Automatic design of discreet discrete filters

Jason M. O'Kane and Dylan A. Shell

***Abstract*— We address the problem of deciding what information a robot should transmit to the outside world, by exploring a setting where some information (e.g., current status of the task) must be shared in order for the robot to be useful, but where, simultaneously, we wish to impose limits which ensure certain information is never divulged. These sorts of conditions arise in several circumstances of increasing relevance: robots that can provide some guarantee of privacy to their users, controllers which safely use untrusted "cloud" services or smart-space infrastructure, or robots that act as inspection devices in information-sensitive contexts (e.g., factories, nuclear plants, etc.) We introduce an algorithm which takes as input an arbitrary combinatorial filter, expressed as a transition graph, and a set of constraints, constituting both upper and lower bounds, that specify the desired informational properties. The algorithm produces a coarser version of the input filter which possesses the desired informational properties, if and only if such a filter exists. We show that determining whether it is possible to satisfy both the distinguishablity and indistinguishablity constraints is NP-hard. The hardness result helps justify the worst-case running time of the algorithm. We describe an implementation of the algorithm along with empirical results showing that, beyond some minimum problem complexity, the algorithm is faster than naïve filter enumeration, albeit with greater memory requirements.**

I. Introduction

Every robot designer faces the problem of deciding how the information from sensors should be processed and stored by their robot. Any robot that responds to its environmental conditions inevitably discloses some information about its internal state or its estimates of the world's state; indeed, doing so often comprises a large part of the robot's purpose. As information is accumulated and integrated, this information is communicated via its choice of actions, status displays (e.g., lights blinking, or interfaces with visualizations of internal variables), or data logs that are written. But disclosure of such information has the potential to be detrimental. It may, for example, violate the privacy of individuals who have been interacting with the robot. In fact, guaranteeing that certain information will not be divulged may be a vital requirement for the robot. A nuclear inspection robot may be barred from entry to site by authorities if it leaks sensitive or proprietary information deemed to be unrelated to its arms control duties. This paper is concerned with rigorously treating questions regarding the sharing of information available to the robot.

To illustrate the general class of problems we address, Figure 1 depicts a somewhat whimsical scenario where a robot has constraints on what information it can divulge.

Jason M. O'Kane (jokane@cse.sc.edu) is with the Department of Computer Science and Engineering, University of South Carolina, Columbia, South Carolina, USA. Dylan A. Shell (dshell@cse.tamu.edu) is with the Department of Computer Science and Engineering, Texas A&M University, College Station, Texas, USA.

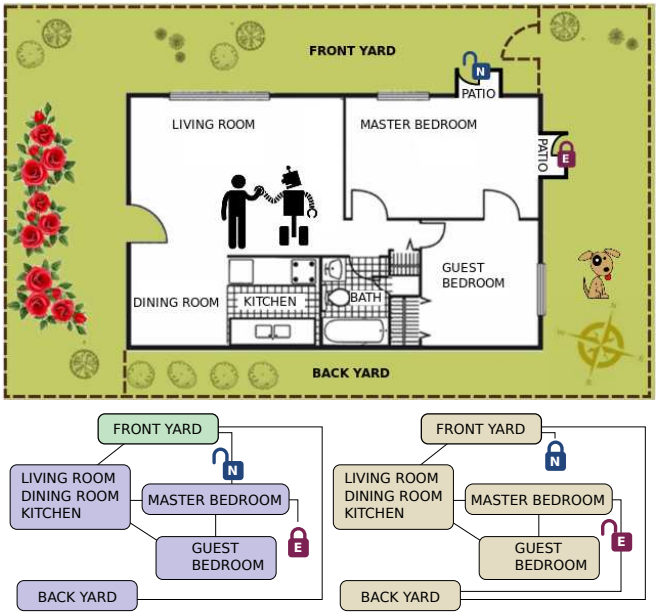

Fig. 1: **[Top]** A mobility impaired user is assisted by a robot escort. We wish to ensure that the robot never divulges information that could distinguish activities conducted in the master bedroom and guest bedrooms, respectively. However, the robot should make information available about the status of the rose garden. **[Bottom left]** This coloring of the robot's filter graph describes information that disambiguates states (and possibly belief states) which satisfy the information disclosure criteria. **[Bottom right]** If the northern patio door is locked and the east patio door unlocked, the coloring no longer serves its purpose. In fact, no controller satisfies the information criteria under those circumstances.

In the example, a helpful robot escort assists its mobility impaired user, horticultural enthusiast seedy John, around the home and garden. To be useful the robot localizes itself and keeps track of the room it occupies at all times. John waters his roses whenever he has opportunity to pass by them so, in order to avoid grievous over-watering, the robot notifies his (intermittent) garden service of the fact that the rosebushes have already been watered that day. It does this by publishing information about John's movement through the front yard. However, care must be taken to ensure that the robot does not divulge information that violates the privacy of the user when communicating with other entities (e.g., caretakers, remote technical support). To protect John's propriety, we guarantee that his nocturnal habits remain confidential by requiring that time spent in one bedroom be indistinguishable from time spent in the other. This miniature example, although twee, captures the essence of information sharing constraints and one may readily scale the description up to larger and more realistic problems.

The lower-left graph in Figure 1 shows the information maintained by the robot (a topological representation where transitions occur as the robot crosses a door/gate threshold).

The information constraints in this problem are satisfied — perhaps rather obviously— by divulging only the colors associated with the graph vertices. However, the existence of a suitable filter is not always so straightforward to determine. Constraints that dictate certain information should not be divulged do not merely imply that those states must have the same color; an astute adversary may disambiguate states in the original filter by examining sequences of colors. Moreover, seemingly minor modifications to the situation can have important consequences. Suppose, in the example, that the East patio door is unlocked (shown with purple the padlock), and the North patio door is sealed (shown with the blue padlock, see also the lower-right graph). Under these apparently innocuous changes, no suitable coloring exists.

