特集　ゲーム情報学

応用

7 その他の2人ゲーム

岸本章宏（東京工業大学大学院情報理工学研究科／科学技術振興機構さきがけ）

2人ゲームと必勝法の解析

数あるゲームの中で，2人のプレイヤでプレイし，片方のプレイヤが勝つときはもう片方が負ける（零和）ことが保証され，さらに自分と相手の状態がすべて公開されている（完全情報），有限手数で必ず終了するゲームは，有限確定二人零和完全情報ゲームと呼ばれている．有限確定二人零和完全情報ゲームでは，両プレイヤが最善手を指し続ければ，ゲーム初期局面の理論的な値は，先手必勝，後手必勝，または引き分け☆1どれかになるということが保証されている．

将棋は，このカテゴリに属する，日本でポピュラーなゲームであり，強い将棋プログラムを作る研究は，日本の研究者を中心に盛んに行われてきた．現在の将棋プログラムの強さは，アマチュアトップ棋士や清水市代女流六段に勝利を収めるなど，トッププロ棋士の背中が見えており，研究として最も面白い段階にある．

一方，有限確定二人零和完全情報ゲームは，世界的にはチェッカー，チェス，シャンチーなどさまざまなものが存在する．これらのゲームも，多くの人々によって盛んにプレイされているだけでなく，人間のチャンピオンに勝てるような強いゲーム・プログラムを作ることを目標に研究も世界的に盛んに行われてきた．実際，オセロやチェスでは，人間の世界チャンピオンとプログラムとの対戦が何度も行われ，最終的にはこれらのプログラムの強さは，人間を凌駕した．

計算機が人間を凌駕してしまったゲームでは，ゲームの「必勝法」を解明し，完璧なプレイヤを作ることが次の研究目標の1つである．

本稿では，人間が実際にプレイしてきたゲームで，すでに必勝法が解析されたゲームの例として，どうぶつしょうぎ，アワリ，五目並べ，およびチェッカーを取り上げ，解析に使われた手法を概説する．

ゲームの解析レベルの定義

あるゲームを解いたと宣言されたとき，そのゲームの解明具合には，次のようなレベルがある[1]．

- **超弱解決**（ultra weakly solved）：初期局面の勝ち負け（または引き分け）の理論的な値のみが解明しているが，どのような指し手を選択すればよいのかは分からない．

  たとえば，盤の大きさが$N \times N$のヘックスというゲームでは，引き分けが存在せず，さらに後手必勝だと仮定すると理論的に矛盾が生じるので，超弱解決で先手必勝である☆2と証明されている．
- **弱解決**（weakly solved）：初期局面の勝ち負け（または引き分け）が証明されており，さらに初期局面の結果を証明する，両プレイヤの最善手の手順が具体的に分かっている．
- **強解決**（strongly solved）：初期局面から至ることのできる局面すべてに対して，勝ち負けと最善手を選択する方法が分かっている．

