An ATMS-Based Tool for Locating Honor Cards in Rubber Bridge

Ali Awada¹, May Dehayni², Antoun Yaacoub³

¹, ²Department of Applied Mathematics, Faculty of Sciences, Lebanese University, Hadath, Lebanon
³Institut de Recherche en Informatique de Toulouse, 118, route de Narbonne, 31062, Toulouse Cedex 9
al_awada@ul.edu.lb, maydehayni@ul.edu.lb, Antoun.yaacoub@irit.fr

ABSTRACT

This paper deals with the problem of providing assistance to a Bridge player using Artificial Intelligence techniques. In order to achieve this goal, we propose an approach that consists in making assumptions about the honor cards location, then to gradually eliminating the inconsistent assumptions until locating all honor cards. Based on the bids, assumptions about honor cards distribution are made using abduction reasoning. The consistency of these assumptions is managed by an Assumption-based Truth Maintenance System (ATMS). As the game progresses, playing a card generates facts, thereby invalidating some assumptions that cause contradiction and transforming some others into facts corresponding to the location of honor cards.

Keywords: Bridge, Honor card location, ATMS, Abduction Reasoning, Consistency management.

1. INTRODUCTION

The bridge is one of the most interesting games for artificial intelligence researchers in game theory. It presents several challenges such as decision making with partial information in very large search spaces, dealing with an opponent agent, and competition and cooperation between agents in a restricted communication frame.

Our concern in this paper is to elaborate a tool to assist a player to locate honor cards during a play session. This tool uses an ATMS to deal with incomplete information in the dynamic environment of the game in a play session. It uses abduction reasoning about incomplete information to speculate a complete picture of the environment by making assumptions, which must be consistent with each other and with the available information on the located cards and the bids. Since the play environment is dynamic, the arrival of new facts transforms some assumptions into facts and calls into question some others so that the consistency of the world is maintained at all times.

The paper is structured as follows. We start with a brief background on the bridge game. In section 3, we review the main works in three areas related to our research, namely abduction, Truth Maintenance System (TMS) and bridge game. In section 4, we follow with a description of TMS in general and ATMS in particular. We then present our approach in detail in section 5: The system components, the application design and the algorithm of card location are discussed. In section 6, we illustrate, on an example, how our application acts to locate honor cards. Finally, we evaluate our tool and propose some concluding remarks and discussion of future works.

2. THE BRIDGE

In 1920, E. Culbertson is widely regarded as the man who made Rubber Bridge [1]. The game is played with four players; it involves a high degree of skill but there is also a fair amount of luck involved in who gets the best cards. We will introduce briefly the game and adjust some rules and constraints in order to present our application.

Four players participate, two against two in partnership: North and South play against East and West. They play with a 52-card pack; the dealer distributes all the cards one at a time so that each player has 13. Turn to deal rotates clockwise. When play begins, the object is to win tricks, consisting of one card from each player in rotation. The cards in each suit rank from highest to lowest: A K Q J 10 9 8 7 6 5 4 3 2. The 52 cards are distributed in four trumps: Spade ♠, Heart ♥, Diamond ♦, and Club ♣. By decreasing order: ace (4 points), king (3 points), queen (2 points), and jack (1 point) are the honors cards.

The game consists in two activities, the auction or the bid and the play of the cards.

2.1 The Auction

This phase is a conversation between two cooperating team members against an opposing partnership. It aims to decide who will be the declarer. Each partnership uses an established bidding system to exchange information and interpret the partner's bidding sequence. Each player has knowledge of his own hand and any previous bids only.

The dealer begins the auction, and the turn to speak passes clockwise. At each turn a player may either pass or make a bid that must be higher than the previous bid if any; it specifies a number of tricks (1 to 7) and a trump (or no-trump). The possible trump suits rank as follows: no-trump (highest), Spade ♠, Heart ♥, Diamond ♦, and Club ♣ (lowest). If anyone bids, the auction continues until there are three consecutive passes, and then stops. The last bid becomes then the contract and the opposing team lays down an initial card and the play phase of the game begins.
The team who made the final bid will now try to make the contract. The first player of this team who mentioned the denomination (suit or no-trump) of the contract becomes the declarer. The declarer's partner is known as the dummy. Note that there are different kinds of auction rules in the bridge game. These rules may be respected or violated in a play [1].

