Севастопольский государственный университет
Институт информационных технологий

Методы и системы искусственного интеллекта

Бондарев Владимир Николаевич

Лекция 18

Продукционные ЭС

В.Бондарев

Понятие ЭС и типовые задачи ЭС

Под ***экспертной системой*** (ЭС) понимают программную систему, аккумулирующую знания эксперта в определенной области и вырабатывающую решения и рекомендации на уровне эксперта. Перечень типовых задач, решаемых ЭС, включает:

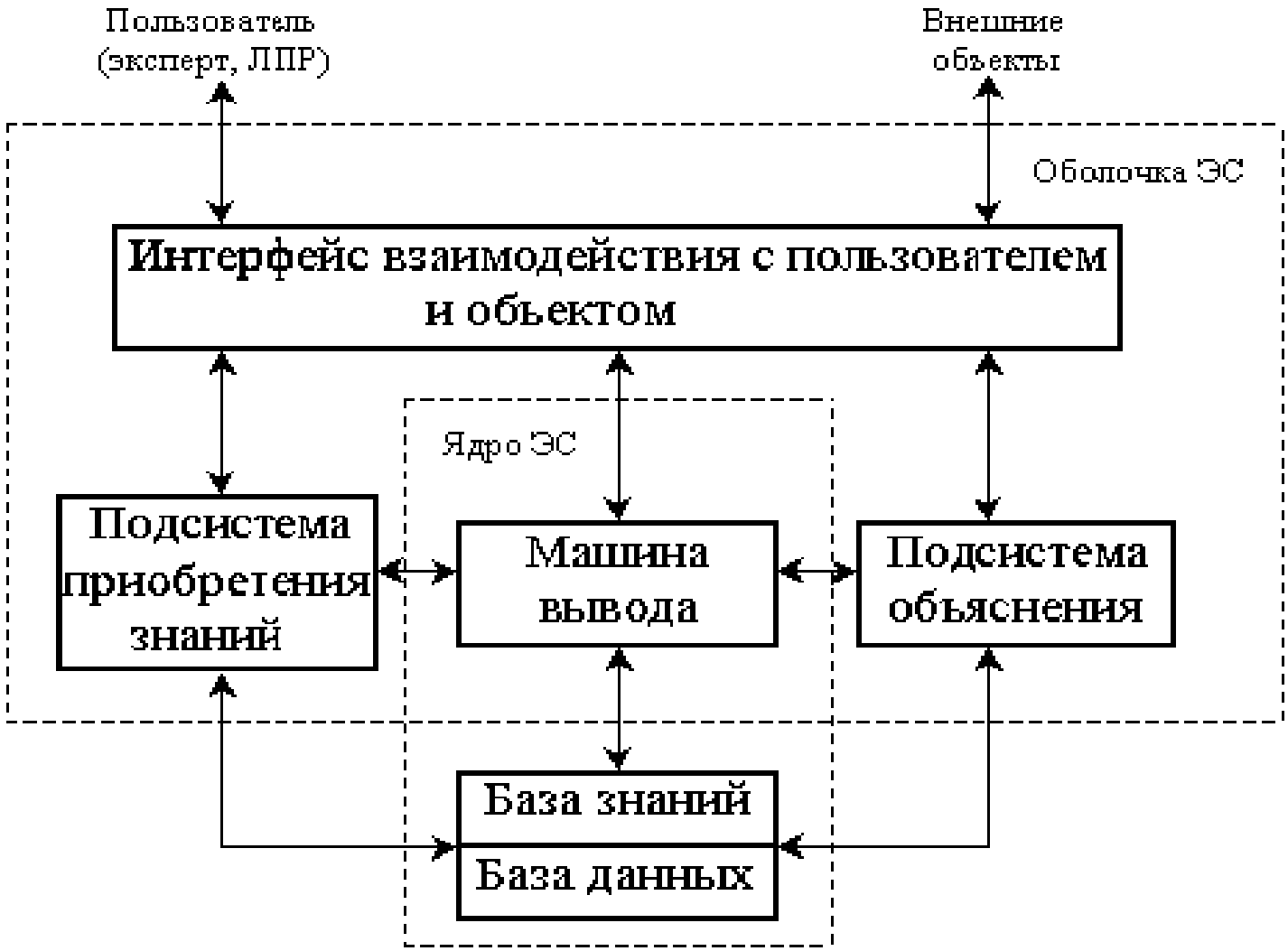
- **интерпретацию** — извлечение информации из первичных данных (распознавание образов) ;
- **диагностику** — обнаружение неисправностей ;
- **мониторинг** — непрерывная интерпретация данных в реальном времени с сигнализацией о выходе тех или иных параметров за допустимые пределы ;
- **прогноз** — предсказание вероятных последствий на основе прошедших и настоящих событий ;
- **планирование** — определение последовательности действий, направленных на достижение заранее поставленных целей ;