ゲーム研究者は，人間にとって難解なゲームで，どのような手順で勝ち負けが決まるのかに興味があるので，ゲームの弱解決または強解決を目標にしている．

ゲームを解く難しさの指標は，探索空間の大きさ

☆1 ルール上，引き分けが存在しないゲームもあり，その場合の初期局面の結果は，先手必勝か後手必勝かのどちらかである．

☆2 ただし，サイズ8×8までは，具体的な先手勝ち手順も計算機によって証明されている．

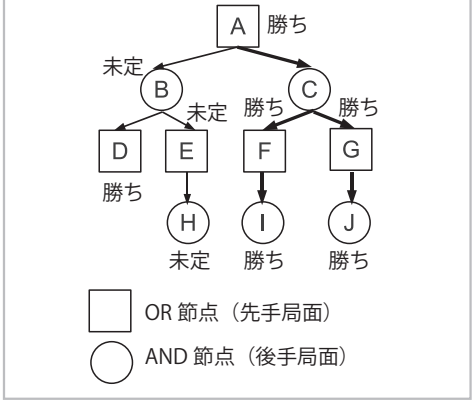

図 -1　AND/OR 木の例（勝ち，負けは先手の観点から記述）

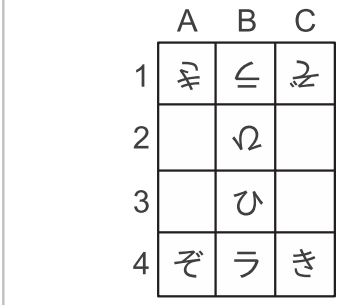

図 -2　どうぶつしょうぎの初期局面（ひ，ぞ，き，ラは，それぞれ，ひよこ，ぞう，きりん，ライオンに対応）

と最善手決定の複雑さである．探索空間が大きければ，計算の組合せ爆発を生じるので，当然解くのが難しくなる．しかし，探索空間が大きなゲームであっても，最善手の候補を理論的にいくつかに限定できれば，その候補のみを調べればよいので，必勝法の解析が簡単である．このため，どの指し手が最善かが分かりにくければ，探索空間があまり大きくなくても解きにくいゲームになり得る．

必勝法解明と AND/OR 木

先手番の局面 P で，先手が勝ちであるためには，P で指せる合法な指し手のうち，少なくとも 1 つが先手勝ちであればよい．一方，後手番の局面 Q で，先手が勝ちであるためには，Q でのどのような指し手に対しても，先手が勝ちである必要がある．先手負けの証明は，先手勝ちの証明とは双対な関係である．

ゲームでの必勝法を見つけるためには，ゲームの局面と指し手をそれぞれ節点と枝に対応させた，**AND/OR 木**を利用して，モデル化が可能である．AND/OR 木では，各節点が持ち得る値は，先手勝ち，先手負け，または未定である[☆3]．

図 -1 に AND/OR 木の例を示す．後手局面である B では，D を選ぶと先手が勝ちであるが，E を選ぶとまだ値が未定である．つまり，まだ後手が勝てる可能性があるので，B の値は未定である．一方，C では，F と G のどちらを選んでも先手勝ちであるので，先手勝ちである．A では，C を選べば先手が勝てるので，A の値は先手勝ちである．

A の先手勝ちを保証するには，C が先手勝ちでありさえすれば，B 以下の値がどのような値でも構わない．つまり，ゲームを弱解決したい場合に最低限勝ち負けを知っておけばよい節点は，図 -1 の太線上にある節点の結果のみ（**証明木**と呼ぶ）であり，その他の計算は省略できる．一方，ゲームを強解決したい場合には，初期局面から到達可能なすべての局面に対しての勝ち負けを計算しなければならないので，B や E，H の勝ち負けの結果も必要である．つまり，計算量の観点からは，ゲームの強解決は弱解決よりもずっと難しい．

どうぶつしょうぎ

どうぶつしょうぎは，2008 年に女流棋士の北尾まどか女流初段によって発明された，子供向けの将棋型ゲームである．どうぶつしょうぎでは，サイズが 3 × 4 の盤面で，将棋を簡略化したルールを用いている（図 -2 を参照）．どうぶつしょうぎの駒には，ライオン（将棋の玉に相当），ぞう（角を 1 マスしか進めなくしたもの），きりん（飛を 1 マスしか進

☆3 引き分けの証明には，AND/OR 木をさらに一般化したミニマックス木でモデル化するか，または引き分けを先手勝ちとして取り扱う AND/OR 木と，引き分けを先手負けとして取り扱う AND/OR 木の 2 種類を保持するかのどちらかで実現可能である．どちらの方法にも，利点と欠点があるが，本稿では説明の簡略化のため，引き分け証明についての詳細は省略する．

めなくしたもの），ひよこ（歩に相当），およびにわとり（ひよこが成れば，金になる）があり，相手のライオンを取れば勝ちである．

どうぶつしょうぎの初期局面から到達可能な局面数は 2.46×10^8 であり，2009年5月に田中によって後手勝ちが証明された[2]．解析には，Opteron 2.6GHzのマシン（メモリは16GB）で約5.5時間要している☆4．どうぶつしょうぎのように，合法局面数が少なければ，局面をすべて数え上げ，勝ち・負け・引き分けのラベルを付けることで必勝法を求めることが可能である．このように解を計算すれば，強解決である．

田中が用いた**後退解析（Retrograde Analysis）**[4]は，全局面に勝敗の結果を計算するのに効率の良い方法である．後退解析アルゴリズムの概要は次の通りである．

