COMMUNICATIONS IN INFORMATION AND SYSTEMS
Vol. 8, No. 2, pp. 85-106, 2008

002

OBSERVERS IN LANGUAGE-BASED CONTROL*

SEAN B. ANDERSSON†, DIMITRIS HRISTU-VARSAKELIS‡, AND MORTEZA LAHIJANIAN†

Abstract. This work discusses the construction of an observer for language-driven control systems. Such systems accept symbolic inputs corresponding to feedback control laws together with logical conditions which determine when each control law is to be applied. Here, we explore the problem of how to identify the symbolic string driving the system through observation of its output. We discuss the identification of individual "instructions" from which inputs are composed, as well as the recovery of more global features of the input. These features can be quite complex depending on the richness of the grammar. Our approach is motivated by settings where language-driven systems (e.g., mobile robots) must cooperate "silently", so that a newcomer who wants to be useful must first determine what others in the team are trying to accomplish. Our ideas are illustrated in a series of numerical experiments involving symbolic control of linear systems using the motion description language MDLe.

1. Introduction. One of R. W. Brockett's influential ideas was that in systems of even mild complexity, such as robots moving about in their environment, control design should take place not in the space of mappings between sensor and actuator signals but rather in a set of control primitives. These primitives, termed *atoms*, corresponded roughly to well-defined feedback control laws [1]. By assigning a symbol to each atom, one could then compose control strings — elements of a so-called Motion Description Language (MDL) — which would be "interpreted" by a system to produce trajectories, as it evolved under the controllers corresponding to the various symbols. This need to "lift" the controller design process from the level of individual sensors and actuators is perhaps evident to anyone working with laboratory robots, in the same way that programmers understand the need for hierarchical programming and high-level languages. There are at least two reasons for doing so, one having to do with the desire to achieve hardware-independence (as Brockett himself has often illustrated using the example of postscript as a language for printers), the other being the reduction in complexity that results when passing to the language-based setting [2] .

With Brockett's original MDL papers [1, 3], these ideas were placed on a systems and control-oriented foundation that codified the existing intuition and – to a certain extent – also coincided with the beginning of a broader effort in the control systems community aimed at making contact with ideas from computer science. His work

*Dedicated to Roger Brockett on the occasion of his 70th birthday.

†Dept. of Mechanical Engineering, Boston University, Boston, MA, USA 02215. E-mail: {sanderss,morteza}@bu.edu

‡Dept. of Applied Informatics, University of Macedonia, Thessaloniki 54006, Greece. E-mail: dcv@uom.gr

set the stage for new ways of applying existing tools, and thinking about control problems at a more abstract level [4]-[8]. It has also lead to the development of other control languages and similar constructs, all of which, at one level or another, seek to take advantage of the complexity reduction offered by symbolic control. Besides the "direct descendants" of MDL, [9]-[11], one finds in the literature a broad collection of works whose viewpoint is clearly influenced by the basic idea of a motion description language and symbolic control. A small sampling of recent works includes [4], [12]-[20].

The goal of this paper is to explore the problem of determining the symbolic inputs driving a control system from observations of its (continuous) outputs. Our work is motivated by a fundamental problem that emerges in the areas of robotics and cooperative control, namely how to deduce what a robot is doing by observing its actions. Since the time of the Brockett's original work on MDLs, robot cooperation has emerged as one of the focal points in robotics research, especially in the context of mobile robots and teams thereof. The complexity and hardware-independence arguments that motivated the development of MDL appear even stronger in this newer context. For the present work, we have in mind scenarios in which multiple language-driven robots must cooperate "silently" for reasons having to do with security, secrecy, energy consumption, or lack of communication equipment.

In a cooperation-without-communication setting, there are a series of challenges that arise, not the least of which is how one should participate in the team once the goal is understood (for example, moving an object requires a much different type of coordination than does searching a large area). Perhaps the most basic problem that an autonomous robot must undertake before being able to assist its peers, however, is to determine *what* they are trying to accomplish. Without communication, the newcomer must do this simply by observing the other's actions. In a language-based setting, this translates to understanding which control primitive(s) (MDL atoms) a subject is executing, based on observations; essentially, we are asking for a *symbolic observer*. This paper is concerned precisely with the construction of such observers. If a control system evolves based on a collection of control primitives (e.g. atoms of a motion description language or similar construction), then instead of a state estimate, the language-based observer should produce an estimate of which symbol "runs" at each time. In cases where the control language supports additional structure (e.g., the "behaviors" and "plans" of the extended Motion Description Language (MDLe) [9]) the observer should also recover that structure. Besides being interesting from a systems viewpoint, such a symbolic observer lies at the heart of "silent cooperation" in multi-robot groups. As we will see, studying the matter in a language-based setting is a good choice in the present context, for the same reason that led Brockett to propose MDLs, namely hardware independence, and "freedom" from too many options (i.e., an infinite number of control laws, vs. combinations of a few basic primitives). The main contribution of the current work is a structured approach for handling the basic

case (a simplified language syntax and linear time-invariant dynamics) by composing existing tools from estimation and hypothesis testing.

As alluded to in the previous discussion, viewing the problem in question as one of *observation* implies that in addition to the usual state-space in which the underlying system evolves, we consider also a discrete state corresponding to the linguistic primitive being used to steer the system at a particular time. Although we will not require any dynamics at the level of linguistic primitives here, the observer viewpoint will be a helpful "guide". Alternatively, one could think of this as an inverse problem where the input space is made up of strings rather than continuous signals. While we will not explore that route here, it is interesting to note the parallels between such an approach and the basic problem behind R. W. Brockett's doctoral dissertation [21].

The remainder of this paper is organized as follows. In Section 2 we give a brief description of the motion description language MDLe. We make extensive use of MDLe in this work, partly for the sake of concreteness, but we note that what is described here could be accomplished with other motion description languages as well, including some of those cited in the Introduction. Section 3 discusses the fundamental challenges involved in constructing a observer for symbolic inputs and outlines a general approach to the problem. Section 4 applies this approach to the special case of a linear system and presents a series of numerical experiments.

2. The motion description language MDLe . Consider an underlying physical system (equipped with a set of sensors and actuators) for which we want to specify a motion control task. Let the system's evolution be governed by

$$x(k+1) = f(x(k), u(k)) + G(x(k))w(k); \quad y(k) = h(x(k)) + \nu(k) \tag{1}$$

where $x \in \mathcal{X}^n$, an n-dimensional manifold, $u(k) \in \mathbb{R}^m$ may be an open loop input or a feedback law of the type $u = u(k, h(x(k)))$, $y(k) \in \mathbb{R}^p$, and G is a matrix whose columns are vector fields in $\mathbb{R}^n$. The system is subject to sensor and actuator noise, $\nu(k)$ and $w(k)$, respectively. The noise processes are taken to be bounded and independent. We assume that $\mathcal{X}^n$ may contain obstacles and that there may be subsets of $\mathcal{X}^n$ that are uncertain or unknown.

