Axiom Schemata as Metalevel Axioms: Model Theory*

Timothy L. Hinrichs and **Michael R. Genesereth**
Logic Group
Computer Science Department
Stanford University
{thinrich, genesereth}@stanford.edu

Abstract

Logicians frequently use axiom schemata to encode (potentially infinite) sets of sentences with particular syntactic form. In this paper we examine a first-order language in which it is possible to write expressions that both describe sentences and assert the truth of the sentences so described. The effect of adding such expressions to a knowledge base is the same as directly including the set of described sentences.

Introduction

Logicians frequently use axiom schemata to encode (potentially infinite) sets of sentences with particular syntactic properties.

As an example, consider the axiom schema shown below, where ϕ is a sentence with a single free variable.

$$\phi(0) \wedge \forall n.(\phi(n) \Rightarrow \phi(n+1)) \Rightarrow \forall n.\phi(n)$$

This schema encodes infinitely many sentences, jointly comprising the principle of mathematical induction. The following sentences are instances.

$$\begin{array}{l} p(0) \wedge \forall n.(p(n) \Rightarrow p(n+1)) \Rightarrow \forall n.p(n) \\ q(0) \wedge \forall n.(q(n) \Rightarrow q(n+1)) \Rightarrow \forall n.q(n) \\ \forall x.r(x,0) \wedge \forall n.(\forall x.r(x,n) \Rightarrow \forall x.r(x,n+1)) \\ \qquad\qquad\qquad \Rightarrow \forall n.\forall x.r(x,n) \end{array}$$

Axiom schemata are differentiated from axioms due to the presence of metavariables or other metalinguistic notation (such as dots or star notation), together with conditions on the variables. They *describe* sentences in a language, but they are not themselves sentences in the language. As a result, they cannot be manipulated by procedures designed to process the language (presentation, storage, communication, deduction, and so forth) but instead must be hard coded into those procedures.

In this paper we examine a language in which it is possible to write expressions that describe sentences and to write sentences that assert the truth of the sentences so described. The effect of adding such sentences to a knowledge base is the same as directly including the (potentially infinite) set of described sentences in the knowledge base.

The use of such a language simplifies the construction of knowledge-based systems, since it obviates the need for building axiom schemata into deductive procedures. It also makes it possible for systems to exchange axiom schemata with each other and thereby promotes knowledge sharing.

The trick is to provide terminology in our language for talking about expressions in our language and a "truth" predicate to assert the truth or falsity of the sentences so described.

This trick is similar to the trick used to develop a first order theory for the notion of truth. Unfortunately, those efforts are fraught with difficulties of dealing with paradoxes. Once we have a way of describing sentences and a truth predicate, it is tempting to assert that a sentence satisfies the truth predicate if and only if it is true; but this causes problems (Tarski 1956). For example, sentences like "This sentence is false" become self-contradictory.

A number of solutions to this problem have been presented over the years. See (Turner 1991) for a description of the problem and some of its solutions. See also (Attardi & Simi 1995; DesRivieres & Levesque 1965; Haas 1986; Perlis 1985; Sato 1992; Weyhrauch 1980; Chen, Kifer, & Warren 1989) for additional material. The various solutions to this problem offer different tradeoffs of understandability, coverage, and implementability.

The framework described here is much simpler than any of these previous approaches. We assert that a sentence satisfies our truth predicate if and only if the sentence is true, but we say this only for sentences that do not contain the truth predicate. This eliminates the paradox mentioned above, since sentences like "This sentence is false" cannot be expressed.

From the point of view of formalizing truth, this is a cheat, since it fails to cover those interesting cases where sentences contain the truth predicate. However, from the point of view of capturing axiom schemata *not* involving the truth predicate, it works just fine. Furthermore, unlike the solutions to the problem of formalizing truth, our framework is easy for users to understand, and it is easy to implement.

This paper presents the model theory for this solution to the problem. A separate paper describes several proof pro-

*Our thanks to Don Geddis, Oliver Duschka, and Nat Love for their editorial and technical comments on the paper. We would also like to thank Hewlett-Packard for their support of the work reported herein.

cedures that implement the techniques developed here.

