requirement on $\mathrm{S}_D(y,d)$ is

$$(y,d,y') \in \mathrm{S}_D \text{ and } (y,d,y'') \in \mathrm{S}_D \implies y' = y''.$$

Proof: Since $\nabla\big|_F^D$ is left-total, every string in $\mathscr{L}(F)$ maps to a non-empty set of sequences. The map preserves sequence length. Then, we show that if $y_0y_1\dots y_m \nabla\big|_F^D x_0x_1\dots x_m$ and $y'_0y'_1\dots y'_m \nabla\big|_F^D x_0x_1\dots x_m$ then each $y_i = y'_i$. Since, $y_0 = x_0 = y'_0$, apply $y_{i+1} = \mathrm{S}_D(y_i, x_{i+1}) = y'_{i+1}$ inductively. No conflation of input occurs, so no information is lost. While intuitive, it remains to show that the necessary translation can be effected by a finite-state sensori-computational device operating symbol-by-symbol. (We revisit the explicit construction below once some algorithms have been presented.) ■

Placing stronger constraints solely on the variator, we obtain another sufficient condition:

Proposition 12. For (D, S_D), if for every $y, y' \in Y(F)$, there exists a unique $d \in D$ so $y' = \mathrm{S}_D(y,d)$ then Question 1's answer is affirmative.

Proof: One easily establishes that $\nabla\big|_F^D$ is left-total. ■

The two previous propositions, while straightforward 'closed-form' sufficient conditions, make demands which seldom hold for realistic sensors. For instance, S_D may be multi-valued in order to model noise. The complete answer for any S_D, deferred until Section IV-B, does appear in Lemma 17.

But first, a camera as an example is, of course, long overdue.

Example 5 (single-pixel camera)**.** A single-channel, 8-bit, single-pixel camera is a device that returns reading in the range $Y = \{0,\dots,255\}$ at any point in time. We might model such a camera via a sensori-computational device F_{cam} that does no state-based computation: have it use 256 vertices $V(F_{\mathrm{cam}}) = \{v_0,\dots,v_{255}\}$, and outputs $C_{\mathrm{cam}} = \{o_0,\dots,c_{255}\}$, so that $c(v_i) = \{o_i\}$, and transitions which consider only the last value $\tau_{\mathrm{cam}}(v_i,v_j) = \{j\}$.

For the observation variator, we again can use standard integers $(\mathbb{Z},+)$. Now we restrict to the subset of triples where the first and third slots only have elements within $\{0,\dots,255\}$; the second slot then clearly only has elements $\{-255,\dots,255\}$.

The device F_{cam} with variator $(\mathbb{Z},+)$ (or the restriction described) is $\nabla\big|_F^D$-simulatable because a derivative F'_{cam} can be constructed for it. □

The integers used to model observation changes for the pixel intensities differ from those with the compass bearings: that is, the elements that remain after $+\colon \mathbb{Z}\times\mathbb{Z}\to\mathbb{Z}$ is restricted in Example 5 and Example 4 have different structure. We shall revisit this.

Rather more obviously, a single-pixel camera has only limited applicability. Next, we consider how to scale up.

A. Modeling complex sensors

We start with a definition that allows aggregation of output variators.

Definition 13 (direct product variator)**.** Given (D_1, S_{D_1}) as an output variator for F_1, and (D_2, S_{D_2}) for F_2, then the set $D_{1\times 2} = D_1 \times D_2$ and $\mathrm{S}_{D_{1\times 2}}$ defined via

$$\big((y_1,y_2),(d_1,d_2),(y'_1,y'_2)\big) \in \mathrm{S}_{D_{1\times 2}} \Longleftrightarrow (y_1,d_1,y'_1) \in \mathrm{S}_{D_1} \wedge (y_2,d_2,y'_2) \in \mathrm{S}_{D_2} \quad (1)$$

is an observation variator for $F_1 \times F_2$, and is termed the *direct product variator*.

