Description-Logic Knowledge Representation System Specification from the KRSS Group of the ARPA Knowledge Sharing Effort

Peter F. Patel-Schneider, co-chair
Bill Swartout, co-chair

1 November 1993

1 Overview

This is the KRSS group specification for description-logic-based KR systems. It describes the required behavior for compliant KR systems. This report is not an overview of description logics, nor is it a rationale for using description logics in knowledge representation.

The description logic in this specification is based closely on the description logic defined by researchers at DFKI [1]. However, it includes several other features, notably role closures and rules.

A knowledge base in this specification is a sequence of statements. The semantics of non-rule, non-closure statements is similar to that in the DFKI proposal. The semantics of role closures is determined by replacing them, in sequence, with the best derivable role maximum or the derivable set of fillers. Rules are treated as epistemic statements, in line with their treatment by Donini *et al* [2].

Compliant implementations are required to parse the entire language, but may replace constructs that they cannot reason about with the closest approximation that they can handle. Compliant implementations are required to be complete on a subset of the logic, selected for its easy, but non-trivial, inferences.

2 Syntax

Major parts of the syntax of knowledge bases are taken from [1]. The top-level syntactic categories in the specification are descriptions, statements, knowledge bases, and inquiries.

Throughout the specification, C, R, A, I, CI, and S, possibly subscripted, are concepts, roles (including attributes), attributes, individuals, concrete individuals, and assertions, respectively; N, CN, RN, AN, IN, AIN, DIN, XN, and GN, possibly subscripted, are names of any sort, concept names, role names (including attribute names), attribute names, individual names, anonymous individual names, distinct individual names, rule names, and group names, respectively; QQ and RR are queries and retrievals, respectively. The syntax of names, numbers, integers, and strings are the same as in LISP.

The three kinds of descriptions in the specification are concepts, roles (including attributes), and individuals. Concepts are either concept names or are formed using the operators in Table 1.[1] Roles (attributes) are either role (attribute) names or are formed using the operators in Table 2 (3). Individuals are either individual names or concrete individuals. Concrete individuals are either numbers or strings. Individual names are either distinct individual names or anonymous individual names.

Statements are formed according to Tables 4 and 5.

A knowledge base is a sequence of statements in which there is exactly one definition of every occurring concept, role, attribute, individual, and rule name. Concept, role, attribute, and individual names must be defined before their first use, so no cyclic definitions are allowed. The name spaces of concepts, roles, individuals, and rules are distinct (i.e., there may be a concept and a role with the same name). The name space of attributes is a sub-space of the name space of roles.

Comment lines, starting with a ';', are allowed in knowledge bases.

3 Semantics

The semantics of the description logic is defined in terms of interpretations and model-sets. The interpretation part of the semantics is mostly taken from [1]. The idea of (epistemic) model-sets is taken from [2].

[1] These tables include the abstract form syntax from [1], so that the logic here can be compared with the many published papers using this abstract form.

