Towards a theory of mechanizable theories: I
FOL Contexts—the extensional view*†

Carolyn Talcott
Stanford University

Richard W. Weyhrauch
Ibuki Inc.

1 Introduction

The FOL system [7], was designed to test some ideas about how to build thinking individuals by mechanizing the activity of reasoning and providing data structures powerful enough to represent both general information and information about reasoning itself. It is our claim that the notions implicit in first order logic are adequate and appropriate for this task if realized in the proper setting.

Informally an understanding or *theory* is a partial explanation of some aspect of the world. It is about some collection of objects; it has its own jargon (vocabulary) for describing both these objects and the notions we have about them; and it provides facts about the objects expressed in this language. It is a *partial explanation* because in general not all questions that can be posed in the language can be determined on the basis of the theory. The notion of a theory construed as a particular structure is the central thrust of this paper. We call these structures *contexts*. We claim that contexts are the appropriate building blocks for giving a general explanation of both reasoning and cognition.

The logic of FOL generalizes the standard (first order) notions of language and model to admit partial knowledge and provides a natural basis for mechanization of the kind of mixed syntactic and semantic reasoning we do in our everyday life. We emphasize that the FOL notion of partial is used in the sense of partially specified or partially computed information. The logic of FOL is NOT a partial term logic or a calculus of partial functions. Every function mentioned in this paper is total and all reasoning is classical. The reply "I don't know" to a query means just that. It does not mean there is no answer, it just means that with the present information and resources a value can not be determined.

*This research was partially supported by DARPA contract N00039-84-C-0211 and NSF grant CCR-8718605

†ECAI90 Stockholm Sweden

The logic of FOL provides a new way to axiomatize theories which is particularly amenable as a basis for presenting mechanized theories. It is more expressive than using sets of formulas in the sense that it permits more direct and natural formal presentations of informal theories —both mathematical and symbolic. Formalizations in FOL reflect the intensional aspects of theories and the FOL paradigm provides an effective, finitistic, explanation of classical reasoning. It attempts to answer the question: How do we apparently carry out classical reasoning about abstract (possibly infinite) structures with only a finite amount of physical stuff (our brains) to represent things.

New notions presented here from the point of view of logic are: simulation structures, FOL contexts, and inference rules as consistency and constraint preserving maps.

Simulation structures are data structures that are presentations of models. They function as an alternative to the use of axioms for providing information about the intended models of a theory. They contain explicit data objects that represent both individuals of the domain and resource limited computations of functions and relations. In this way, even if we do not have enough resources to determine the value of a function or the truth of a proposition, we will always know if we have an answer and, if not, we will have (as a data structure) a computation that can be restarted if more resources become available. Simulation structures include classical models (using infinite resources) but they also include richer structures that allow us to formalize the intermixing of syntactic and semantic reasoning in a very general way.

FOL contexts are data structures that are presentations of theories. A context contains a language, a simulation structure, a connection between symbols of the language and objects of the simulation structure, and a set of facts expressed in the language. A model of an FOL context is a classical model that extends the simulation structure, satisfies the facts, and preserves the connection between symbols and objects. The consistency of an FOL context is defined in the usual way by the existence of a model.

A context can be queried to extract information—the value of a term, the satisfiablitiy of a formula, etc., or it can be mapped to other contexts using inference rules. The resulting contexts may contain more information or less information than the original context, or they may just record the result of a query. The notion of a rule of inference has been weakened from the classical case to include all the consistency preserving maps on FOL contexts. The use of consistency preserving rather than validity preserving maps as is an essential part of the FOL paradigm. This change in point of view permits us to formalize the use of different, even speculative non-validity preserving, methods of problem solving in the same framework without augmenting first order semantics.

2 Languages and simualtion structures

A language is given by a set of basic syntactic objects and a mechanism for building the expressions of the language from these basic objects. We call the basic syntactic objects symbols. In the following we restrict our attention to a particular class of first-order languages we call FOL languages. Thus the kinds of expressions (terms and formulas) and the mechanism for building them is fixed and we will identify a language with its set of symbols. Symbols have internal structure even though they are atomic entities of a language. Each symbol has an identifier, an application type, and possibly other features. For FOL languages, application types are one of **ind** (individuals), $\mathbf{fun}_n$ (n-ary functions), **tv** (truth values), or $\mathbf{rel}_n$ (n-ary relations) for $n \in \mathbf{Nat}$. Given a language $\mathcal{L}$, terms, formulas, and sentences are defined as usual. By an expression ε of $\mathcal{L}$ we mean a term or formula.

