Possible Logical Foundations for Physical Sciences

Paul C Gilmore

September 08

Abstract

There are several motivations for this paper, the most specultive being the one suggested by its title. The thesis of logicism that Frege, Russell and others have advanced is that the foundations of mathematical truth lies in an understanding of the logical truths that underly it. Since theoretical sciences have become more and more dependent on mathematical exposition, it is natural to ask if an understanding of modern physics is not also dependent on an understanding of logical truths. A hint of this belief can be found in a recent book by Smolin. This paper speculates on how the connection might be made more evident.

But there are other more solidly founded motivations. The author has advanced the thesis over the past decade that the source of the logical paradoxes lie in a confusion of use and mention. Logics have been described that make this distinction while providing a consistent purely logical development of recursion theory and theories of natural and real numbers. In this paper novel higher order classical and intuitionist logics CSL and HSL are defined that make no use of quantification variables but only abstraction variables for use with the λ-abstraction operator. They provide a further substantiation of the the thesis of logicism.

1 Introduction

The origins of the logics CSL, Classical Symbolic Logic, and HSL, Heyting Symbolic Logic, are in a classical second order logic NaDSyl, Natural Deduction based Symbolic Logic, [6, 7, 8] with two novel features. To permit an unlimited use of the λ-abstraction operator, abstaction variables were introduced, in addition to the usual first and second order variables. Further, second order terms without free second order variables were admitted as first order terms to substantiate the thesis that the source of the paradoxes of set theory lies in an abuse of use and mention, a thesis advanced earlier by Wilfred Sellars in [17] and [18]. In order not to unnecessarily restrict the use of the λ-abstraction operator, to be a sentence M of the logic was identified with the derivability of the sequent $M \vdash M$ with the sequent assumed derivable when M is atomic. Consequently all sentences of NaDSyL could be ultimately reduced to atomic. Applications of the logic in sciences based on physical observations require that the atomic sen-

tences be thought of as decidable by observation; thus the sentences of NaDsyL can be thought of as potentially *grounded* in a physical observations.

CSL provides both a simplification and a generalization of NaDSyL. The simplification removes all quantification variables and replaces them with abstraction variables only. The generalization results from extending the logic to a complete type theory. The original motivation for NaDSyL is maintained, namely to explore unlimited abstraction within a logic in which consistency is maintained by carefully distinguihing between use and mention, and in which all sentences are *grounded*. HSL is an intuitionist version of CSL with semantics motivated by Kripke's semantics for first order logic.

The relevence of HSL to the motivation of this paper expressed in its title is in the following quote from pages 30 and 31 of Smolin's [19]: "In the history of Physics it has often happened that by the time the physicists had been able to understand the need for a new mathematics, they found that the mathematicians had got there first and had already invented it. ... The mathematicians, it seems, were not aware that they were inventing the right form of logic for cosmology, so they called it other names. In its first forms it was called 'intuitionist' logic. More sophisticated versions which have been studied more recently are known as 'topos theory'."

It appears to be Kripke's multiple world models of intuitionist logic [13] that might make intuitionist logic and its algebraic derivatives useful in quantum cosmology. That HSL might be useful has yet to be explored. That it might be of some value depends not only on its Kripke semantics, but also on the nominalist motivation for the distinction between use and mention that ensures its consistency. Indeed the disputes between Einstein's 'realism' and quantum theorists' views of 'reality' as described in chapter 20 of Isaacson's [12] can be seen as arising from different views of language described by Quine in [16].

A secondary motivation for this paper is to respond to the recent paper [1] by Jamie Andrews. Andrews defined a logic by simplifying NaDSyL in several ways including removing abstraction vaiables. He proved that his logic, like NaDSyL, is consistent; but in addition that there are sentences of his logic that are not sentences of NaDSyL. Among such sentences is one YN in which the Y operator of Curry is used. It is shown in §3.2.2 that Andrew's logic fails to satisfy the criterion of all sentences being grounded, since any attempt to provide a derivation in CSL for the sequent $\mathrm{YN} \vdash \mathrm{YN}$ cycles.

The elementary syntax for both CSL and HSL is defined in §2. The semantics and proof theory for CSL are decribed in §3.1 and §3.2, and for HSL in §4.1 and §4.2. The proof theories of both logics are adaptions of Beth's method of semantic tableaux as used in [6, 7, 8, 9]; an excellent description of the method for first order classical logic is Hodges [11]. Their consistency is proved in §3.2.2 and some examples given of significant sequents that are derivable in both both logics. Sketches are given of the redundancy of Cut for CSL in §3.2.3 and HDL in §4.2.4. Finally in §5 the nominalist motivation for CSL and HSL is sketched along with its relevance for logic and the physical sciences.

2 Elementary Syntax

There are two novel features of the elementary syntax of CSL and HSL described here. Although the logics have quantifiers $\exists$ and $\forall$ over all types, only abstraction variables, used by the abstraction operator λ, appear in the logics. Further, any higher order term in which no unspecified predicate name, called a *free* predicate name, occurs is also a first order term. This feature is justified by a careful distinction between the use and mention of predicate names. Both features are evident in definition 4 of §2.2, and result in a third unusual feature, namely that the syntax defining terms is not decdable, or using a commonly used terminology, is *semi-decidable*.

2.1 Types

The importance of the arity of a predicate name, that is the number of its arguments, is evident: Giving too many or too few or irrelevent arguments to a predicte name results in a string of characters without meaning. As will be seen, applications of the abstraction operator λ can result in terms that will accept as arguments terms of any arity that may themselves be predicates of any arity, resulting in the need for a more general form of arity called *type*.

Definition 1 *Type*
- *1 is the only type of* objects *and* $[\,]$ *is a* predicate *type;*
- *If* σ *is 1 or a predicate type, and* τ *a predicate type, then* (σ, τ) *is a predicate type.*

$[\tau_1, \ldots, \tau_k]$ *abbreviates* $(\tau_1, \ldots (\tau_k, [\,]) \ldots)$. *Thus if* τ *is* $[\ldots]$. *then* (σ, τ) *is abbreviated by* $[\sigma, \ldots]$. *Further,* $[k]$ *abbreviates* $[1, \ldots, 1]$ *where there is a total of* $k, 0 \leq k$, *1's in the sequence* $1, \ldots, 1$.

Apart from notation and the distinction between object and predicate types, this definition repeats the definition A.1.1 in Appenix A, Typed Lambda Calculus, of Barendregt's monograph [2]. The difference in notation reflects a difference in interpretation. The abbreviated type $[\tau_1, \ldots, \tau_k]$ in the abbreviatted Barendregt notation, is written $(\tau_1 \rightarrow \ldots \tau_k \rightarrow [\,])$ where $[\,]$ is the type of the truth values. It is the type of a function with arguments the types $\tau_1, \tau_2, \ldots, \tau_k$, and value of type $[\,]$; that is the type of a *predicate*.

For example, [1,[2],1,1,1] is the type of a predicate; namely the type of a predicate with first argument of type 1, second of type [2], and third, fourth and fifth arguments each of type 1. That the type of the second argument is [2] means that the second argument has arity 2 and therefore has two arguments each of type 1. There is no limit to the depth of nesting of '[' and ']' allowed. For example a type that later will be seen to be involved in inductive or recursive definitions is $[[[1], 1], 1]$.

The notation $N{:}\tau$ used throughout the paper states that N is a term with the type τ.

Predicate and object names are fundamental ingredients of the syntax of the logics.

