

# Принятие решений в условиях неопределенности

Деревья решений

Диаграммы влияния

Динамическое программирование в задачах с неопределенностью

# Теория полезности (Utility theory)

Лотерея – набор исходов и их вероятностей:  $[S_1:p_1, \dots, S_n:p_n]$ .

**Вопрос:** как сравнивать разные лотереи?

Один из подходов (теория полезности фон Неймана – Morgenштерна):

## **Аксиомы фон Неймана - Morgenштерна:**

- *Полнота.* Между любой парой альтернатив выполняется либо  $A \succ B$ ,  $B \succ A$ , либо  $A \sim B$ .
- *Транзитивность предпочтений.*
- *Непрерывность.* Если  $A \succ C \succ B$ , то существует вероятность  $p$ , такая что  $[A:p, B:1-p] \sim C$ .
- *Независимость.* Если  $A \succ B$ , то для любой  $C$  и вероятности  $p$ :  
$$[A:p, C:1-p] \succ [B:p, C:1-p].$$

Оказывается, что аксиомы порождают вещественную меру полезности  $U$ , а полезность лотереи равна *математическому ожиданию полезности ее исходов*.

Теория полезности (Utility theory). Принцип максимизации ожидаемой полезности

*Ожидаемая полезность* от действия  $a$  при условии наблюдения  $o$  задается так:

$$EU(a|o) = \sum_{s'} P(s'|a, o)U(s')$$

Принцип *максимизации ожидаемой полезности* сводится к тому, что мы должны выбирать действие таким образом, чтобы максимизировать ожидаемую полезность:

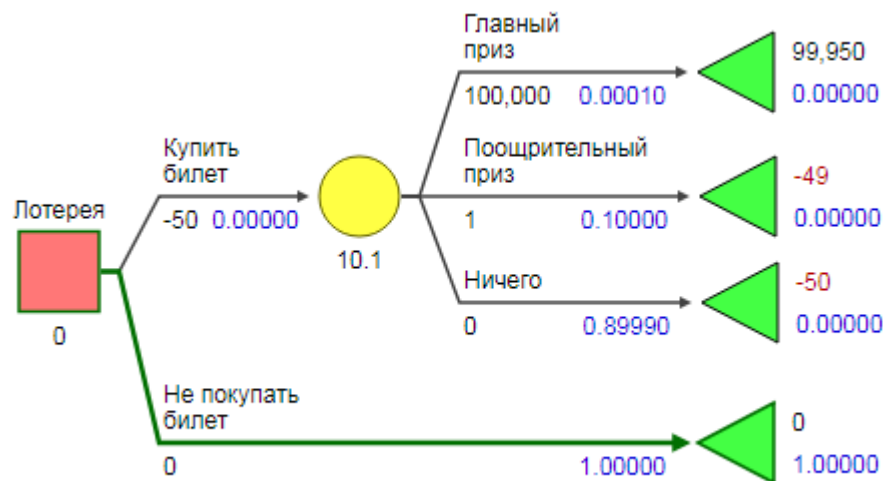
$$a^* = \arg \max_a EU(a|o)$$

# Деревья решений. Структура

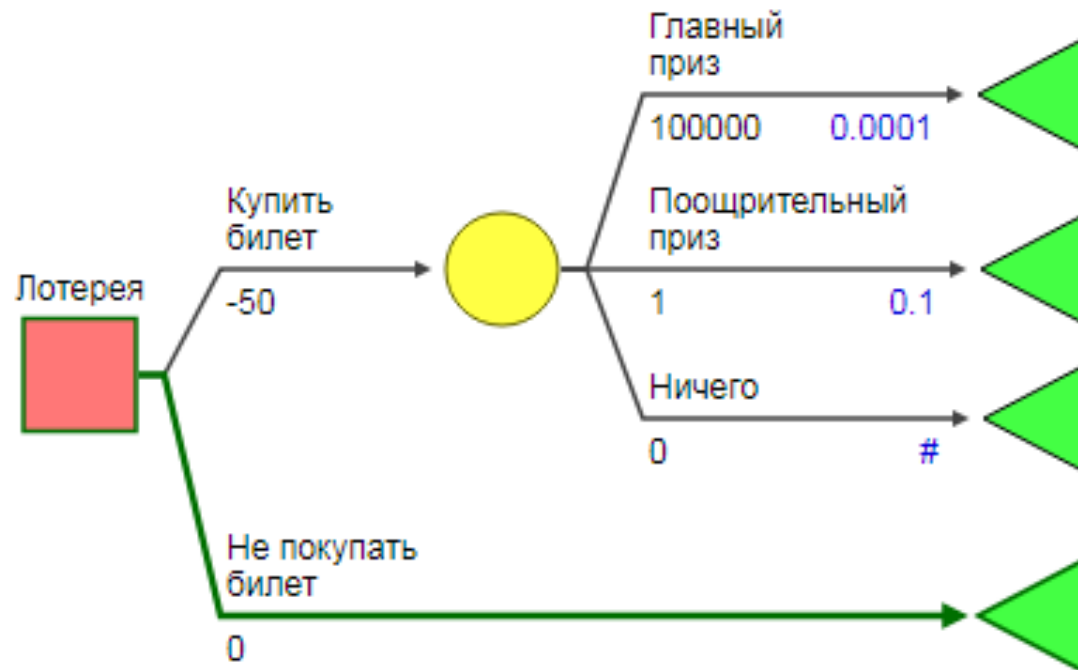
Дерево, содержащее вершины трех типов:

- 1) Решение (управление) – decision node. Варианты решения представлены взаимоисключающими ветвями, выходящими из такой вершины.
- 2) Вероятностная вершина – chance node. Ветви – *полное* множество *взаимоисключающих* исходов СВ.
- 3) Терминальные вершины.

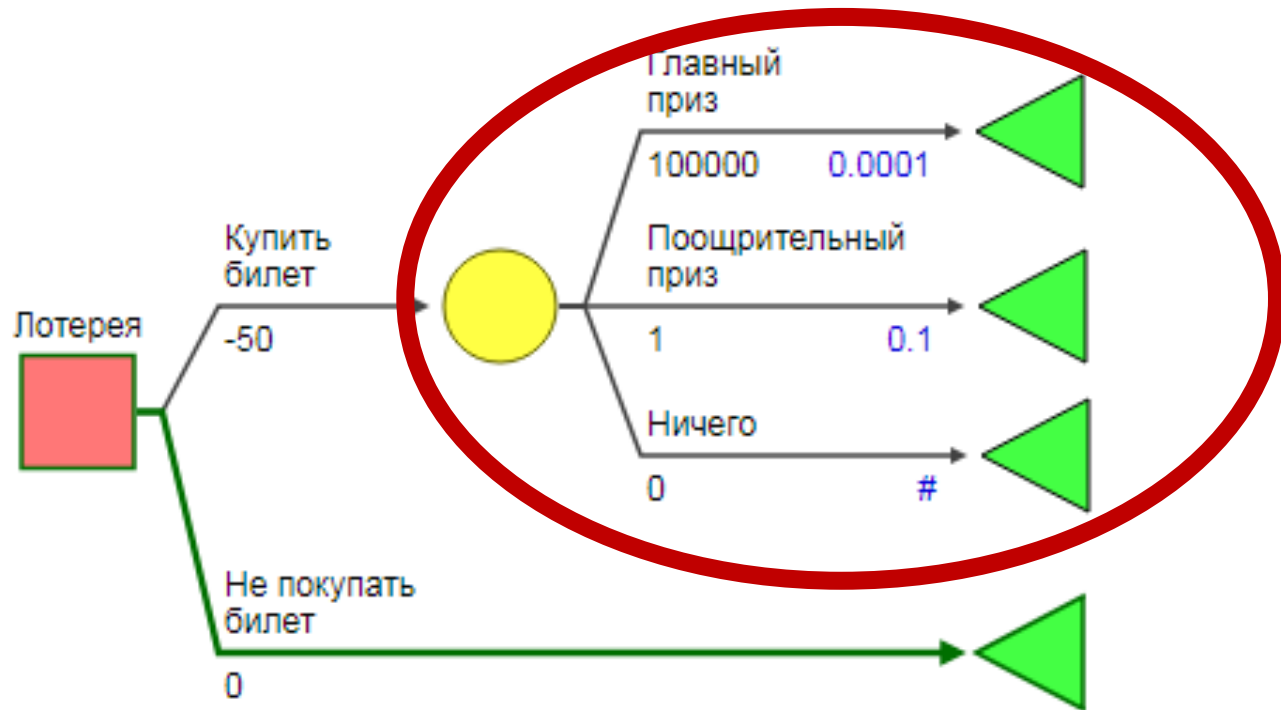
Путь от корня к какой-либо одной терминальной вершине – один из вариантов реализации ситуации принятия решения. Можно представить, что прохождение происходит во *времени*.



## Деревья решений. Обработка (1)

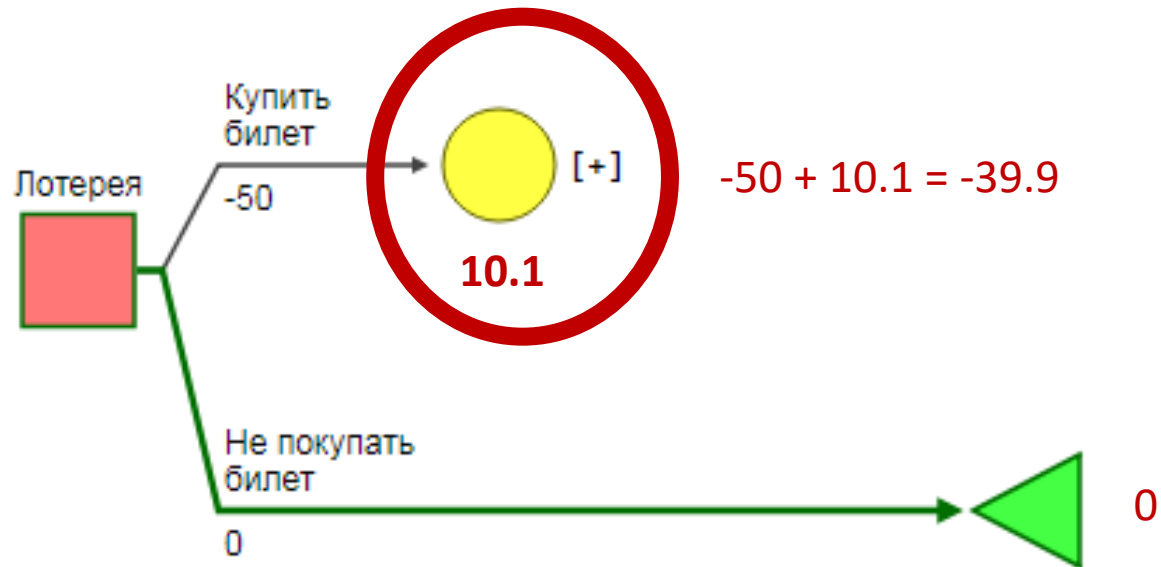


## Деревья решений. Обработка (2)



$$EU = 100000 * 0.00001 + 1 * 0.1 + 0 * (1 - 0.1 - 0.00001) = 10.1$$

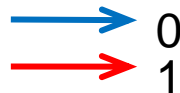
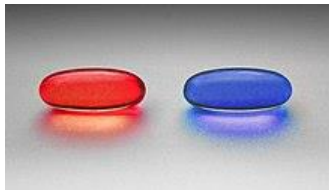
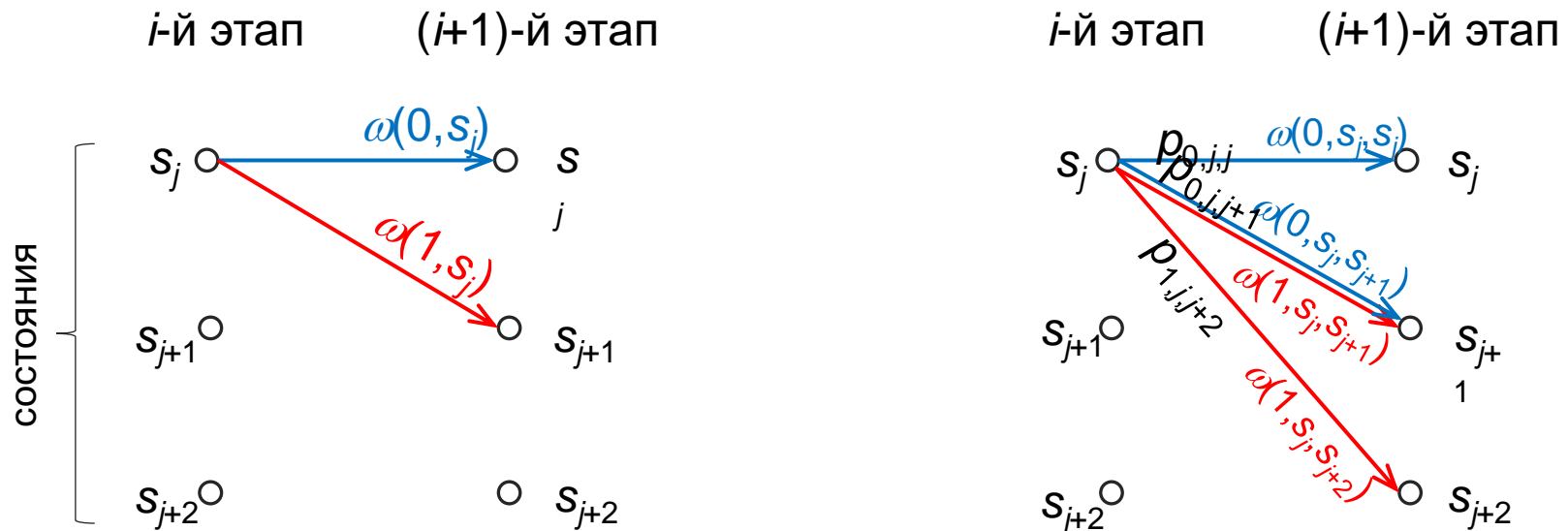
## Деревья решений. Обработка (3)



