Integrating Connective Clue Processing into the Argument Analysis Algorithm Implementation

Trevor J. Smedley
Department of Computer Science

Research Report CS-87-34
June 1987

Integrating Connective Clue Processing into the Argument Analysis Algorithm Implementation

Trevor J. Smedley

Department of Computer Science
University of Waterloo
Waterloo, Ontario
CANADA N2L 3G1

ABSTRACT

The argument analysis algorithm presented by Cohen[1-4] was implemented by Smedley[5]. This was a basic implementation which did not include any clue processing. This report describes the addition of basic connective clue processing to the implementation.

Introduction

Argument Processing

The argument processing algorithm used for this implementation is from Cohen[1-4] and will not be described in detail here. It is used to analyse arguments, where an argument is taken as a one way discourse intended to convince the hearer of some point. Basically, the algorithm incrementally builds a tree showing the claim-evidence relationships present in the argument. It does this by selectively applying the test "Is proposition A evidence for proposition B?" The basic algorithm was implemented by Smedley[5] and this report focuses on adding some clue processing to this basic implementation. The basic algorithm is as follows:

The following loop is executed for each new statement in the argument:

```
forever do:
   if NEW evidence for L then
      if no sons of L are evidence for NEW then
         /* just test rightmost son for evidence */
         attach NEW below L
         set L to NEW
         exit forever loop
      else
         attach all sons of L which are evidence for NEW below NEW
         attach NEW below L
         exit forever loop
      endif
   else
      set L to father of L
   endif
enddo
```

Clues

Clues are words or phrases which a speaker introduces into an argument to help the hearer in understanding. There are two basic types of clues: redirection and connective. Redirection clues provide information regarding where statements fit into an argument. They are things such as "return to...", "about the..." etc. These are not dealt with in this report.

Connective clues are phrases such as "in particular", "for example", "in addition" and "as a result". These give the hearer information as to how the statements fit together. For example, "as a result" indicates that the statement which follows has a previous proposition as evidence.

There are a number of different classes of connective clues. The types which are covered by this implementation are:

category	relation to some prior proposition	example
parallel	brother	in addition
detail	son	in particular
inference	father	as a result
summary	father to multiple sons	in sum

This implementation is by no means complete. There are still many problems which remain to be solved: multiple clues, clues which act as both connective and redirection, clues which fall into more than one category, other types of connective clues, and many others.

The algorithm which has been implemented will handle all arguments without clues exactly as the old algorithm, and will also take advantage of the above classes of connective clues. Thus this implementation supercedes the previous one. It is as follows:

For each new statement in the argument, the following loop is executed. Note that clue1 = parallel, clue2 = detail, clue3 = inference and clue4 = summary.

```
forever do:
   if L = dummy and clue2 then
      INTERRUPT-DISCOURSE (and exit loop)
   endif
   /* see if rightmost son exists */
   if (clue1 or clue3 or clue4) and no rightmost son of L then
      if L = dummy then
         INTERRUPT-DISCOURSE (and exit loop)
      else
         set L to father of L
      endif
   endif
   if NEW evidence for L then
      /* see if sons will reattach */
      if no sons of L are evidence for NEW then
         if (clue3 or clue4) then
            if L = dummy then
               INTERRUPT-DISCOURSE (and exit loop)
            else
               set L to father of L
            endif
         else
            /* normal attaching */
            attach NEW below L
            set L to NEW
            exit forever loop
         endif
      else
         /* some son wants to reattach */
         attach all sons of L which are evidence for NEW below NEW
         attach NEW below L
         exit forever loop
      endif
   else
      set L to father of L
   endif
enddo
```

Implementation

The following is a description of the changes made to the original implementation. For a description of the original implementation see Smedley[5]. As in the original implementation, it is assumed that the input as already been parsed, and that clues have been flagged. The examples show how this is done.

