Proceedings of the 8th International Conference on Applied Informatics
Eger, Hungary, January 27–30, 2010. Vol. 1. pp. 115–125.

Empirical Investigation on Combining Tableaux and Resolution

Gergely Kovásznai[a], Gábor Kusper[b]

[a]Department of Information Technology, Eszterházy Károly College, Eger, Hungary

[b]Department of Computing Science, Eszterházy Károly College, Eger, Hungary

Abstract

There are several attempts to automate tableau calculi in first-order classical logic. The combining of tableaux and resolution is one of the directions of such investigations. In some published papers of ours, we proposed a general splitting technique on resolution derivations, by applying tableaux to represent distinct branches of derivations. In order to make tableaux sufficient for such a role, an additional and costy test must be applied for each clause which is being generated during a resolution derivation. In this paper, we investigate if it is worth to apply our splitting technique in practice, by empirical tests. For the sake of empirical investigations, we implemented several resolution calculi in Java, and then we improved each of them with our tableau method. Then, each of the implemented calculi have been executed over 1642 TPTP problems. In this paper, we present the results, from several aspects, and then, we conclude that tableaux are worth to be combined with resolution.

1. Introduction

In the last decades, *tableau calculi* have proved to be very expressive, easy to use, easy to implement, and quite universal in computer-based reasoning. Originally, Raymond Smullyan proposed this type of calculi in his famous book "First-Order Logic" [15]. In general, a tableau can actually be regarded as a *graph* whose vertices are *labeled with logical formulas*. The majority of tableau calculi apply not a general graph but rather a *tree*, like Smullyan's analytic tableau calculi [15].

It is a quite exciting question if it is worth to combine resolution-based and tableaux-based reasoning methods. *Hyper tableau* calculi (e.g., hyper tableaux [2, 3], constrained hyper tableaux [6], rigid hyper tableaux [11], and hyperS tableaux [8, 9] etc.) are well-known as attempts to combine hyper-resolution and tableaux. In this paper, we propose a general idea for combining resolution calculi and

tableaux. The goal is the same as in the case of hyper tableau calculi: to split (hyper-)resolution derivations into branches. First, we propose a general way of representing any resolution calculus (and illustrate it by examples), and then we introduce a novel method called *resolution tableaux*. Resolution tableaux are more general than hyper tableaux, since any resolution calculus (not only hyper-resolution) can be applied, like, e.g., binary resolution, input resolution, or lock resolution etc. We prove that any resolution tableau calculus inherits *the soundness and the completeness* of the resolution calculus which is being applied. By the use of resolution tableaux, any resolution derivation can be split into separate branches, hence resolution tableaux can be regarded as a kind of *parallelization* of resolution, as it will be illustrated by an example.

The structure of the paper is as follows. In Section 2, basic definitions and concepts are introduced. We propose resolution tableaux in Section 3, and then we give details on the results of empirical tests in Section 4.

2. Preliminaries

In the followings, we assume that the reader is familiar with the basic concepts of first-order logic. Nevertheless, let us present a few crucial concepts.

A *literal* is a formula either A or $\neg A$ where A is an atomic formula. A is classified as a positive, $\neg A$ as a negative literal.

A *clause* is a formula $L_1 \vee L_2 \vee \ldots \vee L_n$ where $n \geq 0$ and each L_i is a literal $(i = 1, \ldots, n)$. A clause can also be regarded as the set of its literals. The *empty clause* is denoted by $\bot$.

A clause is *positive* (negative) iff it consists of solely positive (negative) literals.

Two clauses are *independent* iff there is no variable that occurs in both of them.

$C\sigma$ is called an *instance* of a clause C where σ is a variable substitution. $C\sigma$ is a *new instance* if σ is a variable renaming and its range consists solely of new variables.

A clause C *subsumes* a clause D iff C has an instance $C\sigma$ such that $C\sigma \subseteq D$.

Given a formula A, let $\forall A$ denote the *universal closure* of A.

As usual, $M \models A$ denotes the fact that a formula A is *satisfied by a model* M. In the case of A being open, $M \models A$ iff $M \models \forall A$.

