SIVA: An Educational Tool for the Tableau Reasoning Algorithm

Peter Paulovics, Júlia Pukancová and Martin Homola

Comenius University in Bratislava
Mlynská dolina, 84248 Bratislava
`paulovics95@gmail.com,{pukancova,homola}@fmph.uniba.sk`

Abstract. The tableau algorithm is one of the main reasoning algorithms employed by DL reasoners. It is also often taught as a reasoning technique at DL courses at universities. As the algorithm proves the existence of a model for a knowledge base by constructing a completion tree, the best way to understand this mechanism is to construct this tree graphically. Realization of this process on the blackboard is usually very laborious, and mainly the backtracking is chaotic. We have developed SIVA – a simulation tool for $\mathcal{ALC}$, visualizing the whole process from initializing a vocabulary and a knowledge base, to building a completion tree step by step by application of the tableau rules. It allows easy backtracking to any of the previous states. SIVA is freely available as an online application.

Keywords: description logics · tableau algorithm · visualization · education

1 Introduction

A significant amount of research in DL is focused on tableau reasoning algorithms [25,15,13,14,12,10,11, i.a.]. They are also taught to students during courses on knowledge representation and reasoning. Visualization of the tableau algorithm on the blackboard is helpful, but it is usually quite messy, especially in cases when the algorithm needs to backtrack. Indeed, a body of research in algorithms education has been devoted to development of tools for algorithm visualization [26], e.g., a number of them were developed for various tree-search algorithms.

There is a number of tools [22,28,7,23,24,4,31,18,17,3,6, i.a.] for ontology visualization. Though as noted by a recent survey [16], the issue of coupling reasoning and visualization has not yet been sufficiently explored in the literature. Notably, OntoTrack [18] uses an external reasoner in order to detect problems with the currently open ontology. Also, Protégé [9] with its plugins is able to run the reasoning over an ontology and visualize the inferred data. None of these tools is instrumental in easing the understanding how the tableau algorithm is working.

If we extend our outlook beyond ontologies and description logics, both LoTREC [8] and OOPS [30] are able to visualize tableau proofs for various modal logics. Also the Tree Proof Generator [29] visualizes tableau proofs for first-order logic. Out of this tools LoTREC allows to interact with the tableau once it is generated, otherwise no interaction is possible.

To our best knowledge, none of these tools allows full step-by-step control of the user while working with the visualized proof, where user has to figure out the next step of the proof. This is possible e.g. in the Tableau Editor [19,20] for first-order logic.

We have developed and implemented a web application enabling step-by-step simulation of the tableau algorithm for the DL $\mathcal{ALC}$ [25]. The tool enables to create a knowledge base; choose one of the decision problems from consistency checking, instance checking, and concept satisfiability; and consequently to build a completion tree proving the existence of a model. In accordance with the educational goal, each action needs to be done by the user, but the tool guides the user and evaluates the resulting state after each action.

We believe such a tool is a useful aid for students learning the tableau algorithm. In the future, we plan to extend the application to enable reasoning over more expressive DLs, and to enable the user to add custom tableau rules or blocking methods. We consider the latter two upgrades to be useful also for researchers in their works.

2 Description Logics

We build on top of the $\mathcal{ALC}$ DL [25]. A DL vocabulary consists of countably infinite mutually disjoint sets of individuals N_{I}, roles N_{R}, and atomic concepts N_{C}. Concepts are recursively built using constructors $\neg$, $\sqcap$, $\sqcup$, $\exists$, $\forall$, as shown in Table 1.

A knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ consists of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$. A TBox is a finite set of GCI axioms of the form $C \sqsubseteq D$, where C, D are concepts. An ABox is a finite set of concept assertions of the form $C(a)$, and role assertions of the form $R(a, b)$, where $a, b \in N_{\mathrm{I}}$, C is a concept, and $R \in N_{\mathrm{R}}$.

Table 1. Syntax and Semantics of $\mathcal{ALC}$

Constructor	Syntax	Semantics
complement	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
intersection	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
union	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
existential restriction	$\exists R.C$	$\{x \mid \exists y\, (x, y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
value restriction	$\forall R.C$	$\{x \mid \forall y\, (x, y) \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$
Axiom	Syntax	Semantics
concept incl. (GCI)	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
role incl. (RIA)	$R \sqsubseteq S$	$R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$
concept assertion	$C(a)$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$

An interpretation is a pair $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$, where $\Delta^\mathcal{I} \neq \emptyset$ is a domain, and the interpretation function $\cdot^\mathcal{I}$ maps each individual $a \in N_\mathrm{I}$ to $a^\mathcal{I} \in \Delta^\mathcal{I}$, each atomic concept $A \in N_\mathrm{C}$ to $A^\mathcal{I} \subseteq \Delta^\mathcal{I}$, each role $R \in N_\mathrm{R}$ to $R^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$ in such a way that the constraints in Table 1 are satisfied.