In order to be able to discuss control problems for (1) in a language-based setting, we will first impose some structure on the kinds of inputs (control laws) that will be allowed, as well as on the rules which will govern their compositions. The result will be a formal language in which motion control tasks will be described, and which will allow us to compose complex control laws from simpler ones. See [11] for a more complete description of MDLe's grammar; for background material on formal languages and grammars, see [22].

Consider for the moment the deterministic version of (1) where $\nu(k) = w(k) = 0$ for all time. Let $\mathcal{U} = \{u : \mathbb{R}^p \times \mathbb{R}^+ \to \mathbb{R}^m\}$ be a finite subset of the set of possible

control laws (sometimes termed "control quarks"), including the trivial $u_{null} = 0$. Let $\mathcal{B} = \{\xi = (\psi \texttt{ AND } (t \leq T)),\ \psi : \mathbb{R}^p \to \{0,1\}, T \in \mathbb{R}^+ \cup \{\infty\}\}$ be a finite set of boolean functions of the output variables and time, including the null function $\xi_{null} : \mathbb{R}^p \times \mathbb{R}^*_+ \to 1$. We will refer to elements of $\mathcal{B}$ as *interrupt* functions. The sets $\mathcal{U}$ and $\mathcal{B}$, together with the special symbols "(", ")" and "," define a finite alphabet over which the MDLe strings are formed.

Definition 1 ([11]). *MDLe is the formal language generated by the context free grammar*
$\mathcal{M} := (\mathcal{N}, \mathcal{T}, S, \Pi)$*, where:*
$\mathcal{N} = \{Q\}$ *is the set of* non-terminal *symbols, where* Q *is designated as the* start symbol*;*
$\mathcal{T}$ *is the set of* terminal *symbols, i.e. the set of symbols that form the strings of the language. In other words,* $\mathcal{T} = \mathcal{U} \cup \mathcal{B} \cup \{(), \}$*;*
$\Pi \subset \mathcal{N} \times (\mathcal{N} \cup \mathcal{T})^*$ *is a finite relation which consists of the following* production rules*:*

(P1) $Q \to \epsilon$*, where* ϵ *is the null string.*
(P2) $Q \to (u, \xi),\ u \in \mathcal{U},\ \xi \in \mathcal{B}$
(P3) $Q \to QQ$
(P4) $Q \to (Q, \xi),\ \xi \in \mathcal{B}$

To compose a string in MDLe one begins with a collection of nonterminal symbols (in this case the only choice is the start symbol Q) and applies the production rules P1-P4 until the resulting string contains only terminal symbols. For example, a single application of P2 yields the simplest MDLe strings, termed *atoms*; they are pairs of the form (u, ξ), where the control u is selected from $\mathcal{U}$ and the interrupt ξ from $\mathcal{B}$. To *evaluate* or *run* the atom (u, ξ) means to apply the input u to the dynamical system until the interrupt function ξ is "high" (logical `1`).

MDLe allows us to compose motion control programs (generically referred to as *plans*), from other simpler programs, all the way down to hardware-specific feedback laws. For example, we could use the production rules P4, P3, P2, P2 to form $Q \to (Q, \xi_b) \to (QQ, \xi_b) \to ((u_1, \xi_1)Q, \xi_b) \to ((u_1, \xi_1)(u_2, \xi_2), \xi_b)$. Evaluating the plan $b = ((u_1, \xi_1)(u_2, \xi_2), \xi_b)$ means evaluating the atom (u_1, ξ_1) followed by (u_2, ξ_2) until the interrupt function ξ_b returns "high" (logical `1`), or (u_2, ξ_2) has terminated, whichever occurs first. Plans themselves can be composed to form higher-level strings, e.g., $((b, (u_3, \xi_3)), \xi_4)$. MDLe is a context free language, and it is therefore possible to write an MDLe compiler (e.g., [10, 4]). For more details on the language and its complexity, including example programs, see [11].

3. Observers for Language-driven systems . The basic problem we are concerned with is the following:

Problem 1. *Given a finite alphabet of control and interrupt quarks,* $\mathcal{U}$ *and* ψ*, find the MDLe plan that produced the observed output* y *of (1), assuming that one*

exists.

Solving this problem requires one to be able to identify:

1. the control quarks of which the plan is composed
2. the times at which atom transitions occur
3. the interrupt that causes each transition, and
4. the overall structure of the MDLe plan.

In the sequel we develop a method to accomplish the first three items in the case where the plan structure is straightforward, namely a series of atoms with no higher-level behaviors or nesting. A possible starting point for exploring the general case is [11].

3.1. Control Symbol Detection. Consider the system (1), where the input u is determined by an MDLe atom. If u (equivalently, the control quark corresponding to the atom) were known, then we would have available a model for the closed-loop system, and we could construct an estimator of the conditional density of the state based on the observations. In our setting, we do not know the particular control quark that is running at a given time; we know only that it is one of a given and finite collection as defined by the set $\mathcal{U}$. The task of identification is then the problem of determining which control quark was most likely to have given rise to the observations.

The proposed scheme for identifying control quarks is illustrated in Fig. 1. Each element of $\mathcal{U}$ gives rise to a different possible model of the system dynamics. For each such model, we construct a corresponding estimator which is to maintain a conditional density for the system state. That density is then transformed through the observation function into a density on the next measurement. To determine which model is the one most likely to be currently executing, an $m-$ary hypothesis test is used, based on the conditional densities and driven by actual measurements (see, e.g., [23] for details on hypothesis testing).

In between measurements, the conditional density of the state evolves according to the well-known Fokker-Planck equation. Solving this equation in the general case is not practical, and thus some approximation is required. The filter should be selected so as to take advantage of any structure in the system models. For example, in the linear system setting considered in Sec. 4 below, state estimation is performed using Kalman filters. One notable possibility for handling the general case is to use a particle (or sequential Monte Carlo) filter, or one of its many variants [24].

To apply the m-ary hypothesis framework based on a single measurement, we assume we have *a priori* probabilities π_i, $i = 1, \ldots, N_c$ on the likelihood that control quark i is being run at the current time. As described in Sec. 3.3 below, these probabilities are updated based on the measurements and the collection of plans through the use of a partially-observed Markov chain framework. Hypothesis H_i in the tester is that the current measurement, $y[k]$, comes from the conditional probability

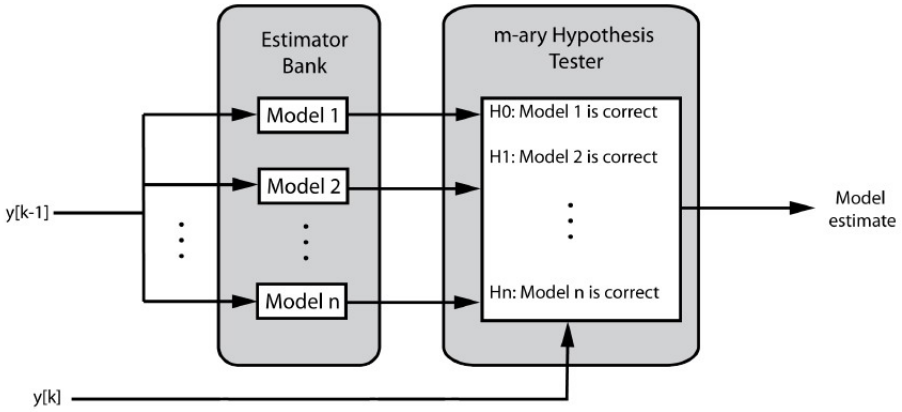


FIG. 1. *Illustrating control quark detection. The current observations are used to drive a bank of estimators, one for each possible control quark. Each estimator is used to generate the predicted density of the next observation. A bank of hypothesis testers compares the next observation to the set of predicted densities to determine whether or not a particular model was likely to have given rise to the measurement.*

density of the observation under the model corresponding to the i^{th} control quark:

$$H_i : y[k] \sim p_i[k]. \tag{2}$$

The hypothesis H_i is accepted if it is most likely to have given rise to the observation $y[k]$.

