A SIMPLIFIED METHOD FOR ESTABLISHING THE CORRECTNESS OF ARCHITECTURAL REFINEMENTS

R. A. RIEMENSCHNEIDER

ABSTRACT. My colleagues and I developed an approach to proving correctness of architectural refinement hierarchies that depended upon treating architectural specifications as axiomatizations of first-order theories. This paper explores the consequences of an alternative approach to formalizing the content of specifications in logic. A specification is treated as a depiction of a particular relational structure, which is intended to be a mathematical model of the system being specified. As a result, specifications now correspond to much stronger (in fact, complete) theories. Although the criterion for refinement correctness — that the theory corresponding to the higher-level specification can be faithfully interpreted in the theory corresponding to the lower-level specification — remains the same, the technique for proving correctness is quite different: proving that a mapping is a theory interpretation is more complex, though still largely a matter of calculation, but faithfulness is trivially guaranteed. The net result is a substantial simplification of correctness proofs, as a comparison of proofs of a simple refinement pattern illustrates.

1. Two Approachs to Establishing Correctness

In a previous paper [5], my colleagues and I presented an approach to proving correctness of architectural refinement patterns. A correspondence between architectural specifications, such as the high-level compiler specification in Figure 1, and theories in first-order logic was defined in terms of a mapping from specification elements to axioms for the theories.[1] For example, the presence of the dataflow connector that carries objects of type AST (i.e., abstract syntax trees) from the parser component to the analyzer/optimizer component in Figure 1 corresponds to the axioms

$$\mathsf{Channel}(\mathsf{ast\_intermediate})$$
$$\forall \mathsf{x}_0\, [\mathsf{AST}(\mathsf{x}_0) \rightarrow \mathsf{Can\_Carry}(\mathsf{ast\_intermediate}, \mathsf{x}_0)]$$

The theory that corresponds to a specification is simply the set of all consequences of the axioms obtained from the elements of the specification, together with general axioms that constrain the meanings of the component, port, and connector predicates that appear in the specification.

This theory is rather weak, in the sense that it does not determine the truth value of many sentences in the language. For example, it does not contain any explicit

[1]Strictly speaking, the content of the informal dataflow diagram was formalized in a textual specification language. For the purposes of this paper, let us eliminate the middleman and pretend that diagrammatic representations are sufficiently precise to serve as formal architectural specifications.

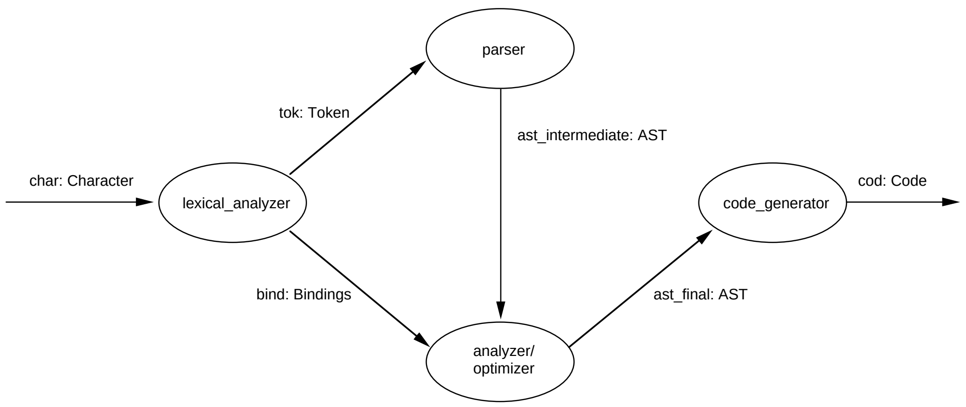

FIGURE 1. Dataflow Architecture for a Compiler

"extremal axioms" to preclude the existence of additional objects not mentioned in the specification. Neither the sentence

$$\neg\, \exists x_0 \exists x_1 \exists x_2\, [\mathsf{Out\_Port}(x_1, \mathsf{parser}) \\ \wedge\ \mathsf{In\_Port}(x_2, \mathsf{lexical\_analyzer}) \wedge \mathsf{Connects}(x_0, x_1, x_2)]$$

which says that there is no dataflow channel from the parser to the lexical analyzer, nor its negation, is a consequence of the theory that corresponds to the architecture in Figure 1. Of course, that no such dataflow channel is shown means that the specifier intended that no such channel be implemented (or, more accurately, that no such channel should be observable at this level of abstraction). Some of these "truth value gaps" can be eliminated by adding axioms, but perhaps not all: the theory of an inductive datatype, such as AST, may well be essentially incomplete. Nonetheless, our theories are treated as if they were complete, in the sense that our criterion for correct refinement is that the higher-level theory must be *faithfully* interpretable in the lower-level theory.[2] In other words, the lower-level theory must not add any detail that could have been expressed at the higher level, such as the existence of additional unmentioned objects in the higher level ontology.

But there is an alternative, perhaps more natural, logical interpretation of the dataflow diagram in Figure 1. Any application of mathematics requires construction of a mathematical model of some phenomena of interest. This mathematical model must share relevant structure with the phenomena, so that conclusions derived from reasoning about the structure of the model are true of the structure of the phenomena being modeled as well. Examples are ubiquitous in computing. For example, one might attempt to derive truths about the behavior of a particular parsing program running on a particular machine by reasoning about a finite

[2]Recall that an interpretation $\mathscr{I}$ of the langauge of theory Θ in the theory Θ' is an interpretation of Θ in Θ' iff for every sentence φ in the language of Θ,

$$\varphi \in \Theta \implies \mathscr{I}(\varphi) \in \Theta'$$

If, in addition, for every sentence φ in the language of Θ,

$$\mathscr{I}(\varphi) \in \Theta' \implies \varphi \in \Theta$$

then $\mathscr{I}$ is said to be *faithful*. See the Appendix of this paper for greater detail.

state automaton that mathematically models it. It seems very natural to suppose that this is exactly the function a system specification, whether behavioral or architectural, is supposed to perform. In short, from this alternative perspective, *architectural specifications are intended to be mathematical models of the systems they specify.*

Just as a finite state automaton, which is formally defined as some sort of mathematical structure, can be depicted as a collection of boxes and arrows, a system architecture diagram can be construed as a depiction of a particular mathematical structure. Consider once again the dataflow diagram of Figure 1. The similarity type of the structure that this diagram depicts is determined by its architectural style. Since this is a dataflow diagram, the structure must provide the extensions for the various predicate parameters of the dataflow language — Channel, AST, Can_Carry, and so on — and must also identify the particular individuals denoted by the individual parameters of the specification, such as parser. (Note that this structure is not finite. Most often, the number of components and connectors in specifications will be finite,[3] but the datatypes are often infinite.)

