \delta(z_k, v_k) & \text{if } z_k \in X_i. \end{cases}$$

Here, v_{ol_i} is the open-loop control that drives the automaton from x_i to X_{i+1}, w_{obs_i} is the feedback control that makes the observer converge on X_i, and w_{attr_i} is the feedback control that drives the automaton to x_i without the trajectory leaving X_i. It is straight forward to check that by using this input, the FRF-automaton $(X, Y, \Sigma', \delta, \gamma)$ drives from any initial state to x_f. ■

The size of the input space is thus

$$\begin{aligned}\mathrm{card}(\Sigma') =& (2\mathrm{card}(V))^{\sum_{i=1}^{n}(1+\mathrm{card}(\gamma(X_i))+\mathrm{card}(X_i))}\\ \leq& (2\mathrm{card}(V))^{\sum_{i=1}^{n} 2\mathrm{card}(X_i)}\\ \leq& \mathrm{card}(V)^{\sum_{i=1}^{n} 4\mathrm{card}(X_i)}\\ \leq& \mathrm{card}(V)^{4nC}\end{aligned}$$

and, hence

$$\frac{\mathcal{C}(A_{FRF}, x_f)}{\mathcal{C}(A_{ol}, x_f)} \leq \frac{4nC}{\mathrm{radius}(A)}.$$

One conclusion to be drawn from Corollary 4.1 is that the increase in the description length caused by the summation over many intermediate goals, can be counter-acted by choosing smaller feedback sets. In the mobile robot case, this would corresponding to using many easily detectable, local landmarks as a basis for the navigation system.[7]

V. Conclusion

In this paper, we formulate and solve some problems involving the search for short descriptions of control procedures. In particular, we investigate the difference in the description lengths of the inputs when controlling dynamical systems with and without reference to sensory information.

We show that when feedback can be used in the description of motor programs, the length of the descriptions can be reduced by a factor that reflects the richness of the available feedback signals. In the domain of task descriptions, where the objective can be stated in terms of reaching a goal state, feedback reduces the description length of the motor programs that execute the task. In particular, we show in Theorem 4.1 that the reduction depends on the ratio between the size of the entire state space and the size of the set of states for which feedback is locally effective. This argument is furthermore used iteratively, leading to further reductions, as seen in Corollary 4.1.

To search for short descriptions of control procedures is an age old problem but the expression of it in a precise language seems to be new. There are many possible applications, and for instance in teleoperated robotics, the control signals are transmitted over communication channels in which the presence of channel noise makes it preferable to transmit instructions that are as short as possible. A related problem arises in the area of minimum attention control, where an attention functional is defined as a measure of the control variability. (See, for example, [5]). The problem then becomes that of minimizing the cost functional under the additional constraint that the servomechanism should perform in a satisfactory way. It can also be argued that this way of imposing complexity measures on control procedures has implications for decentralized or embedded control strategies, where the idea is to minimize the communication between different control modules at the same time as sufficient information must be available in order for the overall system to meet its specifications. Apart from the complexity theorems derived in this paper, a key contribution is thus the model in itself, which allows us to measure the specification complexity of different control procedures.

[7]This result is furthermore consistent with the statistical data recorded in the MapQuest example, in the special case where two observable subsets around x_0 and x_f were used. In this case, the ballistic motions that drive the automaton between these two subsets correspond to the long open-loop motions along the highways connecting different cities.

Appendix
Languages Recognized by Free-Running, Feedback Automata

From Definition 2.2, we see that the variables defining the state of a FRF-automaton are x and l, where x takes on values in a finite set, whereas l takes on values in the nonnegative integers. There is no *a priori* limit on the size of l because there is no *a priori* limit on the length of the input string. Thus, one could ask if the introduction of this infinity affects the languages recognizable by FRF-automata.

We say that a FRF-automaton recognizes the language $L \subset \Sigma^\star$ if all strings in L drive the automaton from the initial state x_0 to a distinct final state x_f. To characterize the language recognized by a given FRF-automaton, we need to introduce the concept of *scope*. Given $x_q \in X$. We say that $\sigma = (v, \kappa, \tau)$ has scope $scope(\sigma, x_q) = r$ if the FRF-automaton, initialized at x_q, maximally advances the state r steps without advancing l.

Lemma 5.1: Consider the FRF-automaton $(X, Y, \Sigma, \delta, \gamma)$. Let $x \in X$ and $\sigma = (v, \kappa, \tau) \in \Sigma$. If there is a periodic orbit of

$$x_{k+1} = \delta(x_k, \kappa(y_k, v)),\ x_0 = x$$

without the interrupt τ triggering, then $scope(\sigma, x)$ is infinite. If there are no such periodic orbits, then $scope(\sigma, x) < \mathrm{card}(X)$.

