Intelligent Information Systems X
Communications of the Symposium held in
Zakopane, Poland, June 18-22, 2001

Changing and Using Beliefs under Incomplete Information

Wojciech Jamroga

Parlevink Group, University of Twente, Enschede
Institute of Mathematics, University of Gdańsk
e-mail: jamroga@cs.utwente.nl
www: http://www.cs.utwente.nl/∼jamroga

Abstract. The concept being proposed here is to use prototype semantics to represent an agent's belief state. Prototype semantics is a linguistic theory that emerged in the 1980s – the key idea is to describe the meaning of an utterance, or a notion, by defining the prototype (the most typical example to which the notion refers) and the extension rules describing 'family resemblances' between various entities and, in consequence, allowing to derive less typical instances from more typical ones. The intuition behind this paper is that in situations of incomplete information such a representation of the agent's knowledge may be easier to deal with than a straightforward probabilistic representation – especially when belief changes are not necessarily monotonic, and informations being acquired by the agent may be vague themselves. Moreover, when an exhaustive search through all the possibilities is impossible the agent may benefit from analyzing the typical situations instead of random ones. This property was tested for a family of games with incomplete information on finite binary trees.

Keywords: belief change, prototype semantics, games with incomplete information, language communication, knowledge representation, nonmonotonic reasoning, multiagent systems.

1 When an Agent Needs to Believe...

An agent must know something (or at least to believe in something) to proceed any sensible action. This knowledge is used in strategy planning, decision-making and so on. Every observed action (a move in the case of a game) brings some new information: either explicitly (when there is explicit communication – as in Bridge bidding, for instance) or implicitly (the action may imply something about the possible strategy or the actual situation of the acting agent). The new information updates the actual knowledge of the observing agent. Sometimes a revision is necessary, when some of the previous beliefs appear to be true no more. Thus, the agent's reasoning may be nonmonotonic.

In a game with incomplete information – from a player's point of view – there are often a number of possible variants of the actual situation (sometimes called the set of the *possible worlds* – Ω). If a probability function is given as the playing agent's actual belief, standard decision theory techniques may be used – for instance, expected utility maximization to choose the best of the alternative game strategies. But anyway, the question is: how to change the actual belief function when a new information appears? Another question is how to represent the beliefs (fuzzy measure functions) to make the updates smart, easy and not very complex.

The *Dempster's rule of combination* may be used to model the change of a belief with another belief However, these two beliefs have to be really independent. Also, they must be of the same importance for

us – none of them should be considered supreme, neither chronologically (as a newer information) nor inherently (better, more relevant information source). For instance, opinions prepared by two separate experts at the same time on a given subject seem to fit the requirements. The experts must be of the same rank and they shouldn't communicate with each other.

2 Belief Change

Imagine two agents – A and B – involved in organization of a conference. The agents meet and A reports:

A: The boss told me that the number of participants would be between 100 and 2000.
B: As usually. He'd better be more precise.
A: Well, I've seen the registration list for a moment. An awful lot of people is going to arrive!
B: About 1000?
A: Even a bit more.

To make any decision (for instance on how many proceedings should be printed), B has to extract some information about the actual possibilities from A's rather obscure statements. However, the pieces of information delivered by A are by no means independent. In fact, every new statement refines the picture shaped by previous messages or even refers directly to the state of belief expressed by B (*About 1000?*). B, provided with a new piece of information, must revise his beliefs (he apparently considered 1000 participants as the most probable number, but the last A's answer made him think of a higher amount... maybe some 1200?). Note that – though the shape of the belief function may change drastically – the interval of possible cases should evolve in a monotonic way (from $[100, 2000]$ to $[1000, 2000]$ in the example above). In other words, the set can be refined, but not revised – if only we assume that A and the boss are not wrong.