In the "model-based" treatment of specifications, the logical theory that corresponds to the specification is simply the theory of the structure that the specification depicts. This theory is, of course, complete, which is important because *every standard interpretation*[4] *of a complete theory in a consistent theory is faithful.* The proof of this fact is easy: for any standard interpretation $\mathscr{I}$ of complete theory Θ in consistent theory Θ' and any sentence φ of the language of Θ,

$$\begin{aligned}
\varphi \notin \Theta \quad &\Longrightarrow \quad \neg\varphi \in \Theta && (\Theta \text{ is complete})\\
&\Longrightarrow \quad \mathscr{I}(\neg\varphi) \in \Theta' && (\mathscr{I} \text{ interprets } \Theta \text{ in } \Theta')\\
&\Longrightarrow \quad \neg\mathscr{I}(\varphi) \in \Theta' && (\text{St'd interp'ns preserve logic})\\
&\Longrightarrow \quad \mathscr{I}(\varphi) \notin \Theta' && (\Theta' \text{ is consistent})
\end{aligned}$$

and so, universally generalizing the contrapositive, $\mathscr{I}$ is faithful.

In the "property-oriented" treatment of our previous paper, proving that the standard interpretation associated with a candidate refinement is a theory interpretation was easy. A standard interpretation maps derivations to derivations.[5] Therefore, if finite axiomatizations of Θ and Θ' are available, then it can be shown that $\mathscr{I}$ interprets Θ in Θ' by deriving the image of each axiom of Θ under $\mathscr{I}$ from the axioms of Θ'. But proving faithfulness is harder. A substantial model-theoretic result served as the basis of the method we proposed. So the fact that faithfulness is automatic on the new approach makes it look quite promising. Unfortunately, there is no simple, mechanical way to obtain axioms for the theory of a structure from a representation of that structure. Indeed, the theory of the structure corresponding to some specifications may very well have no recursive, much less finite, axiomatization. Some other way of recognizing when a basis mapping from the parameters of the language of one structure to the language of another structure

[3]However, it might sometimes be convenient to model a conceptually unbounded number of components or connectors by an infinite number of components or connectors, just as a Turing machine's infinite tape is used to model conceptually unbounded memory

[4]See the Appendix for the definition of *standard interpretation*.

[5]Actually, the derivations that result can contain small "gaps" due to the introduction of bounds on quantifiers, but filling these in is trivial.

determines a standard interpretation of the theory of the former struture in the theory of the latter structure is required.

2. A Model-Theoretic Criterion for Theory Interpretation

Any standard interpretation $\mathscr{I}$ of the language $\mathscr{L}$ in an $\mathscr{L}'$-theory Θ' induces a "dual" mapping $\mathscr{I}^\partial$ from $\mathscr{L}'$-structures to $\mathscr{L}$-structures: roughly speaking, the denotation of a predicate of $\mathscr{L}$ is determined by the extension of the $\mathscr{L}'$-formula that interprets the predicate in the $\mathscr{L}'$-structure, and similarly for individual parameters. More formally, given a model $\mathfrak{A}' = \langle |\mathfrak{A}'|, \mathbf{P}^{\mathfrak{A}'}, \dots, \mathbf{a}^{\mathfrak{A}'}, \dots \rangle$ of Θ', the $\mathscr{L}$-structure $\mathscr{I}^\partial(\mathfrak{A}') = \langle |\mathscr{I}^\partial(\mathfrak{A}')|, \mathbf{P}^{\mathscr{I}^\partial(\mathfrak{A}')}, \dots, \mathbf{a}^{\mathscr{I}^\partial(\mathfrak{A}')}, \dots \rangle$ is determined as follows:

(1) let
$$|\mathscr{I}^\partial(\mathfrak{A}')| = \{x_0 \in |\mathfrak{A}'| : \mathfrak{A}' \vDash \omega_{\mathscr{I}}[x_0]\}$$
where $\omega_{\mathscr{I}}$, the $\mathscr{L}'$-formula that bounds interpreted quantifiers, defines the subset of $|\mathfrak{A}'|$ used to interpret $|\mathfrak{A}|$,

(2) for every n-ary predicate parameter $\mathbf{P}$ of $\mathscr{L}$, let
$$\mathbf{P}^{\mathscr{I}^\partial(\mathfrak{A}')} = \{\langle x_0, x_1, \dots, x_{n-1}\rangle \in |\mathscr{I}^\partial(\mathfrak{A}')|^n : \\ \mathfrak{A}' \vDash \mathscr{I}(\mathbf{P}(\mathsf{x}_0, \mathsf{x}_1, \dots, \mathsf{x}_{n-1}))\,[x_0, x_1, \dots, x_{n-1}]\}$$
and,

(3) for every individual parameter $\mathbf{a}$ of $\mathscr{L}$, let $\mathbf{a}^{\mathscr{I}^\partial(\mathfrak{A}')}$ be the x_0 in $|\mathfrak{A}'|$ such that
$$\mathfrak{A}' \vDash \mathscr{I}(\mathsf{x}_0 \equiv \mathbf{a})\,[x_0]$$

It is easy to show that if $\mathscr{I}$ is a standard interpretation of the language of the structure $\mathfrak{A}$ in the theory of the structure $\mathfrak{A}'$ and $\mathscr{I}^\partial(\mathfrak{A}')$ is isomorphic to $\mathfrak{A}$, then $\mathscr{I}$ interprets the theory of $\mathfrak{A}$ in the theory of $\mathfrak{A}'$.[6] Let f be an isomophism from $\mathfrak{A}$

[6]Recall that the theorem that provided the basis for establishing faithfulness of standard first-order theory interpretations on our original approach was

> Interpretation $\mathscr{I}$ of the theory Θ in the theory Θ' is faithful iff, for every model $\mathfrak{A}$ of Θ, there is a model $\mathfrak{A}'$ of Θ' such that $\mathscr{I}^\partial(\mathfrak{A}')$ can be expanded to a model of the description of $\mathfrak{A}$.

or, equivalently,

> Interpretation $\mathscr{I}$ of the theory Θ in the theory Θ' is faithful iff, for every model $\mathfrak{A}$ of Θ, there is a model $\mathfrak{A}'$ of Θ' and a function from $|\mathfrak{A}|$ into $|\mathfrak{A}'|$ such that
> $$(\mathfrak{A}, a)_{a\in|\mathfrak{A}|} \equiv (\mathscr{I}^\partial(\mathfrak{A}'), f(a))_{a\in|\mathfrak{A}|}$$
> where '$\equiv$' denotes elementary equivalence.

