

Reference number of working document: **ISO/JTC1/SC32/WG2 N 000**

Date: 2001-10-01

Reference number of document: **ISO/WD nnn-n**

Committee identification: **ISO/JTC1/SC 32/WG 2**

Secretariat: **XXX**

Knowledge Interchange Format — Part 1: First-Order KIF

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.

Document type: **International standard**
Document subtype: **if applicable**
Document stage: **(20) Preparation**
Document language: **E**

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 First-Order KIF	2
2	Normative references	3
3	Terms and definitions	3
4	Symbols (and abbreviated terms).....	6
5	Syntax of First-Order KIF	7
5.1	Introduction	7
5.2	Characters	8
5.3	Lexemes.....	9
5.3.1	Character Strings.....	9
5.3.2	Words 9	
5.4	Expressions	10
5.4.1	KIF Languages	10
5.4.2	Terms 11	
5.4.3	Sentences	12
5.5	KIF Modules	14
5.5.1	Expressions for Module Management.....	14
6	Semantics of First-Order KIF.....	15
6.1	Fundamental Principles	15
6.2	Semantics for First-Order KIF Languages	16
6.2.1	Extension Function	16
6.2.2	Semantic Value Function for Terms	17
6.2.3	Satisfaction Function for Sentences	17
7	Semantic Conformance.....	21
7.1	Soundness.....	21
7.2	Completeness	22
8	Syntactic Conformance.....	22
8.1	Full Syntactic Conformance	22
8.2	Restricted Syntactic Conformance.....	23
8.3	Expanded Syntactic Conformance	23
	Annex A (normative) BNF Grammar for First-Order KIF	25
	Annex B (informative) Relationship to Conventional First-Order Logic.....	30
	Holding and applying	30
	Model Theory with Non-Well-Founded Sets	32
	Annex C (informative) Example KIF Sentences.....	34
	Partial Orders	36
	Relations as Arguments to Relations.....	37

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: First-Order KIF*
- *Part 2: Infinitary KIF*
- *Part 3: MetaKIF*

This document is Part 1 of the Knowledge Interchange Format.

- Annex A (BNF Grammar for First-Order KIF) is normative; Annex B (Relationship to Conventional First-Order Logic) and Annex C (Example KIF Modules) 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 (First-Order) specifies the syntax and semantics of a language equivalent to first-order logic. Part 2 (Infinitary KIF) is an expansion of the language of First-Order KIF that is equivalent to a fragment of infinitary logic. Part 3 (MetaKIF) is an expansion of the language of First-Order 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: First-Order KIF

1 Scope

1.1 Scope of KIF

Knowledge Interchange Format (KIF) is a language designed for use in the representation and interchange of knowledge among disparate computer systems.

The following 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 following are within the scope of this standard:

- interchange of knowledge among heterogeneous computer systems;
- representation of knowledge in ontologies and knowledge bases;
- specification of expressions that are the input or output of inference engines.

The following are outside the scope of this standard:

- the specification of proof theory or inference rules;
- specification of translators between the notations of heterogeneous computer systems.

1.2 Scope of First-Order KIF

Part 1 of this standard specifies the syntax and semantics of a first-order language with equality. 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 First-Order KIF:

- 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).

ISO/IEC 14481:1998 (FCD), Information Technology (IT) - Conceptual Schema Modeling Facilities (CSMF).

3 Terms and definitions

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

entails

A KIF theory T entails a sentence Φ if and only if $T \cup \neg\Phi$ is unsatisfiable.

NOTE: This is equivalent to the condition that every model of T satisfies Φ .

extension function

assigns an element in the universe of discourse to every symbol in the nonlogical lexicon of the language

functional

a relation r is *functional on* $\langle e_1, \dots, e_n \rangle$ just in case there is exactly one object e in the domain of discourse O such that $\langle e_1, \dots, e_n, e \rangle$ is in the extension of r

inference system

a computer system that decides whether or not a sentence is satisfiable by a KIF theory.

