Lietuvos matematikos rinkinys ISSN 0132-2818
Proc. of the Lithuanian Mathematical Society, Ser. A Vol. **58**, 2017
DOI: 10.15388/LMR.A.2017.01 pages 1–6

Brodsky's coding method for propositional logic

Romas Alonderis

Institute of Mathematics and Informatics, Vilnius University
Akademijos 4, LT-08663, Vilnius, Lithuania
E-mail: romas.alonderis@mii.vu.lt

Abstract. Brodsky's coding method for propositional logic is considered in the paper. Based on the sequent calculus, the method allows us to determine whether an arbitrary sequent is derivable in the calculus without constructing proof-search trees. The coding method, presented in the paper, can be used as a decision procedure for propositional logic.

Keywords: propositional logic, sequent calculus, Brodsky's coding method.

1 Introduction

Sequent calculi are comparatively convenient means for proof search of sequents/formulas. For example, in the case of propositional logic, it suffices to apply derivation rules in the bottom-up way to sequents in order to check whether the root sequent is derivable. The process is finite and each time the same, routine: determine the outermost symbol and apply the corresponding derivation rule. However, the proof search using sequent calculi has a drawback, it lacks compactness. A proof-search tree may grow very large after several steps. One can use the sequent calculus $\mathbf{LK}_0$ (see further) and be convinced that any maximal backward proof-search tree with the sequent

$$\to \big(\big(\big((A \vee C) \wedge (B \vee C)\big) \wedge (A \vee D)\big) \wedge (B \vee D)\big) \supset \big((E \wedge F) \vee (G \wedge H)\big)$$

at the root, where the letters denote atomic formulas, has 64 branches.

An application of a derivation rule in the bottom-up direction involves duplication of all the conclusion formulas, except that the outermost logical sign of the principal formula disappear above the line, and non-principal formulas are duplicated twice in the case of a two-premise rule.

An interesting solution to the backward proof-search tree expansion problem is given in [3]. The authors present Brodsky's coding method based on the sequent calculus. Instead of constructing a proof-search tree, the considered sequent is coded. The coding enables us to determine whether the sequent is derivable in the sequent calculus the coding method is based on.

The principles of Brodsky's coding method were first presented in [2]. In [3], the coding method is described for propositional and monadic logics. Some modification and complementation of [3] for propositional logic is in [1].

The present paper is organized as follows. In Section 2, we recall the material of [1], except Section 2.3 which is new. In Section 3, we define p-closed code tables

Table 1. Coding table.

	E	$\neg A$	$A \wedge B$		$A \vee B$		$A \supset B$	
In the antecedent	a	s	a	a	a	a	s	a
			$-$		$+$		$+$	
In the succedent	s	a	s	s	s	s	a	s
			$+$		$-$		$-$	

and show that an arbitrary sequent is derivable in the sequent calculus $\mathbf{LK}_0$, iff the corresponding code table is p-closed.

2 Brodsky's coding method for propositional logic

The coding method for propositional logic is based on the Gentzen type propositional sequent calculus $\mathbf{LK}_0$ without structural rules. It can be found in, e.g., [1].

A sequent of the shape $F \to$ or $\to F$, where F is a formula, is called basic. In the paper, we consider coding of the basic sequents.

Any sequent

$$A_1, \ldots, A_m \to B_1, \ldots, B_n$$

is equivalent to the basic sequent

$$\to (A_1 \wedge \cdots \wedge A_m) \supset (B_1 \vee \cdots \vee B_n)$$

(if $m = 0$ or $n = 0$, then the sequent is equivalent to the basic sequents $\to B_1 \vee \cdots \vee B_n$ or $A_1 \wedge \cdots \wedge A_m \to$, respectively).

A proof-search tree, no leaf of which contains logical signs, is called completely expanded.

Given a basic sequent S, let us consider a completely expanded $\mathbf{LK}_0$ backward proof-search tree V with S at the root. Using the coding method, we want to obtain all the leaves of V without constructing the tree itself. We divide the coding of the basic sequent S into three stages:

1. At some point in the course of a backward proof-search tree construction, each atom E in S appears as a separate formula in the antecedent or succedent of a sequent. Also, each logical sign σ becomes outermost. Determine where E as a separate formula and σ as the outermost logical sign will occur, in the antecedent or succedent. Also, determine the number of premises of the $\mathbf{LK}_0$ rule the principal formula of which has the outermost logical sign σ.

