Pragmatic Reasoning in Bridge*

Björn Gambäck
Swedish Institute of Computer Science
Box 1263
S – 164 28 KISTA, Stockholm
Sweden
gam@sics.se

Manny Rayner
SRI International
23 Millers Yard
Cambridge CB2 1RQ, U.K.
manny@cam.sri.com

Barney Pell
Computer Laboratory
New Museum Site
Pembroke Street
Cambridge CB2 3QG
bdp@cl.cam.ac.uk

Technical Report No. 299
University of Cambridge
Computer Laboratory

April 1993

Abstract

In this paper we argue that bidding in the game of Contract Bridge can profitably be regarded as a micro-world suitable for experimenting with pragmatics. We sketch an analysis in which a "bidding system" is treated as the semantics of an artificial language, and show how this "language", despite its apparent simplicity, is capable of supporting a wide variety of common speech acts parallel to those in natural languages; we also argue that the reason for the relatively unsuccessful nature of previous attempts to write strong Bridge playing programs has been their failure to address the need to reason explicitly about knowledge, pragmatics, probabilities and plans. We give an overview of PRAGMA, a system currently under development, which embodies these ideas in concrete form, using a combination of rule-based inference, stochastic simulation, and "neural-net" learning. Examples are given illustrating the functionality of the system in its current form.

---

*The research reported in this paper was undertaken from Summer 1989 to Summer 1991 at the Swedish Institute of Computer Science. Much of the content of the present paper, from Section 1.3 onward, appeared in almost the same form as reference [Gambäck *et al* 1990]. Since publication of this work, a number of advances in Computer Bridge Playing have taken place which largely follow the approach outlined here.

1 Introduction

Presumably the ultimate goal of research in "Natural Language Processing" is to create usable computational models of the processes underlying purposeful communication in unrestricted natural language. Unfortunately, it seems that this is still a long way beyond what it is feasible to attempt given the state of the art; any realistic research project must somehow simplify the problem. If we are prepared to accept the popular division of language into syntax, semantics and pragmatics, there are *a priori* two obvious choices:

1. Keep the syntax and the semantics realistic, but simplify the pragmatics.
2. Keep the pragmatics realistic, but simplify the syntax and the semantics.

Most work up to now has been concentrated on track 1; for example, one can assume that the input forms a connected story describing a visit to a restaurant, or that each sentence is an isolated question to a given relational database. Here we will examine the corresponding strategy from track 2, effecting a drastic simplification of the syntactic and semantic rules while keeping the pragmatics fairly realistic.

At the same time, we want to remove the other great obstacle to progress, the necessity of formalizing large chunks of "common-sense" knowledge. With these thoughts in mind, it is natural to look for a game of skill, which in an integral way involves communication through an artificial language. In the next section, we will argue that Contract Bridge is a good example of this type of game.

We will continue by describing why Bridge is hard and why previous attempts to write good Bridge playing programs have been fairly unsuccessful. We claim that many of the problems are caused by the need to reason about knowledge, probabilities and plans. Here we will describe a new architecture specifically designed to attack these problems: at the bidding level, the key idea is to use a combination of rule-based systems, stochastic simulation and neural nets, each of which take care of a different type of reasoning. Roughly speaking,

- the rule-based part of the program describes the bidding system
- the stochastic simulation part reasons about knowledge and probabilities
- the neural net makes evaluations

Examples are given of the way in which these ideas can be used to make decisions during the bidding phase, and describe how they are realized in a prototype system called PRAGMA. Our long-term plan is to interface the bidding component to a card-player, which is also briefly described.

1.1 A micro-world – bidding in Contract Bridge

Bridge is a four-person card game played between two two-person teams.[1] During the "bidding" stage, the players take turns to make "bids". A bid is either **Pass** or

[1]We regret that it would be quite impossible to provide a full description of the rules of Bridge. Readers unfamiliar with the game are urged to consult one of the many excellent introductory texts. We hope, however, that the introduction given should be enough to make the text comprehensible even to non-Bridge players, since the ideas are more generally applicable.

consists of a number from 1 to 7 followed by a suit denomination,[2] and each bid must be higher than the previous one according to the convention that $< N1, Suit1 >$ is higher than $< N2, Suit2 >$ iff either $N1 > N2$, or $N1 = N2$ and $Suit1 > Suit2$ in the ordering **NoTrump** $> \spadesuit > \heartsuit > \diamondsuit > \clubsuit$. When no player wishes to bid further (i.e. all players pass), the highest bid is regarded as an offer to play a "contract", in effect a bet that the player who has made the final bid $< N, Suit >$ can, together with his partner, take at least $6 + N$ tricks if $Suit$ is trumps.

What makes Bridge interesting from our perspective is, primarily, that it is in general impossible for any individual player to make such a bet with any confidence purely on the basis on the information derived from his own cards; it is consequently greatly to one's advantage to agree with one's partner that bids prior to the final one should be interpreted as conveying information according to some pre-arranged scheme. Such a scheme is called a **bidding system**, and can in a natural way be regarded as defining the semantics of a simple language.

From the point of view of pragmatics a bidding system is much closer to a natural language than to, say, first-order logic: this is largely a result of the constraint mentioned above, that each bid be strictly higher than the preceding one. Since a player can in most cases only bid a small number of times before reaching a dangerously high level, the bidding system must be designed so as to make maximum use of context-sensitivity, presupposition, shared information, and other linguistic devices aimed at increasing the efficiency of communication. Coupled with the fact that the syntax of the language is simplified to the point of non-existence, we get a system where the pragmatic aspects of communication can be studied almost in isolation.

A simple example of pragmatic Bridge reasoning in action follows. Consider the bidding sequence:

North	East	South	West
1♠	pass	2◇	pass
2NT			

In the bidding system most commonly used in Sweden [Nilsland 1978], North is only permitted to make his second bid if he has 11 to 14 high-card points, and a balanced suit distribution. This can be regarded as the overt semantic content of the bid 2NT in this context. However, a number of indirect inferences are available, if we assume that North is using his intelligence and is actively trying to help his partner. Among the most obvious are the following:

- North doesn't have a biddable heart suit
  (then he'd probably have bid 2♡).
- North doesn't have a rebiddable spade suit
  (then he'd probably have bid 2♠).
- (less certainly) North doesn't have diamond support
  (then he might have bid 3◇).

In each case, the fact that North refrained from saying something which could potentially be of use indicates that he was probably unable to say it, since it was

[2] In fact there are two more possible bids: **Double** and **Redouble**, which just raise the stakes.

untrue. Using a little imagination, we can construct a parallel dialogue in English to further illustrate the point:

A: Can I offer you some coffee?
B: Do you have any Earl Grey left?
A: I'll just nip out to the shop.

Here, the explicit content of A's second utterance is presumably a statement that he intends to go to the shop. However, it also seems reasonable to assume the following, given a lack of contrary evidence:

- A has no Earl Grey left
  (otherwise he would have accepted the suggestion).
- A is prepared to abandon his original plan to drink coffee
  (he made no attempt to defend it).
- (less certainly) A has no alternative brand of tea available
  (he might have offered it).

