

Common Logic: Abstract Syntax and Semantics

Common Logic Working Group

Introduction

The purpose of this document is to provide an abstract syntax for the Common Logic standard and a corresponding semantics. The idea here is to provide a syntax at a sufficiently high level of abstraction that it can be satisfied by many "concrete" languages — e.g., KIF, Conceptual Graphs, or the familiar *Principia*-style syntax found in most mathematical logic texts. In the development of a standard language and semantics for use in Ontology and Knowledge Engineering generally, one can take two approaches to this diversity. One approach is simply to legislate that one particular syntax shall be the standard. This, of course, would be the simplest solution, but quite likely the least effective, as it would marginalize those who prefer other languages, many of whom would not be inclined to change. The other approach is to specify a standard at a higher level of abstraction, which, at that level, can encompass all of the distinct concrete languages. CL has chosen the latter path; it does not privilege any one in a large class of concrete languages over another. Nevertheless, it still offers the advantages of the single language approach: Because all compliant languages are, at CL's level of abstraction, the "same" language, translation between those languages — guided by specifications detailing how each language exemplifies the abstract CL syntax — will be straightforward.

Syntax

Lexicon

A CL language is based upon an initial stock of primitive syntactic entities. Specifically, a CL *lexicon* λ consists of the following mutually disjoint sets:

- A countable set *Con* called the *constants* of λ .
- A denumerable set *OrdVar* called the *ordinary variables* of λ ;
- A (possibly empty) set *SeqVar* called the *sequence variables* of λ . If nonempty, *SeqVar* is denumerable.¹

We stipulate that Con , $OrdVar$ and $SeqVar$ be pairwise disjoint. $Var = OrdVar \cup SeqVar$ is known as the set of *variables* of λ . Let $PrimTrm = Con \cup Var$; $PrimTrm$ is known as the set of *primitive terms* of λ .

Note that, unlike traditional logical languages, there is no distinction between individual constants and predicates in CL lexicon. This reflects the underlying perspective that everything is a "first-class" logical citizen.

CL Languages

Every CL lexicon determines a class of languages. Structurally speaking, these languages are all identical, insofar as they all satisfy the same abstract conditions. They differ only in the details of their outward form. We will illustrate this below.

Term Classes

For any set M , let M^* be the set of finite sequences of elements of M , i.e., $M^* = \bigcup_{0 \leq n < \omega} M^n$, where M^n is the set of all n -tuples of elements of M .

Say that T is a *term class* for λ if T is a smallest class that includes the primitive terms of λ and is closed under an operation **App** — called a *generator* for T — such that

- **App** is one-to-one;
- **App** : $(T - SeqVar) \times T^* \rightarrow T$.²

For any term class Trm for λ , let $FuncTrm = Range(\mathbf{App})$,³ where **App** is a generator for Trm , let $FuncTrm$ is the set of *function terms* of λ (relative to Trm), and $SingTrm$ is the class of *singular terms* of λ (relative to Trm).

Formula classes

Let λ be a CL lexicon, and let Trm be a term class for λ . Let $FuncTrm$ be the class of function terms of λ relative to Trm and $SingTrm$ the class of singular terms of λ relative to Trm . Say that F is a *formula class* for λ , relative to Trm , if it is a smallest class that includes $FuncTrm$ and is closed under a set Op — known as a *generator set* for F — of operations **Id**, **Neg**, **Conj**, **Disj**, **Cond**, **Bicond**, **ExQuant**, **UnivQuant** that satisfy the following conditions:

- Each operation is one-to-one;

- The ranges of the operations are pairwise disjoint, and disjoint from Trm
- **Id** : $SingTrm \times SingTrm \rightarrow F$
- **Neg** : $F \rightarrow F$
- **Conj** : $F^* \rightarrow F$
- **Disj** : $F^* \rightarrow F$
- **Cond** : $F \times F \rightarrow F$
- **BiCond** : $F \times F \rightarrow F$
- **ExQuant** : $(Var \cup (Var \times Con))^* \times F \rightarrow F$
- **UnivQuant** : $(Var \cup (Var \times Con))^* \times F \rightarrow F$

The clauses for "quantified" formulas here, i.e., those entities in the ranges of **ExQuant** and **UnivQuant**, allow for the use of restricted quantifiers. Thus, in a KIF-like implementation of CL, these clauses would allow such formulas as

```
(forall (?x Boy)
  (exists (?y Girl)
    (Kissed ?x ?y)))
```

to represent "Every boy kissed a girl."

