N92-16575

DECISION BLOCKS: A TOOL FOR AUTOMATING DECISION MAKING IN CLIPS

Christoph F. Eick
Department of Computer Science
University of Houston
Houston, TX 77204-3475
e-mail: ceick@cs.uh.edu

Nikhil N. Mehta
GE Government Services
1050 Bay Area Boulevard
Houston, TX 77058
e-mail: nmehta@146.154.10.168

Abstract. The human capability of making complex decision is one of the most fascinating facets of human intelligence, especially if vague, judgmental, default or uncertain knowledge is involved. Unfortunately, most existing rule-based forward-chaining languages are not very suitable to simulate this aspect of human intelligence, because of their lack of support for approximate reasoning techniques needed for this task, and due to the lack of specific constructs to facilitate the coding of frequently reoccurring activities in decision making processes. The paper advocates to extend CLIPS by a new component called decision block to provide better support for the design and implementation of rule-based decision support systems. A language called BIRBAL†, which is defined on the top of CLIPS, for the specification of decision blocks is introduced. Empirical experiments involving the comparison of the length of CLIPS-program with the corresponding BIRBAL-program for three different applications are surveyed. The results of these experiments suggest that for decision making intensive applications a CLIPS-program tends to be about three times longer than the corresponding BIRBAL-program.

I. INTRODUCTION

Very often in human life we are forced to make guesses in order to decide where certain objects are located, for reconstructing events that happened in the past, or for building plans in order to achieve a certain goal. The human capability of making complex decisions is one of the most fascinating facets of human intelligence. Usually, decision making involves multiple knowledge sources from which the expert extracts different clues by frequently using a set of fuzzy rules, which encode the expert's general and domain-specific knowledge. Finally, the expert comes up with a decision by combining the evidence received from the individual clues.

The problem of how to design and implement larger systems that rely on these approaches has widely been ignored by current research (some exceptions to this point will be discussed below). There is a lack of programming languages that integrate these approaches of the rule-based expert system shells that support reasoning involving imperfect knowledge adequately, and of knowledge engineering methodologies that can cope with large amounts of imperfect knowledge. Programming involving imperfect knowledge, will be referred to as fuzzy programming in the following sections, still seems to be quite far away for commercial applications. Experimental

† Birbal was a famous minister in the 16th century India, who served as an advisor to king Akbar.

systems that do support reasoning involving imperfect knowledge such as PROSPECTOR([14]), or MYCIN([10]), mostly use "single-valued approaches" for reasoning under uncertainty; that is, they assign a probability-like value to each predicate of interest. However, single-valued approaches have problems coping with ignorance and with different degrees of reliability in different knowledge sources. Two-valued approaches that intend to overcome these problems have been advocated in the literature; most of these approaches use Dempster/Shafer's theory of evidence as the underlying knowledge representation framework ([1], [24], [28]) or they assign priorities to rules and use these priorities in a pragmatic way when combining evidence [36]. These approaches are capable of assigning --- in addition to probabilities --- reliabilities to predicates and rules and have no difficulties with representing ignorance, which makes them very attractive for automating human decision making in uncertain environments. The RUM-system([7]) advocates the use of a 3-layered reasoning strategy, which distinguishes between representation, inference, and control, which is defined on the top of a two-valued approach. Two other experimental systems, INFERNO([27]) and ARIES ([1]), that rely on two-valued approaches have been described in the literature. Finally, some efforts have been made to integrate approximate reasoning methods with Prolog ([4], [21]).

The main topic of this paper is the discussion of techniques and concepts that facilitate the implementation of rule-based decision support systems that have to cope with imperfect knowledge. A language construct, called decision block, that facilitates the automation of decision making is proposed, and its features and its integration with rule-based forward chaining languages are discussed in some detail. The paper is organized as follows: Section 2 introduces decision blocks and a language BIRBAL that supports decision blocks. Section 3 discusses the implementation of BIRBAL.

II. DECISION BLOCKS

In this section we are going to provide the programmer with a language construct, called decision block. This construct eases the automation of the more general aspects of decision making by reducing the number of rules as well as the complexity of individual rules. Decision Blocks rely on the technique called decision making by evidence combination (DBE) that can be characterized as:

(1) Rules in a rule-set are assumed to be independent, providing positive or negative evidence for or against making a certain decision in a certain situation. Rules approximate the basic principles of a particular domain.
(2) Smooth decision making is supported. If the left hand side (LHS) of a rule is only partially true the amount of evidence provided by the rule will be decreased.
(3) After the rules of a rule-set have been processed completely, the evidence provided for different decisions is combined and the best decision is selected. That is, a two-layered inference strategy is used that separates decision making from decision execution.