![Game disposition](Image)

**Fig 1 Game disposition**

### 2.2 The play

The player to the left of the declarer leads to the first trick and may play any card. Immediately after this opening lead, the dummy's cards are exposed. Play proceeds clockwise. Each of the other three players in turn must, if possible, play a card of the same suit that the leader played. A player with no card of the suit led may play any card. A trick consists of four cards, one from each player, and is won by the highest trump in it, or, if no trumps were played, by the highest card of the suit led. The winner of a trick leads to the next, and may lead any card. Dummy takes no active part in the play of the hand and is not permitted to offer any advice or comment on the play. Whenever it is dummy's turn to play, the declarer must say which of dummy's cards is to be played, and dummy plays the card as instructed. Finally, the scoring depends on the number of tricks taken by the declarer team and the contract.

### 3. Related works

Since our work consists in using abduction and TMS in Bridge game, we will describe the most significant works in each of three AI research areas.

#### 3.1 Abduction

Abduction is a useful constructive technique for generating "explanations" or "plans" for given "observations" or "goals". It can be defined as follows: Given a domain theory (axioms T) and observations (set of atoms O) find a minimal set of observations (assumptions A) where A ∪ T is consistent. [2] show that abduction is suitable for automating tasks such as diagnosis, planning, theory and database updates. They provide a survey on research results that show its applicability and utility in software engineering, as a technique for supporting knowledge-based software development, and for facilitating analysis and revision of specifications.

Kakas et al. combine abductive reasoning and constraint solving by integrating the Abductive Logic Programming (ALP) with that of Constraint Logic Programming in a system; it provides a bridge between the high-level problem domain properties and domain-independent solving methods [3].

Also, [4] propose a qualitative study on VoIP products by summarizing the factors that affect the customer behaviors of Skype based on user satisfaction that can be quantified among other by abduction [5].

In [6], Mooney combines abduction and induction to develop machine-learning systems: Abductive reasoning has been used in revising existing KB to improve their accuracy; and Inductive learning is used to acquire accurate abductive theories. Then, he demonstrates their ability in help constructing AI systems for problems in medicine, molecular biology, and intelligent tutoring.

Arrighi and Ferrario inquire the role of abductive reasoning in the interpretation of natural language and use it to represent the mutual ground used by the speakers in a conversation [7].

#### 3.2 TMS

This section summarizes some of the various works that utilize TMS in several application areas with different objectives.

Zlatareva applies TMS to design and test expert systems [8].

Levesque shows that ATMS is a general abduction algorithm for propositional Horn-clause that exhaustively computes all possible explanations which is computationally very expensive for large problems [9]. That is why an amelioration of this algorithm is proposed in [10] to avoid ATMS redundant work by hiding intermediate results. The implementation of this algorithm has been tested on some abduction problems such as text understanding, plan recognition, and devices diagnosis.

In the operating system domain, an intelligent help system for UNIX users was elaborated [11]. It maintains a model of the user based on assumptions about what the user is thinking during a session. An ATMS is used to manage the consistency of these assumptions. In the area of computer networks, Tang et al. propose a system for alert analysis [12] that uses a JTMS to provide administrators a tool to manage/maintain alerts resulting from the intrusion specifications.

In [13], a model for intelligent responsive environments based on semantic web services is developed. ATMS is used to manage the provision of services, based on the relations between preconditions and effects for each service. Thus, the ATMS relies on the description of the available services in order to identify minimal sets of preconditions allowing to obtain a given effect.

Recently, many papers about integrating TMS in other domains have been published. Maeda elaborates a fuzzy ATMS in a consistency maintenance system for spatial navigation [14]; Lorenzi et al. use distributed TMS to propose a multi-agent recommender approach in the...
tourism area [15]. Furthermore, Kono et al. develop a practical multimodal interface system where the formalization of input integration/interpretation is based on ATMS [16].

Finally, we believe that TMS can be used in many diagnostic applications and in those where incomplete data is handled. Our system that is designed to assist a Bridge player to find hidden honor cards falls within this framework.

3.3 Bridge

Bidding phase in multiplayer games, such as Bridge, is a challenge for game theory; that is why many approaches have been proposed. In the following, we will describe the most significant work which uses Artificial Neural Networks (ANN) to have better cooperation between agents in the bidding phase.

