TAL Rules versus ECA Rules: an Attempt for Comparison in the Credit Management Context

Maria Mach
University of Economics, Wroclaw, Poland

mach@manager.ae.wroc.pl

Abstract

Credits are the one of the most important functions in bank management, because, from one side, thanks to a good credit policy a bank can earn money, but from the other side, in the case of weak or wrong credit policy the bank can make substantial losses. Therefore in the field of credit policy management, intelligent information systems can be very helpful, as it is a complex and heterogeneous field, needing complex management and decision-making procedures.

There exist many technical solutions aimed at helping the decision-makers in this field, from "traditional" ones, as databases, to more sophisticated tools, as for example expert systems, the main aim of which is to perform the analysis of applications for a credit, thus helping to make proper credit decisions.

Credit management is closely related to time, in other words, the temporal aspect of credit management can be very clearly seen. Therefore while building intelligent systems in this area, it would be recommended to take this temporal aspect into account.

The article concentrates on the question of searching and choosing an intelligent computer tool which would fulfil the above mentioned requirements, the toll which would help to make necessary credit analyses, to make proper credit decisions, taking into account the temporal aspect of credit management. Two solutions are discussed: TAL language and active databases. Some exemplary credit management rules are encoded both in the TAL language and in the form of ECA rules. Both kinds of rules are analysed and discussed, as well as their advantages and disadvantages.

Keywords: Credit management, TAL rules, active databases, ECA rules, intelligent systems

Introduction

Credits constitute one of the most important functions in bank management process, a function which very often decides about bank's "to be or not to be". It is therefore clear that credit policy is such a field, in which computer tools are needed and useful. The attention must also be paid to the fact, that banking policy is a very complex and heterogeneous field, needing complex management procedures.

There are many concepts and ideas concerning credit management, arising from both computer systems *sensu stricto* as well as from intelligent systems. It is proposed, for example (see e.g. (Ziemba, 1992)) to use expert systems which perform the analysis of applications for a credit, thus helping to make proper credit decisions. Such systems are mainly aimed at minimising the percentage of so-called bad credits.

Credit management is closely related to time, in other words, the temporal aspect of credit management can be very clearly seen. Therefore while building intelligent systems in this area, it would be recommended to take this temporal aspect into account.

In (Mach, 2000) the Temporal Action Language (TAL) was analysed as a tool for performing effective credit management. While analysing the rules encoded in this language, one can notice immediately their similarity to the rules of so-called active databases, namely ECA rules. Therefore it is worth answering a question, whether searching for new tools (such as the TAL language) makes sense, if there exist well-known and good solutions (active databases). To answer the above question, the rules encoded in the TAL language will be analysed along with the same rules encoded in the form of ECA rules. The advantages and disadvantages of both solutions will also be discussed.

The rest of the paper is organised as follows: in section 2 the temporal aspect of credit management is shortly presented. Section 3 is devoted to main features and ideas of the TAL language, while the second part of this section is devoted to a short survey of active databases and ECA rules concepts. Section 4 contains examples of credit rules encoded in the TAL language and as ECA rules. The comparison of both solutions is done in section 5, along with the discussion concerning advantages and disadvantages of both types of rules. Section 6 contains conclusions and future research suggestions.

The Temporal Aspect of Credit Management

The temporal aspect of credit management can be seen already in the definition of credit. In (Bogacka-Kisiel, Karwowski, 1991) for example, the authors give the following definition:

"Credit – an economical relation, in which one party (the lender) transfers a predetermined value in money or commodity to the other party (the borrower), on condition of returning its equivalence with eventual remuneration (interest) in a predetermined period".

Of course the question of a term (or terms) does not have to be restricted only to the aspect of returning the credit, it can also concern the time-ordering of particular credit tranches (Krzyżkiewicz, 1996), (Gospodarowicz, Możaryn, 1998), (Jaworski et al., 1997). The above questions are set up in details in the credit agreement.

The temporal aspect of credit management can be also seen in the temporal criterion of dividing credits, according to which the credits are divided into three groups: short-term ones, intermediate ones and long-term ones.

Grace period – the delay in returning a part of a credit – is another example of temporal aspect in the credit field. It is worth mentioning here that grace period is an excellent example of so-called delayed action or of an action with delayed effect: in this case, a release of a credit constitutes an action, while "moving" the first return instalment in time constitutes the action's delayed effect.

Taking into account the above temporal aspects, it can be seen that temporal computer systems could be very useful in credit policy-making. The main question is not whether it is worth building such systems, but whether one should rely on already existing, well known solutions or better search for new tools. The article is an attempt for answering this question.

