解説

岸本 章宏
公立はこだて未来大学
システム情報科学部情報アーキテクチャ学科

チェッカー解明秘話

Checkers

よく知られたボードゲーム「チェッカー」は，両プレイヤが最善を尽くせば引き分けになることが，アルバータ大学（カナダ）の Jonathan Schaeffer 教授を中心とする研究チームによって，計算機を用いて証明された．

本稿では，チェッカーの解明に利用した技術と筆者がプロジェクトの一員として参加した経緯，およびチェッカー解明までの道程について述べる．

チェッカー解明：黒先引き分け

Jonathan Schaeffer は，1989 年よりチェッカーを題材にした人工知能研究を行っていた．研究の集大成として，18 年後の 2007 年 4 月 29 日にチェッカーの初期局面が引き分けであることを証明し，7 月 19 日午後 2 時（東部標準時間）に Checkers Is Solved[1] という論文が，Science 誌（電子版）に掲載された．

その日の私は，論文掲載の事実を確認して一安心したことと，函館より東京まで出張すること以外は，ごく普通の一日であると考えていた．チェッカーは，欧米では有名なゲームであることは理解していたが，日本では特にメジャーではないので，何も反響がないと考えていた．ところが，この話題が日本経済新聞等に取り上げられていることを知り，非常に驚いた．

海外での反応は，日本以上であった．BBC や CNN をはじめとする数多く（20 以上？）のメディアの Web サイトに Schaeffer とチェッカーに関する記事があった．私自身も，さまざまな国の人からメールを受け取った．メールの中身は，結果を称賛するものが多かったが，チェッカー・プレイヤは今後どうすればよいのかという批判的なものも存在した．

チェッカーとは

チェッカーは，北米人の 9 割以上が一度は遊んだことのあるボードゲームである．チェスでのこの数値は約 5 割であるので，この意味では，北米ではチェッカーはチェスよりポピュラーなゲームであるといえる．

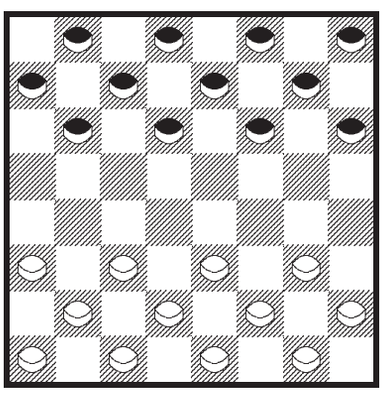

図 -1　チェッカーの初期局面

チェッカーでは，チェス盤に**図 -1** のように黒と白の各 12 個の駒を配置する．黒から開始し，各プレイヤは交互にプレイする．相手の指し手をなくせば，自分の勝ちである．ルールの概要は，以下の通りである．

- 駒は斜め前方に 1 マスしか動けないが，相手陣の 1 段目に到達すれば，キングになる．キングは，斜め後方にも移動できる．
- 斜め前に相手駒があり，さらにその先のマスが空白のときには，相手駒を取りながら，その空白マスにジャンプしなければならない．このルールは，ジャンプができなくなるまで，再帰的に適用される．キングは，後方にもジャンプできる．

チェッカーは，囲碁や将棋などと同様に，**有限確定二人零和完全情報ゲーム**であり，両プレイヤが最善手を指

し続ければ，ゲームの結果(便宜上，**ゲーム値**と呼ぶ)は，勝ちか負けか引き分けのどれかになる．Science 誌の論文では，初期局面（図 -1）から，両プレイヤが最善手を指し続ければ，引き分けになることを示した．なお，現在のチェッカーの公式戦では，最初の 3 手をランダムに選んで進めた局面から開始する 3-move ballot opening (144 個存在する) を利用することが多い．初期局面の証明には，このオープニングの 19 個が利用され，さらに 2 個が解けている．残りのオープニングに関しても計算する予定であるが，解明にどれくらいの時間を要するかは不明である．

