基于规则的产生式动物识别系统

1 实验目的

1. 通过简单动物识别系统的知识表示和推理程序的编写，加深理解产生式知识表示方法和推理
2. 学会使用产生式知识表示方法和推理解决实际问题

2 实验内容

- 动物识别专家系统是流行的专家系统实验模型，它用产生式规则来表示知识，共 15 条规则，设计一个用于动物识别的产生式系统，该系统通过规则库识别老虎、金钱豹、斑马、长颈鹿、企鹅、信天翁、鸵鸟 7 种动物。这些规则既少又简单，可以改造他们，也可以加进新的规则，还可以用来识别其他东西的新规则来取代这些规则。

3 实验要求

1. 确定推理方法（正向还是反向），并根据问题设计实现一个简单的不通用推理机（匹配、冲突消解）
2. 规则库要求至少包含 15 条规则
3. 初始事实可以任意给定，输入初始事实后能够得到推理结果
4. 设计人机界面，解释模块提供查询规则的功能
5. 可以不考虑知识库管理模块
6. 提交实验报告
7. 报告中要有推理树

4 实验代码

4.1 特征库

```
// ignore `std::`
vector<string> features { "Have hair", "Have milk", "Have feather",
                          "Can lay eggs", "Eat meat", "Have sharp teeth",
                          "Have claws", "Eyes forward", "Have hooves",
                          "Rumination", "Tawny", "Dark spots", "Black bars",
                          "Long neck", "Long legs", "Can not fly",
                          "Can fly", "Black and white", "Can swim",
                          "Mammals", "Carnivores", "Ungulates", "Birds"
                        };
```

4.2 规则库

```cpp
std::vector<std::string> status {
                                   // Mammals
                                   "10000000000000000000000",
                                   "01000000000000000000000",
                                   "00000000000000000001000",
                                   "10000000000000000001000",
                                   "01000000000000000001000",
                                   // Birds
                                   "00100000000000000000000",
                                   "00010000000000001000000",
                                   "00000000000000000000001",
                                   "00100000000000000000001",
                                   "00010000000000001000001",
                                   // Carnivores
                                   "00001000000000000000000",
                                   "00000111000000000000000",
                                   "00000000000000000000100",
                                   // Ungulates
                                   "00000000100000000001000",
                                   "10000000100000000000000",
                                   "01000000100000000000000",
                                   "00000000010000000001000",
                                   "10000000010000000000000",
                                   "01000000010000000000000",
                                   "00000000000000000000010",
                                   "10000000010000000001000",
                                   "01000000010000000001000",
                                   // R9
                                   "10001000001100000000000",
                                   "10000111001100000000000",
                                   "10000000001100000000100",
                                   "01001000001100000000000",
                                   "01000111001100000000000",
                                   "01000000001100000000100",
                                   "00001000001100000001000",
                                   "00000111001100000001000",
                                   "00000000001100000001100",
                                   "10001000001100000001000",
                                   "01001000001100000001000",
                                   "10000111001100000001000",
                                   "01000111001100000001000",
                                   "10000000001100000001100",
                                "010000000011000000001100",
                                // R10
                                "100010000010100000000000",
                                "100001110010100000000000",
                                "100000000010100000000100",
                                "010010000010100000000000",
                                "010001110010100000000000",
                                "010000000010100000000100",
                                "000010000010100000001000",
                                "000001110010100000001000",
                                "000000000010100000001100",
                                "100010000010100000001000",
                                "010010000010100000001000",
                                "100001110010100000001000",
                                "010001110010100000001000",
                                "100000000010100000001100",
                                "010000000010100000001100",
                                // R11
                                "000000001001011000001000",
                                "100000001001011000000000",
                                "010000001001011000000000",
                                "000000000101011000001000",
                                "100000000101011000000000",
                                "010000000101011000000000",
                                "000000000001011000000010",
                                // R12
                                "000000001000100000001000",
                                "100000001000100000000000",
                                "010000001000100000000000",
                                "000000000100100000001000",
                                "100000000100100000000000",
                                "010000000100100000000000",
                                "000000000000100000000010",
                                // R13
                                "001000000000011101000000",
                                "000000000000011101000001",
                                "001000000000011101000001",
                                // R14
                                "001000000000000101100000",
                                "000000000000000101100001",
                                "001000000000000101100001",
                                // R15
                                "001000000000000010000000",
                                "000100000000000010000000",
                                  "00000000000000001000001",
                                  "00100000000000001000001",
                                  "00010000000000001000001",
                                };
```

4.3 综合数据库

```
std::vector<std::string> answer { "Sorry, I don't know...",
                                  "It is Leopard.",
                                  "It is Tiger.",
                                  "It is Giraffe.",
                                  "It is Zebra.",
                                  "It is Ostrich.",
                                  "It is Penguin.",
                                  "It is Albatross.",
                                  "It is Mammals.",
                                  "It is Carnivores.",
                                  "It is Ungulates.",
                                  "It is Birds."
                                };
```

