Stably Computable Predicates are Semilinear

Dana Angluin, James Aspnes and David Eisenstat

Population Protocols

Introduced by [Angluin, Aspnes, Diamadi, Fischer and Peralta, PODC '04].

- Anonymous, finite-state agents
- Pairwise (ordered) interactions scheduled by a fair[a] adversary
- No faults (but see [Delporte-Gallet, Fauconnier, Guerraoui and Ruppert, DCOSS '06])
- No termination, stabilizing outputs

Similar models used to study epidemics, chemical reactions.

[a] Any configuration reachable infinitely often is reached infinitely often.

Example: Leader Election

States are $\{leader, follower\}$. When two agents in state $leader$ interact, one switches to $follower$.

If all agents start in state $leader$, eventually only one remains.

Stably Computing a Predicate

- Input is a multiset of symbols $\mathbf{x}$, which we write as a vector.
- Each of $|\mathbf{x}|$ agents gets one symbol, which determines its starting state.
- Each agent has its own 0-1 output, determined by its state.
- Agents must reach stable consensus.

Any boolean combination of stably computable predicates is stably computable.

Example: OR

Two states: $\{0, 1\}$

Input and output maps are the identity.

Joint transition function:

δ	0	1
0	0,0	1,1
1	1,1	1,1

Eventually, all agents have the same output, which is the OR of the inputs.

Example: Parity

Four states: $\{active, passive\} \times \{0, 1\}$

Input map $x \mapsto (active, x)$

Output map $(status, x) \mapsto x$

Active agents combine outputs and reduce their number:

$$(active, x), (active, y) \mapsto (active, x \oplus y), (passive, x \oplus y) \quad (1)$$

Passive agents copy their output from active agents:

$$(active, x), (passive, y) \mapsto (active, x), (passive, x) \quad (2)$$

Invariant: the parity of the active agents' outputs equals the parity of the input.

Eventually, exactly one agent is active (1).

Other agents copy its output (2), resulting in stable consensus.

Parity Protocol in Action

Input: (1,3)

Initial configuration:

(active,1) (active,1)

(active,1) (active,0)

(active,1) (active,1)

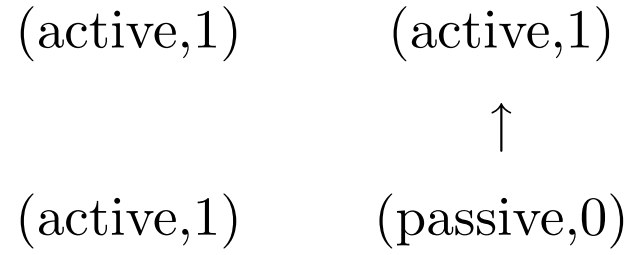
(active,1) (active,0)

$$(\text{active},1) \qquad (\text{active},1)$$

$$(\text{active},1) \quad \rightarrow \quad (\text{active},0)$$

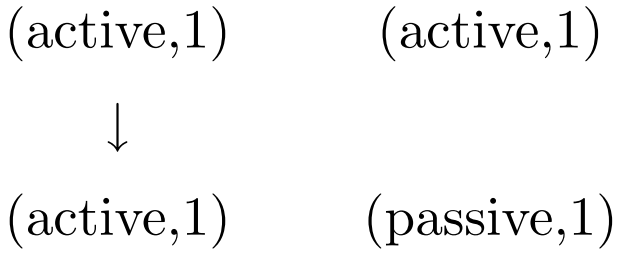
(active,1) (active,1)

(active,1) (passive,0)


(active,1)
(active,1)
(active,1)
(passive,0)

(active,1) (active,1)

(active,1) (passive,1)


(active,1)
(active,1)
(active,1)
(passive,1)

(active,0) (active,1)

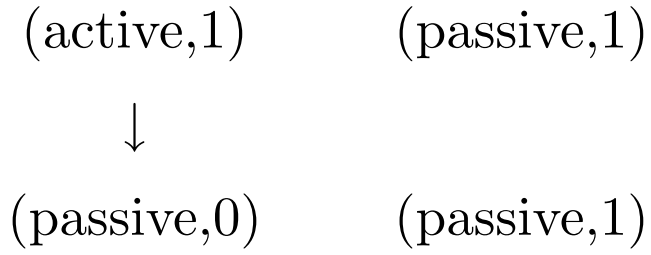
(passive,0) (passive,1)

$$(\text{active},0) \quad \rightarrow \quad (\text{active},1)$$

$$(\text{passive},0) \qquad\qquad (\text{passive},1)$$

(active,1) (passive,1)