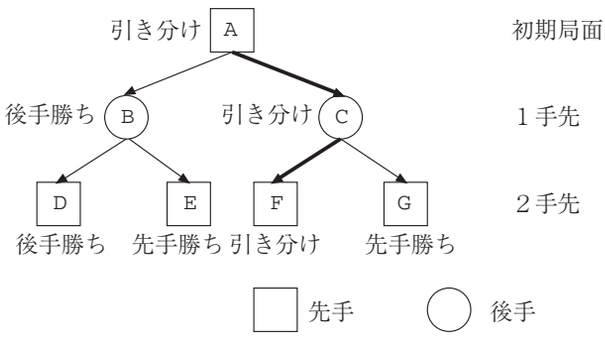

図 -2　ゲーム木探索(2 手先読みをしたとき)

引き分け証明の難しさ

チェッカーのゲーム値が存在することは，簡単に証明できる．しかし，引き分けの証明を行うには，最善手を「具体的」に示し，それが引き分けに至ることを示す必要があるので，簡単な問題ではない．これは，初期局面から，勝ち，負け，引き分けに至るさまざまな局面を調べ上げ，各プレイヤの最善手を見つけ出す**ゲーム木探索**の問題に帰着できる．各局面は，探索木の節点に，指し手は枝に対応する．

図 -2 は，ゲーム木探索の例である．D，E，F，G を，ゲームのルールよりすでにゲーム値の決まっている局面(**終端局面**)とする．後手は，自分が勝てるように最善を尽くすので，B と C のゲーム値は，それぞれ後手勝ちと引き分けになる．同様に，A における先手の最善手は，C を選ぶ手である．よって，A は引き分けであり，各プレイヤの最善手は太線になる．

ゲーム木探索によって，ゲームを解く難しさの指標には，次の 2 つがある．

- **探索空間の大きさ：**探索空間が大きければ，計算量が増えるので，解くのが難しくなる．これまでに解かれた最も難しいゲームに，Awari と Connect Four がある．Awari と Connect Four の探索空間の大きさは，それぞれ 10^{12}= 1 兆局面と 10^{14}= 100 兆局面である．一方，チェッカーの探索空間は，5 × 10^{20}= 5 垓局面であるので，これらのゲームの空間よりも，100 万倍以上も大きい．
- **最善手決定の複雑さ：**探索空間が大きくても，最善手の候補を理論的に絞れれば，その候補のみを調べればよいので，簡単なゲームになる．たとえば，五目並べでは，最善手の候補をかなり限定できる．一方，チェッカーでは，どの合法手も最善手の候補になり得る．

局面のゲーム値の決定には，すべての手を探索する必要はない．たとえば，先手番の局面 P が先手勝ちであるためには，P での指し手の少なくとも 1 つ（これを m とする）が，先手勝ちであればよい．つまり，P での最善手は m であるので，他の手の勝ち負けは無視できる．一方，P で m が指された後には，後手は，すべての指し手が先手勝ちであることを示す必要がある．

局面のゲーム値を証明するのに最低限必要な部分木を**証明木**と呼ぶ．チェッカーの初期局面の解明のためには，5 垓局面の中で，探索の不必要な局面を見つけ，効率良く引き分けの証明木を作ることが鍵となる．

Chinook プロジェクトについて

アルバータ大学コンピューティング・サイエンス科は，カナダ・アルバータ州の州都であるエドモントンにある．エドモントンは，人口 100 万人の都市であり，近くに(といっても車で 5 時間ほどかかるが) 観光地として有名なカナディアン・ロッキーがある．

アルバータ大学にある GAMES グループは，Jonathan Schaeffer をリーダーとするゲーム研究のメッカである．このグループでは，人工知能研究の題材にゲームを利用し，強いプレイができるアルゴリズムを開発している．研究対象は，チェッカー，オセロ，囲碁などのボードゲームから，ポーカーのようなカードゲーム，コマーシャル・ゲームまで，多岐にわたる．チェッカー，オセロ，ポーカーでは，世界最強のコンピュータ・プログラムの開発に成功している．

1989 年より Schaeffer が行っている Chinook プロジェクト（**図 -3** の写真を参照）では，チェッカーを題材にしている．プロジェクトの目標は，人間の世界チャンピオンに勝てるプログラム Chinook[☆1] の開発であった．

Chinook の開発当時，人間で最も強いプレイヤは，Marion Tinsley であった．Tinsley は，1950 年から

☆1 Chinook に関する情報は，http://www.cs.ualberta.ca/~chinook/ にある．なお，Chinook とは，アルバータ州で冬に吹く，気温を急上昇させる風のことである．発音もチヌークではなく，シヌークである．