Changes

Only one predicate of the original implementation had to be changed. That is the predicate *hybrid_*, which handles the main processing of the arguments. The original code was left unchanged and code to handle the clues was added at the beginning. The clauses are ordered to agree with the order of the presentation of the algorithm. The first clause handles the first if clause. The next three are the if part of the next if clause, and the next three are the else part. The last four clauses handle the if clause in the last half of the algorithm which deals with inference and summary clues. The first two are the if part, and the last two are the else part.

The only predicate which was changed to add the clue processing was the predicate "hybrid_". The original code was left unchanged and new code was added. Here is the new code for the "hybrid_" predicate. For the remaining code see Smedley[5].

```
%
% This builds the argument tree for hybrid type of arguments. The
% algorithm is from R. Cohen. See the documentation for a detailed
% description.
%
% The first entries handle connective clues.
%
hybrid_(_ detail(_) dummy) <-
        printf("failed due to detail clue" [])
        nl
        printf("no father found" [])
        nl
        printf("tree so far is:" [])
        nl;
%
hybrid_(_ parallel(_) dummy) <-
        not(rightmost_child(dummy _))
        printf("failed due to parallel clue" [])
        nl
        printf("no possible brothers" [])
        nl
        printf("tree so far is:" [])
        nl;
hybrid_(_ inference(_) dummy) <-
        not(rightmost_child(dummy _))
        printf("failed due to inference clue" [])
        nl
```

```
        printf("no possible son to re-attach" [])
        nl
        printf("tree so far is:" [])
        nl;
hybrid_(_ summary(_) dummy) <-
        not(rightmost_child(dummy _))
        printf("failed due to summary clue" [])
        nl
        printf("no possible sons to re-attach" [])
        nl
        printf("tree so far is:" [])
        nl;
%
hybrid_(S parallel(T) L) <-
        not(rightmost_child(L _))
        printf("parallel clue saves an oracle call" [])
        nl
        printf("``Last'' has no sons to be brothers" [])
        nl
        father(X L)
        !
        hybrid_(S parallel(T) X);
hybrid_(S inference(T) L) <-
        not(rightmost_child(L _))
        printf("inference clue saves an oracle call" [])
        nl
        printf("``Last'' has no sons available to be re-attached" [])
        nl
        father(X L)
        !
        hybrid_(S inference(T) X);
hybrid_(S summary(T) L) <-
        not(rightmost_child(L _))
        printf("summary clue saves an oracle call" [])
        nl
        printf("``Last'' has no sons available to be re-attached" [])
        nl
        father(X L)
        !
        hybrid_(S summary(T) X);
%
hybrid_(_ inference(T) dummy) <-
        no_sons_evidence(dummy inference(T))
        printf("failed due to inference clue" [])
        nl
        printf("no possible sons to re-attach" [])
        nl
```

```
        printf("tree so far is:" [])
        nl;
hybrid_(_ summary(T) dummy) <-
        no_sons_evidence(dummy summary(T))
        printf("failed due to summary clue" [])
        nl
        printf("no possible sons to re-attach" [])
        nl
        printf("tree so far is:" [])
        nl;
%
hybrid_(S inference(T) L) <-
        evidence_oracle(inference(T) L _)
        no_sons_evidence(L inference(T))
        printf("inference clue causes continuation" [])
        nl
        father(X L)
        !
        hybrid_(S inference(T) X);
hybrid_(S summary(T) L) <-
        evidence_oracle(summary(T) L _)
        no_sons_evidence(L summary(T))
        printf("summary clue causes continuation" [])
        nl
        father(X L)
        !
        hybrid_(S summary(T) X);
%
% standard algorithm
%
hybrid_([] N L) <-
        evidence_oracle(N L _)
        no_sons_evidence(L N)
        assert_father(L N);
hybrid_([] N L) <-
        evidence_oracle(N L _)
        attach_sons(L N)
        assert_father(L N);
hybrid_([H | T] N L) <-
        evidence_oracle(N L _)
        no_sons_evidence(L N)
        assert_father(L N)
        hybrid_(T H N);
hybrid_([H | T] N L) <-
        evidence_oracle(N L _)
        attach_sons(L N)
        assert_father(L N)
```

```
        hybrid_(T H L);
hybrid_(S N L) <-
        not(evidence_oracle(N L _))
        father(X L)
        hybrid_(S N X);
```

Examples

The following examples show how the code works with various examples.

