Knowledge Interchange Format — Part 1: KIF-Core

Warning

This document is not an ISO International Standard. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an International Standard.

Recipients of this document are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.
Copyright notice

This ISO document is a working draft or committee draft and is copyright-protected by ISO. While the reproduction of working drafts or committee drafts in any form for use by participants in the ISO standards development process is permitted without prior permission from ISO, neither this document nor any extract from it may be reproduced, stored or transmitted in any form for any other purpose without prior written permission from ISO.

Requests for permission to reproduce this document for the purpose of selling it should be addressed as shown below or to ISO’s member body in the country of the requester:

[Indicate: the full address, telephone number, fax number, telex number, and electronic mail address]

as appropriate, of the Copyright Manager of the ISO member body responsible for the secretariat of the TC or SC within the framework of which the draft has been prepared]

Reproduction for sales purposes may be subject to royalty payments or a licensing agreement.

Violators may be prosecuted.
# Contents

1  Scope......................................................................................................................................................1
  1.1  Scope of KIF ........................................................................................................................................1
  1.2  Scope of KIF-Core...................................................................................................................................2

2  Normative references ...............................................................................................................................2

3  Terms and definitions ...............................................................................................................................2

4  Symbols (and abbreviated terms) ............................................................................................................3

5  Syntax of KIF-Core....................................................................................................................................4
  5.1  Introduction .........................................................................................................................................4
  5.2  Characters ..........................................................................................................................................4
  5.3  Lexemes ..............................................................................................................................................4
    5.3.1  Character Strings..........................................................................................................................4
    5.3.2  Words 4
  5.4  Expressions (Logical Lexicon) ...........................................................................................................5
    5.4.1  Terms 5
    5.4.2  Sentences ....................................................................................................................................6
  5.5  Knowledge Bases ...............................................................................................................................7

6  Semantics of KIF-Core.............................................................................................................................7
  6.1  Introduction .........................................................................................................................................7
  6.2  Semantics for KIF-Compliant Languages............................................................................................7
    6.2.1  Interpretation Function..................................................................................................................8
    6.2.2  Semantic Value Function for Terms.............................................................................................8
    6.2.3  Satisfaction Function for Sentences............................................................................................8

7  Conformance ...........................................................................................................................................10
  7.1  Introduction .......................................................................................................................................10
  7.2  Conformance Dimensions ..................................................................................................................11
    7.3  Introduction .....................................................................................................................................11
    7.3.3  Logical Form.................................................................................................................................11
    7.3.5  Term Complexity.........................................................................................................................11
    7.3.6  Order 11
    7.3.7  Quantification ............................................................................................................................12
  7.4  Common Conformance Profiles .........................................................................................................12
  7.5  Dealing with Differences in Conformance Profiles ............................................................................12

Annex A  (normative)  Infix Notation ...........................................................................................................13
  A.1  Introduction ......................................................................................................................................13
  A.2  BNF Grammar..................................................................................................................................13

Annex B  (informative)  Example Knowledge Bases..................................................................................15
Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3.

Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

International Standard ISO nnn-n was prepared by Technical Committee ISO/JTC 1, Information Technology Standards, Subcommittee SC 32, Data Management and Interchange.

ISO nnn consists of the following parts, under the general title Knowledge Interchange Format

— Part 1: KIF-Core
— Part 2: Sorted KIF
— Part 3: MetaKIF

This document is Part 1 of the Knowledge Interchange Format.

- Annex A (Infix Notation) is normative and Annex B (Example Knowledge Base) is informative.
Introduction

The Knowledge Interchange Format (KIF) is a computer-oriented language for the interchange of knowledge among disparate programs. It has a declarative semantics (i.e. the meaning of expressions in the representation can be understood without appeal to an interpreter for manipulating those expressions); it is logically comprehensive (i.e. it provides for the expression of arbitrary sentences in the first-order predicate calculus); it provides for the specification of class hierarchies; and it provides for the representation of knowledge about knowledge.

