Amorphous Computing

上嶋裕樹・萩谷昌己
東京大学大学院
情報理工学系研究科
CREST, JST

uejima@is.s.u-tokyo.ac.jp
hagiya@is.s.u-tokyo.ac.jp

目次

- Amorphous Computingとは
- Wave Propagationによるパターン形成
  - 例1: Growing-Point Language
  - 例2: Origami Shape Language
  - 例3: Microbial Colony Language
- 生物学との関係
  - 例: Cellular gateと細胞計算
- まとめ

Amorphous Computingとは

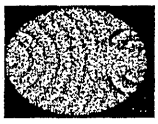
- A.C.の背景
- A.C.の定義
- Computational Particleの性質
- A.C.の論点

Amorphous Computingの背景

- 微細加工技術と細胞工学の発達
- 低コストで様々なプロセッサの製造が可能に。
  ただし，厳密に正確な動作は要求しない。
- 新しい計算パラダイムとしての研究の必要性
  分子計算などのために。
- 1996年頃，MITで提唱された。

Amorphous Computingの定義

- 不規則に配置されて，非同期に局地的に相互作用するような，計算機能の要素 computational particle の集合としてモデル化される計算。

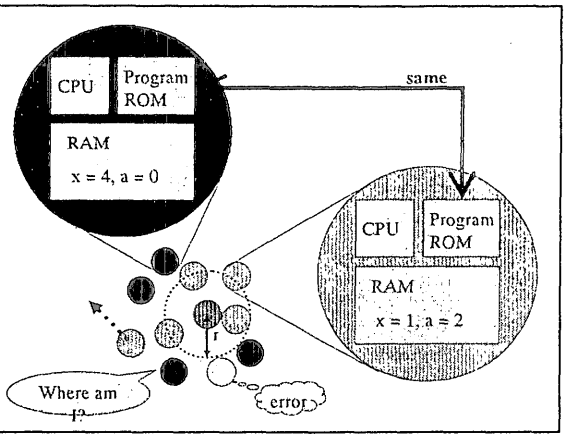


Computational Particleの性質

- 小さい計算能力と少量のメモリーを持つ。
- 全particleは同様にプログラムされている。(局所状態の保持や乱数発生は可能)
- 誤った挙動を示す可能性もある。
- 近接のparticleと短距離(半径r)の通信をする。
- 自分たちの位置や方向に関する情報をもたない。
- 環境に影響される。
- 何らかの動作をするかもしれない。
- 動き回るかもしれない。

全体としては超並列計算システムになっている。

© 日本オペレーションズ・リサーチ学会．無断複写・複製・転載を禁ず．



Amorphous Computingの論点

- 効果的にプログラムするにはどうすればよいか。
  - 生物学的な組織の形成との関係は？
  - ⇒ 様々なA. C.用プログラミング言語
- 生物学との関係
  - 生物のメタファーとしてのA. C.
  - 生物は単なるメタファーでなく，実装に使えないか？
  - ⇒ 遺伝子発現制御機構を利用した計算素子 (Cellular gate)

Wave Propagationによるパターン形成

- ✧ Wave Propagationとは
- ✧ 例1：Growing-Point Language
- ✧ 例2：Origami Shape Language
- ✧ 例3：Microbial Colony Language

Wave Propagationとは

- 近距離通信の繰り返しによるメッセージの伝達
  - 最初のメッセージを出す anchor particle
  - ホップの情報をもつメッセージ(＝距離の情報)
- 生物学的なパターン形成と関連
- 複数個のanchor particleによる制御
  - 「成長の阻害」や「屈動性(tropism)」in GPL
  - 垂直二等分線の形成 in OSL

例1：Growing-Point Language

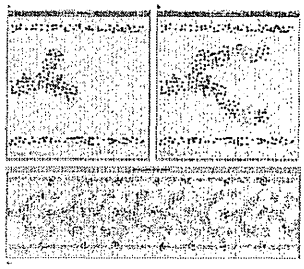
- 成長点(growing-point)の移動軌跡でパターンを形成する。
  - 物理的に移動するのではない。
  - 成長点として振舞うparticleが移り変わっていく。
- 各particleが分泌する「フェロモン」で，成長点の移動を制御する。
  - フェロモンはメッセージ伝達で実現している。
- GPLで書かれたプログラムは，コンパイルしてparticleのプログラムになる。
  - 大局的な観点からプログラミングできる。