Note that, because the operations in a generator set for a formula class Fla for λ are all one-to-one and disjoint in their ranges, every element of Fla will have exactly one "decomposition" under the inverses of those operations, and that all such decompositions are finite. [4](#)

Let $\phi \in Fla$. An object ε in the decomposition of ϕ is an *atom* of ϕ just in case ε is element of the lexicon λ . ϕ is *atomic* if $\phi \in FuncTrm$. ψ is a *subformula* of ϕ if $\psi \in Fla$ and ψ is in the decomposition of ϕ .

We define a CL *language* to be a formula class Fla for a CL lexicon λ . If $\mathbf{L} = Fla$ is a CL language, where Fla is formula class for λ relative to Trm , then we say that \mathbf{L} is a language *for* λ (alternatively, that λ *underlies* \mathbf{L}), and we say that Trm is the set of *terms* of \mathbf{L} . \mathbf{L} is said to be *first-order* if $SeqVar = \emptyset$.

- If λ and λ' are CL lexicons with the same sets of constants and \mathbf{L} and \mathbf{L}' are CL languages for λ and λ' , respectively, then \mathbf{L} and \mathbf{L}' are said to be *equivalent*.

Say that a CL Abstractly speaking, this sense exemplifies the following CL structure.

Semantics

Interpretations

Let \mathbf{L} be a CL language for λ , and let Trm be the terms of \mathbf{L} . A *CL interpretation* \mathbf{I} for \mathbf{L} is a 4-tuple $\langle D, ext, skol, V \rangle$ satisfying the following conditions. First, $D = A \cup R$ is a nonempty set such that $A \cap R = \emptyset$. Intuitively, A represents the set of *individuals* of \mathbf{I} and R the set of relations-in-intension (though, as will be seen, it is possible to think of them extensionally as sets). ext is a function from D into D^* , though we stipulate that, for all $a \in A$, $ext(a) = \emptyset$. Intuitively, ext represents the extension of every relation. However, to smooth the semantics for the highly unrestricted syntax of CL languages, in which there is no syntactic distinction between individual constants and predicates, the individuals of D are assigned extensions as well, albeit always empty ones — thus, the result of predicating one individual of another, while semantically meaningful, will always yield falsehood.

The purpose of $skol$ is to enable the definition of reasonable denotations for function terms when the term occurring in function position in such a term does not denote a total function (i.e., denotes either a partial function, a non-functional relation, or an individual). Toward that end, for $\langle e_1, \dots, e_n \rangle \in D^*$, say that an object $e \in D$ is *defined on* $\langle e_1, \dots, e_n \rangle$ in I just in case there is at least one object $e_0 \in D$ such that $\langle e_0, e_1, \dots, e_n \rangle \in ext(e)$. Define an *I-skolemization* of e to be any (total) function $f: D^* \rightarrow D$ such that, for all $\langle e_1, \dots, e_n \rangle$ on which e is defined we have $\langle f(e_1, \dots, e_n), e_1, \dots, e_n \rangle \in ext(e)$.⁵

Note that (given standard ZFC set theory) every element of D has a skolemization. The object $skol$ in our interpretation $I = \langle D, ext, skol, V \rangle$, then, is stipulated to be a function that maps every element e in the domain D of I to an *I-skolemization* of e . This mechanism enables one to use a term denoting a functional relation as both a predicate and a function symbol with the intuitively correct result. Thus, for example, in a *Principia*-style CL language, one could correctly assert both 'FatherOf(Adam,Cain)' and 'Gardener(FatherOf(Cain))'. This approach, of course, does have as a consequence that certain expressions that are intuitively nonsensical — e.g., 'Gardener(FatherOf(17))' — turn out to be meaningful, and even possibly true. However, this has little practical upshot, as such "unintended" truths are easily rendered innocuous by a proper formulation of one's axioms.⁶

Finally, V is a "denotation" function that assigns appropriate values to the primitive terms of \mathbf{L} . Specifically, if $\tau \in \text{Con} \cup \text{OrdVar}$, then $V(\tau) \in D$, and if $\tau \in \text{SeqVar}$, then, $V(\tau) \in D^*$.