Definition 2 *Predicate and Object Names*
A predicate name is any string of upper and lower case letters and numerals beginning with any uppercase letter other than U, V, W, X, Y *and* Z*. The set of all such predicte names is denoted by* NM*. A* free *predicate name is any* $Nm \in$ NM *typed in* ***bold****, as for example* ***Nm****. The set of all such free predicte names is denoted by* FNM*.*

An object name *is a string of any number of lower case letters and numerals with any first letter other than u, v, w, x, y, and z. The set of all such object names is denoted by* NM*. A* free *object name is any* $nm \in$ NM *typed in* ***bold****, as for example* ***nm****. The set of all such free object names is denoted by* FNM*.*

Definition 3 *Atomic and Grounded Atomic Sentences*
The atomic *sentences take the form* $Nm(tm_1, \ldots, tm_k)$*, where* $Nm{:}[\tau_1, \ldots, \tau_k]$*,* $0 \leq k$*. is a* predicate name*, possibly a* free *predicate name, and each* $tm_i{:}\tau_i$*. If* $Nm{:}[k]$*,* $0 \leq k$*, and is not a free predicate name, then the atomic sentence is said to be* grounded*. Each* tm_i *then may be an* object *name, but not ncessarily an object name.*

Grounded atomic sentences are to be understood as sentences whose truth or falsehood can be verified directly from observation, or indirectly in terms of predicate names that have been defined using only grounded predicate names. An atomic sentences that is not grounded and that does not have a free predicate name is to be understood as defined in terms of grounded atomic sentences. Free predicate and object names are used to define the proof theory of the logics.

2.2 Terms and Their Properties

Only terms and sentences, that is terms of type [0], have been mentioned so far. Now the next important ingrediant of the syntax, namely *abstraction variables*, must be introduced. An *abstraction* variable is a string of lowercase letters beginning with the letters u, v, w, x, y, z with or without numeral subscripts. Along with the λ-abstraction operator, the logical connectives $\wedge$, $\vee$, $\rightarrow$, and $\neg$, and the quantifiers $\exists$ and $\forall$ are introduced.

2.2.1 The λ Abstraction Operator

The λ operator of Church [5] has since that publication been extensively studied, for example in the already cited Barendregt monograh [2] that provides important definitions and results for the elementary syntax of the logics.

Definition 4 *Terms and their Types*
Consider a string of one or more characters each of which is a name from $\text{NM} \cup \text{FNM} \cup \text{NM} \cup \text{FNM}$*, a connective, a quantifier, =, an abstraction variable,* λ*, or one of the braces (and). Such a string is a* term *and is assigned one or more types under the following conditions:*

1. ***Names**: Each predicate name of* NM ∪ FNM *is a* term *assigned a single predicate type, its* primary *type. Each object name of* NM ∪ FNM *is a* term *assigned the object type 1.*
   There are denumerably many names of each type.
   ***Connectives**: Each of* $\wedge, \vee$ *or* $\rightarrow$ *is a* term *assigned the type* $[[0],[0]]$*, and* $\neg$ *is a* term *assigned the type* $[[0]]$*. These are the* primary *types of the connectives.*
   ***Quantifiers**: Each occurrence of* $\exists$ *or* $\forall$ *is a* term *assigned a single type* $[[\tau]]$*, the* primary *type of the occurrence.*
   ***Identity**: Each occurrence of* $=$ *is a* term *assigned a single type* $[\tau,\tau]$*, the* primary *type of the occurrence.*
   ***Variables**: Each occurrence of a variable is a* term *assigned a single type* τ*, the* primary *type of the occurrence.*

2. *Let* F *be assigned the primary type* $[\dots]$ *and let each and every of the types* $\tau_1,\dots,\tau_k$ *be assigned to one or more of all the free occurrences of a variable* v *in* F*. Then* $(\lambda v.F)$ *is assigned each of the types* $[\tau_i,\dots]$*,* $1 \leq i \leq k$*, each a* primary *type. If* v *has no free occurrence in* F*, then the primary type* $[\dots]$ *of* F *is the* primary *type of* $(\lambda v.F)$*.*

3. *Let the types* $[\tau_i,\dots]$*,* $1 \leq i \leq k$*, be all tha primary types assigned to a term* F*, and let each of* τ_i*,* $1 \leq i \leq k$*, be either a primary or secondary type of* G*. Then* (FG) *is a* term *of primary type* $[\dots]$*.*

4. *Let* F *be a term assigned a predicate type in which no* $Nm \in$ FNM *has an occurrence and in which only variables assigned type 1 have a free occurrence. Then* F *is also assigned the* secondary *type 1.*

F:τ *is the notation used throughout the paper to express that a specified occurrece of the term* F *is assigned the type* τ*.*

As noted in the introduction, item (4) of the definition provides the basis for all inductive definitions in the logics. The proviso that no member of FNM has an occurrence in F is essential for the consistency of the logics; to ignor this proviso would be to ignore the distinction between the use and mention of predicate names.

Examples may be useful in clarifying the effects of the last two clauses. Let A, B:[1] with $A, B \notin$ FNM and consider the lambda terms $(\lambda u.[A(u) \vee B(u)])$ and $(\lambda v.v(v))$, where in the latter the first occurrence of v:[1] and the second v:1. Then
a) $\exists(\lambda u.[A(u) \vee B(u)])$:[ ] and
b) $(\lambda v.v(v))(\lambda u.[A(u) \vee B(u)])$:[ ]
since $(\lambda u.[A(u) \vee B(u)])$:1 is a *secondary* type assignment, it is not needed for (a) but is for (b).

The notation $[G/v]F$ denotes the string of characters that results from replacing each free occurrence of a variable v in a term F by a a term G. It is assumed that all the complications of such substitutions involving free and

bound variables are understood by the reader; [2] can be consulted for details. The notation is only used in the following circumstances: Each free occurrence of the variable v in F has been assigned a type that has also been assigned to G. Note that the type of $[G/v]F$ is the type of F since each new occurrence of G in $[G/v]F$ takes the type of the free occurrence of v it replaces.

Changes of bound variables do not affect the meaning of terms. To ensure orderly changes of bound variables, variables are linearly ordered and tuples of variables are ordered accordingly.

Definition 5 *λ-reduction*
The λ-reduction relation $\rightsquigarrow$ between occurrences of terms is defined:

1. $((\lambda v.F)G) \rightsquigarrow [G/v]F$;
2. $(\lambda v.(Fv)) \rightsquigarrow F$ *and* $(\lambda v.F) \rightsquigarrow F$, *provided v has no free occurrence in F;*
3. $F \rightsquigarrow F'$, *provided F' is obtained from F by changes of bound variables that result in the tuple of bound variables in order of appearance in F' preceding the corresponding tuple of F.*
4. $(\lambda v.G) \rightsquigarrow (\lambda v.G')$, *provided $G \rightsquigarrow G'$;*
5. $(FG) \rightsquigarrow (F'G)$, *provided $F \rightsquigarrow F'$; and* $(FG) \rightsquigarrow (FG')$, *provided $G \rightsquigarrow G'$;*
6. *Let $F \rightsquigarrow G$ for terms F and G of primary type τ, and let them satisfy item 4 of definition 4 of Terms and their Types. Then also $F \rightsquigarrow G$ when they have the secondary type 1.*
7. $F \rightsquigarrow G$ *and* $G \rightsquigarrow H \Rightarrow F \rightsquigarrow H$.

The reductions in items (1), (2) and (3) are related to conversions sometimes called respectively as β, ξ and α.

Definition 6 *Reducible, Irreducible, and Grounded Terms*
A term F is reducible *if for some term F', $F \rightsquigarrow F'$; otherwise it is* irreducible.
A term F is grounded *in an irreducible term F_0 if $F \rightsquigarrow F_0$.*

The common terminology of normal form is not used to guard against misunderstandings.

That each term has a unique term in which it is grounded is assured by definition 5 since item (3) ensures an orderly change of bound variables. Justification for this conclusion can be found in Barendregt's monograph [2]. Although a term may not have more than one type in Barendregt's typed lambda calculus, this novelty does not change in any substantial way the results. cited there.

Lemma 1 *Unique Grounding*
For each term F there is an irreducible term F_0, necessarily unique, for which $F \rightsquigarrow F_0$.

2.2.2 Closed Terms

Closed terms, that is terms in which no variable has a free occurrence, are the only terms of immediate interest for the semantics and proof theories of CSL and HSL. For that reason for the remainder of the paper by a *term* is to be understood always to mean a *closed term*, unless a stement to the contrary appears. A term F that is not closed and contains free occurrences of a variable v is always prefixed with a substitution operator $[P/v]$ so that $[P/v]F$, that results from F by replaceing each free occurrence of v in F, is a closed term.

This notation is used, for example, in the statement of the $\pm\lambda$ rules of CSL and HSL; see §3.2.1. The rules are justified by definition 5.