I-variant

for an interpretation $I = \langle O, ext, \sigma, \tau \rangle$ and for any variables v_1, \dots, v_n , say that an interpretation $I' = \langle O, ext, \sigma', \tau \rangle$ is an *I-variant on* v_1, \dots, v_n if σ' differs from σ at most in what it assigns to one or more of v_1, \dots, v_n .

model

A structure is a model of a KIF module T if and only if it satisfies each sentence in T .

satisfaction function

maps sentences into the truth values *true* or *false*

satisfiable

a KIF module T is satisfiable if and only if there exists a structure that satisfies every sentence in T .

satisfies

A structure satisfies a sentence Φ if and only if the interpretation function of the structure assigns the sentence the value *true*.

semantic value function

assigns objects in the universe of discourse to terms in the language

structure

a nonempty set O of objects with an extension function, a semantic value function, and a satisfaction function.

universe of discourse

a nonempty set of individuals.

unsatisfiable

a KIF module T is unsatisfiable if and only if there does not exist a structure that satisfies every sentence in T .

4 Symbols (and abbreviated terms)

KIF: Knowledge Interchange Format

E : set of individuals in a universe of discourse

O : universe of discourse

R : set of relations in a universe of discourse

ext: extension function

i : interpretation function

s : semantic value function

t : satisfaction function

L_I : language of an inference system I as specified by its EBNF.

L_{KIF} : language of First-Order KIF

λ_e : syntactic mapping $\lambda_e : L_I \rightarrow L_{KIF}$ from expressions in L_I to sentences in L_{KIF} .

λ_i : syntactic mapping $\lambda_i : L_{KIF} \rightarrow L_I$ from sentences in L_{KIF} to expressions in L_I .

O^n : set of sequences of length n of members of O

O^* : set of all sequences over O

$\langle \rangle$: sequence of length 0

$cc(s_1, \dots, s_n)$: concatenation of the rows s_1, \dots, s_n .

v : variable

ρ : relation word

τ : term

φ : sentence

σ : word in namespace

5 Syntax of First-Order KIF

5.1 Introduction

The syntax of First-Order KIF 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).

```
<alpha> ::= 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
           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
<marker> ::= @ | # | ? | ` | . | = | / | [ | : | %
<other> ::= ! | $ | % | & | * | + | , | - | ; | ] | ^ | _ | { | } | ~ | > | '
<white> ::= <space> | <tab> | <return> | <linefeed> | <page>
<lexbreak> ::= <white> | ( | ) | " | \
<wordchar> ::= <alpha> | <digit> | <marker> | <other>
```

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.

```
<stringchar> ::= <wordchar> | <white> | (|)
<docstring> ::= "<stringchar>*"

```

5.3.2 Words

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

```
<word> ::= {<alpha> | <other>} <wordchar>
<wordbody> ::= <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 Object Variables

An object variable is a word in which the first character is ?.

```
<objvariable> ::= ?<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 | implies | iff |
           forall | exists

```

5.3.2.3 Constants

All other words are called constants.

```
<constant> ::= <word> - <objvariable> - <sentop>

```

5.4 Expressions

The legal expressions of KIF-Core are formed from lexemes according to the rules presented in this clause.

5.4.1 KIF Languages

A namespace N consists of a recursive set of words. The elements of this namespace are *constant names* that refer to the distinguished individuals, properties, classes, and relations in a conceptualization.

The KIF *language* λ_N *generated by the namespace* N is the set of expressions generated by the above BNF when N is the set of all <constant>s. In particular, the expressions generated by <sentence>, <constant>, <term>, <objterm>, and <fnterm> are known as the *sentences*, *constants*, *terms*, *object terms*, and *function terms* of λ_N , respectively.

NOTE 1: there is no distinction between individual, function or relation symbols, so that any name can be used in any of these roles. A particular KIF language may have a more rigidly stratified vocabulary, but this is not required by KIF itself.

5.4.2 Terms

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

5.4.2.1 Object Terms

```
<objterm> ::= <objvariable> | <constant> | <fnterm>
```

NOTE : object variables and other terms may occur in the first position of a term. This allows expressions with variables in the relation or function position, or complex terms that denote relations or functions.

5.4.2.2 Function Terms

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

```
<fnterm> ::= (<objterm> <objterm>*)
```

NOTE: there is no syntactic restriction on the number of argument terms; arity restrictions in KIF-Core are treated semantically.

5.4.3 Sentences

The following BNF defines the set of legal sentences in KIF-Core. There are four types of sentences.