We use a set of symbols to present a language rather than the traditional notion of signature. It is easy to see that for a given set of symbols there is a corresponding first-order signature. Dually, with the self-contained notion of symbol, we can define a general notion of well-formed term or formula and given any well-formed expression we can determine its language – i.e., the set of (non-logical) symbols occurring in the expression.

In programming terms we can think of symbols as as objects that respond at least to the messages asking for an identifier or a application-type, or as records with at least identifier and application-type fields. All Fol structures are specified in a similar "loose" manner. The point is that we can easily add features to such structures without affecting notions and definitions that do not make use of this additional structure. This becomes particularly important at the mechanization level when one may wish to add intensional information to syntactic structures without affecting operations that do not use the added information.

Simulation structures provide a mechanism for representing parts of models and for reasoning by computation in a model. A simulation structure $\mathcal{S}$ contains at least a universe of individuals, U, a collection of computation descriptions, D, a set of computation rules R, and step function Step mapping a description and a rule to the description obtained by applying the rule. We assume the universe contains two distinct elements `t` and `f` (representing booleans). We let ? (pronounced "I don't know") be an element not in U (or any other set unless explicitly included). A simulation structure also provides a map $d2u$ that maps completed computations to elements of U and incomplete descriptions to ?. We write $d \to d'$ if there is some finite (possibly empty) sequence of steps leading from the description d to d'. Since by applying different rules computations may proceed along different paths, we restrict computation so that the same value is obtained by all routes leading to a completed description. In particular we require that the evaluation function Val defined by

$$\mathrm{Val}(d) = \begin{cases} u & (\exists d')(d \to d' \wedge d2u(d') = u \in U) \\ ? & (\forall d')(d \to d' \;\Rightarrow\; d2u(d') = ?) \end{cases}$$

is well defined.

Computational meaning of expressions of a first-order language in a a simulation structure is given by providing a meaning map, δ, mapping expressions to descriptions. (In the original FOL implementation, δ was determined by semantic attachments to symbols of the language. The current formulation extends this and provides for more powerful mechanisms for integrating syntactic and semantic reasoning.) We require that computational meanings of expressions be compatible with the classical interpretation of propositional operations and quantifiers. The full formal requirements are given in the full version of this paper [6]. The requirements for propositional operations are similar to the Kleene [3] strong 3-valued logic, but our interpretation of ? is closer to that found in Feferman [2]. In the case of a quantified expression a perfectly reasonable meaning is the "I don't know" description—a description that never returns a value. If the meaning of a universal quantifier is determined computationally to be $\mathtt{t}$ then there can be no instances whose meaning is determined to be $\mathtt{f}$.

An M-structure $\mathcal{M}$ is a simulation structure together with a language and meaning map satisfying the above conditions. Let $\mathcal{M}_0$, $\mathcal{M}_1$ be M-structures with common language $\mathcal{L}$, and with universes U_0, U_1, and meaning maps δ_0, δ_1. $\mathcal{M}_0$ is *extended by* $\mathcal{M}_1$ just if $U_0 \subseteq U_1$, and for every well-formed expression closed ε of $\mathcal{L}$ and $u \in U_0$, if $\mathrm{Val}_0(\delta_0(\varepsilon)) = u$ then $\mathrm{Val}_1(\delta_1(\varepsilon)) = u$.

Let $\mathcal{M} = (\mathcal{L}, \mathcal{S}, \delta)$ be and M-structure. $\mathcal{M}$ has *witnesses* if an existential witness or universal counterexample can be found in U whenever it can be determined that one should exist using the meaning map and computation mechanism. $\mathcal{M}$ is *open* if it is always consistent to add a new element to the universe. The point is that computationally the best we can do is produce a witness to the falsity of a universal statement or the truth of an existential statement. $\mathcal{M}$ is *complete* if it has witnesses and $\mathrm{Val}(\delta(\varepsilon)) \in U$ for every well-formed closed expression ε of the language. If $\mathcal{M}$ is complete then we can treat $\mathcal{M}$ as a classical first-order model for $\mathcal{L}$ where for each sentence φ of $\mathcal{L}$ we have $\mathcal{M} \models \varphi$ just if $\mathrm{Val}(\delta(\varphi)) = \mathtt{t}$. Not all simulation structures have complete extensions, i.e. it is possible for a simulation structure to be inconsistent. However, an M-structure that is open and has witnesses can be extended to a complete structure.

