Team Automata and Ordinal Register Machines

Ryan Bissell-Siders
University of Helsinki

2009 January 12
New Worlds of Computation
Orléans

team automata and infinitary computation

team automata (definition)

- ✔ Only change state when they collide or split;
- ✔ only collide in pairs.
- ✔ the pair of new states is a function of the old states.
- ✔ can split; this always occurs before any other collision.
- ✔ move in model M from $a \in M$ at time s to
  - ✘ time $s+1$ and the unique $b \in M$ such that $M \models \phi(a, b)$, or
  - ✘ times $t > s$ and locations $b \in M$ such that $\{(b, t) : M \models \phi((a, s), (b, t))\}$ is a bijection between times and locations.

team automata (definition).

Teams of automata exploring a graph and marking interesting locations are studied by, for instance, (P. Flocchini, D. Ilcinkas, N. Santoro, *Ping Pong in Dangerous Graphs: Optimal Black Hole Search with Pure Tokens*, 5218, LNCS, Springer, 227-241 (2008)).

team automata and ordinal register machines

An Ordinal Register Machine stores ordinals $\alpha_0 \dots \alpha_{19}$ in its registers and runs for ordinal time.
A universal ORM program exists, which reads an input n as a code and runs it. We simulate this universal ORM with a team of automata $A_0 \dots A_{19}$, because the ORM always passes over limit times with its registers in fixed state (either increasing linearly or stationary).
For instance, to compute $\lfloor \alpha_0/5 \rfloor$, we set α_1 to 0 and let it grow with speed 5 until it collides with α_0; the last value it takes will be the greatest multiple of 5 below α_0, and the elapsed time is $\lfloor \alpha_0/5 \rfloor$.
(Bissell-Siders and Koepke, 2006 CiE)

team automata and ordinal Turing machines

Given a team $A_0 \dots A_n$ of automata moving on the ordinals, a Turing machine with the ordinals as its tape and with a finite set of heads $H_0 \dots H_n$ simulates the team of automata by computing the movement of A_n and keeping H_n there; if A_n divides and leaves a marker, H_n markes the Turing tape.
The behavior of a Turing machien with the ordinals as its tape can be modeled by an Ordinal Reigster Machine.
(Bissell-Siders and Koepke, 2008 AML)

computations on Turing Machines and team automata

A Turing Machine rolling right

Given a Turing machine program P which can notice
L-the left end of the tape,
we divide its tape into $L = B \dots C, A \dots$, where C and A are adjacent cells. We create a program with tape:
$0 \dots 0 A \dots B \dots C$ which pushes the interval (B, C) further to the right whenever the interval $(A \dots)$ needs to add something to its right, and which frequently erases the value of the tape at A and adds it to the interveral (B, C).

A Turing Machine rolling right.

We write states as sentences of sem-natural language.
The program P is a set of tuples (state, character, where to go, what to write, what state to enter).
The new program PR has, for each command $(s_0, \chi_0, d, \chi_1, s_1) \in P$, $((s_0, 0), \chi_0, d, \chi_1, (s_1, 1)) \in PR$.
$(s_0, L, d, \chi_1, s_1) \in PR$ is replaced by $(s_0, A, d, \chi_1, s_1) \in PR$.
The program PR has, additionally, the following two sequences of commands
Beginning in state $(s_0, 1)$, read the character χ_0 into memory. Mark this location as D. find A, erase it, move right, read the character as χ_1, write A, move to C, print χ_1, move right, print C, find D, print χ_0, execute the rule $(s_0, \chi_0, d, \chi_1, s_1) \in P$ – print χ_1 and move in direction d and enter state $(s_1, 0)$.

A Turing Machine rolling right..

Beginning with state $(s_0, 1)$, read the character we see into memory as χ_0, print F, move right, check whether we read B. If not, move left, print χ_0, and excute the sequence above. If we read B, print D, move to C, move right, print B. Move right. Print E. (Find D, erase it, move right, read χ_1. print D Move to E. Print χ_1. Move right. Print E.) Repeat until the χ_1 we read is C. Erase it. Find E. Print χ_1. Move right. Print C. Find F. print χ_0 and execute the sequence above.
This program PR has the characteristic that it moves A every time the program P makes one move. PR moves the interval (B, c) whenever the program P desires to write on the n-th square, and $|(A, B)| = n$.

A Turing Machine rolling right...