```
<sentence> ::= <atomsent> | <logsent> | <quantsent>
              | (<sentence> <docstring>)
```

5.4.3.1 Equations

An equation consists of the = operator and two object terms.

```
<equalsent> ::= (= <objterm> <objterm>)
```

5.4.3.2 Atomic Sentences

An atomic sentence consists of an object terms followed by an arbitrary number of argument object terms.

```
<atomsent> ::= (<objterm>+) | <equalsent>
```

NOTE 1: There is no difference in syntactic form between terms and atomic sentences, although it is possible to recognise when any expression token is being used as a sentence.

NOTE 2: As with functional terms, there is no syntactic restriction on the number of argument terms in an atomic sentence. Terms occurring in the predicate position in an atomic sentence are not assigned a unique arity.

NOTE 3: KIF-Core allows functions and relations to be variably polyadic – it allows them to be true of varying numbers of arguments.

NOTE 4: Any term may occur in predicate position in an atomic sentence, in function position in a function term, or in argument position on an atomic sentence or function term.

5.4.3.3 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|or} <sentence>*)|
            ({implies|iff} <sentence> <sentence>)
```

5.4.3.4 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 variables.

```
<quantsent> ::= ({forall|exists}(<objvariable>*) <sentence>)
```

5.5 KIF Modules

A KIF module is a set of sentences.

NOTE: It is important to keep in mind that a KIF module is not a sequence of sentences; and, therefore, the order of sentences within a KIF module 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.

5.5.1 Expressions for Module Management

```
<specialname> ::= %<wordbody>|documentation
<restriction> ::= :<word>
<importname> ::= #<wordbody>
<quote>       ::= `<wordbody>'
<docform>    ::= (documentation [<word>] <docstring>)
```

6 Semantics of First-Order KIF

6.1 Fundamental Principles

The basis for the semantics of KIF is a conceptualization of the world in terms of objects and relations among those objects.

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).

KIF does not require every user to share the same universe of discourse.

Any name can be used as a function or a relation name; the semantics must assign a truth-value to expressions even if their function or relation symbol does not denote a function or relation. The semantics must also be able to handle self-application (an expression where the same name is used to refer to a relation or function as to one of its arguments.)

Within KIF, relations and functions are also objects within the universe of discourse. KIF utilises a basic semantic device which associates an *extension* with every object in the domain. The nature of the extension associated with an object determines what kind of object it is: relations have associated extensions which are sets of n-tuples, functions are a special class of relations whose extensions are functional, and individuals have **empty** extensions.

NOTE 1: Since relations are distinguished from their extensions, the extension of an object may contain the object (or an n-tuple which contains it), providing a way to describe self-application without violating the axiom of foundation in the underlying set theoretic semantics.

NOTE 2: KIF is not in any sense a higher-order logic. While a KIF structure must provide a denotation for every term in the language, and hence an extension for every term used as a relation or function, there is no presumption that the interpretation must contain any particular universe of relations. In contrast, classical higher-order logic requires that every domain contain all relations over the domain of individuals; higher-order syntax with the Henkin semantics requires that they contain all definable relations.

6.2 Semantics for First-Order KIF Languages

An interpretation for a First-Order KIF language is a tuple $I = \langle O, ext, \sigma, \tau \rangle$ as specified in the following clauses.

6.2.1 Extension Function

Within an interpretation I , the extension function ext has the following properties:

- $O = E \cup R$;
- $E \cap R = \emptyset$;
- If $s \in E$, then $ext(s) = \emptyset$;
- If $s \in R$, then $ext(s) \subseteq O^*$.

6.2.2 Semantic Value Function for Terms

Within an interpretation I , the semantic value function σ has the properties specified in the following clauses.

6.2.2.1 Semantic Value of Object Variables

The semantic value of an object variable v is the object assigned to that variable in the interpretation I :

$$\sigma(v) \in O$$

6.2.2.2 Semantic Value of Constants

The semantic value of a constant s is the object assigned to that constant in the interpretation I :

$$\sigma(s) \in O$$

6.2.2.3 Semantic Value of Function Terms

If $\sigma(F)$ is functional, then $\sigma((F t_1 \dots t_n))$ is the unique $e \in E$ such that $cc(\sigma(t_1), \dots, \sigma(t_n), e) \in ext(\sigma(F))$ in the interpretation I ;

otherwise, $\sigma((F t_1 \dots t_n)) = \sigma(F)$.