In what follows, the baselevel language we will be describing, named L_B, is a standard, first-order language. The metalevel language of descriptions, named L_M, is also a standard, first-order language and is disjoint from L_B. It includes a mechanism for asserting the truth of described L_B sentences: a single, unary predicate tr. The first-order language produced from the union of L_B's vocabulary and L_M's vocabulary define the language that we investigate in this paper. It will be denoted by L.

The first section explains in more detail how L_B sentences are described and what it means for a set of sentences to be finitely describable. The second section discusses the machinery used to assert the truth or falsity of a sentence so described, i.e. it axiomatizes tr. The third section proves a notion of soundness and completeness for our scheme.

Expression Descriptions

With the exceptions noted below, we use standard infix notation. Whether a symbol is a variable, object constant, function constant, or relation constant will be clear from context. There are the usual logical operators $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\Leftarrow$, and $\Leftrightarrow$; and there are the usual quantifiers $\forall$ and $\exists$. A logical expression is any sequence of symbols in the logic, e.g. $\neg \wedge \wedge p$, $p(a \wedge b)$, $p(b)$. A logical sentence can be either open or closed but in both cases is well-formed, e.g. $p(x)$, $\forall x.p(x)$. The definitions of model, variable assignment, and logical implication are standard. See (Genesereth & Nilsson 1987).

Expression Descriptions

First we discuss a choice of method for describing sentences in the baselevel language L_B. Such sentences can be described in a variety of ways, e.g. Gödel numbers or expression constructors of various sorts (Haas 1986; Perlis 1985). Our scheme constructs the metalevel language L_M from the vocabulary of the baselevel language L_B. In what follows we introduce a new object constant for each of the symbols in L_B that happens to look like the symbol surrounded by quotes, and we use a set of constructor functions $forall$, $exists$, $impl$, $bicond$, $disj$, $conj$, neg, $relnSent$, and $funcTerm$.[1] For example, we use the term $relnSent(\text{“}p\text{”}, \text{“}a\text{”}, \text{“}b\text{”})$ to denote the sentence $p(a, b)$ and $conj(relnSent(\text{“}p\text{”}, \text{“}a\text{”}), relnSent(\text{“}q\text{”}, \text{“}a\text{”}))$ to denote $p(a) \wedge q(a)$. (This allows us to describe expressions that are not sentences, but this causes no problems.)

In order to simplify our examples that involve quoted expressions, we use a backquote-like notation as syntactic sugar. In particular, to denote an expression, we use matching quotes around the expression; and, to unquote an expression within these quotes, we use matching brackets. For example, $p(\text{“}q(<x>, b)\text{”})$ is equivalent to $p(relnSent(\text{“}q\text{”}, x, \text{“}b\text{”}))$. We intend these quoted expressions to be interpreted as macros which translate into the appropriate constructor terms.

Example 1 To be more precise about our method for describing sentences of L_B, suppose L_B is built from the following finite set of relation, function, and object constants with an infinite set of variables. An infinite set of variables and finitely many constants is not required for our scheme, but it is a common case.

relations of L_B: $\{p, q\}$
functions of L_B:$\{f, g, h\}$
objects of L_B: $\{a, b, c\}$
variables of L_B: $\{v_1, v_2, \ldots\}$

The description language, L_M, includes the object constants “p”, “q”, “f”, “g”, “h”, “a”, “b”, “c” and “v_i” for each variable v_i. To differentiate which quoted symbol is of which type, L_M includes the unary relation constants $relation$, $function$, $object$, and $variable$. Thus the vocabulary for L_M includes the following. [2]

relations of L_M:$\{variable, relation, function, object, =\}$
functions of L_M:$\{forall, exists, impl, bicond, disj, conj, neg, relnSent, funcTerm\}$
objects of L_M: $\{\text{“}p\text{”}, \text{“}q\text{”}, \text{“}f\text{”}, \text{“}g\text{”}, \text{“}h\text{”}, \text{“}a\text{”}, \text{“}b\text{”}, \text{“}c\text{”}, \text{“}v_1\text{”}, \text{“}v_2\text{”}, \text{“}v_3\text{”}, \ldots\}$

We require the vocabularies of L_M and L_B to be disjoint. L's vocabulary is the union of these two vocabularies. □

