

Invited Keynote

Molecular computers for molecular robots as hybrid systems

Masami Hagiya, Shaoyu Wang, and Satoshi Kobayashi

Abstract

The project “Molecular Robotics” is currently developing two kinds of molecular robots: amoeba robots and slime mold robots. An amoeba robot is a vesicle enclosed by an artificial membrane made of lipid, called a liposome, and contains molecular devices including chemical reaction circuits on the surface of and inside the membrane. The body of a slime mold robot is made of a gel which can also work as an actuator of the robot driven by molecular devices in the gel. In this paper, we examine possible approaches to make such intelligent molecular controllers for amoeba and slime mold robots. In particular, we regard such controllers as hybrid systems because the environment, the robot and the controller are all state transition systems having discrete and continuous states and transitions. For modeling and designing hybrid systems, formal frameworks such as hybrid automata are commonly used. In this paper, we examine how molecular controllers are modeled as hybrid automata and how they can be realized in those molecular robots. As an example, we examine how a timed automaton can be realized as a molecular controller.

Introduction

If a single molecule or a group of molecules has its own functionality such as sensing or actuation, it is called a molecular device. Typical examples of molecular devices are membrane receptors and protein motors. In addition to those derived from living cells, various artificial molecular devices have been developed so far. Many of them are made of nucleic acid strands, i.e., DNA and RNA.

The next research step is to develop an integrated system from such molecular devices [Murata 2013]. A molecular system is said to be autonomous if it reacts to its environment by observing the environment, making a decision based on the observation and performing an appropriate action without explicit external control. An autonomous molecular system should therefore consist of sensors, computers and actuators, all made from molecular devices, with some energy source and a body enclosing those devices. We call such a molecular system a molecular robot. To repeat, a molecular robot autonomously reacts to its environment by observing the environment (and itself) by its sensors, making decisions by its computers, and performing actions to the environment. Molecular computers should therefore be intelligent controllers of molecular robots.

There are a number of research projects developing molecular robots or investigating technologies for making molecular robots. The project “Molecular Robotics” funded by MEXT, Japan is one of such research projects [Murata 2013, Hagiya2014+]. It is a five-year project beginning in (July) 2012 and

consists of about 50 investigators from many fields including chemistry, biophysics, computer science, control theory, and robotics. It is one of the Innovative Areas promoted by MEXT, which are expected to create a new (typically interdisciplinary) field of science.

One of the unique features of the project is to develop two kinds of prototypes of molecular robots. One is called “amoeba robot” and the other is called “slime mold robot”. An amoeba robot is a vesicle enclosed by an artificial membrane made of lipid, called a liposome, and contains molecular devices including chemical reaction circuits on the surface of and inside the membrane. Actuators of the robot (e.g., protein motors) are expected to change the shape of the liposome and eventually lead to locomotive actions. The robot may contain internal liposomes, which can explode and emit molecules in the body. The robot itself can eventually explode.



While the body of an amoeba robot is made of a liposome, that of a slime mold robot is made of a gel (typically a hydrogel), which can also work as an actuator of the robot because crosslinks in a gel can be made of DNA and such a gel can shrink or swell. Molecular devices are immobilized in the gel and DNA circuits can work inside the gel. The gel is expected to shrink or swell by the signals emitted from the devices. In relation to slime mold robots, Hagiya et al. have formulated a computational model called gellular automata, in which cellular space is created by walls of gels [Hagiya 2014]. Each cell is surrounded by gel walls and contains a solution in which chemical reactions take place. They may produce molecules called decomposers and composers, which dissolve or (re)construct a gel wall of the corresponding type.

One can also imagine combinations of amoeba and slime mold robots. For example, one can introduce internal liposomes into a solution surrounded by gel walls. If an internal liposome explodes, the molecules emitted from the liposome may dissolve a gel wall, or shrink or swell it. In the research project, we envision evolution of molecular robots as in the following figure, which is taken from [Murata 2013]. Beyond amoeba and slime mold robots and their combinations, we foresee next generations of molecular robots including multi-cellular robots and robots that are hybrids of molecular and electronic devices. Applications of such molecular robots are expected to include intelligent drug delivery (emitting drug only at the right place), artificial internal organ (sensing the blood sugar level and emitting insulin at the right time), intelligent stent (exploring the blood stream and extending blood vessel at the right place), contaminated soil cleaner, brain-machine interface, and eventually artificial brain.