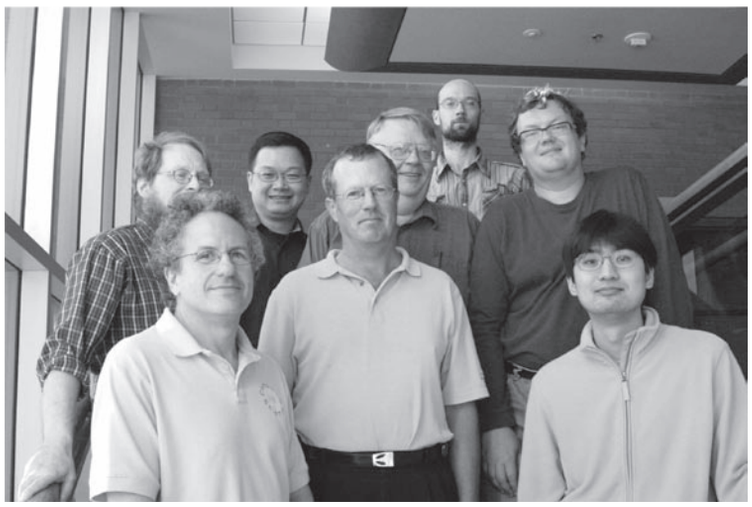

図 -3　Chinook グループの主要メンバ，前列左端が Schaeffer 教授，右端が筆者

1992 年までの公式戦で，3 度しか負けたことがないような伝説的な存在であった．

Chinook は，1990 年の全米選手権で，Tinsley に次ぐ成績を収め，Tinsley へのタイトル挑戦権を獲得した．1992 年に行われたタイトル戦では，Tinsley の 4 勝 2 敗 33 分であったが，Chinook は過去 40 年間に 3 度の敗戦しかない Tinsley に，2 度の黒星を付けた．

Chinook と Tinsley は，世界タイトルを懸けて，1994 年に再び相見えた．6 度の引き分け後，Tinsley は，病気によりタイトル戦を棄権した．このために，Chinook は，Tinsley にチェッカーで勝たずに，世界チャンピオンになった．Tinsley は，1995 年に癌のため死去した．Chinook は，1996 年までタイトルを防衛した．さらに，Chinook は，同年の全米選手権で，他の強豪に大差を付けて優勝した後，引退した．

Chinook は，コンピュータが世界チャンピオンになった初めての例であり，ギネスブックにも紹介されている．しかし，チェッカー・プレイヤの間では，Tinsley と Chinook のどちらが強いのかという疑問が常に生じていた．Tinsley の死後，Chinook プロジェクトに残された研究テーマは，チェッカーを解くことであった．絶対に負けないプログラムを作ることが，コンピュータの優位性を示すための唯一の手段であった．

チェッカーを解くためのアプローチ

引き分けを証明したプログラム（以下，**チェッカー・ソルバ**または**ソルバ**と呼ぶ）は，**終盤データベース**と**探索アルゴリズム**を利用している．終盤データベースは，終端局面から初期局面に向かって，ゲーム値を求めて，構築される．逆に，探索アルゴリズムは，初期局面から終端局面の方向に探索し，ゲーム値を求める．両手法によって，チェッカー・ソルバが初期局面の証明木をカバーすれば，証明が終了する．本章では，各手法の概要を説明する．

駒数	局面数
1	120
2	6,972
3	261,224
4	7,092,774
5	148,688,232
6	2,503,611,964
7	34,779,531,480
8	406,309,208,481
9	4,048,627,642,976
10	34,778,882,769,216
11	259,669,578,902,016
12	1,695,618,078,654,976
13	9,726,900,031,328,256
14	49,134,911,067,979,776
15	218,511,510,918,189,056
16	852,888,183,557,922,816
17	2,905,162,728,973,680,640
18	8,568,043,414,939,516,928
19	21,661,954,506,100,113,408
20	46,352,957,062,510,379,008
21	82,459,728,874,435,248,128
22	118,435,747,136,817,856,512
23	129,406,908,049,181,900,800
24	90,072,726,844,888,186,880

