1 次元可逆セル・オートマトンの クリーン可逆シミュレーションの実現

2014SE048 木村孝大　2014SE114 矢澤拓海

指導教員：横山哲郎

1　はじめに

文献 [1] の研究において可逆セル･オートマトンのクリーン可逆シミュレーションを可逆プログラミング言語 Janus を用いて実現する方法が提案された．無限長のセル空間を可逆スタックを用いて実現し，Janus で直接可逆セル・オートマトンをシミュレートすることでクリーンな可逆シミュレーションが実現された．しかし，プログラムに対する解析が不十分であり，また，改善の余地があった．本研究では，Janus を用いて具体的なプログラムを与えることを目的とする．このとき，無限長であるセル空間を，分割セル・オートマトンで扱い易い形で実現する．

2　関連研究

本章では，本研究に関連する研究について述べる．本章ではセル・オートマトンの基本概要や定義について述べ，3 章以降にて，実装するために必要な概念や定義などをより詳しく述べる．

2.1　セル・オートマトン

2.1.1　オートマトン

オートマトンとは常に 1 つの内部状態を持っており外部から連続している情報が入力され，それによって内部状態が遷移し，何らかの情報を出力するシステムのことをオートマトンと呼ぶ．オートマトンの内，内部状態が有限個であるものを有限オートマトンと呼ぶ．

2.1.2　セル・オートマトンの概要

セル・オートマトン (cellular automaton, 以下 CA) とはセルと呼ばれる大量の有限オートマトンを規則正しく配置したものであり，時間とともにそれぞれのセルの状態が他のセルの影響を受けて変化していくシステムである．化学や物理学など，様々な学問のシミュレーションに利用されている．

2.1.3　セル・オートマトンの定義

CA は以下の定義で与えられる．

$$A = \left(\mathbb{Z}^k, Q, (n_1 \dots n_m), f, \#\right)$$

$\mathbb{Z}^k$ は k 次元ユークリッド空間中の整数座標を持つ点集合であり，この点にセルを配置する．セルが配置されている，この空間をセル空間と呼ぶ．Q は各セルが取り得る内部状態の空でない有限集合である．$(n_1 \dots n_m)$ は $\left(\mathbb{Z}^k\right)^m (m = 1, 2, \dots)$ の要素である．これは近傍と呼び，セルの状態が遷移する際に参照するセルのことである．関数 $f : Q^m \to Q$ は各々のセルの状態を決める局所関数である．$q_1, \dots, q_m, q \in Q$ に対し関係 $f : (q_1, \dots, q_m) = q$ が成り立つときそれを遷移規則と呼ぶ．したがって，f は遷移規則の集合で記述できる．$\#$ は静止状態を表し，$f(\#, \dots, \#) = \#$ をみたす．これは空白に相当し，指定されない CA も存在する．$\alpha : \mathbb{Z}^k \to Q$ であるような写像 α を集合 Q 上の k 次元の状相と呼ぶ．α は A の状相ともいう．したがって，$x \in \mathbb{Z}^k$ とするとき，$\alpha(x)$ は座標 x の位置にあるセルの状態を表す．集合 Q 上の k 次元状相すべての集合を $\mathrm{Conf}_k(Q)$ で表す．つまり $\mathrm{Conf}_k(Q) = \{\alpha \mid \alpha : \mathbb{Z}^k \to Q\}$ である．k は基本的に前後関係からわかるため省略する．静止状態が指定された CA には有限状相と無限状相の概念が存在し，集合 $\{x \mid x \in \mathbb{Z}^k \wedge \alpha(x) \neq \#\}$ が有限の場合を有限状相，そうでない場合を無限状相と呼ぶ．