The standard is divided into three parts. Part 1 (KIF-Core) specifies the syntax and semantics of a language equivalent to first-order logic. Part 2 (Sorted KIF) is an expansion of the language of KIF-Core that specifies the syntax and semantics of a many-sorted first-order logic. Part 3 (MetaKIF) is an expansion of the language of KIF-Core that formalizes the syntax and semantics of the metatheory of KIF-Core.

All parts of the KIF standard preserve the semantics of the symbols in the logical lexicon of KIF-Core.
Knowledge Interchange Format – Part 1: KIF-Core

1 Scope

1.1 Scope of KIF

Knowledge Interchange Format (KIF) is a language designed for use in the interchange of knowledge among disparate computer systems (created by different programmers, at different times, in different languages, and so forth).

The following categorical features are essential to the design of KIF.

- The language has declarative semantics. It is possible to understand the meaning of expressions in the language without appeal to an interpreter for manipulating those expressions.
- The language is logically comprehensive—at its most general, it provides for the expression of arbitrary logical sentences.
- The language provides for the representation of knowledge about knowledge.

In addition to these essential features, KIF is designed to maximize the following additional features (to the extent possible while preserving the preceding features).

- Implementability. Although KIF is not intended for use within programs as a representation or communication language, it should be usable for that purpose if so desired.
- Readability. Although KIF is not intended primarily as a language for interaction with humans, human readability facilitates its use in describing representation language semantics, its use as a publication language for example knowledge bases, and its use in assisting humans with knowledge base translation problems.

KIF is not intended as a primary language for interaction with humans.

KIF is not intended as an internal representation for knowledge within computer systems or within closely related sets of computer systems.

NOTE: Different computer systems can interact with their users in whatever forms are most appropriate to their applications (for example Prolog, conceptual graphs, natural language, and so forth).

NOTE: Typically, when a computer system reads a knowledge base in KIF, it converts the data into its own internal form (specialized pointer structures, arrays, etc.). All computation is done using these internal forms. When the computer system needs to communicate with another computer system, it maps its internal data structures into KIF.

NOTE: The purpose of KIF is roughly analogous to that of Postscript. Postscript is commonly used by text and graphics formatting systems in communicating information about documents to printers. Although it is not as efficient as a specialized representation for documents and not as perspicuous as a specialized wysiwyg display, Postscript is a programmer-readable representation that facilitates the independent development of formatting programs and printers. While KIF is not as efficient as a specialized representation for knowledge nor as perspicuous as a specialized display (when printed in its list form), it too is a programmer-readable language and thereby facilitates the independent development of knowledge-manipulation programs.
1.2 Scope of KIF-Core

Part 1 of KIF specifies the syntax and semantics of a language that is equivalent to first-order logic. This consists of logical symbols for connectives (conjunction, disjunction, negation, implication, equivalence), equality, and quantifiers (existential and universal) that range over a universe of discourse that includes objects and those functions and relations that are denoted by words within the nonlogical lexicon of the language.

The following are outside the scope of KIF-Core:

- many-sorted languages
- second-order logic
- generalized quantifiers
- free logics
- conditional logics
- intuitionistic logics
- modal logics

2 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this part of ISO nnn. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this part of ISO nnn are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of ISO and IEC maintain registers of currently valid International Standards.

ISO/IEC 10646-1:1993, Information Technology (IT) - Universal Multiple-Octet Coded Character Set (UCS).

3 Terms and definitions

For the purposes of this part of ISO nnn, the following terms and definitions apply.

antecedent

completeness

conceptual graphs

consequent

consistent

function

interpretation
4 Symbols (and abbreviated terms)

KIF: Knowledge Interchange Format

\( O \) : universe of discourse

\( i \) : interpretation function

\( s \) : semantic value function

\( t \) : satisfaction function

\( v \) : variable assignment function

\( \nu \) : variable

\( \rho \) : relation word

\( \tau \) : term

\( \phi \) : sentence

\( \phi((\nu_1,...,\nu_n)/(d_1,...,d_n)) \) : the sentence in which all variables \( \nu_i \) in \( \phi \) have been substituted with objects \( d_i \) from the universe of discourse

\( \sigma \) : word in nonlogical lexicon
5 Syntax of KIF-Core

5.1 Introduction

The syntax of KIF-Core is described in three layers. First, there are the basic characters of the language. These characters can be combined to form lexemes. Finally, the lexemes of the language can be combined to form grammatically legal expressions.

