An Improved Knowledge Representation Language for XTT. Extended Tabular Graphs with Variables, Constraints and Control*

Antoni Ligęza[1]

AGH – University of Science and Technology, al. Mickiewicza 30, 30-059 Cracow, Poland, ligeza@agh.edu.pl

Abstract. *This paper presents certain extensions to the XTT knowledge representation model. XTT is a way of tabular specification of rule-based systems. The proposed extensions concerns introduction of variables, dependent attributes, functional constraints, fuzzy rules and time-delayed values. Moreover, inference control specification language is considered.*

1. Introduction Over thirty years rule-based systems prove to constitute one of the most substantial technologies within applied Artificial Intelligence (AI) [6, 4, 11]. Rules of various particular forms implement the core of numerous applications, including expert systems, decision support systems, control and monitoring systems and knowledge-based systems in general [6, 4, 3, 12, 5]. Knowledge specification with rules is used both for definition of domain knowledge as well as meta-knowledge concerning inference control. With use of rules one can specify vast bases of purely declarative knowledge (both flat as hierarchical ones), procedural knowledge including control features, and documentation knowledge covering explanations.

The technology of rule-based systems (RBS) is based on strong logical foundations [1, 8]. In fact logic constitutes the core formalism for knowledge specification and inference. And rule-based programming paradigms, including Prolog [2, 8], are based on logical deductive inference.

Although rules constitute perhaps one of the simplest and most transparent programming paradigms, practical implementation of RBS encounters some important problems. A non-trivial system may contain less than fifty rules and simultaneously it may be difficult to handle. The main problems encountered concern *complete* specification of *non-redundant* and *consistent* set of rules. This turns out to be very tedious task requiring far-going effort.

The main issue for successful design and development of a rule-based system is to find an appropriate language for knowledge representation. A recent proposal consists in using the so-called XTT – eXtended Tabular Trees [10, 8], a combination of advanced decision tables with organized within tree-like strictures. The current approach however suffers from several limitations.

This paper presents a proposal of significant improvement of the XTT technology with respect to knowledge representation and several related issues. In particular, the following aspects are the main focus of this work:

- introduction of named variables as attribute values,
- introduction of new attributes (variables) as functions of initial attributes,
- introduction of functional constraints over these variables,
- introduction of functional (calculable) values defined by rule antecedent,
- new organization of the structure of tables – an inference control graph (ICG) is introduced,
- hierarchical structure of new XTT – multi-layer structure of tables.

Moreover, the problems of using fuzzy inference rules, adaptation and learning, with use of a Wang-Mendel algorithm are mentioned in brief. The design and development approach can be extended over incorporating cases and learning from examples by generalization.

Research supported from a MNiSW Research Project *HEKATE* No.: N516 024 32/2878, Agreement No.: 2878/T02/2007/32

As the last point, the approaches to verification and validation of such rule-based systems do not solve the problem in an entire way. Verification performed after the system is designed is both costly and late. Moreover, if errors are detected, they must be corrected, and after introducing the corrections the cycle of verification must be repeated. The problem consist in the possibility of introducing new errors through correction of the old ones – there in no warranty that the verification-correction cycle is finite. In the paper an incremental verification approach is pursued.

2. XTT – the Basic Extended Tabular Trees Model In this Section the basic form of eXtended Tabular Trees [10, 7] is presented and analyzed. The XTT model for knowledge representation employs granular attributive logic [8, 9] for specification of rules in a tabular form. Similar rules operating within the same context are grouped together and form a table similar to an attributive decision table. The complete knowledge of a system is encoded with a set of partially ordered tables connected with links defining the control flow.

Specification of knowledge is performed with use of Set Attributive logical Language (SAL) allowing for non-atomic attributes values [7]. Rules of the same or similar scheme, i.e. ones based on the same attributes are combined into a special form of decision table. The *Extended Attributive Table* (XAT) (or *eXtended Table* (XT), for short) (see [8, 10]) takes the following generic form as presented with Fig. 1.

Each rule is represented by a single row. In the first column there is the rule identifier, the *Ctx* is the context common for all the rules, $A_1 \ldots A_n$ are the preconditions attributes, and $B_1 \ldots B_b$ are the retract, $C_1 \ldots C_c$ are the assert, and $H_1 \ldots H_h$ are the conclusion-defining ones. The *Ctrl* part defines the (N) next rule to be executed (if present) or the (E) else rule to be executed in case of failure. Hence, the table can specify both declarative knowledge (rules) and control knowledge (the *Ctrl* column). For further details refer to [8].

