CEL—A Short System Demonstration*

Boontawee Suntisrivaraporn, Franz Baader, Carsten Lutz
Theoretical Computer Science, TU Dresden, Germany
{meng,baader,clu}@tcs.inf.tu-dresden.de

Introduction to the CEL system.

Description logics (DLs) are an important family of formalisms for reasoning about ontologies. CEL (Classifier for $\mathcal{EL}$) is a free (for non-commercial use) LISP-based reasoner for the description logic $\mathcal{EL}^+$ [2], supporting as its main reasoning task the computation of the subsumption hierarchy induced by $\mathcal{EL}^+$ ontologies. The most distinguishing feature of CEL is that, unlike other modern DL reasoners, it implements a polynomial-time algorithm, which allows it to process very large ontologies in reasonable time. The underlying description logic $\mathcal{EL}^+$ is a practically useful sub-language of $\mathcal{EL}^{++}$ introduced in [1]. Despite being less expressive than other description logics such as $\mathcal{SHIQ}$ and OWL, it offers a selected set of expressive means that are tailored towards the formulation of biological and medical ontologies, some of the most prominent of which are the Gene Ontology (Go) [7], the Galen Medical Knowledge Base (Galen) [5], and the Systematized Nomenclature of Medicine (Snomed) [3, 6].

The purpose of the present short paper is to demonstrate the functionalities of CEL and their use in applications. The implemented algorithm is described in detail in [2]. In the following section, a quick guide to system usage will be given, including the system requirements. In the last section, performance evaluation of CEL on real-world life science ontologies will be discussed.

CEL at work.

The CEL system is available as a binary executable which can run on most Linux platforms. The latest version is CEL `v0.9b` which includes all features illustrated in this system demonstration. The distribution bundle can be obtained from:

`http://lat.inf.tu-dresden.de/systems/cel/`

The package consists of the CEL executable, the user manual, and some example $\mathcal{EL}^+$ ontologies. After extracting the bundle, the executable `cel` under `bin` can be invoked without installation. However, the following system requirements are assumed:

- Linux operating system;[1]

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

[1]It has been tested successfully on RedHat, Debian, and SuSE.

Concept	DL Syntax	CEL Syntax
the top concept	$\top$	top
the bottom concept	$\bot$	bottom
conjunction	$C_1 \sqcap \cdots \sqcap C_n$	(and $C_1 \cdots C_n$)
existential restriction	$\exists r.C$	(some r C)

Table 1: Syntax of $\mathcal{EL}^+$ concepts.

Ontology axioms	DL Syntax	CEL Syntax
primitive concept definition	$A \sqsubseteq D$	(define-primitive-concept A D)
concept definition	$A \equiv D$	(define-concept A D)
general concept inclusion	$C \sqsubseteq D$	(implies C D)
concept equivalence axiom	$C \equiv D$	(equivalent C D)
concept disjointness axiom	$C_1 \sqcap \cdots \sqcap C_n \sqsubseteq \bot$	(disjoint $C_1 \cdots C_n$)
role domain axiom	$\exists r.\top \sqsubseteq C$	(define-primitive-role r :domain C)
role hierarchy axiom	$r \sqsubseteq s$	(define-primitive-role r :parent s)
transitive role axiom	$r \circ r \sqsubseteq r$	(define-primitive-role r :transitive t)
right-identity axiom	$r \circ s \sqsubseteq r$	(define-primitive-role r :right-identity s)
left-identity axiom	$s \circ r \sqsubseteq r$	(define-primitive-role r :left-identity s)
complex role inclusion	$r_1 \circ r_2 \sqsubseteq s$	(role-inclusion (compose r_1 r_2) s)

Table 2: Syntax $\mathcal{EL}^+$ ontology axioms.

- Physical memory at least 128MB;[2]
- At least 8MB of available hard-disk space.

CEL as a stand-alone reasoner. In order to use CEL to classify an ontology, the user must already have the ontology formulated in $\mathcal{EL}^+$ in a small extension of the KRSS syntax [4], henceforth called CEL syntax. With this LISP-like syntax, it is easy to port existing ontologies that have been used with well-known DL reasoners like FaCT and RACER. Table 1 shows the concept constructors available in $\mathcal{EL}^+$ that can be used to form concept descriptions. For building up ontologies, the expressive means shown in Table 2 can be used, where conventionally A, B denotes a named concept, C, D concept descriptions, and r, s named roles. Though only implies and role-inclusion axioms can sufficiently model any $\mathcal{EL}^+$ ontology, it is often very useful and also makes the ontology more comprehensible to provide auxiliary axioms. An $\mathcal{EL}^+$ ontology is effectively a text file containing axioms of the forms shown in the right column of Table 1.