2.3 Intensional Identity

From the *name* of a predicate the meaning, that is the *intension* of the predicate can be determined. To use an example of Frege, the name 'The morning star' is the name given to the planet Mars when seen in the morning and 'The evening star' when it is seen in the evening. The names of the two predicates are distinct although they have the same *extension*, namely the planet Mars; indeed that these distinct names denotes a single planet was a significant astronomical discovery. Conversely the several occurrences of a single predicate name in different contexts are all intended to denote the same predicate; indeed it is difficult to imagine how some facts might be communicated without this assumption. Unlike conventional predicate logics, CSL and HSL have the means for expressing identity between two predicate names through the use of the identity predicate $=$ of type $[1,1]$ as well as the intensional identity of two predicate names through the use of the identity predicate $=$ of type $[\tau,\tau]$. The relationship between these two identities is expressed in the $\pm Int$ rules of CSL and HSL; see §3.2.1.

2.4 Grounded Sentences

Since a *sentence* is a term of type $[0]$, it follows from lemma 1 that for every sentence M there is an irreducible sentence M_0 for which $M \rightsquigarrow M_0$.

Consider now an irreducible sentence $(\dots(PQ_1)\dots Q_k)$, $0 \leq k$ where $P \notin$ FNM, and P is not (P_1P_2) for some P_1 and P_2. Necessarily P cannot be a λ-term, and must therefore be one of the following:

1. $P \in$ NM of some predicate type $[\dots]$.
2. P one of $\wedge$, $\vee$ or $\rightarrow$ of type $[[0],[0]]$, or $\neg$ of type $[[0]]$.
3. P one of $\exists$ or $\forall$ of type $[[\tau]]$ for some τ.
4. P is $=$ of type $[\tau,\tau]$ for some τ.

Consider these case in turn:

1. The sentence is *atomic*; it is also *grounded* if $P{:}[k]$ is grounded. If it is not grounded, then the assumption has been made that P is definable in terms of grounded predicate names of type $[k]$, $0 \leq k$, in which case the sentence may become reducible when the definition of P replaces P.

2. In this case k is 2 or 1, and the truth or falsehood of the sentence requires considering the truth or falsehood of component sentences with fewer connectives.

3. In this case k is 1, and the truth or falsehood of the sentence depends upon the truth or falsehood of sentences $(Q_1 Tm)$, where $Tm : \tau$ and no free names occur within it.

4. In this case k is 2 and the sentence is $((= Q_1)Q_2)$, $Q_1, Q_2{:}\tau$. The sentence expresses the intensional identity of the two predicates as discussed in §2.3.

2.5 Conventional Notation

Throughout the paper a conventional logic notations will be substituted for the defined notations when greater clarityof structure is achieved. For example the defined notaion $(\exists F)$ would be written $\exists x.F(x)$ where if $F : [\tau]$ then x is an abstraction variable of type τ.

2.5.1 Functional Notation

The syntax for CSL and HSL provides a notation for functions with values that are predicate types. For example, consider functions with arguments of type $\sigma_1, \ldots, \sigma_k$ and a value of type $[\tau]$. Such a function has a type $[\sigma_1, \ldots, \sigma_k, \tau]$; for when argument of type $\sigma_1, \ldots, \sigma_k$ are applied to a predicate of type $[\sigma_1, \ldots, \sigma, \tau]$, a predicate of type $[\tau]$ results. But without the novel feature of the syntax of terms in item 4 of definition 4, which allows some higher order terms to have a secondary type 1, a notation would not be available for functions with values of type 1. An example that will be encountered again in §**??** is the zero and successor function for the natural numbers. The zero is defined to be $(\lambda u.\neg u = u)$ and successor $(\lambda u, v.u = v)$, where $u, v{:}1$. The primary types of these terms are $(\lambda u.\neg u = u) : [1]$, and $(\lambda u, v.u = v) : [1, 1]$, but since also $(\lambda u.\neg u = u){:}1$, it follows that $(\lambda u, v.u = v)(\lambda u.\neg u = u){:}[1]$ and that therefore the numeral one can be defined to be $(\lambda u, v.u = v)(\lambda u.\neg u = u){:}1$. This functional notation provides the means to give a recursive definition for the predicate Num of natural numbers.

3 CSL

The elementary syntax of CSL has been described in §2. The semantics of CSL is described in §3.1 and the proof theory, a proof of its consistency, and a sketch of its completenes is given in §3.2. Some definitions of terms relating to natural numbers and recursion theory are given in §3.3 along with a proof of why the logic of Andrews' paper [1] cannot substitute for CSL.

3.1 CSL Semantics

The values that may be assigned to the terms of CSL are members of the following domains, defined inductively on definition 1 of type.

Definition 7 *The Domains D_τ*
1. D_1 is the set of irreducible terms of type 1; $D_{[]}$ is the set $\{+,-\}$.
Recall that by the meaning given to 'term' in §2.2.2, terms are without free variables, and that by item 4 of definition 4, no member of FNM *occurs in a term of type 1.*
2. Assume D_σ and D_τ are defined. $D_{(\sigma,\tau)}$ is the set of all total functions with arguments from D_σ and single-values from D_τ.

Definition 8 *Valuations*
A valuation *is a function $\Phi(F,\tau)$ defined for arguments F and τ for which $F{:}\tau$ and with a single value in D_τ. A given valuation Φ satisfies the following conditions defined inductively on definition 4 of terms:*

1. ***Names:*** $F \in \mathrm{NM} \cup \mathrm{FNM} \Rightarrow \Phi(F,\tau) \in D_\tau$;
   $F \in \mathrm{NM} \cup \mathrm{FNM} \Rightarrow \Phi(F,1)$ *is* F.
   Connectives: $\Phi(\wedge,[[0],[0]])$, $\Phi(\vee,[[0],[0]])$ *and* $\Phi(\rightarrow,[[0],[0]])$ *are respectively:*
   $\{(+,+.+),(+,-,-),(-,+,-),(-,-,-)\}$;
   $\{(+,+.+),(+,-,+),(-,+,+),(-,-,-)\}$; *and*
   $\{(+,+.+),(+,-,-),(-,+,+),(-,-,+)\}$.
   $\Phi(\neg,[[0]])$ *is* $\{(-.+),(+,-)\}$.
   Quantifiers: $\Phi(\exists,[[\tau]])$ *and* $\Phi(\forall,[[\tau]])$ *are respectively*
   $\{(D',+)||D' \subseteq D_\tau, D'$ *not empty*$\} \cup \{(D',-)||D'$ *empty*$\}$; *and*
   $\{(D_\tau,+)\} \cup \{(D',-)||D' \subset D_\tau\ D'$ *not* $D_\tau\}$.
   Identity: $\Phi(=,[\tau,\tau])$ *is* $\{(P,Q,+)||\Phi(P,\tau)$ *is* $\Phi(Q,\tau)\} \cup$
   $\{(P,Q,-)||\Phi(P,\tau)$ *is not* $\Phi(Q,\tau)\}$.
   Variables: $\Phi(v,\tau) \in D_\tau$.

2. *Let $F:[\ldots]$, and let each and every of the free occurrences of v in F be assigned one of the types $\tau_1 \ldots, \tau_k$. Then for each $i, 1 \leq i \leq k$, $\Phi((\lambda v.F),[\tau_i,\ldots])$ is $(\Phi(v,\tau_i),\Phi(F,[\ldots])$.*
   Should $v{:}\tau$ have no free occurrence in F, then $\Phi((\lambda v.F),[\tau,\ldots])$ is $\Phi(F,[\ldots])$.

3. *Consider the case $(FG){:}[\ldots]$. There are two subcases to consider, when F is one of the terms of item 1 of definition 4, and when it is a λ-term of item 2. Consider these cases in turn.*

In the first case $F{:}[\tau,\dots]$ and $G{:}\tau$ for some type τ. It may be assumed that $\Phi(F,[\tau,\dots]) \in D_{[\tau,\dots]}$ and $\Phi(G,\tau) \in D_\tau$, where by definition 7, $D_{[\tau,\dots]}$ is the set of total single-valued functions with arguments from D_τ and values from $D_{[\dots]}$. Then $\Phi((FG),[\dots])$ is the $y \in D_{[\dots]}$ for which $(\Phi(G,\tau),y) \in D_{[\tau,\dots]}$.

