CICLOP

François de Bertrand de Beuvron, François Rousselot
Martina Grathwohl, David Rudloff, Michael Schlick
LIIA-ENSAIS, FRANCE
beuvron@corbu01.u-strasbg.fr

Abstract

CICLOP (Customizable Inference and Concept Language for Object Processing) is a Description Logic system, which has been developed as a system for practical use. Its facilities and extensions are motivated by its applications. CICLOP offers means to organize the knowledge base in multiple disjoint terminologies and corresponding parts of the world model. Each terminology represents a model unit with a proper expressiveness and optionally the closure facility. CICLOP has been developed in Java and has proven to be easy to extend. Its reasoner uses sound and complete Tableaux-based algorithms.

Language

The CICLOP system supports the concept constructors listed in figure 1, the role constructors listed in figure 2, the TBox axioms listed in figure 3, and the ABox axioms listed in figure 4.

CICLOP manages four disjoint[1] name spaces for concepts, attributes, transitive roles, and standard roles (roles which are neither attributes nor transitives). Elements of these sets will be respectively denoted by CN, RN_S, RN_A, and RN_+ in this paper. RN will denote a role name, either standard, attribute or transitive. C and D will refer to arbitrary concept description, R and S to arbitrary role descriptions, A to an attribute description, I and J will denote individuals in the ABox. $\mathcal{R}$ denotes the field of real numbers.[2]

The semantics for CICLOP is a standard descriptive semantics. As usual, concepts refer to subsets of the domain, and roles to relations between individuals in the domain. CICLOP supports cyclic axioms : the definition of a concept CN may directly or indirectly refer to CN. Least or greatest fixpoint semantics may produce more intuitive semantics for some recursive concept definitions [1], but they are not yet available in the system.

Other Features

Applying the existing facilities (within LIIA's previous DL system C3L[2]) of Description Logic-based knowledge representation and reasoning to domains like configuration [3] or knowledge base modeling and querying [4] has demonstrated the need for extended expressivenesses as well as for other knowledge base management facilities.

In order to meet these application demands the system CICLOP provides the facility to define multiple TBoxes and ABoxes. Thus, concepts can be grouped according to their context. Each TBox has a proper expressiveness. It can be associated with several ABoxes having the same expressiveness and using the concepts of the TBox to define their individuals. Different TBoxes can be connected by means of so-called connector roles, which relate a concept belonging to a certain TBox to a concept belonging to another one.

Within this multiple-TBox/multiple-ABox framework CICLOP uses sound and complete tableaux algorithms. Expressivenesses like $\mathcal{ALC}$, $\mathcal{ALC}\boldsymbol{f}$, and $\mathcal{ALCIH}\boldsymbol{f}_{R+}$ are supported, as well as the use of concrete domains. TBoxes with the numeric or the string expressiveness can be defined. A numeric TBox enables the representation of constraints in form of equations and inequations. In a string TBox concepts can be defined by associating a set of possible and impossible strings to it.

Furthermore, in CICLOP facilities for terminology closure have been implemented. A TBox can be de-

[1] many DL systems do not require concept and role names to be disjoint. In fact, whether an name refers to a concept or a role can always be determined by the context. Nevertheless, CICLOP does not allow concepts and roles to share the same name because this may be confusing within applications.

[2] the current implementation of NumBox is in fact based on floating point numbers. CICLOP satisfiability testing algorithm is therefore lacking the soundness property within the NumBox due to the well known limitations of floating point arithmetic. For exemple, the concepts `(= 1E35)` and `(= (+ 1E35 1E-15))` will be classified as equivalent in all implementations with less than fifty significant digits.

fined to be closable, concepts to be concrete or abstract. The division of the knowledge base in multiple TBoxes and ABoxes enables the selective closure of parts of the knowledge base.

Implementation

CICLOP is implemented in pure Java (JDK 1.1.7), and runs on any operating system providing a Java Virtual Machine. CICLOP can also be executed as an applet within any Java-supporting web browser.