表 -1　各駒数に対する局面数（文献 2）より）

終盤データベースの構築

チェッカーでは，ゲームが進むにつれて，駒の数が減少する．駒数が少ない場合には，すべての局面を数え上げて，各局面のゲーム値を求められる．ソルバでは，駒数が 10 以下の局面のゲーム値を，前もって計算し，終盤データベースとして，ハードディスクに保持している．駒数が N 個の終盤データベースは，**N 駒データベース**と呼ばれる．

表 -1 に各駒数に対する可能な局面数を示す．10 駒データベースの構築には，高速な CPU だけでなく，大規模なメモリとハードディスクも必要である．1989 年には，4 駒データベース（$7.09 \times 10^6 = 709$ 万局面）しかなかった．10 駒データベースまで（合計 $3.9 \times 10^{13} = 39$ 兆局面）の構築を可能にしたのは，ハードウェアの進歩がある．10 駒データベースの構築は，2001 年に開始し，2005 年に終了した．利用した計算機環境は，7 ～ 8 台の

Athlon 1.6MHz のマシン（メモリ 1 〜 4GB），32CPU でメモリ 16GB の SGI マシン，12 ノードのクラスタ（各ノードの CPU は Opteron/250 × 2，メモリは 16GB），約 10 ノードのクラスタ（2001 年頃のハードウェア，詳細は不明）など，数多くである．他と共有の計算機環境（アイドル時間のみを利用）も多い．

当然のことながら，ソフトウェア技術の進歩も重要である．独立したデータベースは，並列に構築している．さらに，終盤データベースの圧縮率は，かなりよい．たとえば，10 駒データベースまでの各局面の結果を 2 ビットで表現したとすると，$3.9 \times 2 \times 10^{13} \div 8 = 9.57$TB のハードディスクが必要である．一方，圧縮したデータベースは，237GB である．その上，圧縮データを実時間で利用する工夫も行っている．

Chinook プロジェクトで最も時間を要したのは，間違いなく終盤データベースの構築である．この計算には，長い時間を要する上に，膨大な I/O アクセスが生じる．さらに，多数のコンピュータを利用すると，ネットワーク・エラー，ディスク・エラー，ソフトウェアのバグなどと格闘する必要がある．たとえば，ネットワークを介したデータ・コピー等のエラーは，月に 1 度程度は起こるので，エラー回復手法も必須である．

データベース構築の大変さを示すエピソードとして，8 駒データベース（合計 $4.4 \times 10^{11} = 4{,}400$ 億局面）の構築の話がある．この構築は，カリフォルニア州リバモア[☆2]にある国立研究所の BBN TC2000（128CPU, 1GB 共有メモリ）やアルバータ大学のマシンなど，合計 200CPU 以上のアイドル時間を利用して行われた．データベースの構築には，1991 年から 1996 年までかかった．この際に，データベースの結果の間違いのために，何カ月も要した計算が無駄になり，再計算をすることが幾度もあった[☆3]．なお，苦労して計算した 8 駒データベースは，別グループが独自に計算した終盤データベースの結果と一致しないために，2001 年に 1 カ月かけて再計算している．

終盤データベースの利用によって，探索アルゴリズムの仕事を大幅に減らせる．この効果は，探索の深さ 100 くらいの削減に相当すると推測している．実際に，10 駒データベースには，たとえば深さ 279 の探索が必要なゲーム値が保存されている．

☆2 Google マップによると，エドモントン・リバモア間は，2,700 キロである．56,000bps のネットワークで，合計 100MB 以上のデータを何度も送受信し，これがバンクーバー・シアトル間のネットワーク通信の 80 〜 90% を占めていた．さらに，テープとディスクを持って，車で研究所を訪問している．

☆3 文献 2）によると，Schaeffer は午前 2 時に起きて，データベースの計算状況をチェックするのが日課であった．

探索アルゴリズム

探索アルゴリズムは，**フロントエンド探索**と**バックエンド探索**からなる．フロントエンド探索は，初期局面からの探索木を管理し，現在の探索木で，証明木の構築に必要そうな未展開局面（**先端節点**）を選択し，バックエンド探索に引き渡す．バックエンドは，フロントエンドが選んだ先端節点を一定時間探索し，ゲーム値か勝ち負けの推測値（**評価値**）を返す．フロントエンドは，バックエンドの探索結果を受け取り，探索木を更新する．この過程を証明が終わるまで繰り返す．