It follows that requiring $\mathscr{I}^\partial(\mathfrak{A}')$ to be isomorphic to $\mathfrak{A}$ is not necessary for faithfulness. But this is simply an artifact of the limited expressive power of first-order logic; in ω-order logic, also known as *higher-order logic* and *the theory of types*, isomorphism is both necessary and sufficient for faithfulness. On the approach advocated in this paper, formal logical derivations have been eliminated — derivations are no longer used in proving correctness, and the question of whether a formula can be derived from the theory corresponding to a specification has been replaced by the question of whether the structure corrseponding to that specification is a model of that formula — and so there is no compelling meta-theoretical reason to restrict ourselves to first-order logic. It would even be tempting to replace the notion of interpreting one *theory* in another by that

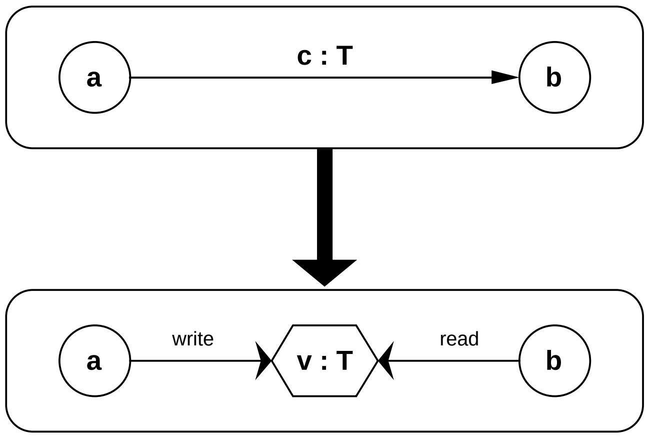

FIGURE 2. Refinement Pattern: Dataflow Channel to Variable

to $\mathscr{I}^{\partial}(\mathfrak{A}')$. Then, for every $\mathscr{L}$-formula φ and every assignment x of values in $|\mathfrak{A}|$ to the free variables of φ,

$$\begin{aligned}\mathfrak{A} \models \varphi\,[x] \quad &\Longleftrightarrow \quad \mathscr{I}^{\partial}(\mathfrak{A}') \models \varphi\,[f \circ x] \\ &\qquad\qquad \text{(Definition of } \textit{isomorphism}\text{)} \\ &\Longleftrightarrow \quad \mathfrak{A}' \models \mathscr{I}(\varphi)\,[f \circ x] \\ &\qquad\qquad \text{(Theorem of Appendix A)}\end{aligned}$$

A fortori, $\mathscr{I}$ is a theory interpretation.

This theorem suggests that $\mathscr{I}^{\partial}$ be viewed as an *abstraction mapping* that corresponds to standard interpretation $\mathscr{I}$. It says that structure $\mathfrak{A}'$ is a correct refinement of structure $\mathfrak{A}$, relative to an interpretation $\mathscr{I}$ of the language of $\mathfrak{A}$ in the language of $\mathfrak{A}'$, if $\mathscr{I}^{\partial}(\mathfrak{A}')$ — the result of abstracting away the features of $\mathfrak{A}'$ not expressible in the language of $\mathfrak{A}$ — is $\mathfrak{A}$ (up to isomorphism).

3. An Example

To enable direct comparison with the correctness proof sketch in our previous paper, the same refinement pattern, replacement of a single dataflow channel by writing and reading of a variable, will be proven correct. This pattern is presented graphically in Figure 2.

A refinement pattern consists of a pair of *schematic diagrams*, diagrams in which some terms have been replaced by syntactic variables. The syntactic variables of this pattern are **a**, **b**, **c**, **v**, and **T**. A pair of structures $\langle \mathfrak{D}, \mathfrak{M} \rangle$ *matches* this pattern iff, for some assignment of values to the syntactic variables, the diagram above the double arrow depicts $\mathfrak{D}$ and the diagram below the arrow depicts $\mathfrak{M}$. The pattern

of interpreting one *structure* in another ([4], pp. 212), except we exploit intuitions about what a specification says to motivate the necessity of faithful interpretation. By focussing on *isomorphism* rather than *finite isomorphism*, the usual algebraic characterization of *elementary equivalence* [2], the dependence of this approach on a particular choice of logic is largely eliminated. However, the account below will be restricted to first-order theories of structures, to enable more direct comparison of the new approach and the original.

is *correct* iff, for every pair of structures $\langle \mathfrak{D}, \mathfrak{M} \rangle$ that matches the pattern, $\mathfrak{M}$ is a correct refinement of $\mathfrak{D}$. So, if a pair of structures matches a correct refinement pattern, the second is a correct refinement of the first.

The class of structures depicted by the schematic diagram above the arrow can be more formally defined as follows:

$$\begin{aligned}
|\mathfrak{D}| &= \{a, b, c, p_o, p_i\} \cup T \\
\mathsf{Procedure}^{\mathfrak{D}} &= \{a, b\} \\
\mathsf{Channel}^{\mathfrak{D}} &= \{c\} \\
\mathsf{In\_Port}^{\mathfrak{D}} &= \{\langle p_i, b\rangle\} \\
\mathsf{Out\_Port}^{\mathfrak{D}} &= \{\langle p_o, a\rangle\} \\
\mathsf{Can\_Carry}^{\mathfrak{D}} &= \{\langle c, t\rangle : t \in T\} \\
\mathsf{Might\_Supply}^{\mathfrak{D}} &= \{\langle p_o, t\rangle : t \in T\} \\
\mathsf{Can\_Accept}^{\mathfrak{D}} &= \{\langle p_i, t\rangle : t \in T\} \\
\mathsf{Connects}^{\mathfrak{D}} &= \{\langle c, p_o, p_i\rangle\} \\
\mathbf{T}^{\mathfrak{D}} &= T \\
\mathbf{a}^{\mathfrak{D}} &= a \\
\mathbf{b}^{\mathfrak{D}} &= b \\
\mathbf{c}^{\mathfrak{D}} &= c \\
\mathbf{a}\mathsf{\_oport}^{\mathfrak{D}} &= p_o \\
\mathbf{b}\mathsf{\_iport}^{\mathfrak{D}} &= p_i
\end{aligned}$$

where a, b, c, p_o, and p_i are some five distinct objects, none of which is a member of the set T.[7] Similarly, the class of structures depicted by the schematic diagram that gives the implementation of the dataflow in terms of shared memory is defined as follows:

$$\begin{aligned}
|\mathfrak{M}| &= \{a, b, v, k_w, k_r\} \cup T \\
\mathsf{Procedure}^{\mathfrak{M}} &= \{a, b\} \\
\mathsf{Variable}^{\mathfrak{M}} &= \{v\} \\
\mathsf{Call}^{\mathfrak{M}} &= \{\langle k_w, a\rangle, \langle k_r, b\rangle\} \\
\mathsf{Can\_Hold}^{\mathfrak{M}} &= \{\langle v, t\rangle : t \in T\} \\
\mathsf{Might\_Put}^{\mathfrak{M}} &= \{\langle k_w, t\rangle : t \in T\} \\
\mathsf{Can\_Get}^{\mathfrak{M}} &= \{\langle k_r, t\rangle : t \in T\}
\end{aligned}$$

[7]The names and types of the ports are not explicitly given by the graphical specification; the names are generated and the types are inferred, according to the principle that the type of a port is the type of the channel connected to it unless there is some explict indication to the contrary.

$$\begin{aligned}
\mathsf{Writes}^{\mathfrak{M}} &= \{\langle k_w, v\rangle\} \\
\mathsf{Reads}^{\mathfrak{M}} &= \{\langle k_r, v\rangle\} \\
\mathbf{T}^{\mathfrak{M}} &= T \\
\mathbf{a}^{\mathfrak{M}} &= a \\
\mathbf{b}^{\mathfrak{M}} &= b \\
\mathbf{v}^{\mathfrak{M}} &= v \\
\mathbf{a}\_\mathsf{call}^{\mathfrak{M}} &= k_w \\
\mathbf{b}\_\mathsf{call}^{\mathfrak{M}} &= k_r
\end{aligned}$$

where a, b, v, k_w, and k_r are some five distinct objects, none of which is a member of the set T. Here, the names of the calls have been generated and the signatures of the calls inferred from types and sorts of calls.

