# Explaining Subsumption in $\mathcal{ALEHF}_{R^+}$ TBoxes

Thorsten Liebig and Michael Halfmann
University of Ulm, D-89069 Ulm, Germany
liebig@informatik.uni-ulm.de
michael.halfmann@informatik.uni-ulm.de

**Abstract**

This paper presents current work on generating textual explanations for subsumption within an expressive fraction of OWL Lite TBoxes. Our approach is based on a tableau-style algorithm. We describe how to explain subsumptions within $\mathcal{ALE}$ extended by role hierarchies, transitivity, cardinality restrictions, and domain as well as range restrictions. We also illustrate some optimization features, comment on our implementation, and discuss future extensions concerning more expressive languages.

## 1 Motivation

Authoring ontologies is a difficult conceptual task. An Ontology is an explicit and formal specification which requires a basic understanding of the underlying semantics and its reasoning services. Especially novices are faced with comprehension problems, concerning the proper use of role hierarchies, the influence of transitive roles on reasoning or the effect of domain and range restrictions, etc.

A better understanding can be fundamentally supported by on-demand explanations of subsumption, a core inference service. Embedded in an interactive editing environment explanations can improve the authoring process to a large degree. Another important issue is the acceptance of reasoning results, which is closely related to trust in its inference services. Belief in a system can be increased by evidence of how and why it was derived.

In this paper, we describe ongoing work towards a system for explaining subsumption for a significant fraction of the DL underlying OWL Lite. Our approach generates quasi-natural language from text templates, which are distilled from a corresponding tableau proof. First we describe how to use a tableau algorithm for subsumption explanation by extending the work of [2]. We then present some optimization techniques, describe our current implementation and further improvements. We conclude with a discussion of related work and an outlook.

# 2 Tableau-Based Explaining

Our approach is based on the ideas of extending a Description Logic tableau prover for explaining purpose as suggested in [2]. Our current implementation covers definitorial $\mathcal{ALEHF}_{R^+}$ TBoxes with global domain and range restrictions. We restrict TBox axioms to be unique. I.e. all axioms of a TBox $\mathcal{T}$ are of the form $A \sqsubseteq D$ or $A \equiv D$, where $A$ is atomic and has no other definition within $\mathcal{T}$.

As commonly known, tableau systems implement a refutation strategy. E. g. in order to prove the subsumption $A \sqsubseteq D$ a refutation prover will show the unsatisfiability of $A \sqcap \neg D$ which is the negation normal form of the complementary query $\neg(A \sqsubseteq D)$. Apparently, this kind of argumentation is not intuitive or natural in the human way of thinking. However, this is inconsequential, as long as the user is only interested in the answer of the TBox query itself.

## 2.1 Tagging

When using a tableau prover for the task of explaining *how* a query answer actually has been derived, the refutation strategy explicitly has to be hidden to the end-user. Instead, the original (positive) query should be used for illustration of the derivation steps. In order to achieve this, a technique called tagging has been introduced in [2]. For a subsumption query $A \sqsubseteq D$ the right hand side is tagged in the corresponding refutation problem (namely $A \sqcap \neg D^{\dagger}$). To generate an explanation, all tagged expressions have to be negated again in order to comply to the original query. E. g. the tagged expression $A \sqcap \bot^{\dagger}$ corresponds to the TBox query $A \sqsubseteq \top$ which can be explained by the statement: "*Everything is subsumed by the most general concept*". Obviously a tableau-based explainer has to carefully distinguish between the rhs and lhs of a given subsumption problem in order to be able to reconstruct the original query at any time. It follows, that many optimizing simplification strategies are no longer applicable.

