Synthesising Tableau Decision Procedures

Renate A. Schmidt[⋆]
[⋆]University of Manchester
renate.schmidt@manchester.ac.uk

Dmitry Tishkovsky[†]
[†]University of Manchester
dmitry.tishkovsky@manchester.ac.uk

Abstract

We formalise sufficient conditions for synthesising sound, complete, and tableau decision procedures for a logic of interest. Given a specification of the formal semantics of a logic, the method generates a set of tableau inference rules which can then be used to reason within the logic. The method guarantees that the generated rules form a calculus which is sound and constructively complete. If the logic can be shown to admit finite filtration with respect to a well-defined first-order semantics then adding a general blocking mechanism produces a terminating tableau calculus. The process of generating tableau rules can be completely automated and produces, together with blocking, an automated procedure for generating tableau decision procedures for logics.

1 Synthesising Tableau Calculus

Our interest is the problem of automatically generating a tableau calculus for a given logic. We assume that the logic is defined by a high-level specification of the formal semantics of the logic. Our aim is to turn this specification into a set of inference rules that provide a sound and complete tableau calculus for the logic. In addition, for a decidable logic we further want to generate a terminating calculus.

In previous work we have described a framework for turning sound and complete tableau calculi into decision procedures (Schmidt and Tishkovsky, 2008). The prerequisites in the framework are that the logic admits an effective finite model property shown by a filtration argument, and that

(i) the tableau calculus is sound and constructively complete, and

(ii) a weak form of subexpression property holds for tableau derivations.

Constructive completeness is a slightly stronger notion than completeness. It means that for every open branch in a tableau there is a model which reflects all the expressions (formulae) occurring on the branch. The subexpression property says that every expression in a derivation is a subexpression of the input expression with respect to a finite subexpression closure operator.

In order to be able to exploit the 'automatic termination framework' results in (Schmidt and Tishkovsky, 2008), our goal is to generate tableau calculi that satisfy the prerequisites (i) and (ii). It turns out that provided that the semantics of the logic is well-defined in a certain sense, the subexpression property can be imposed on the generated calculus. Crucial is the separation of the syntax of the logic from the 'extras' in the meta-language needed for the semantic specification of the logic. The process can be completely automated and gives, together with the unrestricted blocking mechanism and the results in (Schmidt and Tishkovsky, 2007, 2008), an automated procedure for generating tableau decision procedures for logics, whenever they have an effective finite model property with respect to a well-defined first-order semantics.

That the generated calculi are constructively complete has the added advantage that models can be effectively generated from open, finished branches in tableau derivations. This means that the synthesised tableau calculi can be used for model building purposes.

The method works as follows.

(a) The user defines the formal semantics of the given logic in a many-sorted first-order language so that certain well-definedness conditions hold.

(b) The method automatically reduces the semantic specification of the logic to Skolemised implicational forms which are then rewritten as tableau inference rules. These are combined with some default closure and equality rules.

The set of rules obtained in this way provides a sound and constructively complete calculus.

Theorem 1 *There exists a procedure for synthesising a sound and constructively complete tableau calculus from a well-defined specification of the semantics of the logic.*

Under certain conditions it is possible to refine the rules of the calculus in order to reduce branching and redundancy in the syntax of the calculus.

2 Synthesising Decision Procedures

If the logic can be shown to admit finite filtration, then the generated calculus can be automatically turned into a terminating calculus by adding the unrestricted blocking mechanism from (Schmidt and Tishkovsky, 2007):

$$\text{(ub)}: \frac{x \approx x,\ y \approx y}{x \approx y \mid x \not\approx y}.$$

Let $t < t'$, whenever the first appearance of term t' in the branch is strictly later than the first appearance of term t. The conditions that blocking must satisfy are:

(i) If $t \approx t'$ appears in a branch and $t < t'$ then all further applications of rules which introduce new terms to formulae containing t' are not performed within the branch.

(ii) In every open branch there is some node from which point onwards before any application of a rule introducing a new term all possible applications of the (ub) rule have been performed.

Theorem 2 *If for every formula φ and model satisfying φ, there is a finite filtrated model satisfying φ, then extending the tableau calculus which is synthesised in Theorem 1 with the* (ub)*-rule gives a terminating calculus.*

This gives a nondeterministic decision procedure which can be turned into a deterministic decision procedure by using appropriate search strategies that are fair.

3 Applications

The method can be applied to many familiar logics that are first-order representable. These include propositional intuitionistic logic, many standard, Kripke complete modal logics, and a wide set of description logics such as $\mathcal{ALCO}$ and $\mathcal{SO}$. $\mathcal{ALCO}$ is the description logic $\mathcal{ALC}$ with singleton concepts, or nominals. $\mathcal{SO}$ is the extension of $\mathcal{ALCO}$ with transitive roles. For these logics, the synthesised tableau calculi are very close to common 'classical' tableau calculi used for satisfiability checking in these logics.

4 Further Details

For further details we refer to (Schmidt and Tishkovsky, 2009), which introduces the method of synthesising tableau calculi, and to (Schmidt and Tishkovsky, 2008), which describes 'automatic termination' of a tableau calculi for logics having the effective finite model property.

5 Conclusion

The results of the paper can be regarded as a mathematical formalisation and generalisation of tableau development methodologies. The formalisation separates the creative part of tableau calculus development which needs to be done by a human developer and the automatic part of the development process which can be left to an automated (currently first-order) prover and an automated tableau synthesiser.

The tableau calculi generated are Smullyan-type tableau calculi, i.e., ground semantic tableau calculi. We believe that other types of tableau calculi can be generated using the same techniques.

Our future goal is to further reduce human involvement in the development of calculi by finding appropriate automatically verifiable conditions for optimal calculi to be generated.

References

Renate A. Schmidt and Dmitry Tishkovsky. Automated synthesis of tableau calculi, 2009. Submitted to *TABLEAUX'09*, http://www.cs.man.ac.uk/~dmitry/papers/astc2009.pdf.

Renate A. Schmidt and Dmitry Tishkovsky. A general tableau method for deciding description logics, modal logics and related first-order fragments. In Alessandro Armando, Peter Baumgartner, and Gilles Dowek, editors, *Proc. IJCAR'08*, volume 5195 of *Lect. Notes Comput. Sci.*, pages 194–209. Springer, 2008.

Renate A. Schmidt and Dmitry Tishkovsky. Using tableau to decide expressive description logics with role negation. In *Proc. ISWC'07*, volume 4825 of *Lect. Notes Comput. Sci.*, pages 438–451. Springer, 2007.