Population Protocols over Ordered Agents

Anonymous author
Anonymous affiliation
Anonymous author
Anonymous affiliation
Anonymous author
Anonymous affiliation
Anonymous author
Anonymous affiliation
Anonymous author
Anonymous affiliation
Anonymous author
Anonymous affiliation

Abstract

Population protocols are a distributed computation model in which a collection of anonymous, finite-state agents interact in randomly chosen pairs and update their states according to a fixed transition function. The computation is defined by the eventual stabilization of the population to a consensus that represents the output. In practice, it is natural to allow each agent to carry a unique identifier and compare it with that of another agent before interacting. We model this extension by having agents be totally ordered and interactions between two agents to be fireable only if their pair of identifiers falls in some condition set. For instance, $\mathsf{PP}[<]$ allows for two agents to interact only if the first one appears before the second one.

We study population protocols over ordered agents $\mathsf{PP}[\mathcal{N}]$ where $\mathcal{N}$ is a set of predicates available to restrict transition firing. We also study $\mathsf{IO\text{-}PP}[\mathcal{N}]$, the *immediate observation* fragment of $\mathsf{PP}[\mathcal{N}]$ where only one agent changes state per interaction. Our main result is that $\mathsf{IO\text{-}PP}[<]$ recognizes exactly the unambiguous star-free languages, which admits many other characterizations, such as two-variable first-order logic or two-way deterministic partially-ordered automata. We also provide a logic and an automaton model that fits in $\mathsf{PP}[<]$. We further show that if the successor predicate appears in a set $\mathcal{N}$ of $\mathsf{NSPACE}(n)$-computable predicates, then $\mathsf{IO\text{-}PP}[\mathcal{N}] = \mathsf{PP}[\mathcal{N}] = \mathsf{NSPACE}(n)$. Finally, we investigate the problem of deciding whether a given population protocol always stabilizes to a consensus. While this problem is decidable for unordered population protocols, we show that this is undecidable already for $\mathsf{PP}[<]$ and $\mathsf{IO\text{-}PP}[+1]$, but conditionally decidable for $\mathsf{IO\text{-}PP}[<]$.

2012 ACM Subject Classification Theory of computation → Formal languages and automata theory; Theory of computation → Distributed computing models; Theory of computation → Logic and verification

Keywords and phrases Population protocols, first-order logic, partially-ordered automata, unambiguous star-free languages

Digital Object Identifier 10.4230/LIPIcs.ICALP.2026.

53rd EATCS International Colloquium on Automata, Languages, and Programming (ICALP 2026).
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

To facilitate the reviewing process, we provide a table of contents below. We further provide hyperlinks for navigating between statements in the main text and proofs in the appendix. To go to the proof of a statement, click "↓" in the left margin, and then click "↑" to go back to the statement in the main text.

Contents

1 Introduction

Population protocols form a well-established model of distributed computing where anonymous agents, with very limited individual computational power, work collectively to achieve a common task [2]. In this model, an input is scattered among agents that interact pairwise and must take a decision by reaching a consensus that is stable, that is, agents must eventually all agree on the output ("consensus") and stop changing their mind ("stable"). Population protocols provide a theoretical framework for reasoning about a wide range of distributed systems, including networks of mobile sensors, chemical reaction networks, and social networks. [4, 9, 10].

To familiarize the reader with population protocols, we present a classical protocol for the task of majority voting. A population consists of n agents (who are not aware of n), each carrying a state from a finite set. Here, agents start with either a or b as their state. The population aims at collectively determining whether there are initially more a's than b's. At each discrete moment, a pair of agents is chosen arbitrarily and their respective state, from $Q = \{a, b, \underline{a}, \underline{b}\}$, is updated according to these rules:

active to passive	***propagation of winning side***	***tiebreaker***
$a, b \to \underline{a}, \underline{b}$	$a, \underline{b} \to a, \underline{a}$ $b, \underline{a} \to b, \underline{b}$	$\underline{a}, \underline{b} \to \underline{b}, \underline{b}$

Since agents are unordered, each rule $p, q \to p', q'$ also stands for $q, p \to q', p'$. Here are three possible executions of the protocol starting from three different "inputs," that is, assignments of an initial state a or b to each agent:

$$\begin{aligned} aaabb &\to a\underline{a}\underline{b}b \to a\underline{aab}b \to a\underline{aaab} \to a\underline{aaaa}, \\ aabbb &\to a\underline{ab}bb \to \underline{aabb}b \to \underline{abbb}b \to \underline{bbbb}b, \\ aabb &\to a\underline{ab}b \to \underline{aabb} \to \underline{abbb} \to \underline{bbbb}. \end{aligned}$$

Assuming fair scheduling (e.g., choosing agents uniformly at random), one can show that the population stabilizes (almost surely) to the correct outcome: if there is a majority of a's initially, then agents eventually remain in $\{a, \underline{a}\}$, otherwise they eventually remain in $\{b, \underline{b}\}$.

As agents and rules are unordered, each configuration can be seen as a multiset $\boldsymbol{c}\colon Q \to \mathbb{N}$ where $\boldsymbol{c}(q)$ indicates the number of agents in state q. It is known that population protocols compute precisely the subsets of $\mathbb{N}^Q$ that are semilinear [3], or, equivalently, that are definable in Presburger arithmetic (the first-order theory of the naturals with order and addition). In particular, majority voting amounts to computing the predicate $\varphi(\boldsymbol{c}) = \boldsymbol{c}(a) > \boldsymbol{c}(b)$.

From a modeling perspective, the fact that agents are unordered is meant to correspond to a situation in which agents are replicated and anonymous entities, and hence have no identifiers and are indistinguishable. Yet, it is natural to allow replicated agents to be totally ordered, e.g., they could be devices with a unique identifier, such as a serial number, stored in read-only memory. In this context, interactions may depend on the relationships between these identifiers.

From the perspective of automata theory, this corresponds to considering population protocols where configurations are *words* rather than multisets. Standard population protocols can be seen as computing commutative properties of words, such as $|w|_a > |w|_b$, while the word setting additionally allows for noncommutative properties, such as "the middle agent has an a" or "agents strictly alternate between a and b."

Contribution

Motivated by the above, we propose to study population protocols with totally-ordered agents. The class of population protocols so defined, $\mathsf{PP}[\mathcal{N}]$, is parameterized by a set $\mathcal{N}$ of predicates over positions that can be used to restrict transition firings. Central to our study is the class $\mathsf{PP}[<]$, in which a transition $p, q \xrightarrow{<} r, s$ can only be applied to two agents in respective states p and q if the first agent appears *before* the second agent. We consider a well-studied restriction called *immediate observation* [3] where an interaction can only update a single agent, called the "observer."

Our main result, Theorem 10, establishes that $\mathsf{IO\text{-}PP}[<]$ has the same expressive power as unambiguous star-free languages, an important subclass of regular languages that admits a trove of characterizations (see [32] for a lovely survey on the pervasiveness of this class in automata theory). Hence, $\mathsf{IO\text{-}PP}[<]$ is the class of languages captured by these formalisms over finite words:

- partially-ordered unambiguous automata [24];
- partially-ordered two-way deterministic automata [30];
- $\mathsf{LTL}[\mathbf{F}^{-1}, \mathbf{F}]$: linear temporal logic with past and future operators [14];
- $\mathsf{FO}^2[<]$: the two-variable fragment of first-order logic with order [33];
- $\Delta_2[<]$: the intersection of the $\exists^*\forall^*$ and $\forall^*\exists^*$ fragments of first-order logic with order [26];
- languages recognized by finite monoids from the variety DA [29].

This provides the first characterization of this class in terms of distributed computing, rather than automata, logic, or algebra.

In addition, we explore systematically the classes induced by our definitions:

- In Section 3, we provide a toolbox to study population protocols over ordered agents $\mathsf{PP}[\mathcal{N}]$, regardless of $\mathcal{N}$. We study protocols that need to stabilize only if they reach a positive consensus, which we call *semi-deciders*, as opposed to protocols that always stabilize to a consensus, dubbed *deciders*. We also refine the technology of protocols with *stabilizing inputs,* studied in [28], which enables a form of composition between protocols.
- In Section 4, we prove the aforementioned characterization of $\mathsf{IO\text{-}PP}[<]$, and in Section 5, we provide a natural logic and an automaton model expressible in $\mathsf{PP}[<]$. In this latter section, we fall short of showing exact characterizations, but provide conjectures based on our new models.
- In Section 6, we explore the expressiveness of $\mathsf{IO\text{-}PP}[\mathcal{N}]$ and $\mathsf{PP}[\mathcal{N}]$ when the successor predicate is available, that is, when transitions can be restricted to fire only if they act on two adjacent agents. We show that if all the predicates of $\mathcal{N}$ are $\mathsf{NSPACE}(n)$ computable, then $\mathsf{IO\text{-}PP}[\mathcal{N}] = \mathsf{PP}[\mathcal{N}] = \mathsf{NSPACE}(n)$ — this is arguably less surprising than our main result on $\mathsf{IO\text{-}PP}[<]$, as the successor allows for the left-to-right propagation of information.
- Finally, since protocols are only well-behaved when they are deciders, and since this property is semantic, we explore in Section 7 whether we can check if a given population protocol is a decider. We thus ask if the *syntax* of deciders is decidable — this is sometimes called the *well-specification* problem. We show that it is undecidable for $\mathsf{IO\text{-}PP}[+1]$, $\mathsf{PP}[+1]$, and $\mathsf{PP}[<]$, and conditionally decidable for $\mathsf{IO\text{-}PP}[<]$.

Related work

Our model is closely related to the *community protocols* of Guerraoui and Ruppert [17]. These are population protocols where each agent has a unique identifier; each agent can *store* a constant number of identifiers; and interactions depend on these identifiers but only

with respect to their relative order. The motivation of Guerraoui and Ruppert was to devise an extension of population protocols, as mild as possible, which would be fault-tolerant. They proved that community protocols can decide languages from $\mathsf{NSPACE}(n \log n)$ while tolerating Byzantine failures of a constant number of agents [17]. The unique identifier of each agent is considered to be stored in read-only memory, as in real-world low-cost chips, and so exempt from Byzantine failures. Our model corresponds to community protocols where each agent has a *single immutable* register initialized to its unique identifier.

Bournez, Cohen, and Rabie introduce *homonym protocols*, a parameterized restriction of community protocols, where the n agents have $f(n)$ identifiers [7]. The cases of $f(n) = 1$ and $f(n) = n$ are respectively population protocols (everyone has the same identifier) and community protocols (there are as many identifiers as agents). Identifiers are from $[0..f(n)-1]$ and agents can compare them with respect to $x < y$, $x = y$, $x = y + 1$ and $x = 0$.

In [16], Gańczorz et al. introduce *selective population protocols* as an extension of population protocols where the state space is partitioned into finitely many "groups," and where an interaction picks at random an initiator in state s, and then a responder in state s' from the group of s, provided it is nonempty. This is a powerful model that allows for zero-tests, i.e., checking whether no agent holds a certain state. Thus, selective population protocols are orthogonal to our model. However, the authors dedicate a section to the median problem: $\bigcup_{n\geq 0} \Sigma^n a \Sigma^n$. They show that if selective population protocols are extended with the possibility of comparing keys, then they can solve the median problem in time $\mathcal{O}(\log^4 n)$. The authors further provide a short proof that, without leveraging the "selective" aspect of their model (i.e., groups and zero-tests), any population protocol for the median problem must work in expected time $\Omega(n)$. This latter setting, only briefly discussed in [16], corresponds to our model.

Further extensions include *mediated population protocols* [23], where communication edges have an internal state; *population protocols with unordered data* [5], where the input alphabet is infinite; and *population protocols for graph class identification problems* [37, 1], where agents aim at determining whether the communication topology satisfies some property. For other work on immediate-observation protocols, see [13, 35, 11, 34].

2 Preliminaries

Automata and logic. We assume some familiarity with formal languages, automata theory and logic over finite words (e.g., see the textbook [11]). For a word $w \in \Sigma^*$, we write $w[i]$ for the i-th letter of w, starting at 1, and $w[i..j]$ for the infix $w[i]w[i+1]\cdots w[j]$. For $\sigma \in \Sigma$, we write $|w|_\sigma$ for the number of occurrences of σ in w.

We write FO to denote first-order logic over words, where quantifiers range over positions, that is, the set $\{1, \ldots, |w|\}$ for a given word w, and where $a(x)$ holds with respect to w iff $w[x] = a$ (see, e.g., [11, Chap. 8] for formal definitions). We write $\mathsf{FO}[<]$ for the extension of FO with the numerical predicate $<$, that allows to test whether $x < y$ for two positions x and y. For example, the sentence $\varphi = (\exists x)(\forall y)[a(x) \wedge ((x < y) \to b(y))]$ describes the language $\Sigma^* ab^*$. By abuse of notation, $\mathsf{FO}[<]$ stands for both the set of syntactic sentences and for the class of languages described by these sentences. It is well known that $\mathsf{FO}[<]$ is the class of star-free (regular) languages.

We write $\Sigma_i[<]$ (resp. $\Pi_i[<]$) for the fragment of $\mathsf{FO}[<]$ of sentences in prenex normal form with i blocks of alternating quantifiers starting with $\exists$ (resp. $\forall$). For example, $\Sigma^* ab^*$ belongs to $\Sigma_2[<]$ due to the form of our example φ. We let $\Delta_i[<] = \Sigma_i[<] \cap \Pi_i[<]$.

Population protocols over ordered agents. A population protocol (PP) describes how a totally-ordered set of finite-state agents interact and reach a decision about their overall initial states. Interactions can happen between any pair of agents, and predicates are used to restrict how transitions can be taken, based on the position of the agents in the order.

(Syntax.) We extend classical PPs to allow for transitions to carry a test on the *positions* of the totally-ordered agents. Let $\mathcal{N} \subseteq 2^{\mathbb{N}\times\mathbb{N}}$ be any set, whose elements we call *numerical predicates*: these will be used as the allowed tests on a transition.[1] We let $\mathbf{true} = \mathbb{N}^2$ be the always-true predicate. The set $\mathsf{PP}[\mathcal{N}]$ of PPs over $\mathcal{N}$ is the set of transition systems (Q, Σ, O, Δ) where Q is a finite set of *states,* $\Sigma \subseteq Q$ is a distinguished subset of *initial* states, $O\colon Q \to \{\top, \bot\}$ maps each state to an *opinion*, and $\Delta \subseteq Q^2 \times (\mathcal{N} \cup \{\mathbf{true}\}) \times Q^2$ is a set of *transitions*. An element of Δ is denoted $q_1, q_2 \xrightarrow{P} q_3, q_4$, expressing, intuitively, that if two distinct agents meet, the first being in position i and state q_1, the second in position j and state q_2, such that $(i, j) \in P$, then the first agent changes its state to q_3 and the second to q_4.

If $\mathcal{N} = \{P_1, P_2, \ldots\}$, we write $\mathsf{PP}[P_1, P_2, \ldots]$ for $\mathsf{PP}[\mathcal{N}]$. Classical PPs can be seen as the class $\mathsf{PP}[\emptyset]$, recalling that we assume that **true** is always available as a numerical predicate. Our main interest is in the class $\mathsf{PP}[<]$, where $<$ is seen as the set of pairs (i, j) with $i < j$, but we will study more expressive predicates in Sections 3 and 6.

A PP is *immediate-observation* if at most one agent changes state in each interaction, i.e., every transition is of the form $a, b \xrightarrow{P} a, c$ or $a, b \xrightarrow{P} c, b$. We write $\mathsf{IO\text{-}PP}[\mathcal{N}]$ for the class of protocols in $\mathsf{PP}[\mathcal{N}]$ that are immediate-observation.

(Semantics.) Since our agents are totally ordered, we define the *configuration* of a system as a word of Q^+. *Initial configurations* are words of Σ^+. Let u and v be two configurations of the same length, we say that u *leads to* v, denoted $u \to v$, if the two configurations are equal except at potentially two distinct positions i and j, and there is a transition $u[i], u[j] \xrightarrow{P} v[i], v[j] \in \Delta$ with $(i, j) \in P$. We let $\to^*$ be the reflexive transitive closure of $\to$.

Consensus and stability. With w a configuration, let $O(w) \in \{\bot, \top\}$ be the common opinion of all states appearing in w, if there is one; otherwise $O(w)$ is undefined. A configuration w is a b-*consensus* if $O(w) = b$. It is further b-*stable* if $u \to^* v$ implies that v is a b-consensus.

▶ **Example 1.** We give an example of a protocol in $\mathsf{PP}[<]$. Consider the transition system $\mathcal{P} = (\{a, b, q_\bot\}, \{a, b\}, O, \Delta)$, with $O(a) = O(b) = \top$, $O(q_\bot) = \bot$, and transitions $b, a \xrightarrow{<} q_\bot, q_\bot$ and $q_\bot, \_ \xrightarrow{\mathbf{true}} q_\bot, q_\bot$ with "$\_$" standing for any state. Every input configuration belongs to $(a + b)^+$ and is therefore a $\top$-consensus (all agents output $\top$ initially). However, such configurations need not be $\top$-stable. For instance, starting from ba we can apply the rule $b, a \xrightarrow{<} q_\bot, q_\bot$ and obtain $q_\bot q_\bot$, which is $\bot$-stable. In contrast, every configuration in a^*b^* is $\top$-stable: it is a $\top$-consensus and no transition is enabled.

Language of a PP. Consider an infinite sequence $w_0 \to w_1 \to w_2 \to \cdots$ of configurations, which we call a *run*. We say that it is *fair* if for every w_i that appears infinitely often, each configuration of $\{v \mid w_i \to v\}$ appears infinitely often as well. By induction, fairness guarantees the same for each configuration of $\{v \mid w_i \to^* v\}$. Intuitively, fairness ensures that reachable configurations cannot be avoided forever (in a probabilistic setting, the fair runs would be the runs occurring almost surely). We will assume that any configuration can be extended into a run, and so into a fair run, by implicitly adding "no operation" transitions.

[1] These are sometimes called *uniform* numerical predicates in the literature, to emphasize the fact that they do not depend on the *total* number of agents. A natural predicate that is not uniform is $\mathbf{max}(x)$ which is true if x is the position of the last agent. This technical difference will not impact our results.

The *language* of a PP $\mathcal{P}$ is the set $L(\mathcal{P})$ of initial configurations from which there is a fair run that contains a $\top$-stable configuration. Naturally, an initial configuration can be the origin of a fair run containing a $\top$-stable configuration, another run containing a $\bot$-stable configuration, or even a run that contains no stable configurations. We single out PPs that have more crisp behaviors. We say that a PP is a *decider* when for all $w \in \Sigma^*$, there is a $b \in \{\top, \bot\}$ such that *all* fair runs from w contain a b-stable configuration. This is usually called *well-specified* in the literature and we justify our nomenclature in Section 3.1. We say that $L(\mathcal{P})$ is $\mathsf{PP}[\mathcal{N}]$-decidable or $\mathsf{IO\text{-}PP}[\mathcal{N}]$-decidable with the obvious meaning. We will identify $\mathsf{PP}[\mathcal{N}]$ and $\mathsf{IO\text{-}PP}[\mathcal{N}]$ with the class of languages decidable by these protocols.

Note that since population protocols are ill-defined when no agents are present, we adopt the convention, when working with their languages, to disregard the empty word.

▶ **Example 2.** Consider the protocol of $\mathsf{PP}[<]$ from Example 1. We show that it decides the language a^*b^*. Every input in a^*b^* is already $\top$-stable. Conversely, if an input word is not in a^*b^*, then it contains the factor ba. Hence, in any fair run, the transition $(b, a) \to (q_\bot, q_\bot)$ is eventually executed. From that point on, every remaining agent eventually interacts with a $q_\bot$-agent and is converted to $q_\bot$, so the run reaches a configuration in $q_\bot^*$, which is $\bot$-stable.

▶ **Example 3.** Appendix A.1 provides a more intricate and thorough example of a $\mathsf{PP}[<]$.

We provide a generic upper bound on the complexity of languages decided by population protocols; we will exhibit a matching lower bound in Section 6 for $\mathsf{IO\text{-}PP}[+1]$. Recall that $\mathsf{NSPACE}(n)$ is the class of languages recognized by linear-bounded nondeterministic Turing machines, i.e., nondeterministic machines that require space $O(n)$ over inputs of size n. Languages of this class are exactly the context-sensitive languages [20].

▶ **Theorem 4.** *Let $\mathcal{N}$ be a set of numerical predicates, all of which decidable in $\mathsf{NSPACE}(n)$. We have $\mathsf{PP}[\mathcal{N}] \subseteq \mathsf{NSPACE}(n)$.*

Proof. Consider a protocol in $\mathsf{PP}[\mathcal{N}]$. The set of configurations that are *not* $\top$-stable is decidable in $\mathsf{NSPACE}(n)$. Indeed, it is sufficient to nondeterministically guess a partial run (i.e., a finite sequence of configurations $w_0 \to w_1 \to \cdots \to w_n$) that leads to a configuration that is not a $\top$-consensus. Each transition can be guessed, its condition checked, and its effect applied in $\mathsf{NSPACE}(n)$.

By the Immerman–Szelepcsényi theorem, $\mathsf{NSPACE}(n)$ is closed under complement, and so the set of $\top$-stable configurations is in $\mathsf{NSPACE}(n)$. Since the protocol is a decider, to check that a word w is accepted, it is sufficient to nondeterministically guess a partial run from w, and check that it ends in a $\top$-stable configuration. These are all tasks in $\mathsf{NSPACE}(n)$. ◀

3 Semantic restrictions of population protocols: a toolbox

In this section, we define two restrictions of population protocols that will be used to simplify our constructions.

We first define *semi-deciders*, which do not require the full behavior of deciders with respect to stable configurations. Throughout the next sections, we will see that semi-deciders are much easier to define for some languages, and we will rely on the forthcoming Lemma 5 to combine semi-deciders into deciders.

We then define *protocols with stabilizing inputs*, which are protocols where agents can change their mind about their input. Such protocols are harder to design, since they are more robust to change, but we show, in Lemma 9, that they exhibit a strong closure property: they are closed under alphabet rewriting.

3.1 Semi-deciders

▶ **Definition 1.** We say that a PP *semi-decides* $L \subseteq \Sigma^+$ if for all $u \in \Sigma^+$ and every fair run ρ starting from u, we have $u \in L$ iff ρ visits a $\top$-stable configuration. We use the terms $\mathsf{PP}[\mathcal{N}]$*-semi-decidable* and $\mathsf{IO\text{-}PP}[\mathcal{N}]$*-semi-decidable* with the obvious meaning.

Note that a decider is a semi-decider in which we additionally require that $u \notin L$ iff ρ visits a $\bot$-stable configuration; that is, all fair runs contain a stable configuration. Illustrating the differences, Figure 1 shows the situations that can occur for runs from an input w.

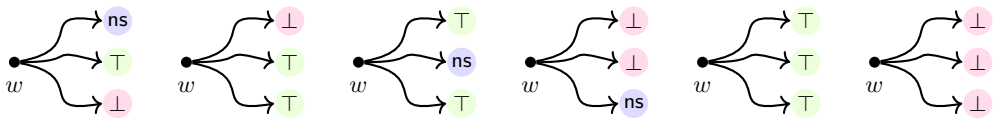


Figure 1 Configurations labeled ⊤, ⊥ and ns are ⊤-stable, ⊥-stable, and configurations from which no stable configurations are reachable. Arrows depict partial, finite runs. Situations (1-6) can happen in PP. Only situations (4-6) can happen in semi-deciders, while only situations (5-6) can in deciders.

Our naming convention is justified by the following property:

▶ **Lemma 5.** *A language L is $\mathsf{PP}[\mathcal{N}]$-decidable iff L and its complement are $\mathsf{PP}[\mathcal{N}]$-semi-decidable. The same holds for $\mathsf{IO\text{-}PP}[\mathcal{N}]$.*

Proof sketch. We combine the two semi-deciders for L and for $\Sigma^+ \setminus L$ by running them in parallel. Each agent stores a pair of states, one for each semi-decider, together with a *belief* indicating which semi-decider it currently trusts. Transitions either simulate one step of one of the semi-deciders, or *flip* the belief so that beliefs can align; in particular, an agent may flip when it meets an agent with the opposite belief, or when its currently trusted component produces a $\bot$-witness. For any input u, exactly one of the two simulations eventually provides such a witness (since exactly one of $u \in L$ or $u \notin L$ holds), which forces all agents to converge to the correct belief and yields a stable consensus, hence a decider. ◀

We note these elementary closure properties:

▶ **Lemma 6.** *If $L_1, L_2 \subseteq \Sigma^+$ are $\mathsf{PP}[\mathcal{N}]$-semi-decidable, then it is also the case for $L_1 \cap L_2$ and $L_1 \cup L_2$. This further holds for $\mathsf{IO\text{-}PP}[\mathcal{N}]$ and deciders.*

3.2 Protocols with stabilizing inputs

Agents of a protocol are generally not aware that they have reached a stable consensus and hence "terminated." To carry out a task A and use its output in a subsequent task B, a protocol has to perform *both* tasks concurrently. The protocol for task B thus guesses what the output of task A is going to be, but ought to be able to self-correct if it becomes clear that the guess was wrong. In this subsection, we introduce a formal notion for this "self-correction" which will simplify the design of composable protocols; this is inspired by a recent presentation of [28] for $\mathsf{PP}[\emptyset]$.