GPLによるプログラムの例

```
(define-growing-point (make-red-branch length)
 (material red-stuff)
 (size 5)
 (tropism (and (away-from red-pheromone)
               (and (keep-constant pheromone-1)
                    (keep-constant pheromone-2))))
 (avoids green-pheromone)
 (actions
  (secrete 2 red-pheromone)
  (when (= length 1)
        (terminate))
        (default
          (propagate (- length 1))))))
```


© 日本オペレーションズ・リサーチ学会．無断複写・複製・転載を禁ず．

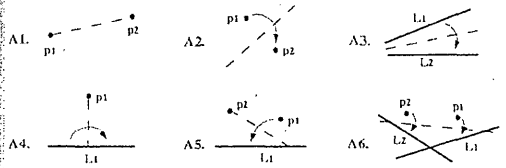
GPLによるパターン形成の例



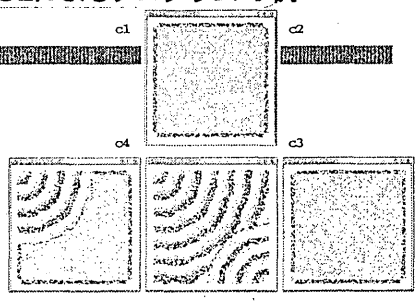
例2：Origami Shape Language

- 4つの「折り紙公理」に基づいてパターン（折り線）を形成する。
  - 折り線に沿って折ることもできる。
  - メッセージ伝達で相対的な位置情報を得ることにより，折り線の形成を実現。
- OSLで書かれたプログラムは，コンパイルしてparticleのプログラムになる。
  - 大局的な観点からプログラミングできる。

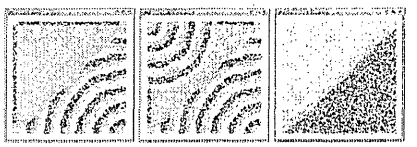
（参考）折り紙公理



OSLによるプログラムの例



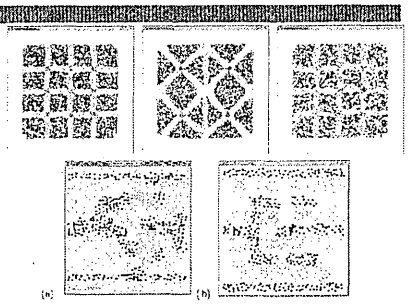

```
(define d1 (crease-p2p c1 c3
green))
```



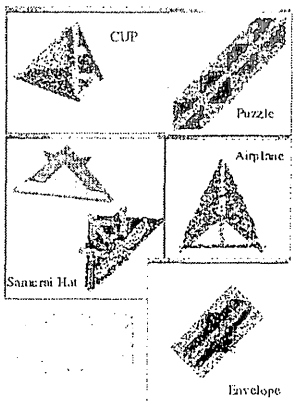
```
(define front (create-region c3
d1))
```

```
(execute-fold d1 apical landmarke=c3)
```

OSLによるパターン形成の例



© 日本オペレーションズ・リサーチ学会．無断複写・複製・転載を禁ず．



例3: Microbial Colony Language

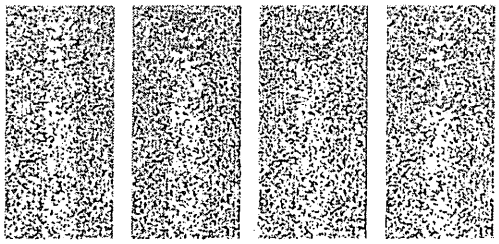
- ruleとmarkerとメッセージによるプログラミングモデル
  - 独立したいくつかのruleによって構成されたプログラム
  - ruleの構成要素はmessage, condition, actionsの3つ。
  - 各particleの状態は2値のmarkerの集合。
  - ruleのactionsがmarkerのset/clearおよびメッセージ発信をする。
  - メッセージはカウンターを持つ。
- particleによって実行されるプログラムを，MCLで直接記述する。
  - 個々の点の動きを規定することによって，全体の動作を制御する。