On one hand, presented basic form of XTT is elegant, concise and transparent, but simultaneously having high expressive power and flexible. It proved to be very efficient in design and implementation of several rule-based systems [10, 8]. Moreover, it facilitates the design and on-line verification of rule-based systems.

On the other hand, for realistic and practical applications the basic XTT suffer from certain shortcomings which are mainly due to limited expression power of the simple attributive logic in use. Below the ideas of some most necessary extensions are put forward and motivated.

3. Variables, Dependent Attributes, Functional Constraints, Functional Output

3.1. Motivation Consider a simple example of two conditional attributes, namely C denoting the cost and G denoting the gain. Let B denote the business result with two symbolic values, i.e. 'good' and 'bad'. We have two simple intuitive rules:

$$\begin{array}{l} rule_1 \colon G > C \longrightarrow B =' good' \\ rule_2 \colon G < C \longrightarrow B =' bad' \end{array} \tag{1}$$

Unfortunately, in the classical attributive logic it is impossible to encode these rules and hence they cannot be represented within the pure XTT formalism.

There are at least two solutions to this problem. First, one can define a new, functionally dependent attribute $Z = G - C$ and specify the rules in the XTT form as:

Table 1. A simple table with dependent attribute Z

Rule	G	C	Z	B
1	_	_	> 0	'good'
2	_	_	< 0	'bad'

Unfortunately, it is not always the case that a new, dependent attribute can be defined in a simple way. Then, one has to use functional constraints imposed on the set of attributes and a procedure for checking if these constraints are satisfied. For intuition, this can be expressed with the following Table 2.

Info		Prec	Retract	Assert	Decision	Ctrl	
I	Ctx	$A_1 \dots A_n$	$B_1 \dots B_b$	$C_1 \dots C_c$	$H_1 \dots H_h$	N	E
1	ψ	$t_{11} \dots t_{1n}$	$b_{i1} \dots b_{1b}$	$c_{11} \dots c_{1c}$	$h_{11} \dots h_{1h}$	g_1	e_1
$\vdots$	$\vdots$	$\vdots \ddots \vdots$	$\vdots \ddots \vdots$	$\vdots \ddots \vdots$	$\vdots \ddots \vdots$	$\vdots$	$\vdots$
i	ψ	$t_{i1} \dots t_{in}$	$b_{i1} \dots b_{ib}$	$c_{i1} \dots c_{ic}$	$h_{i1} \dots h_{ih}$	g_i	e_i
$\vdots$	$\vdots$	$\vdots \ddots \vdots$	$\vdots \ddots \vdots$	$\vdots \ddots \vdots$	$\vdots \ddots \vdots$	$\vdots$	$\vdots$
k	ψ	$t_{k1} \dots t_{kn}$	$b_{k1} \dots b_{kb}$	$c_{ik} \dots c_{kc}$	$h_{k1} \dots h_{kh}$	g_k	e_k

Fig. 1. The basic form of an XAT

Table 2. A simple table with functional constraints $f(X, Y)$

Rule	G	C	f(X,Y)	B
1	X	Y	$X > Y$ 0	'good'
2	X	Y	$X < Y$ 0	'bad'

Note that new variables, namely X and Y were introduced here to denote the current values of attributes G and C, respectively. The reason for introducing such variables is four-fold:

- attributes are functions mapping objects into some predefined domains; the introduced variables denote just their current values,
- some functional constraints can be defined once (using a specific set of variables names) and reused for several different sets of attributes,
- coreference constraints can be expressed with variables,
- more sophisticated rules can be expressed with use of relational symbols between attributes and variables, e.g. $G \subset X$ and then with constraints over the variables; further, constrain propagation can be defined.

Obviously, the definition of constraints must be in the form of an evaluable formula. The check for its satisfaction can be left to an external deterministic procedure.

Consider another problem when the task is to define that if the gain is up to 1000, one has to pay 20% tax, while if it is above 1000, one has to pay 200 and 30% of the additional part. Again this cannot be expressed within simple XTT formalism. For intuition, we have to define rules as presented in Fig. 2.

Here the solution requires the possibility to specify the values of output attributes being functionally dependent on the input attribute values.

