Implementing System BV of the Calculus of Structures in Maude

Ozan Kahramanoğullari
University of Leipzig and Dresden University of Technology, ICCL
ozan@informatik.uni-leipzig.de

Abstract. System BV is an extension of multiplicative linear logic with a non-commutative self-dual operator. We first map derivations of system BV of the calculus of structures to rewritings in a term rewriting system modulo equality, and then express this rewriting system as a Maude system module. This results in an automated proof search implementation for this system, and provides a recipe for implementing existing calculus of structures systems for other logics. Our result is interesting from the view of applications, specially, where sequentiality is essential, e.g., planning and natural language processing. In particular, we argue that we can express plans as logical formulae by using the sequential operator of BV and reason on them in a purely logical way.

1 Introduction

The calculus of structures is a proof theoretical formalism, like natural deduction, the sequent calculus and proof nets, for specifying logical systems syntactically. It was conceived in [10] to introduce the logical system BV, which extends multiplicative linear logic by a non-commutative self-dual logical operator. Then it turned out to yield systems with interesting and exciting properties for existing logics such as classical logic [2], linear logic [19] and modal logics [18], and new insights to their proof theory. In [20], Tiu showed that BV is not definable in any sequent calculus system. Bruscoli showed in [3] that the non-commutative operator of BV captures precisely the sequentiality notion of process algebra, in particular CCS [16].

In contrast to sequent calculus, the calculus of structures does not rely on the notion of main connective and, like in term rewriting, it permits the application of the inference rules deep inside a formula (structure). In this paper, exploiting this resemblance, we present a general procedure turning derivations in logical systems of the calculus of structures into rewritings in term rewriting systems modulo equality. We illustrate our procedure on

Proceedings of the Ninth ESSLLI Student Session
Paul Egré and Laura Alonso i Alemany (editors)

system BV of the calculus of structures. Then, we encode the resulting term rewriting system in *Maude* [5, 4] which results in an implementation of an automated proof search tool for system BV. We also argue that we can employ system BV on applications where sequentiality is essential. In particular, we refer to our encoding of the conjunctive planning problems in the language of BV which allows to express plans as logical formulae. Space restrictions do not permit us to present this encoding in detail, we refer to [14].

2 System BV

In this section, we will shortly present the system BV of the calculus of structures, following [10]. Systems in the calculus of structures for other logics [2, 19, 18] are designed by respecting the scheme in this section.

There are countably many *atoms*, denoted by $a, b, c, \ldots$ Structures of the language BV are denoted by $R, S, T, \ldots$ and are generated by

$$S ::= \circ \mid a \mid \langle \underbrace{S; \ldots; S}_{>0} \rangle \mid [\underbrace{S, \ldots, S}_{>0}] \mid (\underbrace{S, \ldots, S}_{>0}) \mid \overline{S} \quad , \qquad (1.1)$$

where a stands for any atom and $\circ$, the *unit*, is not an atom. $\langle S; \ldots; S \rangle$ is called a *seq structure*, $[S, \ldots, S]$ is called a *par structure*, and $(S, \ldots, S)$ is called a *copar structure*, $\overline{S}$ is the *negation* of the structure S. Structures are considered equivalent modulo the relation $\approx$, which is the smallest congruence relation induced by the equations shown in Figure 1.1.[1] There $\vec{R}$, $\vec{T}$ and $\vec{U}$ stand for finite, non-empty sequences of structures. A *structure context*, denoted as in $S\{\ \}$, is a structure with a hole that does not appear in the scope of negation. The structure R is a *substructure* of $S\{R\}$ and $S\{\ \}$ is its *context*. Context braces are omitted if no ambiguity is possible: for instance $S[R, T]$ stands for $S\{[R, T]\}$. A structure, or a structure context, is said to be in *negation normal form* when the only negated structures appearing in it are atoms, no unit $\circ$ appears in it and no parentheses can be equivalently eliminated.

