Michael A. McRobbie
Paul B. Thistlewaite
Robert K. Meyer

A MECHANIZED DECISION PROCEDURE
FOR NON-CLASSICAL LOGICS:
THE PROGRAM KRIPE
(ABSTRACT)

The relevant logics $E$, $R$ and $NR$ are given Hilbert-style axiomatizations and are studied in detail in [1]. By dropping the axioms governing the extensional connectives $\land$ and $\lor$ from the axiomatizations of these logics we obtain their implication/negation fragments – respectively the system $\neg\neg\rightarrow$, $\neg\neg\rightarrow$ and $\neg\neg\rightarrow$. By adding axioms for the intensional connectives fusion $\circ$ and fission $+$ to $\neg\neg\rightarrow$, we obtain the pure intensional fragments of the systems $R$ and $NR$ – called in [4] $R_i$ and $NR_i$ respectively. By dropping from $R$ just the axiom that governs the distributional properties of $\land$ and $\lor$, i.e. $A \land (B \lor C) \rightarrow (A \land B) \lor C$, we obtain the system called $OR$ in [10] and studied in this book.

A theorem was announced in [3] which was extended in [2], [5] and [7] to show that the relevant logics $\neg\neg\rightarrow$, $R_i$, $NR_i$ and $OR$ and the modal $S4$ are decidable. This theorem was shown in [8] to be equivalent to the number-theoretic theorem was shown in [8] to be equivalent to the number-theoretic theorem known as Dickson’s theorem (for details of which see [9]). Given any formula $A$ (in the appropriate vocabulary) the decision procedure for each of these logics describes a way of recursively constructing out of $A$ a proof-search which will contain as a sub-tree a proof of $A$ if there is one and which, it is proved will always be finite. However in practice this decision procedure tends to be impossible to use due to the exponential rate at which the proof search tree for $A$ usually grows. Hence in [4] the question of whether this decision procedure could be mechanized via computer was asked.
In this paper we describe a PASCAL computer program called KRIPKE which mechanizes the Kripke decision procedure for these logics as they are formulated in [5]. KRIPKE achieves this mechanization by actually recursively constructing depth-first the proof search tree for A. Space-time efficiency is achieved firstly by means of a pre-analysis of A which makes for optimal internal program code and secondly by employing, during tree construction, a series of filters at each node in the proof search tree. These filters, by utilizing certain properties that these logics possess, terminate tree construction at certain nodes at which it would not otherwise have been terminated. Some of these properties, e.g. the use of a number of facts concerning the occurrences of positive and negative formulas in theorems of these logics and the use of a set of logical matrices based on a lattice known as the crystal lattice are not new and rely on theorems that can be found in either [1] or [10]. Some of the other properties of these logics utilized as filters in KRIPKE are new and are described in this paper. KRIPKE is an interactive program. A candidate formula A is simply input at a terminal and KRIPKE then lists the systems in which A is provable or unprovable and outputs a proof of A (if there is one) that is a proof of A in one of the logics formulated as in [5]. We have excellent speeds with KRIPKE – a search for a proof of A usually taking only a few c.p.u. seconds.

References


---

*La Trobe University (McRobbie and Thistlewaite)*

*Australian National University (Meyer)*