Completeness Theorems for Logic with a Single Type

Christopher Menzel

(Under review at *Review of Symbolic Logic*)

1 Common Logic

Common Logic (*CL*) is a standardized logical framework that arose out of the need to represent and share information on high-speed computer networks, the World Wide Web in particular. Arguably, however, its non-standard syntactic and semantic features have rather considerable philosophical significance (Menzel 2011) — notably, CL languages, or *dialects*, contain only a single lexical type the members of which all denote objects of a single semantic type in any interpretation. The Common Logic standard itself (ISO/IEC JTC 1 SC 32 2007[1]) makes solid headway in placing CL on a solid theoretical footing, but it leaves a considerable gap: it provides the syntax and semantics of CL dialects only, leaving the choice of proof theory up to the user.[2] However, because of its non-standard features — notably, a number of features characteristic of second-order languages — it is important to verify that CL dialects can be outfitted with a sound and complete proof theory. That is the central purpose of this paper. In fact, two completeness theorems will

[1] Authorship of an ISO/IEC document is typically attributed to an otherwise anonymous subcommittee of the overseeing "joint technical committee" — hence the moniker "JTC 1 SC 32". The present author was an active member of SC 32 for several years and was intimately involved in the development of the CL standard, although the bulk of ISO/IEC document itself was written by Dr. Patrick Hayes.

[2] There are several reasons for this. First, although it contains a natural first-order fragment (which shall be the focus in this paper), full CL in fact exceeds first-order logic in expressive power and, hence, full CL-validity is not recursively axiomatizable. Second, the primary motivation for CL is to serve as a general framework for representing and exchanging information rather than for theorem proving. Relatedly, third, in the typical case, information represented or exchanged in the context of the Web will be reasoned upon by automated systems based upon logics that are computationally decidable and, hence, less powerful than full first-order logic (cf., e.g., Horrocks et al. 2005). It was therefore thought best to avoid including a proof theory even for the first-order fragment, so as to avoid the marking implementations of CL that adopt a weaker proof theory as non-conformant.

be provided for CL — a "direct" theorem that follows the contours of a typical Henkin completeness theorem, albeit modified to accommodate the novel features of CL; and an "indirect" theorem that proceeds by way of a translation scheme from an arbitrary CL language into a "traditional" first-order language. Completeness then follows straightaway from the completeness of traditional first-order logic by way of a few simple theorems that demonstrate that the translation scheme "preserves meaning" in an appropriate sense.

Because the syntax for CL is stated so generally in order to accommodate a wide variety of dialects, it would be both very awkward and theoretically superfluous to try to define a proof theory and prove a completeness theorem in comparably general terms. We therefore will work in a particular dialect that is in fact included in an appendix to the ISO standard, the Common Logic Interchange Format, or CLIF — so-called because it is a descendent of, KIF, the Knowledge Interchange Format (Genesereth 1998).[3] CLIF is a particularly good choice because it is well-suited to the non-traditional syntactic features afforded by CL. CLIF will initially be outfitted with a more or less standard-looking logical system. However, the syntactic features of CL, as manifested in CLIF, suggest two natural, easily axiomatized extensions to the semantics that will introduced in §6 below. Completeness (by the first of our two methods) will be proved for the extended semantics as well.

As the motivation and significance of CL's syntax is perhaps not obvious on its face, the following section consists entirely of an informal discussion of its distinctive features.[4] Additionally, the final (sub)section of the paper contains some brief reflections on the connection between CL and traditional first-order logic.

2 CL's Distinctive Features and Their Motivation

CL's most distinctive features can be summed up reasonably well as: *type-freedom*, *variable polyadicity*, and *"higher-order" quantification*.

[3] KIF is in fact still broadly used, not only because it was developed at Stanford in the 1980s by an influential group of AI researchers who promoted its use, but also simply because of the ease with which KIF statements can be constructed by means of an ASCII keyboard and hence easily distributed by standard ASCII-based communication protocols like email.

[4] For motivations more directly related to computational issues in artificial intelligence, database theory, and the like, see Chen et al. 1993. Entirely independent of CL, Chen and his colleagues developed a logic — HiLog — whose syntax and semantics are quite similar to CL's but whose purpose was to serve as a more flexible foundation for logic programming than traditional predicate logic.

Type-freedom. In general, a logic is type-free, in some respect, if it does not heed one or another traditional division into strict syntactic or semantic types, or categories. For example, the individual constants, n-function symbols, and n-place predicates of a language are typically considered to constitute distinct, mutually disjoint lexical categories and, hence, are considered to be different syntactic types.[5] Concomitantly, the semantic values of the members of those types are traditionally drawn from domains D and F_n and R_n — individuals, n-place functions, and n-place relations, respectively. As R_n is typically taken to be a subset of $\wp(D^n)$ and F_n a subset of $\wp(D^{n+1})$, n-place functions and might overlap with $n+1$-place relations. But, at the least, n-place functions (for $n > 0$) and n-place relations jointly constitute semantic types that are distinct from the type of individuals.

However, limited forms of type freedom, at least, are well-warranted by natural language and by common practices in knowledge representation. Many knowledge representation (KR) systems presuppose a hierarchical ontology of classes that correspond semantically to predicates. At the same time, classes are often considered (first-order) individuals which can themselves be subjects of predication, suggesting a breach in the traditional division between the semantic values of individual constants and predicates. Moreover, many KR systems have a top level class ENTITY which is itself considered an entity. Hence, 'ENTITY is an ENTITY', or the like, is typically taken to be a theorem of these systems, suggesting a breach in the syntactic division between individual constants and predicates as well.

Likewise, a significant measure of semantic type freedom is warranted by the ubiquitous phenomenon of *nominalization* in natural language, whereby noun phrases are generated from adjectives and verb phrases via, notably, gerundive constructions (e.g., *wise* $\longrightarrow$ *being wise*, *runs* $\longrightarrow$ *running)*. Intuitively, a predicable expression — an adjective or verb phrase — and its nominalized counterpart have the same semantic value: an individual is wise just in case being wise is among her properties; she runs just in case she engages in running. But nominalizations are singular terms. Hence, to represent the information expressed by nominalization correctly, the strict division between the semantic values of predicates and denoting expressions must be breached. Moreover, differences in surface grammatical form aside, since they have the same meaning there is no logical reason to represent them by means of the same lexical item and, hence, the corresponding lexical division between predicates and singular terms breaks down as well.[6]

[5]Individual constants can of course be identified with 0-place function symbols.

[6]See, e.g., Cocchiarella 1972, Menzel 1986. These logics are not only semantically type-free in the sense of Bealer 1982 but also syntactically type-free insofar as predicates also count as terms and, hence, can serve as arguments in atomic formulas. Their occurrences in such formulas, of course,

However, as the strict divisions between lexical types and divisions between semantic types begin to break down and one type bleeds into the other, a more radical perspective emerges: the traditional divisions are best represented, not as fixed, pre-existing types but rather as contextually defined *roles*. This is CL's approach: the traditional divisions between lexical/semantic types are abolished entirely: There is but a single primitive syntactic category of *names* and a corresponding semantic category of *things*. Vestiges of the traditional syntactic divisions survive only as contextually defined syntactic roles that any name (or, more generally, any term) can play; likewise for their denotations. Thus, in terms of CLIF's syntax, any list of names between parentheses $(\alpha\ \ \beta_1\ \ \ldots\ \ \beta_n)$ can serve as both a complex term and an atomic sentence.[7] *Qua* term, the leftmost occurrence of the name α in the expression plays the *applicative* role and the thing a_α it denotes plays the *function* role; likewise, *qua* sentence, that occurrence of α plays the *predicative* role and its denotation a_α the *relation* role. And, in both contexts, the terms β_i are all playing *argument* roles and their denotations the *object* role — in the one case as the objects to which a_α is applied and, in the other, the things of which a_α is predicated. Moreover, as there are no restrictions whatever on the formation of atomic sentences from names, we have, in particular, that $(\alpha\ \ \alpha)$ is both a legitimate formula and a legitimate function term and, hence, both self-predication and self-application are expressible with ease.[8]

A thing plays the object role simply in virtue of being an argument to (something playing the role of) a function or relation, more or less as in the semantics of standard predicate logic. But what does it mean for a thing to play a function or relation role? And how can it play both roles simultaneously as in self-predications/applications like $(\alpha\ \ \alpha)$? The answer is that every thing in the domain of an interpretation is assigned both a function extension and a relation ex-

is meant to represent the nominalizations of the expressions represented by their occurrences as predicates.

[7] I will usually exhibit standard quote-avoidance behavior and take object language expressions used in the metalanguage as names for themselves. Avoidance of corner quotes should be thought of as analogously justified.

[8] In fact, complete type-freedom is only a property of so-called *non-segregated* dialects of CL. In segregated dialects, some names are allowed only to play applicative or predicative roles and, moreover, are not permitted to denote anything in the domain of things. (A separate domain is set aside for their denotations.) CL dialects in which *only* segregated names can play those roles therefore exhibit no type-freedom. An early version of CL known as *Simple Common Logic* (Menzel and Hayes 2003) approached type-freedom from the opposite direction — languages were defined in terms of the traditional syntactic categories, but the categories were allowed to bleed into one another. In the limit, the categories could overlap completely, yielding, in effect, a single lexical type. Name notwithstanding, due to its approach and generality, Simple CL was rather considerably more complex than CL.

tension. Thus, the denotation of $(\alpha \ \ \beta_1 \ \ \ldots \ \ \beta_n)$, *qua* term, is the value $f(b_{\beta_1},...,b_{\beta_n})$ of the function extension $f = e_{fn}(a_\alpha)$ assigned to the denotation a_α of α on the sequence $\langle b_{\beta_1}, ..., b_{\beta_n}\rangle$ consisting of the denotations $b_{\beta_1}, ..., b_{\beta_n}$ of $\beta_1, ..., \beta_n$, respectively. Likewise, *qua* atomic formula, $(\alpha \ \ \beta_1 \ \ \ldots \ \ \beta_n)$ is true just in case $\langle b_{\beta_1}, ..., b_{\beta_n}\rangle$ is in the relation extension $r = e_{rel}(a_\alpha)$ assigned to a_α.

The coherence of self-application and self-predication on this picture should be clear: Nothing whatever prevents an object a from being in the domain of its function extension or from being in its relation extension. Hence, *qua* function term, $(\alpha \ \ \alpha)$ denotes $f(a_\alpha)$, where f is the function extension of a_α; and, *qua* atomic formula, $(\alpha \ \ \alpha)$ is true just in case a_α is a member of its own relation extension r. And note that, because things are not identified with their either their function or relation extensions, there is no need for non-well-founded set theory to make sense of these semantic conditions.

Variable Polyadicity. Variable polyadicity is a well-known feature of many natural language verbs: *eats*, for example, is not restricted to a single argument; it can be true of John, of John and his cheeseburger, of John and his cheeseburger and the restaurant in which, and the time at which, he eats it, and so on. CL incorporates variable polyadicity explicitly into its syntax. Specifically, there is no notion of an *arity*, or *adicity*, associated with any name: in CLIF, in particular, for any name α and *any* finite number of terms $\beta_1, ..., \beta_n$, $(\alpha \ \ \beta_1 \ \ ... \ \ \beta_n)$ is both a term and an atomic sentence.

To represent variable polyadicity semantically, we simply place no corresponding arity restrictions on function and relation extensions. In particular, where D is the domain of a given CL dialect, the relation extension of the thing a_α denoted by a name α is simply stipulated to be a subset of the set $D^* = \bigcup_{n<\omega} D^n$ of all finite sequences of elements of D. Note that this does not entail that all, or even any, relations in a given domain must be semantically variably polyadic. It's just that this constraint is viewed as one to be expressed semantically, not enforced syntactically. For example, that the property F is unary, say — and, hence, that it can only be exemplified by 1-tuples — can be enforced schematically; in CLIF (which uses prefix Boolean operators):

$$\texttt{(not (or (F) (exists (x}_1 \ldots \texttt{x}_n\texttt{) (F x}_1 \ldots \texttt{x}_n\texttt{))))}, \text{ for } n > 1.^{9,10}$$

[9] Of course, in a non-modal framework like CL, this condition is shy of perfect, as it also holds of unexemplifiable properties and relations F.