Consider the basis mapping $\mathscr{K}$ from the parameters of the language of $\mathfrak{D}$ to formulas in the language of $\mathfrak{M}$ defined as follows:

$$\begin{aligned}
\omega_{\mathscr{K}} &= \mathsf{x}_0 \equiv \mathsf{x}_0 \\
\mathscr{K}(\mathsf{Procedure}) &= \mathsf{Procedure}(\mathsf{x}_0) \\
\mathscr{K}(\mathsf{Channel}) &= \mathsf{Variable}(\mathsf{x}_0) \\
\mathscr{K}(\mathsf{In\_Port}) &= \mathsf{Call}(\mathsf{x}_0, \mathsf{x}_1) \wedge \exists \mathsf{x}_2\, \mathsf{Can\_Get}(\mathsf{x}_0, \mathsf{x}_2) \\
\mathscr{K}(\mathsf{Out\_Port}) &= \mathsf{Call}(\mathsf{x}_0, \mathsf{x}_1) \wedge \exists \mathsf{x}_2\, \mathsf{Might\_Put}(\mathsf{x}_0, \mathsf{x}_2) \\
\mathscr{K}(\mathsf{Can\_Carry}) &= \mathsf{Can\_Hold}(\mathsf{x}_0, \mathsf{x}_1) \\
\mathscr{K}(\mathsf{Might\_Supply}) &= \mathsf{Might\_Put}(\mathsf{x}_0, \mathsf{x}_1) \\
\mathscr{K}(\mathsf{Can\_Accept}) &= \mathsf{Can\_Get}(\mathsf{x}_0, \mathsf{x}_1) \\
\mathscr{K}(\mathsf{Connects}() &= \mathsf{Writes}(\mathsf{x}_1, \mathsf{x}_0) \wedge \mathsf{Reads}(\mathsf{x}_2, \mathsf{x}_0) \\
\mathscr{K}(\mathbf{T}) &= \mathbf{T}(\mathsf{x}_0) \\
\mathscr{K}(\mathbf{a}) &= \mathsf{x}_0 \equiv \mathbf{a} \\
\mathscr{K}(\mathbf{b}) &= \mathsf{x}_0 \equiv \mathbf{b} \\
\mathscr{K}(\mathbf{c}) &= \mathsf{x}_0 \equiv \mathbf{v} \\
\mathscr{K}(\mathbf{a}\_\mathsf{oport}) &= \mathsf{x}_0 \equiv \mathbf{a}\_\mathsf{call} \\
\mathscr{K}(\mathbf{b}\_\mathsf{iport}) &= \mathsf{x}_0 \equiv \mathbf{b}\_\mathsf{call}
\end{aligned}$$

The first eight clauses in the definition are determined by the general *style mapping* for interpreting dataflow style in shared memory style. The last six clauses are determined by the *identifier mapping* associated with this particular refinement step.[8] If this basis mapping is extended to an interpretation of formulas in the language of $\mathfrak{D}$ in the standard way,[9] an interpretation of the language of $\mathfrak{D}$ in the theory of $\mathfrak{M}$ results.

[8]See our previous paper [5] for a more detailed account of style and identifier mappings.

[9]See Appendix.

Now the value of the abstraction map $\mathscr{K}^{\partial}$ at $\mathfrak{M}$ can be calculated, using its definition.[10]

$$\begin{aligned}
|\mathscr{K}^{\partial}(\mathfrak{M})| &= \{x_0 \in |\mathfrak{M}| : \mathfrak{M} \vDash \omega_{\mathscr{K}}[x_0]\} \\
&= \{x_0 \in |\mathfrak{M}| : \mathfrak{M} \vDash \mathsf{x}_0 \equiv \mathsf{x}_0\ [x_0]\} \\
&= |\mathfrak{M}|
\end{aligned}$$

$$\begin{aligned}
\mathsf{Procedure}^{\mathscr{K}^{\partial}(\mathfrak{M})} &= \{x_0 \in |\mathfrak{M}| : \mathfrak{M} \vDash \mathscr{K}(\mathsf{Procedure})\ [x_0]\} \\
&= \{x_0 \in |\mathfrak{M}| : \mathfrak{M} \vDash \mathsf{Procedure}(\mathsf{x}_0)\ [x_0]\} \\
&= \{a, b\}
\end{aligned}$$

$$\begin{aligned}
\mathsf{Channel}^{\mathscr{K}^{\partial}(\mathfrak{M})} &= \{x_0 \in |\mathfrak{M}| : \mathfrak{M} \vDash \mathscr{K}(\mathsf{Channel})\ [x_0]\} \\
&= \{x_0 \in |\mathfrak{M}| : \mathfrak{M} \vDash \mathsf{Variable}(\mathsf{x}_0)\ [x_0]\} \\
&= \{v\}
\end{aligned}$$

$$\begin{aligned}
\mathsf{In\_Port}^{\mathscr{K}^{\partial}(\mathfrak{M})} &= \{\langle x_0, x_1\rangle \in |\mathfrak{M}|^2 : \mathfrak{M} \vDash \mathscr{K}(\mathsf{In\_Port})\ [x_0, x_1]\} \\
&= \{\langle x_0, x_1\rangle \in |\mathfrak{M}|^2 : \\
&\qquad \mathfrak{M} \vDash \mathsf{Call}(\mathsf{x}_0, \mathsf{x}_1) \wedge \exists \mathsf{x}_2\, \mathsf{Can\_Get}(\mathsf{x}_0, \mathsf{x}_2)\ [x_0, x_1]\} \\
&= \{\langle k_w, a\rangle\}
\end{aligned}$$

$$\begin{aligned}
\mathsf{Out\_Port}^{\mathscr{K}^{\partial}(\mathfrak{M})} &= \{\langle x_0, x_1\rangle \in |\mathfrak{M}|^2 : \mathfrak{M} \vDash \mathscr{K}(\mathsf{Out\_Port})\ [x_0, x_1]\} \\
&= \{\langle x_0, x_1\rangle \in |\mathfrak{M}|^2 : \\
&\qquad \mathfrak{M} \vDash \mathsf{Call}(\mathsf{x}_0, \mathsf{x}_1) \wedge \exists \mathsf{x}_2\, \mathsf{Might\_Put}(\mathsf{x}_0, \mathsf{x}_2)\ [x_0, x_1]\} \\
&= \{\langle k_r, b\rangle\}
\end{aligned}$$

$$\begin{aligned}
\mathsf{Can\_Carry}^{\mathscr{K}^{\partial}(\mathfrak{M})} &= \{\langle x_0, x_1\rangle \in |\mathfrak{M}|^2 : \mathfrak{M} \vDash \mathscr{K}(\mathsf{Can\_Carry})\ [x_0, x_1]\} \\
&= \{\langle x_0, x_1\rangle \in |\mathfrak{M}|^2 : \mathfrak{M} \vDash \mathsf{Can\_Hold}(\mathsf{x}_0, \mathsf{x}_1)\ [x_0, x_1]\} \\
&= \{\langle c, t\rangle : t \in T\}
\end{aligned}$$

