A Tower Architecture for Meta-Level Inference Systems Based on Omega-Ordered Horn Theories

Pierre E. Bonzon[1]

TR-95-002

January 1995

Abstract

We present a simple meta-level inference system based on a non-ground representation of both base and meta-knowledge given under the form of omega-ordered Horn theories. Processing is done via an extension of the traditional "vanilla" interpreter for logic programs, whose novel lifting mechanism allows one to hop up and down the hierarchy of theories. The resulting computational system ressembles very much the tower architecture defined for functional programming. While lifting does prevent infinite recursion, successful termination depends on the actual ordering of theories. At the end, this situation amounts to facing yet another, meta-meta-level search problem.

The expressive power of this system is illustrated with the solutions to various problems from the current litterature, including the 3 wise men problem. It looks like a reasonable assumption to hypothesize that most (if not all) specialized reasoning performed under the label of "proofs in context" can be formulated within this system.

[1]Author's permanent address: HEC, University of Lausanne, 1015 Switzerland
pbonzon@inforge.unil.ch

1. Introduction

The quest for the Grail in AI, i.e. computer equipped with common sense, has been on for quite some time, but has failed yet to come up with significant results. Numerous formal systems for performing the kind of reasoning presumably required for the task have been proposed. Although some of them have been implemented as computer programs, they are not generally available as user-friendly subsystems. Futhermore, as they now stand, they would not be able to cooperate with other subsystems. These shortcomings, we think, do represent one of the major reasons for the present disappointing state of affairs.

The requirement for effective system integration is particularly critical at a time when vast amounts of computerized common sense knowledge may soon be widely available (if we take for granted that projects like CYC will eventually mature to the point of becoming viable commercial ventures). Without matching common sense reasoning capabilities, we believe, computer systems will not be able to take full advantage of this new breed of data bases.

What is then needed, we conclude, is a kind of a general architecture or facility (not another language, not another formal system) for implementing advanced knowledge representation and processing models. This facility should be provided on *existing software platforms* and be easily interfaced with available knowledge sources.

As a first step toward this goal, we have been looking for a prototyping system that could allow one to easily transcribe and experiment with models of beliefs [10], reflective [14] or context dependent [11] reasoning. The kind of systems we have in mind is best described by the following lines taken from A. Hutchinson's recent book on *Algorithmic Learning* [9]:

> " A clever learning program which can work at meta-levels might include a procedure for generating successive meta-levels. Whenever something within the program suggests that its present system is inadequate, the program could hop up a level, define an extension of its current object system, and then hop down again."

On the theoretical side, F. Gunchiglia & al. [6] , working on a possible foundation for meta-reasoning in the field of AI, introduced the concept of a *hierarchical meta-logic*. Given any axiomatized object theory expressed in propositional logic, this formal system produces theorems both in this base theory and in a related meta-theory. To quote from them, "the assumption of uniformity throughout the hierarchy leads to the possibility of capturing the whole infinite tower of theories with in a single all-encompassing formal system schematic on the levels".

Introduced in the early 80's by B.C. Smith [13] in the context of functional programming, the concept of a tower architecture has not proved so far to lead to significant practical applications. As Hutchinson points out, "considered as a meta-system, it is not very expressive" , and, as such, does not allow to tackle significant problems. To overcome this limitation, Gunchiglia & al. first define a system where the property expressed at each meta level is theoremhood. They allow non logical axioms to be added at appropriate levels, ending up with a natural deduction system for the whole hierarchy. Finally, they prove that the resulting formal system has a proof theory which is sound and complete with respect to a particular semantics capturing the desired relationship between object and meta theory.

An essential hypothesis made throughout these developments, allowing in particular to ensure consistency in theories *from a certain level up*, is that the conclusion from a derivation made at a certain level never depends on an assumption relating to a lower level. As it did turn out, our own developments (which do enforce this very hypothesis) very much look like a Prolog implementation of the above theoretical ideas, extended to the *first order* case. We had indeed decided at once that our investigation would be based on Prolog, which readily allows one to quickly develop and test various meta-level schemes using the non-ground representation [8]. At the same time, it provides a modern interface to leading commercial software applications (were these going to be needed for later developments that have not been yet attempted).