We present an algorithm that accepts an input *combinatorial filter* (a discrete structure that encodes the information available to a robot) and a constraint graph specifying states within that filter which should be distinguishable and indistinguishable from one another. The algorithm outputs a coarser version of the original filter which satisfies the constraints, if only if such a filter exists. The approach works by constructing a graph representing what an adversary could soundly infer on the basis of observations of the filter's outputs. Then this observer graph is pruned to ensure the information constraints are met. Next, a coloring of the original filter is found by constructing a Boolean formula that must be satisfiable precisely when such a coloring exists. Any information transmitted —or actions selected— solely on information in the coloring will not expose knowledge that violates the given constraints. We also show that the problem this algorithm solves is NP-hard, somewhat excusing its inefficiency and, thereby, making use of a Boolean SAT solver defensible.

After reviewing related work in Section II, this paper first defines the constrained filtering problem in Section III in a general way. Then Section IV examines how an eavesdropping adversary can infer information and makes this notion precise. Sections V and VI prove the hardness result and detail the proposed algorithm, representing the main contributions of the paper, respectively. Section VII describes quantitative experiments that measure the performance of our algorithm, and the final section of the paper concludes by outlining some directions for future work.

II. Related Work

The objects of study in this paper are combinatorial filters and information-state graphs that are used for representing uncertainty and its evolution under a sequence of observations and actions. These representations go back a long way in robotics, at least to Erdmann and Mason [2], and Goldberg [3] who consider them for manipulation tasks. These filters were formalized in a general way by LaValle [8], [9]. Recent works that use combinatorial filters consider a wide a range of tasks, including target tracking [17], mobile robot navigation [10], [16], and manipulation [7].

Prior work by the authors has begun to explore algorithmic questions where combinatorial filters are treated as first-class entities. We tackle a related problem of *filter reduction* in [14] where, given an arbitrary filter, we seek an equivalent filter which uses the fewest information states to complete the same filtering task. In [15] we examine the active variant of the problem, proceeding along similar lines, in seeking the most compact plan able to perform a task. Unfortunately, merely minimizing the size of filter (or plan) does not result in one which preserves privacy. Exploring the outputs of these algorithms suggests that, rather than obfuscating state information, usually the algorithms help clarify underlying structure.

This work falls within the broader class of correct-by-design approaches wherein a formal specification of the controller requirements are provided beforehand and synthesis techniques, or verification techniques, or both are employed to guarantee the specifications are met (*cf.* e.g., [5], [6]).

Finally, earlier work by one of the authors [13] examines a related case where, in a specific geometric context, a robot may choose actions to increase its ignorance because everything the robot is aware of can potentially be divulged. In a sense, that work has every state colored uniquely, but the robot chooses its actions to maintain some property (of ambiguity) on those states. This paper considers only the passive case, but there is space to explore the combination of these ideas in future work by having some coloring of the filter that satisfies the distinguishablity constraints and almost satisfies the constraints, and then having the robot choose actions to maximize indistinguishablity.

III. Problem statement

This section formalizes our information disclosure constrained filter coloring problem.

A. I-state graphs and filters

We consider systems in which a robot interacts in discrete time with its environment by executing actions and receiving observations from its sensors. We assume that the robot uses the history of these actions and observations to form some representation of its knowledge about its current state and use the term *local information state (local I-state)* to denote that representation.[1]

In this paper, we are concerned only with *changes* to the local I-state. Notice that both actions that the robot takes and observations that it collects have the same impact, namely to induce a transition from one local I-state to another. Therefore, we adopt the generic term *event* to denote either an action or an observation. Each event corresponds to a discrete unit of information that becomes available to the robot during its execution. The robot experiences a sequence of events in discrete time, all drawn from an *event space* Y of finite size. Formally, we can model such a system as a directed graph [14].

[1]The term "I-state," used in this very general sense, was introduced by LaValle [8]. We use the modifier "local" to distinguish these from the observer I-states introduced below.

Definition 1: An I-state graph $\mathbf{G}$ *is a edge-labelled directed graph supplemented with a starting vertex,* i.e.*,* $\mathbf{G} \triangleq (V, E,\ l : E \to Y, v_0)$*, in which*

1) *the finite set* V *contains vertices which we call "I-states",*
2) *the set* E *consists of ordered pairs of vertices termed directed edges,*
3) *each edge is labelled with an event via the function* l*, and*
4) *the starting I-state is identified as* $v_0 \in V$*.*

In addition, no two edges originating from the same vertex may have the same label.

Under this kind of model, the set of edges outgoing from each local I-state corresponds to the set of events that can plausibly occur when the robot is in that I-state. Depending on the structure of the underlying problem, some events may never occur from some I-states. As a result, the vertices of an I-state graph may have out-degree less than $|Y|$.

Definition 2: An event sequence $y_0, \ldots, y_n$ *is* ***plausible*** *in an I-state graph* $\mathbf{G}$ *if there exists a path of* n *edges through* $\mathbf{G}$*, starting from* v_0 *and crossing edges labelled with those events in order.*

We assume that, at each time step, the robot divulges a single symbol, chosen without loss of generality from the set $\mathbb{N}^+$ of natural numbers and uniquely determined by the current local I-state. Following the tradition of graph theory, we informally refer to these symbols as *colors*.

Definition 3: A filter $\mathbf{F}$ *is an I-state graph supplemented with an assignment of colors to its vertices. That is,* $\mathbf{F} \triangleq (\mathbf{G}, c : V \to \mathbb{N}^+)$*, in which* $\mathbf{G}$ *is an I-state graph and the function* c *assigns a natural number to each I-state.*

We refer to the color assigned the current I-state as the *output* of the filter. At each time step, the robot divulges this output using an untrusted channel, visible to both trusted and untrusted agents alike.