Syntax		Extension
Input	**Abstract**	
TOP	$\top$	$\Delta^{\mathcal{I}}$
BOTTOM	$\bot$	$\emptyset$
NUMBER		the numbers
INTEGER		the integers
STRING		the strings
(and $\mathsf{C_1} \ \ldots \ \mathsf{C_n}$)	$\mathsf{C_1} \sqcap \cdots \sqcap \mathsf{C_n}$	$\mathsf{C_1}^{\mathcal{I}} \cap \cdots \cap \mathsf{C_n}^{\mathcal{I}}$
(or $\mathsf{C_1} \ \ldots \ \mathsf{C_n}$)	$\mathsf{C_1} \sqcup \cdots \sqcup \mathsf{C_n}$	$\mathsf{C_1}^{\mathcal{I}} \cup \cdots \cup \mathsf{C_n}^{\mathcal{I}}$
(not C)	$\neg\mathsf{C}$	$\Delta^{\mathcal{I}} \setminus \mathsf{C}^{\mathcal{I}}$
(all R C)	$\forall\mathsf{R{:}C}$	$\{d \in \Delta_a^{\mathcal{I}} \mid \mathsf{R}^{\mathcal{I}}(d) \subseteq \mathsf{C}^{\mathcal{I}}\}$
(some R)	$\exists\mathsf{R}$	$\{d \in \Delta_a^{\mathcal{I}} \mid \mathsf{R}^{\mathcal{I}}(d) \neq \emptyset\}$
(none R)	$\uparrow \mathsf{R}$	$\{d \in \Delta_a^{\mathcal{I}} \mid \mathsf{R}^{\mathcal{I}}(d) = \emptyset\}$
(at-least n R)	$\geq \mathsf{n}\,\mathsf{R}$	$\{d \in \Delta_a^{\mathcal{I}} \mid \lvert\mathsf{R}^{\mathcal{I}}(d)\rvert \geq n\}$
(at-most n R)	$\leq \mathsf{n}\,\mathsf{R}$	$\{d \in \Delta_a^{\mathcal{I}} \mid \lvert\mathsf{R}^{\mathcal{I}}(d)\rvert \leq n\}$
(exactly n R)	$= \mathsf{n}\,\mathsf{R}$	$\{d \in \Delta_a^{\mathcal{I}} \mid \lvert\mathsf{R}^{\mathcal{I}}(d)\rvert = n\}$
(some R C)	$\exists\mathsf{R.C}$	$\{d \in \Delta_a^{\mathcal{I}} \mid \mathsf{R}^{\mathcal{I}}(d) \cap \mathsf{C}^{\mathcal{I}} \neq \emptyset\}$
(at-least n R C)	$\geq \mathsf{n}\,\mathsf{R{:}C}$	$\{d \in \Delta_a^{\mathcal{I}} \mid \lvert\mathsf{R}^{\mathcal{I}}(d) \cap \mathsf{C}^{\mathcal{I}}\rvert \geq n\}$
(at-most n R C)	$\leq \mathsf{n}\,\mathsf{R{:}C}$	$\{d \in \Delta_a^{\mathcal{I}} \mid \lvert\mathsf{R}^{\mathcal{I}}(d) \cap \mathsf{C}^{\mathcal{I}}\rvert \leq n\}$
(exactly n R C)	$= \mathsf{n}\,\mathsf{R{:}C}$	$\{d \in \Delta_a^{\mathcal{I}} \mid \lvert\mathsf{R}^{\mathcal{I}}(d) \cap \mathsf{C}^{\mathcal{I}}\rvert = n\}$
(equal $\mathsf{R_1}$ $\mathsf{R_2}$)	$\mathsf{R_1} = \mathsf{R_2} \sqcap \exists\mathsf{R_1}$	$\{d \in \Delta_a^{\mathcal{I}} \mid \mathsf{R_1}^{\mathcal{I}}(d) = \mathsf{R_2}^{\mathcal{I}}(d) \wedge \mathsf{R_1}^{\mathcal{I}}(d) \neq \emptyset\}$
(not-equal $\mathsf{R_1}$ $\mathsf{R_2}$)	$\mathsf{R_1} \neq \mathsf{R_2}$	$\{d \in \Delta_a^{\mathcal{I}} \mid \mathsf{R_1}^{\mathcal{I}}(d) \neq \mathsf{R_2}^{\mathcal{I}}(d)\}$
(subset $\mathsf{R_1}$ $\mathsf{R_2}$)	$\mathsf{R_1} \subseteq \mathsf{R_2}$	$\{d \in \Delta_a^{\mathcal{I}} \mid \mathsf{R_1}^{\mathcal{I}}(d) \subseteq \mathsf{R_2}^{\mathcal{I}}(d)\}$
(fillers R $\mathsf{I_1} \ \ldots \ \mathsf{I_n}$)	$\mathsf{R} : \mathsf{I_1} \sqcap \cdots \sqcap \mathsf{R} : \mathsf{I_n}$	$\{d \in \Delta_a^{\mathcal{I}} \mid \mathsf{R}^{\mathcal{I}}(d) \supseteq \{\mathsf{I_1}^{\mathcal{I}}, \ldots, \mathsf{I_n}^{\mathcal{I}}\}\}$
(only-fillers R $\mathsf{I_1} \ \ldots \ \mathsf{I_n}$)		$\{d \in \Delta_a^{\mathcal{I}} \mid \mathsf{R}^{\mathcal{I}}(d) = \{\mathsf{I_1}^{\mathcal{I}}, \ldots, \mathsf{I_n}^{\mathcal{I}}\}\}$
(in A C)	$\mathsf{A} : \mathsf{C}$	$\{d \in \Delta_a^{\mathcal{I}} \mid \mathsf{A}^{\mathcal{I}}(d) \in \mathsf{C}^{\mathcal{I}}\}$
(is A I)	$\mathsf{A} : \mathsf{I}$	$\{d \in \Delta_a^{\mathcal{I}} \mid \mathsf{A}^{\mathcal{I}}(d) = \mathsf{I}^{\mathcal{I}}\}$
(set $\mathsf{I_1} \ \ldots \ \mathsf{I_n}$)	$\{\mathsf{I_1}, \ldots, \mathsf{I_n}\}$	$\{\mathsf{I_1}^{\mathcal{I}}, \ldots, \mathsf{I_n}^{\mathcal{I}}\}$
(minimum Cl)		$\{d \in \Delta_c^{\mathcal{I}} :\mid d \geq \mathsf{Cl}^{\mathcal{I}}\}$
(maximum Cl)		$\{d \in \Delta_c^{\mathcal{I}} :\mid d \leq \mathsf{Cl}^{\mathcal{I}}\}$
(satisfies ...)		see text

