Coherent Translation of FOL

John Fisher
jrfisher@cpp.edu

(DRAFT September 4, 2009 - September 8, 2015)

Abstract

This technical report describes methods for translating FOL (first-order logic) into coherent logic in preparation for computation of the coherent theory by a Skolem Machine. The fore-and-aft translation methods attempt to faithfully preserve FOL formulas already having coherent form.

1 Introduction

In §7 of reference [1], Bezem and Coquand describe a scheme for translating first-order logic wffs into coherent logic form. That general scheme is very straightforward and satisfactory for converting many simple logical theories into coherent logic form, in preparation for the application of a coherent logic prover to the resulting coherent theory. The primary disadvantage of that translation scheme is its preference for producing negative forms, and for insisting on proof by contradiction. For example, a simple coherent first-order axiom $p \rightarrow q$ is translated into the coherent form $true \Rightarrow \neg p \mid q$ rather than into the positive coherent form $p \Rightarrow q$. Also, proof-by-contradiction requires that a conjecture be negated and the negation be shown to contradict the axioms. This preference for negated forms can introduce unnecessary disjunctive cases into the resulting proof searches.

In this report we describe deterministic translation schemas for FOL that avoid injecting negative forms into coherent translations when possible. The schemas preserve FOL inputs already having coherent form.

The subsection 1.1 describes the typed quantifiers for the FOL input language and FOL theories. Subsection 1.2 describes the coherent logic target language, *colog* .

In §2 we work several examples of the translation methods in preparation for the formal presentation of the translation schemas in §3. The fore-and-aft translations schemas of §3 transform each FOL wff of a theory into sequence of *colog* rules. Simplifying transformations can be applied to produce a shorter translation of a wff (fewer *colog* rules and more terse form). Translating all of the wffs of a FOL theory (in order) thus produces a geolog theory when all of the resulting *colog* rules are merged together.

The text form input language for FOL is similar to the TPTP language [5], with various differences: For example, FOL does not use annotations for its wffs. It is straightforward to translate between FOL form and TPTP forms. Appendix A has more details about the syntax for FOL , and Appendix B has worked examples using automatic translation of text forms of FOL .

1.1 typed quantification and FOL

We will assume that FOL quantifiers may be augmented with *types*. The wff

$$(\forall x : k)\ \phi \tag{1}$$

is taken to be logically equivalent to the wff

$$(\forall x)(k(x) \rightarrow \phi) \tag{2}$$

The key symbol k is presumed to be a unary predicate which can occur without restriction in input wffs. Simarly,

$$(\exists x : k)\ \phi \tag{3}$$

is equivalent to

$$(\exists x)(k(x) \wedge \phi) \tag{4}$$

An input wff of the form

$$(\forall x : k_1)\ k_2(x) \tag{5}$$

is thus equivalent to the wff

$$(\forall x)(k_1(x) \rightarrow k_2(x), \tag{6}$$

so the axiom specification (5) orders the types k_1 and k_2 by *inclusion*:

$$k_1 \leq k_2. \tag{7}$$

When a positive wff has the form $(\forall x)\phi$ without a named explicit type, we assume that the key has a generalised form:

$$(\forall x : dom)\ \phi \tag{8}$$

and similarly for $(\exists x)\phi$. The *dom* type designates a *universal* domain. For every explicit key $x : g$ that appears in the theory we assume that the FOL wff

$$(\forall x : g)\ dom(x) \tag{9}$$

is also an implicit axiom of the theory (even if not explicitly provided). Thus, for each type t we have that $t \leq dom$. All types of things belong to the universal domain. User programs may contain $dom/1$ (a unique unary *dom*) provided its special role is understood. In fact, the programmer is required to supply domain and closure axioms in many cases: See §4 for more on this topic.

Typed quantifiers are intended to restrict the range of application of the wff within the scope of the quantifier. One might say that the type in a quantifier *unlocks* or enables the application of the quantifier As an example, suppose that we wish to specify that function f has domain *real* and range *int*. We may wish to express this in the following convenient form

$$(\forall x : real)\ int(f(x)) \tag{10}$$

Including *notations* for quantifier keys allows one to *express* some information about *types* in FOL. We are not presuming any *other* special semantics for typed first-order logic, but we will specify translations to coherent logic in Section 3 which are consistent with the intentions specified in (2) and (4).

A *FOL theory* is a finite sequence of wffs. The wffs in a theory may have typed quantification. Each wff in a theory is translated using the translation schema of Section 3. Section 5 has a discussion of completeness issues regarding such translations.

1.2 *colog* theories

colog is a language for expressing first-order coherent logic in a format suitable for computations using an *abstract machine*. *colog* rules are used as machine instructions for an abstract machine that computes consequences for first-order geometric logic.

A *colog rule* has the general form

$$A_1, A_2, \ldots, A_m \Rightarrow C_1 \mid C_2 \mid \ldots \mid C_n \tag{11}$$

where the A_i are atomic expressions and each C_j is a conjunction of atomic expressions, $m, n \geq 1$. The left-hand side of a rule is called the *antecedent* of the rule (a conjunction) and the right- hand side is called the *consequent* (a disjunction). All atomic expressions can contain variables.

If $n = 1$ then there is a single consequent for the rule (11), and the rule is said to be *definite*. Otherwise the rule is a *disjunctive* or *splitting rule* that requires a case distinction (case of C_1 or case of C_2 or $\ldots$ case of C_n).

The separate cases (disjuncts) C_j must have a conjunctive form

$$B_1, B_2, \ldots, B_h \tag{12}$$

where the B_i are atomic expressions, and $h \geq 1$ varies with j. Any free variables occurring in (12) other than those which occurred free in the antecedent of the rule are taken to be existential variables and their scope is this disjunct (12).

