Terminating Distributed Construction of Shapes and Patterns in a Fair Solution of Automata

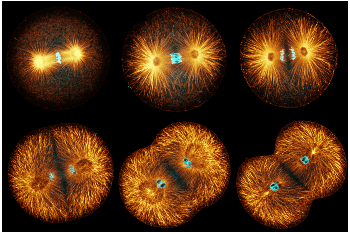
Othon Michail

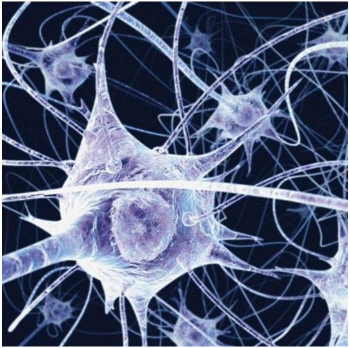
Computer Technology Institute & Press “Diophantus” (CTI)

34th Annual ACM Symposium on Principles of Distributed Computing (PODC)
July 21-23, 2015
Donostia-San Sebastián, Spain

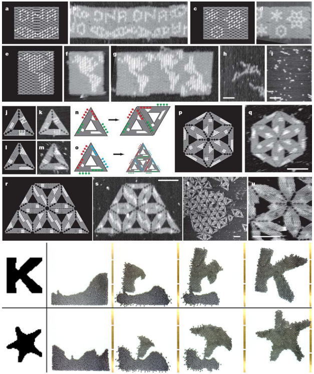
Motivation

- A wide range of physical/biological systems are governed by algorithmic laws
- Usually collections of very large numbers of simple distributed entities
- Higher-level properties are the outcome of coexistence and constant interaction (cooperative and/or competing) of such entities
- Goal:
  - Reveal the algorithmic aspects of physical systems
  - Develop innovative artificial systems inspired by them





Existing Approaches



- Cellular Automata model neural activity, self-replication, bacterial growth, ...
- Population Protocols [AADFP, PODC '04] have been proved formally equivalent to Chemical Reaction Networks [Doty, SODA '14]
- Network Constructors [Michail, Spirakis, PODC '14]: abstract and simple model of distributed network formation
- DNA self-assembly: single-stranded DNA molecules folded into arbitrary nanoscale shapes and patterns [Rothemund, Nature '06]
- Nubot [WCGDWY, ITCS '13]: model for self-assembled structures with active molecular components
- Kilobot [RCN, Science '14]: programmable self-assembly of complex 2D shapes by a swarm of 1000 simple autonomous robots

Long-term Vision: Programmable Matter

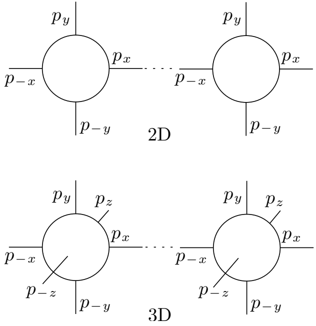
- Manipulate matter via information-theoretic and computing mechanisms and principles
- Incorporation of information to the physical world
- Plausible future outcome of progress in high-volume nanoscale assembly
- Physical realization of any computer-generated object
- Profound implications for how we think about chemistry and materials
- Materials will become user-programmed, smart, and adaptive
- It will change the way we think about engineering and manufacturing

Network Constructors [Michail,Spirakis PODC '14]

- Abstract model of algorithmic (distributed) network construction
- Tiny and weak devices (e.g. tiny nanorobots or programmable molecules), uniform and anonymous
- Passively mobile (e.g. inside the human circulatory system): adversary scheduler
- Cooperate by pairwise interactions and by creating bonds with each other
- Can construct very complex networks by self-organization and by simulating TMs
- Very useful theoretically but less realistic w.r.t. systems

A More Realistic Version

- Adjust some of the abstract parameters of NETs
- Allowable interactions are geometrically constrained (by already formed structures)
- Each device can connect to other devices only via a limited number of ports (4 in 2D and 6 in 3D)



- Connections are made at unit distance and are perpendicular to neighboring connections
- Known universal constructors do not apply in this case

Overview-Contribution

- We overcome the inability of such systems to terminate and sequentially compose routines
- Nodes do not know and cannot store n nor influence time
- Storage: Exploit the ability of nodes to self-assemble into memories
- Counting n: Exploit a "well-mixed" assumption and give the first protocol that terminates counting n w.h.p.
    - Allows to improve stabilization to termination
    - Allows sequential composition of subroutines and results in more natural protocols
- Universality results: Exploit counting to construct w.h.p. arbitrarily complex shapes and patterns by terminating protocols
- We also give direct constructors for some basic problems and study the problem of shape self-replication

The Model

1. Q: finite set of node-states,
2. $q_0 \in Q$: initial node-state,
3. $Q_{out} \subseteq Q$: set of output node-states, and
4. $\delta : (Q \times P) \times (Q \times P) \times \{0,1\} \to Q \times Q \times \{0,1\}$: the transition function, $P = \{u, r, d, l\}$ is the set of ports

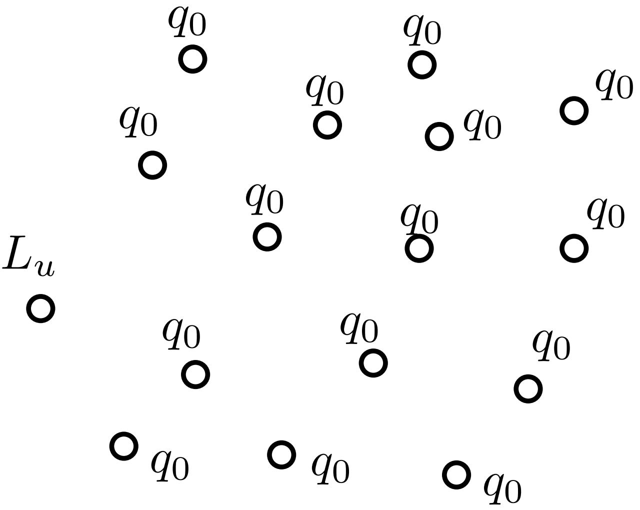
- In every step, a pair $(v_1, p_1)(v_2, p_2)$ is selected by the scheduler and v_1, v_2 interact via their p_1, p_2 ports according to δ
- Valid configuration: its connected components are subnetworks of the 2D grid network
- Uniform random scheduler: selects independently and uniformly at random from the permitted interactions (leading to valid config.)
- Output shape: nodes that are in output (or halting) states and edges between them that are active

