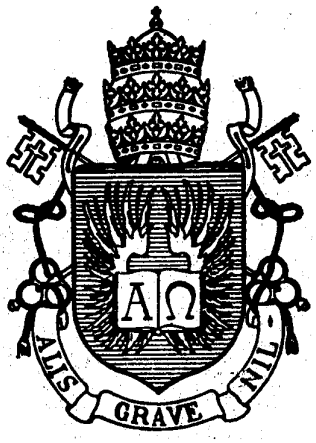

PUC

Series: Monografias em Ciência da Computação
Nº 15/78

COMMUNICATION PREDICATES: A COMPLETE STRATEGY FOR RESOLUTION-BASED THEOREM PROVERS

- AN EVALUATION OF AN IMPLEMENTATION -

by

Roberto L. de Carvalho
Emmanuel P. L. Passos
Sílvia R. G. Peixoto

Departamento de Informática

Pontifícia Universidade Católica do Rio de Janeiro
Rua Marquês de São Vicente, 225 – CEP-22453
Rio de Janeiro – Brasil

Series: Monografias em Ciência da Computação
Nº 15/78

Series Editor: Michael F. Challis August, 1978

COMMUNICATION PREDICATES: A COMPLETE STRATEGY FOR RESOLUTION-BASED THEOREM PROVERS

- AN EVALUATION OF AN IMPLEMENTATION - *

by

Roberto L. de Carvalho
Emmanuel P. L. Passos**
Sílvia R. G. Peixoto

* This work has been sponsored in part by FINEP.
** Instituto Militar de Engenharia, Seção de Informática, RJ.

For copies contact:
Rosane T. L. Castilho
Head, Setor de Documentação e Informação
Deptº de Informática - PUC/RJ
Rua Marquês de São Vicente, 209 - Gávea
22.453 - Rio de Janeiro, RJ - Brasil.

ABSTRACT:

A new and complete strategy for resolution-based theorem provers for first order logic is presented. This strategy is a central result in [1]. The strategy extends Bledsoe's splitting technique which consists of the replacement of a set $S \cup \{c_1 \vee c_2\}$ of clauses by two sets $S \cup \{c_1\}$ and $S \cup \{c_2\}$ of clauses. We extend it in the following senses: a - it enlarges the range of applications by allowing to split sets of clauses, and;
b - it allows the occurrance of common variables in the split parts. The new strategy is implemented and is presented from practical point of view. The examples illustrating the effects of the strategy are discussed.

KEYWORDS: Theorem-proving, Applied Logic, Resolution

RESUMO: Uma nova estratégia para demonstração automática de teoremas, baseada no princípio de Resolução é apresentada. Esta estratégia é o resultado central em [1]. A estratégia extende a técnica de "Splitting" de Bledsoe, a qual consiste de substituição de um conjunto $S \cup \{c_1 \vee c_2\}$ de "cláusulas" por dois conjuntos $S \cup \{c_1\}$ e $S \cup \{c_2\}$. Nós a extendemos em dois sentidos: a) - aumenta o número de aplicações sendo permitida a divisão de conjuntos de "cláusulas", e b) permite a ocorrência de variáveis comuns nas partes separadas. A nova estratégia é implementada e é apresentada de um ponto de vista prático. Os exemplos ilustram os efeitos da estratégia.

PALAVRAS CHAVE: Demonstração automática de teoremas, Lógica Aplicada, Resolução.

1. BACKGROUND MATERIAL

The Resolution Principle was first introduced by J.A. Robinson [6]. In what follows, we assume a certain familiarity with all the concepts discussed in [1,6,7,9,10]. The main result of this paper is proved by using Boyer's Locking Resolution, which can be found in [8].

Locking resolution is a refinement of resolution, which uses a concept similar to that of ordered clauses. According to that concept, given a set S of clauses, we arbitrarily index each occurrence of literals in S with integers. Different occurrences of the same literal in S may be indexed differently, and two different literals can receive the same index. Resolution is then permitted only on literals of lowest index in each clause. Boyer proved that such refinement is a complete strategy. A deduction from S is called a lock deduction if every clause in the deduction is either a clause in S, or a lock resolvent of previous clauses in the deduction.