(1) ゲームのルールによって，勝ち負けが決着した局面をすべて生成する．この集合を S とする．
(2) S から1手前の全局面を計算し，S に加える．
(3) S にある勝ち・負けが未確定な各局面 P に対して，次の操作を行う．ただし，P は先手の局面とする（後手の場合も同様の処理を行う）．
  (a) P での合法手の少なくとも1つによって，S にある先手勝ち局面 Q に至る場合には，指し手 $P \rightarrow Q$ を選べば必ず先手勝ちであるので，P を勝ちとする．
  (b) P での合法手のすべてが，先手負け局面に至る場合には，どの手を選んでも先手負けであるので，P を先手負けとする．
  (c) それ以外ならば，P の値は未定とする．
(4) S にある勝ち・負けの局面数が現在よりも増えなくなるまで (2) と (3) を繰り返す．
(5) 勝ち負けが未定の局面を引き分けとする☆5．

アワリ

アワリは，アフリカを起源とする3500年以上にわたり遊ばれているゲームであり，48個ある石とピットと呼ばれる石置き場からなる石取りゲームである．各プレイヤのピットは6個ずつであり，最初のピット内には4個ずつ石が配置されている（図-3）．盤外の2つの数字は，各プレイヤが取った石の数を表す．

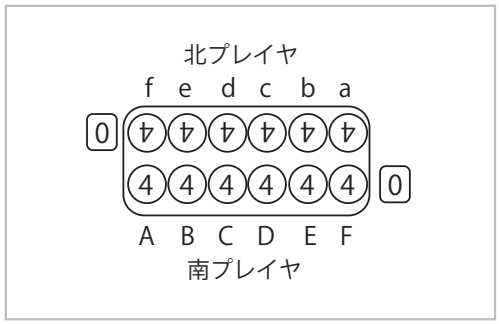

図-3　アワリの初期局面（南が先手）

手番のプレイヤ（プレイヤは北または南と呼ばれ，南が先手である）は，空でないピットから石をすべて取り出し，これらの石を1つずつ，各ピットに反時計回りに置いていく☆6．最後に石を置いたピットが相手のピットであり，ピット内の石の数が2個または3個であれば，そのピットの石を取ることができる．石を取った後に，同様に1つ前の相手ピットに石が2個または3個あれば，それらの石を取ることができる．このルールは，時計回りに再帰的に適用される．25個以上の石をどちらかのプレイヤが取るか，プレイヤの指す手がなくなれば，ゲームは終了である．

アワリの初期局面から到達可能な局面数は，2.04×10^{11} である．Romeinと Balは，2002年に72ノードから構成される計算機環境（各計算ノードは，クロック1.0GHzの2つのPentium IIIプロセッサ，1GBメモリ，20GB IDEハードディスクから構成され，2Gb/sのMyrinetネットワークでつながれている）で，合計144プロセッサとメモリ72GBを使

☆4 田中哲朗氏の報告は2009年5月であるが，鈴木康夫氏も2009年6月にdf-pn探索[3]を用いて弱解決している（計算時間は6.8時間）．詳しくは，http://www.computer-shogi.org/blog/csa_mtg_may_2009/ にある．
☆5 多くのゲームでは，ある局面から指し続けて，元の局面に戻ることを何度か繰り返した場合には，引き分けと定義されている．たとえば，どうぶつしょうぎでは，同一局面が4度出現すると引き分けである．このようなゲームを解析する場合には，同一局面が2回出ると引き分けと取り扱えばよい．後退解析の勝ち負けの決定過程が改善しなくなれば，両プレイヤが最善手を指し続けるとループを作り出す局面のみが，結果が未定として残っている．
☆6 ただし，石を取り出したピットには石を入れない．また，この操作はsowing（種まき）と呼ばれている．

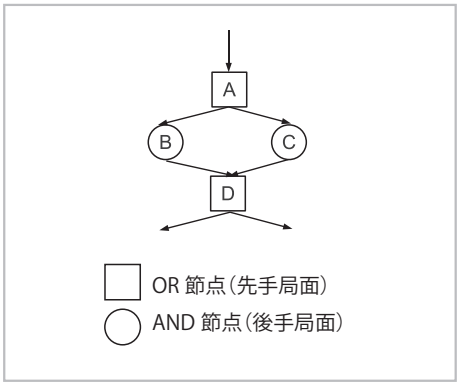

図 -4　分散後退解析の問題点

い，51 時間計算した結果，両プレイヤが最善手を指せば引き分けであることを強解決で証明した☆7．ただし，初期局面で引き分けにできる指し手は，F のみであり，それ以外は南プレイヤの負けである．