Example 1

This shows how the new code works with an example with no clues.

```
Waterloo Unix Prolog [Release 2.0 -- Oct., 1985]
?consult(hybrid.1);

?       analyse([[the,city,is,a,mess],
         [the,playground,area,is,all,run,down],
         [the,sandboxes,are,dirty],
         [the,swings,are,broken],
         [the,parks,are,a,disaster],
         [the,highway,system,also,needs,revamping]]);

The argument is;
[the,city,is,a,mess]
[the,playground,area,is,all,run,down]
[the,sandboxes,are,dirty]
[the,swings,are,broken]
[the,parks,are,a,disaster]
[the,highway,system,also,needs,revamping]

Is
[the,playground,area,is,all,run,down]
evidence for
[the,city,is,a,mess]
(answer y or n)  y

Is
[the,sandboxes,are,dirty]
evidence for
[the,playground,area,is,all,run,down]
(answer y or n)  y

Is
```

```
[the,swings,are,broken]
evidence for
[the,sandboxes,are,dirty]
(answer y or n)  n

Is
[the,swings,are,broken]
evidence for
[the,playground,area,is,all,run,down]
(answer y or n)  y

Is
[the,sandboxes,are,dirty]
evidence for
[the,swings,are,broken]
(answer y or n)  n

Is
[the,parks,are,a,disaster]
evidence for
[the,swings,are,broken]
(answer y or n)  n

Is
[the,parks,are,a,disaster]
evidence for
[the,playground,area,is,all,run,down]
(answer y or n)  n

Is
[the,parks,are,a,disaster]
evidence for
[the,city,is,a,mess]
(answer y or n)  y

Is
[the,playground,area,is,all,run,down]
evidence for
[the,parks,are,a,disaster]
(answer y or n)  y

Is
[the,highway,system,also,needs,revamping]
evidence for
[the,city,is,a,mess]
(answer y or n)  y
```

```
Is
[the,parks,are,a,disaster]
evidence for
[the,highway,system,also,needs,revamping]
(answer y or n)  n

Argument tree;
[the,city,is,a,mess]
        [the,parks,are,a,disaster]
                [the,playground,area,is,all,run,down]
                        [the,sandboxes,are,dirty]
                        [the,swings,are,broken]
        [the,highway,system,also,needs,revamping]
yes
yes
```

Example 2

The processing of a parallel clue is shown with this example.

```
?consult(clue.1);

?       analyse([[the,city,is,in,serious,trouble],
         [there,are,some,fires,going],
         [three,separate,blazes,have,broken,out],
         parallel([in,addition,a,tornado,is,passing,through])]);

The argument is;
[the,city,is,in,serious,trouble]
[there,are,some,fires,going]
[three,separate,blazes,have,broken,out]
parallel([in,addition,a,tornado,is,passing,through])

Is
[there,are,some,fires,going]
evidence for
[the,city,is,in,serious,trouble]
(answer y or n)  y

Is
[three,separate,blazes,have,broken,out]
evidence for
[there,are,some,fires,going]
(answer y or n)  y

parallel clue saves an oracle call
''Last'' has no sons to be brothers
```

```
Is
parallel([in,addition,a,tornado,is,passing,through])
evidence for
[there,are,some,fires,going]
(answer y or n)  n

Is
parallel([in,addition,a,tornado,is,passing,through])
evidence for
[the,city,is,in,serious,trouble]
(answer y or n)  y

Is
[there,are,some,fires,going]
evidence for
parallel([in,addition,a,tornado,is,passing,through])
(answer y or n)  n

Argument tree;
[the,city,is,in,serious,trouble]
        [there,are,some,fires,going]
                [three,separate,blazes,have,broken,out]
        parallel([in,addition,a,tornado,is,passing,through])
yes
yes
```

