

Automatic Proof Search in Logic of Justified Common Knowledge

Yegor Bryukhov

Graduate Center, City University of New York
365 Fifth Avenue, New York, NY 10016 ^{*}
ybryukhov@gc.cuny.edu

Abstract. We consider the logic of justified common knowledge $S4_n^J$ introduced in [Artemov, 2004]. This system captures the notion of *justified common knowledge*, which is free of some of the deficiencies of the usual common knowledge operator, and yet sufficient for analysis of epistemic problems where common knowledge has been traditionally applied. In particular, $S4_n^J$ enjoys cut-elimination, which opens a possibility for an automatic proof search in logic of common knowledge. In this paper, we present an implementation that automatically builds cut-free proofs in $S4_n^J$. Our work is based on the existing matrix-based prover for intuitionistic logic [Schmitt *et al.*, 2001].

1 Introduction

Plato defines knowledge as *justified true belief*. The modal logic approach to knowledge, developed in [Hintikka, 1961; Hintikka, 1962], captures the *true belief* components of Plato’s tripartite definition. The *justification* component was introduced into formal epistemology in [Artemov and Nogina, 2005; Artemov, 2004].

1.1 Common Knowledge

Common knowledge is a standard notion in formal epistemology. If K_i ($i = 1, \dots, n$) are knowledge operators of individual agents and

$$E\varphi = K_1\varphi \wedge \dots \wedge K_n\varphi,$$

we can informally define common knowledge as:

$$C\varphi \leftrightarrow \varphi \wedge E\varphi \wedge \dots \wedge E^m\varphi \wedge \dots$$

The standard way of defining common knowledge axiomatically is via the *Fixed-Point Axiom*:

$$C\varphi \leftrightarrow E(\varphi \wedge C\varphi)$$

^{*} The work was actually done while I was visiting Prof. Jason Hickey at CalTech

together with the *Induction Rule*:

$$\frac{\varphi \rightarrow E(\psi \wedge \varphi)}{\varphi \rightarrow C\psi}$$

This formalization does not behave well proof-theoretically; in particular, the above axiomatization does not admit cut-elimination [Alberucci and Jäger, 2005] – an automatic proof search becomes problematic.

An alternative evidence-based approach to common knowledge was introduced in [Artemov, 2004], using the new expressive power of the logics of knowledge with justification. The basic new atoms in the latter are explicit evidence operators of type $t : \varphi$, where φ is a sentence and t is an evidence term that uses a certain manageable system of operations on justification. New evidence operators $t : \varphi$ are completely described by the Logic of Proofs LP and provide an axiomatic and model description of epistemic assertions

t is a justification for φ .

We can also consider a forgetful projection of evidence that replaces each evidence assertion $t : \varphi$ by a sentence $J\varphi$, where J is a new modality of *justified knowledge*. [Artemov, 2004] considers, among others, the logic $S4_n^J$ with n $S4$ -modalities K_i denoting knowledge of n agents and an additional $n + 1$ -st $S4$ -modality J , together with the axiom scheme $J\varphi \rightarrow K_i\varphi$. It has been established in [Artemov, 2004] that $S4_n^J$ is indeed a forgetful projection of the corresponding logic of knowledge with justification. In particular, [Artemov, 2004] gives a Realization Algorithm that given an $S4_n^J$ -derivation, it recovers explicit evidence terms for each occurrence of the justified knowledge modality J . These results support understanding of the justified knowledge statement $J\varphi$ as

there is an access to a justification of φ .

The logic $S4_n^J$ is a typical justified knowledge system which is also convenient to compare justified knowledge – represented by $S4_n^J$ – with the corresponding common knowledge system $S4_n^C$ based on the Fixed-Point Axiom together with the Induction Rule.

Here are the main properties of the justified knowledge system $S4_n^J$ established in [Artemov, 2004].

Fixed-Point Axiom The justified knowledge modality J satisfies the Fixed-Point Axiom of common knowledge in this form:

$$J\varphi \leftrightarrow E(\varphi \wedge J\varphi).$$

Connection with Common Knowledge Every evidence-based principle is a common knowledge principle but not vice versa:

$$(S4_n^J)^* \subset S4_n^C$$

where $*$ replaces every J -modality with C -modality.

Cut-Elimination $S4_n^J$ enjoys cut-elimination.

Hence Justified Knowledge is a well-behaved, constructive version of Common Knowledge, which can be used as the latter in solving specific problems (cf. [Artemov, 2004]).

In this paper we describe an implemented automatic proof search procedure for $S4_n^J$ which produces cut-free proofs. Given a cut-free proof in $S4_n^J$ one can use the Realization Algorithm to recover evidence terms in this proof.

We successfully formalized, and $S4_n^J$ -prover proved the *Wise Men* and *Muddy Children* puzzles [Fagin *et al.*, 1995]. The Muddy Children Puzzle was formulated in several flavors for four children, and depending on its formulation the proof search and verification time varied from under one second to 90 seconds.