The sort of reasoning that occurs in the two examples is fairly similar: the big difference from the AI modelling point of view is that the second one will probably require orders of magnitude more domain knowledge to formalize properly, and is essentially beyond state of the art. We believe, however, that the Bridge domain is small enough to be formalized in a reasonably complete way. Later in this paper, we will describe what we believe to be a realistic strategy for building up this kind of coverage.

1.2 Different types of "speech acts" in Bridge

The bidding phase of Contract Bridge can be regarded as a series of communication acts using an extremely simple language: the bidding system. The two partners agree before-hand on the meaning of different bids in various situations. (and also inform the opponents of their agreements). Is this "language" then too simple to be of any interest to researchers in Natural Language Processing? In this section, we will give examples showing that this certainly is not the case, and that many commonly used speech acts [Searle 1979, Levinson 1983, Lyons 1977] have direct equivalents in the bidding system.

Examples of how some common speech acts can be performed in the context of Contract Bridge bidding:

1. Asserting. "By making this bid, I assert that my hand has these properties" (e.g. that I have a six-card Spade suit)
2. Denying. "By making this bid, I deny that my hand has these properties" (e.g. that I am interested in playing a major-suit contract)
3. Asking and answering questions. "If your hand is of type 1, make bid 1; if it is of type 2, make bid 2, etc." (e.g. asking how many aces partner has)
4. Interrupting. "The primary purpose of this bid is to stop the opponents communicating" ('pre-emptive bids')

5. Demanding/Ordering. "This bid forces you to do the following" (e.g. make a specific bid on the next round, irrespective of what cards you have)
6. Suggesting/Transferring responsibility. "This bid makes you responsible for the following decision" (e.g. bid game/slam if you judge that it is worthwhile)
7. Promising/Taking responsibility. "This bid promises that I can deal with these possible problems in some way." (e.g. promising that bidder has a stopper in a suit)
8. Lying (Perhaps not a speech act in the normal sense?).

Remember that the semantics of the bidding system is arbitrary, and is defined for maximum utility: having all these "speech acts" isn't accidental, but follows from the principle that they are all useful things that one will want to do. Also note that lying – i.e. making a bid conveying incorrect information about one's hand – is legal in Bridge, as long as one tricks one's partner as well as one's opponents.

Of course, a single bid can have more than one function, and thus fulfill more than one "speech act"; concretely, lies are always lies about something, e.g. an assertion or an answer to a question. The number of speech acts a given bid fulfils in a given situation is, of course, highly context dependent.

1.3 Why Bridge is hard – basic problems

In this section, we will summarize the basic problems that make Bridge a hard game to play. Firstly, it is necessary at almost every stage to reason about the other players' knowledge and beliefs: most bids are made in order to convey information, and it is not in general possible to compare the worth of two possible bids without at some stage reasoning about what the other players (especially one's partner) will know as a result of each bid having been made. Moreover, most knowledge is probabilistic in nature. Exceptionally, one may be in possession of an exact answer to a question (for example, how many aces partner has); but for more common attributes such as lengths of suits, it will normally be the case that there are two or three possibilities, and that the bidding will say something about their relative likelihood.

The points above apply equally to bidding and play: however, for bidding it is at least possible to achieve a certain level of performance with conventional programming methods. (We will quote some examples in the next section). For card-play, there is the added problem of having to be able to construct complex plans and counter-plans; human analysis of card-play situations almost invariably begins by constructing plans for the play in each suit individually, and then uses these as building blocks to construct a plan for the play of the entire hand. This type of analysis is considerably more difficult to model than the tree-searching algorithms used by, for example, most Chess programs.

In the next section, we review previous attempts to write Bridge programs. We will also give some of the reasons why we think that these approaches are unsatisfactory.[3]

[3]In the time since this article was originally published, many of the concerns we raised here have been addressed by other researchers. Two projects of note are the work of Ian Frank and colleagues [Frank *et al* 1992] on abstract reasoning in card-play, and the commercial bridge program DBASE III [DBASE III, 1992], which contains among other features a successful representative deal generator.

1.4 Previous work

Most systems written to date can be lumped under the broad heading of *naive rule-based systems*. By "naive", we mean that the rules are essentially to be viewed as constituting a procedure, rather than a description; this is particularly apparent in programs which encode bidding systems [Lindelöf 1983, Stanier 1975, Wasserman 1970]. However, the same is also true (though perhaps to a lesser degree) in, for example, Quinlan's system for locating high cards [Quinlan 1979], and the PYTHON system [Sterling & Nygate 1990], which identifies squeeze[4] situations.

A naive rule-based system, by our definition, attempts to provide rules to cover every eventuality that can arise. This strategy can prove highly successful in some types of situation; in particular, Lindelöf's COBRA appears to be a world-class bidder when it is allowed to bid undisturbed towards a slam, and Sterling and Nygate's PYTHON is capable of identifying any double-dummy squeeze with a small number of cards. Encouraged by these results, it has recently been suggested by David Levy [Levy 1989] that a combination of a COBRA-like rule-based bidding system and a brute-force card-player (probably implemented on specialized hardware), might be enough to produce a strong mechanical Bridge-player.

We think, however, that this is to be unduly optimistic; there seem in particular to be at least three serious problems that will arise if a system is to be built using this approach. Firstly, the bidding component will only have a very limited ability to reason about its opponents' bids; human players frequently ask questions about the meanings of bids, and use the answers to plan their own strategy. Being able to make use of this kind of information (or even to frame the questions needed to acquire it) without at some stage representing it abstractly seems difficult.

To take a concrete example: suppose that your left-hand opponent opens with 1♠ , your partner doubles, and your right-hand opponent redoubles. You know that the last bids are conventional (i.e. they convey more information than simply raising the stakes), but you are in doubt about the exact meaning of the redouble. This situation is sufficiently unusual that it is unlikely to be mentioned in the opponents' system declaration: conceivable interpretations might include "weak hand, long Spade suit", "8–10 high-card points, at least three Spades", "at least 10 high-card points, defensively oriented hand" or even "biddable Heart suit, moderate high-card strength". Not knowing which of these alternatives is the correct one will clearly be a major handicap when planning one's own next bid, yet adding specific rules to deal with every such situation is a daunting task.

Secondly, successful play (especially in defence) requires that the bidding component is able to pass on the (probabilistic) information extracted from the bidding to the playing component. Levy's suggested architecture for the card-player involves brute-force search for optimal plans in each of a large number of possible distributions of the cards; but if the system is unable to guess which of these distributions are the likely ones, there is no reason to suppose that it will play particularly well.