The good news are that we have been able to do this in a rather straightforward way, coming up with a 15 lines interpreter (not counting the set primitives and system declarations) for a simple meta-level inference system we call *lifted resolution of omega-ordered Horn theories*. The bad news are that at the end, we are left facing a meta-meta-level *search problem* concerning the proper ordering of the various Horn meta-theories defining search control for object domains.

The rest of this paper is organized as follows. In the next section, we describe a simple meta-level inference system through its corresponding Prolog interpreter. In a subsequent section, we relate this system to the usual approaches in the field, and present our solutions to selected problems from the current litterature (including the 3 wise men problem). In the last section, we stress some of the limitations of our own work, and discuss possible extensions to overcome these limitations.

2.- A Simple MetaLevel Inference System and its Prolog Interpreter

2.1 A Prolog Interpreter for Flat Horn Theories

Let us first introduce a simple variant of the traditional Prolog interpreter for logic programs, intended to deal with named definite logic programs (or flat Horn theories) using a non-ground representation. Toward this end, we shall use the following syntactic conventions:

- *named* definite logic programs, or flat Horn theories, will be represented as Prolog structures built with the binary operator ": ", and comprising an *atom* as first operand, standing for the theory's name, and a list of *assertions* as second operand, standing for the theory's definition;

- *assertions* themselves can be either *facts*, represented by single terms , or *rules*, represented by terms built with the binary implication operator " <- ", and comprising a single term as rule *head* and either single or multiple terms standing as rule *conditions*, with multiple conditions enclosed in parentheses and linked by conjunctive "," or disjunctive ";" operators (which both have a higher precedence than the implication operator).

Example:

```
theory1:[us(clinton),
         non binational(clinton),
         european(X) <- swiss(X),
         non european(X) <- (us(X), non binational(X))].
```

Note the use, in the above theory, of the unary operator "non" for representing true negation, in order to avoid interference with Prolog's "not" operator implementing negation based on the closed world assumption.

A Prolog interpreter for solving goals prefixed with the name of a theory can readily be written as follows:

```
solve(Theory:Goal) :- Theory:T,
                      interpret(Goal,T).

interpret((A;B),T):-!,(interpret(A,T);
                       interpret(B,T)).

interpret((A,B),T):-!,interpret(A,T),
                      interpret(B,T).

interpret(A,_):-system(A),!,call(A).

interpret(A,T):- match(A,T);
                 match(A<-C,T),
                 interpret(C,T).
```

where `match (X,T)` retrieves the assertions in the list T which can be unified with X

Example:

```
?- solve(theory1: non european(X)).
X=clinton;

no
```

Without any axiomatic treatment of negation, this interpreter cannot deduce further logical consequences of the above theory, such as the fact that us president clinton, being non european, is also necessary non swiss.

2.2 A Prolog Interpreter for Omega-ordered Horn Theories

As a first extension, let us now consider theories made up of *ordered sets* of flat theories, i.e. comprising lists of lists of assertions with possibly many implication operators associating to the left , thus defining "higher order" conditions for meta-rules. We call the resulting construction *omega-ordered Horn theories.*

Example:

```
theory2:[[us(clinton),
         non binational(clinton),
         european(X) <- swiss(X),
         non european(X) <- (us(X), non binational(X))],

        [non P(X)  <-  (Q(X)<-P(X),  non Q(X))]].
```

Following [6], the first theory corresponds to an *object* theory, and the second one (comprising just one meta-rule, i.e. a simple case of negation introduction), corresponds to a *meta-theory* intended to derive conclusions that otherwise would not be obtainable.

