The Problem of Logical-Form Equivalence*

Stuart M. Shieber
Aiken Computation Laboratory
Harvard University
Cambridge, MA 02138

August 28, 1992

1 History of the Problem

This note discusses the problem of logical-form equivalence, a problem in natural-language generation first noted in print by Douglas Appelt (1987). Appelt describes the problem, which we first came across in pursuing joint work on the GENESYS natural-language generation system, as a problem following from the goal of eliminating the need for a strategic generation component to possess detailed grammatical knowledge. In a paper describing the GENESYS tactical generation component (Shieber, 1988), I claimed that the problem was "AI-complete", in the sense that its resolution would involve a solution to the general knowledge representation problem.

Since the publication of these two papers, several researchers have claimed to have solved the problem of logical-form equivalence. In this paper, I review the problem, and attempt to highlight certain salient aspects of it that have been lost in the pursuing of solutions, in order to reconcile the apparently contradictory claims of the problem's intractability and its resolution.

2 Review of Natural-Language Generation

In order to standardize on a particular context in which to view the problem of logical-form equivalence, I review some concepts in natural-language generation.

A *grammar*, for the purposes of the argument here, can be taken to be a formal statement of a relation between strings of a natural language and representations of their meanings in some logical or other artificial language.

*I have benefited greatly from discussions with Douglas Appelt, Barbara Grosz, David Israel, David McDonald, Fernando Pereira, James Pustejovsky, Jason Stanley, and David Waltz on the topic of this paper. The opinions expressed here are my own, and are not necessarily shared by them, nor are they responsible for any remaining errors of form or substance.

We will call such representations *logical forms*. For each meaning of a string (recalling that strings may be ambiguous), the grammar ought to pair the string with a logical form representing that meaning. There may, in general, be several representations of any given meaning in the logical-form language; the one paired with the string by the grammar will be referred to as the *canonical logical form* for that string (under the given interpretation). It is not necessary for the grammar to pair the string with all logical forms that happen to represent the same meaning, and in fact, such a profligate approach would be unfortunate. It would make it more difficult, for instance, to determine whether a string was truly ambiguous.[1]

Under this view, the parsing problem involves computing the relation in the direction from string to meaning. In particular, from a string, the canonical logical form or forms are computed. The generation problem involves computing the relation in the direction from meaning to string. More precisely, this is the *tactical* generation problem, as opposed to the more difficult *strategic* generation problem (Thompson, 1977)—the problem of deciding what to say. I will assume that this latter problem is solved by some other computational device, which I will refer to as a *reasoner*, that also manipulates logical forms. The reasoner may be a strategic generator proper (as described in the generation literature) or some more specialized program. All that is required is that it be a computational artifact that needs the ability to construct utterances to convey certain meanings, and that it supplies meaning representations, logical forms, to a (tactical) generator for this purpose. See Section 4 for further discussion of these assumptions.

3 Definition of the Problem

Given a logical form (presumably supplied by such a reasoner), a generator[2] must, then, find a string with that meaning, that is, a string whose canonical logical form *means the same as* the given one. This is the *problem of logical-form equivalence*, the problem of constructing a generator that can generate not just from canonical logical forms but from all logical forms that mean the same.

The notion of meaning identity of logical forms is thus crucial to the problem. Meanings are something that we have a priori intuition of only for natural-language sentences, not logical forms; so meaning identity for logical forms

[1] This problem is reminiscent of the "spurious ambiguity" problem noted for combinatory categorial grammar (CCG). A number of distinct CCG analyses for a string may have identical semantics due to a uniformly available choice between application and composition. Identical semantics can arise from coincidental convergence of the meanings of two intrinsically distinct analyses as well. Thus, the number of CCG analyses of a string is not a measure of the string's true ambiguity, nor is the number of semantically distinct analyses.