Ando et al. consider the auction as an interaction among agents with an hypothetical reasoning mechanism [17]. Authors also discuss the effect of ambiguity of information on the behavior of agents and propose a strategy to prevent it.

Uehara proposes a bidding system where information on the combined characteristics of the players’ hands is included as common-sense knowledge. Each agent aims to maximize point gain by cooperating with the partner and minimize point loss by competing with the opponents [18].

Also, bidding is studied in [19] where authors use a self-organizing map (SOM) that is a special form of ANN, to effectively bid no-trump hands. They use a learning framework for agents who exchange continuously their refined strategies [20] and show that a combination of two SOM, that only use assertion, is well adapted to find an optimal strategy for no-trump hands.

Mandziuk and Mossakowski use ANN to estimate the number of tricks in modeling a DDBP (Double Dummy Bridge Problem). They propose to add human knowledge to networks’ inputs in order to improve their results [21].

4. ATMS basics

4.1 TMS

In order to solve some diagnosis problems, a problem solving system must be able to identify responsibilities of conclusions, by keeping track of the way we reach the conclusions from the assumptions. This allows calling into question some information if a problem occurs. A TMS uses a knowledge representation method for representing both beliefs and their dependencies [22]. The name truth maintenance is due to the ability of these systems to restore consistency. TMS is used within problem solving systems in conjunction with Inference Engines (IE) such as rule-based inference systems, to manage the IE beliefs as a Dependency Network.

The IE passes assumptions and justifications to the TMS, which can return beliefs (based on dependencies) or contradictions [23]. Every inference made is communicated to the TMS that uses justifications stored so far in order to decide what data are confirmed or negated. The addition of a justification may cause any belief to change [24]. Thus, the TMS saves traces when data change.

There are several forms of TMS; all answer the same queries but differ in the structure of the dependency record [25]. Some systems work with a single context like JTMS (justification-based) or LTMS (logical-based) while others, like ATMS, deal with multiple contexts.

Choosing the most appropriate TMS should be guided by its simplicity and its suitability for the task to be accomplished. In our survey, we use abductive reasoning to make assumptions about the distribution of honor cards and study the consistency of the multiple contexts associated to these assumptions. Thus, we naturally chose to use ATMS that seems most appropriate to our approach.

4.2 ATMS

Like all other TMS, ATMS aims to determine what data is to be believe in and what is to negate. The problem solver builds records of all inferences and hypothesis to introduce (assumptions). The basic data structure manipulated by an ATMS is a node. Each node is represented as a triplet <datum, justification, label> where the justification describe how it derives from other data, and is labeled with the set of assumptions, called environment, under which it holds. While a justification describes how the datum is derived from immediately preceding antecedents, a label describes how the datum ultimately depends on assumptions [24].

A label must be:
- Consistent: no environment in the label supports the derivation of false,
- Sound: for each environment in the label, the node must be included in the context defined by that environment,
- Complete: any environment in the lattice whose context includes the node is a superset of some environment in the label; inconsistent environment has empty context,
- Minimal: no environment in the label is a subset of another.

There are four types of nodes in an ATMS: facts, assumptions, assumed nodes, and No goods.

- A fact is information that is true and has a justification with no antecedents. The node <f, {}, {}> represents the fact f.
- An assumption is not proved yet but the system decides to believe in. It is a node whose label contains a singleton environment mentioning itself. The node, <A, {}, {A}> represents the assumption A.
- An assumed node is neither a fact nor an assumption and has a justification mentioning an assumption. Given the inference A → a, the node <a, {A}, {}>
The justifications must be Horn clauses. Thus, we decided composed of an honor card and a player that may hold it. a datum, its justification and its label. A datum is manipulated by the ATMS.

An ATMS maintains multiple sets of beliefs simultaneously, thus allowing inferences in multiple contexts at the same time. The tasks of the ATMS are:

- to efficiently generate all possible contexts based-on a set of inferences,
- to manage Nogoods that result from new assumptions that cause contradiction, then to remove every environment that contains a Nogood from every node label.

ATMS algorithm is incremental; it takes a set of correct label, and computes the changes caused by the contribution of supplementary justifications. Justifications serve to propagate updates on labels. The algorithm creates correct local labels and propagates changes to all other labels until they become globally correct.

This paper is not intended to provide a complete survey on ATMS; rather it aims to present the basic ideas and functionality of TMS to be used as a help tool. For more detail, refer to [24].