[10] It should be noted that the language of full CL also allows for the inclusion of so-called *sequence markers*, which are in effect variables over finite sequences of elements of the domain. (The sequences

It is worth noting briefly the special case of n-place atomic formulas when $n = 0$. In this case (using the CLIF dialect once again), we simply have atomic sentences of the form `(`α`)`. Nonetheless, the truth conditions for such sentences are the same as any other atomic sentence: `(`α`)` is true (in an interpretation) just in case the sequence consisting of the denotations of the arguments to α in `(`α`)` — i.e., the empty sequence $\langle\rangle$ — is in the relation extension of the denotation a_α of α.

"Higher-order" Quantification. A further element of CL's type-freedom is better discussed in connection with the last of CL's general, distinctive features, viz., the fact that CL dialects are *syntactically higher-order*. As noted above, CL dialects contain only a single non-logical lexical category of names. One might infer from this that variables are considered part of the *logical* apparatus of a CL dialect. In fact, however, CL dispenses with a separate lexical class of variables altogether; rather, as with predicates and function symbols, in CL, *variable* is simply another role that names can play. Thus, in the CLIF formula `(exists (x) (F x))`, `x` is a name but, in virtue of the initial `x`-binding quantifier, the second occurrence of `x` is playing the (bound) variable role in the formula. Additionally, *any* free occurrence of any name can be quantified. Hence, as names can play applicative and predicative roles in atomic formulas and terms, occurrences of names in those roles can be quantified as well. Thus, for example, in CLIF,

```
(if (G a) (exists (F) (F a)))
```

is perfectly well-formed (and, indeed, a theorem of the proof theory below). Again, combining its second-order and type-free features, the formula

```
(exists (F) (forall (G) (iff (F G) (not (G G)))))
```

is entirely legitimate as well.

Intuitively, the preceding CLIF formula expresses the existence of a "Russell property", i.e., a thing that (*qua* property) is true of all and only those things that are not

are not themselves assumed to be members of the domain.) In this framework, the above schema can be expressed in a single sentence of the object language (where `@s` is a sequence marker):

```
(not (or (F) (exists (x y @s (F x y @s))))).
```

That is, roughly, F does not hold *qua* proposition, nor are there x and y and no (possibly empty) finite sequence z_1, ..., z_n of things such that F is true of $\langle x,y,z_1,...,z_n\rangle$. The addition of sequence markers adds considerable expressive power to CL but, of course, at a cost — one can, for example, express finitude, but that immediately implies that compactness fails and, hence, that validity for the framework is not axiomatizable.

true of themselves. By familiar reasoning one can quickly derive a contradiction from this formula in the proof theory provided below. One might then rightly wonder how inconsistency is avoided in CL. The answer is that, its "higher-order" features notwithstanding, there are no constraints whatever on the "property roles" that must be played in a given interpretation of a language in a given CL dialect. Hence, there are no valid, general comprehension principles; there are no principles telling us that, for any expressible condition there is a thing whose relation extension includes exactly the (n-tuples of) things that satisfy that condition — the condition `(not (G G))` in particular. Hence, the existence of a Russell property is not provable and, indeed, that there is no such property — i.e., the negation of the sentence above — is a theorem of the proof theory below.

3 Syntax

As noted, the CLIF dialect derives from KIF. As defined in the ISO standard, CLIF is in fact a single language containing numerals and countably many names. Here we will follow rather more standard practice and define an entire class of languages that includes the ISO language as a special case.

3.1 Languages

A *CLIF language* consists of the following *lexical items*:

- Logical operators: `if`, `not`, `forall`
- Identity: =
- Names: A denumerable set N_L of nonempty strings of unicode text characters (i.e., no whitespace) other than the logical operators
- The unicode SPACE character (U+0200)
- Parentheses: `(`, `)`

Note that, while a CLIF language must contain the identity symbol, the definition above allows it to be counted among the names of the language. As this turns out to be fairly significant for our purposes, we distinguish two types of languages:

Definition 1. A CLIF language L is *inclusive* if it includes the identity symbol '=' among its names. L is *conventional* if it does not.

Henceforth, let L be some arbitrary CLIF language and, for purposes of the completeness theorem below, assume that there are denumerably many unicode text strings that are not among the names of L.

3.2 Grammar

An *expression* of L is any (possibly empty) string of lexical items. We define the *terms* and *sentences* of L simultaneously:

1. Every name of L is a *term* of L.
2. If α, β_1, ..., β_n are terms of L ($n \geq 0$), then the expression (α β_1 ... β_n) is both a term and a *sentence* of L.
   - If L is conventional and α and β are terms of L, then the expression (= α β) is a sentence of L.
3. If φ is a sentence of L, so is (`not` φ).
4. If φ and ψ are sentence of L, so is (`if` φ ψ).
5. If φ is a sentence of L and $\nu \in N_L$, then (`forall` (ν) φ) is a sentence of L.
6. Nothing else is a term or sentence of L.

The sentences given by clause 2 are *atomic*. Note in particular that a consequence of this definition is that, for any term α, (α) is a sentence of L. We let Σ_L be the set of sentences of L. (Henceforth, reference to L will usually be suppressed unless the context demands otherwise.)

A sentence ψ is a *subsentence of* another sentence φ iff there are expressions ϵ, ϵ' of L such that φ is $\epsilon\psi\epsilon'$. Freedom and bondage are defined for names in a way analogous to their usual definitions for variables. Specifically: An occurrence of a name ν in a sentence φ is *bound* if that occurrence is in a subsentence of φ of the form (`forall` (ν) ψ); otherwise we say that that occurrence is *free*. By $(\varphi)^{\mu}_{\alpha}$ we will understand the result of replacing every free occurrence of the name μ in φ with the term α. (Parentheses will be dropped when clarity does not demand them.) A term α is said to be *free for ν in φ* if, for every name μ occurring in α, no free occurrence of ν in φ is in a subsentence of φ of the form (`forall` (μ) ψ).

For ease of expression in a printed medium (and also as a nod to comfortable tradition), we will henceforth typically abbreviate sentences of the form (`forall` (ν) ψ) as ($\forall\nu$ ψ).

Definition 2. Atomic sentences φ and ψ are *alphabetic variants* (of each other) if $\varphi = \psi$. If φ and φ' are alphabetic variants, then: (i) $(\texttt{not}\ \varphi)$ and $(\texttt{not}\ \varphi')$ are alphabetic variants; (ii) if ψ and ψ' are also alphabetic variants, so are $(\texttt{if}\ \varphi\ \psi)$ and $(\texttt{if}\ \varphi'\ \psi')$; and (iii) if (a) μ is free for ν in φ' and (b) μ does not occur free in φ' if $\mu \neq \nu$, then $(\forall\nu\ \varphi)$ and $(\forall\mu\ (\varphi')^{\nu}_{\mu})$ are alphabetic variants. Finally, sentences are alphabetic variants only if their being so follows from the preceding clauses.

Henceforth, unless otherwise stated, ν and μ (possibly with primes or numerical subscripts) will indicate names of L; α, β, and γ will indicate terms of L; φ, ψ, and θ will indicate sentences of L; and Γ and Σ and will indicate sets of sentences of L.

Defined Logical Operators. Other common logical operators are defined as usual.

- $(\texttt{and}\ \varphi\ \psi) =_{df} (\texttt{not}\ (\texttt{if}\ \varphi\ (\texttt{not}\ \psi)))$
- $(\texttt{or}\ \varphi\ \psi) =_{df} (\texttt{if}\ (\texttt{not}\ \varphi)\ \psi)$
- $(\texttt{iff}\ \varphi\ \psi) =_{df} (\texttt{and}\ (\texttt{if}\ \varphi\ \psi)\ (\texttt{if}\ \varphi\ \psi))$
- $(\texttt{exists}\ (\nu)\ \varphi) =_{df} (\texttt{not}\ (\forall\nu\ (\texttt{not}\ \varphi)))$

As with the universal quantifier, we will typically abbreviate sentences of the form $(\texttt{exists}\ (\nu)\ \varphi)$ as $(\exists\nu\ \varphi)$.

The ISO syntax for standard Common Logic dialects in fact allows conjunction and disjunction to take any finite number of arguments and for quantifiers to bind any (nonzero) finite number of names. For the theoretical purposes here, however, it will be more convenient to treat these boolean operators as binary and to restrict quantifier binding to single variables and introduce the generalized operators as abbreviations. Accordingly, let ν be any name of L and let $\top$ be the sentence $(\texttt{if}\ (\nu)\ (\nu))$ and $\bot$ its negation (note that $\top$ will turn out to be a logical truth and, hence, $\bot$ a logical falsehood, in the semantics below):

- $(\texttt{and}) =_{df} \top$; $(\texttt{or}) =_{df} \bot$
- $(\texttt{and}\ \varphi) =_{df} (\texttt{or}\ \varphi) =_{df} \varphi$
- $(\texttt{and}\ \varphi_1\ \ldots\ \varphi_n\ \varphi_{n+1}) =_{df} (\texttt{and}\ (\texttt{and}\ \varphi_1\ \ldots\ \varphi_n)\ \varphi_{n+1})$, for $n \geq 2$
- $(\texttt{or}\ \varphi_1\ \ldots\ \varphi_n\ \varphi_{n+1}) =_{df} (\texttt{or}\ (\texttt{or}\ \varphi_1\ \ldots\ \varphi_n)\ \varphi_{n+1})$, for $n \geq 2$
- $(\forall\ (\nu_1\ \ldots\ \nu_n\ \nu_{n+1})\ \varphi) =_{df} (\forall\ (\nu_1\ \ldots\ \nu_n)\ (\forall\nu_{n+1}\varphi))$, for $n \geq 1$

Generalizing the convention above, we will usually write $(\forall\ (\nu_1\ \ldots\ \nu_n)\ \varphi)$ as $(\forall\nu_1 \ldots \nu_n\ \varphi)$.

4 Semantics

For nonempty sets A, let A^n be the set of finite sequences of elements of A and let $A^* = \bigcup_{n=0}^{\infty} A^n$, i.e., the set of finite sequences of elements of A. (We let $A^1 = A$ and, hence, for sequences $\langle a \rangle \in A^1$ of length 1, $\langle a \rangle = a$.) A *general relation* over A is a subset of A^* and a *general function* over A is a total function $f : A^* \longrightarrow A$.[11] An *L-interpretation* $\mathcal{I}$ is a 4-tuple $\langle D, e_{rel}, e_{fn}, V \rangle$, where D is a nonempty set, $e_{rel} : D \longrightarrow \wp(D^*)$ and $e_{fn} : D \longrightarrow \{f \mid f : D^* \longrightarrow D\}$, respectively, assign a general relation and a general function to each element of D, and $V : N \longrightarrow D$ is a denotation function on the names of our arbitrary language L. If L is inclusive, it is required that $e_{rel}(V(=)) = \{\langle a, a \rangle : a \in D\}$. $\mathcal{I}$ is said to be *countable* (*denumerable*, *uncountable*) insofar is D is.

Let $\mathcal{I} = \langle D, e_{rel}, e_{fn}, V \rangle$ be an L-interpretation. For $a \in D$, let $V_a^\nu(\mu) = V(\mu)$ if $\mu \neq \nu$, and $V_a^\nu(\nu) = a$; and let $\mathcal{I}_a^\nu = \langle D, e_{rel}, e_{fn}, V_a^\nu \rangle$. For terms and sentences of L, we define the *general denotation function* d_V for $\mathcal{I}$ and the notion of *truth in* $\mathcal{I}$ simultaneously as follows:

- For names ν of L, $d_V(\nu) = V(\nu)$.
- $d_V(\texttt{(}\alpha\ \ \beta_1\ \ \ldots\ \ \beta_n\texttt{)}) = e_{fn}(d_V(\alpha))(d_V(\beta_1), \ldots, d_V(\beta_n))$.
- $\texttt{(}\alpha\ \ \beta_1\ \ \ldots\ \ \beta_n\texttt{)}$ is *true in* $\mathcal{I}$ iff $\langle d_V(\beta_1), \ldots, d_V(\beta_n) \rangle \in e_{rel}(d_V(\alpha))$.
  - If L is conventional, $\texttt{(=}\ \ \alpha\ \ \beta\texttt{)}$ is true in $\mathcal{I}$ iff $d_V(\alpha) = d_V(\beta)$.
- $\texttt{(not}\ \ \varphi\texttt{)}$ is true in $\mathcal{I}$ iff φ is not true in $\mathcal{I}$.
- $\texttt{(if}\ \ \varphi\ \ \psi\texttt{)}$ is true in $\mathcal{I}$ iff either φ is not true in $\mathcal{I}$ or ψ is true in $\mathcal{I}$.
- $\texttt{(}\forall\nu\ \ \varphi\texttt{)}$ is true in $\mathcal{I}$ iff, for all $a \in D$, φ is true in $\mathcal{I}_a^\nu$.