Our interpreter can be extended to do this as follows:

```
interpret(A,[T1|Tn]):- match(A,T1);
                       lift(A<-C,[T1|Tn]),
                       interpret(C,[T1|Tn]).

lift(A,[T1,T2|Tn]):- append(T1,T2,T12),
                     interpret(A,[T12|Tn]).

?-  solve(theory2: non  swiss(X)).
X=clinton;

no
```

A few words of explanation about the "lifting" mechanism just introduced, and which stands at the heart of our interpreter's extension, are in order here. After failing to match the initial query with an assertion in the object theory, the interpreter will call on lift to first merge the object theory with the meta-theory (using append), and then go through a recursive call to interpret. After matching the lifted query (i.e. `non swiss(X) <- C`) with the head of the meta-rule, execution of the initial query will resume and go on interpreting the corresponding meta-conditions, i.e. `(european(X) <- swiss(X), non european(X))`. While the first of these conditions will follow a direct matching with the object theory, the second one will in turn require a lifting, leading to a match of the second lifted query (i.e. `non european(X) <- C`) with the lifted object theory.

The following is another simple example embodying a case of "heterogeneous knowledge compilation" in a system operating with a decision theoretic formulation, as advocated by S. Russel [12]. Note that in this case, access to the lifted object theory is required in order to satisfy the conditions of the meta-rule for `opportunity` (following two consecutives lifting). Note also the versatility allowed in the grouping of assertions within theories , which are ordered here according to their complexity .

```
trade:[[cargo(oil,venezuela,100000),
        price(oil,venezuela,10),
        price(oil,us,12),
        shipping(oil,venezuela,us,0.5),
        utility(profit(_,_)),
        action(contract(_,_,_,_))],

       [contract(X,T,P,O) <- (cargo(X,O,T),price(X,O,P)),

        profit(F,D) <- (contract(X,T,P,O),
                        price(X,D,M),
                        shipping(X,O,D,S),
                        F is T*(M-P-S),
                        F >0)],


       [opportunity((Action,Utility))
               <- Conditions
                     <-  (action(Action),
                          utility(Utility),
                          Action <- ActionCond,
                          Utility <- UtilityCond,
                          compile(Action <- ActionCond,
                                  Utility <- UtilityCond,
                                  Conditions))]].

?- solve(trade:opportunity((X,Y))).
X = contract(oil,100000,10,venezuela) ,
Y = profit(150000,us) ;

no
```

While the unspecified compiling procedure which produces `Conditions` simply involves the partial evaluation of possible actions (here, just the `contract` assertion) with respect to given utilities (here, the `profit` assertion) , the overall computation amounts to *learning and applying* the rule given in [12], i.e " if the current US market price of crude oil is M then buying a cargo of T tons in Venezuela at price P will yield a net profit f(M,T,P)" for some known f.

A careful analysis of our preceeding examples leads to the conclusion that a flat theory comprising all assertions, together with a traditional interpreter, will do the job (in our last example, the derived `Conditions` would then have to be relegated at the end of the meta-rule). However, the pattern "guarding" the lifting procedure (i.e. `[T1,T2|Tn]`), associated with the merged pattern commanding the recursive call to the interpreter (i.e. `[T12|Tn]`), plays another essential role, i.e. *prevents infinite recursion*. This will prove necessary in the more complex examples yet to come.

As a conclusion, our lifting process encompasses both the matching process for retrieving rules from theories and the merging of object and meta-theories. By itself, it does embody the one and only extension needed to answer queries against omega-ordered theories. We call the resulting interpretation process *lifted resolution of omega-ordered Horn theories*.

3. Meta-level architectures and application models: a review

3.1 Meta-level architectures

Meta-level architectures, as described extensively in [7], are by now well defined systems enjoying common properties (foremost, the explicit representation of control knowledge). At the same time, they can be further classified according to specific characteristics (such as the "locus of action", the linguistic relation between levels, and so on).

By the obvious virtues of its associated interpreter, the system introduced in the previous section falls into the category of *monolingual* [2] , *declarative, meta-level inference system*. As such, it does not require the use of a naming relation (as opposed to similar, *amalgated* systems like Alloy[2]). As argued in [7], nothing prevents the use of a unique language, both at the object and meta-levels, the only requirement being "that the two languages, although of the same type, are syntactically distinct". The layered structure of the tower of omega-ordered theories, we believe, does account for this requirement.

The relation of our system with a sound and complete formal system (i.e the hierarchical meta-logic of [6]), has already been stressed in the introduction. Standing on top of an incomplete inference system, i.e. Prolog (for a discussion of other sources of incompletness, see below), our construction is itself necessarily *incomplete*. As for soundness, we refer to the results obtained in [6], which pertains to the propositional case, and to the perspectives given there for the first order case.