When using this approach, no artificial dependencies between rules need to be introduced, and errors in a rule-set can be more easily detected, because each rule-set returns a ranking of the available decisions and no longer only the chosen decision. Using the above approach rule-sets that encode decision making processes look as follows:

(R_1 (if A) (then provide evidence for D_1 with amount a1))
(R_2 (if C) (then provide evidence for D_1 with amount a2))
(R_3 (if B) (then provide evidence for D_2 with amount a3))
(R_4 (if "true") (then provide evidence for D_3 with amount a4))

In general, assuming that the decision with the highest amount of positive evidence is chosen, a particular selection a1,...,a4 is correct, as long as it satisfies the following equations:

$$a1 \oplus a2 > a3$$
$$a3 > a1,\ a3 > a2$$
$$a1 > a4,\ a2 > a4$$

The symbol ⊕ refers to the operator that combines evidence received from different rules in the context of the underlying method for approximate reasoning. For example, the first equation expresses that the result of combining the amount of evidence a1 and a2 has to be greater than the amount a3, reflecting that if both A and C are present D1 and not D2 should be selected. Or, to give another example, a3 has to be greater than a2, because D2 is preferred if B and C, but not A are observed.

So far, our discussion abstracted from the underlying approach for approximate reasoning. We consider a method M to automate approximate reasoning to be a pair M=(O, T) that consists of a set of Operators O operating on the type T. In order to be suitable for decision making by evidence combination, we require that O provides at least the following operators:

$\wedge$	$\in$	$T \times T$	$\rightarrow$	T is called the and-operator
$\vee$	$\in$	$T \times T$	$\rightarrow$	T is called the or-operator
not	$\in$	T	$\rightarrow$	T is called the not-operator
$\longrightarrow$	$\in$	$T \times T$	$\rightarrow$	T is called the modus-ponens-operator
$\oplus$	$\in$	$T \times T$	$\rightarrow$	T is called the combination-operator
$\subseteq$	$\in$	$T \times T$	$\rightarrow$	BOOLEAN. Furthermore, $\subseteq$ has to be an order relation: it has to be reflexive, anti-symmetrical, and transitive.

T represents the type used by the underlying methodology to measure the truth of imperfect knowledge. In a Bayesian system T would be set to [0 1], in the case of MYCIN certainty factors T would be [-1 1], and in the case of two-valued interval approach T would be:

$\{(x,y) \in \Re \mid 0 \leq x \leq y \leq 1\}$; where $\Re$ is a set of real numbers.

In general, various abstract data-types have been proposed in the literature, e.g. ([1], [6], [7], [10], [27]), that are suitable for decision making by evidence combination. In the following we assume that the above operators are applied in infix-form. For example, if the following rule R

(R (if (A and (B or not (C))) then (provide evidence for D amount x))

is processed, its amount of evidence for D would be computed as follows:

$(a \wedge (b \vee (not(c))) \longrightarrow x$

in which *a, b, c, x* $\in$ T, describes the belief associated with A, B, C, and with the rule R respectively. Or, in our original example D2 is considered to be better than D1 if:

$(a \longrightarrow a1) \oplus (c \longrightarrow a2) \subseteq (b \longrightarrow a3)$

In summary, when automating decision making relying on the above framework, we will first apply the operators to process the LHS of a rule, then we will use the operator $\longrightarrow$ to compute the evidence provided by a particular rule, and then we will combine the evidence inferred by different rules for the same predicate using the operator ⊕ for each decision candidate. Finally, the order relation $\subseteq$ is used to select the best decision candidate(s).

Decision blocks enable programmers of decision support systems to identify independent units of decision making. More specifically, a decision block consists of (see Figure 1):

Decision Block

Environment	Decision Candidates	Decision Making Policy	Rule-Set
	Actions		

Figure 1. Components of a Decision Block

(1) A *decision specification* that enumerates decision candidates from which the best decision(s) has (have) to be selected, and which associates an *action* with each decision candidate which will be executed in the case that a particular decision is chosen.
(2) An *environment* which consists of a set of context variables that describe the context in which a particular decision has to be made.
(3) A *rule-set*, whose members provide positive and/or negative evidence for or against the particular decision candidate. Rule-sets are further subdivided into disjoint *ruleclusters* that represent rules that are related to the same knowledge source. Evidence provided by rules from belonging to different ruleclusters is considered to be independent evidence, whereas rules belonging to the same rulecluster are assumed to provide dependent evidence. Furthermore, *exceptions* can be associated with rules. Exceptions are represented by augmenting production rules by an exception LHS and an alternate right hand side (RHS). In the case that the rule's LHS and the exception's LHS are satisfied, the corresponding alternate RHS is executed.
(4) A *decision making policy* that encodes the decision procedure for selecting a decision or a set of decisions after all clues relevant for the decision making have been analyzed.

Phase 1	Phase 2	Phase 3	Phase 4
Evidence Generation	Evidence Combination	Decision Selection	Decision Execution
Rules and their associated exceptions are evaluated giving preference to the most specific exception.	Evidence received from different rules is combined.	A set of decisions is selected depending on the decision making policy.	The selected decisions are carried out by executing their associated actions.

Figure 2. Phases of a Decision Block

Decision blocks automate rule-based decision making by using a four layered inference strategy, as depicted in Figure 2. First, rules and their associated exceptions are evaluated, giving priority to the most specific exception which is applicable for the rule. Second, dependent evidence within each rulecluster, and independent evidence associated with different ruleclusters is combined. For combining dependent evidence a second evidence combination function was provided, which is given in Appendix 1, and whose properties are discussed in more detail in ([15]). Third, a set of decisions is selected according to the strategy outlined in the decision block's policy. Decision making policies supported by decision blocks, which are evaluated with respect to the operator $\sqsubseteq$ include: select the best decision, select the best n decisions, select all decisions, or the best n decisions better than a threshold value α ($0 \leq \alpha \leq 1$). Fourth, the selected decisions are carried out, executing the actions associated with the selected decisions.