Consider now the second case where F is $(\lambda v.H)$. Let the types $[\tau_i,\dots]$, $1 \le i \le k$, be all the primary types assigned to H, and let each of τ_i, $1 \le i \le k$, be either a primary or secondary type of G. Then $\Phi(((\lambda v.H)G),[\dots])$ is $\Phi(H,[\dots])$, where the value $\Phi(v,\tau_i)$ of a free occurrence of v in H of type τ_i is $\Phi(G,\tau_i)$.

4. *Let F be a term assigned a predicate type in which no $Nm \in$ FNM has an occurrence and in which $v_1,\dots,v_k{:}1$ are the only variables with a free occurrence in F. Then $\Phi(F,1)$ is the $F_0 \in D_1$ for which $[\Phi(v_1,1)/v_1]\dots[\Phi(v_k,1)/v_k]F \rightsquigarrow F_0$.*

3.1.1 Satisfaction and Validity of Sequents

A *sequent* $\Gamma \vdash \Theta$ consists of finite sets Γ and Θ of sentences of CSL and HSL of which at most one may be empty. Given a valuation Φ for CSL, the sequent is said to be *satisfied* by Φ if for some $M \in \Gamma$, $\Phi(M,[0])$ is $-$ or for some $N \in \Gamma$, $\Phi(N,[0])$ is $+$. The sequent is said to be *valid* if it is satisfied by every valuation. A valuation Φ is a *counter-example* for the sequent if for all $M \in \Gamma$, $\Phi(M,[0])$ is $+$ and for all $N \in \Gamma$, $\Phi(N,[0])$ is $-$. The proof theories for both CSL and HSL are designed to seek a countr-example for a given sequent; if the search for a countr-example can be sown to lead to a contradiction, then the sequent is *derivable* in the logic. This style of proof theory was first described by Beth in [3] and [4] for first order classical and intuitionist logics and recently more extensively for the classical logic by Hodges in [11].

3.2 CSL Proof Theory

The rules of deduction for both CSL and HSL are described next. The necessary restrictions on the rules for HSL are described in definition 14 in §4.2, and they prevent any of these rules being defined in terms of others, unlike CSL. In these rules each of M,N,P,Q is a closed term with $M,N{:}[0]$. $F{:}[0]$ is a term in which only a single variable has a free occurrence.

3.2.1 Rules of Deduction

$$+\wedge \quad \frac{+[M \wedge N]}{+M} \qquad \frac{+[M \wedge N]}{+N} \qquad -\wedge \quad \frac{-[M \wedge N]}{-M \quad -N}$$

$$+\vee \quad \frac{+[M \vee N]}{+M \quad +N} \qquad -\vee \quad \frac{-[M \vee N]}{-M} \qquad \frac{-[M \vee N]}{-N}$$

$$+\rightarrow \quad \frac{+[M \rightarrow N]}{-M \quad +N} \qquad -\rightarrow \quad \frac{-[M \rightarrow N]}{+M} \quad \frac{-[M \rightarrow N]}{-N}$$

$$+\neg \quad \frac{+\neg M}{-M} \qquad -\neg \quad \frac{-\neg M}{+M}$$

$$+\exists \quad \frac{+\exists P}{+P(\mathbf{nm})} \qquad -\exists \quad \frac{-\exists P}{-P(Q)}$$

new $\mathbf{nm} \in \mathrm{FNM} \cup \mathrm{NM}$

$$+\forall \quad \frac{+\forall P}{+P(Q)} \qquad -\forall \quad \frac{-\forall P}{-P(\mathbf{nm})}$$

new $\mathbf{nm} \in \mathrm{FNM} \cup \mathrm{NM}$

$$+\lambda \quad \frac{+[P/v]F}{+[Q/v]F} \qquad -\lambda \quad \frac{-[P/v]F}{-[Q/v]F}$$

$$P \rightsquigarrow Q$$

$$+= \quad \frac{+P = Q}{+\forall x.[x(P) \rightarrow x(Q)]} \qquad \frac{+P = Q}{+\forall x.[x(Q) \rightarrow x(P)]}$$

$$-= \quad \frac{-P = Q}{-\forall x.[x(P) \rightarrow x(Q)] \qquad -\forall x.[x(Q) \rightarrow x(P)]}$$

$$+\mathrm{Int} \quad \frac{+P =^1 Q}{+P =^\tau Q} \qquad -\mathrm{Int} \quad \frac{-P =^1 Q}{-P =^\tau Q}$$

$$\mathrm{Cut} \quad \frac{}{+M \quad -M}$$

Note that the $\pm Int$ rules are the only ones to change a type from premiss to conclusion.

3.2.2 CSL Derivations

The rules of deduction of CSL can be used to systematicaly search for a *counter-example* for a given sequent. Here the systematic search is described as the

construction of trees with nodes that are signed sentences of CSL. All the rules of deduction other than Cut may be used.

Definition 9 *Search Tree for a Sequent*

1. *A search tree for a sequent* $\Gamma \vdash \Theta$ *is a set of* branches*, each of which is a sequence of signed sentences the initial members of which are* $+M$ *for each* $M \in \Gamma$ *and* $-N$ *for each* $N \in \Theta$*.*

2. *Each member following the initial members of a branch is a single conclusion of one of the rules of deduction with premiss a previous member of the branch.*

3. *A premiss of the rules* $+\wedge, -\vee, -\rightarrow$ *and* $+=$*, each with two possible conclusions, may be used twice in a given branch as a premiss, once for each conclusion.*

4. *An application of the rules* $-\wedge, +\vee, +\rightarrow$ *and* $-=$*, each with alternative conclusions, results in a splitting of the branch of the premiss of the rule into two branches with identical members preceding the conclusions.*

5. *The rules* $+\exists$ *and* $-\forall$ *may be used only once with the same premiss and add* $+P(\boldsymbol{a})$*, respectively* $-P(\boldsymbol{a})$*, to the branch; here* $\boldsymbol{a} \in \mathrm{FNM} \cup \mathrm{FNM}$*, with type determined by the quantifier, has no previous occurrece in the branch.*

6. *The rules* $-\exists$ *and* $+\forall$ *may be used any number of times with the same premiss but only once with the same conclusion* $-P(Q)$*, respectively* $+P(Q)$*; here* $\exists, \forall{:}[[\tau]]$*,* $P{:}[\tau]$*, and* $Q{:}\tau$*.*

7. *A branch is* closed *if both* $+M$ *and* $-M$ *are members for some sentence* M*; otherwise it is said to be* open*. A search tree is* closed *if each of its branches is closed; otherwise it is said to be* open*.*

Because being a sentence of CSL, or HSL, is not decidable, item (7) in allowing a branch to be closed by signed sentences $+M$ and $-M$ assumes that the sentencehood of M has been confirmed. An alternative would be to require M to be *atomic*, since being an atomic sentence is decidable and it is easily established that the sequent $M \vdash M$ is CSL derivable for every sentence M of CSL.

A sequent is *derivable* if a search tree constructed from it is *closed*. A proof of the following lemma is sketched with final details left to the reader.

Lemma 2 *Every Derivable Sequent is Valid*

proof
Let $\Gamma \vdash \Theta$ be a derivable sequent and Φ a valuation. It is necessary to prove that every such sequent is satisfied by every valuation. A proof by induction on the number n of rules applied in the derivation of the sequent will be sketched.

Let n be 0 so that Γ and Θ have identical membership; the sequent is necessarily satisfied by any Φ.

Assume that all sequents derivable with n or fewer applications is valid. Cnsider a sequent with a derivation with $n+1$ applications and consider the first rule applied in its derivation. Proofs for only two of the possible first rules applied will be given since the proofs for the other cases take a similar form.