Table 1: Concept Syntax and Semantics

Syntax		Extension
Input	**Abstract**	
top	$\top$	$\Delta_a^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
bottom	$\bot$	$\emptyset$
identity	id	$\{(d,d) \mid d \in \Delta_a^{\mathcal{I}}\}$
(and $\mathsf{R_1}$ ... $\mathsf{R_n}$)	$\mathsf{R_1} \sqcap \cdots \sqcap \mathsf{R_n}$	$\mathsf{R_1}^{\mathcal{I}} \cap \cdots \cap \mathsf{R_n}^{\mathcal{I}}$
(or $\mathsf{R_1}$ ... $\mathsf{R_n}$)	$\mathsf{R_1} \sqcup \cdots \sqcup \mathsf{R_n}$	$\mathsf{R_1}^{\mathcal{I}} \cup \cdots \cup \mathsf{R_n}^{\mathcal{I}}$
(not R)	$\neg\mathsf{R}$	$(\Delta_a^{\mathcal{I}} \times \Delta^{\mathcal{I}}) \setminus \mathsf{R}^{\mathcal{I}}$
(inverse R)	R^{-1}	$(\mathsf{R}^{\mathcal{I}})^{-1} \cap (\Delta_a^{\mathcal{I}} \times \Delta^{\mathcal{I}})$
(restrict R C)	$\mathsf{R} \vert \mathsf{C}$	$\mathsf{R}^{\mathcal{I}} \cap (\Delta_a^{\mathcal{I}} \times \mathsf{C}^{\mathcal{I}})$
(compose $\mathsf{R_1}$ ... $\mathsf{R_n}$)	$\mathsf{R_1} \circ \cdots \circ \mathsf{R_n}$	$\mathsf{R_1}^{\mathcal{I}} \circ \cdots \circ \mathsf{R_n}^{\mathcal{I}}$
(range C)		$\Delta_a^{\mathcal{I}} \times \mathsf{C}^{\mathcal{I}}$
(domain C)		$(\mathsf{C}^{\mathcal{I}} \cap \Delta_a^{\mathcal{I}}) \times \Delta^{\mathcal{I}}$
(domain-range $\mathsf{C_1}$ $\mathsf{C_2}$)	$\mathsf{C_1} \times \mathsf{C_2}$	$(\mathsf{C_1}^{\mathcal{I}} \cap \Delta_a^{\mathcal{I}}) \times \mathsf{C_2}^{\mathcal{I}}$
(transitive-closure R)	R^{+}	$\bigcup_{n \geq 1} (\mathsf{R}^{\mathcal{I}})^n$
(transitive-reflexive-closure R)	R^{*}	$\{(d,d) \mid d \in \Delta_a^{\mathcal{I}}\} \cup \bigcup_{n \geq 1} (\mathsf{R}^{\mathcal{I}})^n$
(satisfies ...)		see text

Table 2: Role Syntax and Semantics

Syntax		Extension
Input	**Abstract**	
bottom	$\bot$	$\emptyset$
identity	id	$\{(d,d) \mid d \in \Delta_a^{\mathcal{I}}\}$
(and $\mathsf{A_1}$ $\mathsf{R_2}$ ... $\mathsf{R_n}$)	$\mathsf{A_1} \sqcap \mathsf{R_2} \sqcap \cdots \sqcap \mathsf{R_n}$	$\mathsf{A_1}^{\mathcal{I}} \cap \mathsf{R_2}^{\mathcal{I}} \cap \cdots \cap \mathsf{R_n}^{\mathcal{I}}$
(restrict A C)	$\mathsf{A} \vert \mathsf{C}$	$\mathsf{A}^{\mathcal{I}} \cap (\Delta_a^{\mathcal{I}} \times \mathsf{C}^{\mathcal{I}})$
(compose $\mathsf{A_1}$ ... $\mathsf{A_n}$)	$\mathsf{A_1} \circ \cdots \circ \mathsf{A_n}$	$\mathsf{A_1}^{\mathcal{I}} \circ \cdots \circ \mathsf{A_n}^{\mathcal{I}}$

