社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2007－MPS－67 (24)
2007－BIO－11 (24)
2007／12／21

人工化学のための自動推論器の構築

小泉 和真† 冨永 和人†

† 東京工科大学大学院 バイオ・情報メディア研究科

人工化学とは仮想的な化学系を表現する計算モデルである．我々は文字列のパターンマッチと組み換えに基づく人工化学を提案している．人工化学を用いた一般的な研究では，設定をシミュレータに与えて計算機実験を行い，その結果を得る．これに対して我々は，とある結果を引き起こす設定がいかなるものであるか推論する手法の確立を目指している．本研究では，この目的のために自動推論器の試作を行なった．この推論器は，人工化学系の初期状態と目的の分子を与えると，その分子の生成可能性を判定する．推論器の実装にはオブジェクト指向言語 Ruby を用いた．作成した推論器によって例題を解き，推論器が期待通りに動作することを確認した．

Construction of a Prototypical Reasoning Software for an Artificial Chemistry

Kazumasa Koizumi† and Kazuto Tominaga†

† Graduate School of Bionics, Computer and Media Sciences, Tokyo University of Technology

Artificial chemistries are models of virtual chemical systems. We proposed an artificial chemistry based on string pattern matching and recombination. A general research methodology using an artificial chemistry is to give a set-up to a simulator and run a simulation to obtain results. In contrast, we attempt to establish a methodology to infer a set-up to cause a desired result. This study built a prototypical reasoning software, which decides whether a target molecules are produced in a given system of our artificial chemistry, implemented with Ruby, an object-oriented language. We solved example problems with the software, thereby confirmed that it worked as expected.

1 はじめに

人工生命は生命現象をコンピュータを用いて研究する研究分野である．その主な手段のひとつは，人工的なシステムで生命現象におけるさまざまな振る舞いを表現することである．人工生命研究の一手法として人工化学がある[1]．人工化学とは仮想的な化学系を表現する計算モデルである．

人工化学を用いた一般的な研究手法では，設定をシミュレータに与えて計算機実験を行い，結果を得る．これに対して我々は，とある結果を引き起こす設定がいかなるものであるのか推論する方法論を確立することを目的として研究を行っている．我々はこれまでに，そのための推論アルゴリズムを与えた[2]．このアルゴリズムは，系における分子の生成可能性を判定する．本研究では，このアルゴリズムに基づく自動推論ソフトウェアを作成する．実装にはオブジェクト指向言語 Ruby を用いる．

2 文字列のパターンマッチと組み換えに基づく人工化学

我々の研究室では，文字列のパターンマッチと組み換えに基づく人工化学を提案している[3]．本節ではこの人工化学を簡単に説明する．

原体と分体 原体は自然界の原子に相当する．原体は A や Bc など大文字の英字で始まる英数字列で表現する．分体は自然界の分子に相当する．分体は原体の列のひとつ以上の重なりである．分体は O#ABc/1#DE/ のように表記する．

組み換え規則と鬼札 組み換え規則とは，分体間の反応を表わす式である．組み換え規則の例を式 (1) に示す．

$$\mathtt{O\#A/} + \mathtt{O\#B/} \rightarrow \mathtt{O\#AB/} \quad (1)$$

式 (1) は，O#A/ と O#B/ が結合して O#AB/ になることを表わしている．組み換え規則の記述に

鬼札 (wildcard) を用いることができる．我々の人工化学には2種類の鬼札がある．ひとつは <1> のように表記され，任意の1個の原体と適合し，もうひとつは <1*> や <*2> と表記され，任意の0個以上の原体と適合する．鬼札を含む組み換え規則の例を式 (2) に示す．

```
0#<*1><2>C/ + 0#D<3><4*>/ →
        0#<*1><2>C/0#D<3><4*>/        (2)
```

式 (2) は，図1のような分体間の反応を表わす組み換え規則である．

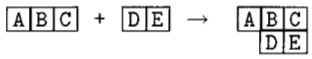

図 1: 分体の組み換え

図1の反応では，原体鬼札 <2> が原体 B に，原体鬼札 <3> が原体 E に適合している．また，列鬼札 <*1> は原体 A に適合している．列鬼札 <4*> は0個の原体に適合している．