The TAL Language, Active Databases, and ECA Rules

The TAL Language: Main Features and Ideas[*]

The TAL (*Temporal Action Language*) is derived form Sandewall's PMON logic (Sandewall, 1994), (Doherty, 1994). It's main features as a language for describing temporal dependencies include: the notion of time independent from actions, the possibility of defining causal dependencies apart from actions' definitions, and the possibility of describing concurrent interactions.

The language consists of two levels (layers): the so-called surface language, which is used to describe narratives (for more information see (Karlsson et al., 1998), (Doherty et al., 1998)), and the so-called base language, namely the logic of events, which is an ordered 1st order predicate logic. Any correct narrative description, after being transformed into the description in the base language constitutes a finite set of 1st order wffs.

The surface language layer consists of:

- temporal expressions,
- value expressions,
- atomic expressions,
- narrative statements,
- additional macro-operators and abbreviations (e.g. the durational reassignment operator, the reassignment operator, the occlusion operator etc.).

The base language layer (the logic of events) contains, among others, temporal predicates: HOLDS, OCCURS, OBSERVE, DUR, PER and others (the exact definitions of the predicates can be found e.g. in (Doherty et al., 1998)).

The surface language does not have formal semantics, although it has a formal syntax. The whole formal inference process is conducted after "translating" the description in the surface language into the description in the base language.

It is also worth mentioning here, that the description (specification) of a scenario in the TAL language consists of:

- type description,
- action definitions and descriptions,
- domain constraints specification,
- temporal dependencies specification.

Active Databases and ECA Rules

The concepts concerning the problem of monitoring certain conditions and of triggering rules is not new. Chapters on this topic can be found in almost every book about database concepts. Usually the above topic is followed by a topic concerning active databases.

There are many definitions of active databases. In (Chakravarthy, 1992) such databases are described as the ones accepting a declarative specification of situation-action (event-condition-action) rules and man-

[*] Based on (Karlsson et al., 1998), (Doherty et al., 1998).

aging them. Silberschatz, Korth and Sudarshan (Silberschatz et al., 1997) define an active database system as the one carrying out actions in response to events. Similar definitions can be found elsewhere.

The three typical architectures of active databases are the following (Chakravarthy, 1992): application based, layered, integrated.

As it has already been said, active databases contain the so-called ECA rules, which serve to make the database reactive. The ECA rules, also called the active ones, specify when and what actions to carry out (Silberschatz et al., 1997).

An ECA rule can have different, although similar structure. The most general structure is of the following form (Silberschatz et al., 1997):

on *event* **if** *condition* **then** *action*

The above structure can also have the form (Pankowski, 1997):

on *event* **if** *condition* **do** *action*

In (Theodoulidis et al., 1992) we find the following structure of the ECA rule:

rule (Rule_name, Rule_priority), **event** (Event_expression), **condition** ([Conditions_list]), **action** ([Actions_list])

And in [CHHA97] the structure of an ECA rule used by Ariel system is given:

define rule *rule-name* [**in** *ruleset-name*] [**priority** *priority-val*] [**on** *event*] [**if** *condition*] **then** *action*

In the paper the most general form (ON event IF condition THEN action) will be used, with no respect to any specific language.

The events in response to which an action occurs, can be divided into primitive and complex ones, moreover, primitive events can be divided into database, time and external/abstract ones (Chakravarthy et al., 1992).

In the general structure of an active rule, an event is an expression specifying, under what circumstances the rule can be activated (Theodoulidis et al., 1992). Condition (or conditions) is the part of an active rule responsible for firing the rule. The rule is fired if and only if the condition is satisfied. Action is a database operation, external procedure call or the operation “abort”.

What do we need active rules for? There can be pointed out several purposes (see (Silberschatz et al., 1997), (Pankowski, 1997) for further details), as for example:

- alerting,
- checking integrity constraints,
- checking so-called business rules,
- reacting to changes in a database,
- modelling processes initiated in the system, etc.

In the context of credit management policy, the usefulness of active rules can be seen even intuitively. Taking into account the above presented features of these rules, it is easy to notice that an active database system would:

make credit repayment control easier: an active database could alert if a credit instalment has not been paid off;

automate certain actions which are to be taken in case of irregularities: e. g. If the credit is not paid off, a system could automatically generate a remark which is then sent to a client.

Examples of Credit Rules in the TAL Language and as ECA Rules

To implement solutions encoded in the TAL language, one can choose the VITAL tool (Kvarnström, Doherty, 1997), developed at Linköping University, Sweden. The tool is very easy to use and automatically translates scenarios from the surface language (TAL) into the base language, therefore the person constructing a scenario does not have to be an expert in temporal predicate logic.