An Illustration: Spanning Square

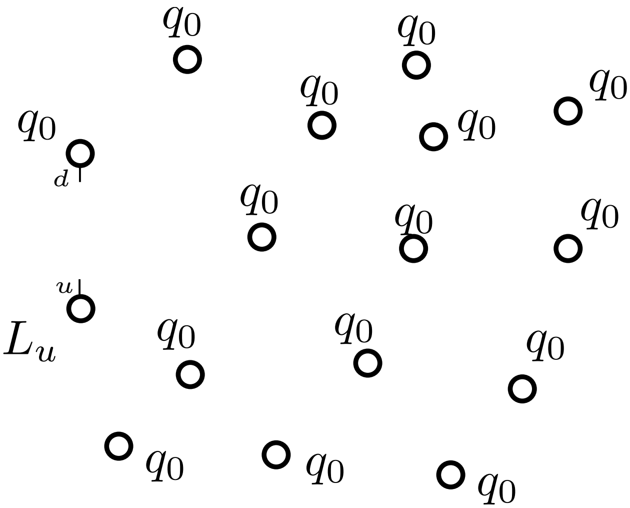
- δ:

$$
\begin{aligned}
(L_u, u), (q_0, d), 0 &\rightarrow (q_1, L_r, 1)\\
(L_r, r), (q_0, l), 0 &\rightarrow (q_1, L_d, 1)\\
(L_d, d), (q_0, u), 0 &\rightarrow (q_1, L_l, 1)\\
(L_l, l), (q_0, r), 0 &\rightarrow (q_1, L_u, 1)\\
(L_u, u), (q_1, d), 0 &\rightarrow (L_l, q_1, 1)\\
(L_r, r), (q_1, l), 0 &\rightarrow (L_u, q_1, 1)\\
(L_d, d), (q_1, u), 0 &\rightarrow (L_r, q_1, 1)\\
(L_l, l), (q_1, r), 0 &\rightarrow (L_d, q_1, 1)
\end{aligned}
$$