# Динамическое программирование и задачи с неопределенностью

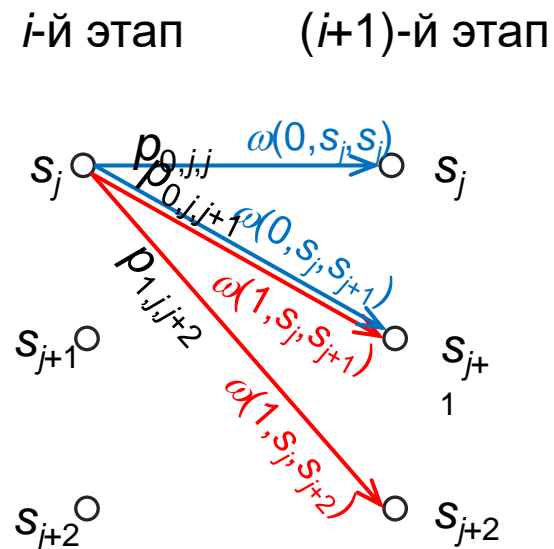
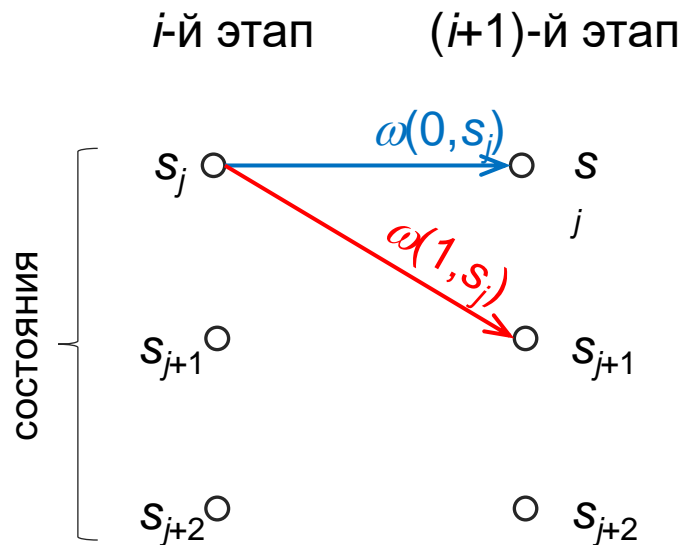


# Неопределенность в задаче управления



- разные значения управления

# Неопределенность в задаче управления



$$W_i(s_j) = \max_{x \in X} \{ \omega(x, s_j) + W_{i+1}(\varphi(s_j)) \}$$

CB

$$W_i(s_j) = \max_{x \in X} \{ M[ \omega(x, s_j, \varphi(x, s_j)) + W_{i+1}(\varphi(x, s_j)) ] \} =$$

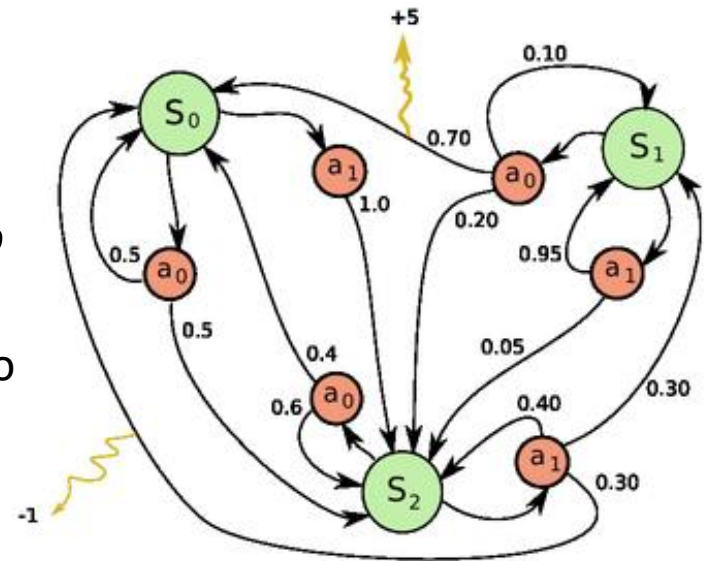
$$= \max_{x \in X} \left\{ \sum_q p_{x,j,q} ( \omega(x, s_j, s_q) + W_{i+1}(s_q) ) \right\}$$

# Траектории

- Детерминированный случай:
  - Траектория полностью определена. Решив задачу, можем восстановить путь от начального состояния до конечного.
- Задача с неопределенностью:
  - Даже после выбора оптимального управления для каждого состояния мы можем оценить лишь распределение вероятностей для каждого из шагов.
  - Траектория «рождается» по мере реализации процесса. Фактическое выполнение дает нам новую **информацию**.
    - А значит, мы можем уточнять оценку стоимости, переходя от математического ожидания к фактическим значениям.

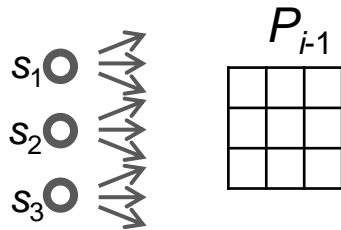
# Марковский процесс принятия решений

- Дискретное время (этапы)
- Конечное число состояний
- Переходные вероятности между состояниями описывают *марковскую цепь*
  - Условное распределение последующего состояния не зависит от предыстории состояний
- Структура вознаграждений представима в виде матрицы дохода при переходе

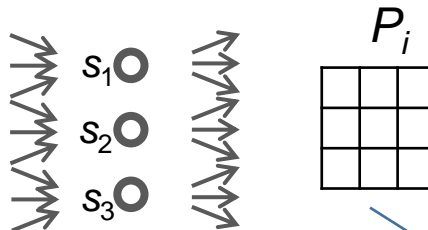


# Марковский процесс (марковская цепь)

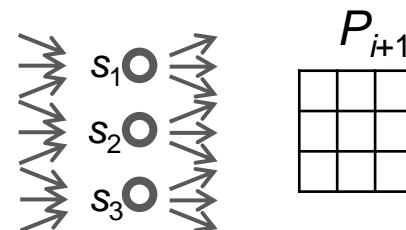
Шаг  $i-1$



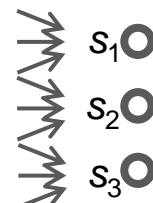
Шаг  $i$



Шаг  $i+1$



Шаг  $i+2$



Для перехода от шага  $i$  к шагу  $i+1$ :

$$p_{i+1} = p_i P_i$$

$$P_i(k,j) = p(S^{(i+1)} = s_j \mid S^{(i)} = s_k)$$

$p_i$  – вектор-строка вероятностей

# Марковский процесс (марковская цепь)

Для перехода от шага  $i$  к шагу  $i+1$ :

$$p_{i+1} = p_i P_i$$

$$\mathbb{P}(S^{(i+1)} = s_j) = \sum_{s_k} \mathbb{P}(S^{(i+1)} = s_j | S^{(i)} = s_k) \mathbb{P}(S^{(i)} = s_k)$$

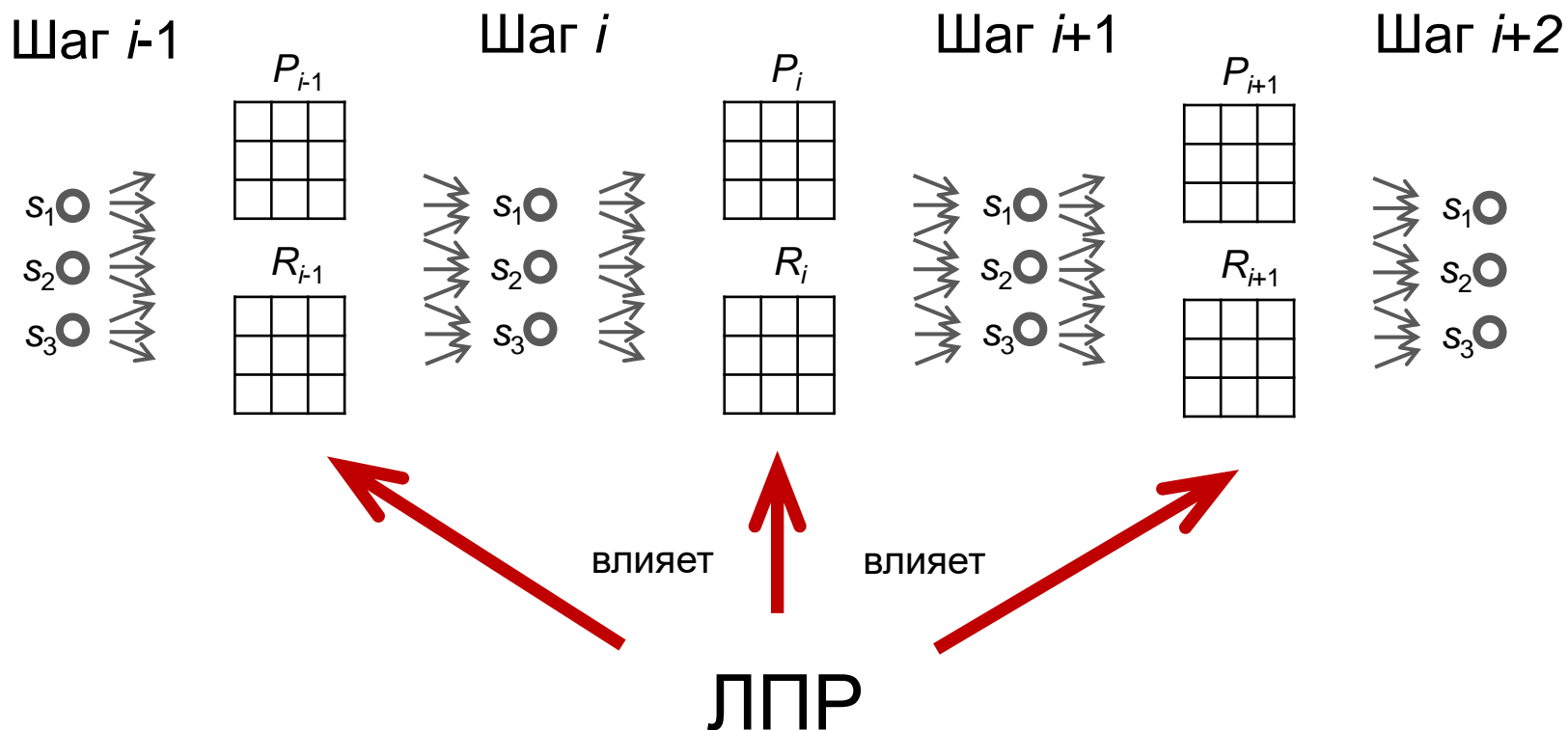
$\mathbb{P}(S^{(i)} = s_1)$	$\mathbb{P}(S^{(i)} = s_2)$	$\mathbb{P}(S^{(i)} = s_3)$
-----------------------------	-----------------------------	-----------------------------

.

