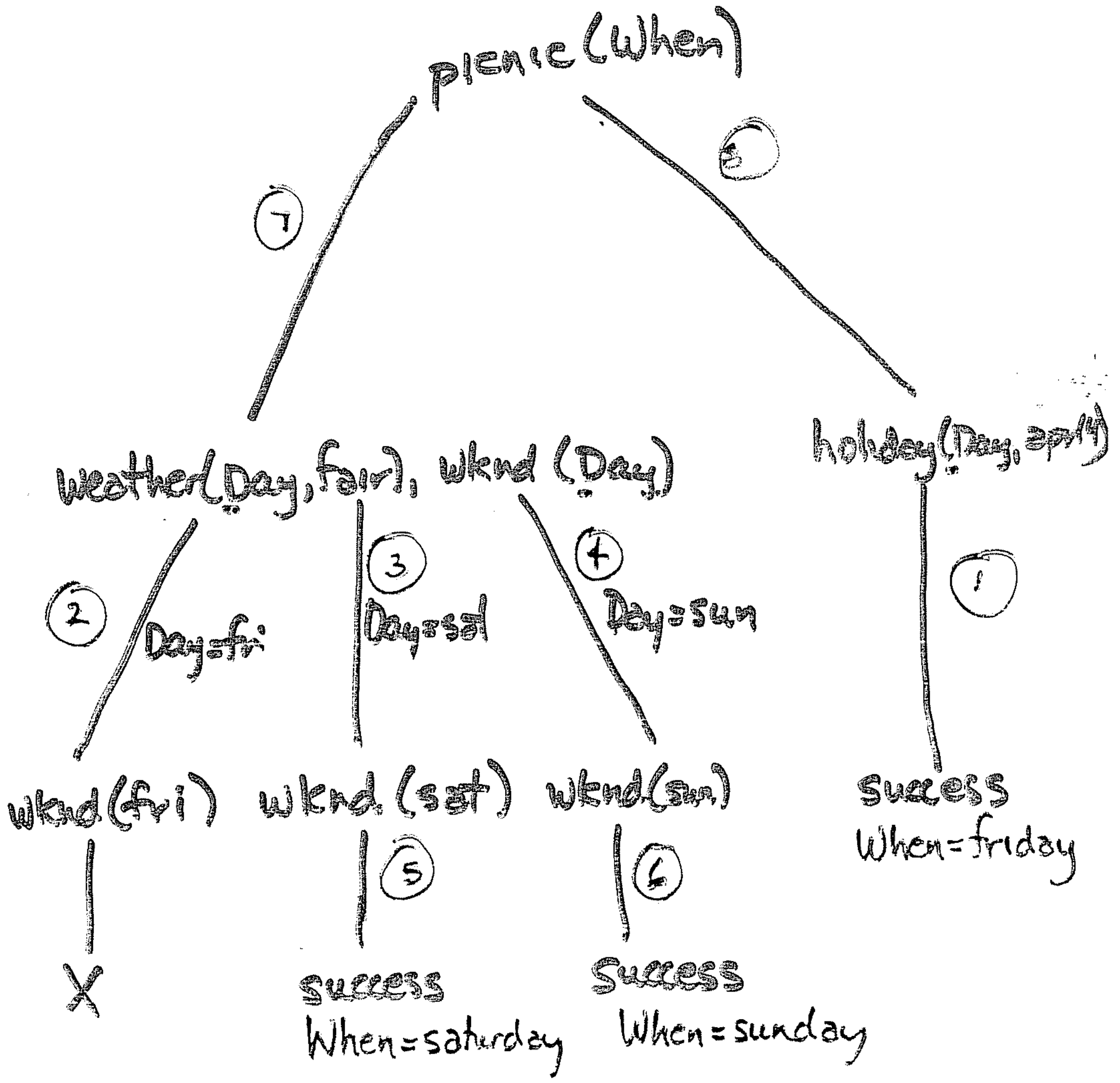
Prolog Search Tree

(Prolog uses depth-first search (DFS))