Figure 3. gives an example (for detailed explanation of different examples see [25]) of a typical decision block, named scholarships, which encodes the awarding of scholarships to the students on the basis of their academic performance, experience as teaching assistant and age.

```
(fuzzy functions
   (curr-perform (arity 1))
   (overall-perform (arity 1))
   (exp-as-TA (arity 1))
   (financial-need (arity 1))
   (oldness (arity 1)))
(decision-block scholarships
   (environment (?univ))
   (action-specifications
    (qualification
      ((qual-of ?ssno ?univ) with (?lb ?ub))
      =>
      (printout t "Qualified Student is " ?ssno ?univ)))
 (decision making policy (select ALL decisions with mv > 0.8))
 (rule-set criteria
   (sc1
      (student (ssno ?ssno) (univ ?univ) (cgpa ?cgpa) (ogpa ?ogpa))
      (%curr-perform ?cgpa)
      (%overall-perform ?ogpa)
      (test (and (between ?cgpa 0.0 4.0) (between ?ogpa 0.0 4.0)))
      =>
     (infer (qual-of ?ssno ?univ) with (0.9 1.0)))
  (sc2
     (student (ssno ?ssno) (univ ?univ) (expTA ?expTA))
     (%exp-as-TA ?expTA)
     =>
     (infer (qual-of ?ssno ?univ) with (0.8 1.0)))
  (sc3
     (student (ssno ?ssno) (univ ?univ) (fin-need ?fneed))
     (%financial-need ?fneed)
     =>
    (infer (qual-of ?ssno ?univ) with (0.75 1.0)))
 (sc4
    (student (ssno ?ssno) (univ ?univ) (age ?age))
    (%oldness ?age)
    =>
    (infer (qual-of ?ssno ?univ) with (0.65 1.0)))))
```

Figure 3. Example of a Decision Block: scholarships

Before a decision block can be used, all fuzzy functions (that is, functions that return intervals as results) with it's arity (i.e., number of arguments) have to be identified. The action-specifications of the decision block enumerates the decision candidates, which are represented as LHS patterns

((qual-of ?ssno ?univ) with (?lb ?ub)),

and RHS gives the action(s) which has(have) to be performed, if the corresponding decision is selected. For example, if a student from the university of Houston with ?ssno equal to 123456789 is selected depending upon the decision making policy then "Qualified Student is 123456789 Houston" is printed out. The environment definition defines the context variables used by the decision block. These variables have to be initialized by the call of the decision-block. These variables are local to the decision-block and need to be passed (in this case ?univ) whenever the decision-block is invoked. The decision making policy specifies what decisions need to be selected from the set of decisions taken, here the decision making policy applied is specified as

"(select ALL decisions with mv > 0.8)"

i.e. the combined evidence received from each of the rules from the rule-set should have the mean-value greater than 0.8.

Now let's see what each rule in the rule-set expresses. First we will see intuitively what is being signified by intervals specified (interested reader should refer to the Appendix 1) with the Two-valued approach we have taken. We can express the fact that we know nothing about certain proposition P by assigning interval [0 1], i.e., unknown can be directly expressed in the interval approach. Also the usage of classical probability becomes the special case of Two-valued approach because we can express single probability values by assigning same lower and upper bounds, e.g., [0.4 0.4], where uncertainty is 0 and reliability is 1, and we are sure about the proposition. Intuitively speaking if the difference between the lower and upper bound increases the uncertainty of the proposition increases and if the difference between them decreases certainty about the proposition increases. Also the evidence provided for the proposition is considered as negative if the mean-value of the interval is less than 0.5.

The LHS of any rule could be any valid CLIPS LHS pattern plus the fuzzy function(s) (C functions), if any, identified by '%' followed by the arguments of the functions. Let's take the first rule sc1 which provides the evidence towards the qualification of the student depending on the academic performance of the student, i.e., student's current GPA and overall GPA. The performance of the student depending on GPA is difficult to quantify since the student having overall GPA of 3.75 and other having 3.80 lies in the same category. Also same can be said for current GPA , but the combined performance might differ extensively since the current GPA will affect the overall GPA drastically depending upon the value. Hence the combined performance is fuzzy. The infer statement in the first rule suggests that it has higher contributing power towards the final decision since the interval applied to it is [0.9 1.0]. The second rule sc2 will contribute towards the final decision depending upon the experience as teaching assistant but the rule has lower contributing power than the first one since the interval applied to it is [0.8 1.0]. Similarly other rules will contribute towards the final decision.

