15 Years of Consequence-Based Reasoning

David Tena Cucala, Bernardo Cuenca Grau, and Ian Horrocks

University of Oxford, Oxford OX1 3QD, UK
david.tena.cucala@cs.ox.ac.uk
bernardo.cuenca.grau@cs.ox.ac.uk
ian.horrocks@cs.ox.ac.uk

Abstract. Description logics (DLs) are a family of formal languages for knowledge representation with numerous applications. Consequence-based reasoning is a promising approach to DL reasoning which can be traced back to the work of Franz Baader and his group on efficient subsumption algorithms for the $\mathcal{EL}$ family of DLs circa 2004. Consequence-based reasoning combines ideas from hypertableaux and resolution in a way that has proved very effective in practice, and it still remains an active field of research. In this paper, we review the evolution of the field in the last 15 years and discuss the various consequence-based calculi that have been developed for different DLs, from the lightweight $\mathcal{EL}$ to the expressive $\mathcal{SROIQ}$. We thus provide a comprehensive and up-to-date analysis that highlights the common characteristics of these calculi and discusses their implementation.

Keywords: description logics · automated reasoning · ontologies · knowledge representation.

1 Introduction

Description logics (DLs) are a prominent family of languages for knowledge representation and reasoning with well-understood formal properties [3]. Interest in DLs has been spurred by their applications to the representation of ontologies: for instance, the DL $\mathcal{SROIQ}$ provides the formal underpinning for the Web Ontology Language OWL 2 [46].

A central component of most DL applications is a scalable reasoner, which can be used to discover logical inconsistencies, classify the concepts of an ontology in a subsumption hierarchy, or answer database-style queries over an ontology and a dataset. Two traditional approaches to concept classification (and to DL reasoning more broadly) are tableaux [4] and resolution [9].

Tableau and hyper-tableau calculi underpin many of the existing DL reasoners [40, 44, 18, 17, 41]. To check whether a concept subsumption relationship holds, (hyper-)tableau calculi attempt to construct a finite representation of an ontology model disproving the given subsumption. The constructed models can, however, be large—a source of performance issues; this problem is exacerbated in classification tasks due to the large number of subsumptions to be tested.

Another major category of DL reasoning calculi comprises methods based on first-order logic resolution [9]. A common approach to ensure both termination and worst-case optimal running time is to parametrise resolution to ensure that the calculus only derives a bounded number of clauses [34, 22, 37, 21, 15, 28]. This technique has been implemented, for instance, in the KAON2 reasoner [32] for $\mathcal{SHIQ}$. Resolution can also be used to simulate model-building (hyper)tableau techniques [20], including blocking methods which ensure termination [16].

Consequence-based (CB) calculi emerged as a promising approach to DL reasoning combining features of (hyper)tableau and resolution [2, 24, 26, 10]. On the one hand, similarly to resolution, they derive formulae entailed by the ontology (thus avoiding the explicit construction of large models), and they are typically worst-case optimal. On the other hand, clauses are organised into *contexts* arranged as a graph structure reminiscent of that used for model construction in (hyper)tableau; this prevents CB calculi from drawing many unnecessary inferences and yields a nice goal-oriented behaviour. Furthermore, in contrast to both resolution and (hyper)tableau, CB calculi can verify a large number of subsumptions in a single execution, allowing for one-pass classification. Finally, CB calculi are very effective in practice, and systems based on them have shown outstanding performance. Leading reasoners for lightweight DLs such as ELK [27] or Snorocket [31] are based on consequence-based calculi. Furthermore, prototypical implementations of consequence-based calculi for more expressive languages, such as Sequoia [11] or Avalanche [45], show promising results.

The first CB calculi were proposed by Franz Baader, Sebastian Brandt, and Carsten Lutz for the $\mathcal{EL}$ family of DLs [12, 2]. They were later extended to more expressive logics like Horn-$\mathcal{SHIQ}$ [24], Horn-$\mathcal{SROIQ}$ [35], and $\mathcal{ALCH}$ [38]. A unifying framework for CB reasoning was developed in [39] for $\mathcal{ALCHI}$, introducing the notion of *contexts* as a mechanism for constraining resolution inferences and making them goal-directed. The framework has been extended to the DLs $\mathcal{ALCHIQ}$, which supports number restrictions and inverse roles [10]; $\mathcal{ALCHOI}$, which supports inverse roles and nominals [42]; $\mathcal{ALCHOQ}$, supporting nominals and number restrictions [23], and finally to $\mathcal{ALCHOIQ}$, which supports all of the aforementioned constructs [43].

This paper reviews the development of the consequence-based approach to DL reasoning since its first appearance fifteen years ago. In Section 2 we introduce the core ideas behind consequence-based reasoning, using a simplified version of the original CB calculus in [12]. In Section 3 we discuss the evolution of consequence-based calculi in the first decade after the introduction of the original calculus, which focused mostly in lightweight or Horn DLs. In Section 4 we discuss the introduction in [39] of a unifying framework for consequence-based reasoning. This piece of work describes an abstract structure which more explicitly captures the defining features of consequence-based calculi. By varying the parameters in this structure, it becomes possible to simulate many previously existing calculi. Finally, in Section 5 we discuss recent progress in consequence-based reasoning, including the design of calculi for more expressive DLs, and the introduction of hybrid methods that combine the consequence-based approach

with other well-known reasoning techniques such as tableaux or integer linear programming.

We assume familiarity with the basics of Description Logics, and refer the reader to [6] for a comprehensive introduction to DLs.

