10. セルオートマトン

2016/12/12

1　準備

講義で説明した、一次元セルオートマトンのシミュレーションを行います。その準備として新たにプロジェクト CA を作成します。以下の URL から Java ソースファイルを取得し、解凍した後、プロジェクト CA のソースファイルとして置きます。

http://http://aoba.cc.saga-u.ac.jp/lecture
/ModelingAndSimulation/javasrc/CA/src.zip

また、ライブラリ MyLib も設定しましょう。

2　クラス CA

2.1　状態更新規則

ca パッケージ中のクラス CA は、一次元セルオートマトンを表すクラスです。周期境界条件が設定されています。セルの状態を表す配列が、cells です。コンストラクタを表 1、メソッドを表 2 に示します。

時間発展は、3 個の連続するセルの状態に対して、中央のセルの次の時刻での状態を対応させることで決定します。3 個の隣接するセルの状態は 3 ビットで表現されるため、全部で 8 通りです。この対応関係を表す配列が、ruleMap です。

表 1: CA クラスのコンストラクタ

コンストラクタと説明
CA(int n, int rule) セルの総数 n とルール番号 rule を与えて初期化する。

課題 1　その 8 通りの状態に 0 または 1 を割り当てることでルールが定められます。つまり、ルールは 2^8 通り可能です。ルール番号 rule を二進数とし、各ビット

の値を newRuleMap という配列に割り当てるメソッドが newRuleMap() です。この戻り値を ruleMap に代入します。

10 進数で与えられた rule の二進数表示の最下位ビットの値を newRuleMap[0]、次のビットの値を newRuleMap[1] と保存し、再上位ビットの値を newRuleMap[7] に保存します。この内容を記述しましょう。

課題 2　クラス CA を実行すると、rule-184 の場合について、ruleMap の内容を印刷します。実行して、正しい内容であることを確認しましょう。また、rule-90 の場合に変更し、確認しましょう。

表 2: CA クラスのメソッド

修飾子と型	メソッドと説明
int[]	getCells() 現在のセルの状態を返す。
int	getN() セルの総数を返す。
int	getNumDifference() 状態更新によって値の変化したセルの数を返す。
int	getNumR() 初期化の際、値が 1 であったセルの数を返す。
int[]	getRuleMap() ruleMap を返す。
void	initialize() 確率 1/2 で、各セルの値を 1 とし、残りを 0 と初期化する。
void	initialize(double r) 確率 r で、各セルの値を 1 とし、残りを 0 と初期化する。
void	initializeSingle() 中央のセルの値を 1 とし、残りを 0 と初期化する。
static int[]	mkruleMap(int r) 与えられた整数の各ビット値を長さ 8 の配列に格納する。
static String	ruleMap2Int(int ruleMapDummy[]) ra[] を整数として返す。
static String	ruleMap2String(int ruleMapDummy[]) ra[] を文字列として返す。
void	setRule(int rule) ルール番号 rule を設定する。private void mkruleMap() で、対応する配列 ruleMap へ変換する。
String	state2String() セルの状態を文字列として返す。
int[]	update() 状態を更新し、最新のセルの状態を返す。

2.2 状態更新

次に、セルの状態更新を考えます。メソッド `public int[] update()` が状態更新のメソッドです。戻り値は、次の時刻の状態を表します。

セルの状態は配列 `cells` に保存されています。状態更新はすべてのセルに対して同時に行われるようにしなければなりません。もしも、セルの状態を端から更新すると、同時に更新した結果と異なってしまいます。そこで、次の時刻の状態を配列 `cells` に直接には書き込まず、ダミー配列 `cellsDummy` に一旦保存します。すべてのセルに対して計算が終わった後に、ダミー配列 `cellsDummy` から配列 `cells` に書き戻すことにします。