Theorem 1.1

[Boyer] let S be a set of clauses, where every literal in S is indexed by an integer. S is unsatisfiable if and only if there is a lock deduction of the empty clause ( ) form S.

2. COMMUNICATION PREDICATES

A common way to prove that a certain class A of objects is equal to another class B of objects, in a given theory, is the following:

a. To prove that A is contained in B

b. To prove that B is contained in A.

In general, (a) and (b) are independent actions, in the sense that the reasoning used in (a) is not related with that used in (b).

Such a procedure can be mechanized in a resolution based theorem-prover. We can assume that equality of classes is defined by

$$\forall x \forall y \; (x = y \iff (x \leq y \vee y \leq x)) \quad \ldots \quad (2.1)$$

or in clause form

$$\left.\begin{array}{l} A1.\ \neg x = y \vee x \leq y \\ A2.\ \neg x = y \vee y \leq x \\ A3.\ \ x = y \vee \neg x \leq y \vee \neg y \leq x \end{array}\right\} \quad \ldots \quad (2.2)$$

In a refutation procedure, to prove that $A = B$ we begin by denying $A = B$. Then in (2.2) we deal with A3.

From $(\neg A = B)$ and A3, we obtain $\neg A \leq B \vee B \leq A$. This must be reduced against a set of clauses, say S, which comes from the axioms of the theory in which A and B are objects, and from a certain set of conditions on these objects (additional hypothesis).

Our approach to mimic (a) and (b) is as follows:

(i) Replace A3 by

$$\left.\begin{array}{l} A3'.\ \ x = y \vee \neg x \leq y \vee P(x,y) \\ A4'.\ \ \neg P(x,y) \vee \neg y \leq x \end{array}\right\} \quad (2.3)$$

(ii) Try to obtain $P(x,y)\,\sigma$ from SU $\{A3'\}$, for some substitution σ.

(iii) Try to obtain ( ) (the empty clause) from SU $\{(\neg y \leq x)\,\sigma\}$

The predicate symbol P introduced in (2.3) is a new one, and we say that P is a communication predicate (c.p., for short).

In this paper we will show that the above procedure is a complete strategy, even in a more general setting.

The motivation for introducing c.p.'s is briefly described as follows:

Given a set S of clauses and a set P of new predicate symbols, we replace S by another set S' of clauses, which contains predicate symbols in

P, in such a way that:

1. S is unsatisfiable if and only if S' is unsatisfiable.

2. It takes advantage of certain special features of S' for the design of theorem-provers, which in some sense mimic a human theorem-proven.

In what follows we will present three ways of introducing the c.p.'s.

Theorem 2.1

Let S be a set of clauses, $C = C_1 \vee C_2$ a clause, P a new predicate symbol and $\bar{v}$ a vector formed with variables common to C_1 and C_2, then SU $\{C_1 \vee C_2\}$ is unsatisfiable if and only if SU $\{C_1 \vee P(\bar{v}), C_2 \vee \neg P(\bar{v})\}$ is unsatisfiable.

Notice that no restrictions were imposed on the subclauses C_1 and C_2. The following example shows that we need the restrictions imposed on $\bar{v}$, i.e., $\bar{v}$ must contain all variables common to C_1 and C_2.

Example 2.1

The set $\{P(x) \vee R(x), \neg P(a), \neg R(b)\}$ is satisfiable, but if we introduce a new predicate symbol Q, we will obtain an unsatisfiable set of clauses.