## 2.2 Lazy Unfolding

Generally, concepts are defined with help of references to other concept definitions of the TBox. In order to successfully build a tableau proof those references require to unfold to their given definition. A well known optimization technique of DL tableau algorithms is lazy unfolding, which delays the unfolding of a concept until it is actually required. In order to keep the user informed, each unfolding step has to be explained. Therefore, all necessary unfolding steps within a tableau node are collected and explained before further processing. E. g. a query $C \sqsubseteq D$ could expand to an expression $\exists r.A \sqsubseteq \exists r.B$, which would lead to an explanation statement "*We have to check whether* $C \sqsubseteq D$ *which unfolds to* $\exists r.A \sqsubseteq \exists r.B$".

## 2.3 $\mathcal{ALE}$

A methodology for explaining the subsumption relationship between $\mathcal{ALC}$ concepts based on a sequent proof, derived from a tableau-based algorithm, can be found in [2]. This approach proposes some straight-forward extensions to a naive tableau algorithm and build the basic procedure of our explanation generator. Since our aim is to cover a sensible fraction of OWL Lite, namly OWL Lite$^-$ [4], we build on top of the slightly less expressive language $\mathcal{ALE}$. Note, that in $\mathcal{ALE}$ disjunction comes implicitly on rhs due to the refutation strategy of the tableau approach. In this instance, each disjunctive element leads to an alternative tableau expansion. On the explanation level a disjunction on rhs corresponds to a conjunction and will therefore be explained with help of an enumeration of sub-explanations.

In a successfully closed tableau a clash occurs in some branch of the tableau tree (resp. in every alternative expansion in case of disjunction). The type of the clash is important for explanation generation. Due to a lack of space we refer to [2] or [1] for the possible set of qualitatively different clashes within $\mathcal{ALC}$ and discuss those types of clashes which extend $\mathcal{ALE}$ with role hierarchies, transitive and functional roles as well as domain and range restrictions ($\mathcal{ALEHF}_{R^+}$) in the following.

## 2.4 Cardinality Restrictions and Merging

Following the language expressivity of OWL Lite we support functional roles and a limited form of cardinality restrictions within our explainer. Particulary, we cover cardinality restrictions $(\geq n\ r)$ and $(\leq n\ r)$ with $n \in \{0, 1\}$ as well as global constraints restricting a role to have at most one filler. In fact, the "at least" restrictions are already covered by $\mathcal{ALE}$, because $(\geq 0\ r)$ reduces to $\top$ and $(\geq 1\ r)$ is equivalent to $\exists r.\top$. In contrast, the "at most" restrictions and the combination of conflicting cardinality restrictions require an extension of the tableau as well as the explanation process. Since we have to take the origin of the restrictions (lhs vs. rhs) into account we have to distinguish between four types of cardinality clashes. Each of which result in a different explanation statement:

- $(\leq n\ r) \sqcap (\geq m\ r) \sqsubseteq \ \ldots \quad$ (with $m > n$)
  "*There can't be at-least m and at-most n fillers for role r. The subsumee is equivalent to $\bot$ which is subsumed by everything.*"

- $\ldots \sqsubseteq (\leq n\ r) \sqcup (\geq m\ r) \quad$ (with $m \leq n + 1$)
  "*There are always either less than n or more than m fillers for role r. The subsumer therefore is equivalent to $\top$ which subsumes everything.*"[1]

[1] This kind of clash/explanation can only occur in the $\mathcal{ALC}$ language family.

- $(\leq n\ r) \sqsubseteq (\leq m\ r)$ (with $m \geq n$)
  "*At most n fillers for r is subsumed by at most m fillers for r.*"

- $(\geq m\ r) \sqsubseteq (\geq n\ r)$ (with $m \geq n$)
  "*At least m fillers for r is subsumed by at least n fillers for r.*"[2]

Another extension of the reasoning procedure is concerned with the combination of existential quantifications and at most restrictions. Moreover, a $(\leq 1\ r)$ restriction forces a role $r$ to have at most one $r$-successor. In case of additional existential quantification this requires to merge all existing successor nodes to one single node. Since we do not allow for full negation such a merge can only locally occur on lhs. Consider the query $\exists r.A \sqcap \exists r.B \sqcap (\leq 1\ r) \sqsubseteq \exists r.(A \sqcap B)$. As a result of the at most restriction the $r$-successor node will be explained as follows: "*Since there has to be at least one filler for each of the types A and B and at most one filler for r on lhs this filler has to be of type* $(A \sqcap B)$".