$s_j$

...	$\mathbb{P}(S^{(i+1)} = s_j   S^{(i)} = s_1)$	...
...	$\mathbb{P}(S^{(i+1)} = s_j   S^{(i)} = s_1)$	...
...	$\mathbb{P}(S^{(i+1)} = s_j   S^{(i)} = s_1)$	...

# Марковский процесс принятия решений

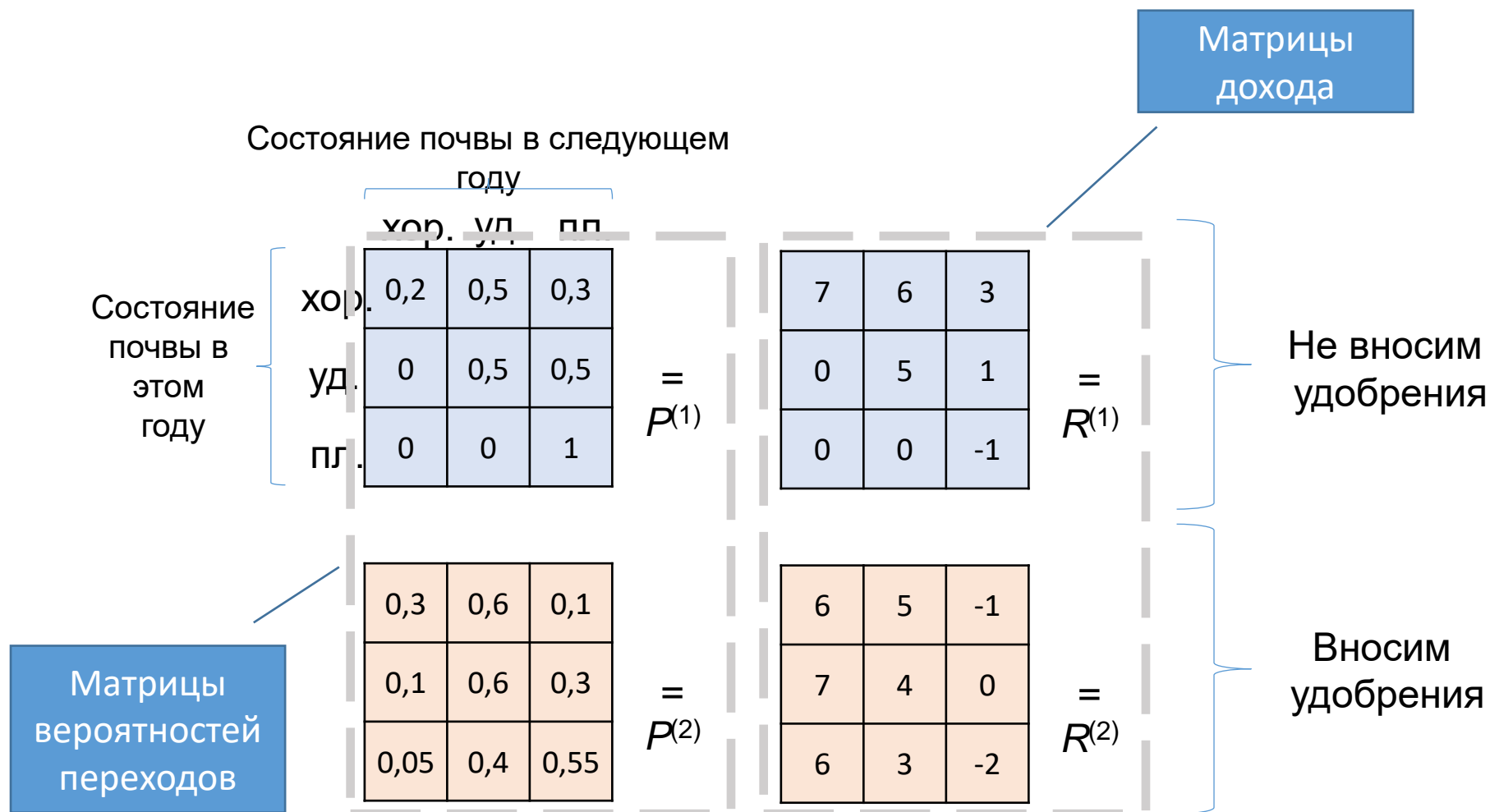


# Задача о садовнике

- Продуктивность сада зависит от состояния почвы на начало сезона и может оцениваться как
  - 1) «хорошая»,
  - 2) «удовлетворительная»,
  - 3) «плохая».
- Состояние почвы в текущем году зависит **только от состояния почвы в предыдущем году** (с учетом проведенных агротехнических мероприятий).
- Необходимо выбрать наилучшую стратегию внесения удобрений на конечное число этапов.



# Задача о садовнике

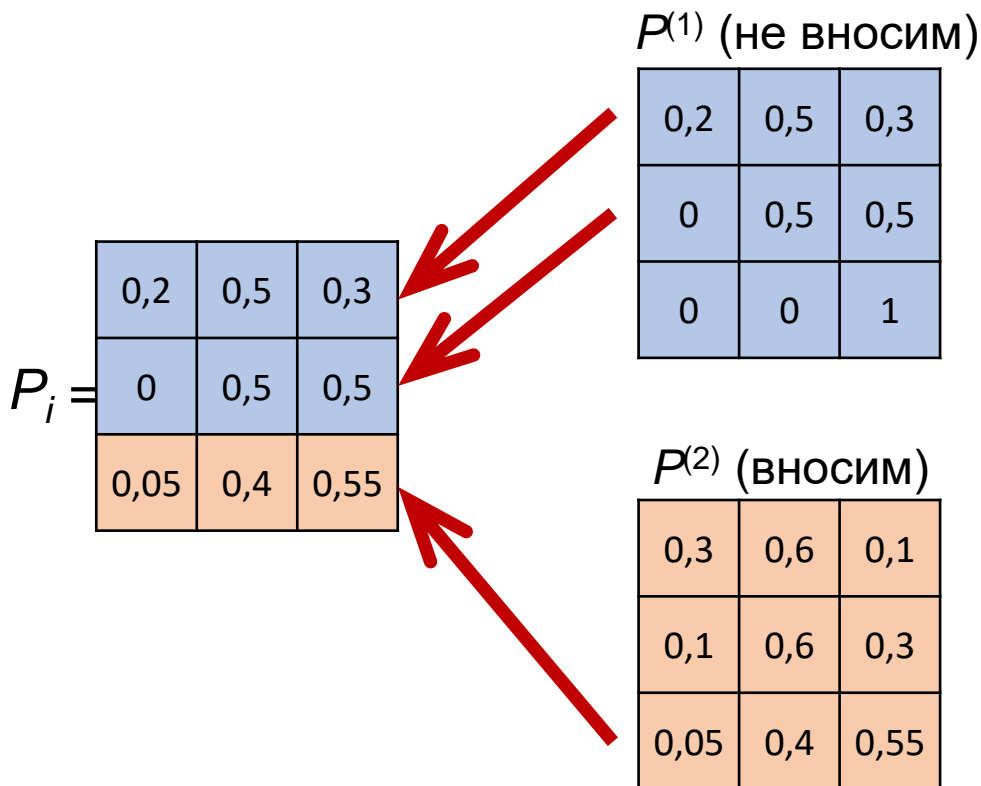


# Стратегия

- Стратегия – **правило** выбора действия **для каждого возможного состояния**
- Например, полностью заданная стратегия для некоторого **одного** шага может «говорить», что:
  - Если состояние почвы «хорошее», то «не вносить».
  - Если состояние почвы «удовлетворительное», то «не вносить».
  - Если состояние почвы «плохое», то «вносить».

# Стратегия в МППР для одного шага

Состояние	Действие (управление)
Хорошее	Не вносить
Удовл.	Не вносить
Плохое	Вносить



# Стратегия в МППР для одного шага

Состояние	Действие (управление)
Хорошее	Не вносить
Удовл.	Не вносить
Плохое	Вносить

$$P_i =$$

0,2	0,5	0,3
0	0,5	0,5
0,05	0,4	0,55

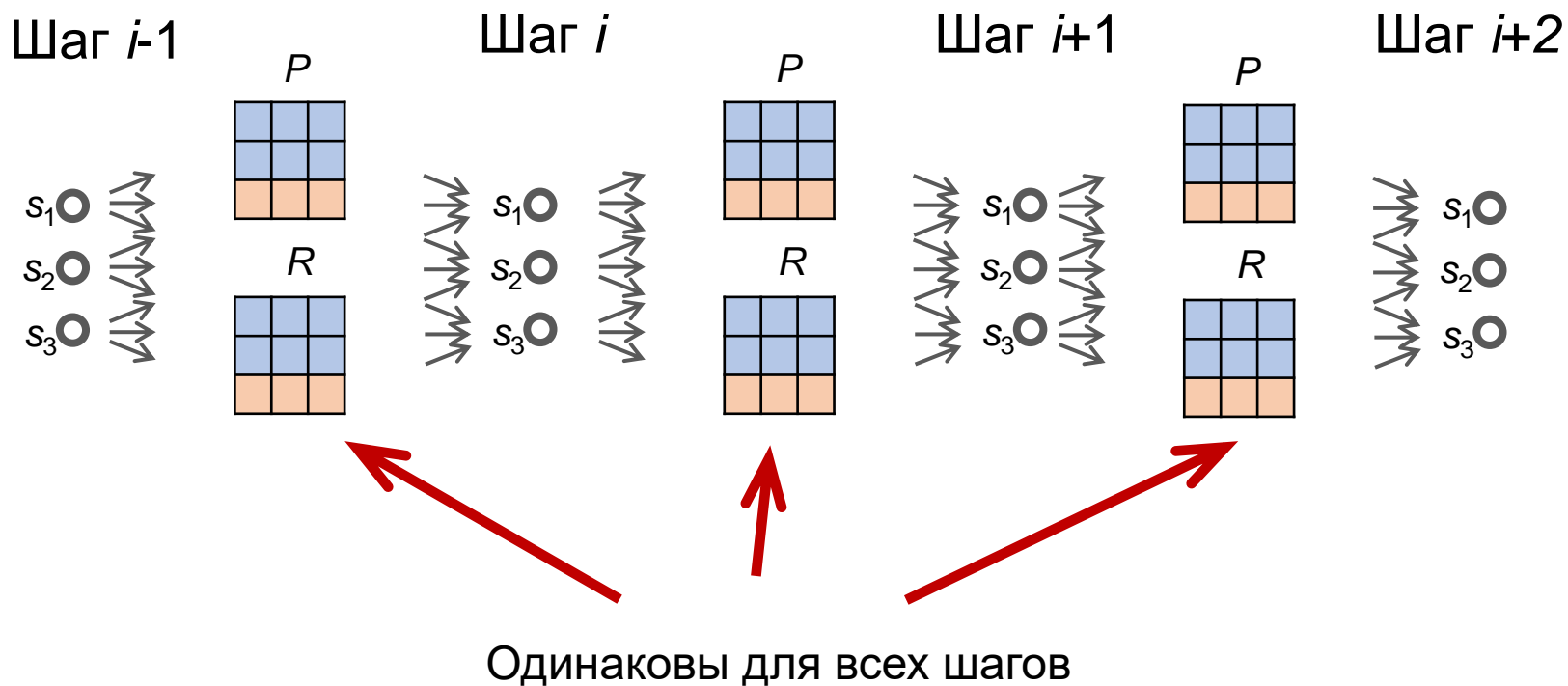
$$R_i =$$

7	6	3
0	5	1
6	3	-2

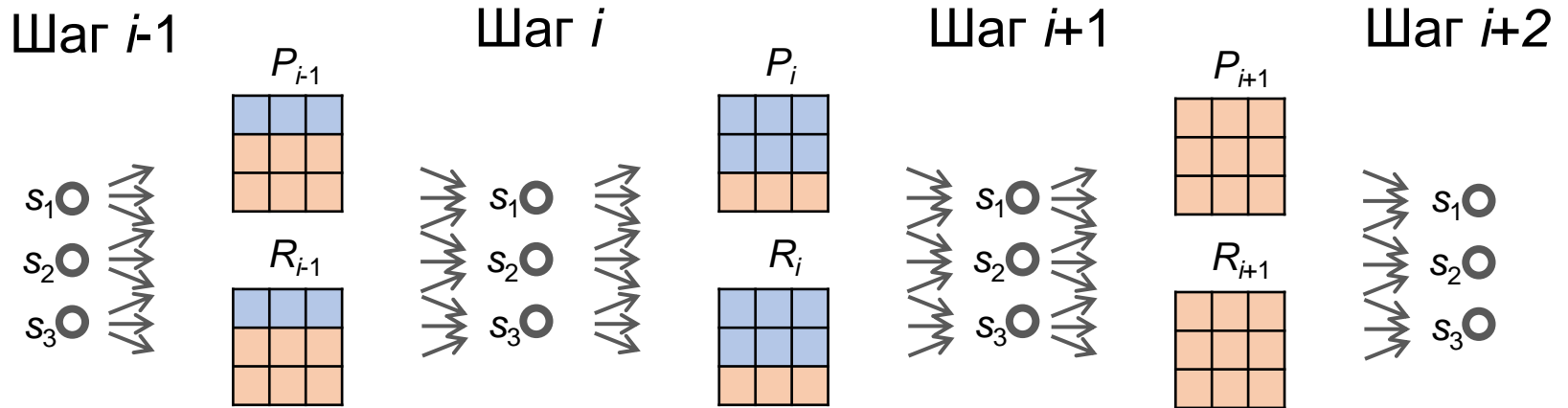
# Задачи

- Оценка стратегии (предсказание)
- Оптимизация стратегии (управление)