3 Contexts and operations on contexts

An FOL context $\mathcal{C}$ is a structure that contains a language $\mathcal{L}$, a simulation structure $\mathcal{S}$, a pair of maps $(\delta, \hat{\delta})$ connecting the language and the simulation structure, a set of facts in the language, and possibly other features. We require that $(\mathcal{L}, \mathcal{S}, \delta)$ be an M-structure. $\hat{\delta}$ maps a description to an expression of $\mathcal{L}$ or to ?. It is inverse to δ in the sense that we require that if $\hat{\delta}[d] = \varepsilon$ and $\delta[\varepsilon] = d'$ then if both d and d' return values they must return the same value. We also require that if $\varepsilon = \hat{\delta}[d]$, $\varepsilon' = \hat{\delta}[d']$, and $d \to d'$ or if $\varepsilon' = \hat{\delta}[\delta[\varepsilon]]$ then ε and ε' are either both terms or both formulas. A fact in some language is a structure which has an associated formula of that language. Generally facts will have additional features. For example the formula of a fact may be decomposed into hypotheses and conclusion. A fact may also include a justification, for example, a proof of the conclusion from the hypotheses or a reason for asserting the fact as an axiom. When other features are irrelevant we make no notational distinction between a fact and its associated formula. The set of facts of a context will not in general be deductively closed.

For terms (resp. formulas) ε, ε' we write $\mathtt{mkeq}(\varepsilon, \varepsilon')$ for the formula asserting equality (resp. equivalence) of ε and ε'. $\mathtt{mkeq}(\varepsilon, \varepsilon')$ is a computational consequence of $\mathcal{C}$ if there are descriptions d, d' such that either $\hat{\delta}[d] = \varepsilon$, $\hat{\delta}[d'] = \varepsilon'$, and $d \to d'$; or $\delta[\varepsilon] = d$ and $\hat{\delta}[d] = \varepsilon'$. A model $\mathcal{M}$ for $\mathcal{L}$ is a model of (satisfies) $\mathcal{C}$ just if the following conditions hold: (i) $\mathcal{M}$ extends $\mathcal{L}, \mathcal{S}, \delta$, (ii) $\mathcal{M}$ satisfies all the facts of $\mathcal{C}$, and (iii) $\mathcal{M}$ satisfies all the computational facts of $\mathcal{C}$. A context is *consistent* if it has a model. A formula φ in the language of a context $\mathcal{C}$ is a *consequence* of $\mathcal{C}$ just if the φ is true in every model of $\mathcal{C}$. Note that computational consequences of a context are consequences. The *theory* presented by a context is its set of consequences.

There are many interesting operations on contexts. Two examples are queries and inference rules. A query is a request to determine the value a term or formula in the context. Queries are expressed and answered in the language of the context. Whether or not a query succeeds depends on whether the context actually determines the value and on the mechanism used for evaluation. For example one might just use Val, ignoring any facts. Or one might use classical deduction based only on the facts. Or a combination of these methods might be used. When semantic deduction is used success also depends on being able to map the semantic descriptions involved to expressions in the language.

Inference rules map contexts to contexts. Inference rules may add information, forget information, or make implicit information explicit. For example maps induced by signature and model morphisms can be considered inference rules. Also adding and forgetting language and adding and forgetting semantic connection information can be considered inference rules. The FOL notion of inference rule is considerably more general that the traditional notion, and in this setting there are many interesting properties of inference rules. An inference rule is *consistency preserving* if it maps consistent contexts onto consistent contexts. It is *satisfaction preserving* if it is consistency preserving and non-decreasing with respect to language and the set of consequences. It is *conservative* if it is satisfaction preserving and preserves consequences of the initial language. It is *validity preserving* if it preserves language and the set of consequences

Standard first-order inference rules are validity preserving. A non-standard example of a validity preserving rule is the Step rule of inference. It takes an expression of the language, ε, and a computation rule r as parameters. Assume $\mathrm{Step}[\delta[\varepsilon], r] = d$. If $\hat{\delta}[d] = \varepsilon'$ then application of the Step rule adds $\mathtt{mkeq}(\varepsilon, \varepsilon')$ to the facts of the context and leaves other components unchanged. If $\hat{\delta}[d] = ?$ then application of the Step rule leaves the context unchanged.