2 Consequence-based Reasoning

This section introduces the consequence-based approach to DL reasoning. We start by describing one of the simplest and best known consequence-based calculi: the classification algorithm for $\mathcal{EL}$. We next give a summary of the common features of consequence-based calculi, and use the $\mathcal{EL}$ calculus as an illustrative example of these characteristics.

2.1 The $\mathcal{EL}$ Consequence-based Calculus

The calculus presented in this section is a restricted form of the classification procedure given in [12]. The original presentation used a notation different from the terminology used in this section, which is due to [24] and has been used in many of the subsequent consequence-based calculi, as well as in a recent textbook on Description Logics [6].

Consider an arbitrary $\mathcal{EL}$ ontology $\mathcal{O}$ in normal form, which is defined as a set of axioms of the form $A \sqsubseteq B$, $A_1 \sqcap A_2 \sqsubseteq B$, $A \sqsubseteq \exists R.B$, or $\exists R.A \sqsubseteq B$, with A, B atomic concepts, and R an atomic role. It is well-known that any $\mathcal{EL}$ ontology can be normalised to this form in polynomial time. The calculus builds a set $\mathcal{S}$ containing inclusions entailed by $\mathcal{O}$, which are called *consequences* of $\mathcal{O}$. Set $\mathcal{S}$ is initialised by using rules IR1 and IR2 from Figure 1. Next, the algorithm repeatedly applies rules CR1-CR4 to saturate $\mathcal{S}$. These rules use the existing consequences in $\mathcal{S}$ and the axioms of $\mathcal{O}$ to derive further consequences; e.g., rule CR1 uses consequence $A \sqsubseteq B$ and the fact that $B \sqsubseteq C$ is an axiom of $\mathcal{O}$ to conclude that $A \sqsubseteq C$ is also a consequence.

$$\mathsf{IR1}\ \frac{}{A \sqsubseteq A} \qquad \mathsf{IR2}\ \frac{}{A \sqsubseteq \top} \qquad \mathsf{CR1}\ \frac{A \sqsubseteq B \qquad B \sqsubseteq C \in \mathcal{O}}{A \sqsubseteq C}$$

$$\mathsf{CR2}\ \frac{A \sqsubseteq B \qquad A \sqsubseteq C \qquad B \sqcap C \sqsubseteq D \in \mathcal{O}}{A \sqsubseteq D}$$

$$\mathsf{CR3}\ \frac{A \sqsubseteq B \qquad B \sqsubseteq \exists R.C \in \mathcal{O}}{A \sqsubseteq \exists R.C} \qquad \mathsf{CR4}\ \frac{A \sqsubseteq \exists R.B \qquad B \sqsubseteq C \qquad \exists R.C \sqsubseteq D \in \mathcal{O}}{A \sqsubseteq D}$$

Fig. 1. Inference rules for the simplified $\mathcal{EL}$ calculus

The calculus in Figure 1 ensures that any subsumption of the form $A \sqsubseteq B$ that is logically entailed by the ontology $\mathcal{O}$ will be contained in $\mathcal{S}$ after saturation;

hence, the classification of $\mathcal{O}$ can be read directly from $\mathcal{S}$. The resulting algorithm works in worst-case polynomial time in the size of the input ontology $\mathcal{O}$.

2.2 The Defining Characteristics of Consequence-Based Reasoning

Although consequence-based calculi described in the literature may appear rather different from each other at first glance, they all share several defining characteristics. We discuss these common features using the $\mathcal{EL}$ calculus as an example.

Materialisation of derived consequences. Similarly to resolution-based approaches, consequence-based calculi proceed by deriving formulae entailed by the input ontology. This approach can have significant practical advantages over (hyper-)tableau calculi, since these construct (a representation of) a model of the input ontology, and such (representations of) models may be very large. We illustrate this with an example: suppose we would like to check whether the subsumption $B_0 \sqsubseteq C$ is entailed by the following $\mathcal{EL}$ ontology:

$$\{B_i \sqsubseteq \exists R.B_{i+1} \mid 0 \leq i \leq n-1\} \qquad B_n \sqsubseteq C \qquad \exists R.C \sqsubseteq C$$
$$\{B_i \sqsubseteq \exists S.B_{i+1} \mid 0 \leq i \leq n-1\} \qquad\qquad \exists S.C \sqsubseteq C$$

If we use a tableau algorithm for this task, the size of the generated model will depend on the order in which inference rules are applied. In particular, building the tableau in a breadth-first manner will lead to an exponentially large model.

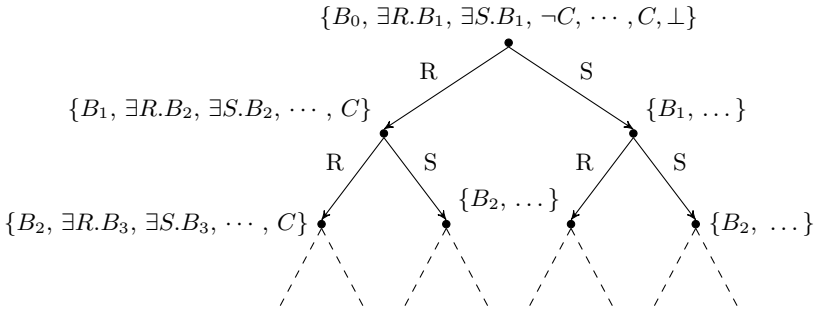

Fig. 2. Model built by a tableau-like procedure. All node labels are derived from the root downwards, except for C, which is derived first at the leaves and then propagated upwards, and $\bot$.