Example 3

An inference clue is demonstrated; along with the way incoherence is handled.

```
?consult(clue.2);

?       analyse([[the,parks,are,a,mess],
         [the,park,benches,are,a,mess],
         [the,playgrounds,are,a,mess],
         inference([as,a,result,the,highways,are,a,mess])]);

The argument is;
[the,parks,are,a,mess]
[the,park,benches,are,a,mess]
[the,playgrounds,are,a,mess]
inference([as,a,result,the,highways,are,a,mess])

Is
[the,park,benches,are,a,mess]
evidence for
```

```
[the,parks,are,a,mess]
(answer y or n)  y

Is
[the,playgrounds,are,a,mess]
evidence for
[the,park,benches,are,a,mess]
(answer y or n)  n

Is
[the,playgrounds,are,a,mess]
evidence for
[the,parks,are,a,mess]
(answer y or n)  y

Is
[the,park,benches,are,a,mess]
evidence for
[the,playgrounds,are,a,mess]
(answer y or n)  n

inference clue saves an oracle call
''Last'' has no sons available to be re-attached
Is
inference([as,a,result,the,highways,are,a,mess])
evidence for
[the,parks,are,a,mess]
(answer y or n)  n

Is
[the,parks,are,a,mess]
evidence for
inference([as,a,result,the,highways,are,a,mess])
(answer y or n)  n

%
% The argument is determined to be incoherent
%
failed due to inference clue
no possible sons to re-attach
tree so far is:
Argument tree;
[the,parks,are,a,mess]
        [the,park,benches,are,a,mess]
        [the,playgrounds,are,a,mess]
yes
yes
```

Example 4

```
Both a parallel and a summary clue in one argument.
?consult(clue.3);

?       analyse([[it,snowed,today],
         [it,rained,yesterday],
         parallel([and,there,is,hail,forecasted,for,tomorrow]),
         summary([in,sum,the,weather,is,terrible])]);

The argument is;
[it,snowed,today]
[it,rained,yesterday]
parallel([and,there,is,hail,forecasted,for,tomorrow])
summary([in,sum,the,weather,is,terrible])

Is
[it,rained,yesterday]
evidence for
[it,snowed,today]
(answer y or n)  n

Is
[it,snowed,today]
evidence for
[it,rained,yesterday]
(answer y or n)  n

parallel clue saves an oracle call
''Last'' has no sons to be brothers
Is
[it,rained,yesterday]
evidence for
parallel([and,there,is,hail,forecasted,for,tomorrow])
(answer y or n)  n

summary clue saves an oracle call
''Last'' has no sons available to be re-attached
Is
parallel([and,there,is,hail,forecasted,for,tomorrow])
evidence for
summary([in,sum,the,weather,is,terrible])
(answer y or n)  y

Is
[it,rained,yesterday]
evidence for
```

```
summary([in,sum,the,weather,is,terrible])
(answer y or n)  y

Is
[it,snowed,today]
evidence for
summary([in,sum,the,weather,is,terrible])
(answer y or n)  y

Argument tree;
summary([in,sum,the,weather,is,terrible])
        [it,rained,yesterday]
        parallel([and,there,is,hail,forecasted,for,tomorrow])
yes
yes
```

Example 5

Shows how a detail clue can signal incoherence.

```
?consult(clue.4);

?       analyse([[it,snowed,today],
         [it,rained,yesterday],
         detail([in,particular,there,is,hail,forecasted,for,tomorrow]),
         summary([in,sum,the,weather,is,terrible])]);

The argument is;
[it,snowed,today]
[it,rained,yesterday]
detail([in,particular,there,is,hail,forecasted,for,tomorrow])
summary([in,sum,the,weather,is,terrible])

Is
[it,rained,yesterday]
evidence for
[it,snowed,today]
(answer y or n)  n

Is
[it,snowed,today]
evidence for
[it,rained,yesterday]
(answer y or n)  n

Is
detail([in,particular,there,is,hail,forecasted,for,tomorrow])
```

```
evidence for
[it,rained,yesterday]
(answer y or n)  n

failed due to detail clue
no father found
tree so far is:
no
yes
```

