CEL—A Polynomial-time Reasoner for Life Science Ontologies*

Franz Baader, Carsten Lutz, Boontawee Suntisrivaraporn

Theoretical Computer Science, TU Dresden, Germany
{baader,lutz,meng}@tcs.inf.tu-dresden.de

Abstract. CEL (Classifier for $\mathcal{EL}$) is a reasoner for the small description logic $\mathcal{EL}^+$ which can be used to compute the subsumption hierarchy induced by $\mathcal{EL}^+$ ontologies. The most distinguishing feature of CEL is that, unlike all other modern DL reasoners, it is based on a polynomial-time subsumption algorithm, which allows it to process very large ontologies in reasonable time. In spite of its restricted expressive power, $\mathcal{EL}^+$ is well-suited for formulating life science ontologies.

The Description Logic underlying CEL

The system CEL[1] is a first step towards realizing the dream of a description logic system that offers both sound and complete polynomial-time algorithms and expressive means that allow its use in real-world applications. It is based on recent theoretical advances that have shown that the description logic (DL) $\mathcal{EL}$, which allows for conjunction and existential restrictions, and some of its extensions have a polynomial-time subsumption problem even in the presence of concept definitions and so-called general concept inclusions (GCI) [1]. The DL $\mathcal{EL}^+$ handled by CEL extends $\mathcal{EL}$ by so-called role inclusions (RI). On the practical side, it has turned out that the expressive power of $\mathcal{EL}^+$ is sufficient to express several large life science ontologies. In particular, the Systematized Nomenclature of Medicine (SNOMED) [4] employs $\mathcal{EL}$ with RIs and acyclic concept definitions. The Gene Ontology (GO) [3] can also be expressed in $\mathcal{EL}$ with acyclic concept definitions and one transitive role (which is a special case of an RI). Finally, large parts of the Galen Medical Knowledge Base (GALEN) [5] can be expressed in $\mathcal{EL}$ with GCIs and RIs.

Because of the space limitations, we cannot introduce the syntax and semantics of $\mathcal{EL}^+$ in detail. We just mention the syntax elements, and illustrate their use by a small example. Full definitions can be found in [1, 2]. Like in other DLs, $\mathcal{EL}^+$ *concepts* are inductively defined starting with the sets of *concept names* $\mathsf{N_C}$ and *role names* $\mathsf{N_R}$. Each concept name A is a concept, and so are the *top concept* $\top$, *conjunction* $C \sqcap D$, and *existential restriction* $\exists r.C$. An $\mathcal{EL}^+$ *ontology* is a finite set of *general concept inclusions (GCI)* of the form $C \sqsubseteq D$ for concepts

* This work was partially supported by the EU funded IST-2005-7603 FET Project Thinking Ontologies (TONES).

[1] CEL can be downloaded from `http://lat.inf.tu-dresden.de/systems/cel/`.

C, D, and *complex role inclusions (RI)* of the form $r_1 \circ \cdots \circ r_n \sqsubseteq s$ for roles $r_1, \ldots, r_n, s$. A *primitive concept definition* (PCDef) $A \sqsubseteq D$ is a GCI with the left-hand side a concept name, while a (non-primitive) *concept definition* (CDef) $A \equiv D$ can be expressed using two GCIs. It is worthwhile to note that RIs generalize at least three expressive means important in bio-medical applications: role hierarchy, transitive role, and so-called right-identity axioms [4]. One of the most prominent inference problems for DL ontologies is *classification*: compute the subsumption hierarchy of all concept names occurring in the ontology.

$$\begin{aligned}
\mathsf{Endocardium} &\sqsubseteq \mathsf{Tissue} \sqcap \exists \mathsf{cont\text{-}in}.\mathsf{HeartWall} \sqcap \exists \mathsf{cont\text{-}in}.\mathsf{HeartValve} \\
\mathsf{HeartWall} &\sqsubseteq \mathsf{BodyWall} \sqcap \exists \mathsf{part\text{-}of}.\mathsf{Heart} \\
\mathsf{HeartValve} &\sqsubseteq \mathsf{BodyValve} \sqcap \exists \mathsf{part\text{-}of}.\mathsf{Heart} \\
\mathsf{Endocarditis} &\sqsubseteq \mathsf{Inflammation} \sqcap \exists \mathsf{has\text{-}loc}.\mathsf{Endocardium} \\
\mathsf{Inflammation} &\sqsubseteq \mathsf{Disease} \sqcap \exists \mathsf{acts\text{-}on}.\mathsf{Tissue} \\
\mathsf{HeartDisease} &\equiv \mathsf{Disease} \sqcap \exists \mathsf{has\text{-}loc}.\mathsf{Heart} \\
\mathsf{part\text{-}of} &\sqsubseteq \mathsf{cont\text{-}in} \\
\mathsf{has\text{-}loc} \circ \mathsf{cont\text{-}in} &\sqsubseteq \mathsf{has\text{-}loc}
\end{aligned}$$