φ is *false in* $\mathcal{I}$ iff it is not true in $\mathcal{I}$. Note in particular that a consequence of the above definition is that, for any term α, the sentence $\texttt{(}\alpha\texttt{)}$ is true in $\mathcal{I}$ just in case $\langle\rangle \in e_{rel}(d_V(\alpha))$, i.e., just in case the null sequence is in the relation extension assigned to α's denotation.

Let Γ be a set of sentences of L. An L-interpretation $\mathcal{I}$ is said to be an *L-model of* Γ iff every member of Γ is true in $\mathcal{I}$, and Γ is said to be *L-satisfiable* if it has an

[11] Thus, a "general" relation is any relation, whether variably polyadic or of fixed adicity. All functions in CL are variably polyadic, as they are totally defined over all of A^*.

L-model. A sentence φ of L is an *L-semantic consequence of* Γ — $\Gamma \models_L \varphi$ — if it is true in every L-model of Γ. φ is *L-valid* — $\models_L \varphi$ — if $\varnothing \models_L \varphi$, i.e., if it is true in every L-interpretation; φ is *L-contradictory* if it is false in every L-interpretation. When the language L in question is understood in a given context, we shall often drop the 'L-' prefix and the 'L' subscript.

This section and the following will contain a number of lemmas that are useful in the proofs of soundness and completeness below. First, as noted above:

Lemma 1. *$\top$ is valid and (hence) $\bot$ is contradictory.*

Proof. Let $\mathcal{I}$ be an arbitrary interpretation. Recall that $\top = (\texttt{if}\ \ (\nu)\ \ (\nu))$, for some arbitrary name ν. The clause for '`if`' will of course guarantee that $\top$ is true in $\mathcal{I}$ so long as the truth condition for 0-place atomic formulas like (ν) is well-defined. And it is: (ν) is true in $\mathcal{I}$ iff $\langle\rangle \in e_{rel}(d_V(\nu))$. So $\top$ is valid and, hence, its negation $\bot$ is contradictory. □

The following lemma is useful in the proof of completeness.

Lemma 2. *Let $\mathcal{I} = \langle D,, e_{rel}, e_{fn}, V\rangle$ be an L-interpretation. Then (a) for any term α, $d_V(\alpha^\nu_\beta) = d_{V^\nu_{V(\beta)}}(\alpha)$ and (b) if β is free for ν in φ, φ^ν_β is true in $\mathcal{I}$ iff φ is true in $\mathcal{I}^\nu_{V(\beta)}$.*

Proof. We prove (a) by induction on terms α. If ν does not occur in α then the lemma holds trivially. So suppose ν occurs in α. If α is a name, then $\alpha = \nu$ and so $d_V(\alpha^\nu_\beta) = d_V(\nu^\nu_\beta) = d_V(\beta) = V(\beta) = d_{V^\nu_{V(\beta)}}(\nu) = d_{V^\nu_{V(\beta)}}(\alpha)$. So suppose α is $(\gamma\ \ \beta_1\ \ \ldots\ \ \beta_n)$ and (a) holds for γ and the β_i. Then

$$\begin{aligned} d_V((\gamma\ \ \beta_1\ \ \ldots\ \ \beta_n)^\nu_\beta) &= d_V((\gamma^\nu_\beta\ \ (\beta_1)^\nu_\beta\ \ \ldots\ \ (\beta_n)^\nu_\beta)) \\ &= e_{fn}(d_V(\gamma^\nu_\beta))(d_V((\beta_1)^\nu_\beta), \ldots, d_V((\beta_n)^\nu_\beta)) \\ &= e_{fn}(d_{V^\nu_\beta}(\gamma))(d_{V^\nu_\beta}(\beta_1), \ldots, d_{V^\nu_\beta}(\beta_n)) \\ &= d_{V^\nu_\beta}((\gamma\ \ \beta_1\ \ \ldots\ \ \beta_n)). \end{aligned}$$

We prove (b) by induction on sentences. So suppose β is free for ν in φ. If φ is an atomic sentence $(\alpha\ \ \beta_1\ \ \ldots\ \ \beta_n)$, then

$$\begin{aligned} (\alpha\ \ \beta_1\ \ \ldots\ \ \beta_n)^\nu_\beta \text{ is true in } \mathcal{I} \quad &\text{iff} \quad (\alpha^\nu_\beta\ \ (\beta_1)^\nu_\beta\ \ \ldots\ \ (\beta_n)^\nu_\beta) \text{ is true in } \mathcal{I} \\ &\text{iff} \quad \langle d_V((\beta_1)^\nu_\beta), \ldots, d_V((\beta_n)^\nu_\beta)\rangle \in d_V(\alpha^\nu_\beta) \\ &\text{iff} \quad \langle d_{V^\nu_\beta}(\beta_1), \ldots, d_{V^\nu_\beta}(\beta_n)^\nu_\beta\rangle \in d_{V^\nu_\beta}(\alpha^\nu_\beta) \text{ (by (a))} \\ &\text{iff} \quad (\alpha\ \ \beta_1\ \ \ldots\beta_n) \text{ is true in } \mathcal{I}^\nu_{V(\beta)}. \end{aligned}$$

For conventional L, the atomic case for identities is similar. The boolean cases of our proof are straightforward. So suppose for our induction hypothesis that (b) holds for ψ and that φ is $(\forall\mu\ \ \psi)$. We note first that, in the case where $\nu = \mu$, (b) holds trivially for φ. So suppose otherwise; then we have:

$$\begin{array}{lll} (\forall\mu\ \ \psi)^{\nu}_{\beta} \text{ is true in } \mathcal{I} & \text{iff} & (\forall\mu\ \ \psi^{\nu}_{\beta}) \text{ is true in } \mathcal{I} \\ & \text{iff} & \text{for all } a \in D,\ \varphi^{\nu}_{\beta} \text{ is true in } \mathcal{I}^{\mu}_{a} \\ & \text{iff} & \text{for all } a \in D,\ \varphi \text{ is true in } \mathcal{I}^{\mu\,\nu}_{a\,V(\beta)} \text{ (by our ind. hyp.)} \\ & \text{iff} & \text{for all } a \in D,\ \varphi \text{ is true in } \mathcal{I}^{\nu\quad\ \mu}_{V(\beta)\,a} \\ & \text{iff} & (\forall\mu\ \ \psi) \text{ is true in } \mathcal{I}^{\nu}_{V(\beta)}. \end{array}$$

□

5 Proof Theory: The System $\mathbf{C}_L$

Despite its novel features, the preceding semantics can be outfitted with a proof theory that features a rather standard-looking set of axiom schemas.[12] We present the proof theory in this section and prove some standard metatheoretic facts about it that will be used in the completeness theorem below.

Definition 3. A *generalization of* a sentence φ is either φ itself or any sentence of L of the form

$$(\forall\nu_1 ... \nu_n\ \ \varphi).$$

Any generalization of any instance of any of the following is an *axiom of* C_L:

A1 Propositional tautologies

A2 $(\texttt{if}\ \ (\forall\nu\ \ \varphi)\ \ \varphi^{\nu}_{\alpha})$, where α is free for ν in φ

A3 $(\texttt{if}\ \ (\forall\nu\ \ (\texttt{if}\ \ \varphi\ \ \psi))\ \ (\texttt{if}\ \ (\forall\nu\ \ \varphi)\ \ (\forall\nu\ \ \psi)))$

A4 $(\texttt{if}\ \ \varphi\ \ (\forall\nu\ \ \varphi))$, where ν does not occur free in φ

A5 $(=\ \ \nu\ \ \nu)$, for any name ν of L

A6 $(\texttt{if}\ \ (=\ \ \nu\ \ \mu)\ \ (\texttt{if}\ \ \varphi\ \ \varphi^{\nu}_{\mu}))$, where μ is free for ν in φ

[12] I have drawn chiefly upon Enderton 2001, Ch. 2.

The system C_L has one *rule of inference*:

- *Modus Ponens* (*MP*): ψ follows from φ and $(\texttt{if}\ \ \varphi\ \ \psi)$.

Let Λ_{C_L} be the set of axioms of C_L. A *proof* in C_L is a finite sequence $\varphi_1,...,\varphi_n$ of sentences of L (the *elements* of the proof) such that each sentence φ_k in the sequence is either a member of Λ_{C_L} or follows from preceding sentences in the sequence by MP, that is, there are positive integers $i, j < k$ such that $\varphi_i = (\texttt{if}\ \ \varphi_j\ \ \varphi_k)$. φ is a *theorem* of C_L ($\vdash_{\mathrm{C}_L} \varphi$) if there is a proof in C_L whose last sentence is φ.

A *proof from* a set $\Gamma \subseteq \Sigma_L$ in C_L is a finite sequence $\varphi_1,...,\varphi_n$ of sentences of L such that each sentence φ_k in the sequence is either a member of $\Lambda_{C_L} \cup \Gamma$ or follows from preceding sentences in the sequence by MP. φ is a *theorem of* Γ in C_L ($\Gamma \vdash_{\mathrm{C}_L} \varphi$) if there is a proof from Γ in C_L whose last sentence is φ. Γ is *consistent in C_L*, or *C_L-consistent*, if there is a sentence φ of L such that $\Gamma \nvdash_{\mathrm{C}_L} \varphi$; Γ is *inconsistent in C_L*, or *C_L-inconsistent*, if it is not consistent in C_L.

Henceforth, unless clarity demands otherwise, we will usually avoid explicit reference to the deductive system at hand.

Lemma 3. *The following are equivalent: (a) Γ is inconsistent; (b) $\Gamma \vdash \bot$; (c) for some sentence φ, $\Gamma \vdash \varphi$ and $\Gamma \vdash (\texttt{not}\ \ \varphi)$.*

Proof. Recall that $\bot =_{df} (\texttt{not}\ \ (\texttt{if}\ \ (\nu)\ \ (\nu)))$, for some arbitrarily chosen name ν. That (a) implies (b) is trivial from the definition of inconsistency. So suppose $\Gamma \vdash \bot$. Then a proof of $\bot$ can be extended to a proof of any sentence ψ by appending $(\texttt{if}\ \ \bot\ \ \psi)$ (a tautology and, hence, a member of axiom group **A1**) and then ψ, which now follows by MP. To see that (c) implies (a), simply note that, for any sentence ψ, $(\texttt{if}\ \ (\varphi\ \ (\texttt{if}\ \ (\texttt{not}\ \ \varphi)\ \ \psi)))$ is a tautology. From this fact a proof of ψ is quickly constructed from proofs of φ and $(\texttt{not}\ \ \varphi)$ from Γ. □

Let $\Gamma, \varphi \vdash \psi$ be short for $\Gamma \cup \{\varphi\} \vdash \psi$.

Theorem 1. (Deduction Theorem) *If $\Gamma, \varphi \vdash \psi$, then $\Gamma \vdash (\texttt{if}\ \ \varphi\ \ \psi)$.*

Proof. As usual, by induction on length of proof. So suppose the theorem holds for proofs of length $< n$ and that there is a proof $\varphi_1,...,\varphi_n = \psi$ of length n of ψ from $\Gamma \cup \{\varphi\}$. Suppose first $\psi \in \Lambda_L \cup \Gamma \cup \{\varphi\}$. If $\varphi = \psi$, we're done, as in that case $(\texttt{if}\ \ \varphi\ \ \psi)$ is a tautology and hence a logical axiom and therefore a theorem of Γ. So suppose $\varphi \neq \psi$. If ψ is an axiom or a member of Γ, then $\Gamma \vdash \psi$ and, hence, as $(\texttt{if}\ \ (\psi\ \ (\texttt{if}\ \ \varphi\ \ \psi)))$ is a tautology, $\Gamma \vdash (\texttt{if}\ \ \varphi\ \ \psi)$. So suppose ψ follows by

MP, i.e., that there are positive integers $i, j < k$ such that $\varphi_i = (\texttt{if}\ \ \varphi_j\ \ \psi)$. Since any proper initial segment of our proof is a proof from $\Gamma \cup \{\varphi\}$ of length $< n$, it follows by our induction hypothesis that $\Gamma \vdash (\texttt{if}\ \ (\varphi\ \ \varphi_i))$ and $\Gamma \vdash (\texttt{if}\ \ (\varphi\ \ (\texttt{if}\ \ (\varphi_j\ \ \psi))))$. By the latter (and propositional logic), we have $\Gamma \vdash (\texttt{if}\ \ (\texttt{if}\ \ \varphi\ \ \varphi_j)\ \ (\texttt{if}\ \ (\varphi\ \ \psi)))$ and, hence, by the former, $\Gamma \vdash (\texttt{if}\ \ (\varphi\ \ \psi))$. □

Corollary 1. *If* $\Gamma \cup \{\varphi\}$ *is inconsistent, then* $\Gamma \vdash (\texttt{not}\ \ \varphi)$.