The evidence provided by the LHS fuzzy predicates such as performance, experience, financial-need and age will be used in computations using logical connectives ($\wedge$, $\vee$, not) which in our case is $\wedge$, and modus ponen functions ($\longrightarrow$) and new evidence interval will be computed. This new value will be contributed by each rule towards the final decision, i.e., qualification of the student for receiving a scholarship. Each rule can either provide positive or negative evidence. Next the evidence provided by each rule is combined repeatedly using the operator $\oplus$ until the evidence provided by each rule towards the final decision is combined. In the third phase depending upon the decision making policy the decision regarding the qualification is made from the set of competing candidates and in the last phase the action specified by action-specifications is taken, in our case it's simply printing out all the candidates whose mean-value of the interval is greater than 0.8. In general the action-specifications can be used to sort the candidates in the ascending order of mean-values or could involve further computations for the final selection or could invoke another decision-block or simply prints the first 10 candidates called for the interview putting other candidates on the waiting list, etc. We can see from the previous discussion how smooth the entire process is and how simple it is to code the decision-block. We will see in the next section implementation of the decision-block, that if we need to do all that is specified in the previous discussion using pure CLIPS how much extensive coding is required and burdensome for the programmer which is not the case with the decision block since the compiler takes away that burden from the programmer.

III. IMPLEMENTATION OF DECISION BLOCKS

In this section, we will discuss the implementation of a rule-based language that supports decision blocks, and will report on some empirical results concerning the benefits of decision blocks. More specifically, we will report on the integration of decision blocks into a rule-based forward chaining language called CLIPS([19]), which was developed by NASA. The extended language is called BIRBAL. A BIRBAL-programmer can in addition to CLIPS rules define decision blocks and call these decision blocks within a CLIPS-program.

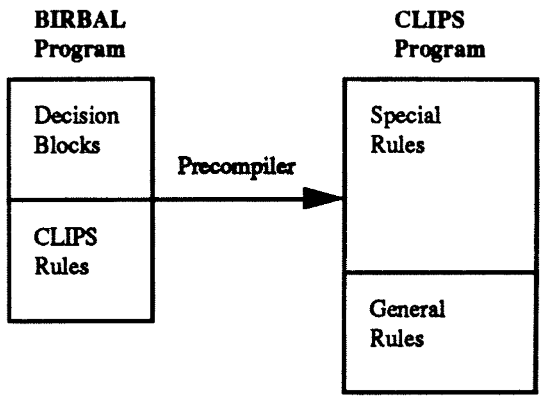

Figure 4. Implementation of Decision-Blocks

A precompiler has been provided, as a part of a Master's thesis ([25]), that maps a BIRBAL-program that consists of decision blocks and CLIPS rules into a program that uses pure CLIPS (see Figure 4). The precompiler is written in C has about 1400 lines of symbolic code. It was developed using the UNIX-compiler generating tools *lex* and *yacc*. The developed precompiler relies on a rule-mapping that maps BIRBAL-rules into CLIPS rules; that is, every rule defined within a decision block is transformed into a single CLIPS-rule that simulates its behavior. However, the generated CLIPS rules are much larger in size, which can be attributed to:

- CLIPS which does not support reasoning under uncertainty. This means that computations involving operators for approximate reasoning have to be provided manually for each rule.
- CLIPS does not support context variables, which implies that they have to be stored as assertions in the CLIPS working memory. The variable management is further complicated by the fact that there could be multiple active decision blocks, which use equally named variables.
- Exceptions are not supported; therefore they have to be programmed out by the programmer.
- It has to be made sure that evidence providing rules are only fired in phase 1 of the execution of a decision block; or, in more general terms, the context under which evidence-providing rules are allowed to fire has to be expressed by adding additional LHS conditions to the generated CLIPS-rule.

In order to make the above comments more transparent and to give the reader a better feeling what a programmer has to do if no decision blocks are provided, let us see how our precompiler maps the rule (refer to Figure 5).

In CLIPS's syntax a rule's LHS and RHS are separated by => symbol, and all condition elements that appear in a rule's LHS are assumed to be connected by 'and'. Furthermore, in CLIPS match-variables are prefixed by '?'. The above rule starts with two control assertions (active-db_ scholarships) and (status_ scholarships phase 1) that make sure that the generated rule is only fired when the decision block scholarships is active and in phase 1. The rule sc1 refers to one context-variable. Context variables are represented in the CLIPS environment by assertions of the form (var_ <db-name> <var-name> <var-value>); e.g., (var_ scholarships univ Houston), expresses that variable 'univ' of decision block scholarships has the value Houston. The values of the context variables have to be retrieved by adding one LHS condition to the rule. Furthermore, the LHS conditions of the BIRBAL-rules have to be simulated. Non fuzzy-conditions do not pose any particular problem; they are simply copied into the generated CLIPS-rule. In this particular case, it is the test pattern which checks whether the GPA of the student is within the valid range or not. The implementation of the fuzzy conditions is much more complicated. First we assume that the C-functions that implement the definition of curr-perform and overall-perform has been provided by the user.

```
(sc1
   (student (ssno ?ssno) (univ ?univ) (cgpa ?cgpa) (ogpa ?ogpa))
   (%curr-perform ?cgpa)
   (%overall-perform ?ogpa)
   (test (and (between ?cgpa 0.0 4.0) (between ?ogpa 0.0 4.0)))
   =>
   (infer (qual-of ?ssno ?univ) with (0.9 1.0)))
The above rule sc1 is transformed into the following CLIPS-rule:
(defrule sc1-phase1
   (declare (salience 555))
   (active-db_ ?db-name_&scholarships)
   (status_ ?db-name_ phase ?phase_&1)
   (var_ ?db-name_ univ ?univ)
   (student (ssno ?ssno) (univ ?univ) (cgpa ?cgpa) (ogpa ?ogpa))
   (test (and (between ?cgpa 0.0 4.0) (between ?ogpa 0.0 4.0)))
   =>
   (bind ?l1-u1_ (curr-perform ?cgpa))
   (bind ?l1_ (nth 1 (str-explode ?l1-u1_)))
   (bind ?u1_ (nth 2 (str-explode ?l1-u1_)))
   (bind ?l2-u2_ (overall-perform ?ogpa))
   (bind ?l2_ (nth 1 (str-explode ?l2-u2_)))
   (bind ?u2_ (nth 2 (str-explode ?l2-u2_)))
   (bind ?y1_ (* ?l1_ ?l2_))
   (bind ?y2_ (* ?u1_ ?u2_))
   (if (and (!= ?y1_ 0) (>= (+ ?y1_ 0.9) 1) (>= (+ ?y2_ 1.0) 1)) then
      (bind ?lb_ (/ (- (+ 0.9 ?y1_) 1) ?y1_))
      (bind ?ub_ (/ (- (+ 1.0 ?y2_) 1) ?y2_))
      (assert (sc1 ?db-name_ 2 qual-of ?ssno ?univ with ?lb_ ?ub_))))
```