As an example, consider the *colog* rule

$$s(X,Y) \Rightarrow e(X,Y) \mid dom(Z), r(X,Z), s(Z,Y). \tag{13}$$

The names for *colog* predicates start with lower-case letters and the names for *colog* variables start with upper-case letters, as for Prolog language syntax. The variables X and Y are universally quantified and have scope covering the entire formula, whereas Z is existentially quantified and has scope covering the last disjunct in the consequent of rule. A fully quantified first-order logical formula representation of this *colog* rule would be the coherent logic wff

$$(\forall X)(\forall Y)[s(X,Y) \rightarrow e(X,Y) \vee (\exists Z)(dom(Z) \wedge r(X,Z) \wedge s(Z,Y))] \tag{14}$$

Now we come to two special cases of rule forms, the *true* antecedent and the *goal* or $false$ consequents. Rules of the form

$$true \Rightarrow C_1 \mid C_2 \mid \ldots \mid C_n \tag{15}$$

are called *factuals*. Here '*true*' is a special constant term denoting the empty conjunction. Factuals are used to express initial information in *colog* theories. Rules of the form

$$A_1, A_2, \ldots, A_m \Rightarrow goal \tag{16}$$

are called *goal* rules. Here '*goal*' is a special constant term. A *goal* rule expresses that its antecedent is sufficient (and relevant) for *goal*. Similarly, rules of the form

$$A_1, A_2, \ldots, A_m \Rightarrow false \tag{17}$$

are called $false$ rules. Here '$false$' is a special constant term denoting the empty disjunction. A $false$ rule expresses rejection of its antecedent.

The constant terms *true*, *goal* and $false$ can only appear in *colog* rules as just described. *true* plays the assumption role of $\top$ in a FOL theory and *goal* plays the conjecture role of $\top$. $false$ plays the denial role of $\bot$ in a FOL theory. All other predicate names, individual constants, and variable names are the responsibility of the *colog* programmer.

A *colog theory* (or *program*) is defined to be finite sequence of *colog* rules. A theory may have any number of factuals and any number of *goal* or $false$ rules.

Coherent form logical formulas, and a bottom-up approach to reasoning with those logical formulas, finds its earliest precursor (1920) in a particular paper by Thoralf Skolem [**?**].

2 Worked fore-and-aft translations

The concepts underlying fore-and-aft translation of first-order logic may seem somewhat obscure initially. The naturalness of this approach requires a small stretch of intuition ...

Ships of logic
sail on winds
fore to truth
aft to conjecture

We present three translation examples worked by hand using conventional FOL notation (augmented for fore-and-aft modes). The Appendix contains translations produced by an automated translator (*colog*) using text input formats of FOL .

Example 1. The wff $((a \vee b) \rightarrow c) \rightarrow d$ is converted to *colog* rules, as follows:

$true \Rightarrow \overrightarrow{((a \vee b) \rightarrow c) \rightarrow d}.$

$\overrightarrow{((a \vee b) \rightarrow c) \rightarrow d}, \overleftarrow{((a \vee b) \rightarrow c)} \Rightarrow \overrightarrow{d}.$

$\overleftarrow{\neg(a \vee b)} \Rightarrow \overleftarrow{((a \vee b) \rightarrow c)}.$

$\overleftarrow{c} \Rightarrow \overleftarrow{((a \vee b) \rightarrow c)}.$

$\overleftarrow{\neg a}, \overleftarrow{\neg b} \Rightarrow \overleftarrow{\neg(a \vee b)}.$

$not\_a \Rightarrow \overleftarrow{\neg a}.$

$not\_b \Rightarrow \overleftarrow{\neg b}.$

$c \Rightarrow \overleftarrow{c}.$

$\overrightarrow{d} \Rightarrow d.$

If any resulting geolog rule has an occurrence of a negated predicate $\neg p$ in the consequent of the rule the translator adds a corresponding consistency axiom.

$p(\hat{X}), not\_p(\hat{X}) \Rightarrow false.$

where $\hat{X}$ is a sequence of variable arguments matching the arity of p. In this example, no consistency rule is added.

The *fore operator* $\overleftarrow{(\ldots)}$ is *tried* whenever formal implication is encountered. The trial fails when an exceptional case is encountered while trying to proceed in the fore direction. In this example *fore translation* goes smoothly, without exception. The *aft operator* $\overrightarrow{(\ldots)}$ always applies.

See Appendix B.1 for the results of automated translation.

Example 3 below will illustrate how fore translation can lead to an exceptional case, and the example illustrstes how to translate when an exception arise.

The next example illustrates some fore and aft translation patterns using quantified variables. Again, the fore translation goes smoothly.

Example 2. The FOL wff $(\forall x : a)(\neg p(x) \rightarrow (\exists y : b)(\exists z : c)q(x, y, z))$. has coherent form. We expect the translation to preserve its coherence.

$true \Rightarrow \overrightarrow{(\forall(x : a)(\neg p(x) \rightarrow (\exists y; b)(\exists z : c)q(x, y, z))}.$

$\overrightarrow{(\forall(x : a)(\neg p(x) \rightarrow (\exists y : b)(\exists z : c)q(x, y, z))}, a(X) \Rightarrow \overrightarrow{\neg p(x) \rightarrow (\exists y : b)(\exists z : c)q(x, y, z)}(X).$

$\overrightarrow{\neg p(x) \rightarrow (\exists y : b)(\exists z : c)q(x, y, z)}(X), \overleftarrow{\neg p(x)}(X) \Rightarrow \overrightarrow{(\exists y : b)(\exists z : c)q(x, y, z)}(X).$

$not\_p(X) \Rightarrow \overleftarrow{\neg p(x)}(X).$

$\overrightarrow{(\exists y : b)(\exists z : c)q(x, y, z)}(X) \Rightarrow b(Y), \overrightarrow{(\exists z : c)q(x, y, z)}(X, Y).$

$\overrightarrow{(\exists z : c)q(x, y, z)}(X, Y) \Rightarrow c(Z), \overrightarrow{q(x, y, z)}(X, Y, Z).$

$\overrightarrow{q(x, y, z)}(X, Y, Z) \Rightarrow q(X, Y, Z).$

The geolog rules are verbosely expressed in order to illustrate translation steps incrementally. Automated translators can skip or combine several steps without loss of generality. Also, the variable correspondences are presented in an intuitive fashion (e.g., formal variable x becomes X in the geolog rule).