Romein らの解法では，後退解析を分散並列化し，結果をデータベース化している[5]．どうぶつしょうぎの後退解析では，解析中に現れる局面すべてをメモリに記憶できたが，アワリでは，各計算ノードのメモリだけでなく，ハードディスクにも途中経過を保存する必要がある☆8．このため，メモリにはアクセス頻度が高く，かつランダム・アクセスする情報を保持する．アクセスの頻度が低いものは，逐次アクセスできるようにしつつ，ハードディスクに置いている．

分散環境での後退解析を行う際の問題に，アワリの探索空間では複数の経路で同一局面に到達し得ることがある．この場合に，各局面の後退解析の結果を分散共有するには，通信や同期オーバヘッドなどが生じるので，高性能な並列化手法の開発は難しい．

たとえば，図 -4 のようなグラフに対して，プロセッサ P_1 と P_2 で並列に後退解析を行うと仮定し，P_1 と P_2 はメモリを共有していないとする．節点 D から後退解析を開始し，P_1 が C の解析を行い，P_2 が B の解析を行ったとする．もし，P_1 と P_2 がプロセッサ間通信を行わなければ，P_1 も P_2 も A を生成するので，A の解析を二重に行ってしまう．一方，P_2 に A の結果があるかどうかを P_1 が確認したければ，単純な方法は次に述べる通りであるが，通信と同期の双方のオーバヘッドが生じてしまう．

(1) P_1 から P_2 に，A の結果の送信を依頼する．

(2) P_2 はメッセージを時々チェックし，P_1 からのメッセージが来たことを確認する．

(3) P_2 は，P_1 に A の結果を送信する．

(4) P_1 は，P_2 からの A の解析結果が到着するまで待ち，結果が届けば再利用する．

Romein らの解法では，局面 N を入力として，N の情報を保持するプロセッサを静的に決める関数 $f(N)$ を定義し，N の解析は必ずプロセッサ $f(N)$ が行うようにしている☆9．この方法では，N の解析結果は，必ずプロセッサ $f(N)$ のローカルのメモリ（またはハードディスク）上にあることが保証できるうえに，複数経路から N に至る場合があっても，プロセッサ $f(N)$ が必ず N を担当することになるので，N を重複して解析することはない．さらに，プロセッサ P が N をプロセッサ $f(N)$ に送信してしまえば，P は N の解析結果を受け取る必要がないので，P での同期が不要である．つまり，Romein らは，並列計算の非同期化によって通信遅延を隠蔽し，高い並列効果を出している．さらに，通信オーバヘッド削減のため，複数局面を 1 つのメッセージにまとめて送信している．

並列後退解析で作った局面をすべて保存したデータベースの大きさは 178GB である．また，並列後退解析の際に，ネットワークを介して通信したデータ量は 130TB であり，2002 年の計算機環境を考えるとこれらのデータのサイズは非常に大きい．

五目並べ

五目並べは，碁盤のようなボード上で，各プレイヤが交互に石を置く，日本では非常によく知られたゲームである．縦，横，または斜めに，同じ色の石が連続して 5 個以上並べば，そのプレイヤの勝ち

☆7 アワリのルールにはさまざまなバリエーションがあるが，解かれたのはゲーム研究者がよく利用しているルールである．

☆8 現在の計算機環境では，分散並列化等を行わなくても解ける可能性があるが，2002 年当時の計算機環境で利用できるハードディスクとメモリは現在よりもはるかに少なかった．

☆9 文献 5）には，$f(N)$ の具体的な構築方法は書かれていないが，データベースのインデックスには Gödel 数を使っているので，この情報を利用している可能性が高い．また，別の方法としては，ゲーム・プログラムでよく使われる Zobrist 関数による方法も考えられる．