3.2. Tables with Variables, Dependent Attributes, Functional Constraints and Functional Output The extended form of tables encoding rules allows for use of variables as attribute values. The set of initial, independent attributes, can be extended towards the use of new, complex attributes, functionally dependent on the initial ones. Their values can be calculated since the values of initial attributes are established.

Extended tables can also contain functional constraints specification. Such constraints constitute a powerful tool for expressing required relationships among attribute values.

For efficient decision making and in control applications it may be necessary to calculate the output on the base of current input values. This is now possible thanks to the functional output definition, applicable both for the *retract*, *assert* and decisional part of the rules.

A generic scheme of an extended rules is presented below; for simplicity the number of the rule, the context and the control part are committed.

4. Fuzzy Rules and Fuzzy Inference In practical applications it is often the case that matching rule preconditions against the current values of state variables must be performed with some tolerance. In other words, the boundaries such as threshold values are to certain degree imprecise. In such a case a useful approach accepted by practitioners consists in defining *fuzzy rules* with preconditions specification based on *fuzzy sets*.

The material below is based on the classical approach to fuzzy sets and fuzzy rule-based systems. For details see for example [11].

Rule	G	C	f(X,Y)	g(X,Y)	T
1	X	Y	$X > Y$	$X - Y \leq 1000$	$(X - Y) * 0.2$
2	X	Y	$X > Y$	$X - Y > 1000$	$200 + (X - Y - 1000) * 0.3$

Fig. 2. A simple table with functional output

Fig. 3. The basic form of an XAT

Prec	Dependent	Constraints	Retract	Assert	Decision
$A_1 \ldots A_n$	Z	F	B	C	H
$X_1 \ldots X_n$	$z(X_1, \ldots, X_n)$	$f(X_1, \ldots, X_n)$	$r(X_1, \ldots, X_n)$	$a(X_1, \ldots, X_n)$	$h(X_1, \ldots, X_n)$

4.1. The Case of Discrete Variables Consider a discrete set $Z = \{z_1, z_2, \ldots, z_k\}$. Let μ be a function of the form:

$$\mu\colon Z \rightarrow [0, 1].$$

The pair $\mathcal{Z} = (Z, \mu)$ is called a discrete fuzzy set. In case of finite sets it can be also written as a set of pairs (z_i/μ_i), where $\mu_i = \mu(z_i)$, $i = 1, 2, \ldots, k$. Hence $\mathcal{Z} = \{z_1/\mu_1, z_2/\mu_2, \ldots, z_k/\mu_k\}$. Note that the crisp set Z can be a set of numbers, but it can be a set of purely symbolic values as well.

Normally, a discrete fuzzy set contains several pairs of the form (z_i/μ_i). If $\mathcal{Z} = \{z/\mu\}$ then $\mathcal{Z}$ is called a *singleton*.

4.2. The Case of Continuous Variables In case of continuous values the fuzzy membership function can be defined to be any function with the range falling into the interval $[0, 1]$. In practice, the most typical functions used are triangular and trapezoidal ones as well as based on left or right half of the trapezoid.

Below a single, general function defining the fuzzy degree is introduced. The function covers several well-known cases of particular membership functions.

Definition 1 (The π function (fuzzy)) *Let Z be a convex interval of numeric values, e.g. $Z \subset \mathbb{R}$, and let α, β, γ and δ be numbers belonging to S and such that $\alpha \leq \beta \leq \gamma \leq \delta$. We define the $\pi^{\langle\alpha,\beta,\gamma,\delta\rangle}$ function as follows:*

$$\begin{aligned}
\pi^{\langle\alpha,\beta,\gamma,\delta\rangle}(\alpha) &= 0 \quad \text{for} \quad \alpha < \beta \\
\pi^{\langle\alpha,\beta,\gamma,\delta\rangle}(\alpha) &= 1 \quad \text{for} \quad \alpha = \beta \\
\pi^{\langle\alpha,\beta,\gamma,\delta\rangle}(\beta) &= 1 \\
\pi^{\langle\alpha,\beta,\gamma,\delta\rangle}(\gamma) &= 1 \\
\pi^{\langle\alpha,\beta,\gamma,\delta\rangle}(\delta) &= 1 \quad \text{for} \quad \delta = \gamma \\
\pi^{\langle\alpha,\beta,\gamma,\delta\rangle}(\delta) &= 0 \quad \text{for} \quad \delta > \gamma
\end{aligned}$$

and is constructed by piecewise linear interpolation among these points. The pair $\mathcal{Z} = (Z, \pi)$ is a fuzzy set.