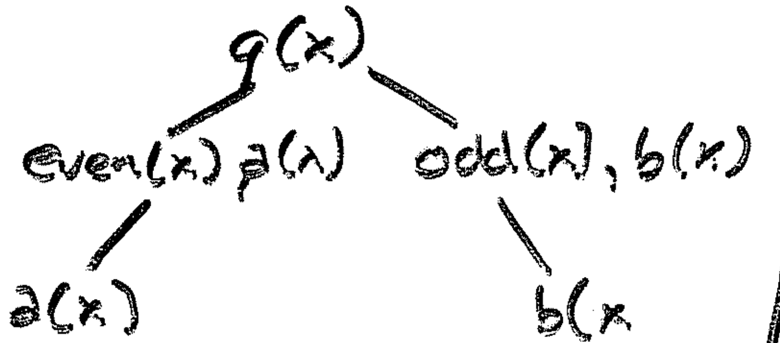


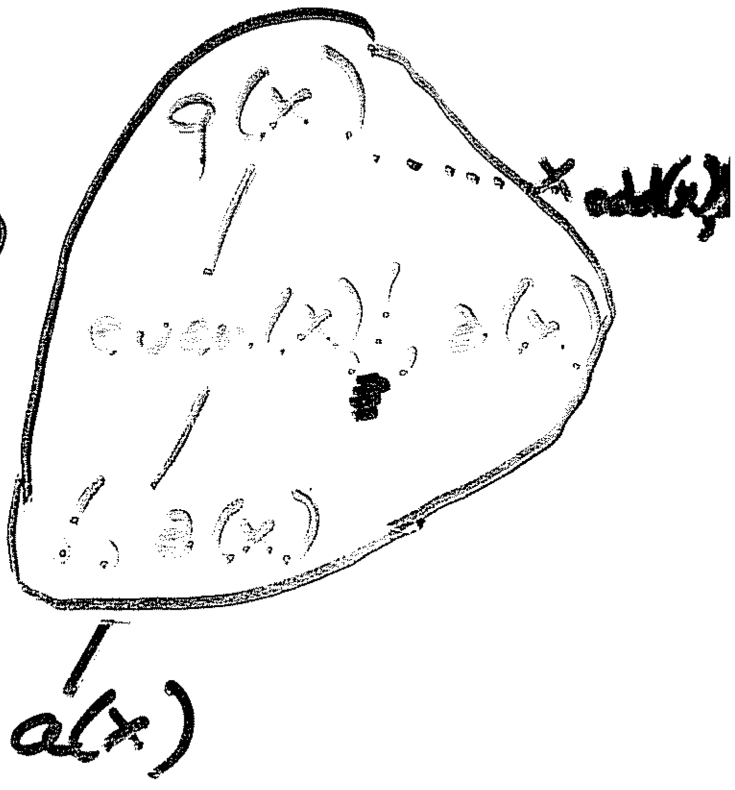
1. Cut Can Reduce Your Search Space

Cut can be used to improve the efficiency of search by reducing Prolog's search space. E.g.,

When two predicates are mutually exclusive.

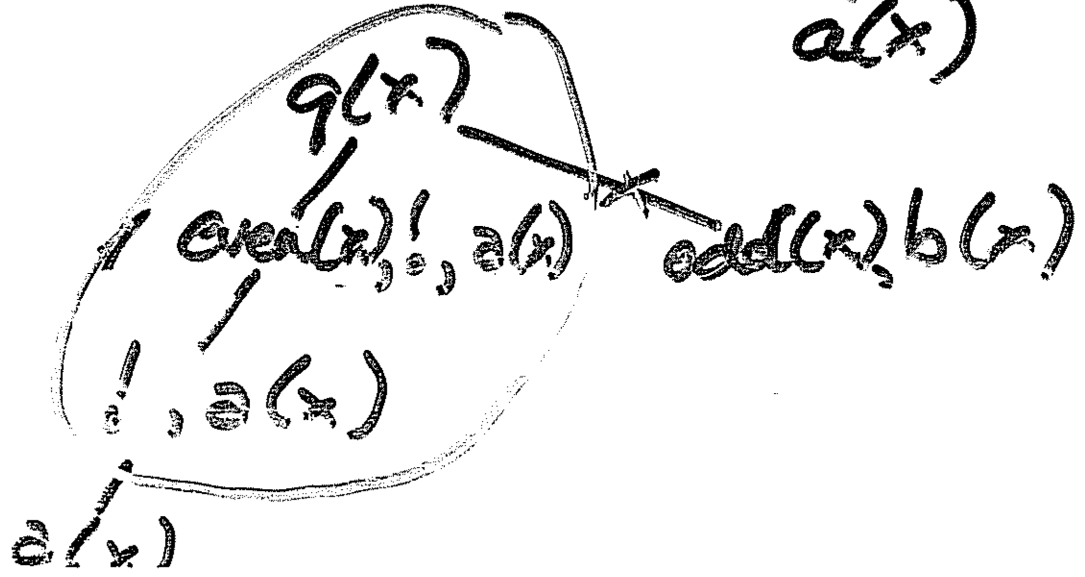
```
q(X) :- even(X), a(X).
q(X) :- odd(X), b(X).
```