By successive applications of theorem 2.1, we obtain the following:

Theorem 2.2

Let S be a set of clauses, $C = C_1 \vee C_2 \vee \ldots \vee C_n$ a clause, $P_1, P_2, \ldots, P_{n-1}$ new predicate symbols, and $\bar{v}_1, \bar{v}_2, \ldots, \bar{v}_{n-1}$ vectors such that $\bar{v}_1$ consists of all variables common to C_1 and $C_2 \vee C_3 \vee \ldots \vee C_n$, and for all i, $i = 2,3,\ldots,n-1$ $\bar{v}_i$ consists of all variables common to $P_{i-1}(\bar{v}_{i-1}) \vee C_i$ and $C_{i+1} \vee \ldots \vee C_n$. Then SU $\{C\}$ is unsatisfiable if and only if SU $\{C_1 \vee P_1(\bar{v}_1), \neg P_1(\bar{y}_1) \vee C_2 \vee P_2(\bar{v}_2), \ldots\ldots \neg P_{n-2}(\bar{v}_{n-2}) \vee C_{n-1}(\bar{v}_{n-1}), P_{n-1}(\bar{v}_{n-1}) \vee C_n\}$ is unsatisfiable.

Theorem 2.3

Let S be a set of clauses of the form $\{C\} \vee S_1$, P a new predicate symbol and $\bar{v}$ a vector consisting of all variables common to C and S_1. Then for any set S' of clauses, not containing P, $S' \cup S$ is unsatisfiable if and only if $S' \cup \{C \vee P(\bar{v})\} \cup (\{\neg P(\bar{y})\} \vee S_1)$ is unsatisfiable.

Theorem 2.4

Let S be a set of clauses of the form $\{C_1, C_2\} \vee S_1$, P a new predicate symbol, and $\bar{v}$ a vector consisting of all variables common to C_1 and S_1 together with those common to C_2 and S_1. Then for any set S' of clauses, not containing P, $S' \cup S$ is unsatisfiable if and only if $S' \cup \{C_1 \vee P(\bar{v}), C_2 \vee P(\bar{v})\} \cup (\{\neg P(\bar{v})\} \vee S_1)$ is unsatisfiable.

For the kind of applications we have in mind, a set P of predicate symbols occuring in a set S of clauses can be considered as a set of c. p.'s.

Definition 2.1

Let S be a set of clauses and P a set of predicate symbols. We say that S is P-separable if and only if each clause in S contains at most one negative occurance of a literal in P.

Definition 2.2

Let S be a set of clauses, and P a set of predicate symbols. We say that S is P-refutable if and only if there is a sequence $C_1, C_2, \ldots, C_n$ of clauses, such that:

(i) For each $i = 1,2,\ldots,n$ C_i is in S or C_i was obtained by resolving two previous clauses C_j and C_k in the sequence.

(ii) If a clause containing negative occurences of literals in P is a parent clause of a clause in the sequence, then the other parent clause containing only literals in P (a pure clause).

(iii) $C_n = (\ )$

We will show that a P-separable set of clauses is unsatisfiable if and only if it is P refutable.

Theorem 2.5

A P-separable set S of clauses is unsatisfiable if and only if it is P refutable.

Proof: Clearly if S is P-refutable then S is unsatisfiable. Now suppose that S is unsatisfiable, and let us consider the following indexing of the literals of S:

1. Index the negative occurences of literals in P with integers 1,2,...,m-1, in such a way that the name predicate symbol in P receives the name index.

2. Index all remaining literals in P (i.e. positive occurences) with the integers m+1, m+2, ..., 2m-1, in such a way that the same predicate symbol in P receives the same index.

3. Index all remaining literals with the integer m.

Then there is a lock refutation C_1, C_2, ..., C_{n-1}, ( ) from S. Notice that:

a. $C_1, C_2, \ldots, C_{n-1}$, ( ) satisfies (i) and (iii) of definition 2.2,

b. (ii) of definition 2.2 is satisfied, because negative occurences of literals in P have the lowest indexes in the clauses where they occur. Only when a pure clause is obtained literals in P are eligible to be resolved upon.

Therefore, the sequence $C_1, C_2, \ldots, C_{n-1}$, ( ) satisfies definition 2.2, and S is P-separable.