An analogous explanation is needed for functional roles even without an explicit at most restriction. E. g. $\exists r.A \sqcap \exists r.B \sqsubseteq \exists r.(A \sqcap B)$ holds in case of a functional role $r$. The corresponding explanation would state: "*Since r is functional there is at most one filler which has to be of type* $(A \sqcap B)$".

## 2.5 Role Hierarchies

Within role hierarchies each role filler also is a filler of all super roles. The effect of role restrictions in a tableau proof is the reverse — from a role to its sub-roles. E. g. quantitative restrictions of a role like domain and range as well as filler types also apply to all sub-roles. Consider a sub-role $p$ of $r$ ($p \sqsubseteq r$) and an expression $(\geq 1\ p) \sqcap \forall r.A$. Each $p$-successor node will get an additional explanation stating that "*All fillers of r are restricted to be of type A. Since p is a sub-role of r this restriction also applies to p.*"

Cardinality restrictions and merging require a likewise handling. As with "direct" cardinality restrictions we have to take the side of each expression into account and therefore need to distinguish between four different types of clashes. These are analogous to the ones described in section 2.4 but additionally explain the corresponding sub-role relationship.

## 2.6 Domain and Range Restrictions

Global domain and range restrictions of a role become proof relevant as soon as a tableau rule generates a successor node for that role. The given range or domain restriction will then be added to the successor resp. predecessor node. Such a restriction may consist of an arbitrary expression of the underlying language. As

[2]Within OWL Lite ($n = 0$) this case does not occur (rhs will reduce to $\top$ beforehand).

a consequence, in case of a resulting clash due to a domain or range restriction one of the previously mentioned explanations will apply. In addition, the source of the domain or range expression has to be explained. Accordingly, we need additional explanation steps for domain and range restrictions. Hence, whenever a role $r$ has a domain or range restriction an additional explanation statement will be given just after a generating $r$-successor rule. As an example, consider the following query $\exists r.A \sqsubseteq \exists r.B$ and a range restriction $B$ on role $r$. The two existential quantifications are then explained as given in section 2.3: "*For role $r$ we have to check whether $A \sqsubseteq B$*". Because of the relevant range restriction an additional explanation step states "*Since role $r$ has a range restriction on $B$ the subsumption evolves to $(A \sqcap B) \sqsubseteq B$*".

### 2.7 Transitive Roles

Restrictions on transitive roles ($\forall r.C$ with $r$ transitive) have to be propagated to all descendants referenced over a sequence of $r$-successors. In the tableau this is achieved by adding the qualifier $C$ as well as the whole construct $\forall r.C$ to all direct successors of the node containing the restriction $\forall r.C$. In order to supply inverse roles in a future version a pair-wise blocking strategy is used to ensure termination. This causes the following sub-proofs to change in an unobvious way. Therefore we add an additional explanation statement: "*Since $r$ is a transitive role $\forall r.C$ also has to hold for all its successors*".

## 3 Optimizations

The preceeding section introduced the basic techniques for explaining expressive TBox queries. As with many approaches, a naive algorithm often leads to sub-optimal results. In the case of explaining TBox queries, the full amount of axioms needed and tableau rules applied in order to complete the proof can lead to confusingly complex explanations. Therefore the major goal of optimizations is to cut down the explanation length wherever possible.