5. Proposed Approach

5.1 ATMS Description

The ATMS is the core of our application, since it serves to locate the hidden cards by maintaining a set of consistent data. In order to make it reusable in other applications, we developed the ATMS as an independent module; the only constraint to use it is to respect its input data format.

In the following, we present basic data manipulated by the ATMS.

5.1.1 Nodes

The ATMS manipulates nodes, each representing a datum, its justification and its label. A datum is composed of an honor card and a player that may hold it. The justifications must be Horn clauses. Thus, we decided to transform rules that are not Horn clauses into several rules each containing negation of the consequences.

Ex: $\{a_1\} >$ corresponds to the assumed datum a. Note that the label of an assumed node may be built using several inferences transitivity.

- Finally, there is a node that represents falsity ($\bot$). The inconsistent environments are called Nogoods because they represent inconsistent conjunctions of assumptions.

$P_1 \land P_2 \land \ldots \land P_n \land \neg C_2 \land \ldots \land \neg C_m \Rightarrow C_1$

Note that we manipulate negative data in the rules. This kind of data is very useful because the system main task consists in managing Nogood environments that result from new facts introduction related to the play of an honor card.

5.1.2 Facts

A fact is a true datum that results from the play of an honor card by a player whose cards are hidden. It's represented by a pair <Player, Card>.

5.1.4 Assumptions

There are 2 types of assumptions in our system:

- Bidding Assumptions: Those concerning the longest trump and the number of points in a player's hand. We represent this kind of assumption as a triplet <Player, Number of cards of the longest trump, Number of points>. This type of assumptions results from the auction phase and is deduced according to the Bidding table (see Table 1).
- Card Assumptions: Those concerning the distribution of honor cards. We represent them as a pair <Player, C> where C may be:
  
  - a list of honor cards held by the player corresponding to a possible combination, or
  - a single card from the preceding list. We take into account this kind of assumption in order to reflect the ATMS behavior that locates cards one by one.

5.1.3 Rules

The player who uses our system is called Self and the two players with hidden cards are Player_1 and Player_2. The Auction phase allows creating several assumptions concerning the number of points and the longest trump of Player_1 and Player_2. Since the total number of points is 40, the following Point Rule (PR) is added to represent compatible assumptions:

$<\text{Player}_1, \text{__NbPoints}>\Rightarrow <\text{Player}_2, 40\text{-Self.points-Dummy.points-NbPoints}>$ \hspace{1cm} (AR)$

A meta-rule indicates that a card cannot be held by two players: Given 2 Card Assumptions $<P_1, L_1>$ and $<P_2, L_2>$, $\{P_1, L_1\}, \{P_2, L_2\}$ is Nogood $<P_1, L_1>, <P_2, L_2> \Rightarrow$ if there is a card C that belongs to L1 and to L2.

$\forall L_1, \forall L_2, \forall P_1, \forall P_2 \exists C / C \in L_1 \land C \in L_2 \land P_1 \neq P_2 \Rightarrow <P_1, L_1>, <P_2, L_2> \Rightarrow \hspace{1cm} (MR_d)$

This rule is essential because it generates Nogoods used to eliminate inconsistent environments. For more clarity, we decided to formulate this rule (and some others) in predicate logic whereas the ATMS manipulates data nodes as propositions.
Another meta-rule concerning Nogoods shows that the possible combinations of a player must contain all the facts concerning him:

\[ \exists P \forall C / <P,C> \text{ is a fact } \land C \not\in L \rightarrow \neg <P,L> \quad (MR) \]

This means that if \( C \) is a fact then its negation is Nogood and hence all environments subsumed by its negation must be eliminated.

Finally, the table of nodes is initialized using inference Combination Rules (CR). A combination rule associates a set of Card Assumptions to each bidding Assumption.

Ex: the following CR indicates that if East has 2 points and does not have ♥Q nor ♦Q, then he must have ♠Q:

\[ <E,_,2> \land \neg <E,Q,\heartsuit> \land \neg <E,Q,\diamondsuit> \rightarrow <E,Q,\spadesuit> \quad (CR) \]

### 5.2 Card Locator Architecture

We designed our application as two independent modules communicating through dataflow. This approach has the advantage of being flexible and portable. These modules are:

- Assumption Generator - this component is responsible for the creation of assumptions that represent possible distributions of honor cards;
- ATMS - the main task of this component is to manage Nogoods in order to eliminate inconsistent assumptions.