CICLOP incorporates optimization techniques such as lexical normalisation, dependency directed backtracking, and model caching (in the ABox). Futhermore, CICLOP uses blocking to deal with terminologies containing cycles, transitive and inverse roles [5].

CICLOP provides a graphical user interface (based on the Java AWT GUI package) for some of the most often used facilities of the system. Other functionalities may be accessed through a text based interface accepting a subset of the KRSS specification [6] as input. Applications should also be built on top of the CICLOP system using the provided java API. For a user-friendly display of classification results, CICLOP is also interfaced with the Mica [7] graphical package to produce graphical representations of concept taxonomies.

Performance

A comprehensive and coherent set of benchmarks results are available on the CICLOP web page.

Availability

CICLOP is available for research purposes via the WWW at :
`http://www-ensais.u-strasbg.fr/liia/ciclop/ciclop.htm`

Future Plans

CICLOP has been designed to be easily adapted to various application domains. Further developments of the system will follow the needs of the applied research. Planned extensions include:

- the graphical user interface should be extended to include a graphical "inspector", and "tracer". The inspector will visualize the graph structure of individuals contained in the ABox, or in the tableaux during a satisfiability test. The trace will catch internal events such as node (individual) creations, constraint expansions, clashes and so on, and give a hook for the user to inspect the current state of the solver via the inspector.
- We plan to use Desciption Logics for the management of natural disaster[8]. The creation of specialized TBoxes to represent time and space constraints is currently under study. We will also extend the system to cope with plannification and ressource management problems.
- Equality of path expression is at the moment restricted on numeric expressions, but we intend to implement feature (attribute) equality in the near future. CICLOP will be used as a unified knowledge representation formalism for syntactic, semantic, and discourse information during natural langage parsing.

References

[1] B. Nebel. Terminological cycles: Semantics and computational properties. In *Principals of Semantic Networks: Explorations in the representation of knowledge.* Morgan-Kaufmann, 1991.

[2] Thomas Kessel. *C3L: Integrating Description Logics and Frame Languages.* PhD thesis, Université Louis Pasteur, 1996.

[3] Bernard Keith, Thomas Kessel, Michael Schlick, and O. Stern. Configuring diagnostic systems in the framework of description logics. In *Proceedings of CAEPIA-95.* Alicante, Spain, 1995.

[4] François de Bertrand de Beuvron, François Rousselot, and David Rudloff. Interpretation of description logics for natural language and for databases. In *International Workshop on Description Logics (DL'97)*, 1997.

[5] R. I. Horrocks. *Optimizing Tableaux Decision Procedures for Description Logics.* PhD thesis, University of Manchester, 1997.

[6] Peter F. Patel-Schneider and Bill Swartout. Description-logic knowledge representation system specification. Technical report, AI Principles Research Department, AT&T Bell Laboratories, 1993.

[7] Software Farm, Inc., 1942 Broadway, Suite 314, Boulder CO 8 03 02, `http://www.swfm.com`. *The Mica Graphics Framework*, 1999.

[8] Martina Grathwohl, François de Bertrand de Beuvron, and François Rousselot. A new application for description logics: Disaster management. In *International Workshop on Description Logics (DL'99)*, 1999.

