基于web的动物知识专家系统研究与实现

莫宇诚 11331247
软件学院
软件工程（计算机应用软件）
2015.5.14 中山大学东校区

大纲

1. 系统开发背景
2. 系统设计
3. 通用接口的研究
4. 系统实现
5. 总结与展望

一、系统开发背景

- 专家系统技术取得广泛应用
- web技术日新月异
- 专家系统逐步走向多技术、多方法的综合集成与多学科、多领域的综合应用型发展

- 以一个典型动物知识专家系统为原型加以简单修改
- 研究实现了PROLOG通用接口协议解决PROLOG专家系统后端到web服务器的通信
- 设计实现了web后台和前端

- 完成一个完整的小型的动物知识web专家系统 -- Lucy

二、系统设计

1. 重视方法的通用性和可适用性。
   - 努力使得 Lucy 的完成能够带来更多可以被借鉴、参考或者直接使用的结果
     - eg1：PROLOG通用接口协议的PROLOG端实现和python客户端实现
     - eg2：专家系统前端编写的方法

2. 保持简单
   - 讨论尽量丰富、完整，但方法、编码、概念尽量简单易懂

3. 分而治之
   – 子系统模块化
   – 功能模块化

系统架构

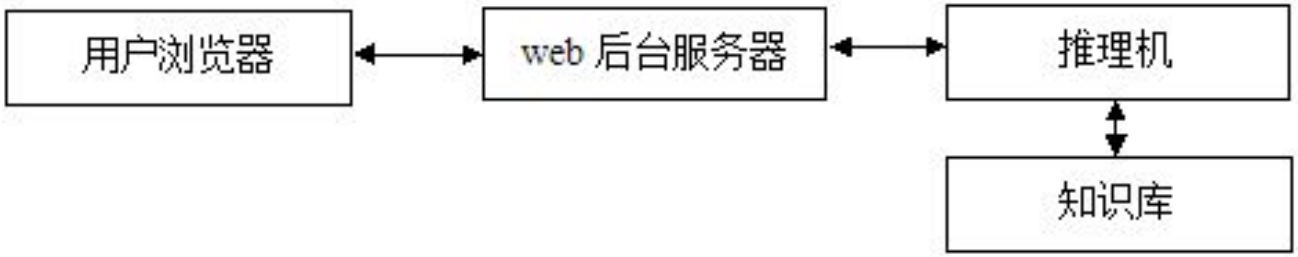


图 3-1 系统架构图

三、通用接口协议的研究

通用接口

- 专家系统广泛地使用PROLOG语言实现
- web开发技术日益丰富多样

- 能否设计一种简单、有效、通用的PROLOG语言到其他语言通信的方法？

- PROLOG通信接口协议

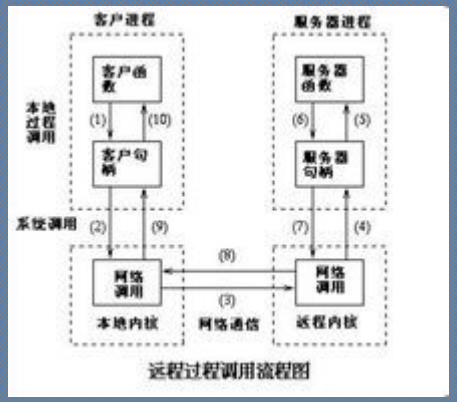
PROLOG通信接口协议

1. 使用socket为不同的用户程序和PROLOG程序建立连接。
2. 连接建立后，用户程序向socket中发送START指令。
3. PROLOG程序收到第一个START后，
4. PROLOG程序发送ASK指令。
5. 用户程序根据收到的ASK指令收集用户反馈reply，发送给PROLOG程序。
6. 如果PROLOG程序已经求出解，转7。否则转4。
7. PROLOG程序发送FINISH指令，双方切断连接。