In this clause, the syntax of KIF-Core is presented using a modified BNF notation. The notation nonterminal* means zero or more occurrences; nonterminal+ means one or more occurrences; The nonterminals space, tab, return, linefeed, and page refer to the characters corresponding to ascii codes 32, 9, 13, 10, and 12, respectively. The nonterminal character denotes the set of all 128 ascii characters. The alphabet of KIF-Core consists of 7 bit blocks of data. In this document, we refer to KIF data blocks via their usual ASCII encodings as characters (as given in ISO 646:1983).

5.2 Characters

KIF characters are classified as upper case letters, lower case letters, digits, alpha characters (non-alphabetic characters that are used in the same way that letters are used), special characters, white space, and other characters (every ascii character that is not in one of the other categories).

upper := A | B | C | D | E | F | G | H | I | J | K | L | M |
        N | O | P | Q | R | S | T | U | V | W | X | Y | Z
lower := a | b | c | d | e | f | g | h | i | j | k | l | m |
        n | o | p | q | r | s | t | u | v | w | x | y | z
digit := 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
special := ! | $ | % | & | * | + | - | . | / | < | = | > |
          ? | @ | _ | ~ |
white ::= space | tab | return | linefeed | page
initialchar ::= upper | lower
wordchar ::= upper | lower | digit | - | _ | special
normal ::= upper | lower | digit | special | white

Use of characters in “special” for word characters is discouraged as they may be given particular meaning in future versions of the standard or its extensions.

5.3 Lexemes

5.3.1 Character Strings

A character string is a series of characters enclosed in quotation marks. The escape character \ is used to permit the inclusion of quotation marks and the \ character itself within such strings.

string ::= “character*”

5.3.2 Words

A constant is a letter or digit followed by any number of other legal word characters.

word ::= initialchar wordchar*
For the purpose of grammatical analysis, it is useful to subdivide the class of words a little further, viz. as variables, operators, and constants.

5.3.2.1 Variables

A variable is a word in which the first character is `?`.

```
variable ::= ?word
```

5.3.2.2 Operators

Operators are used in forming complex expressions of various sorts. Within KIF-Core, there are only sentence operators, which are used in forming complex sentences.

```
sentop ::= = | not | and | or | => | <= | <=> | forall | exists
```

5.3.2.3 Relation Words and Function Words

A distinction is made between words that denote relations and words that denote functions.

```
relword ::= initialchar wordchar*
funword ::= initialchar wordchar*
```

No “relword” and “funword” shall have the same character sequence in a particular knowledge base.

5.3.2.4 Constants

All other words are called constants.

```
constant ::= word - variable – sentop – funword – relword
```

5.4 Expressions (Logical Lexicon)

The legal expressions of KIF are formed from lexemes according to the rules presented in this clause. There are two disjoint types of legal expressions in KIF Core – terms and sentences.

NOTE: Terms are used to denote objects in the world being described; sentences are used to express facts about the world; and definitions are used to define constants.

5.4.1 Terms

There are three types of terms in KIF-Core—variables, constants, and function terms. Variables and constants were discussed in clause 5.3.2.

```
term ::= variable | constant | funterm
```

5.4.1.1 Function Terms

A function term consists of a constant and an arbitrary number of argument terms, surrounded by matching parentheses.

NOTE: there is no syntactic restriction on the number of argument terms; arity restrictions in KIF-Core are treated
semantically.
funterm ::= (funword term+)

5.4.2 Sentences
The following BNF defines the set of legal sentences in KIF-Core. There are four types of sentences.
sentence ::= equation | logsent | relsent | quantsent

5.4.2.1 Equations
An equation consists of the = operator and two terms.
equation ::= (= term term)

5.4.2.2 Relational Sentences
A relational sentence consists of a constant and an arbitrary number of argument terms.
relsent ::= (relword term+)
NOTE: As with functional terms, there is no syntactic restriction on the number of argument terms in a relation sentence.