4.4 推理机

```
int solve() {
  std::string in;
  std::string mask("00000000000000000000000");
  while (std::cin >> in, in != "END") {
    auto it = in.begin();
    bool ok = true;
    while (it != in.end()) {
      ++it;
      if (std::isalpha(*it)) {
        ok = false;
        break;
      }
    }
    if (!ok) {
      std::cout << "Invalid enter, please check the input!" << std::endl;
      continue;
    }
    int idx = std::stoi(in);
    --idx;
    if (0 <= idx && idx < _SIZE_) {
      mask[idx] = '1';
```

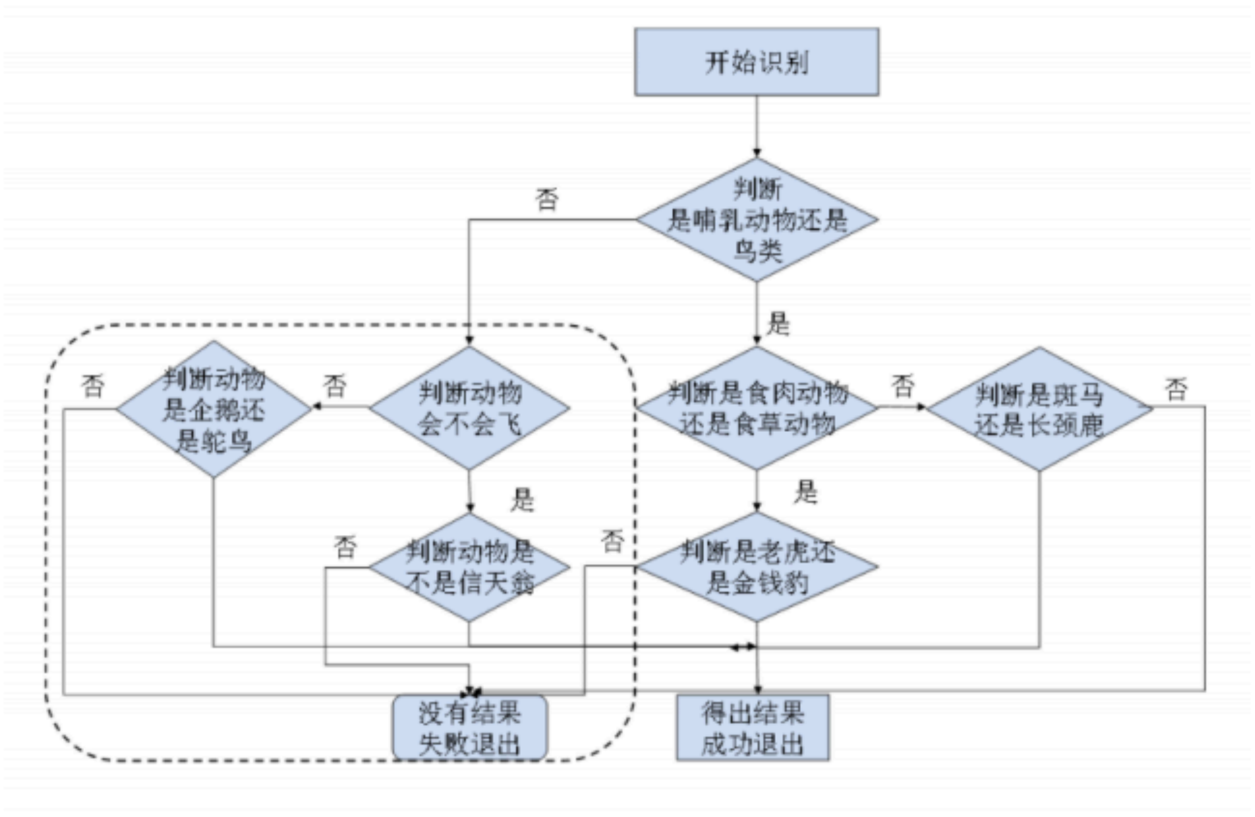
```cpp
    } else {
      std::cout << "Invalid enter, please check the input!" << std::endl;
      continue;
    }
  }
  int pos = -1;
  for (int x = 0; x < _COUNT_; x++) {
    if (mask == status[x]) {
      pos = x;
      break;
    }
  }
  if (0 <= pos && pos < 5) return 8;
  if (5 <= pos && pos < 10) return 11;
  if (10 <= pos && pos < 13) return 9;
  if (13 <= pos && pos < 22) return 10;
  if (22 <= pos && pos < 37) return 1;
  if (37 <= pos && pos < 52) return 2;
  if (52 <= pos && pos < 59) return 3;
  if (59 <= pos && pos < 66) return 4;
  if (66 <= pos && pos < 69) return 5;
  if (69 <= pos && pos < 72) return 6;
  if (72 <= pos && pos < 77) return 7;
  return 0;
}
```