See Appendix B.2 for the results of automated translation. Pay particular attention to the terse translation!

The next example illustrates an exceptional case of fore translation. An embedded formal implication will have to introduce a negation and be expressed using disjunction in a coherent rule.

Example 3. Supose that $\phi = (\forall x)((\forall y)p(x, y) \rightarrow q(x))$. Note that there is an embedded universal quantifier in the antecedent of the formal implication. First, let us see how the exception to fore translation arises ...

$true \Rightarrow \overrightarrow{(\forall x)((\forall y)p(x, y) \rightarrow q(x))}.$

$\overrightarrow{(\forall x)((\forall y)p(x, y) \rightarrow q(x))}, dom(X), \Rightarrow \overrightarrow{((\forall y)p(x, y) \rightarrow q(x))}(X).$

$\star$ $\overrightarrow{((\forall y)p(x, y) \rightarrow q(x))}(X), \overleftarrow{(\forall y)p(x, y)}(X) \Rightarrow \overrightarrow{q(x)}(X).$

EXCEPTIONAL : ...??? $\Rightarrow \overleftarrow{(\forall y)p(x, y)}(X).$

so $\star$ is replace by ...

$\overrightarrow{((\forall y)p(x, y) \rightarrow q(x))}(X) \Rightarrow \overrightarrow{\neg(\forall y)p(x, y)}(X) \mid \overrightarrow{q(x)}(X).$

and now continue ...

$\overrightarrow{\neg(\forall y)p(x,y)}(X) \Rightarrow \overrightarrow{(\exists y)\neg p(x,y)}(X).$

$\overrightarrow{(\exists y)\neg p(x,y)}(X) \Rightarrow dom(Y), \overrightarrow{\neg p(x,y)}(X,Y).$

$\overrightarrow{\neg p(x,y)}(X,Y) \Rightarrow not\_p(X,Y).$

In this example, the attempt to use a fore translation for the embedded formal implication leads to an exception: The primary translation schemas avoid fore translation for a universal form (or negated existential). Such a translation must be redone with aft translation. See Appendix B.3 for the results of automated translation.

3 Formal schemas defining fore-and-aft translations

In the following tables, the right-facing arrow $\overrightarrow{(\ldots)}$ indicates the "aft" direction for a translation and the left-facing arrow $\overleftarrow{(\ldots)}$ indicates the "fore" direction for a translation (as was the case in the previous section).

Fig. 1 specifies the *aft* translations. These are essentially the same as those specified in Reference [1] except for schema (4). Notice that schema (4) is the only aft schema to introduce a *fore* translation. If the fore translation excounters an exception then one uses the aft schema to translate the form.

Fig. 2 specifies the fore translations.

In the schemas, a sequence $\hat{X}$ always denotes an ordered sequence of variable arguments that arose *before* the current schema is supposed to apply. For convenience, we assume that the lower-case variable letters in the FOL wffs become upper-case when the corresponding variable gets elevated to arguments in the geolog rules.

(A1) $\overrightarrow{\phi \wedge \psi}(\hat{X}) \Rightarrow \overrightarrow{\phi}(\hat{X}), \overrightarrow{\psi}(\hat{X})$

(A2) $\overrightarrow{\phi \vee \psi}(\hat{X}) \Rightarrow \overrightarrow{\phi}(\hat{X}) \mid \overrightarrow{\psi}(\hat{X})$

(A3
- a) $\overrightarrow{\neg(\phi \wedge \psi)}(\hat{X}) \Rightarrow \overrightarrow{\neg\phi}(\hat{X}) | \overrightarrow{\neg\psi}(\hat{X})$
- b) $\overrightarrow{\neg(\phi \vee \psi)}(\hat{X}) \Rightarrow \overrightarrow{\neg\phi}(\hat{X}), \overrightarrow{\neg\psi}(\hat{X})$
- c) $\overrightarrow{\neg\neg\phi} \Rightarrow \overrightarrow{\phi}$
- d) $\overrightarrow{\neg(\phi \rightarrow \psi)}(\hat{X}) \Rightarrow \overrightarrow{\phi}(\hat{X}), \overrightarrow{\neg\psi}(\hat{X})$
- e) $\overrightarrow{\neg(\forall x : g)\phi}(\hat{Y}) \Rightarrow \overrightarrow{(\exists x : g)\neg\phi}(\hat{Y})$
- f) $\overrightarrow{\neg(\exists x : g)\phi}(\hat{Y}) \Rightarrow \overrightarrow{(\forall x : g)\neg\phi}(\hat{Y})$
- g) $\overrightarrow{\neg a(\hat{x})} \Rightarrow not\_a(\hat{X})$ % when a is atomic
  $a(\hat{X}),\ not\_a(\hat{X}) \Rightarrow false.$ % add consistency rule
  % if $not\_a(\hat{X})$ occurs in consequent of geolog rule

