

Autonomic Computer Systems

CS321: Bio-Inspired Autonomic Communication, Part I

February 6, 2007

Lidia Yamamoto
lidia.yamamoto@unibas.ch
Computer Science Department
University of Basel

Overview Part I

- Biologically Inspired Computing
 - Overview
 - Case study: Ants algorithms
 - * Ant System applied to TSP (Traveling Salesman Problem)
 - * AntHotNet: Ants for Ad Hoc Network Routing
- Chemical Computing (Part I)
 - Overview
 - Fraglets

Overview Part II

- Chemical Computing (Part II)
 - Resilience to Instruction Loss
 - Genetic Programming Experiments
- The BIONETS Project
 - Overview, architecture, current state
 - Research in Basel
 - * Service and Protocol Evolution
 - * Evolving Chemical Programs
 - * Code Regulation

Introduction

Why seeking inspiration from biology?

- Look for new solutions to difficult problems
- Living organisms are complex adaptive systems: artificial systems are going in that direction too
- Life has many self-* features which are also desirable in artificial systems: self-organization, reproduction, self-healing ability, self-optimization, robustness,...

Biologically Inspired Computing

The typical problems that can be tackled with bio-inspired solutions are characterized by:

- Absence of a complete mathematical model
- Large number of (inter-dependent) variables
- Non-linearity
- Combinatorial or extremely vast solution space

Biologically Inspired Computing

Bio-inspired computing as old as computing itself

- Alan Turing's work on reaction-diffusion models for pattern formation [12]

History of bio-inspiration intrinsically related to that of AI:

- first models of human brain (still too complex for us to understand and extract a model, limited success)
- nouvelle AI: more "modest": behaviour-based AI, look at simple beings such as bacteria, insects

Recently boosted by advances in biology: systems biology, genetics, molecular biology...

Biologically Inspired Computing

Bio-inspired computing can cover most diverse aspects, such as:

- Artificial Life
- Models of social human organization
- Artificial (Gene) Regulatory Networks
- Autonomic Computing and Communication (autonomic nervous system)
- Models of nervous cognition

Biologically Inspired Algorithms

Some examples of bio-inspired algorithms:

- **ACO (Ants Colony Optimization)** (today's lecture)
- ANN (Artificial Neural Networks)
- Evolutionary Computing (Genetic Algorithms, Genetic Programming)

Some bio-inspired distributed algorithms:

- Swarm Intelligence
 - **Ants Routing** (today's lecture)
- Cellular Automata
- AIS (Artificial Immune Systems)
- Epidemic dissemination algorithms
- Many others:
 - Firefly synchronization
 - Chemotaxis-inspired load balancing
 - ...

Swarm Intelligence

- Algorithms inspired by the behavior of *swarms*
- Simple individuals self-organize without central control
- Use self-organization to achieve cooperative behaviour for collectively solving a given problem
 - *Emergent* behaviour
- Examples: flocks of birds or fish, bee hives, ants colonies...

ACO: Ants Colony Optimization [1, 9]

- A class of *Swarm Intelligence* algorithms inspired by the behaviour of ants colonies.
- *Stigmergy*: indirect communication, e.g. via signals released to the environment.
- Ants leave *pheromone trails* that are sensed and followed by other ants to indicate the path towards a food source.
- Pheromones evaporate in time.

Ants and Obstacles [8]