Figure 5. Rule Transformation from Decision Block rule to CLIPS rule.

Each of these functions returns an interval, measuring the performance of the student for which it were called. Second, the approximate reasoning methods described in section II, have to be used to compute the rule's evidence for the qualification of the students. In this particular case two fuzzy conditions are used, which means that the belief in the rule's LHS can be equated to the academic performance of a particular student, if the test conditions for GPAs are satisfied. The first eight RHS actions call the C-functions curr-perform and overall-perform, bind the returned intervals, and bind the lower and upper bounds to variables ?y1_ and ?y2_ after applying logical connectives (see Appendix 1) to the returned intervals. The rest of the rule's RHS simulates the operator '-→' and computes the rule's amount of evidence. In case that the amount of positive evidence is [0 1] then the rule does nothing; otherwise it asserts an assertion specifying the amount of evidence provided for the decision candidate. For example, if the rule sc1 provides a positive evidence of [0.75 1.0], the following pattern

"(sc1 scholarships 2 qual-of 123456789 Houston with 0.75 1.0) "

would be asserted. These assertions are used later in phase 2 to combine the different pieces of evidence associated with the same student. It is also important to note that the generated code would become much more complicated in case that the LHS of the rule references more than two fuzzy conditions connected by different logical connectives and with the presence of exception conditions (see [25]). In that case, it would be necessary to add code that simulates the combination of $\vee$, $\wedge$, and *not* operators.

In general, the example demonstrates that implementing the approximate reasoning methods by hand is highly complicated and time consuming, if no methods for approximate reasoning have been integrated into a rule-based language. Even worse, these computations have to be provided for every rule of a rule-set, even if the computations are similar or the same.

Moreover, the precompiler transforms a decision block's decision making policy into a rule that generates assertions, parameterizing the computations of general rules that simulate the selected decision making policy. Also the decision blocks decision specification is transformed into rules, whose right hand sides consists of the actions associated with a decision candidate, whereas the LHS of the generated rule checks for assertions that state that the corresponding decision has been selected by the decision making policy. Additionally to the rules that are generated specifically for a particular decision block, our implementation relies on a set of general rules that perform general tasks such as evidence combination, switching between the phases of the implementation of a decision block, management of calls of decision blocks, and removal of trash (see Figure 6).

We also made some experiments comparing the length of a decision block with the length of the generated CLIPS program that implements the decision block. These experiments included a slightly more complex version of the scholarships decision block, the decision blocks that simulate the assignment of scholarships at a university, and a larger decision block for a problem of medical parasitology. The results are summarized in Table 1. We claim that these numbers give a good indication of the benefits of decision blocks with respect to a programmer's productivity in decision making intensive applications, such as those analyzed in the benchmark. A pair (a, b) in the Table 1 indicates that the corresponding program has 'a' lines and 'b' characters.

In general, the experiment suggests that the generated CLIPS program is about 4 times longer than the corresponding decision block if DBE is implemented, which makes it quite apparent why the conventional approach has been used quite frequently in practice, and not decision making by evidence combination (DBE). Due to the lack of supportive constructs in commercial forward-chaining languages the received programs tend to be very long, if DBE is used. However, if higher order constructs such as decision blocks are integrated into a rule-based languages, this remark is no longer true. We even go further and claim --- that programs developed using decision blocks tend to be significantly shorter than programs that were developed using the conventional approach. Moreover, the development of a precompiler was not very complex, which demonstrates that decision blocks can be provided at a low cost on the top of rule-based languages such as CLIPS([19]), OPS([18]), or ART([2]), and that decision blocks can be easily integrated into forward chaining systems. In summary, the availability of decision blocks or similar constructs is an important prerequisite to develop decision support systems at reasonable cost that rely on decision making by evidence combination.