[2] Here and following, I will use the term "generator" for what is more strictly referred to by the term "tactical generator". As should be clear from the foregoing discussion, our "reasoner" subsumes "strategic generator".

must be predicated on meaning identity for natural-language sentences. Suppose there are two sentences S_1 and S_2 with canonical logical forms L_1 and L_2. At minimum, the notion of meaning identity for logical forms should be such that if S_1 and S_2 mean the same, then L_1 and L_2 do, and conversely, if S_1 and S_2 do not mean the same, then neither do L_1 and L_2. Any logical forms that are not the canonical form for any sentence are, in a sense, free to mean the same as any other logical forms, but ideally, the logic (again, the term is used broadly) from which the logical forms are taken should define meaning identity in a consistent manner, as for example, through a uniform direct semantics for logical forms, or proof-theoretically, or computationally. Indeed, the reasoner embodies some such notion of meaning identity of logical forms implicitly in whatever computational operations it performs on logical forms. It is natural, then to take a notion of logical equivalence for the logical form language as an approximation to meaning identity. It is important to keep in mind, however, that the standard for meaning identity of logical forms is consistency with natural-language meaning identity, and it is meaning identity, not logical equivalence, that is the basis for the problem of logical-form equivalence.

One might be tempted to remove the requirement that a generator be able to generate from non-canonical logical forms. The reason that it is not sufficient to generate only from canonical logical forms comes from seeing a generator as it fits into a larger computational context. Recall that a generator is a peripheral device for use by some other computational mechanism, the reasoner, when it needs to utter a string to convey a given meaning. If the generator required a canonical logical form as input, the reasoner would be burdened with having to construct the canonical logical-form representation for the meaning it wanted to convey, not merely *some* representation for the meaning. Which of the many logical forms representing a given meaning is canonical is a grammatical issue, not a semantic one, and reasoners should not have to truck with grammatical issues. It is only in the context of a generator's use by a reasoner that the spirit of the logical-form equivalence problem can be seen. (It is no coincidence that the problem first confronted us in the context of using the CL-PATR generator with the KAMP reasoner in building the GENESYS system.)

4 Scope of the problem

The problem of logical-form equivalence is so-called for historical reasons; the interface between the KAMP planner and the CL-PATR generator used representations referred to as *logical forms*. The problem itself holds for any generation system that divides into a language-independent strategic reasoning component and a language-dependent tactical generation component, whether this representation be sentences of a logic narrowly construed or sentences of some other representation language (semantic networks, a knowledge representation language, specialized interface languages, bundles of answers to systemic chooser

questions, and so forth). *Throughout this paper, the term 'logical form' refers to the elements of whatever representation language serves as the interface between the reasoner and generator.* (The term *language* is to be construed very broadly here as well.) As I will argue, for any such language, either the strategic component will have to perform linguistic reasoning, or the interface representation language together with the tactical component will constitute a solution to the AI problem.

To further clarify the generality of the problem, note that the two components—reasoner and generator—need not operate sequentially. The interface representation may be communicated from the strategic to the tactical component in parts, and a single representation need never exist at any one point in time. The sum total of inputs driving the tactical generator constitutes a virtual logical form of the utterance. For instance, the GeneSys system coroutined the KAMP planner and CL-PATR generator, the logical form representation being communicated incrementally between the two. At no point was a single logical representation ever used, or even constructed. Similarly, the tactical component of a systemic generator may ask multiple questions of a reasoner to drive the generation through choice points in the various systems. The recursively structured bundle of answers to the various questions serves as a virtual logical form for the system. See Sections 7 and 9 for further discussion of systemic and functional approaches to generation.

On the other hand, there are restrictions on the scope of the problem. The problem of logical-form equivalence relies on the distinction between a language-free reasoner and a language-sensitive generator. If, as is often proposed, this distinction should be eliminated, then the problem of logical-form equivalence disappears. However, the single-component approach to generation does not thereby achieve an advantage, for the single remaining component encompasses all of the reasoning and linguistic aspects of generation, and is therefore at least as complex as the two components of the model presupposed here. In fact, the elimination of the division opens the door to greater complexity, modular systems with simple interfaces in general being more constrained than nonmodular ones. Whether the elimination of the strategic/tactical distinction is a useful move is, therefore, an independent, primarily empirical question, one concerning which I am personally agnostic and about which this squib will have nothing to contribute.