Example 2 Suppose we want to describe the set of all axioms that require the relations of L_B to be reflexive, symmetric, and transitive; denote this set with Δ. In this example we define a new relation, d, that names all the L_M encodings for sentences in our set. That is, for every sentence $\psi \in \Delta$, we want to ensure that $d(\text{“}\psi\text{”})$ is true.

$$\forall r.(relation(r) \Rightarrow d(\text{“}\forall x.<r>(x,x)\text{”}))$$
$$\forall r.(relation(r) \Rightarrow d(\text{“}\forall xy.<r>(x,y) \Leftarrow <r>(y,x)\text{”}))$$
$$\forall r.(relation(r) \Rightarrow d(\text{“}\forall xyz.<r>(x,y) \Leftarrow <r>(x,z) \wedge <r>(z,y)\text{”}))$$

Notice that r is a metalevel variable and appears unquoted in all the heads of the rules. Had we left the r quoted in the reflexivity axiom for example, i.e. $d(\text{“}\forall x.r(x,x)\text{”}))$, the sentence $\forall x.r(x,x)$ would have been true in d (assuming there is at least one relation in L_B), regardless whether r is a relation in L_B or not. By escaping the r as shown above, we get the desired results, namely that every relation in L_B is reflexive, symmetric, and transitive. □

In this example, the space savings is small if the number of relation constants is finite, but in the case of infinitely many relation constants, the metalevel approach allows the infinite axiom set Δ to be finitely described.

We now give the semantics of the distinguished L_M constants axiomatically. The first set of axioms ensures that every pair of quoted expressions that look different are unequal. In what follows, we will refer to the functions $forall$,

[1] Actually there will be one $relnSent$ for each arity of relation constant in L_B; the same holds for $funcTerm$ and the function constants. In this paper, we will treat a single ternary $relnSent$ and a single ternary $funcTerm$, but the results extend immediately to sets of $relnSent$ and $funcTerm$.

[2] We have left out the relation constant tr, which will be properly introduced in the next section.

$exists$, $impl$, $bicond$, $disj$, $conj$, neg, $relnSent$, and $funcTerm$ collectively as the constructor functions. neg takes one argument; $relnSent$ and $funcTerm$ are ternary, and the rest are binary.

1. Let σ_1 and σ_2 be a distinct pair of constants or variables in L_B. The sentence "σ_1" $\neq$ "σ_2" is true.
2. Let σ be a constant or variable in L_B and c be a constructor function. Then $\forall \overline{x}.$"σ" $\neq c(\overline{x})$ is true.[3]
3. Let c be a constructor function. $\forall \overline{x}\,\overline{y}.(c(\overline{x}) = c(\overline{y})) \Rightarrow \overline{x} = \overline{y})$ is true.
4. Let c_1 and c_2 be distinct constructor functions. $\forall \overline{x}\,\overline{y}.c_1(\overline{x}) \neq c_2(\overline{y})$ is true.

The second set of axioms defines the unary type predicates $relation$, $function$, $object$, and $variable$.

5. Let σ be a relation constant in L_B. $relation($"σ"$)$ is true. Let σ be a function constant, object constant, or variable in L_B. $\neg relation($"σ"$)$ is true. Likewise for $function$, $object$, and $variable$.
6. Let c be a constructor function. The sentence $\forall \overline{x}.\neg relation(c(\overline{x}))$ is true. Likewise for $function$, $object$, and $variable$.

Because all these sentences are in the metalevel language L_M, we will refer to these six groups of axioms as Γ_M.

Finite Describability

The primary use of L_M is to express axiom schemata. In this paper, we are interested only in axiom schemata that can be finitely described.

Definition 1 (Designator) *A* designator *for a set of L_B-sentences Δ is a set of L_M-sentences D with a distinguished, unary relation d such that for any sentence ψ in Δ, $\Gamma_M \cup D \models d($"$\psi$"$)$ and for any L_B-expression ψ not in Δ, $\Gamma_M \cup D \models \neg d($"$\psi$"$)$. D does not include the relation constant tr, and $D \cup \Gamma_M$ is satisfiable.*

Definition 2 (Finite Describability) *A set of L_B-sentences Δ is* finitely describable *if and only if there is a finite designator for Δ.*

Example 3 Let us return to the example that required all relations in the language to be reflexive, symmetric, and transitive. Recall we named this infinite set Δ; we will use D to denote our attempt at designating Δ.