(A4) $\overrightarrow{\phi \rightarrow \psi}(\hat{X}), \overleftarrow{\phi}(\hat{X}) \Rightarrow \overrightarrow{\psi}(\hat{X})$ if translation of $\overleftarrow{\phi}$ is *NOT EXCEPTIONAL*
$\overrightarrow{\phi \rightarrow \psi}(\hat{X}) \Rightarrow \overrightarrow{\neg\phi}(\hat{X}) \mid \overrightarrow{\psi}(\hat{X})$ otherwise

(A5) $\overrightarrow{(\forall x : g)\phi}(\hat{Y}), g(X) \Rightarrow \overrightarrow{\phi}(\hat{Y}, X)$

(A6) $\overrightarrow{(\exists x : g)\phi}(\hat{Y}) \Rightarrow g(X), \overrightarrow{\phi}(\hat{Y}, X)$

(A7) $\overrightarrow{p(\hat{t})}(\hat{T}) \Rightarrow p(\hat{T})$ for predicate p, where terms $\hat{T}$ are substitution results for terms $\hat{t}$

When a fore translation is encountered at (A4), it either completes without exception, and the rules it produces using translations from Fig. 2 are included in the converted theory, or an exception arises and the corresponding rules are discarded, and then the aft version of (A4) is used to obtained the converted rules.

Figure 1: **Aft** *translations*

(F1) $\overleftarrow{\phi}(\hat{X}), \overleftarrow{\psi}(\hat{X}) \Rightarrow \overleftarrow{\phi \wedge \psi}(\hat{X})$

(F2) $\overleftarrow{\phi}(\hat{X}) \Rightarrow \overleftarrow{\phi \vee \psi}(\hat{X})$
$\overleftarrow{\psi}(\hat{X}) \Rightarrow \overleftarrow{\phi \vee \psi}(\hat{X})$

(F3 a) $\overleftarrow{\neg\phi}(\hat{X}) \Rightarrow \overleftarrow{\neg(\phi \wedge \psi)}(\hat{X})$
$\overleftarrow{\neg\psi}(\hat{X}) \Rightarrow \overleftarrow{\neg(\phi \wedge \psi)}(\hat{X})$

b) $\overleftarrow{\neg\phi}(\hat{X}), \overleftarrow{\neg\psi}(\hat{X}) \Rightarrow \overleftarrow{\neg(\phi \vee \psi)}(\hat{X})$

c) $\overleftarrow{\phi} \Rightarrow \overleftarrow{\neg\neg\phi}$

d) $\overleftarrow{\phi}(\hat{X}), \overleftarrow{\neg\psi}(\hat{X}) \Rightarrow \overleftarrow{\neg(\phi \rightarrow \psi)}(\hat{X})$

e) **EXCEPTIONAL** : . . .??? $\Rightarrow \overleftarrow{\neg(\exists \hat{x})\phi}(\hat{Y})$

f) $\overleftarrow{(\exists x)\neg\phi}(\hat{Y}) \Rightarrow \overleftarrow{\neg(\forall \hat{x})\phi}(\hat{Y})$

g) $not\_a(\hat{X}) \Rightarrow \overleftarrow{\neg a(\hat{x})}$ %when a is atomic
$a(\hat{X}),\ not\_a(\hat{X}) \Rightarrow false.$ % add consistency rule
% if $not\_a(\hat{X})$ occurs in consequent of geolog rule

(F4) $\overleftarrow{\neg\phi}(\hat{X}) \Rightarrow \overleftarrow{\phi \rightarrow \psi}(\hat{X})$.
$\overleftarrow{\psi}(\hat{x}) \Rightarrow \overleftarrow{\phi \rightarrow \psi}(\hat{X})$

(F5) **EXCEPTIONAL** : . . .??? $\Rightarrow \overleftarrow{(\forall \hat{x})\phi}(\hat{Y})$

(F6) $g(X), \overleftarrow{\phi}(\hat{Y}, X) \Rightarrow \overleftarrow{(\exists x : g)\phi}(\hat{Y})$

(F7) $p(\hat{T}), g_1(X_1), \ldots, g_k(X_k) \Rightarrow \overleftarrow{p(\hat{t})}(\hat{T})$.
for a predicate p, where terms $\hat{T}$ are substitution results for terms $\hat{t}$, and $< x_1 : g_1, \ldots, x_k : g_k >$ are the variables in $\hat{T}$ which do not occur in $p(\hat{t})$

Figure 2: **Fore** *translations*

The exceptional cases involve the obligation to provide a fore translation for a universally quantified wff. We describe an alternative approach for coherent

conjectures in Section 6. The translation methods are sound. Issues of completeness are discussed in Section 5. We illustrate the required complication in rule (F7) using another example.

Example 4. Consider $(\forall x)(\forall y)(p(x) \rightarrow q(x,y))$ and notice that the antecedent of the formal implication does not contain the universal variable y.

$$true \Rightarrow \overrightarrow{(\forall x)(\forall y)(p(x) \rightarrow q(x,y))}.$$

$$\overrightarrow{(\forall x)(\forall y)(p(x) \rightarrow q(x,y))}, dom(X), dom(Y) \Rightarrow \overrightarrow{p(x) \rightarrow q(x,y)}(X,Y).$$

$$\overrightarrow{p(x) \rightarrow q(x,y)}(X,Y), \overleftarrow{p(x)}(X,Y) \Rightarrow \overrightarrow{q(x,y)}(X,Y).$$

$\star$ $p(X), dom(Y) \Rightarrow \overleftarrow{p(x)}(X,Y).$

$$\overrightarrow{q(x,y)}(X,Y) \Rightarrow q(x,y).$$

Notice at $\star$ how $dom(Y)$ is needed in the antecedent to universally quantify Y. Otherwise the occurrence in the consequent would be interpreted as an existential variable in the consequernt of the geolog rule. See Appendix B.4 for the results of automated translation.

