A Framework for Connection Calculi

H. Mantel[1] and E. Sandner[2]

[1] Deutsches Forschungszentrum für Künstliche Intelligenz GmbH
Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany
mantel@dfki.uni-sb.de

[2] Fachgebiet Intellektik, Fachbereich Informatik, Technische Hochschule Darmstadt
Alexanderstr. 10, 64283 Darmstadt, Germany
sandner@intellektik.informatik.th-darmstadt.de

Abstract. What is a matrix characterization? We give a formal answer to this question by the definition of matrix systems. They provide a framework in which all major existing matrix characterizations can be presented uniformly and also support the development of new ones. We describe how to develop a matrix characterization from a sequent calculus. Once a matrix characterization exists, a close relation to possible world semantics supports the development of such semantics for a logic.

1 Introduction

In the computer science community interest in non–classical logics has tremendously grown during the last years [9]. These logics arise from classical logic by adding new operators or by removal of structural rules in an appropriate sequent calculus. Structural rules have an important impact on the expressiveness and complexity of a logic. Dropping the contraction rule results in direct logic [3, 14] which is decidable. On the other hand, a controlled application of contraction and weakening (i.e. contraction and weakening is only applicable to a designated subclass of formulas) results in linear logic [12] which is undecidable already for the propositional case. From a practical point of view these logics are well suited for resource sensitive tasks like planning. They yield a much more natural representation of knowledge than classical logic in such application domains.

Hand in hand with this development a need for appropriate automated proof methods for these logics becomes apparent. *Matrix characterizations* of logical validity have already been successfully applied to non–classical logics. A matrix characterization defines a notion of *complementarity* of matrices. That validity of formulas and complementarity of respective matrices correspond must be ensured by a *characterization theorem.* A matrix characterization of a logic yields a representation of the search space which avoids many redundancies inherent to methods based on sequent–style or tableau proofs. Starting with Bibel's [4, 6] connection method for classical logic matrix characterizations have later been extended to many non–classical logics by Wallen [20]. Recently, matrix characterizations for fragments of linear-logic have been developed [11, 16, 15].

Different approaches have been undertaken to represent matrix characterizations in a uniform way [20, 7]. They allow to share results among different characterizations, e.g. based on Wallen's style of formulation a uniform proof method [17] and a uniform procedure for transforming matrix into sequent–style proofs [19] could be developed. However, these approaches do not exactly give an answer to the question what a matrix-characterization is.

It is our aim to fill this gap by an abstract definition of matrix characterizations. We identify a language of matrices, a notion of complementarity and translation functions as major components of existing characterizations. Based on these components *matrix systems* are defined as a framework. We see mainly two benefits from such a unifying approach. First, it is of practical interest to have a framework in which different characterizations can be presented uniformly and then be compared. A uniform method to prove characterization theorems appears to be achievable. Assuming further progress, we regard this work as foundation of a building kit for matrix characterizations. Second, besides practical benefits it is interesting by itself to extract the common concepts underlying the different approaches and to study the interrelationships between matrix systems and logics in general and in particular with sequent calculi. In so far, we regard a matrix-system as a mathematical object of its own interest.

In section 2 we present Wallen's matrix-characterization of the first-order modal-logic S4 as an example. In the subsequent section we identify the main concepts in that characterization as a motivation for the definition of matrix systems. We investigate the relation of sequent calculi to matrix systems and sketch how matrix systems naturally induce Kripke-like semantics. We conclude with a comparison to other approaches and an outline of future work.

2 A Matrix Characterization of Modal Logics

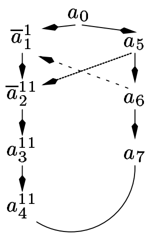
In this section we present Wallen's [20] matrix characterization for the first-order modal logic S4 with cumulative domains (short: S4). The language of S4 is an extension of the classical first-order language with unary modalities $\Box$ (necessity) and $\Diamond$ (possibility). The meaning of the connectives, modalities and quantifiers can be given by possible world semantics or by a cut free sequent calculus [20]. Due to the symmetries in this calculus we present a one-sided version which presumes that formulas are in negation-normal form. The resulting calculus is similar to one for classical logic with two extra rules for modalities:

$$\frac{\Rightarrow \Gamma, A}{\Rightarrow \Gamma, \Diamond A}\;\Diamond \qquad\qquad \frac{\Rightarrow \Gamma^{\Diamond}, A}{\Rightarrow \Gamma, \Box A}\;\Box \qquad \Gamma^{\Diamond}=\{F\in\Gamma \,|\, \exists F'{:}F=\Diamond F'\}$$

Performing proof search based on this calculus goes hand in hand with several types of redundancies. Due to the sub-formula property it suffices to maintain which part of the endsequent occurs how often at which place instead of copying these parts around in a sequent proof, i.e. avoiding *notational* redundancies. It is also preferable to *look ahead* when applying sequent rules, i.e. to avoid detours in the proof and therefore *irrelevance*. Another problem is the involved treatment of so called *in-permutabilities*, i.e. constraints on the order of rule applications. These three types of redundancies are more or less present in every sequent calculus. We describe Wallen's way out of this problem using $\Diamond\exists x.\neg p(x) \vee \forall y.\Box p(y)$ as an example. The reader familiar with [20] should be warned: our presentation is different from the one given there in order to make definitions in subsequent sections more comprehensible.

