An Adaptation of Proof-Planning to Declarer Play in Bridge *

Ian Frank **David Basin** **Alan Bundy**

Department of Artificial Intelligence
University of Edinburgh
80 South Bridge
Edinburgh EH1 1HN
Scotland

Abstract. We present FINESSE, a system that forms optimal plans for declarer play in the game of Bridge. FINESSE generalises the technique of *proof-planning*, developed at Edinburgh University in the context of mathematical theorem-proving, to deal with the disjunctive choice encountered when planning under uncertainty, and the context-dependency of actions produced by the presence of an opposition. In its domain of planning for individual suits, it correctly identified the proper lines of play found in many examples from the Bridge literature, supporting its decisions with probabilistic and qualitative information. Cases were even discovered in which FINESSE revealed errors in the analyses presented by recognised authorities.

1 Introduction

The technique of *proof-planning* has been developed by the Mathematical Reasoning Group at Edinburgh University to find proofs for mathematical theorems [1]. In this technique, proof construction strategies are encoded in *tactics*, which are programs that construct proofs in the spirit of LCF tactics [6]. The preconditions and postconditions of these tactics are in turn specified in a *meta-language* to form *methods*. Planning consists of combining methods by reasoning about their specifications. In practice, the meta-level search space formed by reasoning about these specified strategies is many orders of magnitude smaller than the space at the *object-level* where proof construction consists of applying primitive inference rules.

The search spaces involved in mathematical theorem-proving and declarer play in Bridge are similar in that although they are both large and untenable by humans, experts are remarkably adept at navigating within them, having built up a large variety of strategies that they can draw upon to successfully manoeuver. Since proof planning has proved successful in theorem-proving by declaratively capturing such human problem solving strategies [3], we wanted to apply the same techniques to Bridge.

Bridge, however, presents new demands on proof-plans that are not encountered in mathematics, the most significant being the need to cope with an opposition. Since cards played may not be retracted once plan execution has begun, the value of a Bridge planning system hinges on its ability to judge the most beneficial course of action to take in any particular situation. This is not an issue in mathematical proofs, since one may always backtrack from failed branches until a successful proof is found. Furthermore, as information on the distribution of the opponents' cards is initially unknown, probabilistic information must be used to choose among possible plans. It is therefore the problem of making an *informed disjunctive choice* between various possible courses of action that was successfully tackled in building the FINESSE system.

FINESSE is a modular body of Prolog code consisting of a *pre-planner*, a *planner*, and an *interpreter*. The input to FINESSE is a description of a game state in a single suit[1]. Prior to actually initiating any planning, a small amount of pre-computation is carried out by the pre-planner in order to determine the significant missing high cards. The planner then takes the combination of this information with the original game state and uses proof-planning techniques to produce a plan that consists of a tree of tactics. These plans are not

*The research reported in this paper was supported by the SERC rolling funding grant GR/F/71799, an SERC Advanced Studentship to Ian Frank, an NSF/Nato fellowship to David Basin, and an SERC Senior Fellowship to Alan Bundy.

[1] In order to execute a plan successfully in Bridge it is important to have the leads in the right hands at the right times. This problem of *communication* is avoided in the current FINESSE system by restricting the planner to producing plans for a single suit only, implicitly assuming that communication can be maintained by executing plays in the suits not currently under consideration.

immediately executable, however, since they contain branching points at which a choice between available paths will have to be made. In order to make decisions under these circumstances, the plans are analysed by the interpreter, which produces qualitative statements about the subdivisions of the outstanding cards that would lead to the 'success' or 'failure' of the lines of play contained in the plan. This information is used to calculate probabilities upon which an informed choice can be made between different available courses of action.

FINESSE proved to be a very capable system, consistently identifying a 'best'[2] plan that concurred with the recommended lines of play found in the Bridge literature. In some cases, FINESSE's analysis of this plan even revealed shortcomings in the explanations presented by recognised authorities. These latter examples were generally situations in which authors had overlooked particular subdivisions of the outstanding cards that would lead to the plan succeeding. These 'discoveries' demonstrate the fallibility of humans, even when they are 'experts', in dealing with problems that involve combinatorial explosion. In FINESSE, as in theorem-proving, this explosion can be controlled by using proof-planning to combine high level strategies at the meta-level. In Bridge, this results in a search space typically four or five orders of magnitude smaller than the object-level space (i.e., the space that would result from considering all the possible ways the declarer could play his cards). With this reduction, analysis of the entire search space becomes computationally feasible.