For example, Figure 1 shows a medical ontology about the concept Endocarditis (an inflammation of a heart valve) formulated in CEL syntax. The user can either load this ontology into the system by calling `(load-ontology "med.tbox")` or enter interactively at the prompt each axiom of the ontology. The preprocess is carried out while the ontology is being loaded, and once this is finished, `(classify-ontology)` can be invoked

[2]Considerably more memory may be needed for larger ontologies.

```
(define-primitive-role contained-in)
(define-primitive-role part-of :parent contained-in :transitive t)
(define-primitive-role has-location :right-identity contained-in)

(define-primitive-concept Endocardium
                          (and Tissue
                               (some contained-in HeartWall)
                               (some contained-in HeartValve)))
(define-primitive-concept HeartWall
                          (and BodyWall
                               (some part-of Heart)))
(define-primitive-concept HeartValve
                          (and BodyValve
                               (some part-of Heart)))
(define-primitive-concept Endocarditis
                          (and Inflammation
                               (some has-location Endocardium)))
(define-primitive-concept Inflammation
                          (and Disease
                               (some acts-on Tissue)))

(define-concept Heartdisease
                (and Disease
                     (some has-location Heart)))

(implies (and Heartdisease
              (some has-location HeartValve))
         CriticalHeartDisease)
```

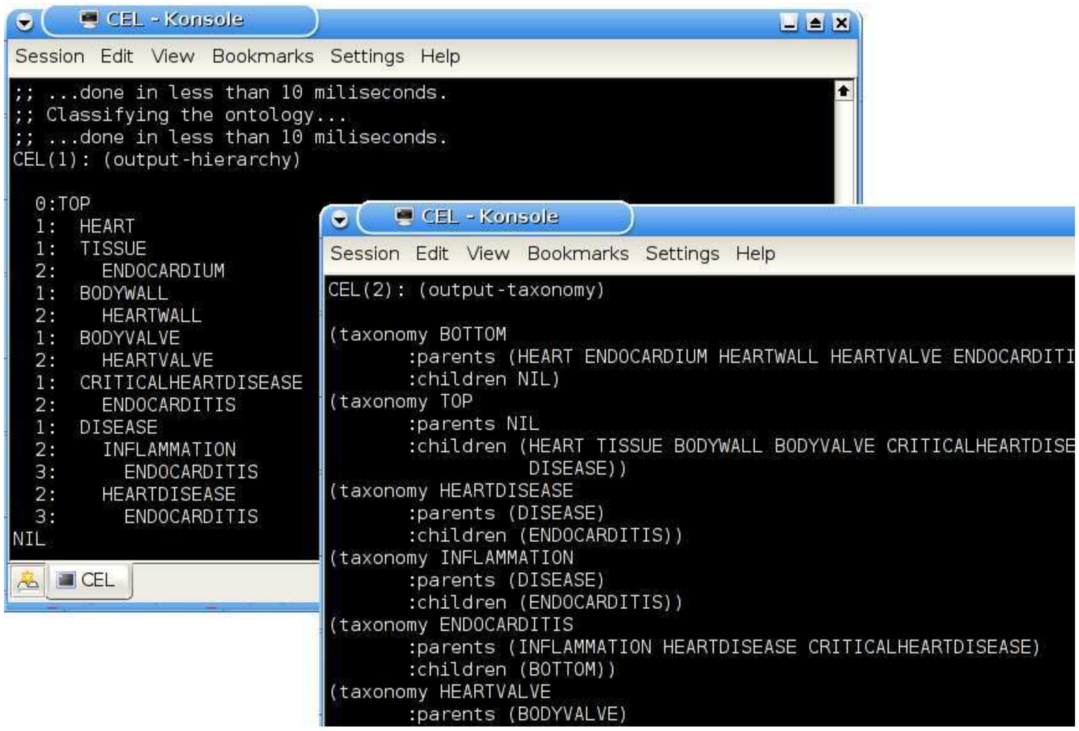
Figure 1: An $\mathcal{EL}^+$ ontology from file `med.tbox`.

to classify all concept names occurring in the ontology (eager subsumption approach). Subsumption query between two concept names can be queried using `(subsumes? B A)`. If this is called after classification, it simply looks up in the computed subsumption hierarchy. Otherwise, it runs a single subsumption test and answers 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 subsumer sets for all concept names occurring in the ontology; `(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. As an example, Figure 2 depicts screen shots of the results of `(output-hierarchy)` and `(output-taxonomy)` after classifying the ontology `med.tbox`.