D is not a valid designator of Δ because it only satisfies the first half of the definition. For every sentence ψ in Δ, $\Gamma_M \cup D \models d($"$\psi$"$)$. But the second condition in the definition is that for every expression ψ in L_B that is not in Δ, $\Gamma_M \cup D \models \neg d($"$\psi$"$)$. $d(x)$ does not meet that requirement. For instance, $\Gamma_M \cup D \not\models \neg d($"$\forall xy.p(x,y)$"$)$.

To fix that, we need to close the definition of d.

$$\begin{aligned}&\forall t.(d(t) \Leftrightarrow \exists r.(relation(r) \wedge \\ &\quad (t = \text{“}\forall x.<r>(x,x)\text{”} \vee \\ &\quad t = \text{“}\forall xy.<r>(x,y) \Leftarrow <r>(y,x)\text{”} \vee \\ &\quad t = \text{“}\forall xyz.<r>(x,y) \Leftarrow <r>(x,z) \wedge <r>(z,y)\text{”})))\end{aligned}$$

[3] If c has arity n then $\overline{x}$ is a sequence of n distinct variables.

This closed version of D is a designator for Δ. $\square$

Note that completeness is not necessary for a designator. A designator need only ensure that for every L_B-expression, the distinguished relation is either true or false for that expression. The fact that other elements might or might not be true is of no consequence.

Nonstandard Models of Γ_M

The axioms of Γ_M were chosen to capture certain types of L_M models–those whose domains represent exactly the expressions in L_B. But Γ_M only approximates that class; every model satisfying Γ_M maps every quoted L_B expression onto a distinct element in its domain, thus ensuring the domain is infinite. Clearly there are models of Γ_M with elements besides those needed to represent L_B-expressions. In fact, Löwenheim-Skolem-Tarski(LST) ensures there are models of every infinite cardinality. Nevertheless, we can view the L_M domains as the set of all expressions built out of the L_B vocabulary extended with those extra elements.

Suppose L_B's vocabulary is limited to a single relation constant p and a single object constant a. The elements in every domain represent the sentences $p(a)$, $p(a) \vee \neg p(a)$ as well as the expressions $p(a,a)$, $p(p)$, $p \wedge \neg a$, $p(a \vee a)$, etc. The application of the constructor functions to the vocabulary of L_B produces all the expressions in L_B. This is the minimum set of elements required by Γ_M.

Consider an element, π, that does not correspond to any L_B-expression. Think of π as another constant in the vocabulary and apply the constructor functions to that extended vocabulary, producing new expressions where π is a constant. Because there are infinitely many L_B-expressions to start, there will be infinitely many more elements in the metalevel domain: one for each expression including π.

This application of constructor functions does not change the cardinality of the domain since there are a finite number of constructor functions, each of which operates on only a finite number of elements at a time. But also notice that every constructor function operates on every element in the metalevel domain; thus every element can be considered an expression. We will refer to the elements in the metalevel domain not representing expressions in L_B as Z-expressions.

It is tempting to try removing these nonstandard models, but LST ensures we cannot. Moreover, we might not want to remove them even if we could. Suppose the only elements in the domain of all models are the L_B-expressions. Consider the infinite set of sentences $\{p(\text{“}v_1\text{”}), p(\text{“}v_2\text{”}), \ldots, p(\text{“}v_n\text{”}), \ldots\}$, which together say that p is true for all the variables in L_B. Without those nonstandard models, the sentence $\forall x.(var(x) \Rightarrow p(x))$ would be entailed but would admit no finite proof. Thus, the nonstandard models are necessary for compactness, and as it turns out they are irrelevant for our purposes, as will be explained at the end of the next section.

true Sentences

In this section we will present axioms that establish a formal connection between sentences in L_B and terms in L_M that describe sentences in L_B. These axioms define a predicate,

tr, to assert the 'truth' or 'falsity' of a sentence so described, e.g. $tr(\text{“}p(a)\text{”})$ will ensure $p(a)$ is true. The axioms state that the encoding for an L_B sentence satisfies the tr predicate if and only if that sentence is true. This is the seventh axiom set we have introduced.