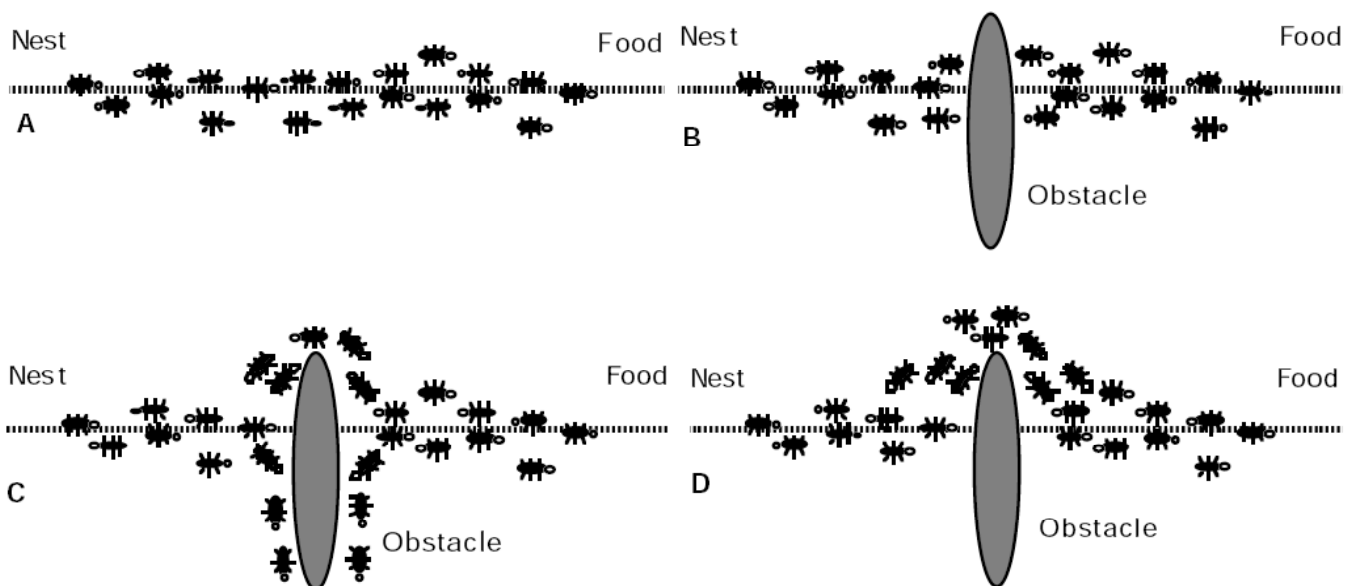


Fig.1. (A) Real ants follow a path between nest and food source. (B) An obstacle appears on the path: Ants choose whether to turn left or right with equal probability. (C) Pheromone is deposited more quickly on the shorter path. (D) All ants have chosen the shorter path.

Ants Double Bridge Experiment

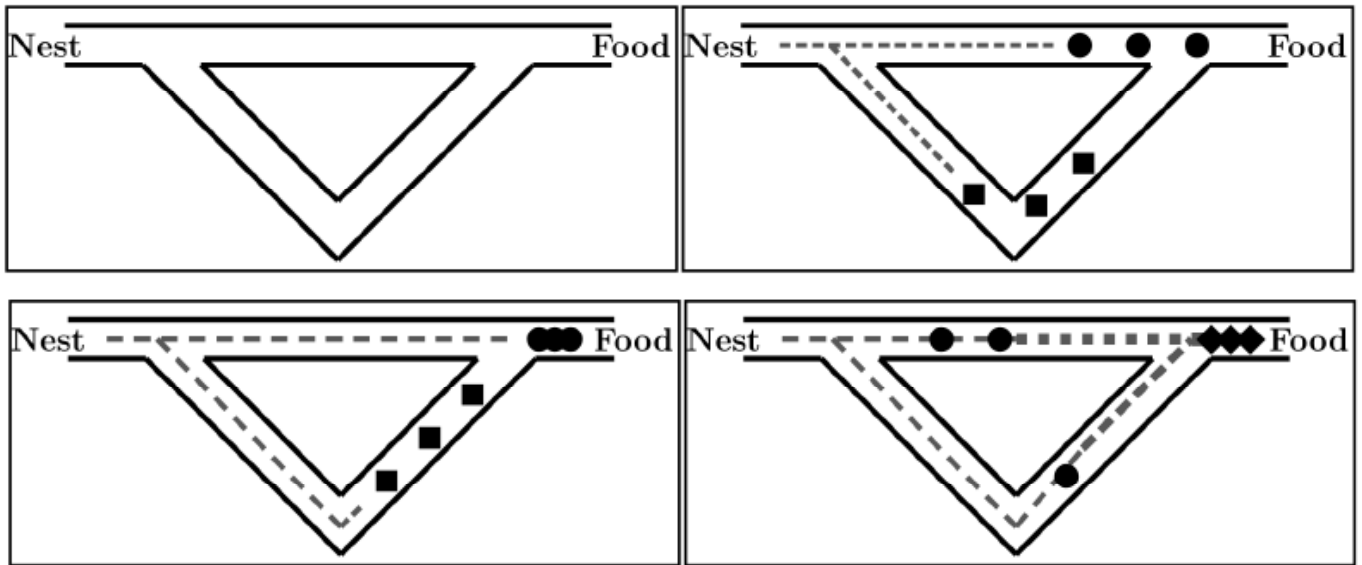
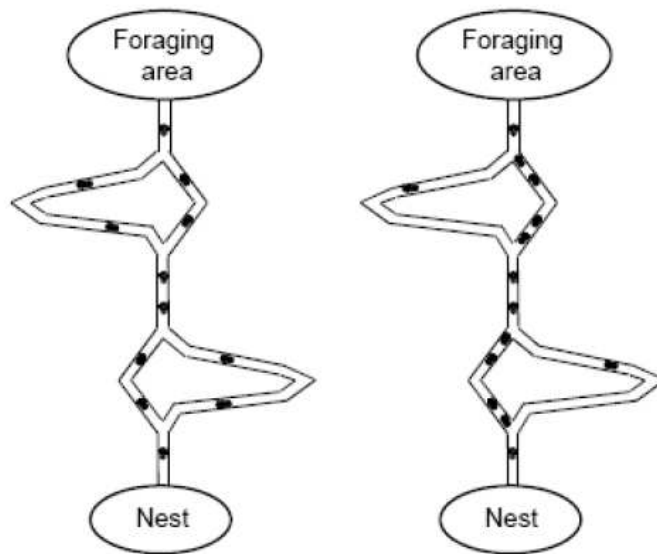