Example 6 (iRobot Create bump sensors)**.** Recalling the iRobot Create of Example 1, these robots have left and right bump sensors, both of which provide binary values (obtained via Sensor Packet ID: #7 (bits 0 and 1, right and left, respectively, encoding '`0` = no bump, `1` = bump' [12, pg. 22]). As these are binary streams, just like the wall sensor, they can each be transformed with variator $(D_2 = \{\bot,\top\}, \mathrm{S}_{D_2})$ that tracks bit flips. And to track both, one constructs $\{\bot,\top\}\times\{\bot,\top\}$, and the ternary relation $\mathrm{S}_{D_{2\times 2}}$. In this way, a $d_i = (\bot,\bot)$ would indicate that neither bump sensor's state has changed since previously. □

Example 7 (1080p camera)**.** A conventional 1080p camera has 3 channels at a resolution of 1920 × 1080. Using the F_{cam} of Example 5, apply the direct product (of Definition 3) to form an aggregate device, and the direct product (of Definition 13) to the variator $(\mathbb{Z},+)$. The triple relation is large (with 4.0×10^{11} elements). □

In order for this camera, assembled from the single-pixel ones, to not be unwieldy we must show how output simulation also aggregates. We do this in Proposition 15, but need the following definition first.

Definition 14 (length compatible relations)**.** A pair of relations, R_1 and R_2, each on sets of sequences $\mathrm{R}_1 \subseteq A \times B$ (with $A \subseteq {\Sigma_a}^*$, $B \subseteq {\Sigma_b}^*$), $\mathrm{R}_2 \subseteq C \times E$ (with $C \subseteq {\Sigma_c}^*$, $E \subseteq {\Sigma_e}^*$) are *length compatible* if for every $a \in A$ and $c \in C$ with the same length (i.e., $|a| = |c|$), there exists some b and e of identical length ($|b| = |e|$) with $a\,\mathrm{R}_1\,b$ and $c\,\mathrm{R}_2\,e$.

In other words, when we consider the image of any sequence under R_1, along with the image of any sequence of identical length under R_2, both contain *some* pair of common-lengthed sequences.

We now express the property of interest.

Proposition 15 (product output simulation)**.** Given sensori-computational devices F_1 and F_2 which are output simulatable modulo length compatible relations $\mathrm{R}_1 \subseteq A \times B$ and $\mathrm{R}_2 \subseteq C \times E$, respectively, then device $F_1 \times F_2$ is output simulatable modulo relation $\mathrm{R}_{1\times 2}$ defined via:

$$\begin{gathered}\big((a_1,c_1)\dots(a_n,c_n)\big)\,\mathrm{R}_{1\times 2}\,\big((b_1,e_1)\dots(b_m,e_m)\big)\\ \Updownarrow \\ (a_1\dots a_n)\,\mathrm{R}_1\,(b_1,\dots,b_m) \wedge (c_1\dots c_n)\,\mathrm{R}_2\,(e_1,\dots,e_m).\end{gathered} \quad (2)$$