7. Let ϕ be a closed sentence in L_B. $tr(\text{“}\phi\text{”}) \Leftrightarrow \phi_B$ is true.

The B subscript on ϕ_B indicates ϕ is in the baselevel language, and because L_B does not include the relation constant tr, these axioms do not admit the standard paradoxes. We use the symbol Γ to refer to axiom groups 1-7.

Note the importance of the condition that only closed sentences are included in the tr axioms. If we were to relax this assumption, we could quickly get contradictions. Consider, for example, the sentence $tr(\text{“}p(x) \vee q(x)\text{”})$. If we had axiom (7) for all sentences, we could infer from this $p(x) \vee q(x)$, from which we could infer $tr(\text{“}p(x)\text{”}) \vee tr(\text{“}q(x)\text{”})$. Originally, we had the sentence asserting that, for each x, either $p(x)$ is true or $q(x)$ is true. From this we infer that either $p(x)$ is true for all x or $q(x)$ is true for all x.

Because of the disjointness of L_M and L_B, the truth value of an L_B-sentence is undefined in an L_M-model, and the truth value of an L_M-sentence is undefined in an L_B-model. Moreover, since all the axioms in Γ_M are in L_M, they do nothing to foster a relationship between L_M and L_B either. The only connection between these two languages comes from the tr axioms, which are in neither L_M nor L_B but are members of the language L.

We will thus treat L-models as the union of an L_B-model and an L_M model, constructed by gluing the two models together. The result is a two-sorted interpretation. One domain, the baselevel domain, is equivalent to the L_B-model's domain, and the other, the metalevel domain, is equivalent to the L_M-model's domain. All the constants of L_B map into the baselevel domain exactly as they did in the original, and all the constants in L_M map into the metalevel domain just as in the original.

An L-model with two domains requires additional notation to clarify which domain a particular quantifier ranges over. When confusion may arise, we will subscript quantifiers with B and M to indicate their association with the baselevel and metalevel domains, respectively. The sentences of Γ_M will always be confined to the metalevel domain, as will any sentences used to define a designator. The subscript B in axiom group (7) restricts its quantifiers to the baselevel domain.

The last section stated the existence of nonstandard models of Γ_M by way of Z-expressions. The same holds for L-models of Γ, where the Z-expressions occur in the metalevel domain. The existence of these nonstandard models leads to some unexpected consequences. For instance, the following two sentences are satisfiable when a is an object constant in L_M: $tr(\text{“}p(<a>)\text{”})$ and $tr(\text{“}\neg p(<a>)\text{”})$. If a maps to a Z-expression, we claim there is a model that satisfies both of these statements simultaneously.

Consider any L-model N that satisfies Γ and has Z-expressions. Notice that the only constraint on tr in N is from the axiom set $tr(\text{“}\phi\text{”}) \Leftrightarrow \phi_B$, where ϕ is a closed sentence in L_B. Because of the restriction that ϕ is a closed sentence in L_B, N is free to satisfy tr for any of the Z-expressions it likes. Thus for the closed sentences in L_B, $N(tr)$ consists of exactly those sentences satisfied by the L_B reduct of N. For all other expressions, $N(tr)$ is entirely unrestricted. (The L_B reduct of an L-model N is the baselevel domain and all mappings for the elements in L_B.)

These nonstandard models would cause problems if we were interested in the consequences of particular L_M axiomatizations, but because our only interest lies in answering L_B queries about the sentences described by the L_M axioms, the nonstandard models are inconsequential.

Formally, let K be a class of L-models. K entails an L_B sentence ϕ_B if and only if the L_B reducts of K entail ϕ. The next section proves the L_B reducts of the L-models that satisfy Γ and a description of Δ are exactly the L_B-models that satisfy Δ itself. We can therefore choose any subset of the models of Γ that we like as long as no L_B reduct is lost. In particular, we can choose to consider only the standard models of Γ, i.e. those whose metalevel domains consist of exactly the L_B-expressions.

Soundness and Completeness

Formally we have not yet defined what a description for a set of sentences is, but we have laid the groundwork by defining a *designator* in the section on Expression Descriptions. A *description* for a set of sentences Δ is the set $D \cup \{d(x) \Rightarrow tr(x)\}$, where d is the distinguished relation for the designator D.

