ゲーム情報学 1－2
（1999．6．24）

関係代数を用いたfeature中の論理式の効率的評価方法

金子 知適　　山口 和紀　　川合 慧
東京大学大学院 総合文化研究科 広域科学専攻
kaneko@graco.c.u-tokyo.ac.jp

概 要

ゲーム固有の概念を論理式により表現した評価関数は，高度な局面判断を学習できる可能性があることが示されている。しかし局面の論理式による評価は他の表現による評価と比較して非常に遅いため，学習に必要な十分な量のデータが扱えないという問題点を抱えている。本稿では演繹データベースにおけるルールの評価手法にもとづき，論理式の評価を関係代数の計算に変換することで効率化する。この手法では，Prologを用いて論理式を評価した場合に生じる冗長な計算を省くことができる。プロトタイプシステムを実装し実験したところ約9倍の高速化が達成されたことを確認した。

An Efficient Evaluation Method for Logical Features by the Relational Algebra

Tomoyuki KANEKO　　Kazunori YAMAGUCHI　　Satoru KAWAI
Graduate School of Arts and Sciences
The University of Tokyo
3-8-1 Komaba, Meguro-ku, Tokyo, 153-8902, JAPAN
kaneko@graco.c.u-tokyo.ac.jp

For mechanical acquisition of features for game state evaluation, researches[4] suggest that logical feature, that is the feature described by logical formula, is promising. However, the evaluation of logical feature by Prolog based on the SLD-resolution and backtracking is quite slow prohibiting it from being applied to more advanced games. In this paper, we propose an evaluation method for logical formula by employing the techniques developed by deductive database research[11]. The method is to translate a logical feature into an equivalent relational algebra, which can be calculated efficiently without redundant search. We show that the method can be improved further by optimizing the translated algebra. The prototype system we developed shows that this method makes the evaluation approximately 9 times faster than the naive method for our sample features and game states.

1　はじめに

本稿では局面に対して，その局面の“良さ”を数値化する関数を**評価関数**と呼び，評価関数中でゲーム固有の概念を表す部分をfeatureと呼ぶ。featureは局面に対して数を返す。例えばオセロ

```
f2(A):-owns(x,A). % PIECES FOR BLACK
```

図 1: Feature の例

において「黒石の数」は良い feature である。評価関数は，これらの feature から，その返す値を線形結合などのアルゴリズムを用いて最終的な局面の評価を与える。

パラメータ学習による評価関数の学習では，feature は人間により与えられたものに固定されていた。これに対して Zenith システム [4] では，feature の表現として論理式を採用し，ゲームのルールと対戦から自動的に生成した feature を用いて，オセロにおいて評価関数を獲得，改良することに成功している。

しかしながら，feature の表現として論理式を採用したモデルでは[1]，局面の評価のコストが高いという問題点がある。これは局面に対して論理式を評価する際の全解探索にかかるオーバーヘッドによるものである。本稿では，この問題を解決するために，新しい論理式の評価方法を提案する。

2 論理式の評価に関する問題

2.1 論理式を用いた feature

feature の表現として論理式を用いるモデルは Zenith システム [4][6] で提案され，Metagame[8] でも使われている。このモデルでは，feature は「変数」と「論理式」から構成され，feature の表す値は論理式を満たす解の数と定義される。但し，解の数は与えられた変数にのみに注目して数える。以後，この解の数え上げを，feature による局面評価と呼ぶ。論理式の中で使われる述語は，domain theory (ゲームのルール) としてあらかじめ定義しておく。domain theory は Prolog で書かれておりルールと事実からなる。事実には，盤面の図形的関係の様な固定的なものと，局面に応じて定められるものがある。

Prolog の記法で書いたオセロの feature の例を図 1 に示す[2]。この例では，A が変数で，owns(x,A) が論理式である。文献 [6] のオセロの domain theory では，x は黒番プレイヤーを，owns(P,S) はオセロでプレイヤー P がマス S に石をおいていることを示す。従って初期局面ではこの feature の値は 2 である。