If we assume that each control quark produces a unique observation, then in the absence of noise and after some possible transient behavior, the m-ary test will settle on a single hypothesis corresponding to the current control quark. In any practical setting, however, noise cannot be ignored and single measurements are not expected to be unique. To improve the performance of the observer, the hypothesis test can be modified to use multiple measurements. To do so, one builds the conditional joint distributions based on the previous m measurements rather than on a single measurement. Under the assumption that the observation noise process is white, the ith hypothesis becomes

$$H_i : y[k] \sim \rho_m[k] := p_i[k]p_i[k-1] \dots p_i[k-m]. \tag{3}$$

The use of multiple observations helps to overcome the effects of noise on the measurements by effectively increasing the signal-to-noise ratio. As m increases, however, the time it takes the hypothesis tester to settle also increases, thus delaying detection and identification of symbol transitions. Intuitively, we expect that more samples will be needed if the measurement noise is large. There is a tradeoff, then, between the robustness of the symbol identification and the speed of detection of symbol transi-

tions. Here, we use confidence ellipsoids to determine the number of measurements to use in the identification of a control symbol.

Suppose that $y[k]$, $k = 1, ..., m$ are a series of measurements from one of the estimators, and that their joint conditional distribution is ρ_m. There will be one such series for each estimator, though here we refer to only one of them in order to avoid subscripts. Assuming that the estimators have reached steady state, and that their output sequences are unique, there will be precisely one of these distributions with zero mean error. The problem we are faced with is to determine which of the series came from a zero-mean distribution. We could view this as the problem of estimating the (unknown) mean of each series of samples. Alternatively, one can compute confidence ellipsoids for the mean. Given the second and higher moments of ρ_m together with the distribution of the sample mean, then the α-confidence ellipsoid for each series for a given $0 < \alpha < 1$ is determined by integrating ρ_m over the appropriate region. Depending on the distance between the steady state means for the estimators and on the noise level present, one expects that the confidence ellipsoids may overlap for a small number of measurements. We can thus determine the number of measurements needed for a given confidence level by making additional measurements until only a single ellipsoid contains the origin.

We note that there may be regions in the environment for which two control quarks produce similar measurement sequences, leading to control sequences which are indistinguishable. Such issues are related to the notion of *observability* of a set of control quarks. In such a scenario, the hypothesis tester may settle on an incorrect symbol while the confidence ellipsoids of two or more measurement sequences will contain the origin and overlap. In Section 3.3 below, we show that such estimation errors can be handled by placing the identification problem in a stochastic setting. In short, we will maintain a probability distribution over the set of control symbols using a partially observed Markov chain structure. By running an MDLe program multiple times in either a real or simulated environment, we can build up a statistical description of the probability of detecting control u_i given that control u_j is running. This "observation matrix" can then be used to update a probability density over the set of controls and in turn over a set of possible plans (c.f. Sec. 3.3) using the Markov chain.

3.2. Interrupt Symbol Detection. As with control quarks, we will assume that the "alphabet" of possible interrupt functions is known in advance. Once a transition between atoms is detected, and the control quarks running before and after the transition have been identified according to the scheme proposed above, the time history of sensor data at (or near) the switching time can be used to decide which interrupt "fired". We begin by considering MDLe strings consisting of a sequence of atoms with no higher-level nesting. In such a setting, identifying the interrupt

associated with a transition can be accomplished by calculating the output of each ξ_i given the state estimate and time. If the interrupts are designed to be unique, meaning that only one of them can be active at any given time, the process described here will uniquely identify the interrupt. Coupling this to the atom detection scheme discussed above yields complete identification of the last executed atom.

There are two primary challenges to interrupt detection which need to be addressed. First, one does not expect interrupts to be unique in general. Depending on noise and similarities in the environment, there may be more than one ξ_i indicating a transition at the time of a switch. As an example, consider the statements "I see a red door" and "I see a door", viewed as Boolean functions. To overcome this ambiguity we will once again turn to a Markov chain structure, to be discussed in Sec. 3.3 below. As in the case of uncertainty in control symbol detection, here we can determine the probability of detecting interrupt ξ_i given that ξ_j was the "true" interrupt via multiple executions of an MDLe plan. The corresponding "observation matrix" can then be coupled into the Markov chain structure.

The second challenge is that the switching time, t_s, may be uncertain. If that is the case, then the Markov chain must include not only a probability distribution over the collection of atoms but also over the possible values of t_s.

Remark: When using hypothesis testing or confidence ellipsoids to detect atom transitions, we must contend with the fact that the time-window during which measurements are made may include a switching time. Of course, we do not know with certainty *when* a transition has occurred. On the other hand, the previous discussion assumed that the filter bank used to decide which control symbol is running has reached steady-state when measurements are made, so that the distribution from which we are sampling is stationary. Practically, we will deal with this by relying on the confidence intervals for switch detection, and by introducing a short delay between the detection of a switch and the execution of the identification algorithm. The time delay is to be determined by the properties of the estimators, including their convergence rates. Of course, the sum of the delay and measurement times must be less than the minimum time during which an atom may possibly run.

3.3. Plan Detection. In the motion description language MDLe, an atom is generated by pairing a control and interrupt quark; a behavior is generated by combining a sequence of atoms with an interrupt quark; and a plan is determined by combining a sequence of behaviors with an interrupt quark. Thus, given even a finite set of control and interrupt quarks, there is an infinite set of possible plans which can be constructed. If we allow for nesting in our MDLe plans, then identifying what plan is being run based on a series of measurements of the control and interrupt quarks is an exceedingly challenging problem. That is because, despite the "serial" nature of MDLe strings (i.e., control quarks are listed from left to right), the triggering of vari-

ous interrupt during execution mean that different executions of the same string (e.g., starting from different initial conditions) will produce different identified sequences of control quarks (and corresponding interrupts).