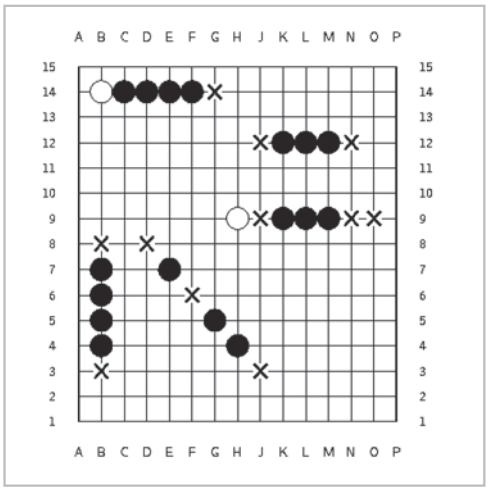

図-5　脅威の例（文献1），6）より）

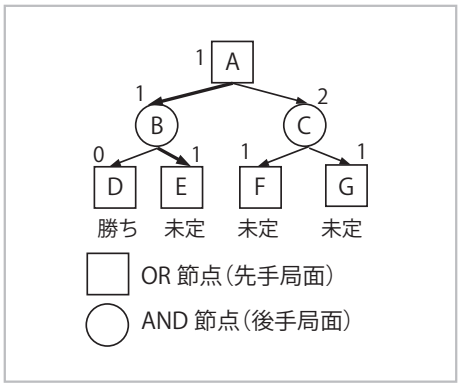

図-6　証明数（各節点のそばにある数値が証明数）

である．公式には，このゲームはサイズが15×15の盤を使うのであるが，最初の局面で選択できる指し手は225であり，探索空間自体は非常に大きい．

五目並べは，Allisと Van den Herik，およびHuntjensによって1992年に弱解決された(先手勝ち)[6]．

文献1）によると，Allisらは11台のSUN SPARC Station（各マシンのメモリは64または128MB）を利用し，CPU時間で15.1日かけて五目並べを解いた☆10．

五目並べで重要な概念に**脅威**（**threat**）がある．このゲームでは，局面によっては，すべての合法手を生成する必要はなく，一部の指し手（脅威手と呼ぶ）のみを考えれば，求める結果の正しさを保証できる．このようにして，五目並べを解くのに探索しなければならない空間を大幅に減らせる．たとえば，図-5で，×と書かれた部分のどこかには，白が必ず打たなければならない．これらの手を打っても，白の勝ちは保証できないが，白の負けを少なくとも先延ばしにできる．

Allisらは，脅威の概念に加えて，**証明数探索**（**proof-number search**）[1]を利用している．証明数探索は，**証明数**（**proof number**）と**反証数**（**disproof number**）という概念を利用して，最も有望そうな節点を優先的に展開する最良優先探索である．

ある局面Pの証明数とは，現在構築した探索木に基づいて，Pの先手勝ちを示すのに展開が必要な先端節点（まだ展開されていない節点）の数の最小値である．証明数は，先手勝ちの証明のしやすさの指標であり，証明数が小さければ，先手が勝ちやすいと推測できる．逆に，反証数は，証明数とは双対の概念であり，Pの反証数とは，Pの先手負けを示すのに展開する必要のある先端節点数の最小値である．反証数が小さければ，先手負けの証明を行いやすいとみなす．

図-6を用いて，証明数を具体的に説明する．この探索木において，Aが先手勝ちであるためには，BまたはCが先手勝ちであればよい．Cが先手勝ちであるためには，少なくともFとGの両方が勝ちであることを示す必要があるので，Cでの証明数は2である．一方，Bでは，Dがすでに先手勝ちである（証明数は0）ので，Eを展開すれば先手勝ちであることが分かれば，一番理想的である．このため，Bの証明数は1である．BとCとの証明数より，AではBを選択した方が労力が少なく先手勝ちを示せそうなので，B以下を優先的に探索する．反証数に関しても同様の考え方で子供節点の選択を行う．

五目並べを解くとき，脅威を用いてもまだ選択可能な指し手が多いときがある．五目並べは，人間の解析では，確実に先手勝ちであろうと言われていた☆11．Allisらは，この事実を利用し，ヒューリ

☆10 文献1）には，1日あたりの利用CPU時間は約150時間とあるので，解くのに必要な時間は実質数日であったと推測される．
☆11 このため，五目並べ型のゲームである連珠では，先手が作ってはいけないパターンをルールで定義している．