バックエンド探索には，Chinook と**バックエンド・ソルバ**の 2 つがある．Chinook は，$\alpha\beta$ 法を利用して，評価値（または勝ち負け）を計算する．一方，バックエンド・ソルバは，ゲーム値または少なくとも負け（または勝ち）はないという部分的なゲーム値のみを計算する．バックエンド探索では，最初に，Chinook が 15 秒程度（状況によってはさらに延長する）で評価値を計算する．Chinook で勝ち負けが分からなければ，バックエンド・ソルバを 100 秒制限で呼び出す．バックエンド・ソルバでも，ゲーム値が分からなければ，評価値をフロントエンドに返す．

Chinook は，大体深さ 17 から 23 の探索を行う．一方，バックエンド・ソルバには，深さ制限がなく，有力な変化があれば，いくらでも深く読める．

フロントエンドには，**証明数探索** [3] を改良した手法 [1] を利用している．一方，バックエンド・ソルバには，長井の **df-pn 探索** [4] を利用している．df-pn は，詰将棋を題材にして生まれた日本独自の手法で，最良優先である証明数探索を等価な深さ優先に変換し，メモリおよび探索効率を改善している．

フロントエンド探索とバックエンド・ソルバに共通する重要な概念は，**証明数**と**反証数**である．証明数とは，現在の探索木を利用して，ある局面の先手勝ち証明にかかる最小の手間を見積もる指標である．証明数が小さければ，その局面は先手が勝ちやすいと推測できる．逆に，証明数とは双対の概念である反証数では，後手勝ちの証明にかかる最小の手間を推測する．

図 -4 を用いて，証明数を説明する．図形の中の数値が証明数である．この探索木において，A が先手勝ちであることを示すためには，$A \to B \to D \to I$ という経路で，I が先手勝ちであることを示すか，$A \to C$ という経路で，F と G と H の 3 つが先手勝ちであることを示すかの 2 通りの戦略がある．I のみを展開した方が，E と F と G の 3 つを展開するよりも労力が少なく見える．

フロントエンドとバックエンド・ソルバは，先手は証明数が最小，後手は反証数が最小の局面を選択して到達する未展開局面を展開し，証明数・反証数を再計算する

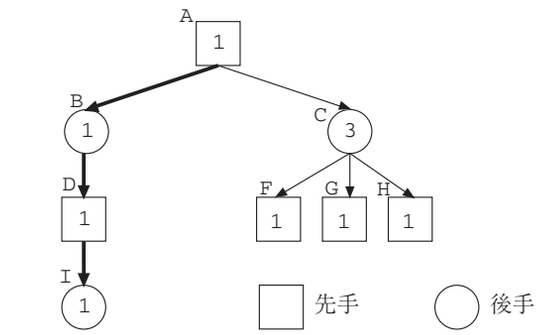

図 -4　証明数

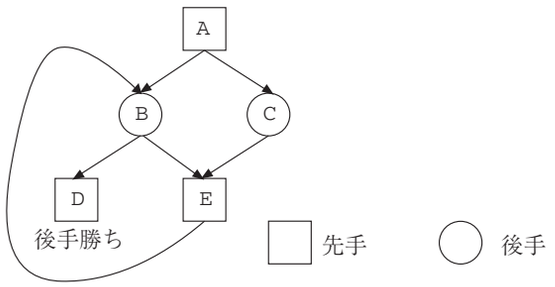

図 -5　GHI 問題

作業を繰り返す．さらに，フロントエンドでは，人間の過去のゲームより有力であると分かっている手順から最初に読むようにしている．これは，アルゴリズムとしては不要なものであるが，探索効率を上げるためには必要であった．

チェッカーに存在するややこしい問題に **GHI（Graph History Interaction）問題** [5] がある．チェッカーでは，別手順によって同一局面に至ることがある．ソルバは，局面をハッシュキーにしたハッシュ表を利用して，探索を効率化している．たとえば，局面 P の引き分けを探索が証明したとき，P をハッシュキーにして，ハッシュ表に引き分けを保存する．次に，別手順で P に至れば，ハッシュ表に P の引き分けがあるので，P 以下の探索は不要である．ところが，引き分けルールは，「繰り返し手順」で以前の局面に戻ることに依存している．ハッシュ表は，「手順を無視」し，ゲーム値を保存するので，実際には勝ち（または負け）の局面を誤って引き分けと判断することがある．