Denotations and Truth

Given the notion of an interpretation, we can now define what it is for a formula of a CL language to be *true* in an interpretation. First, we need some notation. For n -tuples s_1, \dots, s_n , let $\text{cnct}(s_1, \dots, s_n)$ be the concatenation of s with s' .⁷ This definition is useful for defining semantic values for function terms.

Further notation will be useful in defining truth for quantified formulas (i.e., formulas in the range of **ExQuant** and **UnivQuant**). First, for $\chi \in (\text{Var} \cup (\text{Var} \times \text{Con}))$, let $|\chi| = \chi$ if $\chi \in \text{Var}$, and if $\chi = \langle \nu, \kappa \rangle \in (\text{Var} \times \text{Con})$, let $|\chi| = \nu$. Now let $\mathbf{I} = \langle D, \text{ext}, \text{skol}, V \rangle$ be an interpretation for \mathbf{L} . For $\chi_1, \dots, \chi_n \in (\text{Var} \cup (\text{Var} \times \text{Con}))$, say that V' is a $[\chi_1, \dots, \chi_n]$ -variant of V iff

- $V'(\nu) \in \text{ext}(V(\kappa))$, if $\chi_i = \langle \nu, \kappa \rangle$;
- $V'(\tau) = V(\tau)$, if $\tau \neq |\chi_i|$ for any i , $1 \leq i \leq n$.

We then say that an interpretation $\mathbf{I}' = \langle D, \text{ext}, \text{skol}, V' \rangle$ for \mathbf{L} is a $[\chi_1, \dots, \chi_n]$ -variant of \mathbf{I} iff V' is a $[\chi_1, \dots, \chi_n]$ -variant of V .

In plain but somewhat less precise English, a $[\chi_1, \dots, \chi_n]$ -variant \mathbf{I}' of \mathbf{I} is an interpretation that is just like \mathbf{I} except that its denotation function V' possibly assigns something else to each $|\chi_i|$; in particular, in the case where χ_i is a pair $\langle \nu, \kappa \rangle \in (\text{Var} \times \text{Con})$, V' assigns a member of the extension of $V(\kappa)$ to ν . This somewhat complicated definition enables the formulation of the very simple definition of truth for quantified formulas below.

So let \mathbf{L} be a CL language for a lexicon λ , Trm the set of terms of \mathbf{L} , and **App** a generator for Trm , and let $\mathbf{I} = \langle D, \text{ext}, \text{skol}, V \rangle$ be an interpretation for \mathbf{L} . Given \mathbf{I} , the denotations of the non-primitive terms of \mathbf{L} in \mathbf{I} are completely determined by skol and V . This can be expressed in terms of a unique extension $V^\#$ of V such that, for any term $\tau \in \text{Trm}$:

- If $\tau \in \text{Con} \cup \text{Var}$, then $V^\#(\tau) = V(\tau)$.
- If $\tau \in \text{SeqVar}$, then $V^{\#\#}(\tau) = V^\#(\tau)$, otherwise $V^{\#\#}(\tau) = \langle V^\#(\tau) \rangle$.
- If $\tau = \mathbf{App}(\varepsilon, \tau_1, \dots, \tau_n) \in \text{FuncTrm}$, then $V^\#(\tau) = \text{skol}(V^\#(\varepsilon))(e_1, \dots, e_m)$, where $\langle e_1, \dots, e_m \rangle = \text{cnct}(V^{\#\#}(\tau_1), \dots, V^{\#\#}(\tau_n))$.