2.2 Prolog による局面評価

Zenith や Metagame では feature による局面評価に Prolog を用いているが，Prolog の処理系が SLD-resolution と単純なバックトラックにより feature の論理式の評価を行うと，冗長な解の数え上げが頻繁に発生する[3]。これを，変数 `[A,B,C]` 論理式 `p1(A,B),p2(B,C)`, を持つ feature `f` を評価する場合で説明する。各述語 `p1`，`p2` はいくつかのルールで定義されており，ある局面でそれらの全解探索の結果は表 1 であるとする。

---

[1]他の feature の表現としては，例えばオセロの図形的パターン (Glem [2])，バックギャモンの升目そのもの [9] などがある。

[2]文献 [6] では `f2(B):-count([A],(owns(x,A)),B).` と Prolog で直接実行できる形式で記述されているが，本稿では count を除いて簡略化して表す。

[3]詳細は処理系の実装に依存する。

表 1: 述語の全解探索の結果

p1(X,Y)	a1	b1
	a2	b1
	a3	b1
p2(Z,W)	b1	c1
	b1	c2

p1			p2			(数え上げ)		
A	B		B	C		A	B	C
a1	b1	→	b1	?				
			↓ (p2(b1,C) の数え上げ)					
			b1	c1	→	a1	b1	c1
				↓(backtrack)				
			b1	c2	→	a1	b1	c2
				↓(backtrack)				
			fail					
↓ (backtrack)								
a2	b1	→	b1	?				
			↓ (p2(b1,C) の数え上げ)					
			b1	c1	→	a2	b1	c1
			↓(backtrack)					
			b1	c2	→	a2	b1	c2
			↓(backtrack)					
			fail					
↓ (backtrack)								
a3	b1	→	b1	?				
			↓ (p2(b1,C) の数え上げ)					
			b1	c1	→	a3	b1	c1
			↓(backtrack)					
			b1	c2	→	a3	b1	c2
			↓(backtrack)					
			fail					

図 2: バックトラックによる全解探索のながれ

この時，f の局面評価を行うと図 2 に示したように項 p2(b1,C) を満たす変数 C の数え上げが 3 回起こる。p2(b1,C) を満たす解は一定であるから，2 回目と 3 回目の数え上げは冗長である。

この冗長性は，局面を固定して多数の feature を評価する場合には，feature の間に共通する述語があるために頻発する。即ち，一般に評価関数は複数の feature からなるが，ある feature の局面評価のために数え上げた計算が，他の feature の評価に再利用されることがない。また評価関数が連続して局面を評価する場合には，局面に依存しないルールに関する計算が再利用されないという形でも起っている。

3 演繹データベースによる局面評価

本節では，演繹データベースのルールの評価方法を用いて，関係代数 [10][11] により feature の評価を行う手法を提案する。大まかな手順を示す。

1. 与えられた局面について，全てのルールについて解の一覧を求める。(3.1 節参照)
2. 各 feature に対して次の手順で値を評価する。
   (a) feature の論理式を関係代数式に翻訳する。(3.2 節参照)
   (b) 翻訳された関係代数式から値を計算する。(3.3 節参照)

手順 1 で domain theory に表れる全てのルールに対し解の一覧をあらかじめ求めて，以後それを用いるため，前節であげたような冗長な再計算は発生しない。

3.1 演繹データベースによる解の数え上げ

まずルールに対して解の一覧を求める手法について説明する。演繹データベースは 3 要素 $\langle F, R, C\rangle$ で構成される。F は Prolog の「事実」の集合に対応し，R は Prolog のルールの集合に相当する。C は integrity constraints であるがここでは考えない。演繹データベースでは，事実やルールを満たす解の集合を「関係」(relation) と呼び，質問に対して関係を返す。事実からルールにより導出される関係を求めるには，関係代数 [10][11] を用いる。

このステップでは，局面として与えられた事実から，domain theory で定義されたルールについて解を数え上げ，関係に格納する。関係を求める手法には，top-down evaluation[11] を採用した。ルール間の依存関係を調べ，依存されているルールから関係を求めることで効率的に行うことができる。この時，後に述べる自然接合のための索引を作っておく。なお解が局面に依存しない一部のルールについてはこの関係と索引を再利用することができる。