Coherence Theorem. *A terse fore translation of a FOL wff already having coherent form preserves its coherent form.*

Examples 3 and 4 illustrate the theorem well.

4 Domain constants and function closure axioms

When *colog* translates the following (text form) coFOL theory

```
(q(f(a)) => goal).
p(a).
![X]:q(X).
```

we get the *colog* theory

```
q(f(a)) => goal.
true => p(a).
true => dom(a).  % domain axiom
dom(X) => q(X).
dom(X) => dom(f(X)). % function closure axiom
```

Fig. 3 illustrates a *colog* proof for the program.

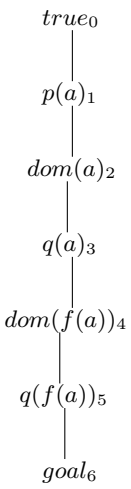

Figure 3: Proof tree

It is clear that the domain axiom and the function closure axiom (in the *colog* translation) are required to effect the proof of the query. The *colog* translator automatically supplies them when it translates axioms or conjecture that contain symbolic constants or functions. The **Basic Completeness Theorem** in the next section requires them.

The *colog* translator detects constant c *as if* it had been explicitly declared in a FOL program as

$$dom(c). \tag{18}$$

and it detects a function symbol f and presumes that the intended closure is

$$(\forall x_1, \ldots, x_n)\ dom(f(x_1, \ldots, x_n)). \tag{19}$$

for any n-ary function symbol f appearing in the theory.

So, for example, if a FOL program contains a literal $p(f(a, b), c)$, *colog* will presume that a, b, c are constants and that f is a binary function symbol of arity 2.

In effect, *colog* computes *colog* code for computing the Herbrand universe using the domain predicate 'dom' and the constants and functions in the FOL program that it translates.

For domains other than *dom* the programmer must supply function closures. For example, the axioms

$$int(3) \tag{20}$$

$$(\forall x : int, y : int)\ int(x + y) \tag{21}$$

would express the closure for the symbolic domain *int* under symbolic addition + and that 3 is in domain *int*,

```
int(3).
![X:int,Y:int]: int({x+y}).
```

These axioms are *not* automatically generated and must be supplied by the FOL programmer.

5 Completeness issues

The worked examples so far are intended to illustrate the mechanics of fore-and-aft translation when applied to individual FOL wffs. In this section we will more carefully describe the translation of FOL theories into geolog theories and the application of a coherent logic prover to the resulting geolog theory.

Recall that a FOL theory is a finite sequence of wffs. The translator for the *colog* system translates the FOL theory wffs in order and adds the resulting geolog rules to the geolog theory. Any consistency rules are added at the beginning of the geolog theory and the geolog form of the dom rules 9 are added at the end of the geolog theory.

A FOL conjecture is a wff that is typically the first wff in the FOL theory. A *goal form conjecture* is expressed in the following manner:

$$\gamma \rightarrow goal \qquad \% \ goal \ conjecture \tag{22}$$

where the intent is to find a *positive* proof for the wff γ. A *refutation form conjecture* has the form

$$\neg\gamma \qquad \% \ refuted \ conjecture \tag{23}$$

where the intent is to find a refutation of $\neg\gamma$ (proof for γ).

A conjecture is a wff which is proposed as a logical consequence of the other wffs in the FOL theory, for which a proof is desired. If no proof is attainable, the programmer would like *colog* to compute a model which serves to *disprove* the conjecture. In general, FOL is not decideable, so one must be content with limited completeness results.

Basic Completeness Theorem. *Suppose that $\gamma, \alpha_1, \ldots, \alpha_n$ is a FOL program and that γ is a logical consequence of $\alpha_1, \ldots, \alpha_n$. Then the fore-and-aft translation of $\neg\gamma, \alpha_1, \ldots, \alpha_n$ into a geolog theory (including all of the domain axioms and function closure axioms) has a refutation proof: a saturated geolog tree having $false$ leaves.*

The theorem remains true when all of the wffs are positive (contain no logical negation) and the conjecture has the special form (22). Notice, however, that negation will be introduced if γ is a universal statement. See the next section for a description of coherent conjecture, which do not require injecting negative forms.

Theories containing logical negation do not necessarily have proofs by Skolem Machine for positive form conjectures. For example, the FOL tautology $p \vee \neg p$ has no coherent proof when posed as a positive conjecture $p \vee \neg p \Rightarrow goal$.

There may be a close association between intuitionistic proof and positive form conjectures and proof by Skolem machine.

6 Coherent goals

Special forms of γ in (22) are called *coherent* goals:

$$\gamma = (\forall \hat{x})\alpha \tag{24}$$

$$\gamma = (\forall \hat{x})(\exists \hat{z_1})\zeta_1 \vee \ldots \vee (\exists \hat{z_k})\zeta_k \tag{25}$$

$$\gamma = (\forall \hat{x})[\alpha \rightarrow (\exists \hat{z_1})\zeta_1 \vee \ldots \vee (\exists \hat{z_k})\zeta_k] \tag{26}$$

where α, ζ_i are conjunctions of atomic wffs, $(k \geq 1)$ and possibly there are no universal variables.

In Reference [2] we defined *coherent queries* to have the form (25) (but with no universal variables).

Notice that fore translation of $\gamma \rightarrow goal$ will raise the exception if there are universal variables. We describe here an alternate translation scheme for these coherent goals. The universal variables in the quantifier $(\forall \hat{x})$ are replaced by unique Herbrand constants. This approach corresponds to the mathematical tactic that says that, for example, if $\phi(@x)$ can be derived, for a general value $@x$ of the variable x, then $(\forall x)\phi(x)$ is also derived. The $@x$ *constant* stands for *a general x*. See Reference [3] for a discussion of this technique in a more general logical context. Each of the forms (24), (25), (26) is converted to a new goal form.