To represent matrix proofs a *tree ordering* $\ll$ based on the formula tree of a formula A is used. To each sub-formula B of A a *polarity* $k \in \{0,1\}$ is assigned, which depends on the number of implicit and explicit negations before B. The polarity is 0 (1) for an equal (odd) number of negations. We call a formula/polarity pair $\langle B, k\rangle$ a *signed formula*. Each $\langle B, k\rangle$ has a type (α, β, γ, δ, ν, or π) determined according to the tableau scheme [20]. The tree ordering $\ll$ can be represented by *positions*, i.e. objects that uniquely address a sub-formula in the formula tree. If a is a position designating a signed formula $\langle B, k\rangle$ then $lab(a) = B$ is its *label*, $pol(a) = k$ its polarity and $Ptype(a)$ its type. At a γ- or δ–position a the actual variable in $lab(a)$ is replaced in the successor formula with the corresponding γ– or δ–position. Formulas of type γ or ν, so called *generative formulas*, may be needed more then once in a proof. Therefore, we assign a *multiplicity* μ to all positions in $\ll$, where $\mu(a) \geq 1$ if $Ptype(a) \in \{\gamma, \nu\}$ and otherwise $\mu(a) = 1$. In $\ll$ each *generative position* a is extended by inserting $\mu(a)$ copies of the corresponding subtree. To discriminate the new positions a string reflecting the number of copies of the actual and all proceeding generative formulas is attached. The resulting new formula is denoted by A^{μ}. The sets of γ (Γ), ν ($\mathcal{V}$) are called *variable* and δ (Δ), π (Π) *constant* positions. We use *prefixes*, i.e. strings of positions to integrate the modal in-permutabilities into the characterization. Each position a in $\ll$ has an associated prefix $pre(a)$, which is constructed by collecting all positions $\hat{a} \in \mathcal{V} \cup \Pi$ when going from the root of $\ll$ to a.

The tree ordering $\ll$ and the corresponding signed (sub-)formulas of our example are shown below. We choose $\mu(\overline{a}) = 1$ for all variable positions $\overline{a}$ (marked with an overbar). To define paths we start at the root of $\ll$ and successively



a	$Ptype(a)$	$\langle lab(a), pol(a)\rangle$	$pre(a)$
a_0	α	$\langle \Diamond\exists x.\neg p(x) \vee \forall y.\Box p(y), 0\rangle$	a_0
$\bar{a}_1^1$	ν	$\langle \Diamond\exists x.\neg p(x), 0\rangle$	$a_0\bar{a}_1^1$
$\bar{a}_2^{11}$	γ	$\langle \exists x.\neg p(x), 0\rangle$	$a_0\bar{a}_1^1$
a_3^{11}	α	$\langle \neg p(\bar{a}_2^{11}), 0\rangle$	$a_0\bar{a}_1^1$
a_4^{11}	-	$\langle p(\bar{a}_2^{11}), 1\rangle$	$a_0\bar{a}_1^1$
a_5	δ	$\langle \forall y.\Box p(y), 0\rangle$	a_0
a_6	π	$\langle \Box p(a_5), 0\rangle$	a_0a_6
a_7	-	$\langle p(a_5), 0\rangle$	a_0a_6

$\sigma_P(\bar{a}_1^1) = a_6$
$\sqsubset_P = \{(a_6, \bar{a}_1^1)\}$
$\sigma_Q(\bar{a}_2^{11}) = a_5$
$\sqsubset_Q = \{(a_5, \bar{a}_2^{11})\}$
$\sigma_P^{\#}(pre(a_5)) = a_0$
$\preceq a_0a_6 = \sigma_P^{\#}(pre(\bar{a}_2^{11}))$

replace positions with their successors. When reaching a β-position (one where the sequent proof would branch) we split into two paths, each containing one successor of β. Paths which contain only non-reducible positions, i.e. leaves in the tree, are called *atomic*. An important concept is the *connection*, which is a two-element set of atomic positions with similar labels but different polarities. In our example, we have just one path $\{a_4^{11}, a_7\}$ and one connection $\{a_4^{11}, a_7\}$.

The existence of a sequent proof is guaranteed by an *admissible combined substitution* $\sigma := \langle\sigma_P, \sigma_Q\rangle$, consisting of a prefix and a quantorial substitution. A prefix substitution $\sigma_P : \mathcal{V} \mapsto T_P^*$ ($T_P = \Pi \cup \mathcal{V}$) induces a relation $\sqsubset_P$ on $T_P \times T_P$ by *if* $\sigma_P(\bar{b}_1) = b_2$ *and* $b_2 \notin \mathcal{V}$ *then for all* $b_3 \varepsilon b_2, b_3 \sqsubset_P \bar{b}_1$. The homomorphic extension to T_P^* is $\sigma_P^{\#}$. Let $\mathcal{T}$ by a set of terms created over $T_Q \cup \Gamma \cup \Delta$. A quantorial substitution $\sigma_Q : \Gamma \mapsto \mathcal{T}$ induces a relation $\sqsubset_P$ on $\Delta \times \Gamma$

by *if* $\sigma_Q(\bar{b}_1) = t$ *then for all* $b_2 \in \Delta$ *which are sub-terms of* t *holds* $b_2 \sqsubset_P \bar{b}_1$. The relations $\sqsubset_P$ and $\sqsubset_Q$ reflect the restrictions on rule applications caused by modal and quantorial in-permutabilities. We define a new relation $\lhd := (\ll \cup \sqsubset_P \cup \sqsubset_Q)^+$ ('$^+$' denotes the transitive closure) representing the combined restrictions on rule applications caused by $\ll, \sqsubset_P$ and $\sqsubset_Q$. Now a combined substitution $\sigma := \langle \sigma_P, \sigma_Q \rangle$ is admissible, if

1. σ_P induces a homomorphism on prefixes, i.e. $p \preceq q$ implies $\sigma_P^{\#}(p) \preceq \sigma_P^{\#}(q)$.
2. $\lhd$ is irreflexive.
3. Let $t \in \mathcal{T}$ be a term and $P(t)$ denote the set of all γ and δ positions in t. For any $\overline{a}$ with $\sigma(\overline{a}) = t$ must hold for all $b \in P(t) : \sigma_P^{\#}(pre(b)) \preceq \sigma^{\#}(pre(\overline{a}))$.