Let us consider protocols where each agent keeps a copy of its input. Formally, a protocol $\mathcal{P} = (Q, \Sigma, O, \Delta)$ is said to be *input-saving* if

- $Q = \Sigma \times R$ for some finite set R;

- Each $\sigma \in \Sigma$ is identified with (σ, r_σ) for some $r_\sigma \in R$; and
- The first component σ of any state $(\sigma, r) \in Q$ is left unchanged by all transitions of Δ.

For all $w \in Q^+$, let $\iota(w) \in \Sigma^+$ be the projection of w onto its first component. Given $u, v \in Q^n$, we write $u \rightsquigarrow v$ if either $u \to v$, or v equals u except at a single position i where $u[i] = (\sigma, r)$ and $v[i] = (\sigma', r)$. This second type of transitions models a "change of mind" of agent i on its input. We write $\rightsquigarrow^*$ for the reflexive transitive closure of $\rightsquigarrow$. Note what we do not change the definition of run, which still relies on $\to$ only.

▶ **Definition 2.** We say that $\mathcal{P}$ semi-decides $L \subseteq \Sigma^+$ *with stabilizing inputs* if (a) $\mathcal{P}$ is input-saving, and (b) for all $u \in \Sigma^+$, all $u \rightsquigarrow^* v$ and every fair run ρ starting from v, it is the case that $\iota(v) \in L$ iff ρ visits a $\top$-stable configuration. We use the term "*decides*" if (b) is strengthened with the condition that $\iota(v) \notin L$ iff ρ visits a $\bot$-stable configuration.

Note that this is more robust than simply semi-deciding: if $\mathcal{P}$ semi-decides L with stabilizing inputs, then it semi-decides L. Intuitively, after $u \rightsquigarrow^* v$, the configuration of the population (i.e., the projection of v onto the second component) may be *incompatible* with $\iota(v)$ since agents have possibly changed their mind several times on their input. Computing with stabilizing inputs means that the protocol is able to fix its configuration so that it reflects the expected output on $\iota(v)$.

It is known that any language $L \in \mathsf{PP}[\emptyset]$ can be decided, and hence semi-decided, with stabilizing inputs[2] [28]. Let us turn to an example, which will be useful later:

▶ **Proposition 7.** *The language $0^*1^* \cdots k^*$ is IO-$\mathsf{PP}[<]$-semi-decidable with stabilizing inputs.*

Proof. Let $\Sigma = \{0, 1, \ldots, k\}$ and $L = 0^*1^* \cdots k^*$. The protocol $\mathcal{P} = (Q, \Sigma, \Delta, O)$ for L is defined by $Q = \Sigma \times \{\top, \bot\}$, $O((\sigma, o)) = o$, each $\sigma \in \Sigma$ identified with $(\sigma, \top)$, and these rules:

$$(x, \top), (y, o) \xrightarrow{<} (x, \bot), (y, o) \qquad \text{for } x > y,$$
$$(x, \bot), (y, o) \xrightarrow{\textsf{true}} (x, \top), (y, o).$$

The purpose of the first rule is to detect a misordering. The second rule allows any agent to nondeterministically reset itself. Correctness is shown in the appendix. ◀

We now turn to closure properties of protocols with stabilizing inputs. We first cover union and intersection, then move on to (nondeterministic) alphabet rewriting.

▶ **Lemma 8.** *If $L_1, L_2 \subseteq \Sigma^+$ are $\mathsf{PP}[\mathcal{N}]$-semi-decidable with stabilizing inputs, then it is also the case for $L_1 \cap L_2$ and $L_1 \cup L_2$. This further holds for IO-$\mathsf{PP}[\mathcal{N}]$ and deciders.*

Given $f \colon \Sigma \to 2^\Gamma$ and $w \in \Sigma^n$, let $f(w) = \{w'_1 \cdots w'_n \mid w'_i \in f(w_i) \text{ for each } i \in [1..n]\}$. For example, if $f(0) = \{a, b\}$ and $f(1) = \{b, c\}$, then $f(01) = \{ab, ac, bb, bc\}$. We extend this notion to languages: $f(L) = \bigcup_{w \in L} f(w)$.

▶ **Lemma 9.** *Let $f \colon \Sigma \to 2^\Gamma$. If $L \subseteq \Sigma^+$ is $\mathsf{PP}[\mathcal{N}]$-semi-decidable with stabilizing inputs, then $f(L)$ is $\mathsf{PP}[\mathcal{N}]$-semi-decidable with stabilizing inputs. This further holds for IO-$\mathsf{PP}[\mathcal{N}]$.*

Proof sketch. The construction builds a protocol that semi-decides $f(L)$ by simulating, in its second component, the input-saving semi-decider for L on a *guessed* word $v \in \Sigma^+$ compatible with the real input $u \in \Gamma^+$ (i.e., $u \in f(v)$). Agents may revise this guess: whenever a $\bot$

[2] It was claimed earlier by [1], but definitions and proofs were deferred to a full paper that never appeared.

opinion is observed in the simulated component, an agent is allowed to change its guessed letter to any σ consistent with its input letter (preserving $u \in f(v)$ letterwise). If $u \in f(L)$, fairness ensures that the population can eventually rewrite the guessed word into some $v' \in L$, after which the simulation of the semi-decider for L reaches a $\top$-stable configuration and changes become disabled; if $u \notin f(L)$, reaching a $\top$-stable configuration would force the simulated input to lie in L, which is not possible. ◀

4 Expressiveness of IO-PP[<]

In this section, we provide a precise characterization of the languages decided by IO-PP[<]. For any language $L \subseteq \Sigma^*$, let us write $\equiv_L$ for the syntactic congruence of L, i.e., $w \equiv_L w'$ iff $uwv \in L \Leftrightarrow uw'v \in L$ for all $u, v \in \Sigma^*$. A language is in the class DA if it is regular, and satisfies, writing $\alpha(w) \subseteq \Sigma$ for the set of letters appearing in a word w:

$$(\forall w \in \Sigma^*)\left[w \equiv_L w^2 \to (\forall h \in \alpha(w)^*)[w \equiv_L w \cdot h \cdot w]\right], \tag{1}$$

It is a fascinating result of Pin and Weil [26] that $\mathsf{DA} = \Delta_2[<]$. We will leverage both characterizations to show the theorem below. The left-to-right inclusion will rely on (1), and the converse on $\mathsf{DA} = \Delta_2[<]$, semi-deciders, and protocols with stabilizing inputs.

▶ **Theorem 10.** IO-PP[<] = DA.

4.1 IO-PP[<] ⊆ DA

We first show that the set of stable configurations in a PP[<], and thus in an IO-PP[<], admits a simple description. Given a finite alphabet Σ and two words $u, v \in \Sigma^*$, we say that u is a *subword* of v, written $u \preceq v$, if u can be obtained from v by removing letters. It is well known that $\preceq$ is a *well-quasi-order* over Σ^* [19]. A consequence is that every strictly decreasing sequence of subword-closed sets must be finite.

▶ **Lemma 11.** *Any protocol in* PP[<] *is a well-structured transition system w.r.t. $\preceq$, that is, for all $u, u', w \in Q^*$ with $u \preceq w$ and $u \to u'$, there exists w' such that $u' \preceq w'$ and $w \to w'$. Further, the set of b-stable configurations is subword-closed and computable.*

As expected from the definition of DA, the core of the argument showing the inclusion of IO-PP[<] in DA will rely on a pumping argument. As a first observation, we show that, in an IO-PP[<], an infix of the form wzw with $\alpha(z) \subseteq \alpha(w)$ can, in some sense, mimic the behavior of the protocol on the simpler infix w.

▶ **Lemma 12.** *In an* IO-PP[<]*, if $uwv \to^* u'w'v'$ with $|u| = |u'|$ and $|v| = |v'|$, then for all z with $\alpha(z) \subseteq \alpha(w)$, there exists z' such that $uwzwv \to^* u'w'z'w'v'$ and $\alpha(z') \subseteq \alpha(w')$.*

We use this first result to show a pumping lemma on IO-PP[<]. It results from the fact that the sets of $\top$-stable and $\bot$-stable configurations are downward-closed and Lemma 12.

▶ **Lemma 13.** *Let $L \in$* IO-PP[<]*. There exists $m \geq 1$ such that, for all $w_1, \ldots, w_m \in \Sigma^+$ and $z \in \Sigma^*$ with $\alpha(w_1) = \ldots = \alpha(w_m) \supseteq \alpha(z)$, we have $w_1 \cdots w_m \equiv_L (w_1 \cdots w_m)z(w_1 \cdots w_m)$.*

We can now show that IO-PP[<]-decidable languages satisfy the definition of DA. In Lemma 14, we show that Equation 1 is satisfied, and in Lemma 15, that the languages are regular, concluding the proof.

▶ **Lemma 14.** *Let $L \in$* IO-PP[<] *and let $w \in \Sigma^*$ be such that $w \equiv_L w^2$. It is the case that $w \equiv_L w \cdot h \cdot w$ for all $h \in \alpha(w)^*$.*

Proof. With m obtained from Lemma 13, the latter implies that $w^m h w^m \equiv_L w^m$, and since $w \equiv_L w^2$, we obtain $w \equiv_L w \cdot h \cdot w$ as claimed. ◀

▶ **Lemma 15.** *Every* IO-PP[<]*-decidable language is regular.*

Proof sketch. We show that beyond some length, every word must contain a pattern of the form $(w_1 \cdots w_m)z(w_1 \cdots w_m)$ with $\alpha(w_1) = \ldots = \alpha(w_m) \supseteq \alpha(z)$. By Lemma 13, this means that every sufficiently long word is $\equiv_L$-equivalent to a shorter one, meaning that $\equiv_L$ has finitely many equivalence classes. ◀

4.2 DA $\subseteq$ IO-PP[<]

Recall that $L \in \Pi_2[<]$ iff $\Sigma^* \setminus L \in \Sigma_2[<]$. Since DA is equal to $\Delta_2[<] = \Sigma_2[<] \cap \Pi_2[<]$, it is enough to prove that $\Sigma_2[<]$ languages are IO-PP[<]-semi-decidable, appealing to Lemma 5 to conclude. Note that $\Sigma_2[<]$ is the set of languages expressible as finite unions of languages of the form $L = A_0^* a_1 A_2^* \cdots a_{m-1} A_m^*$ with the A_i's being subalphabets (see, e.g., [25, Thm. 8.8]). Since semi-deciders are closed under union by Lemma 6, we need only show that L is IO-PP[<]-semi-decidable to conclude. We start with a technical proposition that extends a result of [28] to immediate-observation protocols, then provide a semi-decider for L in Proposition 17.

▶ **Proposition 16.** *The language* $\{w \in \Sigma^+ \mid |w|_a = 1\}$ *is* IO-PP[$\emptyset$]*-semi-decidable with stabilizing inputs.*

Proof sketch. Each agent stores its current input, its last input, and a belief on the output. If the current input of an agent mismatches its former one, then it resets itself to the current one. Each agent eventually stabilizes to a fixed input $(\sigma, \sigma, \cdot)$. The belief component then controls the consensus: an $(a, a, \top)$ agent can turn any $(\sigma, \sigma, \cdot)$ with $\sigma \neq a$ to $\top$, while the presence of any $(\sigma, \sigma, \bot)$ can spread $\bot$ to other agents. Finally, if an a-agent observes another a, it switches to $\bot$; combined with the rule that lets an a-agent reset itself to $\top$, this makes a-agents alternate forever between $\top$ and $\bot$ whenever there are at least two a's, preventing stabilization in that case. As a result, if the input contains no a then every fair run reaches a $\bot$-stable consensus; if it contains exactly one a then every fair run reaches a $\top$-stable consensus; and if it contains at least two a's then some agent flips its belief forever. ◀

▶ **Proposition 17.** *The language* $L = A_0^* a_1 A_2^* \cdots a_{m-1} A_m^*$ *is* IO-PP[<]*-semi-decidable (with stabilizing inputs).*

Proof. Let $\Gamma = \{0, 1, \ldots, m\}$, $K = 0^*1^* \cdots m^*$ and $K' = \bigcap_{a \in \Gamma, a \text{ odd}} \{w \in \Gamma^+ \mid |w|_a = 1\}$. By Proposition 7, K is IO-PP[<]-semi-decidable with stabilizing inputs. Furthermore, by Proposition 16 and Lemma 8, K' is IO-PP[$\emptyset$]-semi-decidable with stabilizing inputs. By Lemma 8, $K \cap K'$ is IO-PP[<]-semi-decidable with stabilizing inputs. Let $f(i) = A_i$ for even i, and $f(i) = \{a_i\}$ otherwise. We are done by Lemma 9 since $L = f(K \cap K')$. ◀

5 Expressiveness of PP[<]

The crisp characterization of the previous section ties IO-PP[<] to a wealth of computational models with strikingly different flavors. Chief among them, DA is characterized by a logic, $\Delta_2[<]$, and by partially-ordered unambiguous automata. A natural question is thus whether PP[<] also admits such a diverse array of characterizations. We fall short of providing exact characterizations, but offer, in this section, two large classes of languages, one logically-defined and one based on partially-ordered automata, that are PP[<]-decidable.

Our formalisms will involve Presburger arithmetic, the first-order theory of the naturals with order and addition; e.g., $\phi(x) = (\exists y \in \mathbb{N})[y \geq 1 \wedge x = 2y]$ holds iff x is a positive even number. Write $\equiv_c$ for the equivalence of naturals modulo c. It is well known that Presburger arithmetic together with $\equiv_c$ with any $c \geq 2$ admits quantifier elimination. For our purposes, a *Presburger formula* is a (quantifier-free) Boolean combination of predicates of the form $\sum_{i=1}^{n} a_i x_i < b$ or $\sum_{i=1}^{n} a_i x_i \equiv_c b$, where $a_i, b \in \mathbb{Z}$, $c \geq 2$ and variables x_i are over $\mathbb{N}$.

5.1 First-order logic over word intervals

▶ **Definition 3.** For $w \in \Sigma^n$ and $\sigma \in \Sigma$, let $\#_\sigma \colon \{1, \dots, n\} \times \{1, \dots, n\} \to \mathbb{N}$ be the function that counts the number of occurrences of σ between two positions of w, i.e., $\#_\sigma(x, y) = |w[x..y]|_\sigma$. We define FO^{int} as first-order logic over word *intervals*, that is, with access to numerical values $\#_a(x, y)$ where x and y are either first-order variables, the first position (denoted "1") or the last position (denoted "**max**"). We will study $\mathsf{FO}^{\text{int}}[<, +, \equiv]$, where we allow numerical values to be compared, added, and tested modulo c for any constant c. Note that a variable x can be expressed as $\sum_{\sigma \in \Sigma} \#_\sigma(1, x)$, we will thus assume that the atomic formulas only have terms of the form $\#_a(x, y)$, though we write 1 for $\sum_{\sigma \in \Sigma} \#_\sigma(1, 1)$, and **max** for $\sum_{\sigma \in \Sigma} \#_\sigma(1, \mathbf{max})$. The logics Σ_k^{int}, Π_k^{int}, and Δ_k^{int} are naturally defined.

▶ **Example 18.** Let us show that the *median language* $\bigcup_{n \in \mathbb{N}} \Sigma^n a \Sigma^n$ belongs to $\Delta_1^{\text{int}}[<, +, \equiv]$. The following predicate asserts that x is the middle position: $\psi(x) = \sum_{\sigma \in \Sigma} \#_\sigma(1, x) = \sum_{\sigma \in \Sigma} \#_\sigma(x, \mathbf{max})$. The median language can either be expressed by $(\exists x)[\psi(x) \wedge a(x)]$, or by $(\forall x)[(\psi(x) \to a(x)) \wedge \mathbf{max} \equiv_2 1]$. Note that $a(x)$ is the same as $\#_a(x, x) = 1$.

We now provide a convenient characterization of languages in $\Sigma_1^{\text{int}}[<, +, \equiv]$.

▶ **Lemma 19.** *Any language from $\Sigma_1^{\text{int}}[<, +, \equiv]$ is a finite union of languages of the form $K = \{a_0 w_1 a_1 \cdots w_m a_m \mid w_i \in \Sigma^*, a_i \in \Sigma, \varphi(x_{i,\sigma} \mapsto |w_i|_\sigma, y_{i,\sigma} \mapsto |a_i|_\sigma)\}$ where $m \geq 0$ and φ is a Presburger formula over variables $\{x_{i,\sigma} \mid i \in [1..m], \sigma \in \Sigma\} \cup \{y_{i,\sigma} \mid i \in [0..m], \sigma \in \Sigma\}$.*

Thanks to the previous lemma, we are able to build a protocol $\mathsf{PP}[<]$ semi-deciding any language in $\Sigma_1^{\text{int}}[<, +, \equiv]$, by building semi-deciders with stabilizing inputs for each K, and then using the closure properties presented in Section 3.2.

▶ **Proposition 20.** *Any language from $\Sigma_1^{\text{int}}[<, +, \equiv]$ is $\mathsf{PP}[<]$-semi-decidable.*

Proof. Let $L \in \Sigma_1^{\text{int}}[<, +, \equiv]$. By Lemma 19, L is a finite union of languages of the form $L' = \{a_0 w_1 a_1 \cdots w_m a_m \mid w_i \in \Sigma^*, a_i \in \Sigma, \varphi(x_{i,\sigma} \mapsto |w_i|_\sigma, y_{i,\sigma} \mapsto |a_i|_\sigma)\}$ where $m \geq 0$ and φ is a Presburger formula. It suffices to show that L' is $\mathsf{PP}[\mathcal{N}]$-semi-decidable with stabilizing inputs. Indeed, by Lemma 8, that class of languages is closed under union.

Let $\underline{\Sigma} = \{\underline{\sigma} \mid \sigma \in \Sigma\}$, $A_i = \underline{\Sigma} \times \{i\}$, $W_i = \Sigma \times \{i\}$ and $\Gamma = \bigcup_{i \in [0..m]} A_i \cup \bigcup_{i \in [1..m]} W_i$. We justify that the three following languages K, K' and K'' over alphabet Γ are semi-decided with stabilizing inputs:

- $K = A_0^* W_1^* A_1^* \cdots W_m^* A_m^*$,
- $K' = \{w \in \Gamma^+ \mid \varphi(x_{i,\sigma} \mapsto |w|_{(\sigma,i)}, y_{i,\sigma} \mapsto |w|_{(\underline{\sigma},i)})\}$,
- $K'' = \left\{w \in \Gamma^+ \mid \sum_{\gamma \in A_1} |w|_\gamma = 1 \wedge \cdots \wedge \sum_{\gamma \in A_m} |w|_\gamma = 1\right\}$ (recall $A_i \cap A_j = \emptyset$ for $i \neq j$).

Let $f(2i) = A_i$ and $f(2i+1) = W_{i+1}$. We have $f(0^*1^*2^* \cdots (2m)^*) = K$. By Proposition 7 and Lemma 9, K is $\mathsf{PP}[<]$-semi-decidable with stabilizing inputs. Since $K', K'' \in \mathsf{PP}[\emptyset]$, these two languages are $\mathsf{PP}[\emptyset]$-decidable with stabilizing inputs [1, 28].

By Lemma 8, the language $K \cap K' \cap K''$ is $\mathsf{PP}[<]$-semi-decidable with stabilizing inputs. Let $g((\sigma, i)) = g((\underline{\sigma}, i)) = \{\sigma\}$. We are done by Lemma 9 since $L' = g(K \cap K' \cap K'')$. ◀

▶ **Corollary 21.** $\Delta_1^{\text{int}}[<,+,\equiv] \subseteq \mathsf{PP}[<]$.

The language $(ab)^+$ over alphabet $\{a,b\}$ plays a key role in the study of DA; indeed, DA is the largest class of regular languages, closed under the so-called variety operations, that does *not* contain $(ab)^+$. Strikingly, $\Delta_1^{\text{int}}[<,+,\equiv]$ cannot either express this language, although it belongs to $\Pi_1^{\text{int}}[<,+,\equiv]$, since it is described by $(\forall x)[(x \equiv_2 1) \leftrightarrow a(x)]$.

▶ **Proposition 22.** *The language $(ab)^+$ does not belong to $\Sigma_1^{\text{int}}[<,+,\equiv]$.*

5.2 Partially-ordered Parikh automata

A *partially-ordered Parikh automaton* (*poPA*) is a tuple $\mathcal{A} = (Q, \Sigma, \delta, q_0, F, \Psi)$ where:

- Q is a finite set of *states* equipped with a partial order $\leq$;
- Σ is a finite *alphabet*;
- $\delta \subseteq Q \times \Sigma \times Q$ is the *transition relation*, that satisfies $p \leq q$ for all $(p,\sigma,q) \in \delta$; if $p = q$, the transition is dubbed a *self-loop*, and otherwise, a *progress transition*;
- $q_0 \in Q$ is the *initial state*, and $F \subseteq Q$ is the set of *final states*;
- Ψ is a Presburger formula over variables δ.

For all $t = (q,\sigma,q') \in \delta$, we write $q \to^t q'$ and $\pi_\Sigma(t) = \sigma$. We naturally lift these notations to sequences. A word $w \in \Sigma^*$ is *accepted* by $\mathcal{A}$ if there exist $\rho \in \delta^*$ and $q \in F$ such that $q_0 \to^\rho q$, $\pi_\Sigma(\rho) = w$ and $\Psi(t \mapsto |\delta|_t)$ holds. The *language* of $\mathcal{A}$ is the set $L(\mathcal{A})$ of words it accepts. Figure 2 depicts two examples of poPA.

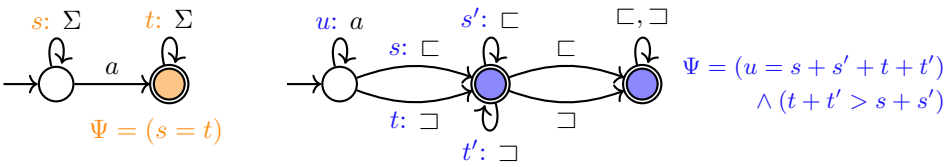


Figure 2 Example of poPA for the median language $\bigcup_{n\geq 0} \Sigma^n a \Sigma^n$ (left) and the coDyck-witness language $\{a^n v \mid n \geq 1, v \in \{\sqsubset,\sqsupset\}^{\geq n}, |v[1..n]|_\sqsupset > |v[1..n]|_\sqsubset\}$ (right).

Lemma 19 allows us to translate a formula from $\Sigma_1^{\text{int}}[<,+,\equiv]$ into a poPA by guessing a factorization $a_0 w_1 a_1 \cdots w_m a_m$ with a progress transition for each a_i, and a self-loop for each w_i; and then using the Presbuger acceptance formula of the automaton to verify the guess. We can also translate a poPA into a formula of $\Sigma_1^{\text{int}}[<,+,\equiv]$ by guessing the position of the progress transitions and verifying the validity of the path. This yields:

▶ **Theorem 23.** *A language is recognized by some poPA iff it belongs to $\Sigma_1^{\text{int}}[<,+,\equiv]$.*

Since $\Sigma_2[<]$ is known to be equivalent to partially-ordered automata, and since the latter trivially form a subclass of poPA, it is the case that $\Sigma_2[<] \subseteq \Sigma_1^{\text{int}}[<,+,\equiv]$. As $\Sigma_2[<]$ is a subset of the regular languages, this implies that $\Delta_2[<] \subset (\Delta_1^{\text{int}}[<,+,\equiv] \cap \text{Reg})$. The strict inclusion follows from the fact that $\Delta_2[<]$ cannot express the language $E = \{a^{2n} \mid n \geq 1\}$ over alphabet $\{a\}$, whereas it trivially belongs to $\Delta_1^{\text{int}}[<,+,\equiv]$, with the formula $\mathbf{max} \equiv_2 0$.