The purpose of $V^{\#\#}$ is simply to enable us to form proper sequences out of the semantic values of sequences of terms, some of which denote objects in D and others of which (viz., the sequence variables) denote sequences of such objects. For example (in a *Principia*-style CL language again), suppose 'moved' denotes a 3-place relation that holds between an object o and locations p_1 and p_2 just in case o has moved from p_1 to p_2 . Thus, in an interpretation where $V('a')$ is o and $V('@r')$ is the pair $\langle p_1, p_2 \rangle$ ('@r' here being a sequence variable), we want 'moved($a, @r$)' to be true just in case the triple $\langle o, p_1, p_2 \rangle \in \text{ext}(\text{'moved'})$. We accomplish this simply by formulating the semantic clause in terms of the concatenation of sequences. But for this we need, not the object o itself, but its singleton sequence $\langle o \rangle$. And thus 'moved($a, @r$)' is true in our interpretation just in case $\text{cnct}(V^{\#\#}('a'), V^{\#\#}('@r')) = \text{cnct}(\langle o \rangle, \langle p_1, p_2 \rangle) = \langle o, p_1, p_2 \rangle \in \text{ext}(\text{'moved'})$.

Given V , we define truth for the formulas of \mathbf{L} in our interpretation \mathbf{I} as follows. Let $\phi \in \mathbf{L}$.

- If $\phi = \mathbf{App}(\varepsilon, \tau_1, \dots, \tau_n) \in \text{FuncTrm}$, then ϕ is true in \mathbf{I} iff $\text{cnct}(V^{\#\#}(\tau_1), \dots, V^{\#\#}(\tau_n)) \in \text{ext}(V^\#(\varepsilon))$.
- If $\phi = \mathbf{Id}(\tau, \tau')$, then ϕ is true in \mathbf{I} iff $V(\tau) = V(\tau')$.
- If $\phi = \mathbf{Neg}(\psi)$, then ϕ is true in \mathbf{I} iff ψ is not true in \mathbf{I} .
- If $\phi = \mathbf{Disj}(\psi_1, \dots, \psi_n)$, then ϕ is true in \mathbf{I} iff ψ_i is true in \mathbf{I} for some $i, 1 \leq i \leq n$.
- If $\phi = \mathbf{Conj}(\psi_1, \dots, \psi_n)$, then ϕ is true in \mathbf{I} iff ψ_i is true in \mathbf{I} for each $i, 1 \leq i \leq n$.
- If $\phi = \mathbf{Cond}(\psi, \psi')$, then ϕ is true in \mathbf{I} iff ψ is not true in \mathbf{I} or ψ' is true in \mathbf{I} .
- If $\phi = \mathbf{BiCond}(\psi, \psi')$, then ϕ is true in \mathbf{I} iff ψ and ψ' are either both true in \mathbf{I} or both false in \mathbf{I} .
- If $\phi = \mathbf{ExQuant}(\chi_1, \dots, \chi_n, \psi)$, then ϕ is true in \mathbf{I} iff ψ is true in some $[\chi_1, \dots, \chi_n]$ -variant of \mathbf{I} .

- If $\phi = \mathbf{UnivQuant}(\chi_1, \dots, \chi_n, \psi)$, then ϕ is true in \mathbf{I} iff ψ is true in all $[\chi_1, \dots, \chi_n]$ -variants of \mathbf{I} .

Note that this semantics requires no "pruning" to serve as an appropriate semantics for first-order CL languages, as interpretations themselves involve no non-first-order entities; given an interpretation $I = \langle D, ext, skol, V \rangle$ the range D^* of the sequence variables itself lies outside of I proper. If \mathbf{L} is first-order, the clauses for formulas containing sequence variables simply never kick in.

KIF: A Paradigmatic CL Language

CL is in fact an outgrowth of the KIF project. CL simply abstracts away from the particularities of KIF — its choice of basic syntactic elements, its use of parenthesis, its choice of keywords, etc. A full CL language will therefore be structurally identical to a KIF language, but might differ in its surface form. Not surprisingly, then, it is rather straightforward to demonstrate that any KIF language is also a CL language. For purposes here, we will take KIF expressions to be strings.⁸ Let A be the usual set of alphanumeric characters together with the characters "-" and "_". Our first task is to specify a lexicon for a KIF language. Although the syntax for KIF is now based more generally upon unicode, for purposes here we can take the set Con of constants of a lexicon to be a set of (finite) strings over A , minus the usual KIF keywords (i.e., "forall", "exists", "not", etc.). $OrdVar$ consists of the result of prefixing "?" to all strings over A , and $SeqVar$ the result of prefixing "@" to those same strings. Clearly, Con , $OrdVar$, and $SeqVar$ are all mutually disjoint, and the sets $OrdVar$ and $SeqVar$ are denumerable. Hence, Con , $OrdVar$, and $SeqVar$ jointly constitute a CL lexicon. Relative to this lexicon, then, Var is the set $OrdVar \cup SeqVar$ and $PrimTrm$ is the set $Con \cup Var$.