Table 3: Attribute Syntax and Semantics

Syntax		Semantics
Input	**Abstract**	
(define-concept CN C)	$\mathsf{CN} \doteq \mathsf{C}$	$\mathsf{CN}^{\mathcal{I}} = \mathsf{C}^{\mathcal{I}}$
(define-primitive-concept CN C)	$\mathsf{CN} \sqsubseteq \mathsf{C}$	$\mathsf{CN}^{\mathcal{I}} \subseteq \mathsf{C}^{\mathcal{I}}$
(define-disjoint-primitive-concept CN (GN_1 ... GN_n) C)		see text
(define-role RN R)	$\mathsf{RN} \doteq \mathsf{R}$	$\mathsf{RN}^{\mathcal{I}} = \mathsf{R}^{\mathcal{I}}$
(define-primitive-role RN R)	$\mathsf{RN} \sqsubseteq \mathsf{R}$	$\mathsf{RN}^{\mathcal{I}} \subseteq \mathsf{R}^{\mathcal{I}}$
(define-attribute AN A)	$\mathsf{AN} \doteq \mathsf{A}$	$\mathsf{AN}^{\mathcal{I}} = \mathsf{A}^{\mathcal{I}}$
(define-primitive-attribute AN R)	$\mathsf{AN} \sqsubseteq \mathsf{R}$	$\mathsf{AN}^{\mathcal{I}} \subseteq \mathsf{R}^{\mathcal{I}}$
(define-distinct-individual DIN)		see text
(define-anonymous-individual AIN)		see text
(define-rule XN CN C)		see text
(state S)		$\mathsf{S}^{\mathcal{I}}$
(close-role IN R)		see text
(close-role-fillers IN R)		see text

Table 4: Statement Syntax and Semantics

Syntax		Semantics
Input	**Abstract**	
(and S_1 ... S_n)		${\mathsf{S}_1}^{\mathcal{I}} \wedge \cdots \wedge {\mathsf{S}_\mathsf{n}}^{\mathcal{I}}$
(or S_1 ... S_n)		${\mathsf{S}_1}^{\mathcal{I}} \vee \cdots \vee {\mathsf{S}_\mathsf{n}}^{\mathcal{I}}$
(not S)		$\neg \mathsf{S}^{\mathcal{I}}$
(instance IN C)	$\mathsf{IN} \in \mathsf{C}$	$\mathsf{IN}^{\mathcal{I}} \in \mathsf{C}^{\mathcal{I}}$
(related IN I R)	$\langle \mathsf{IN}, \mathsf{I} \rangle \in \mathsf{R}$	$\langle \mathsf{IN}^{\mathcal{I}}, \mathsf{I}^{\mathcal{I}} \rangle \in \mathsf{R}^{\mathcal{I}}$
(equal IN_1 IN_2)		${\mathsf{IN}_1}^{\mathcal{I}} = {\mathsf{IN}_2}^{\mathcal{I}}$

Table 5: Assertion Syntax and Semantics

Semantics are defined directly only for simple knowledge bases. A simple knowledge base is a knowledge base without any role or role fillers closure statements or disjoint primitive definitions. Simple knowledge bases can also contain disjointness statements of the form (disjoint CN_1 CN_2) where the concept names have been primitively defined.

A non-simple knowledge base is transformed into a simple knowledge base by modifying, in order, each role or role filler closure or disjoint primitive definition as follows:

1. A role closure, (close-role IN R), is replaced by (instance IN (at-most n R)), where n is the largest integer such that (instance IN (at-least n R)) follows from the simplified version of the portion of the knowledge base before the role closure, provided that there is such an n. Otherwise the role closure is ignored.

2. A role fillers closure, (close-role-fillers IN R), is replaced by (instance IN (only-fillers R I_1 ... I_n)), where the I_i are the individuals such that (instance IN (fillers R I_i)) follows from the simplified version of the portion of the knowledge base before the role fillers closure.

3. A disjoint primitive definition, (define-disjoint-primitive-concept CN (GN_1 ... GN_n) C), is changed to (define-primitive-concept CN C), (disjoint CN CN_1), ..., (disjoint CN CN_m), where the CN_i are the disjoint primitive concepts in the portion of the knowledge base before this definition that have any of the GN_j in their definition.

An interpretation, $\mathcal{I}$, consists of a domain, $\Delta^\mathcal{I}$, and a mapping, $\cdot^\mathcal{I}$, from concept names, role names, attribute names, and individual names to their extensions in the interpretation. The interpretation function is extended to all concepts, roles, attributes, individuals, and assertions as defined below.

All domains contain the rationals and strings over some alphabet of size at least 2, this is called the concrete part of the domain, $\Delta^\mathcal{I}_c$. The rest of the domain, $\Delta^\mathcal{I}_a$, is called the abstract part of the domain.

The extension of concepts are subsets of $\Delta^\mathcal{I}$. The extension of roles are set-valued functions from $\Delta^\mathcal{I}_a$ to $\Delta^\mathcal{I}$. The extensions of attributes are single-valued, partial functions from $\Delta^\mathcal{I}_a$ to $\Delta^\mathcal{I}$. (The extension of roles and attributes will sometimes also be treated as the equivalent subset of $\Delta^\mathcal{I}_a \times$

$\Delta^{\mathcal{I}}$. The extension of attributes are also sometimes treated as set-valued functions.) The extensions of individual names are elements of the abstract part of the domain. The extension of a distinct individual name is different from the extension of all other distinct individual names. The extension of a number or a string is the appropriate rational or string. The extension of assertions is either true or false.