In the calculus of structures, a typical *(deep) inference rule* is a scheme of the kind

$$\frac{S\{T\}}{S\{R\}}\,\rho$$

where ρ is the *name* of the rule, T is its *premise* and R is its *conclusion*. Such a rule specifies the implication $T \Rightarrow R$ inside a generic context $S\{\ \}$, which

[1] In [10] axioms for context closure are added. However, because each equational system includes the axioms of equality, context closure follows from the substitutivity axioms.

Associativity

$$\langle \vec{R}; \langle \vec{T} \rangle; \vec{U} \rangle \approx \langle \vec{R}; \vec{T}; \vec{U} \rangle$$
$$[\vec{R}, [\vec{T}]] \approx [\vec{R}, \vec{T}]$$
$$(\vec{R}, (\vec{T})) \approx (\vec{R}, \vec{T})$$

Singleton

$$\langle R \rangle \approx [R] \approx (R) \approx R$$

Commutativity

$$[\vec{R}, \vec{T}] \approx [\vec{T}, \vec{R}]$$
$$(\vec{R}, \vec{T}) \approx (\vec{T}, \vec{R})$$

Unit

$$\langle \circ; \vec{R} \rangle \approx \langle \vec{R}; \circ \rangle \approx \langle \vec{R} \rangle$$
$$[\circ, \vec{R}] \approx [\vec{R}]$$
$$(\circ, \vec{R}) \approx (\vec{R})$$

Negation

$$\overline{\circ} \approx \circ$$
$$\overline{\langle R; T \rangle} \approx \langle \overline{R}; \overline{T} \rangle$$
$$\overline{[R, T]} \approx (\overline{R}, \overline{T})$$
$$\overline{(R, T)} \approx [\overline{R}, \overline{T}]$$
$$\overline{\overline{R}} \approx R$$

Figure 1.1: The equational system underlying BV.

$$\frac{}{\circ}\,\circ\!\downarrow \qquad \frac{S\{\circ\}}{S[a, \bar{a}]}\,\mathrm{ai}\!\downarrow \qquad \frac{S([R, T], U)}{S[(R, U), T]}\,\mathrm{s} \qquad \frac{S\langle [R, T]; [R', T'] \rangle}{S[\langle R; R' \rangle, \langle T; T' \rangle]}\,\mathrm{q}\!\downarrow$$

Figure 1.2: System BV

is the implication being modeled in the system[2]. An inference rule is called an *axiom* if its premise is empty. Rules with empty contexts correspond to the case of the sequent calculus.

A (formal) *system* $\mathscr{S}$ is a set of inference rules. A derivation Δ in a certain formal system is a finite chain of instances of inference rules in the system. A derivation can consist of just one structure. The topmost structure in a derivation, if present, is called the *premise* of the derivation, and the bottommost structure is called its *conclusion*. The *length* of a derivation is the number of instances of inference rules appearing in it.

The system $\{\circ\!\downarrow, \mathrm{ai}\!\downarrow, \mathrm{s}, \mathrm{q}\!\downarrow\}$, shown in Figure 1.2, is denoted BV and called *basic system* V, where V stands for one non-commutative operator[3]. The rules of the system are called *unit* ($\circ\!\downarrow$), *atomic interaction* ($\mathrm{ai}\!\downarrow$), *switch* (s) and *seq* ($\mathrm{q}\!\downarrow$). We consider $\mathrm{ai}\!\downarrow$ to be a schema for all positive atoms a.

There is a straightforward correspondence between structures not involving seq and formulae of multiplicative linear logic. For example $[(a, b), \bar{c}, \bar{d}]$ corresponds to $(\,(a \otimes b) \parr c^{\perp} \parr d^{\perp})$, and vice versa. Units 1 and $\perp$ are mapped

[2]Due to duality between $T \Rightarrow R$ and $\bar{R} \Rightarrow \bar{T}$, rules come in pairs of dual rules: a down-version and an up-version. For instance, the dual of the $\mathrm{ai}\!\downarrow$ rule is the cut rule. In this paper, we only consider the down rules, which provide a sound and complete system.

[3]This name is due to the intuition that W stands for two non-commutative operators.

into $\circ$, since $1 \equiv \bot$, when the rules mix and mix0 are added to MELL. For a detailed discussion on the proof theory of BV and the precise relation between BV and multiplicative linear logic the reader is referred to [10].