Consider now the set S of strings over the set $A \cup \{ "?", "@", " ", "(", ")" \}$ (" " being the space character). Define an operation \mathbf{App}^* on $(S - SeqVar) \times S^*$ such that $\mathbf{App}^*(t, s_1, \dots, s_n) = [t s_1 \dots s_n]$ ("[" and "]" here are quasi-quotes, or "Quine corners".) Let Trm be the smallest set containing $PrimTrm$ that is closed under \mathbf{App}^* , and let \mathbf{App} be \mathbf{App}^* restricted to Trm . \mathbf{App} is easily seen to be a generator for Trm , and that Trm is the set of KIF function terms that is determined by the given lexicon.

Consider now the following operations:

- $\mathbf{Id}^*(s_1, s_2) = [= s_1 s_2]$ for $s_1, s_2 \in S$;
- $\mathbf{Neg}^*(s) = [\text{not } s]$ for $s \in S$;
- $\mathbf{Conj}^* = [\text{and } s_1 \dots s_n]$ for $s_1, \dots, s_n \in S$;

- $\mathbf{Disj}^* = \{\text{or } s_1 \dots s_n\}$ for $s_1, \dots, s_n \in S$;
- $\mathbf{Cond}^*(s_1, s_2) = \{\text{if } s_1 \ s_2\}$ for $s_1, s_2 \in S$;
- $\mathbf{BiCond}^*(s_1, s_2) = \{\text{iff } s_1 \ s_2\}$ for $s_1, s_2 \in S$;
- $\mathbf{ExQuant}^*(v_1, \dots, v_n, s) = \{\text{exists } (v_1^* \dots v_n^*) \ s\}$ for $v_1, \dots, v_n \in (Var \cup (Var \times Con))$, $s \in S$, where v_i^* is v_i if $v_i \in Var$, and v_i^* is $\{x \ c\}$ if $v_i = \langle x, c \rangle \in Var \times Con$;
- $\mathbf{UnivQuant}^*(v_1, \dots, v_n, s) = \{\text{forall } (v_1^* \dots v_n^*) \ s\}$ for $v_1, \dots, v_n \in (Var \cup (Var \times Con))$, $s \in S$, where v_i^* is as above.

Let \mathbf{F} be the smallest set containing $FuncTrm (= Range(\mathbf{App}))$ that is closed under these operations. \mathbf{F} is exactly the set of formulas of the KIF language determined by the given lexicon, i.e., the above operations, restricted to \mathbf{F} , jointly constitute a generator set for \mathbf{F} .

Conformance

CL languages are both syntactically anarchic (hence rather non-traditional) and quite powerful expressively. In many contexts, one might desire to use a language that is more structured, less expressive, or both — as is the case with most familiar first-order languages. If conformance with CL required one to adopt both the anarchic syntax of CL languages and its full expressive power, it is unlikely that CL would receive wide acceptance. Hence, conformance comes in two varieties: direct, for languages that fully embrace both CL's anarchic syntax and its semantic expressiveness, and indirect, for more languages that are more constrained syntactically, semantically, or both.

Direct Conformance

A given language is *directly conformant* with the CL standard just in case it is an instance of a CL language. Hence, as just illustrated, to demonstrate that a language \mathbf{L}^* is directly conformant, one must simply:

1. Specify the sets constituting the lexicon λ that underlies \mathbf{L}^* — specifically, the sets of constants, ordinary variables, and sequence variables of λ — and demonstrate that they are pairwise disjoint;
2. Specify the operation that serves as the generator for the set of terms of \mathbf{L}^* ;
3. Specify the operations in the generator class for the relevant formula class for λ that we identify with \mathbf{L}^* .

