Exercises

Answer Set Solving in Practice

Exercise 2 (Answer Set Programming by Examples - with Variables)

You can check your answers running `clingo` with the program files in the directory `exercise02`. For example, if the file is named `example.lp`, the command `clingo example.lp 0` will compute all stable models.

Exercise 2.1 (Positive programs and negation with variables)

Follow some programs and their stable models:

```
a(X) :- b(X).
b(1). b(2).
---------------
{ a(1), a(2),
  b(1), b(2)  }
```

```
a(X) :- b(X), c(X).
b(1). b(2). c(1).
--------------------
{ a(1),
  b(1), b(2), c(1) }
```

```
a(X) :- b(X,Y), c(Y).
b(1,5). b(2,10). c(10).
-------------------------
{ a(2),
  b(1,5), b(2,10), c(10) }
```

```
a(X) :- b(X), Y=X+1,
        not b(Y).
b(1). b(2).
--------------------
{ a(2),
  b(1), b(2) }
```

```
a(X) :- b(X),
        not c(X).
b(1). b(2). c(1).
--------------------
{ a(2),
  b(1), b(2), c(1) }
```

```
a(X) :- b(X,Y),
        not c(Y).
b(1,5). b(2,10). c(10).
-------------------------
{ a(1),
  b(1,5), b(2,10), c(10) }
```

Find the stable models of the next programs:

```
a(X) :- b(X).
b(1). b(2).
b(3).
-------------
```

```
a(X) :- b(X), c(X).
b(1). b(2). c(1).
c(2). c(3).
-------------------
```

```
a(X) :- b(X,Y), c(Y).
b(1,5). b(2,10). c(10).
c(5).
-----------------------
```

```
a(X) :- b(X), Y=X+1,
        not b(Y).
b(1). b(3).
--------------------
```

```
a(X) :- b(X),
        not c(X).
b(1). b(2). c(1).
b(3). c(2).
-----------------
```

```
a(X) :- b(X,Y),
        not c(Y).
b(1,5). b(2,10).
-----------------
```

Exercise 2.2 (Programming with choice rules, constraints and variables)

Follow some programs and their stable models:

```
{ a(X) : b(X) }.
b(1). b(2).
---------------
{ b(1), b(2) }
{ a(1),
  b(1), b(2) }
{ a(2),
  b(1), b(2) }
{ a(1), a(2),
  b(1), b(2) }
```

```
1 { a(X) : b(X) } 1.
b(1). b(2).
c(X) :- a(X).
--------------------
{ a(1), c(1),
  b(1), b(2) }
{ a(2), c(2),
  b(1), b(2) }
```

```
{ a(X) : b(X,Y), c(Y) }.
b(1,5). b(2,10). c(10).
d(X) :- a(X).
-------------------------
{ b(1,5), b(2,10), c(10) }
{ a(2),   d(2),
  b(1,5), b(2,10), c(10) }
```

```
{ a(X) : b(X) }.
b(1). b(2).
:- b(X), Y=X-1, a(Y).
---------------------
{ b(1), b(2) }
{ a(2),
  b(1), b(2) }
```

```
1 { a(X) : b(X) } 1.
b(1). b(2).
c(X) :- a(X).
:- b(X), not a(X).
--------------------
```

```
{ a(X) : b(X,Y), c(Y) }.
b(1,5). b(2,10). c(10).
d(X) :- a(X).
:- a(X), d(X).
-------------------------
{ b(1,5), b(2,10), c(10) }
```

Find the stable models of the next programs:

```
{ a(X) : b(X) }.
b(1). b(2).
b(3).
---------------
```

```
1 { a(X) : b(X) } 1.
b(1). b(2).
c(X) :- a(X).
b(3).
--------------------
```

```
{ a(X) : b(X,Y), c(Y) }.
b(1,5). b(2,10). c(10).
d(X) :- a(X).
b(3,10).
-------------------------
```

```
{ a(X) : b(X) }.
b(1). b(2).
:- b(X), Y=X-1, a(Y).
b(3).
---------------------
```

```
1 { a(X) : b(X) } 1.
b(1).
c(X) :- a(X).
:- b(X), not a(X).
--------------------
```

```
{ a(X) : b(X,Y), c(Y) }.
b(1,5). b(2,10). c(10).
d(X) :- a(X).
:- a(X), not d(X).
-------------------------
```