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


CEL - Konsole
Session Edit View Bookmarks Settings Help
CEL

```
;; ...done in less than 10 miliseconds.
;; Classifying the ontology...
;; ...done in less than 10 miliseconds.
CEL(1): (output-hierarchy)

  0:TOP
  1:  HEART
  1:  TISSUE
  2:    ENDOCARDIUM
  1:  BODYWALL
  2:    HEARTWALL
  1:  BODYVALVE
  2:    HEARTVALVE
  1:  CRITICALHEARTDISEASE
  2:    ENDOCARDITIS
  1:  DISEASE
  2:    INFLAMMATION
  3:      ENDOCARDITIS
  2:    HEARTDISEASE
  3:      ENDOCARDITIS
NIL
```

CEL - Konsole
Session Edit View Bookmarks Settings Help

```
CEL(2): (output-taxonomy)

(taxonomy BOTTOM
       :parents (HEART ENDOCARDIUM HEARTWALL HEARTVALVE ENDOCARDITI
       :children NIL)
(taxonomy TOP
       :parents NIL
       :children (HEART TISSUE BODYWALL BODYVALVE CRITICALHEARTDISE
                  DISEASE))
(taxonomy HEARTDISEASE
       :parents (DISEASE)
       :children (ENDOCARDITIS))
(taxonomy INFLAMMATION
       :parents (DISEASE)
       :children (ENDOCARDITIS))
(taxonomy ENDOCARDITIS
       :parents (INFLAMMATION HEARTDISEASE CRITICALHEARTDISEASE)
       :children (BOTTOM))
(taxonomy HEARTVALVE
       :parents (BODYVALVE)
```

Figure 2: CEL with its innate interactive interface

manual (available on the CEL homepage).

CEL as a backend reasoner. Alternatively, the user can also exploit CEL reasoning capabilities through the DIG interface[3] and a graphical ontology editor. To do this, CEL has to be started as a DIG reasoning server by the following command line:

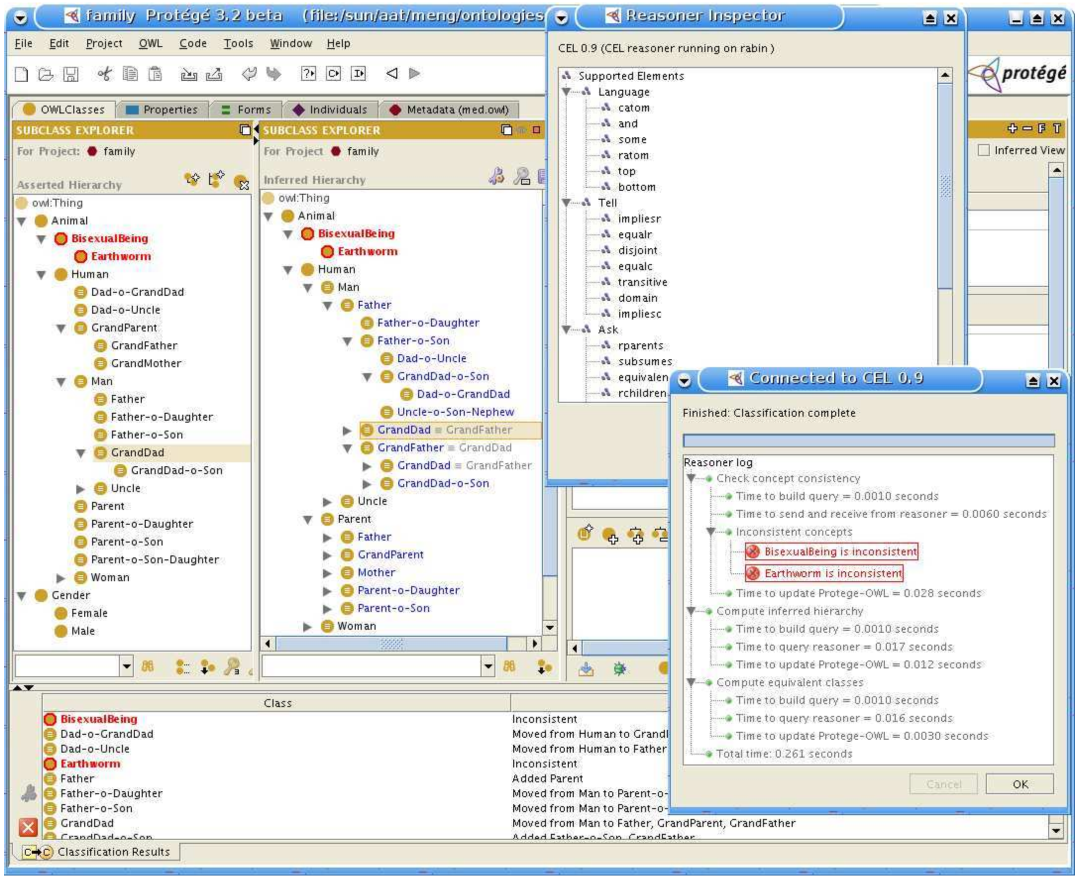
```
$cel -digServer [port]
```

where *port* is defaulted to 8080 but can be overridden.