figure from [5]

Ants Double Bridge Experiment

- *Ant foraging* behavior:
 - Ants tend to follow the strongest pheromone trail
 - Ants deposit more pheromone as they go
- Result: positive reinforcement of strongest path
- Shorter versus longer path
 - foraging favours shortest path: shortest path is completed earlier, therefore receives pheromone reinforcement quicker.

Ants and the Shortest Path



asymmetric bridge experiment [6]

ACO: Ants Colony Optimization

ACO is a *Metaheuristic*:

- It defines a class of optimization heuristics for solving difficult combinatorial problems (typically NP-hard).
- It can be used in various problem domains.
- Typical example: Traveling Salesman Problem (TSP)
- Other applications: scheduling, vehicle routing, graph coloring, constraint satisfaction problems, data mining, circuit design, multiobjective optimization, bioinformatics, telecommunication.

- **Ant System** (first algorithm, by Marco Dorigo, 1992)
(today's lecture)
- Max-Min Ant System
- Ant Colony System (ACS)

Ant System applied to TSP [5]

- **TSP = Traveling Salesman Problem (NP-Hard)**
 - A set of cities with roads between them.
 - Salesman must visit each city only once with minimum cost or travel time, i.e. must find a *Hamiltonian tour* of minimum length.
- ACO formulation of TSP:
 - $G = (V, E)$
 - V = set of vertices i representing cities
 - E = set of edges i, j representing roads with distances $d_{i,j}$
 - $\tau_{i,j}$ = pheromone value for edge i, j

Ant System applied to TSP

Algorithm (iterated until stop condition or time limit reached):

- ants colony: set of n artificial ants
- each ant starts at a random node, then walks to an unvisited node using an edge, until it has visited all nodes; then goes back to original node closing the tour.
- each ant keeps a list S of visited nodes.
- from current node i , an ant chooses the next node j with probability $p(i, j)$:

$$p(i, j) = \frac{\tau_{i,j}}{\sum_{(k \notin S)} \tau_{i,k}}$$

Ant System applied to TSP

Algorithm (cont.):

- pheromone evaporates in all links after all ants have completed their tour:

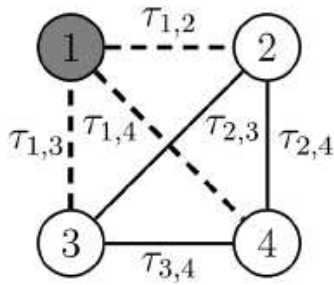
$$\tau_{i,j} = (1 - \rho) \cdot \tau_{i,j}, \quad \forall (i, j) \in E, \quad 0 < \rho < 1$$

- after that, each ant starts its return trip, depositing pheromone along the path S according to the cost function $f(s)$:

$$f(s) = \sum_{(i,j) \in S} d_{i,j}$$

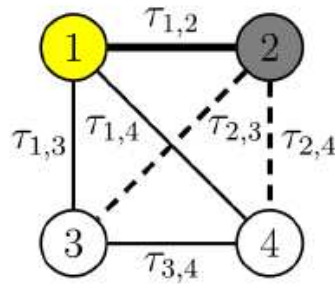
$$\tau_{i,j} = \tau_{i,j} + \frac{Q}{f(s)}, \quad Q > 0$$