3.2 feature の論理式の関係代数式への変換

前節で計算した各ルールに対応する関係を用いて，feature の評価を関係代数の演算により行う。feature の値は論理式を満たす解の数であるから，feature に対応する関係を求めサイズ[4]をとれば良い。この関係は，feature をルールとみなして top-down evaluation により直接求めることができるが，ここでは後の節で述べる効率化を行うため関係代数式へ変換と実行を分けて考える。

[4]ここでは関係の要素数 (タプル数) を関係のサイズと呼ぶ。

top-down evaluation において feature の関係を求める関係代数式は，論理式の各項に対する関係を自然接合し，feature の変数に射影するものである。項に対応する関係とは，項の述語の関係から，ATOV アルゴリズム [11] により求める。但し，以下の例のように項の引数が単純な (変数が複合項に含まれずかつ 1 度しか表れない) 場合は，名前に対応する関係そのもの，あるいは定数についての選択をするだけでよい。

例として，f は以下の様に定義された feature であるとする。Prolog 同様，大文字 (A, B, C, D) は変数を，小文字 (o, x) は定数を表すとする[5]。

```
f(A,B,C) :- span(A,B,C,o), neighbor(B,C,D), owns(x,D).
```

この時，f(A,B,C) に対応する関係は，span, neighbor, owns に対する関係，span, neighbor, owns を用いて，次の関係代数式から求めることができる。ここで，π は射影，σ は選択，$\bowtie$ は自然接合である。また関係の右肩の記号は属性を表し，X^n は匿名の属性を表す。

$$\pi_{A,B,C}(\sigma_{X^1=o}(\underline{\mathtt{span}}^{A,B,C,X^1}) \bowtie \underline{\mathtt{neighbor}}^{B,C,D} \bowtie \sigma_{X^2=x}(\underline{\mathtt{owns}}^{X^2,D})).$$

3.3 関係代数式からの値の計算

feature の値を求めるには，前節で変換した関係代数式から関係を計算しサイズをとれば良い。この計算は，式の中の関係のサイズの情報を利用して効率的に実行することができる。

自然接合の並び替え

自然接合は可換でる。また，関係 r_1 と r_2 の自然接合にかかる時間は，r_1 のサイズを n_1，r_2 のサイズを n_2，m を出力サイズとすると，$O(n_1+n_2+m)$ である。そこで自然接合を繰り返す場合には，計算途中の出力サイズがもっとも小さくなる様な順番で自然接合を行ってゆくことが望ましい。そのために以下の手順で自然接合を並び替える [11]。

1. 最初に，もっともサイズの小さい関係を選ぶ。
2. 以下，選択されている関係に含まれる変数をもっとも多く持つ関係を選ぶ。

自然接合の省略

関係を求めなくてもサイズが求まる場合は，関係を求めることを省くことができる。共通な属性のない関係の自然接合は直積となるため，元の関係のサイズのみから，結果のサイズを求めることができる。

$$\|r1 \bowtie r2\| = \|r1\| \times \|r2\|.$$

この方法は，各 feature の項を共通する変数を持つものにグループ分けしておくことで効率的に適用できる。このグループ分けは feature の式のみに依存するため feature 生成時に行えばよい。

また，式中にサイズが 0 である関係が一つでもある場合は，結果のサイズは 0 である。これは自然接合の並び替えの中の手順 1 で判定すれば良い。

[5] この feature は文献 [6] のオセロの domain theory を用いて生成したもので，o は白石，x は黒石，span は連続する石の並び，neighbor はマスの隣接関係を表している。