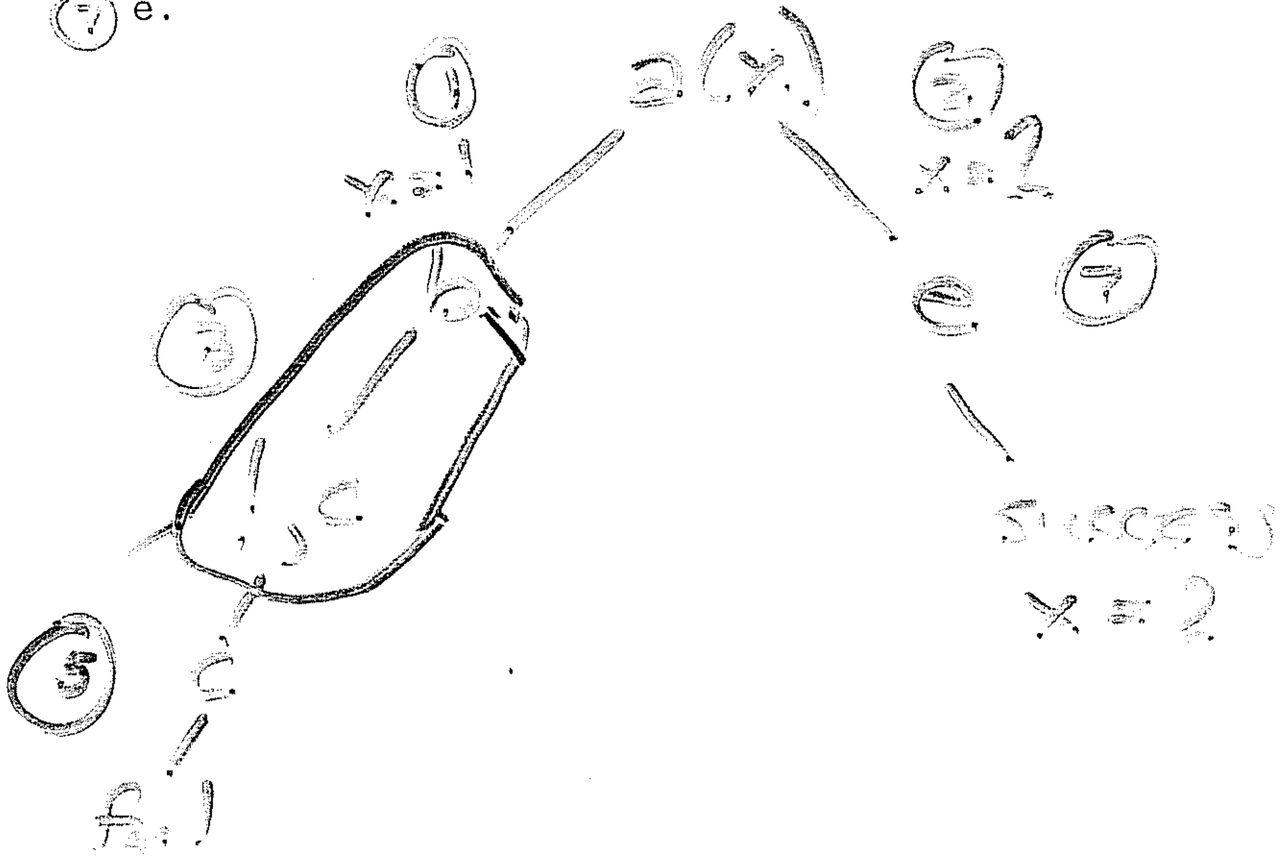
With cut

```
q(X) :- even(X), ! a(X).
q(X) :- odd(X), b(X).
```



1. Reducing Search Space (cont.)

```
a(1) :- b.
a(2) :- e.
b    :- !, c.
b    :- d.
c    :- fail.
d.
e.
```



2. Cut Can Implement Exceptions to Rules

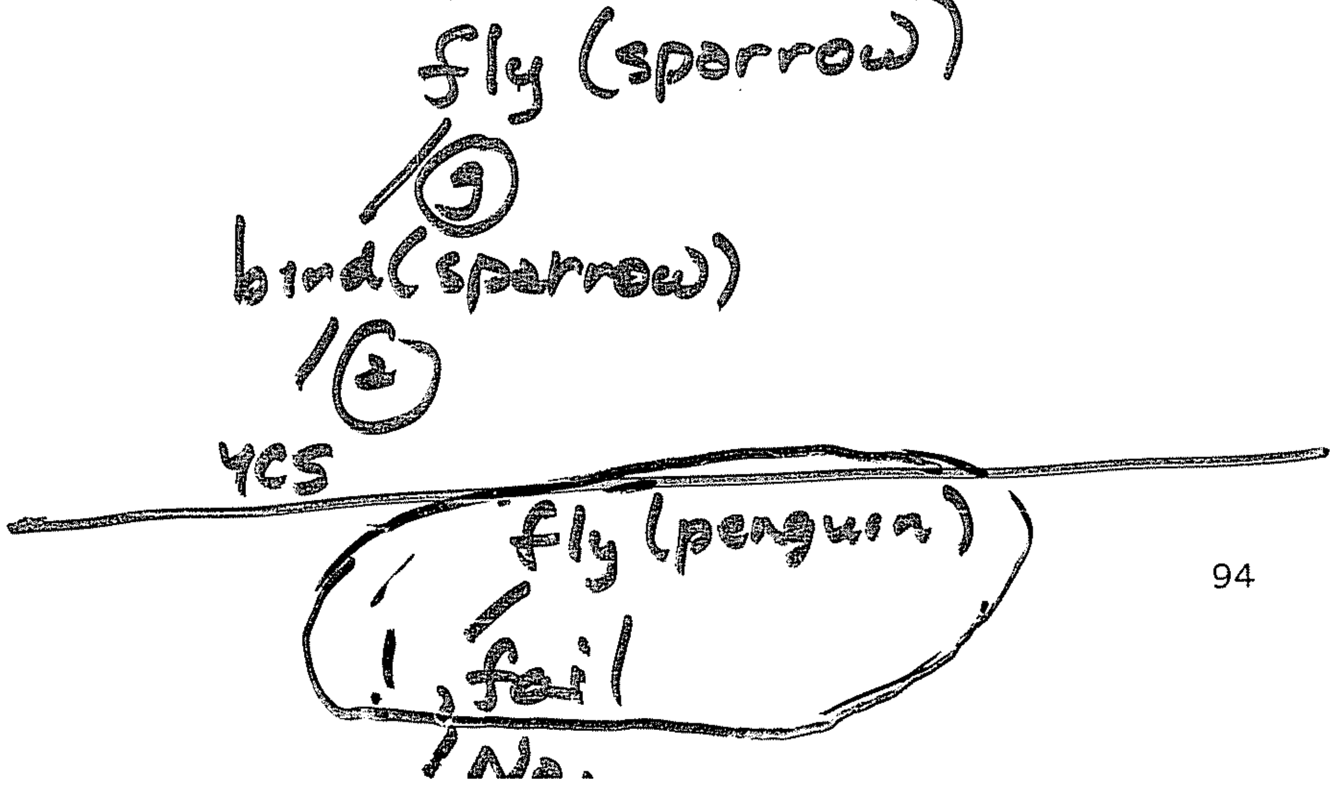
I.e., "To get the right answer".

