Appendix A. Annotated ASP Theory

```
% ---------- scene ----------
%!trace_rule {"We locate a candidate from the detected objects,
               object with ID %", ID}
state(TO,ID) :- scene(TO), object(ID).

% ---------- end ----------
%!trace_rule {"The answer to the question is %", V}
ans(V) :- end(TO), attr_value(TO,V).
%!trace_rule {"The answer to the question is %", V}
ans(V) :- end(TO), attr(TO,V).
%!trace_rule {"The answer to the question is %", V}
ans(V) :- end(TO), rel(TO,V).
%!trace_rule {"The answer to the question is %", V}
ans(V) :- end(TO), bool(TO,V).

% ---------- ans ----------
% At least one answer must be derivable
:- not ans(_).
#show ans/1.

% ---------- select ----------
%!trace_rule {"We select object % because of being
               in the class %", ID, CLASS}
state(TO,ID) :- select(TO, TI, CLASS), state(TI, ID),
               has_attr(ID, class, CLASS).

% ---------- filter ----------
%!trace_rule {"We check that the object has % %", ATTR, VALUE}
state(TO,ID) :- filter(TO, TI, ATTR, VALUE), state(TI, ID),
               has_attr(ID, ATTR, VALUE).

%!trace_rule {"We check that the objects have % %", ATTR, VALUE}
state(TO,ID) :- filter_any(TO, TI, VALUE), state(TI, ID),
               has_attr(ID, ATTR, VALUE).

% ---------- relate ----------
%!trace_rule {"Object with ID % is % object with ID %", ID', REL, ID}
state(TO, ID') :- relate(TO, TI, CLASS, REL, subject), state(TI, ID),
                  has_attr(ID', class, CLASS), has_rel(ID', REL, ID).

%!trace_rule {"Object with ID % is % object with ID %", ID', REL, ID}
state(TO, ID') :- relate(TO, TI, CLASS, REL, object), state(TI, ID),
                  has_attr(ID', class, CLASS), has_rel(ID, REL, ID').

%!trace_rule {"Object with ID % is % object with ID %", ID', REL, ID}
state(TO, ID') :- relate_any(TO, TI, REL, subject), state(TI, ID),
                  has_rel(ID', REL, ID).

%!trace_rule {"Object with ID % is % object with ID %", ID', REL, ID}
```

```
state(TO, ID') :- relate_any(TO, TI, REL, object), state(TI, ID),
                  has_rel(ID, REL, ID').

%!trace_rule {"We establish an attribute relationship between
               objects % and % based on attribute % having
               the same value %", ID, ID', ATTR, VALUE}
state(TO, ID') :- relate_attr(TO, TI, CLASS, ATTR), state(TI, ID),
                  has_attr(ID, ATTR, VALUE),
                  has_attr(ID', class, CLASS),
                  has_attr(ID', ATTR, VALUE'),
                  VALUE==VALUE', ID!=ID'.

% ---------- query ----------
%!trace_rule {"We detect that object with ID % has % with value %",
               ID, ATTR, VALUE}
{ has_attr(ID, ATTR, VALUE) : is_attr_value(ATTR, VALUE)} = 1 :-
      query(TO, TI, ATTR), state(TI, ID), ATTR != name,
      ATTR != class, ATTR != hposition, ATTR != vposition.

%!trace_rule {"We detect that object with ID % has % with value %",
               ID, ATTR, VALUE}
attr_value(TO,VALUE) :- query(TO, TI, ATTR), state(TI, ID),
                        has_attr(ID, ATTR, VALUE).

% ---------- verify ----------
%!trace_rule {"We detect that object with ID % has % with value %",
               ID, ATTR, VALUE}
bool(TO, yes) :- verify_attr(TO, TI, ATTR, VALUE), state(TI, ID),
                 has_attr(ID, ATTR, VALUE).

%!trace_rule {"We detect no object of % with value %", ATTR, VALUE}
bool(TO,no) :- verify_attr(TO, TI, ATTR, VALUE), not bool(TO,yes).

%!trace_rule {"We detect that object with ID % is in a relation %,
               ID, REL}
bool(TO, yes) :- verify_rel(TO, TI, CLASS, REL, subject), state(TI, ID),
                 has_attr(ID', class, CLASS), has_rel(ID', REL, ID).

%!trace_rule {"We detect no object in a relation %", REL}
bool(TO,no) :- verify_rel(TO, TI, CLASS, REL, subject),
               not bool(TO,yes).

%!trace_rule {"We detect that object with ID % is in a relation %,
               ID, REL}
bool(TO, yes) :- verify_rel(TO, TI, CLASS, REL, object), state(TI, ID),
                 has_attr(ID', class, CLASS), has_rel(ID, REL, ID').