5 Examples of the Problem

To construct some examples, we need to specify a particular logical-form language and its relation to natural language sentences, thus imposing a notion of meaning identity on the logical forms. Let us take the language to be a first-order logic and consider the following pairings of strings and canonical logical forms that a hypothetical grammar might induce.

(*Item*)	String *Canonical Logical Form*
(*i*)	John threw a large red ball. $\exists x.throw(j,x) \wedge large(x) \wedge red(x) \wedge ball(x)$
(*ii*)	John threw a red ball that is large. $\exists x.throw(j,x) \wedge red(x) \wedge ball(x) \wedge large(x)$
(*iii*)	John threw a large ball that is red. $\exists x.throw(j,x) \wedge large(x) \wedge ball(x) \wedge red(x)$

Because these sentences mean the same (although they may differ in pragmatic effect) while their logical forms differ in permutation of conjuncts, we will take meaning identity for first-order logical forms to include commutativity and associativity of conjunction. In fact, we can go further, by assuming (for the nonce) that meaning identity is reasonably well approximated by the standard first-order-logic notion of logical equivalence.

By the commutativity and associativity of conjunction, all three of these sentences specify the same proposition. A generator confronted with a request to construct a sentence conveying the proposition represented by, say, $\exists x.throw(j,x) \wedge ball(x) \wedge large(x) \wedge red(x)$ ought to be able to generate all of these strings, or at least one of them. (Ideally, further specification of the intended pragmatic effect ought to control which string is generated, but such issues are incidental to the problem being discussed here.) Similar equivalences based on logical properties of other operators—including associativity and commutativity of disjunction, de Morgan's laws, laws of distributivity, equivalence of quantifier prefixes, and so forth—all introduce instances of the problem for a generator.

These considerations lead us to the reason for calling the problem of logical-form equivalence a *problem*. Logical equivalence is in general not computable, even for relatively inexpressive logics (like first-order logic). There are restricted cases in which it is computable; for instance, propositional logic has a decidable equivalence problem. However, even for such restricted logics, the problem is not effectively solved unless the notion of canonical form implicit in the grammar corresponds exactly to a notion of normal form for the logic (e.g., disjunctive normal form for propositional logic). Otherwise, we cannot use the existence of a normal form to compute the canonical form to drive the generator. Rather, all that can be done is to generate sentences blindly, and test equivalence post facto, which is an extremely profligate approach.[3]

As an approximation to meaning identity, the logical equivalence of first-order logic is too fine-grained. Intuitively at least, more logical forms should be equivalent than are provably equivalent by the laws of the logic alone. These further equivalences can be captured through their statement in a theory, thereby forming a better approximation to the meaning identity relation than logically valid equivalence alone. Not surprisingly, such theory-relative equivalences can

[3]However, see the later discussion of Levine's proposal that works exactly this way.

engender instances of the problem of logical-form equivalence, too. Suppose we are using a first-order language augmented with some generalized quantifiers and equality to represent meanings. A grammar might have the following pairings for a set of synonymous sentences:[4]

(*Item*)	String *Canonical Logical Form*
(*iv*)	Clapton was the leader of Derek and the Dominos. $the(x, leader\text{-}of(dd, x), c = x)$
(*v*)	The leader of Derek and the Dominos was Clapton. $the(x, leader\text{-}of(dd, x), x = c)$
(*vi*)	Clapton led Derek and the Dominos. $led(c, dd)$

It is a (presumably nonlogical) fact that

$$\forall x, y.led(x, y) \equiv leader\text{-}of(y, x)$$

Relative to this fact, along with certain logical equivalences concerning the referential uniqueness induced by the quantifier *the* and substitution of equals for equals, the (theory-relative) equivalence of all of these logical forms can be proved. As before, a generator ought to be able to generate any of these strings from any of their logical forms. And as before, the generator should also be taking into account pragmatic issues such as focus and presupposition in guiding the generation. Thus, we should be able to direct the generator to utter a sentence that means $led(c, dd)$ while focusing c and have it generate sentence (*iv*) above.