Let $\langle A, 0 \rangle$ be a signed formula, $\ll$ its position tree, p a path through $\langle A, 0 \rangle$, and $\sigma := \langle \sigma_P, \sigma_Q \rangle$ an admissible combined substitution. A connection $\{b_1, b_2\}$ is called *σ–complementary*, if $\sigma_P^{\#}(pre(b_1)) = \sigma_P^{\#}(pre(b_2))$ and $\sigma_Q^{\#}(lab(b_1)) = \sigma_Q^{\#}(lab(b_2))$, where $\sigma_Q^{\#}$ is the homomorphic extension of σ_Q. A path p is σ–complementary if it contains a σ–complementary connection. A set of σ–complementary connections *span* $\langle A, 0 \rangle$ if each path through $\langle A, 0 \rangle$ is σ–complementary. Putting this together we obtain:

Theorem 1. *A formula A is valid if and only if there is a multiplicity μ, an admissible combined substitution $\sigma := (\sigma_P, \sigma_Q)$, and a set of σ–complementary connections that spans the signed formula $\langle A^\mu, 0 \rangle$.* [20]

In our example we use dashed lines to indicate the induced relations $\sqsubset_P$ and $\sqsubset_Q$. $\lhd$ is irreflexive, the combined substitution $\sigma = \langle \sigma_P, \sigma_Q \rangle$ admissible, and the connection σ–complementary. According to theorem 1 the formula is therefore valid.

3 Matrix Systems

Four major components can be identified in Wallen's matrix characterization of S4. These are a language of matrices, a translation of formulas into matrices, a notion of complementarity, and a re-translation into formulas. Matrices are represented as trees ($\ll$) where nodes are denoted by positions. To each position a label, a polarity, and a type are associated. Labels are simply modal logic formulas, positions are either 0 or 1, and possible types are α, β, γ, δ, ν, or π. The translation from formulas into matrices is pretty straightforward because of a one-to-one correspondence between nodes in a matrix and nodes in a formula tree. The notion of complementarity is more subtle. Paths are defined as sets of positions and are determined purely from the structure of a matrix. Connections are sets of two positions with opposite polarities but identical labels. Sets of connections span a matrix iff every path contains a connection. A prefix is a string of positions which can be calculated from the structure of a matrix. Two kinds of unification problems arise: one from terms and the other from prefixes. Complementarity of a matrix requires the existence of a spanning set of connections which is unifiable such that the induced reduction ordering is irreflexive.

In this section we define matrix systems as a framework for presenting and developing matrix characterizations. Our framework is sufficiently expressive to present all major existing matrix characterizations within it. Furthermore, some concepts have been extended in order to capture future developments.

3.1 Matrix Concept

A matrix concept consists of a matrix language and a multiplicity concept. Each element of a matrix language represents a matrix. A multiplicity concept defines which parts of a matrix can be duplicated. The precise number of duplications is specified by a multiplicity.

A language of *matrices* is defined as a first order language. As usual, a language $\mathfrak{T}$ of *terms* is defined recursively from a set of variables V and families F_i $(i \in \mathbb{N})$ of function symbols of arity i, e.g. $f(t_1, \ldots, t_i) \in \mathfrak{T}$ for $f \in F_i$ and $t_j \in \mathfrak{T}$ $(j \in \{1, \ldots, i\})$. A language $\mathfrak{L}$ of *literals* is defined from a language of terms $\mathfrak{T}$ and families R_i $(i \in \mathbb{N})$ of relation symbols of arity i, e.g. $r(t_1, \ldots, t_i) \in \mathfrak{L}$ for $r \in R_i$ and $t_j \in \mathfrak{T}$ $(j \in \{1, \ldots, i\})$.

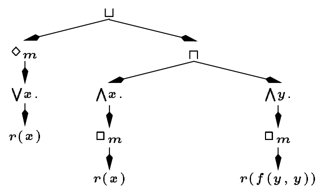
Definition 2. Given a language of literals $\mathfrak{L}(V, F_{i|i\in\mathbb{N}}, R_{i|i\in\mathbb{N}})$ and an ordered set $(S, <)$ a *language of matrices* $\mathfrak{M}$ is defined recursively:

1. Each literal $L \in \mathfrak{L}$ and the empty matrix $\bullet$ are atomic matrices.
2. If $M_1, M_2 \in \mathcal{M}$ then $M_1 \sqcup M_2$, and $M_1 \sqcap M_2$ are matrices.
3. If $M \in \mathcal{M}$ and $x \in V$ then $\bigvee(x, M)$ and $\bigwedge(x, M)$ are matrices.
4. If $M \in \mathcal{M}$ and $s \in S$ then $\Diamond_s M$ and $\Box_s M$ are matrices.

In order to emphasize that $\bigvee$ and $\bigwedge$ are binding operators we also write $\bigvee x.M$ and $\bigwedge x.M$ instead of $\bigvee(x, M)$ and $\bigwedge(x, M)$, respectively.

According to definition 2 each non-atomic matrix can be decomposed into its *sub-matrices*, e.g. M_1 and M_2 are *direct sub-matrices* of $M = M_1 \sqcap M_2$. This yields a binary sub-matrix relation $\succ$, e.g. $M_1 \succ M$. The function *succ* returns the list of direct sub-matrices for a given matrix, e.g. $succ(M) = (M_1, M_2)$. Each matrix M can be viewed as a directed tree. The nodes of such a *matrix tree* correspond to sub-matrices of M and the edges indicate $\succ$. The transitive closure of $\succ$ is denoted by $\gg$ and the transitive reflexive closure by $\ggeq$.