Cut can be used to encode exceptions to rules. This is use in AI default reasoning.

```
1 bird(eagle).
2 bird(sparrow).
3 bird(penguin).
4 fly(penguin) :- !, fail.
5 fly(X) :- bird(X).
```

Query: fly(sparrow).

Query: fly(penguin).



+ not/1 length/2

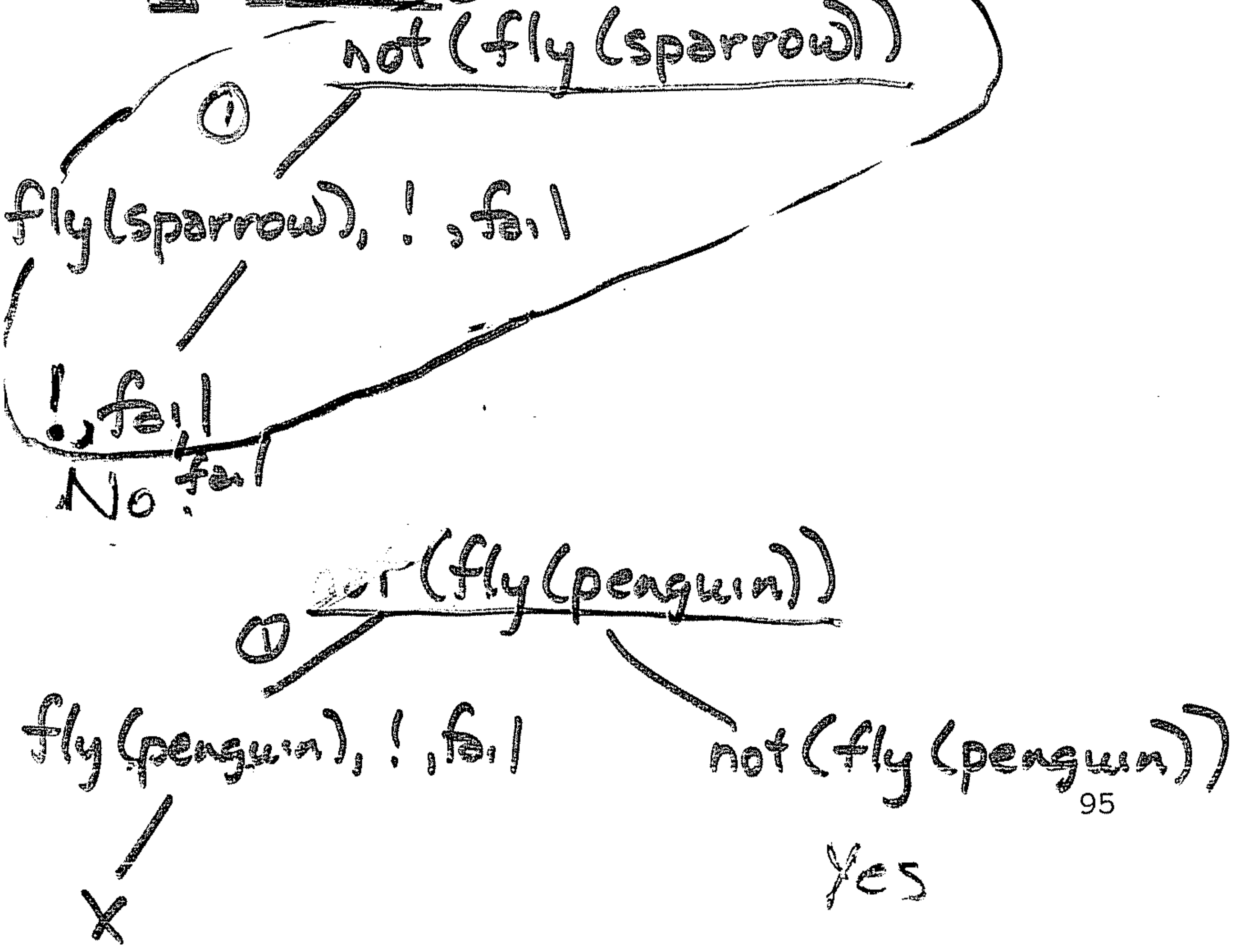
3. Cut Can Implement NAF

Cut can be used to implement negation as failure.