A poPA $\mathcal{A}$ is *weakly unambiguous*[3] if every $w \in L(\mathcal{A})$ is accepted by at most one path (w.r.t. F *and* Ψ). Note that the automata of Figure 2 are weakly unambiguous and can be

[3] The nomenclature, introduced in [6], stems from prior studies [8] which called "unambiguous" the PA with an unambiguous underlying automaton. We note that the examples of Figure 2 can be shown, using the tools of [8], not to be expressible with unambiguous PA.

complemented. By Theorem 23, this means that both languages belong to $\Delta_1^{\text{int}}[<,+,\equiv]$. More generally, we conjecture that weakly-unambiguous poPA are closed under complement.

▶ **Observation 24.** *If weakly-unambiguous poPA are closed under complement, then any language recognized by a weakly-unambiguous poPA belongs to* $\Delta_1^{\text{int}}[<,+,\equiv]$.

Recall that DA is also characterized by two-way *deterministic* partially-ordered automata. We can show that two-way poPA are equivalent to poPA. However, defining two-way deterministic PA as two-way deterministic automata with a Presburger constraint, it is known that the model is as expressive as unambiguous PA [15], which cannot express the languages of Figure 2.

6 Expressiveness of IO-PP[$\mathcal{N}$] and PP[$\mathcal{N}$] when successor is available

We now consider protocols where transitions may be fired only when two agents are adjacent. Formally, we look at IO-PP[$\mathcal{N}$] and PP[$\mathcal{N}$] with $+1 \in \mathcal{N}$, where $+1$ (sometimes written **succ** in the literature) is the predicate $\{(x, x+1) \mid x \in \mathbb{N}\}$. This study is reminiscent of the classic study of FO^2[$<,+1$] [14, 21, 33, 36] that followed that of FO^2[$<$], the two-variable fragment of FO[$<$] that is as expressive as IO-PP[$<$].

▶ **Example 25.** Recall that the set of all b-stable configurations of a PP[$<$] is subword-closed, and hence regular. The following PP[$+1$] has a non-regular set of $\top$-stable configurations:

$$a^{\blacksquare}, b^{\blacksquare} \xrightarrow{\text{true}} a^{\blacktriangleright}, b^{\blacktriangleleft} \qquad a^{\blacktriangleright}, a \xrightarrow{+1} \checkmark, a^{\blacksquare} \qquad a^{\blacktriangleright}, b^{\blacktriangleleft} \xrightarrow{+1} \times, \times$$
$$b, b^{\blacktriangleleft} \xrightarrow{+1} b^{\blacksquare}, \checkmark$$

Let us set the opinion of each state to $\top$, except for $\times$. The set of $\top$-stable configurations cannot be regular since, from $w \in a^{\blacksquare}a^*b^*b^{\blacksquare}$, the state $\times$ will appear iff $|w|_a = |w|_b$, e.g.,

$$a^{\blacksquare}abb^{\blacksquare} \to a^{\blacktriangleright}abb^{\blacktriangleleft} \to \checkmark a^{\blacksquare}bb^{\blacktriangleleft} \to \checkmark a^{\blacksquare}b^{\blacksquare}\checkmark \to \checkmark a^{\blacktriangleright}b^{\blacktriangleleft}\checkmark \to \checkmark\times\times\checkmark.$$

The main result of this section is:

▶ **Theorem 26.** *Let $\mathcal{N}$ be a set of* NSPACE(n)*-decidable numerical predicates that contains $+1$. We have* IO-PP[$\mathcal{N}$] = PP[$\mathcal{N}$] = NSPACE(n).

To show this result, it is enough to prove that NSPACE(n) $\subseteq$ IO-PP[$+1$], since PP[$\mathcal{N}$] $\subseteq$ NSPACE(n) has been established beforehand in Theorem 4.

6.1 IO-PP[$+1$] = PP[$+1$]

We start by establishing that IO-PP[$+1$] are as expressive as PP[$+1$].

▶ **Lemma 27.** *Every language semi-decided by a protocol in* PP[$+1$] *is also semi-decided by a protocol in* IO-PP[$+1$].

Proof sketch. The construction turns a PP[$+1$] protocol $\mathcal{P}$ into an immediate-observation protocol $\mathcal{P}'$ by replacing each transition $\delta = (a,b) \xrightarrow{+1} (c,d)$ with a short *four-step handshake* along the successor relation, using auxiliary markers a^δ and $b^{\mathsf{ack},\delta}$. The four rules are:

$$a, b \xrightarrow{+1} a^\delta, b \qquad a^\delta, b \xrightarrow{+1} a^\delta, b^{\mathsf{ack},\delta}$$
$$a^\delta, b^{\mathsf{ack},\delta} \xrightarrow{+1} c, b^{\mathsf{ack},\delta} \qquad c, b^{\mathsf{ack},\delta} \xrightarrow{+1} c, d$$

Intuitively, one agent announces that it wants to perform δ, the neighbor acknowledges, and the two agents then update one after the other; additional clean-up rules erase incomplete handshakes so that executions cannot block when several handshakes overlap. ◀

6.2 NSPACE(n) $\subseteq$ PP[+1]

Thanks to Lemma 27, we can focus on showing that NSPACE(n) $\subseteq$ PP[+1].

▶ **Lemma 28.** *The language of a linear-bounded nondeterministic Turing machine is* PP[+1]*-semi-decidable.*

To match the semi-decision semantics of protocols, we replace a linear-bounded Turing machine M by a variant M_{rst} in which (i) from any non-accepting state the machine may reset to the initial configuration on the same input, and (ii) once an accepting state is reached, the machine enters a final sweeping phase that overwrites the whole tape with a symbol f.

The protocol $\mathcal{P}_M$ simulates a linear-bounded machine M_{rst} using the successor relation. The main difficulty is that protocol inputs do *not* come with explicit endmarkers: initially, every agent locally *guesses* that it is both the left and the right boundary, and stores its input letter forever. The protocol then proceeds as follows. As long as neighboring boundary guesses are consistent, the agents interpret their "tape component" as the current tape content and faithfully simulate one step of M_{rst} (including the head marker) using local successor interactions. If an inconsistency is detected, the protocol triggers a *reset* by spawning left- and/or right-moving reset tokens. These tokens propagate along the successor order, restoring each visited cell to its stored input symbol and thereby erasing any partial simulation. When a token reaches the corresponding boundary, it restores the appropriate boundary component and disappears, leaving behind a clean segment. Thanks to this reinitialization mechanism, any run can be forced to reach a well-formed tape with unique boundaries, from which the simulation behaves exactly like M_{rst}. Thanks to (ii), upon acceptance, the whole tape is overwritten by f, yielding a $\top$-stable configuration.

▶ **Corollary 29.** NSPACE(n) $\subseteq$ PP[+1].

Proof. Let $L \in$ NSPACE(n). As NSPACE(n) is closed under complement, there exist two linear-bounded nondeterministic Turing machines M and $\overline{M}$ recognizing L and $\Sigma^+ \setminus L$, respectively. By the above construction, we can build two PP[+1] protocols, say $\mathcal{P}_M$ and $\mathcal{P}_{\overline{M}}$, that semi-decide L and $\Sigma^+ \setminus L$, respectively. Lemma 5 allows to conclude. ◀

7 Decidability of checking whether a population protocol is a decider

A good portion of our results assumes that a given population protocol is a decider or a semi-decider in order to *construct* an object (a Turing machine in Theorem 4, a formula in Theorem 10, etc.). To fully understand how constructive these proofs are, we ought to study whether it is decidable, given a population protocol, to check if it is a decider. In other words, *is the syntax of deciders decidable?* From the perspective of formal verification, this is also a natural problem, known as the well-specification problem. It is decidable for PP[$\emptyset$] [12], and undecidable when the alphabet is infinite, except for immediate-observation protocols [34].

We show that the problem is undecidable already for PP[<] and IO-PP[+1], and provide a natural conjecture that would entail that the problem is decidable for IO-PP[<].

7.1 The syntax of PP[<], IO-PP[+1] and PP[+1] deciders are undecidable

First, we consider the *emptiness problem*: given a PP[$\mathcal{N}$] protocol $\mathcal{P}$, is there an execution from an initial configuration u to a $\top$-stable one? We use a technique from [34] to show a general reduction to the syntax problem.

▶ **Lemma 30.** *The emptiness problem for* PP[$\mathcal{N}$] *(resp.* IO-PP[$\mathcal{N}$]*) reduces to deciding the complement of the syntax of* PP[$\mathcal{N}$] *(resp.* IO-PP[$\mathcal{N}$]*) deciders.*

Proof sketch. From a protocol $\mathcal{P} = (Q, \Sigma, O, \Delta)$, we construct a protocol $\mathcal{P}'$ such that $\mathcal{P}'$ is *not* a decider iff $\mathcal{P}$ can reach a $\top$-stable configuration. The construction adds a fresh sink state $q_\perp$ with $O(q_\perp) = \perp$, and for every state q with $O(q) = \perp$ we add a transition allowing an agent in q to switch to $q_\perp$.

As a consequence, from any configuration that contains a $\perp$-agent, a fair execution can drive the system to the $\perp$-stable consensus $q_\perp^*$. On the other hand, if a $\top$-stable configuration is reachable in $\mathcal{P}$, then the same configuration is reachable in $\mathcal{P}'$ and remains $\top$-stable there: none of the added transitions to $q_\perp$ are enabled from $\top$-states. Therefore $\mathcal{P}'$ is not a decider.

Conversely, if $\mathcal{P}$ has no reachable $\top$-stable configuration, then no fair run of $\mathcal{P}'$ stabilizes to $\top$. Every fair run stabilizes to $q_\perp^*$ and $\mathcal{P}'$ decides the empty language. The construction preserves immediate observation, as each new transition updates at most one agent. ◀

This reduction, combined with the translation from linear-bounded Turing machines to IO-PP[+1] from Section 6, already yields the following result.

▶ **Corollary 31.** *The syntax of* IO-PP[+1] *and* PP[+1] *deciders are undecidable.*

▶ **Theorem 32.** *The emptiness problem is undecidable for* PP[<]*. Hence the syntax of* PP[<] *deciders is also undecidable.*

Proof sketch. We reduce from the Post correspondence problem. Specifically, we take two homomorphisms $h_1, h_2 \colon B^* \to A^*$ and construct a PP[<] that can reach a $\top$-stable configuration if and only if there is a word u such that $h_1(u) = h_2(u)$. We obtain it as a product of three protocols. One makes sure that we can only reach a $\top$-stable configuration from an initial one of the form $\#_1 v \#_2 u \#_3$ with $v \in B^*$ and $u \in A^*$. The two others check that we can reach a $\top$-stable configuration only when $h_1(u) = v$ and $h_2(u) = v$, respectively. Each one does so by making agents simulate two reading heads going through u and v in lockstep and verifying that $h_i(u) = v$. Initially all agents have opinion $\perp$, and in order to change it they must carry the reading head at some point, hence we cannot skip any letter in order to reach a $\top$-stable configuration.

Hence, a $\top$-stable configuration is reachable iff there exist u, v with $h_1(u) = v = h_2(u)$. ◀

7.2 The syntax of IO-PP[<] deciders is decidable, conditionally

We present a conjecture on the reachability relation of IO-PP[<], then show that it entails decidability of the syntax of their deciders:

▶ **Conjecture 33.** *The set of configurations reachable* from *a* DA *language in an* IO-PP[<] *protocol is also a* DA *language.*

▶ **Theorem 34.** *If Conjecture 33 holds, then the syntax of* IO-PP[<] *deciders is decidable.*

Proof sketch. We use two semi-decision procedures. The first one checks, for each length n, if the protocol is a decider on words of length n. If the input protocol is not a decider, we will observe it for some n. The second one looks for invariants witnessing that the protocol is a decider. We look for two disjoint languages $K_\top, K_\perp$ such that every input word is in one of the two, they are closed under transitions of the protocol, and from everywhere in K_b we can reach a b-stable configuration. Assuming the conjecture, we can assume that those invariants are in DA. We show that we can enumerate potential regular invariants and check the requirements, yielding the second semi-decision procedure. ◀

8 Open questions

The most tantalizing open question left open by this work is the characterization of $\mathsf{PP}[<]$. We conjecture that the logic and automata models introduced in Section 5 are tight:

▶ **Conjecture 35.** $\mathsf{PP}[<]$ *is the class of languages recognized by* $\Delta_1^{\mathrm{int}}[<,+,\equiv]$ *and the class of languages recognized by weakly unambiguous poPA.*

The previous conjecture would entail in particular that weakly unambiguous poPA are closed under complement. We note that it is not known whether weakly unambiguous PA themselves are closed under complement, but it may be easier to show such closure under the partially ordered assumption.

Conjecture 33 is also very natural and left open.

References

1. Dana Angluin, James Aspnes, Melody Chan, Michael J. Fischer, Hong Jiang, and René Peralta. Stably computable properties of network graphs. In *First IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 63–74, 2005. doi:10.1007/11502593_8.
2. Dana Angluin, James Aspnes, Zoë Diamadi, Michael J. Fischer, and René Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed Computing*, 18(4):235–253, 2006. doi:10.1007/s00446-005-0138-3.
3. Dana Angluin, James Aspnes, David Eisenstat, and Eric Ruppert. The computational power of population protocols. *Distributed Computing*, 20(4):279–304, 2007. doi:10.1007/s00446-007-0040-2.
4. James Aspnes and Eric Ruppert. An introduction to population protocols. In *Middleware for Network Eccentric and Mobile Applications*, pages 97–120. Springer, 2009. doi:10.1007/978-3-540-89707-1_5.
5. Michael Blondin and François Ladouceur. Population protocols with unordered data. In *Proc. 50th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 115:1–115:20, 2023. doi:10.4230/LIPICS.ICALP.2023.115.
6. Alin Bostan, Arnaud Carayol, Florent Koechlin, and Cyril Nicaud. Weakly-unambiguous Parikh automata and their link to holonomic series. In *Proc. 47th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 114:1–114:16, 2020. doi:10.4230/LIPICS.ICALP.2020.114.
7. Olivier Bournez, Johanne Cohen, and Mikaël Rabie. Homonym population protocols. *Theory of Computing Systems*, 62(5):1318–1346, 2018. doi:10.1007/S00224-017-9833-2.
8. Michaël Cadilhac, Alain Finkel, and Pierre McKenzie. Unambiguous constrained automata. *International Journal of Foundations of Computer Science*, 24(7):1099–1116, 2013. doi:10.1142/S0129054113400339.
9. Ho-Lin Chen, David Doty, and David Soloveichik. Deterministic function computation with chemical reaction networks. *Natural Computing*, 13(4):517–534, 2014. doi:10.1007/S11047-013-9393-6.
10. Zoë Diamadi and Michael J Fischer. A simple game for the study of trust in distributed systems. *Wuhan University Journal of Natural Sciences*, 6(1):72–82, 2001. doi:10.1007/BF03160228.
11. Javier Esparza and Michael Blondin. *Automata theory: An algorithmic approach*. The MIT Press, 2023.
12. Javier Esparza, Pierre Ganty, Jérôme Leroux, and Rupak Majumdar. Verification of population protocols. In *Proc. 26th International Conference on Concurrency Theory (CONCUR)*, pages 470–482, 2015. doi:10.4230/LIPICS.CONCUR.2015.470.

13 Javier Esparza, Pierre Ganty, Rupak Majumdar, and Chana Weil-Kennedy. Verification of immediate observation population protocols. In *Proc. 29th International Conference on Concurrency Theory (CONCUR)*, volume 118, pages 31:1–31:16, 2018. doi:10.4230/LIPICS.CONCUR.2018.31.

14 Kousha Etessami, Moshe Y. Vardi, and Thomas Wilke. First-order logic with two variables and unary temporal logic. *Information and Computation*, 179(2):279–295, 2002. doi:10.1006/INCO.2001.2953.

15 Emmanuel Filiot, Shibashis Guha, and Nicolas Mazzocchi. Two-way Parikh automata. In *Proc. 39th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 40:1–40:14, 2019. doi:10.4230/LIPICS.FSTTCS.2019.40.

16 Adam Ganczorz, Leszek Gasieniec, Tomasz Jurdzinski, Jakub Kowalski, and Grzegorz Stachowiak. Selective population protocols. In *Proc. 26th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS)*, pages 225–239, 2024. doi:10.1007/978-3-031-74498-3_16.

17 Rachid Guerraoui and Eric Ruppert. Names trump malice: Tiny mobile agents can tolerate Byzantine failures. In *Proc. 36th International Colloquium Automata, Languages and Programming (ICALP)*, pages 484–495, 2009. doi:10.1007/978-3-642-02930-1_40.

18 Simon Halfon. *On Effective Representations of Well Quasi-Orderings. (Représentations Effectives des Beaux Pré-Ordres)*. PhD thesis, Université Paris-Saclay, France, 2018. URL: https://tel.archives-ouvertes.fr/tel-01945232.

19 Graham Higman. Ordering by divisibility in abstract algebras. *Proceedings of the London Mathematical Society*, s3-2(1):326–336, 1952. doi:10.1112/plms/s3-2.1.326.

20 Sige-Yuki Kuroda. Classes of languages and linear-bounded automata. *Information and Control*, 7(2):207–223, 1964. doi:10.1016/S0019-9958(64)90120-2.

21 Kamal Lodaya, Paritosh K. Pandya, and Simoni S. Shah. Around dot depth two. In *Proc. 14th International Conference on Developments in Language Theory (DLT)*, pages 303–315, 2010. doi:10.1007/978-3-642-14455-4_28.

22 M. Lothaire. *Algebraic Combinatorics on Words*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2002.

23 Othon Michail, Ioannis Chatzigiannakis, and Paul G. Spirakis. Mediated population protocols. *Theoretical Computer Science*, 412(22):2434–2450, 2011. doi:10.1016/J.TCS.2011.02.003.

24 J. Andres Montoya. Asymptotic reasoning with two variables. In *Proc. 31st International Workshop on Logic, Language, Information, and Computation (WoLLIC)*, pages 38–55, 2025. doi:10.1007/978-3-031-99536-1_3.

25 Jean-Éric Pin. Syntactic semigroups. In *Handbook of Formal Languages, Volume 1: Word, Language, Grammar*, pages 679–746. Springer, 1997. doi:10.1007/978-3-642-59136-5_10.

26 Jean-Éric Pin and Pascal Weil. Polynomial closure and unambiguous product. In *Proc. 22nd International Colloquium on Automata, Languages and Programming (ICALP)*, pages 348–359, 1995. doi:10.1007/3-540-60084-1_87.

27 Frank P Ramsey. On a problem of formal logic. In *Classic Papers in Combinatorics*, pages 1–24. Springer, 1987. doi:10.1007/978-0-8176-4842-8_1.

28 Michael Raskin. Modular population protocols. In *Proc. 20th International Symposium on Algorithmics of Wireless Networks (ALGOWIN)*, pages 173–187, 2024. doi:10.1007/978-3-031-74580-5_13.

29 Marcel-Paul Schützenberger. Sur le produit de concaténation non ambigü. *Semigroup Forum*, 13(1):47–75, 1976. doi:10.1007/bf02194921.

30 Thomas Schwentick, Denis Thérien, and Heribert Vollmer. Partially-ordered two-way automata: A new characterization of DA. In *Proc. 5th International Conference on Developments in Language Theory (DLT)*, pages 239–250, 2001. doi:10.1007/3-540-46011-X_20.

31 Michael Sipser. *Introduction to the theory of computation*. Thomson Course Technology, second edition, 2006.

32 Pascal Tesson and Denis Thérien. Diamonds are forever: The variety DA. In *Semigroups, algorithms, automata and languages*, pages 475–499. World Scientific, 2002. doi:10.1142/5050.

33 Denis Thérien and Thomas Wilke. Over words, two variables are as powerful as one quantifier alternation. In *Proc. 30th Annual ACM Symposium on the Theory of Computing (STOC)*, pages 234–240, 1998. doi:10.1145/276698.276749.

34 Steffen van Bergerem, Roland Guttenberg, Sandra Kiefer, Corto Mascle, Nicolas Waldburger, and Chana Weil-Kennedy. Verification of population protocols with unordered data. In *Proc. 51st International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 156:1–156:20, 2024. doi:10.4230/LIPICS.ICALP.2024.156.

35 Chana Weil-Kennedy. *Observation Petri Nets*. PhD thesis, Technical University of Munich, Germany, 2023.

36 Philipp Weis and Neil Immerman. Structure theorem and strict alternation hierarchy for FO^2 on words. In *Proc. 21st International Workshop on Computer Science Logic (CSL)*, pages 343–357, 2007. doi:10.1007/978-3-540-74915-8_27.

37 Hiroto Yasumi, Fukuhito Ooshita, and Michiko Inoue. Population protocols for graph class identification problems. In *Proc. 25th International Conference on Principles of Distributed Systems, (OPODIS)*, pages 13:1–13:19, 2021. doi:10.4230/LIPICS.OPODIS.2021.13.

A Appendix

A.1 Example of a PP[<] for the median language

Consider the *median language* $L = \bigcup_{n\in\mathbb{N}} \Sigma^n a \Sigma^n$. Let us describe a protocol $\mathcal{P} = (Q, \Sigma, O, \Delta)$ that decides L. The states are defined as

$$Q = \Sigma \times \{\text{👉}, \text{👑}, \text{👈}\} \times \{\bot, \top\}.$$

The components respectively represent the input letter; a belief on whether the center is on the right, here, or on the left; and a belief on the output.

We identify input a with state $(a, \text{👑}, \top)$, and each input $\sigma \neq a$ with $(\sigma, \text{👑}, \bot)$. We set $O((x, y, z)) = z$. The set Δ is defined by these rules, each describing a family of transitions:

	Population halving		
(1)	$(x, \text{👑}, z),\ (x', \text{👑}, z')$	$\xrightarrow{<}$ $(x, \text{👉}, \bot),\ (x', \text{👈}, \bot)$	
	Center finding		
(2)	$(x, \text{👑}, z), (x', \text{👉}, z')$	$\xrightarrow{<}$ $(x, \text{👉}, \bot), (x', \text{👑}, x' = a)$	
(3)	$(x, \text{👈}, z), (x', \text{👑}, z')$	$\xrightarrow{<}$ $(x, \text{👑}, x = a), (x', \text{👈}, \bot)$	
	Output propagation		
(4)	$(x, y, z), (x', \text{👑}, z')$	$\xrightarrow{\text{true}}$ $(x, y, z'), (x', \text{👑}, z')$	for $y \in \{\text{👉}, \text{👈}\}$
(5)	$(x, y, \top), (x', y', \bot)$	$\xrightarrow{\text{true}}$ $(x, y, \bot), (x', y', \bot)$	for $y, y' \in \{\text{👉}, \text{👈}\}$

By fairness, the first rule must be used until one or zero 👑 remains. Moreover, by fairness, the second and third rules will respectively move the 👉's to the left, and the 👈's to the right. If some $(x, \text{👑}, y)$ remains, then, by fairness and the fourth rule, it will propagate its output y, which is $\top$ iff $x = a$, by the choice of initial states and by rule (2–3). Otherwise, if the population is of even length, the fifth rule will be used to propagate $\bot$.

Figure 3 depicts all configurations reachable from the initial configuration $aab \in L$. Any *fair* run of a population protocol leads to a bottom strongly connected component of such a reachability graph. Thus, in this example, every fair runs leads to $a^{\top}_{\text{👉}} a^{\top}_{\text{👑}} b^{\top}_{\text{👈}}$, which is $\top$-stable. Note that $a^{\bot}_{\text{👉}} a^{\bot}_{\text{👈}} b^{\bot}_{\text{👑}}$ is a $\bot$-consensus, but is not $\bot$-stable.

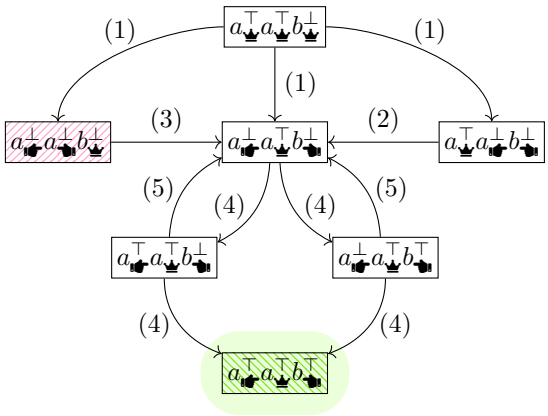
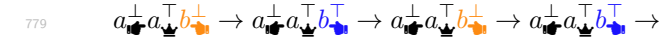
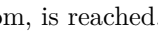
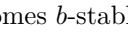


Figure 3 Configurations reachable from aab, where x^z_y stands for (x, y, z). Self-loops arising from "no operation" transitions are omitted. The hatched nodes are consensuses; the bottom one is stable.

Further observe that the reachability graph of Figure 3 has non-trivial cycles, for instance:



Informally, the first and second agents are fighting to convince the third agent. However, by fairness, this is not allowed to happen indefinitely. Eventually, the configuration  , at the bottom, is reached.