Let us suppose that $\hat{x} = x_1, \ldots, x_k$ and that $@x_1, \ldots, @x_k (k \geq 0)$ are unique constants that can replace the variables. For any unquantified wff β containing those variables, let β' be the same wff but with the variables replaced by the corresponding constants. The alternative translation schemes for conjectures $\gamma \rightarrow goal$ are as follows:

The case of (24) is translated as if it were $\alpha' \rightarrow goal$.
The case of (25) is translated as if it were $\exists \hat{z_1})\zeta_1' \vee \ldots \vee (\exists \hat{z_k})\zeta_k' \rightarrow goal$.
And the case of (26) is translated as if it were two axioms.

$$\alpha' \tag{27}$$

$$\exists \hat{z_1})\zeta_1' \vee \ldots \vee (\exists \hat{z_k})\zeta_k' \rightarrow goal \tag{28}$$

Example 5 provides an excellent example illustrating the translation of an interesting mathematical theory. We use the last case (26) for translating the goal. The original formulation of this problem for coherent logic was described by Marc Bezem: *The Diamond Property (DP) in rewriting theory states that, if x rewrites to both y and z, then the latter two rewrite both to some u (all in one step). We prove that if some rewrite relation satisfies DP, then its reflexive closure also satisfies DP. "Reflexive closure" means adding point loops to every point.* Here, we provide a FOL formulation for DP.

Example 5.

$$((\forall x, y, z)(re(x, y) \wedge re(x, z) \rightarrow (\exists w)(re(y, w) \wedge re(z, w))) \rightarrow goal$$

$$(\forall x)X = X.$$

$$(\forall x, y)(X = Y \rightarrow Y = X).$$

$$(\forall x, y, z)((X = Y \wedge Y = Z) \rightarrow X = Z).$$

$$(\forall x, y, z)((X = Y \wedge re(Y, Z)) \rightarrow re(X, Z)).$$

$$(\forall x, y)(X = Y \rightarrow re(X, Y)).$$

$$(\forall x, y)(r(X, Y) \rightarrow re(X, Y)).$$

$$(\forall x, y)(re(X, Y) \rightarrow (X = Y \vee r(X, Y))).$$

$$(\forall x, y, z)((r(X, Y) \wedge r(X, Z)) => (\exists u)(r(Y, U) \wedge r(Z, U))).$$

Notice the coherent conjecture (the *goal* rule). See Appendix B.5 for the FOL version, an automated translation and a nice proof tree generated by *colog* .

Now, coherent conjectures for coherent cofol programs are translated to *colog* theories with proofs if the FOL program is logically valid. This result follows (more or less directly) from the completeness theorems in Reference [2].

7 Related topics

7.1 optimizations for negative normal tranlations

Chapter 1 of A. Polonsky's thesis [4] studies translations from FOL to geolog which are derived from the negation normal forms of the axioms and conjecture. That approach generates collections of translated theories which contain various combinations of contrapositives of the initial negative forms. Heuristics are used to generate target geolog theories, with the intent of generating geolog theories having smaller geolog trees (shorter proofs). The *colog* prover has refinements that were designed in order to improve the proof performance for these negative normal translations. It is good news that many of the refinements work for the fore-and-aft translations also. In particular, *colog* implements something called QEDF search, which is a particular search strategy for a Skolem Machine, see Reference [**?**].

7.2 generalized Herbrandization for universals

It may possible to specify fore translations for universally quantified forms in a more general way, using some tricks. The general approach requires introducing herbrand *functions* rather than constants as explained in §6. After some initial experimentation, this approach has been abandoned for the present. It seems that the new schemes make some translations too obscure, even though this would obviate the need for the exceptional cases for translations.

7.3 typed first-order logic

The stated conventions for keyed quantifiers (2) and (4) obviate the need to use special logical semantics for interpretations of FOL having keyed quantifiers. Thus, when the Basic Completeness Theorem refers to "logical consequence" that means the classical concept of logical consequence, not a modified interpretation regimen where the keys are interpreted using type theory (or set theory) artifacts. The intended interpretations are classical first-order logic interpretations or, after translation, constructive geolog trees.

References

[1] M. Bezem and T. Coquand, Automating Coherent Logic. In G. Sutcliffe and A. Voronkov, editors, *Proc. LPAR-12*, LNCS 3825, 2005, pages 246-260.

[2] John Fisher and Marc Bezem, Skolem Machines, *Fundamenta Informaticae*, 91 (1) 2009, pp.79-103.

[3] *Wikipedia* article on Herbrandization, `http://en.wikipedia.org/wiki/Herbrandization`.

[4] Andrew Polonsky, *Proofs, Types, and Lambda Calculus*, Ph.D. thesis, University of Bergen, Norway, December 4, 2010.

[5] The TPTP Problem Library for Automated Theorem Proving `http://www.cs.miami.edu/~tptp/`

[6] Polymorphic TPTP TFF website `https://sites.google.com/site/polymorphictptptff/home`

A Guide to *FOL* input notation used by *colog*

Text input *FOL* syntax resembles that of the TPTP logic language. Logical quantifiers use the bracket notation, e.g. `![X:t]:`, `![X]:` or `?[X:t]:` and several variables may be quantified in one expression, e.g., `![X:t, S:'seq<t>']:` . The universal quantifier designator is `!` and the existential quantifier designator is `?`.

FOL variables need to start with a capital letter, e,g, `Var`.