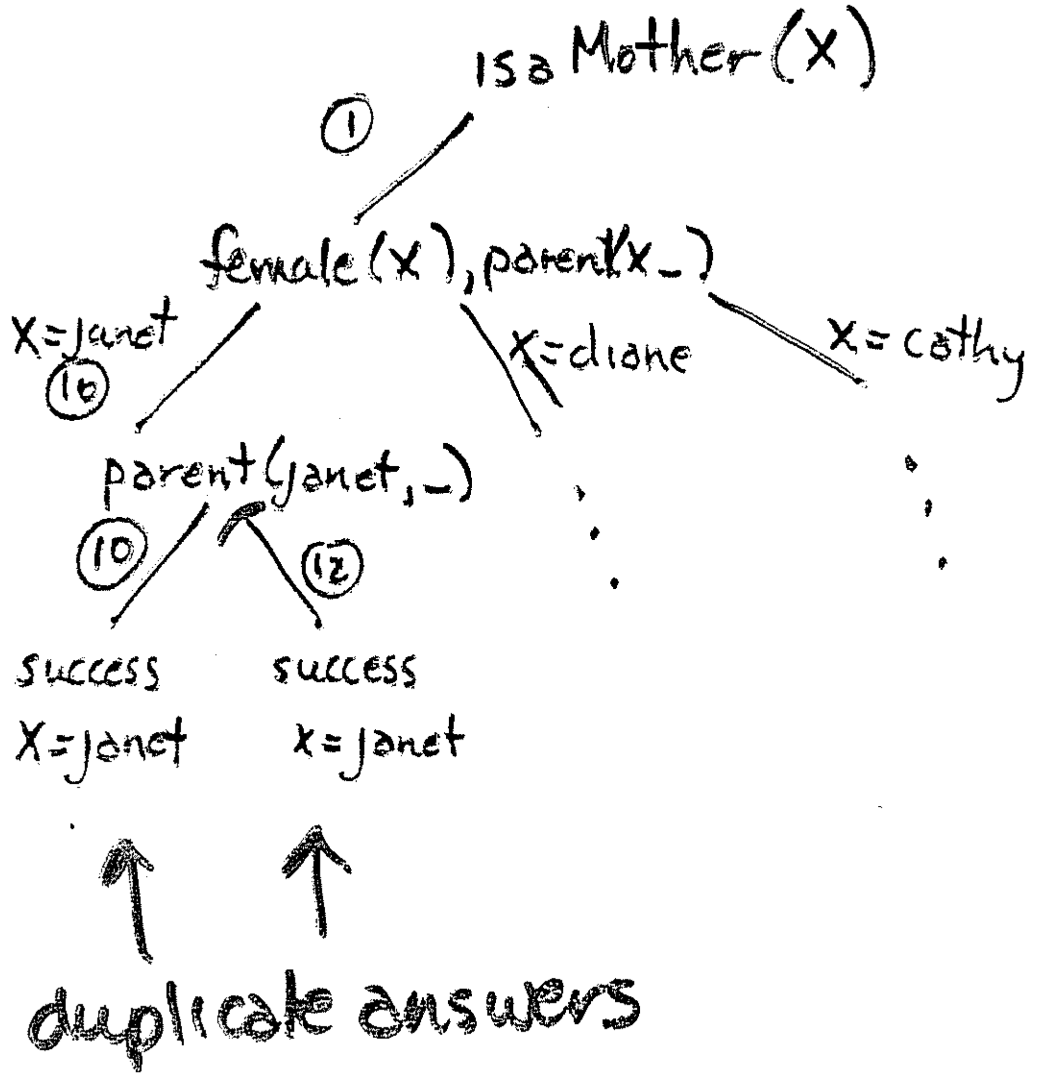
```
(1) not(X) :- X, !, fail.
(2) not(X).
```

Note that not is a meta-logical predicate. It takes a predicate as an argument. E.g.,

```
not(fly(penguin))
```