Ant System applied to TSP



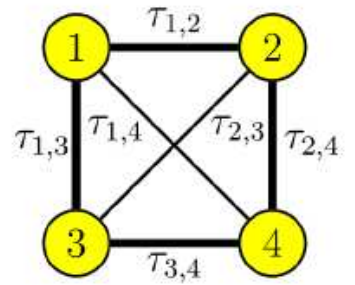
$$p(e_{1,j}) = \frac{\tau_{1,j}}{\tau_{1,2} + \tau_{1,3} + \tau_{1,4}}$$

(a) First step of the solution construction.



$$p(e_{2,j}) = \frac{\tau_{2,j}}{\tau_{2,3} + \tau_{2,4}}$$

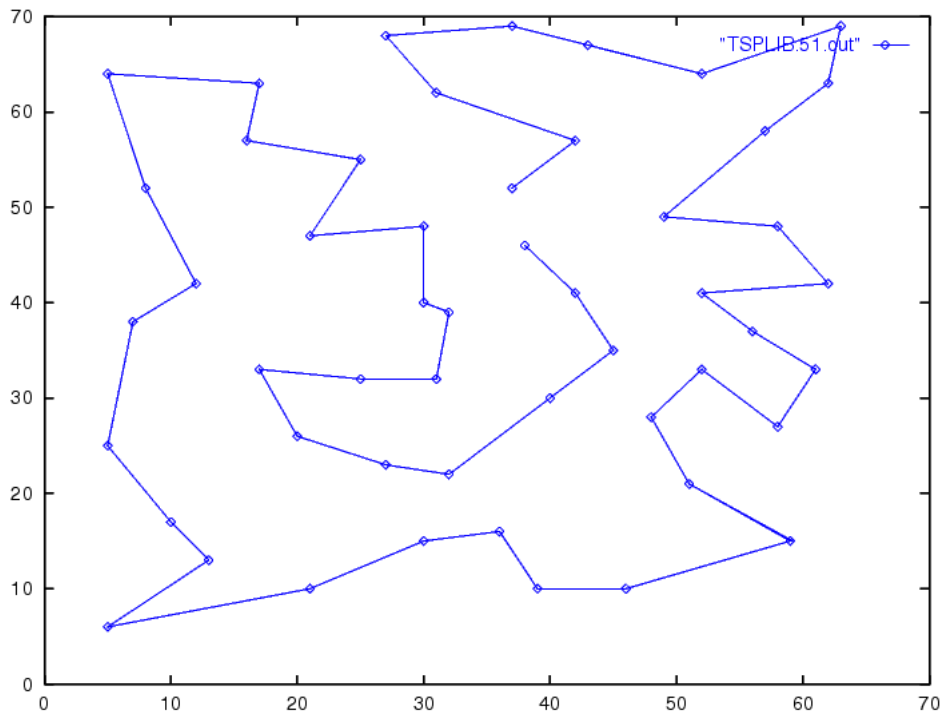
(b) Second step of the solution construction.



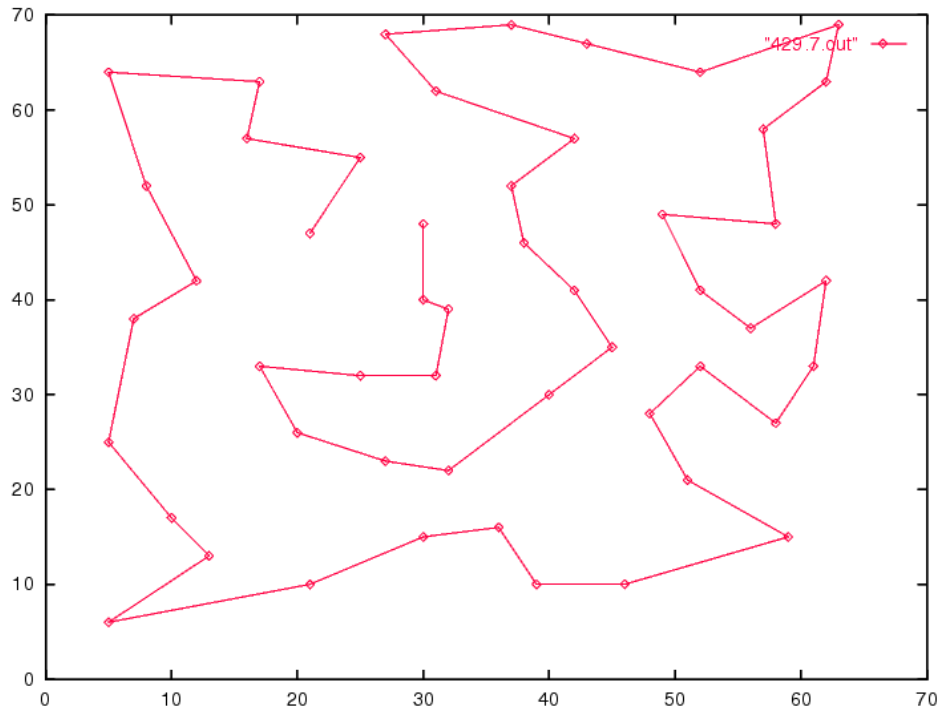
(c) The complete solution after the final construction step.