Indirect Conformance

As indicated, more constrained languages can still be thought of as CL conformant in a robust, well-defined sense. Specifically, where \mathbf{L} is a CL language, we define a subset $\mathbf{L}' \subseteq \mathbf{L}$ to be *indirectly conformant relative to \mathbf{L}* just in case \mathbf{L}' is recursive. Hence, in general, to show that a given language \mathbf{L}' (simply a set of formulas, recall) is indirectly conformant, one must simply provide a recursive specification of \mathbf{L}' relative to some CL language \mathbf{L} . Typically, of course, a specification of an indirectly conformant language will be provided independent of any specific CL language.

The class of interpretations for an indirectly conformant language \mathbf{L}' can simply be thought of as identical to the class of interpretations for its CL superlanguage \mathbf{L} where the *true-in* relation is restricted to \mathbf{L}' . However, typically, once again, it is assumed that the notion of interpretation can be defined for \mathbf{L}' independently of any directly conformant language.

A language will be said to be *CL conformant* (*conformant*, for short) just in case it is either a CL language or indirectly conformant relative to some CL language.

Example: The Language of Traditional FOL

The traditional language of first-order logic, relative to some choice of constants, is a prime example of an indirectly conformant language. Specifically, let \mathbf{L} be a CL language for some lexicon λ such that $SeqVar = \emptyset$. Let *p-arity* and *f-arity* be total recursive functions from disjoint subsets *Pred*, *Func* of *Con* into the set of positive integers. Let $\tau \in Con$. We say that τ is an *n-place predicate* if *p-arity*(τ) = *n*, and that τ is an *n-place function symbol* if *f-arity*(τ) = *n*. Otherwise we say that τ is an *individual constant*. Let $IndCon = Con - (Pred \cup Func)$. Let *Trm** be the smallest set containing *IndCon* and *OrdVar* and containing $\mathbf{App}(\alpha, \tau_1, \dots, \tau_n)$ whenever α is an *n*-place function symbol and $\tau_1, \dots, \tau_n \in Trm^*$. Let *arity* be *p-arity* \cup *f-arity*. We then define the *traditional first-order (TFO) sublanguage \mathbf{L}_1* of \mathbf{L} determined by *arity* to be those formulas ϕ of \mathbf{L} that satisfy the following two conditions:

- All of the atomic subformulas of ϕ are of the form $\mathbf{App}(\pi, \tau_1, \dots, \tau_n)$, where π is an *n*-place predicate and each $\tau_i \in Trm^*$; and
- If $\phi = \mathbf{ExQuant}(\chi_1, \dots, \chi_n, \psi)$ or $\phi = \mathbf{UnivQuant}(\chi_1, \dots, \chi_n, \psi)$ and $\chi_i = \langle \nu, \kappa \rangle$, then κ is a 1-place predicate of \mathbf{L}_1 .

L_1 is clearly indirectly conformant.

First-order CL and Traditional First-order Languages

In this section we clarify the relation between first-order CL (henceforth, "CL₁") languages and more traditional first-order languages. We will show that CL₁ languages and their FO counterparts are equally expressive in a well-defined sense.

Traditional First-order Languages and Their Model Theory

We will exploit the notion of a TFO sublanguage of a CL language to clarify the connection between traditional first-order logic and first-order CL. Specifically, we will clarify the connection between a CL₁ language and its TFO counterparts. There is no loss of generality here, since any first-order language can be thought of as a TFO sublanguage of some CL language.

We begin by altering our general CL semantics slightly to define a variant of traditional first-order model theory for TFO sublanguages. Thus, let L be a CL language and L_1 be a traditional first-order sublanguage of L

determined by *arity*. An interpretation $I = \langle D, ext, skol, V \rangle$ for L is L_1 -agreeable just in case the following conditions hold (where, as usual, $D = A \cup R$):

- For all n -place predicates and $(n-1)$ -place function symbols β of L_1 , $V(\beta) \in R$ and $ext(V(\beta))$ is a set of n -tuples over A . (Intuitively, this condition ensures that n -place predicates and $(n-1)$ -place function symbols are all mapped to n -place relations that hold among individuals.)
- For all n -place function symbols α of L_1 , $ext(V(\alpha)) = skol(V(\alpha))|A^n$, i.e., the restriction of $skol(V(\alpha))$ to A^n . (This condition ensures that the relation to which α is mapped is a total n -place function on individuals.)
- For all $\kappa \in IndCon \cup OrdVar$, $V(\kappa) \in A$.