In the above specific protocol $\mathcal{P}$, for each initial configuration, its reachability graph has a unique trivial bottom strongly connected component, made of one configuration of the form  or  . However, in general, it needs not be unique or trivial. A fair run becomes b-stable iff it visits a bottom strongly connected component whose configurations are *all* b-consensuses.

In a decider, for a given initial configuration, *all* bottom strongly connected components must consist only of stable configurations, all of the *same* output.

A.2 Missing proofs from Section 3.1

▶ **Lemma 5.** *A language L is* PP$[\mathcal{N}]$*-decidable iff L and its complement are* PP$[\mathcal{N}]$*-semi-decidable. The same holds for* IO-PP$[\mathcal{N}]$*.*

Proof. $\Rightarrow$) This is immediate, since a protocol deciding L is a semi-decider for L, and the complement protocol, obtained by swapping $\top$ and $\bot$ in the opinion function, is a semi-decider for $\Sigma^+ \setminus L$.

$\Leftarrow$) This requires combining the two semi-deciding protocols $\mathcal{P}_+, \mathcal{P}_-$, for L and $\Sigma^+ \setminus L$ respectively, into a protocol $\mathcal{P}$. We let $\mathcal{P}_s = (Q_s, \Sigma, O_s, \Delta_s)$, for $s \in \{+, -\}$, and we define $\mathcal{P} = (Q, \Sigma', O, \Delta)$ as follows.

We let $Q = Q_+ \times Q_- \times \{+, -\}$, with $\Sigma' = \{(a, a, s) \mid a \in \Sigma, s = + \text{ iff } a \in L\}$. We identify $a \in \Sigma$ with the state $(a, a, s) \in Q$ where $s = +$ iff $a \in L$. By this identification, the protocol $\mathcal{P}$ recognizes a language over Σ and is correct on single agent runs by definition (since with a single agent no transition can be taken). The third component is called the *belief* of the agent. The function O maps each (q_+, q_-, s) to $O_+(q_+)$ if $s = +$, and $\neg O_-(q_-)$ otherwise. Finally, the transitions of $\mathcal{P}$ are of two kinds:

- *(Simulation.)* A transition of $\mathcal{P}_+$ or $\mathcal{P}_-$ can be used, on the appropriate component Q_+ or Q_-. The two other components stay the same.
- *(Flip.)* If two agents have different beliefs, then one of them can flip its belief. The **true** numerical predicate is used for these transitions. An agent in state (q_+, q_-, s) can also flip its belief by itself if $O_s(q_s) = \bot$.

Consider $u \in \Sigma^+$ of length at least 2. Assume that $u \in L$, the case where $u \notin L$ being symmetric. Let $v \in \mathsf{Post}^*(u)$. Let v_+ and v_- be the projections of v on Q_+ and Q_- respectively. Observe that for both $s \in \{+.-\}$, v_s is reachable from u in $\mathcal{P}_s$. In consequence, since $u \in L$, there is an execution of $\mathcal{P}_+$ from v_+ to a $\top$-stable w_+. There is also an execution of $\mathcal{P}_-$ from v_- to a configuration w_- where at least one agent has opinion $\bot$. By following both those executions one after the other, we can go from v to a configuration w in $\mathcal{P}$, so that the projection of w on Q_+ and Q_- are w_+ and w_-.

- If the agent which has opinion $\bot$ in w_- has opinion $-$, it can switch it to $+$.
- Then, we can flip all the other agents to $+$, by making them observe this agent. The resulting configuration is $\top$-stable.

As mentioned, the case $u \notin L$ is symmetric, showing that $\mathcal{P}$ is a decider. Furthermore, observe that if both $\mathcal{P}_+$ and $\mathcal{P}_-$ are *immediate observation* protocols, then so is $\mathcal{P}$. ◀

▶ **Lemma 6.** *If $L_1, L_2 \subseteq \Sigma^+$ are $\mathsf{PP}[\mathcal{N}]$-semi-decidable, then it is also the case for $L_1 \cap L_2$ and $L_1 \cup L_2$. This further holds for $\mathsf{IO}\text{-}\mathsf{PP}[\mathcal{N}]$ and deciders.*

Proof. The proof is standard, and we will give, in Lemma 8, a slightly more intricate proof with the same ideas, so we omit it here. ◀

A.3 Missing proofs from Section 3.2

▶ **Proposition 7.** *The language $0^*1^* \cdots k^*$ is $\mathsf{IO}\text{-}\mathsf{PP}[<]$-semi-decidable with stabilizing inputs.*

Proof of correctness. Note that $\mathcal{P}$ is input-saving and immediate-observation. Let us show that $\mathcal{P}$ semi-decides L with stabilizing inputs. Let $u \in \Sigma^+$ and $u \rightsquigarrow^* v$. Let $v = w_0 \to w_1 \to \cdots$ be a fair run. For the special case of $|v| = 1$, note that each state starts with output $\top$ and this remains so as no transition is ever enabled. Let us assume that $|v| \geq 2$.

Case $\iota(v) \in L$. The first rule is permanently disabled in w_0. Hence, by fairness, the second rule must swap the last component of each agent to $\top$. Once this happens, the population has reached a $\top$-consensus, and the configuration cannot change anymore.

Case $\iota(v) \notin L$. Recall that the first component of each agent is fixed from w_0 onwards. Let $i < j$ be positions such that $\iota(v)[i] > \iota(v)[j]$. Using the second rule, all agents can change their opinion to $\top$, and then, using the first rule, agent i can change its opinion to $\bot$. By fairness, this happens infinitely often, which means that the run does not stabilize. ◀

▶ **Lemma 8.** *If $L_1, L_2 \subseteq \Sigma^+$ are $\mathsf{PP}[\mathcal{N}]$-semi-decidable with stabilizing inputs, then it is also the case for $L_1 \cap L_2$ and $L_1 \cup L_2$. This further holds for $\mathsf{IO}\text{-}\mathsf{PP}[\mathcal{N}]$ and deciders.*

Proof. Let $\mathcal{P}_1 = (Q_1, \Sigma, \Delta_1, O_1)$ and $\mathcal{P}_2 = (Q_2, \Sigma, \Delta_2, O_2)$ be the protocols that respectively $\mathsf{PP}[\mathcal{N}]$-semi-decide L_1 and L_2 with stabilizing inputs.

Intersection. We simply take the product of both protocols; simulate them independently with a common input component; and output the conjunction of their outputs.

Let $Q_1 = \Sigma \times R_1$ and $Q_2 = \Sigma \times R_2$. Formally, we define $\mathcal{P} = (Q, \Sigma, \Delta, O)$ by $R = R_1 \times R_2$, $Q = \Sigma \times R$, $O((\sigma, q_1, q_2)) = O_1((\sigma, q_1)) \wedge O_2((\sigma, q_2))$, and the following rules:

$$(\sigma, q_1, r), (\sigma', q_2, s) \xrightarrow{P} (\sigma, q_3, r), (\sigma', q_4, s) \quad \text{for } (\sigma, q_1), (\sigma', q_2) \xrightarrow{P} (\sigma, q_3), (\sigma', q_4) \in \Delta_1 \text{ and } r, s \in R_2,$$

$$(\sigma, r, q_1), (\sigma', s, q_2) \xrightarrow{P} (\sigma, r, q_3), (\sigma', s, q_4) \quad \text{for } (\sigma, q_1), (\sigma', q_2) \xrightarrow{P} (\sigma, q_3), (\sigma', q_4) \in \Delta_2 \text{ and } r, s \in R_1.$$

We identify each $\sigma \in \Sigma$ with (σ, σ, σ).

Note that $\mathcal{P}$ is input-saving, and immediate-observation if it is the case of $\mathcal{P}_1$ and $\mathcal{P}_2$. Let us show that $\mathcal{P}$ semi-decides $L = L_1 \cap L_2$ with stabilizing inputs. For every $w \in Q^+$, let $\pi_1(w) \in Q_1^+$ be the projection of w onto its first two components, and let $\pi_2(w) \in Q_2^+$ be the projection of w onto its first and third components. Let $u \in \Sigma^+$ and $u \rightsquigarrow^* v$. Let $v = w_0 \to w_1 \to \cdots$ be a fair run from v. By definition of $\mathcal{P}$, the sequence $\pi_i(w_0), \pi_i(w_1), \cdots$ is a fair run of $\mathcal{P}_i$. Moreover, $\iota(v) = \iota(\pi_1(v)) = \iota(\pi_2(v))$.

Case $\iota(v) \in L$. Since $\iota(v) \in L_1 \cap L_2$, the runs of $\mathcal{P}_1$ and $\mathcal{P}_2$ eventually stabilize to $\top$-stable configurations. Since O is a conjunction, the same holds for the run of $\mathcal{P}$.

Case $\iota(v) \notin L$. Let $i \in \{1, 2\}$ be such that $\iota(v) \notin L_i$. The run of $\mathcal{P}_i$ visits infinitely many configurations containing a state $q \in Q_i$ with $O_i(q) = \bot$. Since O is a conjunction, the

same holds for the run of $\mathcal{P}$. Furthermore, if $\mathcal{P}_i$ is a decider, then the run of $\mathcal{P}_i$ eventually stabilizes to $\bot$-stable configurations, which implies the same for the run of $\mathcal{P}$.

Union. We cannot simply take the previous construction and redefine the output mapping to $O((\sigma, q_1, q_2)) = O_1((\sigma, q_1)) \vee O_2((\sigma, q_2))$. For example, if $\mathcal{P}_1$ and $\mathcal{P}_2$ have two agents with outputs $\top\bot$ and $\bot\top$ respectively, then their disjunction yields $\top\top$, which is incorrect as neither $\mathcal{P}_1$ nor $\mathcal{P}_2$ is in a $\top$-consensus.

Instead, we extend Q with an extra component $\{1, 2\}$ that indicates which output should be used, and define $O((\sigma, q_1, q_2, i)) = O_i((\sigma, q_i))$. The transitions from Δ simply ignore this new component. However, we need to add this (immediate-observation) rule:

$$(\sigma, q_1, q_2, i), (\sigma', q_1', q_2', i') \xrightarrow{\textsf{true}} (\sigma, q_1, q_2, 3-i), (\sigma', q_1', q_2', i') \qquad \text{for } i \neq i' \vee O_i(q_i) = \bot \vee O_i(q_i') = \bot.$$

This way, as long as agents have not agreed on a common choice, or as long as their choice is not in a $\top$-consensus, they may change their mind.

Note that this is correct even for "deciders": If $\mathcal{P}_1$ and $\mathcal{P}_2$ both reach $\bot$-stability, then agents will indefinitely change their choice, but nonetheless steadily output $\bot$. ◀

▶ **Lemma 9.** *Let $f\colon \Sigma \to 2^\Gamma$. If $L \subseteq \Sigma^+$ is $\mathsf{PP}[\mathcal{N}]$-semi-decidable with stabilizing inputs, then $f(L)$ is $\mathsf{PP}[\mathcal{N}]$-semi-decidable with stabilizing inputs. This further holds for IO-$\mathsf{PP}[\mathcal{N}]$.*

Proof. Let $\mathcal{P} = (Q, \Sigma, \Delta, O)$ be the protocol that semi-decides L with stabilizing inputs. Since $\mathcal{P}$ is input-saving, its states are of the form $Q = \Sigma \times R$.

We provide a protocol $\mathcal{P}'$ where, on input $u \in \Gamma^+$, the population internally starts with an arbitrary word v such that $u \in f(v)$ and runs $\mathcal{P}$ on v. With luck, it may be the case that $v \in L$, but this needs not be the case. Therefore, the population may change its choice: If an agent of $\mathcal{P}'$ has input γ and encounters a $\bot$-state of $\mathcal{P}$, then it can change its internal input of $\mathcal{P}$ to any letter σ such that $\gamma \in f(\sigma)$.

Let us now describe the protocol formally. For every $\gamma \in \Gamma$, let σ_γ be an arbitrary letter[4] of Σ such that $\gamma \in f(\sigma_\gamma)$. To handle the corner case of populations with a single agent, if there is choice such that $\sigma_\gamma \in L$, then we take one. We define $\mathcal{P}' = (Q', \Gamma, \Delta', O')$ as follows:

- $Q' = \Gamma \times Q$;
- We identify each $\gamma \in \Gamma$ with (γ, σ_γ);
- $O'((\gamma, q)) = O(q)$;
- The transitions of Δ' are defined by

$$(\gamma, q_1), (\gamma', q_2) \xrightarrow{P} (\gamma, q_3), (\gamma', q_4) \qquad \text{for } \gamma, \gamma' \in \Gamma \text{ and } (q_1, q_2) \xrightarrow{P} (q_3, q_4) \in \Delta,$$
$$(\gamma, q), (\gamma', q') \xrightarrow{\textsf{true}} (\gamma, \sigma), (\gamma', q') \qquad \text{for } \gamma, \gamma' \in \Gamma,\ q, q' \in Q \text{ and } \sigma \in \Sigma \text{ such that } \gamma \in f(\sigma), \text{ and } O(q) = \bot \text{ or } O(q') = \bot.$$

The protocol is input-saving. The second rule is immediate-observation. Moreover, if $\mathcal{P}$ is immediate-observation, then it is also the case of the first rule. It remains to show that $\mathcal{P}'$ semi-decides $f(L)$ with stabilizing inputs. For all $w \in (Q')^+$, let $\pi(w)$ be the projection of w onto its second component. Recall that $\iota(w)$ is the projection of w onto its first component.

[4] If no such σ_γ exists, then we can simply map γ to a dummy state with opinion $\bot$, as no word containing γ belongs to $f(L)$. That being said, we will never invoke the lemma with such an f.

Let $u \in \Gamma^+$ and $u \rightsquigarrow^* v$. Let $v = w_0 \to w_1 \to \cdots$ be a fair run. The case where $|v| = 1$, and hence $\iota(v) = \gamma \in \Gamma$, is trivially correct by the choice of σ_γ. Thus, suppose that $|v| \geq 2$.

Case $\iota(v) \in f(L)$. By hypothesis, there exists $v' \in L$ such that $\iota(v) \in f(v')$. For the sake of contradiction, suppose that no $\top$-stable configuration is visited by the fair run. By assumption, there exist indices $i_0 < i_1 < \cdots$ such that $O'(w_{i_j}) = \bot$ for all $j \geq 0$. By definition of O', this means that $O(\pi(w_{i_j})) = \bot$ for all $j \geq 0$. By fairness and the second rule, we can change the second component of the population to v' (and hence the input of $\mathcal{P}$ to v'). Since $\mathcal{P}$ semi-decides L with stabilizing inputs, we can reach a $\top$-stable configuration of $\mathcal{P}$ in the second component. Consequently, the second rule becomes permanently disabled, which implies that $\mathcal{P}'$ visits a $\top$-stable configuration, a contradiction.

Case $\iota(v) \notin f(L)$. For the sake of contradiction, suppose that there exists $j \geq 0$ such that $O'(w_j) = O'(w_{j+1}) = \cdots = \top$. This means that $w_j \to w_{j+1} \to \cdots$ only uses the first rule. By definition of O', we have $O(\pi(w_j)) = O(\pi(w_{j+1})) = \cdots = \top$. Since $\mathcal{P}$ semi-decides L with stabilizing inputs, this implies that $\iota(\pi(w_j)) \in L$ and hence $\iota(w_j) \in f(L)$, which is a contradiction. ◀

A.4 Missing proofs from Section 4.1

▶ **Lemma 11.** *Any protocol in* PP[<] *is a well-structured transition system w.r.t.* $\preceq$*, that is, for all* $u, u', w \in Q^*$ *with* $u \preceq w$ *and* $u \to u'$*, there exists* w' *such that* $u' \preceq w'$ *and* $w \to w'$*. Further, the set of* b*-stable configurations is subword-closed and computable.*

Proof. In a PP[<], we have $u \to u'$ if and only if there exist $a, b, c, d \in Q$ and $u_1, u_2, u_3 \in Q^*$ such that $u = u_1 a u_2 b u_3$, $u' = u_1 c u_2 d u_3$ and $(a, b) \xrightarrow{<} (c, d) \in \Delta$. Let $w \in Q^*$ be such that $u \preceq w$. We can write w as $w = w_1 a w_2 b w_3$ with $u_1 \preceq w_1$, $u_2 \preceq w_2$ and $u_3 \preceq w_3$. We obtain $w \to w' = w_1 c w_2 d w_3$ and $u' \preceq w'$ as desired.

Let $U_0 = Q^* \setminus O^{-1}(b)^*$, the set of configurations which are *not* b-consensuses. For all i, let $U_{i+1} = U_i \cup \{u \in Q^* \mid (\exists v \in U_i)[u \to v]\}$, the set of configurations that can reach U_0 in at most $i+1$ steps. We let $D_i = Q^* \setminus U_i$. The set of b-stable configurations is $D = Q^* \setminus \bigcup_{i \in \mathbb{N}} U_i$.

A key observation is that since our only predicate is $<$, all D_i are subword-closed. As a consequence, so is their intersection D. Furthermore, the sequence $(D_i)_{i \in \mathbb{N}}$ is a descending chain of subword-closed sets. Since $\preceq$ is a well quasi-order, every such chain eventually stabilizes, hence there exists $i \in \mathbb{N}$ such that $D_i = D_{i+1}$. We can compute D_i for each index until we find one such that $D_i = D_{i+1}$. Such D_i is then the set of b-stable configurations. ◀

We first show that our protocols are robust to the insertion of repeated letters, under specific contexts:

▶ **Lemma 12.** *In an* IO-PP[<]*, if* $uwv \to^* u'w'v'$ *with* $|u| = |u'|$ *and* $|v| = |v'|$*, then for all* z *with* $\alpha(z) \subseteq \alpha(w)$*, there exists* z' *such that* $uwzwv \to^* u'w'z'w'v'$ *and* $\alpha(z') \subseteq \alpha(w')$*.*

Proof. We prove this for a single step, the result follows immediately by induction. Suppose there is a step from uwv to $u'w'v'$, let $(a, b) \xrightarrow{<} (a', b')$ be the associated transition. Since the protocol is IO, we have $a = a'$ or $b = b'$. We assume $a = a'$, the other case being symmetric.

If the observing agent is in u or v, then $w = w'$ and we can go from $uwzwv$ to $u'w'z'w'v' = u'wzwv'$ with the same transition. If the observing agent is in w, then $u = u'$, $v = v'$ and, from $uwzwv$, we start by applying that transition for each b in z, observing the same agent labelled a as in the step from uwv to $u'w'v'$. We thus turn the factor z into z' where all b are turned into b'. We can then apply that same transition twice to turn each of the two copies of w to w'. Note that $\alpha(z') = (\alpha(z) \cup \{b'\}) \setminus \{b\}$, and $\alpha(w') \supseteq (\alpha(w) \cup \{b'\}) \setminus \{b\} \supseteq \alpha(z')$ as w'

is obtained from w by deleting an occurrence of b and adding an occurrence of b'. We obtain $uw'z'w'v$, which is equal to $u'w'z'w'v'$ (since here $u = u'$ and $v = v'$) with $\alpha(z') \subseteq \alpha(w')$. ◀

▶ **Lemma 13.** *Let $L \in$ IO-PP[$<$]. There exists $m \geq 1$ such that, for all $w_1, \ldots, w_m \in \Sigma^+$ and $z \in \Sigma^*$ with $\alpha(w_1) = \ldots = \alpha(w_m) \supseteq \alpha(z)$, we have $w_1 \cdots w_m \equiv_L (w_1 \cdots w_m)z(w_1 \cdots w_m)$.*

Proof. Lemma 11 indicates that for both $b \in \{\top, \bot\}$, the set of b-stable configurations is subword-closed. As a consequence, writing B^ε for $B \cup \{\varepsilon\}$, it is a finite union of languages of the form $A_1^* B_1^\varepsilon \cdots A_k^* B_k^\varepsilon$ with A_i, B_i subsets of Q, the set of states of our protocol [18, Sect. 6.1.1]. Let K_b be the maximal factorisation size k over all those languages, $K = \max(K_\top, K_\bot)$ and $m = 2K + 1$.

Let $w_1, \ldots, w_m, z \in \Sigma^*$ be such that $\alpha(w_1) = \cdots = \alpha(w_m) \supseteq \alpha(z)$, and let $u, v \in \Sigma^*$. We show that for both $b \in \{\top, \bot\}$, if we can reach a b-stable configuration from $uw_1 \cdots w_m v$ then it is also possible from $uw_1 \cdots w_m z w_1 \cdots w_m v$. This implies that $uw_1 \cdots w_m v$ is accepted if and only if $uw_1 \cdots w_m z w_1 \cdots w_m v$ is, proving the lemma.

Let $b \in \{\top, \bot\}$, suppose we can reach a b-stable configuration w' from $uw_1 \cdots w_m v$. Since steps are length-preserving, we can divide w' into $u'w'_1 \cdots w'_m v'$ where u', v' and each w'_i have the same length as u, v and each w_i, respectively.

Since w' is b-stable, there exists $k < K_b$ and an expression $A_1^* B_1^\varepsilon \cdots A_k^* B_k^\varepsilon$ whose language contains w', and only contains b-stable words. Since $m > 2K_b \geq 2k$, there exist i, j such that $u'w'_1 \cdots w'_{i-1} \in A_1^* B_1^\varepsilon \cdots A_j^*$, $w'_{i+1} \cdots w'_m v' \in A_j^* B_j^\varepsilon \cdots B_k^\varepsilon$, and $w'_i \in A_j^*$.

Since $w = uw_1 \cdots w_m v \to^* u'w'_1 \cdots w'_m v' = w'$, by Lemma 12 there is a partial run from

$$\widetilde{w} := (uw_1 \cdots w_{i-1})w_i(w_{i+1} \cdots w_m)z(w_1 \cdots w_{i-1})w_i(w_{i+1} \cdots w_m v)$$

to

$$\widetilde{w'} := (u'w'_1 \cdots w'_{i-1})w'_i z' w'_i(w'_{i+1} \cdots w'_m v')$$

for some z' with $\alpha(z') \subseteq \alpha(w'_i)$, and $|z'| = |(w_{i+1} \cdots w_m)z(w_1 \cdots w_{i-1})|$.

Furthermore, since we have $u'w'_1 \cdots w'_{i-1} \in A_1^* B_1^* \cdots A_j^*$, and $w'_{i+1} \cdots w'_m v' \in A_j^* B_j^\varepsilon \cdots B_k^\varepsilon$, and $z' \in \alpha(w'_i)^* \subseteq A_j^*$, we infer $\widetilde{w'} \in A_1^* B_1^\varepsilon \cdots A_k^* B_k^\varepsilon$, hence $\widetilde{w'}$ is b-stable.

We have shown that if there is an execution from $uw_1 \cdots w_m v$ to a b-stable configuration, then there is also one from $uw_1 \cdots w_m z w_1 \cdots w_m v$ to a b-stable configuration, for all $u, v \in \Sigma^*$ and both $b \in \{\top, \bot\}$. By definition of the language of $\mathcal{P}$, this means that $w_1 \cdots w_m$ and $w_1 \cdots w_m z w_1 \cdots w_m$ are equivalent with respect to $\equiv_L$. ◀

▶ **Lemma 15.** *Every* IO-PP[$<$]*-decidable language is regular.*

Proof. We start by showing that some patterns, reminiscent of *sesquipowers* [22], are unavoidable in long strings. An *m-reducible pattern* is a word $w_1 \cdots w_m z w_1 \cdots w_m$ such that $\alpha(w_1) = \cdots = \alpha(w_m) \supseteq \alpha(z)$. We have:

▷ Claim 36. For all alphabet Q of size ℓ and all $m \in \mathbb{N}$, there exists $B(\ell, m) \in \mathbb{N}$ such that for all $w \in Q^*$, if $|w| \geq B(\ell, m)$ then w contains an m-reducible pattern.

Proof. We proceed by induction on the size ℓ of the alphabet Q. If Q is a singleton, then the result is clear with $B(1, m) = 2m + 1$.

Otherwise, define $B(\ell, m) = mB(\ell - 1, m)\ell^{mB(\ell-1,m)}$. Let $w \in Q^*$ be such that $|w| \geq B(\ell, m)$, let u be an infix of w of maximal length such that $|\alpha(u)| < \ell$, and let $M = |u|$.

- If $M \geq B(\ell - 1, m)$, then by induction hypothesis u contains an m-reducible pattern, thus so does w.