5.3 Application

The game takes place in three phases:

#### 5.3.1 Card distribution

The 52 cards are distributed randomly.

#### 5.3.2 Auction phase

In our application, we adapt standard game rules by adding some constraints in order to use ATMS efficiently. Indeed, a player can speculate on his cards (number of honor cards, the number of cards in a trump ...) in order to mislead opponents. It is clear that such behavior is not compatible with a computer-assisted game. The auction component in our system is actually based on the Bidding table adapted to be used by the ATMS.

<table>
<thead>
<tr>
<th>Number of honor cards in a suit</th>
<th>Number of points</th>
<th>Bid</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;5</td>
<td>12 ≤ nb ≤ 15</td>
<td>1 + trump</td>
</tr>
<tr>
<td></td>
<td>16 ≤ nb ≤ 18</td>
<td>1 + no-trump</td>
</tr>
<tr>
<td></td>
<td>19 ≤ nb ≤ 23</td>
<td>2 + no-trump</td>
</tr>
<tr>
<td></td>
<td>nb &gt; 23</td>
<td>3 + no-trump</td>
</tr>
<tr>
<td>≥5</td>
<td>12 ≤ nb ≤ 15</td>
<td>1 + trump</td>
</tr>
<tr>
<td></td>
<td>16 ≤ nb ≤ 18</td>
<td>1 + no-trump</td>
</tr>
<tr>
<td></td>
<td>19 ≤ nb ≤ 22</td>
<td>2 + trump</td>
</tr>
<tr>
<td></td>
<td>23 ≤ nb ≤ 25</td>
<td>2 + no-trump</td>
</tr>
<tr>
<td></td>
<td>nb &gt; 25</td>
<td>3 + no-trump</td>
</tr>
<tr>
<td>&gt;5</td>
<td>11 ≤ nb ≤ 15</td>
<td>1 + trump</td>
</tr>
<tr>
<td></td>
<td>16 ≤ nb ≤ 20</td>
<td>2 + trump</td>
</tr>
<tr>
<td></td>
<td>nb &gt; 20</td>
<td>3 + trump</td>
</tr>
</tbody>
</table>

Other configurations Pass

Another restriction takes place in the auction phase. Indeed, we divide it into two steps:

- The first time a player bids, he must do it in accordance to the bidding table. The purpose of this phase is to inform the partner (and thus the system) about the number of points he held,
- The second step gives the player the freedom to choose the trump and the bid that best suit his cards.
Consider a player with a hand containing 12 points with 0 points of ♠ and 8 points of ♥. Accordingly to the bidding table, he must bid 1♥ in the first step. In the second step (second round of auction), he may change trump and choose to bid 1♥ (or more).

Since our system requires the number of points in the player hand, we are interested only in the first phase that allows us to make assumptions on the point distribution.

First, the partner of a player who already bids can answer in order to inform his partner (and thus the system) about the number of points he holds. He then estimates the number of points of his partner by consulting the Bidding table in order to retrieve the minimum number of points corresponding to the partner's bid. As an example, a player whose partner has hidden 2♥ + no-trump will suppose that the partner has only 19 points (case of number of honor cards < 5) instead of 22 points (number of honor cards = 5). Since the first phase was designed purely to inform the partner of the number of points, this choice gives him a good margin to rectify his auction in the second phase.

The partner of a player who bids in the first step acts as follows:
- Estimates the partner number of points,
- Add this estimation to his own number of points,
- Consider this sum in order to bid according to the Bidding table.

5.3.3 Play phase

The ATMS starts working in parallel with this phase. After the auction, we may be in front of multiple assumptions, some of which may be contradictory. Honor cards are located successively using abductive validation that identifies and eliminates inconsistency. The following algorithm describes a play session.