Exercise 2.3 (Aggregates and variables)

Follow some programs and the part of their stable models that is shown by `clingo`:

```
b(1..2).
{ a(X): b(X) }.
:- 2 #sum{1,X: a(X)}.
#show a/1.
---------------
{ }
{ a(1) }
{ a(2) }
```

```
b(1..2).
{ a(X) : b(X) }.
:- 2 #sum{1: a(X)}.
#show a/1.
---------------
{ }
{ a(1) }
{ a(2) }
{ a(1), a(2) }
```

```
b(1..2).
{ a(X) : b(X) }.
:- 2 #sum{X: a(X)}.
#show a/1.
---------------
{ }
{ a(1) }
```

```
b(1..2).
1 {a(X,Y): b(X),b(Y)} 2.
:-2 #sum{1,X,Y: a(X,Y)}.
#show a/2.
---------------
{ a(1,1) }
{ a(1,2) }
{ a(2,1) }
{ a(2,2) }
```

```
b(1..2).
2 {a(X,Y): b(X),b(Y)}.
:- 2 #sum{1,X: a(X,Y)}.
#show a/2.
---------------
{ a(1,1), a(1,2) }
{ a(2,1), a(2,2) }
```

```
b(1..2).
2 {a(X,Y): b(X),b(Y)}.
:- 2 #sum{X: a(X,Y)}.
#show a/2.
---------------
{ a(1,1), a(1,2) }
```

Find the stable models of the next programs:

```
b(1..2).
{ a(X) : b(X) }.
c(X):- a(X).
:- 2 #sum{1,X: c(X)}.
#show c/1.
--------------------
```

```
b(1..2).
{ a(X) : b(X) }.
c(X):- a(X).
:- 1 #sum{1: c(X)}.
#show c/1.
--------------------
```

```
b(1..2).
{ a(X) : b(X) }.
c(X):- a(X).
:- 2 #sum{X: a(X), c(X)}.
#show c/1.
------------------------
```

```
b(1..2).
1{a(X,Y): b(X), b(Y)}2.
:-2 #sum{1,Y,X: a(X,Y)}.
#show a/2.
--------------------
```

```
b(1..2).
2 {a(X,Y): b(X), b(Y)}.
:- 2 #sum{1,Y: a(X,Y)}.
#show a/2.
--------------------
```

```
b(1..2).
2 {a(X,Y): b(X), b(Y)}.
:- 2 #sum{Y: a(X,Y)}.
#show a/2.
-------------------------
```

Exercise 2.4 (Conditional literals in the body)

Follow some programs and the part of their stable models that is shown by `clingo`:

```
a(1..2).
b(1..2).
c :- a(X) : b(X).
#show c/0.
----------------
{ c }
```

```
a(1).
b(1..2).
c :- a(X) : b(X).
#show c/0.
----------------
{ }
```

```
a(1).
c :- a(X) : b(X).
#show c/0.
----------------
{ c }
```

```
a(1..2,1..2).
b(1..2,1..2).
c :- a(X,Y) : b(X,Y).
#show c/0.
--------------------
{ c }
```

```
a(1,1..2).
b(1..2,1..2).
c :- a(X,Y) : b(X,Y).
#show c/0.
--------------------
{ }
```

```
a(1,1..2).
c :- a(X,Y) : b(X,Y).
#show c/0.
--------------------
{ c }
```