Example 2.2

Consider the following Set S of clauses:

1. $\neg E(x) \vee V(x) \vee S(x, f(x))$
2. $\neg E(x) \vee V(x) \vee C(f(x))$
3. $P(a)$
4. $E(a)$
5. $\neg S(a,y) \vee P(y)$
6. $\neg P(x) \vee \neg V(x)$
7. $\neg P(x) \vee \neg C(x)$

Notice that S is {V, S, C}-separable, and we can index its literals as we did in the proof of theorem 2.5, obtaining the following:

1'. ${}_4\neg E(x) \vee {}_5V(x) \vee {}_6S(x, f(x))$
2'. ${}_4\neg E(x) \vee {}_5V(x) \vee {}_7C(f(x))$
3'. ${}_4P(a)$
4'. ${}_4E(a)$
5'. ${}_2\neg S(a,y) \vee {}_4P(y)$
6'. ${}_4\neg P(x) \vee {}_1\neg V(x)$
7'. ${}_4\neg P(x) \vee {}_3\neg C(x)$

Thus we have the following lock refutation which is a {V,S,C} - reputation.

8.	${}_5V(a) \vee {}_6S(a, f(a))$	(1', 4')
9.	${}_4\neg P(a) \vee {}_6S(a, f(a)$	(8 , 6')
10.	${}_6 S(a, f(a))$	(9 , 3')
11.	${}_4P(f(a)$	(10, 5')
12.	${}_5V(a) \vee {}_7C(f(a))$	(2', 4')
13.	${}_4\neg P(a) \vee {}_7C(f(a))$	(12, 6')
14.	${}_7 C(f(a))$	(13, 3')
15.	$\neg {}_4P(f(a))$	(14, 7')
16.	( )	(11, 15)

3. APPLICATIONS OF P REFUTATION

Certain classes of problems, when formalized in first order logic, generate a large number of clauses each containing a large number of literals. The next example illustrates this fact.

Example 3.1

The following set of clauses can be considered as axioms for a part of the theory of sets:

A1.	$\neg\neg x \leq y \vee \neg z \in x \vee z \in y$	Defines '$\leq$'
A2.	$x \leq y \vee f(x,y) \in x$	
A3.	$x \leq y \vee \neg f(x,y) \in y$	
A4.	$\neg x = y \vee x \leq y$	Defines '$=$'
A5.	$\neg x = y \vee y \leq x$	
A6.	$x = y \vee \neg x \leq y \vee \neg y \leq x$	
A7.	$\neg x \in (y \cup y) \vee x \in y \vee x \in z$	Defines '$\cup$'
A8.	$x \in (y \cup z) \vee \neg x \in y$	
A9.	$x \in (y \cup z) \vee \neg x \in z$	
A10.	$\neg x \in (y \cap z) \vee x \in y$	Defines '$\cap$'
A11.	$\neg x \in (y \cap z) \vee x \in z$	
A12.	$x \in (y \cap z) \vee \neg x \in y \vee \neg x \in z$	

Suppose we want to prove $\forall x \forall y \forall z\ (x \cap (y \cup z) = (x \cap y) \cup (x \cap z))$. Using the axioms A1 - A12 we obtain (by reducing to the primitive $\in$) the following set of clauses:

1. $a \in A_1 \vee b \in A_1$
2. $a \in A_1 \vee b \in A_2 \vee b \in A_3$
3. $a \in A_1 \vee \neg b \in A_1 \vee \neg b \in A_2$
4. $a \in A_1 \vee \neg b \in A_1 \vee \neg b \in A_3$
5. $a \in A_2 \vee a \in A_3 \vee b \in A_1$

6. $a \in A_2 \vee a \in A_3 \vee b \in A_2 \vee b \in A_3$

7. $a \in A_2 \vee a \in A_3 \vee \neg b \in A_1 \vee \neg b \in A_2$

8. $a \in A_2 \vee a \in A_3 \vee \neg b \in A_1 \vee \neg b \in A_3$

9. $\neg a \in A_1 \vee \neg a \in A_2 \vee b \in A_1$

10. $\neg a \in A_1 \vee \neg a \in A_2 \vee b \in A_2 \vee b \in A_3$

11. $\neg a \in A_1 \vee \neg a \in A_2 \vee \neg b \in A_1 \vee \neg b \in A_2$

12. $\neg a \in A_1 \vee \neg a \in A_2 \vee \neg b \in A_1 \vee \neg b \in A_3$

13. $\neg a \in A_1 \vee \neg a \in A_3 \vee b \in A_1$

14. $\neg a \in A_1 \vee \neg a \in A_3 \vee b \in A_2 \vee b \in A_3$

15. $\neg a \in A_1 \vee \neg a \in A_3 \vee \neg b \in A_1 \vee \neg b \in A_2$

16. $\neg a \in A_1 \vee \neg a \in A_3 \vee \neg b \in A_1 \vee \neg b \in A_3$,