The success of FINESSE provides further support for the value of proof-planning techniques. Moreover, it has given us valuable insight as to how this technique can be extended to 'adversarial domains' that necessitate disjunctive choice based on probabilistic or uncertain information. In addition, the creation of FINESSE's declaratively encoded methods formalised knowledge which was only previously expressed in the form of examples in the Bridge literature. This is analogous to the way in which the study of proof plans at Edinburgh has lead towards a theory of inductive theorem proving, and is in stark contrast to the poverty of the 'theories' underlying existing automated card players (see Section 6).

The remainder of the paper is organised as follows. Section 2 describes Bridge-specific tactics. Section 3 describes methods and how they are used in planning. Section 4 explains how plans can be assessed by a minimax-like algorithm. Section 5 gives examples of the use of FINESSE, including a discovery made by the system of an error in a Bridge textbook, and the final section discusses related work and draws conclusions.

2 Tactics

Bridge is played with a pack of cards, and requires four players (North, South, East, and West) who play in two teams of two (North/South and East/West). The cards are dealt between the players and they play 13 tricks, each consisting of them laying one card in turn in a clockwise direction. Before play begins there is an auction in which one team wins a contract to make a certain number of tricks. One member of this team becomes the *declarer* and the other is the *dummy*. The other team are the *defenders*. For simplicity, we will always assume that South is the declarer, so North is the dummy and East and West are the defenders. Dummy's cards are laid face up on the table after the first card is played by West, increasing the information available to all players. Declarer plays the dummy's cards as well as his own, and can use the increased information to select optimal plays in situations like the following.

♠ A Q

N
W E
S

♠ 2

Here, it is obvious that declarer will win one certain trick (in a No Trumps contract) when he plays the Ace. However, if he were to play this card first, his only chance of making two tricks would be if one of the defenders held the singleton King, so that it would fall when the Ace was played. This play of *cashing* the Ace will succeed in winning two tricks in only two of the possible 2^{10} distributions of the outstanding cards.

A better return is offered by entering the South hand and playing the two. By covering whatever card West plays, declarer can expect to win two tricks whenever West holds the King — a 50/50 chance. This line of play is based on the elementary principle of card play that the best results can be obtained by forcing an opponent to play ahead of you. It is a typical example of a very standard strategy called a *finesse*. When planning a hand, human players will make use of their knowledge of commonly occurring patterns like the finesse to avoid having to consider all

[2]There are competing ways that 'best' may be defined. In the Bridge literature, the most commonly adopted definition is the plan with the highest probability of making the maximum number of tricks. Hence, this definition is taken as the basis for our comparison.

the possible combinations of plays of single cards. FINESSE attempts to replicate this capability by restricting declarer's options at each stage of planning/play to a pre-determined set of such manoeuvers, or *tactics*.

The current version of FINESSE has a total of seven tactics, which it represents using the following Prolog predicates:

1 - 4. `finesse(Type, Player, Card, Suit)` — `Type` represents the type of finesse being used (four different types of finesse were identified when designing FINESSE; in the example above, the finesse of the Queen was a Type 1 finesse); `Player` is the defender being finessed; `Card` and `Suit` specify the finesse card.

5. `cash(Card, Suit)` — represents a trick on which declarer plays the card specified by `Card` and `Suit` from one hand, and plays a low card (or throws away a card from another suit) in the other.

6. `duck(Suit)` — represents a trick on which declarer plays low from both his hands.

7. `sequence(Card, Suit)` — represents a trick on which declarer plays the card specified by `Card` and `Suit` (which must come from a sequence of length 2 or more) from one hand, and plays low (or throws away a card from another suit) in the other.