5.4.2.3 Logical Sentences
The syntax of logical sentences depends on the logical operator involved. A sentence involving the not operator is called a negation. A sentence involving the and operator is called a conjunction, and the arguments are called conjuncts. A sentence involving the or operator is called a disjunction, and the arguments are called disjuncts. A sentence involving the => operator is called an implication; all of its arguments but the last are called antecedents; and the last argument is called the consequent. A sentence involving the <=> operator is called an equivalence.
logsent ::= (not sentence) | (and sentence*) | (or sentence*) | (=> sentence* sentence) | (<= sentence sentence)

5.4.2.4 Quantified Sentences
There are two types of quantified sentences—a universally quantified sentence is signalled by the use of the forall operator, and an existentially quantified sentence is signalled by the use of the exists operator. The first argument in each case is a list of variable specifications. A variable specification is either a variable or a list consisting of a variable and a term denoting a relation that restricts the domain of the specified variable.
quantsent ::= (forall (variable+) sentence) | (exists (variable+) sentence)
NOTE: According to these rules, it is permissible to write sentences with free variables, i.e. variables that do not occur within the scope of any enclosing quantifiers. The significance of the free variables in a sentence depends on the use of the sentence. When we assert the truth of a sentence with free variables, we are, in effect, saying that the sentence is true for all values of the free variables, i.e. the variables are universally quantified. When we ask whether a sentence with free variables is true, we are, in effect, asking whether there are any values for the free variables for which the sentence is true, i.e. the variables are existentially quantified.
5.5 Knowledge Bases

A knowledge base is a set of sentences.

NOTE: It is important to keep in mind that a knowledge base is not a sequence; and, therefore, the order of sentences within a knowledge base is unimportant. Order may have heuristic value to deductive programs by suggesting an order in which to use those sentences; however, this implicit approach to knowledge exchange lies outside of the definition of KIF.

6 Semantics of KIF-Core

6.1 Introduction

The basis for the semantics of KIF is a conceptualization of the world in terms of objects and relations among those objects. A universe of discourse is the set of all objects presumed or hypothesized to exist in the world.

NOTE: The notion of object used here is quite broad. Objects can be concrete (e.g. a specific carbon atom, Confucius, the Sun) or abstract (e.g. the number 2, the set of all integers, the concept of justice). Objects can be primitive or composite (e.g. a circuit that consists of many subcircuits). Objects can even be fictional (e.g. a unicorn, Sherlock Holmes).

Different users of a declarative representation language, like KIF, are likely to have different universes of discourse. KIF is conceptually promiscuous in that it does not require every user to share the same universe of discourse.

In KIF-Core, relationships among objects take the form of relations. Formally, a relation is defined as an arbitrary set of finite lists of objects (of possibly varying lengths). Each list is a selection of objects that jointly satisfy the relation. For example, the < relation on numbers contains the list <2,3>, indicating that 2 is less than 3.

A function is a special kind of relation. For every finite sequence of objects (called the arguments), a function associates a unique object (called the value). More formally, a function is defined as a set of finite lists of objects, one for each combination of possible arguments. In each list, the initial elements are the arguments, and the final element is the value.

EXAMPLE: The 1+ function contains the list <2,3>, indicating that integer successor of 2 is 3.

NOTE: Both functions and relations are defined as sets of lists. In fact, every function is a relation. However, not every relation is a function. In a function, there cannot be two lists that disagree on only the last element. This would be tantamount to the function having two values for one combination of arguments. By contrast, in a relation, there can be any number of lists that agree on all but the last element. For example, the list <2,3> is a member of the 1+ function, and there is no other list of length 2 with 2 as its first argument, i.e. there is only one successor for 2. By contrast, the < relation contains the lists <2,3>, <2,4>, <2,5>, and so forth, indicating that 2 is less than 3, 4, 5, and so forth.

NOTE: Many mathematicians require that functions and relations have fixed arity, i.e. they require that all of the lists comprising a relation have the same length. The definitions here allow for relations with variable arity, i.e. it is perfectly acceptable for a function or a relation to contain lists of different lengths. For example, the relation < contains the lists <2,3> and <2,3,4>, reflecting the fact that 2 is less than 3 and the fact that 2 is less than 3 and 3 is less than 4. This flexibility is not essential, but it is extremely convenient and poses no significant theoretical problems.