Fig. 1. An example $\mathcal{EL}^+$ ontology (motivated by GALEN).

As an example, we consider the $\mathcal{EL}^+$ ontology in Fig. 1, where all capitalized words are concept names and all lowercase words are role names. This small ontology contains 5 GCIs (which are indeed PCDefs), a CDef, and 2 RIs (more precisely a role hierarchy and a right-identity axiom) expressing a piece of clinical knowledge about *endocarditis* and related concepts and roles. It is not hard to infer from this ontology that endocarditis is classified as heart disease, i.e., $\mathsf{Endocarditis} \sqsubseteq_{\mathcal{O}} \mathsf{HeartDisease}$. In fact, (i) Endocarditis implies Inflammation and thus Disease, which yields the first conjunct in the definition of HeartDisease. Moreover, (ii) $\exists$has-loc.Endocardium implies $\exists$has-loc.$\exists$cont-in.HeartWall and thus $\exists$has-loc.$\exists$cont-in.$\exists$part-of.Heart, which, in the presence of both RIs, implies $\exists$has-loc.Heart, satisfying the second conjunct in the definition of HeartDisease.

The CEL System

The algorithm implemented in CEL is based on the restriction to $\mathcal{EL}^+$ of the polytime classification algorithm for the more expressive DL $\mathcal{EL}^{++}$ introduced in [1]. To classify an ontology, the algorithm first transforms it into *normal form*, which requires all GCIs and RIs to be in one of the forms shown in the left part of Fig. 2. By introducing new concept and role names and applying a number of straightforward rewriting rules, any $\mathcal{EL}^+$ ontology $\mathcal{O}$ can be transformed into a normalized one such that subsumption between the concept names occurring in $\mathcal{O}$ is preserved. The normalization can be carried out in linear time, yielding an ontology whose size is linear in the size of the original one [1]. After normalization, the algorithm computes two mappings: $S : \mathsf{N}_\mathsf{C}^\top \longrightarrow 2^{\mathsf{N}_\mathsf{C}^\top}$ and

$A_1 \sqcap \cdots \sqcap A_n \sqsubseteq B$	**CR1**	If $\{A_1, \ldots, A_n\} \subseteq S(X)$, $A_1 \sqcap \cdots \sqcap A_n \sqsubseteq B \in \mathcal{O}$, and $B \notin S(X)$ then $S(X) := S(X) \cup \{B\}$
$A \sqsubseteq \exists r.B$	**CR2**	If $A \in S(X)$, $A \sqsubseteq \exists r.B \in \mathcal{O}$, and $(X, B) \notin R(r)$ then $R(r) := R(r) \cup \{(X, B)\}$
$\exists r.A \sqsubseteq B$	**CR3**	If $(X, Y) \in R(r)$, $A \in S(Y)$, $\exists r.A \sqsubseteq B \in \mathcal{O}$, and $B \notin S(X)$ then $S(X) := S(X) \cup \{B\}$
$r \sqsubseteq s$	**CR4**	If $(X, Y) \in R(r)$, $r \sqsubseteq s \in \mathcal{O}$, and $(X, Y) \notin R(s)$ then $R(s) := R(s) \cup \{(X, Y)\}$
$r \circ s \sqsubseteq t$	**CR5**	If $(X, Z) \in R(r)$, $(Z, Y) \in R(s)$, $r \circ s \sqsubseteq t \in \mathcal{O}$, and $(X, Y) \notin R(t)$ then $R(t) := R(t) \cup \{(X, Y)\}$