接口实现基础

1. 跨编程范式带来的复杂性
   - 跨越逻辑编程范式与常见编程范式
2. 专家系统使用场景的复杂性
   - 能够应对不同的业务场合
3. 专家系统支持功能的复杂性
   - 能够支持多种功能
4. 复杂部署环境的复杂性
   - 不同的智能代理上、不同的程序、环境能够良好地进行通信
5. 复杂编程语言实例复杂性
   - 不同的编程语言实体

1.使用socket，通过TCP连接来进行不同编程语言的通信（类似于远程过程调用）



2.建模

一次专家系统的完整使用上，往往是多次与用户交互：

1.用户求解：请告诉我这是什么动物？
2.专家系统分析：需要知道这个动物是否有羽毛
3.用户回答：这个动物有羽毛
4.专家系统分析：需要知道这个动物是否会飞
5.用户回答：这个动物不会飞
....
N. 专家系统回答：这个动物是企鹅，（出示图片），对吗？
N+1. 用户回答：是的，原来这个东西叫企鹅。

查询：

一次用户和专家系统相互交流，最后得到一个结果的过程

反馈：

用户和专家系统的一步对话

提示：

专家系统为了产生解而对用户进行的一步对话

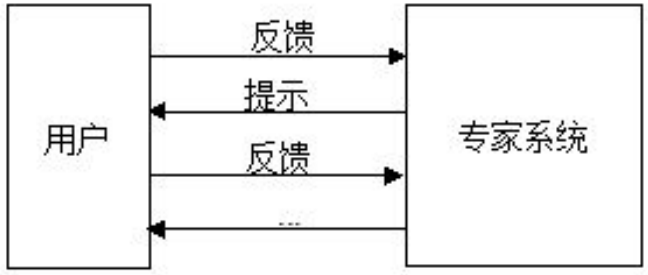


图 4-1 查询的一般过程

专家系统即是这样一个“处理查询”来解决问题的系统。

3.指令

1. START : 表示一次查询开始。
2. ASK(hint) : PROLOG对用户输出提示hint。
3. REPLY(reply) : 用户对PROLOG输入反馈 reply。
4. FINISH(answer) : PROLOG对用户输出查询结果answer。

4.编程实现

- server(Port, Goal)
  - 监听Port端口，每个客户端使用Goal谓词来启动查询
- session_start
  - 当前查询开始时要被调用，用来检查和初始化
- session_ask(Q, R)
  - 在当前查询中，输出提示Q，将R限定为用户反馈
- session_reply(A)
  - 当前查询返回A为最后结果