Two concepts C and C' are equivalent if $C^\mathcal{I} = C'^\mathcal{I}$ for every interpretation $\mathcal{I}$. An interpretation $\mathcal{I}$ satisfies an axiom φ (denoted $\mathcal{I} \models \varphi$) if the respective constraint in Table 1 is satisfied. It is a model of a knowledge base $\mathcal{K}$ if it satisfies all axioms included in $\mathcal{K}$.

Definition 1 (Model). *An interpretation $\mathcal{I}$ is a model of a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ (denoted $\mathcal{I} \models \mathcal{K}$) iff $\mathcal{I} \models \varphi$ for all $\varphi \in \mathcal{T} \cup \mathcal{A}$.*

The main DL decision problems deal with concept satisfiability checking, axiom entailment checking, and consistency checking of a knowledge base.

A concept is satisfiable w.r.t. a knowledge base if there is a model of this knowledge base interpreting the concept into a non-empty set.

Definition 2 (Satisfiability). *A concept C is satisfiable w.r.t. a knowledge base $\mathcal{K}$ iff there is a model $\mathcal{I}$ of $\mathcal{K}$ s.t. $C^\mathcal{I} \neq \{\}$.*

An axiom is entailed by a knowledge base if it is satisfied by all the models of this knowledge base.

Definition 3 (Entailment). *A knowledge base $\mathcal{K}$ entails an axiom φ (denoted $\mathcal{K} \models \varphi$) iff $\mathcal{I} \models \varphi$ for each model $\mathcal{I}$ of $\mathcal{K}$.*

A knowledge base is consistent, if there is at least one interpretation $\mathcal{I}$ such that $\mathcal{I} \models \mathcal{K}$, i.e. it has at least one model.

Definition 4 (Consistency). *A knowledge base $\mathcal{K}$ is consistent, if there is at least one interpretation $\mathcal{I}$ that is a model of $\mathcal{K}$.*

In fact, it is well known that all the decision problems stated above are reducible to the consistency checking problem [1].

Lemma 1 (Satisfiability reduction). *Given a DL knowledge base $\mathcal{K}$, a concept C is satisfiable w.r.t. $\mathcal{K}$ iff $\mathcal{K} \cup \{C(a)\}$ is consistent for some new individual a.*

Lemma 2 (Subsumption entailment reduction). *Given a DL knowledge base $\mathcal{K}$, and concepts C and D, $\mathcal{K} \models C \sqcup D$ iff $\mathcal{K} \cup \{C \sqcap D(a)\}$ is consistent for some new individual a.*

Lemma 3 (Assertion entailment reduction). *Given a DL knowledge base $\mathcal{K}$, a concept C, and an individual a, $\mathcal{K} \models C(a)$ iff $\mathcal{K} \cup \{\neg C(a)\}$ is inconsistent.*

A concept C is in negation normal form (NNF) if the complement constructor $\neg$ stands only in front of atomic concepts in C. From now on, by $\mathrm{nnf}(C)$ we denote a concept C' in NNF that is equivalent to C. At least one such concept always exists [1].

3 Tableau Algorithm

Thanks to reductions, it is sufficient to have an algorithm that solves the consistency checking problem, and so each of the listed decision problems can be solved via reduction. Perhaps the most common reasoning technique in DL is the tableau algorithm.

This algorithm works by proving the existence of a model of $\mathcal{K}$ by constructing it. More precisely, it works on a structure called *completion tree* that is iteratively extended by applying a set of *tableau rules* until it corresponds to a model, or until it is clear that this is not possible.

Definition 5 (Completion tree). *A completion tree (CTree) is a triple $T = (V, E, \mathcal{L})$ where (V, E) is an oriented graph with a set of nodes V and a set of edges E, and $\mathcal{L}$ is a labeling function that assigns a label to all nodes and edges of T as follows:*

- *$\mathcal{L}(v)$ is a set of concepts in NNF for a node $v \in V$,*
- *$\mathcal{L}(\langle x, y\rangle)$ is a set of roles for an edge $\langle x, y\rangle \in E$.*

We say that a node $y \in V$ is a *successor* of a node $x \in V$ in a CTree $T = (V, E, \mathcal{L})$ if $\langle x, y\rangle \in E$, and that a node $y \in V$ is a *descendant* of $x \in V$ if there is a path from x to y. A node y is an *R-successor* if y is a successor of x and $R \in \mathcal{L}(\langle x, y\rangle)$.

