A Satisfiability-Preserving Reduction of IKL to Common Logic

Pat Hayes, Florida IHMC 2009

Introduction

IKL is a version of first-order logic which has one major extension, a syntactic device which provides a name for propositions incorporating a sentence form. IKL can thus *describe* the propositions expressed by its own *sentences*. Originally developed as an interlingua for knowledge exchange between a small set of logic-based KR systems, IKL has proven to be highly adaptable to a large variety of extant formalisms, and is now widely thought of as the most universally expressive first-order logic yet formalized, which is quite surprising given the syntactic simplicity of the proposition-name extension. We have recently discovered that IKL is in fact reducible to a conventional FO logic by eliminating all occurrences of the 'proposition name' construction, using a transformation which generalizes the familar Skolem elimination of existential quantifiers. We refer to this as IKL normalization. It represents an essential basic step towards implementing an IKL reasoning engine, and also solves what was previously an open question regarding the existence of IKL interpretations.

This work was supported by the National Science Foundation under contract #0742022.

IKL Normalization

IKL is similar to CLIF, but with two notable extensions and a few small changes.

(a) IKL syntax allows *proposition names* of the form

(that **<sentence>***)*

with a strong semantic constraint upon the use such a name as a unary relation: the atomic sentence formed by calling such a proposition with no arguments must have the same truth-value is **<sentence>.**

(b) IKL semantics imposes a constraint upon the use of a quoted string as a function name. The term formed by calling such a quoted string with no arguments must denote the same entity as the striong considered as a CL name. That is, all equations of the form

(= ***foo*** *(* ***'foo'*** *))*

are considered to be IKL tautologies. Note that this is perfectly legal CLIF syntax, so this new IKL constraint is a purely semantic restriction. This can be mapped to CL by adding suitable equations of this form as axioms to every set of IKL sentences, one for each name used in the sentences.

(c) IKL syntax does not allow the 'guarded quantifier' construction of CLIF, but it does support a *numerical quantifier* syntax, of the form

(*<quantifier> <numeral>** *(x)* **<sentence>*)

with the usual meaning, so that for example   *(exists 3 (x)(IsLegOf Harry x))*

means

```
(exists (x1 x2 x3)(and
        (allDifferent x1 x2 x3)
        (isLegOf Harry x1) (isLegOf Harry x2) (isLegOf Harry x3)
))
```

This is not really an extension of the language since every case can be re-writtten using conventional quantifier syntax. It is simply a notational convenience.

We will assume that the last two of these are handled by suitable re-writing or addition of equations, as described. However, the first is more complex, and is also where the theoretical interest lies.

Eliminating Proposition names

Suppose that we have an IKL sentence PHI containing a proposition name *(that* RHO*)*:

PHI[*(that* RHO*)*]

where PHI and RHO are sentences; and suppose initially that the sentence RHO contains no names which are bound by any quantifiers in PHI and that this is the only proposition name which occurs in PHI (so that in particular, RHO does not itself contain any proposition names and is in fact a Common Logic sentence.) Now consider the following transformation of PHI where every occurrence of the proposition name is replaced with a new simple name *prop1*:

PHI[*prop1*] *(iff (prop1)* RHO*)*

This is a *text* comprising two sentences, and meaning their conjunction. Call this text S(PHI) with the first sentence the *head* and the other sentence the *tail*. The proposition name has been eliminated from S(PHI), which therefore contains no proposition names and is a CL text.

The semantic relationship between PHI and S(PHI) is as follows. Any IKL interpretation I of PHI defines a CL interpretation J of S(PHI) which satisfies the tail, and vice versa; such that in every case, J assigns the same truthvalue to the head as I does to PHI. Thus, satisfiability is preserved by this mapping from PHI to S(PHI).

The proof of this relationship is rather obvious. Given the IKL interpretation I, we simply make the name *prop1* refer in J to the entity that the proposition name *(that* RHO*)* refers to in I. Then the head sentence is interpreted in J exactly isomorphically to its interpretation in I, and the tail sentence is true in J by virtue of the IKL semantic condition on proposition names. The reverse construction is similar. In fact, the equivalence can be derived from the equation
(= prop1 (that RHO))
as follows: PHI and the equation together entail S(PHI) in IKL; and PHI, the equation and the tail together entail the equivalence of S and the head in Common Logic.