B. Constraints positive and negative

The definition of a filter does not provide any guidance on how the coloring should be selected. Informally, the robot should choose a coloring that simultaneously ensures that its friends receive the information they need to coordinate appropriately, without divulging sensitive information to adversaries that may be observing that same channel. The next definitions formalize this requirement.

Definition 4: A constraint *on an I-state graph* $\mathbf{G} = (V, E,\ l : E \to Y, v_0)$ *is an unordered pair of distinct vertices* $\{v, w\} \subset V$*.*

Definition 5: A coloring c *of* $\mathbf{G}$ satisfies a constraint positively *if, for every plausible event sequence* $y_0, \ldots, y_n$ *leading to* v*, there does not exist any other plausible event sequence* $y'_0, \ldots, y'_n$ *that leads to* w*, and has the same colors at each step. That is, there exists no* y' *sequence leading to* w *such that* $c(y_i) = c(y'_i)$ *for all* $i \leq n$*.*

The intuition is that a positive constraint (v, w) requires that an observer who sees the sequence of colors output by the filter should never be confused about whether the robot's local I-state is v or w. Informally, a positive constraint can be viewed as a *lower bound* on the informative value of the filter's output.

Conversely, we can describe upper bounds on the information that can be inferred from the filter's output using negative constraints.

Definition 6: A coloring c *of* $\mathbf{G}$ satisfies a constraint negatively *if, for every plausible event sequence* $y_0, \ldots, y_n$ *leading to* v*, there exists another plausible event sequence* $y'_0, \ldots, y'_n$ *that leads to* w*, and has the same colors at each step. That is, there exists a* y' *sequence leading to* w *such that* $c(y_i) = c(y'_i)$ *for all* $i \leq n$*.*

A negative constraint requires a coloring that causes an observer never to be sure that the state is v but not w or *vice versa*. Put another way, every sequence of filter outputs that indicates that the robot may be at local I-state v also indicates that the robot may be at local I-state w. Informally, a negative constraint is a *upper bound* on the information communicated by the filter; it is a requirement that the resulting filter remain discreet about certain pieces of information.

C. A Basic Example

Next, we give a minimal example as a concrete illustration of the definitions. Suppose that we have a robot that inhabits a 2×2 grid world, capable of moving "Up", "Down," "Left," and "Right" and always fully aware of which grid cell it occupies. The I-state graph which describes the robot's state information is shown in the left subfigure of Figure 2. The start state s models an initial condition in which any of the four grid cells might be the robot's true start state. Assume that the robot wishes to communicate the column that it currently occupies but never to disclose information which can distinguish the correct row. This can be captured with constraints like the positive constraint $\{01, 10\}$ indicating that 01 is never to be mistaken with 10. Six constraints are needed in all and they are shown in graphical form on the right subfigure of Figure 2 where we denote a positive constraint with a solid edge between the two states involved. Similarly, a negative constraint is represented with a broken edge connecting both states.

This particular problem has a coloring with two colors that is a solution: both 00 and 01 are colored one color; 10 and 11 are colored the other; and s is colored arbitrarily.

D. Goals

With these definitions in place, we can finally state the problem that we address in the balance of the paper.

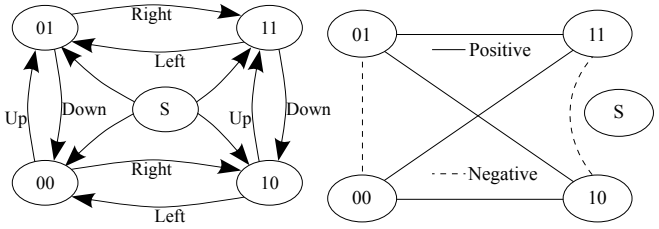

Fig. 2: A minimal example of robot living in a tiny grid world. **[Left]** An I-State graph describing the robot's position, the initial state "S" encodes uncertainty about its pose initially. **[Right]** A specification of information constraints on the robot's communication is neatly summarized via a constraint graph.

Problem: Constrained Filter Coloring (CFC)

Input: An I-state graph $\mathbf{G}$, a set of positive constraints $C^+ = \{(v_1^+, w_1^+), \ldots, (v_{n^+}^+, w_{n^+}^+)\}$ and a set of negative constraints $C^- = \{(v_1^-, w_1^-), \ldots, (v_{n^-}^-, w_{n^-}^-)\}$.

Output: A coloring of vertices of $\mathbf{G}$ that satisfies those constraints, or a statement that no such coloring exists.

IV. Observer I-state graphs

Definitions 5 and 6 directly describe the properties of the filters in which we are interested. However, because those definitions depend on the infinite set of all possible event sequences, it is not immediately clear how to design algorithms for CFC that satisfy those kinds of constraints. In this section, we describe an I-state graph called the *observer I-state graph* of a given filter, and provide alternatives to Definitions 5 and 6 that can be directly and efficiently verified in the observer I-state graph.

Definition 7: For a filter $\mathbf{F} = (\mathbf{G}, c)$, the observer I-state graph $\mathbf{I}$ *is defined as follows.*

1) *The vertex set is $V_I = 2^V - \{\emptyset\}$. Each element of this set, called an* observer I-state*, corresponds to a non-empty set of local I-states.*
2) *For each observer I-state $\eta_1 \in V_I$ and each color k in the range of c, the observer I-state graph contains an edge $\eta_1 \xrightarrow{k} \eta_2$, in which η_2 is the observer I-state corresponding to the set*

$$\bigcup_{v \in \eta_1} \{w \mid \mathbf{F} \text{ has an edge } v \longrightarrow w \text{ and } c(w) = k\},$$

if that set is not empty. If that set is empty, then η_1 has no out-edge with label k.
3) *The initial observer I-state is the vertex corresponding to the singleton set $\{v_0\}$.*

The intuition is that observers who see the output of $\mathbf{F}$ can treat those outputs as the events in an I-state graph of their own. For a given sequence of outputs from $\mathbf{F}$, the resulting observer I-state in $\mathbf{I}$ corresponds to the *set of possible local I-states* that are consistent with that output sequence. This form of worst-case reasoning about possible states is described in more detail in Chapter 11 of LaValle's book [8].