where a is $f(A_1 \cap (A_2 \cup A_3), (A_1 \cap A_2) \cup (A_1 \cap A_3))$ and b is $f(A_1 \cap A_2) \cup (A_1 \cap A_3), A_1 \cap (A_2 \cup A_3))$.

The use of c.p.'s together with P-refutation minimizes this explosive generation of clauses. Let us modify A1 - A12 by the introduction of predicate symbols P_1, P_2 and P_3 as follows:

Replacé A6 by:

A6'. $x = y \vee \neg x \leq y \vee P_1(x,y)$

A6". $P_1(x,y) \vee \neg y \leq x$

Replace A7 by:

A7'. $\neg x \in (y \cup z) \vee x \in y \vee P_2(x,y)$

A7". $\neg P_2(x,y) \vee x \in z$

Replace A12 by:

A12'. $x \in (y \cap z) \vee \neg x \in y \vee P_3(x,y)$

A12". $\neg P_3(x,z) \vee \neg x \in z$

The followwing is a $\{P_1, P_2, P_3\}$-refutation of $\neg(A_1 \cap (A_2 \cup A_3) = (A_1 \cap A_2) \cup (A_1 \cap A_2)$:

A. Initial translation:

1. $a \in A_1 \vee P_1(I, II)$
2. $a \in A_2 \vee P_2(a, A_3) \vee P_1(I, II)$
3. $\neg a \in A_1 \vee P_3(a, A_2) \vee P_1(I, II)$
4. $\neg a \in A_1 \vee P_3(a, A_3) \vee P_1(I, II)$

where I is $A_1 \cap (A_2 \cup A_3)$ and II is $(A_1 \cap A_2) \cup (A_1 \cap A_3)$, a and b as before.

B. Proceed with refutation until a pure clause is found.

5. $P_3(a, A_2) \vee P_1(I, II)$ . . . . (1, 3)

Now (5) is pure and $P_3(a, A_2)$ can be eliminated (an order of elimination of c.p.'s is assumed).

6. $a \in A_2 \vee P_1(I, II)$ ... (5, A12')
7. $P_2(a, A_3) \vee P_1(I, II)$ ... (6, 2)
8. $a \in A_3 \vee P_1(I, II)$ ... (7, A7")
9. $P_3(a, A_3) \vee P_1(I, II)$ ... (1, 4)
10. $a \in A_3 \vee P_1(I, II)$ ... (9, A12")
11. $P_1(I, II)$

Notice that (11) subsumes a 11 clauses generated so far, which can be eliminated from the system; (11) itself is "translated":

1'. $b \in A_1 \vee P_3(b, A_2 \cup A_3)$
2'. $b \in A_1 \vee P_2(b, A_1 \cap A_3)$
3'. $b \in A_2 \vee P_2(b, A_1 \cap A_3)$

Proceedings as before we will obtain the empty clause ( ).

EVALUATION

Experimentation with and evaluation of an automated theorem-proving program require a varied problem set [3]. In this work we made experiments with 400 formulas in $[BAC]_0$, 100 in $[BAC]_1$ and 20 in $[BAC]_2$. Although the system has presented problems to formulas in $[BAC]_i$ for $i \geq 2$, we decide to get some results that we are using in the modification of the program (system).

The following table show summary results of the experiments performed.