Our $S4_n^J$ -prover is based on existing implementation for intuitionistic logics. The cases of **T** and **S5** as the base knowledge systems instead of **S4** are left for future work. The Realization Algorithm is currently implemented for **S4**, but we leave it outside the scope of this paper, since we still need to extend it to the full $S4_n^J$ case and connect it to our $S4_n^J$ -prover.

1.2 Existing Code for First-Order Intuitionistic Logic

We based our work on the existing code of the J-prover in the MetaPRL logical framework [Schmitt *et al.*, 2001]. The implementation is based on a uniform algorithm and matrix characterization for the first order intuitionistic logic J, modal logics K, K4, T, S4, S5, and fragments of linear logic were given [Otten and Kreitz, 1996b; Kreitz and Otten, 1999]. It also contains an algorithm for conversion of an existing matrix proof into a sequent proof [Schmitt and Kreitz, 1995; Kreitz and Schmitt, 2000; Schmitt, 1999].

Note that J here stands for first order intuitionistic logic and has nothing to do with “J” in $S4_n^J$ where it stands for “justified” S4-modality.

The code we started from supported only the first order intuitionistic logic J. Our work is about classical modal logic and does not involve intuitionistic logic. We generalize the matrix characterization of **S4** to $S4_n^J$, and extend that code to support matrix proof search and sequent proof reconstruction for $S4_n^J$.

We also improve the prefix unification algorithm [Otten and Kreitz, 1996a], which is an important part of the matrix proof search algorithm: on some problems it consumes as much as 99% of the total proof search and construction time. The unification is used iteratively with incremental extension of the equation system. Original algorithm was solving new systems from scratch each time. We modified it to reuse the information obtained from the previous iteration on the next iteration when one more equation was added. For problems with a unique most-general unifier, one has only to retain the unifier which itself represents the solved system. Unfortunately, prefix unification problems do not have unique solutions but rather minimal sets of unifiers. Naturally, not all members of a unifier set U for system S survive as members of a unifier set for system $S \cup p = q$.

The algorithm terminates when any unifier is found, but its full state contains the partially explored decision tree which we have to keep for the next iteration. Passing the unexplored part of decision tree from iteration to iteration reduced the running time by at least an order of magnitude on difficult problems.

The rest of the paper is organized in the following way. In Section 2, we introduce the matrix characterization of logical validity for propositional classical logic [Bibel, 1981; Bibel, 1987; Andrews, 1981]. In Section 3, we extend this notion to $S4$ [Wallen, 1990]. Section 4 contains our extension of the matrix method to $S4_n^J$, and Section 5 describes modifications to the prefix unification algorithm, which is a part of matrix method. Section 6 briefly describes the main ideas of matrix proof to sequent proof conversion [Kreitz and Schmitt, 2000; Schmitt, 1999] and our modifications to enable support of $S4_n^J$. And finally, Section 7 describes the Muddy Children Puzzle and our solution to it.

2 Matrix Characterization of Logical Validity

The connection method was introduced in [Bibel, 1981; Bibel, 1987] for classical logic. In [Wallen, 1990] it was extended to a number of non-classical logics. Kreitz *et al* suggested a uniform algorithm for a variety of modal logics, intuitionistic logic, and fragments of linear logic [Otten and Kreitz, 1996b; Kreitz and Otten, 1999]; described a uniform method of reconstruction of Gentzen sequent proofs from matrix proofs [Schmitt and Kreitz, 1995; Kreitz and Schmitt, 2000; Schmitt, 1999]; and implemented it for intuitionistic logic in the MetaPRL proof assistant [Schmitt *et al.*, 2001].

In this section, we will introduce the basic concepts of matrix method and describe logical validity in these terms. We will limit our attention to propositional case because our target logic $S4_n^J$ is propositional.

A *formula tree* of a formula F is basically its syntax tree. Each node s corresponds to one particular subformula F_s . We give unique names $a_1, a_2, \dots$ to each node and call them *positions*. A *label* of a position s is the major (outermost) connective of F_s or is itself F_s if it is atomic. In the latter case, s is called *atomic position* or simply *atom*. The *tree ordering* $<$ on positions is induced by the tree structure where the root is the smallest element with respect to this tree ordering $<$.

Each position is associated with *polarity* in a standard way with the root having polarity 0.

The *principal type* $Ptype(u)$ of a position u is defined by its label and polarity according to the following table:

principal type α	$(A \wedge B)^1$	$(A \vee B)^0$	$(A \rightarrow B)^0$	$(\neg A)^1$	$(\neg A)^0$
subformulae polarities	A^1, B^1	A^0, B^0	A^1, B^0	A^0	A^1
principal type β	$(A \wedge B)^0$	$(A \vee B)^1$	$(A \rightarrow B)^1$		
subformulae polarities	A^0, B^0	A^1, B^1	A^0, B^1		

Atomic positions have no principal type.

Definition 1. For a principal type t , we say that two positions a and b of F are t -related ($a \sim_t b$) if the $<$ -biggest position c of F that is less than a and b has principal type t ($c = \max\{x \mid x < a \wedge x < b\}$). a is t -related to a set S iff $a \sim_t s$ for all $s \in S$.

