Appendix A Example

As an example, consider the following domain, which follows the pattern of the experiments:

```
(define (domain rev-2)
  (:requirements :strips)
  (:predicates (f0) (f1) )

  (:action del-all
   :precondition (and  (f0) (f1) )
   :effect (and  (not (f0)) (not (f1)) ) )

  (:action add-f0
   :effect (f0) )

  (:action add-f1
   :precondition (f0)
   :effect (f1) )
)
```

The tool *plasp* translates it to the following ASP quasi-facts:

```
boolean(true).
boolean(false).

% types
type(type("object")).

% variables
variable(variable("f0")).
variable(variable("f1")).

contains(X, value(X, B)) :- variable(X), boolean(B).

% actions
action(action("del-all")).
precondition(action("del-all"), variable("f0"), value(variable("f0"), true))
  :- action(action("del-all")).
precondition(action("del-all"), variable("f1"), value(variable("f1"), true))
  :- action(action("del-all")).
postcondition(action("del-all"), effect(unconditional), variable("f0"), value(variable("f0"), false))
  :- action(action("del-all")).
postcondition(action("del-all"), effect(unconditional), variable("f1"), value(variable("f1"), false))
  :- action(action("del-all")).

action(action("add-f0")).
postcondition(action("add-f0"), effect(unconditional), variable("f0"), value(variable("f0"), true))
  :- action(action("add-f0")).

action(action("add-f1")).
precondition(action("add-f1"), variable("f0"), value(variable("f0"), true))
  :- action(action("add-f1")).
postcondition(action("add-f1"), effect(unconditional), variable("f1"), value(variable("f1"), true))
  :- action(action("add-f1")).
```

In the simple ELP encoding, the first rules will give rise to multiple possible world views, one that contains answer sets with `chosen("del-all")`, `occurs("del-all",1)`, `holds("f0",true,0)`, `holds("f1",true,0)`, `relevant("f0")`, and `relevant("f1")`, one world view that contains answer sets with `chosen("add-f0")` and `occurs("add-f0",1)`, and one with answer sets containing `chosen("add-f1")`, `occurs("add-f1",1)`, `holds("f0",true,0)`, and `relevant("f0")`.

When we set the constant `horizon` to 2, more world views are created, based on the three mentioned above, one for each pair of actions of the three available ones. Each world view

will contain at most one answer set. Many of these answer sets turn out to be invalid immediately, for instance any answer set containing `occurs("add-f0",1)` and `occurs("add-f1",2)` will be violating a constraint, as the precondition of `add-f1` is not relevant. Others are invalidated because the preconditions of actions are not met. A few others derive `noreversal`, for instance `occurs("del-all",1)`, `occurs("add-f0",2)`, `occurs("add-f0",3)`, as we have `holds("f1",true,0)` but not `holds("f1",true,3)`.

We can check that the only world view with an answer set, in which `noreversal` is not derived, is the one in which `occurs("del-all",1)`, `occurs("add-f0",2)`, `occurs("add-f1",3)` hold. Indeed, `del-all` is the only universally uniformly reversible action, and its only reverse plan of length 2 is ⟨ `add-f0`, `add-f1` ⟩.

The simple ASP encoding works in a very similar way. Since the simple ELP encoding has at most one answer set per world view, we simple turn the "epistemic guesses" into "standard guesses", so instead of an answer set encapsulated in a world view, it is just an answer set, and also there, one answer set exists for the example, in which `occurs("del-all",1)`, `occurs("add-f0",2)`, `occurs("add-f1",3)` hold.

Concerning the general ELP encoding, similar world views as above are created. But in that encoding, multiple answer sets can exist in a world view: for each variable not in the precondition of the chosen action, there will be answer sets in which the variable is true, and answer sets in which it is false. So, any world view, in which `chosen("del-all"))`, `occurs("del-all",1)`, `holds("f0",true,0)`, `holds("f1",true,0)` hold, will still have at most a single answer set, as all variables occur in the precondition of `del-all`. It is easy to see that the reverse plan is then in a single-answer-set world view similar to the one in the simple ELP encoding.