Below a few examples of inference rules will be presented, namely the ones in which a temporal aspect of credit management is clearly visible. They will be followed by a description in the TAL language and by ECA rules. It must be pointed out here, that TAL rules presented below do not constitute complete scenarios, they constitute only the so-called temporal dependencies (see previous section). It must also be said that the ECA rules given for each example are not encoded in any specific language (i.e. SQL), they are presented only as a general idea.

Example 1

At the moment when a debit on the personal account takes place, the bank sends a remark to the account's owner (an example based on (Karlsson et al., 1998)).

forall t [[t] less (balance, i0) -> I([t]remark)]

where:

t	time-point. Its exact value depends on the granularity of time chosen. In the case of credit problems, it can be assumed that the granularity is one day,
less	ordering relation defined on the domain of integers,
balance	an user-defined variable of type *persistent*, which means that an inertia assumption about this variable is made. The inertia assumption means that variable's value does not change until an action is performed affecting this variable,
I	a macro-operator of the general form *I([r]* α*)*, meaning that the formula α is true in time-point *r*. The other legal forms of the macro-operator I are: *I((r, r']* α*)*, *I([r, r']* α*)* etc.,
remark	an user-defined variable with logical values (true or false).

The same rule in the active database could have the following form:

on balance<0 **if** debit_not_allowed=true **then** send_remark

where:

balance	a field in a record referring to specific borrower; therefore the action specified in the active rule responses to a database event, when the field takes on a value less than 0.
debit_not_allowed	a logical variable used to check whether the specific client is allowed to have debits on his/her account (sometimes banks allow debits for a short time and not for every client)
send_remark	an action consisting of sending a remark;

Example 2

If the borrower does not pay the credit off in the predefined time, then the amount which is still to be paid off, is moved to an overdue credit account (Krzyżkiewicz, 1996).

forall t [Ct ([t]not_paid_off) -> (Move(amount) & amount = = sumX)]

where:

Move an user-defined action, consisting of moving the amount that was not paid off on time from a credit account onto the overdue credit account. The definition of such an action could have the following form:

acs [t,t]Move(amount) -> I([t+1] (balplus(overdue_credit_account) = = amount & balplus(credit_account) = = -amount))

where *balplus* is an user-defined function changing the account balance (Karlsson et al., 1998);

Ct a macro-operator *changes to true* meaning, that the expression in parentheses is true;

not_paid_off a logical variable. It's default value is "false". The change into "true" occurs in case of not paying the credit off on time;

sumX the amount of credit not paid off. It could be read by the system from the database.

The same rule in the form of an active rule could have the following form:

on date()>D **if** not_paid_off(x)=true **then** move(x, credit_account, overdue_account)

where:

date() a function returning actual date. Therefore the expression date()>D, where *D* - the date on which the borrower is obliged to pay the credit off, is a time event;

not_paid_off a logical function with an argument *x* where *x* is the sum to be paid;

move an user-defined function with arguments *x*, *credit_account* and *overdue_account*, where *x* is defined as above, *credit_account* is the number of borrower's credit account and *overdue_account* is the number of borrower's overdue account to which the sum x is moved from the credit account.

Example 3

The case similar to the example 2, additionally the bank sends a remark to the client a week after the debit occurs. This is an example of an action with a delayed effect: the action is an occurrence of the debit, the effect – sending a remark.

forall t [Ct ([t]not_paid_off) -> (Move(amount) & amount = = sumX & I([t+7]remark))]

The above formula combines the solutions of examples 1 and 2. Again it has been assumed that the basic time unit is one day, therefore the remark is sent on time-point [t+7].

The active rule could have the form:

on date()>date_n **if** not_paid_off(x)=true **then** move(x, credit_account, overdue_account)

and send(date()+7, remark)

where:

send(date, a) a binary function with arguments *date* and *a*, where *date* is the exact date on which *a* is to be sent while *a* is a variable denoting what to send (a remark, a letter etc.)

Example 4

An example concerning grace period: the time of paying the first instalment off is delayed two months after the credit is released. Here we have the case of an action delayed in respect to some event: the event (which occurs on time-point *t*) is the release of the credit, while action – checking, whether the first instalment was paid off and if not, sending a remark. Note that in the dependency formula (*dep*) the two-month delay is expressed, as the payment of the first instalment is checked in time-point [t+60] with the assumption that the basic time unit is one day.

acs [r,r] Check ~> [r] chck = = false -> I([r] remark)

dep forall t [Ct[t+60] equal(balance, instalment)->I([t+60]chck = = true)]

where:

acs the general definition of action *Check*, stating, that if on any time-point the variable *chck* is false, which means the instalment was not paid off, the remark has to be sent (the variable *remark* is defined as in example 1);

r time-point;

~> an operator separating action symbol and action definition;

chck an user-defined variable; a boolean variable of type *durational*, that is a variable with a predefined default value. In this case the predefined variable value is "false", meaning that the instalment has not been paid off;

equal the ordering relation on the domain of integers;

balance a variable defined as in example 1.