To find out whether a knowledge base $\mathcal{K}$ is consistent, the algorithm tries to construct a corresponding CTree, that is free of any local inconsistency, which is called a *clash*; i.e., it tries to assure that the CTree is *clash-free.*

Definition 6 (Clash). *There is a clash in $\mathcal{L}(v)$ for a node $v \in V$ of a CTree $T = (V, E, \mathcal{L})$, if $\{C, \neg C\} \in \mathcal{L}(v)$ for some concept C. A CTree is clash-free if none of its nodes contains a clash.*

However, some models may be infinite [1]. While the algorithm cannot continue the construction indefinitely, it relies on a technique called *blocking* to recognize infinite models by constructing a CTree that is a finite representation thereof.

Definition 7 (Blocked node). *Given a CTree $T = (V, E, \mathcal{L})$, a node $y \in V$ is blocked if it is a descendant of another node $x \in V$ s.t.:*

- *either $L(y) \subseteq L(x)$;*
- *or x is blocked.*

Finally, the algorithm is presented in Definition 8. For $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ it firstly initializes the CTree T by encoding into it the explicit statements from the ABox $\mathcal{A}$. After this initialization, tableau expansion rules are repetitively applied in order to expand a CTree. If a clash-free and complete CTree is found, the algorithm answers that $\mathcal{K}$ is consistent, otherwise that $\mathcal{K}$ is inconsistent.

Definition 8 (Tableau algorithm for consistency checking). ***Input:*** $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ *in NNF*
Output: *answers if $\mathcal{K}$ is consistent or not*
Steps:

1. *Initialize a CTree T as follows:*
   (a) $V := \{a \mid \textit{individual } a \textit{ occurs in } \mathcal{A}\}$*;*
   (b) $E := \{\langle a, b\rangle \mid R(a,b) \in \mathcal{A} \textit{ for some role } R\}$*;*
   (c) $\mathcal{L}(a) := \{\mathrm{nnf}(C) \mid C(a) \in \mathcal{A}\}$ *for all* $a \in V$*;*
   (d) $\mathcal{L}(\langle a, b\rangle) := \{R \mid R(a,b) \in \mathcal{A}\}$ *for all* $a, b \in V$*;*
2. *Apply tableau expansion rules from Definition 8 while at least one is applicable:*

$\sqcap$*-rule:*	*if* $C \sqcap D \in \mathcal{L}(x)$*,* $x \in V$ *is not blocked, and* $\{C, D\} \nsubseteq \mathcal{L}(x)$ *then* $\mathcal{L}(x) \longleftarrow \mathcal{L}(x) \cup \{C, D\}$
$\sqcup$*-rule:*	*if* $C \sqcup D \in \mathcal{L}(x)$*,* $x \in V$ *is not blocked, and* $\{C, D\} \cap \mathcal{L}(x) = \emptyset$ *then either* $\mathcal{L}(x) \longleftarrow \mathcal{L}(x) \cup \{C\}$ *or* $\mathcal{L}(x) \longleftarrow \mathcal{L}(x) \cup \{D\}$
$\forall$*-rule:*	*if* $\forall R.C \in \mathcal{L}(x)$*,* $x \in V$ *is not blocked, and* y *is R-successor of* x *and* $C \notin \mathcal{L}(y)$ *then* $\mathcal{L}(y) \longleftarrow \mathcal{L}(x) \cup \{C\}$
$\exists$*-rule:*	*if* $\exists R.C \in \mathcal{L}(x)$*,* $x \in V$ *is not blocked, and there is no R-successor y of x s.t.* $C \in \mathcal{L}(y)$ *then create a new anonymous node z and* $V \longleftarrow V \cup \{z\}$ *and* $\mathcal{L}(z) = \{C\}$
$\mathcal{T}$*-rule:*	*if* $C \sqsubseteq D \in \mathcal{T}$*,* $x \in V$ *is not blocked, and* $\mathrm{nnf}(\neg C \sqcup D) \notin \mathcal{L}(x)$ *then* $\mathcal{L}(x) \longleftarrow \mathcal{L}(x) \cup \{\mathrm{nnf}(\neg C \sqcup D)\}$

3. *Answer "$\mathcal{K}$ is consistent" if T is clash-free, otherwise answer "$\mathcal{K}$ is inconsistent".*

The process of building a CTree is nondeterministic. Specifically, there are two kinds of rules. All rules but the $\sqcup$-rule are *AND-rules*: they are deterministic rules with a single possible way of application. The $\sqcup$-rule is an OR-rule: a nondeterministic rules with multiple possible ways of application (two, in this case).

In order to explicitly represent the nondeterministic choices and backtracking done by the algorithm, we track them in a structure called a *tableau.*[1]

[1] We are aware that this notion of tableau is different from how this term is usually used in DL literature [1,25,15,13,14,12,10,11] however it is similar to the notion of a tableau proof in the tableau calculi for propositional and first order logic [2,27].

