Working Version (Draft): Description Logic Specification from the KRSS Effort

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

30 June 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 the specification 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, 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 (KB) 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. The first line of a knowledge base should be a comment containing identifying information. The format of such comments is the same as mode-lines, i.e., it starts and ends with '-*-' and contains a sequence of token definitions, where a token definition is a token name, a ':', a balanced s-expr, and a ';'. It is recommended that the token Mode be defined as KRSS-Description-Logic-KB,

[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 C_1 ... C_n)	$C_1 \sqcap \cdots \sqcap C_n$	${C_1}^{\mathcal{I}} \cap \cdots \cap {C_n}^{\mathcal{I}}$
(or C_1 ... C_n)	$C_1 \sqcup \cdots \sqcup C_n$	${C_1}^{\mathcal{I}} \cup \cdots \cup {C_n}^{\mathcal{I}}$
(not C)	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
(all R C)	$\forall R{:}C$	$\{d \in \Delta_a^{\mathcal{I}} \mid R^{\mathcal{I}}(d) \subseteq C^{\mathcal{I}}\}$
(some R)	$\exists R$	$\{d \in \Delta_a^{\mathcal{I}} \mid R^{\mathcal{I}}(d) \neq \emptyset\}$
(none R)	$\uparrow R$	$\{d \in \Delta_a^{\mathcal{I}} \mid R^{\mathcal{I}}(d) = \emptyset\}$
(at-least n R)	$\geq n\, R$	$\{d \in \Delta_a^{\mathcal{I}} \mid \lvert R^{\mathcal{I}}(d)\rvert \geq n\}$
(at-most n R)	$\leq n\, R$	$\{d \in \Delta_a^{\mathcal{I}} \mid \lvert R^{\mathcal{I}}(d)\rvert \leq n\}$
(exactly n R)	$= n\, R$	$\{d \in \Delta_a^{\mathcal{I}} \mid \lvert R^{\mathcal{I}}(d)\rvert = n\}$
(some R C)	$\exists R.C$	$\{d \in \Delta_a^{\mathcal{I}} \mid R^{\mathcal{I}}(d) \cap C^{\mathcal{I}} \neq \emptyset\}$
(at-least n R C)	$\geq n\, R{:}C$	$\{d \in \Delta_a^{\mathcal{I}} \mid \lvert R^{\mathcal{I}}(d) \cap C^{\mathcal{I}}\rvert \geq n\}$
(at-most n R C)	$\leq n\, R{:}C$	$\{d \in \Delta_a^{\mathcal{I}} \mid \lvert R^{\mathcal{I}}(d) \cap C^{\mathcal{I}}\rvert \leq n\}$
(exactly n R C)	$= n\, R{:}C$	$\{d \in \Delta_a^{\mathcal{I}} \mid \lvert R^{\mathcal{I}}(d) \cap C^{\mathcal{I}}\rvert = n\}$
(equal R_1 R_2)	$R_1 = R_2$	$\{d \in \Delta_a^{\mathcal{I}} \mid {R_1}^{\mathcal{I}}(d) = {R_2}^{\mathcal{I}}(d)\}$
(not-equal R_1 R_2)	$R_1 \neq R_2$	$\{d \in \Delta_a^{\mathcal{I}} \mid {R_1}^{\mathcal{I}}(d) \neq {R_2}^{\mathcal{I}}(d)\}$
