Metacat: A Program That Judges Creative Analogies in a Microworld

James B. Marshall: Metacat: A Program That Judges Creative Analogies in a Microworld, Proceedings of the Workshop on Creative Systems: Approaches to Creativity in Artificial Intelligence and Cognitive Science, ECAI2002, pp.77-84 (2002).

Metacat は，プログラムがユーザから提案された類推も含めた理想的な類推を比較・対照することができるメカニズムを組み込むことで，初期の Copycat モデルを拡張した類推作成と創造性のモデルである．これにより，類推間の類推を作ることにもなる．本稿ではプログラム構成の概要，および実行可能な比較の例をいくつか示す．

1 INTRODUCTION

□ Copycat と Metacat

- Copycat の拡張 [Hofstadter 1994,1995; Mitchell 1993]
  - 自身をモニターするメカニズム [Marshall 1999]
  - 回答到達の理解・類推の比較と対照
- 類似モデル：Letter Spirit（文字魂？） [McGraw 1995]
  - 視覚的文字認識とデザイン
  - 格子内の小文字の認識と分類，新しい文字のデザイン
  - 「創造性の central feedback loop」
    :作成した文字形式の質を判断
    :生成・評価・修正の連続サイクル
- Copycat・Metacat の対象領域（四項類推課題）
  - 実問題の本質側面のみ切り出した小規模・理想的問題領域
  - 構成要素：アルファベット小文字 + 同一・前後関係
  - 例：「abc ⇒ abd; mrrjjj ⇒？」
  - 充分な回答の可能性，創造的・意外な回答の存在
- Copycat の構成（詳細略）
  - Slipnet（明細書網？）：文字列の概念の長期記憶
  - 作業空間：知覚の短期記憶
  - Codelet（小符号？）：問題の文字を調査，解釈の構造を作成するエージェント
    :調査する文字間関係：同一・後続・先行
    :チャンク（例：mrrjjj の j から「同一グループ」，abc から「先行グループ」）
- Copycat の動作
  - Codelet のボトムアップ的集合動作 → 高次動作（中央処理無）
  - 作業空間内の活動から Codelet が活性
  - Slipnet 内の概念を探索
  - 良好な問題解釈をガイド

2 THE ARCHITECTURE OF METACAT

- Metacat の構成
  - Copycat（Slipnet + 作業空間 + Codelet）
  - 新コンポーネント（エピソード記憶, テーマ空間, 一時追跡）

2.1 The Episodic Memory

- 問題解決経験の保持
  - 回答記述のパック：作業空間内の構造＋問題解釈
  - 将来の問題解決時に利用
  - 自己増殖・回答に関わる情報も増加
- テーマ
  - 回答記述内の最重要情報：回答下のキーとなるアイデア
  - 回答間の比較・対照，類似問題の検索キュー
  - 検索・利用のための特徴 (index)

2.2 The Themespace

- テーマ生成

- テーマ
  - Slipnet の概念のペアから生成
    例：a と z の対称性＝「アルファベット位置」＋「反対」
  - 作業空間の構造の如し：作業空間内の構造構築から生成
  - Slipnet の概念の如し：問題解釈表現
  - 時間とともに崩壊，他のテーマに影響
  - テーマの活性レベル
    :正：作業空間内にテーマと同じ構造生成を促進
    :負：代替案を促進

2.3 The Temporal Trace

- 短期記憶
  - エピソード記憶・テーマ空間のための（作業空間と別の）記憶
  - 自己モニタリングの焦点
  - 1 回の実行を越えて情報蓄積（1 実行ごとにフラッシュされない）
  - 問題解決時の重要イベントを一時記録
    :例：重要な概念発見，障害発生（「abc → abd, xyz →?」の z とか），新しい回答発見
    :マクロ（回答発見レベル）からミクロ（Codelet レベル），閾値を越えるもののみ