Definition 9 (Tableau). *A tableau is a triple $\tau = (V, E, \lambda)$ where (V, E) is a tree and λ is a labeling function s.t.:*

- *$\lambda(x) = T$ is a CTree for each $x \in V$;*
- *$\lambda(\langle x, y\rangle) = (r, u)$ if an AND-rule r was applied on a node u from $\lambda(x)$ yielding $\lambda(y)$ in which case y is a sole successor of x;*
- *$\lambda(\langle x, y\rangle) = (r, u, C)$ if an OR-rule r was applied on a node u from $\lambda(x)$ yielding $\lambda(y)$ in which $\mathcal{L}(u)$ was extended by C; in this case x has as many successors as many are possible nondeterministic choices for r on u.*

The root node in τ is the only node without incoming edges and represents the initial state of the reasoning process.

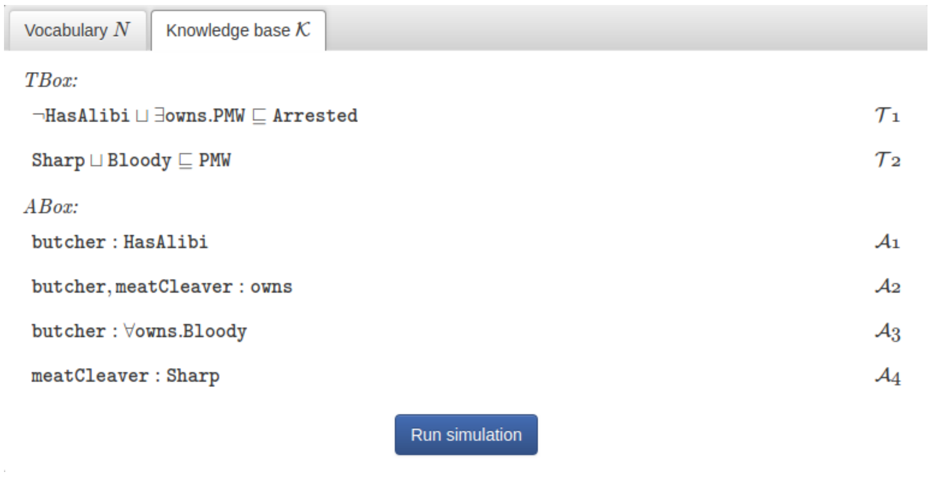
4 Implementation

Within our work, we have proposed and implemented a general purpose algorithm visualization system called SIVA (System of Interactive Visualizations of Algorithms). While designing SIVA, a considerable amount of attention has been paid to its extensibility and modularity. Currently, the most significant contribution of our work to the field of DL is enabling visualization and control over every step of the completion tree version of the tableau algorithm reasoning process in $\mathcal{ALC}$, which is provided by SIVA's modules we have implemented (`DL` module and its submodule `CTreeTableauReasoning`). SIVA's extensibility promotes its future development, due to which support for other description languages or visualization of different algorithms could be implemented.

Our implementation of the tableau algorithm for $\mathcal{ALC}$, utilizes an abstract tableau, which is not constrained to be used with any particular logic formalism. The tableau is stored as a tree-like data structure with exactly one of its nodes representing the initial and one node representing the current state of the reasoning process. Every tableau node except the initial one contains an exact description of reversible actions that transform the state represented by their parents to the state represented by them, which makes them freely traversable. In case of the completion tree version of the tableau algorithm for DL, each node of the tableau describes how an application of a tableau rule affects the completion tree.

The `CTreeTableauReasoning` module enables access to the completion tree data structure and its interactive visualization. The completion tree is represented by a graph-like data structure, whose nodes and edges are labeled by sets of concepts and roles, respectively. While applying tableau expansion rules, the application automatically extends respective labels of nodes and edges and transforms concepts to their NNF.

The `DL` module processes input in the form of DL expressions in original mathematical notation and provides a virtual keyboard with all necessary mathematical symbols. User input is automatically parsed and checked for errors, which have to be fixed before choosing a desired DL problem to solve. SIVA currently supports three types of problems within $\mathcal{ALC}$, which can be interactively solved using the completion tree version of tableau algorithm:



TBox:

$\neg\texttt{HasAlibi} \sqcup \exists\texttt{owns.PMW} \sqsubseteq \texttt{Arrested}$	$\mathcal{T}1$
$\texttt{Sharp} \sqcup \texttt{Bloody} \sqsubseteq \texttt{PMW}$	$\mathcal{T}2$

ABox:

$\texttt{butcher} : \texttt{HasAlibi}$	$\mathcal{A}1$
$\texttt{butcher}, \texttt{meatCleaver} : \texttt{owns}$	$\mathcal{A}2$
$\texttt{butcher} : \forall\texttt{owns.Bloody}$	$\mathcal{A}3$
$\texttt{meatCleaver} : \texttt{Sharp}$	$\mathcal{A}4$

Fig. 1. Visualization of a DL knowledge base

– concept satisfiability checking,
– instance checking,
– knowledge base consistency checking.