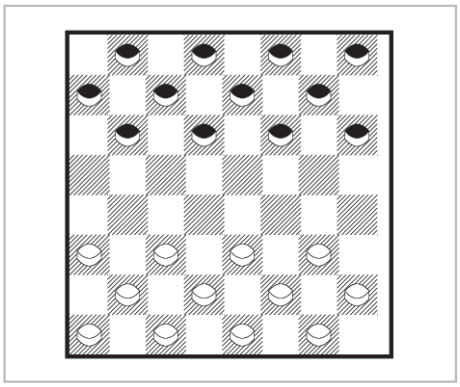

図 -7　チェッカーの初期局面（黒が先手）

ィスティックで先手の生成する指し手を制限している．この指し手生成では，アルゴリズムが後手勝ちであると判定した場合には，先手勝ちの可能性がある．しかし，先手勝ちと判定したときには，結果の正当性が保証できる．

さまざまな効率化を行った Allis らの証明数探索が五目並べを解くのに探索した局面数は，約 630 万局面ほどである．ただし，各節点において，脅威手関係の処理のためにいくつか局面を生成する必要があるので，実際には 5,000 万から 1 億 3,000 万局面程度探索している．

チェッカー

チェッカーは，北米人口の 9 割以上がプレイしたことがあると言われている非常に有名なゲームである．チェッカーでは，サイズが 8 × 8 の盤に黒と白の各 12 個の駒を図 -7 のように配置する．相手の指し手をなくすか，相手の駒をすべて取れば勝ちである．

駒（チェッカーと呼ばれる）は，基本的には斜め前方に 1 マスしか動けないが，相手陣の一段目に到達すれば，キングになる．キングになれば斜め後方にも移動できるようになる．さらに，自分の駒の斜め前に相手の駒があり，かつその先のマスが空白であれば，相手駒を取りながら，その空白マスに移動（ジャンプと呼ぶ）しなければならない．到達した先のマスでも同じ状況になっていれば再び移動しなければならず，移動できなくなるまで，再帰的に行われる．なお，キングは，前方だけでなく後方にもジャンプできる．

チェッカーの局面数は，5×10^{20} であると推測されており，これまでに述べてきたゲームよりも遥かに難しい．五目並べの探索空間自体は，チェッカーよりも大きい可能性があるが，チェッカーでは，脅威手のような指し手を限定する方法は使えないので，五目並べよりも解くのがずっと難しいゲームであった．

チェッカーは 2007 年 4 月に Schaeffer らによって，引き分けであることが弱解決で証明された [7]．筆者の知る限り，多数の計算機を利用しており，何年にもわたり，計算を行った（詳しくは文献 8）を参照）．

チェッカーは，これまでに解かれたゲームよりも解くのがはるかに難しいので，探索や終盤データベースなど，さまざまな手法を組み合わせている．

チェッカーはゲームが進むにつれて，盤上にある駒の数が減少していくゲームである．駒数が少ない場合の局面は，後退解析を用いれば，各局面の勝敗結果をすべてデータベース化できる．Schaeffer らの解法では，駒数が 10 以下の局面の勝敗結果（3.9×10^{13} 局面）は，すべて終盤データベースとしてハードディスクに保存している．つまり，初期局面から探索を行ったときに，駒数が 10 以下の局面に遭遇すれば，それ以上は探索を行わず，終盤データベースにある勝敗を調べることで，探索を省略している．終盤データベースは，圧縮してハードディスクに保持されているうえに，237GB もあるので，メモリ上にデータベースのすべての情報を展開することはできない．

終盤データベース参照の際のハードディスクへのアクセスの頻度を減らすために，OS で基本的な手法であるページングの考え方を利用して，データベースの一部だけをメモリに置いている．しかし，それでもハードディスクのアクセス頻度が高く，CPU 使用率が 10% 以下であることがほとんどであった．

初期局面から，必勝法を求めて探索を行うアルゴリズムは，フロントエンド探索とバックエンド探索と呼ばれる 2 つの探索を行っている．フロントエン

ド探索では，初期局面[☆12]から前述の証明数探索をベースに利用して，解を求めるのに最も有望そうな先端節点を選択する．この先端節点は，バックエンド探索に引き渡し，バックエンド探索が一定時間探索を行う．バックエンド探索に利用した情報は，この探索の終了後にメモリから消去する．バックエンド探索では，$\alpha\beta$法と日本で開発された，証明数探索の改良手法であるdf-pn探索[3]を併用している．バックエンド探索が，勝ち・負け・引き分けの結果を返してくれれば，この情報を元にして，フロントエンド探索が保持している探索木の証明数・反証数を更新する．また，バックエンド探索の探索結果が未定の場合には，フロントエンド探索が先端節点を展開し，証明数・反証数を再計算する．これらの過程を必勝法の解明が終わるまで繰り返す．