This construction suggests how IKL sentences containing proposition names can be transformed into CL texts which are closely related in meaning, so that IKL satisfiability is transformed into CL satisfiability, allowing conventional FOL inference engines to be appplied to testing IKL unsatisfiability entailment. In order to achieve this, however, we have to remove the restrictive conditions.

Quantifier intrusions

First, we must consider the possibility that the proposition name sentence RHO might contain names which are bound by quantifiers in PHI. This situation can arise very naturally. For example, consider the sentences

(FredBelieves (that (forall ((x Human))(exists (y)(hasFather x y)))))
(forall ((x Human))(exists (y)(FredBelieves (that (hasFather x y)))))

The first asserts a reasonable belief to Fred: that everyone has a father. The second says that every human has a corresponding other, of whom Fred believes that it is that particular person's father. In IKL terms, the change is signaled by the fact that outer quantifiers bind names inside proposition names.

If a proposition name P contains a name N which is bound by a quantifier outside the proposition name, then we will say that N is an *intrusion* by the outer quantifier into P. Note that the binding quantifier might be universal or existential. The first stage in the final transformation is to eliminate all but universal intrusions, which we will do in a conventional way by skolemizing the sentence: that is, by replacing every existentially quantified name by a term whose function is a new name and whose argument sequence comprises all the universally quantified names which are bound by a quantifier whose scope contains the existential quantifer being eliminated. However, this must be modified in IKL to exclude quantifiers which occur inside a proposition name. For example, the sentence *(forall ((x Human))(FredBelieves (that (exists (y)(hasFather x y)))))*

is unchanged by skolemization: the inner existential is not eliminated nor its bound name replaced by a skolem term.

Replacing a sentence by its skolemization preserves satisfiability both in IKL and in CL, and for the same reasons. Let us assume, then, that our sentence is skolemized; then all the intrusions into every top-level proposition name are universal, and we can generalize the proposition-name replacement rule by one which replaces the proposition name by a skolem-like term *(prop1 x1 ... xn)*, where *prop1* is a new name and *x1* through *xn* are all the names which intrude into the proposition name. In the tail, we simply universally quantify these:

PHI[(*prop1x1 ... xn)*] *(forall (x1 ... xn)(iff ((prop1 x1 ... xn))* RHO*)*

Our example sentences would then yield the following texts:

(FredBelieves (that (forall ((x Human))(exists (y)(hasFather x y))))) ==>

> *(FredBelieves prop1)*
> *(iff (prop1) (forall ((x Human)(hasFather x (sk1 x))))*

(forall ((x Human))(exists (y)(FredBelieves (that (hasFather x y))))) ==>

> *(forall ((x Human))(FredBelieves (prop1 x)))*
> *(forall (x)(iff ((prop1 x)) (hasFather x (sk1 x))))*

The difference in meaning is made, if anything, more vivid by the transformation, which makes it very clear that the second sentence is attributing a much heavier burden of belief to Fred, if only in the sheer number of propositions for which he can be held accountable. Note however that in the third example, the transformation exposes the inner existential quantifier:

(forall ((x Human))(FredBelieves (that (exists (y)(hasFather x y))))) ==>

> *(forall ((x Human))(FredBelieves (prop2 x)))*
> *(forall (x)(iff ((prop2 x)) (exists (y)(hasFather x y))))*

The transformation described here has the same semantic properties as those described above for the restricted case, and the argument given there generalizes straightforwardly. The function extension of the denotation of the *prop1* function in J is the set of sequence pairs *<<a1 ... an> p>*, where *p* is the denotation of the proposition name in I when I(*xi*)=*ai* for each i between 1 and n.