- Otherwise, we can split w into $r = B(\ell, m)/B(\ell-1, m) = m\ell^{mB(\ell-1,m)}$ factors of length $B(\ell-1, m)$, plus a suffix v: $w = u_1 \cdots u_r v$. Since $M < B(\ell-1, m)$, by definition of M, each u_k contains all letters in Q. Since $r \geq m\ell^{mB(\ell-1,m)}$, there must exist indices i, j such that $i+m < j$ and $u_i \cdots u_{i+m-1} = u_j \cdots u_{j+m-1}$. As all u_k contain the same set of letters, we obtain an m-reducible pattern $(u_i \cdots u_{i+m-1})(u_{i+m} \cdots u_{j-1})(u_j \cdots u_{j+m-1})$. ◀

By Claim 36 applied with m taken from Lemma 13, there is a bound B such that every word of length more than B is equivalent under $\equiv_L$ to a shorter word. As a consequence, every word is equivalent to a word of length at most B.

This means that $\equiv_L$ has finitely many equivalence classes, yielding the result by the Myhill-Nerode theorem. ◀

A.5 Missing proofs from Section 4.2

▶ **Proposition 16.** *The language* $\{w \in \Sigma^+ \mid |w|_a = 1\}$ *is* IO-PP[$\emptyset$]*-semi-decidable with stabilizing inputs.*

Proof. We define $\mathcal{P} = (Q, \Sigma, \Delta, O)$ by $Q = \Sigma \times \Sigma \times \{\top, \bot\}$, $O((\sigma, \sigma', o)) = (o \wedge \sigma = \sigma') \vee ((\sigma = a \neq \sigma')$, and the following rules:

$$(\sigma, a, o), q \xrightarrow{\textsf{true}} (\sigma, \sigma, \bot), q \qquad \text{for } \sigma \neq a, \tag{2}$$

$$(a, \sigma, o), q \xrightarrow{\textsf{true}} (a, a, \top), q, \tag{3}$$

$$(\sigma, \sigma, o), (a, a, \top) \xrightarrow{\textsf{true}} (\sigma, \sigma, \top), (a, a, \top) \qquad \text{for } \sigma \neq a, \tag{4}$$

$$(\sigma, \sigma, o), (\sigma', \sigma', \bot) \xrightarrow{\textsf{true}} (\sigma, \sigma, \bot), (\sigma', \sigma', \bot), \tag{5}$$

$$(a, a, o), (a, a, o') \xrightarrow{\textsf{true}} (a, a, \bot), (a, a, o'). \tag{6}$$

We identify a with $(a, a, \top)$, and each $\sigma \in \Sigma \setminus \{a\}$ with $(\sigma, \sigma, \bot)$.

Note that $\mathcal{P}$ is input-saving and immediate-observation. Let us show that $\mathcal{P}$ decides the language of the statement with stabilizing inputs. Let $u \in \Sigma^+$ and $u \rightsquigarrow^* v$. Let $v = w_0 \to w_1 \to \cdots$ be a fair run.

If $|v| = 1$, then v is of the form $(a, a, \top)$, $(a, \sigma', \bot)$, $(\sigma, a, \top)$, or $(\sigma, \sigma', \bot)$ where $\sigma, \sigma' \neq a$. By definition of O, the first (resp. last) two states have output $\top$ (resp. $\bot$) as desired.

Suppose $|v| \geq 2$. By fairness and rules (2)–(3), we can assume without loss of generality that each agent has its first two components equal and immutable along the fair run.

Case $|\iota(v)|_a = 0$. Since each $\sigma \neq a$ is identified with $(\sigma, \sigma, \bot)$, and since an a can only disappear via rule (2), there must exist a position i and $\sigma \neq a$ such that $v[i] = (\sigma, \sigma, \bot)$. By fairness and rule (5), a $\bot$-consensus can be reached. From there, all rules are disabled.

Case $|\iota(v)|_a = 1$. The single agent whose first component is a can set its third component to $\top$ with rule (3), and set the third component of other agents to $\top$ with rule (4). From there, all rules are disabled and hence a $\top$-stable configuration has been reached.

Case $|\iota(v)|_a \geq 2$. By fairness, rules (3) and (6) cause the third component of a-agents to alternate indefinitely between $\top$ and $\bot$. ◀

A.6 Missing proofs from Section 5.1

▶ **Lemma 19.** *Any language from* $\Sigma_1^{\text{int}}[<, +, \equiv]$ *is a finite union of languages of the form* $K = \{a_0 w_1 a_1 \cdots w_m a_m \mid w_i \in \Sigma^*, a_i \in \Sigma, \varphi(x_{i,\sigma} \mapsto |w_i|_\sigma, y_{i,\sigma} \mapsto |a_i|_\sigma)\}$ *where* $m \geq 0$ *and* φ *is a Presburger formula over variables* $\{x_{i,\sigma} \mid i \in [1..m], \sigma \in \Sigma\} \cup \{y_{i,\sigma} \mid i \in [0..m], \sigma \in \Sigma\}$.

Proof. Let $\varphi = (\exists x_1, \dots, x_m)[\psi] \in \Sigma_1^{\mathrm{int}}[<, +, \equiv]$ where ψ is quantifier-free. Without loss of generality, ψ may assume that $1 = x_0 < x_1 < \cdots < x_m < x_{m+1} = \mathbf{max}$. Indeed,

- For the first and last positions, we add the two extra variables with these two constraints, and replace each other occurrence of 1 and **max** by x_0 and x_{m+1};
- For the ordering, we can change the formula so that it tests all orderings:

$$\bigvee_{\pi \in S_m} (\exists x_0, x_1, \dots, x_m, x_{m+1})[x_1 \le \cdots \le x_m \wedge \psi_\pi],$$

  where S_m is the set of permutations over $[1..m]$, and ψ_π is ψ with x_i replaced by $x_{\pi(i)}$;
- For distinctness, we can similarly enumerate all possible equivalence classes of "=" and associate a variable to each class, e.g., if we guess that $x_0 < x_1 = x_2 < x_3 = x_4 < x_5$, then we change the subformula to $(\exists y_0, y_1, y_2, y_3)[\psi']$ where ψ' is the formula obtained by renaming variables in ψ as follows: $x_0 \mapsto y_0, \{x_1, x_2\} \mapsto y_1$, $\{x_3, x_4\} \mapsto y_2$ and $x_5 \mapsto y_3$.

Once variables are strictly ordered, we can further assume that each expression $\#_a(x_i, x_j)$ satisfies $j \in \{i, i+1\}$. Indeed, if $i < j$, then $\#_\sigma(x_i, x_j)$ can be substituted with

$$\sum_{\ell=i}^{j-1} \#_\sigma(x_\ell, x_{\ell+1}) - \sum_{\ell=i+1}^{j-1} \#_\sigma(x_\ell, x_\ell).$$

Let ψ' denote the formula ψ where each expression $\#_\sigma(x_i, x_{i+1})$ is replaced with $x_{i,\sigma}$, and each $\#_\sigma(x_i, x_i)$ is replaced with $y_{i,\sigma}$. Consider $K = \{a_0 w_1 a_1 \cdots w_n a_m \mid w_i \in \Sigma^*, a_i \in \Sigma, \psi'(x_{i,\sigma} \mapsto |w_i|_\sigma, y_{i,\sigma} \mapsto |a_i|_\sigma)\}$. We claim that $v \models (\exists x_0, \dots, x_m)[\psi]$ iff $v \in K$.

$\Rightarrow$) We take $a_i = v[x_i]$ for each $i \in [0..m]$, and $w_i = v[x_{i-1}+1..x_i-1]$ for each $i \in [1..m]$. By the assumptions on ψ, it is the case that $\psi'(x_{i,\sigma} \mapsto |w_i|_\sigma, y_{i,\sigma} \mapsto |a_i|_\sigma)$ holds.

$\Leftarrow$) We have $v = a_0 w_1 a_1 w_2 \cdots w_m a_m$ where $w_i \in \Sigma^*$, $a_i \in \Sigma$ and $\psi'(x_{i,\sigma} \mapsto |w_i|_\sigma, y_{i,\sigma} \mapsto |a_i|_\sigma)\}$ holds. By taking $x_i = |a_0 w_1 \cdots a_i|$ for $i \in [0..m]$, $\psi(x_0, \dots, x_m)$ holds. ◀

▶ **Corollary 21.** $\Delta_1^{\mathrm{int}}[<, +, \equiv] \subseteq \mathsf{PP}[<]$.

Proof. Let $L \in \Delta_1^{\mathrm{int}}[<, +, \equiv]$. Since $L \in \Sigma_1^{\mathrm{int}}[<, +, \equiv]$, by Proposition 20, L is $\mathsf{PP}[<]$-semi-decidable. As the complement of L also belongs to $\Sigma_1^{\mathrm{int}}[<, +, \equiv]$, the same applies. Thus, we are done by Lemma 5. ◀

▶ **Proposition 22.** *The language $(ab)^+$ does not belong to $\Sigma_1^{\mathrm{int}}[<, +, \equiv]$.*

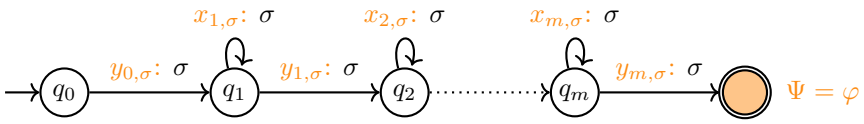
Proof. To derive a contradiction, suppose that $(ab)^+$ belongs to $\Sigma_1^{\mathrm{int}}[<, +, \equiv]$. By Lemma 19, $(ab)^+ = K_1 \cup \cdots \cup K_\ell$ where K_j is of the form $\{a_{j,0} w_{j,1} a_{j,1} \cdots w_{j,m_j} a_{j,m_j} \mid w_{j,i} \in \Sigma^*, a_{j,i} \in \Sigma, \varphi_j(x_{i,\sigma} \mapsto |w_{j,i}|_\sigma, y_{i,\sigma} \mapsto |a_{j,i}|_\sigma)\}$, $m_j \ge 0$ and φ_j is a Presburger formula.

Let $v = (ab)^{n+1}$ where $n = \max(m_1, \dots, m_\ell)$. We have $v \in L$ and so $v \in K_j$ for some $j \in [1..\ell]$. Let $v = a_{j,0} w_{j,1} a_{j,1} \cdots w_{j,m_j} a_{j,m_j}$ be decomposed as in K_j. By $|v| = 2n+2 > 2m_j + 1$ and the pigeonhole principle, there is $i \in [1..m_j]$ with $|w_{j,i}| \ge 2$. Let $w'_{j,i}$ be a word obtained from $w_{j,i}$ by swapping two adjacent letters. Let v' be obtained from v by this change. As the letter counts within $w_{j,i}$ and $w'_{j,i}$ are the same, we have $v' \in K_j$, a contradiction. ◀

A.7 Missing proofs from Section 5.2

▶ **Theorem 23.** *A language is recognized by some poPA iff it belongs to $\Sigma_1^{\mathrm{int}}[<, +, \equiv]$.*

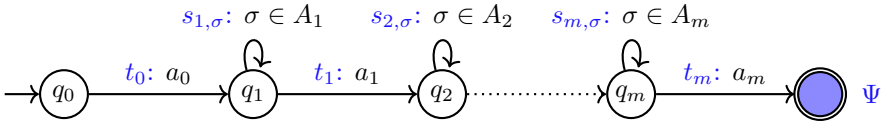
Proof. $\Leftarrow$) Let $L \in \Sigma_1^{\text{int}}[<,+,\equiv]$. By Lemma 19, L can be written as a finite union of languages of the form $K = \{a_0 w_1 a_1 \cdots w_m a_m \mid w_i \in \Sigma^*, a_i \in \Sigma, \varphi(x_{i,\sigma} \mapsto |w_i|_\sigma, y_{i,\sigma} \mapsto |a_i|_\sigma)\}$ where $m \geq 0$ and φ is a Presburger formula. Since poPA are nondeterministic, they are trivially closed under union. Thus, it suffices to give a poPA for K, which we do below. Here, each transition labeled with "σ" stands for $|\Sigma|$ distinct transitions:



$\Rightarrow$) Let $\mathcal{A}$ be a poPA. Let K be the words of $L(\mathcal{A})$ of length at most one. We can construct a formula for each word of K:

$$\varphi_\varepsilon = \bigwedge_{\sigma \in \Sigma} \#_\sigma(1, \mathbf{max}) = 0 \text{ and } \varphi_a = \#_a(1, \mathbf{max}) = 1 \wedge \bigwedge_{\sigma \neq a} \#_\sigma(1, \mathbf{max}) = 0.$$

Let us now consider $K' = L(\mathcal{A}) \setminus K$. As $\mathcal{A}$ is partially ordered, its language can be described as the finite union of poPA organized as straight lines alternating between self-loops and progress transitions. As we consider K' (and so words of length at least 2), we can explicitly read the first and last letters, which yields automata of the following form, where $a_i \in \Sigma$ and $A_i \subseteq \Sigma$:



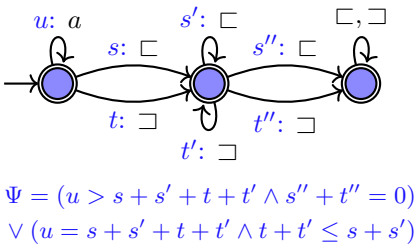
We convert such an automaton into this formula:

$$(\exists x_0, x_1, \ldots, x_m)[1 = x_0 < x_1 < \cdots < x_m = \mathbf{max}] \wedge \bigwedge_{i=0}^{m} \#_{a_i}(x_i, x_i) = 1 \wedge$$
$$\bigwedge_{i=1}^{m} \sum_{\sigma \in \Sigma \setminus A_i} \#_\sigma(x_{i-1}+1, x_i - 1) = 0 \wedge \Psi(t_i \mapsto 1, s_{i,\sigma} \mapsto \#_\sigma(x_{i-1}+1, x_i - 1)).$$

We are done by taking the disjunction of all formulas. ◀

▶ **Observation 24.** *If weakly-unambiguous poPA are closed under complement, then any language recognized by a weakly-unambiguous poPA belongs to* $\Delta_1^{\text{int}}[<,+,\equiv]$.

Proof. First, observe that the two automata of Figure 2 for the median and coDyck-witness languages can be complemented as follows, while remaining weakly unambiguous:



In general, let $\mathcal{A}$ and $\mathcal{A}'$ be poPA with $L(\mathcal{A}') = \overline{L(\mathcal{A})}$. By Theorem 23, $L(\mathcal{A}), L(\mathcal{A}') \in \Sigma_1[<,+,\equiv]$. Thus, $L(\mathcal{A}) = \overline{L(\mathcal{A}')} \in \Pi_1[<,+,\equiv]$ and so $L(\mathcal{A}) \in \Delta_1[<,+,\equiv]$. ◀

A.8 Missing proofs from Section 6.1

▶ **Lemma 27.** *Every language semi-decided by a protocol in* PP[+1] *is also semi-decided by a protocol in* IO-PP[+1].

Proof. Let $\mathcal{P} = (Q, \Sigma, O, \Delta)$ be a protocol in PP[+1]. We construct a protocol $\mathcal{P}' = (Q', \Sigma, O', \Delta')$ in IO-PP[+1] such that $L(\mathcal{P}') = L(\mathcal{P})$.
States and outputs. For each transition $\delta = (a, b) \to (c, d)$ in Δ, we introduce auxiliary states a^δ and $b^{\mathsf{ack},\delta}$. Formally, $Q' = Q \cup \{\, a^\delta,\ b^{\mathsf{ack},\delta} \mid \delta = (a, b) \to (c, d) \in \Delta \,\}$. The output function ignores the auxiliary markers: $O'(q) = O(q)$, $O'(a^\delta) = O(a)$ and $O'(b^{\mathsf{ack},\delta}) = O(b)$.
Transitions. For each $\delta = (a, b) \to (c, d) \in \Delta$, the set Δ' contains these four transitions:

$$
\begin{aligned}
a, b &\xrightarrow{+1} a^\delta, b \\
a^\delta, b &\xrightarrow{+1} a^\delta, b^{\mathsf{ack},\delta} \\
a^\delta, b^{\mathsf{ack},\delta} &\xrightarrow{+1} c, b^{\mathsf{ack},\delta} \\
c, b^{\mathsf{ack},\delta} &\xrightarrow{+1} c, d.
\end{aligned}
$$

Each of these steps updates at most one of the two agents at a time, hence $\mathcal{P}'$ is an immediate-observation protocol. To avoid blocking in the presence of overlapping partial simulations, we also include the following "clean-up" rules:

$$
\begin{aligned}
a^\delta, x &\xrightarrow{+1} a, x && \text{for } x \neq b^{\mathsf{ack},\delta}, \\
y, b^{\mathsf{ack},\delta} &\xrightarrow{+1} y, d && \text{for } y \neq a^\delta.
\end{aligned}
$$

Step simulation.

▷ Claim 37. If $u, v \in Q^*$ and $u \to v$ in $\mathcal{P}$, then $u \to^* v$ in $\mathcal{P}'$.

Proof. Let $\delta = (a, b) \xrightarrow{+1} (c, d)$ be the transition used in $\mathcal{P}$. Let $u = u_1 ab\, u_2$ and $v = u_1 cd\, u_2$. In $\mathcal{P}'$, the four transitions associated with δ yield $u_1 ab\, u_2 \to u_1 a^\delta b\, u_2 \to u_1 a^\delta b^{\mathsf{ack},\delta}\, u_2 \to u_1 c\, b^{\mathsf{ack},\delta}\, u_2 \to u_1 cd\, u_2$. ◀

Projection to $\mathcal{P}$. We define a projection $f \colon Q'^* \to Q^*$ that interprets a $\mathcal{P}'$-configuration as a $\mathcal{P}$-configuration. For $w = w_1 \cdots w_n$, define $f(w) = w'_1 \cdots w'_n$ by:

- if $w_i \in Q$, then $w'_i = w_i$;
- if $w_i = a^\delta$ for some $\delta = (a, b) \to (c, d)$ and $i < n$ with $w_{i+1} = b^{\mathsf{ack},\delta}$, then $w'_i = c$;
- if $w_i = a^\delta$ but the above condition fails, then $w'_i = a$;
- if $w_i = b^{\mathsf{ack},\delta}$ for some $\delta = (a, b) \to (c, d)$, then $w'_i = d$.

▷ Claim 38. Let $u \in Q^*$ and $v \in Q'^*$. If $u \to^* v$ in $\mathcal{P}'$, then $u \to^* f(v)$ in $\mathcal{P}$.

Proof. We prove the claim by induction on the length of $u \to^* v$ in $\mathcal{P}'$. The only step that can change the projected configuration is the acknowledgement step $(a^\delta, b) \xrightarrow{+1} (a^\delta, b^{\mathsf{ack},\delta})$, in which case f precisely applies the corresponding transition $\delta = (a, b) \xrightarrow{+1} (c, d)$ on the projected word. All other rules leave $f(\cdot)$ unchanged. ◀

▷ Claim 39. For every $v \in Q'^*$ reachable in $\mathcal{P}'$, it is the case that $v \to^* f(v)$ in $\mathcal{P}'$.

Proof. Starting from v, repeatedly apply the clean-up rules to remove every pending marker a^δ not followed by the matching $b^{\mathsf{ack},\delta}$, and then complete each matching pair $a^\delta b^{\mathsf{ack},\delta}$ by the two remaining simulation steps $(a^\delta, b^{\mathsf{ack},\delta}) \xrightarrow{+1} (c, b^{\mathsf{ack},\delta}) \xrightarrow{+1} (c, d)$. This yields $f(v)$. ◀

We are now ready to prove the lemma. Let us show that $w \in L(\mathcal{P})$ iff $w \in L(\mathcal{P}')$.

$\Rightarrow$) Let $w \to^* u$ in $\mathcal{P}'$. Thanks to Claim 38, we know that $w \to^* f(u)$ in $\mathcal{P}$. Since $w \in L(\mathcal{P})$, configuration $f(u)$ can reach a $\top$-stable configuration v in $\mathcal{P}$. Hence, thanks to Claim 39 and Claim 37, we conclude that $u \to^* f(u) \to^* v$ in $\mathcal{P}'$. To derive a contradiction, suppose that v is not $\top$-stable in $\mathcal{P}'$. Configuration v can reach a non-$\top$-consensus v' in $\mathcal{P}'$. Consider v' to be such a configuration reachable in a minimal number of steps. The last step must be a transition of the form $(a^\delta, b^{\mathsf{ack},\delta}) \xrightarrow{+1} (c, b^{\mathsf{ack},\delta})$ with $O'(c) = O(c) = \bot$, or $(c, b^{\mathsf{ack},\delta}) \xrightarrow{+1} (c, d)$, with $O'(d) = O(d) = \bot$. Hence, $f(v')$ is not a $\top$-consensus in $\mathcal{P}$. Therefore, $f(v) = v \to^* f(v')$ in $\mathcal{P}$ by Claim 38, contradicting the $\top$-stability of v in $\mathcal{P}$.

$\Leftarrow$) Let $w \to^* u$ in $\mathcal{P}$. By Claim 37, $w \to^* u$ in $\mathcal{P}'$. As $w \in L(\mathcal{P}')$, configuration u can reach a $\top$-stable configuration v in $\mathcal{P}'$. From Claim 39, we have $v \to^* f(v)$ in $\mathcal{P}'$. Note that $f(v)$ is $\top$-stable in $\mathcal{P}'$ by construction. Moreover, from Claim 38, we have $u \to^* f(v)$ in $\mathcal{P}$. To derive contradiction, suppose that $f(v)$ is not $\top$-stable in $\mathcal{P}$. We have $f(v) \to^* v'$ in $\mathcal{P}$ for some non-$\top$-consensus v'. Thanks to Claim 37, we obtain $f(v) \to^* v'$ in $\mathcal{P}'$, contradicting the $\top$-stability of $f(v)$ in $\mathcal{P}'$. Hence, $f(v)$ is $\top$-stable in $\mathcal{P}$, and so $w \in L(\mathcal{P})$. ◀

A.9 Missing proofs from Section 6.2

▶ **Lemma 28.** *The language of a linear-bounded nondeterministic Turing machine is* PP[+1]*-semi-decidable.*

Proof. Let M be a linear-bounded nondeterministic Turing machine over an input alphabet Σ containing fixed endmarkers L and R, which are never overwritten. Thus, any accepted input has the form $L\,w\,R$ with no endmarker within w. We define M_{rst} as M equipped with a *reset* operation. Each non-endmarker cell stores a pair (x, a) consisting of its current symbol x and its initial input symbol a. Initially, M_{rst} replaces each $a \in \Sigma \setminus \{L, R\}$ by (a, a). From any non-accepting state, the machine may enter a reset state, scan the tape and restore every cell (x, a) to (a, a), return to L, and restart in the initial state. Moreover, upon reaching an accepting state, M_{rst} performs a final sweep that overwrites the tape with a designated accepting symbol f.

▶ **Lemma 40.** *For every $w \in \Sigma^*$, w is accepted by M if and only if for every configuration u reachable from w in M_{rst}, there exists a run of M_{rst} from u to an accepting configuration.*

Notations. Let S be the set of states of M_{rst}, let Γ be its tape alphabet, and let $\Sigma \subseteq \Gamma$ be its input alphabet. Write s_0 for the initial state, $F \subseteq S$ for the set of accepting states, and $T \subseteq S \times \Gamma \times \{\triangleright, \triangleleft\} \times S \times \Gamma$ for the transition relation, where "$\triangleright$" and "$\triangleleft$" respectively denote left and right head movements. Let $\Gamma_w = \Gamma \setminus \{L, R\}$ denote the set of non-endmarker symbols. An M_{rst}-configuration is a word over Γ with exactly one head marker, i.e., an element of

$$\bigcup_{s \in S} (L, s)\Gamma_w^* R \;\cup\; \bigcup_{x \in \Gamma_w, s \in S} L\Gamma_w^*(x, s)\Gamma_w^* R \;\cup\; L\Gamma_w^* \bigcup_{s \in S} (R, s).$$

Protocol $\mathcal{P}_M$. Let us first define a protocol $\mathcal{P}'_M$ in PP[+1] which, for all $w = w_1 \cdots w_n \in \Sigma^*$ and $w' = (\underline{L}, w_1, w_1)(w_2, w_2) \cdots (w_{n-1}, w_{n-1})(w_n, w_n, \overline{R})$, accepts w' iff M accepts w. Intuitively, $\mathcal{P}'_M$ takes as input a word w where the leftmost and rightmost agents know that they are, and are in charge of simulating both themselves and the endmarker cells.

We will later expand $\mathcal{P}'_M$ into a protocol $\mathcal{P}_M$, with rules which allow for agents to guess if they are first or last; in $\mathcal{P}_M$, a word w will keep resetting itself until agents have rightfully guessed if they are first or last.