Example 6

A summary clue signals incoherence before any oracle calls.

```
?consult(clue.5);

?       analyse([summary([in,sum,the,weather,was,awful]),
         [it,rained,last,week],
         [and,snowed,all,the,week,before]]);

The argument is;
summary([in,sum,the,weather,was,awful])
[it,rained,last,week]
[and,snowed,all,the,week,before]

failed due to summary clue
no possible sons to re-attach
tree so far is:
no
yes
```

Example 7

This example shows how an inference clue can cause continuation of the processing.

```
?consult(clue.6);

?       analyse([[i,had,to,wait,to,see,if,anyone,called],
         [i,didnt,get,the,letter,saying,i,got,the,job],
         [i,didnt,pick,up,my,mail,at,school],
         [the,mail,room,was,locked,and,i,forgot,my,key],
         inference([as,a,result,i,had,to,stay,home,today])]);

The argument is;
[i,had,to,wait,to,see,if,anyone,called]
[i,didnt,get,the,letter,saying,i,got,the,job]
```

```
[i,didnt,pick,up,my,mail,at,school]
[the,mail,room,was,locked,and,i,forgot,my,key]
inference([as,a,result,i,had,to,stay,home,today])

Is
[i,didnt,get,the,letter,saying,i,got,the,job]
evidence for
[i,had,to,wait,to,see,if,anyone,called]
(answer y or n)  y

Is
[i,didnt,pick,up,my,mail,at,school]
evidence for
[i,didnt,get,the,letter,saying,i,got,the,job]
(answer y or n)  y

Is
[the,mail,room,was,locked,and,i,forgot,my,key]
evidence for
[i,didnt,pick,up,my,mail,at,school]
(answer y or n)  y

inference clue saves an oracle call
''Last'' has no sons available to be re-attached
Is
inference([as,a,result,i,had,to,stay,home,today])
evidence for
[i,didnt,pick,up,my,mail,at,school]
(answer y or n)  y
%
% We have now found a father for [as a result...], but since there no
% sons to re-attach below the staement, and the clue indicates that
% there must be a son, we continue looking for a father farther up in
% the tree.
%
Is
[the,mail,room,was,locked,and,i,forgot,my,key]
evidence for
inference([as,a,result,i,had,to,stay,home,today])
(answer y or n)  n

inference clue causes continuation
Is
inference([as,a,result,i,had,to,stay,home,today])
evidence for
[i,didnt,get,the,letter,saying,i,got,the,job]
(answer y or n)  n
```

```
Is
inference([as,a,result,i,had,to,stay,home,today])
evidence for
[i,had,to,wait,to,see,if,anyone,called]
(answer y or n)  n
%
% The sentence [as a result...] fits in at the top, below "dummy"
%
Is
[i,had,to,wait,to,see,if,anyone,called]
evidence for
inference([as,a,result,i,had,to,stay,home,today])
(answer y or n)  y
%
% and a son gets re-attached, as required
%
Argument tree;
inference([as,a,result,i,had,to,stay,home,today])
        [i,had,to,wait,to,see,if,anyone,called]
                [i,didnt,get,the,letter,saying,i,got,the,job]
                        [i,didnt,pick,up,my,mail,at,school]
                                [the,mail,room,was,locked,and,i,forgot,my,
yes
yes
?quit;
```

References

1. R. Cohen, Investigation of Processing Strategies for the Structural Analysis of Arguments, *Proceedings of ACL Conference*, pp. 71-75 (June, 1981).
2. R. Cohen, A Computational Model for the Analysis of Arguments, *University of Toronto Technical Report CSRG-151*, (October, 1983).
3. R. Cohen, A Theory of Discourse Coherence for Argument Understanding, *Proceedings of CSCSI/SCEIO Conference*, pp. 6-10 (May, 1984).
4. R. Cohen, Interpreting Clues in Conjunction with Processing Restrictions in Arguments and Discourse, *Proceedings AAAI 1987*, (To Appear).
5. T.J. Smedley, An Implementation of a Computational Model for the Analysis of Arguments — An Introduction to the First Attempt, *Technical Report CS-86-26*, University of Waterloo, (1986).