Let us once again look at a simple example. Suppose, as declarer, that South is evaluating two competing plans: one works if West possesses the ♣A, and the other if the Diamond suit is distributed so that East has at least four of the six cards out. In the absence of other information, the first plan is a 50% bet, and the second has less

[4]A squeeze is a particular type of card-play technique in which an opponent is forced to do a, for him, unfavorable discard.

than a 35% chance of succeeding. If, however, West has opened with a non-vulnerable pre-emptive bid of 3♠, one would normally think it unlikely that he had an Ace in another suit; since one expects seven of his cards to be Spades, it also becomes much more probable that he has less than half the Diamonds. Consequently, one would prefer the second plan. This is a relatively simple piece of deduction if one is able to reason abstractly about the kind of hand for which an opening bid of 3♠ is appropriate, but again becomes formidably difficult if it is done in terms of specific hand-coded rules.

Finally, there is a third problem: a system using brute-force double-dummy analysis is by its nature incapable of reasoning about other players' beliefs about its own hand. As declarer, this means that plays based on bluff can never be selected, even though this may be the only way to give the opponents a chance to go wrong. For example, with just small cards on the table against a singleton King in hand, it may pay to play a small card from the table and hope that it is not covered with the Ace; if this is done early on, it will usually be impossible for the right-hand opponent to work out that he has to to play his Ace. In defence, the issue is even more clear-cut. Two different plays might be equivalent if viewed by a double-dummy analysis, even though they in reality will convey different pieces of information to partner.

A brute-force playing component that attempts to construct plans for an entire hand at once is going to be highly inefficient. The search-space for a complete double-dummy analysis that takes partner's and opponents' views of its own hand into account is just going to be infeasibly large. Human players seem to reason about single suits first, then combine the plans to form a plan for the entire hands. This is a much more efficient approach, but even if this might not be the single feasible solution, one can at any rate deduce that brute-force double-dummy analysis won't work.

2 Proposed overall architecture

In this section, we will present an overview of a proposed architecture based on the ideas outlined above. The picture is as shown in Figure 1: the two main components are the subsystems for bidding and play.

The first key idea in the design is to maintain a clear distinction between logical and probabilistic information. Logical information comes from the bidding and play, and the definitions of the agreed-on bidding system. This is maintained as a set of declarative rules. The rules defining the bidding system take as input the observed bidding, and produce a set of *constraints*, which limit the possible card distributions which can explain the known facts: these are then in turn used as input to the probabilistic reasoning component, which, given a set of constraints C, uses a stochastic simulation procedure to produce an arbitrarily large set of random deals R for which C holds. (We will in the sequel refer to elements of these sets as "R-deals"). The larger the size of R, the more accurate the picture it gives of the system's probabilistic knowledge.

All communication between the bidding and playing components is in terms of constraints and sets of R-deals. Initially, the card-player is presented with a set of R-deals summarizing the information gathered during the bidding; if the card-player wishes, it can add extra constraints gained or hypothesized during the course of the play, and ask the stochastic simulator for a new set of R-deals which reflect the extra information.

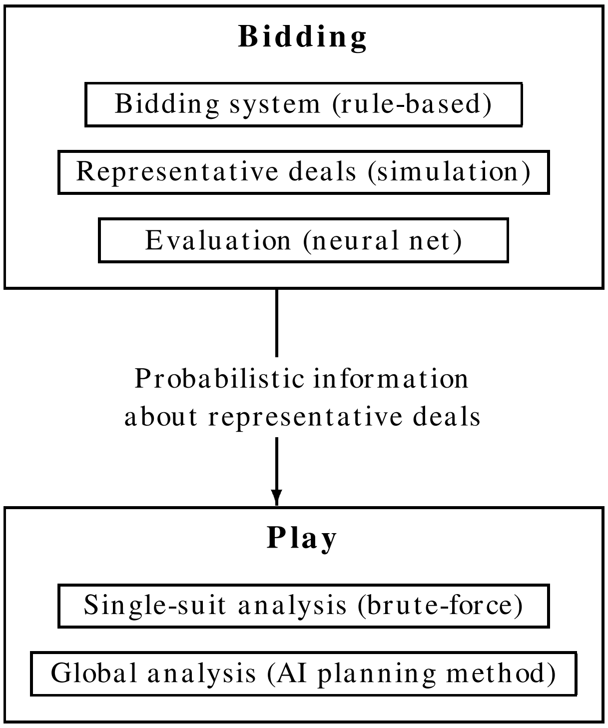

Figure 1: Overview of proposed architecture

Ideally, the link between the bidder and the card-player would be symmetrical: thus the bidder would also be able to ask the card-player what the likelihood would be of making a contract, given an actual deal and sets of R-deals which reflect the state of knowledge of each player. In practice, it seems that this is in general going to be far too expensive in terms of computational resources, and some "quick and dirty" evaluation method is required. In our initial prototype, we have chosen to implement this as a neural network: there are in fact two such nets, one for no-trumps contracts, and one for suit contracts. Each net is trained on a number of deals with known results, and is then capable of taking an unknown deal as input, and producing a set of fourteen outputs, representing the probabilities of making zero to thirteen tricks respectively.

Implementation of the bidding component of the prototype system is now reasonably far advanced, and we will decribe it in more detail in Section 3. In Section 4, we will sketch out the proposed design of the card-player.

3 Architecture of the bidding component

3.1 How the bidding works – an example

We will start by looking an illustrative example, in which the bidding system has to make a simple, but still non-trivial decision; the discussion in this subsection will avoid technical implementation details. Consider the situation in Figure 2:

	♠ K 9 7 ♡ K J 4 2 ♢ 10 9 7 4 ♣ 8 5	
♠ A J 8 2 ♡ Q 6 3 ♢ Q 8 6 ♣ A 9 6	N W E S	♠ 10 5 4 3 ♡ 9 5 ♢ K J 3 2 ♣ 10 7 2
	♠ Q 6 ♡ A 10 8 7 ♢ A 5 ♣ K Q J 4 3	

North	East	South	West
pass	pass	1♣	double
1♡	pass	?	

Figure 2: Making a decision in the bidding

South has opened with 1♣, which is meant as a natural bid showing a club suit and 13 points or more. West has doubled for take-out, indicating an opening hand and asking partner to bid his best suit; on this, North has bid 1♡, which should show something significantly better than a minimum response. (Since partner will have another chance to bid, he can afford to pass with a real minimum). East has passed, which presumably means that he has a very weak hand with no good suit; now South has to decide on his next bid. It is clear that he should be thinking about a contract in Hearts; without simplifying the picture too much, we can say that his only alternatives are either a) to bid game directly with 4♡, or else b) to *invite to game* with 3♡. What this means, intuitively, is that partner should bid 4♡ if he considers that his hand is significantly better than he has indicated by his preceding bids; otherwise he should pass.

An evaluation of the first alternative is quite straight-forward, since we expect it to end the bidding; all that is necessary is to estimate how likely it is (on the basis of the information given) that 4♡ will make. In practice, this will be done by using the stochastic simulator to generate a set of R-deals, and then getting an evaluation of each R-deal from the neural net for suit contracts; the expected value of the contract is then computed by multiplying the probability of each result by the score for making it, and averaging this over the set of R-deals.