Two formulas A and B are *equivalent* (denoted by $A \sim B$) iff for any model M: $M \models A$ iff $M \models B$.

As it is well-known, the *most general unifier* (MGU) of two atomic formulas A and B is the most general variable substitution σ such that $A\sigma = B\sigma$. Let us generalize the definition of MGUs, as follows. The MGU of (A_1, B_1), $(A_2, B_2), \ldots,$ (A_n, B_n), where all A_i and B_i are atomic formulas, is the most general variable substitution σ such that $A_i\sigma = B_i\sigma$ for all $i = 1, \ldots, n$.

Tableaux are regarded as trees whose which vertices are labeled with formulas [15, 7]. Sometimes, for the sake of briefness, we regard a tableau as the set of all its branches. Similarly, a branch is often regarded as the sequence or the set of all the vertices in the branch. Furthermore, let us introduce the following notation:

Notation 2.1. Let $\mathcal{N}$ be a vertex set from a tableau.

1. Let $\widehat{\mathcal{N}}$ denote the conjunction of all the labels (formulas) in $\mathcal{N}$.
2. Let $\check{\mathcal{N}}$ denote the disjunction of all the labels (formulas) in $\mathcal{N}$.

Sometimes it is needed to regard a tableau as a sole formula. This is why we need the following definition:

Definition 2.2 (Formula Represented by a Tableau). The *formula* $\mathcal{F}(T)$ *represented* by a tableau T is defined inductively as follows:

1. If T consists of one single vertex labeled with a formula L, then
$$\mathcal{F}(T) \quad = \quad L$$
2. If T is a compound tableau, i.e., it is in the form as can be seen in Figure 1, where L is a formula and each T_i is a tableau, then
$$\mathcal{F}(T) \quad = \quad L \wedge \left(\bigvee_{i=1}^{n} \mathcal{F}(T_i) \right)$$

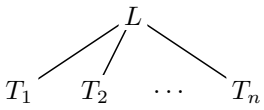


Figure 1: Compound tableau.

Let us note the following obvious fact, which says that any tableau can be regarded as the disjunction of its branches (as conjunctions).

Lemma 2.3. *For any tableau* T*,*
$$\mathcal{F}(T) \quad = \quad \bigvee_{B \in T} \widehat{B}$$

3. Resolution Tableaux

The aim is to introduce a general method for combining resolution calculi and tableaux. This is why a general way of representing resolution calculi is required. We regard a resolution calculus as a *set of inference rules*, which act on clauses. Each resolution inference rule is represented as a function which can assign a clause to one or more clauses. Every time when applying such a rule, it is needed to specify a clause set (denoted by $\mathcal{I}$ and called the *input clause set*) and a sequence of clauses (denoted by $\mathbf{d}$ and called the *resolution derivation*).

Definition 3.1 (Resolution Inference Rule). A *resolution inference rule* is a function $res_{\mathcal{I},\mathbf{d}} : \mathcal{D}om \mapsto \mathcal{C}$, where

- $\mathcal{I}$ is a finite set of clauses;
- $\mathbf{d}$ is a finite sequence of clauses;
- $\mathcal{C}$ is the set of all the clauses;
- $\mathcal{D}om \subseteq P(\mathcal{C})$.

Let us illustrate by examples how well-known resolution calculi can be represented in this form. Of course, other resolution calculi could be represented in a similar way[1].

Example 3.2 (Binary Resolution). The resolution calculus, as was introduced by Robinson [13], can be represented by the set of the following resolution inference rules [1]:

1. *Binary Resolution*:

$$binres_{\mathcal{I},\mathbf{d}}\big(\ A \vee C\ ,\ \neg B \vee D\ \big) \quad = \quad (C \vee D)\sigma$$

   where σ is the most general unifier (MGU) of the atomic formulas A and B.

2. *(Positive) Factoring*:

$$factor_{\mathcal{I},\mathbf{d}}\big(\ C \vee A \vee B\ \big) \quad = \quad (C \vee A)\sigma$$

   where σ is the MGU of the atomic formulas A and B.