Algorithm of card localization

```c
{  Hidden_Cards = Total_Cards - Self.Cards - Dummy.Cards;
  Use Bidding_Table, Auctions, and PR to estimate possible values of Player1.points, Player1.longest_trump,
  Player2.points, Player2.longest_trump;
  Use Player1.points, Player2.points, and Hidden_Cards to determine Possible_Combinations;
  Use Possible_Combinations to define Assumptions;
  Initialize Table_of_Nodes;
  While (3 a non-located honor card)
    {  For each Trick
        If (Player1 or Player2 plays an honor card)
          {  Add it as a fact;
              Use MR1 to retract inconsistent nodes;
              Use MR2 to retract inconsistent nodes;
              Each assumption on the location of a single card having only one label becomes a fact (located card);
              Inform Self of the located honor card;
          }
    }
}
```

6. ILLUSTRATION

The following example illustrates a play session (see Fig.3). Cards are distributed by North:

![Card Distribution](image)

**Fig 3 A play session**

**First step of auction**
- North has 13 points and then bids 1♥ according to the Bidding Table (see table1)
- West bids 2♥
- South has 6 points and then bids 2 No Trump
- East passes

**Second step of auction**
- North may change trump or pass,
- West, South and East pass.

The team South-North wins the auction phase and has to show his cards. The hidden cards are then ♥Q, ♠Q, ♥AKQJ, ♠AK.

The player South has the honor cards locator that makes the following assumptions about the number of points held by East and West according to their auctions:
- West who has hidden 2♥ must have:
  - 5 honor cards in a suit and a number of points between 19 and 22, or
  - More than 5 honor cards in a suit and a number of points between 16 and 20.
- Then West has between 16 and 22 points.

Since North has 13 points and South 6 points, and since the total number of points is 40, West has between 16 and 21 points and East between 0 and 5 points (rules PR).

We have then 12 combinations (0, 1, 2, 3, 4, and 5 for East, and 16, 17, 18, 19, 20, and 21 for West) giving place to 12 assumptions among which only two are correct (2 for East and 19 for West for example).

The ATMS uses the combination rule (CR1) in order to associate combinations of hidden cards to each assumption and then creates a corresponding node (for example, the following combinations correspond to 3 points: {♥Q, ♥J}, {♠Q, ♠J}, {♠KJ}, ♠K}).

The Nogood node is updated using meta-rules MR1 (a card cannot be held by two players) and MR2 (all environments subsumed by a negation of a fact must be eliminated).
The table of nodes in fig.4 summarizes the ATMS behavior while locating honor cards:

**Fig 4 Table of nodes**

<table>
<thead>
<tr>
<th>Node</th>
<th>Data</th>
<th>Justifications</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&lt;E,[*J]&gt;</td>
<td>{J1=(&lt;E,_,1&gt;)}</td>
<td>{{J1}}</td>
</tr>
<tr>
<td>2</td>
<td>&lt;E,[*Q]&gt;</td>
<td>{J2=(&lt;E,_,2&gt;,&lt;E,[AQ]&gt;},&lt;E,[*Q]&gt;}</td>
<td>{{J2}}</td>
</tr>
<tr>
<td>3</td>
<td>&lt;E,[*Q]&gt;</td>
<td>{J3=(&lt;E,_,2&gt;,&lt;E,[*Q]&gt;},&lt;E,[*Q]&gt;}</td>
<td>{{J3}}</td>
</tr>
<tr>
<td>4</td>
<td>&lt;E,[*Q]&gt;</td>
<td>{J4=(&lt;E,_,2&gt;,&lt;E,[AQ]&gt;},&lt;E,[*Q]&gt;}</td>
<td>{{J4}}</td>
</tr>
<tr>
<td>7</td>
<td>&lt;E,[*Q]&gt;</td>
<td>{J7=(&lt;E,_,3&gt;,&lt;E,[*Q]&gt;},&lt;E,[*Q]&gt;}</td>
<td>{{J7}}</td>
</tr>
<tr>
<td>8</td>
<td>&lt;E,[*Q]&gt;</td>
<td>{J8=(&lt;E,_,3&gt;,&lt;E,[*Q]&gt;},&lt;E,[*Q]&gt;}</td>
<td>{{J8}}</td>
</tr>
</tbody>
</table>
30 \(<E,\clubsuit K>\)
\{J_{38}=(<E,[\heartsuit K]>), J_{39}=(<E,[\heartsuit \spadesuit K]>), J_{39K}=(<E,[\heartsuit \spadesuit K]>), J_{3x}=(<E,[\heartsuit \spadesuit K]>), J_{3xK}=(<E,[\heartsuit \spadesuit K]>), \}
\begin{align*}
\text{Label} & = L_{37} \\
\text{Justifications} & = \text{MR}_2, L_{31}, L_{32}, L_{33}, L_{34}
\end{align*}