All instances of these decision problems are solved by reducing them to knowledge base consistency checking. SIVA's extensible design allows broadening its support beyond $\mathcal{ALC}$ and the mentioned set of reasoning problems in DL.

Functionality of external modules can be made accessible to users through SIVA's API by providing a so called *workspace* – a container component, which assembles all visual and functional components required to perform actions specified by the module. Our implementation of `CTreeTableauReasoning` contains a workspace that assembles visualizations and functionality of DL vocabularies, knowledge bases, completion trees and tableaux and enables communication between these components.

We have implemented SIVA as a heavily client-side oriented web application. It has been built using the React framework with TypeScript as the main scripting language. The application's internal workflow follows the Flux architecture, which is standardly used within React-based applications.

5 Web application

During the initial access of SIVA, the user is provided by a list of workspaces available for initialization, which currently contains only the workspace allowing visual simulation of reasoning using the completion tree version of tableau algorithm in $\mathcal{ALC}$. Prior to any actual reasoning, the user has to establish a

DL vocabulary and a knowledge base, either by typing them manually into designated input boxes or by loading existing ones, which have been previously exported using our application.

In Fig. 1 an example knowledge base about owners of potentially murder weapons (`PMW`) and a butcher owning a meat cleaver is shown. If the provided input is error-free, the user can choose one of the supported types of reasoning problems to solve and define its instance using the provided input controls.

While solving a reasoning problem, the application guides the user through the completion tree (Fig. 4) initialization phase, which, due to reduction to knowledge base consistency checking, always consists of creating nodes representing individuals from a non-empty ABox.

Afterwards, the user can freely apply tableau rules, essentially replacing the algorithm's non-determinism. Application of the $\mathcal{T}$-rule is accessed through a dedicated button present in the labels of completion tree nodes and it is performed by choosing a TBox axiom associated with the desired $\mathcal{T}$-rule (Fig. 2). All other tableau rules are applied by simply clicking on concepts in labels of nodes associated with the tableau rule. When applying the $\sqcup$-rule, the user also chooses one of the two ways he wants it to be applied (Fig. 3).

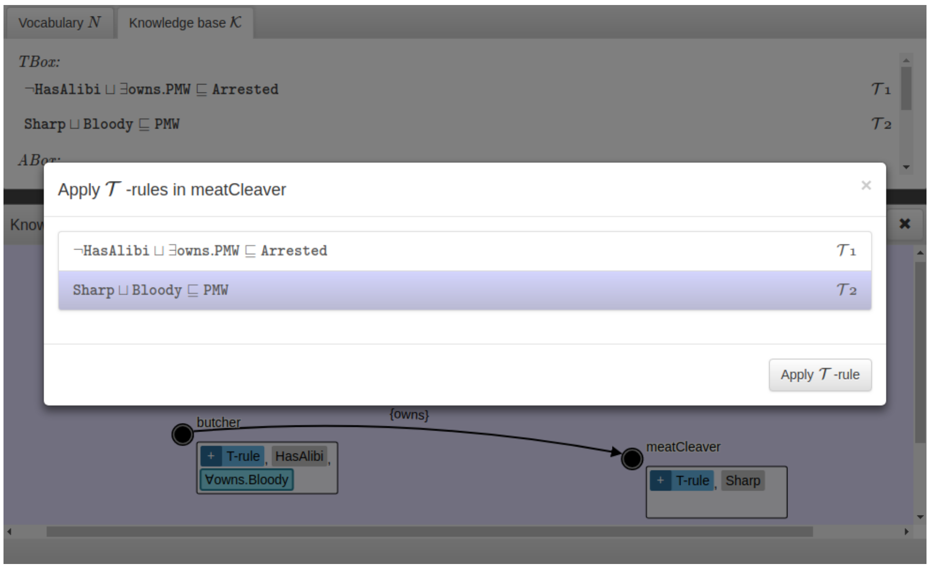
After encountering a clash or a fully expanded completion tree, the application visually informs the user and no more tableau expansion rules can be applied. The user can utilize the tableau (Fig. 5) in order to backtrack the reasoning state to any of the previously discovered ones. The tableau is visualized as a simple rooted tree, whose nodes can be freely traversed by clicking on them, which transforms the current state to the one represented by the clicked tableau node. The user can also quickly traverse the states represented by the current tableau node's parent and its children by using dedicated buttons without the need to access the tableau visualization.