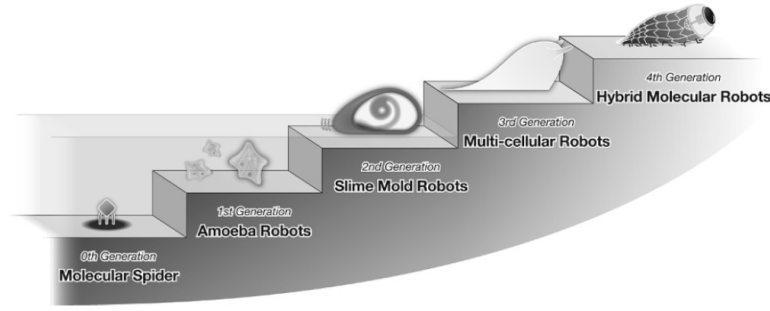


Fig. 1 Evolution of Molecular Robots

Both kinds of amoeba and slime mold robots should have a computer that processes information from sensors and controls actuators. Information processing in such a computer is naturally based on chemical reactions. In the above mentioned project, the research team called “Molecular Intelligence” is designing and implementing chemical reaction circuits that are components of molecular computers, which serve as intelligent controllers of molecular robots.

In this paper, we examine possible approaches to make such intelligent molecular controllers for amoeba and slime mold robots. In particular, we regard such controllers as hybrid systems because the environment, the robot and the controller are all state transition systems having both discrete and continuous aspects, i.e., they have both discrete and continuous states and make both discrete and continuous state transitions. For modeling and designing hybrid systems, formal frameworks such as hybrid automata are commonly used. In this paper, we examine how molecular controllers are modeled as hybrid automata and how they can be realized in molecular robots. As an example, we examine how a timed automaton can be realized as a molecular controller.

Requirements on molecular computers for molecular robots

In this section, we examine the requirements on a molecular computer that controls a molecular robot. Such a computer receives inputs from the sensors of the molecular robot and is expected to send orders to the actuators of the robot. It controls the system consisting of the robot and its environment to preserve certain conditions.

Reactiveness

Just as an electronic computer that controls a mechanical robot, a molecular computer that controls a molecular robot should respond to external signals or changes of the environment by sending orders to the actuators of the robot. In short, a molecular computer should be a real-time reactive system.

In particular, a molecular computer should handle changes of inputs from the environment. We say that a molecular computer is time-responsive if, when inputs to the computer change after its initial computation, outputs are re-computed to reflect the new inputs [Goel 2009, Chiniforooshan 2010]. Molecular computers for molecular robots should therefore be time-responsive.

Statefulness

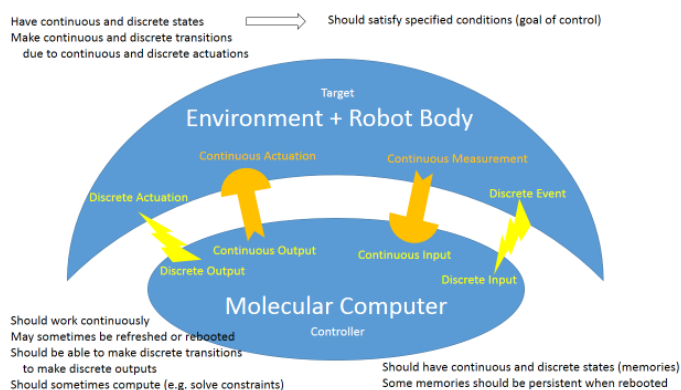
In general, outputs from a molecular computer should depend on the history of its inputs. It therefore should have states that store a part of the history that is necessary for computing outputs, and change them in response to new inputs. As mentioned below, there should be both discrete and continuous states.