In contrast, the subsumption can be proved using the algorithm of Section 2.1 in a linear number of steps. Indeed, initialisation would produce $B_i \sqsubseteq \top$ and $B_i \sqsubseteq B_i$ for each $0 \leq i \leq n$, together with $C \sqsubseteq \top$ and $C \sqsubseteq C$. Rule CR1 would then produce $B_n \sqsubseteq C$; afterwards, rule CR3 would produce at least one of $B_i \sqsubseteq \exists R.B_{i+1}$ and $B_i \sqsubseteq \exists S.B_{i+1}$ for each $0 \leq i \leq n$. Finally, CR4 would generate all inferences of the form $B_i \sqsubseteq C$, including our target subsumption.

Locality of consequences. Another key characteristic of consequence-based calculi is their emphasis on computing "local" consequences only. Unlike methods based on resolution, where many other kinds of entailments can be derived, CB calculi derive only clauses of a very restricted form which constrain only a "central" domain element t and (possibly) domain elements that are directly connected to t via roles (binary predicates). This can be easily seen in the $\mathcal{EL}$ calculus above, where consequences have the same form as ontology axioms, and these are equivalent to first-order clauses universally quantified over a single variable x, which may also introduce an existentially quantified variable y in clauses with a role atom of the form $R(x, y)$ acting as a guard.

Focus on contexts. The search for local consequences in consequence-based calculi is driven by *contexts* [39], which group similar types of local consequences together. For instance, in the $\mathcal{EL}$ calculus, inference rules always use a premise of the form $A \sqsubseteq \Delta$, where A is an atomic concept, and Δ is a concept. Hence, we can define one context per atomic concept A and group together in the same context all consequences having A in the left-hand-side. This is an advantage with respect to resolution-based methods, for it prevents the derivation of irrelevant clauses. For example, a DL-clause $B_1 \sqcap B_2 \sqsubseteq B_3$, where B_1, B_2, B_3 are atomic concepts, is only used in the $\mathcal{EL}$ calculus if we have derived consequences $A \sqsubseteq B_1$ and $A \sqsubseteq B_2$. In contrast, a resolution method could resolve the same axiom with consequences of the form $A_1 \sqsubseteq B_1$ and $A_2 \sqsubseteq B_2$ even if A_1 and A_2 never appear together in the same context, which suggests that the derived consequence would not be relevant for the taxonomy.

Other Features. The main characteristics described above enable other features that are often regarded as distinctive traits of consequence-based reasoning.

- *Goal-oriented behaviour.* It is often possible to focus only on contexts that are relevant for the given query, as well as contexts that are found to be relevant during the saturation phase. For instance, if we use the $\mathcal{EL}$ calculus to check a single subsumption of the form $A \sqsubseteq B$, we can initialise the algorithm simply with $A \sqsubseteq A$ and $A \sqsubseteq \top$, and then restrict the application of rules IR1 and IR2 only to those atomic concepts that are generated in $\mathcal{S}$ during saturation.
- *Reusability of consequences and one-pass classification.* Since consequences derived while answering a query are entailments of the input ontology, they can be re-used in reasoning for any further queries. For instance, in the previous example, any consequences generated while checking whether $B_i \sqsubseteq C$ for some $0 < i < n$ can be reused to determine whether $B_0 \sqsubseteq C$. Therefore, if we initialise the calculus by applying rules IR1 and IR2 to every atomic concept, all subsumptions that follow from an $\mathcal{EL}$ ontology are computed in a single run of the algorithm. Consequences can also be reused when the input ontology $\mathcal{O}$ is extended with new axioms.

– *Parallelisation.* Emphasis on locality makes consequence-based calculi very amenable to parallel implementations, since many inferences in each context can be done independently of other contexts. Furthermore, if one can predict which contexts will interact little or not at all with each other, the workload can be better divided in order to increase the progress that can be made in parallel.

3 The Early Years: 2004 - 2013

This section gives a historical perspective of the first 10 years of consequence-based reasoning, which mostly focused on lightweight or Horn DLs.

3.1 The First Consequence-based Calculus

The first consequence-based calculi were introduced by Franz Baader and colleagues around the year 2004. A first version was presented in [12] for the DL $\mathcal{ELH}$, and this was extended shortly afterwards to handle additional constructs such as the constant for unsatisfiable concepts ($\bot$), concrete domains, and role chains [2]. The $\mathcal{EL}$ family of DLs was starting to receive significant attention at the time: on the one hand, $\mathcal{EL}$ reasoning had been proven to be tractable; on the other hand, the logics in the $\mathcal{EL}$ family had been shown to be expressive enough to capture real-world ontologies. The logic $\mathcal{EL}$++ eventually became the basis for one of the standardised profiles [33] of the ontology language OWL 2.

The first analysis of fragments of $\mathcal{EL}$ used structural subsumption algorithms [7, 5], but the new technique introduced in [12] and [2] was radically different. This technique starts by normalising the ontology, following a procedure analogous to that devised in [30], and then it applies a series of inference rules until saturation is reached. The calculus in Section 2.1 represents an example of this approach. Although these calculi were not referred to as "consequence-based" at the time, they already displayed the defining features that have been discussed in Section 2.2.

The calculus in [2] was implemented in the reasoner CEL [8]; experiments with this system on life-science ontologies showed that efficient reasoning was feasible even for very large ontologies. The elegance and simplicity of this technique was quickly recognised, and it inspired similar methods for more expressive DLs, which we discuss in the following sections.

At the same time, interest was sparked into developing more efficient implementations of consequence-based calculi for the $\mathcal{EL}$ family of DLs. Research in this area has covered topics such as alternative consequence-based calculi for $\mathcal{EL}$ and its variants, efficient strategies for saturation based on tailored data structures and indices, or incremental reasoning. Reasoners such as ELK [27] and Snorocket [31] draw upon this work in order to provide highly efficient, robust, and scalable implementations for lightweight extensions of $\mathcal{EL}$.