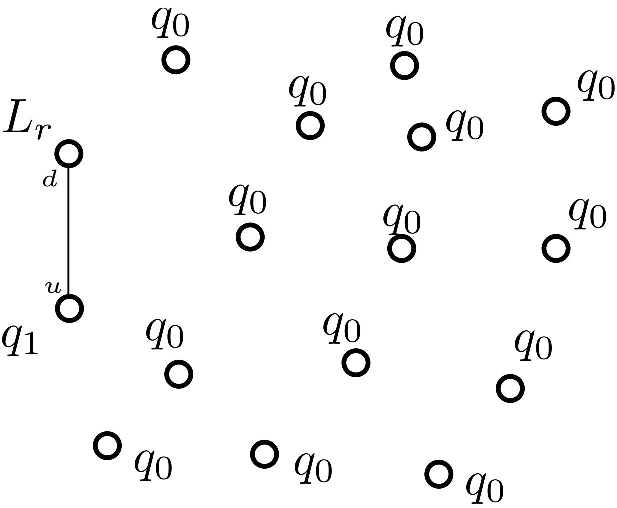
Spanning-Square Protocol

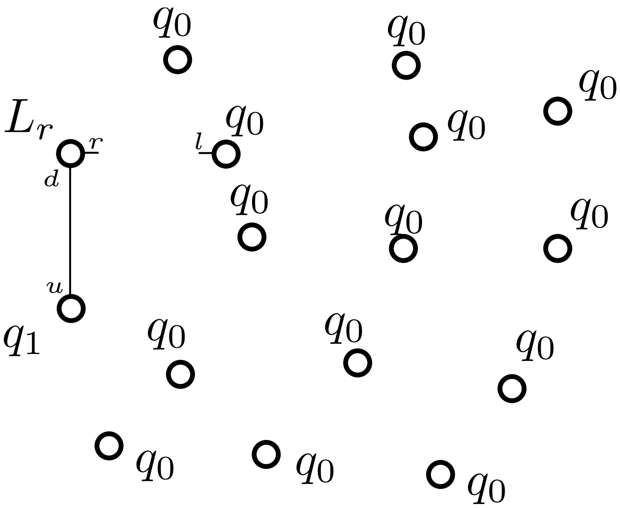


Spanning-Square Protocol

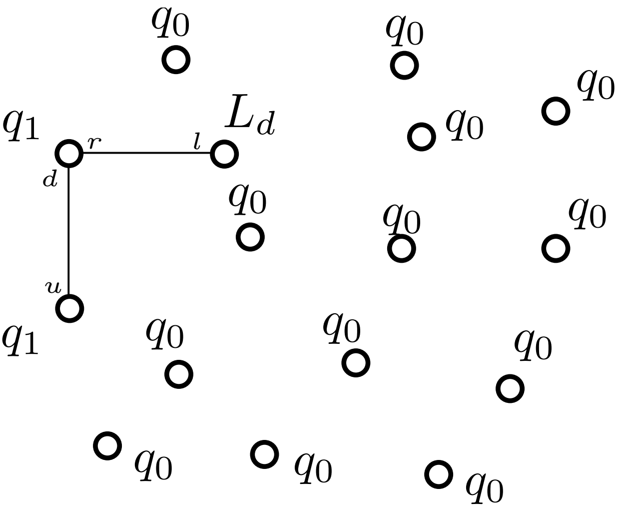


Spanning-Square Protocol

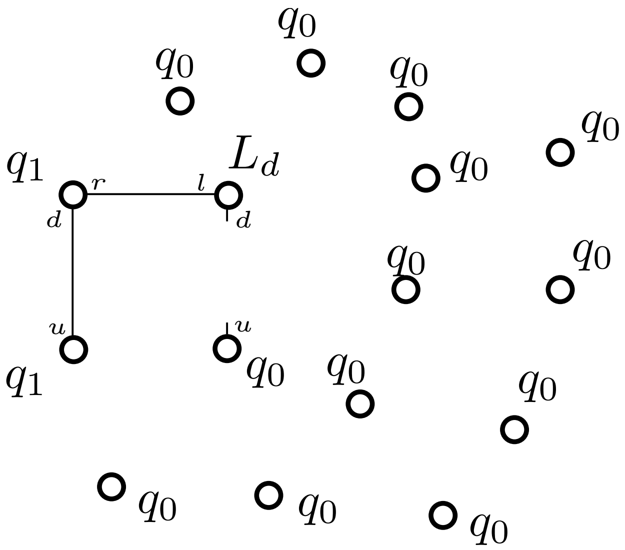



q_0
q_0
q_0
L_r
r
l
q_0
q_0
d
q_0
q_0
q_0
u
q_1
q_0
q_0
q_0
q_0
q_0
q_0

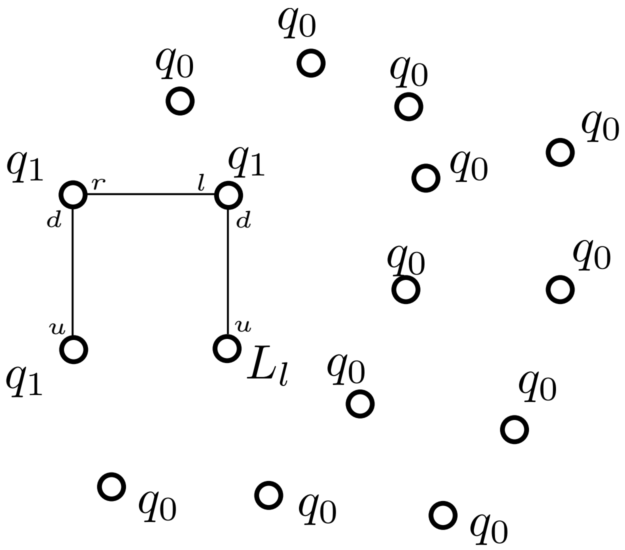
Spanning-Square Protocol



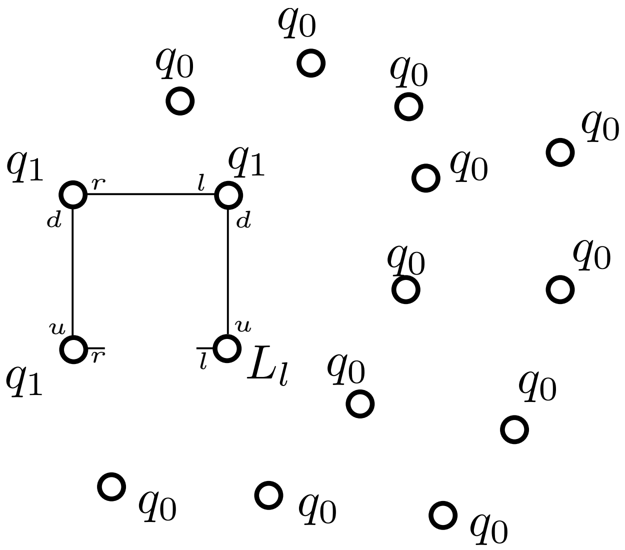