Now we move on to the proofs of soundness and completeness. Soundness and completeness in this context are more limited than is usual. The point of employing L_M and Γ is to ensure descriptions of a set of L_B sentences Δ have the same logical consequences as Δ itself. Thus soundness requires that if a description D along with Γ entail some sentence ϕ in L_B, then the sentences described by D entail ϕ as well. Completeness is the converse: if some set of sentences Δ entails some sentence ϕ then a description of Δ along with Γ also entail ϕ. In both cases, ϕ is a baselevel sentence; thus, soundness and completeness will be shown for L_B queries.

Although in what follows we assume finite describability, the results in this paper apply to any describable set of axioms, finite or not, unless otherwise stated. However, since our interest lies in the construction of finite systems and finite communication, finite describability is essential.

Soundness and completeness build upon two theorems. The Consistency theorem ensures that for every designator D of a set of L_B-sentences Δ, every L_B-model of Δ can be extended to an L-model that satisfies Δ, Γ, and D. The Compatibility theorem shows that every such L-model satisfies not only Δ but also every description of Δ as well.

Theorem 1 (Consistency) *Let Δ be a satisfiable set of closed L_B-sentences and D a designator for Δ. Every L_B-model that satisfies Δ can be extended to an L-model that satisfies $\Delta \cup \Gamma \cup D$.*

Proof: We prove that $\Gamma \cup D$ is consistent with Δ by taking one of Δ's L_B-models N and one of $\Gamma_M \cup D$'s L_M-models M and creating an L-model I that satisfies $\Delta \cup \Gamma \cup D$. The

definition of a designator ensures that $\Gamma_M \cup D$ is satisfiable, thus ensuring the existence of M.

Set the baselevel domain of I to the domain of N. Set the metalevel domain of I to the domain of M. Ensure that $I(c) = N(c)$ for every baselevel constant c and $I(c) = M(c)$ for every metalevel constant c. Thus the baselevel constants map into the baselevel domain and the metalevel constants into the metalevel domain. I must satisfy Δ since the L_B reduct satisfies Δ; likewise, I must satisfy $\Gamma_M \cup D$ since the L_M reduct satisfies $\Gamma_M \cup D$.

To satisfy all of Γ, we need only define $I(tr)$ so that $I \models tr(\text{“}\phi\text{”}) \Leftrightarrow \phi_B$. (Nowhere in $\Gamma_M \cup D \cup \Delta$ is tr mentioned; thus, we are free to assign it however we like.) Build tr to be *false* of exactly those L_B-expressions that are not closed logical consequences of N, i.e. those elements in the metalevel domain that do not represent a closed, L_B-sentence that is satisfied by N. Thus tr is true of all the logical consequences of N, which satisfies the needed axioms. Since I satisfies Δ and $\Gamma \cup D$, it satisfies them both. $\square$

It should be noted that this proof of consistency, at least for the tr predicate, is a special case of Perlis'(1985) classical work that restricts the scope of the truth predicate. In that proof, sentences of the form $tr(\text{“}\phi\text{”}) \Leftrightarrow \phi^*$ are shown to be consistent, where ϕ^* is a transformation of ϕ based on occurrences of tr. In our proof, there are no occurrences of tr within ϕ, which makes $\phi^* = \phi$.

The next theorem strengthens Consistency by showing that the model resulting from the construction above satisfies not only Δ and a designator of Δ, but every description of Δ as well. When this relationship exists between an L-model and an L_B-model for a set of sentences Δ, we say that the L model is *compatible* with the L_B-model for that designator.

Theorem 2 (Compatibility) *Let Δ be a satisfiable set of closed L_B-sentences that is finitely describable. If I is an L_B-model for Δ and D is a designator of Δ, then there is an L-model, I', that is compatible with I for D.*

Proof: Suppose that I is an L_B-model for Δ, and I' is the L-model built from I using the construction in the Consistency proof. Let d be the distinguished relation in the designator D. We need to show that I' satisfies $d(x) \Rightarrow tr(x)$. Every element in the domain not representing an L_B-expression is true of tr, which ensures I' satisfies $d(x) \Rightarrow tr(x)$. The only remaining elements are those representing L_B-expressions.