Above, we saw that the logical equivalence of first-order logic by itself is too fine-grained a notion of equivalence. At the same time, it is too coarse-grained as well. For example, consider conjoining a tautology with the logical form in (*vi*).

$$led(c, dd) \wedge (rain \vee \neg rain)$$

This formula is logically equivalent to (*vi*), but might well be the canonical logical form for the sentence

(*vii*) Clapton led Derek and the Dominos and either it is raining or it is not raining.

[4] The example is derived from that encountered in the work on GeneSys and described by Appelt. The reason that these particular logical forms are deemed the canonical ones for these sentences derives from the relatively direct relation between the syntactic form of the sentences and the associated logical form, as codified in a particular grammar.

Certainly, one would not want a generator to produce (*vi*) and (*vii*) interchangeably; they do not mean the same. This can serve as a reminder that the problem of logical-form equivalence is concerned with logical notions of equivalence *only insofar as these are good approximations to meaning identity.* (Section 8 addresses this point in more detail.) Many logics—relevance logics, for example—do not have these formulae as equivalent; their notions of logical equivalence are presumably more appropriate, for this one case at least, for building reasoners to interact with generators.

6 An Apparent Solution

The problem, in its barest form, may seem to be that a generator must be able to "undo" all of the equivalences given by the axioms (logical and nonlogical) of the logic that it uses to express logical forms. (This appearance of the problem comes about exactly because of the confusion between the meaning identity relation and the logical equivalence approximations thereto. We will purposefully indulge in this confusion for the moment.) For sufficiently expressive logics (and logics as simple as first-order logic are sufficiently expressive), computing whether two formulae are equivalent is undecidable, and for decidable logics such as propositional logic, searching through the infinite number of equivalents is, in any case, impractical.

An obvious method for resolving the problem of logical form equivalence, then, would be to restrict the power of the logic. By doing so, the logical equivalence relation becomes weaker, so that equivalence classes of logical forms are smaller, and searching equivalents for those that are canonical should be that much easier, perhaps even deterministic. In fact, both of the published claims of solution to the logical-form equivalence problem are of this variety. The first such claim that has come to my attention is due to Calder, Reape, and Zeevat (1989). They state that

> We must generate all sentences whose semantic representations are logically equivalent to the semantic representation being generated under the rules of inference and axioms of the semantic representation language.[5] In the case of InL [their representation language], the primary axioms are simply associativity and commutativity. However, these two axioms alone give the equivalence problem factorial complexity.

Since there are only a factorial number of equivalent logical forms, searching for the canonical one is a finite (though, as the authors note elsewhere, still impractical) process. John Levine (1990) reports on an interesting system called

[5] Note the authors' elegant statement of the original definition of the problem given in Section 3, although they use logical equivalence as a replacement for meaning identity. — SMS

PRAGMA that incorporates an apparent solution to the logical-form equivalence problem based on considering formulae identical if they have identical normal forms of a certain sort. The normal form is constructed as a clause form with equalities removed and with literals sorted in a standard way. Again, the notion of equivalence is weak, though stronger than that of Calder et al.

It may be of some historical interest to describe the solution that was implemented in the GeneSys system, a solution that I think has never before been described in published form. It, too, involved setting up a weak notion of equivalence, but of an entirely different sort. We implemented a simple term rewriting system that used a leftmost-outermost rewriting strategy, and specified rewrite rules that attempted to rewrite the given logical form into its canonical equivalent. Some rewrite rules implemented logical equivalences such as commutativity, others nonlogical equivalences such as the *led/leader-of* type of relationship, and so forth. The use of a rewriting system, and especially the leftmost-outermost strategy, gave us extremely fine control over the notion of equivalence that the generator was able to operate within, but the method was essentially ad hoc and thought of purely as an expedient, as opposed to a true solution to the problem.