Example 3.3 (Linear Input Resolution). The linear input resolution calculus [5] can be represented by the same resolution inference rules as binary resolution, but the rule "Binary Resolution" is restricted as follows:

- one of the clauses $C \vee A$ and $D \vee \neg B$ must be the last element of $\mathbf{d}$;
- the other one must be an element of $\mathcal{I}$.

Example 3.4 (Hyper-Resolution). The (positive) hyper-resolution calculus, as was introduced by Robinson [14], can be represented by the following resolution inference rule:

$$\begin{gathered} hypres_{\mathcal{I},\mathbf{d}}(\ A_1 \vee C_1\ ,\ldots,\ A_n \vee C_n\ ,\ \neg B_1 \vee \ldots \vee \neg B_n \vee D\ ) \\ \| \\ (C_1 \vee \ldots \vee C_n \vee D)\sigma \end{gathered}$$

where

[1] It is to be remarked that $\mathcal{I}$ may be defined as a clause sequence (instead of a clause set) in the case of some resolution calculi, where the order of input clauses should not be neglected, like in SLD-resolution [10] and in lock resolution [4, 1, 5].

- $n \geq 1$;
- C_i is a positive or empty clause $(i = 1, \ldots, n)$;
- D is a positive or empty clause;
- A_i and B_i are atomic formulas $(i = 1, \ldots, n)$;
- σ is the MGU of $(A_1, B_1), \ldots, (A_n, B_n)$.

When a resolution calculus (as a set of resolution inference rules) is given, an appropriate tableau can be constructed for a given input clause set $\mathcal{I}$. Such a tableau is called a *resolution tableau*, and can be constructed in a quite simple way. In every deduction steps, some clauses are to be selected, each either from a given branch of the tableau or from the input clause set $\mathcal{I}$. To the selected clauses a resolution inference rule is applied, resulting in a clause D. First D must be split into *independent subclauses*, and then these subclauses are attached to the given branch, forming distinct new branches.

Let us define resolution tableaux inductively, as follows:

Definition 3.5 (Resolution Tableaux). Let $\mathcal{R}$ be a set of resolution inference rules. Let $\mathcal{I}$ be a clause set.

1. One single vertex labeled with $\top$ is a *resolution tableau* for $\mathcal{I}$ w.r.t. $\mathcal{R}$.
2. - Let T be a resolution tableau for $\mathcal{I}$ w.r.t. $\mathcal{R}$.
   - Let B be a branch of T.
   - Let $C_1, \ldots, C_n$ be *new instances* of clauses in $\mathcal{I} \cup B$.
   - Let $res \in \mathcal{R}$ such that $res_{\mathcal{I},B}$ is defined on $C_1, \ldots, C_n$, and let
   $$D = res_{\mathcal{I},B}(C_1, \ldots, C_n)$$
   - Let $D = D_1 \vee \ldots \vee D_k$ such that each distinct D_i and D_j are *independent* clauses $(i, j = 1, \ldots, k)$.[2]

   The tableau that can be seen in Figure 2 is a *resolution tableau* for $\mathcal{I}$ w.r.t. $\mathcal{R}$.

Let us point out that it is mandatory to generate *new instances* of the clauses which have been selected. Note that resolution tableau branches can be regarded (and are used) as separate resolution derivations.

A tableau calculus is regarded sound and complete in the following case: any clause set $\mathcal{I}$ is unsatisfiable iff a closed tableau exists for $\mathcal{I}$. A *closed resolution tableau* is defined as follows:

[2]Furthermore, one can additionally demand that no D_i can further be split into independent subclauses $(i = 1, \ldots, k)$. In this case, decomposition of clauses is unique, and can easily be solved algorithmically.

[3]New vertices labeled with $D_1, \ldots, D_k$ are attached to the leaf of B.