The problem we are faced with then, is: given a series of executions of an MDLe plan (each being a sequence of control-interrupt pairs), find an MDLe plan which could have produced every one of them. The main difficulty here is that given two consecutive atoms, $(a_1, \xi_1), (a_2, \xi_2)$, observed in a single execution of a plan, we are unsure if the original plan contained additional atoms "in between" or not. The interrupt ξ_1 could have been a behavior or higher-level interrupt, or it could be an atom-level interrupt, to be matched with a_1. Furthermore, if on a second execution of the same plan, we observe $(a_1, \xi_3), (a_4, \xi_4), ...$, we are not sure whether a_4 belongs in the same behavior as a_1 (i.e., precedes a_2 in the plan) or whether ξ_3 is the interrupt of a behavior which included a_2, in which case a_2 was "skipped over". Such difficulties are neither superficial nor easy to deal with. Every MDLe plan can be transformed to an equivalent directed graph, with nodes and edges corresponding to control quarks and interrupts, respectively [11]. The problem of finding an MDLe plan given a sufficient number of executions is thus related to the problem of inferring a graph from its paths. Recent work on this last problem [25] has shown that it is strongly NP-hard for planar graphs. In the following, we will focus on the detection of simple plans which contain no nesting[1].

We will assume that we have a collection of plans $\Gamma = \{\Gamma_1, \Gamma_2, ..., \Gamma_n\}$, each plan being a sequence of r atoms,

$$\Gamma_i = \{\sigma_{i1}, \sigma_{i2}, ..., \sigma_{ir}\},\ i \in [1, m], \quad q \in [1, n].$$

where each atom σ_{ij} is drawn from a given alphabet of symbols. Assume for convenience that each plan has the same length, although the extension to plans of different lengths is straightforward. Further assume that the Γ_i contain no subsequences which are repeats of a single atom; that is, terms of the form $\sigma_1, \sigma_1 \ldots, \sigma_1$ are expressly forbidden. This is without loss of generality because any such repeats can be collapsed down to a single occurrence of the atom by suitably redefining the interrupt.

Our approach will be to use a partially observed Markov chain to update the conditional probability distribution over the set of atoms. This information will then be employed to calculate the probability distribution over the set of plans. Thus, let $\sigma[j]$ denote the j^{th} atom in the sequence and let $\hat{\sigma}[j]$ denote the measured atom as determined by the identification scheme described in Sec. 3.1. Let $b[j|j]$ be a row vector of conditional probabilities of occurrence of the atoms at index j, given the

[1] The detection of a simple plan with no nesting is similar to the Sequence Inference from Spectrum Feature problem [25] whose complexity is linear in the length of the plan.

prior information up to index j:

$$b[j|j] = \begin{pmatrix} p(\sigma[j] = \sigma_1 \,|\, \hat{\sigma}[j], \ldots, \hat{\sigma}[1]) \\ p(\sigma[j] = \sigma_2 \,|\, \hat{\sigma}[j], \ldots, \hat{\sigma}[1]) \\ \vdots \\ p(\sigma[j] = \sigma_n \,|\, \hat{\sigma}[j], \ldots, \hat{\sigma}[1]) \\ p(\sigma[j] = \sigma_T \,|\, \hat{\sigma}[j], \ldots, \hat{\sigma}[1]) \end{pmatrix}^T .$$

Here σ_T is a terminal symbol required to ensure the transition matrix below is stochastic. The transition matrix is

$$M[j] = \begin{pmatrix} p(\sigma[j+1] = \sigma_1 \,|\, \sigma[j] = \sigma_1) & \ldots & p(\sigma[j+1] = \sigma_n \,|\, \sigma[j] = \sigma_1) & p(x\sigma[j+1] = \sigma_T \,|\, \sigma[j] = \sigma_1) \\ \vdots & & \vdots & \vdots \\ p(\sigma[j+1] = \sigma_1 \,|\, \sigma[j] = \sigma_n) & \ldots & p(\sigma[j+1] = \sigma_n \,|\, \sigma[j] = \sigma_n) & p(\sigma[j+1] = \sigma_T \,|\, \sigma[j] = \sigma_n) \\ 0 & \ldots & 0 & 1 \end{pmatrix},$$

where the value of $p(\sigma[j+1] = \sigma_T \,|\, \sigma[j] = \sigma_i)$ for $i = 1, \ldots, n$ is chosen such that the sum of each row is 1. M is constructued from the set of plans Γ by determining, for each occurrence of σ_i (in any plan) at step j, the next atom in the sequence. The conditional probability vector is updated according to

$$b[j+1|j] = b[j|j]M[j]. \tag{4}$$

To include the atom detection information, we define an observation probability matrix N as

$$N = \begin{pmatrix} p(\hat{\sigma} = \sigma_1 \,|\, \sigma = \sigma_1) & \ldots & p(\hat{\sigma} = \sigma_n \,|\, \sigma = \sigma_1) & 0 \\ \vdots & & \vdots & \vdots \\ p(\hat{\sigma} = \sigma_1 \,|\, \sigma = \sigma_n) & \ldots & p(\hat{\sigma} = \sigma_n \,|\, \sigma = \sigma_n) & 0 \\ 0 & \ldots & 0 & 1 \end{pmatrix}.$$

A straightforward application of Bayes rule leads to the update law

$$b[j+1|j+1] = b[j+1|j]Z(\hat{\sigma}[j+1]) \tag{5}$$

where $Z(\hat{\sigma}[\cdot])$ is a diagonal matrix determined by a row of N. Specifically, for $\hat{\sigma}[j+1] = \sigma_i$ for some i, the diagonal of $Z(\hat{\sigma}[j+1])$ contains the elements of the i^{th} row of N. This information is then used to calculate the probability distribution of plans up to time step k using the following equation.

$$P_j(\Gamma_i) = \frac{\prod_{l=1}^{j} p(\sigma_{il})}{\sum_{q=1}^{m} \prod_{l=1}^{j} p(\sigma_{ql})}, \tag{6}$$

where $P_j(\Gamma_i)$ indicates the probability of plan i at index j, and $p(\sigma_{il})$ is the probability of the l-th symbol of plan i.

We note that the approach proposed here assumes that the index j correctly counts atoms in the plan Γ_i. This corresponds to identifying every transition with no false positives. One approach to overcoming this brittleness is to extend the probability distribution over the set of plans to include the index into the sequences as well.

4. Example: a linear system under feedback control. To illustrate the proposed approach, we apply Section 3 to a discrete-time linear system given by

$$x[k+1] = Ax[k] + Bu[k] + w[k], \tag{7a}$$

$$y[k] = Cx[k] + v[k]. \tag{7b}$$

Here $w[\cdot]$ and $v[\cdot]$ are independent zero mean Gaussian white noise processes with covariances Q and R respectively. We assume the initial state $x[0]$ is a Gaussian random variable with mean $\bar{x}_0$ and covariance Σ_0 and that (A, B, C) is a minimal realization.

A control quark u_i is defined to be a pair $(K_i, \bar{u}_i)$ with a choice of state feedback gain matrix, K_i, and open-loop input, $\bar{u}_i$. For simplicity, in this work $\bar{u}_i$ is taken to be constant. Therefore, at any time the system will evolve according to

$$x[k+1] = A_i x[k] + B\bar{u}_i + w[k], \tag{8a}$$

$$y[k] = Cx[k] + v[k] \tag{8b}$$

where $A_i = A + BK_i$, for some $i = 1, ..., N_c$. Here N_c is the number of elements in the control set $\mathcal{U} = \{(K_1, \bar{u}_1), \cdots (K_{N_c}, \bar{u}_{N_c})\}$. We will assume that each feedback gain is chosen to make the closed loop system exponentially stable.