Our application is accessible online at `http://siva.6f.sk/`.

6 Conclusions

We have developed SIVA – an educational tool for visual simulation of the tableau algorithm for the description logic $\mathcal{ALC}$. The web application is freely accessible. It allows the user to create a knowledge base, and to solve a decision problem by running a step-by-step simulation of the tableau algorithm. Available decision problems are consistency checking, instance checking, and concept satisfiability.

In the future, we would like to extend the application for more expressive description logics, and to allow the user to specify own tableau rules, blocking strategy, etc. We hope that this could make our tool interesting also for researchers. The application can also be extended by some standard tableau expansion strategies or optimization techniques. We would like to integrate the application with a course ware, such as Moodle [5].

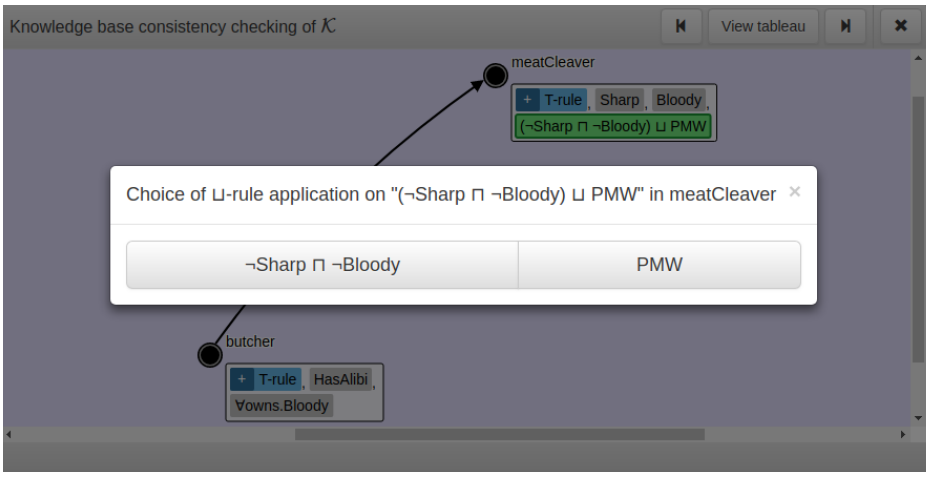


TBox:
¬HasAlibi ⊔ ∃owns.PMW ⊑ Arrested $\mathcal{T}_1$
Sharp ⊔ Bloody ⊑ PMW $\mathcal{T}_2$

Apply $\mathcal{T}$-rules in meatCleaver

¬HasAlibi ⊔ ∃owns.PMW ⊑ Arrested $\mathcal{T}_1$
Sharp ⊔ Bloody ⊑ PMW $\mathcal{T}_2$

Fig. 2. Example of the $\mathcal{T}$-rule application dialog


View tableau
meatCleaver
+ T-rule, Sharp, Bloody, (¬Sharp ⊓ ¬Bloody) ⊔ PMW
¬Sharp ⊓ ¬Bloody
PMW
butcher
+ T-rule, HasAlibi, ∀owns.Bloody