%!trace_rule {"We detect no object in a relation %", REL}
bool(TO,no) :- verify_rel(TO, TI, CLASS, REL, object), not bool(TO,yes).

% ---------- choose ----------
```

```
{has_attr(ID, ATTR, VALUE); has_attr(ID, ATTR, VALUE')} = 1 :-
    choose_attr(TO, TI, ATTR, VALUE, VALUE'), state(TI, ID).
%!trace_rule {"Between % % or %, we choose %",
               ATTR, VALUE, VALUE', VALUE }
attr_value(TO, VALUE) :- choose_attr(TO, TI, ATTR, VALUE, VALUE'),
                         state(TI, ID), has_attr(ID, ATTR, VALUE).

%!trace_rule {"Between % % or %, we choose %",
               ATTR, VALUE, VALUE', VALUE' }
attr_value(TO, VALUE') :- choose_attr(TO, TI, ATTR, VALUE, VALUE'),
                          state(TI, ID), has_attr(ID, ATTR, VALUE').

{has_rel(ID', REL, ID): has_attr(ID', class, CLASS);
      has_rel(ID', REL', ID): has_attr(ID', class, CLASS)} = 1 :-
      choose_rel(TO, TI, CLASS, REL, REL', subject), state(TI, ID).

%!trace_rule {"Between relations % or %, we choose %",
               REL, REL', REL }
rel(TO, REL) :- choose_rel(TO, TI, CLASS, REL, REL', subject),
                state(TI, ID), has_attr(ID', class, CLASS),
                has_rel(ID', REL, ID).

%!trace_rule {"Between relations % or %, we choose %",
               REL, REL', REL' }
rel(TO, REL') :- choose_rel(TO, TI, CLASS, REL, REL', subject),
                 state(TI, ID), has_attr(ID', class, CLASS),
                 has_rel(ID', REL', ID).

{has_rel(ID, REL, ID'): has_attr(ID', class, CLASS);
   has_rel(ID, REL', ID'): has_attr(ID', class, CLASS)} = 1 :-
   choose_rel(TO, TI, CLASS, REL, REL', object), state(TI, ID).

%!trace_rule {"Between relations % or %, we choose %",
               REL, REL', REL }
rel(TO, REL) :- choose_rel(TO, TI, CLASS, REL, REL', object),
                state(TI, ID), has_attr(ID', class, CLASS),
                has_rel(ID, REL, ID').

%!trace_rule {"Between relations % or %, we choose %",
               REL, REL', REL' }
rel(TO, REL') :- choose_rel(TO, TI, CLASS, REL, REL', object),
                 state(TI, ID), has_attr(ID', class, CLASS),
                 has_rel(ID, REL', ID').

% ---------- exist ----------
%!trace_rule {"Object with ID % is a %", ID, N}
bool(TO,yes) :- exist(TO, TI), state(TI,ID), has_attr(ID, name, N).

%!trace_rule {"No such object was detected"}
bool(TO,no) :- exist(TO, TI), not bool(TO,yes).
```

```
% ---------- different/same ----------
%!trace_rule {"Some object share the same % with value %", ATTR, VALUE}
bool(TO,no) :- all_different(TO, TI, ATTR), state(TI, ID),
               state(TI, ID'), has_attr(ID, ATTR, VALUE),
               has_attr(ID', ATTR, VALUE).

%!trace_rule {"All objects are different."}
bool(TO,yes) :- all_different(TO, TI, ATTR), not bool(TO,no).

%!trace_rule {"Not all objects share the same % with value %",
               ATTR, VALUE}
bool(TO,no) :- all_same(TO, TI, ATTR), state(TI, ID),
               state(TI, ID'), has_attr(ID, ATTR, VALUE),
               not has_attr(ID', ATTR, VALUE).

%!trace_rule {"All objects share the same attributes."}
bool(TO,yes) :- all_same(TO, TI, ATTR), not bool(TO,no).

%!trace_rule {"Object of ID % has % %, whereas object of ID % has % %",
               ID, ATTR, VALUE, ID', ATTR, VALUE'}
bool(TO, yes) :- two_different(TO, TI0, TI1, ATTR), state(TI0, ID),
                 state(TI1, ID'), has_attr(ID, ATTR, VALUE),
                 has_attr(ID', ATTR, VALUE'), VALUE != VALUE'.

%!trace_rule {"Object of ID % has % %, whereas object of ID % does not",
               ID, ATTR, VALUE, ID'}
bool(TO, yes) :- two_different(TO, TI0, TI1, ATTR), state(TI0, ID),
                 state(TI1, ID'), has_attr(ID, ATTR, VALUE),
                 not has_attr(ID', ATTR, VALUE).