The second bid is more interesting, since there are two possible continuations of the bidding: we need to reason about what decision North will make on his next bid. As in the first alternative above, we begin by generating a set of R-deals from the information available to us in the present situation. For each of these R-deals, we must decide whether North will consider his hand to be "significantly better than he has already indicated". Let us refer to a typical member of this set of R-deals as R1, and further use N1 and S1 to refer to the North and South hands in R1, S1 being the known South hand.

We now switch our point of view to North: we have assumed that his hand is N1. From this, together with the bidding, we can generate a set of R-deals which represent the possible hands North could think that South had. Once again, we call a typical member of the set R2, and let S2 be the South hand. (Remember that North's hand is assumed to be N1). The question now is whether N1 is a good hand from S2's point of view: thus we create a third nested level of R-deals (R3), this time assuming that the South hand is S2. As usual, we call a typical North hand N3.

We now want to know, for each N1, whether the combination N1+S2 is better than the average (over all N3's and S2's) of N3+S2, when playing contracts in Hearts. If it is, we assume that North would have bid 4♡ ; otherwise, we assume he would have passed. Assuming that this bid ends the bidding, we can now estimate the expected value of alternative b), in the same way as we did for a) above.

In this particular case, it turns out that bidding 4♡ at once is probably the right choice; 4♡ will often have a good chance of making, even if North has a hand which he would consider poor, and on which he would consequently pass 3♡ . In other words, we can calculate that North should not be encouraged to make the decision, because he is likely to decide wrong.

In the following subsections, we will describe in more detail how the various functionalities needed to realize the above analysis are implemented in our prototype version of the bidding component.

3.2 The bidding system

As we have described above, the definition of the bidding system in most earlier programs has been essentially procedural (perhaps excepting the treatment in [Quinlan 1979]); the rules take as input the bidding to date and the program's own cards, and produce the next bid. This is insufficient for several reasons. Firstly, we are not interested in creating a special, hand-crafted bidding system like Lindelöf's COBRA; we want to use a "natural" (non-conventional) system, where bids are defined in terms of common-sense concepts that assume an ability to reason and deduce.[5]

Even more importantly, we most often do not want to know which sequences are possible for given cards; we rather want to know what conditions obtain on the deal, if a given sequence is to be possible. In other words, the bidding system should be able to take the bidding history as input, and derive constraints on the cards each player must have held to make the bids he has made.

In Figure 3 we can see three examples of rules from the bidding system. The first is one of the rules that defines when a take-out double is legal (there are actually four more cases). This particular rule says that a takeout double is possible if the only non-pass bid so far has been a bid of 1 or 2 in a major suit, under the conditions that either our hand has a high-card strength of at least 13 points, or the other major is a biddable, but not doubly re-biddable suit.

---

[5]The program's bidding system is based on Modern Standard [Nilsland 1978]. MS is currently the most popular system in Sweden, and combines elements from the Standard American and British ACOL systems.

```
overcall(d,Bidding,Conditions) :-
     single_non_pass_bid_so_far(Bidding,bid(Level,Suit)),
     is_a_major(Suit),
     in_between(1,2,Level),
     other_major(Suit,OtherMajor),
     Conditions = [strength(13,maxPoints),
                   biddability(OtherMajor,1),
                   no(biddability(OtherMajor,3))].

response_to_1_in_a_suit(rd,[d,bid(1,Suit)|_],Conditions) :-
     real_suit(Suit),
     Conditions = [strength(8,10),
                   minLength(Suit,3),
                   balanced].

response_to_1_in_a_suit(rd,[d,bid(1,Suit)|_],Conditions) :-
     real_suit(Suit),
     Conditions = [strength(10,maxPoints),
                   balanced].
```

Figure 3: Examples of bidding rules

The second and third rules handle redoubles of take-out doubles at the one-level: the interpretation is that the redouble shows a balanced hand, with either 8 to 10 high-card points and at least three cards in the doubled suit, or more than 10 points.

In Figure 4 we see an example of a bidding sequence, together with some of the more important facts known by West after the bid of 3♡ (West has announced at least 5 points, lack of a fit in Hearts, and a biddable Spade suit; North has less than 12 points; East has a rebiddable Heart suit, and either 18 to 19 points with a balanced hand, or 16 to 20 points with an unbalanced one).

3.3 Generating representative deals

Now, the stochastic simulation module is briefly described; this is the component that receives sets of constraints, and outputs sets of R-deals. The basic mechanism is a simple version of the *reversed simulated annealing* algorithm, and is summarized in the flow-chart below (Figure 5). Initially, the unknown cards are distributed randomly between the three unseen hands. The simulator then enters a loop; at each iteration, it first checks to see whether all constraints are already satisfied, and if not tries to find a pair of cards which it can swap to improve the degree of fit, continuing until the hand satisfies the constraints.

The actual algorithm is a bit more sophisticated. Rather than always rejecting swaps which decrease the degree of constraint satisfaction, the algorithm rejects swaps with a probability which increases on progressive iterations (hence the analogy to simulated annealing). Also, the simulation continues making harmless swaps for a number of iterations after all constraints are met, to increase the randomness of resulting R-deals.

Figure 6 continues the example from Figure 4, and shows a typical set of R-deals produced by the simulation module: the situation is viewed from West's viewpoint,

		♠Q 10 9 7 ♡Q 8 7 4 ♢2 ♣9 5 4 2		
♠K J 7 3 ♡5 ♢A K 9 6 3 ♣J 8 6	W	N S	E	♠A ♡A J 10 9 6 2 ♢Q J 5 4 ♣Q 3
		♠8 6 4 2 ♡K 3 ♢10 8 7 ♣A K 10 7		

North	East	South	West
pass	1♡	pass	1♠
pass	3♡		

```
'W',[str(5,maxp),no(confirmed_fit('H')),biddability('S',1)].
'N',[str(0,12)].
'E',[str(18,19),balanced,biddability('H',2)].
'E',[str(16,20),unbalanced,biddability('H',2)].
```

Figure 4: Example of output from the bidding system

and we can see a set of ten possible East hands generated from the given constraints.[6]

There are several things to note about the above set of deals. Firstly, they correctly reflect the intuition that unbalanced hands are much more likely than balanced ones if the heart suit is long enough to be rebiddable, and that weaker (16–17 points) hands are more common than stronger (18–20 points) ones. More surprisingly, they show that the given constraints make it quite likely that East has a singleton or void in Diamonds. This is not so easy to explain, but seems to be a result of the requirement that East's hand should contain at least 16 points; since West knows he has the Ace and King, a long Diamond suit in East's hand will not contribute many points on average.

3.4 Evaluating possible contracts

Let us now look at the neural networks used for evaluating contracts; there are two of these, one for no-trump and one for suit contracts. Each net takes as input the sets of cards from the four hands (with the supposed declarer first), and outputs a set of 14 real numbers, representing the probabilitities that 0 to 13 tricks will be made. In the case of the net for suit contracts, the order of the suits is re-arranged so that the trump suit is first.