2.1.4　フレドキンの自己複製セル・オートマトン

フレドキンの自己複製 CA とは，有限個の黒丸からなるパターンを時刻 0 に与えると一定時間後にそのパターンが複製されているという CA である．空白を状態 0，黒丸を状態 1 とすると，以下のように記述される．

$$A_{\mathrm{F}} = \left(\mathbb{Z}^2\{\{0,1\}\}, N_{\mathrm{N}}, f_{\mathrm{F}}, 0\right) \tag{1}$$

$$N_{\mathrm{N}} = ((0,0), (0,1), (1,0), (0,-1), (-1,0)) \tag{2}$$

$$\forall q_0, q_1, \dots, q_4 \in \{0,1\} : f_{\mathrm{F}}(q_0, q_1, q_2, q_3, q_4) = q_1 \oplus q_2 \oplus q_3 \oplus q_4 \tag{3}$$

この CA の挙動の例を図 1 に示す．図 1 において，灰色のセルは直前の時刻に状態が 1 であったことを表す．

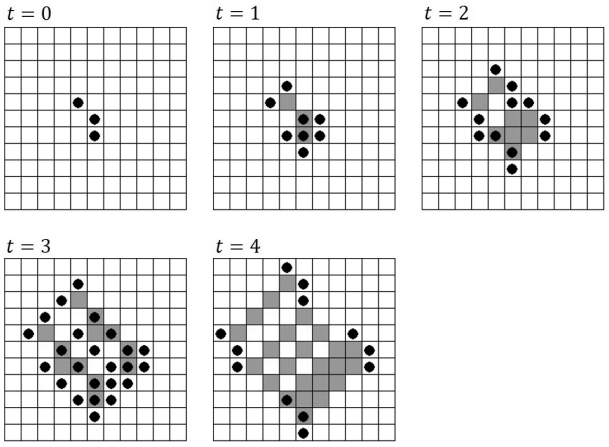

図 1　フレドキンの自己複製 CA の挙動

2.2 セル・オートマトンにおける可逆性

CA における可逆性は，大域関数に対する制約により定義される．可逆セル・オートマトン (reversible cellular automaton, 以下 RCA) とは，大域関数が全単射であるような CA, すなわち現在の状相に対し，直前の状相がちょうど 1 つ存在するような CA のことを言う．大域関数が単射であるか否かを判定するアルゴリズムは CA が 1 次元である場合にしか存在せず，RCA を実装するためには何らかの便法を用いる必要がある．

3 可逆セル・オートマトン

本章では，2 章で述べた CA ついて，可逆性の観点からより深く述べる．

3.1 セル・オートマトンにおける可逆性

RCA を定義する際に局所関数 $f : Q^{|N|} \to Q$ は $|Q| > 1 \wedge |N| > 1 \to |Q^{|N|}| > |Q|$ より，$|Q| = 1 \vee |N| = 1$ の自明な場合を除き単射になり得ない．局所関数の単射性として可逆性を定義することは特殊な設計をしなければ不可能である．そのため，CA における可逆性は大域関数に対する条件として定義せざるを得ない．CA における可逆性を定義するために単射 CA と反転可能 CA という概念を導入する．

3.1.1 単射セル・オートマトン

$A = (\mathbb{Z}, Q, N, f)$ を任意の CA，F をその大域関数とすると，以下の式が成り立つとき，CA A を単射セル・オートマトンと呼ぶ．

$$\forall \alpha_1, \alpha_2 \in \mathrm{Conf}\,(Q)\,[\alpha_1 \neq \alpha_2 \Rightarrow F\,(\alpha_1) \neq F\,(\alpha_2)] \quad (4)$$

式 (4) を満たす F は単射である．

3.1.2 反転可能セル・オートマトン

$A = (\mathbb{Z}, Q, N, f)$ を任意の CA，F をその大域関数とする．さらにある CA $A' = \left(\mathbb{Z}^k, Q, N', f'\right)$ が存在し，F' をその大域関数とする．この時，次式が成り立つとき A を反転可能セル・オートマトンと呼ぶ．

$$\forall \alpha, \beta \in \mathrm{Conf}\,(Q)\,[F\,(\alpha) = \beta \Leftrightarrow F'\,(\beta) = \alpha] \quad (5)$$

また，単射 CA と反転可能 CA の間には次の定理が成り立つことがわかっている．CAA に対し，それが単射 CA である時，かつそのときに限り反転可能 CA である．これは単射 CA と反転可能 CA の概念が等価であるということであり，あらためてこれらを RCA として定義することができるようになる．CA A は，それが単射 CA であるとき，あるいはそれが反転可能 CA であるとき，可逆セル・オートマトンと呼ばれる．