チェッカーでは，$\alpha\beta$法が勝ちそう（または負けそう）だと推測した局面は，ほぼその通りである．この性質を利用して，フロントエンド探索では，勝ち負けの信頼度を定義する閾値thを用意し，バックエンド探索の$\alpha\beta$探索が，このthにくらべて，勝ち（負け）である可能性が高いと判定した局面を勝ち（負け）であるとみなしている．thには，最初は小さな値を設定し，フロントエンド探索が初期局面を解いた後に，少しずつ大きくし，再探索する．つまり，最初は後回しにしていた節点の勝ち負け判定を徐々に正確に取り扱う．最初の必勝法の解は，疑似証明であるが，$th=\infty$の探索終了後には，正しい結果になる．

チェッカーの証明木の大きさは，大きく見積もると約10^{14}局面である．ただし，終盤データベースによって省略された部分は，証明木の大きさの計算には含まれていないので，実際には必勝法の解明のためには，この数値よりも大きな証明木を構築しなければならなかった可能性もある．

☆12 実際には，並列に解析できるように，図-7の初期局面からいくつか手順を進めた局面の中で，引き分け証明に必要な局面を複数取り出し，これらの局面から探索を開始している．

今後の展望

本稿では，すでに計算機によって必勝法が解析されたゲームに用いられた手法について簡単に説明した．

チェッカーの次に難しく，有名なゲームにはオセロがあり，探索空間は10^{28}程度である．現在，オセロはサイズが6×6のものが弱解決（後手勝ち）されているが，サイズが8×8のオセロは，チェッカーの約10^{8}倍大きな問題でかつ，チェッカーで大幅な探索削減効果のあった終盤データベースの考え方を使えないのが，難しい点である．これらの点を改善する技術的革新が起これば，オセロ解明の日も来るかもしれない．

参考文献

1) Allis, L. V. : *Searching for Solutions in Games and Articial Intelligence*, PhD thesis, University of Limburg (1994).
2) 田中哲朗：「どうぶつしょうぎ」の完全解析，第22回ゲーム情報学研究会，情報処理学会 (2009).
3) Nagai, A. : *Df−pn Algorithm for Searching AND/OR Trees and Its Applications*, PhD thesis, Department of Information Science, University of Tokyo, Tokyo (2002).
4) Thompson, K. : Retrograde Analysis for Certain Endgames, *ICCA Journal*, Vol.9, No.3, pp.131-139 (1986).
5) Romein, J. W. and Bal, H. : Solving the Game of Awari Using Parallel Retrograde Analysis, *IEEE Computers*, Vol.36, No.10, pp.26-33 (2003).
6) Allis, L. V., van den Herik, H. J. and Huntjens, M. P. H. : Go-moku Solved by New Search Techniques, In *AAAI Fall Symposia*, pp.1-9 (1993).
7) Schaeffer, J., Burch, N., Björnsson, Y., Kishimoto, A., Müller, M., Lake, R., Lu, P. and Sutphen, S. : Checkers is Solved, *Science*, Vol.317, No.5844, pp.1518-1522 (2007).
8) 岸本章宏：チェッカー解明秘話，情報処理，Vol.48, No.11, pp.1257–1263 (Nov. 2007).

(2011年10月28日受付)

田中哲朗様は，執筆の機会をお与えくださっただけでなく，本稿にさまざまな有益なご提案をくださいました．心より感謝申し上げます．

岸本章宏（正会員）
kishimoto@is.titech.ac.jp

東京工業大学大学院情報理工学研究科数理・計算科学専攻助教および科学技術振興機構さきがけ研究者．探索アルゴリズムと並列処理に興味を持つ．東大将棋の開発に従事．IJCAI 2005やICAPS 2009などで最優秀論文賞を受賞．チェッカー解明の研究が，*Science*誌の2007年のBreakthrough of the yearのトップテンに選ばれる．