The logical connectives are `&, |, , =>, and <=>` (and, or, not, if-then, if-and-only-if, resp.) which are written infix. Subformulas formed using logical connectors require braces `{...}` or parens `(...)`.

```
{{p & q} => {r | s}}.
((p & q) => {r | s}).  // mix {..} or (..) for logic
![X]:{ p(X) => ?[Y]:{ q(Y) & r(Y) }.
```

However, both `&` and `|` can have multiple juncts. For example, {`p & q & r`} is read and stored as {`p & {q & r}`} .

Predicates and functors can be written in the conventional manner, such as `p(X,a)`. Functor names start with a lower-case letter. A functor expression is read in context. For example,

```
{ p(f(a),b) & q(a) }
```

is read so as to make p and q predicates, f is a function, and f(a) is a term. Functors may have names that are quoted using either single quotes or double quotes (as in `'list(t)'` in the following sample theory).

Functors can also start with `$`. This allows the *colog* reader to translate functors intended for evaluation in later versions of colog. However, at the present time, *colog* does not implement any evaluation and $functors are treated symbolically, like other functors.

A *FOL* theory is a finite sequence of wffs, each of which is terminate with a period. Wffs may extend over several lines. The following example illustrates with a small theory of lists.

```
// goal test
{ member(a,add(c,add(b,add(a,nil)))) => goal }.

// data
{ t(a) & t(b) & t(c) }.

// constructor logic
'list(t)'(nil).
![X:t]: 'list(t)'(add(X,nil).
![X:t,R:'list(t)']: 'list(t)'(add(X,R)).

// accessor logic
```

```
![X:t,R:'list(t)']: member(X,add(X,R)).
![X:t,Y:t,R:'list(t)']: { member(X,R) => member(X,add(Y,R)) }.

// empty list
empty(nil).
```

FOL recognizes several special infix predicates: `= != < <= > >= == !==`. These are only symbols, however. For example to use `=` as an equality will require that the *FOL* theory contain the usual equality axions.

```
![X]: X=X.
![X,Y]: {X=Y => Y=X}.
![X,Y,Z]: { {X=Y & Y=Z} => X=Z }.
```

FOL recognizes infix binary operators `* + / - #`. The infix `-` must be followed by a space. There is one symbolic prefix operator `-` which should not have a space following it. (The former might indicate a *subtraction* and the latter a *negative*.)

Compound terms constructed with operators must be fully parenthesized (no braces) , as in

```
?[X]: X = ((f(a) * c) + b).
```

TPTP [5] first-order logic theories (THP, etc.) can be easily translated to *FOL* theory form. Typed TPTP theories (TFF) require special handling of type declarations for translation to *FOL* . CNF theories can be translated as sequences of *FOL* disjunction wffs, but this often leads to a *FOL* theory which is inefficient for solving, so it is generally preferable to translate the first-order logical theory to *FOL* rather than its CNF translation.

B The worked *FOL* examples auto-translated to *colog*

The text input formats for FOL are similar to those used for the TPTP system.

B.1 Example 1 automated

FOL input:

```
(((a | b) => c) => d).
```

Verbose translation:

```
true => '>>{(((a | b) => c) => d)}'.
'>>{(((a | b) => c) => d)}', '<<{((a | b) => c)}' => d.
'<<~{(a | b)}' => '<<{((a | b) => c)}'.
'<<{c}' => '<<{((a | b) => c)}'.
'<<~{a}', '<<~{b}' => '<<~{(a | b)}'.
not_a => '<<~{a}'.
not_b => '<<~{b}'.
c => '<<{c}'.
```

Terse translation:

```
not_a, not_b => d.
c => d.
```

The terse forms are obtained from the verbose forms using a combination of fore and aft "folding" of the geolog rules. It is the terse forms of translation that are typically used as the finished translation of the FOL wffs.

Colog only adds consistency geolog rules for predicates having a negative occurrence of the predicate in some resulting geolog rule, so translation of this simple input wff yields no consistency axioms.

B.2 Example 2 automated

FOL input:

```
![X:a]:(~p(X) => ?[Y:b,Z:c]:q(X,Y,Z)).
```

Verbose translation:

```
true => '>>{![X:a]:(~p(X) => ?[Y:b,Z:c]:q(X,Y,Z))}'.
'>>{![X:a]:(~p(X) => ?[Y:b,Z:c]:q(X,Y,Z))}', a(X) =>
      '>>{(~p(X) => ?[Y:b,Z:c]:q(X,Y,Z))}'(X).
'>>{(~p(X) => ?[Y:b,Z:c]:q(X,Y,Z))}'(X), '<<{~p(X)}'(X) =>
      '>>{?[Y:b,Z:c]:q(X,Y,Z)}'(X).
'<<~{p(X)}'(X) => '<<{~p(X)}'(X).
not_p(X), a(X) => '<<~{p(X)}'(X).
```

```
'>>{?[Y:b,Z:c]:q(X,Y,Z)}'(X) => b(Y), c(Z), q(X,Y,Z).
a(Z) => dom(Z).
b(Z) => dom(Z).
c(Z) => dom(Z).
```

Terse translation:

```
a(X), not_p(X) => b(Y), c(Z), q(X,Y,Z).
a(Z) => dom(Z).
b(Z) => dom(Z).
c(Z) => dom(Z).
```

We see that the translator does indeed preserve the coherence of the FOL wff.