3.2 Going Beyond $\mathcal{EL}$: Horn-$\mathcal{SHIQ}$

In 2009, a new consequence-based calculus was introduced in [24] for the DL Horn-$\mathcal{SHIQ}$, which includes expressive constructs not supported in any variant of $\mathcal{EL}$, such as universal restrictions ($\forall R.A$), inverse roles, and number restrictions. The addition of these constructs led to new challenges for the development of consequence-based calculi. In particular, it becomes necessary to allow for the derivation of additional types of consequences in order to ensure completeness. For instance, a consequence like $A \sqsubseteq \exists R.B$ combined with an axiom like $A \sqsubseteq \forall R.C$ entails consequence $A \sqsubseteq \exists R.(B \sqcap C)$, which then becomes relevant for deriving additional atomic subsumptions. The calculus in [24] allows for consequences of the forms $\bigsqcap_{i=1}^{n} A_i \sqsubseteq B$ and $\bigsqcap_{i=1}^{n} A_i \sqsubseteq \exists R.(\bigsqcap_{j=1}^{m} B_j)$, where A_i, B_j are atomic concepts, and R is an atomic role. These inferences may lead to conjunctions that can be as long as the number of atomic concepts in the ontology. This syntax is also useful to represent consequences of axioms involving number restrictions, as seen in the following inference rule from the calculus in [24]:

$$\mathsf{R5}\ \frac{\begin{array}{llll} M \sqsubseteq \exists R_1.N_1 & N_1 \sqsubseteq B & R_1 \sqsubseteq S \in \mathcal{O} & \\ M \sqsubseteq \exists R_2.N_2 & N_2 \sqsubseteq B & R_2 \sqsubseteq S \in \mathcal{O} & M \sqsubseteq \leq 1S.B \end{array}}{M \sqsubseteq \exists R_1.(N_1 \sqcap N_2)}$$

In this inference rule, M, N_1, and N_2 are conjunctions of atomic concepts, B is an atomic concept, and R_1,R_2, and S are atomic roles. The calculus shares many desirable properties with the $\mathcal{ELH}$ procedure: it is worst-case optimal (it works in exponential time for a logic that is ExpTime-complete) and it allows for one-pass classification. Furthermore, it displays a pay-as-you-go behaviour, as it becomes analogous to the $\mathcal{ELH}$ procedure on an $\mathcal{ELH}$ ontology. Such behaviour is very convenient in applications, because the calculus can deal very effectively with ontologies that are "mostly" $\mathcal{ELH}$. This was proved in practice when the calculus was implemented in a reasoner called CB [24], which classified for the first time the full version of the GALEN ontology.

3.3 Reasoning with Nominals: Horn-$\mathcal{SROIQ}$ and $\mathcal{ELHO}$

Although the calculus in [2] for $\mathcal{EL}$++ included nominals, it was later found to be incomplete for handling them [26]. Complete consequence-based calculi for logics involving nominals were later developed for Horn-$\mathcal{SROIQ}$ [35] and $\mathcal{ELHO}$ [26]. In order to ensure completeness, the calculus in [26] had to keep track of "conditional" consequences, which only hold in models where some atomic concepts are non-empty. To see why this may be necessary, observe that an axiom of the form $C \sqsubseteq \{o\}$ in an ontology splits the models of the ontology into two kinds: those in which C is interpreted as the empty set and those where C is interpreted as a singleton; this obviously affects also the subconcepts of C.

Both of the aforementioned calculi introduced additional syntax to keep track of this kind of consequences. The Horn-$\mathcal{SROIQ}$ calculus in [35] introduces a

predicate rel which identifies non-empty concepts and inference rules to propagate this predicate when necessary. Similarly, the $\mathcal{ELHO}$ calculus from [26] introduces a new type of consequences which can be written as $G : C \sqsubseteq D$. This represents that inclusion $C \sqsubseteq D$ holds whenever concept G is not empty. Furthermore, the calculus also uses the dependency $G \rightsquigarrow H$ to represent that concept H is non-empty whenever concept G is non-empty. Notice that such consequences are not always "local" in the sense described in Section 2.2; indeed, the restriction to local consequences is difficult to marry with the presence of nominals in ontologies. It is often possible, however, to carefully constrain the syntax of consequences so that, in the absence of nominals, local forms are recovered.

Another interesting aspect of the Horn-$\mathcal{SROIQ}$ method is the introduction of a predicate equal to represent equality between individuals, together with inference rules that ensure such relations are congruences. Later calculi for DLs with number restrictions and/or nominals also require the use of equalities, as we show in Section 5.1, but instead of axiomatising equality, they use paramodulation-based inference rules [47]. The Horn-$\mathcal{SROIQ}$ calculus introduces also a rule to deal with the "terrible trifecta," a simultaneous interaction between nominals, inverse roles, and functional restrictions. The rule ensures that any two concepts or types related by a role R to the same nominal, and such that the inverse of R is functionally restricted, are satisfied by a single, unique domain element. This is recorded with the help of a special predicate same, which ensures that such a pair behaves, for the purposes of the calculus, like a nominal. Similar strategies have been used for other consequence-based calculi or in other approaches to reasoning for $\mathcal{SHOIQ}$, such as the resolution-based calculus in [28] or the tableau calculus in [19].

Both calculi are worst-case optimal, and the $\mathcal{ELHO}$ algorithm is also pay-as-you-go in the sense that it reduces to the standard $\mathcal{ELH}$ calculus in the absence of nominals.