It is nearly trivial to transform an L_1 -agreeable CL interpretation into a more traditional first-order interpretation. The definition of what it is for a formula ϕ of L_1 to be $true_1$ in an L_1 -agreeable interpretation $\mathbf{I} =$

$\langle D, ext, skol, V \rangle$ of L is identical with the definition of truth in \mathbf{I} except for

the quantificational clauses, which differ slightly. Specifically, say that a $[\chi_1, \dots, \chi_n]$ -variant V' of V is L_1 -agreeable iff $V'(\chi_i) \in A$ whenever χ_i is a variable.⁹ Then:

- If $\phi = \mathbf{ExQuant}(\chi_1, \dots, \chi_n, \psi)$, then ϕ is true in \mathbf{I} iff ψ is true in some L_1 -agreeable $[\chi_1, \dots, \chi_n]$ -variant of \mathbf{I} .
- If $\phi = \mathbf{UnivQuant}(\chi_1, \dots, \chi_n, \psi)$, then ϕ is true in \mathbf{I} iff ψ is true in all L_1 -agreeable $[\chi_1, \dots, \chi_n]$ -variants of \mathbf{I} .

ϕ is *valid*₁ iff it is true₁ in all L_1 -agreeable interpretations. It is easy to see that the valid₁ formulas of L_1 are exactly the traditional first-order validities.

Interpreting TFO Sublanguages in CL_1

In this section we show that TFO languages are interpretable in CL in two very natural senses, one interpretation-relative and the other interpretation-independent.

Interpretability I: Interpretability Relative to an Interepretation

The first notion of interpretability plays off the fact that, relative to an L_1 -agreeable interpretation, the definition of truth₁ differs from the definition of truth only in the range of the quantifiers. This suggests the following set of definitions.

Let L be a CL_1 language – for definiteness, let us assume KIF syntax – and let L_1 be the TFO sublanguage of L generated by an arity function *arity*.

Let L^+ be the result of adding a single predicate 'Ind' to L . For any

L_1 -agreeable interpretation $\mathbf{I} = \langle D, ext, skol, V \rangle$ for L (where, as usual, $D = A \cup R$), let $\mathbf{I}^+ = \langle D^+, ext^+, skol^+, V^+ \rangle$ be any interepretation such that

- $D^+ = A \cup R^+$, where R^+ is the result of adding a single element *Ind* to R ;
- $ext \subseteq ext^+$ and $ext^+(ind) = A$;
- $skol \subseteq skol^+$;
- $V^+(\text{Ind}) = Ind$.

Now define a simple mapping $+$ from the sentences of L_1 into those of L^+ as follows:

- $\phi^+ = \phi$, for atomic formulas ϕ ;
- $(\text{not } \phi)^+ = (\text{not } \phi^+)$;
- $(\text{and } \phi \ \psi)^+ = (\text{and } \phi^+ \ \psi^+)$; similarly for the remaining boolean operators;
- $(\text{exists } (v_1 \dots v_n) \ \phi)^+ = (\text{exists } (v_1^* \dots v_n^*) \ \phi^+)$, where v_i^* is $\lfloor v_i \text{ Ind} \rfloor$ if v_i is a variable, and v_i^* is v_i otherwise (i.e., if v_i is a variable/constant pair $\lfloor x \ c \rfloor$);
- $(\text{forall } (v_1 \dots v_n) \ \phi)^+ = (\text{forall } (v_1^* \dots v_n^*) \ \phi^+)$, where v_i^* is as above.