- 回答到達の記録
  - 処理中に再処理されたイベントを回答記述に
  - 誘導類推チックな香り

3 ANSWER JUSTIFICATION

- （外部からの）類推の評価
  - プログラムの類推判断機能の利用
  - 後向き動作・過去の回答との比較

- 後付け的理解
  - 人間にとってはわりと簡単
    :希な回答「abc ⇒ abd, mrrjjj ⇒ mrrjjjj」，理解は容易
    :冗談っぽい回答「abc ⇒ abd, mrrjjj ⇒ abd」，笑える
  - 回答以外の追加説明は不要

- Metacat の「判断モード」
  - （ユーザからの）問題・解答の入力から解釈開始
  - 文字認識の構造を作成
  - テーマのパターンでアイデアを表現

- 例「abc ⇒ abd, xyz ⇒ wyz」
  - 最初に対応する文字をリンク
  - 「abc ⇒ abd」→「右端文字の文字カテゴリを後続に変える」
  - 「xyz ⇒ wyz」→「左端文字の文字カテゴリを先行に変える」（→ Figure1）
  ↓
  - 対応関係が不一致（c と x，c を後続と x を先行）
  - 対称性の提案（右端と左端，後続と先行）
    :テーマ「文字位置:逆, 方向:逆, グループタイプ:逆」等を利用
  - テーマ空間の活性テーマと関連付け・分岐探索
  - a-x・c-z の結合で終了（→ Figure2）
    :最終的に含まれるテーマ「アルファベット位置:逆」

4 ANSWER COMPARISON

- 回答と問題解釈（回答記述）
  - 問題「abc ⇒ abd, xyz ⇒ wyz」の例を思い出して頂いて …
    :abc と xyz は「逆方向」, 対称位置,「右端文字の文字カテゴリを後続に変える」の鏡像

- 別解「abc ⇒ abd, xyz ⇒ wyd」の例（→ Figure3）
  :abc と xyz は「同方向」, 同位置, 「右端文字を d に変える」

- 類題「rst ⇒ rsu, xyz ⇒?」
  - 回答 wyz ? (r-z の関係 ≠ a-z の対称性)
  - 先の問題：xyd<wyz, この類題：xyu>wyz
  - 2 題の別解「abc ⇒ abd, xyz ⇒ dyz」「rst ⇒ rsu, xyz ⇒ uyz」
  → 文字列世界のメタレベルの類推

- 類似点と相違点（→ Table1）
  - 回答 xyz と dyz の差= abc ⇒ abd を抽象的に／字義通り解釈
  - 「abc ⇒...」と「rst ⇒...」の wyz の差= dyz と uyz の差=対称性の有無

Table1. 6 つの回答と関連する回答記述（一部略）

問題/回答	テーマ	ルール
abc ⇒ abd, xyz ⇒ wyz	アルファベット位置:逆 文字列位置:逆	抽象
rst ⇒ rsu, xyz ⇒ wyz	文字列位置:逆	抽象
abc ⇒ abd, xyz ⇒ xyd	文字列位置:同	字義
rst ⇒ rsu, xyz ⇒ xyu	文字列位置:同	字義
abc ⇒ abd, xyz ⇒ dyz	アルファベット位置:逆 文字列位置:逆	字義
rst ⇒ rsu, xyz ⇒ uyz	文字列位置:逆	字義

- Metacat の回答比較
  - テーマ等の分析による理想的な類推の比較・対照（Copycat の拡張）
  - エピソードの検索（→コメント生成）

4.1 Program-Generated Commentary

- Metacat の解説生成
  - アイデア・問題の観察・試行の列 (train of thought)
  - 一時追跡のイベント記録の連鎖に相当
  - Codelet のメッセージ・「声に出して考える (thinking out loud)」
    :例：障害に当たった，解を見つけた，新しいアイデアと評価（→ Figure4）
  - テンプレートからの生成

