الجامعة التكنولوجية / قسم علوم الحاسوب

فرع الذكاء الاصطناعي – المرحلة الثالثة – الكورس الثاني

مادة الأنظمة الخبيرة – أ.م.د. حسنين سمير

Controlling the Reasoning Strategy (1)

The control strategy is determined as comparing the number of initial state(s) to the number of goal state(s), therefore and according to the fact that say "the search will be from less to more" we can determine the control strategy for any system (if the set of initial state(s) and goal state(s) are clear and complete) easily.

For the classification system

The number of initial state(s) : The number of goal state(s)

Many (properties) 1 (the target class)

The search will be from less to more

Thus the preferred control strategy is "backward" chaining.

Classification Program with Backward Chaining (Bird, Beast, Fish) Version1

database

db_confirm(symbol, symbol)

db_denied(symbol, symbol)

```
clauses
guess_animal :- identify(X), write("Your animal is a(n) ",X),!.

identify(giraffe) :-
                    it_is(ungulate),
                    confirm(has, long_neck),
                    confirm(has, long_legs),
                    confirm(has, dark_spots)
identify(zebra) :-
                    it_is(ungulate),
                    confirm(has, black_strips),!.
identify(cheetah) :-
                    it_is(mammal),
                    it-is(carnivorous),
                    confirm(has, tawny_color),
                    confirm(has, black_spots),!.
identify(tiger) :-
                    it_is(mammal),
                    it-is(carnivorous),
                    confirm(has, tawny_color),
                    confirm(has, black_strips),!.

identify(eagle) :-
                    it_is(bird),
                    confirm(does, fly),
                    it-is(carnivorous),
                    confirm(has, use_as_national_symbol),!.
```

```
identify(ostrich) :-
                    it_is(bird),
                    not(confirm(does, fly)),
                    confirm(has, long_neck),
                    confirm(has, long_legs),!.
identify(penguin) :-
                    it_is(bird),
                    not(confirm(does, fly)),
                    confirm(does, swim),
                    confirm(has, black_and_white_color),!.

identify(blue_whale) :-
                    it_is(mammal),
                    not(it-is(carnivorous)),
                    confirm(does, swim),
                    confirm(has, huge_size),!.
identify(octopus) :-
                    not(it_is(mammal),
                    it_is(carnivorous),
                    confirm(does, swim),
                    confirm(has, tentacles),!.
identify(sardine) :-
                    it_is(fish),
                    confirm(has, small_size),
                    confirm(has, use_in_sandwiches),!.
identify(unknown).          /* Catch-all rule if nothing else works.
*/
```

```
it-is(bird):-
        confirm(has, feathers),
        confirm(does, lay_eggs),!
it-is(fish):-
        confirm(does, swim),
        confirm(has, fins),!.
 it-is(mammal):-
        confirm(has, hair),!.
it-is(mammal):-
        confirm(does, give_milk),!.
it-is(ungulate):-
        it-is(mammal),
        confirm(has, hooves),
        confirm(does, chew_cud),!.
it-is(carnivorous):-
        confirm(has, pointed_teeth),!.
it-is(carnivorous):-
        confirm(does, eat_meat),!.

confirm(X,Y):- db_confirm(X,Y),!.
confirm(X,Y):- not(denied(X,Y)),!, check(X,Y).

denied(X,Y):- db-denied(X,Y),!.

Check(X,Y):- write(X, " it ", Y, \ "n"), readln(Reply), remember(X, Y,
Reply).

remember(X, Y, yes):- asserta(db_confirm(X, Y)).
remember(X, y, no):- assereta(db_denied(X, Y)), fail.
```

Controlling the Reasoning Strategy (2)

According to the same assumptions, we can reach to same facts that say:

For the classification system

The number of initial state(s) : The number of goal state(s)

Many (properties) 1 (the target class)