An inference rule that is essentially validity preserving is the Let rule. It is similar to but more powerful than the Step rule. Let takes an expression ε, a computation rule r, and a new constant symbol c of the same kind as ε. Again, assume $\mathrm{Step}[\delta[\varepsilon], r] = d$. Then application of the Let rule adds $\mathtt{mkeq}(\varepsilon, c)$ to the facts of the context, updates δ to map c to d, updates $\hat{\delta}$ to map d to c and leaves other components unchanged. Technically this rule is not validity preserving since it introduces new language. However it is *strongly* conservative in the sense that every model of the original context can be extended to a model of the new context by simply extending the meaning map, and every model of the resulting context becomes a model of the original by forgetting the new constant c.

The $[^{\mathrm{Forget-Valid-Consequences}}\}$ rule says that if for $i = 0, \ldots, n-1$

$$\mathcal{C}_i \; [^{\mathrm{r}_i}\} \; \mathcal{C}_{i+1}$$

where each $\mathcal{C}_i$ is a validity preserving rule that simply adds new facts to a context, then

$$\mathcal{C}_0 \; [^{\mathrm{Forget-Valid-Consequence}}\} \; \mathcal{C}$$

where $\mathcal{C}$ is $\mathcal{C}_n$ with any subset of the added facts (added by some $[^{\mathrm{r}_i}\}$) deleted. This rule formalizes the usual activity of proving a lemma and then forgetting the steps in the lemma's proof.

4 The samefringe context

As an example we describe a context $\mathcal{C}_{\mathrm{sf}}$ presenting a theory of binary trees.

$$\mathcal{C}_{\mathrm{sf}} = (\mathcal{L}_{\mathrm{sf}}, \mathcal{S}_{\mathrm{sf}}, \delta_{\mathrm{sf}}, \hat{\delta}_{\mathrm{sf}}, \mathcal{F}_{\mathrm{sf}}).$$

The language $\mathcal{L}_{\mathrm{sf}}$ of $\mathcal{C}_{\mathrm{sf}}$ has indiviual constants a, b, a binary operation p, and binary relations Eq, Sf. The intended interpretation is that p is a pairing operation, Eq is the equality relation on atoms, and Sf is true of two trees if the have the same fringe.

The simulation structure, $\mathcal{S}_{\mathrm{sf}}$, of $\mathcal{C}_{\mathrm{sf}}$ has universe **Sexp**, the set of binary trees (S-expressions) over a set of atoms **At** (which includes the booleans $\mathtt{t}, \mathtt{f}$). A non-atomic tree with left and right subtrees s_0, s_1 is written $\mathtt{<}s_0, s_1\mathtt{>}$. Descriptions are tuples with a descriptor and additional data. They include: completed computations $(\mathbf{ind}, s)$ and $(\mathbf{tv}, s)$; the undetermined computation $(?)$; $(\mathtt{eq}, s_0, s_1)$, describing the binary relation of equality on atoms; $(\mathtt{sf}, s_1, s_2)$, describing the relation of having the samefringe; for $s, s_0, s_1 \in \mathbf{Sexp}$.

The rules of computation include $\{\mathtt{l}, \mathtt{r}, \mathtt{e}\}^*$ with the following single steps

$$(\mathtt{sf}, \mathtt{<<}s_0, s_1\mathtt{>}, s_2\mathtt{>}, s) \overset{\mathrm{l}}{\mapsto} (\mathtt{sf}, \mathtt{<}s_0, \mathtt{<}s_1, s_2\mathtt{>>}, s)$$

$$(\mathtt{sf}, s, \mathtt{<<}s_0, s_1\mathtt{>}, s_2\mathtt{>}) \overset{\mathrm{r}}{\mapsto} (\mathtt{sf}, s, \mathtt{<}s_0, \mathtt{<}s_1, s_2\mathtt{>>})$$

$$(\mathtt{eq}, s_0, s_1) \overset{\mathrm{e}}{\mapsto} \begin{cases} (\mathbf{tv}, \mathtt{t}) & \text{if } s_0 = s_1 \in \mathbf{At} \\ (\mathbf{tv}, \mathtt{f}) & \text{if } s_0 \neq s_1 \in \mathbf{At} \\ (?) & \text{if } \{s_0, s_1\} \not\subset \mathbf{At} \end{cases}$$

The map δ from expressions to descriptions maps the constants a, b to descriptions of distinct atoms $\mathtt{a}, \mathtt{b}$, and lifts homomorphically to closed terms. Closed atomic formulas $R(\ldots, t_j, \ldots)$ map to $(r, \ldots, s_j \ldots)$ where the s_j are images of the t_j and r corresponds to R (via naming convention). For example $\delta[p(a, b)] = (\mathbf{ind}, \mathtt{<a,b>})$, and $\delta[Sf(a, b)] = (\mathtt{sf}, \mathtt{a}, \mathtt{b})$ Other expressions are mapped to $(?)$. The map $\hat{\delta}$ is the inverse of δ on its range.