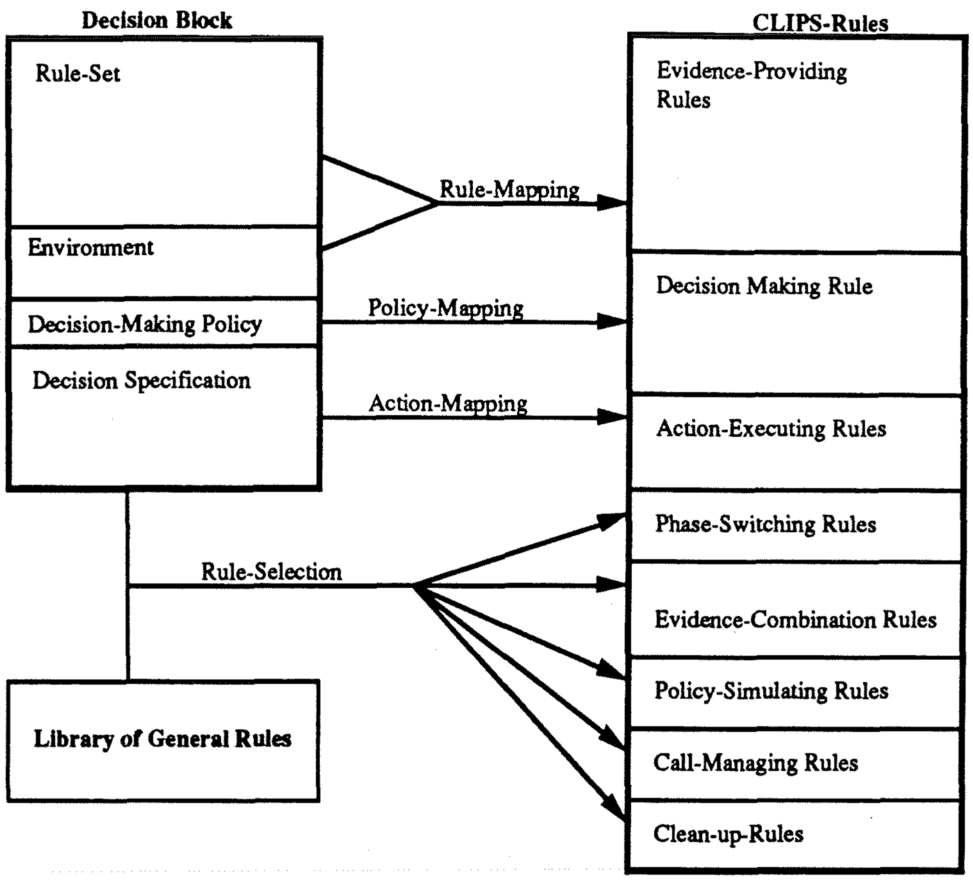

Figure 6. Mapping Decision Blocks to a Rule-Based Language

Example / Language	Opening-bid in the game of Bridge	Medical Parasitology	Scholarship Awarding
BIRBAL	(45, 1175)	(167, 5234)	(37, 1062)
CLIPS	(174, 5743)	(701, 22486)	(180, 4720)

Table 1. Comparison of the Complexity of CLIPS and BIRBAL

IV. CONCLUSION

This paper studied the problem of designing and implementing complex decision support systems. Computerized decision making that treats decision making as a problem of evidence combination was introduced. A language construct called decision block that facilitates the implementation has been proposed. A language called BIRBAL that integrates decision blocks with CLIPS has been provided. Decision blocks offers several advantages over classical rule-based languages such as ART, CLIPS and OPS for automating complex decision making processes. First, the availability of exception handling facilities and evidence combination techniques enables one to program close to the expert level, which is very important for explaining the system's behavior and for system validation. Second, our approach supports smooth decision making. Minor changes will only slightly affect the final ranking of the decision candidates. Third, decision blocks provide encapsulation and allow to modularize complex decision making processes. Fourth, we claim that decision blocks increase a programmer's productivity significantly because the precompiler takes care of many tasks such as reasoning under uncertainty, combination of evidence, removal of trash, ranking of decision candidates, or execution of decisions, which no longer have to be coded by a BIRBAL-programmer. Finally, we showed that decision blocks can be provided at a low cost on the top of existing rule-based programming languages, such as CLIPS.

REFERENCES

[1] L. Appelbaum, and E.Ruspini, "ARIES An Approximate Reasoning Inference Engine," in *[18]*, pp. 745-765.

[2] Inference Corporation, *ART Reference Manual.* Los Angeles Inference Corporation, 1986.

[3] J.F. Baldwin, and N. Gould, "Feasible Algorithms for Approximate Reasoning using Fuzzy Logic," *Fuzzy Set Systems,* vol. 3, pp. 225-251, 1980.

[4] J.F. Baldwin, "Evidential Support Logic Programming," *Fuzzy Sets and Systems,* Vol. 24, pp. 1-26, 1987.

[5] V. Barker, and D. O'Connor, "Expert Systems for Configuration at Digital XCON and Beyond," *CACM,* Vol. 32, No. 3, pp. 298-318, 1989.

[6] A. Basu, and A. Dutta, "Reasoning with Imprecise Knowledge to Enhance Intelligent Decision Support," *IEEE Transactions on Systems, Man and Cybernetics,* Vol. 19, No. 4, pp.756-770, 1989.