MCLによるプログラムの例

```
(start
 Creat
 ((send (make-seg C 1) 3)))

((make-seg seg-type seg-index)
 (and Tube (not C) (not D))
 ((set seg-type)
  (set seg-index)
  (send created 3)))

(((make-seg) (= 0))
 Tube
 ((set Bottom)))

(((make-seg) (> 0))
 Tube
 ((unset Bottom)))
```

```
(created                  } message
 (or C D)                 } condition
 ((set Waiting 10)))      } actions

(*
 (and Bottom C 1 (Waiting (= 0)))
 ((send (make-seg D 1) 3)))

(*
 (and Bottom D 1 (Waiting (= 0)))
 ((send (make-seg C 2) 3)))

(*
 (and Bottom C 2 (Waiting (= 0)))
 ((send (make-seg D 2) 3)))

(*
 (and Bottom D 2 (Waiting (= 0)))
 ((send (make-seg C 3) 3)))
```

MCLによるパターン形成の例



生物学との関係

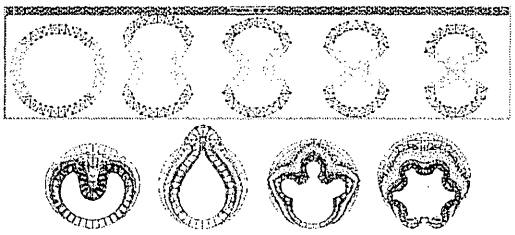
- 生物のメタファーとして
- 例1：Nagpallによるモデル
- 例2：Cellular gateと細胞計算
- 細胞計算を実現するためには

© 日本オペレーションズ・リサーチ学会．無断複写・複製・転載を禁ず．

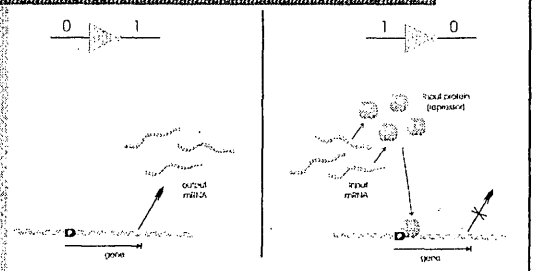
生物のメタファーとして

- 形態形成の過程との類似
  - 胚の成長における原腸形成など
  - Nagpalによるモデル化
- 「正しい」答を得ることが目的ではない。
- 信頼性の低い部品を使って，満足な答えを高確率で出す。
  - どうシステムを抽象的に構成するか。

例1：Nagpalによるモデル



例2：Cellular gateと細胞計算 (inverterを作る)



細胞計算を実現するためには

- 生物化学的な知識が不足している。
  - DNA-binding proteinやmatching repressor patternの知識
  - 動力学的な定数
  - 遺伝的制御機構外でのたんぱく質の相互作用
  - 細胞の本来の機能に干渉しない新しい機構の挿入方法
- 回路を実装するには多くの種類のタンパク質が必要。
  - ゲート間の干渉を防ぐため。
- 生物回路に関するツールが必要。
  - simulator, verifier, plasmid compiler
- 発生論的に有機体を作ってしまうという方法もある。

まとめ

✧ Amorphous Computingの将来像

Amorphous Computingの将来像

- 細胞計算は(超並列だとしても)遅い。
- 細胞外の複雑な構造物の構成や，ナノレベルでの制御の基礎とする。
  - たとえば分子スケールの電子部品を作る。
- Ubiquitous Computingの基盤として
  - Smart paint
  - Smart dust
- 相互作用型の一般的な計算モデル

© 日本オペレーションズ・リサーチ学会．無断複写・複製・転載を禁ず．

参考文献(1)

- Amorphous Computing Home Page http://www.swiss.ai.mit.edu/projects/amorphous/
- H. Abelson, D. Allen, D. Coore, C. Hanson, G. Homsy, T.F. Knight, Jr., R. Nagpal, E. Rauch, G.J. Sussman, R. Weiss: Amorphous Computing, Communications of the ACM, Vol. 43, No. 5, May 2000, pp 74-82.
- Daniel N. Coore: Botanical Computing: A Developmental Approach to Generating Interconnect Topologies on an Amorphous Computer, PhD thesis, MIT, Department of Electrical Engineering and Computer Science, February 1999.

参考文献(2)

- Radhika Nagpal: Programmable Self-Assembly: Constructing Global Shape using Biologically-inspired Local Interactions and Origami Mathematics, PhD thesis, MIT, Department of Electrical Engineering and Computer Science, June 2001.
- Ron Weiss: Cellular Computation and Communications using Engineered Genetic Regulatory Networks, PhD thesis, MIT, Department of Electrical Engineering and Computer Science, September 2001.

© 日本オペレーションズ・リサーチ学会. 無断複写・複製・転載を禁ず.