3.4 Embracing Disjunction: $\mathcal{ALCH}$

The consequence-based calculus presented in [38] for $\mathcal{ALCH}$ was the first to support concept disjunction. The introduction of disjunction leads to difficulties: while previous calculi are such that a canonical model can be built from the saturated set of consequences to disprove any subsumption $A \sqsubseteq B$ absent from this set, such a model may not exist for ontologies with disjunction. The calculus proposed in [38] used an explicit representation of disjunctions which was similar to that used in resolution calculi for fragments of first-order logic. The calculus also introduced ordering, which dramatically reduces the number of inferences to consider [36] and helps to single out a model as "canonical," in case one exists. Some of the more recent calculi have also adopted this approach for dealing with disjunction [39, 10].

This calculus, like those for Horn DLs, transforms the input ontology into a normal form, which now allows for conjunctions of concepts of the form A, $\exists R.A$ and $\forall R.A$, and negation in front of atomic concepts. The presence of disjunction

increases the complexity of the representation of consequences, which may now be of the form:

$$\prod_{i=1}^{n} L_i \sqsubseteq \bigsqcup_{i=1}^{m} A_i \sqcup \exists R. \left(\prod_{i=n+1}^{k} L_i \right),$$

where each expression of the form L_i is an atomic concept or its negation, and each A_i is an atomic concept. Once again, the calculus is worst-case optimal and pay-as-you-go, as it simulates the Horn-$\mathcal{ALCH}$ and the $\mathcal{ELH}$ approaches for ontologies of the corresponding expressivity. One-pass classification is retained.

4 Interlude: A Unifying Framework for CB Reasoning

In 2014, Simančík and colleagues [39] extended the calculus in [38] to $\mathcal{ALCHI}$ while ensuring worst-case optimality, pay-as-you-go behaviour, and one-pass classification. As part of this work, they introduced a unifying framework for consequence-based reasoning that explicitly captures many of the aspects described in Section 2.2. This framework describes a graph-based *context structure*, where each node represents a context and the presence of an edge between contexts indicates that information can be transferred between those contexts. The set $\mathcal{S}$ of consequences derived by the calculus is split into sets $\mathcal{S}(v)$ associated to each context v. Furthermore, each context is associated to a particular set of concepts, called the *core* of the context; consequences in $\mathcal{S}(v)$ are relevant for all elements of a model that satisfy the core. Inference rules are applied to individual contexts, or to pairs of contexts that share an edge. Edges between contexts v and w are defined in cases where each element satisfying the core of v may have an R-filler that satisfies the core of w for some role R; each edge is labelled by the existential concept generating this connection. Figure 4 shows how the example of Section 2.2 could look like in this framework.

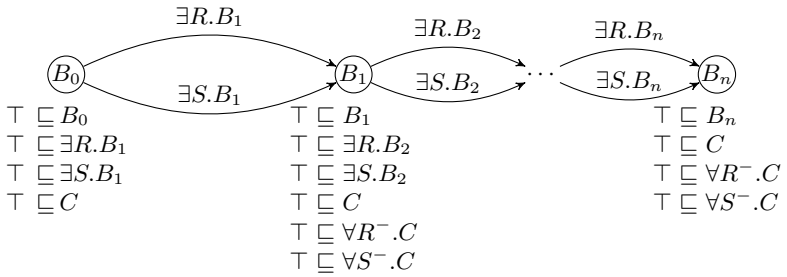

Fig. 3. Context structure for the example in Section 2.2. Cores of contexts are written inside their respective nodes, and below each node all the consequences derived for that context are listed. The first three inclusions in each of these sets are derived first and left-to-right; the inclusions involving C are derived afterwards and right-to-left.

Contexts in the example have only atomic concepts as cores; however, conjunctions of concepts are also allowed to represent cores. This may be relevant whenever consequences of the form $\top \sqsubseteq \exists R.A$ and $\top \sqsubseteq \forall R.B$ appear in the same context; in that case, one may wish to draw an edge to a context with core $A \sqcap B$. However, the calculus also makes it possible to draw an edge to a context with core A. Furthermore, if the context with the chosen core does not exist, the calculus may create it fresh. The policy for deciding how to draw edges between contexts and whether to introduce new contexts or re-use those already in the context structure is a parameter of the calculus called the *expansion strategy*. By suitably choosing this parameter, the framework can be used to simulate some of the calculi from previous sections. For instance, by allowing only cores with atomic concepts, the calculus described in Section 2.1 can be simulated; in contrast, to simulate the Horn-$\mathcal{SHIQ}$ calculus in [24], we need a strategy such that, for every relevant conjunction of atomic concepts M, a new context v_M is introduced with core M.

The calculus for $\mathcal{ALCHI}$ discussed in this section does not support nominals, and therefore it cannot simulate the calculi described in Section 3.3. However, one can find correspondences between constructs introduced in Section 3.3 to deal with nominals, and properties of a context structure created according to a suitable expansion strategy. For instance, assume an expansion strategy that uses a successor context v_A with core A whenever $\top \sqsubseteq \exists R.A$ appears in a context w (and otherwise it re-uses a context $v_\bot$ with empty core); then, the existence of a path from a context u to a context v implies that $\mathsf{core}_u \rightsquigarrow \mathsf{core}_v$, where $\rightsquigarrow$ is the reachability relation in the $\mathcal{ELHO}$ calculus in Section 3.3. Similarly, any context clause $\Gamma \sqsubseteq \Delta$ in a context v reachable from context u can be seen as a conditional consequence $\mathsf{core}_u\colon \Gamma \sqsubseteq \Delta$ from the $\mathcal{ELHO}$ calculus. These correspondences are exploited by some of the calculi for DLs with nominals discussed in Section 5.1.