図 -5 に，GHI 問題の例を示す．同一局面に戻る場合は引き分けであると仮定する．この例では，$A \to B \to D$ および $A \to C \to E \to B \to D$ が後手勝ちであるので，A の正しいゲーム値は，後手勝ちである．ところが，ハッシュ表を利用し，次のように探索すれば，A は引き分けであると間違えて判定してしまう．

(1) $A \to B \to E \to B$ より，E は引き分けである．ハッシュ表に「E は引き分け」と保存する．
(2) $A \to B \to D$ は後手勝ちであるので，B は後手勝ちである．
(3) $A \to C \to E$ のとき，ハッシュ表には「E は引き分け」とあるので，C は引き分けである．
(4) B は後手勝ちで，さらに C は引き分けであるので，A は引き分けである．

この GHI 問題への解決策には，私の手法 [6] を利用している．

引き分け証明の正しさ

コンピュータによる証明が絶対に正しいと主張することは，原理的に不可能である．何年もかけて走らせた計算結果には，ソフトウェア・エラーだけでなく，ハードウェア・エラーの可能性もある．さらに，構築した証明木は，非常に大きい．フロントエンドの証明木の大きさは，約 10^7 = 1,000 万局面である．この証明木の各先端節点では，バックエンド探索（約 10^7 = 1,000 万局面の探索）が呼ばれているので，初期局面の引き分け証明に必要な証明木の大きさは，約 10^{14} = 100 兆局面である．この数字には，終盤データベースによって省略された証明木は，含まれていない．

結果の正当性を 100% 保証することできないが，次の理由から，まず正しいであろうと考えられる [1]．

- 終盤データベースに関しては，別研究グループの計算結果と比較している．8 駒データベースの結果は完全に一致しており，10 駒データベースの一部にも間違いは見つかっていない．
- 探索アルゴリズムについては，別研究グループとの比較は行っていないが，プロジェクト内で開発した複数のプログラムの探索結果は一貫している．
- 人間の強いプレイヤが証明木の一部をチェックしたが，間違いは発見されていない．
- 仮に探索の深い部分で間違いがあったとしても，それが最終結果に影響する可能性は著しく低い．

Chinook プロジェクト参加への経緯

チェッカーをプレイしたこともない私が，Chinook プロジェクトに参加することになった経緯を述べる．参加時期の関係上，私の直接知る話は，プロジェクト後半のみであるが，何卒ご容赦いただきたい．プロジェクト前半の詳細は，Schaeffer の著書 [2] にある．

Jonathan Schaeffer との出会い

私は，将棋とコンピュータが好きな少年だったので，将棋などの思考ゲームの実装方法に興味を持つのは，自然なことであった．当時の私には難しかったが，大学1年の頃に，サイエンス社の「コンピュータ将棋」という本を買って，勉強したりしていた．

進学した東大の理学部情報科学科には，同じようにコンピュータ将棋に興味を持つ友達がほかに3人もいた．棚瀬寧をリーダーとするIS将棋（東大将棋として商品化されている）の開発に参加することになり，チェスやチェッカーなどに使われるアルゴリズムの論文を読みあさっていた．感銘を受けた論文の多くは，アルバータ大学で書かれたものであり，著者には必ずSchaefferの名前が含まれていた．

運の良いことに，1999年に箱根で開かれたゲームプログラミングワークショップの招待講演者は，Schaefferであった．ワークショップ会場で，拙い英語でSchaefferに留学する交渉をした．Schaefferは，研究に対して非常にエネルギッシュな先生であり，ぜひアルバータ大学に留学して欲しいと言ってくれた．今から考えると，英語のできないどこの馬の骨かも分からない日本人学生をよく入学させてくれたと思う．

Chinook プロジェクト・メンバへ

Schaefferは，Chinookを開発する前にチェッカー解明の可能性を調べたが，当時は難しすぎた．また，1997年にチェッカーを解こうと試みたが，8駒データベースでは不可能であった．私がGAMESグループに入ったのは，Chinookプロジェクトが小休止していた2000年であり，当初はチェッカーとは直接は関係のない研究を行っていた．

