Computability in Population Protocols

Peva Blanchard
EPFL 2014/2015
Population Protocol

Agent: no id, small memory
Population Protocol

Population arbitrary size $n$
Population Protocol

Initially: sensors give data
Population Protocol

Initially: sensors give data

red  green  blue
Population Protocol

Initially: sensors give data

red  green  blue

initial states
Population Protocol

Agents move
Population Protocol

Agents move
Population Protocol

Agents move
Population Protocol

Agents move
Population Protocol

Protocol rule

\[ a,b \rightarrow c,d \]
Population Protocol

Protocol rule

\[ a, b \rightarrow c, d \]
Population Protocol

Protocol rule

a, b \rightarrow c, d
What is computable?
What is computable?

What can the agents know about the initial configuration?
What is computable?

What can the agents know about the initial configuration?

- No id
- Only numbers of "species"
What is computable?

What can the agents know about the initial configuration?

\[ \#\text{green} \leq 4 \]
What is computable?

What can the agents know about the initial configuration?

\[ \#\text{green} \leq 4 \]
\[ \#\text{green} \leq \#\text{blue} \]
What is computable?

What can the agents know about the initial configuration?

\[ \#\text{green} \leq 4 \]
\[ \#\text{green} \leq \#\text{blue} \]
\[ \#\text{green} \leq \#\text{blue} + 2\#\text{red} \]
\[ \#\text{green} \leq \#\text{blue} \times \#\text{red} \]
What is computable?

What can the agents know about the initial configuration?

- \#green \leq 4
- \#green \leq \#blue
- \#green \leq \#blue + 2 \#red
- \#green \leq \#blue \times \#red

OK

NOK
predicate $P$ computable
There exists a protocol A such that

predicate P computable
There exists a protocol $A$ such that for any population size predicate $P$ computable.
There exists a protocol \( A \) such that
for any population size
for any input assignment
There exists a protocol $A$ such that
for any population size
for any input assignment
eventually all agents output $P\left(\ldots\right)$
\#green - 2 \#blue \leq 4 \quad (naive)
Assume a unique leader

with counter, initially 0
Assume a unique leader

with counter, initially 0

\[ \#\text{green} - 2 \#\text{blue} \leq 4 \quad \text{(naive)} \]
Assume a unique leader

with counter, initially 0

\[ \#\text{green} - 2 \#\text{blue} \leq 4 \quad \text{(naive)} \]
#green - 2 #blue ≤ 4 (naive)

Assume a unique leader with counter, initially 0

and mark them as seen.
Assume a unique leader
with counter, initially 0

\[
\text{green} - 2 \text{ blue} \leq 4 \quad \text{(naive)}
\]

and mark them as seen.

Leader output 1 iff counter \(\leq 4\)
other agents copy leader's output.
Assume a unique leader with counter, initially 0.

#1. How to elect a leader?

L, counter -= 2

L, counter += 1

#2. How to bound memory?

and mark them as seen.

Leader output 1 iff counter ≤ 4

other agents copy leader's output.

\[ \text{green} - 2 \text{ blue} \leq 4 \] (naive)
\#green - 2 \#blue \leq 4

Leader election: each agent has a leader bit initially, all leaders
$$\#\text{green} - 2 \leq \#\text{blue} \leq 4$$

Leader election: each agent has a leader bit initially, all leaders

![Penguin images showing leader election process]
\#green - 2 \#blue \leq 4

Leader election: each agent has a leader bit initially, all leaders
#green - 2 #blue \leq 4

Leader election: each agent has a leader bit initially, all leaders
\#green - 2 \#blue \leq 4

Counter issue

Fix a large enough limit, e.g. \( s \geq 5 \)
#green - 2 #blue \leq 4

Counter issue

Fix a large enough limit, e.g. \( s \geq 5 \)

All agents have counter \(-s \leq u \leq s\)
Counter issue

Fix a large enough limit, e.g. \( s \geq 5 \)

All agents have counter \(-s \leq u \leq s\)

initially \( u_{init} = -2 \)

initially \( u_{init} = 1 \)
Counter issue

Fix a large enough limit, e.g. \[ s \geq 5 \]

All agents have counter \[-s \leq u \leq s\]

initially \[ u_{\text{init}} = -2 \]

\[ u_{\text{init}} = 1 \]

\[ \sum \text{agents} u = \# \text{green} - 2 \# \text{blue} \leq 4 \]
#green - 2 #blue ≤ 4

Counter issue

\[ L \overset{u}{\rightarrow} * \overset{u'}{\rightarrow} L \overset{q}{\rightarrow} \downarrow \overset{r}{\rightarrow} \]
Counter issue

\[(u, u') = u + u'\]
\[r(u, u') = 0\]
Counter issue

\[ q(u, u') = \max\{-s, \min\{s, u + u'\}\} \]

\[ r(u, u') = u + u' - q(u, u') \quad \text{remainder} \]
$\#\text{green} - 2 \#\text{blue} \leq 4$