The same holds for games with incomplete information. The Dempster's rule seems inappropriate in a number of ways:

- the playing agent is not given several independent belief functions to combine them into one. Instead, the agent has the initial belief (say, a fuzzy measure) which is then updated by the next moves of the other players;
- the implications of the actual move are often vague and difficult to represent in a numerical form;
- when there's an explicit communication in the game (as it is in Bridge), the communication moves (bids, for instance) often refer to the agent's actual knowledge (which is partly shared by the communication partners). Thus, such a move is in fact to *update* the interlocutors' beliefs, not to *create* a completely new view!

The idea presented in this paper emerged from an observation of the way humans communicate when bidding in a Bridge game. The implications of every bid can be modeled on three levels:

1. the prototype (for instance, opening **1♠** suggests 13-14 PC with exactly five Spades and no suit shorter than two cards),
2. the typical (plausible) cases set, defined by plain logical constraints (for opening **1♠**: 11-22 PC, five or more Spades),
3. non-typical, but acceptable cases (opening **1♠**: the declarer may have a little less points, but with a stronger Spade suit).

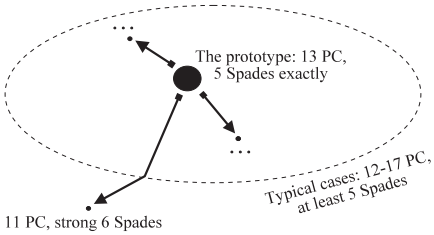
Bidding languages are usually defined in terms of the second level. However, it is the prototype level that's often used by human players to reason about the next move. Why? It reduces the complexity of the problem.

Reasoning on the prototype level recalls the *default logic* in a manner. A player accepts *by default* that the actual card set of the declarer fulfills the prototype conditions. Say, our partner signaled five or more Diamonds and Heart suit shorter than Diamond one. By default we usually assume that number of Diamonds he possesses is exactly five – thus the partner has no more than four Hearts. In the next bid he signals five Hearts... The default judgment is now inconsistent so it has to be changed to the nearest accessible value – in this case, six Diamonds and five-card long Heart suit.

2.1 Belief Change via Prototypes and Extension Rules

(Kronenfeld et al 1985) and (Lakoff 1982) describe the idea of *prototype semantics* (Wittgenstein is often credited as the source of the idea). A notion (or the meaning of some utterance) is defined by the *prototype* (the most typical object for a given word, clause or notion) and *extension rules* which enable deriving less typical representatives from more typical ones through a 'family resemblance'. Connecting some 'measure of typicality' with the extension rules generates fuzzy classification; traditional (sharp) categories can be obtained through fixing threshold values for this measure.

Fig. 1. The prototype and extensions – **1♠** opening in Bridge



The very idea is to represent belief functions with prototypes, extension rules, and mappings providing some numerical measure of typicality. A belief modification would be then proceeded through:

- **change of the prototype.** Usually a new information affects only the case that we consider the most probable, not the way we shift from one case to another;
- **change of the extension rules.** Often, however, the obtained information changes the actual belief more radically, affecting also the 'family resemblance' between the possible situations (possible worlds). In other words, the interrelations between likelihood of the situations change;
- sometimes it's convenient to determine the plausible cases set explicitly. The set boundaries can be updated then, too.

2.2 A View to a Bidding – an Example

Suppose that a Bridge playing agent has two alternative strategies: s_1 and s_2. To reduce the complexity of the search, the agent takes into account only one aspect of the card distribution – say the joint power of his and his partner's hand, expressed with Point Count (PC), the most popular evaluation factor of card power in Bridge. The agent's general knowledge includes the conditional expectancy values $E(s|PC = x)$ for both strategies. Assume (for the sake of simplicity) that the agent has to determine his strategy after the second bid of his partner, and that the expectancies don't change throughout the bidding. Since the player knows exactly how much PC he possesses in his own hand, the only thing to believe he's interested in is the number of points his partner has.