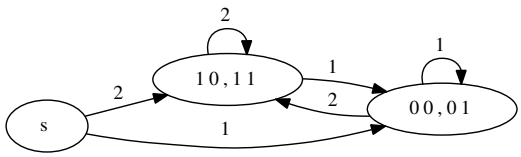

Fig. 3: The observer I-state graph for the example in Figure 2, under a coloring for which $c(00) = c(01) = 1$ and $c(10) = c(11) = c(s) = 2$.

Notice that every event sequence $y_0, \ldots, y_n$ in $\mathbf{F}$ leading to some local I-state v corresponds, by construction of $\mathbf{I}$, to a path in $\mathbf{I}$ crossing edges labeled $c(y_0), \ldots, c(y_n)$ that leads to an observer I-state containing v. Likewise, every plausible color sequence in $\mathbf{I}$ corresponds to one or more paths in $\mathbf{F}$ whose output matches those colors. Informally, this means that the observer I-state graph accurately tracks the sets of possible local I-states that match the observed outputs of $\mathbf{F}$. More directly, we can say that the observer I-state graph shows the sound inferences about the local I-state that can be made by observing the outputs of the filter.

These observations lead directly to the following connection between the observer I-state graph and positive and negative constraints in CFC problems.

Lemma 1: For an I-state graph $\mathbf{G}$ and coloring c of $\mathbf{G}$:

1) *Any positive constraint $\{v, w\}$ is satisfied if and only if no reachable observer I-state contains both v and w.*
2) *Any negative constraint $\{v, w\}$ is satisfied if and only if every reachable observer I-state contains neither v nor w, or both v and w.*

Both the hardness proof in the next section and the algorithm in Section VI rely heavily on this alternative view of the constraints.

Figure 3 shows an example observer I-state graph for the grid example described in Section III-C, under the constraint-satisfying 2-coloring described there.

V. Constrained Filter Coloring is NP-hard

In this section, we show that the CFC problem introduced in Section III-D is NP-hard. We proceed using the common approach of reduction from a known NP-complete problem, in this case a standard graph coloring problem:

Decision Problem: Graph 3-Coloring (GRAPH-3C)

Input: An undirected graph $\mathbf{G}$.

Output: *True* if there exists coloring of $\mathbf{G}$ using exactly 3 colors, such that no pair of adjacent vertices shares the same color; *False* otherwise.

This problem is known to be NP-complete [1]. Therefore, it suffices to show a polynomial time reduction from GRAPH-3C to CFC.

Given an undirected graph $\mathbf{G_1} \triangleq (V_1, E_1)$ as an instance of GRAPH-3C, we construct an instance of CFC with I-state graph $\mathbf{G_2} \triangleq (V_2, E_2, l, s)$ and constraint sets C^+ and $C^-\}$ as follows:

1) Create a start vertex in V_2 called s.

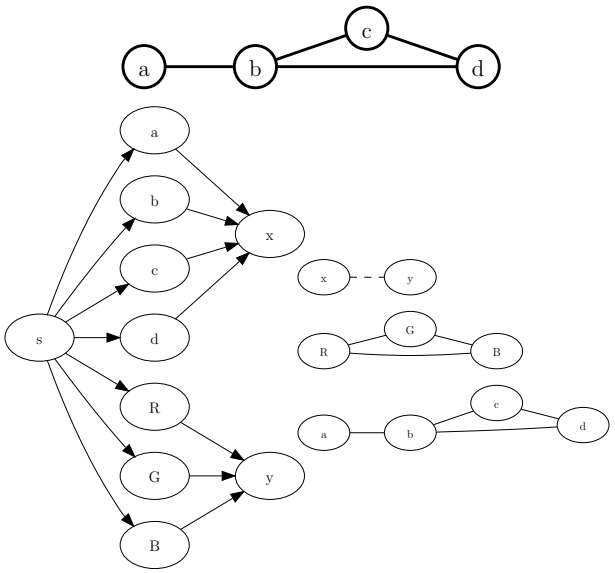

Fig. 4: **[Top]** An illustrative 4 vertex example instance of GRAPH-3C. **[Bottom left]** The CFC input graph for the corresponding problem instance. **[Bottom right]** The constraints for the corresponding instance of CFC form three separate graphs.

2) Add additional vertices x and y to V_2.
3) For each vertex in V_1, create a corresponding vertex in V_2. For each such vertex v, add edges $s \longrightarrow v$ and $v \longrightarrow x$ to E_2.
4) Create additional vertices in V_2 called $\{R, G, B\}$. For each $v \in \{R, G, B\}$, add edges $s \longrightarrow v$ and $v \longrightarrow y$ to E_2.
5) Arbitrarily assign a unique label $l(e)$ to each $e \in E_2$.
6) Define $C^+ = \{(R, G), (G, B), (B, R)\} \cup E_1$ for the positive constraints and $C^- = \{(x, y)\}$ for the negative ones.

