MATRIX PROOF METHOD IN ANNOTATED PARACONSISTENT LOGIC

Celso A.A. KAESTNER

Departamento Acadêmico de Informática - Centro Federal de Educação Tecnológica do Paraná

Departamento de Informática - Pontíficia Universidade Católica do Paraná

CEFET/PR - DAINF - Av. Sete de Setembro, 3165 - 80.230-901 Curitiba - PR - BRAZIL

e-mail: kaestner@dainf.cefetpr.br

Décio KRAUSE

Departamento de Matemática - Universidade Federal do Paraná

Centro Politécnico - Caixa Postal 19.081 - 81.531-970 Curitiba - PR - BRAZIL

e-mail: dkrause@mat.ufpr.br

Abstract

The matrix connection method (MCM) is an alternative procedure for theorem proving than the usual resolution technique. We already have used the MCM for finding models in a real-time knowledge-based system generator. In this paper, we adapt the MCM to the particular case of some annotated propositional paraconsistent logics. Further developments related to these ideas are also outlined.

Keywords:
Paraconsistent Logics, Annotated Logics, Theorem Proving, Matrix Connection Method.

1 Introduction

Automatic proof methods are actually widely used in Artificial Intelligence applications. In the particular case of the Real-Time Knowledge-Based Systems domain, knowledge-based embedded programs have the convenience of reacting conveniently stimulus within imposed time restrictions to the environment. In [KAE 92a], [KAE 92b], [KAE 93a], [KAE 93b], [KAE 95], it was presented the RETIKS system, which is a real-time knowledge-based system generator based on a annotated paraconsistent propositional logic, which makes use of the synchronous approach for time modeling.

In order to attain its main purposes, RETIKS exhaustively calculates its outputs for each possible input. This compilation procedure is equivalent to that one of finding all models of the theory obtained from the rules that constitute the Knowledge-Base of the system [KAE 93b]. If it is desirable that each input determines an unique output, then the paraconsistent treatment should be used. The final execution structure obtained by the compilation procedure is a finite automata; this fact grants good performance and permits the verification of the desired time requirements.

In this paper, we describe the method used for finding models in the RETIKS system. Although theorem provers for similar paraconsistent logics based on *resolution* were already proposed, e.g. [SUB 87][BLA 88], the RETIKS system uses a method based on a variation of the Wallen and Bibel *matrix connection method* [BIB 82], [WAL 87], [WAL 90], [GOC 90], suited for these particular logics.

We begin by adapting the basic concepts of the matrix connection method in order to consider a peculiar paraconsistent case. We restrict ourselves to the propositional case, which is the base of the RETIKS compilation procedure. Meanwhile, the main ideas discussed here can be suitable adapted for first-order logics.

The paper is organized as follows: in the next section the fundamental concepts of the matrix connection method are given and the ground formalism is briefly introduced; in the section 3 the semantics for this formalism is outlined; in the section 4 the concepts of *path*, *connections*, and a characterization of validity are introduced; in the section 5 the case of theorem proving and the technics for finding models for the characterized systems are also considered. Finally, further research and applications are also suggested.

2 Matrix concepts and basic syntax

The terminology is based on [BIB 82] and [BLA 88]. Let A be a non-empty finite ordered set of propositional symbols. To each element $a \in A$, a non-empty finite lattice $\mathcal{T}_a$ is associated. The elements of $\mathcal{T}_a$ will be named *annotated constants* and denoted by μ, ν.

Definition 2.1 A (*ground*) *literal* is a triple (a, μ, p) where $a \in A$, $\mu \in \mathcal{T}_a$ and $p \in \{0, 1\}$. p is the *polarity* of the literal. Literals will be denoted by K, L, M. We also use ${}^qL$, $q \in \{0, 1\}$ to denote the literal $(a, \mu, (p + q) \bmod 2)$.