各セル毎に次の状態を決めるには、その両隣のセルの状態を知る必要があります。注目しているセルの番号を i とすると、$i \pm 1$ のセルの状態が必要です。システムは周期境界条件を有する、つまり輪のようにつながっていると考えていますから、セルの総数を n とすると、$i = 0$ の左隣は $n-1$ 番のセル、$i = n-1$ のセルの右隣は 0 番のセルとなります。そこで、セル i の左隣のセルの番号を l、右隣のセルの番号を r とすると、次式で表すことができます。

$$l = (i - 1 + n) \bmod n \tag{2.1}$$

$$r = (i + 1) \bmod n \tag{2.2}$$

ここで $\bmod n$ は n で除した余りを意味します。

課題 3　例えば $n = 100$ の場合、$i = 0$ 及び $i = 50$ の両隣のセルが、式 (2.1) 及び (2.2) で求められることを確認しましょう。

課題 4　メソッド `update` の部分を作成しましょう。

3　シミュレーション実行

デフォルトパッケージ (パッケージに入らない部分) にあるクラス `CLIMain` は、rule-90 の場合について、実行すると結果をファイルに出力します。

課題 5　挙動が予想されているものか、確認しましょう。

課題 6　`gui` パッケージ中のクラス `CAMain` を実行すると、GUI が開きます。他のルール、例えば講義で扱ったルールでの挙動を観察しましょう。

4　単体テスト

これまで、動作の確認は結果の目視によって行ってきました。しかし、動作が正しいことを確実に確認するためには、体系的な確認方法が必要です。その一つが、単体テストです。

単体テストは、関数などのプログラムの構成要素を、単体として動作を確認するテストです。Java では、JUnit というツールによって、メソッドの動作を確かめることができます。現在のバージョンは 4.12 です。

今回はクラス CA の一部のメソッドの単体テストを実行しましょう。

1. 「プロジェクト」を表示し、クラス CA にマウスを合わせて、右ボタンを押します。
2. 現れたメニューから「ツール」→「テストの作成／更新」を選びます。
3. どの部分コードを生成するかを尋ねる画面が出ますが、今回はそのまま「OK」を押します。
4. 「テストパッケージ」中にクラス ca.CATest が生成されます。

いくつかのメソッドが自動で生成されています。

- setUpClass() メソッドは、このクラス内のテストの一番最初が実行される前に、一度だけ呼ばれます。
- tearDownClass() メソッドは、このクラス内のテストの一番再度が実行された後に、一度だけ呼ばれます。
- setUp() メソッドは、このクラス内の各テストが実行される前に呼ばれます。
- tearDown() メソッドは、このクラス内の各テストが実行された後に呼ばれます。

今回は、特に行うべきことがないので、上記の各メソッドはそのままにしておきます。この後ろに test で始まる、テストメソッドが続いています。

今回は mkRuleMap をテストする testMkruleMap() だけを残して、他を削除してください。また、このメソッド内の fail() メソッドも消してください。なお、この fail() メソッドは、テストを必ず失敗させるもので、テストメソッドを実装せずにおくことを防ぐ目的で置かれています。

testMkRuleMap() の中の assertArrayEquals(expResult, result) に注目します。このメソッドは mkRuleMap() によって得られた結果の配列 result と期待される結果の配列 expResult を比較するメソッドです。

課題 7　rule-90 の場合をテストするように、結果として期待される配列 expResult を定義し、testMkRuleMap() 内を記述しなさい。
テストの実行には、クラス ca.CATest を実行します。

クラス CA には、ルールを表す配列を整数で表すメソッド ruleMap2Int があります。これを用いれば、整数で表されたルールを mkRuleMap を用いて配列へ一旦変換し、その配列を ruleMap2Int を用いて整数へ戻すことで、もとの整数と等しいことを確かめることで正しい動作であることをテストすることができます。

課題 8　整数で表されたルールが正しく変換されていることを確かめる、上述のテストに対応した新しいテストメソッド testRuleConv を作成しなさい。ルールの整数は 1 から 254 まで変化させ、すべての場合について確かめなさい。なお、整数型の結果 result と期待する結果 expResult を比較するには、メソッド assertEquals(expResult,result) を用います。