3 From Derivations to Rewritings

In this paper, we assume that the reader is familiar with the notions of term rewriting such as terms, positions, replacements, substitutions, equations and rewrite rules as can be found in e.g. [1, 17]. However, we will recapitulate the definition of the rewrite relation R/E that will be used extensively. This section is partly a summary of the technical report [12].

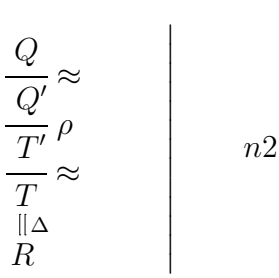
Given terms s, t, a term rewriting system R and an equational system E, *s rewrites to t wrt R and E*, denoted by $s \rightarrow_{R/E(\rho,\pi,\sigma)} t$ if there are terms s', t', a rewrite rule $\rho = l \rightarrow r$, a position $\pi \in pos(s')$ and a substitution σ such that $s \approx_E s'$, $s'|_\pi = \sigma(l)$, $t' = s'|\sigma(r)|_\pi$ and $t' \approx_E t$. In other words, $s \rightarrow_{R/E(\rho,\pi,\sigma)} t$ iff $(\exists s', t')\, s \approx_E s' \rightarrow_{R(\rho,\pi,\sigma)} t' \approx_E t$.

3.1 Replacing Equivalence Classes by Equality Steps

For this purpose, we separate the notion of a structure from the equivalence class defined by the equations shown in Figure 1.1. From this point on, a structure is an expression of the form delivered in (1.1) and no longer an equivalence class of these expressions.

A structure R is a *derivation from R to R*. If Δ is a derivation from structure R to structure T, $T \approx T'$, there is an instance of an inference rule ρ with conclusion T' and premise Q', and $Q' \approx Q$ then the derivation on the left-hand-side of Figure 1.3 is a *derivation from R to Q*. For notational convenience we combine two subsequent equality steps occurring in a derivation into a single equality step. The notion of a proof can be analogously redefined, that is, Δ is a proof of R iff Δ is a derivation from R to T and $T \approx \circ$.

Because $\approx$ is the finest congruence relation generated by the equational system shown in Figure 1.1, each derivation and each proof as defined in Section 2 can be transformed into a derivation and a proof as defined in this subsection, respectively. We have thus clarified the role of the equational theory underlying derivations in BV. The same kind of changes to BV have already been considered in [2].



$$n22(S) = \begin{cases} \circ & \text{if } S = \circ, \\ S & \text{if } S \text{ is an atom}, \\ \overline{n22(R)} & \text{if } S = \overline{R}, \\ \langle n22(R); n22(\vec{T})\rangle & \text{if } S = \langle R; \vec{T}\rangle, \\ (n22(R), n22(\vec{T})) & \text{if } S = (R, \vec{T}), \\ [n22(R), n22(\vec{T})] & \text{if } S = [R, \vec{T}]. \end{cases}$$

Figure 1.3: **Left:** A derivation from R to Q **Right:** Transformation $n22$

3.2 Replacing n-ary Operators by binary Ones

We will now restrict ourselves to binary operators. The recursive transformation on the right-hand-side of Figure 1.3 turns each structure into a structure, where only the binary operators $\langle \_; \_\rangle$, $(\_, \_)$ and $[\_, \_]$ are used.[4] As a consequence, we will also simplify the equations defining the syntactic equivalence leading to a refined set of equations as shown in Figure 1.5, where the equations for singleton become superfluous. Because the inference rules for BV (see Figure 1.2) use only binary seq-, par- and copar-operators, there is no need to change them.

Because $n22(S) \approx S$, derivations wrt n-ary seq-, par- and copar- operators can be equivalently turned into derivations with only binary seq, par- and copar-operators and vice versa. This may lead to less intelligible structures, but the n-ary operators may be reintroduced as abbreviations (e.g. [8, 13]).