5 Recent Developments: 2014 - 2019

This section discusses recent developments in consequence-based reasoning, including the extension of calculi to expressive DLs and the hybridisation of this approach with other well-known reasoning techniques.

5.1 Calculi for Expressive DLs

The framework described in Section 4 has been recently modified and extended to the DLs $\mathcal{SRIQ}$ [10] and $\mathcal{SROIQ}$ [43]. One of the most noteworthy aspects of these calculi is their use of first-order logic syntax with equality to represent both ontology axioms and derived consequences. This choice is motivated by the need of representing consequences stemming from the interactions between number restrictions, inverse roles, and/or nominals. Equalities participate in inferences that implement well-known paramodulation rules for equality reasoning [47].

These calculi are still pay-as-you-go and retain the one-pass classification property. In addition, the $\mathcal{SRIQ}$ calculus is also worst-case optimal, while the

$\mathcal{SROIQ}$ calculus terminates within worst-case optimal bounds except when ontologies feature a simultaneous interaction of disjunction with nominals, inverse roles, and "at-most" quantifiers. In practice, these occurrences are very rare.

The $\mathcal{SRIQ}$ calculus has been implemented in the reasoner Sequoia [11], which has proved to be competitive with state-of-the-art reasoners despite being an early prototype. The main outstanding practical issues in systems such as Sequoia are the following:

- The application of inference rules in the system can lead to the generation of clauses involving a large number of disjuncts. Resolution of such clauses with others can lead to combinatorial explosions due to repetitive inferences. For instance, suppose a clause of the form $\top \rightarrow A_1(x) \vee \cdots \vee A_n(x)$ is generated in a context, and suppose the ontology contains clause $A_i(x) \rightarrow B_i(x) \vee C_i(x)$ for each $1 \leq i \leq n$. If the ordering between literals in the context makes atoms of the form $B_i(x)$ and $C_j(x)$ smaller than atoms of the form $A_k(x)$, the calculus will derive 2^n clauses in the context. Most of these clauses are typically irrelevant since they do not participate in the derivation of target subsumptions.
- The system performs poorly in the presence of number restrictions involving elements of the model with many different successors by a particular role. For instance, if clauses $\{\top \rightarrow R(x, f_i(x)) \,|\, 1 \leq i \leq n\}$ are derived in a context, and the ontology contains a number restriction enforcing that no element may have more than m successors by R, then the calculus will derive $\binom{n}{m+1}$ different clauses. Furthermore, each of these clauses will have no less than $\binom{m+1}{2}$ disjuncts in the head; these long disjunctions may, in turn, exacerbate the issue discussed above.

5.2 Hybrid Methods

The consequence-based approach has been successfully combined with other well-known reasoning methods that can help overcome some of the aforementioned practical limitations. It has been suggested that the problem of generating long disjunctions may be addressed by means of an algebraic reasoner [28]. There currently exist consequence-based calculi that follow this strategy. The calculus in [45] for the DL $\mathcal{ELQ}$ incorporates calls to an external Integer Linear Programming component that is able to find solutions to algebraic constraint satisfaction problems based on the numeric restrictions appearing in derived consequences. This approach has been extended to the DL $\mathcal{SHOQ}$ in [23]. From a theoretical perspective, it remains unclear whether these calculi are worst-case optimal or whether they show pay-as-you-go behaviour. However, these systems have been implemented into the reasoner Avalanche [45], which shows promising results in ontologies with large number restrictions.

A different kind of hybridisation is proposed in [41], which describes the reasoner Konclude. This system works very efficiently in a wide range of expressive DLs. In contrast to all consequence-based calculi discussed so far, Konclude is

based on a tableau calculus, and therefore it attempts to answer queries by building a model. However, Konclude uses an incomplete version of a consequence-based calculus in order to generate as many (sound) consequences of the ontology as possible. These consequences are then used to aid in the construction of the tableau, and the result is a system that is highly competitive even in lightweight description logics such as $\mathcal{EL}$.

Another method for building a hybrid of tableau and consequence-based calculus is presented in [25] for the description logic $\mathcal{ALCHI}$. This calculus is once again based in deriving ontology entailments. However, it deals with disjunction in a way that is very different from the method described in Section 3.4. Instead of freely using resolution-like inference rules to generate long disjunctions, this algorithm makes non-deterministic choices during the saturation procedure which are reminiscent of those used by tableau-based algorithms. This calculus can explore alternative branches and backtrack when necessary, unlike previously discussed consequence-based calculi, which are purely deterministic. This prevents the problem of generating long disjunctions. The calculus is also worst-case optimal and enjoys pay-as-you-go behaviour. A prototype of this calculus has been implemented, and the evaluation shows promising results.

The reasoner MORe [1] provides yet another way of combining consequence-based reasoning with other techniques. The algorithm presented in [1] addresses the problem of ontology classification by decomposing an input ontology into modules, and then classifying each module using a reasoner most suited for the language used in that module. MORe has shown it can effectively classify ontologies using a consequence-based reasoner on $\mathcal{ELH}$ modules, and a reasoner such as HermiT [17], which is based on a tableau calculus, on the remaining modules. This approach is particularly useful for ontologies which have most axioms in a lightweight DL and a few axioms in an expressive logic.