Spanning-Square Protocol

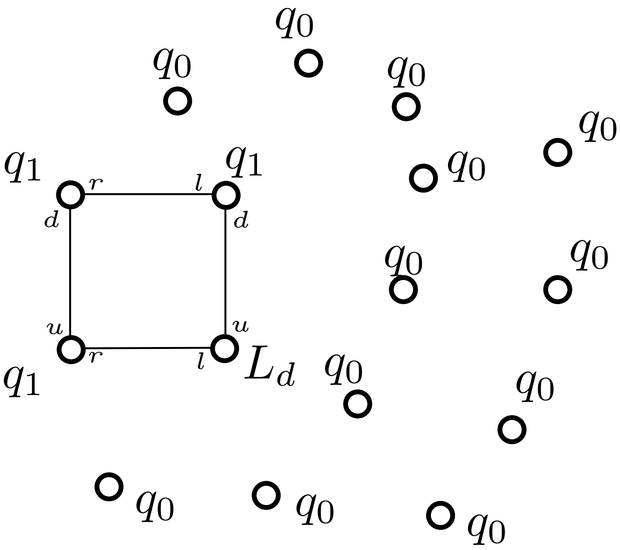


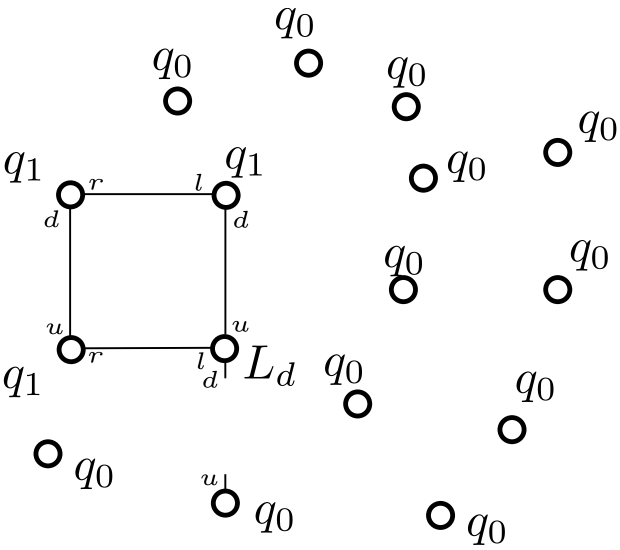
Spanning-Square Protocol



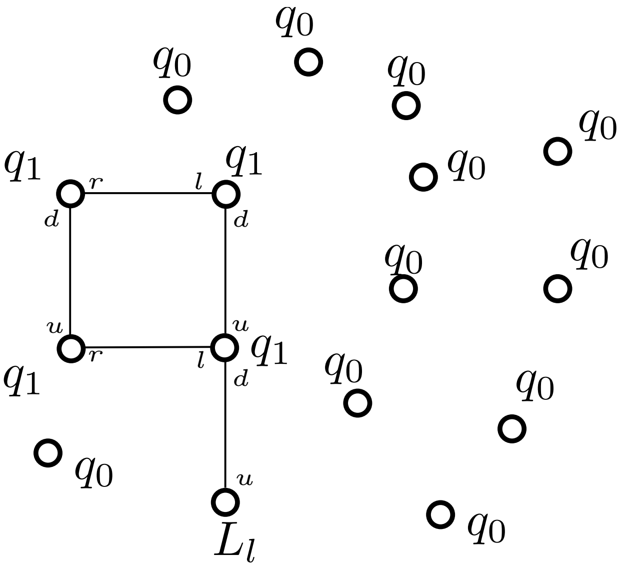
Spanning-Square Protocol



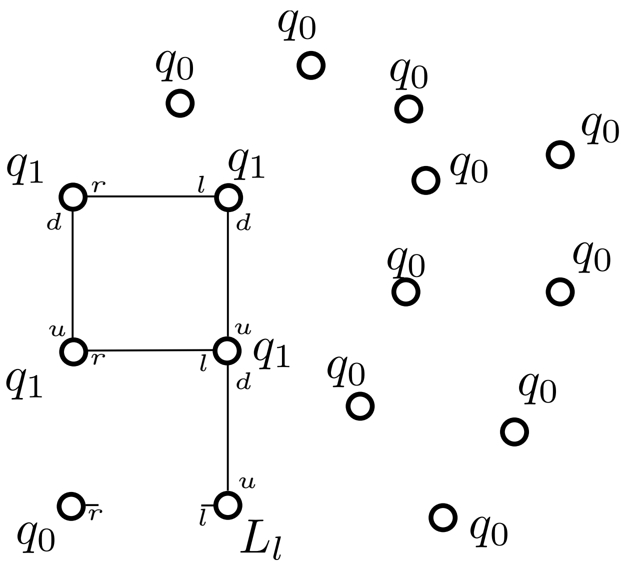

q_0
q_0
q_0
q_0
q_1
q_1
q_0
r
l
d
d
q_0
q_0
u
u
r
l
L_d
q_0
q_1
q_0
q_0
q_0
q_0


q_0
q_0
q_0
q_0
q_1
q_1
r
l
q_0
d
d
q_0
q_0
u
u
r
l
L_d
q_1
d
q_0
q_0
q_0
u
q_0
q_0