3.3 Replacing Structures by Terms

We replace the structures by terms, and consider terms over variables, thus formalizing the concept of structures with variable occurrences. Let

$$\Sigma_{\mathsf{BV}} = \{\circ,\ \overline{\_},\ [\_, \_],\ (\_, \_),\ \langle \_; \_\rangle\} \cup \{a \mid a \text{ is a positive atom}\}.$$

Then, structures as defined in Section 2 are simply Σ_{BV}-terms over the empty set of variables, i.e., ground Σ_{BV}-terms. On the other hand, by considering a non-empty set $\mathcal{V}$ of variables, we obtain Σ_{BV}-terms over $\mathcal{V}$, which correspond to structures with variables.

[4] While applying this transformation, due to associativity of the structures, it is important to observe the equivalence $[R, \vec{T}] = [R, T_1, \ldots, T_n] = [R, [T_1, \ldots, T_n]]$ of structures where $n \geq 1$.

$$R_{Neg} = \left\{ \begin{array}{lcl|l} \bar{\circ} & \to & \circ & \texttt{eq - o = o .} \\ \overline{\langle R;T\rangle} & \to & \langle \bar{R};\bar{T}\rangle & \texttt{eq - < R ; T > = < - R ; - T > .} \\ \overline{[R,T]} & \to & (\bar{R},\bar{T}) & \texttt{eq - [R , T] = \{ - R , - T \} .} \\ \overline{(R,T)} & \to & [\bar{R},\bar{T}] & \texttt{eq - \{ R , T \} = [- R , - T] .} \\ \bar{\bar{R}} & \to & R & \texttt{eq - - R = R .} \end{array} \right.$$

Figure 1.4: **Left:** T. R. System R_{Neg} **Right:** Corresponding Maude code

This way, we can use the notions structure and $\Sigma_{\mathtt{BV}}$-term synonymously and replace the notion of context in derivations within BV by the notion of a position, thus being precise about which substructure or subterm is replaced in a derivation step: the notion of positions, subterms and the replacement of a subterm by another one at a particular position take over the role of a context in BV.

3.4 Orienting the Equalities for Negation

The inference rules of BV can be applied only to the structures which are not under the scope of negation sign. Since these rules do not introduce any new negation signs, neither when they are applied bottom-up nor top-down, we can orient the equalities for negation as rewrite rules from left to right to get the negation normal form at the beginning of a derivation:

Lemma 1. *Term rewriting system R_{Neg} on the left-hand-side of Figure 1.4 is (i.) terminating, (ii) confluent. (iii.) Let s be a $\Sigma_{\mathtt{BV}}$-term. The normal form of s with respect to R_{Neg} is in negation normal form.*

Sketch of Proof (*i*) It suffices to take the lexicographic path order $\overline{\_} >_{lpo} [\_,\_] >_{lpo} (\_,\_) >_{lpo} \langle \_;\_\rangle >_{lpo} \circ$ as stated in [1]. (*ii*) Since R_{Neg} is terminating, the result follows from the analysis of the critical pairs. (*iii*) s being in negation normal form and applicability of a rewrite rule of R_{Neg} are contradictory. □

3.5 Replacing Inference Rules by Rewrite Rules

In the final step, we define the term rewriting system RBV and the equational theory EBV corresponding to BV such that derivations in BV correspond to rewritings $\to_{\mathtt{RBV}/\mathtt{EBV}}$. The context occurring in inference rules is eliminated

Associativity	**Commutativity**	**Unit**
$\langle R;\langle S;T\rangle\rangle \approx \langle\langle R;S\rangle;T\rangle$	$[R,T] \approx [T,R]$	$\langle\circ;R\rangle \approx \langle R;\circ\rangle \approx R$
$[R,[S,T]] \approx [[R,S],T]$	$(R,T) \approx (T,R)$	$[\circ,R] \approx R$
$(R,(S,T)) \approx ((R,S),T)$		$(\circ,R) \approx R$

Figure 1.5: The equational system EBV.