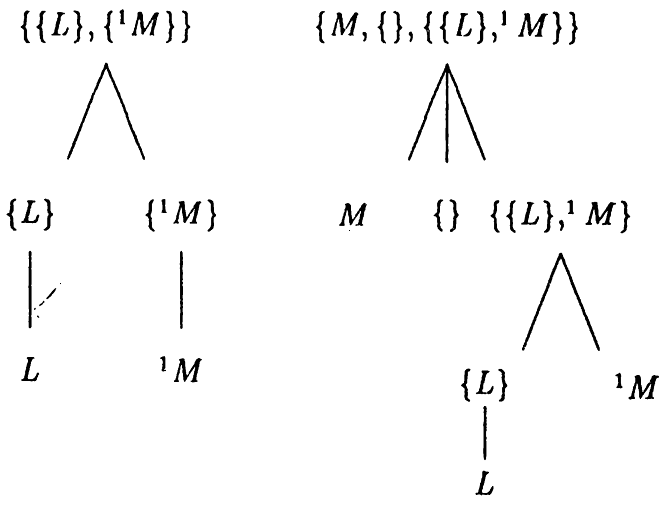

Figure 1: Tree representation of matrices

Let R be an alphabet of *occurrences* or *positions*. The elements of R are denoted by r.

Definition 2.2 By induction, we define the concepts of (*propositional*) *matrices* over (A,R), denoted by D,E,F, so as their *size* $\sigma(F)$, their *positions* $\Omega(F) \subset R$ and their *depth* $\delta(r)$ of r in F for any $r \in \Omega(F)$:

- For any literal L and for any $r \in R$, the pair $(L,r) = L^r$ is a matrix with $\sigma(L^r) = 0$, $\Omega(L^r) = \{r\}$ and $\delta(L^r) = 0$.
- If $F_1, \ldots F_n$, $n \geq 0$ are matrices such that $\Omega(F_i) \cap \Omega(F_j) = \emptyset$ for $i \neq j$ and $1 \leq i,j \leq n$, then the set $F = \{F_1, \ldots F_n\}$ is a matrix where:
  - $\sigma(\emptyset) = 0$ for $n = 0$ and $\sigma(F) = 1 + \sum_{i=1}^{n} \sigma(F_i)$ for $n > 0$;
  - $\Omega(F) = \Omega(F_1) \cup \ldots \cup \Omega(F_n)$;
  - $\delta(r) = m + 1$ for any $r \in \Omega(F_i)$, $1 \leq i \leq n$, where m is the depth of r in F_i.

According to this definition, the atomic parts of the matrices are ground literals, and in general a matrix is a nested set of occurrences of literals.

Example 2.1 Let us consider $A = (a,b,c,d)$ with $\mathcal{T}_a = \mathcal{T}_b = \mathcal{T}_c = \mathcal{T}_d = 2^{\{0,1\}}$ (the Boolean lattice of the power set of $\{0,1\}$ ordered by inclusion) and $R = \{0,1,2,3\}$ be an alphabet of positions. Then $L = (a,\{0\},0)$ and $M = (c,\{0,1\},1)$ are ground literals, while $\{\{L\},\{^1M\}\}$ and $\{M,\{\},\{\{L\},^1M\}\}$ are matrices over (A,R).

A matrix can also be viewed as a *tree*, where some leaves are associated to literals. Figure 1 presents the trees corresponding to the matrices of above example.

Definition 2.3 Let F be a matrix and $l, m \in \{0,1\}$. The set of *formulas* $\tilde{F}$ represented by F with respect to (l,m) is inductively defined as follows:

- if F is a literal $F = L^r$ and $l = 0$ then $\tilde{F} = L$;
- if F is a literal $F = L^r$ and $l = 1$ then $\tilde{F} = ^1L$;
- if $F = \{F_1, \ldots, F_n\}, n \geq 0$ and if $m = 1$, then $\tilde{F} = \wedge(\tilde{F}_1, \ldots, \tilde{F}_n)$, where the $\tilde{F}_i$ are formulas represented by F_i with respect to $(l, 0)$, $i = 1, \ldots n$;
- if $F = \{F_1, \ldots, F_n\}$, $n \geq 0$ and if $m = 0$ then $\tilde{F} = \vee(\tilde{F}_1, \ldots, \tilde{F}_n)$, where $\tilde{F}_i$ are formulas represented by F_i with respect to $(l, 1)$, $i = 1, \ldots n$.

Definition 2.4 A formula $\tilde{F}$ is *positively represented* by a matrix F if it is represented by F with respect to $l = m = 0$; $\tilde{F}$ is *negatively represented* if it is represented by F with respect to $l = m = 1$. A *propositional formula* is any formula represented by some matrix. Formulas are also denoted by D, E, F.