So, for example, let ϕ be the sentence

```
(forall (?x (?d Direction) (?t1 Time) (?t3 Time))
  (if (and (moving-in ?x ?d ?t1)
           (not (exists (?y (?t2 Time))
                        (and (between ?t2 ?t1 ?t3)
                             (acts-on ?y ?x ?t2))))))
      (moving-in ?x ?d ?t3)))
```

expressing the proposition that if an object is moving in a certain direction at a given time t_1 then it is still moving in that direction at a later time t_3 if nothing acts upon it at any intervening time. Then ϕ^+ is

```
(forall ((?x Ind) (?d Direction) (?t1 Time) (?t3 Time))
  (if (and (moving-in ?x ?d ?t1)
           (not (exists ((?y Ind) (?t2 Time))
                        (and (between ?t2 ?t1 ?t3)
                             (acts-on ?y ?x ?t2))))))
      (moving-in ?x ?d ?t3)))
```

Given our two $+$ functions we have the following simple theorem: Let \mathbf{I} be any L_1 -agreeable interpretation for L . Then for any L_1 sentence ϕ , ϕ is true $_1$ in \mathbf{I} iff ϕ^+ is true in \mathbf{I}^+ . The proof is a simple induction on the formulas of L_1 .

Interpretability II: TFO Languages as CL_1 Theories

The function *arity* that generates a TFO sublanguage L_1 of a CL_1 language L separates the syntactically undifferentiated constants of L into n -place predicates, n -place function symbols, and individual constants. This partitioning of the constants explicitly reflects the intuitive, but *typed*, ontology of traditional first-order logic that is expressed in its semantics – relations (including functions) and individuals. CL_1 basically shares the same ontology, albeit *untyped*: relations are first-class logical citizens. We can therefore make the implicit ontology of TFO languages in L despite its undifferentiated constants simply by extending L to a new language L^{++}

by introducing appropriate new constants and interpreting them in a way that parallels the semantics of traditional first-order logic. And we can capture its typed character by suitably restricting the quantifiers, as in the first notion of interpretability. Moreover, we can *axiomatize* this ontology explicitly as a theory of L^{++} . Specifically, we show that, for any TFO sublanguage L_1 of a CL_1 language L , there is a theory in a "natural" extension L^{++} of L whose models are exactly (slightly modified versions of) the class of L_1 -agreeable interpretations.

[Details forthcoming...]

Intepreting CL_1 Languages as TFO Theories

In logic, as in life generally, turnabout is fair play. Just as we can "translate" any given TFO language into a CL_1 theory, so we can translate any CL_1 language into a theory in a TFO language.

[Details forthcoming...]

Notes

1. If $SeqVar$ is empty, then languages built on λ will be first-order. We allow $SeqVar$ to be empty to allow first-order languages to be fully conformant without embracing sequence variables.
2. T is a smallest such class just in case it satisfies the conditions in question but is not a proper superclass of any other class satisfying those conditions.
3. Where $Range(\mathbf{App}) = \{ \tau \in Trm : \text{for some } x, \mathbf{App}(x) = \tau \}$.
4. Suppose $\phi \in Fla$ has an infinite decomposition. Then $Fla - \{ \phi \}$ still satisfies the conditions above, as closure under the operations in the generator set only requires finite compositions of those operations applied to the base sets. But then Fla is not a smallest class of the relevant sort, contrary to its definition. This will be important for the semantics of CL languages below.
5. We identify the first rather than the last element of an n-tuple in the extension of a function with the value of the function, as this increases the usability of an important sublanguage of any CL language. This will be discussed below.
6. Note that, on this semantics, ' $(\exists x)F(x,y)$ ' entails, but is not entailed by, ' $(\exists x)x = F(y)$ ' (indeed, as in classical first-order logic, the latter is a logical

truth of our semantics). Hence, perhaps somewhat paradoxically at first sight, from the fact that something is identical to the *FatherOf* 17, it doesn't follow that 17 has a father. Note, however, that this is going to be an issue for any language that

1. Includes a classical quantification theory;
2. Includes in its intended domain objects of vastly different ontological categories; and
3. Contains terms that intuitively express functions which only sensibly apply to the objects of some ontological categories but not others.

7. More exactly, where $s = \langle e_1, \dots, e_n \rangle$ and $s' = \langle e_{n+1}, \dots, e_{n+m} \rangle$,
 $cnct(s, s') = \langle s_1, \dots, s_{n+m} \rangle$.

8. It is perhaps most natural to take the expressions of KIF to be strings, but other construals (e.g., sequences) are of course possible.

9. We only need to consider this case because when $\chi_i = \langle \nu, \kappa \rangle$ we are already guaranteed by the definitions of a $[\chi_1, \dots, \chi_n]$ -variant and an L_1 -agreeable interpretation that $V'(\nu) \in A$.