State estimation for each of the N_c models is performed using a bank of Kalman filters, each of them being driven by the observations. Given the measurements up to time k, the filters produce the conditional means $\hat{x}_i[k+1|k]$ and state covariances $P_i[k+1|k]$ at time $k+1$, under the assumption of control quark i is running, from the standard Kalman filter equations

$$\hat{x}_i[k+1|k] = A_i\hat{x}_i[k|k] + B\bar{u}_i[k], \tag{9a}$$

$$P_i[k+1|k] = A_i P_i[k|k] A_i^T + Q, \tag{9b}$$

$$K_i[k+1] = P_i[k+1|k]C^T(CP_i[k+1|k]C^T + R)^{-1}, \tag{9c}$$

$$\hat{x}_i[k+1|k+1] = \hat{x}_i[k+1|k] + K_i[k+1](y[k+1] - C\hat{x}_i[k+1|k]), \tag{9d}$$

$$P_i[k+1|k+1] = (\mathbb{1} - K_i[k+1]C)P_i[k+1|k]. \tag{9e}$$

These, in turn, yield the mean and covariance of the measurement $y[k]$ assuming that the ith control symbol is currently being run

$$E\{\hat{y}_i[k+1]\} = C\hat{x}_i[k+1|k] \tag{10}$$

$$\Sigma_{\hat{y}_i} = CP_i[k+1|k]C^T + R. \tag{11}$$

The resulting densities are then used in the $m-$ary hypothesis tester to determine the current symbol.

The window size is selected according to the α-confidence interval scheme described in Sec. 3.1. In the case of linear dynamics with Gaussian noise, we can

specialize the approach and write an equation for the confidence ellipsoid generated by each Kalman filter (and thereby corresponding to each possible control symbol). Suppose that $z(i)$, $i = 1, ..., N_c$, are the innovations terms associated with one of the Kalman filters over the time interval $[t_1, ..., t_N]$ for some N (here, we limit our discussion to a single Kalman filter to avoid subscripts). Assuming the filter is in steady state, the $z(i)$ will have a Gaussian distribution, $\rho(z)$, defined by the steady-state mean and variance of the estimation error. The question of whether the corresponding plant-controller model used to construct the associated Kalman filter matches the true parameters of the system under observation is equivalent to the question of whether the mean of that distribution is zero or not.

Let $\bar{z} = \frac{1}{N}\sum_i z(i)$ denote the sample mean. If $\rho(z)$ is Gaussian with parameters (μ, Σ) then the sample mean will be Gaussian as well, with mean μ and variance Σ/N. Thus, the random variable

$$\omega = \left(\sqrt{\Sigma}\right)^{-1} (\bar{z} - \mu)\sqrt{N}$$

is zero-mean, unity variance. Note that in our case, Σ can be computed from the steady-state Kalman filter equations while μ is unknown. To determine whether μ is zero, consider the ellipsoid that contains the sample mean with high probability, α, and let $z_\alpha \in \mathbb{R}$ be such that

$$P(\omega^T \omega < z_\alpha) = \alpha.$$

For a given α, the value of z_α can be calculated using the known distribution for ω. Using the relationship between ω and $\bar{z}$, we then have:

$$P\left((\mu - \bar{z})^T \Sigma^{-1} (\mu - \bar{z}) \leq \frac{z_\alpha}{N}\right) = \alpha.$$

Thus with probability α, the mean of the distribution from which the $z(i)$ are drawn is no further than distance $\sqrt{\frac{z_\alpha}{N}}$ from the sample mean (where the distance is measured according to the metric defined by Σ^{-1}. If the ellipsoid $(\mu - \bar{z})^T \Sigma^{-1} (\mu - \bar{z}) = \frac{z_\alpha}{N}$ contains the origin, we can be confident (at the given level α) that the corresponding $z(i)$ process is zero-mean.

There will be one such ellipsoid defined for every Kalman filter constructed (equivalently, one per control symbol in our alphabet). Thus, it may happen that the origin is initially included in more than one ellipsoid, depending on the amount of measurement noise present as well as the controller parameters. Each of these ellipsoids will be centered at the sample mean produced by the corresponding filter; its size will be determined by the steady-state covariance and will be inversely proportional to the number of samples used, N. By taking N sufficiently large and assuming uniqueness of control symbols, there will be only one ellipsoid containing the origin.

During the identification process, additional measurements are included in the $m-$ary hypothesis test until only a single confidence ellipsoid contains the origin,

thus determining the measurement window size. At that point, the output of the hypothesis tester is declared to be the identified control symbol. The probabilities over the set of tasks are then updated according to the Markov chain. Following this, at each time step, we continue to calculate the sample mean for the estimator corresponding to the identified control quark, over a moving window. The size of the window is constant, as determined by the confidence ellipsoid calculation. If for two consecutive time steps the origin falls outside this confidence ellipsoid centered on the sample mean for the currently identified atom, then an interrupt is declared. After a short period of time to allow the Kalman filters to move towards their new equilibrium values, the process begins again with symbol detection.

4.1. Simulations. We simulated a discrete-time linear system with parameters

$$A = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -1 & \frac{1}{2} & -\frac{1}{2} \end{pmatrix}, \quad B = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \quad C = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}. \tag{12}$$

Three control quarks were defined by the feedback gains

$$K_1 = \begin{pmatrix} -1.59 \\ 1.76 \\ -2.06 \end{pmatrix}, \quad K_2 = \begin{pmatrix} -1.18 \\ 1.62 \\ -1.59 \end{pmatrix}, \quad K_3 = \begin{pmatrix} -1.52 \\ 1.57 \\ -2.05 \end{pmatrix} \tag{13}$$

together with the constant input $\bar{u}_i = 10$, $i = 1, 2, 3$. The input and measurement noises were taken to be Gaussian white noise processes with $w[k] \sim \mathcal{N}(0, 1)$ and $v[k] \sim \mathcal{N}(0, 1)$. The confidence level for the adaptive window-sizing was set to $\alpha = 0.999$. The delay time after a switch was set to three time steps.

Based on the three available control symbols we defined three plans, each consisting of a sequence of six atoms:

$$\begin{aligned} \Gamma_1 &= \{\sigma_1, \sigma_2, \sigma_3, \sigma_1, \sigma_2, \sigma_3\}, \\ \Gamma_2 &= \{\sigma_1, \sigma_3, \sigma_2, \sigma_1, \sigma_2, \sigma_3\}, \\ \Gamma_3 &= \{\sigma_1, \sigma_2, \sigma_3, \sigma_1, \sigma_2, \sigma_1\}, \end{aligned}$$

where each atom contained the corresponding control quark. For simplicity, the interrupt associated with each atom was a simple timer that triggered a transition after 100 time steps. Note that the first and third plans are identical up until the final symbol while the second plan differs from the other two at the second symbol. The priors π_i used in the $m-$ary hypothesis tester were determined from the set of possible tasks and the current sequence index. The priors determined by the three tasks are given in Table 1.