系　系は三つ組み $\langle \Sigma, R, P_0 \rangle$ である．ここで Σ は原体の集合，R は組み換え規則の集合である．P_0 は分体の多重集合であり，系の初期状態である．

動作　この人工化学の系は以下のように動作する．

1. 分体の作業用多重集合を P_0 で初期化する．
2. 分体の作業用多重集合の中の分体に，R に含まれる組み換え規則のひとつを適用する．
3. 2へ戻る．

3　自動推論器

本節では，本研究で構築した自動推論器について説明する．この自動推論器は，系の初期状態と目的の分体を与えると，その分体が生成可能であるかどうかを答える．

3.1　推論アルゴリズム

この自動推論器は我々の研究室で考案したアルゴリズム[2]に基づく．その概略は以下の通りである．ここで，アルゴリズム中の「世界」とは，生成できる分体の多重集合と，生成すべき分体の多重集合の対である．このアルゴリズムはバックトラックを用いる．

1. 生成したい分体群が，生成できる分体群の部分多重集合であるか確認する．そうであれば「生成可能」を出力して終了する．
2. 分体と組み換え規則を選び，その分体に組み換え規則の逆適用を行い，現在の世界の一つ前の状態を求める．
3. ステップ1へ戻る．

全ての生成可能性を試して生成可能でなければ，「生成不可能」を出力して終了する．

このアルゴリズムは健全ではあるが完全ではない．すなわち，分子が実際には生成可能であっても，生成不可能と判定することがある．

3.2　プログラムの設計と実装

ここでは，前節に述べた推論アルゴリズムに基づく自動推論器の設計と実装について述べる．この自動推論器は，再帰的呼び出しを用いて逆適用などを繰り返し，解を探す．

世界を表現する多重集合変数を表1に示す．$\langle W, H \rangle$ が世界を表わす．このプログラムは組み換え規則の逆適用を行う．この処理は組み換え規則の右辺の項と W にある分体のマッチングを行う．そして，マッチする分体があれば組み換え規則を逆適用し，ひとつ前の状態を求める．この時，組み換え規則の右辺の項に鬼札があれば分体の各行ごとに以下の処理を行う．

表 1: 多重集合変数

多重集合変数	格納する多重集合
W	生成すべき分体
H	生成できる分体

1. 組み換え規則の右辺の項を左列鬼札 l，右列鬼札 r，その2つに挟まれた部分 m に分ける．
2. W にある分体と m のマッチングを行う．この時，m に原体鬼札があればその鬼札と原体のマッチングも行う．
3. ステップ2で，m とマッチしなかった部分に対して列鬼札 l と r のマッチングを行う．

全ての行でマッチングが成功したら逆適用を行う．

その後，W と H の比較を行い，双方に同じ分体があるとき相殺を行う．

このプログラムは逆適用と相殺を繰り返すことにより推論を行う．

3.3 実行例

ここでは，作成した自動推論器の実行例を示す．例として，以下に示す系において，分体 O#ABCD/ が生成できるかを推論する．

- 分体の初期多重集合
  O#A/ 100 個
  O#B/ 100 個
  O#C/ 100 個
  O#D/ 100 個
- 組み換え規則の集合

$$\texttt{O\#<1>/ + O\#<2>/} \rightarrow \texttt{O\#<1><2>/} \quad (3)$$

$$\texttt{O\#<*1><2>B/ + O\#C<3><4*>/} \rightarrow \texttt{O\#<*1><2>BC<3><4*>/} \quad (4)$$

この系では，分体 O#A/ と O#B/ に組み換え規則 (3) が適用され，分体 O#AB/ が生成される．同様に，分体 O#C/ と O#D/ に (3) が適用され，O#CD/ が生成される．組み換え規則 (4) は末尾が B である長さ 2 以上の分体と，先頭が C である長さ 2 以上の分体を結合する．この規則によって分体 O#ABCD/ が生成される．図 2 はこの例を自動推論器に与えて推論を行ったときの実行画面である．この図では，推論器は最後に成功と出力して終了している．図 3 はこの例における分体の初期多重集合から分体 O#A/ を取り除いて自動推論器に与えて推論を行った場合の実行画面である．この図では，推論器は最後に失敗と出力して終了している．以上の動作は期待した通りのものである．