Proof. If $\Gamma \cup \{\varphi\}$ is inconsistent, then $\Gamma, \varphi \vdash \bot$ and, hence, by the Deduction Theorem, $\Gamma \vdash (\texttt{if}\ \ \varphi\ \ \bot)$ and so, by some simple propositional logic, $\Gamma \vdash (\texttt{not}\ \ \varphi)$. □

Corollary 2. *If* $\Gamma \nvdash \varphi$*, then* $\Gamma \cup \{(\texttt{not}\ \ \varphi)\}$ *is consistent.*

Proof. Suppose $\Gamma \nvdash \varphi$. Then Γ is consistent. If $\Gamma \cup \{(\texttt{not}\ \ \varphi)\}$ is inconsistent and, hence, by the preceding corollary, $\Gamma \vdash \varphi$ and, hence, Γ is inconsistent, after all. So $\Gamma \cup \{(\texttt{not}\ \ \varphi)\}$ must be consistent. □

The ability to generalize on names is an important property of C_L.

Theorem 2. (Generalization) *If there is a proof of* φ *from* Γ *and no element of the proof is a member of* Γ *in which* ν *occurs free, then* $\Gamma \vdash (\forall\nu\ \ \varphi)$.

Proof. By induction on length of proof. Suppose the theorem holds for proofs of length $< n$ and suppose there is a proof $\varphi_1, ..., \varphi_n$ of φ ($= \varphi_n$) of length n. Then either $\varphi \in \Gamma \cup \Lambda$ or φ follows by MP from preceding sentences in the proof. Suppose first that $\varphi \in \Lambda$. Then $(\forall\nu\ \ \varphi) \in \Lambda$, as it is a generalization of an axiom; hence $\Gamma \vdash (\forall\nu\ \ \varphi)$. If $\varphi \in \Gamma$, then (by assumption) ν does not occur free in φ and, hence, ν doesn't occur free in φ. Thus, $(\texttt{if}\ \ \varphi\ \ (\forall\nu\ \ \varphi))$ is an instance of axiom schema **A4**. Hence, φ, $(\texttt{if}\ \ \varphi\ \ (\forall\nu\ \ \varphi))$, $(\forall\nu\ \ \varphi)$ is a proof of $(\forall\nu\ \ \varphi)$ from Γ. Suppose then that φ follows by MP, i.e., that there are positive integers $i, j < k$ such that $\varphi_i = (\texttt{if}\ \ \varphi_j\ \ \varphi)$. Since any proper initial segment of our proof from Γ is itself a proof from Γ of length $< n$ that satisfies the conditions of the theorem, it follows by our induction hypothesis that

$$\Gamma \vdash (\forall\nu\ \ \varphi_j) \tag{1}$$

and

$$\Gamma \vdash (\forall\nu\ \ (\texttt{if}\ \ \varphi_j\ \ \varphi)) \tag{2}$$

By an appropriate instance of axiom schema **A3**, (2) yields

$$\Gamma \vdash (\texttt{if}\ \ (\forall\nu\ \ \varphi_j)\ \ (\forall\nu\ \ \varphi)) \tag{3}$$

Hence, by (1), (3), and MP, we have $\Gamma \vdash (\forall\nu\ \ \varphi)$. □

Given Generalization, we can show:[13]

Lemma 4. *If φ and φ' are alphabetic variants, then* $\vdash$ `(if` φ φ'`)`.

Proof. Suppose the theorem holds for all sentences of complexity $< n$ and suppose φ is of complexity n. The theorem follows trivially if φ is atomic (since $\varphi = \varphi'$ in this case) and is straightforward if φ is boolean. So suppose φ is of the form $(\forall\nu\ \ \psi)$ and let φ' be an alphabetic variant $(\forall\mu\ \ \psi'^{\nu}_{\mu})$. It follows by Definition 2 that (a) ψ and ψ' are alphabetic variants and (b) μ is free for ν in ψ' and (c) μ does not occur free in ψ' if $\mu \neq \nu$. By (a) and our induction hypothesis, $\vdash$ `(iff` ψ ψ'`)`. By the Generalization theorem and some propositional logic it follows that $\vdash (\forall\nu$ `(if` ψ ψ'`))` and hence, by axiom group **A3**,

$$\vdash (\texttt{if}\ \ (\forall\nu\ \ \psi)\ \ (\forall\nu\ \ \psi')). \tag{4}$$

By (b) and axiom group **A2** we have $\vdash (\texttt{if}\ \ (\forall\nu\ \ \psi')\ \ \psi'^{\nu}_{\mu})$. Hence, by Generalization once again it follows that $\vdash (\forall\mu\ \ ((\texttt{if}\ \ (\forall\nu\ \ \psi')\ \ \psi'^{\nu}_{\mu}))$. So by (c), axiom groups **A3** and **A4**, and MP we have $\vdash (\texttt{if}\ \ (\forall\nu\ \ \psi')\ \ (\forall\mu\ \ \psi'^{\nu}_{\mu}))$ and, hence, by (4), $\vdash (\texttt{if}\ \ (\forall\nu\ \ \psi)\ \ (\forall\mu\ \ \psi'^{\nu}_{\mu}))$. □

6 Two Logical Extensions of CLIF

There are two rather natural logical extensions to CLIF concerning function and relation extensions.

[13] We can also easily show now that the nonexistence of a Russell property is a theorem. By **A3** and MP we have:

(∀G (iff (F G) (not G G))) ⊢ (iff (F F) (not (F F)).

By **A1**,

⊢ (if (iff (F F) (not (F F))) ⊥),

so by some simple propositional logic we have

(∀G (iff (F G) (not G G))) ⊢ ⊥

and hence, by Lemma 3, {(∀G (iff (F G) (not G G)))} is inconsistent. Hence, by Corollary 1,

⊢ (not (∀G (iff (F G) (not G G)))).

By Generalization, some propositional logic, and the definition of ∃ it follows that

⊢ (not (∃F (∀G (iff (F G) (not G G))))).

6.1 Argument Monotonicity and Its Logic

In the semantics of Section 4, there are no constraints whatsoever on the extensions of variably polyadic relations; $e_{rel}(a)$ is any arbitrary subset of D^*. However, examples of variable polyadicity in natural language suggest that things are more systematic. For example, if John is eating an apple in the kitchen,

`(Eating John anApple theKitchen)`,

then John is eating an apple, `(Eating John anApple)`, John is eating, `(Eating John)`, and, indeed, there is simply *eating*, `(Eating)`. More generally put, in many cases, the variable polyadicity of a relation reflects the fact that the relation accommodates the addition of new information in a way that preserves old information. Accordingly, let us call this property of variably polyadic relations *argument monotonicity*.

A very simple semantic condition ensuring argument monotonicity is this:

M′ If $\langle b_1,...,b_{n+1}\rangle \in e_{rel}(a)$, then $\langle b_1,...,b_n\rangle \in e_{rel}(a)$, for any $n \geq 0$.

However, this condition is undesirable, as it forces all relations with anything other than $\langle\rangle$ in their extensions to be variably polyadic; under this condition, there could be, e.g., no purely binary relations, relations that hold only of ordered pairs of objects. We will therefore simply restrict the condition above so it applies to variably polyadic relations only:

M If $e_{rel}(a)$ is variably polyadic, then if $\langle b_1,...,b_{n+1}\rangle \in e_{rel}(a)$, then $\langle b_1,...,b_n\rangle \in e_{rel}(a)$, for any $n \geq 0$.

This constraint of course yields new logical truths which therefore need to be captured in new axioms if we want a complete proof theory. In general, variable polyadicity is the property of containing m- and n-tuples for at least two distinct numbers m, n and, of course, it is not possible to express such facts in a single axiom without introducing dedicated (non-logical) arithmetic machinery, not something we want in a general purpose logical theory like CL. A second possibility is to introduce dedicated syntactic machinery, e.g., a predicate `VarP` or a stock of dedicated names that can only take objects with variably polyadic relation extensions as values. The first suggestion, however, is undesirable insofar as the property of being variably polyadic, while perhaps broadly logical, is at the least nonstandard. And the second suggestion would be out of keeping with CL's type-free, egalitarian ideology. Fortunately, our new semantic condition can be expressed schematically without any ideological compromise as follows:

A7 $(\texttt{if}\ (\rho\ \sigma_1\ \dots\ \sigma_m)\ (\texttt{if}\ (\rho\ \tau_1\ \dots\ \tau_{n+1})\ (\rho\ \tau_1\ \dots\ \tau_n)))$, for $m,n \geq 0$, $m \neq n+1$.

That is, if a relation holds both of a certain number m of things $c_1, \dots, c_m$, respectively, and also of a certain distinct nonzero number $n+1$ of things $b_1, \dots, b_n, b_{n+1}$, respectively, then it is variably polyadic and, hence, because of argument monotonicity, it must also hold of $b_1, \dots, b_n$, respectively.

A7 is a strong condition. It is in fact not difficult to imagine variably polyadic relations that hold of different numbers of things but do not exhibit argument monotonicity across the board. For example, a contest might dole out prizes only to individuals and married couples but prohibit winners *qua* couples from winning *qua* individuals. Hence, if x and y, *qua* couple, are *winners*, `(W x y)`, it does not follow that x is a winner, `(W x)` (indeed, according to the rules, it follows that x is *not* a winner). Hence, in some application domains, it might be that not all variably polyadic relations exhibit argument monotonicity. In these cases, non-logical axioms of the form of **A7** could simply be introduced piecemeal and condition **M** would not be part of the semantic apparatus.

6.2 Functional and Relational Guises and Their Logic

The second extension reflects the fact that there is often a more systematic connection between the relational and functional roles of objects than is captured in the semantics of Section 4. As things stand, there are no constraints connecting the functional and relations extensions of an object. In actual cases in which a name has both functional and relational "guises", the relational extension is intuitively just the graph of the corresponding function; thus, for example, so long as we restrict our attention to things that have fathers, we have `(father-of x y)` if and only if `(= (father-of x) y)`. A simple condition that captures this idea is that, whenever the relational extension $e_{rel}(a)$ of an object a is functional, it must be a subset of the object's function extension $e_{fn}(a)$. However, when we know that a relation — *parent of*, say — holds between an object and one or more other things, ordinary language permits the use of the indefinite article to form an expression — *a parent of* — that refers, perhaps arbitrarily, to one of those things. Thus, if we know that both parents of Johan live in Ulm, we know that a parent of Johan lives there. This suggests a more general logical connection between function extensions and relation extensions than the condition just noted, viz.:

G If, for at least one c, $\langle b_1, \dots, b_n, c\rangle \in e_{rel}(a)$, then for some such c, $e_{fn}(a)(b_1, \dots, b_n) =$

c.[14]

This condition can be expressed elegantly in a simple axiom schema:

A8 `(if (`$\alpha\ \beta_1\ \ldots\ \beta_n\ \nu$`) (`$\alpha\ \beta_1\ \ldots\ \beta_n$` (`$\alpha\ \beta_1\ \ldots\ \beta_n$`)))`.

Thus, referring back to our example above, we have as an instance of this schema:

```
(if (parentOf Johan x) (parentOf Johan (parentOf Johan))).
```

Hence, from

```
(∀x (if (parentOf Johnn x) (livesIn x Ulm)))
```

and

```
(∃x (parentOf Johan x))
```

we can infer

```
(livesIn ((parentOf Johan) Ulm).
```

Some readers will likely have noticed that the form of schema **A8** is more or less a special case of Hilbert's (1922) "transfinite axiom" for the ε-calculus; in effect, each name in its functional guise serves as a sort of ε-operator for its relational guise: $(\alpha\ \beta_1\ \ldots\ \beta_n)$ picks out *an* object ν such that $(\alpha\ \beta_1\ \ldots\ \beta_n\ \nu)$.