B.3 Example 3 automated

FOL input:

```
![X]:(![Y]:p(X,Y) => q(X)).  // Note universal antecedent of =>
```

Verbose translation:

```
p(X0,X1), not_p(X0,X1) => false.
true => '>>{![X:dom]:(![Y:dom]:p(X,Y) => q(X))}'.
'>>{![X:dom]:(![Y:dom]:p(X,Y) => q(X))}', dom(X) =>
      '>>{(![Y:dom]:p(X,Y) => q(X))}'(X).
'>>{(![Y:dom]:p(X,Y) => q(X))}'(X) =>
      '>>~{![Y:dom]:p(X,Y)}'(X) | q(X).
'>>~{![Y:dom]:p(X,Y)}'(X) => '>>{?[Y:dom]:~p(X,Y)}'(X).
'>>{?[Y:dom]:~p(X,Y)}'(X) => dom(Y), '>>{~p(X,Y)}'(X,Y).
'>>{~p(X,Y)}'(X,Y) => not_p(X,Y).
```

Terse translation:

```
p(X0,X1), not_p(X0,X1) => false.
dom(X) => dom(Y), not_p(X,Y) | q(X).
```

Notice that this translation does require addition of a consistency rule, since the negation is introduced into the consequent of the first (terse) geolog rule.

B.4 Example 4 automated

FOL input:

```
![X,Y]:(p(X) => q(X,Y)).
```

Verbose translation:

```
true => '>{![X:dom,Y:dom]:(p(X) => q(X,Y))}'.
'>{![X:dom,Y:dom]:(p(X) => q(X,Y))}', dom(X), dom(Y) =>
      '>{(p(X) => q(X,Y))}'(X,Y).
'>{(p(X) => q(X,Y))}'(X,Y), '<{p(X)}'(X,Y) => q(X,Y).
p(X), dom(X), dom(Y) => '<{p(X)}'(X,Y).
```

Terse translation:

```
dom(Y), p(X) => q(X,Y).
```

B.5 Example 5 automated

The *FOL* program (file dpe.fol):

```
// positive-form conjecture
(![X,Y,Z]:((re(X,Y) & re(X,Z)) => ?[W]:(re(Y,W) & re(Z,W)) ) => goal).
// axioms
![X]:X = X.
![X,Y]:(X=Y => Y=X).
![X,Y,Z]:((X=Y & Y=Z) => X=Z).
![X,Y,Z]:((X=Y & re(Y,Z)) => re(X,Z)).
![X,Y]:(X=Y => re(X,Y)).
![X,Y]:(r(X,Y) => re(X,Y)).
![X,Y]:(re(X,Y) => (X=Y |r(X,Y))).
![X,Y,Z]:((r(X,Y) & r(X,Z)) => ?[U]:(r(Y,U) & r(Z,U))).
```

Terse (coherent-form, terse) translation to *colog* :

```
/*1*/ re(@Y,W), re(@Z,W) => goal.
/*2*/ true => dom(@Z), dom(@Y), dom(@X).
/*3*/ true => re(@X,@Y), re(@X,@Z).
/*4*/ dom(X) => X=X.
/*5*/ X=Y => Y=X.
/*6*/ X=Y, Y=Z => X=Z.
/*7*/ X=Y, re(Y,Z) => re(X,Z).
/*8*/ X=Y => re(X,Y).
/*9*/ r(X,Y) => re(X,Y).
/*10*/ re(X,Y) => X=Y | r(X,Y).
/*11*/ r(X,Y), r(X,Z) => dom(U), r(Y,U), r(Z,U). /* Existential */
```

Notice that the *colog* translator automatically generates the Herbrand constants for the *colog* theory. (A verbose translation yields 48 rules.)

The *colog* prover automatically generates the proof tree (and the Latex picture) displayed in Fig.4.

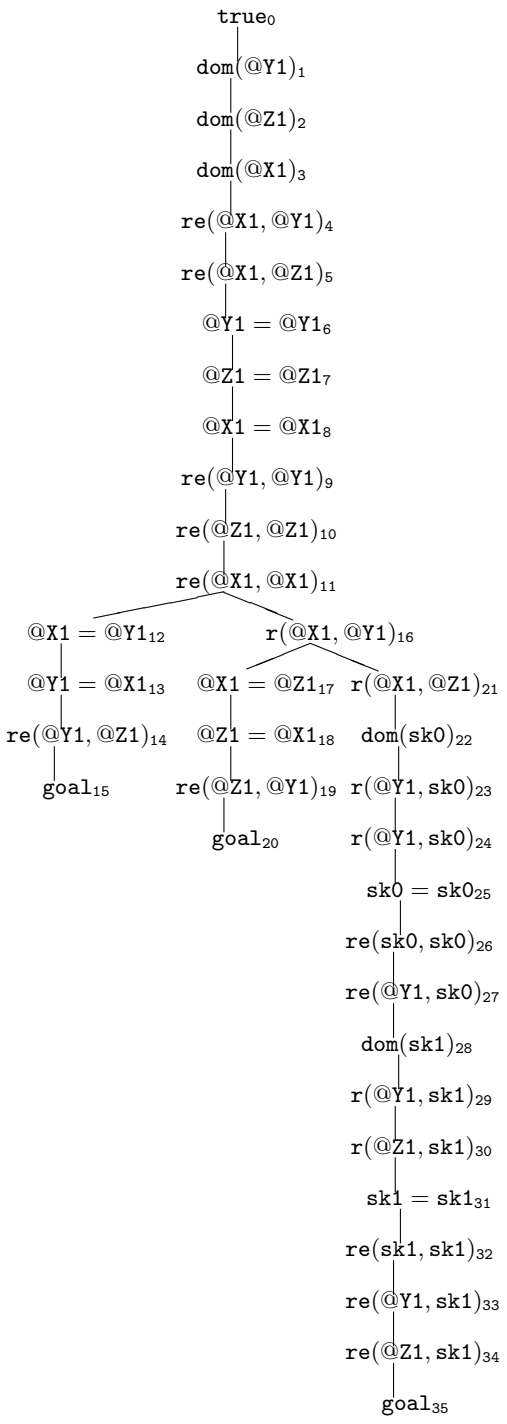

Figure 4: *colog* proof tree for dpe.fol