Counter issue

$q(u, u') = \max\{-s, \min\{s, u + u'\}\}$

$r(u, u') = u + u' - q(u, u')$  \text{remainder}

Invariant  \[ \sum_{\text{agents}} u = \#\text{green} - 2 \#\text{blue} \]
#green - 2 #blue \leq 4

Putting things together

$u_{\text{init}} = -2$

$u_{\text{init}} = 1$

$q(u, u') = \max\{-s, \min\{s, u + u'\}\}$

$r(u, u') = u + u' - q(u, u')$

non-leaders copy leader's output
Proof strategy

A. Eventually a single leader

B. Eventually, the leader collects the value

\[ \max\{-s, \min\{s, \# - 2 \#\}\} \]

C. Eventually, the agents produce correct outputs
Proof strategy

A. Eventually a single leader

B. Eventually, the leader collects the value

$$\max\{-s, \min\{s, \# - 2 \# \}\}$$

C. Eventually, the agents produce correct outputs
Proof strategy

A. Eventually a single leader

B. Eventually, the leader collects the value

\[ \max\{-s, \min\{s, \# \cdot -2 \#\}\} \]

C. Eventually, the agents produce correct outputs
Proof strategy

\[ u_L = \max \{-s, \min \{s, \# - 2 \# \ldots \}\} \]
Proof strategy

\[ u_L = \max\{-s, \min\{s, \# - 2\#\}\} \]

If \( \# - 2\# \leq 4 \) then \( u_L = \begin{cases} 
\# - 2\# \\
\text{or} \quad -s 
\end{cases} \)

If \( \# - 2\# > 4 \) then \( u_L = \begin{cases} 
\# - 2\# \\
\text{or} \quad s 
\end{cases} \)
Proof strategy

C. Eventually, correct outputs

\[ u_L = \max \{ -s, \min \{ s, \# \phantom{-2} - 2 \# \} \} \]

If \( \# \phantom{-2} - 2 \# \leq 4 \) then \( u_L = \begin{cases} \# \phantom{-2} - 2 \# \\ \text{or} \ - s \end{cases} \)

If \( \# \phantom{-2} - 2 \# > 4 \) then \( u_L = \begin{cases} \# \phantom{-2} - 2 \# \\ \text{or} \ s \end{cases} \)

Leader gets correct output
Proof strategy

\[ u_L = \max\{-s, \min\{s, \# - 2 \#\}\} \]

If \( \# - 2 \# \leq 4 \) then \( u_L = \begin{cases} \# - 2 \# \text{ or } -s \end{cases} \)

If \( \# - 2 \# > 4 \) then \( u_L = \begin{cases} \# - 2 \# \text{ or } s \end{cases} \)

Leader gets correct output

Others get correct output on meeting the leader
Proof strategy

C. Eventually, correct outputs

\[ u_L = \max\{-s, \min\{s, \# - 2 \#\}\} \]

- If \( \# - 2 \# \leq 4 \) then \( u_L = \begin{cases} \# - 2 \# \\
\text{or} \quad -s \end{cases} \)

- If \( \# - 2 \# > 4 \) then \( u_L = \begin{cases} \# - 2 \# \\
\text{or} \quad s \end{cases} \)

Q.E.D.

Leader gets correct output

Others get correct output on meeting the leader
Presburger arithmetics

\[ a_1 \cdot \#\sigma_1 + \cdots + a_k \cdot \#\sigma_k < c \]

\[ a_1 \cdot \#\sigma_1 + \cdots + a_k \cdot \#\sigma_k = c \mod m \]

boolean combinations
\[ a_1 \cdot \#\sigma_1 + \cdots + a_k \cdot \#\sigma_k < c \]

\[ a_1 \cdot \#\sigma_1 + \cdots + a_k \cdot \#\sigma_k = c \mod m \]

boolean combinations

Presburger arithmetics

(beware: integer coefficients)
What about multiplication?

\[ \# \cdot \# \leq 10 \]
What about multiplication?

\[ \# \cdot \# \leq 10 \quad \text{NO!} \]
What about multiplication?

Intuition

\[
\begin{align*}
1 \cdot \# & \leq 10 \\
2 \cdot \# & \leq 10 \\
3 \cdot \# & \leq 10 \\
4 \cdot \# & \leq 10 \\
5 \cdot \# & \leq 10 \\
\end{align*}
\]

\text{etc.}
What about multiplication?

Intuition

\[
1 \cdot \# \leq 10 \quad 2 \cdot \# \leq 10 \\
3 \cdot \# \leq 10 \quad 4 \cdot \# \leq 10 \\
5 \cdot \# \leq 10 \quad \text{etc.}
\]

previous approach requires too much memory