If the Turing tape's first ω-many cells have been written into $(0,1) \subset \mathbb{R}$ by a program whose curves ϕ are constant, then the length of the tape's elements (n,ω) diminishes exponentially with n (it is p^n for some fraction n). The time taken to execute program PR is $\sum_{n\in\omega} t_n$, where t_n is the time taken to move A to the right (and perhaps move (B,C) to the right) and execute the n-th command in P. This requires that the program traverse subsets of the interval (A,ω) at most $2n+4$-many times (at most n elements in (B,c) to copy, which takes n passes left and right, then back to the active cell, then to A, to C, and back to the active cell). That would take time $t_n < p^n(2\times n+4)$.

Team Automata write an ordinal in $\mathbb{R}$

A program to write an infinite Turing Machine tape in a bounded interval of $\mathbb{R}$.
Speed is a total order on equivalence classes of states. We have finitely many states, so we name them $n\ A_i\ \sigma$, where n is the speed of the state and A_i is the automaton which is in this state, and σ is further information.
Let automaton A be

- ✔ state $1A0$. After collision with B enter state $1A1$.
- ✔ state $1A1$. Before any collision occurs, A writes a location with state $0L$ and enters state $2A0$.
- ✔ state $2A0$. After collision with B enter state $2A1$.
- ✔ state $2A1$. Before any collision occurs, A write a location with state $0L$ and enters state $1A0$.

Team Automata write an ordinal in $\mathbb{R}$.

Let automaton B have the same instructions (only, the labels 'A' are exchanged for labels 'B').
Let the initial state have automata A and B moving with different speeds so that they will pass each other once. From this start, they will infinitely often: pass each other, write a location, and pass each other again. The amount of time taken to eject the location determines how long it will be until they meet again. Depending on how these times are chosen, the wellordered set of locations which A and B write can be of any countable infinite order type.

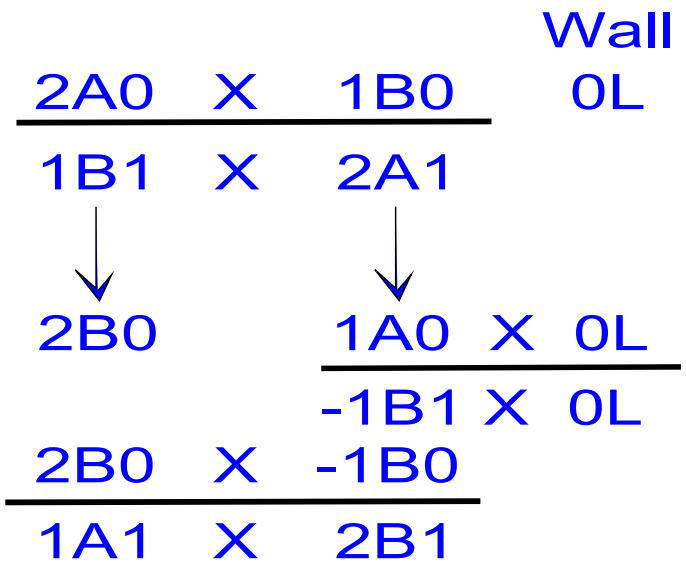
Team Automata write an ordinal in $\mathbb{R}$..

As in (Durand-Lose, *About the Universality of the Billiard ball model* in *Universal Machines and Computations (UMC 98)* edited by M. Margenstern, 2, Université de Metz, 118–133 (1998)) we consider whether this program can be realized physically. Ignoring the difference between state $2A1$ and state $2A0$, this is a realization:

Each automaton corresponds to a divisible particle. Each particle, when passed from the left, is catalzyed to split; the split leaves on fixed part, and one part which proceeds to the right with double velocity. If the initial velocities are different, but not double each other, then each particle will in turn catch the other, catalyze its split, and then be caught by the other.

Team Automata hit a wall

Consider the interaction of this team with a wall – a location L always in state $0L$ such that when automaton in state $bA_i\sigma$ collides with $0L$, the automaton enters state $(-b)A_i\sigma$. When the automata A and B interact, the state $(-b)A_i\sigma$ has the same output as $bA_i\sigma$.



Team Automata hit a wall.

Therefore, if team automata can write the ordinal α and it is desired to write a larger ordinal, we let each location in α split into a wall (for the automata below it) and two automata in states $1A1$ and $2B$. These automata will write between each element of α some infinite ordinal, depending on what delays occur between collisions and splittings, the result will be $\geq \omega \times \alpha$.
If some automaton can generate a sequence $\langle e_n \rangle$ of codes for team automata programs, such that e_i codes a program to write an ordinal $\geq \alpha_i$, then writing the sequence $e_0|e_1|\ldots$, where $|$ represents a wall and then interpreting the codes e_i, we generate the ordinal $\sum_{i<\omega} \alpha_i$.

team automata simulate a Turing machine in $\mathbb{R}$.