[7] P. Bonissone, S. Gans, and K. Decker, "RUM A Layered Architecture for Reasoning in Uncertainty," in *Proc. 10th IJCAI-conference,* Milan, pp.891-898, 1987.

[8] P. Bonissone, D. Cyrluk, J. Goodwin, and J.Stillman, "Uncertainty and Incompleteness Breaking the Symmetry of Defeasible Reasoning" in *5th Workshop on Uncertainty in AI,* pp. 34-45, Detroit, 1989.

[9] L.Browston, R. Farrell, E. Kant, and N.Martin, *Programming Expert Systems in OPS5,* Reading Addison-Wesley, 1985.

[10] B. Buchanan, and E. Shortliffe, *Rule-Based Expert Systems,* Reading Addison Wesley, 1984.

[11] P. Cheeseman, "Probabilistic versus Fuzzy Reasoning," in L. Kanal, *Uncertainty in AI,* Amsterdam North Holland, pp. 85-102, 1986.

[12] P. Cohen, *Heuristic Reasoning about Uncertainty An Artificial Intelligence Approach,* New York Pitman Publishing Limited, 1985.

[13] A.P. Dempster, "A Generalization of Bayesian Inference," *Jour. Royal Stat. Soc.,* B, V 30, pp. 205-247, 1968.

[14] R. Duda, J. Gaschnig, and P. Hart, "Model Design in the PROSPECTOR Consultant System for Mineral Exploration," in Michie's *Expert Systems in the Micro Electronic Age,* Edinburgh University Press, 1979.

[15] C.F. Eick, "Uncertainty Management for Fuzzy Decision Support Systems," in *Proc. 4th Workshop on Uncertainty in AI,* St.Paul, August 1988, pp. 98-108.

[16] C.F. Eick et al., "Computer Bridge - A Challenge for AI," in Z. Ras (eds.), *Methodologies for Intelligent Systems, 5.* New York North-Holland, pp. 59-67, 1990.

[17] C.F. Eick, Yao and H. Fu, "More Flexible Use of Variables in Rule-Based Programming," in *Proc. 2nd Int. Symposium on Artificial Intelligence,* Monterrey, Oct. 1989.

[18] C. Forgy, "OPS83 Report," Technical Report, Dept. of Computer Science, Carnegie-Mellon University, 1983.

[19] J. Giarratano, and G. Riley, *Expert Systems Principles and Programming,* Boston PWS-Kent Pub. Co., 1989.

[20] M. Gupta, A. Kandel, W. Brandler, and J. Kiszka, *Approximate Reasoning in Expert Systems,* Amsterdam, North Holland, 1985.

[21] C.J. Hinde, "Fuzzy Prolog," *Int. Journal of Man-Machine Studies,* pp. 569-595, 1986.

[22] A. Kandel, *Fuzzy Mathematical Techniques with Applications,* Read. Addison Wesley, 1986.
[23] R. Loui, "Evidential Reasoning in a Network Usage Prediction Testbed," *Proc. 4th Workshop on Uncertainty in AI,* St. Paul, 1988, pp. 257-265.
[24] S. Lu, and H. Stephanou, "A set-theoretical framework for the processing of uncertain information," in *Proc. AAAI-conference,* Austin, 1984, pp. 216-221.
[25] N.N. Mehta, "BIRBAL - A Rule-Based Language for Decision Making," Master's Thesis, University of Houston, December 1990.
[26] J. Prugh, "A Knowledge-Based Approach to Bridge Defense," Master's Thesis, University of Houston, May 1989.
[27] J. Quinlan, "INFERNO - A Cautious Approach to Uncertain Inference," *Computer Jour.*, Vol. 26, pp. 255-266, 1983.
[28] G. Shafer, *A Mathematical Theory of Evidence.* Princeton University Press, 1976.
[29] N. Shirouzu, *Norihiko Time for Some Fuzzy Thinking,* in TIME September 25, 1989, pp. 79.
[30] E.Soloway, J.Bachant, and K. Jensen, "Accessing the Maintainability of XCON-in-RIME Coping with Problems of a VERY Large Rule-Base," in *Proc. 6th National Conf. on Artificial Intelligence,* Seattle, 1987, pp. 824-829.
[31] H. Stephanou, and A. Sage, "Perspectives on Imperfect Information Processing," *IEEE Trans. on Systems, Man, and Cybernetics,* Vol. 17, no. 5, pp. 780-798, Sept. 1987.
[32] D. Touretzky, *The Mathematics of Inheritance Systems,* Los Altos M. Kaufman Pub., 1986.
[33] E. Trillas, and L. Valverde, "On Mode and Implication in Approximate Reasoning," in *[18],* pp. 157-166.
[34] C. Tsen, "A Knowledge-Based Approach to Bridge Bidding," Master's Thesis, University of Houston, June 1988.
[35] D. Vaughan, B. Perrin, R. Yadrick, and P. Holden, "Comparing Expert Systems Built using Different Uncertain Inference Systems," in *Proc. Fifth Workshop on Uncertainty in AI,* Detroit, August 1989, pp. 369-376.
[36] R. Yager, "Nonmonotonic Reasoning via Possibility Theory," in *Proc. 4th Workshop on Uncertainty in AI,* St.Paul, 1988, pp. 368-373.
[37] L.A. Zadeh, "The Role of Fuzzy Logic in the Management of Uncertainty in Expert Sys.," in *[18],* pp. 3-31.