(passive,0) (passive,1)


(active,1)
(passive,1)
(passive,0)
(passive,1)

(active,1) (passive,1)

(passive,1) (passive,1)

Example: Majority 1s

Six states: $\{active, passive\} \times \{-1, 0, 1\}$

Input map:

$$0 \mapsto (active, -1)$$

$$1 \mapsto (active, 1)$$

Output map:

$$(status, -1) \mapsto 0$$

$$(status, 0) \mapsto 0$$

$$(status, 1) \mapsto 1$$

Active agents combine outputs and reduce their number. If $x \neq y$,

$$(active, x), (active, y) \mapsto (active, x + y), (passive, x + y) \quad (3)$$

Passive agents copy their output from active agents as before.

Invariant: the sum of the data fields of active agents is the number of 1s in the input minus the number of 0s.

Rule (3) resolves all conflicts between active agents, which leads to stable consensus.

Semilinear Predicates

Revisiting the three predicates, a pattern emerges.

Let $\mathbf{x} = (x_0, x_1)$ be the input.

OR	$(x_0, x_1) \cdot (0, 1) \geq 1$
Parity	$(x_0, x_1) \cdot (0, 1) \equiv 1 \pmod 2$
Majority 1s	$(x_0, x_1) \cdot (-1, 1) > 0$

The table suggests two basic kinds of predicates:

$$\mathbf{x} \cdot \mathbf{v} \geq c$$

$$\mathbf{x} \cdot \mathbf{v} \equiv c \pmod{m},$$

with parameters $\mathbf{v}$, an integral vector, and integers c and m.

The boolean closure of these predicates is the class of **semilinear** predicates.

Semilinearity

Alternate characterizations:

- The multiplicities of strings of regular (or context-free) languages
- Predicates definable in the first order theory of $(\mathbf{N}, +)$
- Finite unions of sets of the form $b + \mathbf{N}p_1 + \ldots + \mathbf{N}p_k$, where $b, p_1, \ldots, p_k$ are vectors.

Some predicates that are not semilinear:

- $x_0 = 2^{x_1}$
- $x_0 < x_1\sqrt{2}$
- $x_0 x_1 = x_2$

Main Result

[AADFP '04]: all semilinear predicates are stably computable.

This paper: all stably computable predicates are semilinear.

Corollaries

Complete characterizations of the immediate, delayed, and queued transmission models from [Angluin, Aspnes, Eisenstat and Ruppert, OPODIS '05]

Characterization of the (composable) stabilizing inputs model of [Angluin, Aspnes, Chan, Fischer, Jiang and Peralta, DCOSS '05].

Characterization of a reversible model where executions may include "backwards" transitions.

Output Stability

A configuration is **output-stable** if

- all agents have the same output and
- no agent can ever change its output.

Positive example: in OR protocol, agents in states $1, 1, 1, 1, 1, 1$.

Negative example: in OR protocol, agents in states $0, 0, 1$.

All inputs eventually reach an output-stable configuration (stable computation).

Any subconfiguration of an output-stable configuration is output-stable.

Corollary: Any non-output-stable configuration contains a minimal non-output-stable subconfiguration.

In the OR protocol, the unique minimal non-output-stable configuration is $0, 1$.

In the Parity protocol, the minimal non-output-stable configurations are

- any configuration with exactly two agents that have different outputs
- $(active, 1), (active, 1)$, which can reach $(active, 0), (passive, 0)$.

Higman's Lemma

Any subset of $\mathbf{N}^d$ has finitely many $\leq$-minimal elements.

There are finitely many minimal non-output-stable configurations.

There is a constant k such that a configuration with k agents in each state contains each such minimal configuration.

If an output-stable configuration contains k agents in a given state, adding more won't destabilize it.

A Pumping Lemma

Suppose the predicate is true for x. Assume

$$x \xrightarrow{*} c$$

$$c + y \xrightarrow{*} c' \geq c,$$

and c is output-stable. If the agents in $c' - c$ are in states with multiplicity $\geq k$ in c, c' is output-stable.