- 意図
  - 実行下のイベント提示
  - 回答比較の理解容易な説明
    :例（→ Figure5,6,7,8）

よし,「abc」が「abd」に変わるなら,「xyz」は何に変わりますか？うーん…
ううおお，私は小さな問題において衝突したように思えます．文字 Z のカテゴリを変えるにも，xyz には次の文字がありません．
ううおお，私はまた小さな問題において衝突したように思えます．文字 Z のカテゴリを変えるにも，xyz には次の文字がありません．
「xyd」という答えを思いつきました．この答えはかなり平凡だと思います．
この答えは問題「rst → rsu, xyz →?」の答え「xyu」を大いに思い起こさせます．
「xyz」という答えも思いつきましたが，これはとても悪いです．
ええとですね…
問題「abc → abd, xyz →?」の答え xyz と xyd の本質的な違いは，abc から abd への変化は，答え xyz のほうが xyd の場合より，より字義通り考えた方法と見ることができます．どちらの答えも（abc と xyz 両方の場合で），2 つの文字列を同じタイプのグループが同じ方向に行くものと見ることができるかに依存します．総じて，私は xyd がより良い答えと言うでしょうが，abc から abd への変化をより抽象的な方法で見ることを含むからです．

Figure4 の直訳 「abc ⇒ abd; xyz ⇒?」で xyd と xyz の回答を見つけた実行に対する Metacat の解説

問題「abc ⇒ abd; xyz ⇒?」の答え wxy は，abc と xyz を反対側に向かう対称的な先行-後続であり，かつアルファベット位置が文字列間で対称のグループとして見ることに基づいていますが，回答 xyd は abc と xyz を同じ方向に向かう同じタイプのグループとして見ることに基づいています．xyd の場合，abc と xyz をアルファベット位置が対称という考え方は出てきません．他に，この回答の間の重要な違いとして，abc から abd への変化を，回答が xyz の場合は xyd の場合と比べ，より抽象的な方法で見ていることがあります．総じて，より多くのアイデアに基づいているので，xyz のほうがより良いアイデアであると言えるでしょう．

Figure5 の直訳 「abc ⇒ abd; xyz ⇒ xyd」対「abc ⇒ abd; xyz ⇒ wyz」

問題「abc ⇒ abd; xyz ⇒?」の答え xyd は，問題「rst ⇒ rsu; xyz ⇒?」の答え xyu と本質的には同じです．どちらの答えも（abc と xyz の場合と rst と xyz の場合で）2 つの文字列を同じタイプのグループが同じ方向に行くものと見ることができるかに依存します．さらに，abc から abd への変化は本質的に rst から rsu への変化と同じ方法で見ています．総じて，どちらも平凡な回答と言えるでしょう．

Figure6 の直訳「abc ⇒ abd; xyz ⇒ xyd」対「rst ⇒ rsu; xyz ⇒ xyu」

問題「abc ⇒ abd; xyz ⇒?」の答え wyz は，abc と xyz の間のアルファベット位置の対称性を見ることに幾らか基づいています．これに対し，問題「rst ⇒ rsu; xyz ⇒?」の答え wyz の場合は，rst と xyz の間のアルファベット位置の対称性というアイデアは出てきません．総じて，最初の wyz のほうが良い回答と言えるでしょうが，より豊かなアイデアに基づくからです．

Figure7 の直訳 「abc ⇒ abd; xyz ⇒ wyz」対「rst ⇒ rsu; xyz ⇒ wyz」

問題「abc ⇒ abd; xyz ⇒?」の回答 dyz は，abc と xyz の間のアルファベット位置の対称性を見ることに幾らか基づいています．これに対し，問題「rst ⇒ rsu;xyz ⇒?」の答え wyz の場合は，rst と xyz の間のアルファベット位置の対称性というアイデアは出てきません．しかし，回答 dyz は abc と xyz の間の抽象的な類似性を（abc と xyz を反対側に向かう対称的な先行-後続であり，かつアルファベット位置が文字列間で対称のグループとして）見ることを含みますが，同時に abc から abd への変化をより字義的な方法で見ているので，一貫性がないように思えます．回答 uyz も rst と xyz の間の抽象的な類似性を（abc と xyz を反対側に向かう対称的な先行-後続であり，かつアルファベット位置が文字列間で対称のグループとして）見ることを含みますが，同時に rst から rsu への変化をより字義的な方法で見ているので，一貫性がないように思えます．しかし，総じて uyz のほうが良い回答と言えるでしょうが，dyz ほど一貫性がないとは思えないからです．