In order to adequate our notation to the usual one, we will introduce the following notations:

- if $n = 0$ $\wedge(F_1, \ldots, F_n) = \wedge()$ is abbreviated by $\mathbf{T}$, and $\vee()$ by $\mathbf{F}$;
- if $n = 1$ both $\wedge(F)$ and $\vee(F)$ are abbreviated by F;
- if $n \geq 2$ $\wedge(F_1, \ldots, F_n)$ is a *conjunction* and $\vee(F_1, \ldots, F_n)$ is a *disjunction*;
- for any literal L, the formula $\neg^k L$ is called a *hyper-literal*; if L is a hyper-literal, then $\neg^k L = (a, \neg^k(\mu), p)$, where $\neg : \mathcal{T}_a \longrightarrow \mathcal{T}_a$ denotes some fixed function (that gives the meaning of the negation), and k is a multiplicity factor (a natural number);
- if F is a formula which is not a hyper-literal, then $\neg F$ is defined by:
  - if $F = \wedge(F_1, \ldots, F_n)$, $n \geq 0$ then $\neg F = \vee(\neg F_1, \ldots, \neg F_n)$;
  - if $F = \vee(F_1, \ldots, F_n)$, $n \geq 0$ then $\neg F = \wedge(\neg F_1, \ldots, \neg F_n)$.
- any formula $\neg F \vee G$ may be written as $F \rightarrow G$;
- any formula $(F \rightarrow G) \wedge (G \rightarrow F)$ may be written as $F \leftrightarrow G$;
- any formula $F \rightarrow ((F \rightarrow F) \wedge \neg(F \rightarrow F))$ may be written as $\sim F$ and it is called the *strong negation* of F;
- parentheses are eliminated in the obvious way.

According to the above conventions, every well-formed formula (defined in the standard way) determines an unique matrix; notwithstanding, a matrix may represent more than one formula.

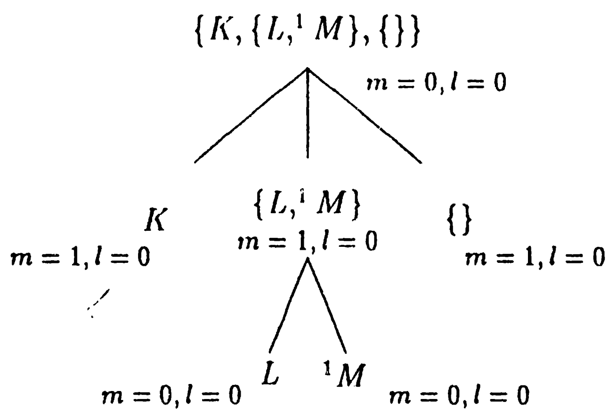

Figure 2: Positive representation of $F = \{K, \{L, M\}, \{\}\}$

Example 2.2 Let $F = \{K, \{L, M\}, \{\}\}$ be a matrix. The tree in figure 2 is the positive representation of F.

Example 2.3 Let be $\tilde{F} = K \rightarrow (L \vee {}^1M)$; F is an abbreviation of $\neg K \vee (L \vee {}^1M)$. This formula is represented by the matrix $\{\neg K, L, {}^1M\}$. Obviously the formula $\neg K \vee L \vee {}^1M$ has the same matrix.

Results presented in [BIB 82], chapter 2 remain applicable here, such as for instance:

- If a formula $\tilde{F}$ is positively represented by a matrix F then $\sim \tilde{F}$ is negatively represented by F;
- If two formulas $\tilde{F}_1$ and $\tilde{F}_2$ are positively represented by the same matrix F, then $\tilde{F}_1$ and $\tilde{F}_2$ are logically equivalents in the sense of the annotated logics [BLA 88].

These results justify the use of matrices instead of formulas.

The following example justify the name *matrix* employed firstly by Bibel [BIB 82] and used also here.