Example 1. The matrix tree for $\Diamond_m \bigvee x.r(x) \sqcup ((\bigwedge x.\Box_m r(x)) \sqcap (\bigwedge y.\Box_m r(f(y,y))))$ is depicted below.



$$\begin{tikzcd}
 & \sqcup \arrow[dl] \arrow[dr] & & \\
\Diamond_m \arrow[d] & & \sqcap \arrow[dl] \arrow[dr] & \\
\bigvee x. \arrow[d] & \bigwedge x. \arrow[d] & & \bigwedge y. \arrow[d] \\
r(x) & \Box_m \arrow[d] & & \Box_m \arrow[d] \\
 & r(x) & & r(f(y,y))
\end{tikzcd}$$

pos	*op*(*p*)	*pos*	*op*(*p*)	*pos*	*op*(*p*)
0	$\sqcup$				
00	$\Diamond_m$	01	$\sqcap$		
000	$\bigvee x.$	010	$\bigwedge x.$	011	$\bigwedge y.$
0000	$r(x)$	0100	$\Box_m$	0110	$\Box_m$
		01000	$r(x)$	01100	$r(f(y,y))$

Definition 3. A *multiplicity concept* $\mathfrak{D}$ is a subset of $\{\bullet, \sqcup, \sqcap, \bigvee, \bigwedge, \Diamond_s, \Box_s\}$[1].

[1] Note that this set depends on S. We will use similar notations at other places.

Usually, matrices are investigated with respect to a fixed multiplicity concept $\mathfrak{D}$. When depicting a matrix, we underline operators from $\mathfrak{D}$ in order to exemplify the multiplicity concept under consideration, e.g. the matrix from example 1 would be written $\underline{\Diamond_m \bigvee} x.r(x) \sqcup ((\bigwedge x.\Diamond_m(x)) \sqcap (\bigwedge y.\Diamond_m r(y)))$ when considering $\mathfrak{D}_{S4} = \{\bigvee, \Diamond_m\}$. This convention is not a change in syntax but rather syntactic sugar. An operator is subject to deletion or duplication iff it is underlined.

Definition 4. A *matrix concept* $\mathcal{M}$ is a pair $(\mathfrak{M}, \mathfrak{D})$.

Basic Positions, Prefixes, and Extended Matrices. *Basic positions* are strings over $\{0, 1\}$ which point into a matrix (. denotes concatenation). A specific basic position is defined as a reference to a sub-matrix. The resulting one-to-one correspondence between basic positions and sub-matrices allows us to identify both concepts in the sequel. Thus, definitions based on the one concept will be used with the obvious meaning for the other concept. When building a theorem prover basic positions can be used in order to avoid notational redundancy.

Definition 5. For any matrix M the *basic position of a sub-matrix* is determined recursively over the structure of M. $bpos_M$ denotes the set of basic positions.

1. M has basic position 0 in M.
2. If $M_1 \overset{\sqcup}{\sqcap} M_2$ has basic position p then M_1 (M_2) has basic position $p.0$ ($p.1$).
3. If $\overset{\bigvee}{\bigwedge} x.M_1$, $\overset{\Diamond_m}{\Box_m} M_1$, has basic position p then M_1 has basic position $p.0$.

We define *lab* as the function which returns the corresponding sub-matrix for a basic position. The main operator of that sub-matrix can be retrieved by the function *op*. The table in example 1 illustrates the correspondence between positions and sub-matrices. A basic position p is called *atomic* iff $lab(p) \in \mathfrak{L}$.

According to the table below *types* are assigned to basic positions depending on the operator. All basic positions with type α, β, γ, δ, ϕ_s, and ψ_s compose the set $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, Γ, Δ, Ψ_s, and Φ_s, respectively. Each basic position p is associated with a *prefix*. The prefix of p is the ordered string of basic positions which are of type ϕ_s or ψ_s (for some $s \in S$) and precede p w.r.t. $\gg$. The empty prefix is denoted by ε. The prefix of a basic position can be retrieved by the function *pre*, e.g. in example 1 $pre(0) = \varepsilon$, $pre(00) = 00$, and $pre(01000) = 0100$ holds.

$op(p)$	$type(p)$	$op(p)$	$type(p)$	$op(p)$	$type(p)$
$\sqcup$	α	$\bigvee$	γ	$\Diamond_s$	ϕ_s
$\sqcap$	β	$\bigwedge$	δ	$\Box_s$	ψ_s

A matrix can be transformed into an *extended matrix* by replacing bound variables by the basic position of the respective binding operator, e.g. the extended matrix for example 1 is $\underline{\Diamond_m \bigvee} x.r(000) \sqcup ((\bigwedge x.\Box_m r(010)) \sqcap (\bigwedge y.\Box_m r(f(011, 011))))$. We denote the language of terms which results from the extension of $\mathfrak{T}$ by basic positions as constants by $\mathfrak{T}_p$.

Positions, Multiplicities, and Expanded Matrix Trees. The definition of *positions* becomes more complicated when *multiplicities* are taken into account. Positions are strings over $\{0,1\} \cup \{0_i, 1_i\}_{i \in \mathbb{N}^+}$.

Definition 6. Assume a multiplicity concept $\mathfrak{D}$, a matrix $M \in \mathfrak{M}$, and a set of numbers $N = \{n_w \in \mathbb{N} \mid w \in (\{0,1\} \cup \{0_i, 1_i\}_{i \in \mathbb{N}^+})^*\}$. We define a set of positions $pos_{M,N}$ and a function bp which returns the basic position of a given position.