There are two facts:

$$1: \quad (\forall x) Sf(x, x)$$

$$2: \quad (\forall z, x_0, x_1)(Sf(p(z, x_0), p(z, x_1)) \Leftrightarrow Sf(x_0, x_1))$$

In the remainder of this section we give two example of reasoning in the samefringe context.

Example 1. Let $\varphi_{\mathrm{eq}} = Eq(a, b)$ then we have

$$\delta[\varphi_{\mathrm{eq}}] = (\mathtt{eq}, \mathtt{a}, \mathtt{b})$$

$$\mathrm{Step}[\mathtt{e}][\delta[\varphi_{\mathrm{eq}}]] = (\mathbf{tv}, \mathtt{f})$$

thus, by the Step rule

$$\mathcal{C}_{\mathrm{sf}} \; [^{\mathrm{Step}[\mathtt{e}]}\} \; \mathcal{C}'_{\mathrm{sf}}$$

where $\mathcal{C}'_{\mathrm{sf}}$ is $\mathcal{C}_{\mathrm{sf}}$ with the fact

$$Eq(a, b) \Leftrightarrow \mathtt{False}$$

then using ordinary first-order reasoning we have

$$\mathcal{C}'_{\mathrm{sf}} \; [^{\mathrm{Logic}[\mathtt{PROP}]}\} \; \mathcal{C}''_{\mathrm{sf}}$$

where $\mathcal{C}''_{\mathrm{sf}}$ contains the additional fact $\neg Eq(a, b)$. This shows how simulation structures and semantic reasoning can be used as an alternative axiomatization in the case of finitely many distinct objects without having to write down a large number of axioms. In particular FOL contexts can be used in this way to help solve the frame problem.

Example 2. Let

$$\varphi_{\mathrm{sf}} = Sf(p(p(a, b), p(a, b)), p(p(a, p(b, a)), b))$$

then we have

$$\delta[\varphi_{\rm sf}] = (\texttt{sf}, \texttt{<<a,b>,<a,b>>}, \texttt{<<a,<b,a>>,b>})$$

$$\mathrm{Step}[\texttt{l}][\delta[\varphi_{\rm sf}]] = (\texttt{sf}, \texttt{<a,<b,<a,b>>>}, \texttt{<<a,<b,a>>,b>})$$

$$\mathrm{Step}[\texttt{l},\texttt{r}][\delta[\varphi_{\rm sf}]] =$$
$$(\texttt{sf}, \texttt{<a,<b,<a,b>>>}, \texttt{<a,<<b,a>,b>>})$$

$$\hat{\delta}[\mathrm{Step}[\texttt{l},\texttt{r}]][\delta[\varphi_{\rm sf}]]] =$$
$$Sf(p(a,p(b,p(a,b))),p(a,p(p(b,a),b)))$$

thus, by the Step rule

$$\mathcal{C}_{\rm sf}\ [^{\mathrm{Step}[\texttt{l},\texttt{r}]}\}\ \mathcal{C}'_{\rm sf}$$

where $\mathcal{C}'_{\rm sf}$ is the same as $\mathcal{C}_{\rm sf}$ except that $\mathcal{C}_{\rm sf}$ has one additional fact

$$Sf(p(p(a,b),p(a,b)),p(p(a,p(b,a)),b))$$
$$\Leftrightarrow\ Sf(p(a,p(b,p(a,b))),p(a,p(p(b,a),b)))$$

then using the initial facts and ordinary first-order reasoning we have

$$\mathcal{C}'_{\rm sf}\ [^{\mathrm{Logic}[\forall\mathsf{E},\mathsf{PROP}]}\}\ \mathcal{C}''_{\rm sf}$$

where $\mathcal{C}''_{\rm sf}$ contains the additional fact

$$Sf(p(a,p(b,p(a,b))),p(a,p(p(b,a),b)))$$
$$\Leftrightarrow\ Sf(p(b,p(a,b)),p(p(b,a),b)).$$