(example from [5])

Example: Optimum Tour



Example: Tour Found by Ants Algorithm



AntHocNet [7]: Adaptive Routing in Ad Hoc Networks

- Hybrid reactive and proactive multipath routing
- Reactive path setup phase: path discovered when needed
 - *Reactive forward ants* launched by source: find multiple paths to destination
 - *Backward ants* set up paths
- *Pheromone table* indicate path quality
- Data packets routed *stochastically* over different paths using pheromone table
- During session, paths are probed, maintained and improved proactively using *proactive forward ants*

AntHocNet: Reactive Path Setup

- source $s \rightarrow$ destination d
- s broadcasts *reactive forward ant* F_d^s
- node i : pheromone table T^i with entries T_{nd}^i : indicate how good is the path $i \rightarrow n \rightarrow \dots \rightarrow d$ where n is a neighbor of i .
- if T^i contains no entries for d , then F_d^s is broadcast
- else: F_d^s chooses next hop n with probability P_{nd} :

$$P_{nd} = \frac{T_{nd}^i}{\sum_{j \in N_{jd}^i} T_{jd}^i}$$

N_{jd}^i = set of neighbors of i over which a path to d is known

AntHocNet: Reactive Path Setup

- F_d^s keeps a list P of nodes it has visited.
- upon arrival at d , it is converted into a *backward ant* which travels back to s retracing P .

AntHocNet: Reactive Path Setup

- backward ant carries:
 - h = number of hops travelled so far
 - t_d^i = an estimate of the travel time of a packet from i to d (calculated using estimate of link layer delay and queue occupation for every node visited)
- at each node $i \in P$, backward ant updates T_{nd}^i :

$$T_{nd}^i = \gamma T_{nd}^i + (1 - \gamma) \tau_d^i, \quad \gamma \in [0, 1]$$

$$\tau_d^i = \left(\frac{t_d^i + ht_1}{2} \right)^{-1}$$

- t_1 = estimation of the time to travel one hop in unloaded conditions.

AntHocNet: Proactive Path Update

- Ants periodically resent to rediscover path in case of changes (e.g. mobility,...)
- *Hello* messages (like in OLSR) to keep track of neighbor

AntHocNet: Results and Discussion

When compared to AODV, AntHocNet [7] has:

- lower delay (good)
- higher delivery ratio (good)
- higher control overhead (bad!)

Criticism of Ants algorithms in general

- time to convergence?
- many parameters, how to tune them?
- how to compare with other approaches? parameter dependency...

Adaptive routing debate: route oscillations?

Chemical Computing

Computation models inspired by chemistry:

- Program = set of “molecules” (data and/or code) that float in a solution (“soup”)
- Execution = “molecules” are consumed and produced in “chemical reactions”
- Computation stops when no more reactions can take place (inert solution)

Why?

- High parallelism: no artificial sequentiality imposed by algorithm: closer to specification

Chemical Computing

Well-known chemical computing models:

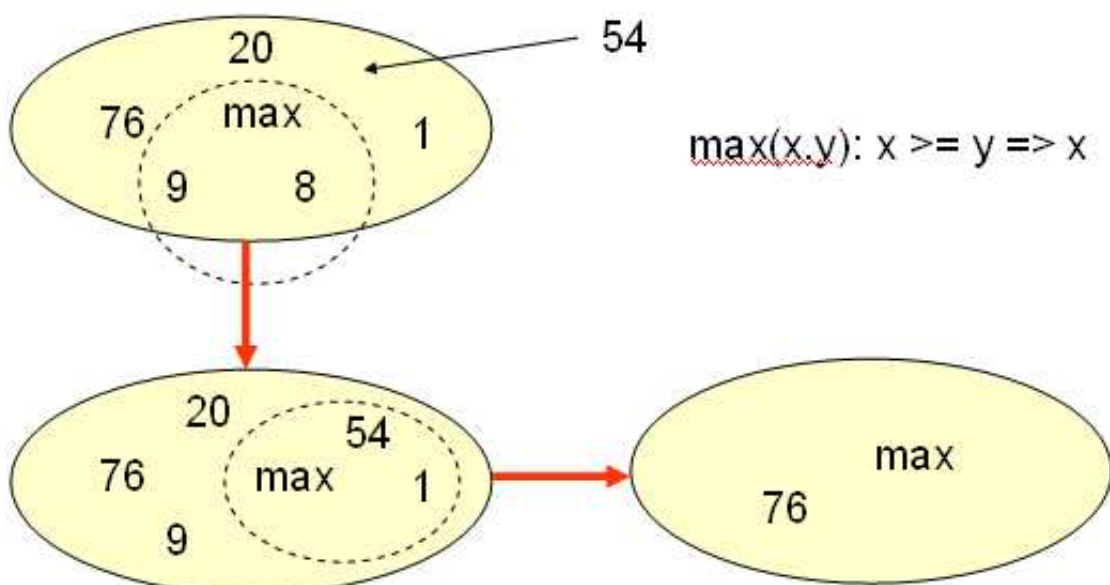
- Gamma (1986) [2, 3]
- Chemical Abstract Machine (CHAM) [4]
- Membrane Computing or P Systems [10]

In Basel:

- Fraglets [11]: for communication networks

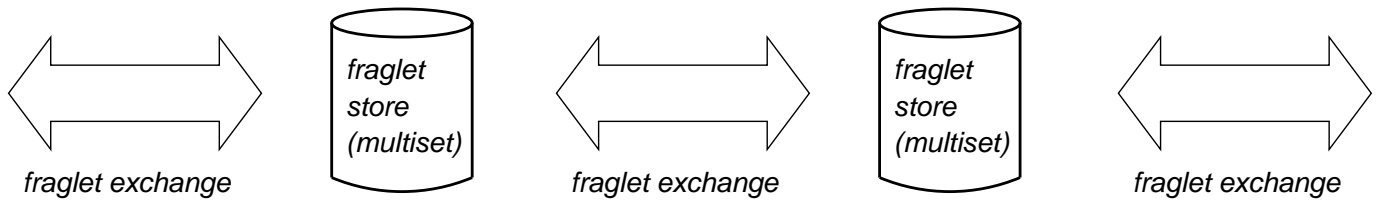
Chemical Computing: An Example in Gamma

➤ Chemical Programming: Gamma



Fraglets

Fraglet = computation fragment = “packet” = “rule” = “molecule”



- Fraglet = string W of symbols:
 $W = [s_0 s_1 \dots s_n]$ (sequence of pkt headers)
- Head symbol selects fraglet processing \rightarrow fraglet rewriting
- Two types of fraglet processing (shall be $O(1)$):
 - single fraglet transformation
 - chemical reaction (involving two fraglets)

Fraglets: Background

- Creation around 2001 by C. Tschudin [11]
- Goals:
 - Automated protocol synthesis and evolution
 - Unified code and data representation (active+passive networking)
 - Efficient packet processing engine: simple instructions with constant (short!) processing time

Fraglets Syntax

- Execution environment: multiset of fraglets
 - multiset = unordered set in which elements may appear more than once
- Syntax: string $n[s_1 s_2 \dots s_n]_m$
 - n = node where fraglet executes
 - m = multiplicity counter: number of occurrences of fraglet in multiset
- Goal: simple syntax that can be easily manipulated by automatic means (e.g. genetic programming)

Basic Instruction Set

Reaction rules: involve two fraglets

- Merge if match:
 $[match\ t\ tail1], [t\ tail2] \rightarrow [tail1\ tail2]$
- Persistent match (“catalyst”):
 $[matchp\ t\ tail1], [t\ tail2] \rightarrow [matchp\ t\ tail1], [tail1\ tail2]$

Basic Instruction Set

Transformation rules: involve a single fraglet

```
[dup t s tail]      --> [t s s tail]      # duplicate symbol
[exch t s1 s2 tail] --> [t s2 s1 tail] # swap symbol
[split f1 * f2]     --> [f1], [f2] # break at '*' position
[nul tail]          --> []              # discard fraglet
a[send ch b tail]  --> b[tail]         # UDP-style send
```

Examples:

```
[dup twice my fraglet] --> [twice my my fraglet]
[exch yes should I go] --> [yes I should go]
[split left side * right] --> [left side], [right]
[nul I will be killed] --> []
a[send ch b msg with detail] --> b[msg with detail]
```