Knowledge base consistency checking of $\mathcal{K}$

Choice of ⊔-rule application on "(¬Sharp ⊓ ¬Bloody) ⊔ PMW" in meatCleaver

Fig. 3. Example of the ⊔-rule application dialog

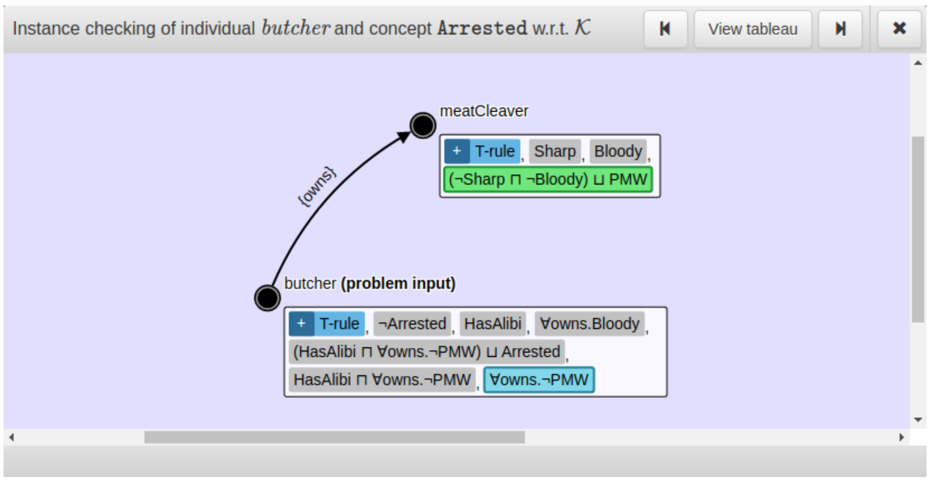

Instance checking of individual *butcher* and concept Arrested w.r.t. $\mathcal{K}$
View tableau
meatCleaver
{owns}
+ T-rule, Sharp, Bloody,
(¬Sharp ⊓ ¬Bloody) ⊔ PMW
butcher **(problem input)**
+ T-rule, ¬Arrested, HasAlibi, ∀owns.Bloody,
(HasAlibi ⊓ ∀owns.¬PMW) ⊔ Arrested,
HasAlibi ⊓ ∀owns.¬PMW, ∀owns.¬PMW

Fig. 4. Visualization of a completion tree

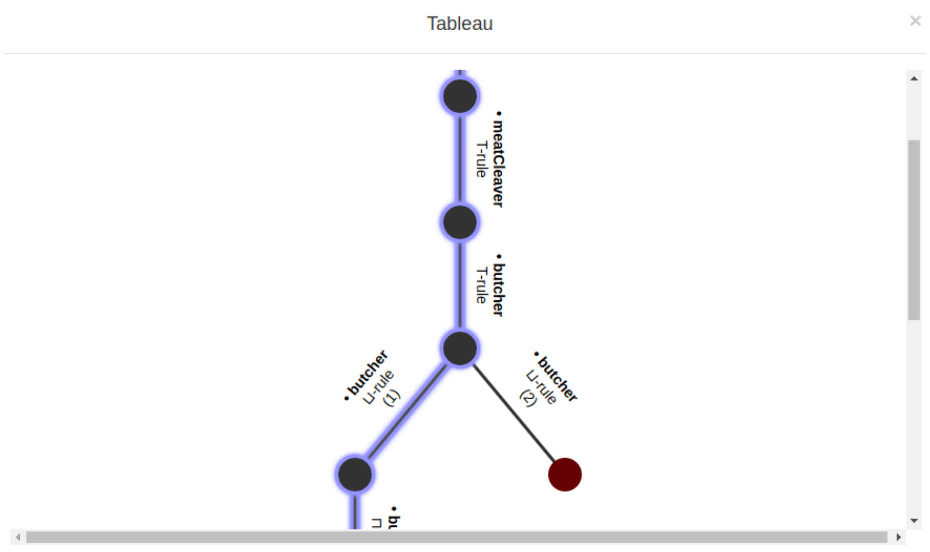


Fig. 5. Visualization of a tableau

Acknowledgements. This work presents the results of the Master's thesis by Peter Paulovics [21]. It was supported by the Slovak national project VEGA 1/0778/18. Júlia Pukancová is also supported by an extraordinary scholarship awarded by Faculty of Mathematics, Physics, and Informatics, Comenius University in Bratislava, and by the Comenius University grant no. UK/266/2018.

References

1. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
2. Beth, E.W.: The foundations of mathematics. North Holland (1959)
3. Boinski, T., Jaworska, A., Kleczkowski, R., Kunowski, P.: Ontology visualization. In: Information Technology (ICIT), 2010 2nd International Conference on. pp. 17–20. IEEE (2010)
4. Bosca, A., Bonino, D., Pellegrino, P.: Ontosphere: more than a 3d ontology visualization tool. In: SWAP 2005 – Semantic Web Applications and Perspectives, Proceedings of the 2nd Italian Semantic Web Workshop, University of Trento, Trento, Italy. CEUR-WS, vol. 166 (2005)
5. Dougiamas, M., Taylor, P.: Moodle: Using learning communities to create an open source course management system. In: EdMedia: World Conference on Educational Media and Technology. pp. 171–178 (2003)
6. Dudás, M., Hanzal, T., Svátek, V.: What can the ontology describe? Visualizing local coverage in PURO modeler. In: Proceedings of the International Workshop on Visualizations and User Interfaces for Knowledge Engineering and Linked Data Analytics, VISUAL@EKAW 2014, Linköping, Sweden. CEUR-WS, vol. 1299, pp. 28–33 (2014)
7. Eklund, P., Roberts, N., Green, S.: OntoRama: Browsing RDF ontologies using a hyperbolic-style browser. In: Cyber Worlds, 2002. Proceedings. First International Symposium on. pp. 405–411. IEEE (2002)
8. Gasquet, O., Herzig, A., Longin, D., Sahade, M.: Lotrec: Logical tableaux research engineering companion. In: Automated Reasoning with Analytic Tableaux and Related Methods, International Conference, TABLEAUX 2005, Koblenz, Germany, Proceedings. LNCS, vol. 3702, pp. 318–322. Springer (2005)
9. Gennari, J.H., Musen, M.A., Fergerson, R.W., Grosso, W.E., Crubézy, M., Eriksson, H., Noy, N.F., Tu, S.W.: The evolution of Protégé: an environment for knowledge-based systems development. International Journal of Human-computer studies 58(1), 89–123 (2003)
10. Horrocks, I., Kutz, O., Sattler, U.: The irresistible SRIQ. In: Proceedings of the OWLED*05 Workshop on OWL: Experiences and Directions, Galway, Ireland. CEUR-WS, vol. 188 (2005)
11. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$. In: Proceedings, Tenth International Conference on Principles of Knowledge Representation and Reasoning, Lake District of the United Kingdom, June 2-5, 2006. pp. 57–67. AAAI (2006)
12. Horrocks, I., Sattler, U.: A tableau decision procedure for $\mathcal{SHOIQ}$. Journal of automated reasoning 39(3), 249–276 (2007)
13. Horrocks, I., Sattler, U., Tobies, S.: Practical reasoning for expressive description logics. In: Logic Programming and Automated Reasoning, 6th International Conference, LPAR'99, Tbilisi, Georgia. LNCS, vol. 1705, pp. 161–180. Springer (1999)