$[a,\bar{a}]$	$\rightarrow$	$\circ$	$\mathfrak{ai}\downarrow$
$[\langle R;R'\rangle,\langle T;T'\rangle]$	$\rightarrow$	$\langle[R,T];[R',T']\rangle$	$\mathfrak{q}\downarrow$
$[(R,T),U]$	$\rightarrow$	$([R,U],T)$	$\mathfrak{s}$

Figure 1.6: The rewrite system RBV corresponding to BV.

and inference rules are turned into rewrite rules. Each inference rule occurring in BV as shown in Figure 1.2 except $\circ\downarrow$ is turned into a rewrite rule as shown in Figure 1.6 by dropping the context S. As before, $\mathfrak{ai}\downarrow$ is a schema for all positive atoms a.

Proposition 2. *Let s and t be two Σ_{BV}-terms or structures, where t is in negation normal form. (i) There is a derivation in BV from s to t having length n iff there exists a rewriting $s \overset{*}{\rightarrow}_{R_{Neg}} s' \overset{n}{\rightarrow}_{\mathsf{RBV/EBV}} t$. (ii) There is a proof of s in BV having length n iff there exists a rewriting $s \overset{*}{\rightarrow}_{R_{Neg}} s' \overset{n}{\rightarrow}_{\mathsf{RBV/EBV}} \circ$.*

Sketch of Proof The proof of (i) follows immediately from the discussion in this and the previous subsections and Lemma 1, by induction on the length of the derivation in BV and on the number of rewrite steps in RBV/EBV, respectively, for the if part and the only if part, respectively. (ii) follows immediately from (i). □

4 Implementation in Maude

The language Maude [4] allows implementing term rewriting systems modulo equational theories due to the built in very fast matching algorithm that supports different combinations of associative, commutative equational theories, also with the presence of units. Another important feature of Maude

that makes it a plausible platform for implementing systems of the calculus of structures is the availability of the search function since the 2.0 release of Maude. This function implements breadth-first search which is vital for complete search for derivations and proofs.

The *Maude system module* in Figure 1.7 implements the system RBV modulo EBV where the equalities for associativity, commutativity and unit become operator attributes "`assoc`", "`comm`" and "`id : o`". The module presumes that the Σ_{BV}-terms are in negation normal form. To get the negation normal form of a Σ_{BV}-term, we can employ a functional module with the *Maude equations* on the right-hand-side of Figure 1.4.

```
mod BV is
 sorts Atom Unit Structure .
 subsort Atom < Structure .
 subsort Unit < Structure .

  op o  : -> Unit .
  op -_      : Atom -> Atom [ prec 50 ] .
  op [_,_]   : Structure Structure -> Structure [assoc comm id: o] .
  op {_,_}   : Structure Structure -> Structure [assoc comm id: o] .
  op <_;_>   : Structure Structure -> Structure [assoc id: o] .
  ops a b c d e : -> Atom .

  var R T U V : Structure .      var A : Atom .

 rl [ai-down] : [ A , - A ]               => o .
 rl [s]       : [ { R , T } , U ]         => { [ R , U ] , T } .
 rl [q-down]  : [ < R ; T > , < U ; V > ] => < [R,U] ; [T,V] > .
endm
```

Figure 1.7: The system module that implements BV.

We can then use the Maude 2 `search` command for searching for proofs or derivations: `search [- c,[< a ; {c,- b} >,< - a ; b >]] =>+ o .` After a successful search, we can display the proof steps by using the command "`show path <state_number_displayed> .`".

```
Maude> search [- c,[< a ; {c,- b} >,< - a ; b >]] =>+ o .
search in BV : [- c,[< a ; {c,- b} >,< - a ; b >]] =>+ o .

Solution 1 (state 2229)
states: 2230  rewrites: 196866 in 930ms cpu (950ms real) (211683
    rewrites/second)
empty substitution

No more solutions.
states: 2438  rewrites: 306179 in 1460ms cpu (1490ms real) (209711
```

```
    rewrites/second)
Maude> show path 2229 .
state 0, Structure: [- c,[< a ; {c,- b} >,< - a ; b >]]
===[ rl [< R ; T >,< U ; V >] => < [R,U] ; [V,T] > [label q-down] . ]===>
state 20, Structure: [- c,< [a,- a] ; [b,{c,- b}] >]
===[ rl [A,- A] => o [label ai-down] . ]===>
state 178, Structure: [b,[- c,{c,- b}]]
===[ rl [U,{R,T}] => {T,[R,U]} [label s] . ]===>
state 634, Structure: [b,{- b,[c,- c]}]
===[ rl [A,- A] => o [label ai-down] . ]===>
state 1492, Structure: [b,- b]
===[ rl [A,- A] => o [label ai-down] . ]===>
state 2229, Unit: o
```