Hybridness

Inputs from the sensors may be instantaneous signals from the external environment or may be continuous measurements of the environment. The former kind of input is a discrete event. Photo-irradiation for a short period of time is a typical example. Another example is pouring a solution to the environment which leads to a discrete change of concentrations of some molecular species in the environment.

A molecular robot may sometimes be required to invoke a discrete event. For example, in order to make a fast conformational change of an amoeba robot by protein motors, the concentration of ATP should change instantaneously to start the motors in a coordinated fashion. To invoke such discrete events, the computer should also be able to make discrete state transitions. Explosion of an internal liposome is a possible discrete state transition because it is a kind of phase transition and occurs instantaneously when a certain condition of the membrane is satisfied [Hamada 2010].

In short, the system consisting of the robot and its environment, including the computer controlling the robot, is a typical hybrid system. They have both discrete and continuous states and make both discrete and continuous state transitions. Discrete transitions may change the differential equations governing the continuous temporal evolution of the system, and continuous changes of the system may accumulate to cause discrete transitions. The computer controls the combination of the environment and the robot body (target of control) in terms of discrete and continuous inputs and outputs.



Persistency (reusability and rebootability)

Persistency is a prerequisite for reactivity and statefulness. However, most of the existing frameworks of DNA computing do not satisfy reusability, i.e., computational devices can be used only once. Therefore, molecular computers should have a continuous supply of new devices and other necessary resources such as energy. If such a supply is not easy to provide, entire molecular computers should be refreshed (or rebooted) from time to time so that used devices are replaced with new ones. In this case, how to implement states becomes crucial. Molecular computers should have persistent memories in the sense of persistent objects that are kept even after they are rebooted.

Examination of existing frameworks of DNA computing

DNA logic circuits such as enzyme-free circuits by Seelig, et al. and DNAzyme-based circuits by Stojanovic et al. are all stateless simply because they are combinatory circuits [Seelig 2006, Stojanovic 2003]. They consist of non-reusable gates, and therefore they are not time-responsive. Their inputs and outputs are supposed to be only discrete, i.e., 0 or 1. Logic gates implemented by seesaw gates by Qian et al. are also stateless, not time-responsive and only discrete [Qian 2011]. Hopfield networks implemented by seesaw gates are similarly not time-responsive because inputs cannot be changed after outputs have been computed for old inputs [Qian 2011+]. Note that seesaw gates themselves are reversible, but they are used as irreversible components in logic gates or neural networks.

As for circuits with states, Benenson, et al. implemented finite state automata using class IIS restriction enzymes [Benenson 2001]. States of the automata are only discrete and the number of states is restricted. They are not reactive in the strict sense because inputs to the automata are not in the form of external signals. The whiplash machines proposed by Hagiya, et al. employ hairpin conformations as states [Hagiya 1999]. They can reactively make state transitions if they receive displacing strands as inputs [Rose 2006, Komiya 2010]. However, state transitions are stochastic and next states are not unique in general. Note also that these machines are implemented by single molecules, so they can only work in a very small compartment as a state machine.

Although reusable gates and time-responsive circuits were proposed in some previous studies, they had problems in the implementation aspects such as scalability, time efficiency and need of enzymes [Goel 2009, Chiniforooshan 2010, Genot 2011]. Actual implementation is only reported by Genot et al. [Genot 2011]. These circuits are time-responsive but supposed to be only discrete. In order to change an input from 1 to 0, it is necessary to put a strand complementary to the old input.

In principle, arbitrary chemical reaction networks can be implemented by strand displacement reactions of DNA [Soloveichik 2010]. It is therefore possible to implement time-responsive circuits using DNA though some strands are consumed in the course of computation and should be supplied continuously. One can also use DNA Toolbox to implement time-responsive circuits though they are always dissipative [Montagne 2010]. Note that with large molecular counts, concentrations of molecular

species in a chemical reaction network evolve continuously. So, discrete transitions should be implemented by relatively fast reactions, and discrete values should be realized in terms of appropriate representations.