14. Horrocks, I., Sattler, U., Tobies, S.: Practical reasoning for very expressive description logics. Logic Journal of IGPL 8(3), 239–263 (2000)
15. Horrocks, I.R.: Optimising tableaux decision procedures for description logics. Ph.D. thesis, University of Manchester (1997)
16. Katifori, A., Halatsis, C., Lepouras, G., Vassilakis, C., Giannopoulou, E.: Ontology visualization methods – a survey. ACM Computing Surveys 39(4), 10 (2007)
17. Krivov, S., Williams, R., Villa, F.: Growl: A tool for visualization and editing of OWL ontologies. Journal of Web Semamntics 5(2), 54–57 (2007)
18. Liebig, T., Noppens, O.: Ontotrack: A semantic approach for ontology authoring. Journal of Web Semamntics 3(2-3), 116–131 (2005)
19. Nyitraiová, A.: Educational tools for first order logic. Bachelor's Thesis, Comenius University in Bratislava (2018)
20. Nyitraiová, A.: Tableau editor. Computer software, available: `https://fmfi-uk-1-ain-412.github.io/tableauEditor/` (2018)
21. Paulovics, P.: Educational tool for tableau algorithm for DL. Master's thesis, Comenius University in Bratislava (2018)
22. Pietriga, E.: IsaViz: A visual authoring tool for RDF. Computer software, available: `http://www.w3.org/2001/11/IsaViz`
23. Plaisant, C., Grosjean, J., Bederson, B.B.: SpaceTree: Supporting exploration in large node link tree, design evolution and empirical evaluation. In: The Craft of Information Visualization, pp. 287–294. Elsevier (2003)
24. Robertson, G.G., Mackinlay, J.D., Card, S.K.: Cone Trees: Animated 3D visualizations of hierarchical information. In: Proceedings of the SIGCHI conference on Human factors in computing systems. pp. 189–194. ACM (1991)
25. Schmidt-Schauß, M., Smolka, G.: Attributive concept descriptions with complements. Artificial intelligence 48(1), 1–26 (1991)
26. Shaffer, C.A., Cooper, M.L., Alon, A.J.D., Akbar, M., Stewart, M., Ponce, S., Edwards, S.H.: Algorithm visualization: The state of the field. ACM Transactions on Computing Education 10(3), 9 (2010)
27. Smullyan, R.R.: First-order logic. Springer (1968)
28. Sure, Y., Erdmann, M., Angele, J., Staab, S., Studer, R., Wenke, D.: OntoEdit: Collaborative ontology development for the semantic web. In: The Semantic Web – ISWC 2002, First International Semantic Web Conference, Sardinia, Italy, Proceedings. LNCS, vol. 2342, pp. 221–235. Springer (2002)
29. Tree proof generator. Computer software, available: `https://www.umsu.de/logik/trees/`
30. van Valkenhoef, G., van der Vaart, E., Verbrugge, R.: OOPS: an $S5_n$ prover for educational settings. Electr. Notes Theor. Comput. Sci. 262, 249–261 (2010)
31. Wang, T.D., Parsia, B.: Cropcircles: Topology sensitive visualization of OWL class hierarchies. In: The Semantic Web – ISWC 2006, 5th International Semantic Web Conference, ISWC 2006, Athens, GA, USA, Proceedings. LNCS, vol. 4273, pp. 695–708. Springer (2006)