We distinguish the following special cases of function π:

1. $\Delta^{\alpha,\beta,\delta} = \pi^{\langle\beta,\beta\rangle}$ *obtained for $\alpha < \beta = \gamma < \delta$; this is the so-called* triangle function*,*
2. $\Gamma = \pi^{\langle\alpha,\beta,\gamma\rangle}$ *obtained for $\alpha < \beta < \gamma = \delta$; this function is a fuzzy step-wise function,*
3. $Z = \pi^{\langle\beta,\gamma,\delta\rangle}$ *obtained for $\alpha = \beta < \gamma < \delta$.*

In majority of practical applications the trapezoidal function together with the functions based on it provides a satisfactory solution for defining fuzzy membership functions. Note that the shape of trapezoidal function is defined by four distinct parameters, the shape of triangular function by three, and the shape of the other two function just by two parameters.

4.3. Fuzzy Rule Preconditions Definition of rule precondition in the fuzzy case is analog to the crisp case. In the crisp case the basic atomic condition is of the form: $A \in t$, and the check results in a single logical value, i.e. *true* if the current value of attribute A belongs to set t and *false* otherwise. In the case of fuzzy preconditions one has to check if $A \in \mathcal{Z}$, and the result is the fuzzy membership coefficient belonging to interval $[0, 1]$. It is normally assumed that the check produces negative answer if $\mu(A) = 0$ and positive one if $\mu(A) > 0$.

Now consider a rule having j atomic preconditions incorporating fuzzy sets. Let μ_i denote the fuzzy coefficient obtained for the $i-th$ atomic condition. The total coefficient is obtained as

$$\mu = \mu_1 \circ \mu_2 \circ \ldots \circ \mu_j,$$

where $\circ$ is any t-norm operator (such as min or times). A rule is fired iff and only if $\mu > 0$. In typical fuzzy rule-based systems it is possible to fire several rules at the same time. If more than one rule is fired, the results of the rules are combined to form a single output.

4.4. Fuzzy Rule Output In case a single rule is fired, the current value of μ can be applied in the following ways:

- instead of a single crisp-case output value d a singleton (d, μ) is produced; such a pair can be interpreted as 'd with degree μ',
- μ can be used to shape the output fuzzy value (limit) as in the case of Mamdani inference rules,
- μ can be used a kind of *certainty degree* in case of logical output values.

In case when several rules are fired together, the value of μ is used to shape the fuzzy output functions (the Mamdani system) or as weighting coefficients (the Takagi-Sugeno system).

4.5. Learning Fuzzy Rules A fuzzy rule-based system can become a learning one. This is especially important when a number of training cases are available, but no expert knowledge covering them is available. In such a case a learning or adaptation algorithm can be used to build a set of rules.

In the case of fuzzy rule-based systems practical results can be obtained with relatively simple approach of Wang-Mendel algorithm.

5. Dynamic Rules and Time Delays Most of the rule-based systems operate according to a very simple principle: the *current* state of the input variables is analyzed and the rules having satisfied preconditions are fired. However, in certain more complex applications, e.g. in the control or signal analysis domain, the preconditions of the rules must incorporate not only current values of input signals, but the *past* values as well.

5.1. Motivation Consider a simple example of signal analysis. The input consists of a sequence of values $X(t_0 + \Delta t \times i) = X_i$, where t_0 is some initial instant of time and Δt is the basic time interval of signal measurement. For example, determining the current sign of the first derivative can be done with the following three rules represented within Table 3.

Table 3. Rules for determining the sign of the first derivative

X_{i-1}	X_i	$X_i - X_{i-1}$	Sign
-	-	> 0	+
-	-	< 0	-
-	-	= 0	0

In general, applications which detect, analyze and use information about changes of the input signals, such as first, second and higher derivatives or accumulated values, such as average for certain period of time, must access and use past, historical values of the signals. A classical example of such task is the prediction and control of dynamic systems.