Simple Programs

- Rewrite header tag
- Read from sensor
- Confirmed Delivery Protocol
- (Flow Control with Credits and Reordering)
- Lossy Link Emulation

Rewrite (rename) header tag

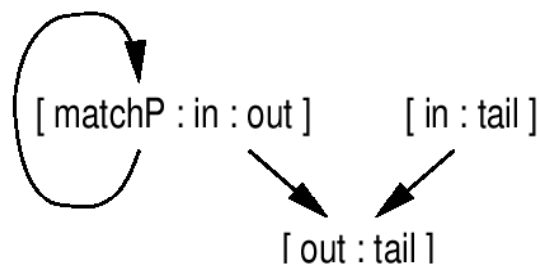
Goal:

[in tail] --> [out tail]

Solution:

[match in out]

Execution trace: 1 reaction:



Reading from a sensor

Read temperature indicated by a local sensor:

[match temp tempis]

...

[temp 25]

Result:

[tempis 25]

Reading from a sensor

Read temperature collected by a sensor at a remote node:

```
a[send ch b match temp send ch a tempbis]
```

```
b[temp 30]
```

Trace:

```
a[send ch b match temp send ch a tempbis] -->
```

```
b[match temp send ch a tempbis], b[temp 30] -->
```

```
b[send ch a tempbis 30] -->
```

```
a[tempbis 30]
```

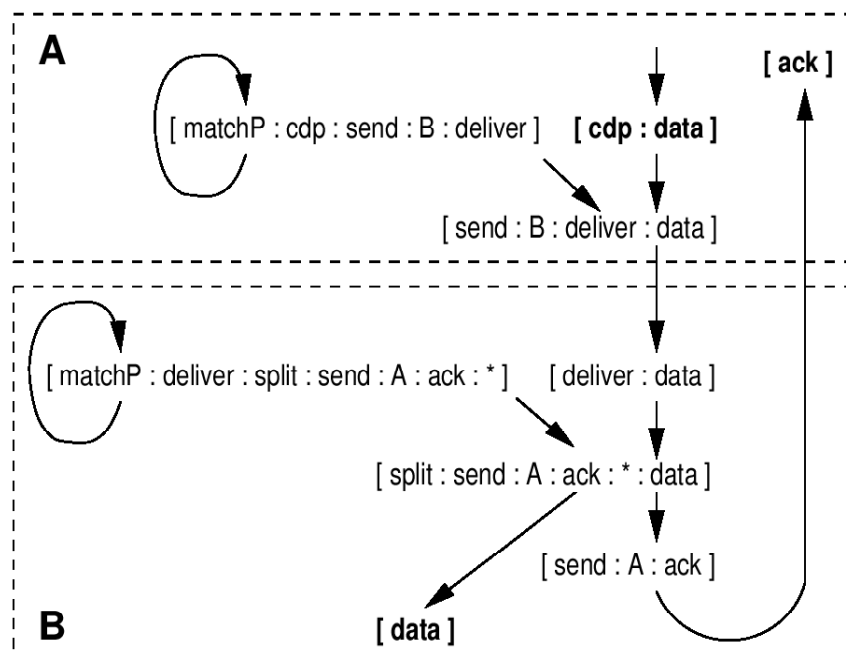
Code Mobility! Result:

```
a[tempbis 30]
```

Confirmed Delivery Protocol

```
a[matchp cdp send b deliver] # (channel omitted)
```

```
b[matchp deliver split send a ack *]
```



Confirmed Delivery Protocol

Active version:

```
a[matchp cdp send b split send a ack *]
```

Trace:

```
a[matchp cdp send b split send a ack *], [cdp data] -->
```

```
a[send b split send a ack * data] -->
```