# Стационарная стратегия в МППР



# Стратегия в МППР для всех шагов



Рекуррентная формула для вычисления условного ожидаемого выигрыша на шагах, начиная  $i$ -того (оценка стратегии):

$$f_i(s_j) = \sum_{s_k \in S} P_i(s_j, s_k) (R_i(s_j, s_k) + f_{i+1}(s_k))$$

# Оценка выигрыша при заданной стратегии

```
1. def finite_horizon_mdp_eval(P, R, periods):
2.     # Количество состояний
3.     n_states = P.shape[1]
4.
5.     profit = np.zeros((1, n_states))
6.     for period in reversed(range(periods)):
7.         profit = np.sum(P[period] * (R[period] + profit), axis=-1)
8.     return profit
```

$$f_i(s_j) = \sum_{s_k \in S} P_i(s_j, s_k) (R_i(s_j, s_k) + f_{i+1}(s_k))$$

[https://bitbucket.org/M\\_H/dmt-examples/src/master/Dynamic%20Programming.ipynb](https://bitbucket.org/M_H/dmt-examples/src/master/Dynamic%20Programming.ipynb)



# Оптимизация стратегии методом ДП

Уравнение Беллмана:

$$W_i(s_j) = \max_{u \in U} \sum_{s_k \in S} P_i^{(u)}(s_j, s_k) \left( R_i^{(u)}(s_j, s_k) + W_{i+1}(s_k) \right)$$

# Оптимизация стратегии (Python)

```
1. # Предполагается, что P и R имеют размерность
2. # 'количество управлений' x 'кол-во состояний' x 'кол-во состояний'
3. def finite_horizon_mdp_solver(P, R, periods):
4.     # Количество состояний
5.     n_states = P.shape[1]
6.     # Сформируем результирующие матрицы
7.     profit = np.zeros((periods, n_states))      # Условно оптимальные выигрыши
8.     control = np.zeros((periods, n_states),
9.                          dtype='int32')         # Условно оптимальные управления
10.    # Для последнего этапа выигрыш на "следующем" этапе
11.    # должен быть равен нулю
12.    profit_next = np.zeros((1, n_states))
13.    for period in reversed(range(periods)):
14.        tmp = np.sum(P * (R + profit_next), axis=-1)
15.        profit[period, :] = np.max(tmp, axis=0)
16.        control[period, :] = np.argmax(tmp, axis=0)
17.        profit_next = profit[period, :]
18.    return profit, control
```

# Исчисление вероятностей для сложных проблемных областей

# Априорная и апостериорная вероятности

**Безусловная, или априорная,** вероятность  $P(a)$ , связанная с высказыванием  $a$ , представляет собой степень уверенности, относящуюся к этому высказыванию в *отсутствии любой другой информации*.

$P(\text{Cavity} = \text{true}) = 0.1$  или  $P(\text{cavity}) = 0.1$

Для всех возможных значений СВ – **распределение априорных вероятностей СВ.**

Для нескольких СВ – **совместные распределения вероятностей.**

Для всех СВ модели – **полное совместное распределение вероятностей.**

**Условная, или апостериорная,** вероятность  $P(a|b)$  представляет собой степень уверенности в том, что истинно высказывание  $a$ , если *всё, что нам известно*, это  $b$ .

# Логический вывод с использованием полных совместных распределений

**Вероятностный вывод** - вычисление апостериорных вероятностей для высказываний, заданных в виде запросов, на основании наблюдаемых свидетельств.

Простейший вариант: БЗ – полное совместное распределение.

$$P(X|e) = \alpha P(X, e) = \alpha \sum_y P(X, e, y)$$

$X$  – переменная запроса;

$E$  – множество переменных свидетельства;

$e$  – наблюдаемые значения переменных свидетельства;

$y$  – возможные значения ненаблюдаемых переменных из множества  $Y$ ;

$P(X|e)$  – запрос;

$\alpha$  – константа нормализации ( $1/P(e)$ ).

## Пример вывода с использованием полного совместного распределения (1)

	<i>influenzaEpidemic</i>		<i>not influenzaEpidemic</i>	
	<i>fever</i>	<i>not fever</i>	<i>fever</i>	<i>not fever</i>
<i>influenza</i>	0.05	0.001	0.02	0.001
<i>not influenza</i>	0.009	0.04	0.33	0.549

3 логические  
переменные:  
- *Influenza*  
- *InfluenzaEpidemic*  
- *Fever*

$$\Sigma = 1$$

## Пример вывода с использованием полного совместного распределения (2)

	<i>influenzaEpidemic</i>		<i>not influenzaEpidemic</i>	
	<i>fever</i>	<i>not fever</i>	<i>fever</i>	<i>not fever</i>
<i>influenza</i>	0.05	0.001	0.02	0.001
<i>not influenza</i>	0.009	0.04	0.33	0.549

→ 0.072

Маргинальная  
вероятность

$$P(\textit{influenza}) = 0.05 + 0.001 + 0.02 + 0.001 = 0.072$$

## Пример вывода с использованием полного совместного распределения (3)

	<i>influenzaEpidemic</i>		<i>not influenzaEpidemic</i>	
	<i>fever</i>	<i>not fever</i>	<i>fever</i>	<i>not fever</i>
<i>influenza</i>	0.05	0.001	0.02	0.001
<i>not influenza</i>	0.009	0.04	0.33	0.549

$$P(\textit{influenza}) = 0.05 + 0.001 + 0.02 + 0.001 = 0.072$$

$$\begin{aligned} P(\textit{Influenza} \mid \textit{influenzaEpidemic}) &= \alpha P(\textit{Influenza}, \textit{influenzaEpidemic}) = \\ &= \alpha \langle 0.051, 0.049 \rangle = \langle 0.51, 0.49 \rangle \end{aligned}$$



# Пример вывода с использованием полного совместного распределения (4)

	<i>influenzaEpidemic</i>		<i>not influenzaEpidemic</i>	
	<i>fever</i>	<i>not fever</i>	<i>fever</i>	<i>not fever</i>
<i>influenza</i>	0.05	0.001	0.02	0.001
<i>not influenza</i>	0.009	0.04	0.33	0.549

$$P(\textit{influenza}) = 0.05 + 0.001 + 0.02 + 0.001 = 0.072$$

$$\begin{aligned} P(\textit{Influenza} \mid \textit{influenzaEpidemic}) &= \alpha P(\textit{Influenza}, \textit{influenzaEpidemic}) = \\ &= \alpha \langle 0.051, 0.049 \rangle = \langle 0.51, 0.49 \rangle \end{aligned}$$

$$\begin{aligned} P(\textit{Influenza} \mid \textit{influenzaEpidemic} \cap \textit{fever}) &= \\ &= \alpha P(\textit{Influenza}, \textit{influenzaEpidemic}, \textit{fever}) = \\ &= \alpha \langle 0.05, 0.009 \rangle \approx \langle 0.85, 0.15 \rangle \end{aligned}$$

# Краткий анализ

**Полезный вывод:** полное совместное распределение *может* быть использовано для получения ответа на *любые* вероятностные запросы.

**Но совершенно непрактично:**

- Объем  $2^n$  значений, где  $n$  – количество переменных (СВ);
  - Где хранить?
  - Как наполнять?
- Обработка таблицы –  $O(2^n)$ .

# Правило Байеса

$$P(b|a) = \frac{P(a|b)P(b)}{P(a)}$$

*Три вероятности для вычисления одной. В чём выигрыш?*



# Виды информации (в системах диагностики)

**Диагностическая информация** – наличие симптома  $a$  свидетельствует в пользу (заболевания, неисправности)  $b$ .  
Встречается *редко*, подвержена влиянию внешних условий.

**Причинная информация** – (заболевание, неисправность)  $b$  с определённой вероятностью вызывает симптом  $a$ .  
Встречается *часто*, менее подвержена влиянию внешних условий.

$$P(\text{заболевание}|\text{симптом}) = \frac{P(\text{симптом}|\text{заболевание})P(\text{заболевание})}{P(\text{симптом})}$$

Диагностическая

Причинная

# Комбинирование свидетельств

Что, если нам известно несколько свидетельств (симптомов)?

$$P(Flu|fever = true \wedge sorethroat = true)$$

Применение правила Байеса даёт:

$$P(Flu|fever \wedge sorethroat) = \frac{P(fever \wedge sorethroat|Flu)P(Flu)}{P(fever \wedge sorethroat)}$$

**Но! Непосредственное применение этого правила требует определения условных вероятностей для всех возможных сочетаний свидетельств!**

# Комбинирование свидетельств. Независимость

**Независимость СВ:**

$$P(a|b) = P(a)$$
$$P(a \wedge b) = P(a)P(b)$$

**Условная независимость СВ:**

$$P(a \wedge b|c) = P(a|c)P(b|c)$$

т.е.  $a$  и  $b$  являются независимыми, при условии, что задано значение некоторой третьей СВ. Чаще всего, и  $a$ , и  $b$ , зависят от  $c$  но не зависят друг от друга.

Из предположения об условной независимости *fever* и *sorethroat*:

$$P(Flu|fever \wedge sorethroat) = \alpha P(fever|Flu)P(sorethroat|Flu)P(Flu)$$

# Байесовская сеть

**Байесовская сеть** — это ориентированный граф, в котором каждая вершина помечена количественной вероятностной информацией.

