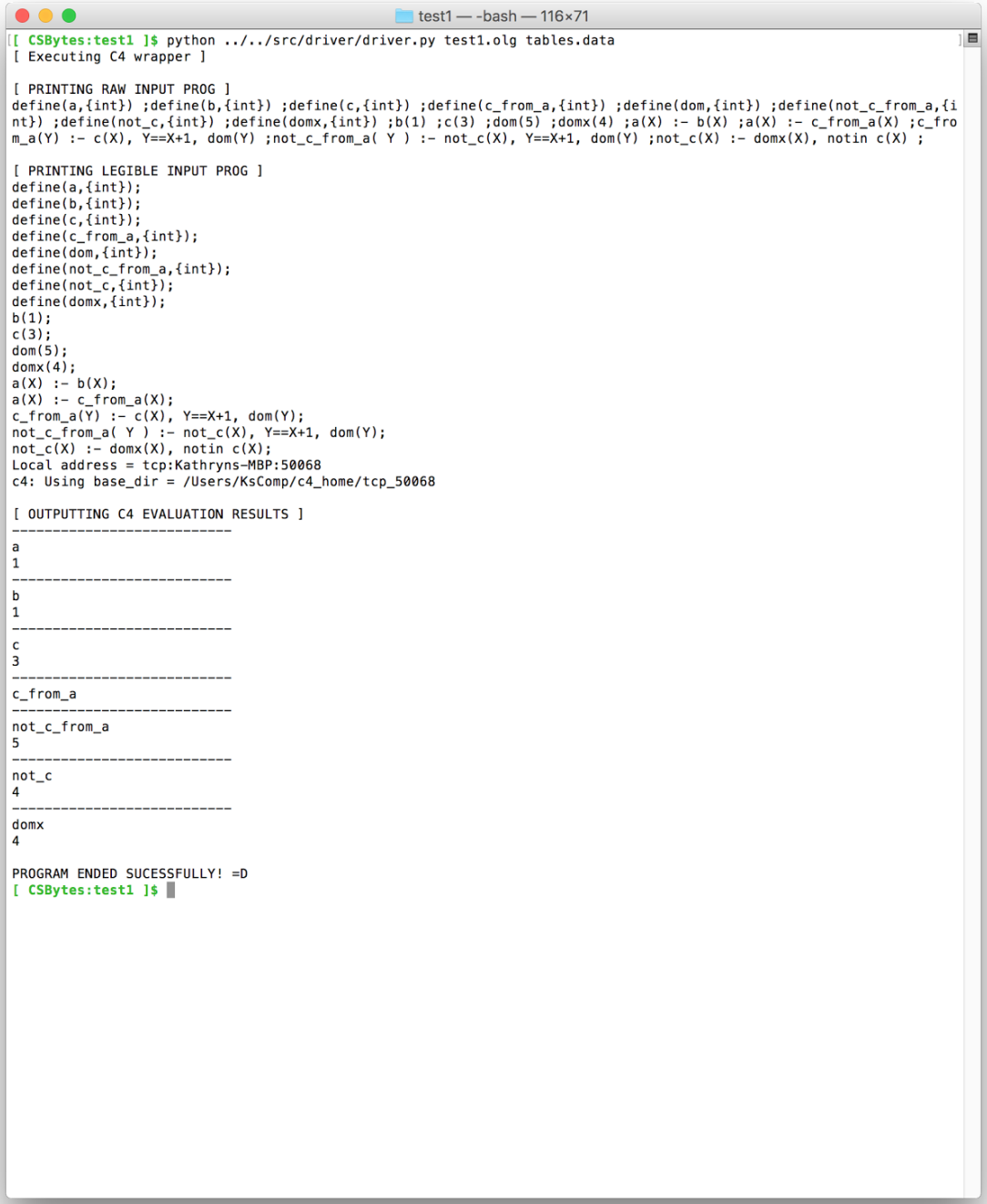
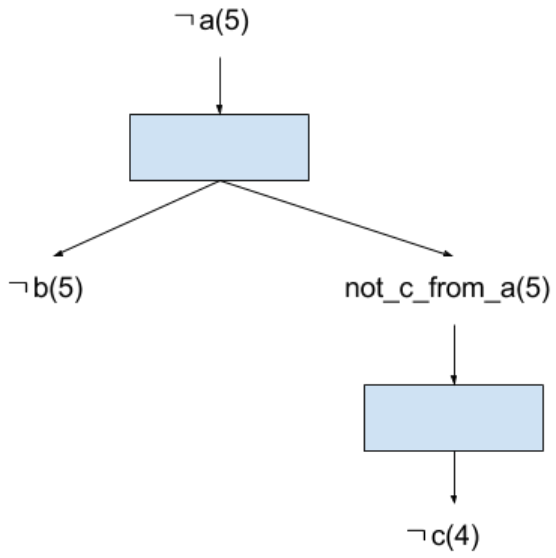
Program :

```
define(a,{int}) ;
define(b,{int}) ;
define(c,{int}) ;
define(c_from_a,{int}) ;
define(dom,{int}) ;
define(not_c_from_a,{int}) ;
define(not_c,{int}) ;
define(domx,{int}) ;

b(1) ;
c(3) ;

dom(5) ;
domx(4) ;

a(X) :- b(X) ;
a(X) :- c_from_a(X) ;
c_from_a(Y) :- c(X), Y==X+1, dom(Y) ;

not_c_from_a( Y ) :- not_c(X), Y==X+1, dom(Y) ;
not_c(X) :- domx(X), notin c(X) ;
```

Results :


test1 — -bash — 116×71
[CSBytes:test1]$ python ../../src/driver/driver.py test1.olg tables.data
[Executing C4 wrapper]
[PRINTING RAW INPUT PROG]
define(a,{int}) ;define(b,{int}) ;define(c,{int}) ;define(c_from_a,{int}) ;define(dom,{int}) ;define(not_c_from_a,{int}) ;define(not_c,{int}) ;define(domx,{int}) ;b(1) ;c(3) ;dom(5) ;domx(4) ;a(X) :- b(X) ;a(X) :- c_from_a(X) ;c_from_a(Y) :- c(X), Y==X+1, dom(Y) ;not_c_from_a(Y) :- not_c(X), Y==X+1, dom(Y) ;not_c(X) :- domx(X), notin c(X) ;
[PRINTING LEGIBLE INPUT PROG]
define(a,{int});
define(b,{int});
define(c,{int});
define(c_from_a,{int});
define(dom,{int});
define(not_c_from_a,{int});
define(not_c,{int});
define(domx,{int});
b(1);
c(3);
dom(5);
domx(4);
a(X) :- b(X);
a(X) :- c_from_a(X);
c_from_a(Y) :- c(X), Y==X+1, dom(Y);
not_c_from_a(Y) :- not_c(X), Y==X+1, dom(Y);
not_c(X) :- domx(X), notin c(X);
Local address = tcp:Kathryns-MBP:50068
c4: Using base_dir = /Users/KsComp/c4_home/tcp_50068
[OUTPUTTING C4 EVALUATION RESULTS]
a
1
b
1
c
3
c_from_a
not_c_from_a
5
not_c
4
domx
4
PROGRAM ENDED SUCESSFULLY! =D
[CSBytes:test1]$

Provenance Graph :



Notes :

The domains per attribute per rule are populated using the evaluation results from the previous iteration. Observe the domain for the ***X*** attribute in the ***not_c*** rule is precisely the domain of ***Y*** for ***c_from_a*** minus ***1***. Applying the complementary arithmetic transformation on the domain constraints is essential for preserving semantics.