3 Planning with Methods

FINESSE's tactics do not specify complete lines of play to be followed for any particular card combination, but only continuations for the next trick. Lines of play are built up by the planning algorithm which constructs a tree of tactics that resembles a minimax tree. In order to build this tree, the planner must be able to determine the tactics which are applicable to any state. This is achieved by specifying the minimal applicability preconditions for each tactic, producing a set of Prolog clauses of the form:

```
applicable(State, Tactic) :- PreConds.
```

To form a plan, an `applicable(+State, -Tactic)` goal is used to find a `Tactic` applicable to the current `State`, the possible responses by the defence are generated, and the post-conditions are determined. This process is continued recursively for the resulting states until branches for each applicable tactic have been generated. In Edinburgh's theorem-proving system (CLAM, [14]), the post-conditions and preconditions of an operator are combined together to form a *method* which in essence resembles a STRIPS type operator [4]. This kind of representation is ill-suited for planning in Bridge, however, because the results of applying a tactic depend on the particular response chosen by the defenders. For example, in the finesse tactic described in Section 2, the card played by the third hand depends on the card chosen by West. Wilkins [17] points out that such *context-dependency* often forces one to provide an operator for every possible situation in which an action might be taken. To avoid doing this, FINESSE uses a database of rules which, for each tactic, allows the declarer's cards to be chosen and played, taking into account the cards chosen by the defence. This deduction of context-dependent effects is very similar to the way SIPE [16] uses domain rules to alleviate problems in operator representation caused by the STRIPS assumption.

Note that there will in general be more than one tactic applicable to a particular state, and also that the defenders may make a number of different responses, leading to the kind of structure illustrated in Figure 1.

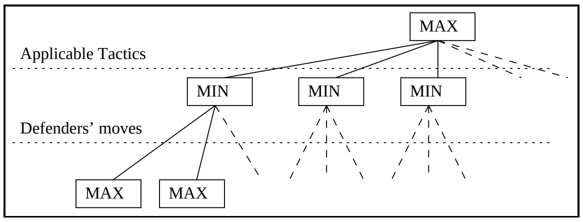

Figure 1: Sample Tree

The results presented in Section 5 illustrate that using tactics as the fundamental planning actions is far more efficient than considering the most fundamental operation: the play of a single card. However, the 'knowledge' contained in most Bridge books is generally in the form of examples. Hence, one of the challenges in constructing the FINESSE system was to encode the implicit strategies behind these examples into the form of tactics and to specify these tactics with preconditions and postconditions at the meta-level.

4 Interpreting the suit-plans

In order to execute the plans produced by the planner we need to have some way of selecting branches at MAX nodes. This is achieved by a bottom-up algorithm which first looks at the number of tricks won at each leaf node (a node at which no tactic's applicability conditions are satisifed). The MIN parent of a leaf node may have several leaf node daughters, but the defenders are restricted in their choice of which branch to follow by the actual distribution of the outstanding cards. For example, East must hold the King if the defenders are to choose a branch in which East plays it. Thus, each leaf node may only be reached under a subset of the possible distributions of the outstanding cards. FINESSE represents this circumstance by constructing an *interpretation-list* at the MIN node of the form

$$[0\text{-}F_0,\ 1\text{-}F_1,\ 2\text{-}F_2,\ \cdots],$$

where each F_n is a predicate formula describing the distributions under which the defenders could give declarer n tricks. There may, however, be some overlap between the distributions described by the formulae in an interpretation-list, so next the concept of *best defence* (that the defenders will always try to restrict declarer to as few tricks as possible) is implemented by updating the list so that all the distributions that are allowed by a formula F_i are removed from any F_j with $j > i$.

Selection of branches at a MAX node is based on the probabilities of the distributions described by the interpretation formulae of its daughter MIN nodes. The MAX node inherits the interpretation-list from its 'best'[2] MIN daughter and the algorithm recursively continues its pass through the plan, updating the information in the interpretation-lists as new cards are encountered.

An important feature of this algorithm is that it reasons qualitatively about card distributions, rather than using bare probabilities. This not only allows FINESSE to generate textual explanations for its plans' chances of success, but also keeps open the possibility of utilising inferences drawn from the bidding or from the play in other suits, should FINESSE be augmented by a bidding component, or extended to the task of planning a whole hand.