Chinookプロジェクトとのかかわりが出てきたのは，博士論文用の研究を始めた2003年からである．研究内容は，囲碁の部分問題である詰碁を解くソルバの開発であった．詰碁ソルバの研究は，コンピュータ将棋での詰将棋ソルバ開発の経験を活かせる上に，チェッカーにも応用できた．同じ頃，Chinookプロジェクトでは，10駒データベースを作成していた．

詰碁ソルバには，詰将棋で大成功を収めていたdf-pn[4]を利用したが，すぐに性能に関する問題とGHI問題に出くわした．性能の問題は簡単に解決策[7]に気づいたが，GHI問題の解決に関しては，なかなか良いアイディアが浮かばなかった．実際，日本で詰将棋ソルバを開発していたときにもGHI問題があり，妥協的な対策を行っていた．2003年3月頃に，指導教官であるMartin Müller（GAMESグループの准教授で，コンピュータ囲碁の研究者）のある一言で，GHI問題解決への糸口をつかみ，囲碁に取り込んだ．

このGHI解決策はチェッカーにも使えると思い，Chinookプロジェクトのミーティングで話した．この頃のChinookプロジェクトでは，GHI問題を無視していたのだが，私の提案する手法は採用されなかった．当時の私のGHI解決策は，囲碁の実験では，従来のGHI問題を無視する場合よりも約2倍くらいは遅く，それに対する抵抗があったと思われる．

その後，しばらくは採用されなかったGHI解決策についての論文を書いていた．論文締切寸前の2003年5月末に，GHI解決策の改良点に気づき，興奮しながら，囲碁で実験した．実験では，わずか数パーセントのオーバーヘッドで，GHI問題を回避できることと，GHI問題が予想以上の頻度で起こることが分かった．2003年7月にGHI解決策について，再びミーティングで発表した．発表後に，Schaefferからチェッカー・ソルバに私の手法を実装することを勧められた．

バックエンド・ソルバの開発の話

Chinookプロジェクトでは，バックエンド・ソルバを開発した．元々のソルバには，アイスランド人の助手のYngvi Björnssonによって書かれた$\alpha\beta$法を利用したバックエンド・ソルバがあったが，GHI問題を無視していた．それを，GHI解決策を施したdf-pnに変更することが，私の役目になった．

実装には，私の詰碁プログラムを再利用した．プログラムから，囲碁の知識を取り除き，Björnssonの助けを借りながら，チェッカー依存の知識を追加した．ソルバ自体は，Cで書かれていて，機械生成されたヘッダファイルを除き，36,000行程度であった．そこに，約6,900行の私のC++のコードを追加した．

バックエンド・ソルバは，数カ月で動くようになったのだが，ソルバを数日間走らせると，バックエンド・ソルバが再現性がなく落ちた．何日もバグの原因が分からず，非常に苦しかったのだが，たまたま教えてもらったvalgrind[☆4]というツールによって，未初期化変数の利用が原因であることが，一瞬にして判明した．これ以来，常にvalgrindのお世話になっている．

Schaefferによると，私のバックエンド・ソルバは，Björnssonのものよりもわずかに性能が良いだけであった．そこで，Björnssonのバックエンド・ソルバにGHI解決策を施してみた．私が実際にコードを付加したのだが，正しくない証明木を構築する場合が頻発した．探索木を分析した結果，Björnssonのバックエンド・ソルバには，本質的な問題があり，それを解決できないことも

☆4 http://valgrind.org/ を参照．あまりの便利さにGAMESグループの他のメンバもすぐに使うようになった．

分かった☆5ので，私のバックエンド・ソルバを利用することに決定した．

チェッカーが解けるまで

ソルバが利用した計算機環境は，前述の10駒データベースを構築した環境や256CPUでメモリ256GBのSGIマシン，64CPUのIBMマシン，Opteronマシン（2.2GHz，メモリ8GB）×6台等である．アイドル時間のみを利用した環境も多い．ソルバ起動後は，私はただ待っていただけであるが，他のメンバは，フロントエンド探索や終盤データベース参照の効率化等を数度行っている．

バグの修正は，私の知る限りでは2回ある．バグの1つは，white doctorという有名なオープニングが解けた(結果は引き分け)ときに判明した．フロントエンドでは，駒割に7個以上差がある局面を終端局面であるとみなし，探索を省略していたというバグであった．この条件は，人間には明らかに正しいと思えるが，理論的に正しいとは限らないので，ソルバを修正し，対象となる局面を探索しなおした．

