2次元可逆分割セルオートマトンの可逆プログラミング言語上でのクリーン可逆シミュレーション

M2018SE013 矢澤拓海

目次

- 研究背景
- 研究目的・研究課題
- 準備
- 関連研究
- 1次元可逆分割セルオートマトン(1D-RPCA)
- 2次元可逆分割セルオートマトン(2D-RPCA)
- 状相の表現の効率化
- 結論・今後の課題

研究背景

- セルオートマトン(CA)は様々なシミュレーションに用いられる計算モデル
  - 例：結晶の成長，交通モデル
- 可逆計算は様々な分野で活用
  - 計算に関わる消費エネルギーに深い関係
  - RTMなどの計算モデル
  - 観測以外の演算が可逆性をもつ量子計算
- CA + 可逆制約 = 可逆CA (RCA)
- 多数の可逆シミュレーション

研究背景

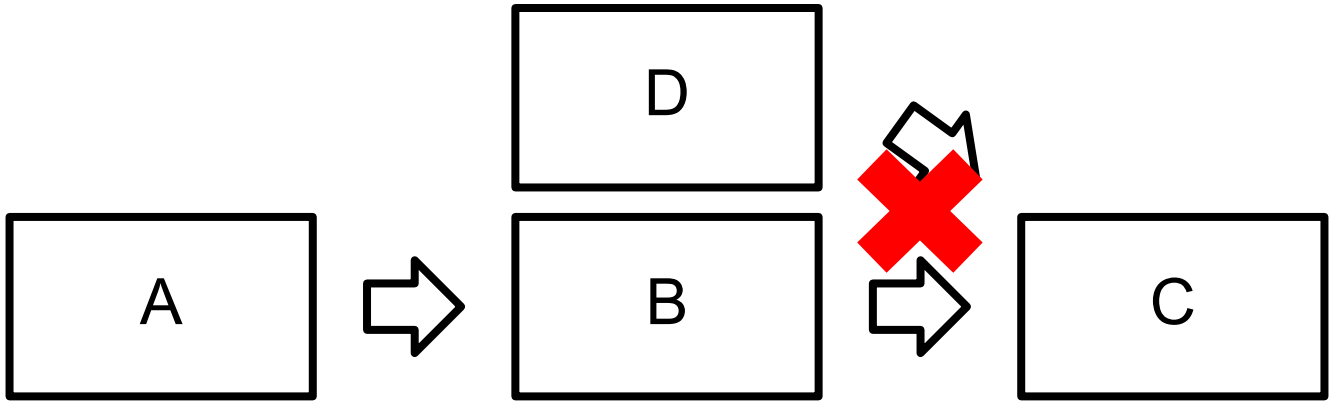
- 有限状相の1D-RPCAはJanus上で実現
  - 周期的境界という制約あり[Moriyama92]
  - 周期的境界という制約なし[渡邉13]
    - 実行時間に比例する非効率なメモリ使用量
  - [渡邉13]のメモリ使用量の効率化[木村,矢澤18]
- 有限状相の効率的な2D-RPCAは可逆言語では未実現
  - 一般解法は可能
    （実行時間に比例する非効率なメモリ使用量）

研究目的 と 研究課題

- 研究目的
  - 2D-RPCAの可逆言語上でのクリーンな実現
- 研究課題
  - 任意の2次元有限状相の表現
    - Janusの拡張
  - 2次元状相の表現の更新の単射的実現
    - 効率的な状相の表現

準備

可逆計算:
全ての時間の状態から直前の状態が
たかだか一つに特定できるような計算



可逆セルオートマトン

- セルオートマトン(CA)
  - 有限オートマトンを規則的に配置・接続した計算システム
- 可逆セルオートマトン(RCA)
  - 状相の遷移を行う大域関数に単射性の制約を与えたCA

可逆分割セルオートマトン

- 分割されたセルを持つCA
- セルの遷移を行う局所関数が単射なので大域関数が単射
- プログラムでは有限状相のみ実現

可逆プログラミング言語Janus[Lutz86][Y+08]