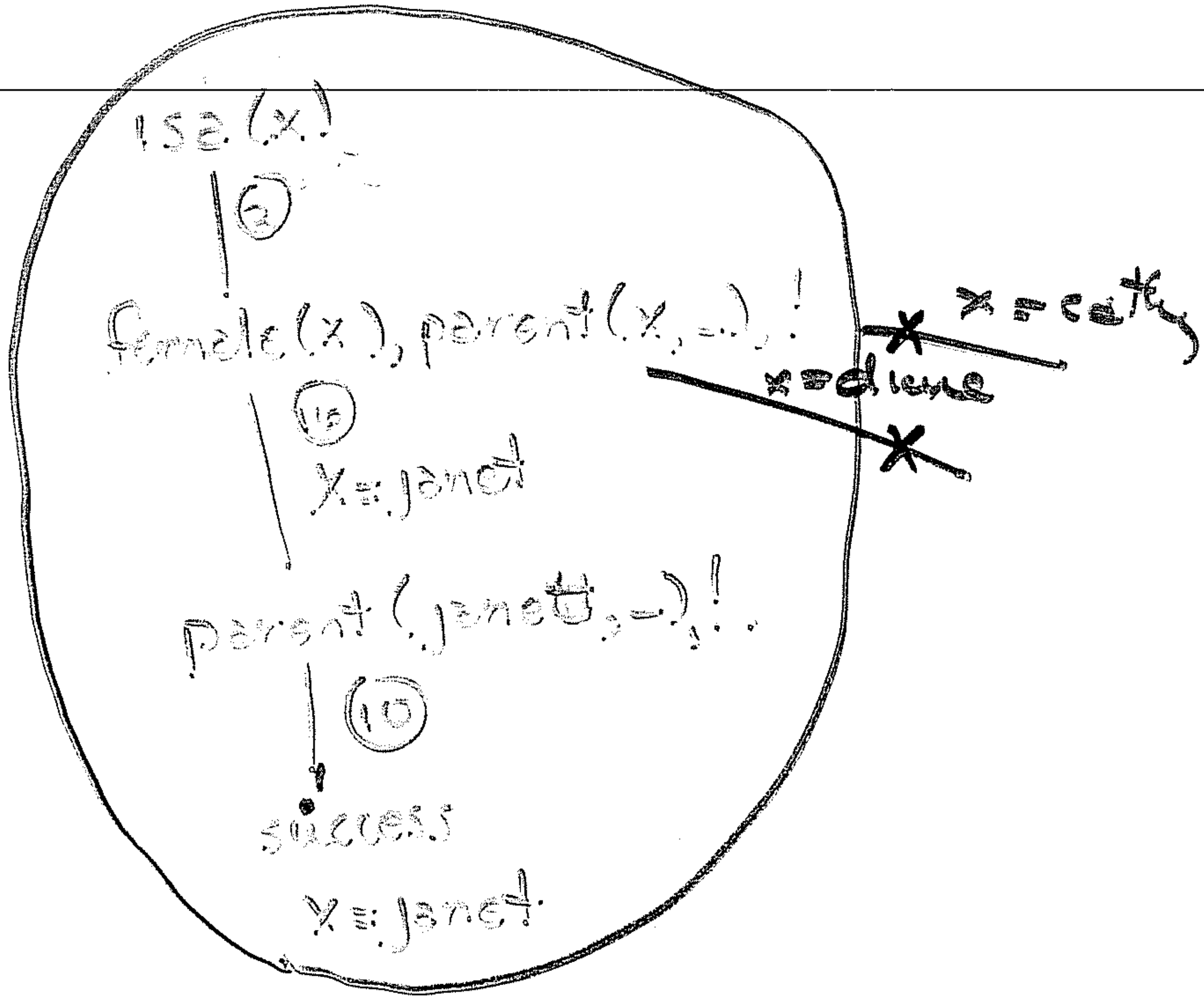


1)


isaMother(X)
1
female(X), parent(X,_)
X=janet
10
X=diane
X=cathy
parent(janet,_)
10
12
success
X=janet
success
x=janet
duplicate answers

Incorrect

3)