In summary, the apparent solutions to the problem of logical-form equivalence rest on the observation that the choice of logical equivalence is free. It can therefore be chosen so as to make the task of the generator easier.

7 The Inadequacy of the Apparent Solution

There is a simple *reductio* against such apparent solutions to the logical-form equivalence problem. The extreme version of this approach to a solution is to weaken the notion of logical equivalence to one of syntactic identity. This notion of equivalence is extremely easy to compute, so that a generator can easily produce all strings whose canonical logical forms are equivalent to the given logical form. Nonetheless, there is an overwhelming feeling that the question has not so much been solved as begged.

The reason is perhaps obvious. This notion of logical equivalence is not a good approximation of meaning identity; it is far too intensional. But the entire basis for pursuing the use of a logical equivalence relation was that it served as an approximation to meaning identity. Thus, such solutions that weaken the notion of equivalence miss the spirit of the logical-form equivalence problem.

Such solutions do not unburden a reasoner from reasoning about canonicality of logical forms. Any divergence between the logic defining equivalence for the generator and the notion of meaning identity assumed by the reasoner implicitly embeds a claim: that the reasoner makes insufficiently many semantic distinctions. These distinctions are manifest in the meanings of natural language sentences, meanings that the reasoner is reasoning about; therefore it should be cognizant of these distinctions. For instance, if the generator logic

eschews commutativity of conjunction, whereas the reasoner incorporates it, the claim is thereby made that the properties conveyed by the nominals "large red ball" and "large ball that is red" differ; this claim is much stronger than the claim that they differ in pragmatic effect. In a similar vein, the notions of equivalence implicit in the work of Calder et al. and Levine do not conflate the logical forms (*i*), (*ii*), and (*iii*). The reasoner must choose the appropriate one by looking ahead to the generation process.

Thus, Calder et al. and Levine solve an artificial variant of the logical-form-equivalence problem; logical equivalence has become an end in itself, rather than a means to an end.[6] They show that the notion of equivalence in the logic used for the generator's logical forms is computable. But the actual problem would require that the notion of logic be appropriate for the reasoner to use as well, that it embody a notion of meaning identity. Without meaning identity to keep us honest, we could take advantage of the divergence between the two notions of equivalence to solve the problem by fiat.

In addition to the theoretic inadequacy due to divergence of the generator's equivalence notion from true meaning identity, there is a practical problem in the particular notions that Calder et al. and Levine choose. Although equivalence is decidable in their schemes, it is still computationally extremely expensive. In Calder et al.'s logic, there are a factorial number of equivalent logical forms. To directly incorporate their solution into a generator requires constructing the factorial number of equivalents and generating separately from each of them. This is a sufficiently expensive process that they do not actually implement the ability to generate from noncanonical logical forms in their system, wisely preferring to direct their efforts to defining notions of "semicanonicality" that admit of more efficient processing.

Levine's notion of equivalence is stronger (in that it finds more formulae to be equivalent) so much so that generation of all logical equivalents is impossible—there are infinitely many, in general. Levine states that "the problem of logical form equivalence can be solved relatively easily for the sentence generator...since it only attempts to equate the goal logical form with the constructed logical form when tree formation is complete." Thus, all that is necessary is a method for checking equivalence, not generating equivalents, and this is efficiently computable. It does, however, mean that the canonical logical form cannot itself be used to guide the generation process (since the particular logical form that will serve that purpose is not known until "tree formation is complete"). The generation process is therefore not particularly goal-directed, and many currently proposed tactical generation algorithms cannot be used. This presumably leads to less efficient generation, though Levine does not indi-

[6] It should be emphasized that the work of Calder et al. and Levine is only incidentally concerned with the logical-form-equivalence problem, and includes many other interesting ideas. The discussion here is not intended to denigrate their research efforts in any way. I am merely using their side discussions of the logical-form-equivalence problem to clarify some alternative conceptions.

cate any such problem. Surprisingly, Levine states that "this process is guided by the syntactic features attached to the sentence level node, but some semantic information from the goal logical form is also used to cut down the search space", which seems to contradict the idea of allowing divergence between the constructed and goal logical forms throughout the generation process.