Definition 2. For each principal type t , there are **secondary types** $t_0, \dots, t_k$ where k is the arity of connectives that produce type t . For position u of principal type t , its $<$ -successors have secondary types t_i according to their relative order as arguments of u . We denote the secondary type of a position v as $Stype(v)$. The root has no secondary type.

The *matrix(-representation)* of a formula F is a two-dimensional representation of its atomic subformulae without connectives (but with parentheses); it is mainly used for illustrative purposes. In this representation, α -related positions are arranged side by side; β -related positions are arranged one on top of another.

Example 1. The matrix representation of the formula $\neg((\neg A \vee \neg C) \wedge ((A \wedge B) \vee (B \wedge C)))$ is:

$$\begin{pmatrix} A^0 \\ C^0 \end{pmatrix} \begin{pmatrix} (A^1 \ B^1) \\ (B^1 \ C^1) \end{pmatrix}$$

Definition 3. A **path** through a formula F is a maximal set of α -related positions. It can be visualized as a maximal “horizontal” line through the matrix of F (or more precisely, a maximal line that never runs vertically).

Definition 4. A **connection** is a pair of positions with the same labels but opposite polarities.

Theorem 1. A propositional formula F is (classically) valid iff every path through F has a connection.

A proof of this theorem (first-order case) can be found in [Bibel, 1987].

3 Matrix characterization for S4

The matrix characterization for S4 was given in [Wallen, 1990]. For this, we need to add two extra principal types ν , π for modal positions; define prefixes and unification over them; define multiplicity and what is an indexed formula; and define complementary connections and admissible substitutions.

We will use only one modality $\Box$ because $\Diamond$ has no explicit counterpart in the logic of knowledge with justification in which we want to realize $S4_n^J$ -theorems.

Two new principal types are defined according to the table below:

principal type ν	$(\Box A)^1$
subformula polarity	A^1
principal type π	$(\Box A)^0$
subformula polarity	A^0

We denote the set of all positions of primary type ν as $\mathcal{V}$, of primary type π as Π , all positions of secondary type ν_0 as $\mathcal{V}_0$, and of secondary type π_0 as Π_0 .

Definition 5. For a position u , the **modal prefix** $\text{pre}_M(u)$ is a string $u_1 \dots u_n$ of all positions $u_1 < u_2 < \dots < u_n \leq u$ such that $u_i \in \mathcal{V}_0 \cup \Pi_0$.

Definition 6. The modal substitution $\sigma : \mathcal{V}_0 \rightarrow (\mathcal{V}_0 \cup \Pi_0)^*$ induces a relation $\sqsubset_M \subseteq (\mathcal{V}_0 \cup \Pi_0) \times \mathcal{V}_0$: if $\sigma_M(u) = p$, then $v \sqsubset u$ for all $v \in \mathcal{V}_0 \cup \Pi_0$ occurring in p .

In other words, ν_0 -positions are considered to be variables and π_0 -positions are considered to be constants. All variables in the result of $\sigma(u)$ are less than u with respect to the ordering induced by σ .

Definition 7. A modal substitution σ is **S4-admissible** iff the induced reduction ordering $\triangleleft = (< \cup \sqsubset)^+$ is irreflexive.

In S4-tableaux, ν -formulae can be used several times to populate several prefixes (worlds); in order to reflect it in the matrix characterization, we introduce modal multiplicity $\mu : \mathcal{V}_0 \rightarrow \mathbb{N}$ and define *indexed formula* F^μ which is just a pair – a formula and a multiplicity function over it. A position $u \in \mathcal{V}$ has $\mu(u)$ instances of its successor trees with root v , $u < v$. Whenever we refer to index formula, we imply that positions are duplicated according to the multiplicity function.

Definition 8. A connection $\{u, v\}$ is **complementary** under a substitution σ iff $\sigma(\text{pre}(u)) = \sigma(\text{pre}(v))$. A path is *complementary* iff it contains a complementary connection.

Definition 9. A set of connections C **spans** the formula A iff any path through A contains some connection from C .

Theorem 2. A formula is S4-valid iff there is a multiplicity μ , an S4-admissible substitution σ , and a set of σ -complementary connections C that spans A [Wallen, 1990].

4 Matrix Characterization of S4_n^J

S4_n^J can be described as n modalities K_i with S4 behavior, modality J with S4 behavior and the connection axiom schema

$$Jp \rightarrow K_i p, \text{ for all } i. \quad (4.I)$$

To extend matrix characterization from singular S4 to several such modalities, one has to label tree positions with the appropriate modalities and make such positions incompatible for prefix unification purposes.

In terms of tableau rules, the connection axiom 4.I can be expressed as

$$\frac{T \sigma \quad Jp}{T \sigma.a_i p} \text{ for an existing prefix } \sigma.a_i$$

This means that J -modality is compatible with worlds generated by any modality $K_1, \dots, K_n, J$. For unification purposes, we allow variable positions related to J -modalities to unify with any string.