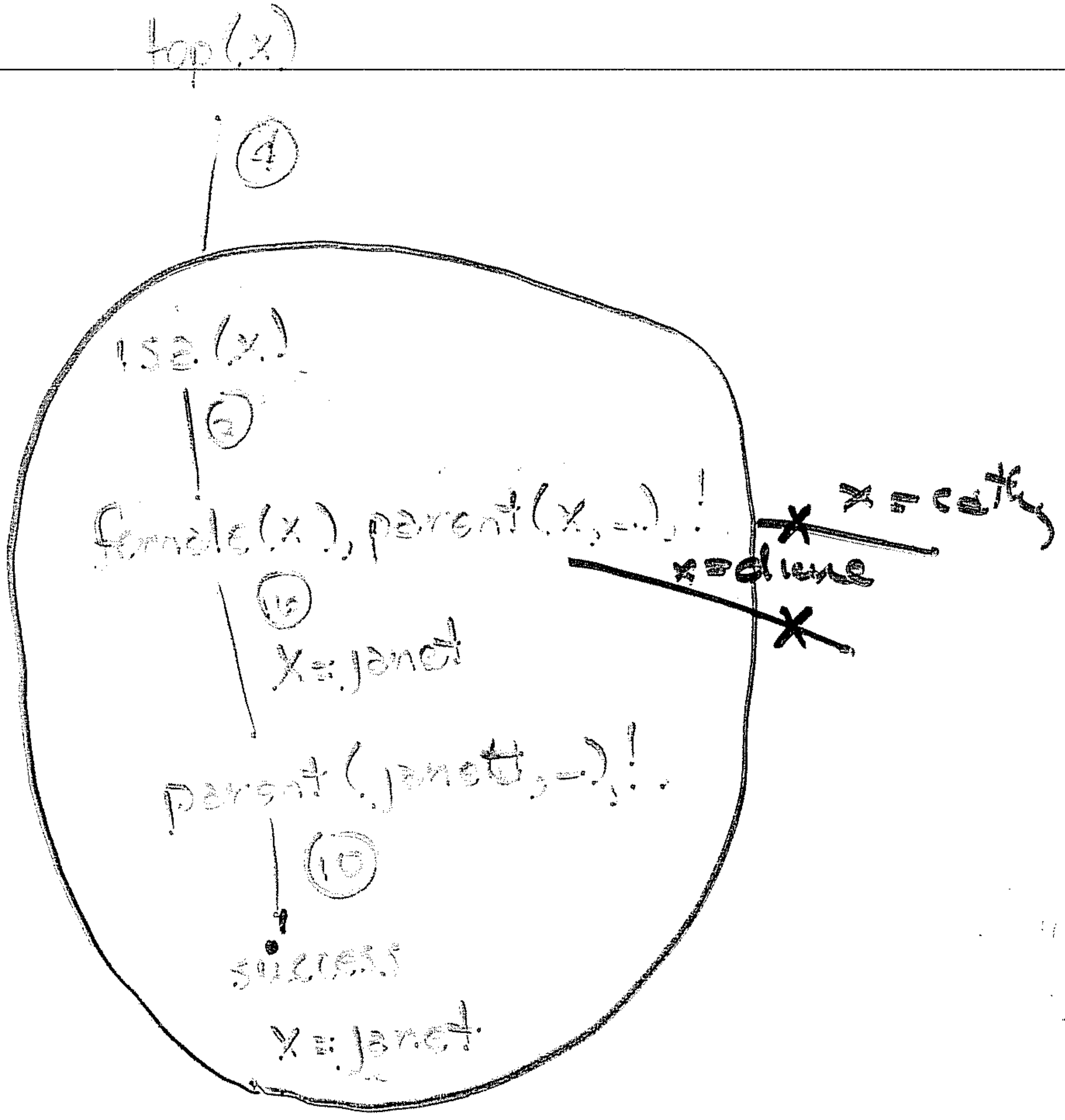


!se(x)
2
female(x), parent(x,_),!
15
x=janet
x=cathy
x=diane
parent(janet,_),!
10
success
x=janet

Incorrect

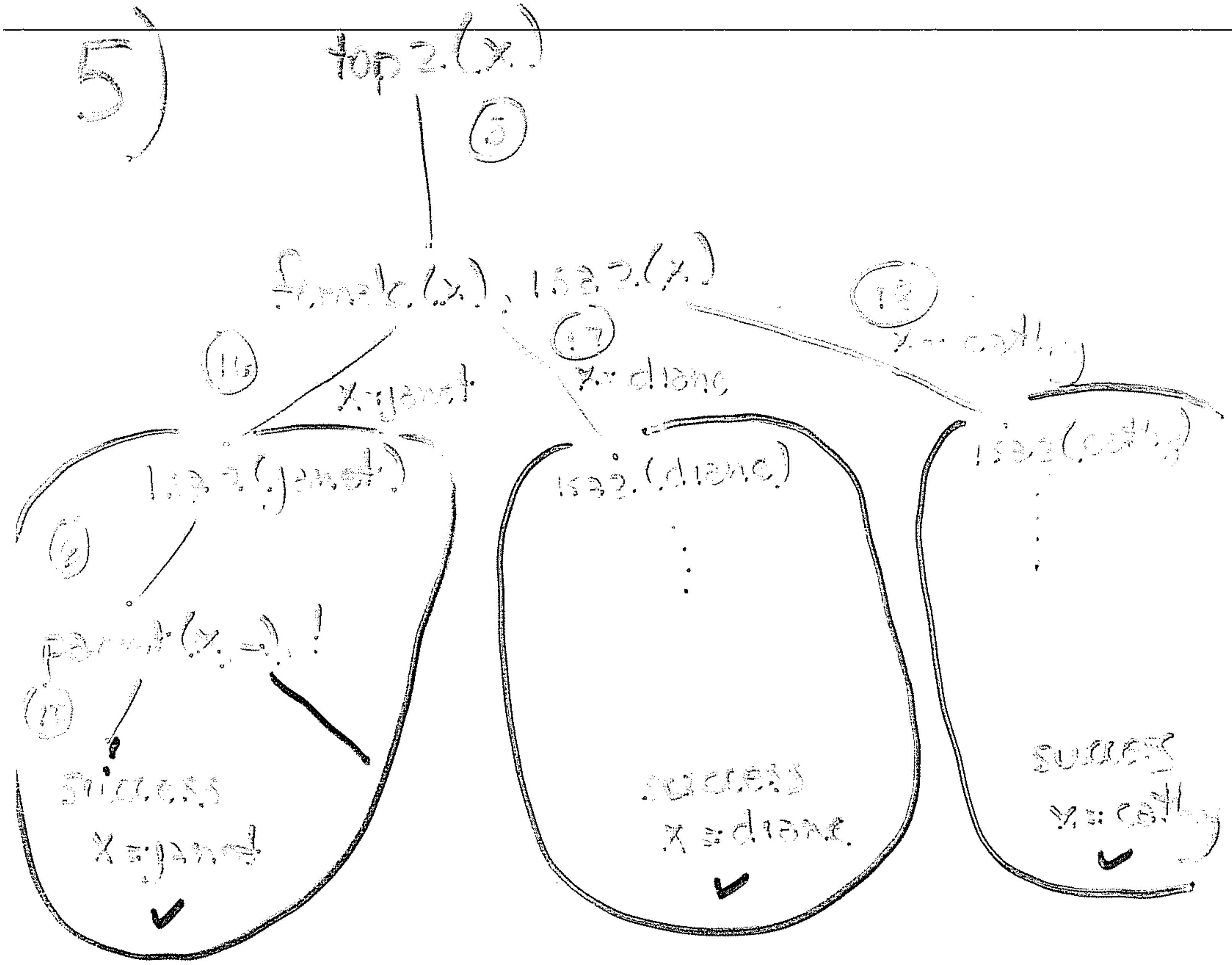
4)