An example of this construction is shown in Figure 4. The intuition is to use the information constraints to induce an observer I-state graph of special form, shown in Figure 5. The resulting graph has initial vertex $\{s\}$, followed by a fan-out to an intermediate (or "middle") set of vertices, which then fan back in to a single vertex $\{x, y\}$. The intermediate vertices encode the original coloring problem with positive constraints as follows. The $\{R, G, B\}$ vertices represent colors (intended as "red," "green," and "blue," respectively) associated with the graph $\mathbf{G_1}$; positive constraints ensure that R, G, and B are always kept separate from one another in the observer I-state graph. The vertices in $\mathbf{G_1}$ fall within this intermediate region, and are forced apart in the observer I-state graph by positive constraints that are added for each of edge in the original graph. The negative constraint ensures that each vertex the middle layer of the observer I-state graph contains at least one vertex of $\mathbf{G_1}$ and at least one of the $\{R, G, B\}$ vertices. Taken together, these constraints ensure that any constraint-satisfying coloring induces an observer I-state graph with exactly three vertices reachable in one step from the start.

We now formalize this idea, showing that the construction correctly produces an instance of CFC that is equivalent to the original GRAPH-3C instance.

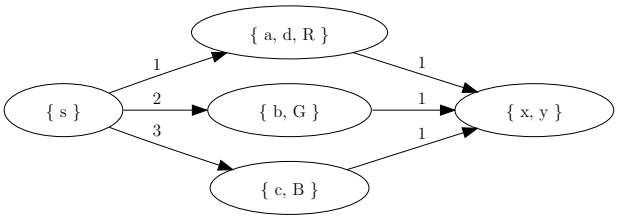

Fig. 5: The observer I-state graph for the problem in Figure 4.

Lemma 2: For any instance $\mathbf{G_1} \triangleq (V_1, E_1)$ *of* GRAPH-3C *for which the correct output is "True," the correct output for the* CFC *instance as described above is a coloring of the vertices of* $\mathbf{G_2}$ *satisfying* C^+ *and* C^-.

Proof: Suppose the contrary that $\mathbf{G_1}$ is 3-colorable with coloring $c_1 : V_1 \to \{0, 1, 2\}$, but that CFC of $\mathbf{G_2}$ determines that no coloring exists. Then consider the following coloring of V_2: Let $c_2 : v \mapsto 0$ for $v \in \{s, x, y\}$, $c_2 : R \mapsto 0$, $c_2 : G \mapsto 1$, $c_2 : B \mapsto 2$, and $c_2 : v \mapsto c(v), \forall v \in V_1$. Now examine the observer I-state graph $\mathbf{I_2}$ constructed from $(\mathbf{G_2}, c_2)$. The initial state of $\mathbf{I_2}$ is $\{s\}$, with edges labeled $\{0, 1, 2\}$ leading to three vertices labelled $\{R\} \cup \{v \in V_1 \mid c(v) = 0\}$, $\{G\} \cup \{v \in V_1 \mid c(v) = 1\}$, and $\{B\} \cup \{v \in V_1 \mid c(v) = 2\}$ respectively. Those "middle" three vertices each have an outgoing edge labelled 0 which connects to a vertex $\{x, y\}$.

Lemma 1 permits examination of the vertices of $\mathbf{I_2}$ to verify that the constraints hold for c_2. The constraints in C^+ involve only vertices in the "middle" of $\mathbf{I_2}$ and each is satisfied either by construction for $\{R, G, B\}$ or because c_2 collected vertices not connected by edges in E_1 by inheriting the coloring of c_1. The constraint in C^- is satisfied because x and y only appear together in $\{x, y\}$. This leads to a contradiction, completing the proof. ■

Lemma 3: For any instance $\mathbf{G_1} \triangleq (V_1, E_1)$ *of* GRAPH-3C *for which the correct output is "False," the correct output for the* CFC *instance as described above is that no satisfying coloring exists.*

Proof: Again suppose the contrary, that $\mathbf{G_1}$ is not 3-colorable but that a satisfying coloring for $\mathbf{G_2}$ is provided, say $c_2 : V_2 \to Y \subset \mathbb{N}^+$. Examining the observer I-state graph $\mathbf{I_2}$ constructed from $(\mathbf{G_2}, c_2)$, we observe that it must be of the following form: the initial state of $\mathbf{I_2}$ is $\{s\}$, connecting via edges labelled with $y \in Y$ to "middle" vertices $v \in (2^{\{R,G,B\} \cup V_1} - \{\emptyset\})$. These vertices all have an edge to a vertex $\{x, y\}$. The construction of $\mathbf{I_2}$ and Lemma 1 require that x and y appear together if C^- is to be satisfied.

The constraints on $\{R, G, B\}$ in C^+ (and Lemma 1) imply that there must exist at least three distinct "middle" vertices, one for each of R, G, and B. In fact, there must be exactly three "middle" vertices because if there is some other vertex it must be of the form $v' \in (2^{V_1} - \{\emptyset\})$. Then a sequence of observations y_0, y_1, y_2 with $y_1 = c_2(w)$ for some $w \in v'$ will distinguish x from y, hence violating C^-. Now construct coloring c_1 of $\mathbf{G_1}$ as follows:

$$c_1 : v \mapsto 0 \text{ for } v \in \{u \in V \mid c_2(u) = c_2(R)\}$$
$$c_1 : v \mapsto 1 \text{ for } v \in \{u \in V \mid c_2(u) = c_2(G)\}$$
$$c_1 : v \mapsto 2 \text{ for } v \in \{u \in V \mid c_2(u) = c_2(B)\}$$

Since c_2 satisfies C^+, no two $u, v \in V_1$ with $(u, v) \in E_1$ have $c_1(u) = c_1(v)$. Thus, $\mathbf{G_1}$ is 3-colored by c_1, yielding the contradiction. ∎

Finally, we must show that the reduction is time-efficient.

Lemma 4: The reduction from GRAPH-3C *to* CFC *described about takes time polynomial in the size of* $\mathbf{G_1}$.