State space. Let $TS = \Gamma \cup (S \times \Gamma)$ be the state of possibly head-marked tape symbols. We use the following state forms:

- the machine input alphabet Σ;
- internal cell states (b, γ) where $b \in \Sigma$ is the stored input letter and $\gamma \in TS$ is the current tape symbol;
- two-sided boundary states $(\underline{\gamma}, b, \gamma', \overline{\gamma''})$; an agent in this state guesses that it is both the first and only agents. It stores its internal cell state as b, γ', and the state of the left and right endmarker as γ and γ''. The notations $\underline{\gamma}$ and $\overline{\gamma''}$ allows us to distinguish easily where the endmarker states are stored.
- left-boundary states $(\underline{\gamma}, b, \gamma')$ and right-boundary states $(b, \gamma, \overline{\gamma'})$ should be interpreted in the same way.
- reset tokens $\mathsf{rst\text{-}left}^b$ and $\mathsf{rst\text{-}right}^b$.

Encoding. We encode the machine configuration $x_L\, x_1 \cdots (x_i, s) \cdots x_n\, x_R$ reachable on input $b_1 \cdots b_n$ as the protocol configuration

$$(\underline{x_L}, b_1, x_1)\,(b_2, x_2) \cdots (b_i, (x_i, s)) \cdots (b_n, x_n, \overline{x_R}).$$

Rules of $\mathcal{P}'_M$.

Starting the simulation. An agent currently carrying the left boundary component may initiate the simulation by placing the head on its tape symbol:

$$(\underline{L}, *, *), * \xrightarrow{\mathsf{true}} (\underline{(L', s_0)}, *, *), *.$$

We introduce symbol L' to prevent a same left endmarker to start several machine execution.

Simulation of the machine. We simulate M_{rst} on the tape components, leaving the stored input letters unchanged. Below, $*$ stands for an arbitrary boundary component (possibly absent). For each transition $(q, x, \triangleright, p, y) \in T$, the set Δ contains:

$$\begin{aligned}
(\underline{(q, x)}, a, z, *), * &\xrightarrow{\mathsf{true}} (\underline{y}, a, (p, z), *), * \\
(*, a, (q, x), \overline{z}), * &\xrightarrow{\mathsf{true}} (*, a, y, \overline{(p, z)}), * \\
(*, a, (q, x)),\ (b, z) &\xrightarrow{+1} (*, a, y),\ (b, (p, z)).
\end{aligned}$$

For transitions of the form $(q, x, \triangleleft, p, y) \in T$, the set Δ contains analogous transitions.

Final sweeping. Let $Acc = \{f\} \cup (F \times \Gamma)$. We further include these rules:

$$\begin{aligned}
(*, a, \gamma), (b, \gamma', *) &\xrightarrow{+1} (*, a, f), (b, f, *) && \text{if } \gamma \text{ or } \gamma' \in Acc, \\
(\underline{\gamma}, a, \gamma', *), * &\xrightarrow{\mathsf{true}} (\underline{f}, a, f, *), * && \text{if } \gamma \text{ or } \gamma' \in Acc, \\
(*, a, \gamma, \overline{\gamma'}), * &\xrightarrow{\mathsf{true}} (*, a, f, \overline{f}), * && \text{if } \gamma \text{ or } \gamma' \in Acc.
\end{aligned}$$

Opinion function. We define O as follows:

$$\begin{aligned}
O(b, \gamma) = \top &\iff \gamma \in Acc, \\
O(\underline{\gamma}, a, \gamma') = \top &\iff \gamma, \gamma' \in Acc, \\
O(a, \gamma, \overline{\gamma'}) = \top &\iff \gamma, \gamma' \in Acc, \\
O(\underline{\gamma}, a, \gamma', \overline{\gamma''}) = \top &\iff \gamma, \gamma', \gamma'' \in Acc,
\end{aligned}$$

where all other states have opinion $\bot$.

Expanding $\mathcal{P}'_M$ into $\mathcal{P}_M$.
Initialization. On input letter $a \in \Sigma \setminus \{L, R\}$, an agent initially guesses it is both the first and the last cell, and stores its input symbol:

$$a, * \xrightarrow{\textsf{true}} (\underline{L}, a, a, \overline{R}), *.$$

Detecting an inconsistency. Below, "$\square$" stands for an arbitrary tape symbol in TS. Whenever two adjacent agents carry incompatible boundary information, we enter a reset mode:

$$\begin{aligned}
(b, \square),\ (\underline{\square}, a, \square) &\xrightarrow{+1} \textsf{rst-left}^b,\ \textsf{rst-right}^a && \text{(mismatch on the left)}\\
(\underline{\square}, b, \square),\ (\underline{\square}, a, \square) &\xrightarrow{+1} (\underline{L}, b, b),\ \textsf{rst-right}^a && \text{(two left-boundaries)}\\
(a, \square, \overline{\square}),\ (b, \square) &\xrightarrow{+1} \textsf{rst-left}^a,\ \textsf{rst-right}^b && \text{(mismatch on the right)}\\
(a, \square, \overline{\square}),\ (b, \square, \overline{\square}) &\xrightarrow{+1} \textsf{rst-left}^a,\ (b, b, \overline{R}) && \text{(two right-boundaries)}\\
(\underline{\square}, a, \square, \overline{\square}),\ (\underline{\square}, b, \square, \overline{\square}) &\xrightarrow{+1} (\underline{L}, a, a),\ (b, b, \overline{R}) && \text{(two-sided vs two-sided)}\\
(\underline{\square}, a, \square, \overline{\square}),\ (\underline{\square}, b, \square) &\xrightarrow{+1} (\underline{L}, a, a),\ \textsf{rst-right}^b && \text{(two-sided vs left)}\\
(a, \square, \overline{\square}),\ (\underline{\square}, b, \square, \overline{\square}) &\xrightarrow{+1} \textsf{rst-left}^a,\ (b, b, \overline{R}) && \text{(right vs two-sided)}\\
(a, \square, \overline{\square}),\ (\underline{\square}, b, \square) &\xrightarrow{+1} \textsf{rst-left}^a,\ \textsf{rst-right}^b && \text{(right vs left).}
\end{aligned}$$

Reset propagation. A right-moving token restores the stored input symbol and moves one step to the right:

$$\begin{aligned}
\textsf{rst-right}^a,\ (*, b, \square) &\xrightarrow{+1} (a, a),\ \textsf{rst-right}^{(b)} && \text{(propagate right)}\\
\textsf{rst-right}^a,\ \textsf{rst-left}^b &\xrightarrow{+1} \textsf{rst-left}^b,\ \textsf{rst-right}^a && \text{(tokens cross).}
\end{aligned}$$

The left-moving reset token rst-left is symmetric.

End of propagation. When rst-right (resp. rst-left) reaches a right (resp. left) boundary, it restores the right-boundary (resp. left-boundary) component:

$$\begin{aligned}
&\textsf{rst-right}^a,\ (*, b, \square, \overline{\square}) \xrightarrow{+1} (a, a),\ (b, b, R)\\
&(\underline{\square}, b, \square, *),\ \textsf{rst-left}^a \xrightarrow{+1} (L, b, b), (a, a).
\end{aligned}$$

▶ **Lemma 41.** *$\mathcal{P}_M$ is a semi-decider of $L(M)$.*

Proof. Let $w \in \Sigma^*$. It suffices to show that $w \in L(M)$ if and only this condition holds:

(P1) For every partial run of $\mathcal{P}_M$ from w reaching a configuration c, there exists a run from c to a $\top$-stable configuration.

On the other hand, by Lemma 40, we have that $w \in L(M)$ if and only if:

(P2) For every configuration u reachable from w in $M_{\textsf{rst}}$, there exists a run of $M_{\textsf{rst}}$ from u to an accepting configuration.

We prove that (P1) and (P2) are equivalent.

(P1) ⇒ (P2). Assume (P1), and let u be any configuration reachable from w in $M_{\textsf{rst}}$. Write $w = a_1 \cdots a_n$. Consider the following run of $\mathcal{P}_M$:

$$\begin{aligned}
a_1 \cdots a_n &\to^+ (\underline{L}, a_1, a_1, \overline{R}) \cdots (\underline{L}, a_n, a_n, \overline{R})\\
&\to\ (\underline{L}, a_1, a_1)\,(a_2, a_2, \overline{R}) \cdots (\underline{L}, a_n, a_n, \overline{R})\\
&\to\ (\underline{L}, a_1, a_1)\,\textsf{rst-left}^{a_2}\,(a_3, a_3, \overline{R}) \cdots (\underline{L}, a_n, a_n, \overline{R})
\end{aligned}$$

$$\to^+ (\underline{L}, a_1, a_1)\,\textsf{rst-left}^{a_2}\,\textsf{rst-left}^{a_3}\cdots\textsf{rst-left}^{a_{n-1}}\,(a_n, \overline{R})$$
$$\to^+ (\underline{L}, a_1, a_1)\,(a_2, a_2)\cdots(a_{n-1}, a_{n-1})\,(a_n, a_n, \overline{R}).$$

Let w' denote the last configuration above. From it, we follow the simulation rules of $\mathcal{P}'_M$ so as to reproduce a machine run from w to u in $M_{\textsf{rst}}$, thus reaching a protocol configuration $\tilde{u}$ that encodes u. By (P1), there exists a partial run from $\tilde{u}$ to a $\top$-stable configuration.

By construction, from $\tilde{u}$ no inconsistency can be detected (the boundary components are consistent and no reset token is present), and the simulation has already been started on the left. Hence, every enabled transition from $\tilde{u}$ is a simulation step of $M_{\textsf{rst}}$. Therefore, the above fair run projects to a run of $M_{\textsf{rst}}$ starting from u. Since the target configuration is $\top$-stable, the machine must eventually enter an accepting state and perform its final sweeping phase, yielding a configuration in which all tape symbols are f. In particular, u reaches an accepting configuration in $M_{\textsf{rst}}$, proving (P2).

(P2) $\Rightarrow$ (P1). Assume (P2). Let $c = c_1 \cdots c_n$ be any protocol configuration reachable from the input $w = a_1 \cdots a_n$. We show that there exists a fair continuation from c to a $\top$-stable configuration. First, if c still contains some agent in a state $a \in \Sigma$, we turn them into symbols $(\underline{L}, a, a, \overline{R})$. Observe that, once initialized, the leftmost (resp. rightmost) agent can never lose its left-boundary (resp. right-boundary) information. Furthermore, using repeatedly the reset-propagation rules (namely the rules moving rst-right one step to the right and rst-left one step to the left, together with the token-crossing rule), we can reach a configuration with no reset token.

Case 1: c contains wrong boundary components. We first eliminate them by repeated resets. Let $N(c)$ be the number of *wrong* boundary components, namely

$$N(c) = \big|\{1 < i \le n \mid c_i \text{ carries a left-boundary component } \underline{\gamma}\}\big| + \big|\{1 \le i < n \mid c_i \text{ carries a right-boundary component } \overline{\gamma}\}\big|.$$

We claim that if $N(c) > 0$, then c has a continuation to some configuration c' such that $N(c') < N(c)$, and, moreover, the whole segment between the closest left boundary to the left, and the closest right boundary to the right is reset (i.e., its tape components become of the form (a, a)).

To prove the claim, pick a wrong left boundary component; that is, choose a position $i > 1$ such that c_i carries $\underline{\gamma}$. Then c_i can interact with its predecessor at position $i - 1$, yielding a successor configuration

$$c_1 \cdots c_{i-2}\, c'_{i-1}\, c'_i\, c_{i+1} \cdots c_n.$$

By inspection of the inconsistency rules of $\mathcal{P}_M$, the pair (c'_{i-1}, c'_i) is necessarily of one of the following forms:

- $\textsf{rst-left}^y$, $\textsf{rst-right}^z$: we propagate rst-left to the nearest left boundary and rst-right to the nearest right boundary, resetting the whole intermediate segment;
- $(\underline{s}, b, b)$, $\textsf{rst-right}^z$: here position $i - 1$ is the nearest left boundary to the left of i, and propagating rst-right to the nearest right boundary resets the segment to its right;
- $(\underline{s}, a, a)$, $(b, b, \overline{s'})$: in this case both boundaries are restored immediately around the cut, and the enclosed segment is reset.

In all cases, after the (finite) propagation of reset tokens, we reach a configuration c' such that $N(c') < N(c)$ and the affected segment has been reset to (a, a) symbols. Repeating this

process, we eventually reach a configuration with $N(\cdot) = 0$, hence with correct boundary placement. In particular, we can reach a configuration of the form

$$c_{\text{init}} = (\underline{L}, a_1, a_1)\,(a_2, a_2)\cdots(a_{n-1}, a_{n-1})\,(a_n, a_n, \overline{R}).$$

From c_{init}, the protocol can simulate, step by step, any run of M_{rst} on input w (using only simulation rules, as no inconsistency rule is enabled anymore). By (P2), from the corresponding machine configuration there exists a run to an accepting configuration; simulating it yields a protocol run to a configuration in which the machine is accepting and the sweeping phase has written f everywhere, i.e., a configuration of the form

$$(\underline{\gamma_0}, a_1, \gamma_1)\,(a_2, \gamma_2)\cdots(a_{n-1}, \gamma_{n-1})\,(a_n, \gamma_n, \overline{\gamma_{n+1}}) \qquad \text{with } \gamma_i \in \mathit{Acc} \text{ for all } i.$$

From such a configuration, no inconsistency can be detected and no simulation step is enabled, hence it is terminal; since every agent has opinion $\top$, it is $\top$-stable. Hence, $w \in L(\mathcal{P}_M)$.

Case 2: c has correct boundary placement. Assume c has a unique left-boundary component at position 1 and a unique right-boundary component at position n. We can write

$$c = (\underline{\gamma_0}, a_1, \gamma_1)\; c'\; (a_n, \gamma_n, \overline{\gamma_R}),$$

where the middle factor c' carries no boundary component. Using repeatedly the reset-propagation rules (namely the rules moving rst-right one step to the right and rst-left one step to the left, together with the token-crossing rule), we can reach a configuration with no reset token, of the form

$$v = (\underline{\gamma_L}, a_1, \gamma_1)\,(a_2, \gamma_2)\cdots(a_{n-1}, \gamma_{n-1})\,(a_n, \gamma_n, \overline{\gamma_R}).$$

We claim that v contains *at most one* head-marked tape symbol (q, x). Assume for contradiction that v contains two head markers. Since the only rules that *create* a head marker are the machine-start rules at the left boundary (of the form $(\underline{L}, *) \to (\underline{(L', s_0)}, *)$ and the subsequent simulation rules), this can only happen in one of the following ways:

1. two *distinct* positions carrying a left-boundary component have started a machine execution (possibly before some boundary components were later removed), or
2. the *same* left-boundary position has started the execution twice.

We call them *starters*.

(1) Two distinct starters. Let p be the rightmost starter. The corresponding head marker must lie at or to the right of p (it is created at p and then moves by simulation rules). Since v has correct boundary placement, the left-boundary component that was present at p at start time must have been removed at some point. By the definition of the inconsistency-detection rules, removing a left-boundary component that is not at position 1 can only occur via one of the following rule patterns:

a) a rule that creates a right-moving reset token, i.e., a transition whose right-hand side contains rst-right$^{\cdot}$, or
b) a rule that restores a right boundary locally, i.e., a transition whose right-hand side contains a state of the form $(\square, \square, \overline{R})$.

In case (a), the created rst-right token is propagated until it reaches the next right boundary using the reset-propagation rules, and the end-of-propagation rule applies at that boundary. Along this propagation, the whole traversed segment is reset, and in particular any head marker within that segment is removed (the simulated head never crosses a right boundary

component). In case (b), the starter position becomes (or is adjacent to) a right boundary, hence the head marker cannot move past it by any simulation rule, and it is eliminated when that boundary-restoration rule fires. In both subcases, the head marker created by p cannot survive up to v, a contradiction.

(2) The same starter starts twice. For the same left-boundary position to start twice, it must have been reset in between (otherwise the machine-start rule would not be enabled again at that position). By construction, any reset affecting the left boundary can only occur through one of the following rule families:

a) creation of a rst-right token at or next to the left boundary (a rule whose right-hand side contains $\mathsf{rst\text{-}right}^a$),
b) a two-sided inconsistency rule producing both restored boundaries in one step (a rule whose right-hand side contains both $(\underline{L}, \square, \square)$ and $(\square, \square, \overline{R})$),
c) interaction between the left boundary and a left-moving token, i.e., a rule of the form $(\underline{\square}, \square, \square),\ \mathsf{rst\text{-}left}^a \xrightarrow{+1} (\underline{L}, \square, \square),\ (a, a)$.

We show that in each case, any existing head marker is deleted.

(a) Creation of rst-right. Consider the step that creates $\mathsf{rst\text{-}right}^a$ at the left boundary. Immediately before this step, the head marker cannot lie strictly to the left of that boundary. If it lay strictly to the right while the boundary component still existed, then to reach the left side it would have to cross a left boundary component by a simulation rule, which is impossible since simulation rules never move the head across boundary components. Therefore the head marker must be at the boundary position (or in the reset segment adjacent to it) and is removed when the reset is created and propagated.

(b) Two-sided inconsistency. If a two-sided inconsistency rule fires and produces a restored left boundary $(\underline{L}, \square, \square)$ and a restored right boundary $(\square, \square, \overline{R})$ in one step, then the head marker cannot lie beyond either boundary (simulation rules never cross boundary components). Hence the head marker lies inside the segment that is reset by that rule and is deleted.

(c) Interaction with rst-left. Before reaching the left boundary, a token rst-left is created by an inconsistency involving a right boundary component (i.e., by one of the inconsistency-detection rules whose right-hand side contains $\mathsf{rst\text{-}left}^a$). During the subsequent propagation to the left (by the reset-propagation rules), the token resets every visited position. Since the head marker cannot cross a right boundary component, it must lie within the segment traversed by rst-left, and is therefore deleted when the token passes.

In all cases we reach a contradiction. Therefore, v contains at most one head marker.

Restarting a clean simulation. From v, we can use the internal reset mechanism of M_{rst} (as simulated by $\mathcal{P}_M$) to reset the tape contents to the stored input and restart from the initial machine configuration

$$(\underline{(L, s_0)}, a_1, a_1)\,(a_2, a_2) \cdots (a_{n-1}, a_{n-1})\,(a_n, a_n, \overline{R}),$$

provided the head is not already in an accepting state. If it is already accepting, then by our assumption on M_{rst}, the machine performs its sweeping phase and writes f everywhere, yielding $\top$-stable configuration of the form

$$(\underline{\gamma_0}, a_1, \gamma_1)\,(a_2, \gamma_2) \cdots (a_{n-1}, \gamma_{n-1})\,(a_n, \gamma_n, \overline{\gamma_{n+1}}) \qquad \text{with } \gamma_i \in \mathit{Acc} \text{ for all } i$$

Otherwise, starting from the clean initial configuration, we simulate (step by step) a run of M_{rst} that reaches an accepting configuration, which exists by (P2). Simulating its sweeping phase again yields a $\top$-stable configuration as above. Hence, $w \in L(\mathcal{P}_M)$. ◀

A.10 Missing proofs from Section 7.1

▶ **Lemma 30.** *The emptiness problem for* PP[$\mathcal{N}$] *(resp.* IO-PP[$\mathcal{N}$]*) reduces to deciding the complement of the syntax of* PP[$\mathcal{N}$] *(resp.* IO-PP[$\mathcal{N}$]*) deciders.*

Proof. Take a protocol $\mathcal{P} = (Q, \Sigma, O, \Delta)$. We build a protocol $\mathcal{P}'$ such that $\mathcal{P}'$ is *not* a decider if and only if a $\top$-stable configuration is reachable in $\mathcal{P}$.

If Σ contains a letter a with opinion $\top$, then the configuration a is $\top$-stable and reachable. In that case, we set $\mathcal{P}'$ as an arbitrary protocol which is not a decider.

Otherwise, we build $\mathcal{P}' = (Q', \overline{\Sigma}, O', \Delta')$ from $\mathcal{P}$ as follows: For each initial state $a \in \Sigma$ in $\mathcal{P}$, we add another state $\overline{a}$ with opinion $\bot$. We denote $\overline{\Sigma}$ the set of those additional states. We also add another state $q_\bot$ with opinion $\bot$. The set of initial states of $\mathcal{P}'$ is $\overline{\Sigma}$. For each $\overline{a} \in \overline{\Sigma}$ and $q \in Q'$ we add a transition $(\overline{a}, q) \xrightarrow{\mathsf{true}} (a, q)$. As a consequence, from every initial configuration $\overline{a_1} \cdots \overline{a_k}$ with $k \geq 2$, we can reach its counterpart in $a_1 \cdots a_k$ in Σ^*.

Further, for every pair of states q_1, q_2 such that at least one of the two has opinion $\bot$, we add a transition $(q_1, q_2) \xrightarrow{\mathsf{true}} (q_1, q_\bot)$. Note that by this last transition, from every configuration of length ≥ 2 which is not a $\top$-consensus, we can reach a configuration of ${q_\bot}^*$.

Suppose that there is a run in $\mathcal{P}$ from an initial configuration $u = a_1 \cdots a_k \in \Sigma^+$ to a $\top$-stable configuration v in $\mathcal{P}$. We must have $k \geq 2$ since we assumed that all $a \in \Sigma$ have opinion $\bot$. In $\mathcal{P}'$, the configuration $\overline{u} = \overline{a}_1 \cdots \overline{a}_k$ is initial, and it can reach both ${q_\bot}^k$ and v, respectively a $\bot$ and a $\top$-stable configuration. As a result, the protocol is not a decider.

Now suppose that a $\top$-stable configuration is unreachable in $\mathcal{P}$. Let u be an initial configuration and let $u \to^* v$. Since v is not $\top$-stable, we have $v \to^* w$ for some configuration w containing a state with opinion $\bot$. Thus $w \to^* q_\bot^*$ and the latter is a $\bot$-consensus. As a result, the protocol is a decider (and recognizes the empty language).

Finally, observe that if $\mathcal{P}$ is immediate-observation then so is $\mathcal{P}'$. ◀

▶ **Corollary 31.** *The syntax of* IO-PP[+1] *and* PP[+1] *deciders are undecidable.*

Proof. In Section 6, we presented an effective construction translating a linear-bounded Turing machine into an IO-PP[+1] protocol with the same language. Since emptiness is undecidable for linear-bounded Turing machines [31, Thm. 5.10], it must also be undecidable for IO-PP[+1] protocols. By Lemma 30, so is the syntax of IO-PP[+1] deciders. ◀

▶ **Theorem 32.** *The emptiness problem is undecidable for* PP[<]. *Hence the syntax of* PP[<] *deciders is also undecidable.*

Proof. We reduce from the Post correspondence problem (PCP), which is undecidable. We are given two finite alphabets A, B, and two homomorphisms $h_1, h_2 \colon B^* \to A^*$, and must determine whether there exists $w \in B^*$ such that $h_1(w) = h_2(w)$.

We construct three protocols, each in charge of verifying a property of the initial word:

- $\mathcal{P}_0$ checks that the input word is in $\#_1 A^* \#_2 B^* \#_3$;
- $\mathcal{P}_1$ checks that $h_1(u) = v$ for a word $\#_1 v \#_2 u \#_3$;
- $\mathcal{P}_2$ checks that $h_2(u) = v$ for a word $\#_1 v \#_2 u \#_3$.

For the first one, we simply use the fact that $\#_1 A^* \#_2 B^* \#_3$ is in DA, and hence recognized by an (immediate-observation) protocol $\mathcal{P}_0$. In particular an input word can reach a $\top$-stable configuration if and only if it is in that language.

We now construct $\mathcal{P}_i$ for $i \in \{1, 2\}$. The set of states of both protocol is

$$\{a, a^{head}, a^{next} \mid a \in A\} \cup\ B \cup \{(b, \pi) \mid b \in B, \pi \text{ prefix of } h(b)\} \cup\ \{\#_1, \#_2, \overline{\#_2}, \#_3, q_\top\}.$$

The opinion of $q_\top$ is $\top$, all other states have opinion $\bot$.

Protocol rules are as follows, for all $a, c \in A$, $b, d \in B$ and prefix π of $h_i(b)$:

Start	***Head***
$\#_1, a \xrightarrow{\leq} q_\top, a^{head}$	$a^{head}, (b, \pi) \xrightarrow{\leq} a^{next}, (b, \pi a)$
$\#_2, b \xrightarrow{\leq} \overline{\#_2}, (b, \varepsilon)$	if πa is a prefix of $h_i(b)$
Next (A)	***Next (B)***
$a^{next}, c \xrightarrow{\leq} q_\top, c^{head}$	$(b, h_i(b)), d \xrightarrow{\leq} q_\top, (d, \varepsilon)$
$a^{next}, \overline{\#_2} \xrightarrow{\leq} q_\top, q_\top$	$(b, h_i(b)), \#_3 \xrightarrow{\leq} q_\top, q_\top$

Our final protocol $\mathcal{P}$ is simply the product of those three protocols, the opinion of a triple of states being the conjunction of their opinions. This way, we can reach a $\top$-stable configuration from an initial configuration if and only if we can reach a $\top$-stable configuration from it in all three protocols.

