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

2010SE265 渡邉恭平
指導教員：横山哲郎

1 はじめに

可逆局所関数を用いた有限状相可逆セル・オートマトン (reversible cellular automata on finite configuration with reversible local maps, 本論文では単に RCA と呼ぶ) のクリーン可逆シミュレーションが，時間・空間オーバーヘッドを伴えば，可逆プログラミング言語によって実現できることが知られている．例えば，RCA を可逆チューリングマシン (RTM: reversible Turing machine) で，さらに RTM をスタックの備わった可逆プログラミング言語 Janus で，2 段階でシミュレートする方法が知られている．このオーバーヘッドを取り除くには，RCA を Janus で直接的にシミュレーションすればよい．

しかし，RCA のシミュレーションを Janus で実現するには 2 点の課題がある．第 1 に，RCA の無限個のセルを有界なメモリしかない実行環境で実現できるよう Janus プログラムを作成しなければならない．Janus を用いた RCA のクリーン可逆シミュレーションとして，固定長配列で有限個のセルを実現したものが知られている．しかし，固定長配列では有界でない個数のセルを表現することができない．第 2 に，RCA の効率的な可逆シミュレーションはクリーンでなければならない．RCA の状相を更新する大域関数をいかにクリーンに実現するかが問題である．

本論文では，RCA の無限個のセルの実現，クリーンな状相の遷移の実現によってこれら 2 点の課題を解決し，1 次元 RCA について，可逆スタックを用いた RCA のクリーン可逆シミュレーションが実現できるという知見を得た．

2 関連研究

2.1 セル・オートマトン

セル・オートマトン（CA）[4] は，多数のセルを規則的に連結・配置し，そのセルの集合の状態を表す状相が，動的に変化するシステムである．CA は

$$(\mathbb{Z}^k, Q, N, f, \#)$$

により定義される．ここで，Q はセルの状態と呼ばれる空でない有限集合，$N(=\{n_1,\ldots,n_m\})$ は近傍と呼ばれる $\mathbb{Z}^k$ の部分集合，局所関数 $f : Q^m \to Q$ は各セルの状態遷移を定める．なお，$\# \in Q$ は静止状態と呼ばれ，$f(\#,\ldots,\#) = \#$ を満たす．集合 Q 上の k 次元の状相とは $\alpha : \mathbb{Z}^k \to Q$ であるような写像 α をいう．Q 上の k 次元状相すべての集合を $\mathrm{Conf}_k(Q) = \{\alpha \mid \alpha : \mathbb{Z}^k \to Q\}$ で表す．状相 α は，集合 $\{x \mid x \in \mathbb{Z}^k \wedge \alpha(x) \neq \#\}$ が有限であるとき，有限と呼ばれる．

関数 $F : \mathrm{Conf}_k(Q) \to \mathrm{Conf}_k(Q)$ を

$$\forall \alpha \in \mathrm{Conf}_k(Q), \forall x \in \mathbb{Z}^k : \\ F(\alpha)(x) = f(\alpha(x+n_1),\ldots,\alpha(x+n_m)) \qquad (1)$$

と定める．F を状相の遷移を定める大域関数と呼ぶ．

2.2 可逆セル・オートマトン

CA は，その大域関数 F が単射であるとき，可逆セル・オートマトンと呼ばれる [4]．任意の可逆セル・オートマトン A には，状相の遷移がちょうど逆であるような逆セル・オートマトン A' が存在する．

RCA の例として，1 次元 3 近傍分割 CA[4] を挙げる．この CA は，1 次元 CA を構成するセルを 3 つごとに分割した CA である．この CA は，単射な局所関数 f を全セルに適用することで単射な大域関数 F を定義した例である [3][4]．各分割の各セルの左，中央，右の各部分の内部状態の集合を L, C, R とおく．各セルの内部状態を $Q = L \times C \times R$ とすれば，この CA の局所関数 $f : Q \to Q$ を定義できる．時刻 t において，図 1 のように l, c, r とおくと，それぞれ局所関数 f にしたがって l', c', r' へと遷移する．この CA の遷移規則は $f(l, c, r) = (l', c', r')$ である．例えば，表 1 に示すような遷移規則を実現する単射な局所関数 f が存在する [4]．

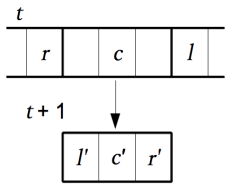

図 1 1 次元 3 近傍分割 CA における遷移規則

表 1 遷移規則

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

3　可逆シミュレーションの実現

スタックを備えた Janus を用い，セルの個数が有界でない RCA のクリーン可逆シミュレーションを実現することが本論文の狙いである．このスタックを，本論文では可逆スタック [5] と呼ぶ．RCA のシミュレーションは，可逆スタックによる記憶空間上で行われる．

3.1　可逆スタックによる記憶空間の実現

CA の各セルの状態を表す要素 1, 0 があり，0 を静止状態とする．0 が無限に連続する場合は空スタック [] で代替表現する．図 2 のように，現在の位置 h を基準とし，その右側，左側を可逆スタック s_l, s_r で表現し，可逆スタックによる記憶空間 $(s_l,\, h,\, s_r)$ を

$$(1 :: [],\, 0,\, 0 :: 1 :: [])$$

と表す．実際の Janus プログラム上では [] を省略している．例えば，状相 $\{\ldots, 0, \ldots\}$ を表現する際，h で表す要素は，[] ではなく 0 で現れる．そのまま h が別の要素を表すと，

$$sl = 0 :: []$$

のような表現が予想されるが，[] は 0 の無限個の連続を表しており，0 が [] の直前に現れる表現を許さないという制約を課す．これにより，可逆スタックの可逆性を保証しているという [5].