We will mark position a with a superscript i if it is related to modality K_i . For J we will use superscript 0.

Definition 10. For a position a^i , we define $\text{sort}\{a^i\} = i$

Definition 11. To describe prefix unification for $\mathbf{S4}_n^J$, we define a relation $\preceq$ on positions. For two positions a and b , we say that $a \preceq b$ iff a is ν_0^0 or a is ν_0^i , and b is ν_0^i or π_0^i . Basically, it states when b can be substituted for a .

Corollary 1. $\preceq$ is reflexive and transitive.

Definition 12. A substitution $\{V_1 \setminus s_1, \dots, V_n \setminus s_n\}$ is $\mathbf{S4}_n^J$ -admissible iff it is $\mathbf{S4}$ -admissible and for all i and all a from s_i , holds $V_i \preceq a$.

Theorem 3. A formula is $\mathbf{S4}_n^J$ -valid iff there is a multiplicity μ , an $\mathbf{S4}_n^J$ -admissible substitution σ , and a set of σ -complementary connections C that spans A .

Proof. (Sketch) The only difference between $\mathbf{S4}$ and $\mathbf{S4}_n^J$ tableaux are the rules of propagation of $\Box$ from prefix to prefix. Our adjustments in the definition of admissible substitution reflects those differences exactly.

We can modify our definition of a path through a formula A^μ as follows. We say that some sets of α -related positions are paths (and our original paths become atomic paths). Instead of $P \cup \{v\}$, we will write P, v .

- (A) Root is a path.
- (B) If P, u is a path, and u is an α -position with subformulae u_0 and u_1 , then P, u_0 and P, u_1 are paths as well.
- (C) If P, u is a path, and u is a β -position with subformulae u_0 and u_1 , then P, u_0, u_1 is a path.
- (D) If P, u is a path, and u is a π -position with subformula u_0 , then P, u_0 is a path.
- (E) If P, u is a path, and u is a ν -position with subformula u_0 , then $P, u_0, 1, \dots, u_0, \mu(u)$ is a path where u_0, i are instances of u_0 .

If we augment each position in paths with its prefixes, paths will become $\mathbf{S4}_n^J$ -tableau branches. Atomic paths will be closed branches (if there is a spanning set of connections). Our path generation rules become valid tableau rules as long as prefixes are introduced by π -rules before they are used by ν -rules. But we can always apply the ν -rule after the appropriate π -rule – they cannot block each other because their active positions are from different branches of the syntax tree. Hence if there is a matrix proof then there is a tableaux proof as well.

Once we have a tableau proof, we can find the appropriate multiplicity and an $S4_n^J$ -admissible substitution. Each tableau branch represents one or many atomic paths, each closed branch has a complementary connection, and all branches of a closed tableau cover all atomic paths.

Note Our implementation does not use tableaux in any way, we refer to them only for the sake of the proof(-sketch).

[Cerrito and Mayer, 1997] give an upper bound for multiplicity function for a number of propositional modal logics, including $S4$. We tried to implement it but it proved to be unpractically high.

5 Prefix Unification for $S4_n^J$

[Otten and Kreitz, 1996a] present a prefix unification algorithm for a number of logics, including $S4$ (the algorithm is the same but it uses different sets of rules for different logics).

We consider strings over $\mathcal{V} \cup \mathcal{C}$ where $\mathcal{V}$ serves as a set of variables and $\mathcal{C}$ is a set of constants. Symbols are actually positions, so for the $S4_n^J$ case, we assume that each symbol has a sort : $\mathcal{V} \cup \mathcal{C} \rightarrow \mathbb{N}$ which we denote as a superscript, and we have a sort compatibility relation $\prec$ over symbols.

Definition 13 (T-string property). *Two strings a and b over an alphabet $\mathcal{A}$ have **T-string property** iff*

- (A) $\forall_{1 \leq i, j \leq |s|} . i \neq j \rightarrow s_i \neq s_j$ and $\forall_{1 \leq i, j \leq |t|} . i \neq j \rightarrow t_i \neq t_j$
- (B) $\exists_{0 \leq k \leq \min(|s|, |t|)} . (\forall_{1 \leq i \leq k} . s_i = t_i \& \forall_{k < i \leq |s|} \forall_{k < j \leq |t|} . s_i \neq t_j)$

In other words, these strings form a tree.