Find the stable models of the next programs:

```
a(1..2).
b(1..2).
c(1..2).
c :- a(X) : b(X), c(X).
#show c/0.
----------------------
```

```
a(1).
b(1..2).
c :- a(X) :     b(X),
             not c(X).
#show c/0.
----------------------
```

```
a(1).
b(1..2).
c :- a(X) : b(X), c(X).
#show c/0.
----------------------
```

```
a(1..2,1..2).
b(1..2,1..2).
c :- a(X,Y) :   b(X,Y),
               not c(X).
#show c/0.
----------------------
```

```
a(1,1..2).
b(1..2,1..2).
c(1).
c :- a(X,Y) : b(X,Y),
                 c(X).
#show c/0.
----------------------
```

```
a(1,1..2).
c(1).
c :- a(X,Y) : b(X,Y), c(X).
#show c/0.
--------------------------
```

Exercise 2.5 (Optimization)

Follow some programs and the part of their stable models that is shown by `clingo` (use option `--opt-mode=optN` to compute them):

```
b(1..2).
1 { a(X) : b(X) }.
#minimize{ 1,X: a(X)}.
#show a/1.
----------------------
{ a(1) }
{ a(2) }
```

```
b(1..2).
1 { a(X) : b(X) }.
#minimize{ 1: a(X)}.
#show a/1.
--------------------
{ a(1) }
{ a(2) }
{ a(1), a(2) }
```

```
b(1..2).
1 { a(X) : b(X) }.
#minimize{ X: a(X)}.
#show a/1.
--------------------
{ a(1) }
```

```
b(1..2).
1 {a(X,Y): b(X), b(Y)}.
#minimize{1,X,Y:a(X,Y)}.
#show a/2.
------------------------
{ a(1,1) }
{ a(1,2) }
{ a(2,1) }
{ a(2,2) }
```

```
b(1..2).
2 {a(X,Y): b(X), b(Y)}.
#minimize{1,X: a(X,Y)}.
#show a/2.
------------------------
{ a(1,1), a(1,2) }
{ a(2,1), a(2,2) }
```

```
b(1..2).
2 {a(X,Y): b(X), b(Y)}.
#minimize{X: a(X,Y)}.
#show a/2.
------------------------
{ a(1,1), a(1,2) }
```

Find the *optimal* stable models of the next programs:

```
b(1..2).
1 { a(X) : b(X) }.
c(X) :- a(X).
#minimize{ 1,X: c(X)}.
#show c/1.
----------------------
```

```
b(1..2).
1 { a(X) : b(X) }.
c(X) :- a(X).
#minimize{ 1: c(X)}.
#show c/1.
--------------------
```

```
b(1..2).
1 { a(X) : b(X) }.
c(X) :- a(X).
#minimize{ X: a(X), c(X)}.
#show c/1.
--------------------------
```

```
b(1..2).
1 {a(X,Y): b(X), b(Y)}.
#minimize{1,Y,X:a(X,Y)}.
#show a/2.
------------------------
```

```
b(1..2).
2 {a(X,Y): b(X), b(Y)}.
#minimize{1,Y: a(X,Y)}.
#show a/2.
------------------------
```

```
b(1..2).
2 {a(X,Y): b(X), b(Y)}.
#minimize{ Y: a(X,Y)}.
#show a/2.
------------------------
```

Exercise 2.6 (Additional functions and statements)

Run the following program and try to understand it:

```
% python functions definition
#script(python)
def get_value():
    return 1
#end.

% #true and #false
true :- #true.
notfalse :- not #false.

% python functions usage
a(X) :- X = @get_value().

% constants
#const n=2.
a(n).

% intervals
a(3..4).

% show statements
#show.                                    % show no atoms if no predicate is #shown
#show a/1.                                % show true atoms of predicate a/1
#show hold("true and notfalse") : true,   % show term if condition holds
                               notfalse.

%
% projection: use with options
%   0
% and
%   0 --project
%
{ b(1) }.
{ c(1) }.
#project b/1.
#show b/1.
#show c/1.
```