Note also that the semantic condition that guarantees the validity of schema **A8** is, in effect, that $e_{fn}(a)$ must is a sort of choice function with regard to $e_{rel}(a)$. More exactly: Let R be any general relation. Define the equivalence relation $\sim$ on R such that, for $s, s' \in R$, $s \sim s'$ if and only if there are $b_1, \ldots, b_n, c, d$ $(n \geq 0)$ such that $s = \langle b_1, \ldots, b_n, c\rangle$ and $s' = \langle b_1, \ldots, b_n, d\rangle$. $\sim$, that is, simply groups together those $n+1$-tuples of R that share their first n elements. Let $R/\!\sim$ be the partition on $R \setminus \{\langle\rangle\}$ induced by $\sim$.[15] Then for any interpretation $\langle D, e_{rel}, e_{fn}, V\rangle$, our condition **G** is

[14] Note in particular that we do not require that the relation extension of an object reflect its function extension, that is, we don't require that $e_{fn}(a) \subseteq e_{rel}(a)$. The reason for this is simply that function extensions are total functions on the set D^* of n-tuples of D. The condition $e_{fn}(a) \subseteq e_{rel}(a)$ would, therefore, require that all of D^* be in the field of every relation extension. This would yield a large class of new, unintuitive logical truths, e.g., `(∀r (∃xy (r x y)))`. (The corresponding functional proposition `(∀r (∃xy (= (r x) y)))` is also logically true, of course, but it is not a *new* logical truth; it is classically valid simply in virtue of the fact that, in classical logic, function terms have to denote and, hence, function extensions have to be totally defined D^*.)

[15] If $\langle\rangle \in R$, then $\sim$ doesn't partition R, as $\langle\rangle \nsim r$, for all $r \in R$.

simply that, for $a \in D$, $e_{fn}(a) \upharpoonright e_{rel}(a)/\!\sim$ is a choice function for $e_{rel}(a)/\!\sim$. Thus, in light of this, our axiom schema **A8** can also be thought of as a restricted, "purely logical" manifestation of the axiom of choice.

6.3 Interpretations$^+$ and the System C_L^+

In some contexts, at least, it seems that conditions **M** and **G** could quite reasonably be thought to capture logical properties of relation and function extensions and hence that instances of the corresponding schemas **A7** and **A8**, in those contexts, could reasonably considered logical truths. Even if there is not universal agreement about this, however, as their addition adds no serious formal complications to the completeness result that follows, we shall augment the definitions above to accommodate them. Accordingly, say that a *+-interpretation* of our language L, or *L-interpretation*$^+$, is an interpretation of L that satisfies conditions **M** and **G**. We then define *satisfiability*$^+$, the *semantic consequence*$^+$ relation ($\Gamma \models_L^+ \varphi$), *validity*$^+$ ($\models_L^+ \varphi$), and *contradictoriness*$^+$ accordingly. Likewise, let C_L^+ be the result of adding schemas **A7** and **A8** to our system C_L. $\Gamma \vdash_{\mathrm{C}_L^+} \varphi$ thus indicates that φ is a theorem of Γ in C_L^+. We note the following:

Lemma 5. *Corresponding versions of all of the metatheoretic results of Sections 4 and 5 hold for L-interpretations*$^+$ *and for the system* C_L^+.

7 Soundness and Completeness for C_L and C_L^+

In the following two sections we prove that C_L is sound and semantically complete, and we describe the simple modifications required to convert these proofs into corresponding proofs for C_L^+.

7.1 Soundness

Theorem 3. **(Soundness of C_L)** *If* $\Gamma \vdash_{\mathrm{C}_L} \varphi$, *then* $\Gamma \models_L \varphi$.

Proof. It is a nearly trivial exercise to verify that *MP* is truth preserving and that the CLIF axioms are all valid relative to the semantics. Axiom schema 5 is perhaps the only one that deserves mention: the validity of its instances is immediate given the semantic constraint on the extension assigned to the denotation of the identity predicate. □

Corollary 3. *Every L-satisfiable set of sentences is C_L-consistent.*

Proof. Suppose Γ is satisfiable and, hence, has a model $\mathcal{M}$. Then Γ must be C_L-consistent. For suppose otherwise. Then by Lemma 3, $\Gamma \vdash_{C_L} \bot$ so, by Soundness, $\Gamma \models_L \bot$. $\mathcal{M}$, however, is a model of Γ. So by the definition of the consequence relation $\models_L$, $\bot$ is true in $\mathcal{M}$, contradicting Lemma 1 that $\bot$ is false in all interpretations and, hence, in all interpretations. □

Theorem 4. (**Soundness of C_L^+**) *If* $\Gamma \vdash_{C_L^+} \varphi$, *then* $\Gamma \models_L^+ \varphi$.

Proof. All we need to add to Theorem 3 is the observation that it is also an easy exercise to verify that conditions **M** and **G** guarantee the truth of schemas **A7** and **A8** in every interpretation$^+$. □

Reasoning as above, together with Lemma 5, delivers an analogous corollary:

Corollary 4. *Every L-satisfiable$^+$ set of sentences is C_L^+-consistent.*

7.2 Completeness

As usual, most of our efforts are directed toward proving the following lemma:

Lemma 6. (**Model Existence**) *Every C_L-consistent set of sentences has an L-model.*

So let Γ be an arbitrary C_L-consistent set of sentences. We will prove that Γ has an L-model by means of a Henkin-style construction.[16] Toward that end, we prove a series of auxiliary lemmas.

Let $K = \{\kappa_0, \kappa_1, ...\}$ be a denumerable set of new names not in N_L;[17] let L_K be the language whose set of names is $N_{L_K} = N_L \cup K$. It is easy to show first that:

Lemma 7. Γ *is C_{L_K}-consistent.*

Proof. Let φ be a sentence of L. It is easy to verify that any proof of φ in C_{L_K} can be transformed into a proof of φ in C_L by uniformly replacing occurrences of names in K in the proof with names of L that do not occur in the proof.[18] Hence, in particular, if $\Gamma \vdash_{C_{L_K}} \bot$, then $\Gamma \vdash_{C_L} \bot$. □

[16] The general structure of the proof follows Enderton 2001, ch. 2.

[17] Recall that we (in effect) stipulated in §3.1 that there be such a set.

[18] The reason for this (as could be proved in tedious detail by induction) is simply that, since $\Gamma \subseteq \Sigma_L$, a name from K can enter a proof in C_{L_K} only by occurring in an axiom of L_K in which that name plays a role that could be played just as well by any other name not occurring in the proof and hence by any such name in N_L.

We make use of the new names to (intuitively) ensure that there is a "witness" to every false universal quantification. Specifically, let $\langle\alpha_0, \psi_0\rangle$, $\langle\alpha_1, \psi_1\rangle$, ... be an enumeration of $N_{L_K} \times \Sigma_{L_K}$ and, for $i \in \mathbb{N}$, let θ_i be the sentence

$$\texttt{(if (not (}\forall\alpha_i\ \ \psi_i\texttt{)) (not (}\psi_i)^{\alpha_i}_{\kappa_j}\texttt{))},$$

where j is the least number such that κ_j does not occur in ψ_i or in any θ_k, for $k < i$. Let $\Theta = \{\theta_i : i \in \mathbb{N}\}$.

Lemma 8. $\Gamma \cup \Theta$ *is* C_{L_K}*-consistent.*

Proof. Suppose otherwise, i.e., that $\Gamma \cup \Theta$ is inconsistent (in C_{L_K}). For $m \in \mathbb{N}$, let $\Theta_m = \{\theta_i : i < m\}$. Since proofs are finite, there must be an n such that $\Gamma \cup \Theta_n$ is inconsistent. But this is impossible. For $\Gamma \cup \Theta_0$ $(= \Gamma)$ is consistent by Lemma 7. And if $\Gamma \cup \Theta_n$ is consistent, so is $\Gamma \cup \Theta_{n+1}$. For suppose otherwise, i.e., that $\Gamma \cup \Theta_{n+1} = \Gamma \cup \Theta_n \cup \{\theta_n\}$ is inconsistent. Then, by Corollary 1, $\Gamma \cup \Theta_n \vdash \texttt{(not }\ \theta_n\texttt{)}$, i.e.,

$$\Gamma \cup \Theta_n \vdash \texttt{(not (if (not (}\forall\alpha_i\ \ \psi_i\texttt{)) (not (}\psi_i)^{\alpha_i}_{\kappa_j}\texttt{)))}. \tag{5}$$

By a bit of propositional logic, from (5) we have

$$\Gamma \cup \Theta_n \vdash \texttt{(not (}\forall\alpha_i\ \ \psi_i\texttt{))} \tag{6}$$

and

$$\Gamma \cup \Theta_n \vdash (\psi_i)^{\alpha_i}_{\kappa_j} \tag{7}$$

Since κ_j does not occur in $\Gamma \cup \Theta_n$ or ψ_i, it follows by Theorem 2 from (7) that

$$\Gamma \cup \Theta_n \vdash \texttt{(}\forall\kappa_j\ \ \psi_i)^{\alpha_i}_{\kappa_j}\texttt{)}. \tag{8}$$

But $\texttt{(}\forall\kappa_j\ \psi_i)^{\alpha_i}_{\kappa_j}$ and $\texttt{(}\forall\alpha_i\ \ \psi_i\texttt{)}$ are alphabetic variants, so by Theorem 4,

$$\Gamma \cup \Theta_n \vdash \texttt{(}\forall\alpha_i\ \ \psi_i\texttt{)},$$

which, together with (6) and the fact that $\Theta_n \subseteq \Theta$, contradicts our assumption that $\Gamma \cup \Theta$ is consistent. □

Let $\varphi_0, \varphi_1, \ldots$ be an enumeration of the sentences of L_K and consider the following (familiar) construction:

$$\Delta_0 = \Gamma \cup \Theta$$

$$\Delta_{n+1} = \begin{cases} \Delta_n \cup \{\varphi_n\} & \text{if } \Delta_n \cup \{\varphi_n\} \text{is consistent} \\ \Delta_n \cup \{(\texttt{not}\ \ \varphi_n)\} & \text{otherwise.} \end{cases}$$

Lemma 9. *For all $m \in \mathbb{N}$, Δ_m is consistent.*

Proof. Δ_0 is consistent by Lemma 8, so suppose that Δ_n is consistent. If $\Delta_n \cup \{\varphi_n\}$ is consistent, then, trivially, Δ_{n+1} is. So suppose that $\Delta_n \cup \{\varphi_n\}$ is not consistent and, hence, that . Then Δ_{n+1} must again be consistent. For, since $\Delta_n \cup \{\varphi_n\}$ is inconsistent, it follows by Corollary 1 that $\Delta_n \vdash (\texttt{not}\ \ \varphi_n)$. Hence, if Δ_{n+1} $(= \Delta_n \cup \{(\texttt{not}\ \ \varphi_n)\})$ were also inconsistent, it would follow by Corollary 1 that $\Delta_n \vdash (\texttt{not}\ \ (\texttt{not}\ \ \varphi_n))$, contradicting our inductive assumption that Δ_n is consistent. □

Definition 4. A set Σ of sentences of a language L' is *L'-maximal* iff, for every sentence φ of L', either $\varphi \in \Sigma$ or $(\texttt{not}\ \ \varphi) \in \Sigma$.

Let $\Delta = \bigcup_{n \in \mathbb{N}} \Delta_n$.

Lemma 10. *Δ is a consistent, L_K-maximal set.*

Proof. Δ is obviously L_K-maximal. But it must also be consistent. For, suppose otherwise. Then, by Lemma 3, $\Delta \vdash \bot$. But proofs are finite. Hence, for some $\Delta_n \subseteq \Delta$, $\Delta_n \vdash \bot$ and, hence, by Lemma 3 again, it is inconsistent, contradicting the preceding lemma. □

Lemma 11. *For every term α of L_K, there is a name $\kappa \in K$ such that $\Delta \vdash (= \ \kappa \ \ \alpha)$.*

Proof. Let $\nu \in N_{L_K}$ be a name other than α. $(\texttt{not}\ \ (= \ \nu \ \ \alpha))$ is a formula of L_K. Hence, for some i and j, the formula

$$(\texttt{if}\ \ (\texttt{not}\ \ (\forall\nu\ \ (\texttt{not}\ \ (=\ \nu\ \ \alpha))))\ \ (\texttt{not}\ \ (\texttt{not}\ \ (=\ \kappa_j\ \ \alpha))))$$

that is, given our definitions above, the formula

$$(\texttt{if}\ \ (\exists\nu\ \ (=\ \nu\ \ \alpha))\ \ (\texttt{not}\ \ (\texttt{not}\ \ (=\ \kappa_j\ \ \alpha))))$$

is θ_i and, hence, a member of $\Theta \subseteq \Delta$. However, $\vdash (\exists\nu\ \ (=\ \nu\ \ \alpha))$, by axiom groups **A2** and **A5**; hence, by MP and some propositional logic, $\Delta \vdash (= \ \kappa \ \ \alpha)$. □

Lemma 12. *If, for all $\kappa \in K$, $\varphi^{\nu}_{\kappa} \in \Delta$, then* $(\forall \nu \;\; \varphi) \in \Delta$.