もう1つは，バックエンド・ソルバで落ちるバグがあった．2006年6月22日にメールを受け取り，私のせいで，2年以上かけた計算のやり直しなのかと緊張した．幸運にも，そのバグは非常に特殊な条件でしか起こらない上に，これまでの計算結果に影響を与えなかった．このバグの修正が，私の行った最後の変更である．バグを直したときには，非常にほっとした．

そして，2007年4月29日，Schaefferは出張先のカリフォルニアから大学にログインし，いつものようにソルバの計算状況を確認したが，すべてのソルバが停止していた．ソルバが不正終了したと思い，慌ててログをチェックしたが，ログ・ファイルには「Draw」と書かれていた☆6．

その日（日本時間では4月30日）の私は，ゴールデンウィークの家族サービスで，身重の妻と函館市内の温泉に滞在していた．その夜，妻の陣痛が始まり，翌日に3,532gの大きな（チェッカーの探索空間に比べると随分小さな数字であるが)元気な女の子が無事に産まれた．

おわりに

チェッカーを解くために取り扱った数字の大きさや何年も動きつづけるプログラムを書くことの困難さを感じていただければ，幸いである．

私自身は，運に恵まれたと感じている．日本人でありコンピュータ将棋に精通していたこと，留学したこと，チェッカーが解けるハードウェアがあったこと，GHI問題に取り組んだことのどれが欠けても，このプロジェクトに参加することはなかったであろう．

さて，次に解ける可能性のあるゲームは，日本人にもなじみの深いオセロであろう．オセロの探索空間は，10^{30}= 100穣局面もある．さらに，ゲームの性質上，終盤データベースを作れないので，しばらくは難しいのでないかと個人的には考えている．

謝辞　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，本稿に数々の有益なご提案をくださいました．また，松原仁様より，草稿へのさまざまなコメントをいただきました．Jonathan Schaeffer，Neil Burch，Steve Sutphen，Paul Lu，Robert Lakeの各皆様は，プロジェクトに関する質問にお答えくださいました．心より感謝申し上げます．

参考文献

1) Schaeffer, J., Björnsson, Y., Burch, N., Kishimoto, A., Müller, M., Lake, R., Lu, P. and Sutphen, S. : Checkers Is Solved, Science, Vol.317, pp.1518-1522 (2007). DOI: 10.1126/science.1144079.
2) Schaeffer, J. : One Jump Ahead : Challenging Human Supremacy in Checkers, Springer-Verlag (1997).
3) Allis, L. V., van der Meulen, M. and van den Herik, H. J. : Proof-Number Search, Artificial Intelligence, Vol.66, No.1, pp.91-124 (1994).
4) Nagai, A. : Df-pn Algorithm for Searching AND/OR Trees and Its Applications, PhD Thesis, Department of Information Science, University of Tokyo, Tokyo (2002).
5) Campbell, M. : The Graph-History Interaction : On Ignoring Position History, In 1985 Association for Computing Machinery Annual Conference, pp.278-280 (1985).
6) Kishimoto, A. and Müller, M. : A General Solution to the Graph History Interaction Problem, In 19th National Conference on Artificial Intelligence, pp.644-649, AAAI Press (2004).
7) Kishimoto, A. and Müller, M. : Df-pn in Go : An Application to the One-Eye Problem, In Advances in Computer Games 10, pp.125-141, Kluwer Academic Publishers (2003).

（平成19年8月28日受付）

岸本 章宏（正会員）　kishi@fun.ac.jp

1974年生．公立はこだて未来大学システム情報科学部情報アーキテクチャ学科助教．1997年東京大学理学部情報科学科卒業．2001年アルバータ大学コンピューティング・サイエンス科修士課程修了．2005年同博士課程修了．博士（理学）．人工知能，並列アルゴリズムに興味を持つ．東大将棋の開発に従事．IJCAI2005にて，Distinguished Paper Award受賞．

☆5 df-pnと$\alpha\beta$法の性能差がわずかである理由を分析していないが，この問題と関係があるのかもしれない．

☆6 http://cnews.canoe.ca/CNEWS/Rogers/Macleans/2007/07/25/4367082-mac.html を参照．