It is also possible to display all the one step rewrites of a Σ_{BV}-term by using the Maude command "`search <term> =>1 R .`".

5 Planning within BV

In [14], we present an encoding of the conjunctive (multiset rewriting) planning problems (see e.g. [9]) in the language of BV, where plans are not extracted from the proof of a planning problem, but are explicit premises of derivations, which result from bottom-up search. However, in such an encoding, being restricted to BV, while going up in a derivation, the actions in the problem structure at the conclusion of the derivation must be used precisely once. In order to overcome this, there, we employ system NEL [11], the extension of BV with the exponentials of linear logic, to express the availability of actions arbitrarily many times.

In [3], Bruscoli showed that there is a correspondence between system BV and a fragment of CCS [16]: the sequential composition corresponds to the non-commutative operator *seq*. Parallel composition is naturally mapped to the commutative linear logic operator *par*. However, as it is the case in CCS, there only the actions (labels) are included in the language, but not the resources that are consumed and produced by the actions.

Similar to [3], by exploiting the non-commutative operator of system BV, and the commutative logical operator *par*, we are able to observe concurrent plans, where the parallelism between plans is respected. Since our encoding is propositional, no unification mechanism is needed. This allows system NEL to give the complete operational semantics of our method, and establish the first step of a uniform formalism that connects concurrency and planning. This way, it becomes possible to transfer methods from concurrency to planning.

6 Discussions

In this paper, we showed that system BV of the calculus of structures can be expressed as a term rewriting system which can be implemented in Maude for automated proof search and automated application of inference rules. This way, we have also provided a tool for implementing a fragment of CCS which was shown to be equivalent to BV in [3].

We observed that orienting the equalities for *unit* by modifying the inference rules to preserve completeness causes a gain in efficiency in proof search. In [15] we present equivalent systems to system BV where equalities for unit become redundant. Furthermore, due to the non-deterministic application of inference (rewrite) rules, often there are several rewritings of a structure, but in general, only a few of them lead to a proof. A similar problem was solved in [21] by employing the conditional rules of Maude and by means of a strategy at the meta-level [7].

The methods presented in this paper can be analogously applied to the existing systems in the calculus of structures for classical logic [2] and linear logic [19], since these systems can also be expressed as term rewriting systems [12]. However, termination of proof search in our implementation is a consequence of BV being a multiplicative logic. For the logics with an additive behavior, e.g., classical logic, some strategy must be introduced. Different Maude modules for the systems in the calculus of structures, including BV, classical logic and linear logic, are available for download at `http://www.informatik.uni-leipzig.de/ozan/maude_cos.html`.

Carrying our results to full linear logic is of particular interest, since the sequent calculus presentation of linear logic was previously encoded into rewriting logic within Maude modules (see, e.g., [6]). However those modules are not directly executable, in particular due to the promotion rule: in contrast to the calculus of structures, in the sequent calculus, promotion rule requires a global view of the formulae, which makes it difficult to express as an implementable rewriting rule.

Acknowledgments This work has been supported by the DFG Graduiertenkolleg 446. I am grateful to Alessio Guglielmi, Steffen Hölldobler, Steven Eker, Lutz Straßburger and the members of the proof theory group at the International Center for Computational Logic at the TU Dresden. I would like to thank anonymous referees for valuable remarks and improvements.

Bibliography

[1] Franz Baader and Tobias Nipkow. *Term Rewriting and All That*, volume 1. Cambridge University Press, 1998.

