ulo $\mathrm{P}_{\mathcal{N}}$. First, we show that for every string $s \in \mathscr{L}(F)$, $\mathrm{P}_{\mathcal{N}}(s) = \{t : \Sigma^* \mid s\,\mathrm{P}_{\mathcal{N}}\,t\} \subseteq \mathscr{L}(G)$; since $s\,\mathrm{P}_{\mathcal{N}}\,s$ this also establishes condition 1 in Definition 5. Suppose that there exists a string $s = s_1 s_2 \ldots s_n$ in $\mathscr{L}(F)$ with $t = t_1 t_2 \ldots t_m \in \mathrm{P}_{\mathcal{N}}(s)$ and $t \notin \mathscr{L}(G)$. Let $t_{1\ldots i} = t_1 t_2 \ldots t_i$ be a prefix of t such that $t_{1\ldots i} \in \mathscr{L}(G)$, but $t_{1\ldots i+1} \notin \mathscr{L}(G)$. If $t_{i+1} \in \mathcal{N}$ then $t_{1\ldots i+1}$ cannot crash on G due to the introduction of self loops in lines 4–5. So $t_{i+1} \notin \mathcal{N}$. But then s, without entering any of those self loops, must also crash on G. By construction Algorithm 3 ensures $\mathscr{L}(F) \subseteq \mathscr{L}(G)$, and $s \notin \mathscr{L}(G)$ then violates the assumption.

Secondly, we will show that for every string $s \in \mathscr{L}(F)$ and $t \in \mathrm{P}_{\mathcal{N}}(s)$, we have $\mathscr{C}(F,s) \supseteq \mathscr{C}(G,t)$. To facilitate the analysis, we focus on the structure G_n that is constructed between lines 1–5 ('n' stands for a non-deterministic version of G). Since t is obtained by pumping some labels in $\mathcal{N}$ on s and those pumped labels can be consumed in the self loops in G_n, at least one trace of t in G_n must reach a copy of the state reached by s in F. Since state determinization (from G_n to G) in lines 6–17 produces an output that is common across all tracings (the intersection in line 10), we have $\mathscr{C}(G,t) \subseteq \mathscr{C}(F,s)$.

$\Longleftarrow$: If there exists a solution G for Question 3, then we will show that Algorithm 3 will return a sensori-computational device. Suppose Algorithm 3 does not give a sensori-computational device. Then it returns a 'No Solution' at line 12. As a consequence, there are multiple traces of some string t in G_n such that these traces do not share any common output. Those traces must be images for a set S of different strings in F under relation $\mathrm{P}_{\mathcal{N}}$. Otherwise, they should reach the same state in G_n and have the same output. Hence, there exists a set of strings $S \subseteq \mathscr{L}(F)$ such that $\forall s \in S, s\,\mathrm{P}_{\mathcal{N}}\,t$, and $\cap_{s\in S}\mathscr{C}(F,s) = \varnothing$. As a consequence, there is no proper output chosen for t to satisfy the conditions of output simulation. Since G is a solution for Question 3, t must also be in $\mathscr{L}(G)$. But condition 2 in Definition 5 cannot hold on G for all the $s \in S$; hence, a contradiction. ∎

A. Relationships between shrinking and pumping

As understanding the connection between these two relations (shrink and pump) supplies some insight, we start with some basic facts.

Property 24. Immediately these relationships follow:

1) $\pi_{\mathcal{N}} = \mathrm{P}_{\mathcal{N}} \fatsemi \pi_{\mathcal{N}}$. The preceding statement generalizes to $\pi_{\mathcal{N}} = \mathrm{X}_1 \fatsemi \cdots \fatsemi \mathrm{X}_n \fatsemi \pi_{\mathcal{N}}$, where by $\mathrm{X}_1 \fatsemi \cdots \fatsemi \mathrm{X}_n$ we denote any sequence of concatenations under '$\fatsemi$' of relations **id**, $\mathrm{P}_{\mathcal{N}}$, and $\pi_{\mathcal{N}}$.
2) If $\mathcal{N} \neq \varnothing$, then $\mathrm{P}_{\mathcal{N}} \subseteq \pi_{\mathcal{N}} \fatsemi \mathrm{P}_{\mathcal{N}}$. More precisely: if some string in $s_1 s_2 \cdots s_n$ contains an $s_i \in \mathcal{N}$, then $\mathrm{P}_{\mathcal{N}}$ is a strict sub-relation; otherwise the two relations are equal.
3) If $\mathcal{N} \neq \varnothing$, then $\pi_{\mathcal{N}} \subsetneq \pi_{\mathcal{N}} \fatsemi \mathrm{P}_{\mathcal{N}}$.