Definition 14. *A set of strings S has **T-string property** iff all pairs of strings $s, t \in S$ have T-string property.*

Definition 15. *A system of equations $\{s_1 = t_1, \dots, s_n = t_n\}$ has T-string property iff the set $\{s_1, \dots, s_n, t_1, \dots, t_n\}$ has T-string property.*

Definition 16. *A substitution σ is a **T-unifier** of a system of equations $\{s_i = t_i | 1 \leq i \leq n\}$ iff $\sigma s_i = \sigma t_i$ for all $1 \leq i \leq n$.*

Definition 17. *A substitution σ is an **instance** of τ if there is δ such that $\sigma = \delta \circ \tau$ (δ applied after τ).*

Definition 18. *A set of substitutions Σ is a **(minimal) set of most general unifiers** for a system of equations Γ iff:*

Correctness *Every $\sigma \in \Sigma$ is a T-unifier for Γ .*

Completeness *Every T-unifier τ for Γ is an instance of some $\sigma \in \Sigma$.*

Minimality *No $\sigma \in \Sigma$ is an instance of another $\sigma' \in \Sigma$.*

A nondeterministic version of the unification algorithm is the following. It assumes that each equation in Γ has the form $s = r|t$ where $r = \varepsilon$ at the beginning:

```

while  $\Gamma \neq \emptyset$  do
  select the left most system  $s = t \in \Gamma$ 
  set  $\Gamma$  to  $\Gamma \setminus \{s = t\}$ 
  select a transformation rule  $\Pi \rightarrow \Delta, \Theta$  from  $\mathcal{T}$  which is applicable to  $s = t$ ;
  if no rule is applicable stop with failure
  apply rule  $\Pi \rightarrow \Delta, \Theta$  to  $s = t$ , let  $\Delta', \Theta'$  be the results
   $\Gamma := \Theta'(\Gamma)$ ,  $\sigma := \Theta(\sigma)$ 
   $\Gamma := \Delta' \cup \Gamma$ ,  $\sigma = \sigma \cup \Theta'$ 
stop with success and return  $\sigma$ .

```

The set of most general unifiers is a result of all successful runs of the algorithm.

Definition 19. For two variable positions a^i and b^j , we say that $a^i \simeq b^j$ iff $i = j$ or one of them is 0.

Corollary 2. $\simeq$ is an equivalence relation.

If $a^i \simeq b^j$, then the least constraining sort of variables that can be substituted for both a^i and b^j is $\max(i, j)$. Indeed, if $\max(i, j)$ is positive, then it is the only possible sort; if it is 0, then it can be replaced with any other sort, but if we choose any positive sort, we will then be bound to it.

The original list of rules for T-unification for **S4** described in [Otten and Kreitz, 1996a] changes to the one described in Table 1. There we added the Rules 11 through 14 and sort compatibility conditions to the Rules 7 through 10.

Corollary 3. (T-String Preserving) Let Γ be system of equations with T-string property and σ be a most general unifier for one of its equations, then a system of equations $\sigma\Gamma$ still has T-string property.

Definition 20. Let $\text{mgu}(\Gamma)$ be the set of all substitutions produced by the given algorithm with the given set of rules.

Corollary 4. (Correctness) Let $\Gamma = \{s = \varepsilon|t\}$ be an equation of T-strings. Then all $\sigma \in \text{mgu}(\Gamma)$ are T-unifiers for Γ .

Proof is the same as in [Otten and Kreitz, 1996a].

Theorem 4. (Termination) Let $\Gamma = \{s = \varepsilon|t\}$ be an equation of T-strings. Then the given algorithm with the given set of rules always terminates.

Proof. [Otten and Kreitz, 1996a] prove it by defining an ordering on equations and showing that we need at most two rule applications to obtain a smaller equation with respect to that ordering. This proof works for our set of rules as well.

R1	$\{\varepsilon = \varepsilon \varepsilon\}$	$\rightarrow \{\}, \{\}$
R2	$\{\varepsilon = \varepsilon t^+\}$	$\rightarrow \{t^+ = \varepsilon \varepsilon\}, \{\}$
R3	$\{Xs = \varepsilon Xt\}$	$\rightarrow \{s = \varepsilon t\}, \{\}$
R4	$\{Cs = \varepsilon Vt\}$	$\rightarrow \{Vt = \varepsilon Cs\}, \{\}$
R5	$\{Vs = z \varepsilon\}$	$\rightarrow \{s = \varepsilon \varepsilon\}, \{V \setminus z\}$
R6	$\{Vs = \varepsilon C_1t\}$	$\rightarrow \{s = \varepsilon C_1t\}, \{V \setminus \varepsilon\}$
R7	$\{Vs = z C_1C_2t\}$	$\rightarrow \{s = \varepsilon C_2t\}, \{V \setminus zC_1\}$, where $V \preceq C_1$
R8	$\{Vs^+ = \varepsilon V_1t\}$	$\rightarrow \{V_1t = V s^+\}, \{\}$, where $V_1 \preceq V$
R9	$\{Vs^+ = z^+ V_1t\}$	$\rightarrow \{V_1t = V' s^+\}, \{V \setminus z^+V'\}$, where $V \simeq V_1$, and $\text{sort}\{V'\} = \max(\text{sort}\{V\}, \text{sort}\{V_1\})$
R10	$\{Vs = z Xt\}$	$\rightarrow \{Vs = zX t\}, \{\}$, where $V \preceq X$, and $V \neq X$, and $s = \varepsilon$ or $t \neq \varepsilon$ or $X \in \mathcal{C}$
R11	$\{Vs = z V_1t\}$	$\rightarrow \{s = \varepsilon V_1t\}, \{V \setminus z\}$, where $(z \neq \varepsilon \text{ and } V \not\preceq V_1) \text{ or } (z = \varepsilon \text{ and not } V \simeq V_1)$
R12	$\{V = \varepsilon V_1t\}$	$\rightarrow \{V = \varepsilon t\}, \{V_1 \setminus \varepsilon\}$, where $V_1 \preceq V$ but $V \not\preceq V_1$
R13	$\{V = \varepsilon V_1t\}$	$\rightarrow \{V_1t = V' \varepsilon\}, \{V \setminus V'\}$, where $V_1 \preceq V$ but $V \not\preceq V_1$, and $\text{sort}\{V'\} = \text{sort}\{V\}$
R14	$\{Vs = zV_1 Ct\}$	$\rightarrow \{s = \varepsilon Ct\}, \{V \setminus zV_1\}$, where $V \not\preceq C$