[2] Kai Brünnler. *Deep Inference and Symmetry in Classical Proofs*. PhD thesis, Technische Universität Dresden, 2003.

[3] Paola Bruscoli. A purely logical account of sequentiality in proof search. In Peter J. Stuckey, editor, *Logic Programming, 18th Int. Conference*, volume 2401 of *Lecture Notes in Comp. Science*, pages 302–316. Springer-Verlag, 2002.

[4] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. Talcott. The Maude 2.0 system. In Robert Nieuwenhuis, editor, *Rewriting Techniques and Applications, Proceedings of the 14th International Conference*, volume 2706. Springer, 2003.

[5] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. Talcott. Maude 2.1 manual. Technical report, Computer Science Laboratory, SRI International, 2004. http://maude.cs.uiuc.edu/manual/.

[6] Manuel Clavel. *Reflection in Rewriting Logic: Metalogical Foundations and Metaprogramming Applications*. CSLI Publications, 2000.

[7] Manuel Clavel, Francisco Durán, Steven Eker, José Meseguer, and Mark-Oliver Stehr. Maude as a formal meta-tool. In Jeannette M. Wing, Jim Woodcock, and Jim Davies, editors, *FM'99 — Formal Methods, World Congress on Formal Methods in the Development of Computing Systems, Toulouse, France, September 20–24, 1999 Proceedings, Volume II*, volume 1709 of *Lecture Notes in Computer Science*, pages 1684–1703. Springer, 1999.

[8] Melvin C. Fitting. *First–Order Logic and Automated Theorem Proving*. Springer, Berlin, 2nd edition, 1996.

[9] G. Große, S. Hölldobler, and J. Schneeberger. Linear deductive planning. In *Journal of Logic and Computation*, volume 6 (2), pages 233–262. 1996.

[10] Alessio Guglielmi. A system of interaction and structure. Technical Report WV-02-10, TU Dresden, 2002. to appear in ACM Transactions on Computational Logic.

[11] Alessio Guglielmi and Lutz Straßburger. A non-commutative extension of MELL. In M. Baaz and A. Voronkov, editors, *LPAR 2002*, volume 2514 of *Lecture Notes in Artificial Intelligence*, pages 231–246. Springer-Verlag, 2002.

[12] S. Hölldobler and O. Kahramanoğulları. From the calculus of structures to term rewriting systems. Technical Report WV-04-03, TU Dresden, 2004.

[13] Steffen Hölldobler. *Logik und Logikprogrammierung*. Synchron Publishers GmbH, second, extended edition, 2001.

[14] Ozan Kahramanoğulları. Plans as formulae with a non-commutative operator. Technical report, TU Dresden, 2004. submitted.

[15] Ozan Kahramanoğulları. System BV without the equalities for unit. Technical report, TU Dresden, 2004.

[16] Robin Milner. *Communication and Concurrency*. International Series in Computer Science. Prentice Hall, 1989.

[17] David A. Plaisted. Equational reasoning and term rewriting systems. In Dov Gabbay, Christopher Hogger, and J. A. Robinson, editors, *The Handbook of Logic in Artificial Intelligence and Logic Programming, Volume 1: Deductive Methodologies*, pages 274–367. Oxford University Press, Oxford, 1993.

[18] Charles Stewart and Phiniki Stouppa. A systematic proof theory for several modal logics. Technical Report WV-03-08, TU Dresden, 2003. Submitted.

[19] Lutz Straßburger. *Linear Logic and Noncommutativity in the Calculus of Structures*. PhD thesis, TU Dresden, 2003.

[20] Alwen Fernanto Tiu. Properties of a logical system in the calculus of structures. Technical Report WV-01-06, Technische Universität Dresden, 2001.

[21] A. Verdejo and N. Martí-Oliet. Implementing CCS in Maude 2. In Fabio Gadducci and Ugo Montanari, editors, *Proceeding Fourth Int. Workshop on Rewriting Logic and its Applications, WRLA 2002, Pisa, Italy, 2002*, volume 71 of *Electronic Notes in Theoretical Computer Science*. Elsevier, 2002.