In the framework of analog computation proposed by Kobayashi, et al., analog amplifier, adder, multiplier, and divider can be implemented by simple reversible strand displacement reactions like seesaw gates [Kobayashi 2014]. Computation is defined as establishment of a function $y(t) = f(x(t))$ from $x(t)$ to $y(t)$, where $x(t)$ and $y(t)$ are concentrations of input and output molecular species at time t when the reaction system is in a steady state. The input concentration $x(t)$, which is the concentration of a certain molecular species in the environment, is assumed to be constant during the computation, i.e., it is assumed that the environment of a molecular robot does not change so fast. Proposed analog computing devices are time-responsive and can be used for a long time. Therefore, they can be used as components in a hybrid molecular controller.

Hybrid systems

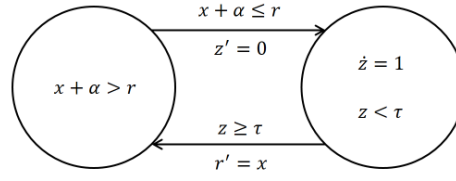
State transition systems are called hybrid systems if they have both discrete and continuous states and can make both discrete and continuous state transitions. Many formal models of hybrid systems have been proposed. Hybrid automata are one of the most popular models [Henzinger 1996]. A hybrid automaton consists of the following components.

- Real-valued variables
- Discrete states, called control modes, each having invariant conditions (e.g., $x \geq 18$) and flow conditions (e.g., $\dot{x} = -0.1x$).
- Discrete transitions, called jumps, each having event labels and jump conditions (e.g., $x < 19$) and updating the values of variables (e.g., $z' = 0$).

Based on the framework of hybrid automata, a controller and its target are both modeled as hybrid automata. The entire system is then modeled as the parallel composition of those automata, which is also a hybrid automaton. If one automaton makes a jump with a shared event label, then the other should also make a jump with the same label. One automaton can therefore send a message, which the other synchronously receives. One can make an internal discrete transition as a jump with a non-shared event label while the other makes no transition. Implicit communication between the target and the controller is also possible via a shared real-valued variable. One changes the shared variable and the other observes it.

Timed automata are restricted hybrid automata [Alur 1994]. All real-valued variables, called clocks, increase with the same speed. Jumps can only reset real-valued variables to zero as in $z := 0$. Jump conditions are made of comparisons with a threshold of the form $z \geq \tau$.

Let us give a concrete example motivated by the work of Azuma et al., who modeled chemotaxis controllers of bacteria [Azuma 2013].



It is assumed that in the left mode of the above figure, the bacterium makes forward motion, and in the right mode, it randomly changes its direction. The variable x is an input from the environment, which tracks the concentration of some nutritious substance, and r is a local variable holding the value of x . When x becomes smaller than $r - \alpha$, the current mode is changed from left to right, and the clock z is reset to zero. The current mode then changes from right to left when z expires, i.e., it reaches its wakeup time τ .

Molecular controller as hybrid automaton

In this section, we examine each component of a molecular controller of as a hybrid system.

Real-valued input from the environment

The concentration of a molecular species shared by the environment and the molecular robot is considered as a real-valued input from the environment, and is observed by a sensor of the robot and referred to by its controller. For example, a membrane channel of a liposome selectively passes the molecular species, or a gel wall allows it to freely diffuse. Compared with the molecular robot, the environment can be regarded as a reservoir of the molecular species, whose concentration is changed by the environment but is clamped at a fixed level for some duration of time.

Real-valued variable local to the controller

In addition to the molecular species shared by the environment, the molecular controller can have its own molecular species whose concentrations continuously evolve according the chemical reactions that are local to the controller. Therefore, the concentration of such a molecular species is considered as a real-valued local variable of the controller. The total concentration (or combination) of different species can also be used. Such a local variable is typically used as a clock or as a record (memory) of a past input from the environment to the robot.

A physical parameter of a macroscopic structure in the robot is also considered as a real-valued local variable. Examples are the stability of a lipid membrane (which is usually determined by the concentration of a certain molecule on the membrane) and the width of a gel wall.