(subset R_1 R_2)	$R_1 \subseteq R_2$	$\{d \in \Delta_a^{\mathcal{I}} \mid {R_1}^{\mathcal{I}}(d) \subseteq {R_2}^{\mathcal{I}}(d)\}$
(fillers R I_1 ... I_n)	$R : I_1 \sqcap \cdots \sqcap R : I_n$	$\{d \in \Delta_a^{\mathcal{I}} \mid R^{\mathcal{I}}(d) \supseteq \{{I_1}^{\mathcal{I}}, \ldots, {I_n}^{\mathcal{I}}\}\}$
(only-fillers R I_1 ... I_n)		$\{d \in \Delta_a^{\mathcal{I}} \mid R^{\mathcal{I}}(d) = \{{I_1}^{\mathcal{I}}, \ldots, {I_n}^{\mathcal{I}}\}\}$
(in A C)	$A : C$	$\{d \in \Delta_a^{\mathcal{I}} \mid A^{\mathcal{I}}(d) \in C^{\mathcal{I}}\}$
(is A I)	$A : I$	$\{d \in \Delta_a^{\mathcal{I}} \mid A^{\mathcal{I}}(d) = I^{\mathcal{I}}\}$
(set I_1 ... I_n)	$\{I_1, \ldots, I_n\}$	$\{{I_1}^{\mathcal{I}}, \ldots, {I_n}^{\mathcal{I}}\}$
(minimum CI)		$\{d \in \Delta_c^{\mathcal{I}} :\mid d \geq CI^{\mathcal{I}}\}$
(maximum CI)		$\{d \in \Delta_c^{\mathcal{I}} :\mid d \leq CI^{\mathcal{I}}\}$
(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 R_1 ... R_n)	$R_1 \sqcap \cdots \sqcap R_n$	${R_1}^{\mathcal{I}} \cap \cdots \cap {R_n}^{\mathcal{I}}$
(or R_1 ... R_n)	$R_1 \sqcup \cdots \sqcup R_n$	${R_1}^{\mathcal{I}} \cup \cdots \cup {R_n}^{\mathcal{I}}$
(not R)	$\neg R$	$(\Delta_a^{\mathcal{I}} \times \Delta^{\mathcal{I}}) \setminus R^{\mathcal{I}}$
(inverse R)	R^{-1}	$(R^{\mathcal{I}})^{-1} \cap (\Delta_a^{\mathcal{I}} \times \Delta^{\mathcal{I}})$
(restrict R C)	$R \mid C$	$R^{\mathcal{I}} \cap (\Delta_a^{\mathcal{I}} \times C^{\mathcal{I}})$
(compose R_1 ... R_n)	$R_1 \circ \cdots \circ R_n$	${R_1}^{\mathcal{I}} \circ \cdots \circ {R_n}^{\mathcal{I}}$
(range C)		$\Delta_a^{\mathcal{I}} \times C^{\mathcal{I}}$
(domain C)		$(C^{\mathcal{I}} \cap \Delta_a^{\mathcal{I}}) \times \Delta^{\mathcal{I}}$
(domain-range C_1 C_2)	$C_1 \times C_2$	$({C_1}^{\mathcal{I}} \cap \Delta_a^{\mathcal{I}}) \times {C_2}^{\mathcal{I}}$
(transitive-closure R)	R^+	$\bigcup_{n \geq 1} (R^{\mathcal{I}})^n$
(transitive-reflexive-closure R)	R^*	$\{(d,d) \mid d \in \Delta_a^{\mathcal{I}}\} \cup \bigcup_{n \geq 1} (R^{\mathcal{I}})^n$
(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 A_1 R_2 ... R_n)	$A_1 \sqcap R_2 \sqcap \cdots \sqcap R_n$	${A_1}^{\mathcal{I}} \cap {R_2}^{\mathcal{I}} \cap \cdots \cap {R_n}^{\mathcal{I}}$
(restrict A C)	$A \mid C$	$A^{\mathcal{I}} \cap (\Delta_a^{\mathcal{I}} \times C^{\mathcal{I}})$
(compose A_1 ... A_n)	$A_1 \circ \cdots \circ A_n$	${A_1}^{\mathcal{I}} \circ \cdots \circ {A_n}^{\mathcal{I}}$