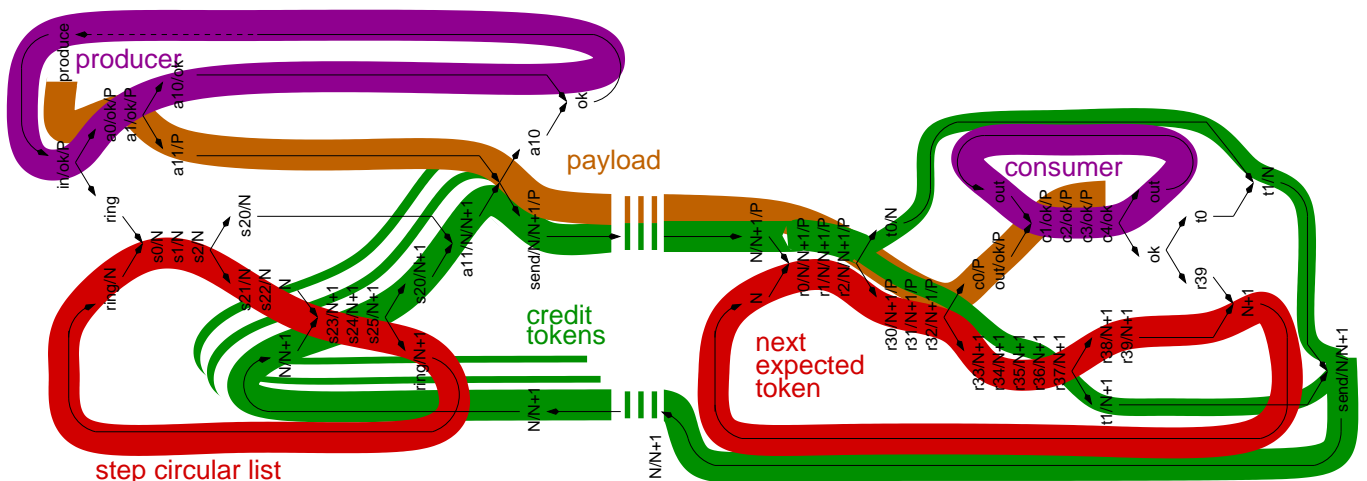
```
b[split send a ack * data] -->
```

```
b[data]
```

```
b[send a ack] -->
```

```
a[ack]
```

Flow Control with Credits and Reordering



Lossy Link Emulation

50% loss on average:

```
a[transmit b msg]100
a[matchp transmit send ch]
a[matchp transmit nul]
```

Result:

```
b[msg]44
```

.

Lossy Link Emulation

25% loss on average:

```
a[transmit b msg]100
a[matchp transmit send ch]3
a[matchp transmit nul]
```

Result:

```
b[msg]74
```

Easy to emulate other loss patterns, delays (`nop`, `wait`), etc.

References

- [1] ACO: Ants Colony Optimization.
<http://iridia.ulb.ac.be/mdorigo/ACO/ACO.html>.
- [2] J.-P. Banâtre, P. Fradet, and Y. Radenac. “*A Generalized Higher-Order Chemical Computation Model with Infinite and Hybrid Multisets*”. 1st International Workshop on New Developments in Computational Models (DCM’05), pages 5–14, 2005. To appear in ENTCS (Elsevier).
- [3] J.-P. Banâtre and D. L. Métayer. “*A new computational model and its discipline of programming*”, Sep. 1986. Technical Report RR0566, INRIA.
- [4] G. Berry and G. Boudol. “*The Chemical Abstract Machine*”. Theoretical Computer Science, 96:217–248, 1992.
- [5] C. Blum. “*Ant colony optimization: Introduction and recent trends*”. Physics of Life Reviews, 2(4):353–373, 2005.
- [6] E. Bonabeau, M. Dorigo, and G. Theraulaz. “*Swarm Intelligence: From Natural to Artificial Systems*”. Oxford University Press, 1999.
- [7] G. Di Caro, F. Ducatelle, and L. M. Gambardella. “*AntHocNet: An Adaptive Nature-Inspired Algorithm for Routing in Mobile Ad Hoc Networks*”. European Transactions on Telecommunications, Special Issue on Self-organization in Mobile Networking, 16(5), Oct. 2005.
- [8] M. Dorigo and L. M. Gambardella. “*Ant colonies for the traveling salesman problem*”. TR/IRIDIA/1996-3, Université Libre de Bruxelles.
<http://www.idsia.ch/luca/acs-bio97.pdf>.
- [9] M. Dorigo and T. Stützle. “*Ant Colony Optimization*”. MIT Press, 2004.
- [10] G. Paun. “*Computing with Membranes*”. Journal of Computer and System Sciences, 61(1):108–143, 2000.
- [11] C. Tschudin. “*Fraglets – A Metabolic Execution Model for Communication Protocols*”. Proc. 2nd Annual Symposium on Autonomous Intelligent Networks and Systems (AINS), Menlo Park, USA, Jul. 2003.
- [12] A. Turing. “*The chemical basis of morphogenesis*”. Philosophical Transactions of the Royal Society of London, Series B, 237(641), Aug. 1952. off-print, available at <http://www.turingarchive.org/browse.php/B/22>.