APPENDIX 1. BIRBAL'S UNCERTAINTY MANAGEMENT

This section describes the operators that are supported in BIRBAL for approximate reasoning. BIRBAL relies on a two-valued interval approach to automate approximate reasoning which measures the *belief* that a certain proposition P is true by assigning an interval [a b] to P, expressing the following semantics:

(1) The probability that P is true is at least a.
The *confirmation* of P is a: conf[a b] = a.
(2) The probability that P is false is at least (1-b).
The *disconfirmation* of P is (1-b): disconf([a b]) =1-b.
(3) The *uncertainty* of our belief concerning P is measured by unc([a b]) = b-a.
(4) The *mean-value* of our belief concerning P is measured by $\left\{\frac{a+b}{2}\right\}$: mv([a b]) = $\left\{\frac{a+b}{2}\right\}$.

For example, if we assign an interval [0.40 0.99] to P we express the following, the confirmation of P is 40% and the disconfirmation of P is 1%. That is, 40% of the probability is assigned to P, and 1% of the probability is assigned to (not P). It is unknown how the remaining probability (59%) is distributed we don't know how much of this probability is assigned to P and how much is assigned to (not P). The uncertainty is 59%, and the mean-value is 69.5%. A special case is the interval [0 1], which expresses the fact that we know nothing about a proposition P. We used the following operators for $\vee$, $\wedge$, not, $\rightarrow$, $\oplus$, and $\subseteq$ in the project, whose definitions are as follows:

Let [a b], [c d], $[l_1\ u_1]$, $[l_2\ u_2]$ be intervals:

$[a\ b] \wedge [c\ d] := [\min(a, c)\ \min(b, d)]$

$[a\ b] \vee [c\ d] := [\max(a, c)\ \max(b, d)]$

$\text{not}([a\ b]) := [1-b\ \ 1-a]$

$(a\ b) \longrightarrow (c\ d) :=$ if $(a \neq 0 \wedge (c+a \geq 1) \wedge (b+d \geq 1))$

then $\left[\frac{c+a-1}{a}\ \frac{b+d-1}{b}\right]$

else [0 1]

$$[l_1\ u_1] \oplus [l_2\ u_2] := \left[\frac{l_1 \times u_2 + l_2 \times u_1 - l_1 \times l_2}{1 - l_1 \times (1-u_2) - l_2 \times (1-u_1)} \quad \frac{u_1 \times u_2}{1 - l_1 \times (1-u_2) - l_2 \times (1-u_1)}\right]$$

$$[a\ \ b] \subseteq [c\ \ d] := \frac{a+b}{2} \leq \frac{c+d}{2}$$

The operators for and-, or-, and negation are identical to those advocated by Fuzzy Logic -- only generalized in the context of intervals. The modus ponens operator $\longrightarrow$ is used as follows to associate evidence with predicates appearing on the LHS of rules. For example, if we have a rule:

(R (if E) (then (infer H with [c d]))),

and our belief in E is measured by [a b], then the above rule provides evidence for H, whose amount is measured by: $(a\,b) \longrightarrow (c\,d)$.

The proposed modus ponens operator generalizes a modus ponens operator given in ([33]) for intervals. The proposed operator supports smooth decision making, as can be seen for the rule r1, given below

(r1 (if (tall $x)) (then (infer (strong $x) with (0.5 0.9))))

Varying our belief assigned to the tallness of a person, we receive,

Interval for the Tallness	**r1's Positive Evidence**
[1.0 1.0]	[0.500 0.900]
[0.9 0.9]	[0.444 0.889]
[0.8 0.8]	[0.375 0.875]
[0.7 0.7]	[0.285 0.857]

Note that the mean-value of the rule's conclusion decreases if the probability of the LHS decreases means that conclusions based on uncertain knowledge are less reliable and provide less positive evidence. In the above framework negative evidence is treated as positive evidence for the negation of the hypothesis of interest, e.g., when processing: (r1 (if E) (then not(H) with (0.7 1))), we will infer an interval that describes the positive evidence for not(H), and convert this interval to negative evidence for H by applying the negation function, introduced before, to the received interval. In the example, if E is definitely true [1 1], an interval [0 0.3] will be associated with H.

The operator $\oplus$, we use for combination of evidence, is Dempster's rule of combination ([13]). Intervals can easily be interpreted as probability assignment functions. If an interval [a b] is assigned to a predicate P the corresponding probability assignment function would be: m(P) = a, m(not(P)) = 1-b, $m(\theta)$ = b-a, and $m(\emptyset) = 0$. By applying Demster's rule of combination the above formula is obtained. Finally, we use the mean-value of intervals to rank decision candidates, giving preference to decisions with the highest mean-value for the combined evidence. For example, if an interval of [0 1] is associated with D1 and an interval of [0.1 0.8] is associated with D2, D1 is preferred because its mean-value is 0.5, whereas the mean-value associated with D2 is 0.45, which is less than 0.5.

C-2