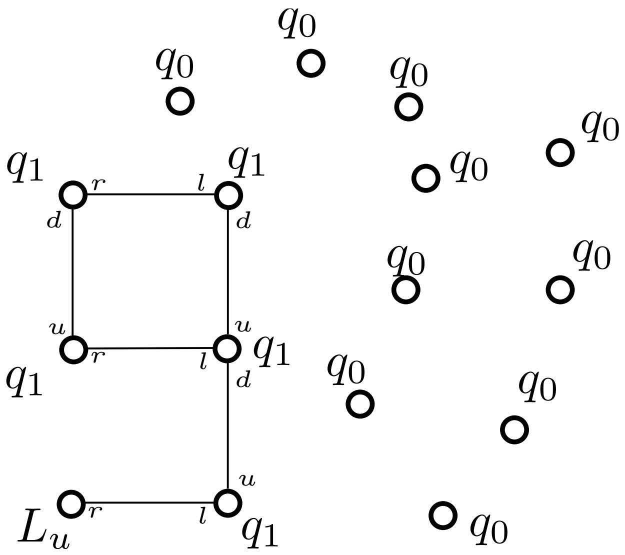
Spanning-Square Protocol



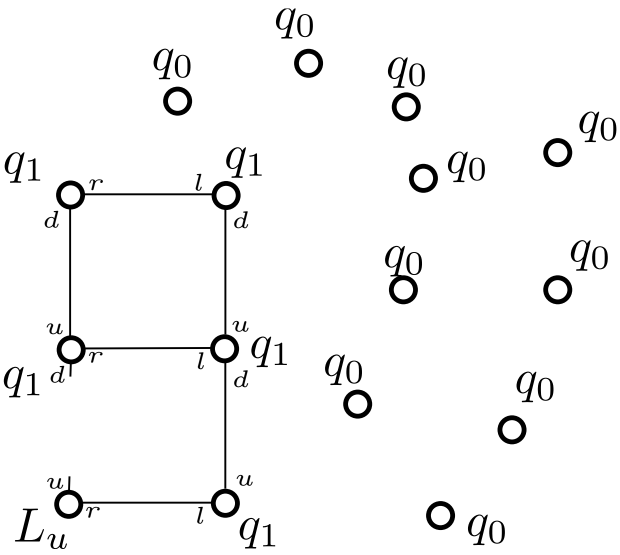
Spanning-Square Protocol



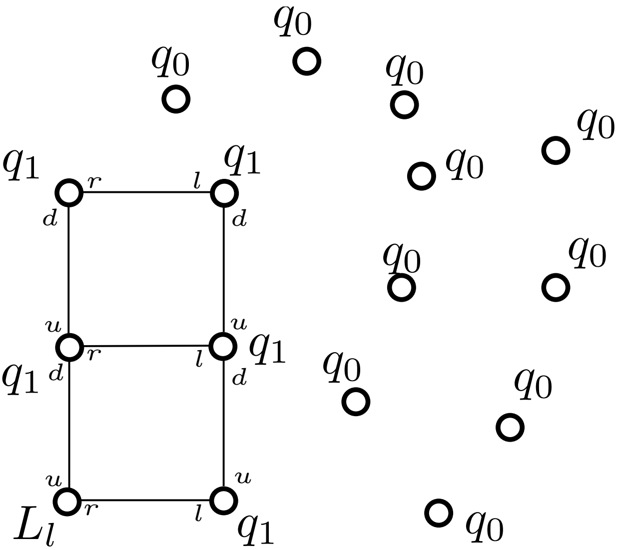
Spanning-Square Protocol



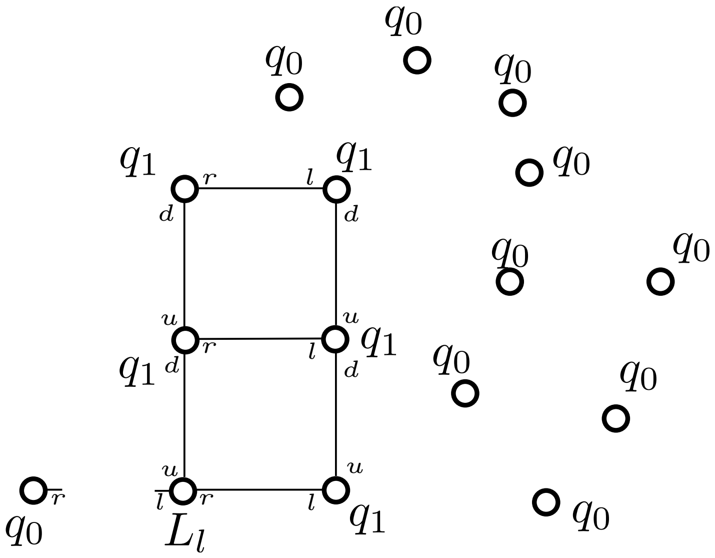
Spanning-Square Protocol

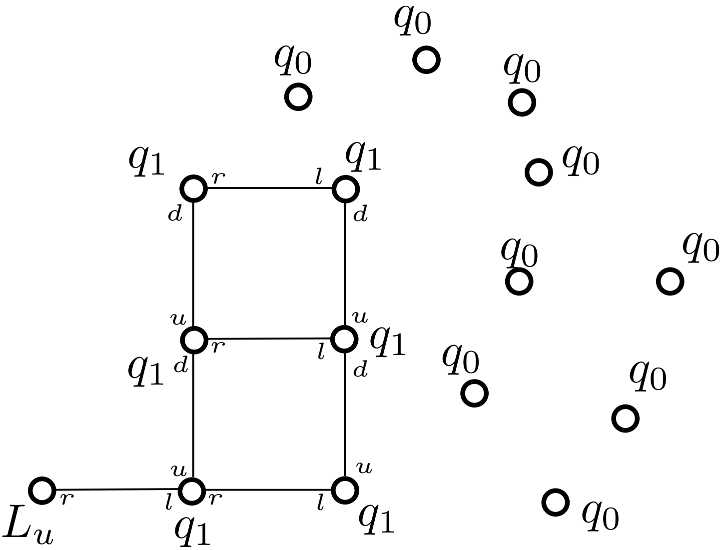


Spanning-Square Protocol

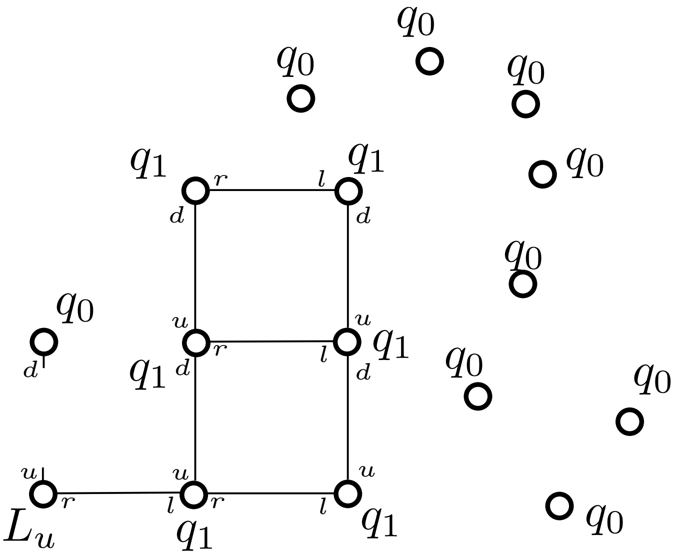


Spanning-Square Protocol

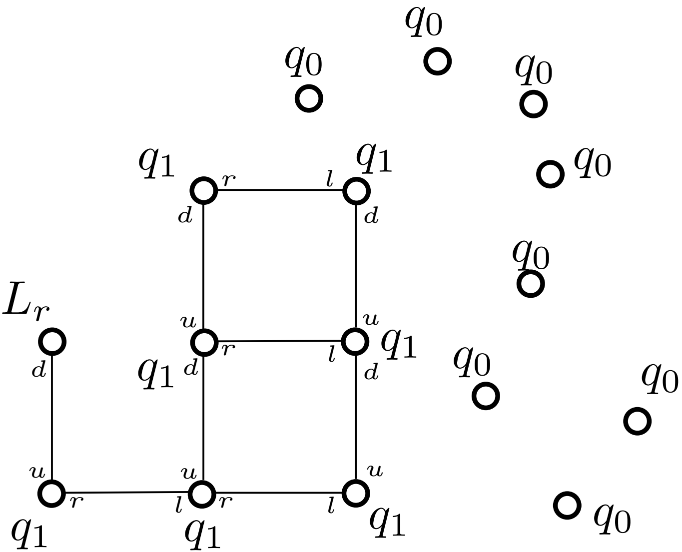



q_0
q_0
q_0
q_1
q_1
r
l
d
d
q_0
q_0
q_0
u
u
q_1
r
l
q_1
d
d
q_0
q_0
u
u
r
l
r
l
L_u
q_1
q_1
q_0