%!trace_rule {"Object of ID % has % %, whereas object of ID % does not",
               ID', ATTR, VALUE, ID}
bool(TO, yes) :- two_different(TO, TI0, TI1, ATTR), state(TI0, ID),
                 state(TI1, ID'), not has_attr(ID, ATTR, VALUE),
                 has_attr(ID', ATTR, VALUE).

%!trace_rule {"Both objects share the same %", ATTR}
bool(TO,no) :- two_different(TO, TI0, TI1, ATTR), not bool(TO,yes).

%!trace_rule {"Two objects of IDs %, % share % of value %",
               ID, ID', ATTR, VALUE}
bool(TO, yes) :- two_same(TO, TI0, TI1, ATTR), state(TI0, ID),
                 state(TI1, ID'), has_attr(ID, ATTR, VALUE),
                 has_attr(ID', ATTR, VALUE'), VALUE == VALUE'.

%!trace_rule {"No two objects share attrbute %", ATTR}
bool(TO,no) :- two_same(TO, TI0, TI1, ATTR), not bool(TO,yes).

% ---------- common ----------
%!trace_rule {"Objects with ID %, % share attrbute %",
               ID, ID', ATTR}
```

```
attr(TO, ATTR) :- common(TO, TI0, TI1), state(TI0, ID),
                  state(TI1, ID'), has_attr(ID, ATTR, VALUE),
                  has_attr(ID', ATTR, VALUE), ATTR != name,
                  ATTR != class, ATTR != hposition, ATTR != vposition.

{attr(TO, ATTR): is_attr(ATTR)} = 1 :- common(TO, TI0, TI1).

% ---------- compare ----------
%!trace_rule {"Object of ID % has % %, whereas object of ID % does not",
               ID, ATTR, VALUE, ID'}
state(TO,ID) :- compare(TO, TI0, TI1, VALUE, true), state(TI0, ID),
                state(TI1, ID'), has_attr(ID, ATTR, VALUE),
                not has_attr(ID', ATTR, VALUE).

%!trace_rule {"Object of ID % has % %, whereas object of ID % does not",
               ID', ATTR, VALUE, ID}
state(TO,ID') :- compare(TO, TI0, TI1, VALUE, true), state(TI0, ID),
                 state(TI1, ID'), not has_attr(ID, ATTR, VALUE),
                 has_attr(ID', ATTR, VALUE).

%!trace_rule {"Object of ID % has % %, whereas object of ID % does not",
              ID, ATTR, VALUE, ID'}
state(TO,ID') :- compare(TO, TI0, TI1, VALUE, false), state(TI0, ID),
                 state(TI1, ID'), has_attr(ID, ATTR, VALUE),
                 not has_attr(ID', ATTR, VALUE).

%!trace_rule {"Object of ID % has % %, whereas object of ID % does not",
               ID', ATTR, VALUE, ID}
state(TO,ID) :- compare(TO, TI0, TI1, VALUE, false), state(TI0, ID),
                state(TI1, ID'), not has_attr(ID, ATTR, VALUE),
                has_attr(ID', ATTR, VALUE).

% ---------- boolean ----------
%!trace_rule {"We check the conjunction.
                In this case both branches are true"}
bool(TO,yes) :- and(TO, TI0, TI1), bool(TI0,yes), bool(TI1,yes).

%!trace_rule {"We check the conjunction. A branch is false"}
bool(TO,no) :- and(TO, TI0, TI1), not bool(TO,yes).

%!trace_rule {"We check the disjunction.
               In this case both branches are true"}
bool(TO,yes) :- or(TO, TI0, TI1), bool(TI0,yes), bool(TI1,yes).

%!trace_rule {"We check the disjunction.
               In this case the left branch is true"}
bool(TO,yes) :- or(TO, TI0, TI1), bool(TI0,yes).

%!trace_rule {"We check the disjunction.
               In this case the right branch is true"}
bool(TO,yes) :- or(TO, TI0, TI1), bool(TI1,yes).
```

```
%!trace_rule {"We check the disjunction. Both branches are false"}
bool(TO,no) :- or(TO, TI0, TI1), not bool(TO,yes).

% ---------- unique ----------
%!trace_rule {"The object with ID % is the only object detected
               of this kind", ID}
{state(TO,ID): state(TI,ID)} = 1 :- unique(TO, TI).
:~ unique(TO, TI), state(TO,ID), has_obj_weight(ID, P). [P, (TO, ID)]

% ---------- negate ----------
%!trace_rule {" We negate the existance of object with ID %", ID}
state(TO, ID) :- negate(TO, TI0, TI1), state(TI1, ID),
                 not state(TI0, ID).

%!show_trace ans(V).
```