### 3.1 Switching Explanation Mode

Special cases within subsumption determination are those where at some point of processing either the subsumee is equivalent to $\bot$ or the subsumer is equivalent to $\top$ independently from each other. On tableau side this turns out as a clash caused by equivalence to $\bot$ on either side. Explanation statements for those cases finally state "*Everything is subsumed by the most general concept*" for the lhs or "*The most specific concept is subsumed by everything*" for the rhs.

However, it requires two independent unsatisfiability tests (for rhs and lhs) to detect unsatisfiability on either side. Consider the subsumption $\forall r.(\exists s.((\leq$

$0\ t) \sqcap (\geq 1\ t))) \sqsubseteq \forall r.C$. The latter holds because of the unsatisfiability of the subsumees s-successor node on lhs. As a general consequence, instead of explaining subsumption an optimized approach will switch to unsatisfiability explaining as soon as the rhs or lhs of a node becomes independently unsatisfiable.

The previous example would then be explained as follows:
"*For role r the left hand side* $(\exists s.((\leq 0\ t) \sqcap (\geq 1\ t)))$ *is equivalent to* $\bot$ *which is subsumed by everything.*"
"*For role s the left hand side* $((\leq 0\ t) \sqcap (\geq 1\ t))$ *is equivalent to* $\bot$."
"*There can't be* $(\geq 1)$ *and* $(\leq 0)$ *fillers for role t so this is equivalent to* $\bot$."

### 3.2 Filtering of Disjunction on rhs

There are specific query types which are typically considered as obviously true by end users. We propose a structural approach for these cases which avoids additional tableau branches by pruning out disjunction on rhs. Consider the query $A \sqcap (\leq 1\ r) \sqcap C \sqsubseteq A \sqcap (\leq 1\ r)$. Quite obviously this subsumption holds, since the subsumee is a direct specialization of the subsumer. On tableau side the subsumer is represented as $\neg A \sqcup (\geq 2\ r)$. A naive approach would split up the proof by providing an explanation for each part of the disjunction. On explanation side this case analysis is very likely unnecessary because of the intuitive and commonly agreed relationship between conjunction and specialization. Therefore, an optimized explainer could simply provide an explanation statement like "*Obviously* $A \sqcap (\leq 1\ r) \sqcap C$ *is subsumed by* $A \sqcap (\leq 1\ r)$ *since the former is a simple specialization of the latter*". To provide such explanations, the corresponding situations have to be recognized and filtered. This can be achieved by a structural comparison of subsumer and subsumee.

## 4 Implementation and Future Work

### 4.1 Prototype

Our prototype called MEX is implemented in Lisp and capable of explaining subsumption within $\mathcal{ALEHF}_{R^+}$ TBoxes. The syntax follows the KRSS syntax.

An explanation provided by MEX is a list of explanation steps. These steps are created "on-the-fly" during proof generation. For each relevant tableau step one or more explanation steps are added. When split into multiple branches each branch consists of an independent sub-subsumption explanation. A tableau clash causes an explanation branch to terminate with a type-specific explanation step. In order to keep track of the structure of the explanation branches the tableau tree depth is stored with each step. We have also implemented a Java-based visualization component which displays an explanation as an expandable tree list. As explaining is most powerful when used in combination with a tool

for editing ontologies. Therefore we have integrated MEX and its visualization component as a plugin into ONTOTRACK [4].

Although we haven't done extensive testing concerning MEXs performance, it is quite obvious, that it is far behind the performance of highly optimized reasoners like RACER or Pellet. On one hand this is because additional operations for explanation generation are necessary. On the other hand this is due to the fact that none of the common tableau optimization techniques (except *lazy-unfolding*) are used. Those techniques have not been applied because they typically change the structure of the involved terms and ignore the seperation of lhs and rhs, which is harmful in terms of providing explanations.

In order to allow for switching from subsumption to unsatisfiability explaining, unsatisfiability on rhs or lhs needs to be identified before clash detection. Therefore, both sides of a node are checked for unsatisfiability with help of the RACER reasoner before further expansion rules are applied. Optionally, the highly optimized RACER system can be used to further prune the MEX tableau tree to those clauses which actually contribute to the clash, by making use of the dual-reasoner strategy proposed in [3].