Proof: The size of $\mathbf{G_2}$ is linear in the size of $\mathbf{G_1}$, and each element of $\mathbf{G_2}$ can trivially be constructed in constant time. Likewise, the size of C^+ is linear in the number of edges of $\mathbf{G_1}$, and each constraint is trivial to construct. Finally, C^- is constant. ∎

Importantly, the construction imposes structure that ensures the observer I-state graph is structured as desired, but the construction never forms the observer I-state graph—which may have size exponential in the size of $\mathbf{G_2}$—directly.

We now have all of the constituent parts needed to state our hardness result.

Theorem 1: CFC *is NP-hard.*

Proof: Combine Lemmas 2, 3, and 4. ∎

Note that, while this result shows that CFC is NP-hard, whether CFC is in NP—and, therefore, NP-complete—remains unproven. Indeed, it is not immediately clear that, given an instance of CFC and a coloring for the I-state graph, one can verify the correctness of that coloring in polynomial time, since the most direct way of performing that verification requires construction of the potentially exponential-size observer I-state graph.

VI. Algorithm description

This section describes an algorithm for CFC. Although its worst-case runtime is exponential in the size of the input graph—as it likely must be, in light of Theorem 1—the algorithm always produces the correct output, and does so much faster than direct brute force for many problem instances.

Two main ideas underlie the algorithm's operation. First, we use a "universal" observer I-state graph containing *every* I-state transition that the observer might make, across all colorings of the input graph. We then view the problem of satisfying the given constraints as a problem of selecting a coloring that removes enough edges from this graph to ensure that no path exists from the start to any observer I-state that violates any of the constraints. Section VI-A describes this graph in detail.

Second, to find a coloring of the filter that cuts the universal observer I-state graph in this way (or to show that no such coloring exists), our algorithm forms a Boolean formula that has a satisfying assignment if and only if the original filter has a coloring that satisfies the constraints. Moreover, filter coloring can be extracted directly from a satisfying assignment for this formula. After constructing this formula, our algorithm uses a highly optimized complete SAT solver to search for a solution. Details about this process appear in Section VI-B.

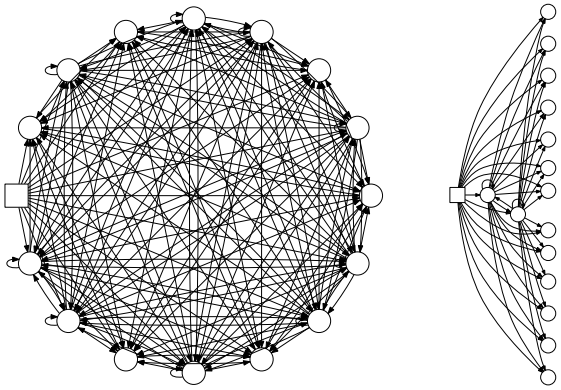

Fig. 6: **[Left]** The universal observer I-state graph for the I-state graph in Figure 2. Vertex labels are omitted due to space limitations. The start vertex is shown as a square (□); all other vertices are shown as circles (◯). **[Right]** The optimized universal observer I-state graph actually computed by our algorithm. It omits outgoing edges from observer I-states that violate the constraints.

A. The universal observer I-state graph

The first step of our algorithm is to construct a graph that contains all I-state transitions that an observer might make, across all colorings of the input filter.

Definition 8: For a given I-state graph $\mathbf{G} = (V, E, l, v_0)$, *the* universal observer I-state graph $\mathbf{U}$ *is an unlabeled directed graph.*

1) *Every distinct subset of the vertices of* $\mathbf{G}$ *corresponds to a vertex* η *of* $\mathbf{I}$. *We write* $V(\eta)$ *for the vertex set associated with* η.
2) *An edge* $\eta_1 \longrightarrow \eta_2$ *exists if and only if there* exists *some coloring of* $\mathbf{G}$, *under which an observer that knows the robot is in a state in* $V(\eta_1)$ *could receive some color as the filter output, and conclude that the robot might be in some state in* $V(\eta_2)$ *at the next step.*
3) *The start vertex* η_0 *of* $\mathbf{U}$ *corresponds to the singleton set* $\{v_0\}$.

Informally, we can think of $\mathbf{U}$ as a graph that collects all of the edges from all potential observer I-state graphs. The left portion of Figure 6 shows the universal observer I-state graph for the example of Section III-C.

1) Generating edges in $\mathbf{U}$*:* To find the out-edges of a given I-state η—without iterating over the exponentially many colorings of $\mathbf{G}$—we first compute the *successor set* of η:

$$S(\eta) = \bigcup_{v \in V(\eta)} \{w \mid \mathbf{F} \text{ has an edge } v \longrightarrow w\},$$

We then generate an edge in $\mathbf{U}$ from η to the vertex corresponding to each element in $2^{S(\eta)}$. This process follows directly from the definition of $\mathbf{U}$; each edge $\eta_1 \longrightarrow \eta_2$ generated in this way exists precisely when all of the vertices in $V(\eta_2)$ share a color amongst themselves, but they do not share a color with any other vertex in $S(\eta_1)$.

2) Separating η_0 *from constraint-violating I-states:* The value of $\mathbf{U}$ to our algorithm is that we can view the selection of a particular coloring of $\mathbf{G}$ as a deletion of the edges of

$\mathbf{U}$ that are not consistent with that coloring (and a labeling of each remaining edge the appropriate color). In addition, it is easy to identify the vertices that violate the given set of positive and negative constraints, by applying Lemma 1. These two facts, taken together, allow us to recast the original problem in the following way.