Proof. If ν does not occur free in φ, then the lemma follows directly from axiom group 4. So suppose ν occurs free in φ and that the hypothesis of the lemma holds but that $(\forall \nu \;\; \varphi) \notin \Delta$. By the maximality of Δ, it follows that $(\texttt{not} \;\; (\forall \nu \;\; \varphi)) \in \Delta$. The pair $\langle \nu, \varphi \rangle$ is $\langle \alpha_i, \psi_i \rangle$ of our enumeration of $N_{L_K} \times \Sigma_{L_K}$, for some i. Hence, by construction, for some $\kappa \in K$, $(\texttt{if} \;\; (\texttt{not} \;\; (\forall \nu \;\; \varphi)) \;\; (\texttt{not} \;\; \varphi^{\nu}_{\kappa})) \in \Delta$. Hence, $\Delta \vdash (\texttt{not} \;\; \varphi^{\nu}_{\kappa})$. But it follows from our assumption that $\varphi^{\nu}_{\kappa} \in \Delta$ and, hence, that $\Delta \vdash \varphi^{\nu}_{\kappa}$, which contradicts the consistency of Δ. □

We now construct an L_K-interpretation $\mathcal{I} = \langle D, e_{rel}, e_{fn}, V \rangle$. For terms α of L_K, let $\overline{\alpha} = \{\beta : (= \alpha \;\; \beta) \in \Delta\}$. Let $D = \{\overline{\alpha} : \alpha \text{ is a term of } L_K\}$ and for $\overline{\alpha} \in D$, let $e_{rel}(\overline{\alpha}) = \{\langle \overline{\gamma_1}, \ldots, \overline{\gamma_n} \rangle : (\alpha \;\; \gamma_1 \;\; \ldots \;\; \gamma_n) \in \Delta\}$ and, for $\overline{\gamma_1}, \ldots, \overline{\gamma_n} \in D$, let $e_{fn}(\overline{\alpha})(\overline{\gamma_1}, \ldots, \overline{\gamma_n}) = \overline{(\alpha \;\; \gamma_1 \;\; \ldots \;\; \gamma_n)}$. It can be proved straightaway from our identity axiom schemas **A5** and **A6** that if $\alpha = \beta \in \Delta$, then $\overline{\alpha} = \overline{\beta}$ and hence that e_{rel} and e_{fn} are well-defined. For names $\nu \in N_{L_K}$, let $V(\nu) = \overline{\nu}$. If L is inclusive, it follows that $V(=) = \overline{=}$. Hence, $e_{rel}(\overline{=}) = \{\langle \overline{\gamma_1}, \overline{\gamma_2} \rangle : (= \;\; \gamma_1 \;\; \gamma_2) \in \Delta\}$, from which, given our identity axioms, it quickly follows that $\langle \overline{\gamma_1}, \overline{\gamma_2} \rangle \in e_{rel}(V(=))$ iff $\overline{\gamma_1} = \overline{\gamma_2}$ and, hence, that $e_{rel}(V(=)) = \{\langle a, a \rangle : a \in D\}$, as required. $\mathcal{I}$, therefore, is an L_K-interpretation.

Lemma 13. *For every term α of L_K, $d_V(\alpha) = \overline{\alpha}$.*

Proof. By induction on terms. If α is a name ν, then $d_V(\alpha) = V(\alpha) = \overline{\nu}$. So suppose α is $(\gamma \;\; \beta_1 \;\; \ldots \;\; \beta_n)$ and that the lemma holds for γ and the β_i. Then

$$\begin{aligned} d_V(\alpha) &= d_V((\gamma \;\; \beta_1 \;\; \ldots \;\; \beta_n)) \\ &= e_{fn}(d_V(\gamma))(d_V(\beta_1), \ldots, d_V(\beta_n)) \\ &= e_{fn}(\overline{\gamma})(\overline{\beta_1}, \ldots, \overline{\beta_n}) \\ &= \overline{(\gamma \;\; \beta_1 \;\; \ldots \;\; \beta_n)} \\ &= \overline{\alpha}. \end{aligned}$$

□

Lemma 14. *For every sentence φ of L_K, φ is true in $\mathcal{I}$ iff $\varphi \in \Delta$.*

Proof. By induction on sentences. If φ is atomic $(\alpha \;\; \beta_1 \;\; \ldots \;\; \beta_n)$, then:

$$\begin{aligned} (\alpha \;\; \beta_1 \ldots \;\; \beta_n) \text{ is true in } \mathcal{I} \quad &\text{iff} \quad \langle d_V(\beta_1), \ldots, d_V(\beta_n) \rangle \in e_{rel}(d_V(\alpha)) \\ &\text{iff} \quad \langle \overline{\beta_1}, \ldots, \overline{\beta_n} \rangle \in \{\langle \overline{\gamma_1}, \ldots, \overline{\gamma_n} \rangle : (\alpha \;\; \gamma_1 \;\; \ldots \;\; \gamma_n) \in \Delta\} \\ &\text{iff} \quad (\alpha \;\; \beta_1 \;\; \ldots \;\; \beta_n) \in \Delta \text{ (by schemas } \mathbf{A5} \text{ and } \mathbf{A6}). \end{aligned}$$

For conventional L (i.e., languages where '=' is not a name and, hence, does not have a relation extension), the atomic case for identities is similar. And, again, the boolean cases are straightforward. So assume for our induction hypothesis that the condition of the lemma holds for ψ and that φ is $(\forall\mu\ \ \psi)$:

$$
\begin{aligned}
(\forall\mu\ \ \psi) \text{ is true in } \mathcal{I} \quad & \text{iff} \quad \text{for all } a \in D,\ \psi \text{ is true in } \mathcal{I}^{\nu}_{a} \\
& \text{iff} \quad \text{for all } a \in \{\overline{\alpha} : \alpha \text{ is a term of } L_K\},\ \psi \text{ is true in } \mathcal{I}^{\nu}_{a} \\
& \text{iff} \quad \text{for all } a \in \{\overline{\kappa} : \kappa \in K\},\ \psi \text{ is true in } \mathcal{I}^{\nu}_{a} \text{ (by Lemma 11)} \\
& \text{iff} \quad \text{for all } \kappa \in K,\ \psi \text{ is true in } \mathcal{I}^{\nu}_{\overline{\kappa}} \\
& \text{iff} \quad \text{for all } \kappa \in K,\ \psi \text{ is true in } \mathcal{I}^{\nu}_{V(\kappa)} \\
& \text{iff} \quad \text{for all } \kappa \in K,\ \psi^{\nu}_{\kappa} \text{ is true in } \mathcal{I} \text{ (by Theorem 2)} \\
& \text{iff} \quad \text{for all } \kappa \in K,\ \psi^{\nu}_{\kappa} \in \Delta \text{ (by our induction hypothesis)} \\
& \text{iff} \quad (\forall\mu\ \ \psi) \in \Delta \text{ (by Lemma 12).}
\end{aligned}
$$

□

The Model Existence Lemma 6 now follows immediately:

Proof of Lemma 6. Since $\mathcal{I} = \langle D, e_{rel}, e_{fn}, V\rangle$ is an L_K-model of Δ and $\Gamma \subseteq \Delta$, $\mathcal{I}$ is obviously an L_K-model of Γ. The L-interpretation $\langle D, e_{rel}, e_{fn}, V \restriction N_L\rangle$ that results simply from restricting the denotation function V to the set N_L of names of L is therefore an L-model of Γ. □

Theorem 5. (Completeness) *If* $\Gamma \models_L \varphi$*, then* $\Gamma \vdash_{C_L} \varphi$.

Proof. Suppose $\Gamma \nvdash_{C_L} \varphi$. By Corollary 2, $\Gamma \cup \{(\texttt{not}\ \ \varphi)\}$ is C_L-consistent. Hence, by Lemma 6, $\Gamma \cup \{(\texttt{not}\ \ \varphi)\}$ has an L-model $\mathcal{M}$. So there is an L-model of Γ in which φ is false, i.e., $\Gamma \not\models_L \varphi$. □

As a direct consequence of the Model Existence Lemma we have:

Corollary 5. (Löwenheim-Skolem Theorem) *If a set* Γ *of sentences of L is L-satisfiable, it has a countable L-model.*

Proof. Suppose Γ is L-satisfiable. By Corollary 4, Γ is C_L-consistent. Thus, letting it be the set of sentences in the proof of the Lemma 6, the domain D of the L-model of Γ constructed in that proof is that of the L_K-model constructed in Lemma 12. But D is a partition of the (denumerable) set of terms of the language L_K and, hence, is no larger than denumerable. □

Compactness follows as usual:

Corollary 6. (Compactness) *If every finite subset of a set* Γ *of sentences is satisfiable, then* Γ *is satisfiable.*

Proof. Suppose Γ is not satisfiable. Then by Lemma 6, Γ is inconsistent and, hence, by Lemma 3, $\Gamma \vdash \bot$. But again, as in Lemma 10, proofs are finite. Hence, for some finite $\Gamma' \subseteq \Gamma$, $\Gamma' \vdash \bot$ and so, by Lemma 3 once again, Γ' is inconsistent. Hence, by Corollary 4, Γ' is unsatisfiable. So not every finite subset of Γ is satisfiable. □

Given the preceding corollaries, by a well-known theorem of Lindström (1969),[19] it follows that, its "second-order" quantifiers notwithstanding, the logical framework of CLIF is entirely first-order.[20]

7.2.1 Completeness for $\mathbf{C}_L^+$

Given Lemma 5, the only thing that has to be done to turn the above proof into a completeness proof C_L^+ (beyond changing semantic notions to their +-counterparts and references to C_L to C_L^+) is to show — assuming now axioms **A1–A8** — that the structure $\mathcal{I} = \langle D, e_{rel}, e_{fn} V\rangle$ defined in the manner following the proof of Lemma 12 satisfies the principles **M** and **G** and, hence, is in fact an L_K-interpretation$^+$. This is guaranteed by the corresponding axiom schemas **A7** and **A8**.

Lemma 15. *Condition **M** holds in* $\mathcal{I}$.

Proof. Let $a \in D$ and suppose $e_{rel}(a)$ is variably polyadic and $\langle b_1, ..., b_{n+1}\rangle \in e_{rel}(a)$. Then there are terms α, γ_1, ..., γ_{n+1} of L_K, such that a is $\overline{\alpha}$, each b_i is $\overline{\gamma_i}$, and $(\alpha\ \ \gamma_1\ \ ...\ \ \gamma_{n+1}) \in \Delta$. Since $e_{rel}(a)$ is variably polyadic, by construction we must

[19]Strictly speaking Lindström's theorem does not apply to CLIF since the theorem is proved with regard to languages with fixed signatures. But there seems to be no reason in principle why the theorem wouldn't generalize to include CL languages. Alternatively, we can appeal to the translation scheme from a CLIF language into a "traditional" first-order language discussed in the following section and apply Lindström's theorem to CLIF indirectly.