Example 2.4 Let $\tilde{F}$ be the formula:

$$(K \wedge {}^1L \rightarrow {}^1N) \wedge M \wedge \neg L \rightarrow ({}^1N \wedge {}^1K)$$

where K, L, M and N are literals; if we put $\tilde{F}$ in the disjunctive normal form (as usual), we will obtain:

$$(K \wedge {}^1L \wedge \neg^1N) \vee \neg M \vee \neg^2L \vee ({}^1N \wedge {}^1K)$$

This formula may be presented in a bidimensional arrangement, where the literals placed in a fixed column are connected by "$\wedge$", and the columns are connected by "$\vee$", as follows:

$$F = \begin{bmatrix} K & & & {}^{1}N \\ {}^{1}L & \neg M & \neg^{2}L & \\ \neg^{1}N & & & {}^{1}K \end{bmatrix}$$

3 Semantics

As usual, we admit that the truth of a certain knowledge depends on the truth values of its atomic constituents.

Definition 3.1 An *interpretation* $\mathcal{M}$ for our formalism is a function which associates an element of the lattice to every propositional symbol. By denoting $\mathcal{M}(a) = \mu_a$, we may write $(a, b, c, \ldots) \mapsto (\mu_a, \mu_b, \mu_c \ldots)$.

Let us consider the two "special" matrices introduced earlier:
$\vee() = \mathbf{F} = \emptyset = \{\}$ and $\wedge() = \mathbf{T} = \{\emptyset\} = \{\{\}\}$.

Now we will define the "truth value" $\mathcal{M}(F)$ of a matrix F as follows:

- if F is a literal (a, μ, p), then $\mathcal{M}(F) = \mathbf{T} = \{\emptyset\}$ iff $\mathcal{M}(a) \geq \mu$ when $p = 0$ and $\mathcal{M}(a) \not\geq \mu$ when $p = 1$. Otherwise $\mathcal{M}(F) = \mathbf{F} = \emptyset$.
- if F is a matrix $F = \{F_1, \ldots, F_n\}$, $n \geq 0$, then $\mathcal{M}(F) = \bigcup_{k=1}^{n} \mathcal{M}(F_k)$ when $m = 0$ and $\mathcal{M}(F) = \bigcap_{k=1}^{n} \mathcal{M}(F_k)$ when $m = 1$.

We will write $\mathcal{M}$ **sat** $\tilde{F}$ (and also $\mathcal{M}$ **sat** F) iff $\mathcal{M}(F) = \mathbf{T}$ for a matrix F which represents $\tilde{F}$.

Definition 3.2 A matrix F is *valid* iff $\mathcal{M}(F) = \mathbf{T}$ for every possible interpretation. It is called *contradictory* iff $\mathcal{M}(F) = \mathbf{F}$ for every interpretation.

We can note that if F is valid then $\sim F$ is contradictory, and the converse is also true.

Other semantical concepts can be introduced in such a way so that the standard semantical results can be obtained, but we will not present such details here.

4 Paths, Connections, and a Characterization of Validity

Definition 4.1 A *path* through a matrix F is a set of occurrences of literals, defined as follows:

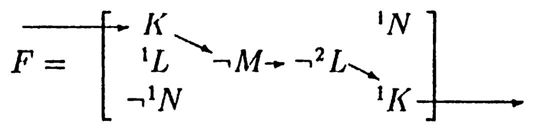

Figure 3: The path $\{K, \neg M, \neg^2 L, {}^1K\}$ through the matrix F

- if $F = \emptyset$ then the only path through F is $\emptyset$;
- if $F = L^r$ then the only path through F is the set $\{L^r\}$;
- if $F = \{F_1, \ldots, F_m, F_{m+1}, \ldots, F_{m+n}\}$, $m, n \geq 0$, $m + n \geq 1$ for m literals $F_1, \ldots, F_m$ and for n matrices which are not literals $F_{m+1}, \ldots, F_{m+n}$, then for any matrix $E_i \in F_{m+i}$ and for any path p_i through $E_i, 1 \leq i \leq n$, the set $\bigcup_{j=1}^{m} \{F_j\} \cup \bigcup_{i=1}^{n} p_i$ is a path through F.

Example 4.1 Let F be the matrix in example 2.4; a path through F is a crossing from left to right, constrained to pass by the literals (to be interpreted as "gates") as shown in figure 3.