3.2 分割セル・オートマトン

分割セル・オートマトンは各セルがいくつかの部分に分割された構造を持つ CA である．分割 CA は可逆 CA を設計することができる CA の一種である．

3.2.1 分割セル・オートマトンの定義

分割 CA は次の式によって定義される

$$A = \left(\mathbb{Z}^k\,(Q_1, \ldots, Q_m)\,, (n_1, \ldots, n_m)\,, f, (\#_1, \ldots, \#_m)\right) \quad (6)$$

標準的な CA 同様，$\mathbb{Z}^k$ はセルが配置される点集合であり，$(n_1, \ldots, n_m)$ は近傍である．Q_i は各セルの第 i 部分 $(i = 1, \ldots, m)$ が取り得る，内部状態がから出ない有限集合である．これは，各セルが m 個に分割されることを示す．関数 $f : (Q_1 \times \cdots \times Q_m) \to (Q_1 \times \cdots \times Q_m)$ は各々のセルの状態を定める局所関数である．$(\#_1, \ldots, \#_m) \in (Q_1 \times \cdots \times Q_m)$ は $f\,(\#_1, \ldots, \#_m) = (\#_1, \ldots, \#_m)$ を満たす静止状態である．便宜上，静止状態 $\#_1, \ldots, \#_m$ は m 個の部分状態を同一視し，$(\#, \ldots \#)$ として表す．標準的な CA 同様に，静止状態は空白に相当し，それが指定されない CA も存在する．標準的な CA と同様に，$(q_1, \ldots, q_m)\,, (q'_1, \ldots, q'_m) \in Q_1 \times \cdots \times Q_m$ に対して $f\,(q_1, \ldots, q_m) = (q'_1, \ldots, q'_m)$ が成り立つとき，これを A の遷移規則と呼ぶ．$Q = Q_1 \times \cdots \times Q_m$ とし，$\alpha : \mathbb{Z}^k \to Q$ であるような写像 α を標準的な CA と同様に A の状相と呼ぶ．A の状相すべての集合を $\mathrm{Conf}\,(Q)$ で表す．有限状相についても同様で，集合 $\{x \mid x \in \mathbb{Z}^k \wedge \alpha\,(x) \neq (\#, \ldots, \#)\}$ が有限であるような α を有限状相と呼ぶ．ここで関数 pr_i を任意の $(q_1, \ldots, q_m) \in Q$ に対して $\mathrm{pr}_i\,(q_1, \ldots, q_m) = q_i$ となるような射影とする $(i = 1, \ldots, m)$．局所関数 f から導出される A の大域関数 $\tilde{F} : \mathrm{Conf}\,(Q) \to \mathrm{Conf}\,(Q)$ は

$$\forall \alpha \in \mathrm{Conf}\,(Q) \quad (7)$$

$$\forall x \in \mathbb{Z}^k : F\,(\alpha)\,(x) = f\,(\mathrm{pr}_1\,(\alpha\,(x + n_1)), \ldots \mathrm{pr}_m\,(\alpha\,(x + n_m))) \quad (8)$$

と定義される．分割 CA は，局所関数が単射である場合は可逆となることが知られている．[2]

3.2.2 1 次元 3 近傍分割セル・オートマトン

分割 CA の具体例として文献 [1] でプログラムの実現の際に利用された以下のような 1 次元 3 近傍分割 CA を挙げる．

$$A_{\mathrm{p}} = (\mathbb{Z}, (L, C, R)\,, (1, 0, -1)\,, f_{\mathrm{p}}\,(0, 0, 0)) \quad (9)$$

$$L = C = R = \{0, 1\} \quad (10)$$

各セルは右部分，中央部分，左部分のように 3 つに分割されている．遷移の際には右隣のセル左の部分，着目セルの中央部分，左隣のセルの右部分の状態に依存する．分割されたセルの左部分を l, 中央部分を c, 右部分を r, と置くと遷移規則 $f\,(l, c, r) = (l', c', r')$ は図 2 のように表現できる．