6.2 Semantics for KIF-Compliant Languages

A language shall be KIF-compliant if and only if there exist the following functions:

- interpretation function \(i\) that assigns an element in the conceptualization to every symbol in the nonlogical lexicon of the language;

- semantic value function \(s\) that assigns objects in the universe of discourse to terms in the language;
satisfaction function \( t \) that maps sentences into the truth values \( true \) or \( false \).

### 6.2.1 Interpretation Function

In a KIF-compliant language, the interpretation function shall have the following properties:

- If \( \sigma \) is an object constant, then \( i(\sigma) \in O \);
- If \( \sigma \) is a function word, then \( i(\sigma) : O^* \rightarrow O \), and \( i(\sigma) \in O \);
- If \( \sigma \) is a relation word, then \( i(\sigma) \subseteq O^* \) and \( i(\sigma) \in O \).

**NOTE:** The universe of discourse contains not only objects, but also functions and relations that are denoted by words in the language.

### 6.2.2 Semantic Value Function for Terms

In a KIF-compliant language, the semantic value function \( s \) shall have the properties specified in the following clauses.

#### 6.2.2.1 Semantic Value of Variables

The semantic value of a variable is the object assigned to that variable by the given variable assignment:

\[
s(\nu) = v(\nu)
\]

#### 6.2.2.2 Semantic Value of Constants

The semantic value of a constant \( \sigma \) is the object assigned to that constant by the interpretation function:

\[
s(\sigma) = i(\sigma)
\]

#### 6.2.2.3 Semantic Value of Function Terms

The semantic value of a functional term is obtained by applying the function denoted by the function word in the term to the objects denoted by the arguments.

**EXAMPLE:** The value of the term \( (+ 2 3) \) is obtained by applying the addition function (the function denoted by +) to the numbers 2 and 3 (the objects denoted by the object constants 2 and 3) to obtain the value 5, which is the value of the object constant 5.

### 6.2.3 Satisfaction Function for Sentences

In a KIF-compliant language, the satisfaction function shall have the properties specified in the following clauses.

#### 6.2.3.1 Equations

An equation is true if and only if the terms in the equation refer to the same object in the universe of discourse.

\[
t(= (\tau_1, \tau_2)) = \begin{cases} 
true & s(\tau_1) = s(\tau_2) \\
false & otherwise
\end{cases}
\]
6.2.3.2 Relational Sentences

A simple relational sentence without a terminating sequence variable is true if and only if the relation denoted by the relation constant in the sentence is true of the objects denoted by the arguments. Equivalently, viewing a relation as a set of tuples, we say that the relational sentence is true if and only if the tuple of objects formed from the values of the arguments is a member of the set of tuples denoted by the relation constant.

\[
t((\rho, \tau_1, \ldots, \tau_n)) = \begin{cases} 
\text{true} & \left( s(\tau_1), \ldots, s(\tau_n) \right) \in \iota(\rho) \\
\text{false} & \text{otherwise}
\end{cases}
\]

6.2.3.3 Not

A negation is true if and only if the negated sentence is false.

\[
t(\lnot \varphi) = \begin{cases} 
\text{true} & \iota(\varphi) = \text{false} \\
\text{false} & \text{otherwise}
\end{cases}
\]

6.2.3.4 Or

A disjunction is true if and only if at least one of the disjuncts is true.

\[
t(\bigvee \varphi_1 \ldots \varphi_n) = \begin{cases} 
\text{true} & \iota(\varphi_j) = \text{true for some } 1 \leq j \leq n \\
\text{false} & \text{otherwise}
\end{cases}
\]

6.2.3.5 And

A conjunction is true if and only if every conjunct is true.

\[
t(\bigwedge \varphi_1 \ldots \varphi_n) = \begin{cases} 
\text{true} & \iota(\varphi_j) = \text{true for all } 1 \leq j \leq n \\
\text{false} & \text{otherwise}
\end{cases}
\]

6.2.3.6 =>

If every antecedent in an implication is true, then the implication as a whole is true if and only if the consequent is true. If any of the antecedents is false, then the implication as a whole is true, regardless of the truth value of the consequent.