Definition 4.2 We call *complementary literals* a pair of literals $L = (a, \mu, p)$, $M = (a, \nu, q)$ such that:

- $(p + q)$ **mod** $2 = 1$, and
- $\sqcup(\mu) \cup \sqcap(\nu) = \mathcal{T}_a$, where we denote $\sqcup(\mu) = \{\tau \in \mathcal{T}_a : \mu \leq \tau\}$ and $\sqcap(\tau) = \mathcal{T}_a \setminus \sqcup(\tau)$.

Definition 4.3 Paths which have complementary literals as elements are called *connections.*

Example 4.2 If $\mathcal{T}_a = 2^{\{0,1\}}$ then $(a, \{1\}, 0)$ and $(a, \{0\}, 1)$ are complementary literals.

Example 4.3 Let F be the matrix in the example 2.4;
the paths are $\{K, \neg M, \neg^2 L, {}^1N\}$, $\{K, \neg M, \neg^2 L, {}^1K\}$, $\{{}^1L, \neg M, \neg^2 L, {}^1N\}$, $\{{}^1L, \neg M, \neg^2 L, {}^1K\}$, $\{\neg^1 N, \neg M, \neg^2 L, {}^1N\}$, and $\{\neg^1 N, \neg M, \neg^2 L, {}^1K\}$.

The path $\{K, \neg M, \neg^2 L, {}^1K\}$ is obviously a connection, since K and ${}^1K$ are complementary literals. In order to others paths be connections, it is necessary to analyze their compounding literals, the negation function definition, and so on.

Proposition 4.1 (Soundness and Completeness): A matrix F is valid iff every path through F is a connection.

Proof: Adapted from [BIB 82, pp. 30-31], by using induction over the size of σ in the matrix F.

5 Theorem proving technics and models

In order to adapt the usual proof theorem procedures to our case, we should consider the following situation. Suppose that we have a set of formulas

$\Gamma = \{F_1, \ldots, F_n\}$ and a query G. Then, in order to investigate if G is a semantical consequence of the set Γ, we should verify if the matrix provided by $(\bigwedge_{i=1}^{n} F_i) \vee \sim G$ is contradictory.

The paraconsistent case is included in this procedure due to our definition of complementary literals. In fact, in this case the existence of a literal

$L = (a, \mu, p)$ and its 'negation' $\neg L = (a, \neg(\mu), p)$ is not a sufficient condition to assure complementary literals in the path. This exemplifies perfectly well the underlying ideas of the general paraconsistence program [COS 74]. We note that in order to obtain a proof of the query, it is necessary that all paths of the matrix $\sim (\bigwedge_{i=1}^{n} F_i) \vee \sim G)$ have connections, which imply the existence of complementary literals in every path.

If there are no complementary literals in the paths, then the set of paths represents the set of models of $\Gamma \cup \{\sim G\}$ as in usual tableaux semantic method; this was the methodology used in [KAE 93b].

6 Further Developments

The methodology presented in this paper were implemented in an prototype version of the system written in CommomLisp [KAE 93b]. It is interesting to note that this method is adequate also for the paraconsistent treatment, so that it provides an alternative way for theorem proving than the classical resolution procedures presented in the papers described in the Introduction.

As mentioned in [BIB 82, p. 45ff] and repeated by [WAL 87], [WAL 90], [GOC 90], we guess that the method presented here is more suitable for finding models as required by the RETIKS system. In fact, as the mentioned authors sustain, the connection method seems to be algorithmicaly more efficient than resolution in most cases.

It would be also interesting to ask for the algebra of connections, which apparently might differ from the classical case, as presented in [BIB 82].

The first-order case can be obtained without difficulty by adapting the procedure sketched in this paper.

One could also to investigate the possibility of extending the method presented here to the so called *multideductive logics* [COS 95]. In short, multideductive logics are defined so that several deduction symbols, say $\vdash_1$, $\vdash_2$, ... can be introduced in such a way so that the notions of $\vdash_i$-deduction ($i = 1, 2, \ldots$) are defined as usual by means of the stated $\vdash_i$-axioms. Then, if Γ is a set of formulas of the language and if F is a particular formula, the concept of $\Gamma \vdash_i F$ as well as the concept of an i-theorem are well defined. Since the set of i-theorems and the set of j-theorems ($i \neq j$) may contain contradictory formulas, the general underlying logic must be a paraconsistent one, as shown in just mentioned paper.