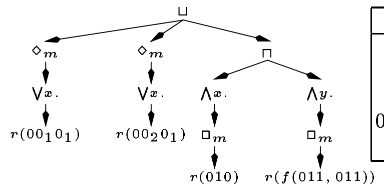
1. M has basic position 0 according to definition 5.
   (a) If $op(0) \notin \mathfrak{D}$ then $0 \in pos_{M,N}$ and $bp(0) = 0$
   (b) else $0_i \in pos_{M,N}$ and $bp(0_i) = 0$ for $0 < i \leq n_0$.
2. If $p \in pos_{M,N}$ and $succ(bp(p)) = (p_1, \ldots, p_l)$ then
   (a) If $op(bp(p).p_k) \notin \mathfrak{D}$ then $(p.p_k) \in pos_{M,N}$ and $bp(p.p_k) = bp(p).p_k$
   (b) else $p.(p_k)_i \in pos_{M,N}$ and $bp(p.(p_k)_i) = bp(p).p_k$ for $0 < i \leq n_{p.p_k}$.

lab, op, and $\succ$ are extended to $pos_{M,N}$ such that for each position p the value for the corresponding basic position is returned.

Definition 7. A *multiplicity* μ is a function from $(\{0,1\} \cup \{0_i, 1_i\}_{i \in \mathbb{N}})^*$ to $\mathbb{N}$.

Usually, a multiplicity μ is used instead of the set N in definition 6. Given a matrix M and a multiplicity μ, we denote the set of positions by $pos_{M,\mu}$. μ needs to be evaluated only at a finite number of points when calculating $pos_{M,\mu}$. For our purposes, it suffices to specify μ just at these points. In a matrix proof just μ must be searched for while the set of positions can then be calculated.

Example 2. The tree for our example matrix and the corresponding positions are depicted below for the multiplicity μ with $\mu(00) = 2$, $\mu(00_10) = 1$ and $\mu(00_20) = 1$. Note that nodes of arity > 2 may occur in such a tree.



pos	*op*(*p*)	*pos*	*op*(*p*)	*pos*	*op*(*p*)
0	$\sqcup$				
$00_1, 00_2$	$\Diamond_m$	01	$\sqcap$		
$00_10_1, 00_20_1$	$\bigvee x.$	010	$\bigwedge x.$	011	$\bigwedge y.$
$00_10_10, 00_20_10$	$r(x)$	0100	$\Box_m$	0110	$\Box_m$
		01000	$r(x)$	01100	$r(f(y,y))$

We call trees like the one in example 2 *implicitly expanded trees*. These trees are based on the notion of positions and therefore – in general – require nodes of arity > 2. *Explicitly expanded trees* provide a means to express a matrix with a given multiplicity with basic positions and without the need for nodes of arity > 2. They also enable us to extend the notion of basic–complementarity to multiplicities as will be explained in the subsequent subsection.

The explicit expansion of a matrix M for a multiplicity μ is defined recursively. If M contains no position p with $\mu(p) \neq 1$ then M is explicitly expanded. If M contains a position p with $\mu(p) = 0$ and M' is the matrix which results from replacing the sub-matrix which corresponds to p by the empty matrix $\bullet$ then the expansion of M for μ equals the expansion of M' for μ. If M contains a position p with $\mu(p) = n > 1$ and there is no proper prefix $\hat{p}$ of p with $\mu(\hat{p}) > 1$, M' is the matrix which results from replacing the sub-matrix corresponding to

p with the matrix $lab(p) \sqcup lab(p)$, p_1 (p_2) is the position of the left-hand (right-hand) formula $lab(p)$ in M', and μ' is a multiplicity which equals μ except for $\mu'(p_1) = n - 1$ and $\mu'(p_2) = 1$ then the expanded matrix of M for μ equals the expanded matrix of M' for μ'. Since they are superfluous, indices of positions are omitted for expanded trees.

Example 3. The explicitly expanded matrix tree and the corresponding positions for the multiplicity μ as defined in example 2 are depicted below. Note that minor changes to a multiplicity cause only local changes to the corresponding explicitly expanded matrix tree.

pos	$op(p)$	*pos*	$op(p)$	*pos*	$op(p)$
0	$\sqcup$				
00	$\sqcup$	01	$\sqcap$		
000, 001	$\Diamond_m$	010	$\bigwedge x.$	011	$\bigwedge y.$
0000, 0010	$\bigvee x.$	0100	$\Box_m$	0110	$\Box_m$
00000, 00100	$r(x)$	01000	$r(x)$	01100	$r(f(y,y))$

$$\begin{tikzcd}
&&& \sqcup \arrow[dll] \arrow[drr] &&& \\
& \sqcup \arrow[dl] \arrow[dr] &&&& \sqcap \arrow[dl] \arrow[dr] & \\
\Diamond_m \arrow[d] && \Diamond_m \arrow[d] && \bigwedge x. \arrow[d] && \bigwedge y. \arrow[d] \\
\bigvee x. \arrow[d] && \bigvee x. \arrow[d] && \Box_m \arrow[d] && \Box_m \arrow[d] \\
r(0010) && r(0000) && r(010) && r(f(011, 011))
\end{tikzcd}$$

Remark. The reader familiar with [4, 6] might be astonished by our rather complicated definition of matrices. However, it is necessary in order to capture many non–classical logics and to keep the approach extensible.

3.2 Complementarity Concept

We first define a *basic–complementarity* concept using basic positions, i.e. leaving multiplicities out of consideration. Using the notion of expanded matrices, we extend our definitions in order to capture multiplicities afterwards.

Paths, Polarities, and Connections. We define *paths* through a matrix M as specific sets of basic positions.

1. $\{0\}$ is a path.
2. If $P \cup \{p\}$ is a path, p is not atomic, and
   (a) if $op(p) \in \{\sqcup, \bigvee, \bigwedge, \Diamond_s, \Box_s\}$ then $P \cup succ(p)$ is path
   (b) if $op(p) = \sqcap$ then $P \cup \{p_k\}$ is a path for each $p_k \in succ(p)$.

A path p is *atomic* if it contains atomic positions only. There are 2 atomic paths through the matrix in example 3 which are $p_1 = \{00000, 00100, 01000\}$ and $p_2 = \{00000, 00100, 01100\}$.