Once started in this mode, ontology editors can be connected to CEL and exploit its reasoning power either locally or remotely via the Internet. The upper floating dialog in Figure 3, "Reasoner Inspector," displays the expressive means that can be handled by CEL in terms of DIG language. The DIG interface for CEL has been tested successfully with Protégé OWL editor.[4] The main window in Figure 3 illustrates the asserted subsumption hierarchy (input) and the inferred subsumption hierarchy (output) within the editor, whereas the small floating dialog, "Connected to CEL 0.9," displays the interactions between the DIG client and the DIG server.

[3]The DIG (DL Implementation Group) interface is an XML-based standard that defines an interfacing language for seamless communication between a DL service provider (DIG server) and a DL application (DIG client). See `http://dl.kr.org/dig/`

[4]See `http://protege.stanford.edu/plugins/owl/`


family Protégé 3.2 beta (file:/sun/aat/meng/ontologies
File Edit Project OWL Code Tools Window Help
OWLClasses Properties Forms Individuals Metadata (med.owl)
SUBCLASS EXPLORER
For Project: family
Asserted Hierarchy
Inferred Hierarchy
Reasoner Inspector
CEL 0.9 (CEL reasoner running on rabin)
Supported Elements
Language
Tell
Ask
protégé
Inferred View
Connected to CEL 0.9
Classification Results

```
Finished: Classification complete

Reasoner log
  Check concept consistency
    Time to build query = 0.0010 seconds
    Time to send and receive from reasoner = 0.0060 seconds
    Inconsistent concepts
      BisexualBeing is inconsistent
      Earthworm is inconsistent
    Time to update Protege-OWL = 0.028 seconds
  Compute inferred hierarchy
    Time to build query = 0.0010 seconds
    Time to query reasoner = 0.017 seconds
    Time to update Protege-OWL = 0.012 seconds
  Compute equivalent classes
    Time to build query = 0.0010 seconds
    Time to query reasoner = 0.016 seconds
    Time to update Protege-OWL = 0.0030 seconds
  Total time: 0.261 seconds
```

Class	
BisexualBeing	Inconsistent
Dad-o-GrandDad	Moved from Human to Grandl
Dad-o-Uncle	Moved from Human to Father
Earthworm	Inconsistent
Father	Added Parent
Father-o-Daughter	Moved from Man to Parent-o-
Father-o-Son	Moved from Man to Parent-o-
GrandDad	Moved from Man to Father, GrandParent, GrandFather

Figure 3: CEL as a DIG reasoner and Protégé OWL editor

Empirical Success of CEL.

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 two right-identity rules similar to the third role axiom in our example (see Fig. 1). These axioms are 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,

	$\mathcal{O}^{\text{Go}}$	$\mathcal{O}^{\text{Galen}}$	$\mathcal{O}^{\text{Snomed}}_{\text{core}}$	$\mathcal{O}^{\text{Snomed}}$
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	$11 + 2$
concept names	20,465	2,740	53,234	379,691
role names	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 3: Benchmarks and Evaluation Results

we also consider a fragment obtained by keeping only concept definitions, 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 3, where the first row shows the numbers of primitive concept definitions, concept definitions, and general concept inclusions, respectively. The results of our experiments are summarized in the lower part of Table 3, where all classification times are shown in seconds and *unattainable* 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 $\mathcal{O}^{\text{Snomed}}_{\text{core}}$.

The empirical results for the performance of CEL described above show that it can compete with, and often outperforms, the fastest tableau-based DL systems. The robustness and scalability of tractable reasoning is visible, especially in the case of Snomed with almost four hundred thousand concepts. We view these results as evidence of empirical success of the CEL system, and also as a strong argument for the use of tractable DLs based on extensions of $\mathcal{EL}$ provided that the expresivity is sufficient in the domain of interest.

References

[1] F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*, Edinburgh, UK, 2005. Morgan-Kaufmann Publishers.

[2] F. Baader, C. Lutz, and B. Suntisrivaraporn. Efficient reasoning in $\mathcal{EL}^{+}$. In *Proceedings of the 2006 International Workshop on Description Logics (DL2006)*, CEUR-WS, 2006.

[3] 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.

[4] P. Patel-Schneider and B. Swartout. Description-logic knowledge representation system specification from the krss group of the arpa knowledge sharing effort. Technical report, DARPA Knowledge Representation System Specification (KRSS) Group of the Knowledge Sharing Initiative, 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.

[6] K.A. Spackman. Managing clinical terminology hierarchies using algorithmic calculation of subsumption: Experience with SNOMED-RT. *Journal of the American Medical Informatics Association (JAMIA)*, 2000. Fall Symposium Special Issue.

[7] The Gene Ontology Consortium. Gene Ontology: Tool for the unification of biology. *Nature Genetics*, 25:25–29, 2000.