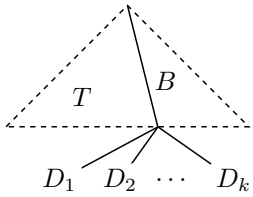


Figure 2: Attaching a clause to a branch B of a resolution tableau T.[3]

Definition 3.6 (Closed Resolution Tableaux). A resolution tableau is *closed* iff each of its branches contains $\bot$.

Assume a resolution calculus $\mathcal{R}$ which is sound and complete in first-order logic (or in a fragment of first-order logic). It is quite obvious that the resolution tableau calculus applying $\mathcal{R}$ inherits soundness and completeness. For example, since hyper-resolution is sound and complete in first-order logic, so is the hyper-resolution tableau calculus[4]. The linear input resolution tableau calculus[5] is sound and complete in Horn logic.

Theorem 3.7. *If a resolution calculus $\mathcal{R}$ is sound and complete (in a fragment of first-order logic), then so is the resolution tableau calculus applying $\mathcal{R}$.*

Proof.

1. Soundness:
   It is to show that if there is a closed resolution tableau for $\mathcal{I}$ w.r.t. $\mathcal{R}$, then $\mathcal{I}$ is unsatisfiable.

   If $\mathcal{R}$ is sound, then each inference rule $res \in \mathcal{R}$ preserves satisfiability. Let

   $$res(C_1, \ldots, C_n) \quad = \quad D_1 \vee \ldots \vee D_k$$

   where each distinct D_i and D_j are independent. It can be seen that for any model M:

   $$\text{if } M \models C_1, \ldots, C_n, \text{ then } M \models D_1 \vee \ldots \vee D_k.$$

   Because of independence:

   $$\forall\,(D_1 \vee \ldots \vee D_k) \quad \sim \quad \forall D_1 \vee \ldots \vee \forall D_k$$

   Summing up, for any model M:

   $$\text{if } M \models C_1, \ldots, C_n, \text{ then } M \models D_1 \text{ or } M \models D_2 \text{ or } \ldots \text{ or } M \models D_k.$$

[4]I.e., the resolution tableau calculus applying hyper-resolution.
[5]I.e., the resolution tableau calculus applying linear input resolution.

Hence, if $\mathcal{I}$ was satisfiable, then at least one branch could not be closed.

2. Completeness:
   It is to show that if $\mathcal{I}$ is unsatisfiable, then there is a closed resolution tableau for $\mathcal{I}$ w.r.t. $\mathcal{R}$.

   This fact is even more obvious than in the case of soundness. Since $\mathcal{R}$ is complete, there is a resolution refutation from $\mathcal{I}$. Each tableau branch can actually be regarded as a "simplified" variant of that refutation, i.e., only subclauses occuring in the refutation can occur in the branch. Since $\bot$ is deduced in the refutation and all literals of the clauses can be resolved out, obviously $\bot$ can occur in each branch.

□

Note that the fact that $D_1, \ldots, D_k$ are pairwise independent has been employed only in the soundness proof.

Example 3.8 (Linear Input Resolution Tableaux). Consider the following input clause set:

$$\mathcal{I} = \left\{ \begin{array}{c} M(a, s(c), s(b)) \\ P(a) \\ M(x, x, s(x)) \vee D(y, x) \\ \neg M(x, y, z) \vee D(x, z) \\ \neg P(x) \vee \neg M(y, z, u) \vee \neg D(x, u) \vee D(x, y) \vee D(x, v) \\ \neg D(a, b) \end{array} \right\}$$

a, b, c are constants, u, v, x, y, z are variables.

In Figure 3, a closed resolution tableau for $\mathcal{I}$ w.r.t. linear input resolution (c.f. Example 3.3) can be seen.