The extension of the concept-, role-, and attribute-forming operators is as in the DFKI proposal [1], extended in the obvious way. See Tables 1, 2, and 3 for details. The extension of the satisfies constructs is unconstrained (within $\Delta^{\mathcal{I}}$ or $\Delta_a^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, of course).

A non-empty set of interpretations is a model-set for a simple knowledge base if each statement in the simple knowledge base is is true in the set. A simple knowledge base (any knowledge base) is inconsistent if it (its simple version) has no model-sets.

A non-rule, non-closure statement is true in a non-empty set of interpretations if it is true in each interpretation in the set. Conditions for the truth of several types of statements are given in Table 4. The statement (disjoint CN_1 CN_2) is true in an interpretation if the extensions of CN_1 and CN_2 are disjoint. Definitions of individuals only serve to distinguish anonymous and distinct individual names.

A non-empty set of interpretations makes (define-rule N C_1 C_2) true for a simple knowledge base if for each individual name, IN, in the simple knowledge base, if $\mathsf{IN}^{\mathcal{I}} \in \mathsf{C}_1{}^{\mathcal{I}}$ in each interpretation, $\mathcal{I}$, in the set, then $\mathsf{IN}^{\mathcal{I}} \in \mathsf{C}_2{}^{\mathcal{I}}$ in each interpretation in the set.

A subsumption relationship, $\mathsf{C}_1 \Longrightarrow \mathsf{C}_2$ ($\mathsf{R}_1 \Longrightarrow \mathsf{R}_2$), follows from a knowledge base if $\mathsf{C}_1{}^{\mathcal{I}} \subseteq \mathsf{C}_2{}^{\mathcal{I}}$ ($\mathsf{R}_1{}^{\mathcal{I}} \subseteq \mathsf{R}_2{}^{\mathcal{I}}$) in each interpretation of each model-set of the simple knowledge base version of the concept, role, attribute, disjointness, and individual definitions (i.e., no rules or assertions or closures) in the original knowledge base. An instance relationship, $\mathsf{IN} \in \mathsf{C}$, follows from a knowledge base if $\mathsf{IN}^{\mathcal{I}} \in \mathsf{C}^{\mathcal{I}}$; a role relationship, $\langle \mathsf{IN}, \mathsf{I} \rangle \in \mathsf{R}$, follows if $\langle \mathsf{IN}^{\mathcal{I}}, \mathsf{I}^{\mathcal{I}} \rangle \in \mathsf{R}^{\mathcal{I}}$; and an individual equality, $\mathsf{IN}_1 = \mathsf{IN}_2$, follows if $\mathsf{IN}_1{}^{\mathcal{I}} = \mathsf{IN}_2{}^{\mathcal{I}}$; all in each interpretation of each model-set of the simple knowledge base version of the original knowledge base.

Query	**Meaning**
(concept-subsumes? C_1 C_2)	$\mathsf{C}_1 \Longrightarrow \mathsf{C}_2$
(role-subsumes? R_1 R_2)	$\mathsf{R}_1 \Longrightarrow \mathsf{R}_2$
(individual-instance? IN C)	$\mathsf{IN} \in \mathsf{C}$
(individual-related? IN I R)	$\langle \mathsf{IN}, \mathsf{I} \rangle \in \mathsf{R}$
(individual-equal? IN_1 IN_2)	$\mathsf{IN}_1 = \mathsf{IN}_2$
(individual-not-equal? IN_1 IN_2)	$\neg(\mathsf{IN}_1 = \mathsf{IN}_2)$

Table 6: Query Syntax and Semantics

(concept-descendants C)
(concept-offspring C)
(concept-ancestors C)
(concept-parents C)
(concept-instances C)
(concept-direct-instances C)
(role-descendants R)
(role-offspring R)
(role-ancestors R)
(role-parents R)
(individual-types IN)
(individual-direct-types IN)
(individual-fillers IN R)

Table 7: Retrieval Syntax

(validate-true QQ)
(validate-not-true QQ)
(validate-set RR N_1 $\ldots$ N_n)

Table 8: Validation Syntax

4 Queries, Retrievals, and Validations

The input language of the specification also contains inquiries about the knowledge base that is being constructed. The input language is thus a sequence of statements, queries (see Table 6), retrievals (see Table 7), and validations (see Table 8).

A query follows from a knowledge base, and returns something other than the symbol NIL, if its meaning in Table 6 follows from the knowledge base before the query. Otherwise, the query is false, and returns the symbol NIL. However, if a concept or role in a subsumption query is incoherent[2] then the result of the query is unspecified. Note that determining if a query follows from a knowledge base is not decidable, nor even recursively enumerable. Therefore, no implementation can possibly be complete.

Retrievals return sets (as lists) of concept names, role names, individual names, and concrete individuals.