Table 1. s, t and z denote (arbitrary) strings, and s^+, t^+, z^+ non-empty strings. X, V, V_1, C, C_1 , and C_2 denote single characters with $X \in \mathcal{V} \cup \mathcal{C} \cup \mathcal{V}'$, $V, V_1 \in \mathcal{V} \cup \mathcal{V}'$ (with $V \neq V_1$), and $C, C_1, C_2 \in \mathcal{C}$. $V' \in \mathcal{V}'$ is a new variable which does not occur in the substitution σ computed so far.

[Otten and Kreitz, 1996a] claim minimality of the resulting unifier sets for their S4-set of rules. We do not claim it for our rules. See our practical considerations at the end of this section.

Theorem 5. (Completeness) *Let $\Gamma = \{s = \varepsilon|t\}$ be an equation of T-strings. $\text{mgu}(\Gamma)$ is complete, i.e. any unifier of Γ is an instance of some $\sigma \in \text{mgu}(\Gamma)$.*

Proof. [Otten and Kreitz, 1996a] claim completeness of the original S4-set of rules. Completeness of our extended set can be obtained by lengthy analysis of all cases when S4-rules are not applicable due to modal sort compatibility requirements.

5.1 Practical considerations

Informally, T-unification problems are not exactly isomorphic to desired solutions of prefix equations. For example, fresh variables introduced in the Rules 9 and 13 are set to ε after a unifier is found. Substitution $\{V_1 \setminus \varepsilon, V_2 \setminus \varepsilon\}$ is an instance of $\{V_1 \setminus V_2\}$; for proof search purposes they are equally good, but if we still have some equations to solve, the former substitution being applied to them produces a smaller problem. These considerations suggest that minimality and completeness of the resulting unifier sets do not guarantee optimality.

We must also pay close attention to the order in which rules are attempted – swapping Rules 12 and 13 changes running time on certain problems by more

than an order of magnitude, and changes (practical) feasibility of some problems. We plan to measure “popularity” of the rules on a large base of problems and reorder them by decreasing order of popularity.

6 From Matrix Proofs to Sequent Proofs

The algorithm for the conversion of matrix proofs to sequent proofs was given by Kreitz and Schmitt [Kreitz and Schmitt, 2000; Schmitt, 1999] for a number of modal logics (K, K4, T, S4, D, D4), first order intuitionistic logic, and fragments of linear logic.

The full description of the algorithm is lengthy and very technical, so we will highlight only the main idea and details that change with our move from S4 to S4_n^J.

The algorithm goes over the formula tree starting from its root and converts each position into an inference rule (or several inference rules); the result is a derivation tree. Suppose, the original formula that we had to prove was $A \wedge B$ (or $\Rightarrow A \wedge B$ as a sequent). The primary type of the root is α and the main connective is $\wedge$, the algorithm chooses this rule:

$$\frac{\Gamma \Rightarrow A, \Delta \quad \Gamma \Rightarrow B, \Delta}{\Gamma \Rightarrow A \wedge B, \Delta}$$

Positions for A and B become *open* and the algorithm picks next position among open ones.

In general, inference rules are not permutable because certain rules can destroy formulae of a sequent that have not yet been used, hence the order in which the algorithm has to explore the formula tree is important. The primary factor is the reduction ordering induced by unifying substitution.

Definition 21. *We say that the processed positions are **solved**, and that P_a is the set of solved atomic positions.*

Definition 22. *We call **open** a not-solved position whose immediate predecessor in the tree ordering is solved. $P_o = \{x \mid x \text{ not solved} \ \& \ y \prec x \text{ solved}\}$ is the set of open positions.*

Basically, open positions represent current formulas in the sequent.

Definition 23. *If $x \sqsubset y$ (substitution induced relation), we say that x is blocking y . And $W_y = \{x \mid x \sqsubset y\}$ is the set of positions blocking y .*

Open positions that are blocked will not be processed until there is no blocking. There are more requirements for first-order S4 in [Kreitz and Schmitt, 2000; Schmitt, 1999], unfortunately, they do not fully agree.