[20]In a bit more detail: The semantics of full second-order logic requires that, in any interpretation, n-place predicate quantifiers range over the entire power set $\wp(D^n)$ of the n^{th}-Cartesian product of the domain D of individuals. (A similar requirement holds if the language contains n-place function quantifiers.) If, as in so-called *general semantics* for second-order logic, or *Henkin semantics* (after Henkin 1950), that requirement is lifted, the expressive power of the logic becomes essentially first-order; see, e.g., Enderton (2001, §4.4). It is easy to see that, in the semantics of CL, quantified variables in predicate (or function) position range, in effect, over a subset of $\wp(D^*)$ no larger than D and, hence, in general, a "second-order" domain much smaller than the full power set.

have $(\alpha\ \sigma_1\ \dots\ \sigma_m) \in \Delta$, for some m ($\neq n+1$) terms σ_1, ..., σ_m. As Δ is both L_k-maximal and $C^+_{L_K}$-consistent, by (the $C^+_{L_K}$ analog of) Lemma 10, the following instance of axiom schema **A7** must be in Δ:

$$(\texttt{if}\ (\alpha\ \sigma_1\ \dots\ \sigma_m)\ (\texttt{if}\ (\alpha\ \gamma_1\ \dots\ \gamma_{n+1})\ (\alpha\ \gamma_1\ \dots\ \c{t}\gamma_n))).$$

Hence, $(\alpha\ \gamma_1\ \dots\ \gamma_n) \in \Delta$. So, by definition of e_{rel}, $\langle b_1, \dots, b_n \rangle \in e_{rel}(a)$. □

Lemma 16. *Condition* ***G*** *holds in* $\mathcal{I}$.

Proof. Suppose for at least one c, $\langle b_1, \dots, b_n, c \rangle \in e_{rel}(a)$. Then there are terms α, β_1, ..., β_n, ν such that a is $\overline{\alpha}$, b_i is $\overline{\beta_i}$, c is $\overline{\nu}$, and $(\alpha\ \beta_1\ \dots\ \beta_n\ \nu) \in \Delta$. Since Δ is L_k-maximal and $C^+_{L_K}$-consistent, the instance

$$(\texttt{if}\ (\alpha\ \beta_1\ \dots\ \beta_n\ \nu)\ (\alpha\ \beta_1\ \dots\ \beta_n\ (\alpha\ \beta_1\ \dots\ \beta_n)))$$

of schema **A8** must be in Δ as well. Hence, $(\alpha\ \beta_1\ \dots\ \beta_n\ (\alpha\ \beta_1\ \dots\ \beta_n)) \in \Delta$. □

The proof of completeness and the proofs of Löwenheim-Skolem and compactness for C^+_L now proceed exactly as they do for C_L.

8 CL and Traditional First-order Logic (TFOL)

There is a fairly obvious method of translating the sentences of a *conventional* CLIF language L (i.e., a language in which '=' is not a name) directly into sentences of a "traditional" first-order language that, in a precise sense, have the same meaning. Our central interest in this translation method for purposes here is that, given a few lemmas, the method yields a separate completeness theorem for C_L as a corollary of the completeness of traditional first-order logic (TFOL). Nonetheless, in light of this method, one has the option of considering the sentences of CLIF simply to be somewhat compressed expressions of their traditional counterparts. We close with some brief reflections on this and other issues concerning the connection between CL and TFOL in §8.3.

8.1 Translating CLIF into TFOL

In this section we introduce the notion of a *traditional counterpart* L^* of a conventional CLIF language L, that is, a first-order language whose lexicon consists of the "traditional" syntactic categories — individual constants and variables, n-place predicates and function symbols — and whose grammar exhibits the concomitant restrictions on the formation of terms and sentences. We then establish the rather obvious way of translating from L to L^* and show that this translation scheme is in a robust way "meaning-preserving". In the following section, we show how to exploit these results to sketch our alternative proof of the completeness of C_L in §8.2.[21]

8.1.1 *L*'s Traditional Counterparts

Let L be a conventional CLIF language. The lexicon of a *traditional counterpart* L^* of L consists of the same logical operators `not`, `if`, and `forall` (written again as $\forall$) as well as the following:

- The set N_L of names of L, which are known as the *individual constants* of L^*.
- For every $n \in \mathbb{N}$, an $n+1$-place predicate $\mathtt{Holds}_n$
- For every $n \in \mathbb{N}$, an $n+1$-place function symbol App_n.
- A denumerable set Var_{L^*} of names (in the sense of §2.1) disjoint from N_L and not containing the predicates and function symbols above. These are the *variables* of L^*.

We will use a CLIF-like grammar for L^*, albeit with the traditional syntactic categories and restrictions and a traditional model theory. The *terms* of L^* are the individual constants and variables of L^* together with those expressions of L^* of the form $(\mathsf{App}_n\ \alpha\ \beta_1\ \ldots\ \beta_n)$, for terms $\alpha,\ \beta_1,\ \ldots,\ \beta_n$ of L^*. The *formulas* of L^* are those expressions of the form $(\mathtt{Holds}_n\ \alpha\ \beta_1\ \ldots\ \beta_n)$ for terms $\alpha,\ \beta_1,\ \ldots,\ \beta_n$ of L^* and, for formulas φ, ψ of L^*, those expressions of the form $(\mathtt{not}\ \varphi)$, $(\mathtt{if}\ \varphi\ \psi)$, and $(\mathtt{forall}\ (\chi)\ \varphi)$ (written, again, as $(\forall\chi\ \varphi)$), for variables χ of L^*. Traditional counterparts of L thus differ only in their choice of variables.

The notion of an L^*-interpretation I will be understood as usual as a pair $\langle D, U\rangle$ consisting of a domain D and an interpretation function U on the basic lexicon of L^* such that $U(\nu) \in D$, for individual constants $\nu \in N_L$ and, for $n \in \mathbb{N}$, $U(\mathsf{App}_n)$ is an $n+1$-place function $f : D^{n+1} \longrightarrow D$ and $U(\mathtt{Holds}_n) \subseteq D^{n+1}$. Truth in I is

[21] Chen et al. (1993) define a similar method for translating HiLog into a traditional first-order language L^* and prove a similar "meaning preservation" theorem. This enables them to prove the completeness of a resolution-based proof theory for HiLog that is analogous to our "indirect" completeness proof below in §8.2.

defined as usual by way of satisfaction by a variable assignment, that is a function $s : Var_{L^*} \longrightarrow D$.

8.1.2 Standard Translations

Let L^* be a traditional counterpart of L. The translation scheme from L into L^* is straightforward. Let x be a fixed one-to-one correspondence from the set N_L of names of L onto Var_{L^*}; for convenience, we shall write x_ν instead of $x(\nu)$. We define two mappings $'$ and $\dagger$, where $'$ maps the terms of L to terms of L^* and $\dagger$ maps the sentences of L to sentences of L^* as follows:

- For names $\nu \in N_L$, $\nu' = \nu$
- For terms $\alpha, \beta_1, ..., \beta_n$ of L,
  - $(= \ \beta_1 \ \ \beta_2)^\dagger = (= \ \beta_1' \ \ \beta_2')$
  - $(\alpha \ \ \beta_1 \ \dots \ \beta_n)' = (\mathtt{App}_n \ \ \alpha' \ \ \beta_1' \ \dots \ \beta_n')$
  - $(\alpha \ \ \beta_1 \ \dots \ \beta_n)^\dagger = (\mathtt{Holds}_n \ \ \alpha' \ \ \beta_1' \ \dots \ \beta_n')$
- For sentences φ,ψ of L and $\nu \in N_L$,
  - $(\mathtt{not} \ \ \varphi)^\dagger = (\mathtt{not} \ \ \varphi^\dagger)$
  - $(\mathtt{if} \ \ \varphi \ \ \psi)^\dagger = (\mathtt{if} \ \ \varphi^\dagger \ \ \psi^\dagger)$
  - $(\forall \nu \ \ \varphi)^\dagger = (\forall x_\nu \ \varphi^{\dagger}{}^{\nu}_{x_\nu})$

Let us call the pair $\langle ', \dagger \rangle$ of functions a *standard translation of L into L^**.[22] The mapping $\dagger$ is not surjective, as no sentence of L^* in which a name ν occurs in the scope of a quantifier binding its corresponding variable x_ν is in the range of $\dagger$. However, for any such sentence there will be an alphabetic variant that is in the range; more exactly:

Lemma 17. *For every sentence ψ of L^* there is a sentence φ of L such that ψ and $\varphi^\dagger$ are alphabetic variants.*

[22]Standard translations differ only in their choice of the one-to-one correspondence x from N_L onto Var_{L^*}. It should be clear that nothing essential hangs on this, as the standard translations determined by distinct correspondences x, y are identical up to alphabetic variance and, hence, cannot be differentiated model-theoretically.

Proof. By induction on the number of quantifiers and connectives in ψ. The atomic and boolean cases are trivial. So suppose ψ is $(\forall x_\nu\ \theta)$. Let $\mu \in N_L$ be such that neither it nor x_μ occurs in θ. By our induction hypothesis, there is a sentence ξ of L such that $\xi^\dagger$ and $\theta^{x_\nu}_{\mu}$ are alphabetic variants. Note that, since $\xi^\dagger$ is in the range of $\dagger$, no occurrence of μ in $\xi^\dagger$ is in the scope of a quantifier binding x_μ. Thus, $(\forall x_\mu\ \xi^{\dagger}{}^{\mu}_{x_\mu})$ and $(\forall x_\mu\ \theta^{x_\nu}_{x_\mu})$ must be alphabetic variants as well. But $(\forall x_\mu\ \theta^{x_\nu}_{x_\mu})$ and $(\forall x_\nu\ \theta)$ are alphabetic variants (by choice of x_μ); so, therefore, are $(\forall x_\nu\ \theta)$ and $(\forall x_\mu\ \xi^{\dagger}{}^{\mu}_{x_\mu})$ which, by definition, is $(\forall x_\mu\ \xi^{\mu}_{x_\mu})^\dagger$. □

8.1.3 Standard Translations are Meaning-preserving

The idea behind the standard translation scheme should be clear. As noted in §1 above, in the semantics of CL, any object can play the role of a (variably polyadic) function or relation. The translation scheme in a sense does not alter the underlying picture but simply expresses it differently in terms of more traditional function symbols and predicates. Thus, intuitively, App_n applied to any objects a, b_1, ..., b_n yields exactly the object c that a, *qua* function, yields when applied to those objects; likewise, $\mathtt{Holds}_n$ is true of a, b_1, ..., b_n if and only if a, *qua* relation, holds of b_1, ..., b_n. Because there is no restriction on the arguments to App_n and $\mathtt{Holds}_n$, it is clear that they preserve the idea that any objects can be the first element of a string of arguments and hence play either a functional or relational role in an atomic formula. Furthermore, the variable polyadicity of functions and relations is reflected in the fact that the same object a can be the first argument to App_n and App_m, or $\mathtt{Holds}_n$ and $\mathtt{Holds}_m$, for any m and n.

The semantic connection between L and L^* is expressed more rigorously by saying that every L-interpretation $I = \langle D, e_{rel}, e_{fn}, V\rangle$ determines a unique L^*-interpretation $\mathcal{I}^* = \langle D, V \cup W_{\mathcal{I}}\rangle$ where, for function symbols App_n and predicates $\mathtt{Holds}_n$ of L^*,

- $W_{\mathcal{I}}(\mathsf{App}_n) = \bigcup\left\{\{a\} \times (e_{fn}(a) \upharpoonright D^n) : a \in D\right\}$;[23]
- $W_{\mathcal{I}}(\mathtt{Holds}_n) = \bigcup\{\{a\} \times (e_{rel}(a) \cap D^n) : a \in D\}$.

Moreover, it should be clear that *every* L^*-interpretation is so determined by some (unique) L-interpretation. For the interpretation function U in any L^* interpretation $\mathcal{J} = \langle D, U\rangle$ can be split into a function V on the individual constants of L^* and another W on the function symbols and predicates of L^*. We then let

[23] I assume here that $\langle a,b\rangle = \{\{a\},\{a,b\}\}$ and that, for $n > 2$, the ordered n-tuple $\langle a_1, a_2 ..., a_n\rangle = \langle a_1, \langle a_2 ..., a_n\rangle\rangle$. So, in particular, as $e_{fn}(a) \upharpoonright D^n$ is a function and, hence, a set of ordered $n+1$-tuples, $\{a\} \times (e_{fn}(a) \upharpoonright D^n)$ is a set of ordered $n+2$-tuples. It should also be clear that, since $e_{fn}(a) \upharpoonright D^n$ is a function, so too is $\{a\} \times (e_{fn}(a) \upharpoonright D^n)$.

- $e_{fn} = \bigcup\{W(\mathtt{App}_n) : n \in \mathbb{N}\}$
- $e_{rel} = \bigcup\{W(\mathtt{Holds}_n : n \in \mathbb{N}\}$.

It is easy to check that $\langle D, e_{fn}, e_{rel}, V\rangle$ is an L-interpretation and that it yields $\mathcal{J}$ under the above mapping. So the mapping above is in fact a one-to-one correspondence.