Similary, by semantic reasoning we can construct a context with the fact

$$Sf(p(b,p(a,b)),p(p(b,a),b))$$
$$\Leftrightarrow\ Sf(p(b,p(a,b)),p(b,p(a,b)))$$

added. Then, applying the second axiom followed by the first axiom, we arrive at a context, $\mathcal{C}_{\rm almost-done}$, that contains the fact $\varphi_{\rm sf}$. Finally,

$$\mathcal{C}_{\rm almost-done}\ [^{\rm Forget-Valid-Consequences}\}\ \mathcal{C}_{\rm done}$$

where $\mathcal{C}_{\rm done}$ is $\mathcal{C}_{\rm sf}$ with the single additional fact φ. Note that all of the inference rules used in this example were validity preserving rules. Also note that although φ is a consequence of $\mathcal{C}_{\rm sf}$ it does not follow either from the axioms or the model alone. This kind of reasoning is not accounted for by customary logical deduction. In the above case even forgeting is validity preserving. There is no analogue to forgetting facts in the usual description of theories.

5 Discussion and future work

The logic of FOL can be considered at three levels: extensional, intensional, and representational. In this paper we treated part of the extensional view of FOL contexts. The extensional view of FOL systems and constraints will be presented in a companion paper. At the intensional level we present a further refinement of the structures described here and show that each such structure has a finite presentation. Furthermore we will show how the logic of FOL itself can be represented using FOL structures. At the representational level we show the representation of FOL structures as particular data structures and show detailed examples. We will then show how to generalize extensional semantics by allowing these data structures to have shared substructures and to be operated on by rules of inference that can destructively update data structures rather than simply create new structures as required by normal extensional semantics. This will create an entirely new class of ideas that can be finitistically presented. At each level the reason for the choices of basic FOL notions will become clearer.

The ideas presented here are a synthesis of the work of Weyhrauch (cf. [7], [1]) on the FOL systems and the work of Talcott (cf. [5], [4]) on computation theories and programming language semantics. It was impossible to present a clear description of FOL semantics before we understood that the correct formal way of characterizing the FOL notion of "I don't know" was to make explicit the steps implicit in finding the value of a function and to formulate the computation theory in such a way that each step of a computation is available for inspection and computations are restartable. This allows us to define the notion of compatability of computation descriptions, in addition to equality of values. This formulation creates a semantics for FOL structures that is not effected by any details of the choice of what to do in situations of limited resources.

There is a sense in which the notion of context is a conservative extension of the usual presentation of a theory by a set of axioms. A context with a (non-empty) effective simulation structure can be replaced by a context with an empty simulation structure, and a suitably extended language and set of facts to account for the consequences entailed by valuation. However, doing this generally results in a less natural presentation and loses expressive power with respect to defining inference mechanisms. The point here is that with the advent of computers it is possible to easily manipulate structures that are richer than strings of symbols or sentences of a language and progress can be made by using these new structures. If we use these new structures in their natural way including sharing and updating with side effects, then viewing FOL contexts as simple conservative extensions of ordinary logic becomes even more cumberson because it will in general require translating to a completely new theory that takes explicit account of sharing. When we go to asynchronous interactions then it is unlikely that still have a conservative extension at all.

This paper is a step toward our goal of providing finite structures that can reason with rules that can be understood with classical semantics.

Acknowledgements. We would like to thank Jacqueline's for wonderful food and atmosphere, and for our table in the corner where much of the discussion leading to this paper was carried out.

References

[1] Aiello, L. and Weyhrauch, R. W. Using metatheoretic reasoning to do algebra, *Proceedings of the 5-th automated deduction conference*, Les Arcs France, 1980.

[2] Feferman, S. Non-extensional type-free theories of partial operations and classifications, I, *Lecture Notes in Mathematics*, **500**, (Springer-Verlag), pp. 73–118. 1974.

[3] Kleene, S. C. *Introduction to metamathematics*, (North-Holland), 1952.

[4] Mason, I. A. and Talcott, C. L. Programming, Transforming, and Proving with function abstractions and memories, *Proceedings of the 16th EATCS Colloquium on Automata, Languages and Programming, Stresa, Italy*. 1989.

[5] Talcott, C. L. *The essence of Rum: A theory of the intensional and extensional aspects of Lisp-type computation*, Ph.D. Thesis, Stanford University. 1985.

[6] Talcott, C. L., and Weyhrauch, R. W. Towards a theory of mechanizable theories I: FOL Contexts—the extensional view, in preparation.

[7] Weyhrauch, R. W. Prolegomena to a theory of formal reasoning, *Artificial Intelligence*, **13**, pp. 133–170. 1980.