5 Sample Results

The following examples illustrate FINESSE's capabilities. The first row in each of the tables compares the size of FINESSE's plans to that of the object-level search space (olss)[3]. This reduction is crucial, since it allows the interpreter to analyse the entire space.

[3] The general formula for the minimum size of the search space when declarer has n_1 cards in one hand and n_2 cards in the other, with the defence holding d cards in the suit is $n_1!n_2!(d+1)!$.

A 6 4

N
W E
S

Q 7 5

Leaf nodes	plan: 75, olss:1.4×10^6
Critical card	King
Applicable tactics	Cash the Ace Finesse the Queen against East (Type 3) Duck a round

This example differs from the one in Section 2, in that declarer has additional low cards and the Queen is now opposite the Ace. Whereas before FINESSE would select the finesse tactic, this time it selects the cash. The interpretation formulae correctly show that cashing leads to two tricks not only when East has the King (a finesse is executed on succeeding trick), but also when West has the singleton King.

A Q 10

N
W E
S

4 3

Leaf nodes	plan: 285, olss:1.4×10^6
Critical cards	King, Jack
Applicable Tactics	Cash the Ace Finesse the Queen against West (Type 1) Finesse the 10 against West (Type 1)

FINESSE also copes well with situations in which a choice of finesses is available. In this example, FINESSE 'discovers' by itself the 'principle' that when it is possible to finesse more than one card in the opposite hand, the best chance of winning most tricks is by finessing the lower card first. Accordingly, the finesse of the 10 is chosen as the best play. FINESSE also produces correct analyses in situations where declarer holds so many cards in a suit that the 'principle' of finessing the lowest honour does not apply.

A J 9

N
W E
S

7 6 4

Leaf nodes	plan: 470, olss:4.4×10^6
Critical cards	King, Queen, 10
Applicable Tactics	Cash the Ace Finesse the Jack against West (Type 1) Finesse the 9 against West (Type 1)

Reese [10] and FINESSE both identify the finesse of the 9 as being the best play in this situation, but disagree on the probability of success. FINESSE's justification for its choice is that "This leads to 2 tricks if West holds at least one of the King or Queen and West holds the ten, or if West holds both the King or Queen, East holds the ten, and East holds the remaining four low cards." This explanation makes it an easy matter to check that it is Reese who has produced the incorrect analysis, having overlooked the distribution where West holds the doubleton King, Queen, thus introducing an error of $1/2^7$ into his result.

6 Related Work and Conclusions

Throop [13] traces the development of Bridge-playing programs (both those of a research nature and those resulting in a consumer product) as far back as 1957. His survey highlights the fact that the best Bridge-playing programs are of a low standard, especially when compared to the most successful chess and checker programs, which can compete at an international level ([5] and [11]).

Bidding seems to be the department of Bridge that lends itself best to play by computers [15]. Novel applications such as expert systems [8] and knowledge-based systems for locating missing high cards [9] do exist, and have proved successful at their intended tasks, but computer programs have yet to make any significant impact on the problem of card play itself.

Related to this poor performance, we suspect, is that many of the available Bridge playing programs are far from principled in their underlying methodology. Throop's description of his own program is typical in its level of detail, neglecting to enter into any deeper explanation than the following:

> "Behaviour of the program is heuristic in nature. The program evaluates trick objectives, formulates goals, and executes the play of the declarer's and the dummy's cards. The program performs in accordance with heuristic principles I have always practised in tournament play and which I incorporated into the computer program."

Throop goes on to claim that "the task of writing a Bridge-playing program that demonstrates a high level of intelligence is actually more difficult than writing a similarly intelligent chess program," — a sentiment echoed by Stanier [12], who puts forward four grounds for the rejection of the usual tree-searching approach when forming plans in Bridge.

Given this context, the results presented in Section 5 are particularly encouraging. Maybe the best way to summarise the design of our system system is to briefly outline how each of Stanier's objections is answered by FINESSE:

1. Stanier argues that any Bridge-playing program should not ignore the body of knowledge gained by Bridge-players in the past decades: FINESSE incorporates this knowledge in the declaratively encoded preconditions of its Bridge methods.