Proof: If the state is advanced more than $\mathrm{card}(X) - 1$ steps then, by the Pigeon Hole Principle (see, for example, [11]), the same state must have been visited twice. Since the control input is kept constant, a periodic orbit must have been encountered. Hence, the scope is infinite. If, on the other hand, no periodic orbit is encountered, then no state is visited more than once, which implies that $scope(\sigma, x) < \mathrm{card}(X)$. ■

Essentially, what this means is that the only way a given σ can have infinite scope is if it makes the FRF-automaton cycle through an interrupt-free orbit.

Theorem 5.1: If $L \subset \Sigma^\star$ is recognized by the FRF-automaton $(X, Y, \Sigma, \delta, \gamma)$, with $\delta(x_f, v) = x_f,\ \forall v \in V$, then L is also recognized by some finite automaton.

Proof: Let $\boldsymbol{\sigma} = \sigma_1 \cdots \sigma_{q_\sigma} \in L$, where L is the language recognized by the FRF-automaton. Let $\delta(x, \sigma^k) \in X$ denote the state reached after applying (v, κ) repeatedly k times, e.g.,

$$\delta(x, \sigma^2) = \delta\Big(\delta\big(x, \kappa(\gamma(x), v)\big), \kappa\big(\gamma(\delta(x, \kappa(\gamma(x), v))), v\big)\Big).$$

Assume that all the input symbols in the input string have finite scope with respect to the recursively defined initial states. Label these initial states as

$$\begin{aligned}\tilde{x}_0^\sigma &= x_0 \\ \tilde{x}_1^\sigma &= \delta\left(\tilde{x}_0^\sigma, \sigma_1^{scope(\sigma_1, \tilde{x}_0^\sigma)}\right) \\ \tilde{x}_2^\sigma &= \delta\left(\tilde{x}_1^\sigma, \sigma_2^{scope(\sigma_2, \tilde{x}_1^\sigma)}\right) \\ &\vdots \\ \tilde{x}_{q_\sigma}^\sigma &= \delta\left(\tilde{x}_{q_\sigma-1}^\sigma, \sigma_{q_\sigma}^{scope(\sigma_{q_\sigma}, \tilde{x}_{q_\sigma-1}^\sigma)}\right).\end{aligned}$$

Since L is recognized by the FRF-automaton we must have that $\tilde{x}_q^\sigma = x_f$.

Now, define the finite automaton $A = (X, Y, \Sigma, \tilde{\delta}, \gamma)$, with initial state x_0 and

$$\tilde{\delta}(\tilde{x}_i^\sigma, \sigma_{i+1}) = \tilde{x}_{i+1}^\sigma,\ i = 0, \ldots, q_\sigma - 1$$

where $\tilde{x}_i^\sigma,\ i = 0, \ldots, q_\sigma$ are as previously defined.

By repeating this argument for all $\boldsymbol{\sigma} \in L$ such that the finite scope assumption holds, the resulting finite automaton recognizes every finite scope word $\boldsymbol{\sigma} \in L$. However, if the assumption is false, i.e., there is a $\boldsymbol{\sigma} \in L$ such that the finite scope assumption does not hold, then there is a σ_j for some $j \in \{1, \ldots, q_\sigma\}$, such that the scope is infinite when starting from the initial state $\tilde{x}_{j-1}^\sigma$. Then, by Lemma 5.1, the FRF-automaton has encountered an interrupt-free periodic orbit. If x_f is not on this orbit then $\boldsymbol{\sigma}$ is not recognized by the FRF-automaton, which is a contradiction. If, on the other hand, x_f is on the orbit then the orbit is, in fact, only consisting of one point $\{x_f\}$ since $\delta(x_f, v) = x_f,\ \forall v \in V$. Hence σ is recognized by the FRF-automaton. The finite automaton can thus be redefined to interpret inputs according to the following rules:

$$\begin{aligned}\tilde{x}_0^\sigma &= x_0 \\ \tilde{x}_1^\sigma &= \delta\left(\tilde{x}_0^\sigma, \sigma_1^{\min\{scope(\sigma_1, \tilde{x}_0^\sigma), \mathrm{card}(X)\}}\right) \\ \tilde{x}_2^\sigma &= \delta\left(\tilde{x}_1^\sigma, \sigma_2^{\min\{scope(\sigma_2, \tilde{x}_1^\sigma), \mathrm{card}(X)\}}\right) \\ &\vdots \\ \tilde{x}_{q_\sigma}^\sigma &= \delta\left(\tilde{x}_{q_\sigma-1}^\sigma, \sigma_{q_\sigma}^{\min\{scope(\sigma_{q_\sigma}, \tilde{x}_{q_\sigma-1}^\sigma), \mathrm{card}(X)\}}\right)\end{aligned}$$