To give some brief interpretation: Property 24.1) leads one to conclude, with $\mathcal{N} \subseteq Y(F)$, that determining if device F is $(\mathrm{P}_{\mathcal{N}} \fatsemi \pi_{\mathcal{N}})$-simulatable, then, is identical to Question 2. This is very intuitive, as the $\mathcal{N}$-shrink has the 'last word' so to speak, and hence will drop all symbols from $\mathcal{N}$ — it does not care whether those symbols were in the input or generated via the $\mathcal{N}$-pump operation.

The generalization mentioned in Property 24.1) implies other specific facts, like that $\pi_{\mathcal{N}}$ is idempotent. Properties 24.2) and 24.3) suggest that $\mathrm{P}_{\mathcal{N}}$ behaves differently from $\pi_{\mathcal{N}}$. In fact, they share many common properties (e.g., $\mathrm{P}_{\mathcal{N}}$ is idempotent too). Actually, as will be established shortly, the question of output simulation modulo each of these relations is equivalent under conditions we shall be directly concerned with.

Lemma 25. With $\mathcal{N} \subseteq Y(F)$ for an F being $\pi_{\mathcal{N}}$-simulatable implies that F is $\mathrm{P}_{\mathcal{N}}$-simulatable.

Proof: Suppose $H \sim F \pmod{\pi_{\mathcal{N}}}$. We may assume without loss of generality that no sequences in $\mathscr{L}(H)$ contain any element of $\mathcal{N}$: if they did, we could remove them (by applying, to every edge label, a set difference with $\mathcal{N}$); removing them does not alter its output simulation of F, as no such sequences are in the image of $\pi_{\mathcal{N}}$. Apply the steps in lines 1–5 of $\textsc{Pump}_{\mathcal{N}}(H)$, in Algorithm 3, and refer to the result as G. To show $G \sim F \pmod{\mathrm{P}_{\mathcal{N}}}$, the two conditions of Definition 5 are: 1) For any $s \in \mathscr{L}(F)$, that $s \in \mathscr{L}(G)$, and $s\,\mathrm{P}_{\mathcal{N}}\,s$. 2) For any $s \in \mathscr{L}(F)$ with $s\,\mathrm{P}_{\mathcal{N}}\,t$, $t \in \mathscr{L}(G)$ because all sequences of elements of $\mathcal{N}$ can be (repeatedly, as needed) traced via the self loops introduced. Further, $\mathscr{C}(G,t) = \mathscr{C}(G, \pi_{\mathcal{N}}(t)) = \mathscr{C}(H, \pi_{\mathcal{N}}(t)) = \mathscr{C}(H, \pi_{\mathcal{N}}(s)) \subseteq \mathscr{C}(F,s)$. ∎

Lemma 26. For F, with $\mathcal{N} \subseteq Y(F)$, and all $s_1 s_2 \ldots s_k \in \mathscr{L}(F)$ having $s_1 \in Y(F) \setminus \mathcal{N}$, then device F being $\mathrm{P}_{\mathcal{N}}$-simulatable implies that F is $\pi_{\mathcal{N}}$-simulatable.

Proof: Construct G via $G = \textsc{Pump}_{\mathcal{N}}(F)$. To show that it output simulates F modulo $\pi_{\mathcal{N}}$ observe that, following Algorithm 3 (lines 4–5), G will have self loops labeled with $\mathcal{N}$ for all vertices except the initial one. Also, we know further that all edges bearing elements in $\mathcal{N}$ must be self loops because any two vertices connected by an edge bearing elements in $\mathcal{N}$ will be merged in the state determinization process (lines 6–17) and their connecting edge deleted. Any string $s \in \mathscr{L}(F)$ maps to a unique $t = \pi_{\mathcal{N}}(s)$, namely s with all elements of $\mathcal{N}$ removed. String t can be traced in G because s can be traced in G, since $s\,\mathrm{P}_{\mathcal{N}}\,s$ and $G \sim F \pmod{\mathrm{P}_{\mathcal{N}}}$. But the particular structure of G means that tracing s will simply stay at any vertices when elements of $\mathcal{N}$ are encountered. Hence, t will visit the same states, without the extra loitering due to self loops. Furthermore, $\mathscr{C}(G,t) = \mathscr{C}(G,s) \subseteq \mathscr{C}(F,s)$. ∎