2. Stanier is concerned that the probabilistic results returned by an evaluation function would make minimaxing a complex task: by building up a qualitative picture of the distributions that would lead to the success of each line of play in a suit, FINESSE avoids the loss of information inherent in a probabilistic approach.

3. The use of the proof-planning paradigm controls search in a way that avoids unpleasantness such as the 'horizon effect' (there is no horizon, since the whole tree is searched).

4. Stanier argues that a successful Bridge-playing program will need to have concepts such as "finesse the Queen of clubs": these are exactly what is represented by the tactics in FINESSE.

From a proof-planning perspective, FINESSE has also provided us with some valuable insights. To date, proof-planning ideas have been successfully applied to the problem of finding proofs for mathematical theorems (CLAM), computer system configuration [7], and to program verification [2]. In Section 1, however, we noted that planning in Bridge (unlike in these other domains) requires an efficient mechanism for making informed disjunctive choice. With the implementation of FINESSE, we have developed such a mechanism, and also produced insights into the meta-level language necessary to deal with context-dependent actions. Since any attempts to apply the proof-planning paradigm to real-world problems such as business decision making will be unable to avoid these kinds of issues, it is to be hoped that the lessons learned from FINESSE will aid and direct any such endeavour.

References

[1] A. Bundy. The use of explicit plans to guide inductive proofs. In R. Lusk and R. Overbeek, editors, *9th Conference on Automated Deduction*, pages 111–120. Springer-Verlag, 1988. Longer version available from Edinburgh as DAI Research Paper No. 349.

[2] A. Bundy, A. Smaill, and J. Hesketh. Turning eureka steps into calculations in automatic program synthesis. In S.L.H. Clarke, editor, *Proceedings of UK IT 90*, pages 221–6, 1990. Also available from Edinburgh as DAI Research Paper 448.

[3] A. Bundy, F. van Harmelen, J. Hesketh, and A. Smaill. Experiments with proof plans for induction. *Journal of Automated Reasoning*, 7:303–324, 1991. Earlier version available from Edinburgh as DAI Research Paper No 413.

[4] R. Fikes, P. Hart, and N. Nilsson. Learning and executing generalized robot plans. In Nilsson and Webber, editors, *Readings in Artificial Intelligence*, pages 231–249. Tioga Publishing, Palo Alto, California, 1981.

[5] P.W. Frey. *Chess Skill in Man and Machine*. Springer Verlag, 1977.

[6] M.J. Gordon, A.J. Milner, and C.P. Wadsworth. *Edinburgh LCF - A mechanised logic of computation*, volume 78 of *Lecture Notes in Computer Science*. Springer Verlag, 1979.

[7] Helen Lowe. Extending the proof plan methodology to computer configuration problems. *Artificial Intelligence Applications Journal*, 5(3), 1991. In press. Also available from Edinburgh as DAI Research Paper 537.

[8] Y. Nygate. Python: A bridge expert on squeezes. Master's thesis, Weizmann Inst. of Science, Rehovot, Israel, 1984.

[9] J.R. Quinlan. A knowledge-based system for locating missing high cards in bridge. In *Proceedings of IJCAI-79*, pages 705–710, Tokyo, 1979.

[10] Terence Reese and T. Dormer. *The Play of the Cards*. Robert Hale Limited, 1991.

[11] A.L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–220, July 1959.

[12] A. Stanier. Bribip: A bridge bidding program. In *Proceedings of the 4th IJCAI*, Tbilisi, USSR, 1975.

[13] T. Throop. *Computer Bridge*. Hayden, 1983.

[14] F. van Harmelen. The CLAM proof planner, user manual and programmer manual. Technical Paper TP-4, Dept. of Artificial Intelligence, Edinburgh, 1989.

[15] A. Wasserman. Realisation of a skillful bridge bidding program. In *Proceedings of FJCC*, Houston, Texas, 1970.

[16] D. E. Wilkins. Domain-independent planning: Representations and plan generation. *Artificial Intelligence*, 22:269–301, April 1984.

[17] D. E. Wilkins. *Practical Planning*. Morgan Kauffmann Publishers, Inc, 1988.