Note Because generated proofs are always checked by MetaPRL engine, it is not critical if the algorithm has bugs and/or incomplete. There are two worst case scenarios: algorithm runs into a deadlock because of these blocking conditions, or it might produce an incorrect proof, which will be rejected by MetaPRL.

Consider the **S4**-rule for $\Box$ -introduction to the right:

$$\frac{\Gamma^* \Rightarrow \Delta, A}{\Gamma \Rightarrow \Delta, \Box A} \quad (\Rightarrow \Box)$$

Where $\Gamma^* = \{\Box X | \Box X \in \Gamma\}$, it is more convenient to formulate it in this form with $*$ operation when we do a bottom-up proof.

If we apply this rule too early, the $*$ operation might delete some formulae that we will need later in our proof. The conversion algorithm has a complicated condition describing when it is not too early to use this rule.

For **S4**_{*n*}^J, the similar rule is:

$$\frac{\Gamma^* \Rightarrow \Delta, A}{\Gamma \Rightarrow \Delta, \Box A} \quad (\Rightarrow \Box)$$

If $\Box$ is J then $\Gamma^* = \{JA | JA \in \Gamma\}$.

If $\Box$ is K_i then $\Gamma^* = \{JA | JA \in \Gamma\} \cup \{K_i A | K_i A \in \Gamma\}$.

So in the case of **S4**_{*n*}^J, not all boxes survive after this rule but only boxes of the same sort or stronger (i.e. J). In order to reflect this in the proof conversion algorithm for **S4**_{*n*}^J, we impose one more condition when a π -position cannot be processed – *if there is an open ν -position of a different non zero sort*. Otherwise, if we process that π -position, we will apply the $\Rightarrow \Box$ -rule to it and it will delete the incompatible ν -positions.

This modification is the only one needed to add **S4**_{*n*}^J support to the existing proof conversion algorithm for **S4**.

7 Muddy Children Puzzle

The paradigmatic example of **S4**_{*n*}^J application is the renowned:

Muddy Children Puzzle There are n children playing in the yard; as they play m of them become muddy. Their father enters the yard and says: “at least one of you is muddy, do you know if you are muddy?” Nobody answers, so he asks again: “Do you know if you are muddy?” He asks this question until all the children are able to answer the question.

After how many questions the father will receive the answers?

We assume that all the children are smart and answer simultaneously.

Let’s formalize this puzzle in **S4**_{*n*}^J for four children, two of them muddy. Propositional letter c_i will stand for “ i^{th} child is muddy”. We will use the formalization, that we heard from Melvin Fitting.

Definition 24. *The i^{th} child knows whether A is true:*

$$KW_i A = K_i A \vee K_i \neg A$$

Definition 25. *It is common knowledge that every child knows about other children if they are muddy:*

$$KAO = J(KW_1c_2 \wedge KW_1c_3 \wedge KW_1c_4 \wedge KW_2c_1 \wedge KW_2c_3 \wedge KW_2c_4 \wedge KW_3c_1 \wedge KW_3c_2 \wedge KW_3c_4 \wedge KW_4c_1 \wedge KW_4c_2 \wedge KW_4c_3)$$

Definition 26. *The initial state before the father said anything was*

$$s_0 = c_1 \wedge c_2 \wedge \neg c_3 \wedge \neg c_4 \wedge KAO$$

Definition 27. *The father's statement "At least one of you is muddy" is formalized as*

$$s_1 = J(c_1 \vee c_2 \vee c_3 \vee c_4)$$

Definition 28. *Some child knows whether (s)he is muddy or not:*

$$SK = KW_1c_1 \vee KW_2c_2 \vee KW_3c_3 \vee KW_4c_4$$

Theorem 6. *After the father asked his question first time, nobody answered. Immediately after that, muddy children (c_1 and c_2) realized that they were muddy:*

$$s_0, J(\neg J(s_1 \rightarrow SK)) \Rightarrow K_1c_1$$

and

$$s_0, J(\neg J(s_1 \rightarrow SK)) \Rightarrow K_2c_2$$

- $s_1 \rightarrow SK$ says that after first question some child knew if (s)he is muddy.
- $J(s_1 \rightarrow SK)$ says that it became common knowledge, i.e. that child publicly said that (s)he knows.
- And finally, $J(\neg J(s_1 \rightarrow SK))$ says that it is common knowledge that no child actually answered after the first question.

Proof. The proof was found automatically by our matrix-based prover and verified by MetaPRL, it took approximately 92 seconds in total (but separately for c_1 and c_2).

Unfortunately, Four Children Puzzle represents a real challenge to our implementation. It was not able to find a proof for the Theorem 6 above with $K_1c_1 \wedge K_2c_2$ in conclusion.

8 Final Remarks

We presented a matrix-based prover for $S4_n^J$. It builds cut-free proofs that can be fed to a Realization Algorithm, that would recover evidence terms (justifications) for justified common knowledge modality J , and build an $S4_n^{LP}$ -proof for a realization of the original formula [Artemov, 2004]. It is our immediate goal to finish this Realization Algorithm and connect it to our $S4_n^J$ -prover.