Table 3: Attribute Syntax and Semantics

Syntax		Semantics
Input	**Abstract**	
(define-concept CN C)	$CN \doteq C$	$CN^{\mathcal{I}} = C^{\mathcal{I}}$
(define-primitive-concept CN C)	$CN \sqsubseteq C$	$CN^{\mathcal{I}} \subseteq C^{\mathcal{I}}$
(define-disjoint-primitive-concept CN (GN$_1$ ... GN$_n$) C)		see text
(define-role RN R)	$RN \doteq R$	$RN^{\mathcal{I}} = R^{\mathcal{I}}$
(define-primitive-role RN R)	$RN \sqsubseteq R$	$RN^{\mathcal{I}} \subseteq R^{\mathcal{I}}$
(define-attribute AN A)	$AN \doteq A$	$AN^{\mathcal{I}} = A^{\mathcal{I}}$
(define-primitive-attribute AN R)	$AN \sqsubseteq R$	$AN^{\mathcal{I}} \subseteq R^{\mathcal{I}}$
(define-distinct-individual DIN)		see text
(define-anonymous-individual AIN)		see text
(define-rule XN CN C)		see text
(state S)		$S^{\mathcal{I}}$
(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$_n$)		${S_1}^{\mathcal{I}} \wedge \cdots \wedge {S_n}^{\mathcal{I}}$
(or S$_1$ ... S$_n$)		${S_1}^{\mathcal{I}} \vee \cdots \vee {S_n}^{\mathcal{I}}$
(not S)		$\neg S^{\mathcal{I}}$
(instance IN C)	$IN \in C$	$IN^{\mathcal{I}} \in C^{\mathcal{I}}$
(related IN I R)	$\langle IN, I \rangle \in R$	$\langle IN^{\mathcal{I}}, I^{\mathcal{I}} \rangle \in R^{\mathcal{I}}$
(equal IN$_1$ IN$_2$)		${IN_1}^{\mathcal{I}} = {IN_2}^{\mathcal{I}}$

Table 5: Assertion Syntax and Semantics

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

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)

(validate-true QQ)
(validate-not-true QQ)
(validate-set RR N_1 ... N_n)

Table 7: Retrieval and Validation Syntax

that the token System be defined as the KR system for which the KB was written, and that the token Compliance be defined as the compliance type for the KB.

Inquiries, encompassing queries, retrievals, and validations, are as in Tables 6 and 7.

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].

Semantics are defined directly only for simple KBs. 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 KB is transformed into a simple KB 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 KB 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 KB 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 KB 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, $\Delta_c^{\mathcal{I}}$. The rest of the domain, $\Delta_a^{\mathcal{I}}$, 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 $\Delta_a^{\mathcal{I}}$ to $\Delta^{\mathcal{I}}$. The extensions of attributes are single-valued, partial functions from $\Delta_a^{\mathcal{I}}$ to $\Delta^{\mathcal{I}}$. (The extension of roles and attributes will sometimes also be treated as the equivalent subset of $\Delta_a^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. 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 KB if each statement in the simple KB is is true in the set. A simple KB (any KB) 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 KB if for each individual name, IN, in the simple KB, 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 KB 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 KB version of the concept, role, attribute, disjointness, and individual definitions (i.e., no rules or assertions or closures) in the original KB. An instance relationship, $\mathsf{IN} \in \mathsf{C}$, follows from a KB 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 KB version of the original KB.

A query follows from a KB if its meaning in Table 6 follows from the KB.

Retrievals return sets (as lists) of concept, role, or individual names.

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).

The role retrievals are defined analogously.

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

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.

4 Compliance

Conforming implementations must parse the entire syntax. If a conforming implementation cannot internally represent a particular statement, it must replace it with the syntactically-closest 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) query definition 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} ... AN_{1n}) (compose AN_{21} ... AN_{2m}))
(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. The actions of conforming implementations on inconsistent knowledge bases are unspecified, but 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. Conforming implementations may produce warning messages for various events, such as

1. incoherent concept and role definitions, and

2. redundant statements.

Conforming implementations must perform complete reasoning on core knowledge bases, except that their behavior on subsumption queries where one or both concepts is incoherent (i.e., has an empty extension in all model sets) is undefined.

4.1 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 Suites

A test suite exists at AT&T Bell Labs.