3)



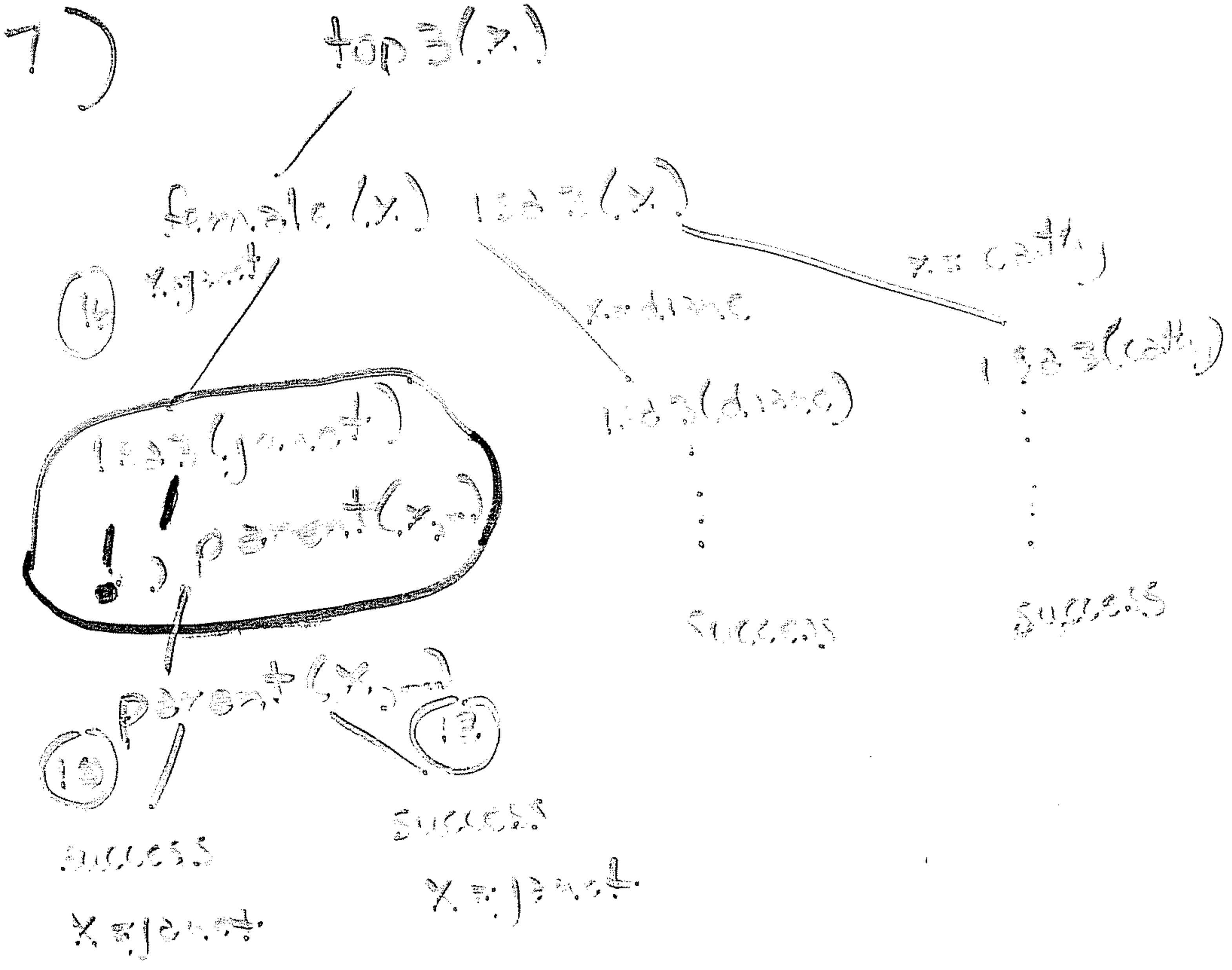
Incorrect

5)



The Correct Version

7)


top3(X)
female(X) !a3(X)
X=janet
X=diane
X=cathy
!a3(janet)
!, parent(X,_)
!a3(diane)
!a3(cathy)
success
success
parent(X,_)
success
success
X=janet
X=janet

Incorrect!