Tableaux Calculus for Unranked Logics [⋆]

Lia Kurtanidze[1] and Mikheil Rukhaia[2]

[1] Faculty of Informatics, Mathematics and Natural Sciences, Georgian University.
lia.kurtanidze@gmail.com
[2] Institute of Applied Mathematics, Tbilisi State University.
mrukhaia@logic.at

After long time of stagnation, tableaux-based reasoning methods became popular with the development of Semantic Web. All major Description Logic reasoners (e.g. Racer [6], FaCT++ [14], Pellet [12], etc.) use tableaux as their main reasoning method (see e.g. [7, 1]).

Tableaux calculus [13] is based on the principle of refutation. When a formula is given, it is negated and according to some rules decomposed to subformulas. This decomposition produces a tree of formulas. If every branch of the tree is closed (meaning that the branch contains contradictory formulas), then the given formula is valid. Tableau has advantage over other proof systems in that it can also build a model for satisfiable formula, or find a counter-example for non valid formula.

There are many refinements and modification of the tableaux calculus in the literature (see e.g. [10, 3]). This includes tableaux for intuitionistic, temporal, modal, substructural, nonmonotonic, many-valued logics and the like. To the best of our knowledge, there is no tableaux method for unranked logics defined in the literature and our aim is to provide such.

Unranked terms are first order terms in which function symbols do not have a fixed arity: The same symbol may have a different number of arguments in different places and some variables (aka sequence variables) can be instantiated by finite (possible empty) sequences of unranked terms.

In recent years, usefulness of sequence variables and unranked symbols has been illustrated in practical applications related to XML [9], knowledge representation [5], automated reasoning, rewriting, functional, functional logic, and rule-based programming, Common Logic [2], just to name a few. There are systems for programming with sequence variables. Probably the most prominent one is Mathematica, with a powerful rule-based programming language that uses (essentially first order, equational) unranked matching with sequence variables. Unranked function symbols and sequence variables bring a great deal of expressiveness in this language, permitting writing a short, concise, readable code.

Unification procedure for unranked terms has been given in [8]. The unranked terms $f(\overline{x}, x, \overline{y}, \overline{z})$ and $f(f(\overline{x}), x, a)$, where $\overline{x}, \overline{y}, \overline{z}$ are sequence variables and x is an individual variable unifies in three different ways with the substitutions $\{\overline{x} \mapsto (), x \mapsto f(), \overline{y} \mapsto (), \overline{z} \mapsto (f(), a)\}$, $\{\overline{x} \mapsto (), x \mapsto f(), \overline{y} \mapsto f(), \overline{z} \mapsto a\}$

[⋆] This work was supported by the project No. FR/51/4-102/13 of the Shota Rustaveli National Science Foundation.

and $\{\overline{x} \mapsto (), x \mapsto f(), \overline{y} \mapsto (f(), a), \overline{z} \mapsto ()\}$. Note that unranked terms may have infinitely many unifiers. For example, $f(\overline{x}, a)$ and $f(a, \overline{x})$ have the unifiers $\{\overline{x} \mapsto ()\}$, $\{\overline{x} \mapsto a\}$, $\{\overline{x} \mapsto (a, a)\}$, etc. Unranked unification, when one of the terms is ground (term without variables), called matching, is finitary.

In this talk we present an extended traditional tableaux inference system to work with formulas built over unranked terms. Unranked unification is used in tableaux as a mechanism that decides whether a path can be closed. It selects terms for replacement in quantification rules. We show, that the calculus is sound and complete, thus non-terminating in general. As it was mentioned above, unranked unification is not finitary in general, that is another reason of non-termination of the given algorithm. Finally, we illustrate the potential of the extended calculus in Web-related applications. In such applications unification problem is reduced to matching and is thus finitary. More details about this work can be found in [4].

We extend classical first-order tableaux for unranked logic. To reduce number of inference rules in the calculus, we consider formulas only in *negation normal form* (NNF). Classical first-order semantics defined for our language allows us to skolemize and transform formulae to NNF in a standard way.

The unranked tableaux calculus for formulae in negation normal form consists of the following rules

$$\frac{A \wedge B}{\begin{array}{c} A \\ B \end{array}} \wedge \qquad \frac{A \vee B}{A \quad\quad B} \vee \qquad \frac{\forall x A}{A[x \mapsto t]} \forall_i \qquad \frac{\forall \overline{x} A}{A[\overline{x} \mapsto \overline{s}]} \forall_s$$

Remark 1. Let us clarify the difference between the rules:

– The $\wedge$-rule chooses to decompose either or both operands in the single path. In contrast, the $\vee$-rule creates alternative paths for each operand.
– In the $\forall_i$ quantification rule the individual variable must be replaced by an individual term, while in case of the $\forall_s$-rule the sequence variable is replaced by an arbitrary sequence of terms, including the empty one.