単射な局所関数 f_{p} として表 1 のような遷移規則が挙げられる．

状態 1 を黒丸，状態 0 を空白とした時の A_{p} の挙動は図 3 のようになる．分割された各セルの右部分にある黒丸は右へ，左部分にある黒丸は左へと動くことが見て取れる．

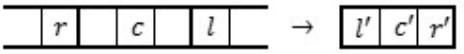

図 2　$f\,(l,c,r)$ の挙動

表 1　1 次元 3 近傍分割 CA の遷移規則の例

r	c	l	l'	c'	r'
0	0	0	0	0	0
0	0	1	1	0	0
0	1	0	0	1	0
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	1
1	1	0	0	1	1
1	1	1	1	1	1

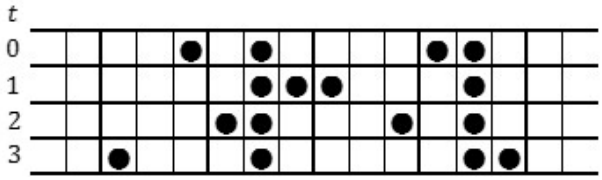

図 3　A_{p} の挙動

4　既存のクリーン可逆シミュレーション

本章では，文献 [1] における RCA のクリーン可逆シミュレーションがどのように実現されたか，その実現方法について論じる．

4.1　クリーン可逆シミュレーションの実現方法

Janus で RCA のクリーン可逆シミュレーションを実現するには，単射な大域関数 F をクリーンに実現する必要がある．RCA のシミュレーションを行う上で，無関係な情報が残らないならばクリーン可逆シミュレーションである．文献 [1] では単射な局所関数 f を用いた単射な大域関数 F_{p} を Janus でクリーンに実現することでオーバーヘッドを削減し，クリーン可逆シミュレーションが実現された．Janus プログラムの実行は，Janus のオンラインインタプリタ [3] を用いる．

4.2　スタックを用いたセル空間の実現

RCA をシミュレートするには無限長のセル空間を有界なメモリ上で実現する必要がある．[1] では，RCA 内部の静止状態のセルを除いたセルの個数が有限個である場合に限定し，Janus に備えられたスタックを用いてセル空間が実現された．

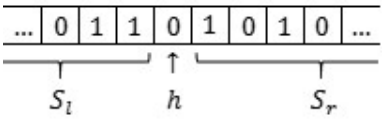

図 4　セル空間の図

4.2.1　スタックを用いたセル空間の実現

静止状態のセルを除いたセルの個数が有限個である RCA では，静止状態のセルが無限に連続する．そこで，連続する静止状態のセルを空スタックに置き換えることにより有限個の要素数で無限長のセル空間を表現し，有界なメモリ上での実現が可能となる．この方法では，静止状態のセルを除いたセルの個数が無限個である RCA，静止状態が 2 つ以上設定されている RCA を実現することができない．セル空間上で基準とするセルを h とし，その左側・右側のセルをそれぞれスタック s_l, s_r で表す．CA の各セルの状態を表す要素として 1,0 が存在するものとする．ここで，0 は静止状態を表す．連続する無限個の 0 は空スタック [] で代替表現する．これを図で表すと，図 4 のように表すことができる．ここで，図 4 のように $\ldots,0,1,1,0,1,0,1,0,\ldots$ と表される状相があり，h の位置が図に表されたものと同じであると仮定すると，

$$h = 0 \tag{11}$$
$$sl = 1 :: 1 :: [] \tag{12}$$
$$sr = 1 :: 0 :: 1 :: [] \tag{13}$$

となる．スタックによるセル空間を (s_l, h, s_r) と表す．このとき，図 4 は $\{(1 :: 1 :: [], 0, 1 :: 0 :: 1 :: [])\}$ と表される．どちらも同じ状相を表しているため，単射な大域関数を用いていたとしても，次の時刻に同じ状相へと遷移し，単射性が失われてしまう．このように，セル空間の表現が一意ではなくなってしまい，可逆性が失われてしまうことを避けるため，空白状態と空スタックが連続した場合は，その空白状態を含めて空スタックで表現し，スタックの底に空白状態が入らないよう状相の表現を統一する．