Multiple proposition names

Finally, we need to consider the case where the IKL sentence contains several proposition names. In this case, we can simply iterate the mapping, each time reducing the number of names by one. An obvious inductive argument shows that this process of normalization must terminate with a CL text. As shown above, each iteration may expose a new quantifier in the tail sentences, which must therefore be re-skolemized before the mapping is applied again. The resulting patterns of skolemization can become quite complex, since the exposed sentence is embedded in a biconditional. The above example would yield:

> *(forall ((x Human))(FredBelieves (prop2 x)))*
> *(forall (x)(if ((prop2 x)) (hasFather x (sk2 x))))*
> *(forall (x y)(if (hasFather x y)((prop2 x))))*

with a skolem term in one half of the biconditional but a double universal quantification in the other half. Such a pattern is typical.

If a tail sentence contains a proposition name then the transformation must of course be applied to those sentences also: in this case, both the head and the tail of the later transformation are treated as part of the tail. Thus, the final text so produced will always have a single head, corresponding in meaning to the original IKL sentence when the tail is true.

We have shown, therefore, that there is a direct syntactic transformation, generalizing the well-known skolemization process used to eliminate existential quantifiers, which takes any IKL sentence to a CL sentence (in fact, a sentence in the CLIF CL dialect) and preserves satisfiability. This achieves a limited, but adequate, reduction of IKL to Common Logic which is sufficient to allow IKL satisfiability and entailment to be detected by a CL reasoner. The transformation can be summarized by the following pseudocode:

```
TAIL := nil
NORMAL(E) :=
    E := Skolemize(E);
    UNTIL E contains no proposition names DO(

        Let P = (that S) be a proposition name in E;
        Let U1,.., Un = the intrusions in P;
        K := gensym( );
        E :=  substitute (K U1 ... Un) for P in E ;
        TAIL:= TAIL +
            NORMAL( (forall (U1 ... Un)(iff ((K U1 ... Un)) S )) ) ;
    )
    RETURN( E + TAIL );
```

Normalizing the paradoxes

As an illustrative exercise, we give here the results of applying this process to the 'paradoxical' sentences reviewed in (Hayes 2009):

1. That Nothing Is True (TNIT).

((that (forall (p)(not (p)))))   ==>

(prop1)
(iff (prop1) (forall (p)(not (p))))

2. The Liar.

(= p (that (not (p))))   ==>

(= p prop2)
(iff (prop2) (not (p)))

3. Kripke's semantic paradox.

(forall (x)(iff (S x)(= x (that (forall (y)(if (S y) (not (y))))))))   ==>

(forall (x)(iff (S x)(= x prop3))
(iff (prop3) (forall (y)(if (S y)(not (y)))))

4. The Knower. (Kaplan & Montague 1950)

(forall (x)(if (K p) (p)))
(forall (x)(K (that (if (K p) (p)))))
(forall (x y)(if (and (if (x)(y)) (K x)) (K y)))
(= D (that (K (that (not (D))))))   ==>

(= D prop4)
(iff (prop4)(K prop5)) (iff (prop5) (not (D)))

The last of these illustrates the treatment of nested proposition names. This is the only example that would require more than one iteration of the main loop of the pseudocode.

IKL Satisfiability and Entailment

The noramlization mapping preserves satisfiability, but the semantic relationship is somewhat more complex than that of simple skolemization. In particular, if the tail sentences are inconsistent in CL, then the semantic argument given would lead to the conclusion that the IKL semantic conditions cannot be satsified by any IKL interpretation of the IKL sentence, i.e. that the IKL sentence in this case has no interpretation at all. We have not found any such example, and believe that it is impossible, but do not at the tim,e of writing have a conclusive proof. However, this possible situation of having no possible interpretation, while not standard, is not, we suggest, a fatal flaw in the IKL semantics even if it can in fact arise. We can simply take the syntactic transformation given here as *defining* the meaning of an IKL sentence – in effect, by treating IKL as asserting the IKL semantic conditions for each of its proposition names explicitly – whcih guarantees that an interpretation is always possible. The 'rogue' case of a setnence which violates the IKL sematnic conditions is then treated simply as an exotic case of IKL inconsistency, as indeed it is; for having no interpretations at all implies having no satisfying interpretations.

This means that the tail sentences of a normalized IKL sentence represent the extra burden placed upon an inference engine to enable it to check the internal consistency of any IKL sentence, required by the IKL truth conditions on proposition names.

Reference

Hayes 2009 *FIrst-order logic can describe its own propositions*. Memorandum, Florida IHMC.