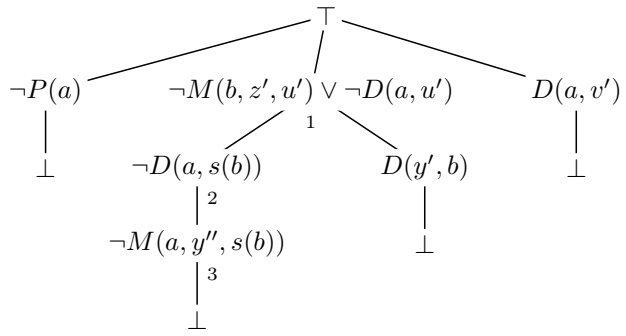
First, the input clauses $\neg P(x) \vee \neg M(y, z, u) \vee \neg D(x, u) \vee D(x, y) \vee D(x, v)$ and $\neg D(a, b)$ are selected; the atomic formulas $D(x', y')$ and $D(a, b)$ are resolved upon by MGU $\{:= x'a, := y'b\}$. As can be seen, the resolvent is split into four independent subclauses. Three branches can obviously get closed by resolving with the unit input clauses $P(a)$ and $\neg D(a, b)$.

Let us focus on the branch which contains $\neg M(b, z', u') \vee \neg D(a, u')$. Since the basis is linear input resolution, this clause (as the label of the last vertex in the branch) must be resolved with an input clause. Currently, that input clause is $M(x, x, s(x)) \vee D(y, x)$. The resolvent is split into two subclauses.

The consequent steps can be similarly performed.

4. Empirical Investigations

In order to examine the practical usefulness of resolution tableaux, we implemented four different resolution calculi: *binary resolution*, *linear resolution*, *linear input resolution*, and *hyper-resolution*. Let us emphasize that purely the basic variants



Selected input clauses:

[1]: $M(x, x, s(x)) \vee D(y, x)$
[2]: $\neg M(x, y, z) \vee D(x, z)$
[3]: $M(a, s(c), s(b))$

Figure 3: Closed linear input resolution tableau.

of those calculi have been implemented. We also implemented improved variants of the aforementioned calculi, only by applying resolution tableaux.

Then, we tested all the original and improved calculi on 1642 TPTP problems [16] (from 232 files). As it had been expectable, those calculi could not solve most of the problems in a reasonable time limit. What we primarily tried to investigate are the following questions:

- *How often* an improved calculus can solve such a problem that the original calculus cannot solve?
- If both an original calculus and its improved variant can solve a problem, *how much time* is gained by using the improved calculus?

	Solution		**Time**		
	gained	*lost*	*gained*	*lost*	*gained/lost*
Binary	0.49%	0.59%	10.91%	29.09%	2.93%
Linear	12.3%	0%	7.45%	0.62%	626.67%
Linear input	27.57%	0%	4.48%	0%	–
Hyper	2.01%	0.97%	46.46%	18.77%	310.43%

Table 1: Empirical results.

Table 1 contains all the statistical data we have collected. Let us give an overview on the columns of the table:

- In connection with those cases when either an original calculus or its improved variant does not provide a solution (in a reasonable time limit) for the same problem, let us summarize the following data:

    1. **Gained solutions:** The frequency of those cases when the original calculus does not provide a solution, but the improved calculus *does*.
    2. **Lost solutions:** The frequency of those cases when the original calculus provides a solution, but the improved calculus *does not*.

- In connection with those cases when both an original calculus and its improved variant provide a solution for the same problem, let us summarize the following data:

    1. **Gained time:** The frequency of those cases when the improved calculus provides a solution in *less time* than the original calculus.
    2. **Lost time:** The frequency of those cases when the improved calculus provides a solution in *more time* than the original calculus.
    3. **Gained time/Lost time:** We calculated the ratio of the length of the gained time to the length of the lost time in order to illustrate how it is worth to apply the improved calculus, in respect to execution time.

As it can be noticed, binary resolution tableaux do not seem very practical, in contrast with linear resolution tableaux and linear input resolution tableaux, which are absolutely worth to apply.

The conclusion in the case of hyper-resolution tableaux is quite ambiguous. Since hyper-resolution itself can be regarded as a quite powerful proof method, only in a few cases can hyper-resolution tableaux provide extra solutions. Nevertheless, the frequency of the cases when hyper-resolution tableaux shorten execution time is extremely high.

5. Conclusion

We proposed a novel technique for enhancing resolution calculi by applying tableaux, in order to split resolution derivations, i.e., practically speaking, to parallelize resolution. We proved that the resulting calculus inherits the soundness and the completeness of the resolution calculus which is being applied.