31 \(<E, \spadesuit K>\)
\{J_{31}=(<E,[\spadesuit K]>), J_{312}=(<E,[\spadesuit \diamondsuit K]>), J_{313}=(<E,[\spadesuit \diamondsuit K]>), J_{314}=(<E,[\spadesuit \spadesuit K]>), J_{32}=(<E,[\spadesuit \spadesuit K]>), J_{33}=(<E,[\spadesuit \spadesuit K]>), J_{34}=(<E,[\spadesuit \spadesuit K]>), J_{3x}=(<E,[\spadesuit \spadesuit K]>), J_{3xK}=(<E,[\spadesuit \spadesuit K]>), \}
\begin{align*}
\text{Label} & = L_{37} \\
\text{Justifications} & = \text{MR}_2, L_{31}, L_{32}, L_{33}, L_{34}
\end{align*}

32 \(<E, \heartsuit Q>\)
\{J_{32}=(<E,[\heartsuit \heartsuit Q]>), J_{323}=(<E,[\heartsuit \heartsuit Q]>), J_{324}=(<E,[\heartsuit \heartsuit Q]>), J_{325}=(<E,[\heartsuit \heartsuit Q]>), J_{326}=(<E,[\heartsuit \heartsuit Q]>), J_{327}=(<E,[\heartsuit \heartsuit Q]>), J_{328}=(<E,[\heartsuit \heartsuit Q]>), J_{329}=(<E,[\heartsuit \heartsuit Q]>), J_{33}=(<E,[\heartsuit \heartsuit Q]>), J_{334}=(<E,[\heartsuit \heartsuit Q]>), \}
\begin{align*}
\text{Label} & = L_{32} \\
\text{Justifications} & = \text{MR}_2, L_{31}, L_{32}, L_{33}, L_{34}
\end{align*}

33 \(<E, \heartsuit Q>\)
\{J_{33}=(<E,[\heartsuit \heartsuit Q]>), J_{333}=(<E,[\heartsuit \heartsuit Q]>), J_{334}=(<E,[\heartsuit \heartsuit Q]>), J_{335}=(<E,[\heartsuit \heartsuit Q]>), J_{336}=(<E,[\heartsuit \heartsuit Q]>), J_{337}=(<E,[\heartsuit \heartsuit Q]>), J_{338}=(<E,[\heartsuit \heartsuit Q]>), J_{339}=(<E,[\heartsuit \heartsuit Q]>), J_{34}=(<E,[\heartsuit \heartsuit Q]>), J_{344}=(<E,[\heartsuit \heartsuit Q]>), \}
\begin{align*}
\text{Label} & = L_{32} \\
\text{Justifications} & = \text{MR}_2, L_{31}, L_{32}, L_{33}, L_{34}
\end{align*}

34 \(<E, \spadesuit Q>\)
\{J_{34}=(<E,[\spadesuit \spadesuit Q]>), J_{343}=(<E,[\spadesuit \spadesuit Q]>), J_{344}=(<E,[\spadesuit \spadesuit Q]>), J_{345}=(<E,[\spadesuit \spadesuit Q]>), J_{346}=(<E,[\spadesuit \spadesuit Q]>), J_{347}=(<E,[\spadesuit \spadesuit Q]>), J_{348}=(<E,[\spadesuit \spadesuit Q]>), J_{349}=(<E,[\spadesuit \spadesuit Q]>), J_{35}=(<E,[\spadesuit \spadesuit Q]>), J_{355}=(<E,[\spadesuit \spadesuit Q]>), \}
\begin{align*}
\text{Label} & = L_{32} \\
\text{Justifications} & = \text{MR}_2, L_{31}, L_{32}, L_{33}, L_{34}
\end{align*}

35 \(<E, \spadesuit J>\)
\{J_{35}=(<E,[\spadesuit J]>), J_{353}=(<E,[\spadesuit J]>), J_{354}=(<E,[\spadesuit J]>), J_{355}=(<E,[\spadesuit \spadesuit J]>), J_{356}=(<E,[\spadesuit \spadesuit J]>), J_{357}=(<E,[\spadesuit \spadesuit J]>), J_{358}=(<E,[\spadesuit \spadesuit J]>), J_{359}=(<E,[\spadesuit \spadesuit J]>), J_{36}=(<E,[\spadesuit \spadesuit J]>), J_{366}=(<E,[\spadesuit \spadesuit J]>), \}
\begin{align*}
\text{Label} & = L_{32} \\
\text{Justifications} & = \text{MR}_2, L_{31}, L_{32}, L_{33}, L_{34}
\end{align*}