Let the first rule applied be the $+Int$ rule. Then it may be assumed that the sequent takes the form $\Gamma, P =^1 Q \vdash \Theta$ where $P, Q{:}1$ since $P, Q{:}\tau$ and no member of FNM occurs in either P or Q. It follows that the sequent $\Gamma, P =^\tau Q \vdash \Theta$ has a derivation with n applications and is satisfied by every Φ. If $\Gamma, P =^1 Q \vdash \Theta$ is not also valid then there exists a Φ that does not satisfy it. But this is only possible if $\Phi(P =^1 Q, [0])$ is $+$ while $\Phi(P =^\tau Q, [0])$ is $-$.

From the **Identity** case of item 1 of the definition 8 of valuations it follows that $\Phi(P =^1 Q, [0])$ is $+$ if and only if $\Phi(P, 1)$ is $\Phi(Q, 1)$; that is, for some irreducible $R_0{:}1$, $P \rightsquigarrow R_0$ and $Q \rightsquigarrow R_0$. Hence by item 6 of definition 5 of $\rightsquigarrow$, $P \rightsquigarrow R_0$ and $Q \rightsquigarrow R_0$ when $P, Q, R_0{:}\tau$, and therefore again by item 1 of definition 8 it follows that $\Phi(P =^\tau Q, [0])$ is $+$, contrdicting the previous value.

Consider one more case, when the first rule used in the derivation of the sequent is $-\forall$. Then the sequent takes the form $\Gamma \vdash \Theta, \forall P$, where $P{:}[\tau]$, and the sequent $\Gamma \vdash \Theta, P(\mathbf{nm})$ is derivable and valid, where $\mathbf{nm} \in$ FNM does not occur in any sentence of $\Gamma \vdash \Theta, \forall P$. It follows that each sequent $\Gamma \vdash \Theta, P(Q)$ for which $Q \in D_\tau$ is also derivable and valid.

Assume that $\Gamma \vdash \Theta, \forall P$ is not valid. It follows that for some Φ, $\Phi(\forall P, [0])$ is $-$ while $\Phi(P(Q), [0])$ is $+$ for every $Q \in D_\tau$. But that contradicts the **Quantifiers** case of item 1 of the definition 8 of valuations.
qed

3.2.3 CSL Completeness

The converse of lemma 2 is *completeness*. Clearly completeness with respect to the domains of definition 7 cannot be proved since there are only denumerably many derivable sequents while the domains D_1 and $D_{[]}$ are finite and all others are non-denumerable; the incompleness theorem of Gödel [21] describes a method of constructing such a sequent from any fully formal theory of natural numbers. Completeness has to be proved with repect to some proper subdomains of the defined domains. Henkin described how such domains could be defined for higher orders predicate logic in [10]. These results can be adapted for CSL. But now a proof of completeness for a logic like CSL proceeds by a method described by Prawitz in [14] for second order logic. An open *saturated* branch of a search tree for a given underivable sequent is first constructed. Here by a saturated branch is meant one in which every possible conclusion of a rule with premiss on the branch appears on the branch. A *partial* valuation that is a counter-example for the sequent is defined from the branch. Then a transfinite process completes the valuation while maintaining it as a counter-example for the given underivable sequent. The domains of this constructed valuation are generally proper subsets of the domains of definition 7. Thus a completeness proof obtained in this way is esentially circular: a logic such as CSL or HSL is compete with respect to the predicates that can be defined within the logic by a

transfinite process. Since a proof of completeness via a proof of the redundancy of Cut for CSL, and later HSL, can be adapted from the proof for ITT given in [9], the proofs are left to the reader.

3.3 Some Derivable Sequents

Here a number of derivable sequents will be stated to illustrat the strength of CSL; in §4 these sequents will also be seen to be derivable in HSL. nnot substitute for CSL. As noted in §2.5 a standard notation is generally employed.

Definition 10 *Some Abbreviating Definitions.*
$0 \stackrel{df}{=} (\lambda u.\neg u = u)$. *The empty predicate of arity 1, nuweral 0; type* $[1]$.
$S \stackrel{df}{=} (\lambda u, v.u = v)$. *The successor predicate; type* $[1,1]$.
$RN \stackrel{df}{=} (\lambda z, x.[x = 0 \vee \exists u.[z(u) \wedge x = S(u)]])$.*Recursion generator for the sequence of natural numbers; type* $[[1],1]$.
$Lt \stackrel{df}{=} (\lambda wg, u.\forall z.[\forall x.[wg(z,x) \rightarrow z(x)] \rightarrow z(u)]])$. *Least Predicate Operator; type* $[[[1],1],1]$.
$Num \stackrel{df}{=} (\lambda u.Lt(RN,u))$. *Natural Number; type* $[1]$.
$Gt \stackrel{df}{=} (\lambda wg, u.\exists z.[\forall x.[z(x) \rightarrow wg(z,x)] \wedge z(u)]])$. *Greatest Predicate Operator; type* $[[[1],1],1]$.
$Un \stackrel{df}{=} (\lambda u.u = u)$. *The universal predicate of arity 1; type* $[1]$.
$Rus \stackrel{df}{=} (\lambda w.\neg w(w))$. *The Russell set; type* $[[1]]$ *and* $[1]$.
$\mathrm{V} \stackrel{df}{=} (\lambda w, v.w(v(v)))$. *Andrews'* V.
$\mathrm{Y} \stackrel{df}{=} (\lambda w.(\mathrm{V}(w)\mathrm{V}(w))$. *Andrews'* Y *combinator.*
$\mathrm{N} \stackrel{df}{=} (\lambda u, x.[x = 0 \vee \exists y.[x = S(y) \wedge u(y)])$. *Andrews'* N.

Examples of derivable sequents of CSL that make use of these definitions are:
1. $\vdash \forall x, y.[S(x) = S(y) \rightarrow x = y]$
2. $\vdash \forall x.\neg S(x) = 0$
3. $\vdash Num(0)$
4. $\vdash \forall x.[Num(x) \rightarrow Num(S(x))]$
5. $\vdash \forall w.[w(0) \wedge \forall x.[w(x) \rightarrow w(S(x))] \rightarrow \forall x.[Num(x) \rightarrow w(x)]]$
6. $\vdash Rus(Emp)$
7. $\vdash \neg Rus(Un)$

The sequents (1-5) are the Peano's axioms for the natural numbers, while (6) and (7) illustrate how the unusual syntax of CSL and HSL can accomodate sentences involving paradoxical predicates like Russell's set. Derivations of these sequents that can be adapted for CSL are given in [9]. However to illustrate one important role for the unusual rules $\pm Int$, a derivation of (1) is given with the types of occurrences of $=$ made explicit.

$-\forall x, y.[S(x) =^1 S(y) \rightarrow x =^1 y]$
$-S(\mathbf{a}) =^1 S(\mathbf{b}) \rightarrow \mathbf{a} =^1 \mathbf{b}$; $\mathbf{a}, \mathbf{b} \in$ FNM
$+S(\mathbf{a}) =^1 S(\mathbf{b})$
$-\mathbf{a} =^1 \mathbf{b}$
$+S(\mathbf{a}) =^{[1]} S(\mathbf{b})$; $+Int$
$+\forall w.[w(S(\mathbf{b})) \rightarrow w(S(\mathbf{a}))]$; $+=$
$+[(\lambda z.(z\mathbf{b}))(S(\mathbf{b})) \rightarrow (\lambda z.(z\mathbf{b}))(S(\mathbf{a}))]$; $[(\lambda z.(z\mathbf{b}))/w]$
$+[((S(\mathbf{b}))\mathbf{b}) \rightarrow (\lambda z.(z\mathbf{b}))(S(\mathbf{a}))]$; $+\lambda$
$+[((S(\mathbf{b}))\mathbf{b}) \rightarrow ((S(\mathbf{a})\mathbf{b}))]$; $+\lambda$
$+[(((\lambda u, v.u = v)(\mathbf{b}))\mathbf{b}) \rightarrow (((\lambda u, v.u = v)(\mathbf{a})\mathbf{b}))]$; def S
$+[\mathbf{b} =^1 \mathbf{b} \rightarrow \mathbf{a} =^1 \mathbf{b}]$; $+\lambda$