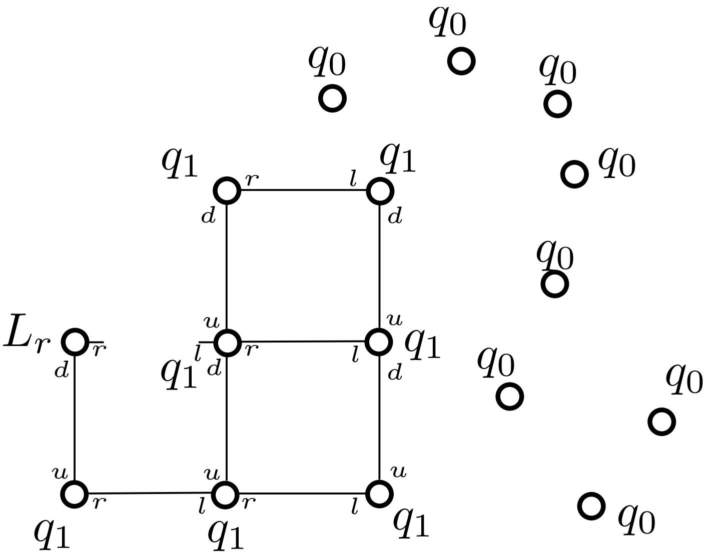
Spanning-Square Protocol



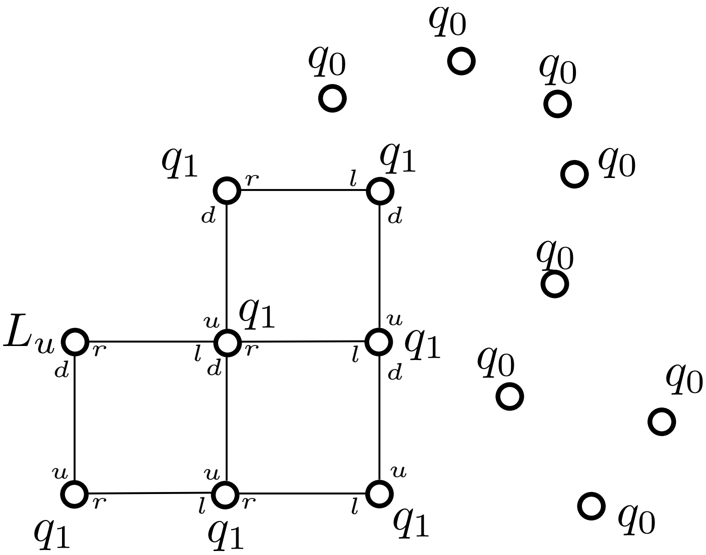
Spanning-Square Protocol



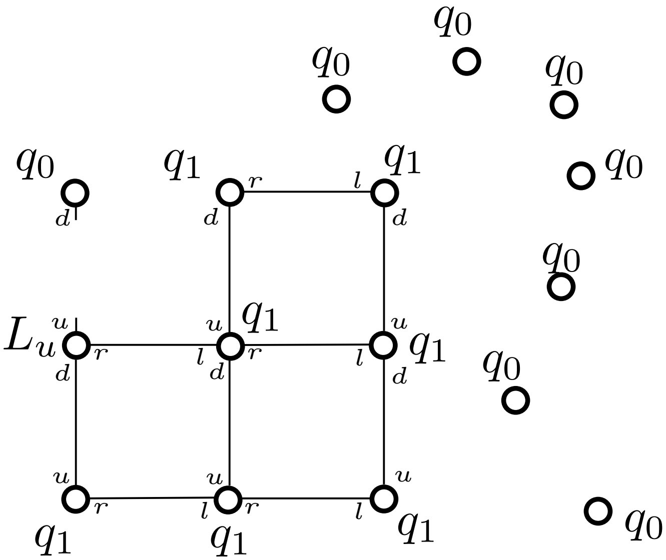
Spanning-Square Protocol



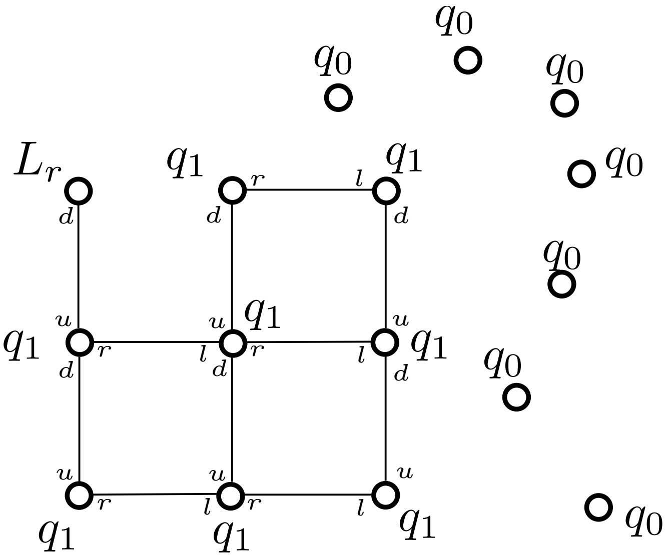
Spanning-Square Protocol



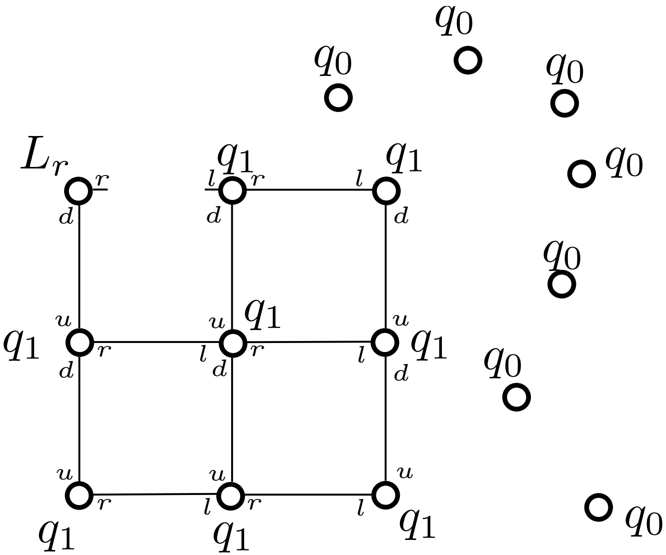
Spanning-Square Protocol



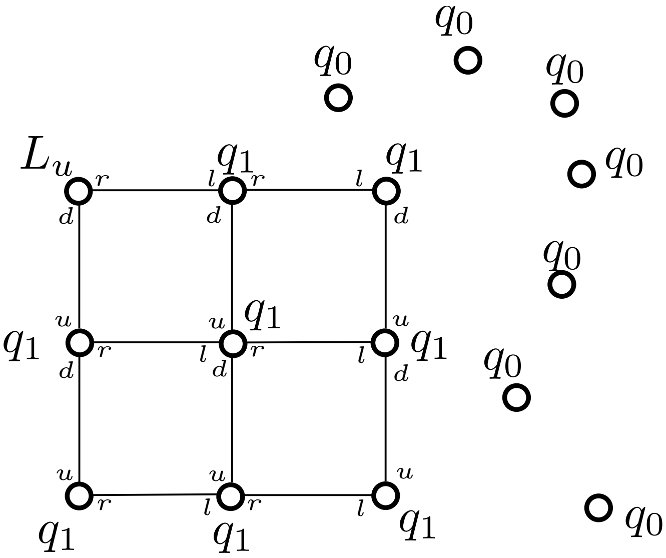
Spanning-Square Protocol



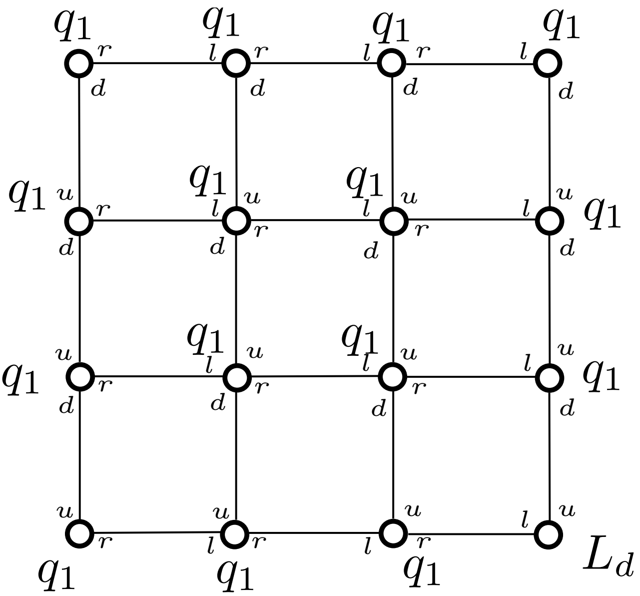
Spanning-Square Protocol



Spanning-Square Protocol



Spanning-Square Protocol



Probabilistic Counting

- Uniform random scheduler
- Terminating protocol that counts n (or a large fraction of n) w.h.p.
- The best that we can hope for
  - There is no protocol that always terminates and is always correct
- Main subroutine of our generic constructors
- Allows for sequential composition and avoids perpetual reinitializations

Fast Probabilistic Counting with a Leader

- Unique leader l, all other in q_0 initially
- Disregard for a while ports, geometry, and link activations
- The scheduler selects in every step equiprobably one of the $n(n-1)/2$ possible node pairs
- Assume for simplicity that the leader can store two n-counters in its memory
- Classical PP with an additional leader with linear memory
- In the sequel we adjust the protocol to make it work in our model
  - We only keep the unique leader assumption (but drop its memory)

Counting-Upper-Bound Protocol

- $l(r_0, r_1)$: The state of l, where r_0, r_1 are the values of the two counters, $0 \leq r_0, r_1 \leq n$
- Rules:

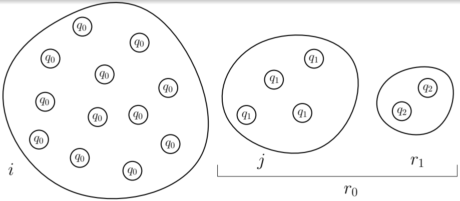