N.B. Logical completeness and soundness, as accounted for here (i.e. relating a given semantics and proof theory), are not to be confused with the concepts of combinatorial completness and soundness. These later concepts are used in particular in [7] to classify meta-level architectures whose primary purpose is to "prune parts of the object level search space". Combinatorial soundness, being the property enjoyed by conservative extensions of object theories (which do not allow new theorems to be added by the meta-theory), is required for such a specific task. It is not however a desirable property of meta-level inference systems intended, as described in our introduction, to extend the results of an object theory.

Being based on the *non-ground* representation of object knowledge within meta-knowledge, our system stands in strong contrast to the more versatile approach using the *ground* representation pioneered in [3]. This is not surprising, since the goals systems based on this later approach usually pursue, " namely the development of software tools (the metaprograms) that manipulate other programs (the object programs) as data, such as debuggers, compilers, program transformers, etc" [4] , are quite different too. When they aim, as we do, at representing and processing application knowledge, systems using the non ground representation, such as Alloy [2] and Reflective Prolog [5], seem to be unnecessarily cluttered by naming relations.

[2] [8] refers to this property as the *non-ground* representation

3.2 Application models

As all architectures, our simple system represents nothing more than an empty shell, i.e. a tool that should possibly be used to model situations and solve practical problems. Potential applications which have already been extensively studied in the literature include, among others, the modelization of *agents beliefs* including self referential statements about truth or knowledge, which arise naturally in common sense reasoning.

As a tentative assessment of the expressive power enjoyed by omega-ordered Horn theories, we present the formulation, within our formalism, of recently published solutions to problems involving such modelizations. This will allow us, at the same time, to relate our proposal to competing formalisms.

3.2.1 *Encoding of private beliefs*

The first problem involves the modelization of an agent's *private* beliefs. The original formulation goes as follows [2]:

"...Mary has the following five basic beliefs: anyone who she believes is thinking they like her, she finds charming; all men think all women are irresistible; everyone think that Mary is a woman; John is a man; everyone think that if they are irresistible then everyone likes them".

The solution given in [2] makes use of a special predicate Name " intended to mean that an atomic sentence Name(t,<t>) can be proved for any term t in any theory". A specialized proof system is then needed to carry the "interleaving of computations at different meta-levels". Our formulation, which does not resort to such a special predicate, otherwise closely follows [2]'s original solution, as far as the object theory is concerned; apart from the two meta-assertions that are required in order to account for the nesting of beliefs, there is no need for a specialized proof theory. It goes as follows:

```
charm:[[believe(mary, believe(X,female(mary))),
        believe(mary, male(john)),
        believe(mary, (charming(X) <- believe(X,likes(X,mary)))),
        believe(mary, believe(X,(likes(Y,Z) <- irresistible(Z)))),
        believe(mary, (believe(X,(irresistible(F) <- female(F)))
                                                  <- male(X)))],

       [believe(X,P)    <- believe(X,Q)  <-   believe(X, P<-Q)],

       [believe(X, believe(Y,P)) <- believe(X,believe(Y,Q))
                                <- believe(X, believe(Y, P<-Q))],

       []].

/* N.B. last empty theory needed for lifting space see discussion below */

?- solve(charm:believe(X,charming(Y))).
X = mary ,
Y = john ;

no
```

3.2.2 Encoding of common knowledge

Our next problem, i.e. the so-called " three wise men problem", is often regarded as a benchmark in the field of meta-reasoning, so we will refrain from restating it here. Among the numerous proposals for solving this problem, we have chosen to try and reproduce the elegant solution given by G. Attardi and M. Simi. In this solution, "common knowledge is grouped in a single theory and lifting rules are provided for each agent to access it" [1]; the required proof is then carried out *by hand*, following a well defined process of natural deduction in and out of nested contexts.