That a standard translation scheme $\langle{}', \dagger\rangle$ is meaning-preserving can now be expressed in an easy lemma:

Lemma 18. *For sentences φ and interpretations $\mathcal{I} = \langle D, e_{rel}, e_{fn}, V\rangle$ of L, φ is true in $\mathcal{I}$ iff $\varphi^\dagger$ is true in $\mathcal{I}^*$=$\langle D, V \cup W_{\mathcal{I}}\rangle$.*

Proof. A simple induction on the complexity of terms yields that, for terms τ of L, $V(\tau) = V \cup W_{\mathcal{I}}(\tau')$. The lemma is then proved by a straightforward induction on φ. □

For a set Γ of sentences of L, let $\Gamma^\dagger = \left\{\varphi^\dagger : \varphi \in \Gamma\right\}$ and let $\models_{L^*}$ indicate the entailment relation for the TFOL model theory of L^*. Then as a corollary to the preceding lemma we have:

Corollary 7. *For sentences φ of L, $\Gamma \models_L \varphi$ if and only if $\Gamma^\dagger \models_{L^*} \varphi^\dagger$.*

In particular, φ is a logical truth of L if and only if $\varphi^\dagger$ is a logical truth of L^*.

Note that Corollary 7 fails if L is inclusive or if we replace $\models_L$ with its +-counterpart $\models_L^+$ as, in each case, there are validities of the relevant semantics whose †-translations are invalid in TFOL. If L is inclusive, then, since '=' is a name, `(∃r (r x x))`, for example, follows from `(= a a)` and, hence, is a valid formula of L. But `(∃x (r x x))`† is `(∃x (Holds`$_2$` r x x))`, which is not valid in TFOL with its standard semantics; likewise the translations $\varphi^\dagger$ of instances φ of the valid$^+$ axiom schemas **A7** and **A8**.

8.2 Completeness by Way of TFOL

With a bit of hand-waving, the preceding lemmas yield an alternative completeness proof for C_L. The argument needs a few more lemmas. First, let C_{L^*} be the TFOL system whose axioms are the standard translations of the axioms of C_L and their alphabetic variants. C_{L^*} is a standard axiomatization of TFOL and, hence, is complete. Let $\vdash_{\mathrm{C}_{L^*}}$ be the provability relation for C_{L^*}.

Lemma 19. *For any sentence ψ of L^* and any set Σ of sentences of L^*, if $\Sigma \vdash_{C_{L^*}} \psi$, then there is a proof of ψ from Σ consisting entirely of sentences of L^* (i.e., formulas of L^* in which no variables occur free).*

Proof. Since ψ and the members of Σ are all sentences, any variable that occurs free in a proof Π of ψ must have been introduced by an instance of one of our axioms. For each variable χ that occurs free in Π, let ν_χ be a name that does not occur in ψ or any member of Σ, or in any formula occurring in Π and suppose that, for distinct variables χ and χ' occurring Π, $\nu_\chi \neq \nu_{\chi'}$. Let Π' be the result of replacing every free occurrence of a variable χ in Π with ν_χ. A simple induction on the length of Π shows that Π' is also a proof of ψ from Σ. □

Lemma 20. *If ψ_1, ..., ψ_n is a proof in C_{L^*} of $\varphi^\dagger$ from $\Gamma^\dagger$, then there are sentences φ_1, ,..., φ_n of L such that $\varphi_1^\dagger$, ,..., $\varphi_n^\dagger$ is a proof of $\varphi^\dagger$ from $\Gamma^\dagger$ in C_{L^*}.*

Proof. By the preceding lemma we may assume that the ψ_i are all sentences. As noted above, the only sentences of L^* that are not in the range of the translation scheme are those in which a name ν occurs in the scope of a quantifier binding the corresponding variable x_ν. A judicious replacement of such sentences with alphabetic variants that are in the range of the translation scheme, as guaranteed by Lemma 17, will yield the desired result. □

Finally, we note that the result of transforming a C_{L^*} proof as in the preceding lemma yields a corresponding proof in C_L:

Lemma 21. *If $\varphi_1^\dagger,...,\varphi_n^\dagger$ is a proof from $\Gamma^\dagger$ in C_{L^*}, then $\varphi_1,...,\varphi_n$ is a proof from Γ in C_L.*

Proof. Suppose so for all C_{L^*} proofs of length $< n$. $\varphi_n^\dagger$ is either a member of $\Gamma^\dagger$, an axiom of C_{L^*}, or follows by MP. The first case is trivial, as that means $\varphi \in \Gamma$. In the second case we are done as well, as the axioms of C_{L^*} are the translations of C_L axioms or alphabetic variants thereof. For the last case we need only observe that there must be $i, j < n$ such that $\varphi_j^\dagger = (\texttt{if}\ \ \varphi_i^\dagger\ \ \varphi_n^\dagger)$ and that, by definition, $(\texttt{if}\ \ \varphi_i^\dagger\ \ \varphi_n^\dagger) = (\texttt{if}\ \ \varphi_i\ \ \varphi_n)^\dagger$. Given our induction hypothesis, $\varphi_1,...,\varphi_{n-1}$ is a proof from Γ and it contains φ_i and φ_j (= $(\texttt{if}\ \ \varphi_i\ \ \varphi_n)$), so $\varphi_1,...,\varphi_n$ is a proof from Γ as well. □

As an immediate corollary we have:

Corollary 8. *If $\Gamma^\dagger \vdash_{C_{L^*}} \varphi^\dagger$, then $\Gamma \vdash_{C_L} \varphi$.*

Our result now follows:

Alternative proof of Completeness of C_L. If $\Gamma \models_L \varphi$, then by Corollary 7 $\Gamma^\dagger \models_{L^*} \varphi^\dagger$. Hence, by the completeness of C_{L^*}, we have $\Gamma^\dagger \vdash_{\mathrm{C}_{L^*}} \varphi^\dagger$ and thus, by Corollary 8, $\Gamma \vdash_{\mathrm{C}_L} \varphi$. □

In light of the observations in the final paragraph of the preceding subsection, this proof breaks down if L is inclusive or if we replace C_L with C_L^+ (hence $\models_L$ with $\models_L^+$). Hence, completeness under those conditions cannot be proved by this method without substantial modifications to the proof theory and semantics of TFOL — in which case it would no longer be *traditional* FOL.

8.3 Brief Reflections on CL and TFOL

The existence of a general translation method from CL dialects to TFOL prompts a few reflections. A pragmatic benefit of the method is that it enables users of CL immediately to take advantage of the many powerful first-order theorem provers to support automated reasoning upon knowledge bases written in CL dialects.

More theoretically speaking, although the method shows that CL is, with regard to expressiveness, theoretically superfluous, CL still arguably has some significant philosophical advantages. First, despite the fact that, intuitively, the Holds predicates and App function symbols express exemplification and function application, they are in fact logically vacuous — given a CLIF language L, there are no constraints on the semantic values assigned to the Holds predicates and App function symbols in a traditional counterpart of L and, hence, they require no axioms. This suggests that they are mere "syntactic sugar" for the conceptually more fundamental syntax of CL.

More generally, I would argue that the traditional, highly typed syntax of TFOL is a vestige of an (arguably) outmoded Fregean metaphysics that stipulates an inviolable gulf between individuals and properties and, likewise, between n-place and m-place relations (for $m \neq n$) — the bountiful linguistic evidence of nominalization and variable polyadicity notwithstanding. Perhaps worse still, Holds/App syntax reflects a sort of schizophrenia with regard to relations: $\mathtt{Holds}_n$ is a predicate and $\mathtt{App}_n$ is a function symbol; hence, they express an n-place relation and an n-place function, respectively. However, in its intended interpretation, the first argument of $\mathtt{Holds}_n$ is itself and n-place relation. Moreover, since, as noted above, these predicates and function symbols are conceptually vacuous, their function is, effectively, simply to *suppress* the predicative nature of relations and the applicative nature of functions that occupy their first argument places. They exist only for those who would acknowledge the type-free, variably polyadic nature of relations and functions but resist its most natural expression in favor of the familiar and the traditional.

It is interesting to note that, while in a certain sense equivalent to TFOL, CL's non-Fregean perspective brings with it some changes in our notion of logical truth. Say that a sentence φ of CLIF is *traditional first-order* (*TFO*) if none of CL's distinctive features are present in φ — no apparent violations of traditional type distinctions (as in, e.g., `(F F)`), no evidence of variable polyadicity (as in, e.g., `(and (F a) (F a b))`), and no higher-order quantification. One might expect that the TFO sentences of CLIF retain all of the traditional logical properties of their TFOL counterparts. But this is not the case. A particularly interesting example was first brought to light in Horrocks and Patel-Schneider 2003. Clearly, things that, *qua* properties, are true of different things cannot be identical. Accordingly,

`(if (∃x (and (F x) (not (G x)))) (not (= F G)))` (9)

is a valid sentence of CLIF (hence also a theorem of C_L, by Theorem 5). (9) is of course not TFO, as it exhibits type-freedom. However, its consequent obviously implies that there are two things. Hence, the following is also a validity of CL:

`(if (∃x (and (F x) (not (G x)))) (∃xy (not (= x y))))` (10)

Unlike the preceding sentence, however, (10) is clearly TFO. But, obviously, as F and G would be predicates in a TFOL language L' in which (10) is a sentence, they would not denote members of the domain of any interpretation of L'. Hence, (10) is not a valid sentence of L'; there are interpretations of L' in which the antecedent of (10) is true but which contain only one thing and, hence, interpretations in which (10) is false.

Attitudes toward the existence of "Horrocks sentences" like (10) vary in accordance with the importance one places upon the idea that the logical properties of a TFO sentence φ ought not to differ from those of a sentence with the identical form in a different first-order logic (so long, of course, as the logics agree on the semantics of the logical operators of φ). Those who are particularly distressed by their existence can perhaps take some comfort in the translation method of the preceding section. But if we accept the implications of our reflections above — that CL's is in fact the most natural and general syntax for logic — then we ought simply to acknowledge that the true logical properties of Horrocks sentences like (10) are simply lost in the Fregean framework of TFOL.

References

Bealer, G. (1982). *Quality and Concept*. Oxford University Press.

Chen, W., Kifer, M., and Warren, D. (1993). HiLog: A Foundation for Programming for Higher-Order Logic. *Journal of Logic Programming*, 15:187–230.

Cocchiarella, N. B. (1972). Properties as Individuals in Formal Ontology. *Nous*, 6(2):165–187.

Enderton, H. (2001). *A Mathematical Introduction to Logic*. Academic Press, San Diego, 2nd edition.

Genesereth, M. (1998). Knowledge Interchange Format: Draft Proposed American National Standard, NCITS.T2/98-004. Technical report, URL = http://logic.stanford.edu/kif/dpans.html.

Henkin, L. (1950). Completeness in the Theory of Types. *Journal of Symbolic Logic*, 15:81–91.

Hilbert, D. (1922). Die logischen Grundlagen der Mathematik. *Mathematische Annalen*, 88(1-2):151–165.

Horrocks, I., Parsia, B., Patel-Schneider, P., and Hendler, J. (2005). Semantic Web Architecture: Stack or Two Towers? In Fages, F. and Saliman, S., editors, *Principles and Practice of Semantic Web Reasoning, Lecture Notes in Computer Science, Vol. 3703*, pages 37–41. Springer, Berlin.

Horrocks, I. and Patel-Schneider, P. F. (2003). Three Theses of Representation in the Semantic Web. In *Proceedings of the Twelfth International World Wide Web Conference*, pages 39–47.

ISO/IEC JTC 1 SC 32 (2007). Common Logic (CL): A Framework for a Family of Logic-based Languages. Technical report, International Organization for Standardisation, Geneva, Switzerland.

Lindström, P. (1969). On Extensions of Elementary Logic. *Theoria*, 35(1):1–11.

Menzel, C. (1986). A Complete, Type-free, "Second-order" Logic and Its Philosophical Foundations. Technical Report CSLI–86–40, Center for the Study of Language and Information, Stanford University, Stanford, CA. http://cslipublications.stanford.edu/TechReports/CSLI-86-40.pdf.

Menzel, C. (2011). Knowledge Representation, the World Wide Web, and the Evolution of Logic. *Synthese*, 182(2):269–295.

Menzel, C. and Hayes, P. (2003). SCL: A Logic Standard for Semantic Integration. In Doan, A., Halevey, A., and Noy, N., editors, *CEUR Workshop Proceedings, Vol. 82: Semantic Integration*. http://ceur-ws.org/Vol-82/.