The retrieval (concept-descendants C) ((concept-ancestors C)) returns the set of concept names, CN, that are defined in the KB, and for which $\mathsf{CN} \Longrightarrow \mathsf{C}$ ($\mathsf{C} \Longrightarrow \mathsf{CN}$) follows from the KB but $\mathsf{C} \Longrightarrow \mathsf{CN}$ ($\mathsf{CN} \Longrightarrow \mathsf{C}$) does not. The retrieval (concept-offspring C) ((concept-parents C)) returns the set of maximal (minimal), under the subsumption relationship in the KB, elements of the result of (concept-descendants C) ((concept-ancestors C)). The retrieval (concept-instances C) returns the set of individual names, IN, that are defined in the KB, and for which $\mathsf{IN} \in \mathsf{C}$ follows from the KB. The retrieval (concept-direct-instances C) returns the subset of the result of (concept-instances C) that are not instances of any member of the result of (concept-descendants C).

Role retrievals are defined analogously.

The retrieval (individual-types IN) returns the set of concept names, CN, that are defined in the KB and for which $\mathsf{IN} \in \mathsf{CN}$ follows from the KB. The retrieval (individual-direct-types IN) returns the set of minimal, under the subsumption relationship in the KB, elements of the result of (individual-types IN). The retrieval (individual-fillers IN R) returns the set of individual names, IN_1, that are defined in the KB and for which $\langle \mathsf{IN}, \mathsf{IN}_1 \rangle \in \mathsf{R}$ follows from the KB, unioned with the set of concrete individuals, I_c, for which $\langle \mathsf{IN}, \mathsf{I}_\mathsf{c} \rangle \in \mathsf{R}$ follows from the KB, unless this latter set is infinite, in which case the retrieval is undefined.

[2]An incoherent concept or role has empty extension in all model sets.

Validations simply check to see if the query or retrieval returned the expected result. If so, the validation is true; otherwise, the validation is false, and may print a message.

5 Compliance

Conforming implementations must parse the entire syntax. If a conforming implementation cannot internally represent a particular statement, it must replace it some syntactically-close representable statement and issue a warning.

For simple knowledge bases, conforming implementations must be sound for queries. Retrievals must be correct with respect to a definition that replaces semantic entailment by the (incomplete) subsumption query in the conforming implementation.

Core knowledge bases are defined as knowledge bases containing only the following sorts of statements:

(define-concept CN cC)
(define-primitive-concept CN cC)
(define-disjoint-primitive-concept CN (GN_1 ... GN_n) cC)
(define-primitive-attribute AN top)
(define-distinct-individual DIN)
(define-rule XN CN cC)
(state (instance IN cC))
(state (related IN I cR))

Here cC is a core concept, which is either a concept name, TOP, BOTTOM, NUMBER, INTEGER, or STRING or formed as follows:

(and cC_1 ... cC_n)
(all cR cC)
(some cR)
(none cR)
(eq (compose AN_{11} ... $\mathsf{AN}_{1\mathsf{n}}$) (compose AN_{21} ... $\mathsf{AN}_{2\mathsf{m}}$))
(minimum CI)
(maximum CI)

Also, cR is either a role name or an attribute name. (In the core cR has to be an attribute, but the number restriction extension allows multi-valued roles also.)

The core syntax was selected to be easy to perform inferences on. Subsumption inferences are polynomial. Other inferences are similarly easy. The inference for rules is that in the presence of the rule (define-rule XN CN C), if (instance IN CN) can be derived, then (instance IN C) can also.

Conforming implementations must accept all consistent core knowledge bases that also have no incoherent concept definitions. The actions of conforming implementations on inconsistent knowledge bases or knowledge bases with incoherent concept definitions are unspecified. It is recommended that an error be signaled or the knowledge base be reverted to a consistent state (or both) as soon as an inconsistency is detected. If an incoherent concept (or role) definition is detected, an error may be signalled and the knowledge base be reverted, or a warning produced.

Conforming implementations must perform complete reasoning on core knowledge bases. Note that because subsumption is unspecified for incoherent concepts or roles conformining implementations do not have to perform complete subsumption on such concepts or roles.

5.1 Extensions

If an implementation is complete with respect to subsumption of incoherent concepts then it satisfies the "incoherent-subsumption" extension.

If an implementation is complete on the core plus the statements:

(define-primitive-role RN top)
(close-role IN cR)
(close-role-fillers IN cR)

with concepts extended to include (at-least n RN), (at-most n RN), and (exactly n RN) then it satisfies the "number-restriction" extension.

References

[1] Franz Baader, Hans-Jürgen Bürckert, Jochen Heinsohn, Bernhard Hollunder, Jürgen Müller, Bernhard Nebel, Werner Nutt, and Hans-Jürgen Profitlich. Terminological knowledge representation: A proposal for a terminological logic. A DFKI note, June 1991.

[2] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, Andrea Schaerf, and Werner Nutt. Adding epistemic operators to concept languages.