Whenever possible, our formulation also closely follows Attardi & Simi's original solution. Our axioms for getting in and out of contexts are transcriptions of their own axioms. In order to make up for the limitations of lifted resolution of omega-ordered Horn theories with regard to natural deduction in full first order theories, we introduce for the following fixes:

- disjunctive facts are broken down into corresponding implications
- double nested negation introduction (or reductio ad absurdum) are rendered through a meta-rule relying on a simple interpreter's extension allowing to carry out a proof under a given assumption

This straightforward interpreter's extension is as follows:

```
interpret((A=>B),[T1|Tn]):-!,append(A,T1,AT1),
                              interpret(B,[AT1|Tn]).
```

Common knowledge and meta-rules for accessing common knowledge and defining reductio ad absurdum can then be expressed as follows:

```
wise:[[answer(X,Answer) <- (possible(X,Answer), believe(X,Answer)),

       possible(1,white1), possible(1, non white1),
       possible(2,white2), possible(2, non white2),
       possible(3,white3), possible(3, non white3),

       believe(X, non believe(2,white2)),
       believe(X ,non believe(1,white1)),

       believe(X, (white1 <- (non white2, non white3))),
       believe(X, (white2 <- (non white3, non white1))),
       believe(X, (white3 <- (non white1, non white2))),

       believe(X,believe(2, white1)) <- white1,
       believe(X,believe(3, white1)) <- white1,
       believe(X,believe(2, non white1)) <- non white1,
       believe(X,believe(3, non white1)) <- non white1,
       believe(X,believe(3, white2)) <- white2,
       believe(X,believe(1, white2)) <- white2,
       believe(X,believe(3, non white2)) <- non white2,
       believe(X,believe(1, non white2)) <- non white2,
       believe(X,believe(1, white3)) <- white3,
       believe(X,believe(2, white3)) <- white3,
       believe(X,believe(1, non white3)) <- non white3,
       believe(X,believe(2, non white3)) <- non white3],
        [believe(X, believe(Y,(A,B)))  <-   (believe(X,believe(Y,A)),
                                              believe(X,believe(Y,B))),

         believe(X, believe(Y,P))  <-   (believe(Z,P), var(Z))],

        [believe(X, believe(Y,P))  <- believe(X,believe(Y,Q))
                                          <- (believe(Z,P<-Q), var(Z))],

         /* schema for  nested  double reductio ad absurdum  */

        [believe(X,P) <- true

          <-   ([non P,non Q] => believe(X,believe(Y,non believe(Z,R))),

                [non P,non Q] => believe(X,believe(Y,believe(Z,R))))

                  <-   ([non P] => believe(X,believe(Y,non P)),

                          [non P] => believe(X,non believe(Y,Q)))]].

?- solve(wise:answer(X,Y)).
X = 3 ,
Y = white3 ;

X = 3 ,
Y = white3 ;

no
```

N.B. The order in which wise1 and wise2 have expressed their beliefs being not taken into account results in duplicate solutions.

4. Conclusions: perspectives, limitations and possible extensions

The ease with which we were able to transcribe the above models is a good indication of the expressive power enjoyed by our own system. Although we are not in a position to back up any serious claim, we are led to believe that most (if not all) all the formal reasoning that can be carried out *by hand* within the framework of "proofs in context" [1] can be *automated* using lifted resolution of omega-ordered Horn meta-theories.

As already mentionned in the introduction, lifted resolution of omega-ordered Horn theories is very sensitive to the actual ordering of the meta-theories defining control knowledge. As an example, the successful termination of the example encoding private beliefs (see above 2.3.1) not only requires the two meta-rules expressing nested beliefs to be in separate theories, but also calls for an extra *empty* meta-theory to accomodate lifting . This situation amounts to facing a *meta-meta-level search* problem topping the infinite tower defining the application domain. As a possible way out, one might consider the following extensions, leading to a *meta-interpretation* process:

- first, *reifying* the interpretation process itself into a (flat) theory I ; this theory should include our Prolog interpreter as well as meta-rules for ordering theories in the application tower T

- then modifying our Prolog interpreter to process the theory I , which, in turn, should process the application tower T.