Proof: There must exist G_1 and G_2 with $G_1 \sim F_1 \pmod{\mathrm{R}_1}$, and $G_2 \sim F_2 \pmod{\mathrm{R}_2}$. Then if we form $G_1 \times G_2$, we see $G_1 \times G_2 \sim F_1 \times F_2 \pmod{\mathrm{R}_{1\times 2}}$, which is verified via the conditions of Definition 5; if $s_1s_2\dots s_n \in \mathscr{L}(F_1 \times F_2)$, then each $s_i = (a_i,c_i)$ with $a_1a_2\dots a_n \in \mathscr{L}(F_1)$, and $c_1c_2\dots c_n \in \mathscr{L}(F_2)$. 1) For all such $s_1s_2\dots s_n$ then there exist $b_1b_2\dots b_m \in \mathscr{L}(G_1)$ and $e_1e_2\dots e_m \in \mathscr{L}(G_2)$, of identical length, with $a_1a_2\dots a_n \,\mathrm{R}_1\, b_1b_2\dots b_m$ and with $c_1c_2\dots c_n \,\mathrm{R}_2\, e_1e_2\dots e_m$. Then sequence $t_1t_2\dots t_m$, where for each $k \in \{1,\dots,m\}$ we take $t_k = (b_k,e_k)$, is $s_1s_2\dots s_n \,\mathrm{R}_{1\times 2}\, t_1t_2\dots t_m$. As required, $t_1t_2\dots t_m \in \mathscr{L}(G_1 \times G_2)$ because neither $u_1u_2\dots u_m$ crashes on G_1, nor $v_1v_2\dots v_m$ on G_2, and simply pairing the respective states visited in sequence gives the states visited in $G_1 \times G_2$. 2) For any $(b_1,e_1)(b_2,e_2)\dots(b_m,e_m) \in \mathscr{L}(G_1 \times G_2)$ with $(a_1,c_1)(a_2,c_2)\dots(a_n,c_n)\,\mathrm{R}_{1\times 2}\,(b_1,e_1)(b_2,e_2)\dots(b_m,e_m)$, the $\mathscr{C}(G_1 \times G_2,(b_1,e_1)(b_2,e_2)\dots(b_m,e_m)) = \mathscr{C}(G_1,b_1b_2\dots b_m) \times \mathscr{C}(G_2,e_1e_2\dots e_m)$. But $\mathscr{C}(G_1,b_1\dots b_m) \subseteq \mathscr{C}(F_1,a_1\dots a_n)$, and, similarly, $\mathscr{C}(G_2,e_1\dots e_m) \subseteq \mathscr{C}(F_2,c_1\dots c_n)$. Hence, $\mathscr{C}(G_1,b_1\dots b_m) \times \mathscr{C}(G_2,e_1\dots e_m) \subseteq \mathscr{C}(F_1,a_1\dots a_n) \times \mathscr{C}(F_2,c_1\dots c_n) = \mathscr{C}(F_1 \times F_2,(a_1,c_1)(a_2,c_2)\dots(a_n,c_n))$. ■

Corollary 16. Given sensori-computational devices F_1 and F_2, with variators (D_1, S_{D_1}) and (D_2, S_{D_2}), a sufficient condition for direct product $F_1 \times F_2$, with direct product variator $(D_{1\times 2}, \mathrm{S}_{D_{1\times 2}})$, to possess a derivative is that F_1 and F_2 do.

Proof: If we know $F_1' \sim F_1 \left(\bmod \nabla\big|^{D_1}_{F_1}\right)$ and also that $F_2' \sim F_2 \left(\bmod \nabla\big|^{D_2}_{F_2}\right)$, then Proposition 15 will give the result that $F_1' \times F_2' \sim F_1 \times F_2 \left(\bmod \nabla\big|^{D_{1\times 2}}_{F_1\times F_2}\right)$ provided we establish two facts. First, that $\nabla\big|^{D_1}_{F_1}$ and $\nabla\big|^{D_2}_{F_2}$ are length compatible. Definition 10 implies (the stronger fact) that delta relations preserve length (i.e., whenever $a \nabla\big|^{D}_{F} b$, then $|a| = |b|$), so length compatibility follows. Secondly, that if $\mathrm{R}_1 = \nabla\big|^{D_1}_{F_1}$ and $\mathrm{R}_2 = \nabla\big|^{D_2}_{F_2}$, then $\mathrm{R}_{1\times 2}$ as given in (2) is $\nabla\big|^{D_{1\times 2}}_{F_1\times F_2}$, i.e., the following does indeed commute:

$$\begin{tikzcd}
(D_1, \mathrm{S}_{D_1}) \arrow[d] & , (D_2, \mathrm{S}_{D_2}) \arrow[r, "\text{Def. 13 via (1)}"] \arrow[d, "\text{Def. 10}"'] & (D_{1\times 2}, \mathrm{S}_{D_{1\times 2}}) \arrow[d, "\text{Def. 10}"] \\
\nabla\big|^{D_1}_{F_1} & , \nabla\big|^{D_2}_{F_2} \arrow[r, "\text{Prop. 15 via (2)}"'] & \nabla\big|^{D_{1\times 2}}_{F_1\times F_2}
\end{tikzcd}$$