The transition matrix M was also determined from the set of available plans. Recall that $M[j]$ is the transition matrix between symbols from atom index j to time

TABLE 1
Priors for the three tasks.

	Sequence index					
Prior	1	2	3	4	5	6
π_1	1	0	0	1	0	$\frac{1}{3}$
π_2	0	$\frac{2}{3}$	$\frac{1}{3}$	0	1	0
π_3	0	$\frac{1}{3}$	$\frac{2}{3}$	0	0	$\frac{2}{3}$

$j+1$. Given the set of plans above we have

$$M[1] = \begin{pmatrix} 0 & \frac{2}{3} & \frac{1}{3} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad M[2] = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & \frac{1}{3} & \frac{2}{3} & 0 \\ 0 & \frac{1}{3} & \frac{2}{3} & 0 \end{pmatrix}$$

$$M[3] = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \quad M[4] = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

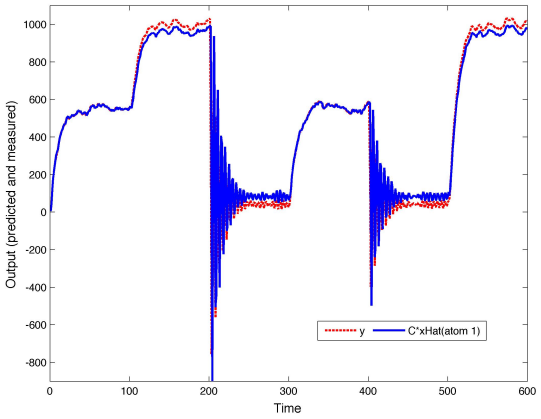
$$M[5] = \begin{pmatrix} 0 & 0 & 0 & 1 \\ \frac{1}{3} & 0 & \frac{2}{3} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Due to uniqueness of equilibria in linear systems, any erroneous identification of the current symbol is due purely to noise. The adaptive window-sizing was designed explicitly to mitigate the effects of this noise and thus the probability of a false identification in these systems was very near to zero. Because of this, the observation probability matrix N was taken to be the identity. We note that in a nonlinear setting in which different symbols may share common equilibria, erroneous identification could occur, leading to a non-trivial N.

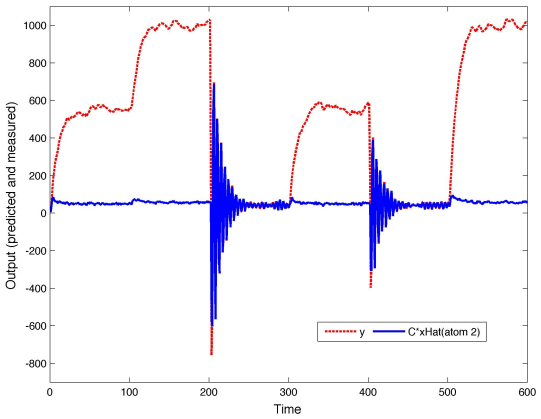
First simulation: Γ_2

In the first simulation the"true" plan was selected to be Γ_2. The measurement and estimated output from each of the three filters is shown in Fig. 2. Notice that the outputs of the estimators corresponding to σ_1 and σ_3 are similar while that of the σ_2 estimator is clearly different. The Kalman filters did a good job of modeling the output when the corresponding atom is being run.

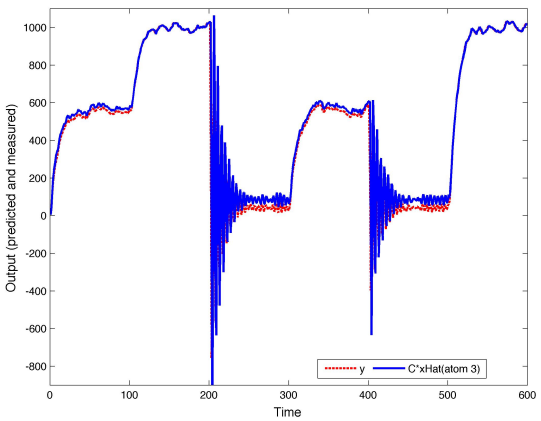
In Fig. 3 we show the output of the hypothesis tester and the window size determined for each identification. The true switching time is indicated with the solid black vertical lines and the time of a detected switch is indicated with a dashed vertical line (Fig. 3(a)). The window size used for estimation varied between one and five steps although a single measurement was often all that was needed for confident identification (Fig. 3(b)). Notice that in the first interval five measurements were needed; this is due to the fact that the output of the third symbol was also very close to the measured output and thus it was also likely to have given rise any of the in-



(a) Model assuming the plant executes σ_1



(b) Model assuming the plant executes σ_2



(c) Model assuming the plant executes σ_3

FIG. 2. *Output of each of the three filters (solid line) and the true measurement (dashed line). The actual plan being executed is* Γ_2*. Atom switches occur every 100 seconds.*

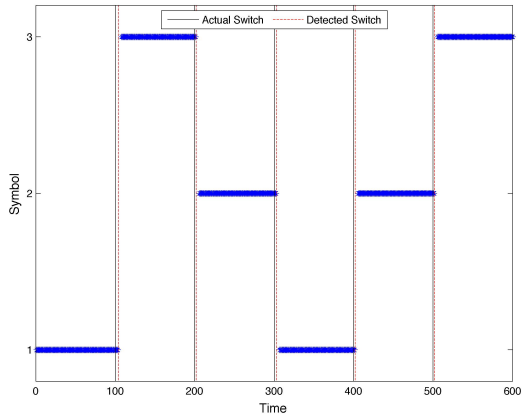
dividual observations. Because the measurement matrices were identity, we expected the probabilities over the set of plans to directly mirror the uniqueness of the plans. Thus, since all three plans were identical in the first segment, the probabilities were initially uniform. In the second step, however, only Γ_2 had σ_3 and thus the system correctly identified that as the current plan at that stage.

To illustrate the benefit of the adaptive window-sizing, in Fig. 4 we show the output of the $m-$ary hypothesis tester if only a single measurement is used to generate the probability distributions. Because the adaptive window sizes in segments two, four, and six were one, good performance was expected during those periods. The figure shows that in fact no erroneous identifications are made at those times. It is perhaps initially surprising that no false measurements are made during the first segment when the confidence intervals indicated that five measurements were needed to distinguish between atoms one and three. The $m-$ary tester, however, simply takes the most likely candidate and it is clear from Fig. 2 that the error corresponding to σ_1 is smaller. In fact the only erroneous identifications are made during the third time period when σ_2 is running. Considering the evolution shown in Fig 2, we surmise this is most likely due to the oscillatory decay of the outputs of the first and third atoms. The corresponding errors thus go through brief periods where they are very close to zero and a single-measurement $m-$ary tester has difficulty distinguishing between the different possibilities.