It might be thought that the problem rests in using logical notations for representing meanings. Systemic grammarians, for instance, might think that their methods are immune to the problem of logical-form equivalence, because they do not represent the meaning as such in a logical-form expression. Rather, to guide generation they must merely query an oracle as to the answers to a definite set of questions. The answers that the oracle provides control the choices in the grammar and lead to the generation of the utterance. This feeling of relief is illusory. At some point, the oracle must answer questions on the basis of its representation of the situation to be described in the utterance. (This representation is analogous to a logical form.) If it answers the questions by direct lookup (as is done in the systemic generators I am aware of), then it requires the representation of the described situation to be in a kind of canonical form. If it answers the questions by reasoning from the representation, then we are back to constructing a canonical form from a noncanonical one, and arbitrary reasoning might be needed. To the extent that the reasoning is constrained but the reasoner is not burdened with grammatical issues, the problem of logical-form equivalence has been solved. Thus, the use of a systemic grammar does not make the problem moot; the problem is well-defined in the systemic context. Furthermore, existing systemic grammar generators do not, to my knowledge, solve the problem.[7]

Solutions to the problem of logical-form equivalence are thus revealed to lie along a continuum with respect to how close the match is between two notions of equivalence: (i) the relation of meaning identity on logical forms and (ii) the notion of logical equivalence that the generator is insensitive to (that is, the generator treats two logical forms as equivalent in this sense if either can equally well serve as the input and yield the same generated output). What is needed to solve the logical-form equivalence problem then? Essentially, any semantic distinction in the logic which ramifies in the syntax of the natural language must be somehow efficiently distinguishable in the semantic representation, and any semantically distinguishable facet of the semantic representation must have syntactic ramifications. The logical form language must allow for the representation of all and only the semantic distinctions that ramify syntactically.[8] It may make these distinctions by representing them explicitly or by having them be efficiently computable (that is, not by arbitrary reasoning), but they must

[7] It might be argued, however, that systemic or other functional approaches to language might have a leg up on constructing a solution. I discuss this possibility in Section 9.

[8] This raises the interesting possibility, noted by Zellig Harris (Harris, 1988, especially Chapter 1), that the only language appropriate for representing the semantics of natural language sentences are the sentences themselves.

be manifestable somehow.

It seems that a resolution of the logical-form equivalence problem must wait until a representation language has been invented that characterizes exactly the semantic distinctions that ramify in natural-language syntax. But this, reiterating my previous argument (Shieber, 1988), is one manifestation of the AI knowledge representation problem. There, I used the fact that the knowledge representation problem has not been solved (and might be considered AI-complete) as a proof, by reduction so to speak, of the AI-completeness of the logical-form equivalence problem. Certainly, the argument is not a proof in principle, but merely, one in practice; such a representation has not been developed *so far* but perhaps one will be devised some day. Nonetheless, I am not sanguine about the prospects.

8 Relations to Philosophy

One reason that ready solution of the problem of logical-form equivalence is not likely, is that it is essentially a philosophical problem, not a technological one. It may be that this accounts for its intransigence. The problem centers on precisely specifying a notion of meaning identity. This is a problem that has arisen in various notorious problems in the philosophy of language over the last century. I list a few examples here.

Identity of predicates: The question of when two predicates are identical ("mean the same") has long been a thorny philosophical problem. Russell addresses it in his philosophy of logical atomism (Russell, 1983). It is a central question in the theory of universals (Armstrong, 1978). Typically, the question is phrased as a concern over lexical predicates (e.g., renate/chordate). The problem of logical-form equivalence is the syntactic construction counterpart to this problem.