\[
t(\Rightarrow \varphi_1 \ldots \varphi_n \varphi) = \begin{cases} 
\text{true} & \text{for some } j \iota(\varphi_j) = \text{false or } \iota(\varphi) = \text{true} \\
\text{false} & \text{otherwise}
\end{cases}
\]
6.2.3.7 <=>

An equivalence is true if and only if all arguments have the same truth value.

\[
t((\leftrightarrow \varphi_1 \varphi_2)) = \begin{cases} 
  \text{true} & t(\varphi_1) = t(\varphi_2) \\
  \text{false} & \text{otherwise}
\end{cases}
\]

6.2.3.8 exists

A simple existentially quantified sentence (one in which the first argument is a list of variables) is true if and only if the embedded sentence is true for some value of the variables mentioned in the first argument.

\[
t((\exists \varphi_1 \varphi_2 \cdots \varphi_n) \varphi) = \begin{cases} 
  \text{true} & \text{there exist } d_1, \ldots, d_n \in O \text{ such that } t(\varphi((\nu_1, \ldots, \nu_n)/ (d_1, \ldots, d_n))) = \text{true} \\
  \text{false} & \text{otherwise}
\end{cases}
\]

6.2.3.9 forall

A simple universally quantified sentence (one in which the first argument is a list of variables) is true if and only if the embedded sentence is true for every value of the variables mentioned in the first argument.

\[
t((\forall \varphi_1 \varphi_2 \cdots \varphi_n) \varphi) = \begin{cases} 
  \text{true} & \text{for every } d_1, \ldots, d_n \in O, t(\varphi((\nu_1, \ldots, \nu_n)/ (d_1, \ldots, d_n))) = \text{true} \\
  \text{false} & \text{otherwise}
\end{cases}
\]

7 Conformance

This clause specifies criteria for building fully knowledge-based systems that fully conform to the KIF standard.

7.2 Introduction

The basic language specification is augmented with a set of “conformance dimensions”. These dimensions are not the same as the “conformance levels” of other languages. Rather, each conformance dimension has a variety of levels within that dimension.

A “conformance profile” is a selection of alternatives from each conformance dimension. System builders are expected to make choices for each dimension and then ensure that their systems adhere to the resulting conformance profile. Systems are expected to use the terminology defined here to share information about their conformance profile with other systems (in a protocol-specific manner).

Although this conformance profile scheme is more complex than one based on conformance levels, it accommodates varying capabilities and/or computational constraints while providing a migration path from more restrictive to more expressive.
7.3 Conformance Dimensions

7.3.3 Introduction

A conformance dimension is a classification of KIF sentences into conformance categories on the basis of a single syntactic criterion. (For example, the quantification dimension provides two categories, quantified KIF and unquantified KIF, based on whether or not a conforming knowledge base contains quantifiers.)

7.3.4 Logical Form

The first conformance dimension concerns logical form. There are five basic categories: atomic, conjunctive, positive, logical, and rule-like.

Rule-like knowledge bases are further categorized as Horn or non-Horn and recursive or nonrecursive.

A knowledge base is atomic if and only if it contains no logical operators.

A knowledge base is conjunctive if and only if it contains no logical operators except for conjunction.

A knowledge base is positive if and only if it contains no logical operators except for conjunction and disjunction.

A knowledge base is logical if and only if it contains no logical operators except for conjunction, disjunction, and negation.

A knowledge base is rule-like if and only if every sentence is either atomic or an implication or reverse implication in which all subexpressions are atomic sentences or negations of atomic sentences. A rule system is a rule-like knowledge base.

A rule system is Horn if and only if every constituent of every rule is atomic (i.e. no negations allowed). Otherwise, the rule system is said to be non-Horn.

The dependency graph for a rule system is a graph whose nodes are the constants in relational position. There is an edge from the node for a given relation constant p to the node of relation constant q if and only if p appears in the body of a rule whose head predicate is p.

A rule system is recursive if there is a cycle in its dependency graph. Otherwise, the rule system is said to be non-recursive.

7.3.5 Term Complexity

The nature of terms defines a second conformance dimension. There are two categories: simple and complex.