$-\mathbf{b} =^1 \mathbf{b}$ $\qquad$ $+\mathbf{a} =^1 \mathbf{b}$
$-\forall w.[w(\mathbf{b}) \rightarrow w(\mathbf{b})]$ $\qquad$ ---
$-[\mathbf{C}(\mathbf{b}) \rightarrow \mathbf{C}(\mathbf{b})]$; $\mathbf{C} \in$ FNM
$+\mathbf{C}(\mathbf{b})$
$-\mathbf{C}(\mathbf{b})$

In §4.2 of [1], Andrews suggests that Y(N) can play the role in his logic that Num plays in CSL, namely as a predicate name for the natural numbers, or more precisely, a *counter set* for the natural numbers to use the terminology of [15]. Why Y(N) cannot take on that role in CSL is demonstrated by any attempt to provide a derivation for the following sequent in which $\mathbf{a} \in$ FNM:
A) $\mathrm{Y(N)}(\mathbf{a}) \vdash \mathrm{Y(N)}(\mathbf{a})$
Using the $\beta-$reduction of Y(N) given in [1], to provide a derivation for (A) it is sufficient to provide a derivation for
B) $(\lambda x.[x = 0 \vee \exists y.[x = S(y) \wedge \mathrm{Y(N)}(y)])(\mathbf{a}) \vdash$
$\qquad\qquad (\lambda x.[x = 0 \vee \exists y.[x = S(y) \wedge \mathrm{Y(N)}(y)])(\mathbf{a})$
A derivation of (B) would have to include all of the following:
$+(\lambda x.[x = 0 \vee \exists y.[x = S(y) \wedge \mathrm{Y(N)}(y)]])(\mathbf{a})$
$-(\lambda x.[x = 0 \vee \exists y.[x = S(y) \wedge \mathrm{Y(N)}(y)]])(\mathbf{a})$
$+[a = 0 \vee \exists y.[a = S(y) \wedge \mathrm{Y(N)}(y)]]$
$-[a = 0 \vee \exists y.[a = S(y) \wedge \mathrm{Y(N)}(y)]]$
$+\exists y.[a = S(y) \wedge \mathrm{Y(N)}(y)]$
$-\exists y.[a = S(y) \wedge \mathrm{Y(N)}(y)]$
$+[a = S(\mathbf{b}) \wedge \mathrm{Y(N)}(\mathbf{b})]$,$b \in$ NM
$-[a = S(\mathbf{b}) \wedge \mathrm{Y(N)}(\mathbf{b})]$
$+\mathrm{Y(N)}(\mathbf{b})$
$-\mathrm{Y(N)}(\mathbf{b})$

Thus a derivation of (A) requires a derivation of
C) $\mathrm{Y(N)}(\mathbf{b}) \vdash \mathrm{Y(N)}(\mathbf{b})$
Any attempted derivation of (A) cycles, just as does any attempted derivation of $Rus(Rus) \vdash Rus(Rus)$!! The string $\mathrm{Y(N}(\mathbf{a})$ is an example of what can be called *ungrounded* abstractions and are properly excluded from the terms of

CSL.

It is possible that Y like *Russ* is definable in CSL and that like 0 and Un for *Russ* one or more terms P can be defined for which $(\mathrm{Y}P) \vdash (\mathrm{Y}P)$ is a derivable sequent.

4 HSL

The primary purpose of this section is to adapt the Kripke semantics for first order intuitionist logic [13] for the higher order intuitionist logic HSL with the same elementary syntax as CSL as given in §2. A secondary purpose is then to explore how this semantics might provide insights into the foundations of experiment based scientific theories in general and quantum mechanics in particular as suggested by Lee Smolin in [19]. This is of course highly speculative. A sounder purpose is to provide a foundation for higher order intuitionist mathematics.

4.1 Kripke Semantics

For CSL a valuation Φ assigned either + 'true' or − 'false' to each sentence. For HSL a Kripke model does the same but not in the same explicit manner, since in Kripke models the fact that no explicit sign is attached to a sentence is to be the equivalent of assigning it a − sign. Assigning a + sign to a sentence in Kripke semantics is to be understood as asserting that the truth of the sentence has been *verified*, not just that the sentence is assumed to be 'true'. Since assigning a − sign to a sentence M is to be understood as asserting that M has not been verified, a wide range of meanings can follow from $-M$. It could mean that M has been verified to be false, in which case $+\neg M$ can be asserted, but it could also mean that currently nothing is known about the truth or falsehood of M.

A Kripke style semantics is described here for HSL in a form much influenced by §5 of volume I of Troelstra and van Dalen's [20]. The concept of a *world* in Kripke semantics is identified here with a recording of the observations made in the world in the form of *grounded* atomic sentences. Observations made in one world may differ from those made in a different world. In a successor world to a given world, additional obervations may be recorded, but previously recorded observtions are always confirmed. The additional observations may, of course, involve objects not observed in a previous world; the domain of observables, which is always a subset of the type 1 objects, may enlarge in a successor world. The grounded atomic sentences may of course reference the observer who has recorder the observation - observers may be objects in a world.

Definition 11 *Kripke Models of HSL*
A Kripke model $\mathcal{KM}$ for HSL is a quadruple $(\mathrm{W}, \sqsubseteq, \mathcal{B}, \textsc{Tup})$*, where:*

1. W *is a non-empty set of* worlds *w partially ordered by the relation $\sqsubseteq$ with* least *element the world w_0.*

2. $\mathcal{B}$, *the* basis *of the model, is a set of sets* $\mathcal{B}^k$ *consisting of a set of subsets* $\textsc{Tup}^k$ *of* D_1^k, $0 \leq k$, *that includes the empty subset* $\{\}$; $\mathcal{B}^0$ *is therefore* $\{\{\}, ()\}$, *where* $()$ *is the empty tuple.*

3. $\textsc{Tup}$ *is a function with arguments* $w \in \mathrm{W}$ *and* $Nm \in \mathrm{NM}^k, 0 \leq k$, $Nm \notin \mathrm{FNM}^k$, *with values* $\textsc{Tup}(w, Nm) \in \mathcal{B}^k$ *satisfying:*
$w_1 \sqsubseteq w_2 \Rightarrow \textsc{Tup}(w_1, Nm) \subseteq \textsc{Tup}(w_2, Nm)$.

The *grounded* atomic sentences $Nm(tm_1, \ldots, tm_k)$, $(tm_1, \ldots, tm_k) \in \mathrm{TUP}(w, Nm))$, $w \in \mathrm{W}$, records all observations currently made and expected to be made in w_0 and all its successors w. Note that the definition of the sets of subsets $\mathcal{B}^k$ in clause (2) together with its use in clause (3) allows for new objects of type 1 to be observed in a successor world. The logical consequences of these observations is defined next in terms of *signed* sentences.

Definition 12 *The Forcing Relation* $\vDash$.
The forcing relation $\mathcal{KM} \vDash \pm M$ *between a model* $\mathcal{KM}$ *specified by* $(\mathrm{W}, \sqsubseteq, \mathcal{B}, \textsc{Tup})$, *and signed sentences* $\pm M$, *holds if* $w_0 \vDash \pm M$, *where* w_0 *is the least world of* W. *The relation* $\vDash$ *for* $\mathcal{KM}$ *is defined as the least relation between worlds* $w \in \mathrm{W}$ *and signed sentences satisfying the conditions stated next in three groups, Current world, Some successor world, and All successor worlds. Each world of the model must be considered in turn as the Current world and must satisfy the specified conditions, and must further have successor worlds with forced signed sentences as required by Some successor world, and All successor worlds.*

Current world w**.**

$\pm A$**:** *Let* A *be a* grounded *atomic sentence* $Nm(tm_1, \ldots, tm_k)$. *Then:*
$(tm_1, \ldots, tm_k) \in \mathrm{TUP}(w, Nm) \Rightarrow w \vDash +A$;
$(tm_1, \ldots, tm_k) \notin \mathrm{TUP}(w, Nm) \Rightarrow w \vDash -A$.

$\pm\wedge$**:** $w \vDash \pm M$ *and, resp. or,* $w \vDash \pm N \Rightarrow w \vDash \pm[M \wedge N]$.

$\pm\vee$**:** $w \vDash \pm M$ *or, resp. and,* $w \vDash \pm N \Rightarrow w \vDash \pm[M \vee N]$.