4.2.2　Janus による実現

Janus には可逆性の保証されたスタックと，そのスタックに対する pop，push 操作が定義されている．int 型変数 x とスタック s に対する操作は以下の通りである．

·push (x, s)

変数 x の値をスタック s の先頭に格納する．変数 x の値は 0 となる．

·pop (x, s)

変数 x の値が 0 であり，かつスタック s が空でない場合，スタック s の先頭の値を変数 x に格納する．

文献 [1] では Janus 上で int 型変数 h，スタック sl, sr を用いて 4.2.1 節のセル空間を実現している．スタック s_l, s_r はそれぞれスタック sl, sr で表現し，セル h は変数 h で

```
1    sl=<1,1],h=0,sr=<1,0,1]
```

図 5 プログラム上でのセル空間

```
1    push(h,sr)
2    pop(h,sl)
```

図 6 右送り

```
1    push(h,sr)
2    pop(h,sl)
```

図 7 左送り

表現している．RCA のシミュレーションを行う際に，セルの中身を読み取る必要があり，直接状態を読み取ることができるのは h のみである．そのため，セル空間全体を右送り，もしくは左送りすることでセルの中身を読み取る．セル空間の右送りは，まず変数 h の要素をスタック sr に push し，その後にスタック sl の要素を変数 h に pop することで実現する．これで h の左側のセルの状態を読み取ることができる．同様に，セル空間全体の左送りは，変数 h の要素をスタック sl に push し，スタック sr の要素を変数 h に pop することで実現し，h の右側のセルの状態を読み取ることができる．このようにして読み取った値を使用し，状相の遷移を行う．

4.2.3 Janus プログラム

文献 [1] において実装されたセル空間の右送りと左送りのプログラムについて記述する．図 4 のような $\{\ldots,0,1,1,0,1,0,1,0,\ldots\}$ という状相を仮定する．この状相は

$$(1::1::[],0,1::0::1::[]) \tag{14}$$

と表される．プログラム上では，このセル空間を int 型変数 h とスタック sl,sr を用いて図 5

となる．右送りを実現するプログラムは図 6 の通りである．

変数 h の要素をスタック sr に push し，スタック sl の要素を変数 h に pop することで右送りを実現している．同様に，図 4 のセル空間に対し，左送りを 1 回行うと，

$$(0::1::1::[],1,0::1::[]) \tag{15}$$

となる．図 7 のように，変数 h の要素をスタック sl に push し，スタック sr の要素を変数 h に pop することで左送りを実現できる．以上が Janus 上でのセル空間の実現の例である．ただし，Janus オンラインインタプリタには空スタックからの pop を行う機能がなく，空スタックからの pop を代替する機能を実装する必要がある．文献 [1] では状相の遷移の際に空スタックからの pop が起きる可能性がある．そのような状況が起きた場合，pop された値が格納される予定の変数に対し，その値が必ず 0 になるように実現している．また，この方法では空スタックに対して空白状態の push が行われた際にそのまま格納してしまいセル空間の表現が一意ではなくなってしまう．しかし，遷移後の状相が異なるため，単射性は損なわれておらず，可逆性は保たれている．また，余計な空白状態が増えてしまう問題がある．

5 おわりに

本稿では，過去に実現された RCA のクリーン可逆シミュレーションの方法についての解析までを行い，今後の課題は 4.2.3 節にて示された問題の解決と，可逆スタックを用いたクリーン可逆シミュレーションに対し，厳密な定義を与えることである．

参考文献

[1] 渡邉恭平：可逆スタックを用いた可逆セル・オートマトンのクリーン可逆シミュレーション．南山大学情報理工学部 2013 年度卒業論文

[2] 森田憲一：可逆計算，ナチュラルコンピューティング・シリーズ Vol. 5, p. 87-118，近代科学社 (2012)

[3] Claus, S.N.and Michael, B.: Janus Playground, DIKU(online), available from `http://topps.diku.dk/janus-playground/`