Note that access to variable history is not limited to numerical signals. The following rules are used by car drivers to predict the next value of the traffic light signal:

Table 4. Rules for traffic lights prediction

Light-before	Light-now	Light-next
red	yellow	green
green	yellow	red
yellow	red	yellow
yellow	green	yellow

5.2. Temporal Formalism The simplest and perhaps most useful from practical point of view temporal formalism is one used in signal analysis and control theory. The time domain is divided into small and equal temporal intervals. The signals are measured only at indexed time instants. The intervals should be small enough to capture the expected changes and the dynamic behavior of the system.

Consider a specific attribute A_i. The value of this attribute at time instant k will be denoted as a pair (A_i, k), or simply as $A_i(k)$. Now, if the rule based system uses the current value as well as

m past values, the table representing such rules should be extended and can have the following form represented with Table 5.

Table 5. Rules with past values of attribute A_i

$A_i(k-m)$	$A_i(k-m+1)$	$\ldots$	$A_i(k)$	D
$d_i(k-m)$	$d_i(k-m+1)$	$\ldots$	$d_i(k)$	d

where $d_i(j)$ is the value of attribute A_i at time instant j.

A more general solution might require complex temporal knowledge representation formalism. In general, there are numerous temporal logics for specific purposes. They fall into two basic categories: ones based on the concept of *interval* of time, and ones based on the concept of *point* of time. They may also refer to *relative* or *absolute* time.

6. Inference Control Diagrams In order to assure satisfactory work of the rule interpreter, an efficient and correct control mechanism should be defined. In general, it is hard to say what does the correctness of the control mechanism mean; however, certain postulates can – at least – be put forward:

1. Determinism: the behavior of the system should be deterministic (at least unless randomly behaving components are introduced on purpose),
2. Continuity: the system should not fall into a deadlock trap, it can stop only when declared to do so,
3. Correctness: under any specific external conditions and memory state, the right, declared action should be executed.

The Control Specification Language should enable specification of the following control actions concerning inference:

- within a table:
  - what to do after some middle rule is fired,
  - what to do when some middle rule fails,
  - what to do after the last rule is fired,
  - what to do after the last rule fails;
- among the tables:
  - backtrack or no-backtrack mode,
  - conditional branching,
  - loops.

Generally speaking, after an operation (a rule is fired or it fails) there are the following potential actions:

- go to *next* (if exists) or *declared* table/rule,
- backtrack and check other possibilities,
- stop and exit.

References.

[1] Mordechai Ben-Ari. *Mathematical Logic for Computer Science*. Springer-Verlag, London, 2001.

[2] Michael A. Covington, Donald Nute, and André Vellino. *Prolog programming in depth*. Prentice-Hall, 1997.

[3] Adrain A. Hopgood. *Intelligent Systems for Engineers and Scientists*. CRC Press, Boca Raton London New York Washington, D.C., 2nd edition, 2001. ISBN 0849304563.

[4] Peter Jackson. *Introduction to Expert Systems*. Addison–Wesley, 3rd edition, 1999. ISBN 0-201-87686-8.

[5] Thomas Laffey and *et al.* Real-time knowledge-based systems. *AI Magazine*, Spring:27–45, 1988.

[6] Jay Liebowitz, editor. *The Handbook of Applied Expert Systems*. CRC Press, Boca Raton, 1998. ISBN 0-8493-3106-4.

[7] Antoni Ligęza. *Logical Foundations for Rule-Based Systems*. AGH University of Science and Technology Press, Kraków, 2005.

[8] Antoni Ligęza. *Logical Foundations for Rule-Based Systems*. Springer-Verlag, Berlin, Heidelberg, 2006.

[9] Antoni Ligęza and Pilar Fuster Parra. A granular attribute logic for rule-based systems management within extended tabular trees. In Robert Trappl, editor, *Cybernetic and Systems*, volume 2, pages 761–766. Austrian Society for Cybernetic Studies, 2006.

[10] Grzegorz J. Nalepa. *Meta-Level Approach to Integrated Process of Design and Implementation of Rule-Based Systems*. PhD thesis, AGH – University of Science and Technology, Institute of Automatics, Cracow, Poland, September 2004.

[11] Michael Negnevitsky. *Artificial Intelligence. A Guide to Intelligent Systems*. Addison-Wesley, Harlow, England; London; New York, 2002. ISBN 0-201-71159-1.

[12] I. S. Torsun. *Foundations of Intelligent Knowledge-Based Systems*. Academic Press, London, San Diego, New York, Boston, Sydney, Tokyo, Toronto, 1995.