Simply following the definitions:
$((y_0,z_0)(y_1,z_1)\dots(y_n,z_n))\,\nabla\big|^{D_{1\times 2}}_{F_1\times F_2}\,((y_0,z_0)(d_1,u_1)\dots(d_n,u_n))$ with $(y_k,z_k) = \mathrm{S}_{D_{1\times 2}}((y_{k-1},z_{k-1}),(d_k,u_k))$, from Def. 10. But, from (1), $y_k = \mathrm{S}_{D_1}(y_{k-1},d_k)$ and $z_k = \mathrm{S}_{D_2}(z_{k-1},u_k)$; so $y_0y_1\dots y_n \nabla\big|^{D_1}_{F_1} y_0d_1,\dots,d_n$ and $z_0z_1\dots z_n \nabla\big|^{D_2}_{F_2} z_0u_1,\dots,u_n$. ■

The proof of Corollary 16 and the foundation it builds upon, Proposition 15, construct the output simulating device via a direct product.

To summarize, we may compose sensori-computational devices to give a more complex aggregate device, and also compose output variators to give a composite variator. Question 1 for the aggregate device, can be answered in the affirmative by consider the same question for the individual devices.

B. Answering Question 1

In most of the examples we have considered, the argument for the existence of an output simulating sensori-computational device has been on the basis of the fact that the variator allows complete recovery of the original stream. Indeed, being based on the same reasoning, conditions like those in Propositions 11 and 12 are rather blunt. Remark 6 has already mentioned that some specific two-observation subsequences might never appear in any string in $\mathscr{L}(F)$, thus a set D smaller than one might naïvely expect may suffice. Even beyond this, and perhaps more crucially in practice, full reconstruction of the $y_0y_1\dots y_m$ from $y_0d_1d_2\dots d_m$ may be unnecessary as whole subsets of $\mathscr{L}(F)$ may produce the same or a compatible output, some element of $\mathscr{C}(F,y_0y_1\dots y_m)$.

Hence, to answer an instance of Question 1 when some input streams are allowed to be collapsed by $\nabla\big|^{D}_{F}$, one must examine the behavior of the specific sensori-computational device at hand. But to provide an answer for a given (D, S_D), a computational procedure cannot enumerate the possibly infinite domain of $\nabla\big|^{D}_{F}$. We give an algorithm for answering the question; the procedure is constructive in that it produces a derivative of F if and only if one exists.

Question 1 (Constructive version). Given F with variator (D, S_D), find an F' such that $F' \sim F \left(\bmod \nabla\big|^{D}_{F}\right)$ if one exists, or indicate otherwise.

In this case, we shall additionally assume that the set D is finite.

The procedure is given in Algorithm 1. It operates as follows: it processes input F to form an F'' (in lines 1–16) by copying and splitting vertices so each incoming edge has a single label. This F'' is processed to compute the differences between two consecutive labels (lines 18–30) and the results are placed on edges of F'. If it fails to convert any pair of consecutive labels, then it fails to compute the change of some string in F and reports 'No Solution'. In the above step, after computing the changes, a single string can arrive at multiple vertices in F'. We further check whether those strings, which will be distinct strings in $\mathscr{L}(F)$ but share the same image under $\nabla\big|^{D}_{F}$, have some common output or not. If that check (in lines 31–40) finds no common output, then it fails to satisfy condition 2 of Definition 5 and, hence, we report 'No Solution'. Otherwise the procedure merges those reached states to determinize the graph. As a consequence, Algorithm 1 will return a deterministic sensori-computational device that output simulates the input modulo the given delta relation. Its correctness appears as the next lemma.

Lemma 17. Algorithm 1 gives a sensori-computational device that output simulates F modulo $\nabla\big|^{D}_{F}$ if and only if there exists such a solution for Question 1.