4个函数

- 客户端

图 5-1 通用 PROLOG 接口类图

- work well

程序见附录

四、系统实现

1.专家系统后端

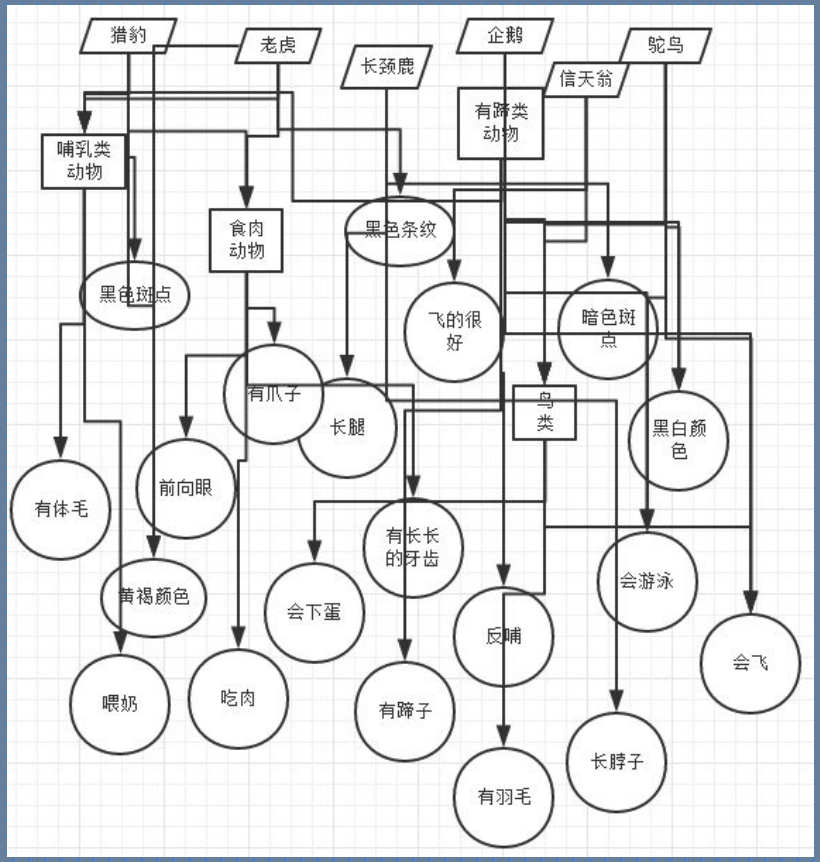


图6-1 Lucy包括的动物知识

```
animal_is(cheetah) :-
    it_is(mammal),
    it_is(carnivore),
    positive(has, tawny_color),
    positive(has, black_spots).
```

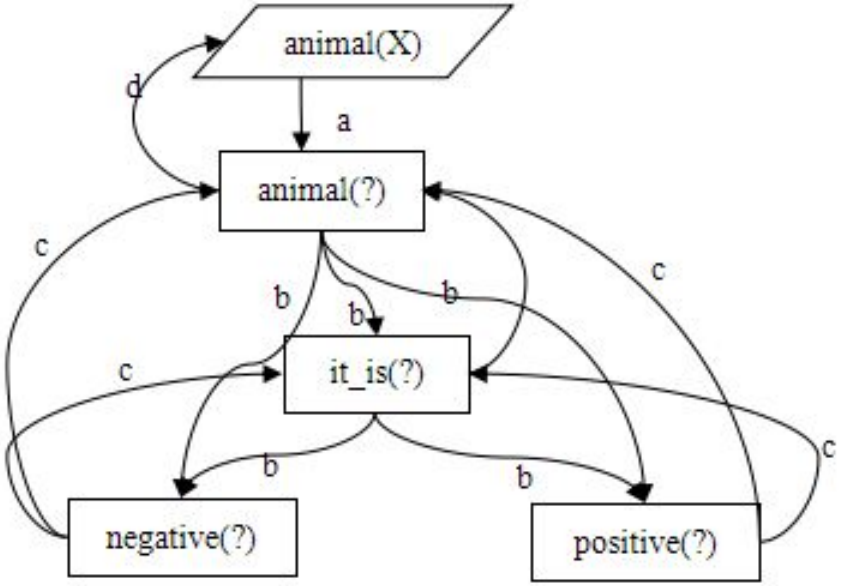


图 6-2 推理机查询过程

2. WEB后台

简单地把用户反馈传输给PROLOG后端

- @app.route("/start")
- @app.route("/restart")
- @app.route("/send")

后台维持和PROLOG的通信：

- 在服务器后台为每个用户的查询维持一个 GPIClient 实例
- 每个连接根据cookie查到对应的实例
- 过一段时间都没有活动的实例程序去清除掉

3.web前端

- 策略：不包装Term类，直接解析字符串
- 解析PROLOG产生的term字符串，实际上在服务器前台完成。为此，实现了三个重要的函数：
- has_word(word) : 该term字符串是否包括单词 word
- select(word_list) : 该term字符串如果包含word_list 中的某一项，返回之。否则返回null
- handler(term_string) : 根据收到的 term_string 字符串，调整web界面收集用户反馈。这里利用了 term_string 与用户界面是一一对应的。
- 通过ajax，浏览器前端抓取web后台传过来的数据 term_string ，并通过 has_word 和 select 将其“翻译”为一个友好的界面。在用户在界面上做出反馈后，通过ajax提交用户的反馈，并再次抓取新的 term_string 。

