MAS Assessed Coursework 1– Argumentation

Due date: Tuesday 19 February 2013

Electronic submission
(Submission in groups of up to three students allowed/welcome)

1. Consider the following Prolog representation of Abstract Argumentation (AA):

`argument(a).`	for `a` is an argument.
`attacks(a,b).`	for argument `a` attacks argument `b`.

   Give a Prolog *meta-interpreter* (i.e. a Prolog program) defining the predicate `grounded/1` such that if

   `grounded(a).`

   succeeds for some abstract argument `a` then `a` belongs to the grounded set of arguments.

   Your implementation does not need to generate the grounded set of arguments, it is sufficient for it to test whether or not the input argument is contained in the grounded extension.

   You should test your program with several AA frameworks, including (but not limited to) the ones below.

   - For the AA framework (Arg, Att) with
     - $Arg = \{a, b, c\}$
     - $Att = \{(a, b), (b, c)\}$

     the grounded extension is $\{a, c\}$, so, for example, `grounded(a).` should succeed, but `grounded(b).` should (finitely) fail.
   - For the AA framework (Arg, Att) with
     - $Arg = \{a, b, c, d\}$
     - $Att = \{(a, a), (a, b), (b, a), (c, d), (d, c)\}$

     `grounded(X).` should not succeed for any argument `X`, since the grounded extension here is the empty set.

   Don't worry too much if your program loops for complicated AA frameworks. However, it should not loop for the frameworks given above.

   It may be useful to use ASPARTIX to check whether you get the right answers for AA frameworks you think of and use for testing.

2. Consider the following Prolog representation of the main concepts in Assumption-Based Argumentation (ABA):

`myAsm(a).`	for a is an assumption
`contrary(a,b).`	for the contrary relation $\overline{a} = b$
`myRule(a,[b,c]).`	for inference rule $a \leftarrow b, c$
`myRule(b,[]).`	for inference rule $b \leftarrow$

   (a) Give a Prolog *meta-interpreter* defining the predicate `argument/1` such that
   if `argument((C,X)).` succeeds then the (set corresponding to the) list of assumptions `X` supports an argument for the claim `C`, that is `X` $\vdash$ `C` is an ABA argument, represented as `(C,X)`.

   (b) Give a Prolog *meta-interpreter* defining the predicate `attacks/2` such that
   if `attacks((C1,X1),(C2,X2)).` succeeds then the argument `(C1,X1)` attacks the argument `(C2,X2)`.

   Your implementations of `argument/1` and `attacks/2` should be able to generate arguments and attacks, respectively, in addition to testing whether they are so. For example, given

```
myAsm(a).
myAsm(b).
contrary(a,p).
myRule(p,[b]).
myRule(p,[]).
```

   your implementation should behave as follows:

`argument((p,X)).`	should give `X=[b]` or `X=[]`
`argument((C,[b])).`	should give `C=b` or `C=p`
`argument(a,[a]).`	should succeed
`attacks((p,X),(C,Y)).`	should give `X=[], C=a, Y=[a]` etc

3. Consider the following setting:
   You are working in a hospital, allocating appointments to patients. Today, there are two patients, Anne (a) and Boris (b). They both tell you that they could probably make the 8am or the 6pm appointment. While talking to them for a bit, Anne tells you that she has a child, so you take a note that the 8am appointment would not be optimal for her because she has to bring her child to the nursery at this time. During your conversation with Boris, he mentions that he is taking sports classes and would consequently prefer not to have the 6pm appointment. However, you are not completely sure that he is telling the truth since he looks very overweight and does not seem to be the kind of person who does sports regularly. On the basis of these information you now have to decide who gets which appointment.

   Your strategy for making a decision will be Assumption-Based Argumentation (ABA), where the ABA framework $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, {}^- \rangle$ has

   - $\mathcal{A} = \{$*free6pm(a)*, *free8am(a)*, *free6pm(b)*, *free8am(b)*,
     child(a), *sports(b)*, *overweight(b)*,
     not_get6pm(a), *not_get8am(a)*, *not_get6pm(b)*, *not_get8am(b)*$\}$
   - $\overline{\textit{free6pm(a)}} = \textit{not\_free6pm(a)}$, $\overline{\textit{free8am(a)}} = \textit{not\_free8am(a)}, \ldots$
     $\overline{\textit{child(a)}} = \textit{not\_child(a)}$, $\overline{\textit{sports(b)}} = \textit{not\_sports(b)}$, $\overline{\textit{overweight(b)}} = \textit{not\_overweight(b)}$,
     $\overline{\textit{not\_get6pm(a)}} = \textit{get6pm(a)}$, $\overline{\textit{not\_get8am(a)}} = \textit{get8am(a)}, \ldots$
   - $\mathcal{R} = \{$*get6pm(a)* $\leftarrow$ *free6pm(a)*, *not_get8am(a)*, *not_get6pm(b)*;
     get8am(a) $\leftarrow$ *free8am(a)*, *not_get6pm(a)*, *not_get8am(b)*;
     get6pm(b) $\leftarrow$ *free6pm(b)*, *not_get8am(b)*, *not_get6pm(a)*;
     get8am(b) $\leftarrow$ *free8am(b)*, *not_get6pm(b)*, *not_get8am(a)*;
     not_free8am(a) $\leftarrow$ *child(a)*;
     not_free6pm(b) $\leftarrow$ *sports(b)*;
     not_sports(b) $\leftarrow$ *overweight(b)*$\}$

   (a) Run your implementations in parts 1 and 2 to determine whether *free6pm(b)* belongs to the grounded extension. Write down the query(-ies) you used and the results.

   (b) Download the SXDD implementation from CATE and have a look at the readme file. Run the SXDD implementation to determine whether *free6pm(b)* belongs to the grounded set (make sure you are using SXDD with grounded, not admissible semantics!). Then,

      i. Write down the computed dispute derivation in tabular form (as shown in the lectures, from slide 50), with an additional column briefly explaining each step.

      ii. Draw the arguments and attacks of this dispute derivation (as in the lectures).

   (c) Who gets which appointment according to the grounded extension? Who would get which appointment if you had used the stable semantics?

Submit a zipped directory 'ArgCW' containing (i) a file grounded.pl with your answer to questions 1 (don't include the examples in your code!) (ii) files argument.pl and attacks.pl with your answers to questions 2a and 2b respectively (don't include examples in your code!) (iii) a file 3.pdf with your answers to questions 3a-c.