Expres-sivenes	concrete syntax	abstract syntax	semantics
all TBoxes	CN TOP BOTTOM (NOT C) (AND C D) (OR C D)	CN $\top$ $\bot$ $\neg C$ $C \sqcap D$ $C \sqcup D$	$CN^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ $\Delta^{\mathcal{I}}$ $\emptyset$ $\Delta^{\mathcal{I}} - C^{\mathcal{I}}$ $C^{\mathcal{I}} \cap D^{\mathcal{I}}$ $C^{\mathcal{I}} \cup D^{\mathcal{I}}$
$\mathcal{ALC}$	(SOME RN C) (ALL RN C)	$\exists R.C$ $\forall R.C$	$\{d \in \Delta^{\mathcal{I}} / R^{\mathcal{I}}(d) \cap C^{\mathcal{I}} \neq \emptyset\}$ $\{d \in \Delta^{\mathcal{I}} / R^{\mathcal{I}}(d) \subseteq C^{\mathcal{I}}\}$
StringBox	(ASET $S_1 \ldots S_n$)	$\{S_1, \ldots, S_n\}$	$\{S_1, \ldots, S_n\}$
NumBox	(SOME A (<op> (path A_1 ... A_n))) where <op> is one of $=, <, >, <=, >=$		$\{d \in dom(A^{\mathcal{I}}) / \exists c \in dom(A_n^{\mathcal{I}}) : A^{\mathcal{I}}(d)$¡op¿$A_1^{\mathcal{I}} \circ A_2^{\mathcal{I}} \circ ... \circ A_n^{\mathcal{I}}\}$

Figure 1: Concept expressions

concrete syntax	abstract syntax	semantics
RN_S	RN_S	${RN_S}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
RN_A	RN_A	${RN_A}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ $\left.\begin{array}{l} \langle d, e\rangle \in {RN_A}^{\mathcal{I}} \\ \langle d, f\rangle \in {RN_A}^{\mathcal{I}} \end{array}\right\} \Rightarrow e = f$
RN_+	RN_+	${RN_+}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ $\left.\begin{array}{l} \langle d, e\rangle \in {RN_+}^{\mathcal{I}} \\ \langle e, f\rangle \in {RN_+}^{\mathcal{I}} \end{array}\right\} \Rightarrow \langle d, f\rangle \in {RN_+}^{\mathcal{I}}$
(AND R S) (DOMAIN C) (RANGE C) (INV R)	$R \sqcap S$ R^{-1}	$R^{\mathcal{I}} \cap S^{\mathcal{I}}$ $C^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ $\Delta^{\mathcal{I}} \times C^{\mathcal{I}}$ $\{\langle d, d'\rangle \mid \langle d', d\rangle \in R^{\mathcal{I}}\}$
(COMPOSE $RN_A^1 \ldots RN_A^n$)	$RN_A^1 \circ \ldots \circ RN_A^n$	$\{\langle d_0, d_n\rangle; \langle d_0, d_1\rangle \in {RN_A^1}^{\mathcal{I}}, \ldots, \langle d_{n-1}, d_n\rangle \in {RN_A^n}^{\mathcal{I}}\}$

Figure 2: Role and attribute expressions

concrete syntax	abstract syntax	semantics
(DEFINE-PRIMITIVE-ROLE RN_S)	RN_S is a standard role	
(DEFINE-PRIMITIVE-ATTRIBUTE RN_A)	RN_A is an attribute	
(DEFINE-TRANSITIVE-ROLE RN_T)	RN_T is a transitive role	
(DEFINE-CONCEPT CN C)	$CN \doteq C$	$CN^{\mathcal{I}} = C^{\mathcal{I}}$
(DEFINE-PRIMITIVE-CONCEPT CN C)	$CN \sqsubseteq C$	$CN^{\mathcal{I}} \subseteq C^{\mathcal{I}}$
(ROLE-IMPLIES RN R)	$RN \sqsubseteq R$	$RN^{\mathcal{I}} \subseteq R^{\mathcal{I}}$
(GENERAL-CONCEPT-INCLUSION C D)	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$

Figure 3: TBox axioms

concrete syntax	abstract syntax	semantics
(INSTANCE I C)	$I \in C$	$I^{\mathcal{I}} \in C^{\mathcal{I}}$
(RELATED I J RN)	$\langle I, J\rangle \in RN$	$\langle I^{\mathcal{I}}, J^{\mathcal{I}}\rangle \in RN^{\mathcal{I}}$

Figure 4: ABox axioms