Tableaux + Constraints

Martin Giese and Reiner Hähnle

Chalmers University of Technology
Department of Computing Science
S-41296 Gothenburg, Sweden
{giese|reiner}@cs.chalmers.se

Abstract. There is an increasing number of publications in which the analytic tableaux calculus is combined with technology based on constraint solving. Although the details, as well as the purpose of these combinations vary widely, the results are invariably referred to as "constraint tableaux" or sometimes "constrained tableaux". We review some of the combinations and propose a more differentiated nomenclature.

1 Classification of Approaches

In this paper, we present an overview of previous work which in some way uses constraints in an analytic tableau framework.

We classify the various approaches syntactically, according to the location *where* constraints are used in the tableau. Semantically of course, this corresponds to *what* is being constrained in each particular case, so it is not too surprising that approaches in the same categories tend to use constraints for related purposes.

Another (orthogonal) possibility for classification concerns the domain over which constraints are interpreted. In most of the approaches mentioned in this paper, the instantiations of free variables are constrained, and the domain is accordingly that of ground terms. It is also possible to use constraints that refer to the labels in some labeled tableau system [3], in which case quite different domains make sense, see the end of Sect. 4. We concentrate on a classification by location because that seems to give a wider variation.

In our proposed nomenclature, we will use the word "constrained" as past participle of the verb "to constrain", while "constraint" is a noun, denoting an entity used to constrain something.

This nomenclature should be understood as a first proposal and a basis for further discussion. We would also like to encourage readers to notify us of relevant work that has not been included in this survey.

2 Constrained Tableaux

The most obvious entity that might be constrained is, of course, the whole tableau. We propose to call such combinations simply *constrained tableau* calculi.

Constrained tableaux are used by Degtyarev and Voronkov [1, 2] in their calculi for ordered superposition based equality handling. The constraints are mainly used to record ordering restrictions between the ground instantiations of free variables. But one also sees another interesting use of constraints, namely that unification constraints for branch closure are added to the global constraint instead of applying substitutions globally. For instance, given two literals $p(X)$, $\neg p(a)$ on the same branch, a constraint $X \equiv a$ would be generated separately from the formulae on the tableau. In a system using unification, the substitution $[X/a]$ would be applied globally. This might enable rewriting steps on other branches, which can shown to be redundant.

Another approach that uses a global constraint on free variable instantiations is employed in the theorem prover SETHEO [15]. Here, the idea is to check regularity of tableaux by gathering constraints which ensure that the free variables are not instantiated in a way that renders two formulae on the same branch identical. Accordingly, this approach uses 'dis-unification' constraints.

3 Constrained Formula Tableaux

Instead of accumulating constraints globally, it is possible to add them to the formulae or signed formulae on a tableau branch. We will then speak of *constrained formula* tableaux.

For instance, if a rule application calls for some substitution $[X/a]$, a constrained tableau method would typically add the constraint $X \equiv a$ to the global constraint. In a constrained formula tableau, the constraint $X \equiv a$ is instead added to every new formula produced by the rule application. One also usually has *constraint propagation*, which means that the constraints of the formulae in the premisses of a rule application accumulated in the constraint of the conclusion.

This approach was used in several papers by Giese for equality handling [8, 7] and simplification rules [5, 9, 10] in a non-backtracking context. The property of constrained formula tableaux exploited in this context is that a rule which merely adds a constrained formula to a tableau branch never introduces a backtracking point, while a rule which modifies a global constraint usually does.

4 Constrained Branch Tableaux

In between constrained tableau and constrained formula tableaux one has the possibility to attach constraints to the branches of a tableau. We call this approach *constrained branch* tableaux.

This was done by van Eijck [22] for constraints on free variable instantiations with a similar intention as described above for constrained formulae. A branch containing a constraint C that is not satisfied by the instantiation for the free variables may be regarded as closed. As formulae within branches are conjunctively connected, and most constraint languages are closed under intersection, this does usually not result in greater expressiveness, but gives more succinct calculi, when branch closure rules are more complicated than a simple test for a complementary pair. Note that these constraints on branches could be simulated by attaching their negations to "false" literals in a constrained formula approach.

To illustrate the constrained branch approach, assume again that a certain tableau expansion requires a substitution $[X/a]$. Van Eijck handles this with constrained branch tableaux by splitting the branch on which the rule would be applied. On the first branch, the rule is actually applied, performing the necessary substitution X/a on the result. On the second branch, the constraint $X \not\equiv a$ is added. The effect is that if the final closing substitution substitutes a for X, the constraint on the second branch becomes false and closes it, so the proof on the (probably simpler) first branch is sufficient. Otherwise, a proof has to be found on the second branch, which does not profit from the rule application.

Note that there are probably efficiency problems with this approach. Take n possible, unrelated rule applications, each requiring a different instantiation. While these can be applied consecutively in any order in a constrained formula tableau, the constrained branch approach generates 2^n branches to account for all combinations of constraint satisfaction.