Theorem 27 (equivalence of pumping and shrinking)**.** Given any sensori-computational device F and $\mathcal{N} \subseteq Y(F)$, such that all $s_1 s_2 s_3 \ldots s_k \in \mathscr{L}(F)$ have $s_1 \in Y(F) \setminus \mathcal{N}$, then

$$F \text{ is } \mathrm{P}_{\mathcal{N}}\text{-simulatable} \iff F \text{ is } \pi_{\mathcal{N}}\text{-simulatable}$$

Proof: Combine Lemma 25 and Lemma 26. ∎

The intuitive interpretation is clear. If elements of $\mathcal{N}$ can be pumped, you cannot conduct any computation that depends on their number. This is true even when F has strings in

its language that include some elements of $\mathcal{N}$ because, when such elements are encountered, they could either be pumped additions or originally in the string—two cases which cannot be distinguished. The following remark does emphasize that some care is needed, however.

Remark 7. Both Lemmas 25 and 26 construct a new device. That this should be necessary for Lemma 25 is scarcely surprising: if $H \sim F \pmod{\pi_{\mathcal{N}}}$, an attempt at redeploying device H directly under pumping could fail immediately since $\mathscr{L}(H)$ need contain no strings with any element of $\mathcal{N}$, yet for $\mathrm{P}_{\mathcal{N}}$, the device must consume many strings full of $\mathcal{N}$ elements. In Lemma 26, the case for construction of a new sensori-computational device is more subtle. We show this as an example.

Example 10. For F with $\mathcal{N} \subseteq Y(F)$, beware that

$$G \sim F \pmod{\mathrm{P}_{\mathcal{N}}} \not\Longrightarrow G \sim F \pmod{\pi_{\mathcal{N}}}.$$

Figure 7 is an example of a simple sensori-computational device, we shall refer to it as F_{tiny}. Figure 8 gives two more devices, G_1 and G_2. All three have $Y(F_{\text{tiny}}) = Y(G_1) = Y(G_2) = \{a,b,n\}$. Both G_1 and G_2 output simulate F_{tiny} modulo $\mathrm{P}_{\{n\}}$, but G_2 fails to output simulate F_{tiny} modulo $\pi_{\{n\}}$. The string $anb \in \mathscr{L}(F_{\text{tiny}})$, but $\pi_{\{n\}}(anb) = ab \notin \mathscr{L}(G_2)$. □

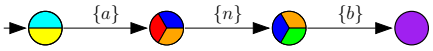
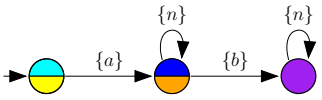
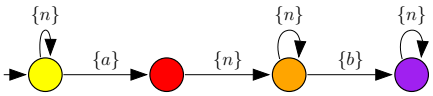


Fig. 7: An example of a simple sensori-computational device F_{tiny}, with $Y(F_{\text{tiny}}) = \{a,b,n\}$.



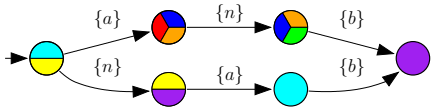
(a) A G_1 which output simulates F_{tiny} modulo $\mathrm{P}_{\{n\}}$, and modulo $\pi_{\{n\}}$ as well. Device G_1 can be obtained from Algorithm 3.



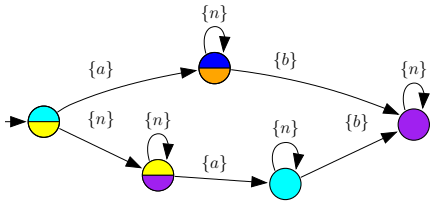
(b) A device G_2 that output simulates F_{tiny} modulo $\mathrm{P}_{\{n\}}$, but which fails modulo $\pi_{\{n\}}$.

Fig. 8: Two devices, related to F_{tiny}, the one in Figure 7, help illustrate how Theorem 27 is a statement about the existence of some device. A device that will output simulate modulo $\mathrm{P}_{\{n\}}$ need not modulo $\pi_{\{n\}}$; however, the devices Algorithm 3 produces will.

Remark 8. The condition on the first element of the sequences in Lemma 26 and Theorem 27 is necessary. The sensori-computational device F_{small} shown in Figure 9a is obtained by adding a string '*nab*' to F_{tiny}. Figure 9b gives a device G_1' that output simulates F_{small} modulo $\mathrm{P}_{\{n\}}$. However, no



(a) A new sensori-computational device F_{small} with an additional string '*nab*' being added to F_{tiny}.