6.2.3 Satisfaction Function for Sentences

Within an interpretation I , the satisfaction function τ has 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 of the interpretation I .

$$\tau((= t_1 t_2)) = \begin{cases} true & \sigma(t_1) = \sigma(t_2) \\ false & otherwise \end{cases}$$

6.2.3.2 Atomic Sentences

An atomic 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 in the extension of the relation denoted by the relation constant in the interpretation I .

$$\tau((R t_1 \dots t_n)) = \begin{cases} true & cc(\sigma(t_1), \dots, \sigma(t_n)) \in ext(\sigma(R)) \\ false & otherwise \end{cases}$$

NOTE: An atomic sentence of the form (R) will be true just in case the empty sequence $\langle \rangle \in ext(\sigma(R))$.

6.2.3.3 not

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

$$\tau((not \varphi)) = \begin{cases} true & \tau(\varphi) = false \\ false & otherwise \end{cases}$$

6.2.3.4 or

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

$$\tau((or \varphi_1 \dots \varphi_n)) = \begin{cases} true & \tau(\varphi_j) = true \text{ for some } j, 0 \leq j \leq n \\ false & otherwise \end{cases}$$

NOTE: (*or*) is vacuously true under any interpretation.

6.2.3.5 and

A conjunction is true if and only if every conjunct is true in the interpretation I .

$$\tau((\text{and } \varphi_1 \dots \varphi_n)) = \begin{cases} \text{true} & \tau(\varphi_j) = \text{true for all } j, 0 \leq j \leq n \\ \text{false} & \text{otherwise} \end{cases}$$

NOTE: (*and*) is vacuously false under any interpretation.

6.2.3.6 implies

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

$$\tau((\text{implies } \varphi_1 \dots \varphi_n \varphi)) = \begin{cases} \text{true} & \text{for some } j, \tau(\varphi_j) = \text{false or } \tau(\varphi) = \text{true} \\ \text{false} & \text{otherwise} \end{cases}$$

6.2.3.7 iff

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

$$\tau((\text{iff } \varphi_1 \varphi_2)) = \begin{cases} \text{true} & \tau(\varphi_1) = \tau(\varphi_2) \\ \text{false} & \text{otherwise} \end{cases}$$

6.2.3.8 exists

An existentially quantified sentence is true if and only if the embedded sentence is true for some value of the variables mentioned in the first argument.

$$\tau((\text{exists}(v_1 \dots v_n) \varphi)) = \begin{cases} \text{true} & \tau(\varphi((v_1, \dots, v_n))) = \text{true under some } I - \text{variant} \\ \text{false} & \text{otherwise} \end{cases}$$

6.2.3.9 forall

A universally quantified is true if and only if the embedded sentence is true for every value of the variables mentioned in the first argument.

$$\tau(\text{forall}(v_1 \dots v_n) \varphi) = \begin{cases} \text{true} & \tau(\varphi((v_1, \dots, v_n))) = \text{true under every } I\text{-variant} \\ \text{false} & \text{otherwise} \end{cases}$$

7 Semantic Conformance

This clause specifies semantic criteria for building inference systems that conform to the KIF standard.

7.1 Soundness

An inference system I has simple semantic conformance to L_{KIF} if and only if there exists a syntactic mapping $\lambda_e : L_I \rightarrow L_{kif}$ such that for any sentence $\Phi \in L_I$ that I decides to be satisfiable with module $T \subset L_I$, there exists $T' \subset L_{KIF}$ such that $\lambda_e(T) \subseteq T'$ and $\lambda_e(\Phi)$ is satisfied by all models of T' .

NOTE: This is equivalent to saying that an inference system is sound with respect to First-Order KIF -- every sentence that the inference system decides to be satisfiable by a module is satisfied by some First-Order KIF model of the module.