Substitution in opaque contexts: It is widely acknowledged that Liebniz's law does not hold in certain contexts (Frege, 1952). Intensional verbs, for instance, disallow such substitution. Thus, logical equivalence diverges from meaning identity with respect to this test at least. Nonetheless, certain substitutions might be allowed—which ones is a contentious topic—for instance, substitutions based on commutativity of conjunction. The issue hinges on examination of synonymy of sentences like the following:

> John knows that Clapton led Derek and the Dominos.
> John knows that Clapton was the leader of Derek and the Dominos.
> John knows that Clapton led Derek and the Dominos and either it is raining or it is not raining.

In a sense, generation is inherently an intensional process. If it were not, then a generator could concern itself only with the truth value of the logical form it was to generate from, uttering either "true" or "false" for all inputs. One can think of the process of generation of an utterance as implicitly incorporating an intensional context—something akin to "I tell you that..." or "I want you to believe that...". Indeed, all dialogue works this way. Russell (1940) expresses a similar sentiment: "In adult life, all speech ... is, in intention, in the imperative mood. When it seems to be a mere statement, it should be prefaced by the words 'know that'." It is not surprising, then, that there is a close relationship between the problem of logical-form equivalence and the philosophy of propositional attitudes and other intensional phenomena.

Reducing meaning to truth: Davidson (1984) attempts to build a theory of meaning on a theory of truth, that is, on a characterization of Tarskian T-sentences such as

> "Clapton led Derek and the Dominos" is true if and only if Clapton led Derek and the Dominos.

Foster (1976) notes the need for excluding such T-sentences as

> "Clapton led Derek and the Dominos" is true if and only if Clapton led Derek and the Dominos and either it is raining or it is not raining.

Although true, these sentences are undesirable in the enterprise of characterizing the meaning of the sentence on the left-hand side. Davidson's reply (1976) postulates the need, inter alia, for a canonical proof procedure that maintains a "translation relationship" between the left and right side of the biconditional in the T-sentences. Much of the argument against Davidson's approach centers on the difficulty or impossibility of constructing such a canonical proof procedure, and counterarguments rest on the existence of such procedures. The question remains open.

Any line of inquiry that requires a precise specification of the notion "means the same as" is, not surprisingly, going to get itself ensnared in a fair amount of philosophical flotsam.

9 Applicable Technology

It is important that researchers be aware of the philosophical nature of the problem, so that they do not attempt to develop technological fixes to what is, at heart, not a technological problem. Nonetheless, it may be that some technology

can be useful in developing approximate solutions to the problem of logical-form equivalence—logics that come closer to embodying meaning identity or methods for characterizing certain classes of meaning-identical logical forms. Some work that may be of potential use in this program of research—but certainly not an exhaustive listing—is discussed in this section.

Functional linguists, including those working in the systemic grammar tradition, have done a great deal of work in codifying the distinctions between logically equivalent sentences. By building on this work, better representational languages may be derivable. Of particular significance along these lines is the work of Hans Weigand (1990) who has explored the idea of designing knowledge representation languages on the basis of "linguistically motivated principles", based on the functional grammar of Dik (1981). He is coming at the same problem, but from the opposite direction.

Some recent work by knowledge representation designers has concentrated on low-power efficient knowledge representation languages. David McAllester's work on efficient inference systems based on analogies with natural-language syntax (McAllester and Givan, 1989b; McAllester and Givan, 1989a) is especially exciting in its potential application to the problem.

The lexical functions of Mel'čuk (1982) were originally proposed as a kind of universal theory of lexical relationships. But they might also serve as a useful taxonomy on which to build a notion of meaning identity. Although the functions that Mel'čuk defines may or may not be exhaustive as far as meaning identity goes, they certainly go a long way towards characterizing the most common such relationships. (For instance, the relationship between (*iv*) and (*vi*) above is related to the lexical function S_1 in Mel'čuk's terminology.)

It will be interesting to see to what extent these or other approaches can aid in approximating solutions to the problem of logical-form equivalence.