We now show that this is the case if and only if that initial configuration is of the form $\#_1 u \#_2 v \#_3$ with $u = h_1(v) = h_2(v)$.

If the initial configuration $\#_1 h(a_1) \cdots h(a_k) \#_2 a_1 \cdots a_k \#_3$ satisfies those conditions, then it is straightforward to build an execution from it to $q_\top^*$ in both $\mathcal{P}_1$ and $\mathcal{P}_2$. Furthermore, it can reach a $\top$-stable configuration in $\mathcal{P}_0$ since it is in $\#_1 A^* \#_2 B^* \#_3$.

Now consider an initial configuration from which we can reach a $\top$-consensus in all three protocols. By definition of $\mathcal{P}_0$, it must be of the form $\#_1 a_1 \cdots a_k \#_2 b_1 \cdots b_\ell \#_3$ with $a_1, \ldots, a_k \in A$ and $b_1, \ldots, b_\ell \in B$.

Now consider protocol $\mathcal{P}_i$, with $i \in \{1, 2\}$. We call the states $\{a^{head}, a^{next} \mid a \in A\} \cup \{\#_1\}$ *special states*. It is easy to see that while we do not use the last ***Next (A)*** rule, there is exactly one special state in the configuration, and there is no special state afterwards. Also observe that only letters to the right of the special state can be modified. Since all agents must reach $q_\top$, this imposes that all agents that have a state in A initially must hold the special state, one by one from left to right.

A symmetric argument shows that all agents with a state in B initially must transform into $q_\top$ using ***Next (B)*** rules, one by one, from left to right. Now let us have a look at the ***Head*** rule: $a^{head}, (b, \pi) \xrightarrow{\leq} a^{next}, (b, \pi a)$. Let w be the word obtained by taking the letter a associated with each of those transitions, in the order in which they appear in the run.

By the arguments above, we must have $w = a_1 \cdots a_k$, and $w = h_i(b_1) \cdots h_i(b_\ell)$. As a result, the input word is indeed of the form $\#_1 h_i(b_1) \cdots h_i(b_\ell) \#_2 b_1 \cdots b_\ell \#_3$.

As we have shown this for both $i \in \{1, 2\}$, we obtain that the initial configuration must be of the form $\#_1 u \#_2 v \#_3$ with $u = h_1(v) = h_2(v)$.

In conclusion, there is a reachable $\top$-stable configuration if and only if the PCP instance has a solution. ◀

A.11 Missing proofs from Section 7.2

▶ **Theorem 34.** *If Conjecture 33 holds, then the syntax of* IO-PP[$<$] *deciders is decidable.*

Proof. Let $\mathsf{Post}(u) = \{v \mid u \to v\}$ and $\mathsf{Post}^*(u) = \{v \mid u \to^* v\}$. Similarly, let $\mathsf{Pre}(u) = \{v \mid v \to u\}$ and $\mathsf{Pre}^*(u) = \{v \mid v \to^* u\}$. We extend these to sets, e.g., $\mathsf{Post}^*(L) = \bigcup_{w \in L} \mathsf{Post}^*(w)$.

We reuse the machinery of Section 4.1, specifically Lemma 11, Lemma 13, and the combinatorial Claim 36.

If a IO-PP[$<$] protocol $\mathcal{P} = (Q, \Sigma, O, \Delta)$ is not a decider, then it is witnessed by the configuration space of words of length n for some $n \geq 2$. This can be checked for each n incrementally. We only have to provide a semi-decision procedure which returns yes if and only if the protocol is a decider.

Observe that assuming Conjecture 33 holds, a protocol $\mathcal{P}$ is a decider if and only if there exist languages $K_\top, K_\bot$ in DA over Q^* such that:

- $K_\top$ and $K_\bot$ are disjoint;
- $\Sigma^* \subseteq K_\top \cup K_\bot$;
- For both b, for all $u \in K_b$ and $v \in \mathsf{Post}(u)$, it is the case that $v \in K_b$;
- For both b, $K_b \subseteq \mathsf{Pre}^*(C_b)$ with C_b the set of b-stable configurations.

The right-to-left direction is clear: if such invariants exist, then the protocol recognizes $K_\top \cap \Sigma^*$. For the other direction, by Theorem 10, if $\mathcal{P}$ is a decider then its language L is in DA, hence so is $\Sigma^* \setminus L$. Take $K_\top = \mathsf{Post}^*(L)$ and $K_\bot = \mathsf{Post}^*(\Sigma^* \setminus L)$. Further, assuming Conjecture 33, $K_\top$ and $K_\bot$ are also in DA. They clearly satisfy all the conditions.

Given $\mathcal{P}$, we enumerate automata $\mathcal{A}, \mathcal{B}_\top, \mathcal{B}_\bot$ recognizing languages in DA.

We check whether:

- $L(\mathcal{A}) \subseteq L(\mathcal{B}_\top)$, $\Sigma^* \setminus L(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B}_\bot)$;
- For both b, for all $u \in L(\mathcal{B}_b)$ and $v \in \mathsf{Post}(u)$, it is the case that $v \in L(\mathcal{B}_b)$;
- For both b, $L(\mathcal{B}_b) \subseteq \mathsf{Pre}^*(C_b)$.

The first condition is easy to check. For the second one, we can build a non-deterministic automaton recognizing $\bigcup_{u \in L(\mathcal{B}_b)} \mathsf{Post}(u)$, by making it guess two positions at which it applies a transition of the protocol, while checking that the resulting word is in $L(\mathcal{B}_b)$. The last condition is more subtle. To check it, we argue that if there is a word in $L(\mathcal{B}_b) \setminus \mathsf{Pre}^*(C_b)$ then there is one of bounded length. It then suffices to check inclusion over words of length below this bound.

First we prove a property of language of DA by a classical use of Ramsey's theorem.

▷ Claim 42. For all language $L \subseteq Q^*$ in DA, there exists $m \in \mathbb{N}$, computable from an automaton recognizing L, such that for all $n \geq m$ and $u_1, \ldots, u_n, z$ with $\emptyset \neq \alpha(u_1) = \cdots = \alpha(u_n) \supseteq \alpha(z)$, we have $u_1 \cdots u_n \equiv_L u_1 \cdots u_n z u_1 \cdots u_n$.

Proof. Let $(M, \cdot, 1_M)$ be the syntactic monoid of L, $\varphi \colon Q^* \to M$ a morphism and $F \subseteq M$ such that $L = \varphi^{-1}(F)$.

Given a sequence of words $u_1, \ldots, u_n$, consider the complete undirected graph (without loops) over $\{0, \ldots, n\}$, where for all $i < j$ the edge $\{i, j\}$ is colored $\varphi(u_{i+1} \cdots u_j)$. By Ramsey's theorem [27], there exists a uniform bound $m \in \mathbb{N}$ such that whenever $n \geq m$ this graph contains a monochromatic 3-clique, i.e., there are $i < j < k$ such that $\varphi(u_{i+1} \cdots u_j) = \varphi(u_{j+1} \cdots u_k) = \varphi(u_{i+1} \cdots u_k)$.

Let $u_1, \ldots, u_n, z$ with $n \geq m$ and $\emptyset \neq \alpha(u_1) = \cdots = \alpha(u_n) \supseteq \alpha(z)$, and let $i < j < k$ be as described above. Since φ is a morphism, $\varphi(u_{i+1} \cdots u_k) = \varphi(u_{i+1} \cdots u_j)\varphi(u_{j+1} \cdots u_k) = \varphi(u_{i+1} \cdots u_k)^2$. In other words, $u_{i+1} \cdots u_k \equiv_L (u_{i+1} \cdots u_k)^2$. By Equation (1), we have $u_{i+1} \cdots u_k \equiv_L u_{i+1} \cdots u_k (u_{k+1} \cdots u_n z u_1 \cdots u_i) u_{i+1} \cdots u_k$.

By appending a prefix $u_1 \cdots u_i$ and a suffix $u_{k+1} \cdots u_n$ to both sides of the equivalence, we obtain $u_1 \cdots u_n \equiv_L (u_1 \cdots u_n) z (u_1 \cdots u_n)$. ◁

We are now ready to show the bound on witnesses for (the negation of) the third condition.

▷ Claim 43. We can compute a bound k such that if $L(\mathcal{B}_b) \setminus \mathsf{Pre}^*(C_b)$ is non-empty then it contains a word of length at most k.

Proof. Since $L(\mathcal{B}_b)$ belongs to DA, we can compute m from $\mathcal{B}_b$ as given by Claim 42.

By Lemma 11, the set of b-stable configurations is subword-closed and computable. There is therefore a computable k such that it is made of a finite union of languages described by expressions of the form $A_1^* B_1^\epsilon \cdots A_\ell^* B_\ell^\epsilon$ with $\ell \leq k$. Let $n = \max(m, k)$.

Let w be a word in $L(\mathcal{B}_b) \setminus \mathsf{Pre}^*(C_b)$ of length $> B(|\Sigma|, n)$ as given by Claim 36. The word w is of the form $x u_1 \cdots u_n z u_1 \cdots u_n y$ with $\emptyset \neq \alpha(u_1) = \cdots = \alpha(u_n) \supseteq \alpha(z)$.

The word $w' = x u_1 \cdots u_n y$ belongs to $L(\mathcal{B}_b)$ since $n \geq m$. It is not in $\mathsf{Pre}^*(C_b)$ since otherwise w would be in $\mathsf{Pre}^*(C_b)$, analogously to the proof of Lemma 13. ◁

In conclusion, to check the third condition we can compute the bound k given by the claim above, and check if there is a word of length at most k in $L(\mathcal{B}_b) \setminus \mathsf{Pre}^*(C_b)$. ◀

B Supplementary constructions and explorations

B.1 Alternative construction for $\Delta_2[<] \subseteq$ IO-PP$[<]$

▶ **Lemma 44.** *Any language in $\Sigma_2[<]$ is* IO-PP$[<]$*-semi-decidable.*

Proof. We know that any language in $\Sigma_2[<]$ is a finite union languages of the form $L_0 a_1 L_1 \dots a_n L_n$ where L_i is a downward-closed language for all $1 \leq i \leq n$. Furthermore, any downward closed language can be written as a finite union of languages of the form $A_0^* a_1 A_1^* \dots a_n A_n^*$ where $A_i \subseteq \Sigma$. Therefore, since semi-decidable languages are closed under finite union, it is sufficient to build a protocol recognizing L of the form $A_0^* a_1 A_2^* \dots a_{n-1} A_n^*$.

CM: ref?

CM: harmoniser avec les expressions de la preuve du claim 8

CM: lemma for this?

The protocol is designed so that an input word u can reach a $\top$-stable configuration if and only if $u \in L$. It proceeds in three steps: (1) Each agent guesses an index in $\{0, \dots, n\}$ corresponding to a position in the pattern. (2) Once an agent has observed at least one agent for each odd position (in an order that is consistent with its own view), it can "lock" itself in a $\top$-state. (3) Any agent that is not locked can always reinitialize (i.e., restart from its input symbol and try a different guess). Even locked agents may reinitialize if they encounter an agent whose guess is incompatible with theirs.

Let us give explicitly the protocol rules.

(1) ***Guessing rules.*** Each agent can make a guess, without observing any neighbor.

$$a \to a(i)(-1) \text{ if } \begin{cases} a \in A_i \text{ and } i \text{ is even,} \\ a = a_i \text{ and } i \text{ is odd.} \end{cases}$$

In state $a(i)(\ell)$, i is the guessed position of the letter in the language structure, whereas $\ell \in \{-1, 1, 3, \dots\}$ is its level in the lock process. A letter is fully locked when $\ell = n - 1$, i.e., the opinion function maps each symbol in $\{\sigma(i)(n-1) | \sigma \in \Sigma, i \leq n\}$ to $\top$, and every other symbol to $\bot$.

(2) ***Locking rules.*** When one agent with a guess i is at lock level ℓ, it can increase its lock level to $\ell + 2$ when either its guess is $\ell + 2$, (and it is the agent $a_{\ell+2}$ of the pattern), or it encounters one agent with guess ℓ to its right (resp. left) and this is coherent with its view, i.e. $i < \ell + 2$ (resp $i > \ell + 2$):

$$(a(i)(\ell), b(\ell+2)(\ell')) \xrightarrow{<} (a(i)(\ell+2), b(\ell+2)(\ell')) \qquad \text{with } i < \ell + 2$$
$$(b(\ell+2)(\ell'), a(i)(\ell)) \xrightarrow{<} (b(\ell+2)(\ell'), a(i)(\ell+2)) \qquad \text{with } i > \ell + 2$$
$$a(\ell+2)(\ell) \to a(\ell+2)(\ell+2)$$

(3) ***Unlocking rules.***

$$(a(i)(\ell),\ b(j)(\ell')) \xrightarrow{<} (a,\ b(j)(\ell')) \text{ if } \begin{cases} i > j \\ i = j \text{ and } i \text{ is odd and greater than } -1. \end{cases}$$
$$a(i)(\ell) \to a \text{ if } \ell < n - 1$$
$$(a,\ b(i)(\ell)) \to (a,\ b)$$
$$(b(i)(\ell),\ a) \to (b,\ a)$$

CM: notation for self-transitions

We claim that a word $u = u_1 \dots u_k$ can reach a word $u_\top = u_1(x_1)(n-1) \dots u_k(x_k)(n_1)$ in $\Sigma_\top$ only if for all odd i between 1 and $n-1$, there exists $j \leq k$ such that $x_j = i$. Indeed, consider the last letter to go from state $a(x)(i-2)$ to $a(x)(i)$ in the execution from u to

$u_\top$: to do that, $a(x)(i-2)$ must observe a letter $b(i)(\ell)$ (maybe itself). For the sake of contradiction, imagine that b wears a different label in $u_\top$: it means that after being observed by a, b uses an unlocking rule to reinitialize itself, and then later guesses a new position and locked itself; hence a is not the last letter to do the i-th locking step, and we reach a contradiction.

Thanks to the first unlocking rule, $u_\top$ is a $\top$-consensus iff $x_1 \leq \cdots \leq x_k$, and for each odd i between 1 and $n-1$, there exist at most one j such that $x_j = i$. Thus, if u is not in L, u cannot reach a $\top$-consensus, because there is no labelling of u through the guessing rule producing an unlockable sequence of x_i.

On the other hand, if u is in L, there exists such a labelling, hence there is a run producing a $\top$-consensus from u. Now it only remains to show that any *fair* run reaches a $\top$-consensus. Observe that, from any configuration u' which is not a $\top$-consensus, there is an execution from u' to u using the unlocking rules: any partially locked letter can reinitialize itself, and a fully locked letter can reinitialize by observing a position contradicting with its own from a letter at its right. Furthermore, any un-labelled letter a can reinitialize any other letter, even fully locked. Thus, any infinite fair run from u would enter and leave u infinitely often, and explore every configuration accessible from u, including a $\top$-consensus. Thus every fair run from u is finite and reaches a $\top$-consensus.

Therefore, the protocol semi-decides L. ◀

B.2 Alternative construction for $\Delta_1^{\text{int}}[<,+,\equiv] \subseteq \mathsf{PP}[<]$

▶ **Proposition 45.** $\Delta_1^{\text{int}}[<,+,\equiv] \subseteq \mathsf{PP}[<]$.

Proof. We show that any language in $\Sigma_1^{\text{int}}[<,+,\equiv]$ is semi-decided by a protocol in $\mathsf{PP}[<]$. Such a language is a finite union of intersections $K = R \cap S$, where R is a language of the form $A_0^* a_1 A_2^* \dots a_{n-1} A_n^*$ and S is a semilinear constraint protocol. It suffices to semi-decide one such K.

IV: it is more than an intersection, otherwise we know how to intersect protocols

We construct a protocol $\mathcal{P}$ with two components: $\mathcal{P}_{struct}$ from Lemma 44 that semi-decides R and $\mathcal{P}_{count}$.

A fair run of $\mathcal{P}_{struct}$ on input w annotates every agent with a position label (k) and a locked/unlocked label. When an agent is labelled with position k, it means this agent believe they are part of the k-th block of R. Furthermore, for any $w \to^* w'$:

- If at least one letter is unlocked in w', there exists a run $w' \to^* w$.
- If w' is fully locked, but an agent observes that their position belief is in contradiction with an other agent belief, they can unlock themself.
- If w' is fully locked with coherent position beliefs (which is possible iff $w \in R$) then no more transition rules can be taken; $\mathcal{P}_{struct}$ stabilizes.

This stabilizing property of $\mathcal{P}_{struct}$ is what will allow us to combine it with $\mathcal{P}_{count}$. Formally, following Angluin et al. [1], an input process is *stabilizing* if it remains invariant after some time T_{stab}. By theorem 4 in [1], there exists a protocol $\mathcal{P}_{count}$ that, given stabilizing inputs, eventually stabilizes to output $\top$ if the inputs satisfy S, and $\bot$ otherwise.

IV: there is a leap from the way Theorem 4 is given and that

Protocol $\mathcal{P}_{count}$ runs only on agents locked by $\mathcal{P}_{struct}$ with a specific feedback loop: if an agent has opinion $\bot$ in $\mathcal{P}_{count}$, they can unlock themself. We show that a stable $\top$-consensus is reachable $\iff w \in K$.

- Case $w \in K$: There exists a correct structural guess. Consider a fair path the agents guess this structure and lock. Since $w \in S$, $\mathcal{P}_{count}$ eventually stabilizes to output $\top$. Once

output $\top$ is reached, no unlocking can occur, and the system remains in a permanently locked structural state with $\top$ output, i.e., a $\top$-consensus.

- Case $w \notin K$: If $w \notin R$, $\mathcal{P}_{struct}$ never locks stably by definition. If $w \in R$ but $w \notin S$, suppose $\mathcal{P}_{struct}$ locks in position w'. Since w' represent a vector violating S. Thus $\mathcal{P}_{count}$ can always unlock an agent. ◀

IV: I don't think this proof can work without giving the details of Pcount. If Pcount works for some time then unlock a letter, we cannot force this letter to completely reset w, there could be some leftovers from the Pcount execution

IV: Here is my attempt at a proof. To make it work, I had to add some precision on how Pstruct and Pcount compose together, and how to carefully clean up when doing feedback

MB: How do you get rid of coefficients from linear combinations? And from modulos? This seems crucial; agents cannot take the role of a position outside of the range $[1..n]$.

Proof. Any language in $\Sigma_1^{\text{int}}[<,+,\equiv]$ is a finite union of languages L where L is defined by

- a finite set of special positions $x_1, \dots, x_n$.
- Semi-linear conditions on the infixes delimited by these special positions.

Formally, L is characterized a number n and a finite set $\mathcal{S}$ of constraints of the form $\sum_{\sigma\in A} \#_\sigma(x_i, x_j) \bowtie \sum_{\sigma\in B} \#_\sigma(x_k, x_\ell)$ with $A, B \subseteq \Sigma$ and $\bowtie \in \{=, <, \leq, >, \geq\}$ and $0 \leq i \leq j \leq n+1$ and $0 \leq k \leq \ell \leq n+1$. For any $u \in \Sigma^*$, $u \in L$ iff there exist positions $x_1, \dots x_n$ in u such that u verifies all constraints in S (positions x_0 and x_{n+1} denotes the first and last positions of the word).

Let us build $\mathcal{P}$ that semi-decides such a L. We do that by composing protocols $\mathcal{P}_{struct}$ and $\mathcal{P}(S)$ for all $S \in \mathcal{S}$.

First, $\mathcal{P}_{struct}$ factorizes u as $u_0a_1u_2 \dots a_{n-1}u_n$ by annotating every agent with a position belief: either (x_k), if the agent believe themself to be special position x_k, or $(k+)$ if they believe to be after x_k (but before x_{k+1} if exists), or (0) for agent before x_0. Then every agent "locks" themself by checking they can see every special position. Any agent can unlock themself if they observe a contradiction between their beliefs and someone else, or someone not fully locked. A $\top$-stable configuration is a configuration where every agent is fully locked with a belief coherent with some factorisation $u_0a_1u_2 \dots a_{n-1}u_n$ of u.

For more details on how this protocol works, see Proof of Lemma 44.

Now, let us give a protocol $\mathcal{P}_S$ which verifies a constraint S of the form $\sum_{\sigma\in A} \#_\sigma(x_i, x_j) \bowtie \sum_{\sigma\in B} \#_\sigma(x_k, x_\ell)$ with $A, B \subseteq \Sigma$ and $\bowtie \in \{=, <, \leq, >, \geq\}$ and $0 \leq i \leq j \leq n+1$ and $0 \leq k \leq \ell \leq n+1$. Assume that $\bowtie = <$ (other cases works similarly):

Here are the transition rules of $\mathcal{P}_S$, for any agents a, b, c, d fully locked such that agent a is fully locked in A with position belief between i and j, agent b is fully locked in B with position belief between k and l, and agent c is in one of these configurations. Agent d is fully locked with irrelevant position belief.

- $a, b \to a^2, b^2$ for $a \in A$, $b \in B$
- $b \to b^{1-\top}$
- $b^{1-\top}, a \to b^2, a^2$
- $b^{1-\top}, c_2 \to b^1, c^{2-\top}$
- $d \to d^\top$

$\mathcal{P}$ is a composition of protocol $\mathcal{P}_{struct}$, and protocols $\mathcal{P}(S)$ for all $S \in \mathcal{S}$. The labels from protocol $\mathcal{P}(S)$ carry a S-identifier, and can co-exist together. A *end-configuration* in $\mathcal{P}$ is a word u' where each agent is locked and has a position belief, and has a $\top$ label from $\mathcal{P}(S)$ for all $S \in \mathcal{S}$ Furthermore, all position beliefs must be coherent. $\Sigma_\top$ is the minimal alphabet such that every end-configuration is a $\top$-consensus. We will show later that end-configurations are exactly $\top$-stable configurations.

We add the following feedback rules, between the composed protocols:

- **Free-to-unlock rule** Letters that are locked but have no label can always unlock themselves. Furthermore, letters that are locked cannot be unlocked unless they bear no label from any $\mathcal{P}_S$.

- **Prepare-to-change-belief rule.** If $\mathcal{P}_{struct}$ can unlock an agent, but this agent is labelled, it can receive an "un-label yourself" order.
- **Unlabel rules.** An agent with such an an "unlabel yourself" order can reverse any transition from any protocol S involving itself and another agent (even if this agent has received no order). It then loses the "unlabel yourself".
- **Unhappy rule.** If an agent has some labels but not all needed, it can give itself the "unlabel yourself" order.

These feedback rules ensure the following properties:

- If $u \in L$, u can reach an end-configuration. Furthermore, end-configuration are $\top$-stable: no feedback or unlock rules can be applied.
- For any run $u \to^* u'$ such that u' is not an end-configuration, there exists a run $u' \to^* u$.
  - Either some agent is not fully labelled: they can use the **Unhappy rule**, then unlabel themself through the Unlabel rules, then unlock themself with the **Free-to-unlock rule**. From there, they can use the **Prepare-to-change-belief rule** on every labelled agent. They all unlabel themself then unlock themself.
  - Or there is a contradiction between beliefs. The agent observing this contradiction can use the **Prepare-to-change-belief rule** on themself, then apply the same process as above.
- The unlabel rules, combined to the rules of any protocol $\mathcal{P}_S$, ensure that, at any point of an execution, all agents labelled by $\top$ form a word that would verify S if their position belief were valid. Therefore, if there is an execution $u \to u'$ where u' is an end-configuration, then $u \in L$. Indeed, position beliefs are coherent in u' coherent, and for any $S \in \mathcal{S}$ every agent is labelled $\top$ by $\mathcal{P}_S$. ◀

B.3 Alternative proof of $\Delta_2[<] \subseteq \Delta_1^{\mathrm{int}}[<, +, \equiv]$

▶ **Proposition 46.** $\Delta_2[<] \subseteq (\Delta_1^{\mathrm{int}}[<, +, \equiv] \cap \mathrm{Reg})$.