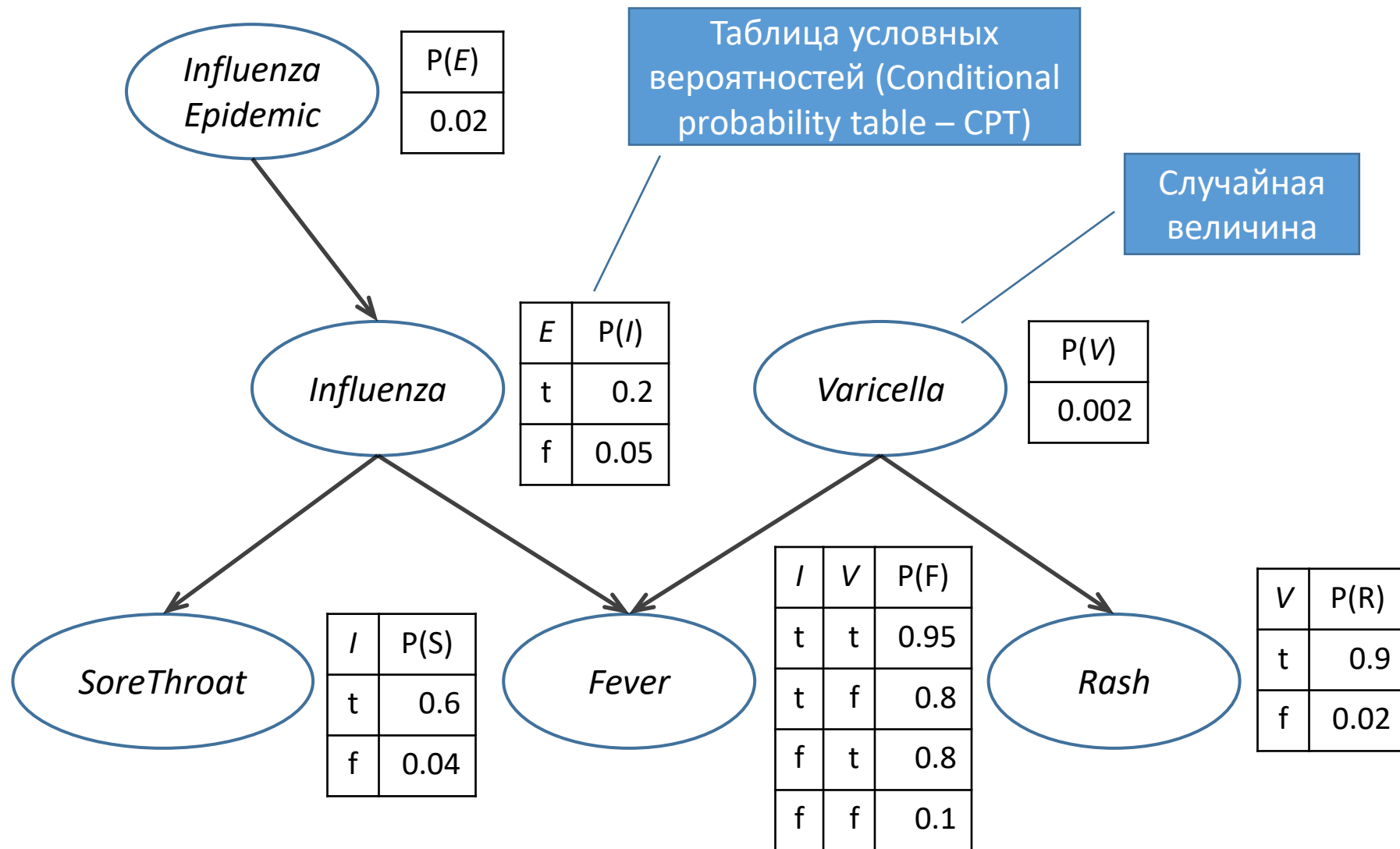
1. Вершинами сети является множество случайных переменных. Переменные могут быть дискретными или непрерывными.
2. Вершины соединяются попарно ориентированными ребрами; ребра образуют множество ребер. Если стрелка направлена от вершины  $X$  к вершине  $Y$ , то вершина  $X$  называется родительской вершиной вершины  $Y$ .
3. Каждая вершина  $X_i$  характеризуется распределением условных вероятностей  $P(X_i \mid \text{Parents}(X_i))$ , которое количественно оценивает влияние родительских вершин на эту вершину.
4. Граф не имеет контуров (Directed Acyclic Graph — DAG).

# Байесовская сеть

**Топология** сети показывает отношения, определяющие условную независимость, которые проявляются в данной проблемной области. *Интуитивный* смысл стрелки в правильно составленной сети обычно состоит в том, что вершина  $X$  оказывает *непосредственное* влияние на вершину  $Y$ .



# Пример байесовской сети



# Представление полного совместного распределения

Каждый элемент в полном совместном распределении вероятностей может быть рассчитан на основании информации, представленной в байесовской сети.

$$P(x_1, \dots, x_n) = \prod_{i=1}^n P(x_i | \text{parents}(X_i))$$

где  $\text{parents}(X_i)$  обозначает конкретные значения переменных в множестве вершин  $\text{Parents}(X_i)$ . Поэтому каждый элемент в совместном распределении представлен в виде произведения соответствующих элементов в таблицах условных вероятностей (Conditional Probability Table — CPT) байесовской сети.

Таким образом, таблицы CPT обеспечивают **декомпонованное представление совместного распределения**.

# Составление байесовской сети

Байесовская сеть служит правильным представлением проблемной области, только если *каждая вершина в ней условно независима от ее предшественников в конкретном упорядочении вершин, после того как заданы ее родительские вершины.*

Необходимо выбрать для каждой вершины родительские вершины так, чтобы соблюдалось это свойство. Интуитивно ясно, что множество родительских вершин вершины  $X_i$  должно включать все такие вершины из множества  $X_1, \dots, X_{i-1}$ , которые *непосредственно* влияют на  $X_i$ .

Правильный порядок вершин (от причин к следствиям). Правила должны быть *причинными*, а не *диагностическими*.

# Точный вероятностный вывод в байесовских сетях

Основной задачей для любой системы вероятностного вывода является вычисление распределения апостериорных вероятностей для множества **переменных запроса**, если дано некоторое наблюдаемое **событие**, т.е. если выполнено некоторое присваивание значений множеству **переменных свидетельства**.

## Система обозначений:

$X$  — обозначает переменную запроса;

$E$  — множество переменных свидетельства,  $E_1, \dots, E_m$ ;

$e$  — конкретное наблюдаемое событие;

$Y$  обозначает переменные, отличные от переменных свидетельства,  $Y_1, \dots, Y_l$  (иногда называемые **скрытыми переменными**).

Полное множество переменных определяется выражением  $X = \{X\} \cup E \cup Y$ .

В типичном запросе содержится просьба определить распределение апостериорных вероятностей  $P(X \mid e)$ .

# Вероятностный вывод с помощью перебора

## Предпосылки:

1. Условную вероятность можно вычислить суммируя элементы из полного совместного распределения:

$$P(X|e) = \alpha P(X, e) = \alpha \sum_y P(X, e, y)$$


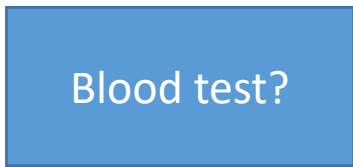

2. Байесовская сеть является представлением полного совместного распределения:

$$P(x_1, \dots, x_n) = \prod_{i=1}^n P(x_i | \text{parents}(X_i))$$



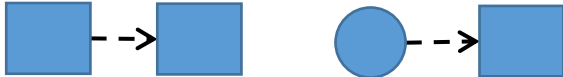
## Идея:

Вычисляем результат запроса, суммируя произведения условных вероятностей из сети.

# Диаграммы влияния. Вершины

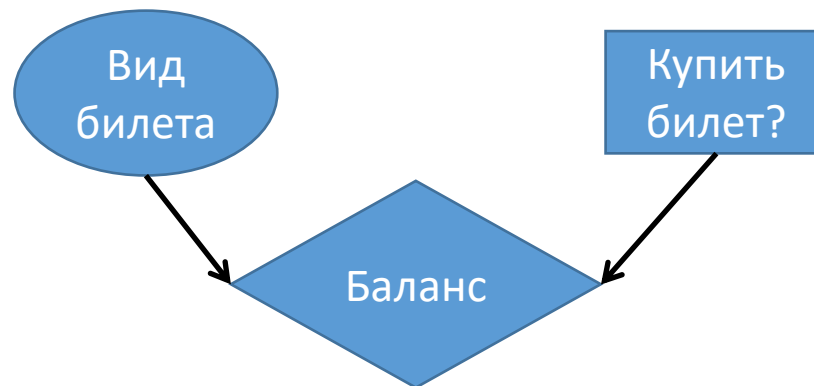
	Вершина «шанса» (chance node), соответствующая случайной переменной	Таблица условных вероятностей
	Вершина принятия решения (decision node)	Перечень допустимых значений
	Вершина полезности (utility node)	Таблица значений полезности для каждого сочетания «входящих» переменных

# Диаграммы влияния. Дуги

$A \longrightarrow B$	<p><i>Условная дуга</i> Значение A участвует в условном распределении B</p>	
$A \longrightarrow B$	<p><i>Функциональная дуга</i> Значение A используется при расчете функции полезности</p>	
$A \dashrightarrow B$	<p><i>Информационная дуга</i> Значение A определяется до значения B</p>	