A path of a tableaux is *closed* if it contains both, formula and its negation (modulo unification); otherwise it is *open*. A tableaux is closed if all of its paths are closed.

The tableaux calculus is sound and complete. Moreover, it is confluent; in other words, backtracking over the rules is not necessary. The only "backtracking" points are substitutions in the $\forall_i$ and $\forall_s$ quantification rules.

We demonstrate basic reasoning capabilities of our calculus using the Clique of Friends example from [11]. This example illustrates some basic reasoning for the Semantic Web. It does not use any particular Semantic Web language itself.

Consider a collection of address books where each address book has an owner and a set of entries, some of which are marked as `friend` to indicate that the person associated with this entry is considered a friend by the owner of the address book.

In the example we consider a collection that contains two address books, the first owned by `Donald Duck` and the second by `Daisy Duck`. Donald's address book has two entries, one for `Scrooge`, the other for `Daisy`, and only `Daisy` is marked as `friend`. Daisy's address book again has two entries, both marked as `friend`. In XML, this collection of address books can be represented in a straightforward manner.

The *clique-of-friends* of Donald is the set of all persons that are either direct friends of Donald (i.e. in the example above only `Daisy`) or friends of friends (i.e. `Gladstone` and `Ratchet`), or friends of friends of friends (none in the example above), and so on. To retrieve these friends, we have to define the relation "being a friend of" and its transitive closure.

We introduce the following abbreviations:

- Fixed arity function symbols: $f_r^o(\texttt{owner}), f_r^n(\texttt{name})$ and $f_r(\texttt{friend})$
- Flexible arity function symbols: $f_u^{ab}(\texttt{address-book})$ and $f_u^e(\texttt{entry})$
- Flexible arity predicate symbols: $p_u^{abs}(\texttt{address-books}), p_u^{fo}(\texttt{friend-of})$ and $p_u^{fof}(\texttt{friend-of-friend})$

Then the above-mentioned XML will be represented as the following fact in knowledge base:

$$\begin{aligned}\texttt{XML} \equiv p_u^{abs}(&f_u^{ab}(f_r^o(\texttt{Donald}), f_u^e(f_r^n(\texttt{Daisy}), f_r), f_u^e(f_r^n(\texttt{Scrooge}))),\\ &f_u^{ab}(f_r^o(\texttt{Daisy}), f_u^e(f_r^n(\texttt{Gladstone}), f_r), f_u^e(f_r^n(\texttt{Ratchet}), f_r)))\end{aligned}$$

We define *friend-of* and *friend-of-friend* relationships in the knowledge base:

$$\begin{aligned}p_u^{fo}(\overline{x}, \overline{y}) &\equiv p_u^{abs}(\_s, f_u^{ab}(f_r^o(\overline{x}), \_s, f_u^e(f_r^n(\overline{y}), f_r), \_s), \_s),\\ p_u^{fof}(\overline{x}, \overline{y}) &\equiv p_u^{fo}(\overline{x}, \overline{y}),\\ p_u^{fof}(\overline{x}, \overline{y}) &\equiv p_u^{fo}(\overline{x}, \overline{z}) \wedge p_u^{fof}(\overline{z}, \overline{y}),\end{aligned}$$

where $\_s$ is an anonymous sequence variable, that can be instantiated by an arbitrary sequence of terms, including the empty one.

The query to be asked is $KB \to \exists \overline{x}\, p_u^{fof}(\texttt{Donald}, \overline{x})$. Note that p_u^{fof} predicate can be represented in a single formula as $p_u^{fo}(\overline{x}, \overline{y}) \vee (\neg p_u^{fo}(\overline{x}, \overline{y}) \wedge p_u^{fo}(\overline{x}, \overline{z}) \wedge p_u^{fof}(\overline{z}, \overline{y}))$. Then the query to refute (after transforming to NNF) will be:

$$\begin{aligned}KB\ &\wedge \forall \overline{x}\ \neg p_u^{fo}(\texttt{Donald}, \overline{x})\\ &\wedge \forall \overline{x}\ (p_u^{fo}(\texttt{Donald}, \overline{x}) \vee \neg p_u^{fo}(\texttt{Donald}, \overline{y}) \vee \neg p_u^{fof}(\overline{y}, \overline{x}))\end{aligned}$$