Then, if it is done a query G and a set of formulas Γ, one could ask if the query is a consequence of Γ by one of the concepts $\vdash_i$. In fact, in a most general idealized case, the 'proof' of the query might require more that one deductive notion. To investigate the applicability of the connection method in this situation could be useful in several cases, such as for instance when a certain knowledge may depends on different canons of inference.

In the particular case of Artificial Intelligence applications, the above discussion appears both in the context of knowledge-based systems obtained from several experts and in the context of cooperative multiagent systems. In the first case, the obtained date-base may be inconsistent in several ways, and the matrix connection approach could be done in a rather different way than that one provided by [COS 89], [COS 90]. In the context of cooperative multiagent systems, the use of multideductive logics seems to be more evident, since to each one of the agents it could be associated a different $\vdash_i$ notion.

References

[BIB 82] W. Bibel. *Automated Theorem Proving.* Friedr. Vieweg & Sohn, Braunschweig, Wiesbaden, 1982.

[BLA 88] H.A. Blair; V.S. Subrahmanian. 'Paraconsistent Logic Programming', *7 th. Int. Conf. on Foundations of Software Tech. & Theoret. Computer Science*, 1988.

[COS 74] N.C.A. da Costa. 'On the Theory of Inconsistent Formal Systems', *Notre Dame Journal of Formal Logic*, **15**, 497-510, 1974.

[COS 89] N.C.A. da Costa; V.S. Subrahmanian. 'Paraconsistent Logics as a Formalism for Reasoning About Inconsistent Knowledge Bases', *Artificial Intelligence in Medicine*, **1**, 167-174, 1989.

[COS 90] N.C.A. da Costa, et al.. 'Automatic theorem proving in paraconsistent logics: theory and implementation', *Proceendings of the 10th International Conference on Automated Deduction*, 72-86, 1990.

[COS 91] N.C.A. da Costa; V.S. Subrahmanian; C. Vago. 'The Paraconsistent Logics *PT*', *Zeitsch. F. Math. Logik und Grundlagen der Math.*, 139-148, 1991.

[COS 95] N. C. A. da Costa et al.. 'Paraconsistent logic: an application to physics', preprint.

[GOC 90] P. Gochet; P. Gribomont. *Logic 1: Methods for Fundamental Informatics* (in french). Hermes, Paris, 1990.

[KAE 92a] C.A.A. Kaestner. 'Real-Time AI Systems: the Synchronous Approach Contribuition to Rule-Based Systems' (in French), *Research Report LAAS* **92221**, Toulouse, France, 1992.

[KAE 92b] C.A.A. Kaestner. 'Artificial Intelligence Real-Time Systems: Synchronous Approach for Rule-Based Systems' (in Portuguese), *Proceedings of the IX Brasilian Symposium of AI*, 1992.

[KAE 93a] C.A.A. Kaestner. 'Proposing a Rule-Based System for Real-Time Applications' (in Portuguese), *Proceedings of the X Brasilian Symposium of AI*, 1993.

[KAE 93b] C.A.A. Kaestner. *A Proposition of a Knowledge-Based System for Real-Time Applications using the Synchronous Approach* (in Portuguese), Ph.D. Thesis, Federal University of Santa Catarina, 1993.

[KAE 95] C.A.A. Kaestner; J.M. Farines. 'A Synchronous Real-Time Knowledge-Based System', *Proceedings of the 7th Euromicro Real-Time System Symposium*, Odense, Denmark, 1995.

[SUB 87] V.S. Subrahmanian. 'On the Semantics of Quantitative Logic Programs', *Proccedings of the 4 th IEEE Symposium on Logic Programming*, Computer Society Press, Washington DC, 1987.

[WAL 87] L.A. Wallen. 'Matrix Proof Methods for Modal Logics', *Proceedings of the International Joint Conference on Artificial Intelligence*, Milan, 1987.

[WAL 90] L.A. Wallen. *Automated Proof Search in Non-Classical Logics*, The MIT Press, 1990.

[THA 89] A. Thayse et all. *Logic Approach of AI - 2. From Modal Logic to Data-Base Logic* (in French), Dunod-Informatique, Paris, 1989.