A knowledge base is simple if and only if the only terms occurring the knowledge base are constants and variables.

A knowledge base is complex if and only if it contains functional terms.

7.3.6 Order

The third conformance dimension concerns the presence or absence of variables.

A knowledge base is ground, or zeroth-order, if and only if it contains no variables. Otherwise, a knowledge base in nonground.

A knowledge base is first-order if and only if there are no variables in the first argument of any explicit functional term or explicit relational sentence.

A knowledge base is higher-order otherwise.
7.3.7 Quantification

For nonground knowledge bases, there are two alternatives—quantified and unquantified.

A nonground knowledge base is quantified if and only if it contains at least one explicit quantifier.

A nonground knowledge base is unquantified if and only if it contains no explicit quantifiers.

7.4 Common Conformance Profiles

A conformance profile is a selection of alternatives for each conformance dimension. Given the dimensions and categories defined in the preceding section, it is possible to define a large number of profiles. A single system may use different profiles in different types of communication. In particular, it is common to use one profile for assertions and another for queries. The following paragraphs define a few common types of systems with their corresponding profiles.

A database system is one in which (1) all assertions are atomic, simple, ground, and baselevel and (2) all queries are positive, simple, unquantified, and baselevel.

A Horn system (e.g. pure Datalog) is one in which (1) all assertions are rules that are Horn, unquantified, and baselevel and (2) all queries are positive, non-recursive, unquantified, and baselevel.

A relational system is one in which (1) all assertions are rules that are simple, unquantified (but may be non-Horn and non-recursive), and baselevel and (2) all queries are logical, non-recursive, unquantified, and baselevel.

A first-order system is one that allows the broadest categories within each conformance dimension except that only first-order expressions are accommodated.

A full KIF system is one that accepts the broadest categories within each conformance dimension, i.e. any KIF knowledge base is acceptable in any context.

7.5 Dealing with Differences in Conformance Profiles

The existence of multiple conformance profiles raises the question of what happens when systems with different profiles must communicate.

Whenever the conformance profile of a receiver is known, a sender should avoid sending expressions that fall outside the receiver’s conformance profile.

Unfortunately, this rule cannot be enforced in all situations. In some cases, conformance information about receivers is unavailable; and, even when conformance information is available, it may be desirable to send a message that falls outside a receiver’s profile, e.g. it may be most efficient for a sender to broadcast a single knowledge base to a large number of receivers with differing conformance profiles rather than sending different knowledge bases to each receiver.

Whenever a receiver receives a non-conforming expression, it is free to ignore the expression, even though it may be able to make sense of portions of that expression. If the receiver ignores a non-conforming expression and the sender requests a reply, the receiver should report a failure.
Annex A
(normative)

Infix Notation

A.1 Introduction

Infix KIF is a syntactic variant of prefix KIF using infix syntax. For every expression in Infix KIF, there is a logically equivalent expression in KIF. However, the reverse is not true; some KIF expressions have been dropped for simplicity, eg. complex conditionals, complex variable specifications in quantified sentences, and documentation strings in definitions.

This chapter gives the full grammar for Infix KIF using nonterminals similar to those in the specification of Prefix KIF. (A few extra nonterminals are used here to define the more complicated syntax of Infix KIF.) This parallel treatment of nonterminals conveys the relationship between the two syntaxes.

Note that Infix KIF is NOT intended for use in the exchange of knowledge between computers. It is provided solely for consumption by humans, in articles about KIF and possibly in knowledge base editors. Prefix KIF is preferred for exchange of knowledge between machines. Its simpler syntax makes it easier to write parsers and makes it possible to write parsers that are more efficient.

A.2 BNF Grammar
Annex B
(informative)

Example Knowledge Bases
Bibliography


[5] IEC 60027 (all parts), *Letter symbols to be used in electrical technology.*

[6] ISO 1000, *SI units and recommendations for the use of their multiples and of certain other units.*


A Bibliography, if present, shall appear after the last annex. The drafting rules set out in ISO 690[7] shall be followed.

The bibliography may include

- documents that are not publicly available,
- documents to which only informative reference is made, and
- documents which have merely served as references in the preparation of the standard.