A very different kind of constrained branch tableau was proposed by Hähnle and Ibens [12], and later refined by Goubault and Schmitt [11]. These contributions deal with labeled tableau systems for linear temporal logic. Integer ordering constraints are used to keep track of the required ordering of points in time. Branches can be closed exactly if they contain unsatisfiable constraints. In other words the question of branch closure is delegated to the constraint system. This reinforces the point made above that constrained branch tableaux are best used in situations, where branch closure is very complicated.

A similar approach is used in the *resource tableaux* recently proposed by Galmiche, Mery and Pym [4] for the logic of bunched implications. Their constraints also refer to labels attached to formulae on tableau branches, but they are interpreted over some preordered monoid. As before, the constraints are essentially used to reduce a complicated branch closure decision to a constraint satisfaction problem.

5 Constrained Subformula Tableaux

To our knowledge, the smallest syntactic entities that have had constraints attached to them are subformulae of the formulae in tableaux. Such *constrained subformula tableaux* have been proposed by Peltier [18, 19]. Instead of just juxtaposing formulae and constraints, Peltier intertwines them. For instance, in a formula

$$\forall x.\forall y.(x = y \vee p(x, y)) \quad ,$$

$x = y$ plays the role of a constraint, which makes the formula $p(x, y)$ available only if x and y are instantiated by different ground terms. The symbol $=$ denotes *syntactic* equality, and not an equality predicate in the usual sense. On the other hand, quantification over the variables x, y used in the syntactic equality constraint is possible. This means that the semantics of such a mixture of formulae and constraints can only be defined with respect to models, in which the elements of the carrier set are ground terms (such as Herbrand models). The possibility of attaching different constraints to different parts of a larger formula is a potential advantage of Peltier's approach.

6 Constraint Merging Tableaux

There is also a family of approaches which don't necessarily add constraints to the tableaux at all. The constraints are used as an intermediate representation to find out whether a tableau is closed. This invariably requires some kind of merging operation on sets of closing instantiations for different sub-tableaux. As these sets are of course represented by some form of constraints, one might as well talk about *constraint merging tableaux*.

Constraint merging tableaux have been introduced in the incremental closure technique of Giese [6, 9]. Van Eijck attempted an implementation in a lazy functional programming language [20, 21], but the resulting provers were unfortunately incomplete. A successful functional implementation was recently given by Sörensson and Hähnle [13].

Constraint merging tableaux can be combined very nicely with constrained formulae. First, constraint merging tableaux are used to avoiding backtracking, and constrained formula calculi don't require backtracking, as we noted earlier. And second, as tableau closure is formulated and checked using constraints anyway, having constraints on the formulae hardly means any extra complexity.

7 Constraint Tableaux

There is finally one approach[17] in which the whole tableau is actually replaced by a potentially infinite but lazily computed constraint. Here, the constraint *is* the tableau, so we can finally talk about a *constraint tableau.*

There are certain difficulties in finding a correct definition for constraint semantics and satisfiability testing for lazily computed infinite constraints. On the other hand, once correct definitions are found, it might turn out that one can reason algebraically on constraints in a way that is not obvious from a tableau representation.

Note that the satisfiability check as presented by Ó Nualláin corresponds quite closely to the ones described under "Constraint Merging Tableaux". Still, this might not necessarily be implied by the constraint tableau idea. That's why we propose keeping them in a separate category.

8 As Yet Unclassified Approaches

One further line of work where the term "constraint propagation" is used in a tableau context, is that of Massacci [16]. What Massacci means by that term seems to be essentially unit propagation, using simplification rules. There are certainly no separate syntactical entities referred to as constraints in this work, so we chose to keep it separate from the other techniques presented in this paper.

It might be interesting to apply our classification to non-tableau formalisms, like the work of Kreitz and Pientka on inductive theorem proving [14], which is formulated in a matrix-based setting. However, this is future work.

9 Conclusion

We attempted to give an overview of the existing literature on combinations of tableaux and constraint systems. We proposed a systematic, differentiated nomenclature, which might be used in the future to distinguish between the different ideas and methods.

We hope to receive comments from the community, both on work we failed to mentioned here, and on the proposed nomenclature.

Acknowledgments

The authors would like to thank the anonymous referees for suggesting a number of improvements.