(b) A device G_1' that output simulates F_{small} modulo $\mathrm{P}_{\{n\}}$.

Fig. 9: After adding a new string, the sensori-computational device in Figure 7 is only output simulatable under $\mathrm{P}_{\{n\}}$, not under $\pi_{\{n\}}$.

device exists that can output simulate modulo $\pi_{\{n\}}$ because $\pi_{\{n\}}(na) = \pi_{\{n\}}(a) = a$, and $\{\text{cyan}\} \cap \{\text{red}, \text{blue}, \text{orange}\} = \varnothing$. This caveat is neither a particular concern nor limitation for us, as sensori-computational devices that are derivatives (Definition 10) have sequences where the first element is distinct. For these, the first element gives the offset or initial value, whereas the remainder has the role of tracking the dynamic variations. It is pumping or shrinking of these variations that is important.

VI. Data acquisition semantics reprise: Monoidal variators

The previous two sections do not break the atomicity of the symbols in the original signal space. For instance, the polling acquisition mode (modeled via the $\mathcal{N}$-pump relation) adds neutral elements to the stream; it does not consider what happens if a change is occurring continuously so that the query arrives amid a change. If we desire to query the sensor with maximal flexibility, such as if one were to model a general asynchronous interaction, then some extra structure is required. To move in this direction, we will need the output variator to possess some additional properties.

Definition 28 (monoidal variator)**.** A *monoidal variator* for an observation set Y, is a monoid $(D, \oplus, 1_D)$ and a right action[3] of D on Y, $\bullet : Y \times D \to Y$.

Being concise, we will write $(D, \bullet)$ for a monoidal variator, the notation showing an operation in the second slot that helps to indicate that it is an action and hence D has additional algebraic structure. (This is consistent with the previous notation when we would include the ternary relation within the pair.)

Some of the earlier examples had output variators that were monoidal or could be extended to be, while not so for others.

[3]Recall that $\bullet$ is a total function with two requirements— identity: $y \bullet 1_D = y$; compatibility: $(y \bullet d_1) \bullet d_2 = y \bullet (d_1 \oplus d_2)$, for all y in Y, and all d_1, d_2 in D.

Example 11 (Lane sensor, again)**.** Building on Example 2, the observation variator was given as $D_{3\text{-lane}} = \{\text{LEFT}, \text{NULL}, \text{RIGHT}\}$. Given that there are 3 lanes, it means that one might wish to combine, say, two RIGHT actions, one after the other. As there are only three elements, two RIGHT actions might map to a RIGHT (as that seems less wrong than LEFT or NULL). Following this might give the following 'operator':

$\oplus_1$	LEFT	NULL	RIGHT
LEFT	LEFT	LEFT	NULL
NULL	LEFT	NULL	RIGHT
RIGHT	NULL	RIGHT	RIGHT

But $\oplus_1$ fails to be a monoid operator as since the associativity rule does not hold: $(\text{LEFT} \oplus_1 \text{LEFT}) \oplus_1 \text{RIGHT} \neq \text{LEFT} \oplus_1 (\text{LEFT} \oplus_1 \text{RIGHT})$.

Here is an alternative which does yield a valid operator, although it is still hard to give it a consistent interpretation:

$\oplus_2$	LEFT	NULL	RIGHT
LEFT	RIGHT	LEFT	NULL
NULL	LEFT	NULL	RIGHT
RIGHT	NULL	RIGHT	LEFT

But now the action causes difficulty. While NULL must map 0, 1 and 2, each to themselves, the form of $\oplus_2$ requires that the action treat RIGHT $\oplus_2$ RIGHT identically with LEFT. This fails to describe lanes 0, 1 and 2, in a consistent fashion. The lanes do not seem to admit any monoidal variator. □

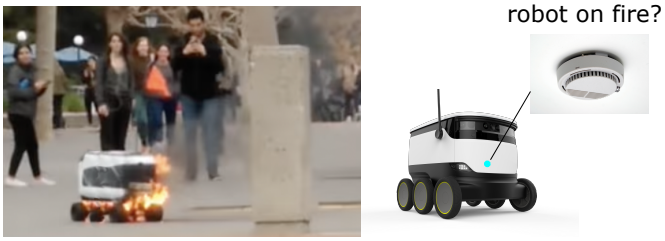


Fig. 10: A delivery robot equipped with a smoke detector in order to determine whether it has caught on fire.