# Диаграммы влияния. Пример 1

Значения	$P()$
Ничего	0.8999
Поощрительный приз	0.1
Главный приз	0.0001

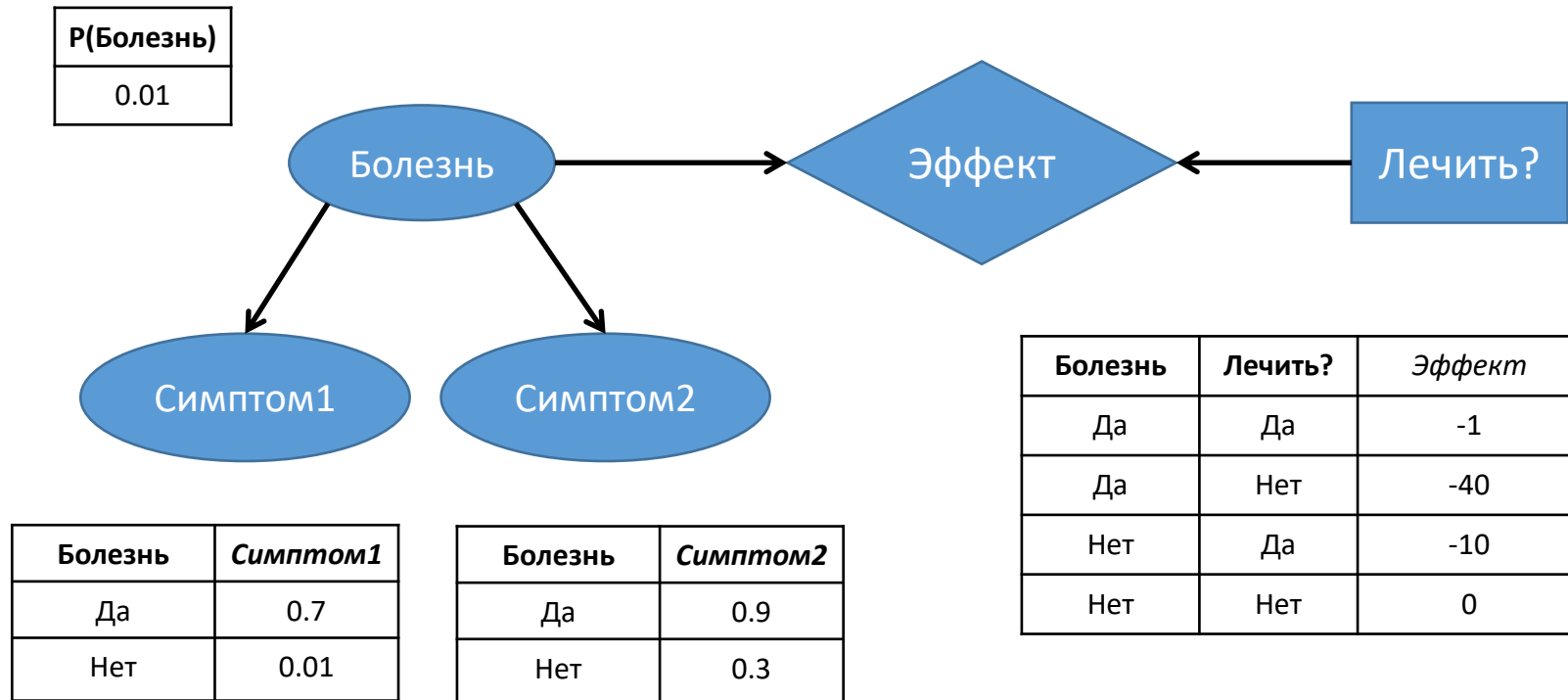


Значения
Да
Нет

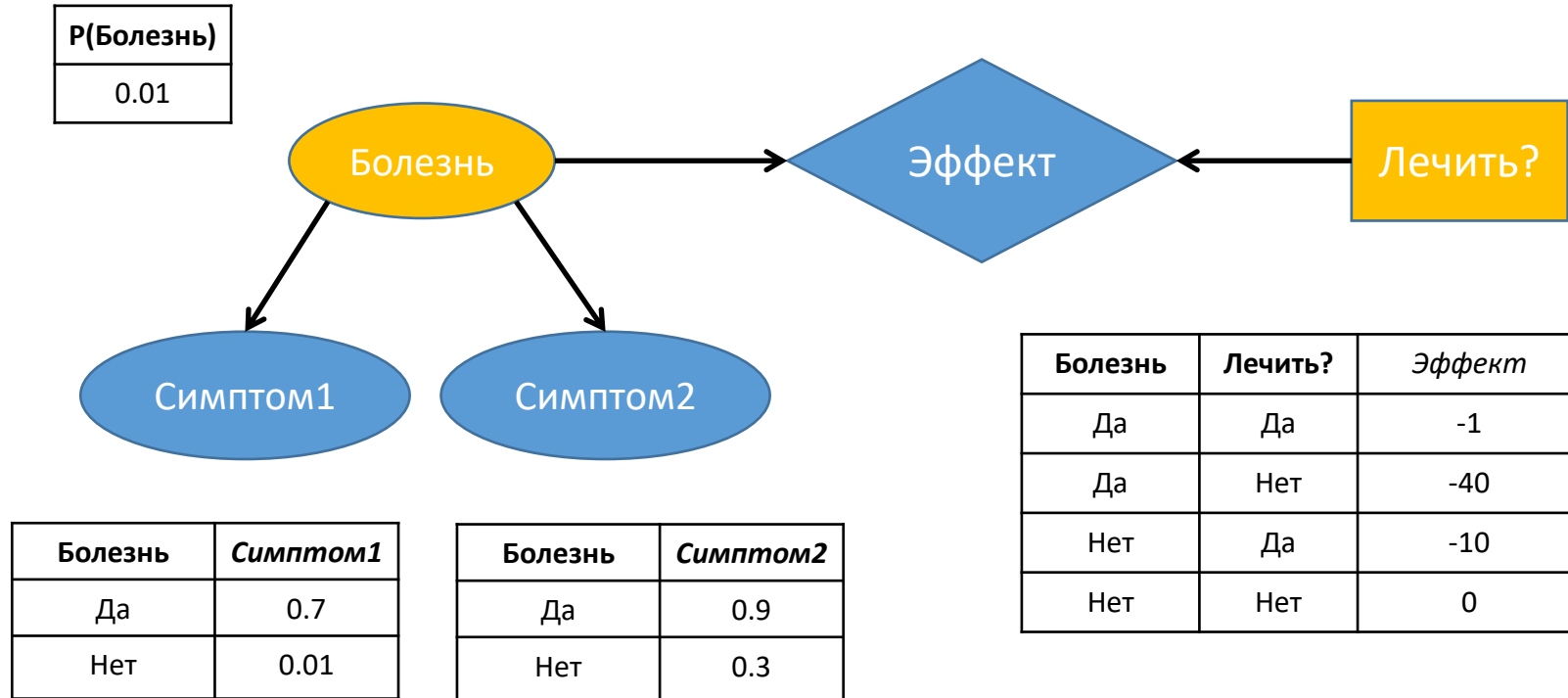
Вид билета	Купить билет?	Баланс
Ничего	Да	-50
Ничего	Нет	0
Поощрительный приз	Да	-49
Поощрительный приз	Нет	0
Главный приз	Да	99950
Главный приз	Нет	0



# 

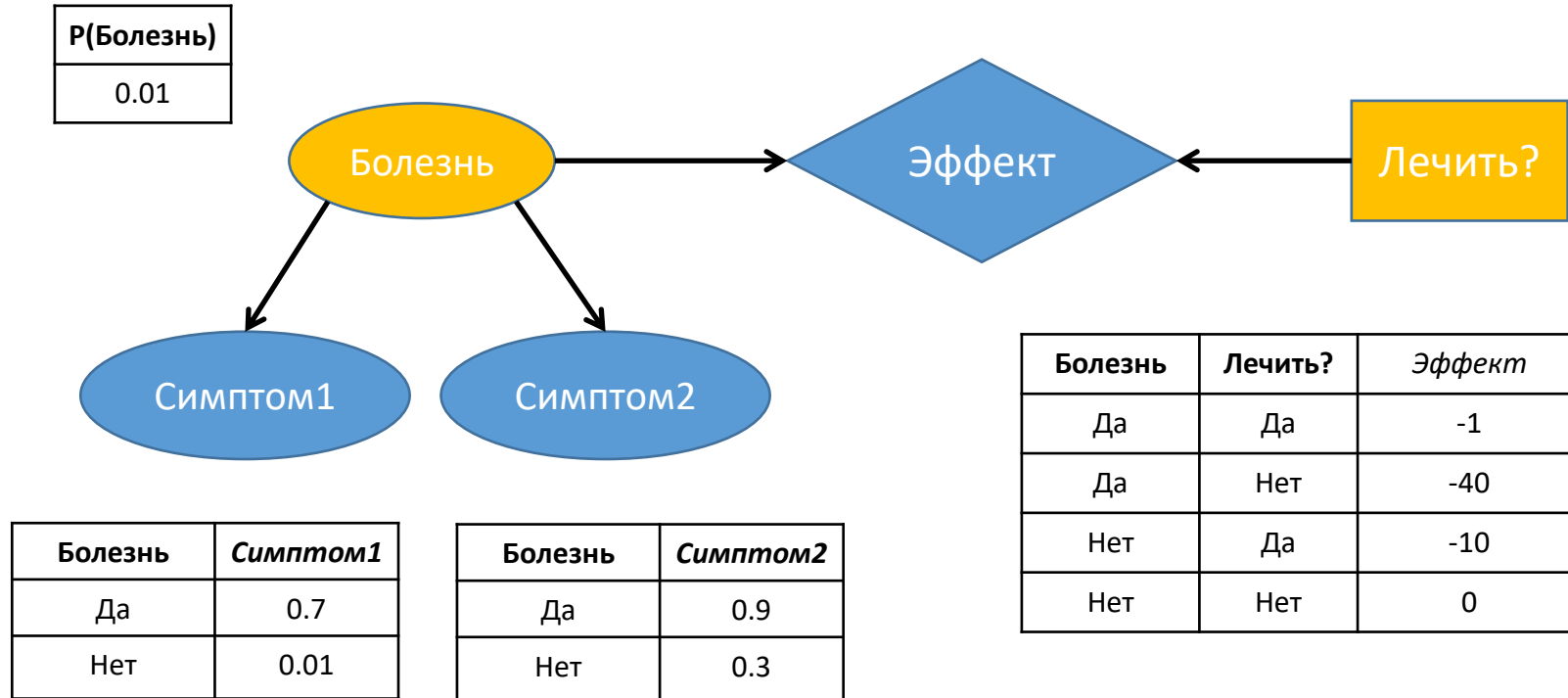


# Диаграммы влияния. Обработка



$$\begin{aligned}
 EU(L = \text{Да}) &= \sum_{b \in \{\text{Да}, \text{Нет}\}} P(B = b | L = \text{Да}) * \mathcal{E}(B = b, L = \text{Да}) = \\
 &= \sum_{b \in \{\text{Да}, \text{Нет}\}} P(B = b) * \mathcal{E}(B = b, L = \text{Да}) = 0.01 * (-1) + 0.99 * (-10) = -9.91
 \end{aligned}$$