Consider an arbitrary L_B-expression ϕ. If ϕ is *not* a member of Δ, then $\Gamma \cup D \models \neg d(\text{“}\phi\text{”})$ by the definition of a designator. Since I' satisfies $\Gamma \cup D$ it cannot satisfy $d(\text{“}\phi\text{”})$, and so $d(x) \Rightarrow tr(x)$ as a whole is true in I'. If ϕ is a member in Δ, then $d(\text{“}\phi\text{”})$ is entailed by $\Gamma \cup D$ and therefore satisfied by I'. To satisfy $d(x) \Rightarrow tr(x)$, I' must therefore satisfy $tr(\text{“}\phi\text{”})$. Since I satisfies Δ it must satisfy ϕ. By construction, I' must therefore satisfy $tr(\text{“}\phi\text{”})$. Again the sentence is true. Since I' satisfies both $\Gamma \cup D$ and $\{d(x) \Rightarrow tr(x)\}$, it satisfies their union. $\square$

The significance of this theorem is that for any description D of Δ, every L_B-model of Δ can be extended to at least one L-model of $\Gamma \cup D$ that satisfies every description of Δ, i.e. using the description does not cause us to lose any important L_B-model reducts, which is essentially enough for soundness. For completeness to hold, it must also be the case that there are no extra L_B-model reducts that result from using the description instead of the sentences themselves. For this, we must show that every L-model that satisfies Γ and some description of Δ, when reduced to L_B, satisfies Δ.

Theorem 3 (Soundness) *Let Δ be a set of closed sentences in L_B with description D. For any L_B-sentence ψ, $D \cup \Gamma \models \psi_B$ implies $\Delta \models \psi$.*

Proof: If Δ is unsatisfiable, the conclusion follows immediately, so suppose Δ is satisfiable. Further suppose $D \cup \Gamma \models \psi_B$, and for the purpose of contradiction that there is an L_B-model I such that I satisfies Δ but not ψ. Because I satisfies Δ, we can apply the Compatibility theorem to ensure there is a compatible L-model I' for the designator of D that satisfies $D \cup \Gamma$. Since it is compatible and I does not satisfy ψ, we know that I' does not satisfy ψ when restricting quantifiers to the baselevel domain. Thus I' does not satisfy ψ_B. This contradicts the assumption that $D \cup \Gamma \models \psi_B$. Therefore I must satisfy ψ and because I was chosen arbitrarily $\Delta \models \psi$. $\square$

Theorem 4 (Completeness) *Let Δ be a set of closed sentences in L_B with description D. For any L_B-sentence ψ, $\Delta \models \psi$ implies that $D \cup \Gamma \models \psi_B$.*

Proof: Suppose that $\Delta \models \psi$, and suppose that I' is any L-model that satisfies $D \cup \Gamma$.

First, we show that I' satisfies $\Delta \cup D \cup \Gamma$ when the quantifiers in Δ are restricted to the baselevel domain. Consider any sentence ϕ_B in Δ_B. By the definition of a description and one step of deduction, we know that $\Gamma \cup D \models tr(\text{“}\phi\text{”})$. By the definition of Γ, we know that the sentence $tr(\text{“}\phi\text{”}) \Leftrightarrow \phi_B$ is an element of Γ. By a simple application of modus ponens, we get $D \cup \Gamma \models \phi_B$. Therefore, any model of $D \cup \Gamma$ is a model of ϕ_B. This argument holds for all elements of Δ_B. Thus, every model of $D \cup \Gamma$ is a model of $\Delta_B \cup D \cup \Gamma$. This includes I'. If Δ were unsatisfiable, there would be no such model I' and the result follows immediately.

Since I' satisfies $\Delta_B \cup D \cup \Gamma$, it must also satisfy Δ_B. Since every sentence in Δ_B is contained in the language L_B and every quantifier ranges over the baselevel domain, the L_B-reduct of I' must satisfy Δ_B. But then, by our supposition, that reduct satisfies ψ_B, and therefore so must I'. $\square$

Example

As a demonstration of the utility of metalevel axiom schemata, consider a first-order axiomatization for the unique names axioms when the language includes a single

object constant a and a single, unary function constant f.

$$\begin{array}{l} a \neq f(a) \\ a \neq f(f(a)) \\ \vdots \\ f(a) \neq f(f(a)) \\ f(a) \neq f(f(f(a))) \\ \vdots \end{array}$$

Note that the sentence set

$$\begin{array}{l} a \neq f(x) \\ x \neq y \Rightarrow f(x) \neq f(y) \end{array}$$

entails the unique names axioms above, but is stronger than we want.