36 \(\perp\)
\{J_{36}=(<E,J>)\}
\begin{align*}
\text{Label} & = L_{36} \\
\text{Justifications} & = \text{MR}_2, L_{31}, L_{32}, L_{33}, L_{34}
\end{align*}

Node 1 (respectively 2 to 4, 5 to 9, 10 to 16 and 17 to 27), corresponds to the combinations where East has 1 (respectively 2, 3, 4 and 5) point(s).

Nodes 28 to 35 denote the following property: each datum is justified by any environment that contains it.

Another set of nodes is associated to the combinations of these cards are played.

During the play, West plays \(\heartsuit J\) (see Fig. 5). The fact \(<W, \heartsuit J>\) is added. Then the system eliminates:
- all the combinations having East as player and containing \(\heartsuit J\) (MR1) by adding the node 36. Thus labels 1, 7, 8, 9, 13, 14, 17, 18, 25, 26, 27, and 35 become inconsistent,
- all those having West as player and not containing \(\heartsuit J\) (MR2).

At now, East plays \(\heartsuit Q\). The fact \(<E, \heartsuit Q>\) is added and the system eliminates:
- all the combinations having West as player and containing \(\heartsuit Q\) (MR1) by modifying the node 36 := West has \(\heartsuit AK\).
- all those having East as player and not containing \(\heartsuit Q\) (MR2). Thus labels 2, 11, 21, 24, 30, 31, and 34 become inconsistent.

The remaining consistent environments are 12, 32, and 33. The first one concerns a combination while the last two concern the location of single cards. Note that there is no more combination for East and thus all his cards are located. Using this information, the ATMS locates West cards.

This example illustrates the power of using ATMS as a bridge player assistant tool. Indeed, it allows locating all the hidden honor cards (8) although only 3 of these cards are played.

7. CONCLUSION

The ATMS is a reasoning maintenance system proved useful in many applications and fields such as diagnosis and reasoning about incomplete data. This paper describes how to use ATMS as a card game assistant tool in order to locate all the hidden honor cards using a minimum number of played ones.

In our approach, we adapt the standard bridge rules by building our own Bidding table. This can be considered as a limitation that enforces the player who uses the assistant tool to respect the auction rules defined in this table. In order to reflect real play, the developed interface offers the user the possibility to parameterize this table by allowing him to set his own preferences at the game beginning so that it best suits his playing style. This facility can be generalized to other players (those who are not using the tool) allowing to create a profile for each of them.

Our software works even if both players of a team do not bid during a session. They will keep their cards hidden without giving the system any useful data. We believe that the information provided by the bidding team is enough to make assumptions. All the possibilities of points from 0 to (40 - bidding team total points) are associated to each of the non-bidding team player and the ATMS can then work normally. This presents the only drawback to increase substantially the number of
assumptions. However, this number decreases rapidly each time an honor card is played by this team.

One limitation of our application is that we take into account only the first phase of auction to get information about hidden card distribution. Indeed, the use of only two parameters (number of points, longest trump length) leads us to omit the next auction phase. We believe that considering other factors such as player profile requires taking into account the next phases.

Another limitation of the ATMS is that it handles propositional justifications only. This could be considered as a problem in applications handling a large number of data [12]. In our case, since we are interested only in a subset of the 16 honor cards spread over two players, the number of assumptions becomes quite small making the number of combinations quite reasonable. The formalism of propositional logic seems appropriate to our problem.

The obtained results seem encouraging and the tool often retrieves the location of all hidden honor cards after four tricks approximately.

Finally, we estimate that the approach that consists in using ATMS and abduction reasoning is well-adapted to games as Rubber Bridge. However, building a robust tool that is closer to the reality requires taking into consideration more factors in the assumption creation process. Those factors may be the player experience and his playing style.

REFERENCES


**APPENDIX**

After 3 tricks, there are no more hidden honor cards

*Fig 5 Example of play session*