GIVEN FORMULA	THEOREM?	(0,0) # Of Resolvents	(0,0) # Of Unif	(0,0) EXEC TIME	(1,0) # Of Resolvents	(1,0) # Of Unif	(1,0) EXEC TIME	(0,1) # Of Resolvents	(0,1) # Of Unif	(0,1) EXEC TIME	(1,1) # Of Resolvents	(1,1) # Of Unif	(1,1) EXEC TIME
1. $A-A=\emptyset$	YES	0	1	0.160	0	1	0.190	0	1	0.160	0	1	0.150
2. $\neg((A\cap A)\cup A\subseteq A)$	NO	0	0	0.076	0	0	0.077	0	0	0.070	0	0	0.063
3. $\neg(\bar{A}=A)$	NO	1	1	0.093	1	1	0.090	1	1	0.080	1	1	0.090
4. $\bar{A}=A$	NO	12	0	0.244	0	0	0.090	12	0	0.243	0	0	0.083
5. $\bar{A}-(A\cap A)=\emptyset$	NO	30	0	0.604	0	0	0.110	6	0	0.253	0	0	0.124
6. $A=B\rightarrow(A\cap B=A\wedge A\cup B=A)$	YES	MEMORY OVERFLOW		20.672	46	23	1.923	TEMPO		-	71	35	3.290
7. $A=B\rightarrow(A\cap B=B\wedge A\cup B=B)$	YES	TEMPO		-	67	30	2.423	MEMORY OVERFLOW		16.599	103	47	4.277
8. $\bar{A}\cup\bar{B}\subseteq A\cap B$	NO	0	0	0.133	0	0	0.123	31	5	1.080	0	0	0.147
9. $A\cap(B\cap(A\cup B))\subseteq(\bar{A}\cap B)\cup(A\cup\bar{B})$	NO	MEMORY OVERFLOW		-	6	3	0.517	MEMORY OVERFLOW		-	15	5	0.670
10. $(\bar{A}\cap B)\cup(A\cap\bar{B})=(A\cap B)\cup(\bar{A}\cap\bar{B})$	YES	130	32	23.355	3	8	3.159	138	60	7.923	4	8	0.903
11. $((\bar{A}\cap B)\cup(A\cap\bar{B})=A\cup\bar{B}\wedge B\subseteq A)\rightarrow A=B$	YES	92	30	2.374	6	4	0.557	MEMORY OVERFLOW		11.156	30	14	1.690
12. $(A\cap\bar{B})\cup(\bar{A}\cap B)=(A\cap(\bar{A}\cup\bar{B}))\cup(B\cap(\bar{A}\cup\bar{B}))$	NO	TEMPO		-	MEMORY OVERFLOW		4.403	MEMORY OVERFLOW		6.770	13	9	1.097
13. $\neg(A\cup B=B\rightarrow A\subseteq B)$	NO	TEMPO		-	23	4	0.680	TEMPO		54.0	5	0	0.223
14. $\neg((A\cap B)-(\bar{A}\cup B)=(A\cap B)\cap(A\cup B)$	NO	32	4	1.080	0	0	0.166	29	26	3.717	0	0	0.227
15. $A\cup B=(\bar{A}-\bar{B})\cap(A\cap\bar{B})$	NO	8	2	0.483	8	2	0.510	8	4	0.716	8	4	0.717
16. $(\bar{A}\cap B)\cap(A\cap\bar{B})=(\bar{A}\cap B)\cup(A\cap\bar{B})$	NO	1084	66	52.493	6	0	0.466	MEMORY OVERFLOW		40.516	11	3	0.680
17. $((A\cap B)-(\bar{A}\cup B)=(A-B)\cap(A\cup B)$	NO	32	4	1.080	0	0	0.166	29	26	3.717	0	0	0.227
18. $(A\cup B)\cap((\bar{A}\cup C)\cap(B\cup C)=(A\cup B)\cap(A\cup C)$	YES	MEMORY OVERFLOW		28.902	12	12	1.033	MEMORY OVERFLOW		87.798	33	23	3.170
19. $(\bar{A}\cap C)\cup(A\cap\bar{C})=B\rightarrow(\bar{A}\cap B)\cup(A\cap\bar{B})=C$	YES	448	82	11.506	29	12	1.220	MEMORY OVERFLOW		12.613	44	23	2.690