[6]It has been pointed out to us by Anders Ottosson that the constraints in this case are actually nowhere near tight enough to capture the information available from the bidding. Most importantly, they do not include two vital pieces of information: East cannot have any Spade support (otherwise he would have shown it), and it is also unlikely that he has a strong biddable Club suit. With these constraints added, the set of R-deals would be substantially different.

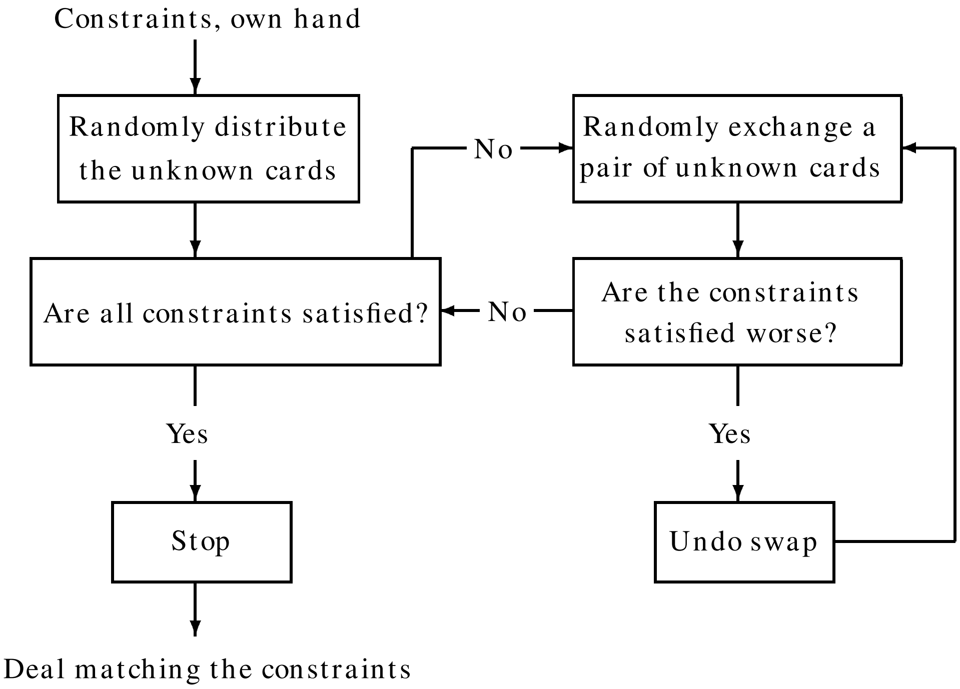

Figure 5: The simulation algorithm

Known West hand:	Constraints on the other hands:
♠ K J 7 3	North has less than 12 points
♡ 5	East has a rebiddable heart suit, and
♢ A K 9 6 3	either 18–19 points balanced
♣ J 8 6	or 16–20 points unbalanced

Possible East hands:

♠ A 10	♠ A Q	♠ A 8 4	♠ Q T 8	♠ A Q 8 6 2
♡ A K 10 3 2	♡ K Q 9 8 6 4	♡ A K Q J 10 7	♡ A K Q 10 3	♡ K Q J 8 3
♢ 10	♢ -	♢ 10	♢ -	♢ 5
♣ A Q 10 9 2	♣ A Q 10 7 5	♣ Q 9 3	♣ A Q T 7	♣ A 5
♠ Q 6 5	♠ Q 2	♠ A 10 8 4	♠ A	♠ Q 10
♡ K J 9 4 2	♡ A K Q 4 2	♡ A 10 9 4 3	♡ K Q J 9 7	♡ A K Q J 3 2
♢ J	♢ 10 8	♢ 7	♢ Q 8 7 4	♢ 7 4
♣ A K Q 5	♣ A K 5 3	♣ A K Q	♣ A 5 2	♣ A K 9

Figure 6: Typical set of R-deals

Neural nets in their purest form take the raw input data, and learn to construct appropriate outputs without doing anything more than recalculating the weights on the connections between their nodes. However, it is in practice considerably more efficient to perform a certain amount of pre-processing on the input, so as to construct values representing features which humans consider important in the domain; the system we have used as a model here is the neural net Backgammon-player described in [Tesauro & Sejnowski 1989]. Some of the input nodes represent these pre-computed feature points, PFPs. The PFPs are dependent on e.g. the high-cards, distribution, controls and shape. The other input nodes represent specific high-cards, or total suit-lengths.

The net consists of one input layer (100 nodes), two hidden layers (16 nodes each), and one output layer (14 nodes). Each node in the hidden layers is connected to each node in the layers above and below it (i.e. the net is fully connected). All the connections are weighted; additionally each node contain an internal threshold value. The output of a node is a function of the weighted sum of its inputs and internal threshold.

The nets are trained using the standard *back-propagation algorithm* (see for example [Rumelhart *et al* 1987, Lippmann 1987]). The weights on the connections are initialized to small random numbers. The nets are fed with a large number of (pre-processed) deals with known results. Actually, for each deal in the training set, the nets are fed with deals representing all permutions of the (non-trump) suits. The nets calculates their outputs, which are then compared with the results on the deals. The weights on the connections between the nodes are corrected accordingly from the output nodes backwards to the input nodes.

After a number of passes through the training set, the net has converged towards a global optimum. If not, the experiment is restarted with new (random) weights on the connections. When the net has converged, it is tested with new data, that were not in the original training set. Continuing the running example from Figure 4, Figure 7 shows the output from the neural net, queried for contracts in Diamonds with West as declarer. The figures at the bottom give the probabilities of making 0 to 13 tricks inclusive. As can be seen, the net estimates (quite reasonably, in our opinion) that 4◇ is as good as certain, 5◇ is a fair chance, and slam is more or less out of the question.

```
61  9 75 47
xx    5   x x  4        1     x  3
      1    x x 4    x   4        4
  xx  4  x     1  x  xx 6    x   2
    x 3        4   x    2  xx  x 4

100 100 100 100 100 100 100 100 100 100  91  78   5   0
```

Figure 7: Example of neural network output

3.5 The top-level — overall co-ordination of decision-making

At top-level, the task of the bidding component can be summed up in a few words: it has to consider its available information, and having performed a reasonable amount of analysis decide which of the possible bids appears to be the best one. What "a reasonable amount of analysis" might be is, naturally, dependent on context, but in a practical system there will at any rate be two obvious reasons to stop thinking: either the best bid is already obvious, or a time-limit has been exceeded. The time-limit can either be fixed, or can be implemented more subtly, so that the system can attempt to weigh up the expected gain in decision quality against the loss of time, while keeping to an overall average time-limit for its bids. This idea has been researched in depth by Russell and his colleagues, [Russell & Wefald 1989], and may be incorporated into a future version of our prototype.