6 Conclusion and Future Directions

Consequence-based reasoning has been a very active area of research for the last 15 years, and progress on this field shows no signs of slowing down. There is still no consequence-based calculi covering all expressive features of OWL 2 DL, as there is yet little progress in the area of consequence-based reasoning in expressive DLs with concrete domains. Further research is also needed in the area of optimisations and implementation techniques for consequence-based reasoning, especially for those calculi for the more expressive DLs. Furthermore, it is yet unclear whether hybrid approaches with algebraic reasoning and non-determinism can be effectively implemented for the whole of $\mathcal{SROIQ}$.

The application of consequence-based calculi to problems other than subsumption or classification (such as conjunctive query answering) remains also a fairly unexplored topic. A solid basis for this line of research is provided by "combined" approaches [29, 14, 13] to query answering that start with a materialisation phase similar to saturation in consequence-based reasoning, which is then followed by query-rewriting techniques.

For all these reasons, we think that consequence-based reasoning will continue being a dynamic area of research, one which holds promise for delivering the next generation of robust, efficient, and scalable reasoners for Description Logics.

References

1. Armas Romero, A., Kaminski, M., Cuenca Grau, B., Horrocks, I.: Module extraction in expressive ontology languages via datalog reasoning. J. Artif. Intell. Res. **55**, 499–564 (2016)
2. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ Envelope. In: Proc. of the 19th Int. Joint Conference on Artificial Intelligence. pp. 364–369. Morgan Kaufmann Publishers, Edinburgh, UK (2005)
3. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press, Cambridge, UK (2003)
4. Baader, F., Sattler, U.: An overview of tableau algorithms for description logics. Studia Logica **69**, 5–40 (2001)
5. Baader, F.: Terminological Cycles in a Description Logic with Existential Restrictions. In: Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence. pp. 325–330. Morgan Kaufmann Publishers, Acapulco, Mexico (2003)
6. Baader, F., Horrocks, I., Lutz, C., Sattler, U.: An Introduction to Description Logic. Cambridge University Press, Cambridge, UK (2017)
7. Baader, F., Küsters, R., Molitor, R.: Computing least common subsumers in description logics with existential restrictions. In: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence. pp. 96–103 (1999)
8. Baader, F., Lutz, C., Suntisrivaraporn, B.: CEL—a polynomial-time reasoner for life science ontologies. In: Automated Reasoning, Third International Joint Conference, Proceedings. vol. 4130, pp. 287–291. Springer-Verlag, Seattle, WA, USA (2006)
9. Bachmair, L., Ganzinger, H.: Resolution Theorem Proving. In: Robinson, A., Voronkov, A. (eds.) Handbook of Automated Reasoning, pp. 19–99. Elsevier Science, London, UK (2001)
10. Bate, A., Motik, B., Cuenca Grau, B., Simancik, F., Horrocks, I.: Extending consequence-based reasoning to SRIQ. In: Principles of Knowledge Representation and Reasoning: Proceedings of the Fifteenth International Conference. pp. 187–196. AAAI Press, Cape Town, South Africa (2016)
11. Bate, A., Motik, B., Cuenca Grau, B., Tena Cucala, D., Simancik, F., Horrocks, I.: Consequence-based reasoning for Description Logics with disjunctions and number restrictions. J. Artif. Intell. Res. **63**, 625–690 (2018)
12. Brandt, S.: Polynomial time reasoning in a Description Logic with existential restrictions, GCI axioms, and—what else? In: Proceedings of the 16th Eureopean Conference on Artificial Intelligence. pp. 298–302. Valencia, Spain (2004)
13. Carral, D., Dragoste, I., Krötzsch, M.: The combined approach to query answering in horn-alchoiq. In: Principles of Knowledge Representation and Reasoning: Proceedings of the Sixteenth International Conference. pp. 339–348. AAAI Press, Tempe, Arizona (2018)
14. Feier, C., Carral, D., Stefanoni, G., Grau, B.C., Horrocks, I.: The combined approach to query answering beyond the OWL 2 profiles. In: Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence. pp. 2971–2977. AAAI Press, Buenos Aires, Argentina (2015)