> *Choose a coloring of* $\mathbf{G}$ *that deletes enough edges from* $\mathbf{U}$ *to ensure that no path exists in* $\mathbf{U}$ *from its start vertex to any I-state that violates any positive or negative constraints.*

3) Optimizing $\mathbf{U}$*:* The universal observer I-state graph introduced in Definition 8 is sufficient for our algorithm, but it may contain some elements that are readily identified as useless to our algorithm. Therefore, the algorithm applies two simplifications as it constructs $\mathbf{U}$:

1) Since our algorithm is concerned only with paths in $\mathbf{U}$ starting from η_0, we construct only the portion of $\mathbf{U}$ that is reachable from η_0. That is, we use a forward search from η_0, instead of enumerating the powerset of V. This reduces the number of vertices in $\mathbf{U}$.
2) Since our algorithm is concerned only with paths that reach some observer I-state η that violates the constraints, we omit out-edges from any such observer I-state. This reduces that number of edges in $\mathbf{U}$.

The right portion of Figure 6 shows an example of the simplified $\mathbf{U}$ resulting from these optimizations.

B. Reduction to SAT

In the previous section, we argued that CFC can be solved by selecting a coloring that deletes at least one edge in $\mathbf{U}$ from every path between η_0 and any constraint-violating observer I-state. In this section, we show how to construct a Boolean expression f that is equivalent to this problem. Specifically, there exist Boolean values for the variables in f under which f evaluates to True, if and only if there exists a coloring of $\mathbf{G}$ that satisfies all of the positive and negative constraints of the problem instance.

We construct f from two distinct types of variables.

- For each pair (v_i, v_j) of distinct vertices in $\mathbf{G}$, we introduce a *same-color variable* c_{ij}. This variable is intended to have the value True if and only if v_i and v_j share the same color in the final filter.
- For each edge $\eta_1 \to \eta_2$ in $\mathbf{U}$, we introduce an *edge-exists variable* $e_{\eta_1\eta_2}$. This variable is intended have the value True if and only if the coloring defined by the same-color variables generates an observer I-state graph in which this edge exists.

Based on these variables, our algorithm constructs f as a conjunction of three different kinds of subexpressions.

- First, we generate subexpressions that force the same-color variables to represent a legitimate coloring. Specifically, they must encode an equivalence relation on V. To force this relation to be symmetric, we include subexpressions for each vertex pair v_i, v_j of this form:

$$c_{ij} = c_{ji}$$

  To force the same-color relation to be transitive, we include subexpressions of this form for every trio of distinct vertices:

$$c_{ij} \text{ and } c_{jk} \to c_{ik}$$

  Finally, an equivalence relation must also be reflexive, so for each vertex v_i of $\mathbf{F}$, we include a subexpression

$$c_{ii} = \mathrm{True}$$

- Next, we generate subexpressions that establish the connection between the same-color variables and the edge-exists variables. Recall from Section VI-A.1 that an edge $\eta_1 \longrightarrow \eta_2$ exists in the observer I-state graph when all of the vertices in $V(\eta_2)$ share the same color, and no other vertices in $S(\eta_1)$ share that same color. We express that directly as a subexpression in f by choosing an arbitrary element $v_i \in V(\eta_2)$ and comparing its colors to those of the other vertices in $V(\eta_2)$ and $S(\eta_1)$:

$$e_{\eta_1\eta_2} = \left(\begin{array}{c} \underbrace{c_{ij_1} \text{ and } \cdots \text{ and } c_{ij_m}}_{v_j \in V(\eta_2)} \\ \text{and} \\ \underbrace{\neg c_{ij_1} \text{ and } \cdots \text{ and } \neg c_{ij_m}}_{v_j \in S(\eta_1) - V(\eta_2)} \end{array} \right)$$

- Finally, we add a collection of subexpressions that require that least one edge from each path $\mathbf{U}$ between η_0 and a constraint-violating I-state to be removed. We use a depth-first search on $\mathbf{U}$ to enumerate such paths. For each path

$$\eta_0 \to \eta_1 \to \cdots \to \eta_m,$$

  we generate a subexpression

$$\neg e_{\eta_0\eta_1} \text{ or } \cdots \text{ or } \neg e_{\eta_{m-1}\eta_m},$$

  which evaluates to True if and only if that path loses at least one if its edges.

After generating all of these subexpressions, our algorithm combines them into a single Boolean expression f using conjunctions, and solves the resulting instance of the Boolean satisfiability problem.

Given an assignment that satisfies f, we can extract a coloring for $\mathbf{G}$ in a straightforward way:

1) Select a vertex v_I of $\mathbf{G}$ for which no color is assigned yet. Assign the next unused color k to it.
2) Find all other vertices v_j for which the variable c_{ij} is True. Assign color k to each v_j.
3) Repeat until every vertex of $\mathbf{G}$ has been colored.

If there is no assignment that satisfies f, then there is no coloring of $\mathbf{G}$ that satisfies the constraints.

The decision to convert our problem instance to an instance of SAT may appear, on the surface, rather counterintuitive, since SAT is a well-known (and, indeed, was the first known) NP-hard problem [1]. However, this also represents an advantage: There has been extensive research on fast solvers for the SAT problem, to the extent that many

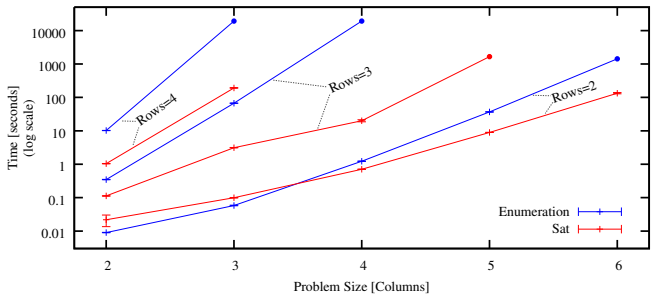

Fig. 7: A comparison of the running-times for the proposed SAT-based algorithm and a naïve enumeration method for a robot moving in grids of various sizes. Data points with error-bars represent the mean and standard deviations of 30 measurements, circles were (tedious) instances terminated after a single measurement. The proposed algorithm is faster beyond some minimum problem complexity but, nevertheless, computation time increases significantly with problem instance size for both methods. Note that the ordinate axis has a logarithmic scale.

problems—including, we believe, CFC—can often be solved more rapidly by state-of-the-art SAT solvers than by domain specific heuristics directly [4], [12].