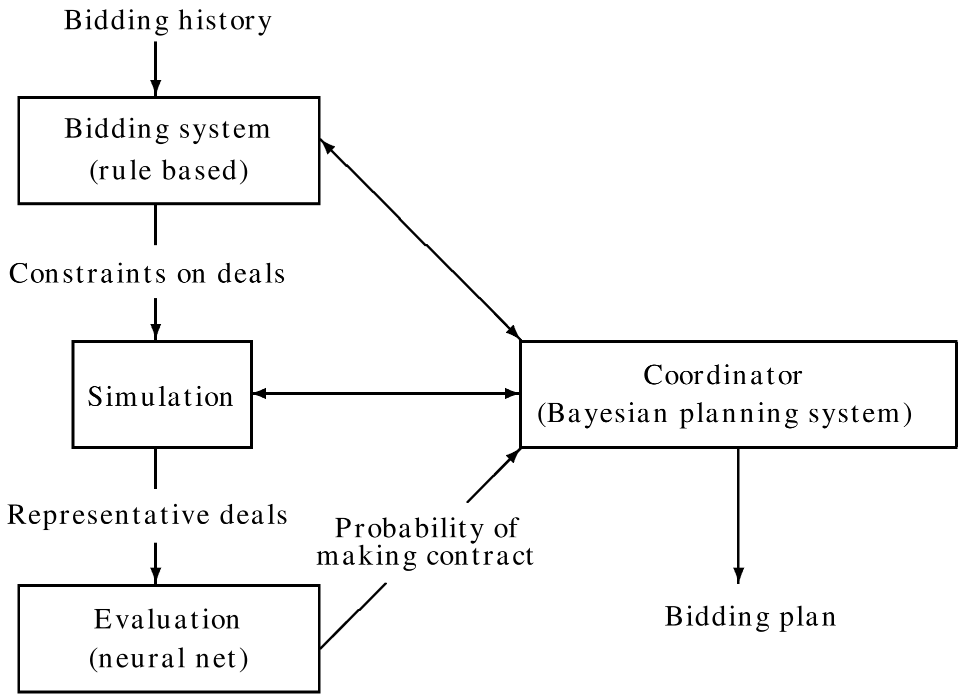

Figure 8: Architecture of the bidding component

A flexible way of attempting to achieve the desired functionality is to structure the top-level as a "refinement loop": at any moment, it is possible to stop thinking and read out the relative scores assigned to the possible options, selecting the one currently given the top rating. There are a number of "knowledge sources" (KS's), which can be used to try and improve the accuracy of the system's evaluation of each bid's worth; the simplest ones give a rough-and-ready estimate at low cost, while the more sophisticated ones take an appreciable time to perform a detailed analysis. Below, we will describe three simple KS's, which are currently being implemented.

Before weighing up the possible alternatives, the bidder must ascertain what they are. The first step in the decision-making process is thus to call the rule-based definition of the bidding system, to determine the bids legal in the present situation. It can of course be the case that only one bid is allowed: for example, partner may have asked for aces with Blackwood, or made a transfer bid which forces a specified

S	South	West	North	East
♠ K 10 8 5 3	1♠	pass	1NT	pass
♡ 9 4	?			
♢ A 10 7 3				
♣ K Q				

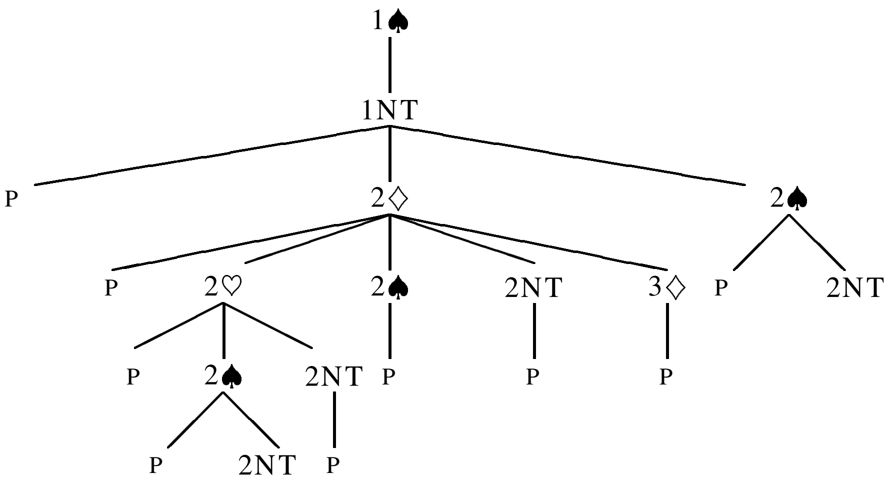

Figure 9: Bidding tree example

response. If this is so, no further thought is necessary. In general, however, there will be two or more bids to choose between.

In contrast to games like Chess, normal tree-searching is an extremely expensive KS, and can rarely be used; the problem is that there are a great many alternatives (since the other players' cards are unknown), and a probabilistic calculation over all these possibilities will normally be infeasibly complex. However, various kinds of cheap tree-searching are still possible, and very useful. Often, it is quite enough simply to consider the possible final contracts that can arise after a candidate bid, without worrying about their relative plausibility. We can approximate this reasonably well by generating a small set of R-deals, and running the bidding system non-deterministically on each of them; let us refer to the possible set of final contracts that a candidate bid can lead to as the *choice set* for the bid.

Thus in the example shown in Figure 9, the choice set for the bid "Pass" is { 1NT }; for 2♢ it is { 2♢ , 2♡ , 2♠ , 2NT , 3♢ }; and for 2♠ it is { 2♠ , 2NT }.

Using these definitions, we can describe our first three KS's, here presented in order of increasing complexity.

3.5.1 Knowledge source 1: simple domination

The *simple domination* KS is based on the following elementary observation. Suppose that we have two alternative bids, X and Y, and that the choice sets for them are B_X and B_Y respectively. If we can ascertain that the worst result in B_X is strictly better than the best result in B_Y, then X must be preferable to Y, irrespective of how play develops; a further analysis of the tree is thus unnecessary. The most obvious example of this kind of situation is that in which the partnership has reached game, and there is clearly no reasonable chance of making a slam; any bid other than "pass" will only result in a contract that is harder to make.

Simple domination can be estimated by creating a set of R-deals, and evaluating (with the help of the neural net) the worth, in each R-deal, of each contract in the choice sets for the various bids. The criterion is thus that the contracts in one choice set outperform *all* the contracts in *all* the other choice sets in every R-deal; this means, naturally, that these contracts cannot occur in any other choice set.

3.5.2 Knowledge source 2: other players deterministic

Tree-searching is usually much more complex in Bridge than in Chess, since one needs to take into account the beliefs the other players hold when they make their decisions. If, however, it is possible to assume that the bidding system will not give the other players any choices, then the problem is simplified sufficiently that normal search methods will be applicable. Situations of this kind can most easily occur when one is sure both that the opponents will pass, and that partner is restricted to forced bids with conventional meaning; this is often the case when bidding towards slam.