4.模拟分布式

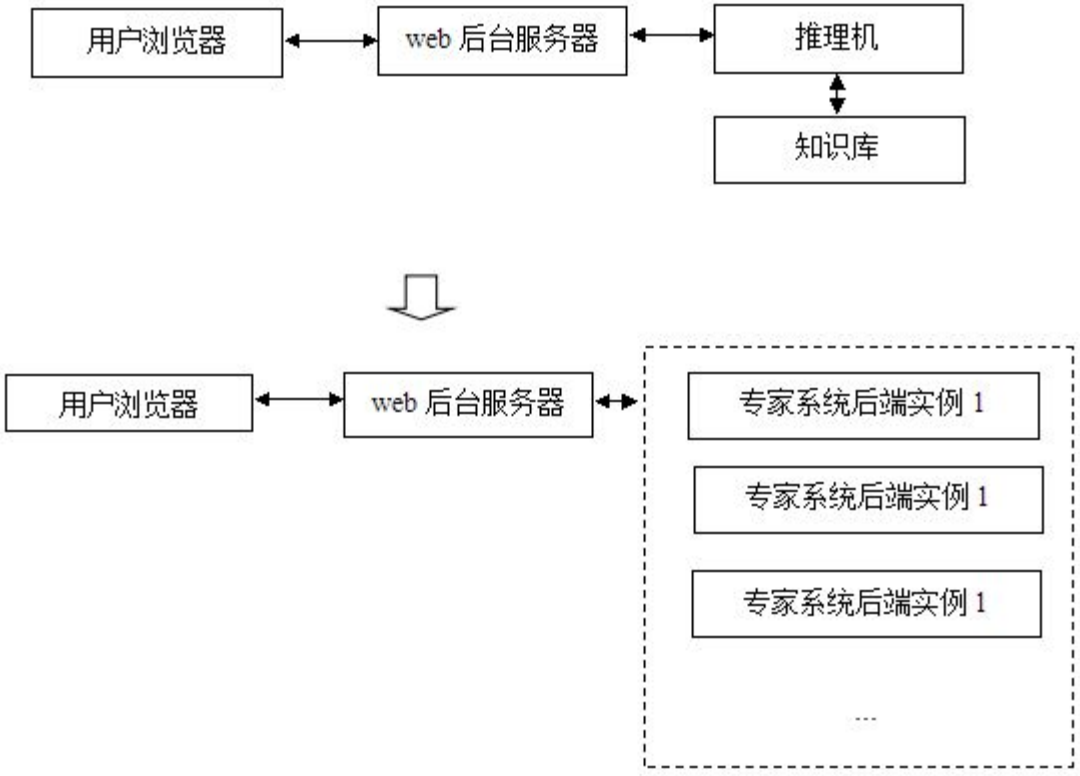


图 8-1 分布式专家系统架构

```
def choice_client(iplist) :
    host, port = random.choice(iplist).split(':')
    print 'SELECT>> %s:%s' % (host, port)
    return GPIClient(host, int(port))
```

图 8-4 随机分派策略的核心代码（调试用）

5. web前端效果截图

图3-2 Lucy界面
（回答问题）

图3-3 Lucy推测出动物名称

五、总结与展望

系统情况

- 工作良好
- 模拟分布式专家系统工作良好

不足

- 完成更加功能丰富、复杂的专家系统
- 参考更多的专家系统实现、了解更多的实际问题

展望

- 提出了一种模块化、完整的专家系统示例
- 从通信接口着手，为专家系统的整合、重用提供思路

感谢各位答辩委员！

感谢我的导师！

感谢所有帮助过我的人！

莫宇诚 11331247
软件学院
软件工程（计算机应用软件）
2015.5.14 中山大学东校区