The refutation consists of the following steps:

$$\frac{KB \wedge \forall \overline{x}\ \neg p_u^{fo}(\texttt{Donald}, \overline{x}) \wedge \ldots}{KB} \wedge$$
$$\frac{\forall \overline{x}\ \neg p_u^{fo}(\texttt{Donald}, \overline{x})}{\neg p_u^{fo}(\texttt{Donald}, \overline{x})\sigma} \forall_s$$
$$\times$$

where $\sigma = \{\overline{x} \mapsto \texttt{Daisy}\}$. If we would like to find all solutions, then we should continue by decomposing the second formula:

$$\cfrac{KB \wedge \forall \overline{x}\ \neg p_u^{fo}(\texttt{Donald}, \overline{x}) \wedge \ldots}{KB}\ \wedge$$

$$\forall \overline{x}\ (p_u^{fo}(\texttt{Donald}, \overline{x}) \vee \neg p_u^{fo}(\texttt{Donald}, \overline{y}) \vee \neg p_u^{fof}(\overline{y}, \overline{x}))$$

$$\vdots$$

$$\cfrac{(\neg p_u^{fo}(\texttt{Donald}, \overline{y}) \vee \neg p_u^{fof}(\overline{y}, \overline{x}))\theta}{\begin{array}{cc} \neg p_u^{fo}(\texttt{Donald}, \overline{y})\theta & \neg p_u^{fof}(\overline{y}, \overline{x})\theta \\ \times & \times \end{array}}\ \vee$$

where θ is either of substitutions: $\{\overline{x} \mapsto \texttt{Gladstone}, \overline{y} \mapsto \texttt{Daisy}\}$ and $\{\overline{x} \mapsto \texttt{Ratchet}, \overline{y} \mapsto \texttt{Daisy}\}$. From these substitutions we can read off the answer to our query:

$$\{\texttt{Daisy}, \texttt{Gladstone}, \texttt{Ratchet}\}$$

References

1. Christoph Benzmüller and Adam Pease. Progress in automating higher-order ontology reasoning. In *Proceedings of the Second International Workshop on Practical Aspects of Automated Reasoning*, 2010.
2. Common Logic Working Group. Common Logic Working Group Documents: Common Logic Standard. http://common-logic.org/, 2007.
3. Marcello D'Agostino, Dov M Gabbay, Reiner Hähnle, and Joachim Posegga. *Handbook of tableau methods*. Springer Science & Business Media, 2013.
4. Besik Dundua, Lia Kurtanidze, and Mikheil Rukhaia. Unranked Tableaux Calculus and its Web-related Applications. In *IEEE Conference on Electrical and Computer Engineering*, pages 1181–1184, 2017.
5. Michael R. Genesereth. Knowledge Interchange Format, draft proposed American National Standard (dpANS). Technical Report NCITS.T2/98-004, 1998.
6. Volker Haarslev and Ralf Möller. Racer system description. In *International Joint Conference on Automated Reasoning*, pages 701–705. Springer, 2001.
7. Ian Horrocks and Andrei Voronkov. Reasoning support for expressive ontology languages using a theorem prover. In *Foundations of Information and Knowledge Systems*, pages 201–218. Springer, 2006.
8. Temur Kutsia. Solving equations with sequence variables and sequence functions. *J. Symb. Comput.*, 42(3):352–388, 2007.
9. Temur Kutsia and Mircea Marin. Can context sequence matching be used for querying XML? In Laurent Vigneron, editor, *Proceedings of the 19th International Workshop on Unification UNIF'05*, pages 77–92, Nara, Japan, 22 April 2005.
10. Alan JA Robinson and Andrei Voronkov. *Handbook of automated reasoning*, volume 1. Elsevier, North Holland, 2001.
11. Sebastian Schaffert. *Xcerpt: a rule-based query and transformation language for the web*. PhD thesis, University of Munich, 2004.
12. Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet: A practical owl-dl reasoner. *Web Semantics: science, services and agents on the World Wide Web*, 5(2):51–53, 2007.
13. Raymond Smullyan. *First-Order Logic*. Springer, 1968.
14. Dmitry Tsarkov and Ian Horrocks. Fact++ description logic reasoner: System description. In *International Joint Conference on Automated Reasoning*, pages 292–297. Springer, 2006.