Example 12 (Robot on fire)**.** Consider a sensor indicating that the robot has encountered some irrecoverable failure. For instance, the delivery robot shown in Figure 10 is equipped with a sensor to detect some irreversible condition. Once the sensor is triggered, it retains this status permanently.

Representing the status of the robot by 0 for 'normal' and 1 for 'abnormal', we may then use a monoid variator $D_2 = \{☺, ☹\}$, with 1_{D_2} is ☺, and the monoid operator $\oplus$ and the right action $\bullet$ in table form as:

$\oplus$	☺	☹
☺	☺	☹
☹	☹	☹

and

$\bullet$	☺	☹
0	0	1
1	1	1

□

Example 13 (Wall sensor, re-revisited)**.** The D_2 and table for S_{D_2} in Example 1 shows that it has the potential to be monoidal, and indeed an appropriate right action can be defined. □

The case in Example 13 also admits an inverse, leading on to the following.

Proposition 29. A sufficient condition for an affirmative answer to Question 1 when $(D,\bullet)$ is a monoidal variator is that D be, additionally, a group (i.e., possesses inverses), and there be a $y_0 \in Y(F)$ for which $y_0 \bullet D = Y(F)$.

Proof: We establish the conditions in Proposition 12: for every $y, y' \in Y(F)$ there exists some $d \in D$ so that $y = y_0 \bullet d$ and $d' \in D$ so $y' = y_0 \bullet d'$. But then let $g = d^{-1} \oplus d'$, then $y \bullet g = y \bullet (d^{-1} \oplus d') = (y \bullet d^{-1}) \bullet d' = y'$. And g must be unique, for if g and g' both have $y' = y \bullet g' = y \bullet g$, then $y \bullet (g' \oplus g^{-1}) = (y \bullet g') \bullet g^{-1} = y' \bullet g^{-1} = (y \bullet g) \bullet g^{-1} = y \bullet (g \oplus g^{-1}) = y \bullet 1_D$, thus $g'^{-1} = g^{-1}$. Hence $g' = g$. ■

Example 14 (90° Minispot, again)**.** In Example 4, we took as the output variator the integers and addition. After discussing restricting the transformation, what remained was, $\mathbb{Z}_8$, the cyclic group of order eight. □

Suppose a down-stream consumer of some device generates its queries in an asynchronous fashion. If that device measures changes, then it reports the change since the last query. When the consumer queries at a high frequency, the change sequence contains many elements, presumably with only moderate changes. Otherwise, the sequence is sparse and the change between symbols would be more considerable. To model this asynchronous data acquisition mode wherein the events reported are causally triggered by the down-stream element, we give the definition that follows. It expresses the idea that the sequences of changes should agree on the accumulated change, regardless of when the queries occur.

Definition 30 (monoid disaggregator)**.** Given the monoid $(D, \oplus, 1_D)$ and observation set Y, the associated *monoid disaggregator* is a relation, $\partial\oplus\big|_Y \subseteq (\{\varepsilon\} \cup (Y \cdot D^*)) \times (\{\varepsilon\} \cup (Y \cdot D^*))$ defined as:

0) $\varepsilon \partial\oplus\big|_Y \varepsilon$, and
1) $y_0 \partial\oplus\big|_Y y_0$ for all $y_0 \in Y$, and
2) $y_0 d_1 d_2 \ldots d_m \partial\oplus\big|_Y y_0 d'_1 d'_2 \ldots d'_n$ if $d_1 \oplus d_2 \oplus \cdots \oplus d_m = d'_1 \oplus d'_2 \oplus \cdots \oplus d'_n$.

Based on the above relation, and adhering to the established pattern, we have the natural question, posed in two forms:

Question 4. For any device F with monoidal variator $(D,\bullet)$ on $Y(F)$, is it $\left(\nabla\big|_F^D \fatsemi \partial\oplus\big|_{Y(F)}\right)$-simulatable?

Question 4 (Constructive). Given F with monoidal variator $(D,\bullet)$, find a sensori-computational device F' such that $F' \sim F \left(\bmod \nabla\big|_F^D \fatsemi \partial\oplus\big|_{Y(F)}\right)$, or indicate none exist.

Remark 9. In the reflection at the beginning of Section V, on Example 4 and its relation to Example 3, attention was directed to the significance of missing an element from the symbol