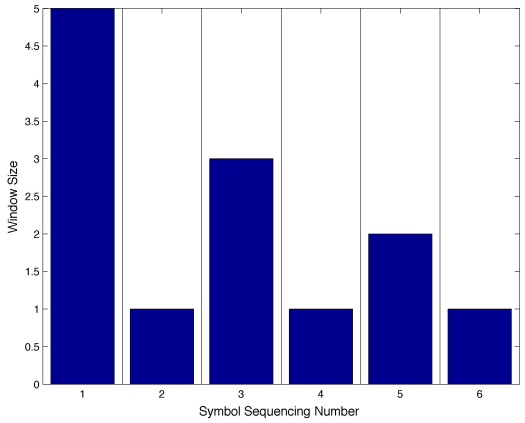
Second simulation: Γ_1

In the second simulation we executed the first plan. Because the first and third plans are identical until the final atom, we expected the probabilities of these two plans to each be $\frac{1}{2}$ from the second segment until the last segment at which Γ_1 should be selected as the correct plan. The measurement and estimated output from each of the three models is shown in Fig. 5. The dynamics were similar to those in the previous simulation. The results of symbol detection, adaptive window sizing, and plan detection are shown in Fig. 6.

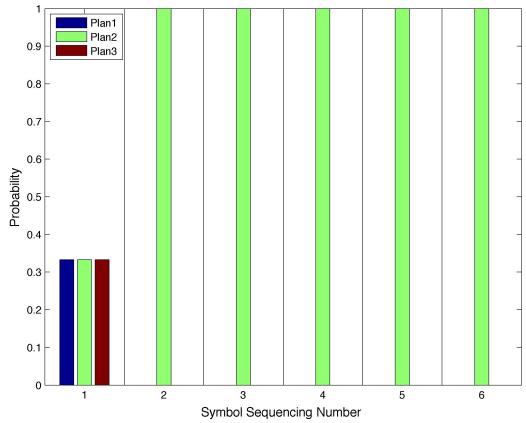
5. Summary. We have discussed the problem of inferring the symbolic inputs of a language-driven system based on observations of its output. This problem is related to that of "silent" cooperation between a group of mobile robots, where each new robot must determine what task the others are trying to accomplish before being able to help. We outlined an approach based on the notion of motion description languages, as advanced by Brockett. To identify the control symbols used to steer the plant and their switching times, we constructed one estimator per possible closed-loop dynamics, and compared the estimator outputs against the true plant output. Depending on the noise level in the system and on the alphabet of allowed atoms, determining the correct estimator may be difficult if only the error based on a single measurement is used. For that reason, we have introduced an adaptive procedure based on confidence



(a) Symbol detection



(b) Window sizes



(c) Probabilities over the plans

FIG. 3. *Results of the simulation when* Γ_2 *was executed.*

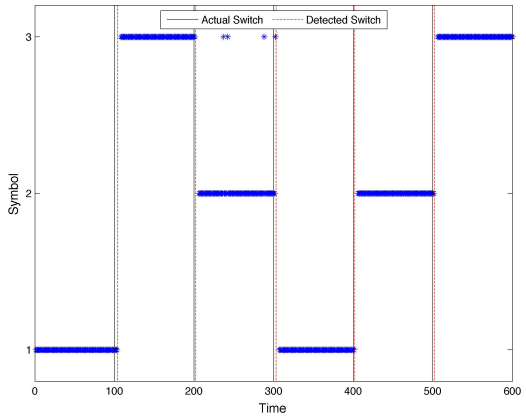


FIG. 4. *Output of the $m-$ary tester based on only a single measurement.*

ellipsoids for determining an appropriate number of measurements. The different estimators are compared using an $m-$ary hypothesis tester to determine the actual atom being executed by the plant. The interrupt condition that is associated from the switch can be inferred from the plant output near the time of a switch and the effect of noise mitigated by appealing to a Markov chain structure. This structure is also utilized to identify the particular plan being executed by the system from among a collection of possible plans. We provided details for the case of linear systems with Gaussian noise, and illustrated the performance of our method using simulations.

REFERENCES

[1] R. W. BROCKETT, *On the computer control of movement*, in: Proc. of the 1988 IEEE Conf. on Robotics and Automation, 1988, pp. 534–540.

[2] M. EGERSTEDT AND R. BROCKETT, *Feedback can reduce the specification complexity of motor programs*, IEEE Trans. Robotics and Automation, 48:2(2003), pp. 213–223.

[3] R. W. BROCKETT, *Formal languages for motion description and map making*, in: Robotics. American Mathematical Society, 1990, pp. 181–193.

[4] S. ANDERSSON AND D. HRISTU-VARSAKELIS, *Symbolic feedback control for navigation*, IEEE Transactions on Automatic Control, 51:6(2006), pp. 926–937.

[5] M. EGERSTEDT AND D. HRISTU-VARSAKELIS, *Observability and policy optimization for mobile robots*, in: Proc. of the 41st IEEE Conf. on Decision and Control, Dec. 2002, pp. 3596–3601.

[6] D. HRISTU-VARSAKELIS AND S. ANDERSSON, *Directed graphs and motion description languages for robot navigation and control*, in: Proc. of the IEEE Int. Conf. on Robotics and Automation, 2002, pp. 2689–2694.

[7] S. ANDERSSON AND D. HRISTU-VARSAKELIS, *Stochastic language-based motion control*, in: Proc. of the 42nd IEEE Conf. on Decision and Control, Dec. 2003, pp. 3313–8.

[8] E. FRAZZOLI, *Maneuver-based motion planning and coordination for multiple unmanned aerial vehicles*, in: Proc of the AIAA/IEEE Digital Avionics Systems Conference, 2002.

[9] V. MANIKONDA, P. S. KRISHNAPRASAD, AND J. HENDLER, *Languages, behaviors, hybrid ar-*

chitectures and motion control, in: Mathematical Control Theory, J. W. J. Baillieul, Ed. Springer, 1998, pp. 199–226.

[10] D. Hristu-Varsakelis, P. S. Krishnaprasad, S. Andersson, F. Zhang, P. Sodre, and L. D'Anna, *The MDLe engine: a software tool for hybrid motion control*, Institute for Systems Research, Tech. Rep. TR2000-54, Oct. 2000.

[11] D. Hristu-Varsakelis, M. Egerstedt, and P. S. Krishnaprasad, *On the structural complexity of the motion description language MDLe*, in: Proc. of the 42nd IEEE Conf. on Decision and Control, Dec. 2003, pp. 3360–5.

[12] A. Bicchi, A. Marigo, and B. Piccoli, *Feedback encoding for efficient symbolic control of dynamical systems*, IEEE Transactions on Automatic Control, 51:6(2006), pp. 1–16.

[13] E. Klavins, *A computation and control language for multi-vehicle systems*, in: Proc. of the 42nd IEEE Conference on Decision and Control, Dec. 2003, pp. 4133–9.

[14] E. Frazzoli, *Explicit solutions for optimal maneuver-based planning*, in: Proc. of the 42nd IEEE Conf. on Decision and Control, Dec. 2003, pp. 3372–77.