Boolean (discrete) local variable representing a mode of the controller

Some local variables are Boolean in the sense that they have only two values: a non-zero value and zero,

or high and low values. Such a variable typically corresponds to each mode. If the controller is in a mode, its corresponding variable is non-zero and variables corresponding to other modes are zero. Those variables are also naturally represented by a concentration of a molecular species in the controller.

Existence or non-existence of a macroscopic structure can also be used as a Boolean variable. Examples are open/close of a liposome and existence/non-existence of (a hole in) a gel wall. When a liposome explodes, the controller usually enters a new mode.

The conformation of a single molecule is also considered as a Boolean or discrete variable. In a whiplash machine, for example, a single-stranded DNA molecule can take one of several hairpin forms. Each form is considered as a discrete state of the molecule and can emit different DNA molecules by combination of a polymerase and a nicking enzyme.

Flow

Chemical reactions in the environment and the controller define how real-valued inputs and local variables continuously evolve in time. In particular, chemical reactions in the controller increase the value of a clock, copy an input to a local variable, etc. In a hybrid system, flow transitions are defined for each mode of the system. Therefore, chemical reactions in the controller should be constrained by the current mode of the controller.

In addition to concentrations of molecular species, physical parameters of macroscopic structures also evolve continuously. Such flow transitions may also be constrained by the current mode of the controller.

Jump by an external signal

In a parallel composition of hybrid automata, a shared transition label allows a synchronous jump transition in the component automata. In the case of molecular systems, such transitions can be made by external signals. For example, photo-irradiation may instantaneously change the conformation of a specific molecular group such as azobenzene. If the molecular group is shared by the environment and the robot, photo-irradiation causes a synchronous jump transition.

Addition of a solution such as NaOH to the entire system is another external signal. It may instantaneously change the concentration of a molecular species and results in a discrete change of a certain solution condition (such as pH). Some external signals can be given in terms of construction or destruction a macroscopic structure. For example, if a hole is made in a gel wall, some molecules flow into the robot.

Jump by the controller

The controller should also be able to trigger jump transitions autonomously. Firstly, it should be able to change values of its local variables. Examples of such jump transitions are resetting a local variable (e.g.,

a clock) to zero, setting a local variable to a constant value (e.g., to change the current mode), and copying an input to a local variable (e.g., to record the current value of the input).

The controller should also be able to start new flows (e.g., start a clock) by a jump transition. Finally, the controller should be able to perform actions, and as discussed in a previous section, some actions can or should be performed by jump transitions. Examples are emitting molecules outside, importing molecules from outside, changing a macroscopic shape (e.g., opening or closing a valve, and turning on a switch), and supplying energy and reactants to refresh or reboot the controller.

Realization of a jump by the controller

How can such jumps mentioned above be realized? The following three methods can be considered. One is to make use of ultrafast chemical reactions in the controller, which lead to a steady state quickly.

The second method is in terms of a macroscopic structural change. For example, a liposome explodes and enclosed molecules are emitted. This is considered as a jump transition because emission and diffusion of molecules are instantaneous compared with other reactions. Consequently, concentrations of some molecular species are increased instantaneously.

The third is to employ a single-molecular conformational change. This is a stochastic but discrete event. An example is a conformational change of a hairpin DNA molecule which takes another hairpin form.

Jump condition

Referring to inputs and local variables, the controller makes decisions for triggering jump transitions and changing its current mode. There are various kinds of conditions for the controller to make decisions. Typical examples are comparisons on inputs such as comparisons between an input and a threshold, between two inputs, and between an input and a local variable. Other examples are comparisons on local variables such as comparisons between a local variable and a threshold. More concretely, the controller should be able to check when a clock reaches its wakeup time. All such comparisons may be constrained by the current mode of the controller. In general, more complex conditions combining comparisons with arithmetical and Boolean operators may be necessary. They require analog computation [Kobayashi 2014].