$\pm\exists$**:** $P{:}[\tau]$.
$w \vDash \pm P(Q)$ *for some, resp. for no* $Q{:}\tau \Rightarrow w \vDash \pm\exists P$.

$\pm =$**:** *For some* R, *resp. for no* R, $P \rightsquigarrow R$ *and* $Q \rightsquigarrow R \Rightarrow \pm P = Q$.

$\pm\lambda$**:** $P \rightsquigarrow Q$ *and* $w \vDash \pm[P/v]F \Rightarrow w \vDash \pm[Q/v]F$.

Some successor world. *For some* $w', w \sqsubset w'$*:*

$-\neg$**:** $w' \vDash +M \Rightarrow w \vDash -\neg M$.

$-\,\rightarrow$**:** $w' \vDash +M, w' \vDash -N \Rightarrow w \vDash -[M \rightarrow N]$.

$-\forall$**:** $P{:}[\tau]$.
$w' \vDash -P(Q)$ *for some* $Q{:}\tau \Rightarrow w \vDash -\forall P$.

All successor worlds. *For all* $w', w \sqsubseteq w'$*:*

$+\rightarrow$**:** $w' \vDash -M$ *or* $w' \vDash +N \Rightarrow w \vDash +[M \rightarrow N]$.

$+\neg$**:** $w' \vDash -M \Rightarrow w \vDash +\neg M$.

$+\forall$**:** $P{:}[\tau]$.
$w' \vDash +P(Q)$ *for all* $Q{:}\tau \Rightarrow w \vDash +\forall P$.

4.2 HSL Proof Theory

It is important to recall that an atomic sentence A recorded as $-A$ in a world may in a later world be changed to a recording of $+A$; this is a consequence of item 3 of definition 11. It follows from definition 12 that the same holds for any sentence M; it is only the sentences recorded as $+M$ that are never changed. It is this fact that permits the same rules of deduction described for CSL in §3.2 to have different conclusions for HSL.

The proof theory of HSL, like that of CSL, is motivated by a search for a counter-example.

Definition 13 *Sequents, their Satisfactiom and Counter-Examples*
Let $\Gamma \vdash \Theta$ *be a sequent and let* $\mathcal{KM}$ *be a model.* $\mathcal{KM}$ satisfies $\Gamma \vdash \Theta$ *if for some* $M \in \Gamma, \mathcal{KM} \vDash -M$*, or for some* $N \in \Theta, \mathcal{KM} \vDash +N$.
$\mathcal{KM}$ *is a* counter-example *for* $\Gamma \vdash \Theta$ *if for each* $M \in \Gamma, \mathcal{KM} \vDash +M$*, and each* $N \in \Theta, \mathcal{KM} \vDash -N$.

For a given sequent there may be both a model that satisfies it as well as a model that is a counter-example. But these are not the sequents of interest, rather it is the sequents without a counter-example. A process will be described for determining of a given sequent whether it has a counter-example. The process makes use of the *rules of deduction* of the logic CSL described in §3.2.1 that, with special restrictions, are the rules of deduction for HSL. They can be seen to be a more formal statement of the Kripke semantics described in §4.1.

4.2.1 Examples

Some examples may be useful in connecting the semantic rules with derivations and counter-examples. The first example compares a derivation of the sequent $\vdash \neg\neg[A \vee \neg A]$ in CSL with a derivation in HSL.

CSL	HSL
$-\neg\neg[A \vee \neg A]$	$-\neg\neg[A \vee \neg A]$
$+\neg[A \vee \neg A]$	$+\neg[A \vee \neg A]$
$-[A \vee \neg A]$	$-[A \vee \neg A]$
$-A$	$-\neg A$
$-\neg A$	$+A$
$+A$	$-[A \vee \neg A]$
	$-A$

Note that it is necessary to repeat the appearance of $-[A \vee \neg A]$ in the HSL derivation.

The next example, $\forall x.[P(x) \vee M] \vdash [\forall x.P(x) \vee M]$, taken from [13], illustrates how a search for a counter-example can lead to the construction of a counter-example. Two different ways of constructing the same counter-example are described in two different ways of dealing with a restriction on the use of the $-\forall$ rule described later in §4.2.2.

$+\forall x.[P(x) \vee M]$			$+\forall x.[P(x) \vee M]$				
$-[\forall x.P(x) \vee M]$			$-[\forall x.P(x) \vee M]$				
$+[P(a) \vee M]$			$-\forall x.P(x)$	$\cdot$	$\cdot$	$\cdot$	$-M$
			$-P(\mathbf{b})$				$+[P(a) \vee M]$
$+P(a)$	$+M$		$+[P(\mathbf{b}) \vee M]$				
$-\forall x.P(x)$	$-M$						$+P(a)$ $+M$
$-P(\mathbf{b})$			$+P(\mathbf{b})$	$+M$			
$+[P(\mathbf{b}) \vee M]$							
$+P(\mathbf{b})$	$+M$						

There is a single open branch with atomic members $+P(a)$, $-P(\mathbf{b})$, $+M$ in the tree on the left. In the pair of trees on the right there is an open branch in each, with atomic members $-P(\mathbf{b})$, $+M$ in the first open branch and $-M$, $P(a)$ in the second. Since the absence of a sentence is to be understood in the same way as its presence with a $-$ sign, these two structures define the same counter-example.

The counter-example $\mathcal{KM}$ defined for this example has two worlds w_0 with members $\{-M, +P(a)\}$, and w_1 with members $\{+P(a), -P(\mathbf{b}), +M\}$. The domain of w_0 is $\{a\}$ and that of w_1 is $\{a, \mathbf{b}\}$. The justification for concluding that $\mathcal{KM}$ provides a counter-example is given next.

$w_0 \vDash +P(a)$	$w_1 \vDash -P(\mathbf{b})$
$w_0 \vDash +[P(a) \vee M]$	$w_0 \vDash -\forall x.P(x)$
$w_1 \vDash +M$	$w_0 \vDash -M$
$w_1 \vDash +[P(\mathbf{b}) \vee M]$	$w_0 \vDash -[\forall x.P(x) \vee M]$

$w_0 \vDash +\forall x.[P(x) \vee M]$

4.2.2 Search Trees for HSL

The definition of a search tree for a given sequent for HSL differs from the definition for CSL first in the most obvious way of requiring the full range of rules for the connectives and quantifiers, More important differences are in restrictions on the applications of the rules $-\neg$, $- \rightarrow$ and $-\forall$, arising from the fact that these rules apply in a possible successor world in which a $-M$ in a previous world could become $+M$. The final difference is in a subtle change in the definition of closed branch. These differences are made explicit in the next definition.

Definition 14 *Search Tree for a Sequent*

1. *A search tree for a sequent* $\Gamma \vdash \Theta$ *is a set of* branches*, each of which is a sequence of signed sentences the initial members of which are* $+M$ *for each* $M \in \Gamma$ *and* $-N$ *for each* $N \in \Theta$*.*

2. *Each member following the initial members of a branch is a single conclusion of one of the rules of deduction with premiss a previous member of the branch that has remained* current *on the branch. A signed sentence* $+M$ always *remains current on a branch. A signed sentence* $-M$ *is current on a branch provided no conclusion of one of the rules* $-\neg$, $- \rightarrow$ *and* $-\forall$ *follows it on the branch. For example, consider*
   $-[\forall x, P(x) \vee M]$
   $-M$
   $-\forall x.P(x)$
   $-P(\boldsymbol{b})$
   The only member of this branch that is current is the last, because it is a conclusion of the $-\forall$ *rule;* $-M$ *is no longer current since* $+M$ *may hold in a successor to the world of* $-M$*. That is the reason for the treatment of* $-[\forall x, P(x) \vee M]$ *in the example given in §4.2.1.*

3. *A current premiss of one of the rules* $-\vee$, $+\wedge$, $- \rightarrow$ *and* $+ =$ *has two possible conclusions; either one or both of which may be added to the branch provided that the premiss remains current for the second coclusion. For the* $-\vee$ *rule an alternative is to duplicate the existing tree and add one of the two conclusions to each tree. An example is given in §4.2.1.*

4. *A current premiss of one of the rules* $+\vee$, $-\wedge$, $+ \rightarrow$ *and* $- =$ *has two alternative conclusions. An application of one of these rules results in a splitting of the branch into two branches with identical members preceeding the conclusions.*

5. *A current premiss of one of the rules* $+\exists$ *and* $-\forall$ *has a single conclusion* $+P(\boldsymbol{nm})$*, respectively* $-P(\boldsymbol{nm})$*, on the branch; here* $\boldsymbol{nm} \in \mathrm{FNM} \cup \mathrm{FNM}$

has no previous occurrece in the branch, and has its type determined by P.