Since c' contains c, c can "absorb" any number of copies of y. Thus the predicate is true for all the elements of $x + \mathbf{N}y$.

A Pumping Lemma

Suppose $x_1 < x_2 < x_3 < \cdots$ is an increasing sequence of inputs on which the predicate is true.

$$x_1 \xrightarrow{*} c_1$$

$$c_1 + (x_2 - x_1) \xrightarrow{*} c_2$$

$$c_2 + (x_3 - x_2) \xrightarrow{*} c_3$$

$$\vdots$$

A corollary of Higman's lemma implies that $c_1, c_2, c_3, \ldots$ has an infinite increasing subsequence.

There exist $i < j$ such that the predicate is true on $x_i + \mathbf{N}(x_j - x_i)$.

Decomposition into Cones

Using the pumping lemma techniques and Higman's lemma, we can write the inputs where the predicate is true as a finite union of integral "cones".

Here, a cone is a set that can be written

$$x + \mathbf{N}y_1 + \mathbf{N}y_2 + \mathbf{N}y_3 + \dots$$

Example: Parity is true on $(0,1) + \mathbf{N}(1,0) + \mathbf{N}(0,2)$.

Example: Majority 1s is true on $(0,1) + \mathbf{N}(1,1) + \mathbf{N}(0,1)$.

Separating the Cones

Write the support and its complement as a finite union of cones.

Strategy: for each pair of cones, find a semilinear predicate that agrees with the computed predicate on the cones except on finitely many hyperplanes.

Use induction to show the semilinearity of the predicate on the hyperplanes and express the computed predicate as a boolean combination of semilinear predicates.

Parity or Majority?

Translate the cones to the origin and look at the intersection.
There are two cases:

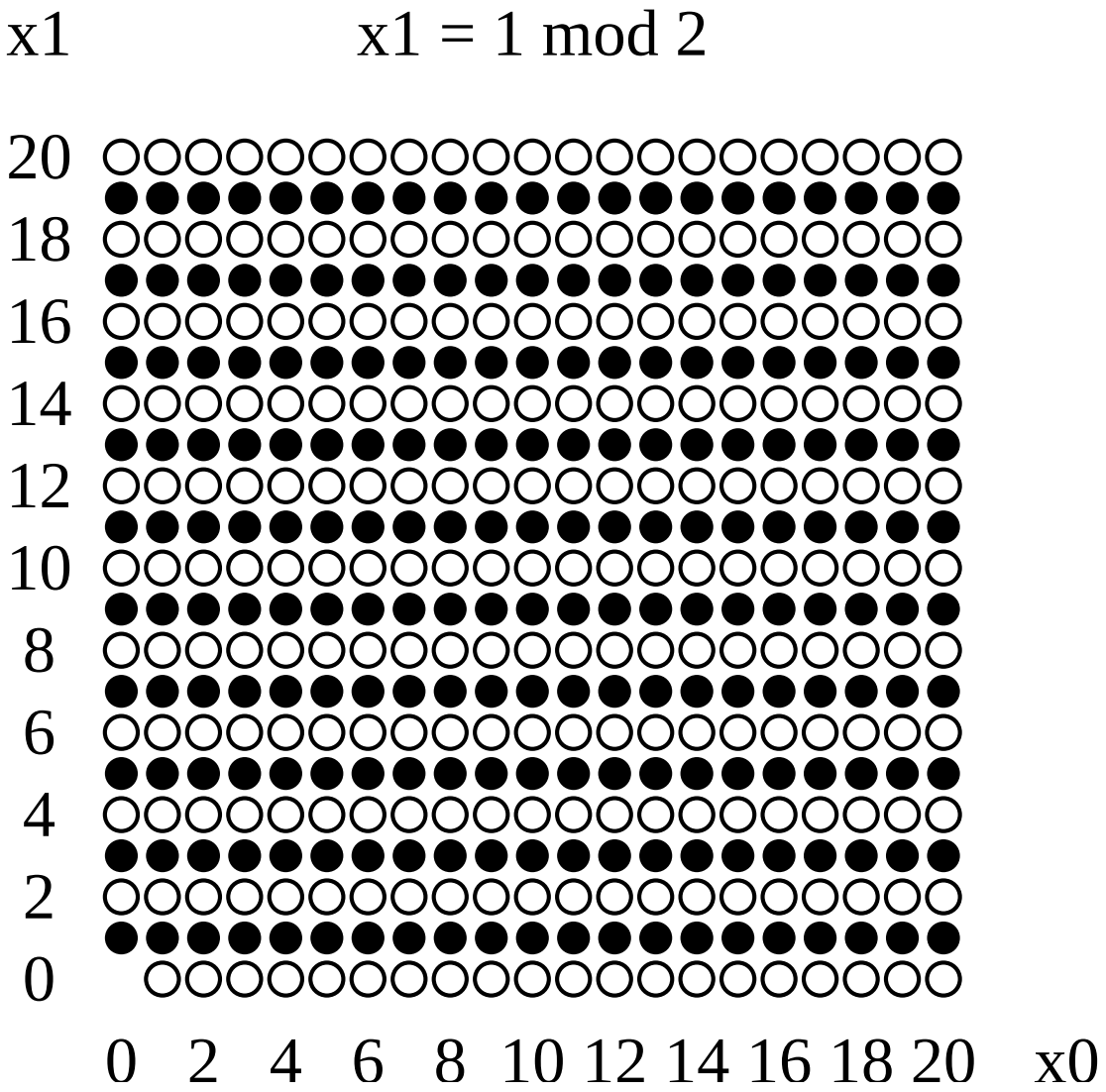
- The intersection **cannot** be contained in a hyperplane.
- The intersection **can** be contained in a hyperplane.