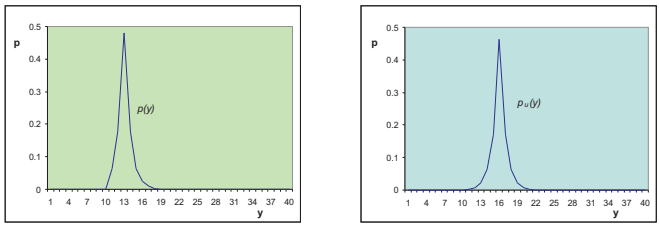
In the actual game the agent has 9PC in his hand, and his partner opened **1♠**, so his knowledge about his partner's possible power (y) may be as follows. The extension rule defines the distance between instances explicitly:

the prototype (*prot*):	13 PC
the plausible cases set (*Pl*):	11..22 PC
the extension rule:	$distance(y_1, y_2) = \begin{cases} \lvert y_2 - y_1 \rvert & \text{if} \quad prot \geq y_1 \geq y_2 \text{ or } prot \leq y_1 \leq y_2 \\ undef & \text{otherwise} \end{cases}$
the probability measure:	$p(y) = k * \hat{p}(y)$, where $\hat{p}(y) = \begin{cases} e^{-distance(prot,y)} & \text{if} \quad y \in Pl \\ e^{-distance^2(prot,y)} & \text{if} \quad y \notin Pl \end{cases}$ and $k = 1/\sum_y \hat{p}(y)$ is the normalization factor.

Now, the agent bids **1NT** and his partner replies **3♠** (which means 'some extra values' in most bidding systems). The agent's belief is updated now:

the prototype:	16PC
the rest:	as before

Fig. 2. The belief function $p(y)$ and the updated measure $p_u(y)$



If the agent has to make his decision right now, he can use the updated probability measure to compute the expected values of his payoff for any available strategy.

The example presented above is somewhat arbitrary, of course. It's a long way to identifying a methodology for defining prototypes and extension rules that suit a given situation in a good manner. But *if* the agent is able to express his knowledge in terms of prototype semantics, he can benefit from it in a number of ways. One of them is making the knowledge updates easier to deal with. Another one relates to using the knowledge with restricted resources; this issue is discussed within the next section.

3 Using Prototypes to Find a Decision

In the previous section the concepts of prototype semantics were proposed to be useful for an agent who has to *learn* some uncertain knowledge. However, the way human players deal with uncertain situations in games like Bridge suggests that also *using* prototypes may be beneficial for the agent. When the space of all possibilities is huge, a complete search through all the alternative strategies would cost too much time of computation. Thus, in order to make a decision, agents often use suboptimal algorithms – such as classical minimaxing, vector minimaxing or payoff-reduction minimaxing instead of a thorough search through all the possible strategies. To reduce the complexity further still, players can do *sampling* – choose an arbitrary subset of Ω – and restrict the search to this 'sample set' of situations only. For instance, they may choose some of the possible worlds randomly and then find a strategy using classical minimaxing (*Monte Carlo sampling*). Such approach was employed by Ginsberg in the design of his GIB bridge playing program, with considerable results (Ginsberg 1999).

It can be shown that the actual choice of *which* possible worlds are actually included in the sample set is really important. For some minimaxing algorithms at least, using the actual prototype can reduce the order of the set while the quality of output remains the same. The results was obtained for a family of games with incomplete information on complete binary trees of arbitrary depth. The minimaxing algorithms used here are: Monte Carlo sampling (*MC*), together with vector minimaxing (*vm*) and payoff-reduction minimaxing (*prm*) – modifications of classical minimaxing, proposed by Frank, Basin & Matsubara (1998) to remove some errors that are inherent to the Monte Carlo approach.

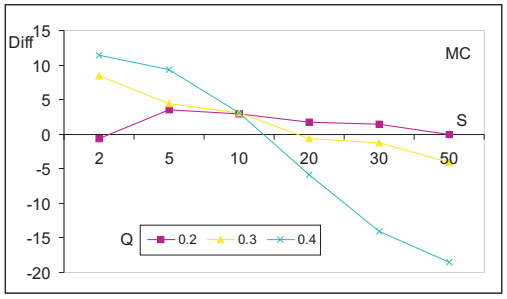
3.1 Experiments on Random Binary Tree Games