Suppose, then, that analysis of the bidding tree shows that the other players' bids will be deterministic in each branch. We take the set R of R-deals used to compute the tree, and assume that each of these is equally probable. Now we recursively descend the bidding tree, starting from the root, and annotate each node N with the subset of R relevant to N; that is to say, the set of R-deals in R for which bidding could potentially reach the situation N denotes. Call this set $R(N)$. Each node N in the tree that represents a choice made by another player will thus involve a partition of $R(N)$ into a set of mutually disjoint subsets, one for each daughter.

When the tree has been fully annotated, we can estimate the expected value of each choice by a modified minimax calculation. As usual, we work backwards, starting with the leaves. Each leaf L of the bidding tree will be labelled with a contract $C(L)$ and a set of R-deals $R(L)$; its worth can thus be computed as the average over $R(L)$ of the worth of $C(L)$ in each of its members.

When backing up values, there are two separate cases to consider, depending on whether the bid is made by ourselves or by another player. In the first case, we can assume that we have a free choice between the different alternatives, and choose the best one; thus the situation is as in normal minimax, and the value of the node is the maximum of the values of its daughters. In the second case, we have assumed that choice is deterministic; the uncertainty derives from the fact that the player's cards are unknown, rather than from his freedom to make a choice. Thus the value of the node is not the minimum, but rather the weighted average of the values of its daughters, where each daughter is weighted by the number of R-deals associated with it.

3.5.3 Knowledge source 3: players other than partner deterministic after next bid

This KS is an elaboration of the previous one. We want to cover situations where we transfer control to partner: the simplest case, naturally, is where we just ask him to make a decision on the final contract. We can do this by first creating a set of

R-deals, and then for each R-deal expanding the bidding tree until we reach partner's next bid; at this point, we switch our viewpoint to partner's hand, and use KS 2 to evaluate the worth of the node. (By assumption, this is possible). As usual, we can now take the average over all R-deals of the evaluation after each bid, to pick the one that gives the highest score.

It is worth noting that KS 3 is very much more expensive to apply than KS's 1 and 2, since it involves nested generation of R-deals; to keep resource expenditure within managable bounds, it will probably be necessary to generate fewer R-deals at the "inner" level than at the outer one.

KS's 1, 2 and 3 are naturally only meant as a beginning; we present them primarily to show that it is possible, using our framework, to construct tests, usable in common bidding situations, which will be able to make sensible evaluations of competing alternative bids.

4 Proposed architecture of the playing component

Work on the card-play component is still at a preliminary stage, and this section is not meant to provide more than a rough sketch of our ideas to date; a prototype based on these notions is currently being implemented by Anders Ottosson, and will be reported elsewhere.[7]

The fundamental intuition we want to capture is that play (especially declarer play) is hardly ever planned as a single process; human experts first consider play in each individual suit on its own, and only then proceed to form a plan for the whole hand. Although it is by no means a simple matter to write programs which function in a similar way, recent work by Campbell and Berliner has shown that such an approach is at least feasible for Chess pawn endings [Berliner & Campbell 1984, Campbell 1988] in this type of chess position, the concept analogous to "suit" is that of a "pawn island".

Let us now look at an illustrative example of how the Campbell/Berliner methology might be adapted to Bridge; to keep things as simple as possible, we restrict ourselves initially to double-dummy declarer play. We begin by noting that play in one suit can only influence play in another in the following ways:

Entries:
: $Suit_1$ can be used as an entry for $Suit_2$, so that play in $Suit_2$ can shift discontinuously from one hand to another. (This includes losing the lead to the opposition, intentionally or otherwise).

Discards:
: Cards from $Suit_1$ can be discarded on those from $Suit_2$, if a hand is unable to follow suit.

Ruffs:
: Cards from the trump suit can be used to ruff cards from $Suit_1$, if a hand is unable to follow suit.

[7] Ian Frank and colleagues [Frank *et al* 1992] have subsequently demonstrated the feasability of an approach similar to the one sketched in this section.

Tricks, entries, ruffs and discards can be viewed as "resources" which a single-suit line of play can either "produce" or "consume".

```
          ♠8654                   ♡AK
            N                       N
    ♠32   W   E ♠KJ9      ♡6542   W   E ♡J1093
            S                       S
          ♠AQ107                  ♡Q87
```

Figure 10: Example single-suit distributions for Spades and Hearts

Consider now the distribution in Spades shown in the left half of Figure 10. There is a line of play in which South consumes two entries to dummy, while producing four tricks, two entries to hand, two discards from West, and one discard from East; there is also another, in which South loses the lead twice to East and regains it each time, to produce two tricks. Without knowing the situation in the other suits, it is obviously impossible to say which of the two plans is the "better" one. Note also that the resources have to be produced and consumed in a given order: for the first plan, this can be written as

```
<consume(entry(n)),
 produce(entry(s)),
 consume(entry(n)),
 produce(entry(s)),
 produce(discard(w)),   produce(trick),
 produce(discard(w)),   produce(discard(e)),   produce(trick))>
```

Now look at the Heart suit on the right: here, there is a plan that produces three tricks, two entries to dummy, and a discard from dummy, while consuming three entries to hand. (There are clearly other plans to produce three tricks as well, which consume fewer entries to hand; these are ignored here). The sequencing of resource production and consumption here will be:

```
<produce(entry(n)),
 consume(entry(s)),
 produce(entry(n)),
 consume(entry(s)),
 produce(discard(n)),   produce(trick)>
```

The point worth noting with the example is that it is now possible to find a plan which will produce seven tricks from the two suits, by just "matching up" the two partial plans on the principle that a "produce" in one matches a "consume" in the other; the basic claim being that finding plans in this way will normally be much cheaper than performing a single search over play in all suits simultaneously.

Thus the search process, in outline, will be something like the following:

1. Carry out a conventional search of each individual suit, and keep the "best" lines of play. Note that "best" need not necessarily mean "taking most tricks", since it can often be worthwhile to lose one or more tricks in order to establish or save an entry, set up a discard, or avoid being thrown in. Play can include

consumption of "resources" in the sense defined above; one of the problems to be solved is how to avoid analysis of lines of play in which obviously unrealistic quantities of resources are consumed.

2. Analyze the search trees constructed during phase 1, and annotate each possible single-suit plan with its relevant properties. Among these would be the number of tricks taken, the total production and consumption of resources, and the sequence in which production and consumption events occurred.

3. Guided by the summary information acquired during phase 2, combine suit plans into a global plan.

5 Current state of the implementation

Implementation of the bidding component of the prototype system is reasonably far advanced, and some basic functionality has been achieved. Most components are encoded in Prolog, although some of the code for the simulator and the neural nets is implemented in C. In summary:

- The semantics of most of the bidding system have been formalized.
- A general stochastic simulation system has been constructed.
- A first version of the neural network(s) exists.