$$\begin{aligned}
\mathsf{Might\_Supply}^{\mathscr{K}^{\partial}(\mathfrak{M})} &= \{\langle x_0, x_1\rangle \in |\mathfrak{M}|^2 : \mathfrak{M} \vDash \mathscr{K}(\mathsf{Might\_Supply})\ [x_0, x_1]\} \\
&= \{\langle x_0, x_1\rangle \in |\mathfrak{M}|^2 : \mathfrak{M} \vDash \mathsf{Might\_Put}(\mathsf{x}_0, \mathsf{x}_1)\ [x_0, x_1]\} \\
&= \{\langle k_w, t\rangle : t \in T\}
\end{aligned}$$

$$\begin{aligned}
\mathsf{Can\_Accept}^{\mathscr{K}^{\partial}(\mathfrak{M})} &= \{\langle x_0, x_1\rangle \in |\mathfrak{M}|^2 : \mathfrak{M} \vDash \mathscr{K}(\mathsf{Can\_Accept})\ [x_0, x_1]\} \\
&= \{\langle x_0, x_1\rangle \in |\mathfrak{M}|^2 : \mathfrak{M} \vDash \mathsf{Can\_Get}(\mathsf{x}_0, \mathsf{x}_1)\ [x_0, x_1]\} \\
&= \{\langle k_r, t\rangle : t \in T\}
\end{aligned}$$

[10]Every parameter of the language of $\mathfrak{D}$ is treated below, for the sake of completeness, but all calculations are similar and there is no need to read them all unless you are so inclined.

$$\begin{aligned}
\mathsf{Connects}^{\mathscr{K}^{\partial}(\mathfrak{M})} &= \{\langle x_0, x_1, x_2\rangle \in |\mathfrak{M}|^3 : \mathfrak{M} \vDash \mathscr{K}(\mathsf{Connects})\,[x_0, x_1, x_2]\} \\
&= \{\langle x_0, x_1, x_2\rangle \in |\mathfrak{M}|^3 : \\
&\qquad \mathfrak{M} \vDash \mathsf{Writes}(\mathsf{x}_1, \mathsf{x}_0) \wedge \mathsf{Reads}(\mathsf{x}_2, \mathsf{x}_0)\,[x_0, x_1, x_2]\} \\
&= \{\langle c, k_w, k_r\rangle\}
\end{aligned}$$

$$\begin{aligned}
\mathbf{T}^{\mathscr{K}^{\partial}(\mathfrak{M})} &= \{x_0 \in |\mathfrak{M}| : \mathfrak{M} \vDash \mathscr{K}(\mathbf{T})\,[x_0]\} \\
&= \{x_0 \in |\mathfrak{M}| : \mathfrak{M} \vDash \mathbf{T}(\mathsf{x}_0)\,[x_0]\} \\
&= T
\end{aligned}$$

$$\begin{aligned}
\mathbf{a}^{\mathscr{K}^{\partial}(\mathfrak{M})} &= (\imath x_0 \in |\mathfrak{M}|)\, \mathfrak{M} \vDash \mathscr{K}(\mathsf{a})\,[x_0] \\
&= (\imath x_0 \in |\mathfrak{M}|)\, \mathfrak{M} \vDash \mathsf{x}_0 \equiv \mathsf{a}\,[x_0] \\
&= a
\end{aligned}$$

$$\begin{aligned}
\mathbf{b}^{\mathscr{K}^{\partial}(\mathfrak{M})} &= (\imath x_0 \in |\mathfrak{M}|)\, \mathfrak{M} \vDash \mathscr{K}(\mathsf{b})\,[x_0] \\
&= (\imath x_0 \in |\mathfrak{M}|)\, \mathfrak{M} \vDash \mathsf{x}_0 \equiv \mathsf{b}\,[x_0] \\
&= b
\end{aligned}$$

$$\begin{aligned}
\mathbf{c}^{\mathscr{K}^{\partial}(\mathfrak{M})} &= (\imath x_0 \in |\mathfrak{M}|)\, \mathfrak{M} \vDash \mathscr{K}(\mathsf{c})\,[x_0] \\
&= (\imath x_0 \in |\mathfrak{M}|)\, \mathfrak{M} \vDash \mathsf{x}_0 \equiv \mathsf{v}\,[x_0] \\
&= v
\end{aligned}$$

$$\begin{aligned}
\mathbf{a}\_\mathsf{oport}^{\mathscr{K}^{\partial}(\mathfrak{M})} &= (\imath x_0 \in |\mathfrak{M}|)\, \mathfrak{M} \vDash \mathscr{K}(\mathbf{a}\_\mathsf{oport})\,[x_0] \\
&= (\imath x_0 \in |\mathfrak{M}|)\, \mathfrak{M} \vDash \mathsf{x}_0 \equiv \mathbf{a}\_\mathsf{call}\,[x_0] \\
&= k_w
\end{aligned}$$

$$\begin{aligned}
\mathbf{b}\_\mathsf{iport}^{\mathscr{K}^{\partial}(\mathfrak{M})} &= (\imath x_0 \in |\mathfrak{M}|)\, \mathfrak{M} \vDash \mathscr{K}(\mathbf{b}\_\mathsf{iport})\,[x_0] \\
&= (\imath x_0 \in |\mathfrak{M}|)\, \mathfrak{M} \vDash \mathsf{x}_0 \equiv \mathbf{b}\_\mathsf{call}\,[x_0] \\
&= k_r
\end{aligned}$$

Clearly, if h from $|\mathfrak{D}|$ to $|\mathfrak{M}|$ is defined by

$$\begin{aligned}
h(a) &= a \\
h(b) &= b \\
h(c) &= v \\
h(p_o) &= k_w \\
h(p_i) &= k_r \\
h(t) &= t \qquad \text{(for every } t \in T)
\end{aligned}$$

then h is an isomorphism from $\mathfrak{D}$ to $\mathscr{K}^{\partial}(\mathfrak{M})$, and so $\mathscr{K}$ is indeed an interpretation of the theory of $\mathfrak{D}$ in the theory of $\mathfrak{M}$.

Compare this proof and the correctness proof sketch in our previous paper. In the latter, we had to show that, for any model of the dataflow theory, there was a model of the shared memory theory with a certain desirable property, viz., that the image of the shared memory model under the abstraction mapping is indistinguishable from the dataflow model using the resources of the relevant logic: see Figure 3. Thus, a mapping from dataflow models to shared memory models — obtained by "inverting" the equations that define the interpretation to obtain something like an inverse interpretation whose dual maps dataflow structures to shared memory structures — had to be introduced. By reducing the class of structures that correspond to a specification to a singleton, the necessity of finding an appropriate mapping from dataflow structures to shared memory structures is eliminated: see Figure 4. As a result, the correctness proof is reduced to straightforward calculation, requiring no creativity on the part of the prover. Correctness proof construction is, therefore, much simpler and much more straightforward when using the new approach in place of the old.