7.2 Completeness

An inference system I has strong semantic conformance to L_{KIF} if and only if there exists a syntactic mapping $\lambda_i : L_{KIF} \rightarrow L_I$ such that for any sentence $\Phi \in L_{KIF}$ and module $T \subset L_{KIF}$, if Φ is satisfied by all models of T then I decides $\lambda_i(\Phi)$ to be satisfiable with a module $\lambda_i(T) \subset L_I$.

NOTE: This is equivalent to saying that an inference system is complete -- every sentence that is entailed by some First-Order KIF model of a module will be decided by the inference system to be satisfiable by the module.

8 Syntactic Conformance

This clause specifies semantic criteria for building inference systems that conform to the KIF standard.

8.1 Full Syntactic Conformance

An inference system I has full syntactic conformance with KIF if and only if it has simple semantic conformance with L_{KIF} .

8.2 Restricted Syntactic Conformance

An inference system I has restricted syntactic conformance with KIF if and only if there exists a sublanguage $L' \subset L_{KIF}$ such that I has simple semantic conformance with L' .

8.3 Expanded Syntactic Conformance

An inference system I has expanded syntactic conformance with KIF if and only if there exists a sublanguage $L' \subset L_I$ such that I has simple semantic conformance with L_{KIF} .

Annex A (normative) BNF Grammar for First-Order KIF

```

<alpha> ::= 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
          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
<marker> ::= @ | # | ? | ` | . | = | / | [ | : | %
<other> ::= ! | $ | % | & | * | + | , | - | | ; | ] | ^ | _ | { | } | ~ | > | '
<white> ::= <space> | <tab> | <return> | <linefeed> | <page>
<lexbreak> ::= <white> | ( | ) | " | \
<wordchar> ::= <alpha> | <digit> | <marker> | <other>
<stringchar> ::= <wordchar> | <white> | (|)
<docstring> ::= "<stringchar>*"

<word> ::= {<alpha> | <other>} <wordchar>
<wordbody> ::= <wordchar>+

<objvariable> ::= ?<word>

<sentop> ::= = | not | and | or | implies | iff |
           forall | exists

<constant> ::= <word> - <objvariable> - <sentop>

<objterm> ::= <objvariable> | <constant> | <fnterm>
<fnterm> ::= (<objterm> <objterm>*)
<sentence> ::= <atomsent> | <logsent> | <quantsent>
             | (<sentence> <docstring>)
<equalsent> ::= (= <objterm> <objterm>)

<atomsent> ::= (<objterm>+) | <equalsent>
<logsent> ::= (not <sentence>)|
             ({and|or} <sentence>*)|
             ({implies|iff} <sentence> <sentence>)
<quantsent> ::= ({forall|exists}(<objvariable>*) <sentence>)

<specialname> ::= %<wordbody> | documentation

```

```
<restriction> ::= :<word>  
<importname> ::= #<wordbody>  
<quote> ::= `<wordbody>'  
<docform> ::= (documentation [<word>] <docstring>)
```


Annex B (informative)

Relationship to Conventional First-Order Logic

KIF can be mapped into a more conventional logical language in two stages. We first describe an embedding from KIF-Core into a restricted KIF language which has a conventional clear distinction between object, function and relation symbols of fixed arity.

Holding and applying

To 'tidy up' the free syntax of KIF we utilize a familiar trick for writing 'higher-order' syntax into a first-order notation, which makes use of a denumerable set of special relations Holds-0 , Holds-1 , ... and function symbols App-0 , App-1 , ... (one for each natural number), and we simply rewrite every expression of the form

$$(t_o \ t_1 \ \dots \ t_n)$$

as

$$(\text{App-n } t_o \ t_1 \ \dots \ t_n)$$

if it is a functional term, and as

$$(\text{Holds-n } t_o \ t_1 \ \dots \ t_n)$$

if it is an atomic sentence, with these translations applied recursively to every subexpression in the obvious way. If θ (or λ_N) is an expression (or a language on N), let $H(\theta)$ (or $H(\lambda_N)$) be the expression (or language) defined by the above translation; we will call this a *holds expression* (or *language*).

In any holds language, all the symbols of the KIF language have become individual names; all the function, relation and individual names are distinct, and each has a unique arity; and no variables or nonatomic terms occur in the relation or function position.