In *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning*. Morgan Kaufmann, October 1992.

A Test Suite

This is a simple test suite for compliance with the specification.

```
;; The following macro calls test the parsing.
(define-concept c1 top)
(define-concept c c1)
(define-role relative top)
(define-primitive-role sibling (and top relative))
(define-primitive-role brother (or (and sibling)))
(define-primitive-role sister (and (or sibling)))
(define-attribute age top)
(define-primitive-attribute name (or relative))
;; defaults to sister as parent:
(define-primitive-role sister2 (and sister brother))
;; Defaults to simple role:
(define-primitive-role relative3 (or sister brother))
(define-primitive-role r1 top)
(define-primitive-role r2 top)
(define-primitive-role r3 top)
(define-primitive-role r top)
(define-primitive-attribute a1 top)
(define-primitive-attribute a2 top)
(define-primitive-attribute a top)

(define-disjoint-primitive-concept animal type top)
(define-disjoint-primitive-concept animal3 type top)
(define-primitive-concept animal2 top)
(define-concept c2a bottom)
(define-concept c2 (all (and relative) c1))
(define-concept c3 (all age (or number)))
(define-concept c4 (all age (and integer)))
(define-concept c5 (all name string))
```

```
(define-concept c6 c5)

(define-concept c7 (and (or c1 c2)))
(define-concept c8 (or c1))
(define-concept c9 (or c1 c2))
(define-concept c10 (or (and (some r1) (some r2))))
(define-concept c11 (not top))
(define-concept c12 (not c1))
(define-concept c13 (all r top))
(define-concept c14 (all (not r) top))
(define-concept c15 (some r1))
(define-concept c16 (some (not r1)))
(define-concept c17 (some r top))
(define-concept c18 (some (not r) top))
(define-concept c19 (at-least 2 r1))
(define-concept c20 (at-least 2 (not r1)))
(define-concept c21 (at-least 2 r1 C1))
(define-concept c22 (at-least 2 (not r1) C1))
(define-concept c23 (at-most 2 r1))
(define-concept c24 (at-most 2 (not r1)))
(define-concept c25 (at-most 2 r1 C1))
(define-concept c26 (at-most 2 (not r1) C1))
(define-concept c27 (exactly 2 r1))
(define-concept c28 (exactly 2 (not r1)))
(define-concept c29 (exactly 2 r1 C1))
(define-concept c30 (exactly 2 (not r1) C1))
(define-concept c31 (some r))
(define-concept c32 (some (not r)))
(define-concept c33 (none r))
(define-concept c34 (none (not r)))

(define-concept c35 (equal (compose a1) (compose a2 a1)))
(define-concept c36 (equal (compose a1) (compose a2 r3)))
(define-concept c37 (equal a1 a2))
(define-concept c38 (equal (range C) a2))
(define-concept c39 (not-equal a1 a2))
(define-concept c40 (subset a1 a2))
(define-concept c41 (fills r i1 i2))
```

```
(define-concept c42 (fills r 5 6))
(define-concept c43 (fills (and (or r)) 5 6))
(define-concept c44 (fills (range C) i1 i2))
(define-concept c45 (fills-only r i1 i2))
(define-concept c46 (fills-only r 5 6))

(define-concept c47 (fills-only (range C) i1 i2))
(define-concept c48 (in a1 C1))
(define-concept c49 (in (and A) C))
(define-concept c50 (in (not A) C))
(define-concept c51 (in A (some r)))
(define-concept c52 (is A I))
(define-concept c53 (is (and A) I))
(define-concept c54 (is (not A) I))
(define-concept c55 (set i1 i2 i3))
(define-concept c55a (set 4 5))
(define-concept c56 (minimum 5))
(define-concept c57 (maximum 5))
(define-concept c58 (satisfies integerp))

(define-distinct-individual mary)
(define-anonymous-individual unnamed-ind)
(define-rule rule1 top (at-least 1 r1))
(close-role-fillers mary (and r2))
(close-role-fillers mary r)
(close-role-fillers mary (or r2 r))
(define-distinct-individual fred)
(close-role fred r2)
(define-primitive-concept athlete top)
(state (instance mary athlete))
(state (not (instance mary athlete)))
(state (equal mary joe))
(state (or (instance mary athlete)))
(state (or (instance mary athlete) (equal mary joe)))
(define-distinct-individual joe)
(state (related mary joe brother))
(state (related mary 35 age))
(state (and (instance mary athlete)))
```

```
(state (and (instance mary athlete) (related mary 35 age)))
(state (and (instance mary athlete) (equal mary joe)))

;; role syntax
(define-concept c59 (all top top))
(define-concept c60 (all bottom top))
(define-concept c61 (all identity top))
(define-concept c62 (all (inverse parent) top))
(define-concept c63 (all (restrict r c) top))
(define-concept c64 (all (compose r1 r2) top))