Proof. Let $L \in \Sigma_2[<]$. As mentioned in Section 4.2, L is a finite union of languages of the form $L' = A_0^* a_1 A_1^* \cdots a_k A_k^*$ where $A_i \subseteq \Sigma$. Let $\chi_{i,j}$ be 1 if $a_i \notin A_j$, and 0 otherwise. The language L' can be expressed by

$$\exists x_1, \dots x_k \ (x_1 < \cdots < x_k) \wedge a_1(x_1) \wedge \cdots \wedge a_k(x_k) \wedge$$
$$\sum_{\sigma \in \Sigma \setminus A_0} \#_\sigma(1, x_1) = \chi_{1,0} \wedge \sum_{\sigma \in \Sigma \setminus A_k} \#_\sigma(x_k, \mathbf{max}) = \chi_{k,k} \wedge$$
$$\bigwedge_{1 \le i < k} \left(\sum_{\sigma \in \Sigma \setminus A_i} \#_\sigma(x_i, x_{i+1}) = \chi_{i,i} + \chi_{i+1,i} \right).$$

Thus, $L' \in \Sigma_1^{\mathrm{int}}[<, +, \equiv]$ and so $L \in \Sigma_1^{\mathrm{int}}[<, +, \equiv]$. Since $L \in \Delta_2[<]$, we have $\overline{L} \in \Sigma_2[<]$ and hence $\overline{L} \in \Sigma_1^{\mathrm{int}}[<, +, \equiv]$ by the same reasoning. Therefore, $L \in \Delta_1^{\mathrm{int}}[<, +, \equiv]$. ◀

B.4 Partially-ordered two-way Parikh automata

A *partially-ordered two-way Parikh automaton* (*p.o. 2-PA*) is a tuple $\mathcal{A} = (Q, \Sigma, \delta, q_0, F, \Psi_{\mathrm{acc}})$ where:

- Q is a finite set of *states* equipped with a partial order $\leq$;
- Σ is a finite *alphabet*;

- $\delta \subseteq Q \times (\Sigma_{\vdash\dashv} \cup \{\varepsilon\}) \times \{\triangleright, \triangleleft\} \times Q$ is the *transition relation*, that satisfies the following, for every $(p, \sigma, d, q), (p', \sigma', d', q') \in \delta$:
  - $p \leq q$ (cycles are self-loops);
  - if $p = q = p' = q'$, then $d = d'$ (a self-loop reads in a single direction);
  - if $p = p'$ and $\sigma, \sigma' \in \Sigma_{\vdash\dashv}$, then $\sigma \neq \sigma$ (nondeterminism arises from ε);
  - if $p = q$, then $\sigma \neq \varepsilon$ (no ε-self-loops);
  - $\sigma = \vdash$ implies $\sigma \neq \triangleleft$, and $\sigma = \dashv$ implies $\sigma \neq \triangleright$ (the head never goes out of bound);
- $q_0 \in Q$ is the *initial state*;
- $F \subseteq Q$ is the set of *final states*, from which there is no transition;
- Ψ_{acc} is a Presburger formula over variables δ, called the *acceptance condition*.

A *configuration* of $\mathcal{A}$ is a triple $(q, i, \boldsymbol{c})$ where $q \in Q$ is the current state; $i \in \mathbb{N}$ is the head position; and $\boldsymbol{c} \colon \delta \to \mathbb{N}$ maps transitions to their visit count. For every $w \in \Sigma^n$, let $w[0] = \vdash$ and $w[n+1] = \dashv$. On input w, a step $(q, i, \boldsymbol{c}) \to^t (q', i', \boldsymbol{c}')$ can occur if transition $t = (q, \sigma, \tau, q') \in \delta$ satisfies the following:

- If $\sigma \in \Sigma_{\vdash\dashv}$, then $w[i] = \sigma$, and $i' = i + 1$ if $\tau = \triangleright$, or $i' = i - 1$ if $\tau = \triangleleft$;
- If $\sigma = \varepsilon$, then $i' = i$;
- $\boldsymbol{c}'(t) = \boldsymbol{c}(t) + 1$ and $\boldsymbol{c}'(s) = \boldsymbol{c}(s)$ for $s \neq t$.

We say that $\mathcal{A}$ *accepts* the input w if $(q_0, 1, \mathbf{0}) \to^* (q, i, \boldsymbol{c})$ where $q \in F$ and $\Psi_{\text{acc}}(\boldsymbol{c})$ holds. The *language* of $\mathcal{A}$ is the set of words it accepts. Figure 4 depicts two examples of p.o. 2-PA.

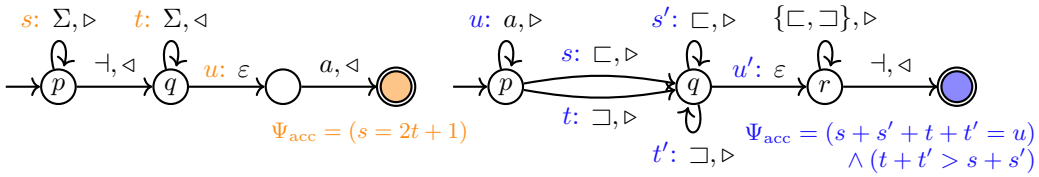


Figure 4 Example of p.o. 2-PA for the median language $\bigcup_{n \geq 0} \Sigma^n a \Sigma^n$ (left) and the coDyck-witness language $\{a^n v \mid n \geq 1, v \in \{\sqsubset, \sqsupset\}^{\geq n}, |v[1..n]|_{\sqsupset} > |v[1..n]|_{\sqsubset}\}$ (right).

For all $t = (p, \sigma, d, q)$, let $\text{out}(t) = p$, $\text{letter}(t) = \sigma$, $\text{dir}(t) = d$ and $\text{in}(t) = q$. We distinguish *looping transitions* $\delta_{\circlearrowleft} = \{t \in \delta \mid \text{out}(t) = \text{in}(t)\}$ and *progress transitions* $\delta_{\rightarrow} = \delta \setminus \delta_{\circlearrowleft}$.

▶ **Proposition 47.** *A language is recognized by some p.o. 2-PA iff it belongs to* $\Sigma_1^{\text{int}}[<, +, \equiv]$.

A resolver is a mechanism that indicates what to do when a nondeterministic choice arises. We introduce a type of resolver that takes a progress transition as soon as some linear constraint holds, and only uses looping transitions otherwise.

Let τ be a function that associates to each progress transition a Presburger formula over Q, such that $(t \neq t' \wedge \text{out}(t) = \text{out}(t') \wedge \varepsilon \in \{\text{letter}(t), \text{letter}(t')\})$ implies $\tau(t) \wedge \tau(t') \equiv \mathit{false}$.

We will interpret formulas over Q as formulas over δ by mapping each q to $\sum_{t \in \delta \mid \text{in}(t) = q} \boldsymbol{c}(t)$. Given $t \in \delta$, we write $(q, i, \boldsymbol{c}) \to_\tau^t (q', i', \boldsymbol{c}')$ iff $(q, i, \boldsymbol{c}) \to^t (q', i', \boldsymbol{c}')$ and either

- $t \in \delta_{\rightarrow}$ and $\tau(t)$ holds; or
- $t \in \delta_{\circlearrowleft}$ and $\neg\tau(s)$ holds for each $s \in \delta_{\rightarrow}$ such that $\text{out}(s) = q$.

We write $L_\tau(\mathcal{A})$ to denote the language recognized by $\mathcal{A}$ using $\to_\tau^*$ instead of $\to^*$. We say that τ is a *semilinear resolver* if $L_\tau(\mathcal{A}) = L(\mathcal{A})$.

▶ **Example 48.** Reconsider automaton $\mathcal{A}_{\text{left}}$ from Figure 4. By assigning $\tau(u) = (p - 2q = 1)$, the automaton accepts the same language, even with $\Phi_{\text{acc}} = \top$. For automaton $\mathcal{A}_{\text{right}}$ from Figure 4, setting $\tau(u') = (p - q = 0)$ yields the same language, even with $\Phi_{\text{acc}} = (t + t' > s + s')$. The advantage here is that it makes both automata "deterministic".

▶ **Proposition 49.** *Given a p.o. 2-PA $\mathcal{A}$ and a semilinear resolver τ, it is possible to construct a p.o. 2-PA $\mathcal{A}'$ and a semilinear resolver τ' such that $L_{\tau'}(\mathcal{A}') = \overline{L_\tau(\mathcal{A})}$.*

Proof. Let $\mathcal{A} = (Q, \Sigma, \delta, q_0, F, \Psi_{\text{acc}})$. Since $(t \neq t' \wedge \text{out}(t) = \text{out}(t') \wedge \varepsilon \in \{\text{letter}(t), \text{letter}(t')\})$ implies $\tau(t) \wedge \tau(t') \equiv \mathit{false}$, two progress transitions of $\mathcal{A}$ are never simultaneously enabled. As $\to_\tau^*$ prioritize progress transitions, this means that each input word has a single run in $\mathcal{A}$. Since $\mathcal{A}$ is partially ordered, it cannot loop forever. Therefore, a word is rejected by $\mathcal{A}$ iff its single run reaches F with $\neg\Psi_{\text{acc}}$, or reaches a dead-end in $Q \setminus F$. We construct $\mathcal{A}'$ and τ' so that they detect these two outcomes.

For every $p \in Q$, let

$$A_p = \{a \in \Sigma_{\vdash\dashv} \mid a = \text{letter}(t) \text{ for some } t \in \delta \text{ with } \text{out}(t) = p\},$$
$$S_p = \{s \in \delta_\to \mid \text{out}(s) = p \wedge \text{letter}(s) = \varepsilon\}.$$

We construct $\mathcal{A}' = (Q', \Sigma, \delta', q_0', F', \Psi'_{\text{acc}})$ and τ' as follows:

- $Q' = Q \cup \{q_{\text{acc}}\}$, $q_0' = q_0$ and $F' = \{q_{\text{acc}}\}$;
- δ' and τ' obtained by extending δ and τ as follows:
  - For each $t = (p, \sigma, d, q) \in \delta_\to$ with $\sigma \neq \varepsilon$, we add $t' = (p, \sigma, d, q_{\text{acc}})$ with $\tau'(t') = \neg\tau(t) \wedge \bigwedge_{s \in S_p} \neg\tau(s)$. Indeed, as t is the only transition that can read σ from p, automaton $\mathcal{A}$ rejects unless an ε-transition is enabled by τ (in which case it is taken).
  - For each $a \in \Sigma_{\vdash\dashv} \setminus A_p$, we add $t_{p,a} = (p, a, d, q_{\text{acc}})$ with $\tau'(t_{p,a}) = \bigwedge_{s \in S_p} \neg\tau(s)$, and $d = \triangleright$ iff $a \neq \dashv$. Indeed, since no transition can read a from p, automaton $\mathcal{A}$ rejects unless an ε-transition is enabled by τ (in which case it is taken);
- $\Psi'_{\text{acc}} = \bigwedge_{t \in \delta' \setminus \delta} (t > 0) \vee \neg\Psi_{\text{acc}}$.

◀

▶ **Proposition 50.** *For every p.o. 2-PA $\mathcal{A}$ and linear resolver τ, $L_\tau(\mathcal{A}) \in \Delta_1^{\text{int}}[<, +, \equiv]$.*

Proof. By Proposition 49, the class of languages recognized by these automata is closed under complement. Thus, it suffices to give a formula in Σ_1^{int}.

Since $\mathcal{A}$ is partially ordered, every accepting run is of the form

$$(q_0, \cdot, \cdot) \to_\tau^{\rho_0} (q_0, \cdot, \cdot) \to_\tau^{t_1} (q_1, \cdot, \cdot) \to_\tau^{\rho_1} (q_1, \cdot, \cdot) \to_\tau^{t_2} \cdots \to_\tau^{\rho_{m-1}} (q_{m-1}, \cdot, \cdot) \to_\tau^{t_m} (q_m, \cdot, \cdot),$$

where each ρ_i only uses looping transitions, each t_i is a progress transition, and $q_m \in F$.

We will construct a formula that guesses $t_1, \ldots, t_m$ and checks that the run is valid. We introduce existential variables $\boldsymbol{x} = (x_1, \ldots, x_m)$ and $\boldsymbol{y} = (y_0, y_1, \ldots, y_m)$ where x_i and y_i are the head position respectively before and after taking t_i. We set $y_0 = 1$.

The validity of ρ_i requires that the interval between x_i and x_{i+1} contains no letter outside of $A_i = \{\text{letter}(s) \mid s \in \delta_\circlearrowleft, \text{out}(s) = \text{out}(t_i)\}$. It is expressed as

$$\varphi_{\text{loop},i} = \bigwedge_{a \in \Sigma_{\vdash\dashv} \setminus A_i} (\#_a(u, v) = 0) \text{ where } u = \min(y_i, x_{i+1}) \text{ and } v = \max(y_i, x_{i+1}).$$

The validity of the positions is expressed as follows:

$$\varphi_{y,i} = \begin{cases} y_i = x_i & \text{if } \mathrm{letter}(t_i) = \varepsilon, \\ y_i = x_i + 1 & \text{if } \mathrm{letter}(t_i) \neq \varepsilon \text{ and } \mathrm{dir}(t_i) = \triangleright, \\ y_i = x_i - 1 & \text{if } \mathrm{letter}(t_i) \neq \varepsilon \text{ and } \mathrm{dir}(t_i) = \triangleleft. \end{cases}$$

It remains to express that transition t_i is valid. Given a Presburger formula φ over Q, and a state q, let us define

$$\varphi_{\mathrm{disabled},i} = \forall x \ ((0 \leq x \leq \mathrm{out}(t_i)) \implies \neg\varphi(\mathrm{out}(t_i) \mapsto x)).$$

Since Presburger arithmetic admits quantifier elimination, $\varphi_{\mathrm{disabled},i}$ can be made quantifier-free. We define

$$\varphi_{\mathrm{progress},i} = \big((\mathrm{letter}(t_i) \neq \varepsilon) \to \#_{\mathrm{letter}(t)}(x_i, x_i) = 1\big) \wedge \tau(t_i) \wedge \bigwedge_{\substack{s \in \delta_\to \setminus \{t\} \\ \mathrm{out}(s) = \mathrm{out}(t) \\ \mathrm{letter}(s) = \varepsilon}} \tau(s)_{\mathrm{disabled},i}.$$

Recall that τ yields Presburger formulas over Q, which techically does not typecheck in $\varphi_{\mathrm{progress},i}$. Formally, we consider that q_i stands for $|y_i - x_{i+1}|$. ◀

▶ **Corollary 51.** *Any language decided by a p.o. 2-PA with linear resolving belongs to* PP[<].

B.5 Alternative proof for $\mathsf{NSPACE}(n) \subseteq \mathsf{IO\text{-}PP}[+1]$

Let us fix a linear-bounded nondeterministic automaton $M = (P, \Sigma, \Sigma', \delta, p_0, p_{\mathrm{acc}}, p_{\mathrm{rej}})$ where P is the set of states; Σ is the input alphabet; $\Sigma' \supseteq \Sigma$ is the tape alphabet; $\delta \subseteq (P \times \Sigma') \times (P \times \Sigma' \times \{\triangleleft, \triangleright\})$ is the transition relation; and $p_0, p_{\mathrm{acc}}, p_{\mathrm{rej}}$ are respectively the initial, accepting and rejecting states.

Traditionally, on input $w \in \Sigma$, a linear-bounded automaton has access to $\vdash w \dashv$; has no left-move on $\vdash$; has no right-move on $\dashv$; and cannot overwrite the endmarkers. For convenience, we consider instead that there are no endmarkers, but that the head stays put when moving left (resp. right) from the first (resp. last) cell. It is simple to construct such a machine with the same language. We further assume that M may only enter q_{acc} with its head on the rightmost cell, and that there is no transition from q_{acc}.

Let $\Gamma = (P \cup \{-\}) \times \Sigma' \times \{\mathrm{fst}, -, \mathrm{lst}\}$. For every $s = (p, a, x) \in \Gamma$, let $\mathsf{sta}(s) = p$ and $\mathsf{pos}(s) = x$. We say that a word $w \in \Gamma^+$, with $|w| \geq 2$, is a *valid configuration* of M if

- $\mathsf{pos}(w[i]) = \mathrm{fst}$ iff $i = 1$,
- $\mathsf{pos}(w[i]) = \mathrm{lst}$ iff $i = |w|$,
- there is a unique position i with $\mathsf{sta}(w[i]) \in P$.

For example, $w = (-, a, \mathrm{fst})(p, a, -)(-, b, \mathrm{lst})$ represents the configuration where M is in state p, with its head on the second cell, and its tape containing aab.

▶ **Proposition 52.** *The following language is* $\mathsf{PP}[+1]$*-semi-decidable with stabilizing inputs:* $L = \{w \in \Gamma^{\geq 2} \mid w \text{ is a valid configuration of } M \text{ and it leads to acceptance}\}$.

Proof. Let us construct a protocol $\mathcal{P} = (Q, \Gamma, O, \Delta)$. We define $Q = \Gamma \times \Gamma \times \Gamma \times \{\bot, \top, \text{👥}\}$ and $O((x, y, z, o)) = (x = y \wedge o = \top)$. We identify each input $\gamma \in \Gamma$ with $(\gamma, \gamma, \gamma, \bot)$.

Intuitively, the state (x, y, z, o) indicates that the agent has input x; has been acting as if its input was y; currents holds z; and has output belief o. Whenever $x \neq y$, the agent has changed its mind on its input, and must consequently reset the whole population.

The transitions of Δ are defined by these rules:

	Simulation of M	
(1)	$(*,*,(p,a,*),*),(*,*,(-,*,*),*) \xrightarrow{+1} (*,*,(-,b,*),*),(*,*,(q,*,*),*)$	for $((p,a),(q,b,\triangleright)) \in \delta$
(2)	$(*,*,(-,*,*),*),(*,*,(p,a,*),*) \xrightarrow{+1} (*,*,(q,*,*),*),(*,*,(-,b,*),*)$	for $((p,a),(q,b,\triangleleft)) \in \delta$
(3)	$(x,*,(p,a,*),*) \xrightarrow{\mathsf{true}} (x,*,(q,b,*),*)$	for $((p,a),(q,b,\triangleright)) \in \delta$ and $\mathsf{pos}(x) = \mathrm{lst}$
(4)	$(x,*,(p,a,*),*) \xrightarrow{\mathsf{true}} (x,*,(q,b,*),*)$	for $((p,a),(q,b,\triangleleft)) \in \delta$ and $\mathsf{pos}(x) = \mathrm{fst}$
	Belief propagation	
(5)	$(x,*,*,*),(x',*,z',*) \xrightarrow{\mathsf{true}} (x,*,*,\top),(x',*,z',*)$	if $\mathsf{pos}(x) = \mathrm{fst} \wedge \mathsf{pos}(x') = \mathrm{lst} \wedge \mathsf{sta}(z') = p_{\mathrm{acc}}$
(6)	$(*,*,*,\top),(*,*,*,*) \xrightarrow{\mathsf{true}} (*,*,*,\top),(*,*,*,\top)$	
	Reset on input changes	
(7)	$(x,y,*,*),(x',y',*,*) \xrightarrow{\mathsf{true}} (x,y,*,*),(x',y',*,\text{👥})$	if $\mathsf{pos}(x') = \mathrm{lst} \wedge (x \neq y \vee x' \neq y')$
(8)	$(*,*,*,*),(x',*,*,\text{👥}) \xrightarrow{+1} (*,*,*,\text{👥}),(x',x',x',\bot)$	
(9)	$(x,*,*,\text{👥}) \xrightarrow{\mathsf{true}} (x,x,x,\bot)$	if $\mathsf{pos}(x) = \mathrm{fst}$
	Erroneous configuration detection	
(10)	$(x,*,*,*),(x',*,*,*) \xrightarrow{\mathsf{true}} (x,*,*,\bot),(x',*,*,\bot)$	if $\mathsf{sta}(x), \mathsf{sta}(x') \in P$
(11)	$(x,*,*,*),(x',*,*,*) \xrightarrow{\mathsf{true}} (x,*,*,\bot),(x',*,*,\bot)$	if $\mathsf{pos}(x) = \mathsf{pos}(x') \in \{\mathrm{fst}, \mathrm{lst}\}$
(12)	$(x,*,*,*),(x',*,*,*) \xrightarrow{+1} (x,*,*,\bot),(x',*,*,\bot)$	if $\mathsf{pos}(x) = \mathrm{lst} \vee \mathsf{pos}(x') = \mathrm{fst}$

Let us show that $\mathcal{P}$ semi-decides L with stabilizing inputs. Let $u \in \Gamma^n$. For the special case of $n = 1$, note that each state starts with output $\bot$ and this remains so as no transition is ever enabled. Thus, let us assume that $n \geq 2$.

Let $u = u_0 \rightsquigarrow u_1 \rightsquigarrow \cdots \rightsquigarrow u_\ell = v$, and let $v = w_0 \to w_1 \to \cdots$ be a fair run. Without loss of generality, we may assume $\ell = 0$ or $\iota(u_\ell) \neq \iota(u_{\ell-1})$. Indeed, it suffices to take ℓ as the last moment where the input changes.

We make a case distinction, where each case assumes that the previous ones do not hold.

Case 1: $\iota(v)$ is valid. By validity, rules (10–12) are permanently disabled from v onwards.

If $\ell > 0$, then there exists a position i such that $v[i] = (x, y, z, o)$ with $x \neq y$. By validity of $\iota(v)$, we have $\mathsf{pos}(\iota(v)[1]) = \text{fst}$ and $\mathsf{pos}(\iota(v)[n]) = \text{lst}$. Thus, by fairness, the population must eventually use rule (7) to assign 👥⚙ to agent n; use rule (8) to reset the population from right to left; and use rule (9) to complete this reset with agent 1. From there, rules (7–9) are permanently disabled. Thus, by fairness, M is simulated faithfully with rules (1–4), and $\top$ eventually spreads with rules (5–6).

The case of $\ell = 0$ is the same except for the fact that rules (7–9) are disabled from the very beginning, where the simulation of M already begins.

Case 2: $\iota(v)$ has several "fst", "lst" or states from P; or a wrongly placed "fst" or "lst". Since the first component of each agent stops changing from v onwards, fairness and rules (10–12) guarantee that at least one $\bot$-belief must occur infinitely often.

Case 3: $\ell = 0$ and $\iota(v)$ is invalid. Rule (5) is never enabled and hence no $\top$ ever appears.

Case 4: $\ell > 0$ and $\iota(v)$ is invalid. By assumption, there is a position i such that $v[i] = (x, y, z, o)$ with $x \neq y$. By the latter, we have $O(v[i]) = \bot$. We consider three subcases.

Case 4-A: $\iota(v)$ has no "lst". By the lack of "lst", rule (9) is permanently disabled and hence agent i is stuck with output $\bot$.

Case 4-B: $\iota(v)$ has no "fst". The only way the population can possibly reach a $\top$-consensus is by resetting agent i. This requires creating at least one 👥⚙. Since $\iota(v)$ contains no "fst", rule (9) can never be used. Therefore, if 👥⚙ appears, then at least one 👥⚙ must remain forever. As $O((*, *, *, 👥⚙)) = \bot$, this means that there is always at least one agent with output $\bot$.

Case 4-C: $\iota(v)$ has no state from P. Since the previous (sub)cases do not hold, agents 1 and n respectively hold "fst" and "lst". As agent i has a mismatch on its first two components, the only way to obtain $\top$ is by carrying a full reset from agent n to agent 1 using rules (7–9). If this happens, then by the lack of state from P, rules (1–5) are permanently disabled. ◀

▶ **Proposition 53.** *The language $L(M)$ is $\mathsf{PP}[+1]$-semi-decidable.*

Proof. Let K denote the set of valid initial configurations of M. Formally, let $K = \Gamma_0\Gamma_1^*\Gamma_2$ where $\Gamma_0 = \{p_0\} \times \Sigma \times \{\text{fst}\}$, $\Gamma_1 = \{-\} \times \Sigma \times \{-\}$ and $\Gamma_2 = \{-\} \times \Sigma \times \{\text{lst}\}$.

Let $f(i) = \Gamma_i$ and $U_i = \{w \in \{0, 1, 2\}^+ : |w|_i = 1\}$. We have

$$K = f(0^*1^*2^* \cap U_0 \cap U_2).$$

Thus, by Propositions 7 and 16 and by Lemmas 8 and 9, the language K is IO-$\mathsf{PP}[+1]$-semi-decidable with stabilizing inputs.

MB: The current Proposition 7 is for IO-PP[<], but it is trivial to adapt it to IO-PP[+1].

Let $K' = \{w \in \Gamma^{\geq 2} \mid w \text{ is a valid configuration of } M \text{ and it leads to acceptance}\}$. By Proposition 52, K' is $\mathsf{PP}[+1]$-semi-decidable with stabilizing inputs. Let $g((x, y, z)) = y$. We have

$$L(M) \cap \Sigma^{\geq 2} = g(K \cap K').$$

So, by Lemmas 8 and 9, the language $L(M) \cap \Sigma^{\geq 2}$ is $\mathsf{PP}[+1]$-semi-decidable.

We can trivially show that $L(M) \cap \Sigma$ is $\mathsf{PP}[\emptyset]$-decidable. Thus, we are done by taking the union, with Lemma 6. ◀