10 Summary

In summary, the problem of logical-form equivalence is really a problem of the divergence between two notions of equivalence, one associated with a generator, a kind of logical equivalence, and one associated with a reasoner, a notion of meaning identity. The crucial role of the reasoner's notion of equivalence was not made clear in earlier descriptions of the problem. Previous proposed solutions to the problem have, not surprisingly therefore, addressed a weaker variant of the problem, defined solely in terms of the generator's notion of equivalence, and consequently, have incorporated weak solutions to the problem. I remain agnostic as to the possibility of a strong solution to the problem of logical-form equivalence, but the relationships to difficult problems in core AI and the philosophy of language cast a long shadow. Nonetheless, approximate solutions based on certain techniques from computational linguistics and AI may provide for some illumination.

References

Appelt, Douglas E. 1987. Bidirectional grammar and the design of natural language generation systems. In Yorick Wilks, editor, *Proceedings of TINLAP-3*, pages 185–191, Las Cruces, NM, January 7–9. New Mexico State University. Reprinted in Yorick Wilks, editor, *Theoretical Issues in Natural Language Processing*, chapter 9.3, pages 199–205. Lawrence Erlbaum Associates, Hillsdale, NJ, 1989.

Armstrong, D. M. 1978. *Universals and Scientific Realism*. Cambridge University Press, Cambridge. Two volumes.

Calder, Jonathan, Mike Reape, and Hank Zeevat. 1989. An algorithm for generation in unification categorial grammar. In *Proceedings of the 4th Conference of the European Chapter of the Association for Computational Linguistics*, pages 233–240, Manchester, England, 10-12 April. University of Manchester Institute of Science and Technology.

Davidson, Donald. 1976. Reply to Foster. In *Truth and Meaning: Essays in Semantics*. Oxford University Press, Oxford, chapter II, pages 33–41.

Davidson, Donald. 1984. Truth and meaning. In *Inquiries into Truth and Interpretation*. Oxford University Press, Oxford, pages 17–36.

Dik, Simon C. 1981. *Functional Grammar*, volume 7 of *Publications in Language Sciences*. Foris, Dordrecht, Holland, 3rd revised edition.

Foster, J. A. 1976. Meaning and truth theory. In *Truth and Meaning: Essays in Semantics*. Oxford University Press, Oxford, chapter I, pages 1–32.

Frege, Gottlob. 1952. On sense and meaning. In Peter Geach and Max Black, editors, *Translations from the Philosophical Writings of Gottlob Frege*. Basil Blackwell, Oxford, pages 56–78.

Harris, Zellig. 1988. *Language and Information*. Columbia University Press, New York ,NY.

Levine, John M. 1990. PRAGMA—A flexible bidirectional dialogue system. In *Proceedings of AAAI-90*.

McAllester, David and Robert Givan. 1989a. Natural language syntax and first order inference. A.I. Memo No. 1176, Massachusetts Institute of Technology, Cambridge, Massachusetts, October.

McAllester, David and Robert Givan. 1989b. Taxonomic syntax for first order inference. A.I. Memo No. 1134, Massachusetts Institute of Technology, Cambridge, Massachusetts, June.

Mel'čuk, Igor. 1982. Lexical functions in lexicographic description. In *Proceedings of the Eighth Annual Meeting of the Berkeley Linguistics Society*, pages 427–444, Berkeley, California. University of California.

Russell, Bertrand. 1940. *An Inquiry into Meaning and Truth*. Allen and Unwin.

Russell, Bertrand. 1983. The philosophy of logical atomism. In John G. Slater, editor, *The Philosophy of Logical Atomism and Other Essays, 1914–1918*. George Allen and Unwin, London, chapter Part III, pages 155–244.

Shieber, Stuart M. 1988. A uniform architecture for parsing and generation. In *Proceedings of the 12th International Conference on Computational Linguistics*, pages 614–619, Budapest, Hungary, 22–27 August. Karl Marx University of Economics.

Thompson, Henry. 1977. Strategy and tactics: A model for language production. In *Papers from the 13th Regional Meeting, Chicago Linguistic Society*, Chicago, Illinois.

Weigand, Hans. 1990. *Linguistically Motivated Principles of Knowledge Base Systems*, volume 12 of *Functional Grammar Series*. Foris Publications, Dordrecht, Holland.