(define-concept c65 (all (range c1) top))
(define-concept c66 (all (domain c1) top))
(define-concept c67 (all (domain-range c1 c2) top))
(define-concept c68 (all (transitive-closure r1) top))
(define-concept c69 (all (transitive-reflexive-closure r1) top))
(define-concept c70 (all (satisfies foo r1) top))

;; Test the queries
(define-primitive-role r1 top)
(define-primitive-role r2 top)
(define-primitive-role r2-child r2)
(define-primitive-role r2-child2 r2)
(define-primitive-role r2-descendant r2-child)
(define-primitive-role sibling top)
(define-primitive-attribute age top)
(define-primitive-concept athlete top)
(define-primitive-concept healthy top)
(define-concept healthy-athlete (and athlete healthy))
(define-primitive-concept very-healthy-athlete healthy-athlete)
(define-distinct-individual mary)
(state (and (related mary 35 age)
            (related mary herbie sibling)
            (related mary joe sibling)
            (related mary martin sibling)))

(define-distinct-individual joe)
(state (and (instance joe athlete)
            (related mary joe r1)))
```

```
(close-role-fillers mary r1)
(define-distinct-individual joe-healthy)
(state (and (instance joe-healthy healthy-athlete)))

(concept-subsumes? athlete healthy-athlete)
(concept-subsumes? healthy-athlete athlete)
(concept-subsumes?
 (and (at-least 1 r2) (all r1 athlete))
 (and (at-least 2 r2-child) (all r1 healthy-athlete)))
;; one containing an error:
(concept-subsumes?
 (and (at-least 1 top) (all r1 athlete))
 (and (at-least 2 r2-child) (all r1 healthy-athlete)))

(role-subsumes? r1 r2-child) ;nil
(role-subsumes? r2 r2-child)
(role-subsumes? r2 r2)
(role-subsumes? top r2)

(individual-instance? mary (all r1 athlete))
(individual-instance? joe athlete)
(individual-instance? joe (all r1 athlete)) ;nil
(individual-instance? joel athlete) ;error

(individual-related? mary joe r1)
(individual-related? mary joe r2)
(individual-related? mary joel r2)

(individual-equal? mary mary)
(individual-equal? mary joe)
(individual-equal? mary mary2)
(individual-equal? mary mary5)

(individual-not-equal? mary mary)
(individual-not-equal? mary joe)
(individual-not-equal? mary mary2)
(individual-not-equal? mary mary5)
```

```
;; Retrieval and Validation Macros

(concept-descendants athlete)
(concept-descendants bottom)
(concept-descendants (and athlete healthy))
(concept-descendants (some r1))

(concept-offspring athlete)
(concept-offspring bottom)
(concept-offspring (and athlete healthy))
(concept-offspring (some r1))

(concept-ancestors athlete)
(concept-ancestors bottom)
(concept-ancestors (and athlete healthy))
(concept-ancestors (some r1))

(concept-parents athlete)
(concept-parents bottom)
(concept-parents (and athlete healthy))
(concept-parents (some r1))

(concept-instances athlete)
(concept-instances bottom)
(concept-instances healthy-athlete)
(concept-instances (and athlete healthy))
(concept-instances (some r1))

(concept-direct-instances athlete)
(concept-direct-instances bottom)
(concept-direct-instances healthy-athlete)
(concept-direct-instances (and athlete healthy))
(concept-direct-instances (some r1))

(role-descendants r2)
(role-descendants r2-child)
(role-descendants r2-descendant)
(role-descendants (and r2-descendant r1))
```

```
(role-offspring r2)
(role-offspring r2-child)
(role-offspring r2-descendant)
(role-offspring (and r2-descendant r1))

(role-ancestors r2)
(role-ancestors r2-child)
(role-ancestors r2-descendant)
(role-ancestors (and r2-descendant r1))

(role-parents r2)
(role-parents r2-child)
(role-parents r2-descendant)
(role-parents (and r2-descendant r1))

(individual-types joe)
(individual-types joe-healthy)
(individual-types mary)
(individual-types joel)

(individual-direct-types joe)
(individual-direct-types joe-healthy)
(individual-direct-types mary)
(individual-direct-types joel)

(individual-fillers mary r1)
(individual-fillers mary age)
(individual-fillers mary r2)
(individual-fillers joel r2)
(individual-fillers mary r2l)
(individual-fillers mary (not r2l))

(validate-true (role-subsumes? r1 r2-child)) ;nil
(validate-true (role-subsumes? r2 r2-child))

(validate-not-true (role-subsumes? r1 r2-child)) ;nil
(validate-not-true (role-subsumes? r2 r2-child))
```

```
(validate-set (individual-fillers mary sibling) joe martin herbie)
(validate-set (individual-fillers mary sibling) martin herbie)
```