Jump condition to a jump

How to trigger a jump transition according to a jump condition seems to be one of the most difficult challenges for making a molecular controller. There are several possible approaches. One approach is to make a jump condition equivalent to the condition for an internal liposome to explode, i.e., when the jump condition is satisfied, the concentration of a certain molecular species, which makes the membrane of the liposome unstable, exceeds its threshold to explode the liposome. Let us call the molecular species

the destructor of the liposome.

Another approach is to dissolve a gel wall when a jump condition is satisfied. In this case, the jump condition $x = 0$ is realized if x is interpreted as the width of a gel wall. One can then define x as the difference between a clock and its wakeup time.

One can also prepare appropriate (ultrafast) reactions so that when a jump condition is satisfied, they lead to a steady state quickly.

Finally, a jump condition may promote a conformational change of a single molecule. For example, a hairpin form can be denatured by strand displacement. Note that this kind of event is stochastic, so only the probability to make a jump transition can be changed.

Let us now examine the above mentioned approaches in the following more concrete examples.

Examples

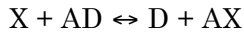
In this section, we examine some concrete examples of jump conditions and transitions, and see how a hybrid molecular controller can be actually constructed. We mainly consider the possibility of employing internal liposomes. Gel walls can also be used though they are less convenient.

Comparison on a real-valued input

We first examine comparisons that involve a real-valued input. Consider the following jump condition.

If the concentration of a molecular species X is greater than the specified threshold θ , then explode an internal liposome.

It is assumed that the molecular species X is shared by the robot and the environment, and its concentration is determined and clamped by the environment, i.e., it is a real-valued input from the environment. It is also assumed that the liposome explodes if the concentration of its destructor D exceeds the threshold η . We prepare the following reaction [Qian 2011, Kobayashi 2014].



Assume that AD and AX are abundant compared with X , i.e., their concentrations are much larger than that of X . We can then achieve the desired condition by adjusting their concentrations to make $\theta [AD] = \eta [AX]$ true, where the equilibrium constant of the reaction is assumed to be 1. Note that in this implementation time responsiveness is achieved, i.e., even if the input is changed by the environment, the jump is triggered as soon as the condition is satisfied.

Next, consider the following jump condition that is constrained by the current mode.

If the concentration of a molecular species X is greater than the specified threshold θ in a mode m , then explode an internal liposome.

The mode m is assumed represented by a non-zero concentration of a molecular species M . We prepare the following reaction.



Here, we can use a seesaw gate by Qian et al. The molecule M works as a catalyst, and if M is present, the reaction is active and leads to a steady state. We can adjust the concentrations of AD and AX as above. When the mode m is left, the concentration of M is supposed to become zero, and it may be necessary to decompose D at this point.

One can also use the multiplier of Kobayashi, et al., and make the concentration of D proportional to the product of the concentrations of X and M [Kobayashi 2014]. Although it is necessary to set the non-zero concentration of M to a fixed level, the concentration of D automatically becomes zero when m is left and the concentration of M is reset to zero.

One may also want to compare an input with a local variable. For example, consider the following jump condition, which appeared in a previous section.

If the concentration of a molecular species X in the environment is greater than $r + \alpha$, where r is a recorded value and α is a positive constant, then explode an internal liposome.

We first prepare the following reaction to transform X to Y.



We assume that fr is recorded as the initial concentration of the molecular species R, where f is an adjustable factor, and prepare the following reaction whose equilibrium constant is K .



When the concentration of D becomes η , $K(fr - \eta) = \eta[Y] = \eta([AY]/[AX])[X]$ holds. Therefore, we can achieve the condition by adjusting $[AX]$, $[AY]$ and f .

Checking a clock

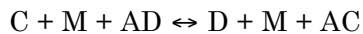
Here, we examine comparisons between a clock and its wakeup time. Consider the following jump condition.

When a clock reaches the specified wakeup time τ in a mode m, explode an internal liposome.

We assume that the molecular species C is generated at a constant speed, i.e., we prepare the following reaction in which C is generated constantly at some rate k .



For example, we can use combination of a polymerase and a nicking enzyme. We also prepare the following reaction.