A *polarity* is a function *pol* which assigns to atomic basic positions elements of a finite boolean algebra $(L, \cup, \cap, \sim, \bot, \top)$ (e.g. [8]), the *polarity concept. pol* assigns to all positions with label $\bullet$ the polarity $\bot$. For a *matrix–polarity pair* (M, pol), a *connection* c is a set of atomic basic positions for which a subset c' exists such that the labels all basic positions in c' have the same relational symbol and that $\bigcup_{p \in c'} pol(p) = \top$ holds. c' is called the *relevant subset* of c.

Example 4. We inspect the lattice with $L = \{\top, 0, 1, \bot\}$ as depicted. We define a polarity *pol* for the expanded tree in example 3 by $pol(00000) = 0$, $pol(00100) = 0$, $pol(01000) = 1$ and $pol(01100) = 1$. Possible connections would be $c_1 = \{00000, 01000\}$, $c_2 = \{00000, 01100\}$, $c_3 = \{00100, 01000\}$, and $c_4 = \{00100, 01100\}$.

$$\begin{tikzcd}
& \top \arrow[dl] \arrow[dr] & \\
0 \arrow[dr] && 1 \arrow[dl] \\
& \bot &
\end{tikzcd}$$

Substitutions, Unification Problems, and Matings. In matrix systems two different kinds of substitutions exist. Both substitutions are required to be idempotent. A *quantorial substitution* σ_Q assigns terms from $\mathfrak{T}_p$ to basic positions from Γ (see [6] for usual binding rules). A *prefix substitution* σ_P assigns strings of basic positions from $\bigcup_{s\in S}(\Phi_s \cup \Psi_s)$ to basic positions from $\bigcup_{s\in S}\Phi_s$. For any $p \in \Phi_s$ with $\sigma_P(p) = t$ must hold $t \in \bigcup_{s' \leq s}(\Phi_{s'} \cup \Psi_{s'})^*$, i.e. t contains no basic positions of bigger sorts. We extend σ_P to $\bigcup_{s\in S}(\Phi_s \cup \Psi_s)$ by $\sigma_P(p) = p$ for any $p \in \bigcup_{s\in S}\Psi_s$ and denote the homomorphic extension to $(\bigcup_{s\in S}(\Phi_s \cup \Psi_s))^*$ by $\sigma_P^\sharp$. Given a Matrix M we define a set of prefixes $\mathit{Pre}(M) = \{t \mid t \in \mathit{bpos}_M\}$, $\mathit{Pre}_{\sigma_P}(M) = \{t \mid t \subseteq \sigma_P^\sharp(t'), t' \in \mathit{Pre}(M)$, and restrict prefix substitutions by $t.p \in \mathit{Pre}_{\sigma_P}(M) \Rightarrow t.p = \sigma_P^\sharp(\mathit{pre}(p))$, i.e. they should result from unifications.

The two kinds of substitutions result in different *unification problems* arising from connections. A *quantorial unification problem* arises from the task to unify the labels of all positions in a relevant subset, i.e. a quantorial substitution is required under which the arguments of the respective literals become identical. For example the substitution with $\sigma_Q(0000) = f(011, 011)$ unifies the labels of c_2. A *prefix unification problem* arises when the prefixes of all elements in a relevant subset need to be unified, e.g. the substitution with $\sigma_P(000) = 0100$ unifies the prefixes of c_2. A *combined unification problem* is a combination of both.

Each substitution σ_Q and σ_P induces a reduction ordering $\sqsubset_Q \subseteq \Gamma \times (\Gamma \cup \Delta)$ and $\sqsubset_P \subseteq \bigcup_{s\in S}\Phi_s \times \bigcup_{s\in S}(\Phi_s \cup \Psi_s)$, respectively. For a substitution σ_Q and any position $p \in \Delta$ if p occurs in $\sigma_Q(p')$ for some $p' \in \Gamma$ then $p \sqsubset_Q p'$. For σ_P and any position $p \in \bigcup_{s\in S}\Psi_s$ if p occurs in $\sigma_P(p')$ for any $p' \in \bigcup_{s\in S}\Phi_s$ then $p \sqsubset_P p'$. We define $\lhd$ as the transitive closure of $\prec \cup \sqsubset_Q \cup \sqsubset_P$.

A *mating* C is a set of connections for a matrix–polarity pair (M, pol). If every path through a matrix M contains at least one connection from C then C *spans* M, e.g. in example 4 $C_1 = \{c_1, c_3, c_4\}$ and $C_2 = \{c_2, c_3\}$ are spanning. A mating is *quantorial unifiable* if there exists a quantorial substitution σ_Q such that all positions from the relevant subset of each connection in C have the same label under σ_Q. A mating is *prefix unifiable* if there exists a prefix substitution σ_P such that the prefixes of all connections in C are identical under σ_P. A mating C is *unifiable* if there exists a combined substitution $\sigma = (\sigma_Q, \sigma_P)$ such that C is quantorial unifiable with respect to σ_Q and prefix unifiable with respect to σ_P, e.g. C_2 is unifiable while C_1 is not.

Complementarity. A matrix–polarity pair (M, pol) is *basic–complementary* if there exists a set of connections C and a combined substitution $\sigma = (\sigma_Q, \sigma_P)$ such that C spans M, σ unifies C, the induced reduction ordering $\lhd$ is irreflexive, and a set of *complementary properties* $\mathcal{P}$ is fulfilled. Typical properties will be discussed in section 4.1.

A matrix–polarity pair (M, pol) is *complementary* if there exists a multiplicity μ for which $\overline{M}$ is the corresponding expanded matrix and $\overline{\mathit{pol}}$ the corresponding polarity which assigns to any atomic position the polarity of the corresponding basic position such that $(\overline{M}, \overline{\mathit{pol}})$ is basic–complementary. A *complementarity concept* $\mathcal{C}$ is defined by $\mathcal{P}$. It induces a binary predicate $\mathcal{C}$ such that $\mathcal{C}(M, \mathit{pol})$ holds iff (M, pol) is complementary.