VII. Comparison to brute force enumeration

We implemented the algorithm described in the preceding section using python, making use of the `bool2cnf` utility and `zChaff` implementation of the Chaff algorithm [11] for the Boolean satisfiability elements. For purposes of comparison, we also wrote a naïve method that directly enumerates all possible colorings of its input filter, generating the associated observer I-state graph and testing whether the constraints are satisfied for each. All executions in this section were performed on a GNU/Linux laptop using a single core of a 2.53GHz Intel Core 2 Duo processor.

We considered generalizations of the 2×2 scenario in Figure 2 to an $m \times n$ case, where the constraints are analogous, *viz.*, positive constraints to divulge columns and negative constraints to obscure rows. We varied both m and n, measuring the run-time for both algorithms. Figure 7 summarizes the findings of the experiments. For all but the smallest problem instances, the proposed algorithm is faster than naïve enumeration. It should be said, however, that the primary limiting resource of the method we propose is its memory use. This suggests that the practitioner can choose to trade between time and space resources.

VIII. Summary and Conclusion

We address the question of how one might design a robot controller subject to discreetness constraints on the information it provides to the outside world. We introduce a formulation of the problem based on combinatorial filters, which are discrete structures for encoding the information available to a robot. We allow the designer to specify both the information needed to perform a task (i.e., the robot *must* share information that distinguishes certain circumstances) and information that may not be divulged (i.e., information should *never* allow a nefarious agent to disambiguate two states). If these requirements conflict, there may be no suitable controller satisfies all constraints; this fact, however, is not always immediately apparent.

We have shown that finding a filter which satisfies informational constraints is not likely to be efficiently solvable, at least not exactly. The exact method we propose appears to have a running time which is lower than straightforward enumeration for all but the tiniest problem instances. Future work should examine whether CFC is in NP. We speculate that it is not. Work could also explore whether there are meaningful approximations to the problem, ideally ones which do not sacrifice guarantees on information that is not to be divulged (*i.e.*, negative constraints). As alluded to in Section II, one direction is to consider active rather than passive variants of the problem.

Acknowledgments This material is based upon work supported by the National Science Foundation under Grant No. IIS-0953503, and IIS-1302393. Interest in information leakage by robots was piqued by early discussions with Keith O'Hara.

References

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. The MIT Press, 2001.

[2] M. Erdmann and M. T. Mason, "An exploration of sensorless manipulation," *IEEE Transactions on Robotics and Automation*, vol. 4, no. 4, pp. 369–379, Aug. 1988.

[3] K. Y. Goldberg, "Orienting polygonal parts without sensors," *Algorithmica*, vol. 10, pp. 201–225, 1993.

[4] C. P. Gomes, H. Kautz, A. Sabharwal, and B. Selman, "Satisfiability solvers," in *Handbook of Knowledge Representation*, F. van Harmelen, V. Lifschitz, and B. Porter, Eds. Elsevier, 2008, ch. 2, pp. 89–133.

[5] M. Kloetzer and C. Belta, "A fully automated framework for control of linear systems from temporal logic specifications," *IEEE Transaction on Automatic Control*, vol. 53, no. 1, pp. 287–297, 2008.

[6] H. Kress-Gazit, T. Wongpiromsarn, and U. Topcu, "Correct, Reactive Robot Control from Abstraction and Temporal Logic Specifications," *IEEE Robotics and Automation Magazine*, vol. 18, no. 3, pp. 65–74, Sep. 2011.

[7] S. Kristek and D. Shell, "Orienting deformable polygonal parts without sensors," in *Proc. International Conference on Intelligent Robots and Systems*, 2012.

[8] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006, available at http://planning.cs.uiuc.edu/.

[9] ——, "Sensing and filtering: A fresh perspective based on preimages and information spaces," *Foundations and Trends in Robotics*, vol. 1, no. 4, pp. 253–372, 2010.

[10] R. Lopez-Padilla, R. Murrieta-Cid, and S. M. LaValle, "Optimal gap navigation for a disc robot," in *Proc. Workshop on the Algorithmic Foundations of Robotics*, 2012.

[11] M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang, and S. Malik, "Chaff: Engineering an efficient SAT solver," in *Proc. Design Automation Conference*, 2001.

[12] O. Ohrimenko, Stuckey, P. J., and M. Codish, "Propagation = Lazy Clause Generation," in *Principles and Practice of Constraint Programming—CP*, ser. Lecture Notes in Computer Science, 2007, vol. 4741, pp. 544–558.

[13] J. M. O'Kane, "On the value of ignorance: Balancing tracking and privacy using a two-bit sensor," in *In Proc. International Workshop on the Algorithmic Foundations of Robotics*, 2008.

[14] J. M. O'Kane and D. A. Shell, "Automatic Reduction of Combinatorial Filters," in *Proc. IEEE International Conference on Robotics and Automation*, 2013.

[15] ——, "Finding concise plans: Hardness and algorithms," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Sytems*, 2013.

[16] B. Tovar, R. Murrieta-Cid, and S. M. LaValle, "Distance-optimal navigation in an unknown environment without sensing distances," *IEEE Transactions on Robotics*, vol. 23, no. 3, pp. 506–518, Jun. 2007.

[17] J. Yu and S. M. LaValle, "Shadow information spaces: Combinatorial filters for tracking targets," *IEEE Transactions on Robotics*, vol. 28, no. 2, pp. 440–456, 2012.