20. $(A\cap B=A\cap(A\cap B=C)\rightarrow A\cap B=A\cap C$	YES	256	70	7.899	11	6	0.673	203	59	11.493	11	6	0.784
21. $((A\cap B)\cup(B\cap C))\cup(C\cap A)=((A\cup B)\cup(B\cup C))\cap(C\cap A)$	NO	MEMORY OVERFLOW		5.339	TEMPO		-	MEMORY OVERFLOW		34.188	8	5	0.834
22. $\bar{A}\cap(\bar{B}\cup C)=(\overline{A\cup B})\cup(\bar{A}\cap C)$	YES	147	33	5.439	11	9	0.660	MEMORY OVERFLOW		16.426	7	8	0.706
23. $(A\cup B)\cup(\bar{A}\cap C)=(\bar{A}\cap\bar{B})\cup(\bar{A}\cap C)$	NO	MEMORY OVERFLOW		31.803	15	5	0.463	557	47	12.803	10	2	0.424
24. $A\cap(B\cup(C\cup\emptyset))\subseteq(A\cap B)\cup(A\cap C)-(A\cap D)$	NO	729	729	26.771	3	3	0.357	MEMORY OVERFLOW		14.043	6	4	0.550
25. $\gamma(A)\cap\gamma(B)\subseteq\gamma(\overline{A\cup B})$	NO	11	2	0.350	11	2	0.360	12	2	0.357	12	2	0.310
26. $A=B\rightarrow\gamma(A)\subseteq\gamma(B)$	YES	7	3	0.260	7	3	0.257	7	3	0.246	7	3	0.254
27. $\smile(A))\cap\smile(B))=\smile(A\cap B)$	NO	TEMPO		-	14	6	0.610	MEMORY OVERFLOW		9.307	4	2	0.416
28. $\gamma(A\cap(A\cup B))=\gamma(A\cap B)$	YES	309	71	11.316	17	9	0.750	MEMORY OVERFLOW		16.806	12	9	0.780
29. $\smile(A\cap(\bar{A}\cup B))=\smile(A\cap B)$	NO	TEMPO		54.0	8	4	0.616	MEMORY OVERFLOW		13.856	6	5	0.607
30. $(\frown(A\cup B)\subseteq(\frown A)\cup(\frown B)$	-	MEMORY OVERFLOW		53.517	MEMORY OVERFLOW		53.830	MEMORY OVERFLOW		14.669	MEMORY OVERFLOW		24.516
31. $\gamma((A\cap B)-C))\subseteq\gamma(A)\cdot\gamma(B-C)$	YES	452	60	11.789	452	60	11.972	14	10	0.730	14	10	0.740
32. $\smile((A-B)\cap(A-C))=\smile(A-(A\cup C))$	YES	MEMORY OVERFLOW		10.213	10	8	0.940	MEMORY OVERFLOW		43.688	11	10	1.297
33. $(A\cap B)\cap C=A\cap(B\cap C)$	YES	24	105	3.660	6	8	0.366	MEMORY OVERFLOW		-	6	6	0.480
34. $(A\cup B)\cup C=A\cup(B\cup C)$	YES	24	114	3.786	6	8	0.353	MEMORY OVERFLOW		-	6	6	0.524
35. $A\cap(B\cup C)=(A\cap B)\cup(A\cap C)$	YES	33	122	4.607	9	9	0.480	MEMORY OVERFLOW		-	8	9	0.777
36. $A\cap(\overline{B\cup C})=(A-B)-C$	YES	24	158	3.654	6	6	0.297	30	95	5.633	6	6	0.523
37. $(A\cap B=A\cap C\vee\bar{A}\cap B=\bar{A}\cap C)\rightarrow B=C$	YES	27	153	2.372	10	27	0.687	MEMORY OVERFLOW		-	6	28	0.684
38. $A\cap B=A\cap C\rightarrow B=C$	NO	20	207	2.672	3	23	0.317	6	138	1.683	1	17	0.204
39. $(A-B)-C=A-(B-C)$	NO	14	89	2.536	4	3	0.360	MEMORY OVERFLOW		-	4	5	0.543
40. $(A-\bar{B})\cdot(B-A)=(\bar{A}\cap B)-(\bar{A}\cap B)$	YES	32	126	6.073	8	2	0.713	MEMORY OVERFLOW		-	12	13	1.82