Fig. 2. Normal Form and Completion Rules

$R : \mathsf{N_R} \longrightarrow 2^{(\mathsf{N_C^\top} \times \mathsf{N_C^\top})}$ where $\mathsf{N_C^\top}$ is $\mathsf{N_C}$ augmented by $\top$. The intuition is that these mappings make implicit subsumption relationships explicit in the sense that $B \in S(A)$ implies $A \sqsubseteq_{\mathcal{O}} B$, and $(A, B) \in R(r)$ implies $A \sqsubseteq_{\mathcal{O}} \exists r.B$. The mappings are initialized by setting $S(A) := \{A, \top\}$ and $R(r) := \emptyset$. Then the sets $S(A)$ and $R(r)$ are extended by applying the completion rules shown in the right part of Fig. 2 until no more rule applies. As a result, the mapping S computed this way satisfies $B \in S(A)$ iff $A \sqsubseteq_{\mathcal{O}} B$, i.e., $S(A)$ contains all subsumers of A. Note that this algorithm computes the subsumption relationships between *all* pairs of concept names.

It is obvious that, when implementing this algorithm, an efficient approach for finding an applicable rule must be developed. To avoid searching for such rules, we use a set of queues, one for each concept name appearing in the input ontology, to guide the application of completion rules. Intuitively, the queues list modifications to $S(A)$ and $R(A)$ that still have to be carried out. The fact that such an addition triggers other rules is taken into account by appropriately extending the queues when the addition is performed (see [2] for a detailed description). With a relatively straightforward implementation (in Common LISP) of this idea, we were able to classify the large SNOMED ontology (see below) in less than 4 hours (see [2] for this and other experimental results). Since then, however, we have further improved the implementation by changing the strategy of rule applications, changing the encoding of concept and role names, and low-level optimizations on the data structures. These optimizations have enhanced the performance of CEL on large real-world ontologies. In particular, CEL can now classify SNOMED in less than half an hour (see below).

The CEL interface. CEL currently accepts input based on a small extension of the KRSS syntax.[2] It is currently equipped with a very simple shell-like interface that provides users with all essential functionalities, including a simple interactive help command. The user can either load an $\mathcal{EL}^+$ ontology formulated in KRSS syntax into the system from a file by calling `(load-ontology`

[2] see `http://dl.kr.org/krss-spec.ps`

filename) or enter interactively at the prompt each axiom of the ontology. The normalization is carried out while the ontology is being loaded, and once normalization is finished, `(classify-ontology)` can be invoked to classify all concept names occurring in the ontology (eager subsumption approach). We have modified the algorithm described above to a goal-directed variant such that a single subsumption query between 2 concept names `(subsumes? B A)` can be answered without needing to classify the whole ontology first (lazy subsumption approach). After having classified the whole ontology, CEL allows the user to output the classification results in different formats: `(output-supers)` to output the sets $S(A)$ for all concept names A occurring in $\mathcal{O}$; `(output-taxonomy)` to output the Hasse diagram of the subsumption hierarchy, i.e., just the *direct* parent-child relationships; and `(output-hierarchy)` to output the hierarchy as a graphical indented tree.

Through its command-line options, CEL can also work as a stand-alone reasoner without interaction from users. For instance, the command line:

```
$cel -l filename -c -outputHierarchy -q
```

can be entered to load and classify an ontology from *filename*, and then output the hierarchy. For a more detailed description of the CEL interface, we refer to the CEL user manual (available on the CEL homepage).

Performance evaluation. The empirical results for the performance of CEL described below show that it can compete with, and often outperforms, the fastest tableau-based DL systems. We have compared the performance of CEL with three of the most advanced DL systems: FaCT^{++} (v1.1.0), RacerMaster (v1.9.0), and Pellet (v1.3b). These systems implement tableau-based decision procedures for expressive DLs in which subsumption is ExpTime-complete. All experiments have been performed on a PC with 2.8GHz Intel Pentium 4 processor and 512MB memory running Linux v2.6.14. For Pellet, we used JVM v1.5 and set the Java heap space to 256MB (as recommended by the implementers).