Definition 8. A *matrix system* $\mathcal{MS}$ is a pair $(\mathcal{M}, \mathcal{C})$.

4 Relating a Matrix System and a Logic

Matrix based proof methods can be applied to a specific logic if an appropriate matrix characterization exists. Four major components of matrix characterizations can be distinguished: a logic, a matrix system, and two translations. In this framework it suffices to define a logic by $\mathcal{L} = (\mathfrak{F}, \mathcal{V})$ where $\mathfrak{F}$ is a language (of well–formed formulas) and $\mathcal{V}$ is a validity concept. $\mathcal{V}(F)$ holds for a formula $F \in \mathfrak{F}$ iff F is valid in the logic.

Definition 9. Assume a logic $\mathcal{L}$, a matrix system $\mathcal{MS} = (\mathcal{M}, \mathcal{C})$, a translation *lm* which maps a formula to a matrix–polarity pair, and a translation *ml* which maps a matrix–polarity pair from the codomain of *lm* ($\textsc{codom}(lm)$) to a formula. We call the 4-tuple $\mathcal{MC} = (\mathcal{L}, \mathcal{MS}, lm, ml)$ a *matrix-characterization* of $\mathcal{L}$ iff for any formula $F \in \mathfrak{F}$ and any matrix–polarity pair $(M, pol) \in \textsc{codom}(lm)$ holds

$$1.)\ \mathcal{V}(F) \text{ iff } \mathcal{C}(lm(F)) \qquad \text{and} \qquad 2.)\ \mathcal{C}((M, pol)) \text{ iff } \mathcal{V}(ml((M, pol)))$$

The asymmetry in the definition is because we require every formula to be mapped to an appropriate matrix–polarity pair but do not enforce that every matrix has a formula as counterpart.

4.1 Relation to Proof Theory

For many logics validity can be expressed by a sequent calculus. We sketch how to get from a specific calculus for a logic $\mathcal{L}$ to a matrix–characterization of that logic. In fact, we describe how to determine $\mathcal{MS}$ and *lm*.

A sequent calculus is composed of a set of rules. Each rule consists of a *conclusion* and possibly multiple *premises*. A *principal formula* of a rule occurs in the conclusion but not in any premise. Formulas which occur in a premise but not in the conclusion are called *active*. All other formulas are the *context*. A rule with multiple principal formulas is called *context–sensitive*, e.g. the $\Box$–rule in section 2 . A principal formula in such a rule is either *reduced*, i.e. sub-formulas of it are active, or otherwise *deleted*.

Usually, formulas are translated to matrices in a recursive manner. Most operators can be translated using the tableaux–scheme [20]. However, the translation must take care of non-permutabilities in the sequent calculus. Quantifiers are translated into $\bigvee$ and $\bigwedge$. Context–sensitive rules can result in $\Diamond_s$ or $\Box_s$ operators in the image of formulas – $\Diamond_s$ for reduced formulas, $\Box_s$ for formulas in the context but none for deleted formulas. For the example in section 2, there is a one-to-one correspondence where $\Box$ and $\Diamond$ connectives are translated into $\Box_m$ and $\Diamond_m$ operators, respectively. For other logics, this is more complicated, e.g. intuitionistic logics [20] or $\mathcal{MLL}$ [15]. However, to construct an efficient matrix representation the number of such operators should be kept to minimum. Similarly, the multiplicity concept should be kept to a minimal size. For our $S4$ example the multiplicity concept $\mathfrak{D}_{S4} = \{\bigvee, \Diamond_m\}$ is sufficient.

After $\mathcal{M}$ and lm have been determined one needs to find a suitable complementarity concept, i.e. a set of complementarity properties. In existing matrix characterizations these properties originate from context–sensitive rules (e.g. K modal logics [20]), domain conditions (e.g. cumulative domains [20]), or missing structural rules (e.g. linear logic [15]). For the first two we refer to [20] and just sketch the last one.

Omitting structural rules (depicted below) from a sequent calculus results in a sub-structural logic. For each missing structural rule a complementarity property needs to be added to the corresponding matrix system. Elements of this subset of $\mathcal{P}$ are called *structural complementarity properties* ($\mathcal{P}_S$). Omitting the contraction rule is the key to resource sensitivity. Without such a rule a *linearity condition* must be added to $\mathcal{P}_S$ (e.g. linear connection method [5]) which requires that each atomic position occurs in at most one connection of the mating. Omitting the weakening rule results in relevance logics. Any sub-formula must contribute to the proof in some axiom. The corresponding complementarity condition is that each atomic position must occur in at least one connection of the mating, i.e. it is *relevant.* If the mix–rule which is a weak version of the weakening rule is removed as well each formula must contribute to an important axiom. For a matrix the spanning set of connections is required to be *minimal*, i.e. removal of any connection would result in a mating which is not spanning any more.

$$\frac{\Rightarrow \Gamma, F, F}{\Rightarrow \Gamma, F}\,C \qquad \frac{\Rightarrow \Gamma}{\Rightarrow \Gamma, F}\,W \qquad \frac{\Rightarrow \Gamma_1, F_1 \quad \Rightarrow \Gamma_2, F_2}{\Rightarrow \Gamma_1, \Gamma_2, F_1, F_2}\,M$$

4.2 Semantics of matrix systems

There is a close relationship between complementarity in matrix systems and *possible world semantics*. Due to the lack of space we only sketch the idea. The key is to denote variable (constant) positions of a matrix as infinite joins (meets) over structures reflecting the structural complementarity properties. In this setting adding structural rules, i.e. forgetting about some restrictions of these properties, is reflected by further narrowing the class of these structures. Since in S4 all structural rules are present, we choose the class of boolean-algebras [8]. In general, we assume that these structures consist of a complete ortholattice [8] with a monoid and several requirements concerning the interaction between the lattice and monoid operations. Semantically, we interpret the operator $\sqcap$ ($\sqcup$) by the monoid operation (its dual).