NOTES:

a) (i, j) has the following meaning:

(0, 0) - no c.p.'s are present
(1, 0) - c.p. for equality is present
(0, 1) - no c.p. for equality, c.p.'s for the function symbols $\cap$, $\cup$, $\sqcap$, $\sqcup$.
(1, 1) - c.p.'s for both equality and function

b) Memory overflow means problem with the memory.

c) "Tempo" means overtime condition (1 minute).

d) 1st column: result 'S' is theorem, 'N' is not theorem.
2nd column: number of unifiers
3th column: number of resolvents
4th column: execution time

After again for other (i,j), unifier, resolvent, time etc.

CONCLUSIONS

The analysis of the table shows that the efficiency of the system is, in general, improved by the introduction of the c.p.'s.

In some cases the improvement is considerable for instance in examples 7, 9, 18, 21, 23, 27, 29, 34, 35, 36, 37, 38, 39, 40.

The analysis shows that (0,1) is in some cases, a bad choice, for instance in examples 6, 16, 18, 19, 24, 28, etc.

In $(BAC)_0$ only for the two formulas below we did not of obtain a good result,

1) $(A \subseteq B \wedge B \subseteq C \rightarrow (\bar{A} \cap B) \cup (B \cap C) = \bar{A} \cap C$.

2) $((\bar{A} \cap B) \cup (B \cap C) = \bar{A} \cap C \wedge A \subseteq B \wedge B \subseteq C$.

For the others formulas with (0,0) or (0,1) or (1,1) we obtain the answer.

In $(BAC)_1$ 18 formulas presented problems and only 3 formulas are not denial.

In $(BAC)_2$ were made 20 experiments and we only had have goods results for 8 of them.

The present implementation was written in SPITBOL, running in a system IBM 370/165, at the Departamento de Informática da Pontifícia Universidade Católica do Rio de Janeiro, and for each terminal we had 150K of memory.

We do not consider our implementation as a final one. We are already working in modifications of these experience, and our purpose is to use it in theorem proving in Point Set Topology (to handle theorems which need the choice axiom).

BIBLIOGRAFIA

[1] Carvalho, Roberto Lins de. "Some Results in Automatic Theorem-Proving with Applications in Elementary set Theory and Topology" - Technical Report No. 71, Nov. 1974 - Department of Computer Science - University of Toronto, Canada.

[2] Peixoto, Silvia Regina Góes. "Experimentação para um sistema de prova automatica de teoremas" - Tese de Mestrado - Departamento de Informática da Pontifícia Universidade Católica do Rio de Janeiro - BRASIL - 1975.

[3] McCharen, J.D.; Overbrek, R.A., and Wos, L.A. "Problems and Experiments for and with Automated Theorem-Proving Programs - IEEE Transactions on Computers. Vol. C-25, No. 8, August 1976.

[4] Passos, E.P.L. and Galda, K. "Theorem Proving, using Quantifier Elimination for Theory for densely ordered sets without first or last element" Monographs nº 6/72 - Departamento de Informática da Pontifícia Universidade Católica do Rio de Janeiro - BRASIL.

[5] Passos, E.P.L. and Silveira, G.G.; "Application of Quantifier Elimination to Mechanical Theorem Proving - Implementation" - North-Holand - 1976, works from 3th Intern. Cong. of Cybernetics and System - August 1975 - Bucarest - Romenia.

[6] Robinson, J.A. "A Machine-Oriented Logic and Based on the Resolution Principle"

[7] Bledsoe, W.W. - "Splitting and Reduction Heuristics in Automatic Theorem Proving" - Artificial Intelligence 2 (1971).

[8] Boyer, R.S. - Locking: A Restriction of Resolution - Texas (1971).

[9] Nevine, A.J. - A Relaxation Approach to Splitting in an Automatic Theorem Prover. MIT (1974).

[10] Slagle, J.R. - Automatic Theorem Proving with Built-in Theories Including Equality, Partial Ordering, and Sets. J. ACM - January 72.