$$(l(r_0, r_1), q_0) \to (l(r_0 + 1, r_1), q_1)$$
$$(l(r_0, r_1), q_1) \to (l(r_0, r_1 + 1), q_2), \text{ and}$$
$$(l(r_0, r_1), \cdot) \to (halt, \cdot) \text{ if } r_0 = r_1$$

- r_0 counts the number of q_0s in the population
- r_1 counts the number of q_1s in the population
- When a q_0 (q_1) is counted it is converted to q_1 (q_2)
- Terminates when $r_0 = r_1$ for the first time
- We also give to r_0 an initial head start of b (constant)

Counting-Upper-Bound Protocol

Theorem

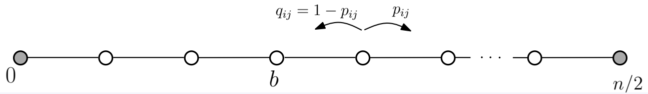
Counting-Upper-Bound halts in every execution. Moreover, if the scheduler is a uniform random one, when this occurs, w.h.p. it holds that $r_0 \geq n/2$.



Proof

- $p_{ij} = i/(i+j)$: *probability that an effective interaction is an* (l, q_0)
- $q_{ij} = 1 - p_{ij} = j/(i+j)$: *probability that it is an* (l, q_1)
- *r.w. on a line with* $n+1$ *positions* $0, 1, \ldots, n$
- *a particle begins from position* b*, absorbing barrier at* 0*, and reflecting at* n*, position corresponds to* $r_0 - r_1 = j$

Counting-Upper-Bound Protocol



- *"difficult" r.w.: the transition probabilities depend on the position j and also on $i + j$ which decreases in time*
- *upper bound* $\mathrm{P}[\textit{failure}] = \mathrm{P}[\textit{reach 0 before } r_0 \geq n/2 \textit{ holds}] \leq$
- *reduce it to a r.w. that does not depend on $i + j$*
  - *Ehrenfest r.w. [Kac, AMM '47]*
- *further reduce to a r.w. that does not depend on j, with barriers 0 and b*
- *repeat the classical ruin problem n times + Boole-Bonferroni*
- $\mathrm{P}[\textit{failure}] \leq \frac{1}{n^{b-2}}$