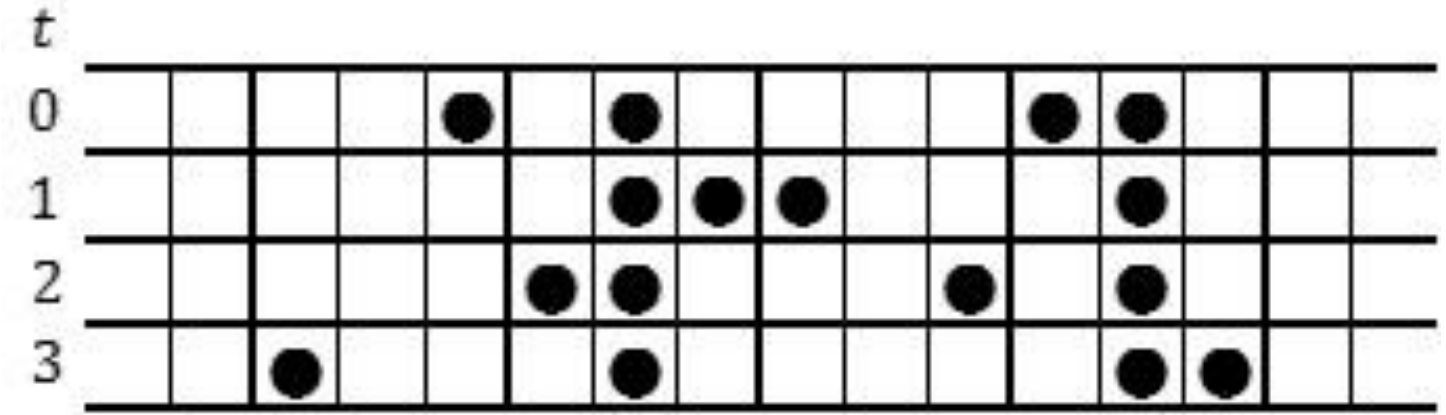
- Cに似た構文を持つ手続き型言語
- プロシージャの逆呼び出しが可能
- スタックを用いたJanusはr-Turing完全
- 可逆性を保つための一部特殊な構文
  - 代入
  - 条件分岐
  - 繰返し

関連研究

- 周期的境界条件を持つ1D-RPCAのシミュレーション
  - 有限の大きさの状相を表現
- 非周期的境界条件を持つ1D-RPCAのシミュレーション
  - 無限長の状相を表現
  - 状相の遷移の度にスタックが拡大
- 1D-RPCAの効率的なシミュレーション