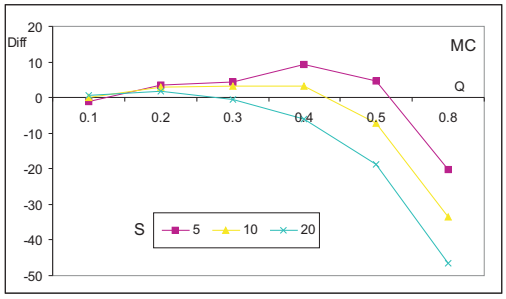
The experiment idea is strongly based on the experiments described in (Frank et al. 1998). The test was conducted for games on complete binary trees of depth D. For any of N possible worlds from Ω a payoff is assigned to every tree leaf; the payoff may be either 0 or 1. If a game ends in leaf l and world w appears to be the actual world, the player wins the payoff value assigned to (l, w), and the opponent loses the same amount. A possible world is described with a list of payoffs for all the tree leaves in this world, called a *payoff vector*. Thus, to generate a random game, one must generate a random payoff vector for each world. Note that two different worlds can have same payoff vectors. The player is called the MAX player, his opponent - the MIN player.

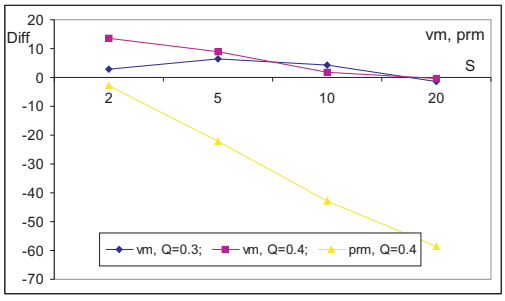
To simulate MAX's knowledge, a prototype payoff vector is chosen before generating an actual game. The 'prototype strength' may be manipulated through the probability Q as in the algorithm presented below. The parameters of an actual experiment were – D: the game tree depth, N: the number of possible worlds, Q: the prototype strength, S: the size of a random sample used to identify a strategy, and the minimaxing algorithm being used. Most experiments were conducted for games with $D = 8, N = 1000$. The results are shown on figure 3.

1. Generate a random prototype payoff vector (a sequence of payoffs for each tree leaf) for a tree of depth D.
2. Generate payoff vectors for N worlds in the following manner:
   – with probability Q, let the actual payoff for leaf l in world w be a copy of the prototype payoff for leaf l;
   – otherwise, choose the payoff randomly.
3. Use *MC*, *vm* or *prm* on a random sample of S worlds to identify the player's strategy. Compute the payoff of the strategy in each of N worlds from Ω, assuming that the opponent is omniscient (i.e. he/she knows which world we're in actually).
4. Use the same algorithm on the prototype world alone ($S = 1$) to identify the strategy. Compute the payoffs under the same assumption.
5. Compare the results, assuming equal likelihood of worlds from Ω. A strategy is better if it succeeds in more worlds, i.e. if the computed payoff string includes more 1s.
6. Repeat the steps 1000 times. Count how many times the strategy based on the prototype only was better (+), equivalent (=) and worse (−) than the strategy derived from the random sample. Compute the difference between the number of better and worse results (*Diff*).

Fig. 3. Example results (in [%]) for tree depth $D = 8$ and $N = 1000$ worlds; MAX is leading. Algorithms: MC, vm and prm.







	+	=	−	*Diff*
MC ($S = 5$, $Q = 0.4$)	21.7	66.0	12.3	9.4
MC ($S = 10$, $Q = 0.3$)	13.3	76.5	10.2	2.1
MC ($S = 20$, $Q = 0.2$)	8.0	85.8	6.2	1.8
vm ($S = 5$, $Q = 0.3$)	15.1	76.1	8.8	6.3
vm ($S = 5$, $Q = 0.4$)	22.0	64.9	13.1	8.9
vm ($S = 10$, $Q = 0.3$)	13.2	77.8	9.0	4.2
prm ($S = 2$, $Q = 0.4$)	9.4	78.3	12.3	-2.9
prm ($S = 5$, $Q = 0.4$)	9.2	59.3	31.5	-22.3

Note that:

– sampling is a search based on a subset of the available knowledge. The more worlds are checked, the better the output should be. However, the results of the experiment show that choosing *which* worlds

will be included in the sample plays an important role, too. For instance, for $Q = 0.2$ Monte Carlo sampling on 1 payoff vector only (the prototype one) led to results of the same quality as sampling on a random set 50 times larger!
- the opponent's omniscience assumption implies that the player is interested in finding these worlds in which the opponent can't beat him (then the player may construct a strategy that fits to the largest number of such worlds). However, the prototype world is not included in Ω automatically. Thus, 'playing for this world' doesn't guarantee anything – since the world doesn't even have to belong to the set of possible worlds! One may suspect that there are many worlds in Ω with payoffs same as in the prototype – but that's not true in most cases. On the contrary, the probability of at least one world same as the prototype being in the actual Ω set is very small. For the standard setting of the experiment ($D = 8, N = 1000$) this probability is virtually 0, unless $Q > 0.9$ (for $Q = 0.1$ the probability is some 10^{-64}, and for $Q = 0.4$ it's about 10^{-37})!
- prototype strength (Q): the higher the probability, the 'stronger' the prototype is. Thus, the strategy based on the prototype should be likely to succeed in more worlds. However, high Q increases also the probability that the prototype (or worlds quite similar to the prototype) is represented in the actual random sample of worlds more accurately.

The idea of sampling algorithms is to restrict the search to a subset of (more or less random) instances. The instances are usually being taken randomly from the whole searchset. However, when the sample set has to be relatively small (for instance, when the original set is huge or when the processing of a sample takes a considerable amount of time), it may be better to restrict sampling to cases 'close' to the prototype rather than trying completely randomly (risking induction from mostly non-typical cases). Thus, manipulating with the 'randomness' of the samples (and with the influence of different cases on the final solution) may be beneficial for the player.

4 Final Remarks

In this paper a concept of belief representation and modification for games with incomplete information is proposed. It's just a rough idea, but it may prove useful and convenient in many cases. Games aren't the only area of application – the method may be used in any situation of uncertainty and information flow, especially when 'natural' communication occurs (including written/spoken language utterances and/or gestures, conventional signals etc.).

The method allows to specify the evolution of an agent's uncertain knowledge in a relatively easy way. Moreover, it makes a tradeoff between optimality of the solution and the complexity of the searching process possible. If the decision making process is somewhat short of resources, it can limit the search and computations to the prototype case only, pruning much of the searchspace and obtaining suboptimal, but often relevant results.

References

1. Carmel D., Markovitch S. (1996), Learning and Using Opponent Models in Adversary Search, Technical Report CIS9609, Technion, Haifa.
2. Frank I., Basin D., Matsubara H. (1998), Finding Optimal Strategies for Imperfect Information Games, In *Proceedings of the Fifteenth National Conference on Artificial Intelligence, AAAI-98*.
3. Ginsberg M. (1999), GIB: Steps Toward an Expert-Level Bridge-Playing Program. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*, pp. 584-589.
4. Jamroga W. (1999), Modelling Artificial Intelligence on a Case of Bridge Card Play Bidding, In *Intelligent Information Systems (IIS'99). Proceedings of the 8th International Workshop*, pp. 267-277, Instytut Podstaw Informatyki PAN, Warszawa.
5. Klir G.J., Folger T.A. (1988), *Fuzzy Sets, Uncertainty and Information*, Prentice Hall, Englewood Cliffs.
6. Kronenfeld D.B., Armstrong J.D., Wilmoth S. (1985), Exploring the Internal Structure of Linguistic Categories: An Extensionist Semantic View. In Dougherty J.W.D., ed., *Directions in Cognitive Anthropology*, pp. 91-110, University of Illinois Press, Urbana.
7. Lakoff G. (1982), *C*ategories and Cognitive Models, L.A.U.T., Trier.
8. Weiss G., ed. (1999), *Multiagent Systems. A Modern Approach to Distributed Artificial Intelligence*, MIT Press, Cambridge, Mass.