Parity

Case: intersection is not lower-dimensional.

Example: Parity, since both shifted cones are $\mathbf{N}(1,0) + \mathbf{N}(0,2)$.

We can show algebraically that a boolean combination of modulo predicates separates the original cones.

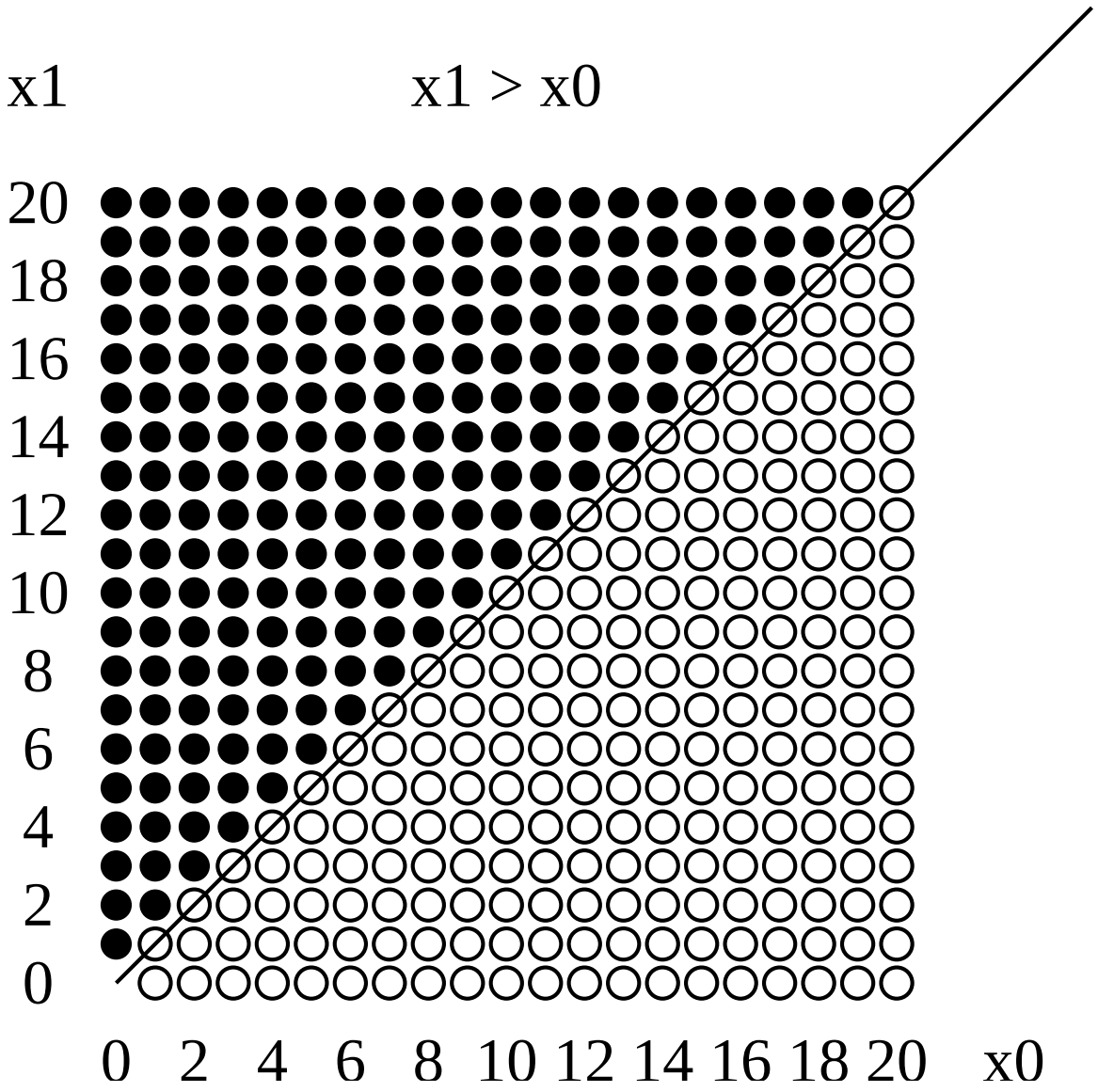

x1
x1 = 1 mod 2
20
18
16
14
12
10
8
6
4
2
0
0 2 4 6 8 10 12 14 16 18 20
x0

Majority

Case: intersection is lower-dimensional.

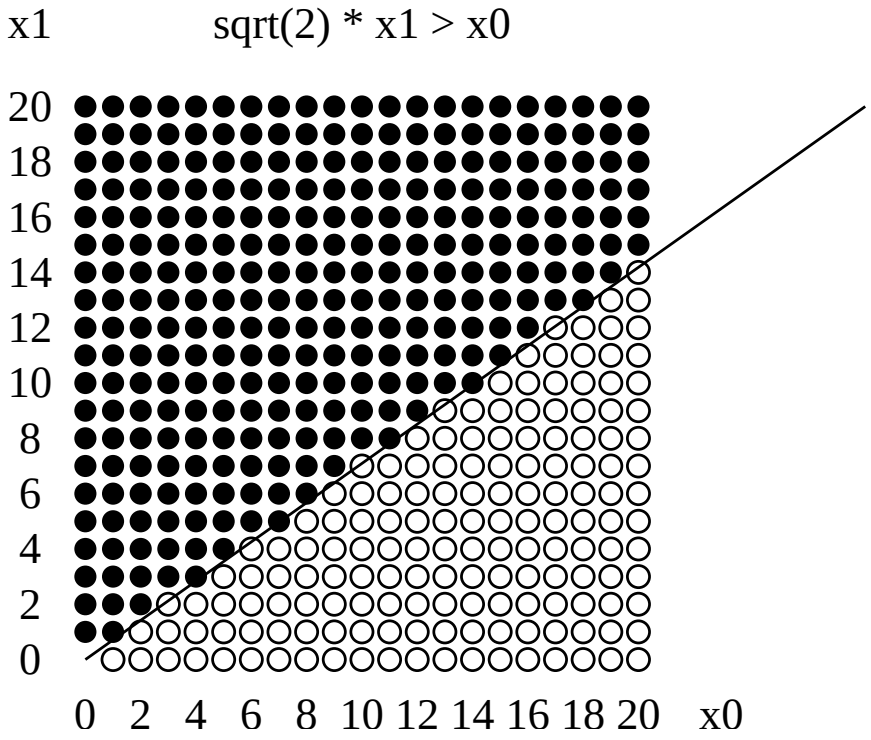
Example: Majority, since
$(\mathbf{N}(1,0) + \mathbf{N}(1,1)) \cap (\mathbf{N}(0,1) + \mathbf{N}(1,1)) = \mathbf{N}(1,1)$.

The interiors of the convex closures of the cones are disjoint, so separate them with a hyperplane (in $\mathbf{R}^d$).


x1
x1 > x0
20
18
16
14
12
10
8
6
4
2
0
0 2 4 6 8 10 12 14 16 18 20
x0

Problem

The hyperplane might not have an integral normal vector.



Solution: use the pumping lemma again.

Thanks