4. Conclusions

The essential difference between the two techniques stems from regarding architectural specifications as mathematical models of the systems being specified, rather than comparatively weak descriptions of those systems. Since our justification for demanding that higher-level specifications be *faithfully* interpretable in lower-level specifications was that an architectural specification should say everything that can truly be said of the system's architecture at a given level of abstraction, it is natural to demand that the logical analogues of these specification be *complete* theories. So, this formalization of the connection between architectural specifications and logic is arguably more natural for our purposes. This is one reason for regarding the approach of this paper to be superior. Another, more practical and important, reason is that correctness proofs become much simpler, as the example shows.

It should be noted that this is a relatively minor modification the original approach: specifications are stronger, proofs of correctness are simpler, but the basic idea of proving particular refinement steps correct by using a library of pre-verified refinement rules remain the same. Ordinary users of our architecture specification tools see no difference between the two approaches — all the refinement patterns that they are used to using are still valid — but the burden for those who wish to extend the system by validating new refinement patterns has been considerably lightened.

It should also be noted that this new approach cannot be directly applied to all conceivable refinement patterns. Some architectural descriptions cannot naturally be thought of as specifying a single architectural structure. For example, there are standard architectures, such as the X/Open Distributed Transaction Processing (DTP) architecture [9] that specify how various types of components are composed, but do not specify the number of components of each type. In such cases, it is much more natural to think of the standard as specifying a class of structures. However the new approach can be used to verify a set of simple, primitive refinement patterns that generate the refinements in a multi-level formal representation of X/Open DTP in the SADL language [6], as well as all the refinement patterns of our previous paper [5].

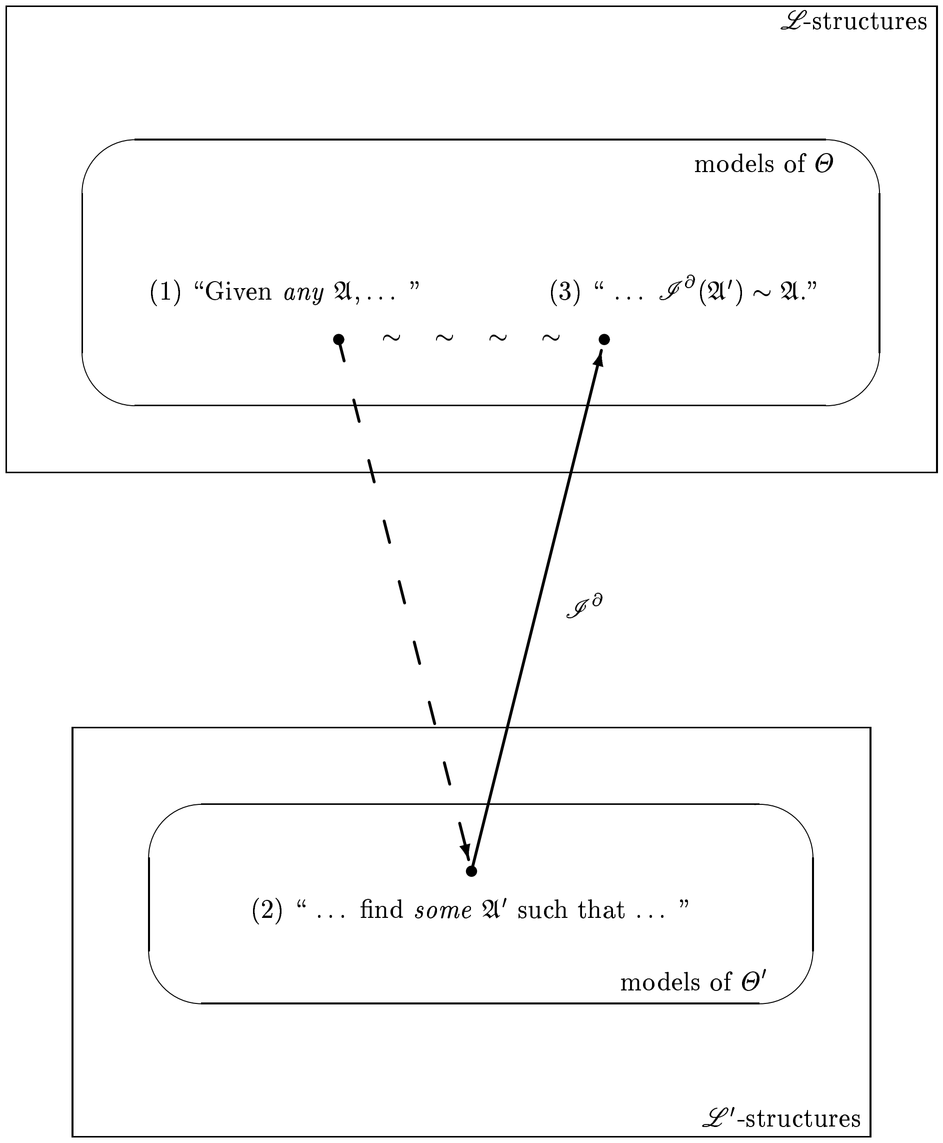
$\mathscr{L}$-structures
models of Θ
(1) "Given *any* $\mathfrak{A}$, ... "
(3) " ... $\mathscr{I}^{\partial}(\mathfrak{A}') \sim \mathfrak{A}$."
$\mathscr{I}^{\partial}$
(2) " ... find *some* $\mathfrak{A}'$ such that ... "
models of Θ'
$\mathscr{L}'$-structures

FIGURE 3. Original proof technique for faithfulness

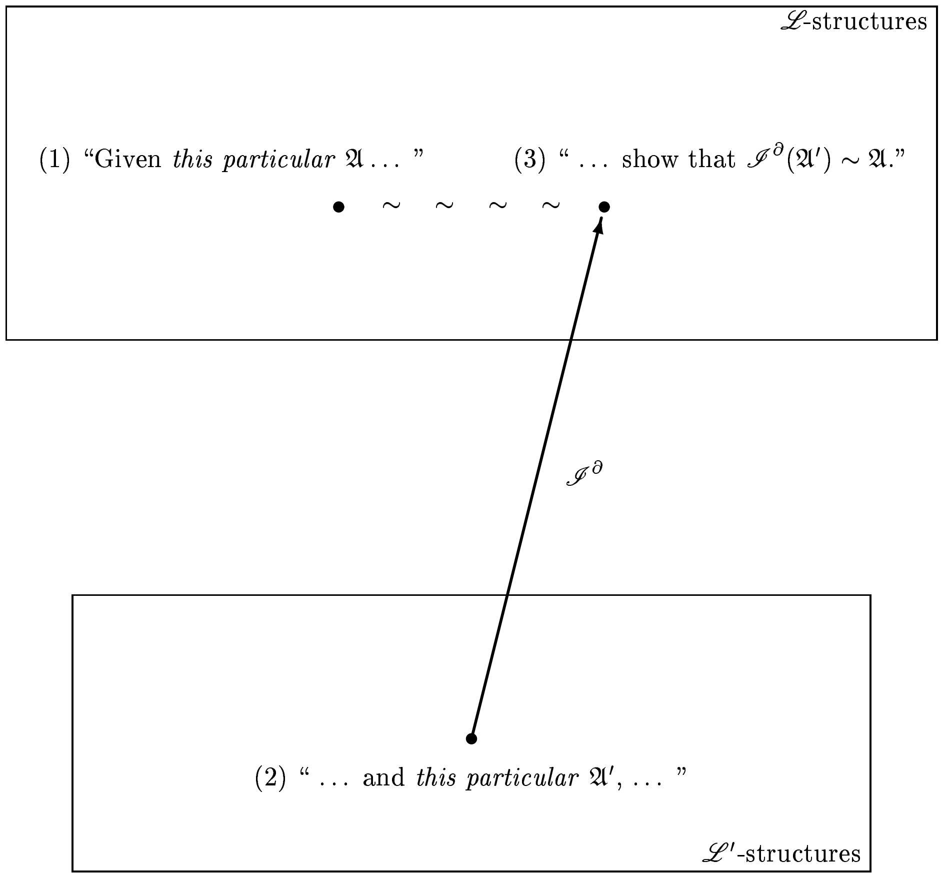
$\mathscr{L}$-structures
(1) "Given *this particular* $\mathfrak{A}$ ... "
(3) " ... show that $\mathscr{I}^{\partial}(\mathfrak{A}') \sim \mathfrak{A}$."
$\mathscr{I}^{\partial}$
(2) " ... and *this particular* $\mathfrak{A}'$, ... "
$\mathscr{L}'$-structures