Run the preceding program. Given a Turing machine P, let P' be the automaton with a state $bP's$ for each $b \in \{-1, 1\}$ and state s of P; let locations have states corresponding to the Turing Machine's tape's symbols, and lets states collide with rules derived from the Turing Machine:

- ✔ If $(s_0, \chi_0, d, \chi_1, s_1) \in P$, then when P' and a location collide with P' in state $bP's_0$ and the location in state $0L\chi_0$, then P' enters state $dP's_1$, and the location enters state χ_1.

relationship to other hypercomputation systems

Abstract Geometric Computation simulate team automata

AGS, or "rational signal machines," almost meet the definition of team automata. It seems clear that splitting, and the rule that splits occur before any other event can replace the collision of three particles and the generation of new signals, and the existence of arbitrarily fast events can be modeled with fast AGS structures (?).

We take an idea from the following article, and produce $0'$ with team automata.

(Durand-Lose, *Reversible conservative rational abstract geometrical computation* in *Logical Approaches to Computational Barriers, 2nd Conf. Computability in Europe (CiE '06)* edited by Beckmann, Arnold and Tucker, John V., 3988, LNCS, Springer, 163–172 (2006))

A program to iterate the jump

If team automata can write the ordinal $\alpha \in \mathbb{R}$, then they can compute $0^{(\alpha)}$.
With each iteration, the class of symbols could grow. Don't let the limit be the union.
On the other hand, if a team of automata compute a finite answer to a question, then computations as in slide 15 can be performed by a machine using only a finite amount of memory. To search infinitely many possibilities to find whether something happens or not, then to use the existence of those witnesses in further computation, is exactly what infinite-time computers do.
Infinite-time machines storing a finite number of natural numbers compute exactly the hyperarithmetic hierarchy Δ^1_1.

The expressive power of geometric computation

This suggests that rational signal machines and teams of automata produce the hyperarithmetic hierarchy Δ^1_1.

questions

questions

- ✔ The inverse operation to splitting an automaton is to merge automata.
- ✔ Frame of reference. Surely it is all the same if all automata are moving to the right at a certain speed.
- ✔ What happens when two infinite families of automata, each family moving at a constant speed, collide?
- ✔ What happens to the automata A and B after they have collided infinitely often, and reach a wall? They cease to be part of my calculation. But what criterion should we use to declare automata to be exhausted?

a paradoxical answer

Suppose we use the criterion that two particles are both exhausted after an infinity of interactions.

- ✔ E.g., Achilles and the Tortoise run. Achilles constantly kicks a stone, which hits the tortoise's shell and stops. Achilles always kicks the stone before moving on. Therefore the stone shuttles between the runners infinitely often. In a physical world, just before Achilles reaches the tortoise, the vibration of the stone between Achilles' foot and the turtle's shell will transfer, in a finite number of bounces, enough energy to the Tortoise so that the Tortoise will pull ahead again.
- ✔ Suppose now that the stone is not a physical object, but an imaginary automaton about which Zeno is telling a story. So the automaton named "stone" shuttles infinitely often between Achilles and the Tortoise. So we put it to rest.

a paradoxical answer.

Suppose we use the criterion that two particles are both exhausted after an infinity of interactions.

- ✔ E.g., Achilles and the Tortoise run. Achilles constantly kicks a stone, which hits the tortoise's shell and stops. Achilles always kicks the stone before moving on. Therefore the stone shuttles between the runners infinitely often. In a physical world, just before Achilles reaches the tortoise, the vibration of the stone between Achilles' foot and the turtle's shell will transfer, in a finite number of bounces, enough energy to the Tortoise so that the Tortoise will pull ahead again.
- ✔ Achilles has collided infinitely often with the stone. Should we put Achilles to rest? Or let him pass the Tortoise and continue running?

questions.

- ✔ Can we state some conditions on the partial order of information dependence which are equivalent to the team of automata computing independently of their trajectories ϕ and of their infinitesmals (the distance traveled after colliding and before splitting)?

team automata on graphs.

These automata can settle on and label functional groups which are not localized in the graph, but which might be localized in space when the graph is realized as a molecule. For instance, a team of automata can find locations of joints in a protein string and, at roughly equal distances from those bends, groups of molecules which might function as a large functional group.
By considering hypercomputation systems, we have settled on a slightly different formulation of team automata than that which has been previously studied on graphs.
e.g. (Pierre Fraigniaud (CNRS LRI, Univ. Paris Sud, Orsay) *Graph Exploration and Graph Searching*, talk at Valparaiso, Jan 9-13, 2006) presents a series of proofs that a team of automata executing the same code will get trapped – they will either clump together and be trapped as a large automaton, or they will spread out and be trapped one by one.
But why should they execute the same code?