Proof: $\Longrightarrow$: Let F' be the sensori-computational device generated by Algorithm 1. Then we will show that $F' \sim F \left(\bmod \nabla\big|^{D}_{F}\right)$. Let $T_s = \left\{t \mid s \nabla\big|^{D}_{F} t\right\}$. First, we show that $\bigcup_{s\in\mathscr{L}(F)} T_s \subseteq \mathscr{L}(F')$. Let G_n denote the F' after lines 1–30 (the 'n' serves as a reminder that it may be non-deterministic). The ε string is captured in the initial state (renamed from v''_ε). The unit-lengthed strings in $\nabla\big|^{D}_{F}$ are are in $\mathscr{L}(G_n)$ because they are identical to $\mathscr{L}(F) \cap Y(F)$ and the (especially created) start state makes sure that the first symbol is processed without

Algorithm 1: Delta Transform: $\textsc{Delta}_D(F) \rightarrowtail F'$

```
   Input  : A device $F$, output variator $(D, \mathbb{S}_D)$
   Output: A determinized device $F'$ that output simulates $F$
           modulo $\nabla\big|^D_F$; 'No Solution' otherwise
1  Create a sensori-computational device $F''$ as a copy of $F$
     with each state $v''$ in $F''$ corresponding to state $v$ in $F$
2  for $v'' \in V(F'')$ do // Make vertex copies
3      $L := Set(v''.IncomingLabels())$
4      if $v'' = v''_0$ then $L := L \cup \{\varepsilon\}$ ;
5      for $\ell \in L$ do
6          Create a new state $v''_\ell$ and set $c(v''_\ell) = c(v'')$
7      for $\ell \in L$ do
8          for every outgoing edge $v'' \xrightarrow{y} w''$, $w'' \neq v''$ do
9              Add an edge $v''_\ell \xrightarrow{y} w''$ in $F''$
10         for every $\ell$-labeled incoming edge $w'' \xrightarrow{\ell} v''$ do
11             if $w'' \neq v''$ then
12                 Add an edge $w'' \xrightarrow{\ell} v''_\ell$ in $F''$
13             else // Self loop
14                 Add edges $v''_k \xrightarrow{\ell} v''_\ell$ in $F''$,for $k \in L$
15     Remove $v''$ from $F''$
16 Rename $v''_\varepsilon$ to $v''_0$
17 Create $F'$ as a copy of $F''$ and $q := Queue([v'_0])$
18 while $q \neq \varnothing$ do // Apply variator
19     $v' := q$.pop()
20     Find the corresponding state $v''$ in $F''$ and
         $\{\ell\} := v''.IncomingLabels()$ // A singleton
21     for $z \in v'.OutgoingLabels()$ do
22         Initialize set $U_z := \varnothing$
23         if $v' = v'_0$ then
24             $U_z := \{z\}$
25         else
26             $U_z := \{d : D \mid (\ell, d, z) \in \mathbb{S}_D\}$
27         if $U_z = \varnothing$ then
28             return 'No Solution'
29         Replace $z$ in $F'$ with label set $U_z$
30     Mark $v'$ as visited, and add children of $v'$ to $q$
31 Reinitialize $q := Queue([v'_0])$
32 while $q \neq \varnothing$ do // State determinization
33     $v' := q.pop()$
34     for $d \in v'.OutgoingLabels()$ do
35         $W' := \{w' : V(F') \mid v' \xrightarrow{d} w'\}$, $X := \cap_{w' \in W'} c(w')$
36         if $X = \varnothing$ then
37             return 'No Solution'
38         if $|W'| > 1$ then Merge all states in $W'$ as $m'$, set
             the $c(m') := X$, and add $m'$ to $q$ ;
39         else
40             Add states in $W'$ to $q$ if not visited
41 return $F'$
```