FIGURE 4. New proof technique for theory interpretation

References

1. J. Baldwin, *Definable second-order quantifiers*, **Model-theoretic logics**, J. Barwise and S. Feferman (eds.), Springer, 1985.
2. H.-D. Ebbinghaus, J. Flum, and W. Thomas, **Mathematical logic**, Springer, second edition, 1994.
3. H. B. Enderton, **A mathematical introduction to logic**, Academic Press, 1972.
4. W. Hodges, **Model theory**, Cambridge, 1993.
5. M. Moriconi, X. Quan, and R. A. Riemenschneider, *Correct architecture refinement*, **IEEE transactions on software engineering**, vol. 21 (1995), pp. 356–372. (Available on the web at URL `<http://www.csl.sri.com/sadl/sadl-intro.ps.gz>`.)
6. M. Moriconi and R. A. Riemenschneider, **Introduction to SADL 1.0**, SRI Computer Science Laboratory Technical Report SRI-CSL-97-01, March 1997. (Available on the web at URL `<http://www.csl.sri.com/sadl/tse95.ps.gz>`.)
7. R. A. Riemenschneider, **Correct transformation rules for incremental development of architecture hierarchies**, SRI Computer Science Laboratory Technical Report SRI-CSL-97-??, ?? 1997. (Available on the web at URL `<http://www.csl.sri.com/sadl/incremental.ps.gz>`.)
8. J. R. Shoenfield, **Mathematical logic**, Addison-Wesley, 1967.
9. X/Open Company, Ltd., **Distributed transaction processing: reference model, version 2**, X/Open, November 1993.

Appendix A. Brief Introduction to Interpretations

Two different sorts of definition of the term *interpretation* appear in the literature. Logic textbooks [3, 4, 8] provide rather concrete definitions, explaining in some detail how mappings from the symbols of one theory to the expressions of another can be extended to mappings from sentences to sentences and exploring the properties of those mappings. For more advanced purposes [1, pp. 470–471], more abstract definitions are often preferred. The main advantage of a more abstract approach is flexibility. Many more mappings can count as interpretations if an abstract approach is adoped. But flexibility has a corresponding cost. Most importantly, if an interpretation is not defined inductively on formulas, then inductive proofs of properties of the interpretation become much more complicated.

For our applications, considerable flexibility is occasionally required [7]. Therefore, our definition is maximally abstract: an *interpretation* $\mathscr{I}$ of the theory Θ in the theory Θ' is a (total) mapping from the sentences of the language $\mathscr{L}$ of Θ to the sentences of the language $\mathscr{L}'$ of Θ' such that, for every $\mathscr{L}$-sentence φ,

$$\varphi \in \Theta \quad \Longrightarrow \quad \mathscr{I}(\varphi) \in \Theta'$$

While it is desirable that interpretations preserve meaning, in some sense, there is no formal requirement that they do so. However, an informal argument that, for every $\mathscr{L}$-sentence φ, $\mathscr{I}(\varphi)$ is semantically stronger than φ may be offered as evidence that the interpretation is "natural". An interpretation $\mathscr{I}$ of Θ in Θ' is *faithful* when, for every $\mathscr{L}$-sentence φ,

$$\mathscr{I}(\varphi) \in \Theta' \quad \Longrightarrow \quad \varphi \in \Theta$$

In many cases, interpretations will be defined in a way that allows straightforward inductive proofs to be performed. Interpretations of theory Θ in theory Θ' will often be defined by giving a *basis mapping* $\mathscr{I}$ from the parameters $\{\square, \mathbf{P}, \dots, \mathbf{a}, \dots\}$[11] of $\mathscr{L}$, the language of Θ, to formulas of $\mathscr{L}'$, the language of Θ', that satisfies the following three conditions:

(*i*) x_0 is the only free variable of $\mathscr{I}(\square)$ — which will generally be called $\omega_{\mathscr{I}}$ rather than $\mathscr{I}(\square)$ — and
$$\Theta' \vDash \exists \mathsf{x}_0\, \omega_{\mathscr{I}}$$
(*ii*) for every n-ary predicate $\mathbf{P}$ of $\mathscr{L}$, the free variables of $\mathscr{I}(\mathbf{P})$ are $\mathsf{x}_0, \mathsf{x}_1, \dots, \mathsf{x}_{n-1}$, and
(*iii*) for every name $\mathbf{a}$ of $\mathscr{L}$, x_0 is the only free variable of $\mathscr{I}(\mathbf{a})$ and
$$\Theta' \vDash \exists \mathsf{x}_1\, \forall \mathsf{x}_0\, [\, \mathscr{I}(\mathbf{a}) \leftrightarrow \mathsf{x}_0 \equiv \mathsf{x}_1].$$

These conditions can be restated as

(*i*) the formula $\omega_{\mathscr{I}}$ defines a non-empty set,
(*ii*) for every n-ary predicate $\mathbf{P}$ of $\mathscr{L}$, the formula $\mathscr{I}(\mathbf{P})$ defines an n-ary relation on the set defined by $\omega_{\mathscr{I}}$, and

[11] $\square$ is a special parameter of the language $\mathscr{L}$, the *universe parameter*, that does not actually occur in $\mathscr{L}$-formulas. This allows us to treat a structure as a mapping from the parameters of $\mathscr{L}$ to appropriate denotations — so $|\mathfrak{A}|$ is simply $\square^{\mathfrak{A}}$ — and similarly allows us to treat the basis of an interpretation as a mapping from the parameters of $\mathscr{L}$ to appropriate $\mathscr{L}'$-formulas.

(*iii*) for every name a of $\mathscr{L}$, the formula $\mathscr{I}(\mathsf{a})$ defines a member of the set defined by $\omega_{\mathscr{I}}$,

which emphasizes the fact that they are syntactic analogues of the three defining conditions for an $\mathscr{L}$-structure $\mathfrak{A}$,

(*i′*) the universe $|\mathfrak{A}|$ of $\mathfrak{A}$ is non-empty,
(*ii′*) for every n-ary predicate P of $\mathscr{L}$, $\mathsf{P}^{\mathfrak{A}}$ is an n-ary relation on $|\mathfrak{A}|$, and
(*iii′*) for every name a of $\mathscr{L}$, $\mathsf{a}^{\mathfrak{A}}$ is a member of $|\mathfrak{A}|$.