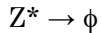
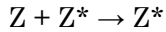
The molecular species M represents the mode m and D is the destructor of the liposome. If C is generated at k and the threshold of D is η , we can achieve the desired condition by adjusting the concentrations of $[AC]$ and $[AD]$ to make $(k\tau - \eta)[AD] = \eta[AC]$ true.

Manipulation of a local variable

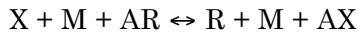
Upon a jump transition, one may want to change values of local variables. For example, when the

current mode is left, the variable representing the mode should be reset. In a timed automaton, some clocks should also be reset to zero.

When an internal liposome explodes, molecules are emitted from inside the liposome to the solution of the controller. They can change values of local variables. For example, assuming that a local variable Z is represented by the concentration of the molecular species Z , we can reset Z to zero by emitting a molecular species Z^* that decomposes Z . We should further assume that Z^* is also eventually decomposed as in the following reactions.



Another example is to copy the value of an input from the environment to a local variable. We prepare the following reaction, where X is a molecular species shared with the environment and R is a molecular species in the controller.



We assume that AR and AX are abundant and have the same concentration (or we can multiply the input by the factor $f = [AR]/[AX]$). Before copy, the current mode is changed to m by increasing the concentration of the molecular species M to non-zero. At the time of copy, the concentration of M is reset to zero, i.e., the mode m is left. The value of the input is kept as the concentration of R .

Can a timed automaton be implemented?

Given the above examples, we can raise the question: can a timed automaton be implemented? In a timed automaton, jumps are restricted to the following form, which is still more restricted than usual.

If $z \geq \tau$ then change the mode from m to n and reset $y := 0$

The modes m and n are assumed represented by the concentrations of molecular species M and N . We can prepare an internal liposome so that molecules emitted from the liposome reset the concentration of M to zero. On the other hand, molecules of the species N are also emitted and its concentration becomes non-zero.

The most challenging problem is how to repeat jump transitions since each internal liposome can only explode once. A possible solution is to prepare many liposomes of the same contents but with slightly different sizes. They are expected to explode one by one. However, the number of such liposomes is still finite, so it may be necessary to supply internal liposomes to the robot by some means.

Concluding Remarks

In this paper, we introduced the research project “Molecular Robotics” and briefly explained two kinds of molecular robots under development, i.e., amoeba robots and slime mold robots. We then summarized the requirements on molecular computers for those molecular robots by pointing out that molecular computers should be hybrid controllers, and examined existing frameworks of DNA computing. Finally,

we discussed how to implement each component of a hybrid automaton in terms of chemical reactions and internal liposomes, and raised a question on whether a timed automaton can be implemented.

Let us finally emphasize that, just as other kinds of machines, the framework of hybrid systems is appropriate for designing and verifying molecular robots and their controllers. It is therefore important to develop general (hopefully universal) methods for implementing hybrid systems in terms of chemical reactions and physical phenomena that can be implemented in molecular robots. In order to achieve the goal, it will be necessary to combine various approaches to develop molecular robots, including the present one employing internal liposomes and other approaches such as those of whiplash machines, dynamical systems and strand displacement reactions [Hagiya 1999, Montagne 2010, Soloveichik 2010].