The active rule concerning the above example could have the following form:

On date()=D **if** balance<>instalment **then** send_remark

where:

date()=D a time event: the present date equals *D*, where *D* is the date on which the borrower is supposed to pay the first instalment off. Of course in our example *D* equals the date on which the credit was released + 60 days. As we see, an action in the above active rule responses to a time event. Moreover, it should be noticed that expressing a certain delay (e.g. date+60) is not explicit, in contrast to TAL rules;

balance a field in a record concerning a particular borrower;

send_remark an action consisting of sending a remark to a borrower.

Comparison of TAL and ECA Rules in the Credit Management Context

At first sight, there is not much difference between the two types of rules. In both cases actions respond to events, and some conditions are checked. So which type of rule should be chosen in the aspect of credit management?

The ECA rules seem to be useful and well suited to credit management purposes because they are to be used in database management systems, therefore the implementation is supposed to be easy and not very expensive in terms of money and time. At the same time, the implementation of TAL rules is not equally easy, as it requires implementing the VITAL tool, writing complete scenarios (which is not an easy task) and building an interface between a database and the VITAL tool. The answer to the above question could therefore be: choose ECA rules. But is it really a good answer?

There can be immediately seen major advantages of TAL rules:

- temporal dependencies in the problem domain can be expressed explicitly, moreover, the actions in the system can depend on time dimension;
- the possibility of manipulating on delayed actions and on actions with delayed effects in an explicit manner. As it has been shown, this quality of TAL rules is particularly useful in the case of grace period, for example;
- the possibility of encoding concurrent actions (for example, the same client is the borrower of two different credits at the same time);
- the possibility of encoding different kinds of constraints.

The above features of the TAL rules do not however apply to ECA rules equally. It must be especially pointed out that situations in which delayed effects of actions occur are difficult to encode in the form of ECA rules. At the same time, TAL rules also have disadvantages, among which the most serious ones are: the lack of formal semantics (Doherty et al., 1998), which makes writing scenarios difficult; only one predefined variable type, namely logical one, which means, that other types needed for performing the reasoning are to be defined in an artificial manner, very often explicitly. Therefore it is impossible to generalise the scenarios, which would have to be written for each case (client) separately.

The main features of both types of rules are summarised in Table 1.

Feature	TAL rules	ECA rules
1. Possibility to use with existing database management systems	NO	YES
2. Possibility to express temporal dependencies **explicitly**	YES	NO
3. Possibility to deal with delayed actions and delayed effects of actions **explicitly**	YES	NOT ALWAYS
4. Possibility to encode concurrent actions **explicitly**	YES	NO
4. Easy to generalise	NO	YES
5. Prioritisation of rules	NO	YES

Table 1. Main features of TAL and ECA rules – a comparison.

In Table 1, two points need a short comment, namely points 2 and 3. In the first case, it should be noted that temporal dependencies can be expressed in the form of ECA rules, but not explicitly, therefore in this case TAL rules have an advantage over ECA rules. As for point 3, please refer to examples 3 and 4 in section 4. Time delay can be expressed explicitly in ECA rules but not in every case, sometimes the problem needs to be overcome in a way. Examples 3 and 4 in section 4 illustrate this very well.

Conclusions

The paper constitutes an attempt for comparison of two tools – namely TAL rules and ECA rules in a well-defined context, in this case the context of credit management policy. Taking into account the temporal aspect which can be clearly seen in this field, the need of using a kind of high-level tool dealing with temporal expressions seems to be obvious. At the same time there is no need to explain why an efficient credit management computer system should be able to react to some events. Therefore, as both tools analysed in the paper enable such kind of reaction, both of them can be taken into account while thinking of a tool for credit management.

Which solution should be then chosen? The answer is neither obvious nor easy.

As for now, a system based on ECA rules seems to have an advantage over a system based on TAL rules. It is due mainly to impossibility of constructing TAL scenarios in full versions. Therefore a future research concerning the topic should head towards constructing scenarios in full versions (not only rules), as well as towards generalising the scenarios so that they can be used repeatedly (constructing a new scenario for each new client – borrower – would make no sense). And while the above goal is reached, the systems based on TAL rules would probably be very useful, mainly because of the possibility of encoding temporal dependencies, delayed actions, delayed action effects and concurrent actions in an explicit manner.