The mapping $\mathscr{I}$ can be extended to a map from $\mathscr{L}$-formulas to $\mathscr{L}'$-formulas in a straightforward fashion. If φ is an atomic $\mathscr{L}$-formula — say, $\mathsf{P}(\mathsf{a},\mathsf{x})$, where P is a binary predicate, a is a name, and x is a variable — then $\mathscr{I}(\varphi)$ is

$$\exists\mathsf{y}\,[\big(\mathscr{I}(\mathsf{a})\big)(\mathsf{x}_0/\mathsf{y}) \wedge \big(\mathscr{I}(\mathsf{P})\big)(\mathsf{x}_0/\mathsf{y},\mathsf{x}_1/\mathsf{x})]$$

where y is a fresh variable. If $\mathscr{I}(\mathsf{a})$ is an equation $\mathsf{x}_0 \equiv \mathsf{a}'$, $\mathscr{I}(\mathsf{P}(\mathsf{a},\mathsf{x}))$ will be simplified to

$$\big(\mathscr{I}(\mathsf{P})\big)(\mathsf{x}_0/\mathsf{a}',\mathsf{x}_1/\mathsf{x})$$

(Note that $\mathscr{I}(\mathsf{P}(\mathsf{x}_0,\mathsf{x}_1,\dots,\mathsf{x}_{n-1}))$ is simply $\mathscr{I}(\mathsf{P})$ and that $\mathscr{I}(\mathsf{x}_0 \equiv \mathsf{a})$ is simply $\mathscr{I}(\mathsf{a})$.) Now that $\mathscr{I}$ has been defined on all atomic formulas, it can be extended to all formulas as follows:

(1) for any $\mathscr{L}$-formula φ,
$$\mathscr{I}(\neg\varphi) = \neg\,\mathscr{I}(\varphi)$$
(2) for any $\mathscr{L}$-formulas φ and ψ,
$$\mathscr{I}(\varphi \wedge \psi) = \mathscr{I}(\varphi) \wedge \mathscr{I}(\psi)$$
(3) for any $\mathscr{L}$-formulas φ and ψ,
$$\mathscr{I}(\varphi \vee \psi) = \mathscr{I}(\varphi) \vee \mathscr{I}(\psi)$$
(4) for any $\mathscr{L}$-formulas φ and ψ,
$$\mathscr{I}(\varphi \rightarrow \psi) = \mathscr{I}(\varphi) \rightarrow \mathscr{I}(\psi)$$
(5) there is an $\mathscr{L}'$-formula $\omega_{\mathscr{I}}$, called the *universe* of the interpretation, such that for any variable x and any $\mathscr{L}$-formula φ,
$$\mathscr{I}(\forall\mathsf{x}\,\varphi) = \forall\mathsf{x}\,[\omega_{\mathscr{I}}(\mathsf{x}_0/\mathsf{x}) \rightarrow \mathscr{I}(\varphi)]$$
and
$$\mathscr{I}(\exists\mathsf{x}\,\varphi) = \exists\mathsf{x}\,[\omega_{\mathscr{I}}(\mathsf{x}_0/\mathsf{x}) \wedge \mathscr{I}(\varphi)]$$

An interpretation that is defined on formulas as well as sentences and satisfies these five conditions will be said to *preserve logic*. Preservation of logic is required if straightforward inductive proofs of properties of an interpretation are to be performed. Mappings defined by starting with a basis and extending it as described above will be called *standard interpretations* of $\mathscr{L}$ in $\mathscr{L}'$.

A standard interpretation $\mathscr{I}$ of language $\mathscr{L}$ in language $\mathscr{L}'$ is an *interpretation* of $\mathscr{L}$ in $\mathscr{L}'$-theory Θ' iff

$$\Theta' \models \exists\mathsf{x}_0\,\omega_{\mathscr{I}}$$

and, for every individual parameter a of $\mathscr{L}$,

$$\Theta' \models \exists\mathsf{x}_1\,\forall\mathsf{x}_0\,[\,\mathscr{I}(\mathsf{x}_0 \equiv \mathsf{a}) \leftrightarrow \mathsf{x}_0 \equiv \mathsf{x}_1]$$

Of course, a standard interpretation $\mathscr{I}$ of $\mathscr{L}$ in $\mathscr{L}'$-theory Θ' is an interpretation of *scrL*-theory Θ in Θ' if, for every sentence φ of the language of Θ,

$$\varphi \in \Theta \implies \mathscr{I}(\varphi) \in \Theta'$$

and it is faithful if, in addition, for every $\mathscr{L}$-sentence φ,

$$\mathscr{I}(\varphi) \in \Theta' \implies \varphi \in \Theta$$

The main result about standard theory interpretations that is needed for this paper is that they always preserve meaning, in a sense made precise by the following theorem.

Theorem. For any standard interpretation $\mathscr{I}$ of language $\mathscr{L}$ in $\mathscr{L}'$-theory Θ', any model $\mathfrak{A}'$ of Θ', any formula φ of $\mathscr{L}$, and any assignment x of values in $|\mathscr{I}^{\partial}(\mathfrak{A}')|$ to the free variables of φ,

$$\mathfrak{A}' \models \mathscr{I}(\varphi)\,[x] \iff \mathscr{I}^{\partial}(\mathfrak{A}') \models \varphi\,[x]$$

Proof. It is immediate from the definition of $\mathscr{I}^{\partial}$ (see page 4) that the equivalence holds for every *atomic* formula φ of $\mathscr{L}$. That it also holds for non-atomic $\mathscr{L}$-formulas follows easily by induction. For example, if φ is $\psi \wedge \chi$, then

$$\begin{aligned}
\mathscr{I}^{\partial}(\mathfrak{A}') \models (\psi \wedge \chi)\,[x] &\iff \mathscr{I}^{\partial}(\mathfrak{A}') \models \psi\,[x] \text{ and } \mathscr{I}^{\partial}(\mathfrak{A}') \models \chi\,[x] \\
&\qquad\qquad \text{(Definition of \textit{satisfaction})} \\
&\iff \mathfrak{A}' \models \mathscr{I}(\psi)\,[x] \text{ and } \mathfrak{A}' \models \mathscr{I}(\chi)\,[x] \\
&\qquad\qquad \text{(Induction hypothesis)} \\
&\iff \mathfrak{A}' \models (\mathscr{I}(\psi) \wedge \mathscr{I}(\chi))\,[x] \\
&\qquad\qquad \text{(Definition of \textit{satisfaction})} \\
&\iff \mathfrak{A}' \models \mathscr{I}(\psi \wedge \chi)\,[x] \\
&\qquad\qquad \text{(Definition of \textit{interpretation})}
\end{aligned}$$

Computer Science Laboratory, SRI International, Menlo Park, CA 94025
E-mail address: rar@csl.sri.com