References

- [Murata 2013] Murata, S., Konagaya, A., Kobayashi, S., Saito, H., and Hagiya, M., Molecular robotics: a new paradigm for artifacts, *New Generation Computing*, 31:27–45, 2013.
- [Hagiya 2014+] Hagiya, M., Konagaya, A., Kobayashi, S., Saito, H., and Murata, S., Molecular robots with sensors and intelligence, *Accounts of Chemical Research*, ACS, to appear.
- [Hagiya 2014] Hagiya, M., Wang, S., Kawamata, I., Murata, S., Isokawa, T., Peper, F., and Imai, K., On DNA-based gellular automata, O.H. Ibarra et al. (Eds.): *UCNC 2014*, LNCS 8553:177–189, 2014.
- [Hamada 2010] Hamada, T., Sugimoto, R., Vestergaard, M.C., Nagasaki, T., and Takagi, M., Membrane Disk and Sphere: Controllable Mesoscopic Structures for the Capture and Release of a Targeted Object, *J. Am. Chem. Soc.* 132:10528–10532, 2010.
- [Goel 2009] Goel, A. and Ibrahimi, M., Renewable, time-responsive DNA logic gates for scalable digital circuits, R. Deaton and A. Suyama (Eds.): *DNA 15*, LNCS 5877:67–77, 2009.
- [Chiniforooshan 2011] Chiniforooshan, E., Doty, D., Kari, L., and Seki, S., Scalable, time-responsive, digital, energy-efficient molecular circuits using DNA strand displacement, Y. Sakakibara and Y. Mi (Eds.): *DNA 16*, LNCS 6518:25–36, 2011.
- [Seelig 2006] Seelig, G., Soloveichik, D., Zhang, D.Y., and Winfree, E., Enzyme-Free Nucleic Acid Logic Circuits, *Science* 314(5805):1585–1588, 2006.
- [Stojanovic 2003] Stojanovic M.N., and Stefanovic, D., A Deoxyribozyme based Molecular Automaton. *Nature Biotechnology*, 21(9):1069–1075, 2003.
- [Qian 2011] Qian, L., and Winfree, E., Scaling Up Digital Circuit Computation with DNA Strand Displacement Cascades, *Science* 332(6034):1196–1201, 2011.
- [Qian 2011+] Qian, L., Winfree, E., Bruck, J., Neural network computation with DNA strand displacement cascades, *Nature* 475:368–372, 2011.
- [Benenson 2001] Benenson, Y., Paz-Elizur, T., Adar, R., Keinan, E., Livneh, Z., and Shapiro E., Programmable and autonomous computing machine made of biomolecules, *Nature* 414:430–434, 2001.
- [Hagiya 1999] Hagiya, M., Arita, M., Kiga, D., Sakamoto, K. and Yokoyama, S., Towards Parallel

- Evaluation and Learning of Boolean μ -Formulas with Molecules. DNA Based Computers III, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 48:57-72, 1999.
- [Rose 2006] Rose, J.A., Komiya K., Yaegashi S., and Hagiya M., Displacement Whiplash PCR: optimized architecture and experimental validation. Lecture Notes in Computer Science, 4287:393–403, 2006.
- [Komiya 2010] Komiya, K., Yamamura, M., and Rose, J.A., Experimental validation and optimization of signal dependent operation in whiplash PCR, Natural Computing 9:207–218, 2010
- [Genot 2011] A. J. Genot , J. Bath , and A. J. Turberfield, Reversible Logic Circuits Made of DNA. J. Am. Chem. Soc., 133(50):20080–20083, 2011.
- [Soloveichik 2010] Soloveichik, D., Seelig, G., and Winfree, E., DNA as a universal substrate for chemical kinetics, PNAS 107(12):5393–5398, 2010.
- [Montagne 2010] Montagne, K., Plasson, R., Sakai, Y., Fujii, T., and Rondelez, Y., Programming an in vitro DNA oscillator using a molecular networking strategy, Molecular Systems Biology 7:466, 2010.
- [Kobayashi 2014] Kobayashi, S., Yanagibashi, K., Komiya, K., and Fujimoto, K., Hagiya, M., Analog DNA computing devices for the control of molecular robots, Proc. of Workshop on Self-organization in Swarm of Robots: from Molecular Robots to Mobile Agents, 2014, to appear.
- [Henzinger 1996] Henzinger, T.A., The theory of hybrid automata, IEEE LICS 11: 278–292, 1996.
- [Alur 1994] Alur, R., and Dill, D.L., A theory of timed automata, Theoretical Computer Science 126(2): 183–235, 1994.
- [Azuma 2013] Azuma, S., Owaki, K., Shinohara, N., and Sugie, T., Performance Analysis of Chemotaxis Controllers, IEEE Conference on Decision and Control 52:1411–1416, 2013.