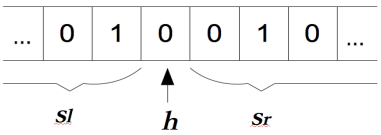

図 2　CA をシミュレートする記憶空間

4　実装

Janus オンラインインタプリタ [6] を用い，可逆スタックを用いて RCA のクリーン可逆シミュレーションを実装した例を示す．ここでは，表 1 の遷移規則を実現する単射な局所関数 f をもつ 1 次元 3 近傍 CA を扱う．時刻 t のとき，状相 $\{\ldots, 0, 1, 0, 1, 0, \ldots\}$ があったとすると，例えば可逆スタック s_l, s_r, int 型変数 h を用いてスタックによる記憶空間 $(s_l,\, h,\, s_r)$ を

$$([],\, 1,\, 0 :: 1 :: [])$$

と表す．h の要素を変更する場合，h の要素を s_l へ push し s_l の要素を h へ pop することで記憶空間の左送りを行うか，h の要素を s_r へ push し s_l の要素を h へ pop することで記憶空間の右送りを行えばよい．この記憶空間に単射な大域関数 F を適用し遷移を行う．遷移後の状相を記録しなければならないが，単一の記憶空間に代入を行えば，情報は失われてクリーンでなくなってしまう．そこで，2 つの記憶空間 $M_0 : (s_{l0},\, h_0,\, s_{r0}), M_1 : (s_{l1},\, h_1,\, s_{r1})$ を用意する．M_0 に F を適用した結果を M_1 に格納し M_0 を ([], 0, []) へとゼロクリアする手続き，M_1 に F を適用した結果を M_0 に格納し M_1 を ([], 0, []) へとゼロクリアする手続きを交互に実行することで，代入を行うことなく，クリーンに状相の遷移を実現する．

M_0 の要素を [] を除いた先頭から [] を除いた最後まで $h0$ に pop し，それを int 型変数 l, c, r に格納しながら l, c, r に f を適用していくことで，M_0 に F を適用する．f を適用する際，[] の要素を参照する場合があるが，Janus オンラインインタプリタにはその機能がない．そこで，例えば [] から pop した要素を l に格納したい場合は，l の値を必ず 0 にし，[] からの pop を代替する．c, r についても同様である．f による遷移規則 $f(l,\, c,\, r) = (l',\, c',\, r')$ は表 1 のようになる．

開始時の時刻を $t = 0$ とすると，プログラムの実行結果は次のようになる．

順実行

$t = 0$　$M_0 : ([],\, 1,\, 0 :: 1 :: []),\, M_1 : ([],\, 0,\, [])$
$t = 1$　$M_0 : ([],\, 0,\, []),\, M_1 : ([],\, 1,\, 0 :: 0 :: 0 :: 0 :: 0 :: 0 :: 1 :: [])$

逆実行

$t = 1$　$M_0 : ([],\, 0,\, []),\, M_1 : ([],\, 1,\, 0 :: 0 :: 0 :: 0 :: 0 :: 0 :: 1 :: [])$
$t = 0$　$M_0 : ([],\, 1,\, 0 :: 1 :: []),\, M_1 : ([],\, 0,\, [])$

5　おわりに

本論文では，無限に連続する静止状態を除いたセルの個数が有限である RCA を有界なメモリしかない実行環境で実現できる Janus プログラムを作成した．また，クリーンな大域関数を定義し，大域関数による遷移をクリーンに実現する方法を用い，可逆スタックを用いた 1 次元 RCA のクリーン可逆シミュレーションを実現できるという知見を得た．

本論文で実現したのは 1 次元 RCA のシミュレーションであるが，今後はこれを 2 次元以上の RCA に拡張することを目標とする．

参考文献

[1] 森田憲一：一次元 2 近傍可逆的セル・オートマトンについて (アルゴリズムと計算量の理論), 数理解析研究所講究録, No.731, pp.108–117 (1990).

[2] 森田憲一:計算における可逆性, 情報処理, Vol.35, No.4, pp.306–314 (1994).

[3] 森田憲一：可逆セル・オートマトン, 情報処理, Vol.35, No.4, pp.315–321 (1994).

[4] 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, pp.87–187, 近代科学社 (2012).

[5] Yokoyama, T., Axelsen, H.B. and Glück, R.: Principles of a reversible programming language, *Proc. Computing frontiers* (*CF*'08), pp.43–54 (2008).

[6] Claus, S.N. and Michael, B: Janus Playground, DIKU(online), available from ⟨`http://topps.diku.dk/janus-playground/`⟩ (accessed 2013-11-17)