Figure8 の直訳 「abc ⇒ abd; xyz ⇒ dyz」対「rst ⇒ rsu; xyz ⇒ uyz」

4.2 Reminding

- 記憶検索
  - 新しい回答発見・回答記述による検索
  - 回答記述のテーマ・概念から距離（類似度）算出
  - 閾値を越える回答を想起
  - 「障害 (snag)」: 障害記述の作成・記憶保存，回答比較に利用
    （→ Figure9 :「rst ⇒ rsu; xyz ⇒?」で wyz を見つけた場合の想起の例）

- 障害記述の利用
  - ある回答のより良い理解
  - 例：実験「eqe ⇒ qeq; abbbc ⇒?」
    :創造的な回答例 :「文字カテゴリ（abc）→文字数（1-3-1）」⇒「aaabccc」
    1.Metacat を初期化,「eqe ⇒ qeq; abbba ⇒ aaabaaa」を見せる
    2.「eqe ⇒ qeq; abbbc ⇒ aaabccc」を見せる（→ Figure10）
    3. 再度初期化,「eqe ⇒ qeq; abbbc ⇒?」を解かせる（障害発生・失敗して qeeeq を回答）
    4.「eqe ⇒ qeq; abbba ⇒ aaabaaa」を見せる（→ Figure11）

問題「eqe ⇒ qeq; abbba ⇒?」の回答 aaabaaa は，問題「eqe ⇒ qeq; abbbc ⇒?」の回答 aaabccc の回答と本質的に同じです．どちらの答えも 2 つの文字列（eqe と abbba と，eqe と abbbc）を同じ方向に向かい，一方の文字列は文字の，他方は数の問題になっていると見ることに依存します（しかし，そう見る良い理由はありません）．さらに，eqe から qeq への変化はどちらの場合も本質的に同じです．総じて，aaabccc は半分まともで，aaabccc はとても良いと言えます．

Figure10 の直訳 障害に当たる前の aaabaaa 対 aaabccc

問題「eqe ⇒ qeq; abbba ⇒?」の回答 aaabaaa は，問題「eqe ⇒ qeq; abbbc ⇒?」の回答 aaabccc の回答と本質的に類似していますが，どちらの答えも 2 つの文字列（eqe と abbba と，eqe と abbbc）を同じ方向に向かい，一方の文字列は文字の，他方は数の問題になっていると見ることに依存するからです. しかしながら，一方の文字列は文字の，他方は数の問題になっていると見ることで，abbbc の文字 a・グループ bbb・文字 c の間の文字カテゴリ入れ替えが不可能であるという事実から生じる障害を避ける後者の eqe と abbbc の場合の違い，前者の場合は一方は文字の，他方は数の問題と見る強制的な理由はありません．総じて，aaabccc は良い回答ですが，根拠のないアイデアが含まれないからです.

Figure11 の直訳 障害に当たった後の aaabaaa 対 aaabccc

5 DISCUSSION

- Metacat・Copycat との一部類似（非決定的・並列・活性伝播）
  - デュアル認知アーキテクチャ [Kokinov 1994,2000]
  - その他 [Falkenhainer 1990; Thagard 1990; Forbus 1995; Barden 1994]
- 創造性・類推への CBR アプローチ [Leak 1996; Kolodner 1993; Kolodner 1994]
  - 記憶に保持する回答記述 - CBR の事例
  - Metacat と CBR の相違点
    :Metacat の目的：○類推問題を解く ×問題解決自身のモデル化
    :CBR はシステム構築指向（最近はそうでないものあるっぽい）
- 誘導類推
  - CBR：最終解のみ - 誘導類推（問題解決過程込）
  - Metacat と誘導類推（＋ CBR）の相違点
    :概念の性質・表現，not 実体
- CBR からのアイデア
  - 事例の記憶保持・新しい状況への再利用
  - CBR 研究には概念の性質・認識との相互作用が欠落

6 CONCLUSION

- Metacat の「おはなし」能力
- 過去の問題想起，回答間の類似点・相違点の説明
  - 表層的でごめんね