Задачи ЭС

- **проектирование** — определение конфигурации системы при заданных ограничениях ;
- **отладку и ремонт** — выполнение последовательности действий по приведению той или иной системы к требуемым режимам функционирования ;
- **обучение** — интерпретация, диагностика и коррекция знаний и умений обучаемого ;
- **управление** — формирование управляющих воздействий, определяющих поведение сложных систем .

.

Типовая архитектура ЭС



Формирование объяснений

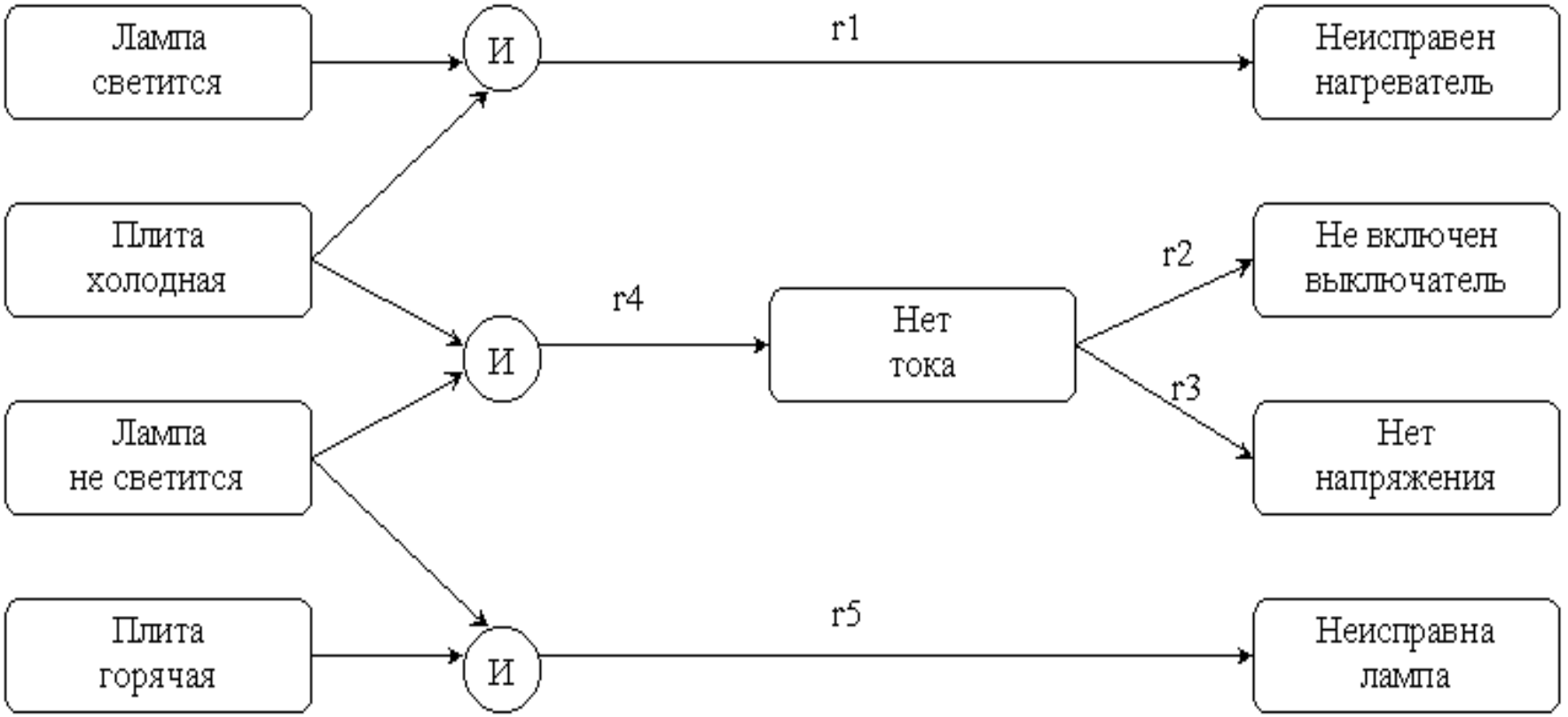
Для получения объяснений выводов пользователи ЭС могут задавать вопросы двух типов:

- *почему* система сочла необходимым задать пользователю определенный вопрос;
- *как* система пришла к соответствующему заключению.