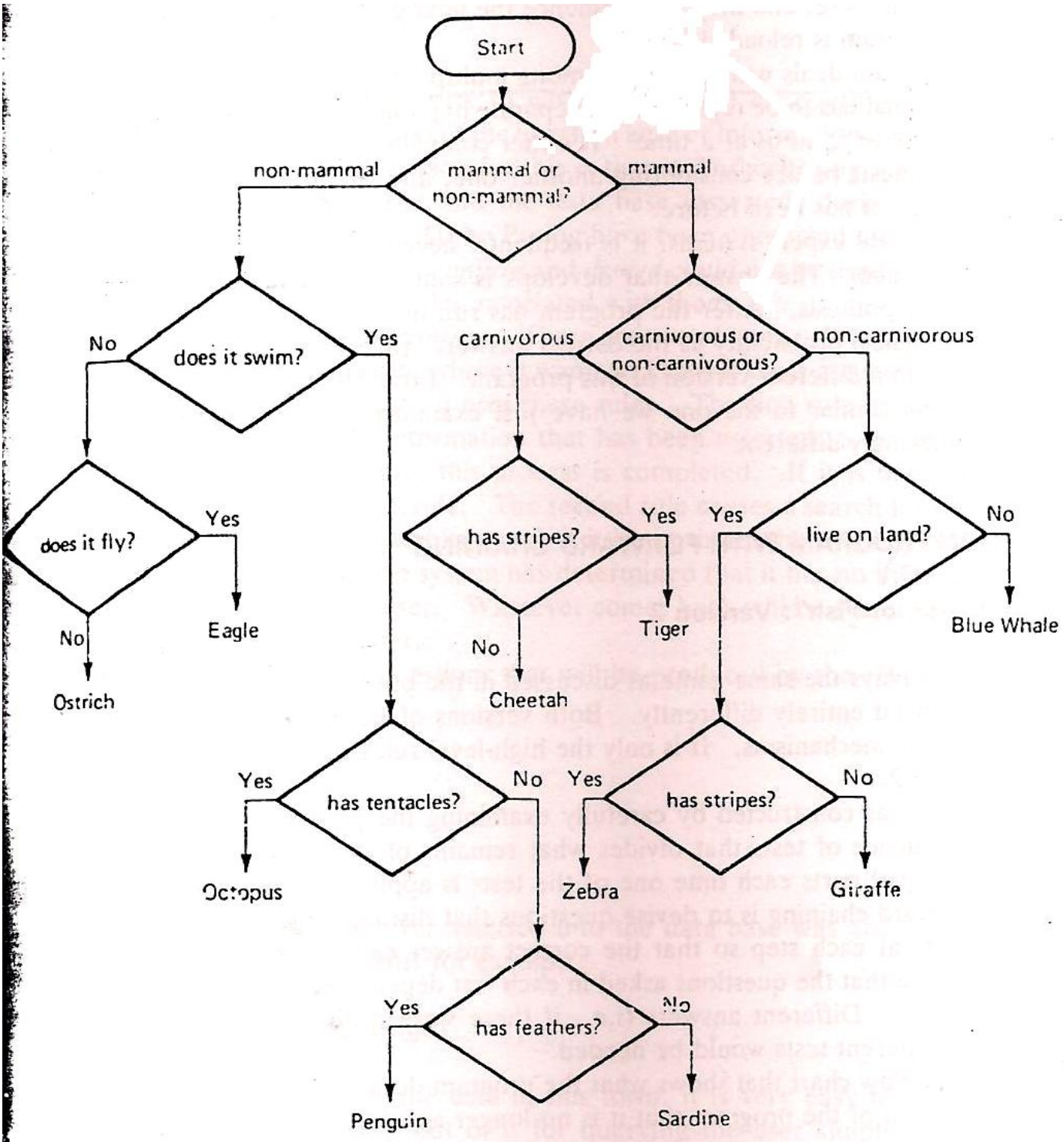
The search will be from less to more

Thus the preferred control strategy is "backward" chaining, but it can be used the "forward" chaining as another strategy to solve the animal classification system but under different conditions as they illustrated in the system requirements such as:

-Decision tree to design the problem.

-The special code for the classification system as a forward chaining.

Classification Program with Forward Chaining (Bird, Beast, Fish) Version2



BBF is a classification program. The forward chaining version makes a series of binary decisions. Each is designed to throw away one half the remaining possibilities (or as close to that as possible until only one is left.

```
database
db_confirm(symbol, symbol)
db_denied(symbol, symbol)
clauses
guess_animal :-
                    find_animal, have_found(X),
                    write("Your animal is a(n) ",X),nl,!.

find_animal:- test1(X), test2(X,Y), test3(X,Y,Z), test4(X,Y,Z,_),!.
Find_animal.

test1(m):- it_is(mammal),!.
test1(n).

test2(m,c):- it_is(carnivorous),!.
test2(m,n).
test2(n,w):- confirm(does, swim),!.
test2(n,n).

test3(m,c,s):- confirm(has, strips),
                asserta(have_found(tiger)),!.
test3(m,c,n):- asserta(have_found(cheetah)),!.
test3(m,n,l):- not(confirm(does, swim)),
                not(confirm(does, fly)),!.
test3(m,n,n):-  asserta(have_found(blue_whale)),!.
test3(n,n,f):- confirm(does, fly),
                asserta(have_found(eagle)),!.
test3(n,n,n):- asserta(have_found(ostrich)),!.
test3(n,w,t):- cofirm(has, tentacles),
```

```
                    asserta(have_found(octopus)),!.
test3(n,w,n).

test4(m,n,l,s):- confirm(has, strips),
                    asserta(have_found(zebra)),!.
test4(m,n,l,n):- asserta(have_found(giraffe)),!.
test4(n,w,n,f):- confirm(has, feathers),
                    asserta(have_found(penguin)),!.
test4(n,w,n,n):- asserta(have_found(sardine)),!.

it-is(bird):- confirm(has, feathers),
                confirm(does, lay_eggs),!.
it-is(fish):- confirm(does, swim),
               confirm(has, fins),!.
 it-is(mammal):- confirm(has, hair),!.
it-is(mammal):- confirm(does, give_milk),!.
it-is(ungulate):- it-is(mammal),
                    confirm(has, hooves),
                    confirm(does, chew_cud),!.
it-is(carnivorous):- confirm(has, pointed_teeth),!.
it-is(carnivorous):- confirm(does, eat_meat),!.

confirm(X,Y):- db_confirm(X,Y),!.
confirm(X,Y):- not(denied(X,Y)),!, check(X,Y).

denied(X,Y):- db-denied(X,Y),!.

Check(X,Y):- write(X, " it ", Y, \ "n"), readln(Reply), remember(X, Y,
Reply).
```

```
remember(X, Y, yes):- asserta(db_confirm(X, Y)).
remember(X, y, no):- assereta(db_denied(X, Y)), fail.
```

Conclusions

1. Code written for backward chaining is clearer. All the rules in version 1 of BBF have a nice declarative reading. They correspond nicely to most people's intuitive idea of how things should be described when they are part of some kind of hierarchy. The description is top down.
2. Code written for backward reasoning is also much easier to modify or expand. It is apparent without much thought what would have to be done to add another animal (class) to the structure: just define it. But it is not always clear where to attach another instance to a forward reasoning rule structure. In fact, if a number of additions have to be made, all the rules may have to be redone to accommodate the additions and at the same time to maintain the same testing efficiency as was there before.
3. Code for the backward reasoning system will be easier to develop in the first place because the built-in inference method in prolog is backward chining.