Let's see how to take advantage of tr to finitely describe this infinite set of axioms. The designator for these sentences can be defined as including the sentence $\phi \neq \psi$ whenever ϕ and ψ are distinct terms in the language L_B.

$$\begin{array}{l} d(relnSent(\text{“}\neq\text{”}, y, z)) \Leftarrow \\ \quad term(y) \wedge term(z) \wedge y \neq z \end{array}$$

The object constant a is a term, as is $f(\phi)$, where ϕ is a term.

$$\begin{array}{l} term(\text{“}a\text{”}) \\ term(funcTerm(\text{“}f\text{”}, y)) \Leftarrow term(y) \end{array}$$

Closing these two definitions and adding in the sentence $d(x) \Rightarrow tr(x)$ gives a full description of the unique names axioms.

$$\begin{array}{l} d(x) \Leftrightarrow \\ \quad \exists yz.(x = relnSent(\text{“}\neq\text{”}, y, z) \wedge \\ \quad\quad term(y) \wedge term(z) \wedge y \neq z) \\ term(x) \Leftrightarrow \\ \quad (x = \text{“}a\text{”} \vee \\ \quad \exists y.(x = funcTerm(\text{“}f\text{”}, y) \wedge term(y))) \\ d(x) \Rightarrow tr(x) \end{array}$$

Conclusion and Future Work

Injecting metalevel axiom schemata into a first order language enhances a logician's ability to concisely axiomatize certain domains. The scheme presented here starts with a particular baselevel language L_B and constructs a metalevel language, L_M, and the infinite axiom set Γ to support L_M. Because Γ depends only on the *language* L_B, building Γ into a proof procedure allows that procedure to manipulate any description of sentences in L_B. That is, with some mild restrictions we can construct a proof system $\vdash_{L_B}$ such that for any description D of L_B-sentences Δ, $D \vdash_{L_B} \phi$ if and only if $\Delta \models \phi$, where ϕ is an L_B-sentence.

In one sense, this implementation is trivial. If the variables and constants of L_B are recursively enumerable, so too is an infinite axiom set with a finite description in L_M. One can interleave this enumeration with a favorite proof procedure and ensure semi-decidability, soundness, and completeness. But such a proof procedure is far from practical. In the sequel, we give a resolution-based procedure that avoids enumerating axiom schemata when possible.

The tr predicate described here was developed for the purposes of knowledge sharing and can be used as a replacement for the truth predicate $true$ described in the KIF (Knowledge Interchange Format) manual (Genesereth, Fikes, & et al. 1992). tr is conceptually simpler than $true$ and, therefore, easier for users to understand. It is also more amenable to implementation.

References

Attardi, G., and Simi, M. 1995. A Formalization of Viewpoints. *Fundamenta Informaticae* 23:149–173.

Chen, W.; Kifer, M.; and Warren, D. 1989. Hilog: A foundation for higher-order logic programming. Technical report, State University of New York at Stony Brook.

DesRivieres, J., and Levesque, H. J. 1965. The consistency of syntactical treatments of knowledge. *Computational Intelligence* 12(1):23–41.

Enderton, H. 2001. *A Mathematical Introduction to Logic*. Hartcourt/Academic Press.

Genesereth, M., and Nilsson, N. 1987. *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann.

Genesereth, M.; Fikes, R.; and et al. 1992. Knowledge interchange format version 3 reference manual. Technical report, Stanford University.

Haas, A. 1986. A syntactic theory of beliefs and knowledge. *Artificial Intelligence* 28(3):245–292.

Perlis, D. 1985. Languages with self-reference, I: Foundations. *Artificial Intelligence* 25(3):301–322.

Sato, T. 1992. Meta-programming through a truth predicate. In *JICSLP*, 526–540.

Tarski, A. 1956. *The Concept of Truth in Formalized Languages*. Clarendon Press. 152–278.

Turner, R. 1991. *Truth and Modality for Knowledge Representation*. The M.I.T. Press.

Weyhrauch, R. 1980. Prolegomena to a theory of mechanized formal reasoning. *Artificial Intelligence* 13(1-2):133–170.