To handle quantorial and prefix positions uniformly we treat quantifiers as multi-modal operators, whereby $\forall x$ and $\exists x$ play the role of modalities for each variable x. Possible valuations play the role of *worlds* and the *accessibility relation* relates valuations that are equal up to x. To cope with quantorial and prefix positions a new argument, the *cluster-valuation pair*, for short *the world* is added to the interpretation. A cluster is representing a zone of permutabilities modulo prefix-types, while valuations represent specific points of permutabilities concerning quantorial permutabilities. Each prefix-type comes with an *accessibility relation* integrating the requirements of the complementarity concept. For instance, in S4 the accessibility relation, say R, is reflexive and transitive because the variable prefix positions can be substituted by any string of basic-positions including the empty one.

Dependencies between different substitutions are represented by an unary relation I over worlds, i.e. cluster-valuation pairs. In S4 interaction between the quantorial and modal positions is represented by the hereditary condition: If $(w, v) \in I$ and $(w, \hat{w}) \in R$ then $(\hat{w}, v) \in I$. This is just another way to express the monotonicity of valuations with respect to the reachability of clusters. By this we have recovered the semantics of first-order modal logic S4 with cumulative domains. Though this result might not be very surprising for S4, our approach offers a way to obtain possible world semantics for any non-classical logic that comes with a matrix system. Instead of using semantics to obtain a matrix system, like e.g. in [4, 20], we *extract* the semantics from the matrix system.

5 Conclusion

We have presented a formal definition of matrix systems as a framework for matrix characterizations, their relation to sequent calculi, and the semantics *naturally induced* by matrix-systems. By this, we substantiate the results of [4, 20, 17, 11] concerning an exact definition of a matrix characterization. Additionally, we extend Wallen's approach [20] concerning multiplicities, connections, the integration of partial-ordered prefix types, and polarities. It is noteworthy that the extension of polarities offers a natural way to handle even multi-valued logics and that our notion of a connection becomes necessary for handling logical constants.

An important problem is clearly how to find the appropriate complementarity properties, i.e. answering the question: given a sequent calculus how to obtain an appropriate matrix characterization ? While this seems easy for the structural complementarity properties, the conditions imposed on the others are more subtle. We expect that further experience will provide the necessary insights to solve this problem for a well-chosen class of sequent calculi. Besides working purely proof-theoretically, it might be interesting to make use of already existing semantics. Indeed the matrix characterizations for modal logics presented in [20] are based on Kripke semantics.

Regarding a matrix system as a mathematical object of its own rights offers a lot room for extensions. One might be interested in performing typical constructions of matrix systems like direct-(co-)products or (co-)limits. For instance, the latter one could be used to *melt together* different matrix systems. The question arising in this setting, is how these constructions could be reflected in the semantics of the matrix system.

In so far we regard the work presented here, as a foundation for a promising new research area in the spirit of Tarski's contribution to meta-mathematics concerning his concept of a *formalized deductive-system* [18].

References

1. P. B. Andrews. Theorem-proving via general matings. *Journal of the Association for Computing Machinery*, 28 (2), 193–214, 1981.
2. A. Avron. Simple consequence relations. *Information and Computing*, 92:105-139, 1991.
3. G. Bellin and J. Ketonen. A decision procecure revisited: Notes on direct logic, linear logic and its implementation. *Theoretical Computer Science*, 95:115–142, 1992.
4. W. Bibel. On Matrices with Connections. *Jour. of the ACM*, 28, 633–645, 1981.
5. W. Bibel. A deductive solution for plan generation. *New Generation Computing*, 4:115–132, 1986.
6. W. Bibel. *Automated Theorem Proving*. Vieweg, 1987.
7. W. Bibel, M. Thielscher. Non-Classical Automated Deduction. *Trends in Theoretical Informatics*, 89, 39–59, 1996.
8. G. Birkhoff. Lattice Theory. American Mathematical Society, New York, 1967.
9. K. Dosen. A Historical Introduction to Substructural logics. In *Substrucal Logics* ed. by K. Dosen and P. Schroeder-Heister. Oxford University Press, 1993.
10. M. Dunn. Partial Gaggles Applied to Logics with Restricted Structural Rules. In *Substructural Logics* ed. by K. Dosen and P. Schroeder-Heister. Oxford University Press, 1993.
11. B. Fronhöfer *The Action-as-Implication Paradigm*, CS Press, 1996.
12. J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.
13. J. McCarthy and P.J. Hayes. Some Philosophical Problems from the Standpoint of Artificial Intelligence. *Machine Intelligence*, 4:463–502, 1969.
14. J. Ketonen and R. Weyrauch. A decidable fragment of predicate calculus. *Theoretical Computer Science*, 32:297-307, 1984.
15. C. Kreitz, H. Mantel, J. Otten, S. Schmitt. Connection-Based Proof Construction in Linear Logic. *CADE–14*, Springer Verlag, 1997. *to appear*
16. H. Mantel. Eine Matrixcharakterisierung für ein Fragment der linearen Logik. *Diplomarbeit*, TH-Darmstadt, Germany, 1996.
17. J. Otten, C. Kreitz. A Uniform Proof Procedure for Classical and Non-Classical Logics. *KI-96: Advances in Artificial Intelligence*, LNAI 1137, pp. 307–319, Springer Verlag, 1996.
18. A. Tarski. Logic, semantics, metamathematics. Oxford University Press, 1956.
19. S. Schmitt, C. Kreitz. Converting non-classical matrix proofs into sequent-style systems. *CADE–13*, LNAI 1104, pp. 418–432, Springer Verlag, 1996.
20. L. Wallen. *Automated Deduction in Non-Classical Logics*. MIT Press, 1990.