We are not satisfied with the performance of our implementation. One way to address it is to implement non-constant multiplicity function, i.e. to increase

multiplicity of generative positions individually, on demand. Although, it is not obvious how to do it. May be, we should move to tableaux-based algorithms, though we need sequent proofs as the output for the Realization Algorithm.

We also want to extend our work to other modal logics (T_n^J and $\mathsf{S5}_n^J$) [Artemov, 2004] and investigate possible connections between logic of proofs LP (and T_n^{LP} , $\mathsf{S4}_n^{\text{LP}}$, $\mathsf{S5}_n^{\text{LP}}$) and description logics.

We thank Karen Kletter for editing the linguistic aspect of the paper and all the anonymous reviewers for their helpful comments.

References

- Alberucci and Jäger, 2005. Luca Alberucci and Gerhard Jäger. About cut elimination for logics of common knowledge. *Annals of Pure and Applied Logic*, 133(1-3):73–99, 2005.
- Andrews, 1981. Peter B. Andrews. Theorem proving via general matings. *JACM*, 28(2):193–214, 1981.
- Artemov and Nogina, 2005. Sergei Artemov and Elena Nogina. On epistemic logic with justification. In R. van der Meyden, editor, *Theoretical Aspects of Rationality and Knowledge. Proceedings of the Tenth Conference (TARK 2005), June 10-12, 2005, Singapore.*, pages 279–294. National University of Singapore, 2005.
- Artemov, 2004. Sergei Artemov. Evidence-based common knowledge. Technical Report TR-2004018, CUNY Ph.D. Program in Computer Science Technical Reports, November 2004.
- Bibel, 1981. Wolfgang Bibel. On matrices with connections. *JACM*, 28(4):633–645, 1981.
- Bibel, 1987. Wolfgang Bibel. *Automated Theorem Proving*. Vieweg Verlag, Braunschweig, 2nd edition, 1987.
- Cerrito and Mayer, 1997. Serenella Cerrito and Marta Cialdea Mayer. Hintikka multiplicities in matrix decision methods for some propositional modal logics. In *Analytic Tableaux and Related Methods (TABLEAUX 97)*, pages 138–152, 1997.
- Fagin *et al.*, 1995. Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning About Knowledge*. Massachusetts Institute of Technology, 1995.
- Hintikka, 1961. Jaakko Hintikka. Modalities and quantification. *Theoria*, 27:119–128, 1961.
- Hintikka, 1962. Jaakko Hintikka. *Knowledge and Belief*. Cornell University Press, Ithaca, NY, 1962.
- Kreitz and Otten, 1999. Christoph Kreitz and Jens Otten. Connection-based theorem proving in classical and non-classical logics. *Journal for Universal Computer Science, Special Issue on Integration of Deductive Systems*, 5(3):88–112, 1999.
- Kreitz and Schmitt, 2000. Christoph Kreitz and Stephan Schmitt. A uniform procedure for converting matrix proofs into sequent-style systems. *Journal of Information and Computation*, 162(1–2):226–254, 2000.
- Otten and Kreitz, 1996a. Jens Otten and Christoph Kreitz. T-string-unification: Unifying prefixes in non-classical proof methods. In U. Moscato, editor, *5th Workshop on Theorem Proving with Analytic Tableaux and Related Methods*, volume 1071 of *Lecture Notes in Artificial Intelligence*, pages 244–260. Springer Verlag, 1996.
- Otten and Kreitz, 1996b. Jens Otten and Christoph Kreitz. A uniform proof procedure for classical and non-classical logics. In G. Götz and S. Hölldobler, editors,

- KI-96: Advances in Artificial Intelligence*, volume 1137 of *Lecture Notes in Artificial Intelligence*, pages 307–319. Springer Verlag, 1996.
- Schmitt and Kreitz, 1995. Stephan Schmitt and Christoph Kreitz. On transforming intuitionistic matrix proofs into standard-sequent proofs. In P. Baumgartner, R. Hähnle, and J. Posegga, editors, *Theorem Proving with Analytic Tableaux and Related Methods: Proc. of the 4th International Workshop TABLEUX'95*, pages 106–121. Springer, Berlin, Heidelberg, 1995.
- Schmitt *et al.*, 2001. Stephan Schmitt, Lori Lorigo, Christoph Kreitz, and Aleksey Nogin. *JProver*: Integrating connection-based theorem proving into interactive proof assistants. In *International Joint Conference on Automated Reasoning*, volume 2083 of *Lecture Notes in Artificial Intelligence*, pages 421–426. Springer-Verlag, 2001.
- Schmitt, 1999. Stephan Schmitt. *Proof Reconstruction in Classical and Non-classical Logics*. PhD thesis, Technical University of Darmstadt, 1999.
- Wallen, 1990. Lincoln A. Wallen. *Automated deduction in nonclassical logics*. MIT Press, Cambridge, MA, USA, 1990.