[15] E. Frazzoli, M. A. Dahleh, and E. Feron, *A maneuver-based hybrid control architecture for autonomous vehicle motion planning*, in: Software Enabled Control: Information Technology for Dynamical Systems, G. Balas and T. Samad, Eds. IEEE Press, 2003.

[16] ——, *Real-time motion planning for agile autonomous vehicles*, AIAA Journal of Guidance, Control, and Dynamics, 25:1(2002), pp. 116–129.

[17] ——, *Maneuver-based motion planning for nonlinear systems with symmetries*, IEEE Trans. on Robotics, 21:6(2005), pp. 1077–1091.

[18] M. Kloetzer and C. Belta, *A fully automated framework for control of linear systems from ltl specifications*, in: Hybrid Systems: Computation and Control: 9th International Workshop, ser. LNCS, J. Hespanha and A. Tiwari, Eds. Berlin: Springer-Verlgag, 2006, vol. 3927, pp. 333–347.

[19] P. Tabuada and G. J. Pappas, *Linear time logic control of discrete-time linear systems*, IEEE Transactions on Automatic Control, 51:12(2006), pp. 1862–1877.

[20] G. Fainekos, S. Loizou, and G. J. Pappas, *Translating temporal logic to controller specifications*, in: Proceedings of the 45^{th} IEEE Conference on Decision and Control, December 2006, pp. 899–904.

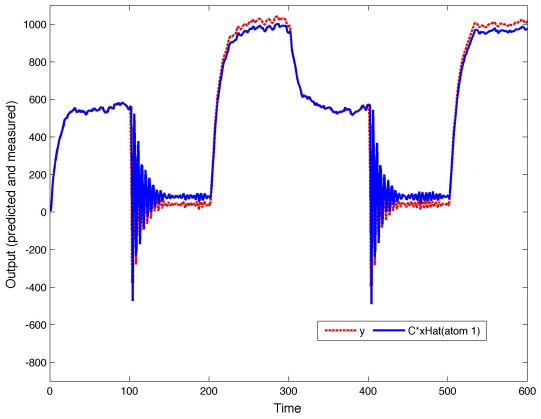
[21] R. W. Brockett, *The invertibility of dynamic systems with application to control*, Ph.D. dissertation, Case Western Reserve University, 1964.

[22] R. M. J.E. Hopcroft and J. Ullman, *Introduction to Automata Theory, Languages, and Computation (2nd Edition)*. Addison Wesley, Reading, MA, 2000.

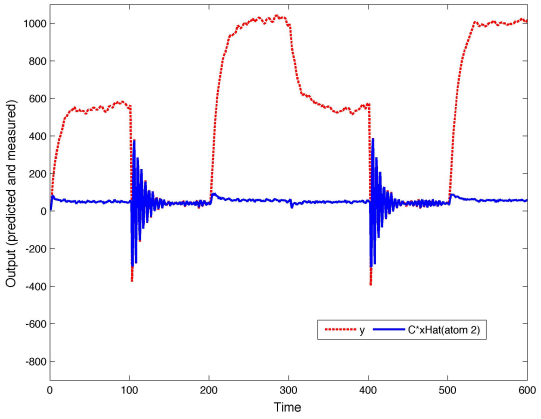
[23] H. V. Poor, *An introduction to signal detection and estimation, second edition*. New York, NY: Springer-Verlag, 1994.

[24] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, *A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking*, IEEE Transactions on Signal Processing, 50:2(2002), pp. 174–188.

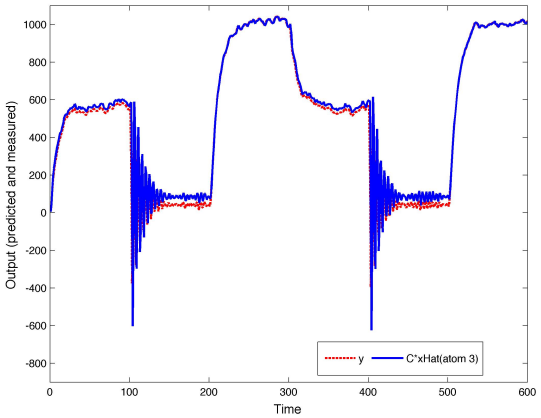
[25] T. Akutsu and D. Fukagawa, *Inferring a graph from path frequency*, in: Combinatorial Pattern Matching. Berlin-Heidelberg: Springer, 2005, vol. 3537, pp. 371–382.



(a) Model assuming the plant executes σ_1

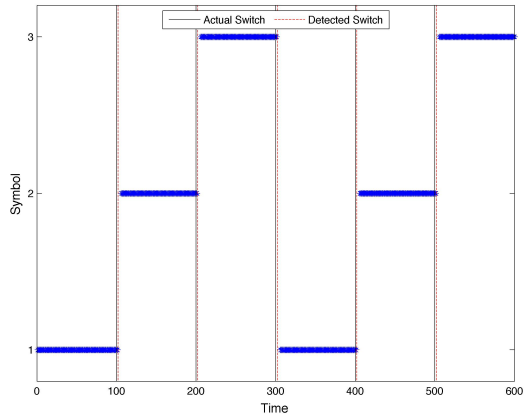


(b) Model assuming the plant executes σ_2

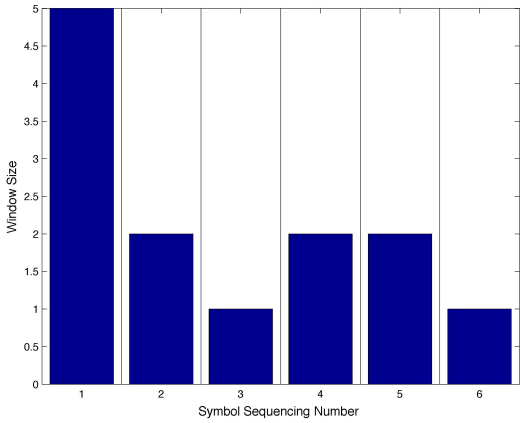


(c) Model assuming the plant executes σ_3

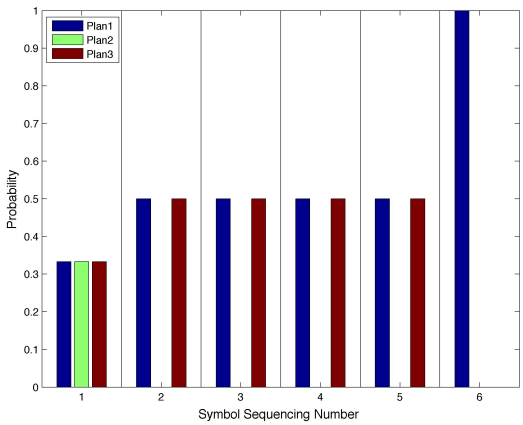
FIG. 5. *Output of each of the three estimators (solid line) and the true measurement (dashed line). The actual plan being executed is* Γ_1*. Atom switches occur every 100 seconds.*



(a) Symbol detection



(b) Window sizes



(c) Probabilities over the plans

FIG. 6. *Results of the simulation when* Γ_1 *was executed.*