On the other hand, in world views containing `chosen("add-f0")` and `occurs("add-f0",1)` four potential answer sets can exist, one with `holds("f0",true,0)` and `holds("f1",true,0)`, one with `holds("f0",false,0)` and `holds("f1",true,0)`, one with `holds("f0",true,0)` and `holds("f1",false,0)`, and one with `holds("f0",false,0)`, `holds("f1",false,0)`.

Let us have a look at the world view containing `occurs("add-f0",1)`, `occurs("del-all",2)`, `occurs("del-all",3)`. For this, `inapplicable` will be derived because the preconditions for `occurs("del-all",3)` are not met in any of the answer sets, and also because the precondition for `occurs("del-all",2)` is not met in those answer sets in which `holds("f1",false,0)` is true. Therefore, the constraint `:- not &k{ ~ inapplicable}.` is violated for this world view.

Maybe more interesting is the world view containing `occurs("add-f0",1)`, `occurs("add-f1",2)`, `occurs("del-all",3)`. Here, `inapplicable` will not be derived, as the preconditions of the actions hold in all answer sets. But consider the answer set containing `holds("f0",true,0)` and `holds("f1",true,0)`: neither `holds("f0",true,3)` nor `holds("f1",true,3)` is derived, so `noreversal` is derived. `noreversal` is also true in the other answer sets except the one containing `holds("f0",false,0)` and `holds("f1",false,0)`. The constraint `:- not &k{ ~ noreversal}.` is thus violated for this world view.

The general ASP encoding works in a rather different way. Here, one candidate answer set will be created for each action to be reversed, one completion of the initial state and one candidate reverse plan.

So there is still only one answer set candidate containing `chosen("del-all"))`, `occurs("del-all",1)`, `holds("f0",true,0)`, `holds("f1",true,0)`. For `occurs("add-f0",1)`, `occurs("del-all",2)`, `occurs("del-all",3)` there will be four answer set candidates, similar to the four answer sets in the world view in the general ELP encoding, similar for `occurs("add-f0",1)`, `occurs("del-all",2)`, `occurs("del-all",3)`, there will also be four answer set candidates.

Let us first see what happens with the reverse plan answer set candidate containing `chosen("del-all"))`, `occurs("del-all",1)`, `holds("f0",true,0)`, `holds("f1",true,0)`. Eventually, `samestate` and `planvalid`, and `reversePlan` are derived for this answer set, which means that `holds(V,Val,T)` will be derived for all combinations of variables, values and times. It should be noted that doing this will not invalidate any derivation done earlier, so it is an answer set.

Now consider the answer set candidates containing `occurs("add-f0",1)`, `occurs("del-all",2)`, `occurs("del-all",3)`. In none of these, `planvalid` will be derived, as either the preconditions of `occurs("del-all",2)` are not met (in answer set candidates having `holds("f1",false,0)`) or the preconditions of `occurs("del-all",3)` are not met. So, `reversePlan` will not hold in any of these answer set candidates, which means that they all violate the constraint `:- not reversePlan.`

Now for the answer set candidates containing `occurs("add-f0",1)`, `occurs("add-f1",2)`, `occurs("del-all",3)`, `planvalid` will be derived in all of them, but `samestate` only in the one containing `holds("f0",false,0)` and `holds("f1",false,0)`. This means, that in the other three answer set candidates `reversePlan` will not hold, violating the constraint `:- not reversePlan.` For the remaining one, the constraint is satisfied, however the saturation rule derives `holds(V,Val,T)` for all combinations of variables, values and times. This "inflated" answer set candidate is not stable any longer: we can form a subset of it, in which exactly the atoms of one of the other three constraint-violating candidates are true plus `reversePlan`; the obtained interpretation also satisfies the program and therefore is a counterexample for the stability of the saturated interpretation. One might object that `reversePlan` is unsupported in the counterexample, but supportedness is not a requirement for countermodels. So none of the candidate answer sets containing `occurs("add-f0",1)`, `occurs("add-f1",2)`, `occurs("del-all",3)` turned out to be answer sets (for quite different reasons).