The implementation of the card play component, which will also be written in Prolog, has just started.

6 Conclusions and future work

This work has described why Bridge is hard and an interesting AI topic, and how a "bidding system" in a natural way can be regarded as a simple language. Previous attempts to write good Bridge playing programs have been fairly unsuccessful, and we have claimed that this is because they have not attempted to properly model the necessary reasoning involved. In contrast, we have outlined an architecture specifically designed to address the vital problems in the Bridge domain: knowledge, probabilities and plans.

The architecture that we have described bids by using a rule-based bidding system, a stochastic simulation of deals and a neural net for contract evaluation. We have also presented a sketch for a playing component which uses single-suit brute force analysis and global analysis by plan combination.

As far as future work goes, the most pressing requirement at the moment is clearly a faster implementation of the simulator, which is still more than an order of magnitude too slow to be practically useful. Without fast generation of R-deals, it is difficult to develop any very sophisticated knowledge sources for the control component of the bidding system. It also appears necessary (as pointed out in footnote 4 above) to pay more attention to representation of "negative" information in the constraints output by the bidding system.

Acknowledgements

The work described in this paper was influenced by several fruitful discussions with Annika Wærn and Anders Ottosson. Mats Nygren helped train the neural net, and greatly improved its performance when he located an obstinate bug in the code. Peter Cheeseman and Carl Witty provided useful input on the construction of the simulator.

The project was funded by the Swedish Institute of Computer Science.

References

[Borel & Chéron 1940]
Borel, E. and Chéron, A. 1940. *Théorie Mathématique du Bridge – A la Portée de Tous*. Paris, France: Gauthier-Villars (in French).

[Berlekamp 1963]
Berlekamp, E.R. 1963. "Program for Double-Dummy Bridge Problems – A New Strategy for Mechanical Game Playing", in: Levy, D. (ed.) *Computer games II*. New York, New York: Springer Verlag.

[Berliner & Campbell 1984]
Berliner, H. and Campbell, M.S. 1984. "Using Chunking to Solve Chess Pawn Endgames". *Artificial Intelligence* **23**.

[Campbell 1988]
Campbell, M.S. 1988. *Chunking as an Abstraction Mechanism*. Ph. D. Thesis, Carnegie-Mellon University.

[DBASE III, 1992]
DBASE III Bridge Program, 1993. Reviewed in Bridge Magazine, February, 1993.

[Epstein 1977]
Epstein, R.A. 1977. *The Theory of Gambling and Statistical Logic*. San Diego, California: Academic Press.

[Frank *et al* 1992] Frank, I., Basin, D. and Bundy, A. 1992. "An Adaptation of Proof-Planning to Declarer Play in Bridge", in: Proceedings of the European Conference on AI.

[Gambäck *et al* 1990]
Gambäck, B., Rayner, M. and Pell, B. 1991. "An Architecture for a Sophisticated Mechanical Bridge Player", in: D.F. Beal and D.N.L. Levy (eds.), *Heuristic Programming in Artificial Intelligence – The Second Computer Olympiad*. Chichester, England: Ellis Horwood.

[Laskey & Lehner 1989]
Laskey, K.B. and Lehner, P.E. 1989. "Assumptions, Beliefs and Probabilities". *Artificial Intelligence* **41**:65–77.

[Levinson 1983]
Levinson, S.C. 1983. *Pragmatics*. Cambridge, England: Cambridge University Press.

[Levy 1989]
Levy, D.N.L., 1989. "The Million Pound Bridge Program", in: D.N.L. Levy and D.F. Beal (eds.), *Heuristic Programming in Artificial Intelligence – The First Computer Olympiad*. Chichester, England: Ellis Horwood.

[Lindelöf 1983]
Lindelöf, E.T. 1983. COBRA – *The Computer-Designed Bidding System*. London, England: Victor Gollancz.

[Lippmann 1987]
Lippmann, R.P. 1987. "An Introduction to Computing with Neural Nets". *IEEE ASSP Magazine* **April**:4–22.

[Lyons 1977]
Lyons, J. 1977. *Semantics: 2*. Cambridge, England: Cambridge University Press.

[Moore 1985]
Moore, R. 1985. "A Formal Theory of Knowledge and Action", in: Hobbs, J. and Moore, R. (eds.) *Formal Theories of the Commonsense World*. Norwood, New Jersey: Ablex Publishing Corp.

[Nilsland 1978]
Nilsland, M. 1978. *Modern Standard*. Malmö, Sweden: Svenska Bridgeförlaget (in Swedish).

[Pearl 1985]
Pearl, J. 1985. *Heuristics – Intelligent Search Strategies for Computer Problem Solving*. Reading, Massachusetts: Addison-Wesley.

[Pearl 1988]
Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, California: Morgan Kaufmann.

[Quinlan 1979]
Quinlan, J.R. 1979. "A Knowledge-Based System for Locating Missing High Cards in Bridge", *the 6th International Joint Conference on Artificial Intelligence*, Tokyo, Japan, pp. 705–707.

[Root & Pavlicek 1981]
Root, W.S. and Pavlicek, R. 1981. *Modern Bridge Conventions*. New York, New York: Crown.

[Rumelhart *et al* 1987]
Rumelhart, D.E., Hinton, G.E. and Williams, R.J. 1987. "Learning Internal Representations by Error Propagation", in: Rumelhart, D.E. and McClelland, J.L. *Parallel Distributed Processing, Vol. 1*. Cambridge, Massachusetts: MIT Press.

[Russell & Wefald 1989]
Russell, S. and Wefald, E. 1989. "Principles of Metareasoning". *the 1st International Conference on Principles of Knowledge Representation and Reasoning,* pp. 400–411.

[Searle 1979]
Searle, J.R. 1979. *Expression and Meaning – Studies in the Theory of Speech Acts*. Cambridge, England: Cambridge University Press.

[Stanier 1975]
Stanier, A. 1975. "BRIBIP: A Bridge-Bidding Program". *the 4th International Joint Conference on Artificial Intelligence*, Tbilisi, Soviet, pp. 374–378.

[Sterling & Nygate 1990]
Sterling, L. and Nygate, Y. 1990. "PYTHON: An Expert Squeezer". *Journal of Logic Programming* **8**:21–40.

[Tesauro & Sejnowski 1989]
Tesauro, G. and Sejnowski, T.J. 1989. "A Parallel Network that Learns to Play Backgammon", *Artificial Intelligence* **39**:357–388.

[Turner 1990]
Turner, R. 1990. *Truth and Modality for Knowledge Representation*. London, England: Bibbles.

[Tzeng 1988]
Tzeng, C-H. 1988. *A Theory of Heuristic Information in Game-Tree Search*. Berlin, Germany: Springer-Verlag.

[Wasserman 1970]
Wasserman, A. 1970. *Achievement of Skill and Generality in an Artificial Intelligence Program*. Ph. D. Thesis, University of Wisconsin.