6. *A current premiss of one of the rules* $-\exists$ *and* $+\forall$ *may have any number of conclusions. Each conclusion has the form* $-P(tm)$, *respectively* $+P(tm)$, *where* tm *has its type determined by* P.

7. *A branch is* closed *if* $+M$ *and a* current $-M$ *are both members for some sentence* M; *otherwise it is said to be* open. *A search tree is* closed *if each of its branches is closed; otherwise it is said to be* open.

Because of the alternative for $-\vee$ in item (3), more than one search tree may be constructed for a given sequent. However only a single closed tree need be found to establish that the sequent is *derivable* in HSL, although an open branch from each of the open search trees is needed to construct a Kripke model that is a counter-example for it.

The comment concerning item (7) of definition 9 for CSL search trees is relevant here: Being a sentence of HSL, or CSL, is not decidable; allowing a branch to be closed by $+M$ and a current $-M$ assumes that the sentencehood of M has been confirmed. An alternative would be to require M to be *atomic*, since being an atomic sentence is decidable and it is easily established that the sequent $M \vdash M$ is HSL derivable for every sentence M of HSL.

Definition 15 *HSL Derivations*
An HSL derivation for a sequent is a closed HSL search tree for the sequent.

Since every derivable sequent of HSL is a derivable sequent of CSL, lemma 2 ensures the consistency of HSL as well as of CSL.

4.2.3 Some HSL Derivations

The first five examples given in §3.3 of sequents derivable in CSL are repeated here:
1. $\vdash \forall x, y.[S(x) = S(y) \rightarrow x = y]$
2. $\vdash \forall x.\neg S(x) = 0$
3. $\vdash Num(0)$
4. $\vdash \forall x.[Num(x) \rightarrow Num(S(x))]$
5. $\vdash \forall w.[w(0) \wedge \forall x.[w(x) \rightarrow w(S(x))] \rightarrow \forall x.[Num(x) \rightarrow w(x)]]$

They are Peano's axioms for the natural numbers. A CSL derivation was given for the first of these sequents; clearly that derivation is also an HSL derivation. Indeed a foundation for all of the recursion theory described in chapter 4 Recursions of [9] can be provided in HSL.

4.2.4 Completeness of HSL

It was remarked in §3.2.3 that there is a circularity implicit in the proof of completeness of CSL, namely in as much as the logic is complete with respect to the predicates that can be defined within the logic by a transfinite process.

Nevertheless, the fact that the redundancy of the Cut rule is a conseqiuence makes the result signifigant since searching for derivations of sequents in CSL is greatly simplified when Cut can be ignored. A similar completeness can be proved for HSL by a similar process for HSL. Again the fact that Cut is proved redundant is important. But for HSL there may be an additional benefit in describing the predicates definable within HSL since it may help clarify what is to be meant by intuitionistically defined predicates. For example, comparisons with the constructive set theory of chapter 11 of [20] would be of great interest.

5 Logic, Mathematics and the Physical Sciences

Nominalism holds that an assertion purportdly about a universal is really an assertion about how the name of the univeral is *used*. For example, the sentence 'yellow is a colour' is to be understood as asserting 'Yellow is a colour word'. The subject of the sentence is thus the word 'yellow' and not the colour yellow; the sentence asserts a fact about the usage of the word in the English language. When someone asserts 'my car is yellow' we learn something about the appearance of his car. The use of number words is similarly interpreted. 'John has two legs' gives us information about a person by *using* the words 'John' and 'two' while 'John has four letters' tells us something about the word 'John' by *using* the word 'four' and *mentioning* the word 'John'.

As Sellars points out in [18], a careful distinction between use and mention disarms Cantor's paradox. The consequences are of signifigance since a recognition of the distinction requires dropping the fundamental axiom of extensionality of Cantor's theory and replacing it with something like the rules $\pm Int$ of CSL and HSL. But really only of significance for such logics in which something like the λ-abstraction operator is used and mathematics depending upon it; the rest of mathematics is unaffected.

However, that is not to say that awareness of a nominalist view of language and its dependence upon the distinction between use and mention is not important for mathematics and science; afterall computers are consumate nominalists and they are of increasing importane for the manipulations and calculations of modern physicak theories. But of course it remains to be seen whether a nominalist motivated logic like HSL based on a semantics derived from Kripke's semantics for first order logic can provide any insights into foundations for quantum cosmology.

References

[1] James H. Andrews, An Untyped Higher Order Logic ith Y Operator, Journal of Symbolic Logic, (2007) 21 pages.

[2] H.P. Barendregt, *The Lambda Calculus, Its Syntax and Semantics*, (Studies in Logic, The Foundations of Mathematics, Volume 103,North-Holland, 1984).

[3] E.W. Beth, Semantic entailment and formal derivability, *Mededelingen de Koninklijke Nederlandse Akademie der Wetenschappen, Afdeeling Letterkunde, Nieuwe Reeks*, **18** (1955) 309–342.

[4] E.W. Beth, Semantic construction of intuitionistic logic, *Mededelingen de Koninklijke Nederlandse Akademie der Wetenschappen, Afdeeling Letterkunde, Nieuwe Reeks*,**19** (1956) 357–388.

[5] Alonzo Church, The Calculi of Lambda Conversion, *Annals of Mathematics Studies*, Number 4, Princeton University Press, 1941.

[6] Paul C. Gilmore, Nominalism, Computers, and the Logic NaDSyL, *Dept of Computer Science Technical Report TR97-09, UBC* **66** (1997) 38 pages.

[7] Paul C. Gilmore, Recursions in NaDSyL, *Dept of Computer Science Technical Report TR97-18, UBC* **66** (1997) 31 pages.

[8] Paul C. Gilmore, NaDSyL and some applications, *Proceedings of the Kurt Gödel Colloquim LNCS Springer* **1289** (2001) 153–166.

[9] Paul C. Gilmore, *Logicism renewed:Logical foundations for mathematics and computer science* (Lecture Notes in Logic 23, Association for Symbolic Logic, 2005)

[10] Leon Henkin, *Completeness in the Theory of Types*, The Journal of Symbolic Logic, vol. 15, 81–91, 1950.

[11] Wilfred Hodges, *An introduction to elementary logic* (Penguin Books, Berlin, 1977, reprinted 1991)

[12] Walter Isaacson, *Einstein, his life and universe* (Simon & Schuster, 2007)

[13] Saul Kripke, Semantical analyses of intuitionistic logic I, in: J.N. Crossely and M.A.E. Dummett, eds., *Formal systems and recursive functions* (North Holland, Amsterdam, 1965) 92–130.

[14] Dag Prawitz, *Completeness and Hauptsatz for Second Order Logic*, Theoria, vol. 3, 246–258, 1967.

[15] Willard Van Orman Quine, *Mathematical Logic*, (Harvard University Press, Cambridge, revised edition 1951).

[16] Willard Van Orman Quine, *from a logical point of view*, (Harper & Row, 1963).

[17] Wilfred Sellars, *Abstract Entities*, Review of Metaphysics, vol. 16 (1963). pp. 625–671.

[18] Wilfred Sellars, *Classes as abstract entities*, Review of Metaphysics, vol. 17 (1963). pp. 67–90.

[19] Lee Smolin, *Three Roads to Quantum Gravity*, (Basic Books, 2001).

[20] A.S. Troelstra and D. van Dalen, *Constructivism in Mathematics, An Introduction*, Volume 121 and 122 in Studies in Logic and the Foundations of Mathematics, (North-Holland, Amsterdam, 1988).

[21] Jean van Heijenoort, *From Frege to Gödel*, (Harvard University Press, 1967).