To enable advanced optimizations in order to increase the explanation quality we suggest to generate explanations after building up the complete tableau in future implementations. Additionally we aim to develop a more sophisticated data structure in order to store additional information (e. g. origin and history) for each construct in the tableau-nodes.

### 4.2 Further Optimization: Hiding

Constructs like domain and range restrictions or propagations of $\forall$-restrictions on transitive roles increase the amount of clauses in a tableau. Occasionally, they do not account for the final clash. When adding general concept inclusions (GCIs) this becomes a serious problem, because lazy unfolding is not applicable here.[3] In order to reduce an explanation to its essential components we propose to hide all those (otherwise mechanically added) constructs that are irrelevant for the terminating clash. In a second step even all those irrelevant constructs could be hidden which explicitly occur in the given definitions. However, to hide individual constructs their clash relevance has to be detected prior clash occurrence. A likewise concept called *Relevant* has been introduced in [2].

## 5 Related Work and Outlook

Work on explaining TBox inference services for expressive DLs has just started mainly because of publishing of OWL to a larger audience. An early work

[3] Potential optimizations like absorption would in turn require other explanation steps.

deals with structural subsumption algorithms in order to provide explanations as proof fragments within the Classic system [5]. An approach for explaining subsumption by computing an interpolation of an intermediate concept in-between subsumer and subsumee for $\mathcal{ALC}$ is given in [6]. The most related work seems to be the dual-reasoner approach [3] which is also based on [2].

Our approach covers a fairly expressive DL. In order to deal with $\mathcal{SHF}$ (without GCIs) we only need to add full negation and disjunction, which we already have to cope within our tableau algorithm. However, full negation requires to explain the transformation steps into negation normal form.

Explaining subsumption within OWL Lite ontologies (which are virtually $\mathcal{SHIF}(\mathbf{D})$ TBoxes) requires support for GCIs (easy in combination with hiding but may significantly decrease performance), concrete domains, inverse roles (which may require to jump back and forth due to dynamic blocking), and multiple definitions (for which we currently have no idea how to explain). It is even more challenging to deal with OWL DL ontologies ($\mathcal{SHOIN}(\mathbf{D})$ TBoxes). Explaining TBoxes with nominals is costly because it adds ABox reasoning and therefore requires ABox explaining. In addition, unrestricted cardinality constraints may blow up the explanation because of a potential huge set of cardinality enforced combinatorial changes.

# References


[1] A. Borgida, E. Franconi, and I. Horrocks. Explaining $\mathcal{ALC}$ subsumption. In *Proc. of the 14th European Conf. on Artificial Intelligence (ECAI 2000)*, pages 209–213, 2000.

[2] A. Borgida, E. Franconi, I. Horrocks, D. McGuinness, and P. F. Patel-Schneider. Explaining $\mathcal{ALC}$ subsumption. In *Proc. of the Int. Workshop on Description Logics (DL99)*, pages 37–40, 1999.

[3] F. Kwong. Explaining Description Logic Reasoning. In *Proc. of the Int. Workshop on Description Logics (DL04)*, Whistler, BC, Canada, 2004.

[4] T. Liebig and O. Noppens. OntoTrack: Combining Browsing and Editing with Reasoning and Explaining for OWL Lite Ontologies. In *Proc. of the International Semantic Web Conf. (ISWC 2004)*, pages 244–258, Hiroshima, Japan, 2004.

[5] D. McGuinness. *Explaining Reasoning in Description Logics*. PhD thesis, Rutgers University, 1996.

[6] S. Schlobach. Explaining Subsumption by Optimal Interpolation. In *Proc. of the European Conf. of Logics in Artificial Intelligence*, pages 413–425, Lisbon, Portugal, 2004.