4.5 程序入口

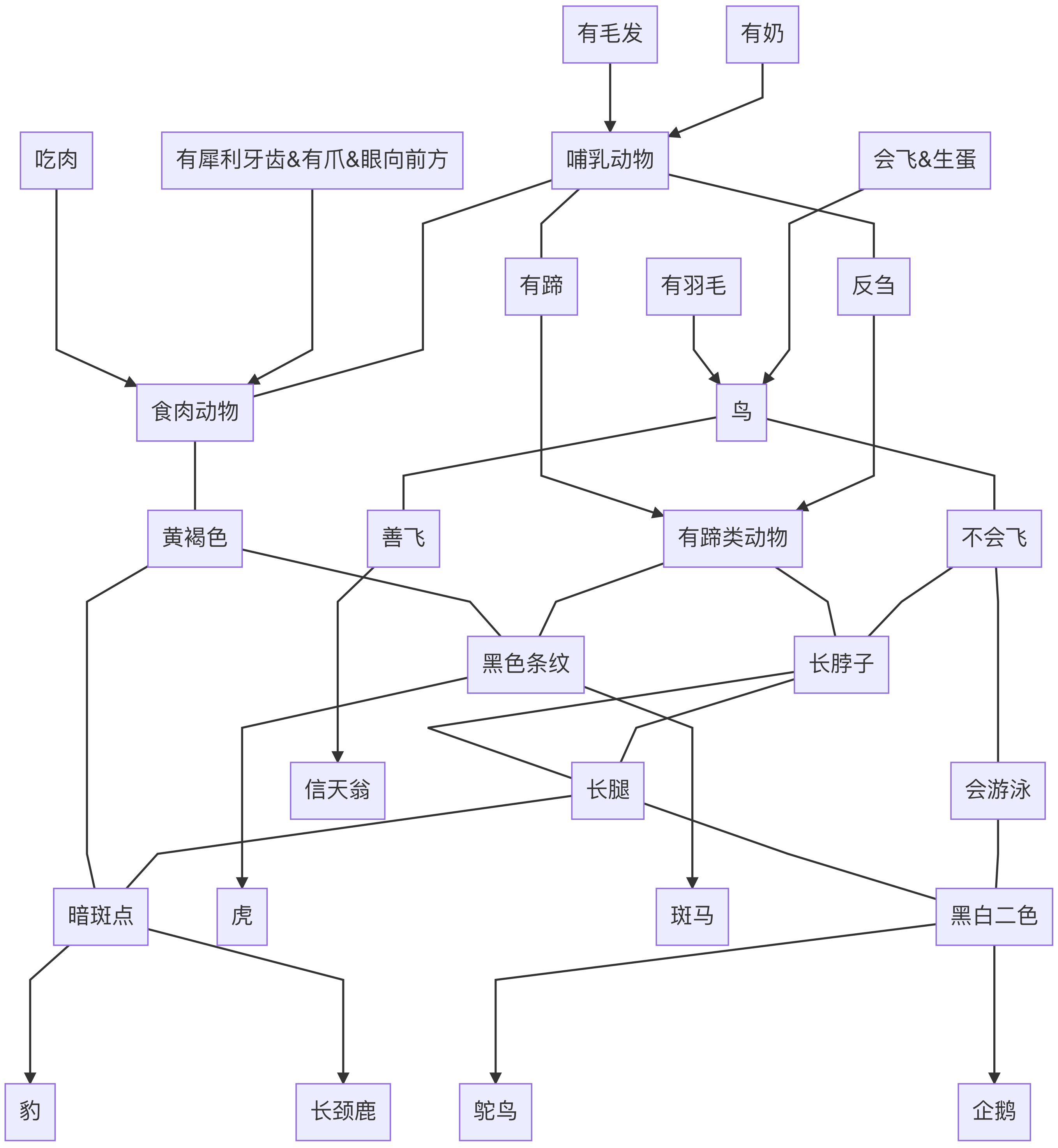
```cpp
int main() {
  std::cout << "Choose feature(s) of the animal you want to identify:" << std::endl;
  std::cout << "And type `END` to stop entering." << std::endl << std::endl;
  for (int id = 0; id < _SIZE_; id++) {
    int out = id + 1;
    if (out >= 10) {
      std::cout << out << ". ";
    } else {
      std::cout << "0" + std::to_string(out) << ". ";
    }
    std::cout << std::left << std::setw(20) << rules[id] << " \n"[out % 3 == 0];
  }
  std::cout << std::endl;
  int idx = solve();
```

```cpp
    std::cout << std::endl;
    std::cout << "My answer is:" << std::endl;
    std::cout << answer[idx] << std::endl;
    return 0;
}
```

5 流程图



6 推理树



有毛发
有奶
吃肉
有犀利牙齿&有爪&眼向前方
哺乳动物
会飞&生蛋
有蹄
有羽毛
反刍
食肉动物
鸟
黄褐色
善飞
有蹄类动物
不会飞
黑色条纹
长脖子
信天翁
长腿
会游泳
暗斑点
虎
斑马
黑白二色
豹
长颈鹿
鸵鸟
企鹅