References

1. Anatoli Degtyarev and Andrei Voronkov. The undecidability of simultaneous rigid E-unification. *Theoretical Computer Science*, 166(1-2):291–300, October 1996.
2. Anatoli Degtyarev and Andrei Voronkov. What you always wanted to know about rigid E-unification. Technical Report 143, Comp. Science Dept., Uppsala University, 1997.
3. Dov M. Gabbay. *Labelled Deductive Systems*, volume 1—Foundations. Oxford University Press, 1996.
4. D. Galmiche, D. Mery, and D. Pym. Resource tableaux (extended abstract). In J. Bradfield, editor, *16th Int. Workshop on Computer Science Logic, CSL 2002, Edinburgh, Scotland*, volume 2417 of *LNCS*. Springer, 2002.
5. Martin Giese. A first-order simplification rule with constraints. In Peter Baumgartner and Hantao Zhang, editors, *3rd Int. Workshop on First-Order Theorem Proving (FTP), St. Andrews, Scotland, TR 5/2000 Univ. of Koblenz*, pages 113–121, 2000.
6. Martin Giese. Incremental closure of free variable tableaux. In Rajeev Goré, Alexander Leitsch, and Tobias Nipkow, editors, *Proc. Intl. Joint Conf. on Automated Reasoning, Siena, Italy*, volume 2083 of *LNCS*, pages 545–560. Springer-Verlag, 2001.
7. Martin Giese. Model generation style completeness proofs for constraint tableaux with superposition. Technical Report 2001-20, Universität Karlsruhe TH, Germany, 2001. URL: `http://www.cs.chalmers.se/~giese/tr01-20.ps.gz`.
8. Martin Giese. A model generation style completeness proof for constraint tableaux with superposition. In Uwe Egly and Christian G. Fermüller, editors, *Proc. Intl. Conf. on Automated Reasoning with Analytic Tableaux and Related Methods, Copenhagen, Denmark*, volume 2381 of *LNCS*, pages 130–144. Springer, 2002.

9. Martin Giese. *Proof Search without Backtracking for Free Variable Tableaux.* PhD thesis, Fakultät für Informatik, Universität Karlsruhe, July 2002.
10. Martin Giese. Simplification rules for constrained formula tableaux. In Marta Cialdea, editor, *Proc. Intl. Conf. on Automated Reasoning with Analytic Tableaux and Related Methods, Rome, Italy*, LNCS. Springer, 2003. To appear.
11. Jean Goubault-Larrecq and Peter H. Schmitt. A tableau system for linear-time temporal logic. In Ed Brinksma, editor, *Proceedings of the Third International Workshop on Tools and Algorithms for the Construction and Analysis of Systems, TACAS'97, Enschede, the Netherlands April 02-04*, volume 1217 of *Lecture Notes in Computer Science*, pages 130–144. Springer, 1997.
12. Reiner Hähnle and Ortrun Ibens. Improving temporal logic tableaux using integer constraints. In Dov M. Gabbay and Hans Jürgen Ohlbach, editors, *Proc. International Conference on Temporal Logic, Bonn, Germany*, volume 827 of *LNCS*, pages 535–539. Springer-Verlag, 1994.
13. Reiner Hähnle and Niklas Sörensson. Fair constraint merging tableaux in lazy functional programming style. In Marta Cialdea, editor, *Proc. Intl. Conf. on Automated Reasoning with Analytic Tableaux and Related Methods, Rome, Italy*, LNCS. Springer, 2003. To appear.
14. Christoph Kreitz and Brigitte Pientka. Connection-driven inductive theorem proving. *Studia Logica*, 69(2):293–326, 2001.
15. Reinhold Letz, Johann Schumann, Stephan Bayerl, and Wolfgang Bibel. SETHEO: A high-perfomance theorem prover. *Journal of Automated Reasoning*, 8(2):183–212, 1992.
16. Fabio Massacci. Simplification: A general constraint propagation technique for propositional and modal tableaux. In Harrie de Swart, editor, *Proc. International Conference on Automated Reasoning with Analytic Tableaux and Related Methods, Oosterwijk, The Netherlands*, volume 1397 of *LNCS*, pages 217–232. Springer-Verlag, 1998.
17. Breanndán Ó Nualláin. Constraint tableaux. In *Position Papers presented at International Conference on Analytic Tableaux and Related Methods, Copenhagen, Denmark*, 2002.
18. Nicolas Peltier. Simplifying and generalizing formulae in tableaux: pruning the search space and building models. In Didier Galmiche, editor, *Proc. International Conference on Automated Reasoning with Analytic Tableaux and Related Methods, Pont-à-Mousson, France*, volume 1227 of *LNCS*, pages 313–327. Springer-Verlag, 1997.
19. Nicolas Peltier. Pruning the search space and extracting more models in tableaux. *Logic Journal of the IGPL*, 7(2):217–251, 1999. Available online at `http://www3.oup.co.uk/igpl/Volume_07/Issue_02/`.
20. Jan van Eijck. LazyTAP — a lazy tableau theorem prover for FOL. Manuscript, Available online at: `http://www.cwi.nl/~jve/dynamo/papers/lazyTAP.ps.gz`, December 2000.
21. Jan van Eijck. CHT—a theorem prover for constrained hyper tableaux, version 0.3. `http://www.cwi.nl/~jve/lazytab/CHT0-3.ps`, October 2001.
22. Jan van Eijck. Constrained hyper tableaux. In Laurent Fribourg, editor, *Proc. Computer Science Logic, Paris, France*, volume 2142 of *LNCS*, pages 232–246. Springer-Verlag, September 2001.