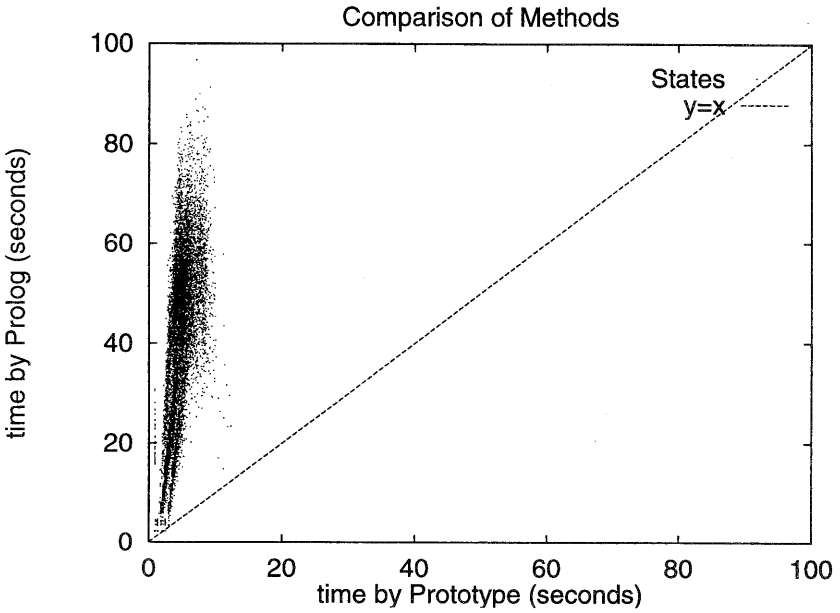

図 3: 本手法と Prolog の局面評価にかかる時間の比較

4 性能測定

プロトタイプシステムを実装し，本稿で提案した方式の評価を行った。実装言語には Scheme[6] を用いた。これと，Prolog[7] を用いて実装したものとの所要時間の比較を図 3 に示す[8]。図の各点は局面の評価に要した時間を表す。横軸が提案した方式で評価に要した時間，縦軸が Prolog を用いた実装での所要時間である。Prolog による実装では，5 節で述べる表引きを効果がでる範囲で実装している。

その結果，平均して約 8.7 倍の効率化を達成した。評価する feature は，Zenith の手法で生成したオセロゲームの 126 個の feature であり，1 つの feature は平均 3.9 個の項を含む。局面はシステムと人間との対戦 39 試合から生成した 10385 局面を用いた。

また各局面について，3.1 節で説明したルールに対する関係の生成にかかった時間を図 4 に示す。横軸が評価にかかった全体の時間，縦軸が関係の生成に要した時間である。この図から，関係の生成にかかる時間はわずかで，今後は自然接合に要する時間の最適化が重要であることが分かる。

5 議論

再計算を抑制する最も単純な方法は，一度計算した結果を表に記憶し 2 回目以降は表引きにより再計算を省略すること (いわゆる memoization) である。しかし，効率的な実装には処理系の拡

[6]SCM version 5c4 and Hobbit compiler version 5x, available at ftp://prep.ai.mit.edu:pub/gnu/jacal/.
[7]SWI-Prolog version 2.9.10, available at ftp://swi.psy.uva.nl/pub/SWI-Prolog/
[8]Pentium Pro/200 の PC を用いた。