Our experiments are based on three important bio-medical ontologies: Go, Galen, and Snomed. Since Galen uses some expressivity that CEL cannot handle, we have simplified it by removing inverse role axioms and treating functional roles as ordinary ones, and obtained an $\mathcal{EL}^+$ ontology $\mathcal{O}^{\text{Galen}}$. (Of course, also the other reasoners, which could have handled inverse and functional roles, were applied to $\mathcal{O}^{\text{Galen}}$ rather than full Galen.) We have obtained two other benchmarks, $\mathcal{O}^{\text{Go}}$ and $\mathcal{O}^{\text{Snomed}}$, from the other two ontologies. However, Snomed has one right-identity rule similar to the last axiom in our example (see Fig. 1). This axiom is passed to CEL, but not to the other reasoners, as the latter do not support right identities. Additionally, to get a smaller version of Snomed that can be dealt with by standard DL reasoners, we also consider a fragment obtained by keeping only CDefs, and call it $\mathcal{O}^{\text{Snomed}}_{\text{core}}$. Some information on the size and structure of these benchmarks is given in the upper part of Table 1, where the first row shows the numbers of PCDef, CDef, and GCI axioms, respectively. The results of our experiments are summarized in the lower part of Table 1, where all classification times are shown in seconds and *unattainable*

	$\mathcal{O}^{\text{GO}}$	$\mathcal{O}^{\text{GALEN}}$	$\mathcal{O}^{\text{SNOMED}}_{\mathsf{core}}$	$\mathcal{O}^{\text{SNOMED}}$
concept axioms	20,465/0/0	2,041/699/1,214	0/38,719/0	340,972/38,719/0
role axioms	1	438	0	$11 + 1$
$\lvert\mathsf{N_C}\rvert$	20,465	2,740	53,234	379,691
$\lvert\mathsf{N_R}\rvert$	1	413	52	52
CEL	5.8	14	95	1,782
FaCT^{++}	6.9	50	740	3,859
RacerMaster	19	14	34,709	*unattainable*
Pellet	1,357	75	*unattainable*	*unattainable*

Table 1. Benchmarks and Evaluation Results

means that the reasoner failed due to memory exhaustion. Notable, CEL outperforms all the reasoners in all benchmarks except $\mathcal{O}^{\text{GALEN}}$, where RacerMaster is as fast. CEL and FaCT^{++} are the only reasoners that can classify $\mathcal{O}^{\text{SNOMED}}$, whereas RacerMaster and Pellet fail. Pellet and the original version of FaCT (not shown in the table) even fail to classify $\mathcal{O}^{\text{SNOMED}}_{\mathsf{core}}$. It seems worth noting that the performance of FaCT^{++} degrades dramatically if $\mathcal{O}^{\text{SNOMED}}$ is extended with real GCIs. For instance, FaCT^{++} needs about 3,000 more seconds to classify $\mathcal{O}^{\text{SNOMED}}$ for each additional GCI of the form $\exists r.C \sqsubseteq D$, whereas the performance of CEL does not change noticeably if we add such GCIs.

Conclusion

We view these results as a strong argument for the use of tractable DLs based on extensions of $\mathcal{EL}$. As illustrated by the above performance evaluation, CEL is suitable for practical reasoning on very large life science ontologies. Developing CEL is ongoing work. We plan to extend its capabilities to the DL $\mathcal{EL}^{++}$ [1], with which one can express, among other things, *nominals* and *disjoint concepts*. We also plan to implement the DIG and OWL interface, so that CEL can be used as a backend reasoner for ontology editors such as OilEd and Protégé, which would also make their sophisticated user-interfaces available to users of CEL.

References

1. F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope. In *IJCAI-05*, Edinburgh, UK, 2005. Morgan-Kaufmann Publishers.
2. F. Baader, C. Lutz, and B. Suntisrivaraporn. Is tractable reasoning in extensions of the description logic $\mathcal{EL}$ useful in practice? In *Proceedings of the 2005 International Workshop on Methods for Modalities (M4M-05)*, 2005.
3. The Gene Ontology Consortium. Gene Ontology: Tool for the unification of biology. *Nature Genetics*, 25:25–29, 2000.
4. R. Cote, D. Rothwell, J Palotay, R. Beckett, and L. Brochu. The systematized nomenclature of human and veterinary medicine. Technical report, SNOMED International, Northfield, IL: College of American Pathologists, 1993.

5. A. Rector and I. Horrocks. Experience building a large, re-usable medical ontology using a description logic with transitivity and concept inclusions. In *Proceedings of the Workshop on Ontological Engineering, AAAI Spring Symposium (AAAI'97)*, Stanford, CA, 1997. AAAI Press.