15. Ganzinger, H., de Nivelle, H.: A Superposition Decision Procedure for the Guarded Fragment with Equality. In: Proc. of the 14th IEEE Symposium on Logic in Computer Science. pp. 295–305. IEEE Computer Society, Trento, Italy (1999)
16. Georgieva, L., Hustadt, U., Schmidt, R.A.: Hyperresolution for Guarded Formulae. Journal of Symbolic Computation **36**(1–2), 163–192 (2003)
17. Glimm, B., Horrocks, I., Motik, B., Stoilos, G., Wang, Z.: HermiT: An OWL 2 Reasoner. Journal of Automated Reasoning **53**(3), 245–269 (2014)
18. Haarslev, V., Hidde, K., Möller, R., Wessel, M.: The RacerPro knowledge representation and reasoning system. Semantic Web **3**(3), 267–277 (2012)
19. Horrocks, I., Sattler, U.: A tableaux decision procedure for $\mathcal{SHOIQ}$. In: Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence. pp. 448–453. Edinburgh, UK (2005)
20. Hustadt, U., Schmidt, R.A.: Issues of Decidability for Description Logics in the Framework of Resolution. In: Caferra, R., Salzer, G. (eds.) Selected Papers from Automated Deduction in Classical and Non-Classical Logics, LNAI, vol. 1761, pp. 191–205. Springer, Berlin (1999)
21. Hustadt, U., Motik, B., Sattler, U.: Deciding Expressive Description Logics in the Framework of Resolution. Information & Computation **206**(5), 579–601 (2008)
22. Hustadt, U., Schmidt, R.A.: Using Resolution for Testing Modal Satisfiability and Building Models. Journal of Automated Reasoning **28**(2), 205–232 (2002)
23. Karahroodi, N.Z., Haarslev, V.: A Consequence-based Algebraic Calculus for $\mathcal{SHOQ}$. In: Proc. of the 30th Int. Workshop on Description Logics. CEUR Workshop Proceedings, vol. 1879. Montpellier, France (2017)
24. Kazakov, Y.: Consequence-Driven Reasoning for Horn $\mathcal{SHIQ}$ Ontologies. In: Proc. of the 21st Int. Joint Conf. on Artificial Intelligence. pp. 2040–2045. Pasadena, CA, USA (2009)
25. Kazakov, Y., Klinov, P.: Bridging the gap between tableau and consequence-based reasoning. In: Informal Proceedings of the 27th International Workshop on Description Logics. pp. 579–590. Viena, Austria (2014)
26. Kazakov, Y., Krötzsch, M., Simančík, F.: Practical Reasoning with Nominals in the EL Family of Description Logics. Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning (2012)
27. Kazakov, Y., Krötzsch, M., Simančík, F.: The incredible ELK — from polynomial procedures to efficient reasoning with $\mathcal{EL}$ ontologies. J. Autom. Reasoning **53**(1), 1–61 (2014)
28. Kazakov, Y., Motik, B.: A resolution-based decision procedure for $\mathcal{SHOIQ}$. J. Autom. Reasoning **40**(2–3), 89–116 (2008)
29. Kontchakov, R., Lutz, C., Toman, D., Wolter, F., Zakharyaschev, M.: The combined approach to query answering in dl-lite. In: Principles of Knowledge Representation and Reasoning: Proceedings of the Twelfth International Conference. AAAI Press, Toronto, Ontario (2010)
30. Lutz, C.: Complexity of terminological reasoning revisited. In: Logic Programming and Automated Reasoning, 6th International Conference. vol. 1705, pp. 181–200. Springer-Verlag, Tbilisi, Georgia (1999)
31. Metke-Jimenez, A., Lawley, M.: Snorocket 2.0: Concrete domains and concurrent classification. In: Informal Proceedings of the 2nd International Workshop on OWL Reasoner Evaluation. vol. 1015, pp. 32–38. Ulm, Germany (2013)
32. Motik, B.: KAON2 - scalable reasoning over ontologies with large data sets. ERCIM News **2008**(72) (2008)
33. Motik, B., Cuenca Grau, B., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C.: OWL 2 Web Ontology Language Profiles, `http://www.w3.org/TR/owl2-profiles/`

34. de Nivelle, H., Schmidt, R.A., Hustadt, U.: Resolution-Based Methods for Modal Logics. Logic Journal of the IGPL **8**(3), 265–292 (2000)
35. Ortiz, M., Rudolph, S., Simkus, M.: Worst-Case Optimal Reasoning for the Horn-DL Fragments of OWL 1 and 2. In: Proceedings of the Twelfth International Conference on the Principles of Knowledge Representation and Reasoning. AAAI Press, Toronto, Canada (2010)
36. Robinson, A., Voronkov, A. (eds.): Handbook of Automated Reasoning. Elsevier (2001)
37. Schmidt, R.A., Hustadt, U.: First-Order Resolution Methods for Modal Logics. In: Voronkov, A., Weidenbach, C. (eds.) Programming Logics—Essays in Memory of Harald Ganzinger. LNCS, vol. 7797, pp. 345–391. Springer (2013)
38. Simančík, F., Kazakov, Y., Horrocks, I.: Consequence-based reasoning beyond Horn ontologies. In: Proceedings of the 22nd International Joint Conference on Artificial Intelligence. pp. 1093–1098. IJCAI/AAAI, Barcelona, Spain (2011)
39. Simančík, F., Motik, B., Horrocks, I.: Consequence-based and fixed-parameter tractable reasoning in description logics. Artif. Intell. **209**, 29–77 (2014)
40. Sirin, E., Parsia, B., Cuenca Grau, B., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. Journal of Web Semantics **5**(2), 51–53 (2007)
41. Steigmiller, A., Liebig, T., Glimm, B.: Konclude: System description. J. Web Semant. **27**(1) (2014)
42. Tena Cucala, D., Cuenca Grau, B., Horrocks, I.: Consequence-based reasoning for description logics with disjunction, inverse roles, and nominals. In: Proceedings of the 30th International Workshop on Description Logics. CEUR-WS.org, Montpelier, France (2017)
43. Tena Cucala, D., Cuenca Grau, B., Horrocks, I.: Consequence-based reasoning for Description Logics with disjunction, inverse roles, number restrictions, and nominals. In: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence. pp. 1970–1976. ijcai.org, Stockholm, Sweden (2018)
44. Tsarkov, D., Horrocks, I.: FaCT++ Description Logic Reasoner: System Description. In: Proc. of the 3rd Int. Joint Conf. on Automated Reasoning (IJCAR 2006). LNAI, vol. 4130, pp. 292–297. Springer, Seattle, WA, USA (2006)
45. Vlasenko, J., Daryalal, M., Haarslev, V., Jaumard, B.: A Saturation-based Algebraic Reasoner for ELQ. In: Proceedings of the 5th Workshop on Practical Aspects of Automated Reasoning co-located with International Joint Conference on Automated Reasoning. pp. 110–124. Coimbra, Portugal (2016)
46. W3C OWL Working Group: OWL 2 Web Ontology Language Overview, `http://www.w3.org/TR/owl2-overview/`
47. Wos, L., Robinson, G.: Paramodulation and theorem-proving in first-order theories with equality. In: Michie, D., Meltzer, B. (eds.) Proceedings of the 4th Annual Machine Intelligence Workshop. Edinburgh University Press, Edinburgh, UK (1969)