being transformed (see line 24). Suppose all strings $s \in \mathcal{L}(F)$ with $|s| \leq k$ have $T_s \subseteq \mathcal{L}(G_n)$. Consider a string $s' \in \mathcal{L}(F)$ with $|s'| = k+1$, denote $s' = y_0 y_1 y_2 \ldots y_{k-1} y_k$ and also let its k-length prefix $s = y_0 y_1 y_2 \ldots y_{k-1}$. Let $U_{s'} = \{(\ell, d, z) : \mathbb{S}_D) \mid \ell = y_{k-1}, z = y_k\}$. Then the set of strings $T_s \times U_{s'} = T_{s'}$. But each of the elements of T_s are in $\mathcal{L}(G_n)$ by supposition and the y_k-edge traced taking s to s' in F will correspond, after being copied, to an edge (going from $v'_{y_{k-1}}$ to v'_{y_k}) that bears $U_{s'}$ (see line 26). Hence, no strings in $T_{s'}$ will crash on G_n. And so on inductively for all k. The final steps of the algorithm (lines 31–40) transform G_n to F' without modifying the language.

Secondly, we need to show that for every string $s \in \mathcal{L}(F)$ and every $t \in T_s$, we have $\mathscr{C}(F,s) \supseteq \mathscr{C}(F',t)$. Hence, string t must reach the state in G_n that is a copy of a state reached by s in F. (Strictly: the state reached is a copy of a state in F'' that was split from the state in F.) After state determinization using lines 31–40, the output of t must be a subset of the output at that state, i.e., $\mathscr{C}(F',t) \subseteq \mathscr{C}(F,s)$, since it takes the intersection of the outputs from the states that are non-deterministically reached (line 35).

$\Longleftarrow$: If there exists a solution G for Question 1, then we will show that Algorithm 1 will return a sensori-computational device. Suppose that Algorithm 1 does not give a sensori-computational device. Then it returns a 'No Solution' either at line 28 or 37. If 'No Solution' is returned at line 28, then there exists some string s in F with subsequence $y_{i-1}y_i$, such that there is no element in the variator D that can characterize the change (as $\ell = y_{i-1}$ and $z = y_i$). Hence, there does not exist a string t in G such that $s\nabla\big|^D_F t$, which violates condition 1 of Definition 5 and contradicts with the assumption that $G \sim F \ \big(\bmod \nabla\big|^D_F\big)$. If 'No Solution' is returned at line 37, then there exists a non-empty set of strings $S \subseteq \mathcal{L}(F)$ that are mapped to the same string t in F'_n, but $\cap_{s \in S} \mathscr{C}(F,s) = \varnothing$. As a consequence, there is no appropriate output chosen for t to be consistent with all strings in S. Since G is a solution, $t \in \mathcal{L}(G)$. Any output chosen for t in G will violate output simulation. ∎

Proposition 11 (Revisited). The proof of Proposition 11 stopped short of giving an actual sensori-computational device. One may simply employ Algorithm 1 to give an explicit construction.

C. Hardness of minimization

Earlier discussion has already anticipated the fact that an interesting question is: What is the minimal cardinality set D that is possible for a given F?

Decision Problem: Observation Variator Minimization (OVM)
Input: A sensori-computational device F, and $n \in \mathbb{N}$.
Output: TRUE if there exists a variator $(D, \mathbb{S}_D)$ and a sensori-computational device F', such that F' output simulates F modulo $\nabla\big|^D_F$ and $|D| \leq n$. FALSE otherwise.

Theorem 18. OVM is NP-hard.

Proof: This is proved by reducing the graph 3-coloring problem, denoted GRAPH-3C, to OVM in polynomial time. Given a GRAPH-3C problem with graph G, we will construct a corresponding OVM instance as follows (similar to the reduction to sensor minimization in [30]):

- Initialize an empty sensori-computational device $F = (V, \{v_0\}, Y, \tau, C, c)$.
- Create one observation in Y for each vertex in G, and an additional distinct (arbitrary) one x.
- Set the output set C to be $\{0,1,2\}$.
- Enumerate the edges in G in an arbitrary order and, for each edge $e = (v,w)$, create a head state h_e, two middle