and, as before

$$\tilde{\delta}(\tilde{x}_i^\sigma, \sigma_{i+1}) = \tilde{x}_{i+1}^\sigma,\ i = 0, \ldots, q_\sigma - 1$$

which concludes the proof. ■

REFERENCES

[1] M. Arbib, Ed., *Algebraic Theory of Machines, Languages, and Semigroups*. New York: Academic, 1968.

[2] H. S. Black, "Stabilized feedback amplifiers," *The Bell System Technical Journal*, pp. 1–18, Jan. 1934.

[3] R. W. Brockett, "On the computer control of movement," in *Proc. 1988 IEEE Conf. Robotics Automation*, New York, Apr. 1988, pp. 534–540.

[4] ——, "Hybrid models for motion control systems," in *Perspectives in Control*, H. Trentelman and J. Willems, Eds. Boston, MA: Birkhauser, 1993, pp. 29–54.

[5] ——, "Minimum attention control," in *Proc. IEEE Conf. Decision Control*, 1997, pp. 2628–2632.

[6] F. Buckley and F. Harary, *Distance in Graphs*. Reading, MA: Addison-Wesley, 1990.

[7] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York: Wiley, 1991.

[8] W. Fleming and P. L. Lions, "Stochastic Differential Systems, Stochastic Control Theory and Applications," in *The IMA Volumes in Mathematics and Its Applications*. New York: Springer-Verlag, 1987, vol. 10.

[9] A. Ginzburg, "Algebraic theory of automata," in *ACM Monograph Series*. New York: Academic, 1968.

[10] Y. Ho, Ed., *Discrete Event Dynamic Systems*. New York: IEEE Press, 1992.

[11] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, 2 ed. Reading, MA: Addison-Wesley, 2000.

[12] M. Itô, *The Cerebellum and Neural Control*. New York: Raven, 1984.

[13] S. E. Kels, *Human Motor Behavior*. Hillsdale, NJ: Lawrence Erlbaum, 1982.

[14] MapQuest [Online]. Available: http://www.mapquest.com/

[15] D. Wätjen, "Feedback automata and their languages," *Inform. Processing Lett.*, vol. 21, pp. 83–86, 1985.

Magnus B. Egerstedt (S'99–M'00) was born in Stockholm, Sweden, in 1971. He received the B.A. degree in philosophy from Stockholm University, Stockholm, Sweden, and the M.S. degree in engineering physics and the Ph.D. degree in applied mathematics from the Royal Institute of Technology, Stockholm, Sweden, in 1996 and 2000, respectively.

He is currently an Assistant Professor in Electrical and Computer Engineering at the Georgia Institute of Technology, Atlanta. He spent 2000–2001 as a Postdoctoral Fellow in the Division of Engineering and Applied Science at Harvard University, Cambridge, MA. His research interests include optimal control as well as modeling and analysis of hybrid and discrete-event systems, with emphasis on motion planning and control of mobile robots.

Roger W. Brockett (S'62–M'63–SM'73–F'74) received the B.S., M.S., and Ph.D. degrees from Case Western Reserve University, Cleveland, OH, in 1960, 1962, and 1964, respectively.

He is the An Wang Professor of Electrical Engineering and Computer Science in the Division of Applied Science at Harvard University, Cambridge, MA. He taught for six years in the Electrical Engineering Department at the Massachusetts Institute Technology, Cambridge, before joining Harvard in 1969. He has contributed extensively to the theory of automatic control with work on stability, nonlinear control, feedback linearization, system identification, nonlinear estimation, pole placement, hybrid systems, and robotics. More recently, his work has involved problems arising in the study of intelligent machines. Areas of particular interest include the problem of motion control, and the investigation of new computational paradigms appropriate for control in the high data rate, sensory rich environments that characterize vision guided systems.

Dr. Brockett has been recognized by the American Automatic Control Council and by the IEEE through their Richard Bellman Award and the Control System Science and Engineering Award, and has received the Reid Prize from the Society of Industrial and Applied Mathematics. He has served on a variety of National Research Council Panels and is a Member of the National Academy of Engineering.