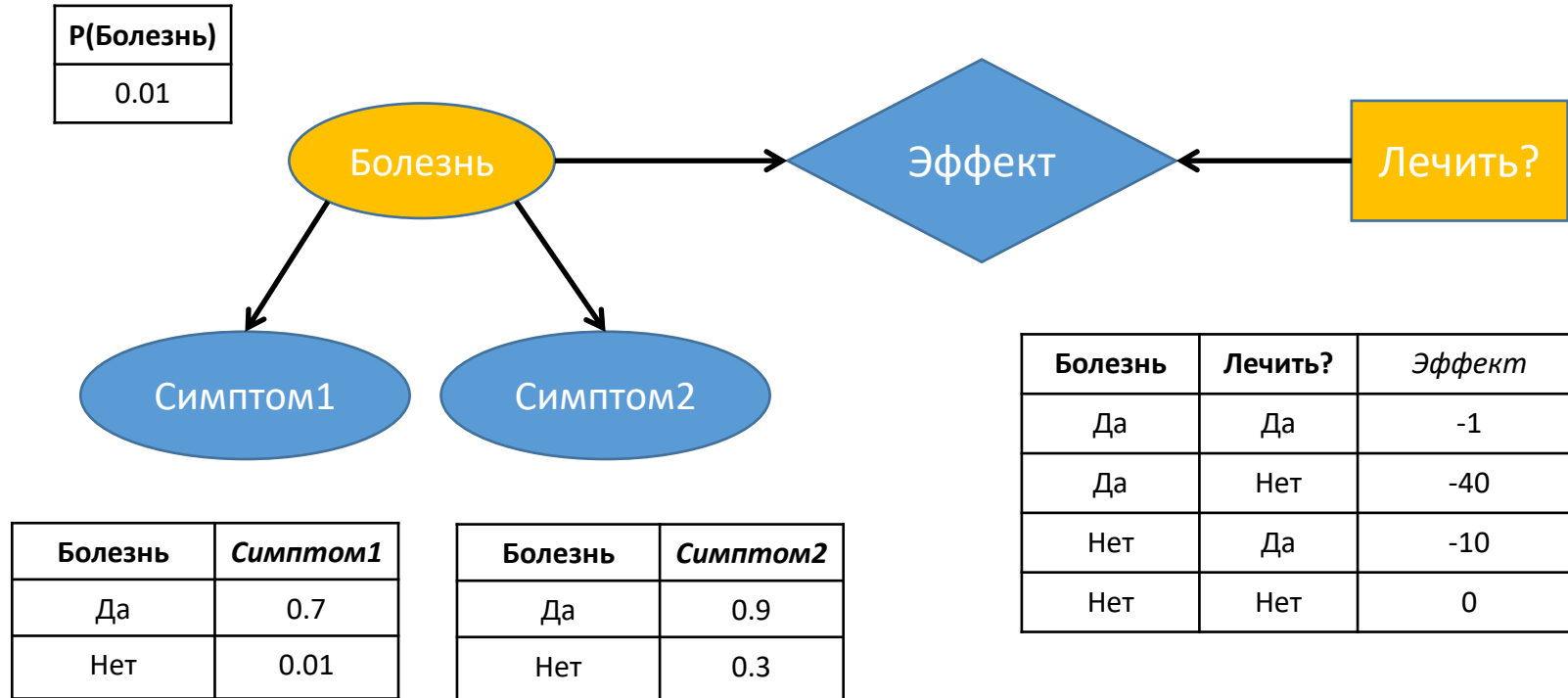
# Диаграммы влияния. Обработка



$$EU(L = \text{Да}) = -9.91$$

$$EU(L = \text{Нет}) = 0.01 * (-40) + 0.99 * (0) = -0.4$$

# Диаграммы влияния. Обработка



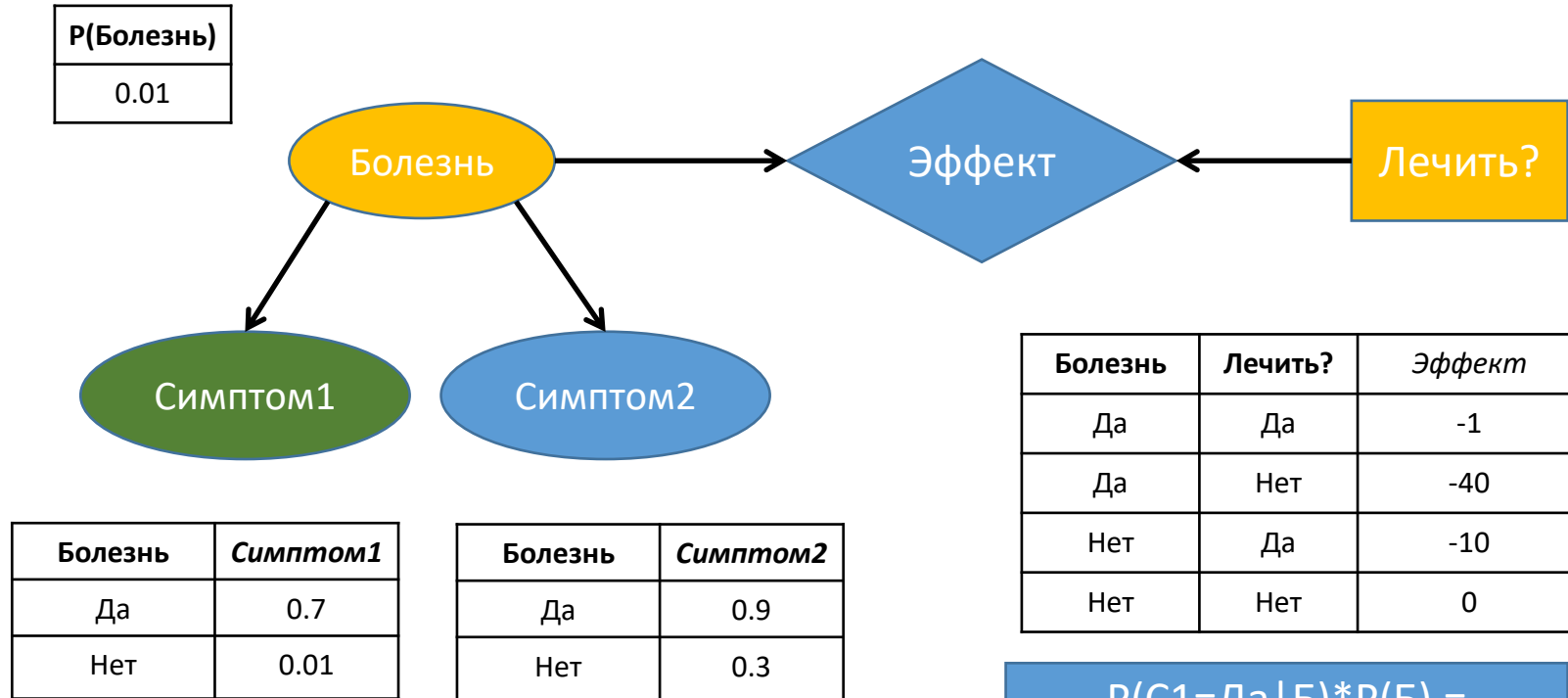
$$EU(L = \text{Да}) = -9.91$$

$$EU(L = \text{Нет}) = 0.01 * (-40) + 0.99 * (0) = -0.4$$

$$L^* = \arg \max_{L \in \{\text{Да}, \text{Нет}\}} EU(L = L) = \text{Нет}$$

Другими словами,  
если мы больше  
ничего не знаем, то  
лучше пациента не  
лечить

# Диаграммы влияния. Обработка

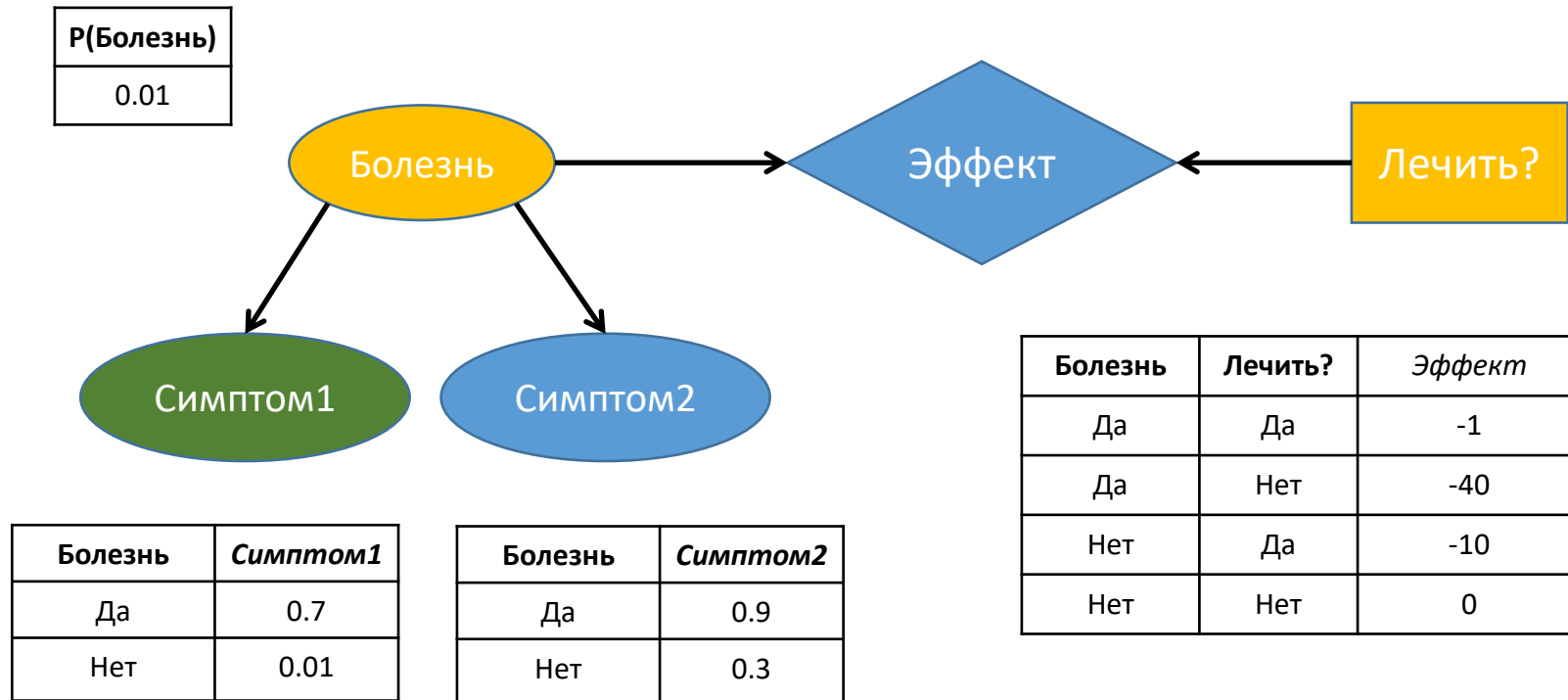


$$P(C1=\text{Да} | Б) * P(Б) = \langle 0.41, 0.59 \rangle$$

$$EU(Л = \text{Да} | \text{C1} = \text{Да}) = \sum_{б \in \{\text{Да}, \text{Нет}\}} P(Б = б | \text{C1} = \text{Да}, Л = \text{Да}) * Э(Б = б, Л = \text{Да}) =$$

$$= 0.41 * (-1) + 0.59 * (-10) = -6.31$$

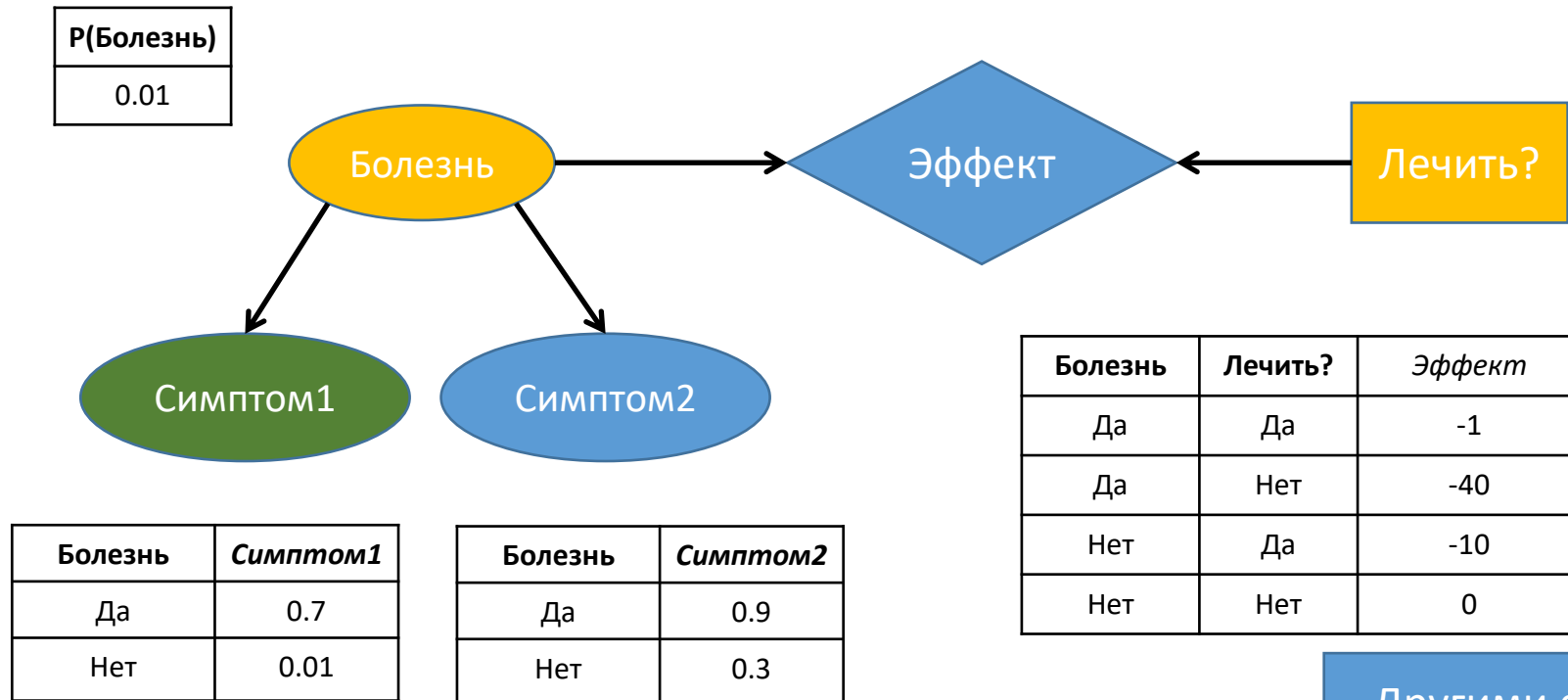
# Диаграммы влияния. Обработка



$$EU(L = \text{Да} | C1 = \text{Да}) = -6.31$$

$$EU(L = \text{Нет} | C1 = \text{Да}) = 0.41 * (-40) + 0.59 * (0) = -16.4$$

# Диаграммы влияния. Обработка



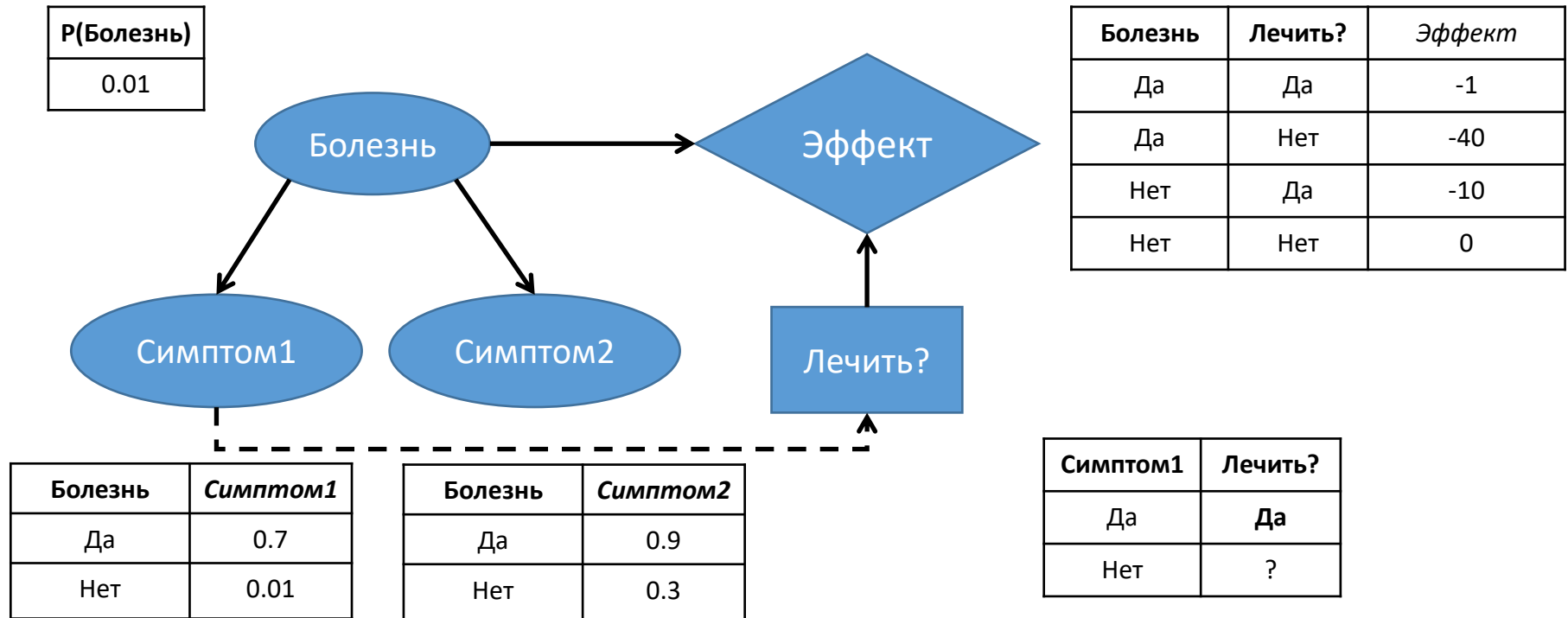
$$EU(L = \text{Да} | C1 = \text{Да}) = -6.31$$

$$EU(L = \text{Нет} | C1 = \text{Да}) = 0.41 * (-40) + 0.59 * (0) = -16.4$$

$$L^* = \arg \max_{L \in \{\text{Да}, \text{Нет}\}} EU(L = L | C1 = \text{Да}) = \text{Да}$$