Выясним механизм формирования ответов на эти вопросы на примере *обратного вывода*.

В качестве примера выберем задачу диагностики. Пусть требуется установить причину неисправности электрической плиты, оборудованной нагревательным элементом, контрольной лампочкой и выключателем. Плита подключена к источнику напряжения с помощью электрического кабеля, который обеспечивает передачу тока. Набор правил “неисправность-причина” представим в виде графа

Формирование объяснений



Правила-продукции диагностической ЭС

1) если лампа(светится) и плита(холодная)
то нагреватель(неисправен).
2) если тока(нет) то выключатель(не_включен).
3) если тока(нет) то напряжения(нет).
4) если плита(холодная) и лампа(не_светится)
то тока(нет).
5) если лампа(не_светится) и плита(горячая)
то лампа(неисправна).

Процесс вывода начинается с ввода в рабочую память возможной гипотезы (причины) неисправности, например, **гипотеза(X)**, где **X** – переменная, которая может быть сопоставлена с любой из возможных причин неисправностей, известных системе: **выключатель(не_включен), нагреватель(неисправен), напряжения(нет), лампа(неисправна).**

Ответ на вопрос «почему?»

В соответствии с правилами в ходе вывода пользователю будут заданы вопросы:

плита(холодная)?

:-да.

лампа(не_светится)?

:-да.

Если пользователь желает выяснить, почему система задает тот или иной вопрос, то он вводит слово “*почему*”. Например:

лампа(не_светится)?

:-почему.

пытаюсь доказать лампа(не_светится)

с помощью правила: правило4

лампа(не_светится)?

В этом случае система демонстрирует пользователю номер текущего правила, в соответствии с которым был связан заданный вопрос.

Ответ на вопрос «как»

Чтобы ответить на вопрос “*как*” в системе сохраняется *дерево решений*, принятых в течение сеанса работы. Просматривая дерево, можно выяснить, достижение каких подцелей привело к данному решению. Например, возможен следующий сценарий:

```
решение: выключатель(не_включен)
объяснить ? [цель/нет]:
:-выключатель(не_включен).
выключатель(не_включен) было доказано с помощью
правило2 :: если тока(нет) то выключатель(не_включен)
объяснить ? [цель/нет]:
:-тока(нет).
тока(нет) было доказано с помощью
правило4 :: если плита(холодная) и лампа(не_светится) то
тока(нет)
объяснить ? [цель/нет]:
:-плита(холодная).
плита(холодная) было введено пользователем
объяснить ? [цель/нет]:
:-нет.
```

Простейшая ЭС

Простейший способ построения ЭС на Прологе — использование в качестве интерпретатора механизма поиска решений, заложенного в Пролог-системе .

```
/* структура правила:  причина:– неисправность */
%правило 1
нагреватель(неисправен):– лампа(светится),  плита(холодная).
выключатель(не_включен):– тока(нет). %правило 2
напряжения(нет):– тока(нет).                %правило 3
%правило 4
тока(нет):– плита(холодная),  лампа(не_светится).
%правило 5
лампа(неисправна):– лампа(не_светится), плита(горячая).
```

Простейшая ЭС

```
лампа(не_светится).                    % признаки неисправности
плита(холодная).
гипотеза (нагреватель(неисправен)).    % гипотезы
гипотеза (выключатель(не_включен)).
гипотеза (напряжения(нет)).
гипотеза (лампа(неисправна)).
найти(X): – гипотеза (X), X.           % проверка гипотез
```

```
? – найти(X).
        X = выключатель(не_включен);
        X = напряжения(нет).
```

Недостатком рассмотренной программы является необходимость внесения наблюдаемых признаков неисправностей непосредственно в текст программы в виде фактов.

Простейшая ЭС

Устранить указанный недостаток можно, запросив значения соответствующих неисправностей у пользователя в процессе выполнения программы:

```
/* вариант 2 */
лампа(не_светится): – спроси('Лампа не светится ?').
плита(холодная): – спроси('Плита холодная ?').
лампа(светится): – спроси('Лампа светится ?').
плита(горячая): – спроси('Плита горячая ?').
спроси(Вопрос): – write(Вопрос), read('да').
```

В этом случае система будет задавать вопросы пользователю относительно неизвестных фактов. Так как ответы пользователя не запоминаются, то возможно повторение вопросов. Для запоминания ответов можно воспользоваться встроенным предикатом **assert**.