Another interesting possibility worthy of attention consists of combining both solutions, or more precisely their advantages in one system. Such combined solution seems interesting and future research should also be aimed at checking whether such a mixture is possible to make and whether it is effective.

Acknowledgements

The author would like to thank Dr Mieczysław Owoc from Wroclaw University of Economics, for his comments, suggestions and help.

References

Bogacka-Kisiel E., Karwowski J., *Bankowość (wybrane zagadnienia). [Banking. Selected problems]*. Skrypty AE, Wydawnictwo AE Wrocław, 1991 (in Polish). To be obtained from the Wroclaw Unversity of Economics.

Chakravarthy S., *Architectures and Monitoring Techniques for Active Databases: An Evaluation*. Technical Report UF-CIS-TR-92-041, University of Florida, 1992.

Chakravarthy S., Hanson E., Su S. Y. W., *Active Data/Knowledge Base Research At The University of Florida*. Technical Report UF-CIS-TR-92-047, December 1992, University of Florida, 1992.

Doherty P., Gustafsson J., Karlsson L., Kvarnström J., *Temporal Action Logics (TAL): Language Specification and Tutorial*. Linköping Electronic Articles in Computer and Information Science Vol. 3 (1998): nr 015. http://www.ep.liu.se/ea/cis/1998/015. October 1, 1998.

Doherty P., *Reasoning about action and change using occlusion*, Proc. of 11th ECAI, Amsterdam, John Wiley and Sons, Ltd., 1994.

Gospodarowicz A., Możaryn H., *Identyfikacja i szacowanie ryzyka kredytowego [Identification and estimation of credit risk]*. Wydawnictwo AE, Wrocław 1998 (in Polish). To be obtained from the Wroclaw Unversity of Economics.

Jaworski W. L., Krzyżkiewicz Z., Kosiński B., *Banki: rynek, operacje, polityka [Banks: market, operations, policy]*. Poltext, Warszawa (Warsaw) 1997 (in Polish).

Karlsson L., Gustafsson J., Doherty P., *Delayed Effects of Actions*. In Proc. ECAI 98 – 13th European Conference on Artificial Intelligence, Brighton. John Wiley & Sons, Ltd., 1998, pp. 542-546.

Krzyżkiewicz Z., *Podręcznik do nauki bankowości [Banking tutorial]*. Biblioteka Menedżera i Bankowca, Zarządzanie i Finanse, Warszawa (Warsaw) 1996 (in Polish).

Kvarnström J., Doherty P., *VITAL research tool*, 1997. http://anton.ida.liu.se/vital/vital.html

Mach M., *Temporalny język akcji (TAL) jako narzędzie wspomagające zarządzanie kredytami [Temporal Action Language (TAL) as an additional tool for credit management]*, in: Gospodarowicz A. (Ed.), *Zastosowania rozwiązań informatycznych w bankowości*. Prace Naukowe AE nr 855, Wydawnictwo AE Wrocław, 2000 (in Polish). To be obtained from the Wroclaw Unversity of Economics.

Pankowski T., *Temporalne reguły ECA ze zdarzeniami złożonymi i ich zastosowanie do modelowania wiedzy w bazach danych [Temporal ECA rules with complex events for knowledge modeling in databases]*, in: Bubnicki Z., Grzech A., *Inżynieria wiedzy i systemy ekspertowe*, tom 1, Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 1997 (in Polish). To be obtained from the Wroclaw University of Technology.

Sandewall E., *Features and Fluents: A Systematic Approach to the Representation of Knowledge about Dynamical Systems*, Oxford University Press, 1994.

Silberschatz A., Korth H. F., Sudarshan S., *Database Systems Concepts*. The McGraw-Hill Companies, Inc., 1997.

Theodoulidis B., Loucopoulos P., Kopanas V., *A Rule-Oriented Formalism for Active Temporal Databases*, http://citeseer.nj.nec.com/43336.html. 1992.

Ziemba E., *Komputerowa implementacja modeli zarządzania ryzykiem kredytowym [Computer implementation of credit risk management models],* in: Gospodarowicz A. (Ed.), *Zastosowania rozwiązań informatycznych w bankowości*, Prace Naukowe AE nr 828, Wrocław 1999 (in Polish). To be obtained from the Wroclaw Unversity of Economics.

Biography

Maria A. Mach is a faculty member of the Wroclaw University of Economics. Her research interests include knowledge management, artificial intelligence systems in marketing and banking, temporal representation and temporal intelligent systems. She authored and co-authored 19 articles concerning the above mentioned and other areas.