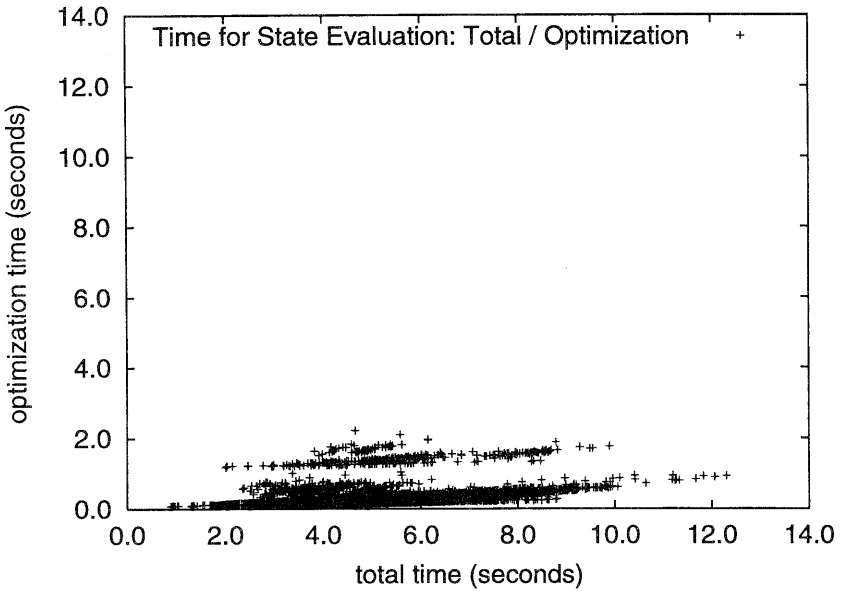

図 4: 関係代数式の生成にかかる時間の全体の評価時間との比較

張が必要である上，Prolog の全ての文法に対応することは困難である。

一般に Prolog で探索結果を記憶する研究 [7][3] では，記憶するべき結果の取捨選択や更新のタイミングが課題である。本稿では，状態の更新を明示的に扱う演繹データベースのモデルを採用することでこの問題をクリアした。

論理式を用いない評価関数と比較すると，効率はまだ改善の必要がある。現在 feature に表れる述語の folding による自然接合の省略や，view update [12][1] による局面間の類似性の利用を検討中である。一般に探索においは連続して評価する局面は近い傾向にあるため有効だと思われる。

本手法では，feature にいわゆるメタ論理述語[9]が使えないという制限がある。これは論理式を関係代数式に変換し解を求めることによる制限である。しかし feature とは状態を変更しないものであり，先行研究でも状態を変更するような述語は feature に使われていないためこの制約は問題にならない考えている。

6 まとめ

本稿では，feature を関係代数式に変換して評価する手法を提案した。この手法では，必要な関係を局面に基づいて初めに計算することで，feature の評価を効率化する。実験で用いた feature と局面に対して 9 倍の近くの高速化をもたらし，極めて有効な手法であることが示された。

[9]例えば assert/1 など。

参考文献

[1] J. A. Blakeley, N. Coburn, and P. Larson. Updating derived relations: Detecting irrelevant and autonomously computable updates. *ACM TODS*, 14(3):369–400, 1989.

[2] M. Buro. From simple features to sophisticated evaluation functions. In *Proceedings of the First International Conference on Computers and Games*. Springer-Verlag, 1998.

[3] G. DeJong and R. Mooney. Explanation-based learning: An alternative view. *Machine Learning*, 1:145–176, 1986.

[4] Tom E. Fawcett and Paul E. Utgoff. A hybrid method for feature construction. In L. A. Birnbaum and G. C. Collins, editors, *Proceedings of the 8th International Workshop on Machine Learning*, pages 137–141, Evanston, IL, 1991. Morgan Kaufmann.

[5] Tom E. Fawcett and Paul E. Utgoff. Automatic feature generation for problem solving systems. In D. Sleeman and P. Edwards, editors, *Proceedings of the 9th International Conference on Machine Learning*, pages 144–153. Morgan Kaufmann, 1992.

[6] Tom Elliott Fawcett. *Feature Discovery for Problem Solving Systems*. PhD thesis, Department of Computer Science, University of Massachusetts, 1993.

[7] T. Mitchell, R. Keller, and S. Kedar-Cabelli. Explanation-based generalization: A unifying view. *Machine Learning*, 1:47–80, 1986.

[8] Barney Darryl Pell. *Strategy Generation and Evaluation for Meta-Game Playing*. PhD thesis, University of Cambridge, 1993.

[9] Gerald Tesauro. Practical issues in temporal difference learning. *Machine Learning*, 8:257–278, 1992.

[10] Jeffrey D. Ullman. *Prinsiples of Database and Knowledge-Base Systems, Volume I:Classical Database Systems*. Computer Science Press, Maryland, 1988.

[11] Jeffrey D. Ullman. *Prinsiples of Database and Knowledge-Base Systems, Volume II:The New Technologies*. Computer Science Press, Maryland, 1989.

[12] Toni Urpí and Antoni Olivé. A method for change computation in deductive databases. In *Proceedings of the 18th International Conference on Very Large Data Bases*, pages 225–237, 1992.