図 2 の推論に対応する探索木を図 4 に示す．図中の枝に添えてある規則の番号や「相殺」は，そこで行った処理である．例えば「規則 (3)」は規則 (3) の逆適用を表す．この自動推論器は深さ優先探索を行う．この例ではバックトラックが起きずに正しい経路を推論器が一度で選択した．しかし，一般には必ず正しい経路をとるとは限らない．

4 議論

本研究では，自動推論器の実装にオブジェクト指向言語 Ruby を用いた．計算機で推論を行う際には論理型言語を用いるのが一般的である．論理型言語のひとつに Prolog がある [4]．我々の人工

図 2: 実行画面 1

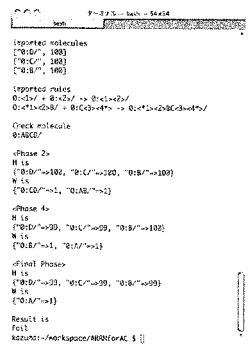

図 3: 実行画面 2

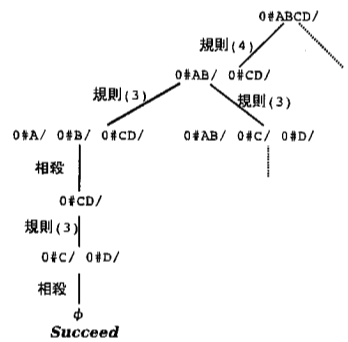

図 4: 探索木

化学の組み換え規則を Prolog の節として表現すると，分体の個数を自然に扱うことができない．例えば，分体 A と B から分体 ABAB を生成する系を図 5 のように書くことを考える．このプログラムを Prolog の処理系で実行し ?- mol(abab). と質問すると mol(abab) は真であることがわかる．しかし実際には，この系では分体 A と B が一つずつ足りず，分体 ABAB を生成できない．これは，mol(a) や mol(ab) という項が A や AB などの個数を表現していないためである．

Prolog に線形論理 [5] を導入した論理型言語に LLP がある [6]．LLP を用いれば分体を項として表現することでその個数を自然に扱うことができよう．しかし本研究の応用では，線形論理がもつ表現力の高い多くの演算子は必要なく，それよりも鬼札の処理のために様々な文字列操作が必要である．このため，LLP を採用せず，文字列を容易に扱える言語である Ruby を用いた．

```
/*molecule*/
mol(a).
mol(b).
/*rule*/
mol(ab):-mol(a),mol(b).
mol(abab):-mol(ab),mol(ab).
```

図 5: Prolog での記述

5 おわりに

本研究では，我々の人工化学のための試作版自動推論器を構築した．この自動推論器はオブジェクト指向言語 Ruby で実装した．そして，作成した推論器によって例題を解き，推論器が期待通りに動作することを確認した．今後の課題としては，推論の効率を上げるために有効な探索戦略を考える，また鬼札の表現力を拡張して探索空間を狭めることなどがある．

参考文献

1) P. Dittrich, J. Ziegler, and W. Banzhaf. Artificial chemistries – a review. *Artificial Life*, Vol. 7, No. 3, pp. 225–275, 2001.

2) Kazuto Tominaga. An approach to reasoning in an artificial chemistry. In Hamid R. Arabnia, editor, *Proceedings of the 2006 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'06)*, Vol. II, pp. 839–845, USA, 2006. CSREA Press.

3) Kazuto Tominaga, Tooru Watanabe, Keiji Kobayashi, Masaki Nakamura, Koji Kishi, and Mitsuyoshi Kazuno. Modeling molecular computing systems by an artificial chemistry - its expressive power and application. *Artificial Life*, Vol. 13, pp. 223–247, 2007.

4) I. Bratko. Prolog への入門. 近代科学社, 1990. 安部 憲広 訳.

5) 竹内外史. 線形論理入門. 日本評論社, 1995.

6) 田村直之, 番原睦則. LLP: A linear logic programming language and its compiler system. http://bach.istc.kobe-u.ac.jp/llp/index-jp.html.