Усовершенствованная ЭС

Усовершенствованный вариант реализации ЭС предполагает запись правил более естественной форме:

правило1 :: если лампа(светится) и плита(холодная)
то нагреватель(неисправен).
правило2 :: если тока(нет)
то выключатель(не_включен).
правило3 :: если тока(нет)
то напряжения(нет).
правило4 :: если плита(холодная) и лампа(не_светится)
то тока(нет).
правило5 :: если лампа(не_светится) и плита(горячая)
то лампа(неисправна).

Усовершенствованная ЭС

Чтобы программа могла интерпретировать правила в такой форме, необходимо определить операторы:

определить_операторы: – op(920, xfy, и), op(950, xfx, то),
op(960, fx, если), op(970, xfx, '::').
: – определить_операторы.

Тогда для указанных правил перечень возможных гипотез неисправностей запишется в виде:

% гипотезы представляются структурой – Имя_факта :: Факт
h1 :: гипотеза(нагреватель(неисправен)).
h2 :: гипотеза(выключатель(не_включен)).
h3 :: гипотеза(напряжения(нет)).
h4 :: гипотеза(лампа(неисправна)).

Усовершенствованная ЭС

Перечислим также признаки неисправностей, значения которых можно запрашивать у пользователя:

```
% признаки, истинность которых запрашивается
q1 :: признак(лампа(светится)).
q2 :: признак(плита(холодная)).
q3 :: признак(лампа(не_светится)).
q4 :: признак(плита(горячая)).
```

Реализуем интерпретатор ЭС в виде предиката **найти(H),** где **H** — возможная гипотеза, которую требуется подтвердить или опровергнуть.

Усовершенствованная ЭС

При этом возможны четыре случая:
1) гипотеза **H** подтверждается фактом, уже известным системе;
2) гипотеза **H** соответствует следствию одного из правил, тогда для подтверждения гипотезы необходимо доказать справедливость предпосылок правила;
3) гипотеза **H**, доказываемая на некотором шаге вывода, представляет собой конъюнкцию условий правила, т.е. **H1** и **H2**, тогда необходимо доказать достижимость конъюнкции подцелей: **найти(H1)** и **найти(H2)**;
4) гипотеза **H** сопоставима с одним из признаков, на основе которых устанавливаются причины неисправности, тогда необходимо задать соответствующий вопрос пользователю.

Для того чтобы исключить повтор вопросов, ответы пользователя будем запоминать в базе данных в виде фактов :

сообщено(Факт, ‘да|нет’).

Усовершенствованная ЭС

Гипотезы, относительно которых можно задавать вопросы, определяются с помощью предиката **запрашиваемая(Н)**. Такие гипотезы сопоставимы с признаками.

С учетом сказанного, интерпретатор опишется следующей совокупностью правил языка Пролог:

```
найти(Н):- Факт :: Н.                                  % случай 1
найти(Н):- Правило :: если Н1 то Н, найти(Н1).% случай2
найти(Н1 и Н2):-найти(Н1), найти(Н2).          % случай 3
найти(Н):- запрашиваемая(Н),                   % случай 4
           сообщено(Н, да).                    % вопрос уже был
найти(Н):- запрашиваемая(Н),                   % случай 4
           not(сообщено(Н,_)),                 % вопроса  не было
           спроси(Н).
```

Усовершенствованная ЭС

```
запрашиваемая(H):- Факт :: признак(H).

спроси(H):- nl, write(H), write('?'), nl,                % вывод вопроса
            read(O), ответ(H,O).                         % ввод ответа
%обработка ответов
ответ(H, да):-   assert(сообщено(H, да)), !.  % добавить ответ в БД
ответ(H, нет):- assert(сообщено(H, нет)), !, fail.
ответ(H,_):- write('правильный ответ: да, нет'), nl,
             спроси(H).                          % повторить вопрос
```

Вызов интерпретатора с целью проверки всех гипотез выполняется следующим образом:

```
 инициализация:-retractall(сообщено(_,_)).
диагностика(H):-    инициализация, Факт :: гипотеза(H),
                    найти(H).
```

Усовершенствованная ЭС

Улучшение разработанной ЭС связано с включением возможности ответов на вопросы типа “почему?” и “как?”.