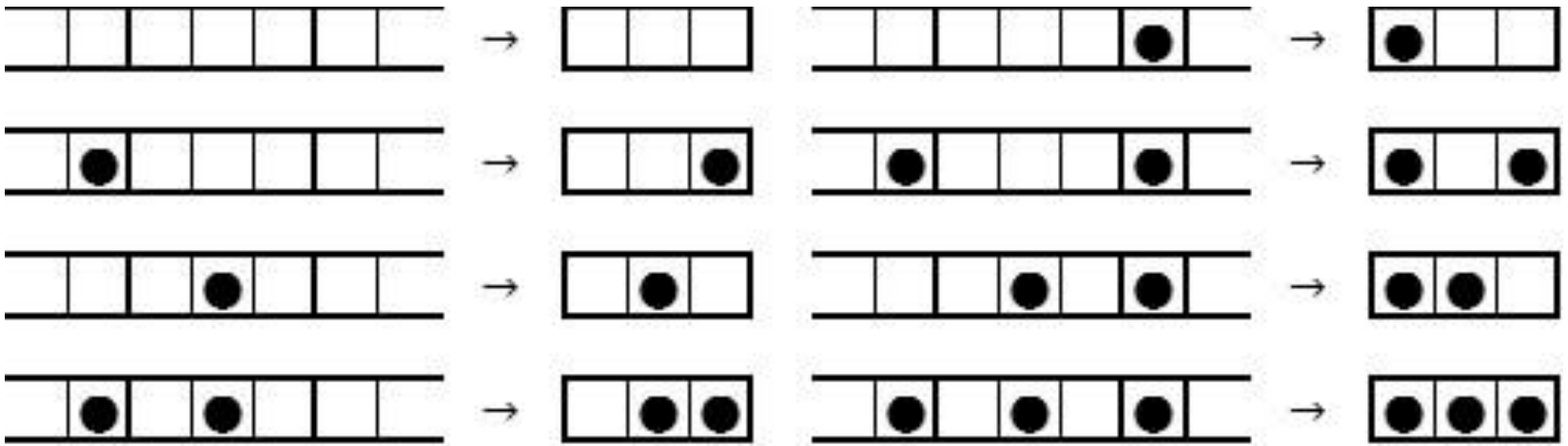
1次元可逆分割セルオートマトン[森田12]の実現

- 無限に広がる状相を表現
- 状相の表現の最適化



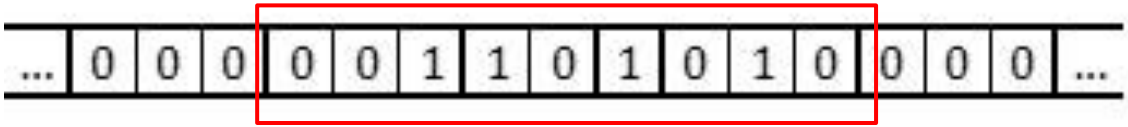
1次元可逆分割セルオートマトンの実現

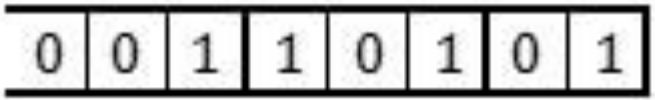
局所関数



1次元可逆分割セルオートマトンの実現

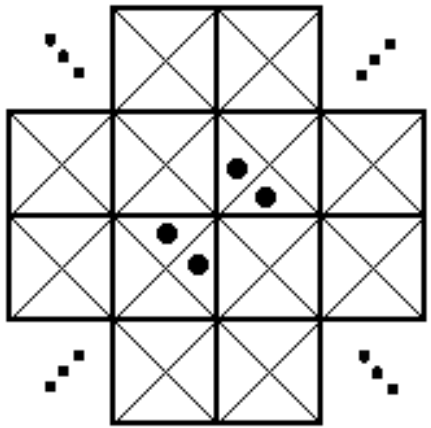
- 分割位置を保ち，静止状態の数が少ない表現

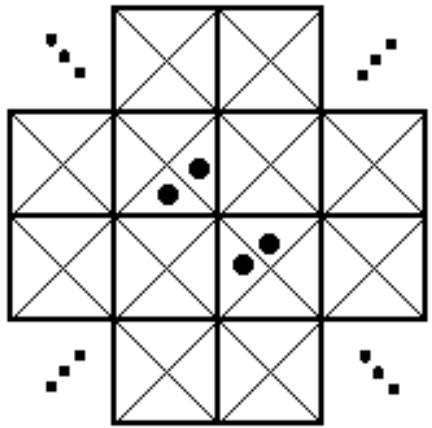




2次元可逆分割セルオートマトン[森田12]の実現

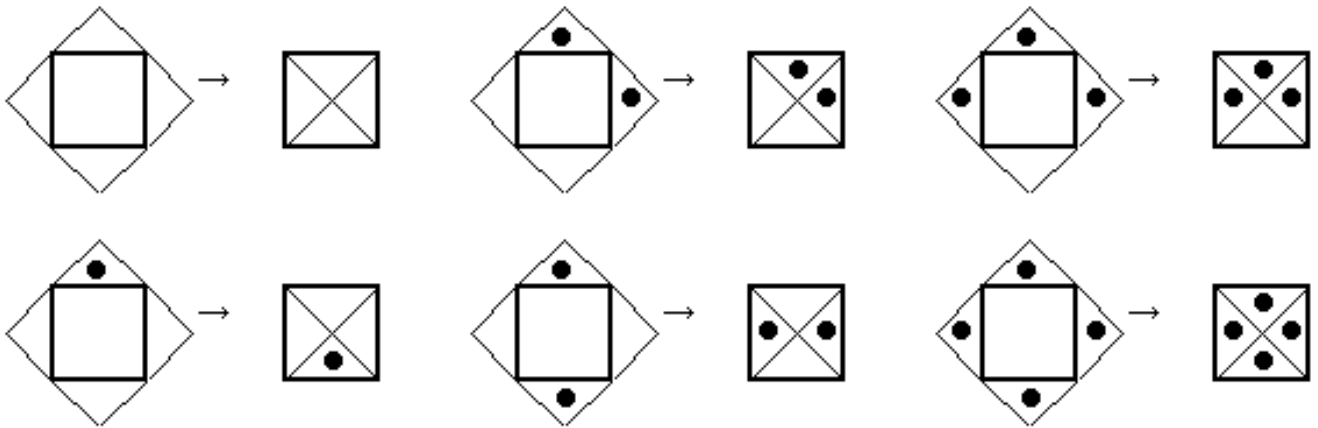
可逆動的表を用いて実現





2次元可逆分割セルオートマトン

遷移規則



大域関数のプログラム

```
procedure global_map(int sr[][], int rule[][])
    local int sl[size(sr) + 2][size(sr) + 2] = {{0}},
        int u = 0, int l = 0, int r = 0, int d = 0
    call table_expand(sr)
    iterate int y = 0 to size(sl) - 1
        iterate int x = 0 to size(sl) - 1
            call get_ulrd(u,l,r,d,sr,x,y)
            local int t = 0
                call local_map(t, u, l, r, d, rule)
                t <=> sl[x][y]
            delocal int t = 0
        end
    end
    call move2(sl, sr)
    delocal int sl[size(sr)][size(sr)] = {{0}}, int u
        = 0, int l = 0, int r = 0, int d = 0
```