2. Determine the number of branches in V.

3. Distribute each atom E in S to the leaves of V.

The coding allows us to imitate in a compact way the process of proof-search tree construction. Now we present each stage in detail.

2.1 Stage 1

The following coding table is used in this stage, where E is an atomic formula; A and B are arbitrary formulas; '$-$' and '$+$' stand for one and two premise rules, respectively; 's' and 'a' stand for 'succedent' and 'antecedent', respectively. See [1] for more information.

2.2 Stage 2

The number of branches of V is determined as follows. Write 1 above each atom; write "$+$" above a logical sign coded by "$+$", and "$\times$" above a logical sign coded by "$-$", but ignore $\neg$. Parenthesize the obtained expression according to the basic sequent, until the order of arithmetic operations application becomes clear. The details can be found in [1].

2.3 Stage 3

The distribution of atoms to leaves is performed by constructing a table of codes from which we obtain all the leaves of V.

The construction of the table is explained by coding the sequent

$$\to \big(\big((A \wedge B) \wedge (C \wedge D)\big) \vee (E \vee \neg G)\big) \wedge F,$$

where the letters denote atomic formulas.

First, we perform Stage 1 and 2 coding:

$$\begin{array}{rccccccccccccccc}
 & \big(\big((1 & + & 1) & + & (1 & + & 1)\big) & \times & (1 & \times & & 1)\big) & + & 1 \\
\to & \big(\big((A & \wedge & B) & \wedge & (C & \wedge & D)\big) & \vee & (E & \vee & \neg & G)\big) & \wedge & F. \\
 & s & s & s & s & s & s & s & s & s & s & s & a & s & s \\
 & & + & & + & & + & & - & & - & & & + & s
\end{array}$$

The arithmetical expression containing only one sort of arithmetical operations is called ground. We replace each number in the top row by the atom with its 'a', 's' code beneath the number and obtain:

$$\big(\big(\big(A^s + B^s\big) + \big(C^s + D^s\big)\big) \cdot \big(E^s \cdot G^a\big)\big) + F^s.$$

We drop the parentheses from each ground expression, leaving only the outer pair:

$$\big(A^s + B^s + C^s + D^s\big) \cdot \big(E^s \cdot G^a\big)\big) + F^s.$$

This expression is called the principal arithmetical expression. We place each subexpression $(E_1^{\theta_1}\sigma \dots \sigma E_n^{\theta_n})$, where $\theta_i \in \{a, s\}$ and $\sigma = + \mid \cdot$, into a column and draw under-lines which replace the relevant parentheses:

$$\underline{\big|\ A^s \ + \ B^s \ + \ C^s \ + \ D^s\ \big|} \cdot \underline{\big|\ E^s \ \cdot \ G^a\ \big|} + \big|\ F^s\ \big|$$

The sign '$+$' expresses the fact that the atoms, joined by it, are on different leaves:

	A^s	B^s	C^s	D^s	$\cdot$	E^s	$\cdot$	G^a	$+$	F^s
1	*									*
2		*								
3			*							
4				*						

We have dropped '+' in the first column and put '$*$' under the appropriate atoms to indicate the fact that the atoms are present on the corresponding leaves.

The sign '$\cdot$' in a column indicates that the propositional symbols, joined with '$\cdot$', are on the same leaf. We drop '$\cdot$' between E^s and G^a and get the table:

	A^s	B^s	C^s	D^s	$\cdot$	E^s	G^a	$+$	F^s
1	$*$					$*$	$*$		$*$
2		$*$							
3			$*$						
4				$*$					

Such a table, where '$*$' occurs beneath each atom, is called a completely marked code table. The span without the arithmetic operations is called a column of the table. The above code table has three columns.

The sign "$\cdot$" between columns indicates that the distribution of propositional symbols to leaves is obtained by taking all the variants: one non-empty row from one column and one non-empty row from the other column. These variants for the above table are:

$$(1,1)$$
$$(2,1)$$
$$(3,1)$$
$$(4,1)$$

We merge the first and second columns and get the following table:

	A^s	B^s	C^s	D^s	E^s	G^a	$+$	F^s
1	$*$				$*$	$*$		$*$
2		$*$			$*$	$*$		
3			$*$		$*$	$*$		
4				$*$	$*$	$*$		

We say that, e.g., D^s occurs in the fourth row of the first column or in the cell $(1,4)$ of the table.