□

Further Remarks

- Expected running time: $O(n^2 \log n)$ interactions
- Experiments show that in most cases the estimation is closer to $(9/10)n$
- Exact value of n: l waits an additional large polynomial of r_0
- We suspect (also experimental evidence) that the unique leader is necessary
  - very interesting open problem
  - w.h.p. all states coexist with $\Theta(n)$ cardinalities [Doty, SODA '14]
  - a node may observe the same as in a fixed population and terminate after meeting a few nodes
- If there is no leader but UIDs we can solve the problem
  - The maximum id can be made to simulate the behavior of a leader

Generic Constructors

- Characterization for the class of constructible 2D shape languages
- Simulate shape-constructing TMs to realize their output-shape in the distributed system

1. Adapt Counting-Upper-Bound to obtain w.h.p. a line of length $\Theta(\log n)$, containing n in binary
2. The leader exploits knowledge of n to construct a $\sqrt{n} \times \sqrt{n}$ square
3. Simulate the TM on the square n distinct times, one for each pixel
   - input: index of pixel and $\sqrt{n}$, in binary
   - output: on or off (decision for the corresponding pixel)
4. Release the connected shape consisting of the on pixels

Theorem

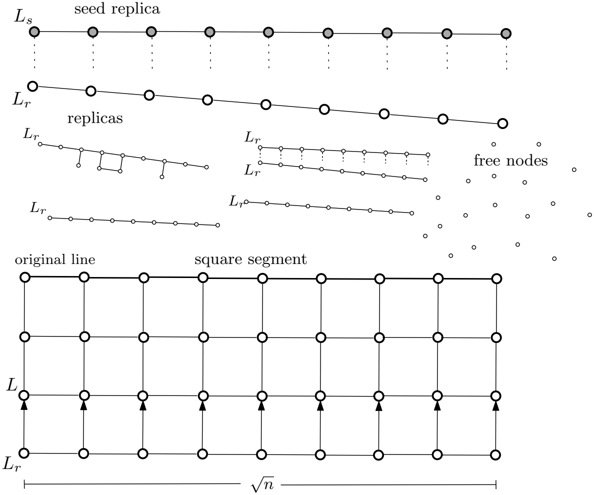
Let $\mathcal{L} = (S_1, S_2, \ldots)$ be a connected 2D shape language, such that $\mathcal{L}$ is TM-computable in space d^2. Then there is a protocol that w.h.p. constructs $\mathcal{L}$. In particular, for all $d \geq 1$, whenever the protocol is executed on a population of size $n = d^2$, w.h.p. it constructs S_d and terminates. In the worst case, when G_d (that is, the shape of S_d) is a line of length d, the waste is $(d-1)d = O(d^2) = O(n)$.

Counting-on-a-line Protocol

- Adapt Counting-Upper-Bound to work in our model
- The same probabilistic process
- The leader constructs a line that stores the two counters in binary
  - The line grows whenever more space is required

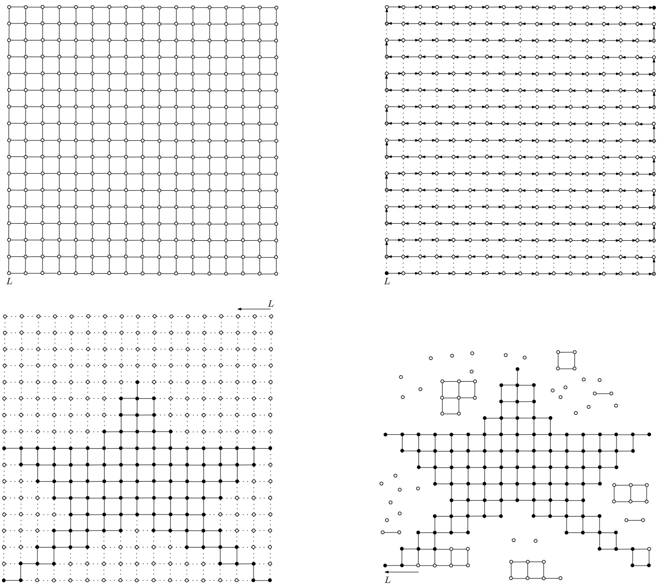
Lemma

Counting-on-a-Line protocol terminates in every execution. Moreover, when the leader terminates, w.h.p. it has formed an active line of length $\log n$ containing n in binary in the r_0 components of the nodes of the line (each node storing one bit).

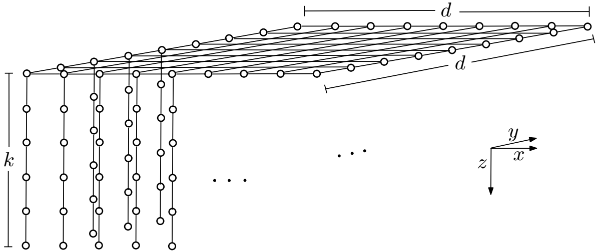


L_s
seed replica
L_r
replicas
L_r
L_r
L_r
L_r
L_r
free nodes
original line
square segment
L
L_r
$\sqrt{n}$

Simulating and Releasing



Parallelizing the Simulations



Theorem

Let $\mathcal{L} = (S_1, S_2, \ldots)$ be a TM-computable connected 2D shape language, such that S_d is computable in space $k = f(d)$ and k is computable in space $O(k \cdot d^2)$. Then there is a protocol that w.h.p. constructs $\mathcal{L}$. In particular, for all $d \geq 1$, whenever the protocol is executed on a population of size $n = k \cdot d^2$, w.h.p. it constructs S_d and terminates, by executing d^2 simulations in parallel each with space $O(k)$.

Open Problems

- Prove/disprove: Terminating counting w.h.p. is impossible if processes are identical
- Is there a faster counting protocol? (mainly, for the exact count)
- Distinction between the speed of the scheduler and the internal speed of a component
- Hybrid models of active and passive mobility
- Take other physical considerations into account
  - mass, strength of bonds, rigid and elastic structure, collisions
- Structures that optimize some global property or that achieve a behavior/functionality
- Develop protocols that efficiently reconstruct broken parts of the structure

Thank You!