In order to investigate whether our approach is worth to employ, we ran empirical tests. 4 resolution calculi (binary resolution, linear resolution, linear input resolution, and hyper-resolution) were chosen to be implemented in Java. We also implemented their improved variants, i.e., we enhanced them by applying tableaux. Then, we chose 1642 TPTP problems, which we loaded by the use of the ANTLR Parser Generator, and finally, for all those problems, all chosen calculi were executed. We made statistical analysis in order to show that resolution tableaux are quite advantageous.

However, one could note that the testing on TPTP problems does probably not provide life-like scenarios. TPTP problems do mostly formalize quite special situations and targeted to testing the capabilities and the overall performance of theorem provers. Nevertheless, a future plan of ours is to test our approach on deductive databases, since

- databases store a huge amount of data, thus, the shortening and parallelizing of derivations is worth considering;
- data tables consist of ground literals, thus, it is probably very often that clauses can be split into subclauses.

References

[1] L. Bachmair, H. Ganzinger, Resolution Theorem Proving, in: J. A. Robinson, A. Voronkov, Handbook of Automated Reasoning, Elsevier and MIT Press, North-Holland, Amsterdam, 2001, Vol. 1, Chapter 2, pp. 19-99.

[2] P. Baumgartner, U. Furbach, I. Niemelä, Hyper Tableaux, *Lecture Notes in Computer Science*, Vol. 1126, 1996, pp. 1-17.

[3] P. Baumgartner, Hyper Tableaux – The Next Generation, *Lecture Notes in Artificial Intelligence*, Vol. 1397, 1998, pp. 60-76.

[4] R.S. Boyer, Locking: A Restriction of Resolution, PhD thesis, University of Thexas at Austin, Austin, USA, 1971.

[5] C. L. Chang, R. C. T. Lee, Symbolic Logic and Mechanical Theorem Proving, Academic Press, 1973.

[6] J. van Eijck, Constrained Hyper Tableaux, *Lecture Notes in Computer Science*, Vol. 2142, 2001, pp. 232-246.

[7] R. Hähnle, Tableaux and Related Methods, in: J. A. Robinson, A. Voronkov, Handbook of Automated Reasoning, Elsevier and MIT Press, North-Holland, Amsterdam, 2001, Vol. 1, Chapter 3, pp. 100-178.

[8] G. Kovásznai, HyperS Tableaux – Heuristic Hyper Tableaux, *Acta Cybernetica*, Vol. 17, 2005, pp. 325-338.

[9] G. Kovásznai, Multi-Hyper Tableaux in Automated Theorem Proving, PhD thesis (in Hungarian), University of Debrecen, Debrecen, Hungary, 2007.

[10] R. A. Kowalski, Logic for Problem Solving, Elsevier North Holland, Amsterdam, 1979.

[11] M. Kühn, Rigid Hypertableaux, *Lecture Notes in Artificial Intelligence*, 1997, Vol. 1303, pp. 87-98.

[12] R. Letz, J. Schumann, S. Bayerl, W. Bibel, SETHEO: A High-Performance Theorem Prover, *Journal of Automated Reasoning*, Vol. 8(2), 1992, pp. 183-212.

[13] J. A. Robinson, A Machine-Oriented Logic Based on the Resolution Principle, *Journal of the ACM*, Vol. 12, 1965, pp. 23-41.

[14] J. A. Robinson, Automated Deduction with Hyper-Resolution, *International Journal of Computer Mathematics*, Vol. 1, 1965, pp. 227-234.

[15] R. M. Smullyan, First-Order Logic, Springer-Verlag, Berlin, Germany, 1968.

[16] G. Sutcliffe, The TPTP Problem Library and Associated Infrastructure: The FOF and CNF Parts, v3.5.0, *Journal of Automated Reasoning*, Vol. 43(4), 2009, pp. 337-362.

Gergely Kovásznai
email: kovasz@aries.ektf.hu

Gábor Kusper
email: gkusper@aries.ektf.hu