As a simple example of such a meta-interpretation process, we give below an interpreter theory that will allocate lifting space in the application tower on demand:

```
interpreter:[interpret(P) <- (P=(A;B), (interpret(A);
                                         interpret(B)),

                              P=(A,B),  interpret(A),
                                         interpret(B);

                              system(P), call(P);

                              match(P);
                              lift(P<-Q),
                              interpret(Q))),
            solve(P) <- (interpret(P) <- Cond,
                         Cond),

            solve(P) <- (write('stretch ? (y/n) '),nl,
                         read(R),nl,
                         (R='y',
                          stretch(P);
                          R='n',
                          write('stop'),nl,fail))].
```

This theory reproduces our Prolog interpreter implementing lifted resolution, extended with an interactive dialog allowing the user to "stretch" the application tower by allocating more lifting space. As the lifting process itself is now taken care of by a control theory, there is no need to introduce it into the interpreter. We can thus revert to the basic Prolog interpreter of section 2.1 , which must be modified as follows:

```
solve(Theory:Goal)  :- Theory:T,
                       interpreter:I,
                       interpret(solve(Goal),T,I).

interpret(A,T,I):- match(A,I);
                   match(A<-C,I),
                   interpret(C,T,I).
```

The extensions required to process the interpreter theory include the following procedures:

```
interpret(match(A),[T1|_],_):- !,match(A,T1).

interpret(lift(A),[T1,T2|Tn],I):-!,append(T1,T2,T12),
                                  interpret(interpret(A),[T12|Tn],I).

interpret(stretch(A),T,I):- !,append(T,[[]],TS),
                            interpret(solve(A),TS,I).
```

Whereas the overhead associated with such a meta-interpretation process should not be overlooked, the above example accounts well for the open-ended facilities offered by our architecture.

5. References

[1] G. Attardi and M. Simi, Building Proof in Context, Pre-proceedings 4th Inter. Workshop on Meta Programming in Logic (META 94), 1994

[2] J. Barklund, K. Boberg and P. Dell'Acqua, A Basis for a MultiLevel Metalogic Programming Language, Pre-proceedings 4th Inter. Workshop on Meta Programming in Logic (META 94), 1994

[3] K. Bowen and R.Kowalski, Amalgating language and metalanguage in logic programming, in:Logic Programming, K. Clark and S. Tarnlund (eds), Academic Press, 1982

[4] I. Cervesato and G. F. Rossi, Logic Meta-Programming Facilities in 'Log, in: A. Pettorossi (ed.), Meta-Programming in Logic, Proc. 3rd Int. Work. (META 92), Lecture Notes In Computer Science (vol 649), Springer Verlag, 1992

[5] S. Costantini and G.A. Lanzarone, A MetaLogic Programming Language, in: C. Levi and M. Martelli (eds.), Proceedings of the Sixth Inter. Conf. on Logic Programming, MIT Press, 1989

[6] F. Giunchiglia, L. Serafini and A. Simpson, Hierarchical Meta-Logics: Intuitions, Proof Theory and Semantics, in: A. Pettorossi (ed.), Meta-Programming in Logic, Proc. 3rd Int. Work. (META 92), Lecture Notes In Computer Science (vol 649), Springer Verlag, 1992

[7] F. van Harmelen, Meta-level Inference Systems, Pitman and Morgan Kaufman Publ. 1991

[8] P.M. Hill and J.W. Lloyd, Analysis of Metaprograms, in: H. Abramson and R Rogers (eds), Metaprogramming in Logic Programming, The MIT Press, 1989

[9] A. Hutchinson, Algorithmic Learning, Oxford University Press, 1994

[10] K. Konolidge, A Deduction Model of Belief, Pitman and Morgan Kaufman Publ. , 1986

[11] J. McCarthy, Notes on Formalizing Context, Proc. 13th Int. Join Conf. on Art. Intell, (IJCAI93), 1993

[12] S. Russell, Execution Architectures and Compilation, Proc. 11th Int. Join Conf. on Art. Intell, (IJCAI89), 1989

[13] B.C. Smith, Reflection and Semantics in Lisp, Conf. Rec. 11th ACM Symp. on Princ. of Progr. Languages (POPL 84), 1984

[14] R. Weyhrauch, Prolegomena to a theory of mechanized formal reasoning, Artificial Intelligence, 13(1), 1980