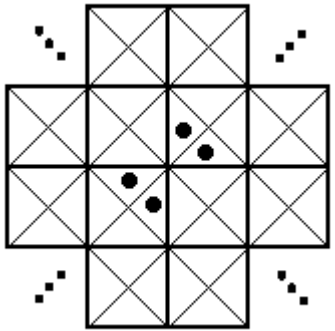
配列を拡張するプログラム

配列の解放・割り付けが独立

```
procedure table_expand(int t[][])
  local int t2[size(t)+2][size(t)+2] = {{0}}
  call move(t, t2)    // zero clear t
  delocal int t[size(t2)-2][size(t2)-2] = {{0}}
  local int t[size(t2)][size(t2)] = {{0}}
  call swap(t2, t)    // zero clear t2
  delocal int t2[size(t)][size(t)] = {{0}}
```

効率的な状相の表現

- 同一の状相の異なる表現を別のものとして扱う
- 遷移の際に必ず1だけ拡大

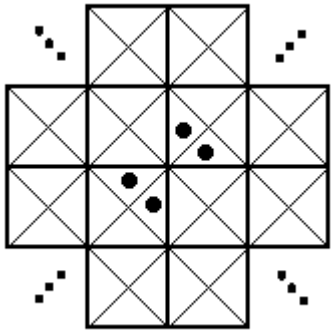


```
{{0, 12},
 3, 0}}
```

```
{{0, 0, 0, 0},
 {0, 0, 12, 0},
 {0, 3, 0, 0},
 {0, 0, 0, 0}}
```

効率的な状相の表現

- 行・列それぞれの先頭と末尾に静止状態以外を格納
- 整形な配列



```
{{0, 12},
 3, 0}}
```

```
{{0, 0, 0, 0},
 {0, 0, 12, 0},
 {0, 3, 0, 0},
 {0, 0, 0, 0}}
```

状相の表現の効率化の考察

- 状相の表現の最適化は非単射
  - ゴミ出力を伴えば実装可能
- 入力消去可逆計算[Bennett89]を応用
  - 単射な計算ならばゴミ出力と入力を消去可能
- 定義より2D-PRCAの状相の遷移は単射

状相の表現の効率化の考察

- 効率のいい遷移前後の状相をそれぞれ α_1, α_2
- 効率の悪い遷移前後の状相をそれぞれ β_1, β_2
- プログラムは α_1 から β_2 を出力
- β_2からゴミ出力を伴い β_1を出力
- α_1から β_1の遷移は単射
- 入力消去可逆計算でゴミ出力と入力を消去可能

結論

- 2D-RPCAの可逆言語上での実現
  - 欠点: 遷移ごとに静止状態を表すセルの増加
  - r-Turing完全性の別証明
- 任意の有限状相の配列を用いた表現
  - 任意の効率的な有限状相の表現を実現
- Janusの構文と意味論の拡張:
  可逆動的表，割付/解放の独立化
- 2次元状相の表現の遷移の単射的実現
  - 欠点: 同一状相の複数表現

今後の課題

- 提案した 有限状相の効率的な表現 の実装
  - 入力消去可逆シミュレーション[Bennett89]を応用
- 1パスでクリーン可逆シミュレーションの実現

参考文献

[1] Moriyama, K.: Reversible Cellular Automata from a Programming Language Perspective, Master's thesis, DIKU, University of Copenhagen (2010).

[2] 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2013年度卒業論文(2014).

[3] 木村孝大，矢澤拓海：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，南山大学2017年度卒業論文(2018).

参考文献

[4]Lutz, C.: Janus: A time-reversible language, Letter to R. Landauer (1986).

[5]Yokoyama, T., Axelsen, H.B. and Gl¨uck, R.: Principles of a Reversible Programming Language, Proc.CF, ACM Press, pp.43–54 (2008).

[6] 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)

[7]Bennett, C.H.: Time/space trade-offs for reversible computation, Vol.18, No.4, pp.766–776 (1989).