Finally, we merge the first and second columns of this table and obtain the table:

	A^s	B^s	C^s	D^s	E^s	G^a	F^s
1	$*$				$*$	$*$	
2		$*$			$*$	$*$	
3			$*$		$*$	$*$	
4				$*$	$*$	$*$	
5							$*$

Such a completely marked code table, consisting of one column, is called a completely expanded code table. The rows of the table indicate the leaves of the completely expanded proof-search tree V:

$$\big(A^s, E^s, G^a\big) \quad \big(B^s, E^s, G^a\big) \quad \big(C^s, E^s, G^a\big) \quad \big(D^s, E^s, G^a\big) \quad \big(F^s\big).$$

The corresponding leaves are:

$$G \to A, E; \quad G \to B, E; \quad G \to C, E; \quad G \to D, E; \quad \to F.$$

Lemma 1. *For any basic sequent S, Stage* 3 *yields a correct distribution of atoms to leaves of any completely expanded proof-search tree, generated by the derivation rules of the calculus $\boldsymbol{LK}_0$ with the sequent S at the root.*

3 p-Closed tables

The expression E^θ, where E is an atomic formula and $\theta \in \{a, s\}$, is called a marked atom.

A pair (E^θ, E^η) of marked atoms, where $\theta \neq \eta$, is called a contradictory pair.

A completely marked code table T, consisting of $n \geqslant 1$ columns with '$\cdot$' between any two of them, is called an analytic table.

The expression T_S (T_S^a), where S is a sequent, denotes a completely marked (analytic) code table of the basic sequent corresponding to S.

The sequence

$$r_1, \ldots, r_n,$$

where r_i is a non-empty row in the i-th column of the analytic table T consisting of $n \geqslant 1$ columns, each '$*$' replaced by the corresponding marked atom, is called a path of T. For example, the table

	A^a	B^a	D^a	$\cdot$	E^s
1	$*$				$*$
2		$*$	$*$		

has the following paths:

$$(1,1) = \big(A^a, E^s\big) \quad \text{and} \quad (2,1) = \big(B^a, D^a, E^s\big).$$

It is clear that all completely marked code tables of any fixed basic sequent S have the same paths.

A table T_S is called p-closed, iff each path of T_S^a contains a contradictory pair; here T_S^a is obtained from T_S by merging some, if any, columns.

Theorem 1. *A sequent S is derivable in $\boldsymbol{LK}_0$, iff T_S is p-closed.*

Proof. The proof follows from Lemma 1.

4 Concluding remarks

In the present paper, Brodsky's coding method for propositional logic is considered. It allows us to determine whether a sequent is derivable in the corresponding sequent calculus, using the analysis of the sequent by means of the code tables without generating proof-search trees. This is a purely syntactic method, based on the corresponding sequent calculus and the analysis of the paths of code tables, covered by contradictory pairs. We have defined the conditions under which the code table is p-closed and proved that a sequent is derivable in $\mathbf{LK}_0$, iff the corresponding code table is p-closed.

It would be interesting to consider Brodsky's coding method for modal and non-classical logics.

References

[1] R. Alonderis. A coding method for a sequent calculus of propositional logic. *Lith. Math. J.*, **48**(2):123–136, 2008. doi:10.1007/s10986-008-9008-6.

[2] I.N. Brodsky. Enthymeme reconstruction. *Bulletin of Leningrad State University*, **6**(1):40–44, 1988.

[3] V.P. Moukhachjov and I.V. Netchitailov. An improvement of Brodsky's coding method for the sequent calculus of first-order logic. In *Logic Joint International Conference of automated Reasoning*, Siena, Italy, 18–25 June, 2001, pp. 113–121. Available from Internet: http://www.philosophy.ru/library/logic/an_improvement_of_brodsky.doc.

[4] A.S. Troelstra and H. Schwichtenberg. *Basic Proof Theory*, second edition. Cambridge University Press, 2000.

[5] H. Wansing. Sequent calculi for normal modal propositional logics. *J. Logic Comput.* **4**(2):125–142, 1994.

REZIUMĖ

Brodskio kodavimo metodas teiginių logikai

R. Alonderis

Straipsnyje pateikiamas Brodskio kodavimo metodas teiginių logikai. Šis metodas pagrįstas kodavimo lentele, sudaryta sekvencinio skaičiavimo pagrindu. Remiantis šia lentele, generuojama nagrinėjamos sekvencijos kodų lentelė. Parodoma, kad bet kuri sekvencija yra įrodomą minėtame skaičiavime tada ir tik tada, jei jos kodų lentelė yra p-uždara.

Raktiniai žodžiai: teiginių logika, sekvenciniai skaičiavimai, Brodskio kodavimo metodas.