Consider a holds language $H(\lambda_N)$, and a theory T written in $H(\lambda_N)$ without using row variables, and a conventional model-theoretic interpretation for this language over some domain D . Such an interpretation maps individual names to elements of D , n -ary function names to functional sets of $(n+1)$ -tuples over D , and n -ary relation symbols to sets of n -tuples over D . Given such a theory T with an interpretation I , it is easy to specify a KIF interpretation \hat{I} for λ_N such that ψ is true under \hat{I} just when $H(\psi)$ is true under I .

The domain of \hat{I} is D . We need to partition this into a subdomain of relations with nontrivial extensions, and a subdomain of nonrelational KIF individuals. Say that x is *relational* just when there is a term t_o in λ_N , $I(t_o) = x$, and T contains an expression or subexpression of the form

$(\psi \ t_o \ t_1 \ \dots \ t_n)$, ie the term t_o occurs immediately after a relation or function symbol; and say that $E = D \rightarrow R$. If $x \in R$, $\text{ext}(x)$ is defined to be set of rows r such that $cc(x,r) \in I(p)$ for some relation or function symbol p in $H(\lambda_N)$. It is then easy to check that this defines an KIF interpretation which is truth-preserving in the required sense. (Note that since a first-order interpretation always assigns functions (functional relations) to function symbols, the 'troublesome' case in the first clause of the KIF truth recursion does not arise.)

Model Theory with Non-Well-Founded Sets

Rather than use the extension function in the definition of structure, there is an alternative semantics for KIF-Core that uses the interpretation function of conventional first-order logic ([2], [3]). However, the existence of relations that can take relations as arguments leads to the use of non-well-founded set theory [1] within the semantics.

In the alternative semantics, a structure consists of a universe of discourse, an interpretation function, a semantic value function, and a satisfaction function. The semantic value function has the same properties as specified in 6.2.2. Within a structure, the interpretation function has the following properties:

- If s is a constant, then $\iota(s) \in O$;
- If s is a relation word, then $\iota(s) \subseteq O^*$ and $\iota(s) \in O$.

The satisfaction function for atomic sentences then has the following specification:

An atomic 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.

$$\tau((R t_1 \dots t_n)) = \begin{cases} true & \langle \sigma(t_1), \dots, \sigma(t_n) \rangle \in \iota(R) \\ false & otherwise \end{cases}$$

In the case of self-application (e.g. $(R R)$), this specification would be equivalent to $R \in \iota(R)$, which violates the axiom of foundation within ZF set theory, but is consistent with the axioms of ZF⁻.

Annex C
(informative)
Example KIF Sentences

The mother of Charles is either Elizabeth or Ann.

```
(forall (?x)
  (=> (mother Charles ?x)
      (or (= ?x Elizabeth)
          (= ?x Ann))))
```

Everyone's age must be greater than 0.

```
(forall (?x)
  (greater (age ?x) 0))
```

Nobody can be both a brother and a sister.

```
(forall (?x ?y)
  (=> (brother ?x ?y)
      (not (sister ?x ?y))))
```

Every person has a mother.

```
(forall (?x)
  (=> (person ?x)
      (exists (?y)
        (and (person ?y)
              (mother ?x ?y)))))
```

Two people are siblings if and only if they are brother or sister.

```
(forall (?x ?y)
  (=> (and (person ?x)
           (person ?y))
      (<=> (sibling ?x ?y)
           (or (brother ?x ?y)
               (sister ?x ?y)))))
```

Partial Orders

Nonlogical lexicon:

Constant symbols: none

Function symbols: none

Relation symbols: precedes

Axioms:

Reflexivity:

```
(forall (?x)
  (precedes ?x ?x))
```

Anti-symmetry:

```
(forall (?x ?y)
  (= > (and (precedes ?x ?y)
            (precedes ?y ?x))
      (= ?x ?y)))
```

Transitivity:

```
(forall (?x ?y ?z)
  (= > (and (precedes ?x ?y)
            (precedes ?y ?z))
      (precedes ?x ?z)))
```

Relations as Arguments to Relations

```
(forall (?r ?x ?y)
  (<=> (symmetric ?r)
      (<=> (?r ?x ?y)
          (?r ?y ?x))))
```

```
(forall (?r)
  (<=> (reflexive ?r)
      (forall (?x)
        (?r ?x ?x))))
```

Bibliography

[1]