Другими словами,  
если мы знаем  
только, что у  
пациента  
наблюдается  
Симптом 1, то  
пациента стоит  
лечить

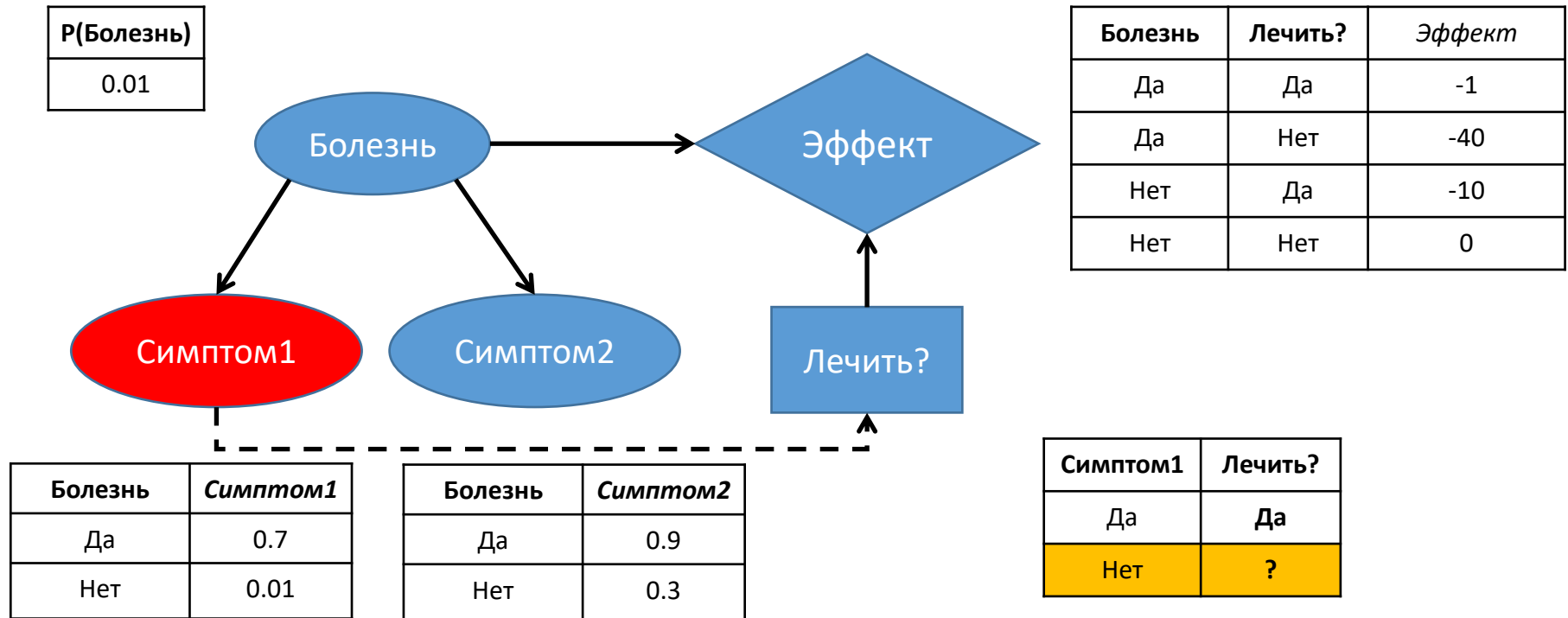
# Диаграммы влияния. Обработка



Цель – определение для каждой вершины решения решающего правила (стратегии, policy), отображающего все возможные комбинации значений переменных, известных на момент принятия решения, значения переменной решения.



# Диаграммы влияния. Обработка

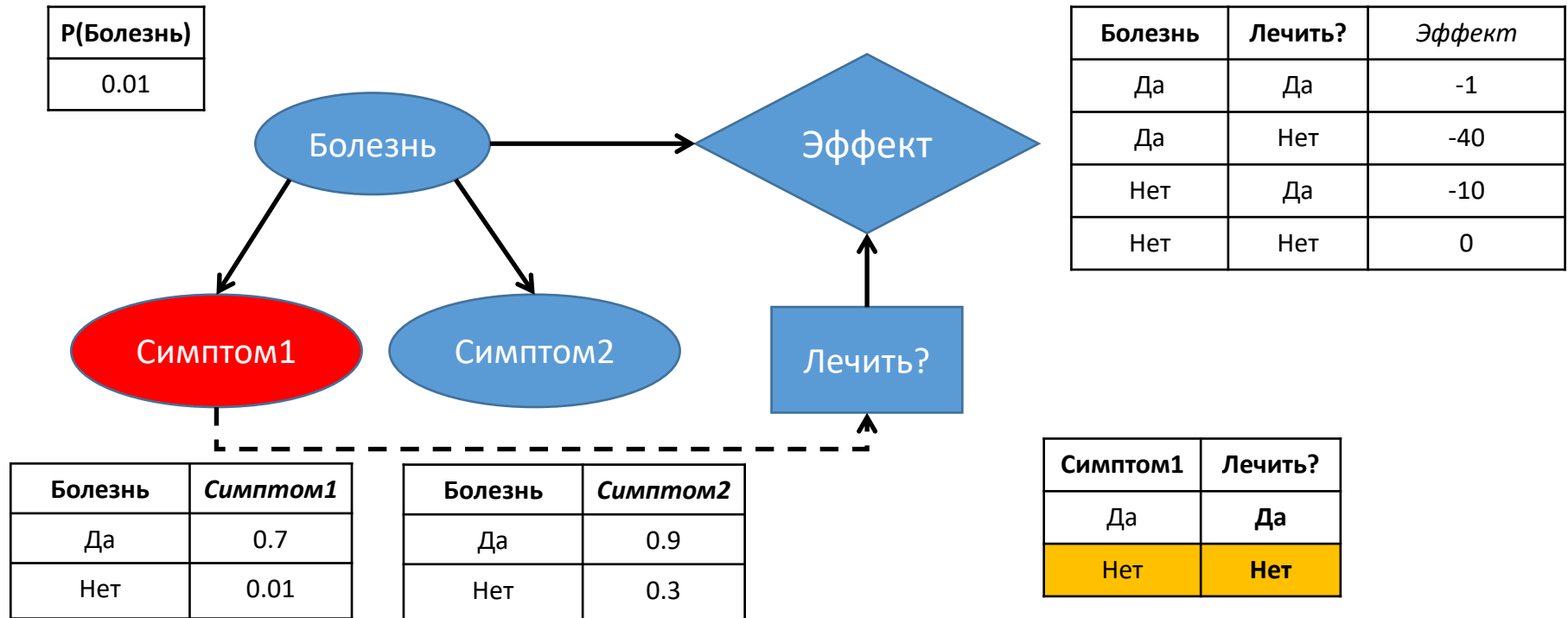


$$P(\text{Болезнь} | C1 = \text{Нет}) = \alpha P(C1 = \text{Нет} | \text{Болезнь}) P(\text{Болезнь}) = \\ = \alpha \langle 0.3 * 0.01, 0.99 * 0.99 \rangle = \langle 0.003, 0.997 \rangle$$

$$EU(L = \text{Да} | C1 = \text{Нет}) = 0.003 * (-1) + 0.997 * (-10) = -9.973$$

$$EU(L = \text{Нет} | C1 = \text{Нет}) = 0.003 * (-40) + 0.997 * (0) = -0.12$$

# Диаграммы влияния. Обработка

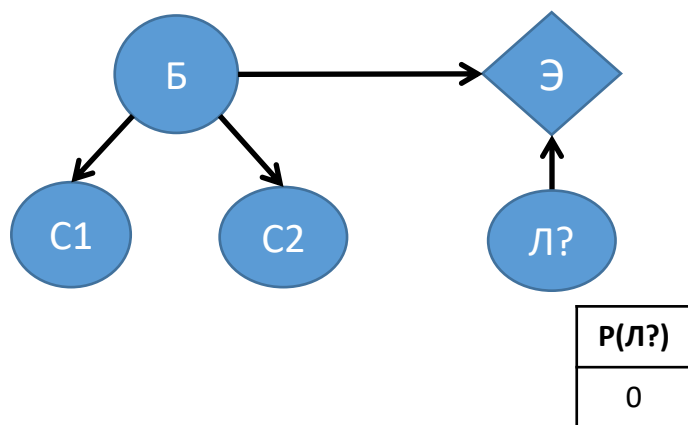


$$P(\text{Болезнь} | C1 = \text{Нет}) = \alpha P(C1 = \text{Нет} | \text{Болезнь}) P(\text{Болезнь}) = \\ = \alpha \langle 0.3 * 0.01, 0.99 * 0.99 \rangle = \langle 0.003, 0.997 \rangle$$

$$EU(L = \text{Да} | C1 = \text{Нет}) = 0.003 * (-1) + 0.997 * (-10) = -9.973$$

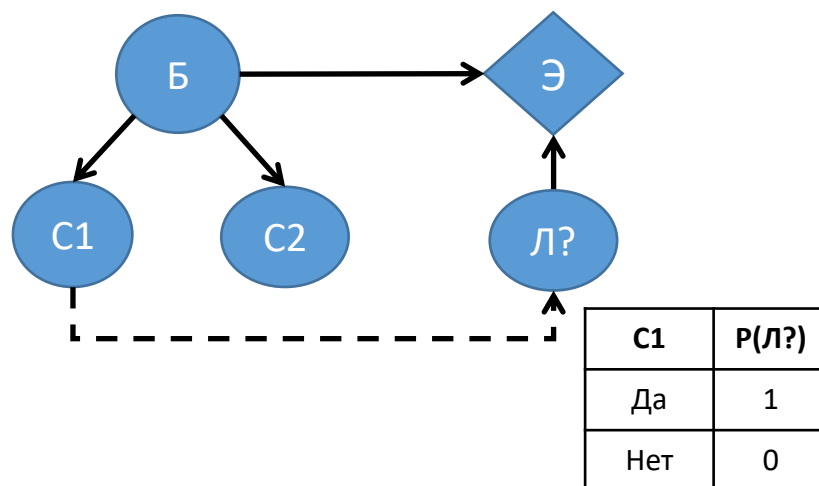
$$EU(L = \text{Нет} | C1 = \text{Нет}) = 0.003 * (-40) + 0.997 * (0) = -0.12$$

# Цена информации



Болезнь	Лечить?	$P(.)$	Эффект
Да	Да	0	-1
Да	Нет	0.01	-40
Нет	Да	0	-10
Нет	Нет	0.99	0

$$EU = -0.4$$



Болезнь	Лечить?	$P(.)$	Эффект
Да	Да	0.007	-1
Да	Нет	0.003	-40
Нет	Да	0.0099	-10
Нет	Нет	0.98	0

$$EU = -0.226$$

$$VOI = -0.226 - (-0.4) = 0.174$$

# Резюме

- Теория полезности фон Неймана - Morgenштерна:
  - Ожидаемая полезность
  - Принцип максимизации ожидаемой полезности
- Динамическое программирование оказывается полезным и при выборе оптимального управления в системах с неопределенностью
  - Например, максимизация ожидаемой выгоды
- Марковский процесс принятия решений (МППР, MDP) – широко распространенное обобщение учета неопределенности в задаче формирования управления
  - Решается (в том числе) с помощью динамического программирования
- Моделирование ситуаций принятия решения в условиях неопределенности:
  - Деревья решений
  - Диаграммы влияния
- **Не затронули (а важно!):**
  - Откуда брать вероятности (особенно, субъективные)?
  - Что делать с функцией полезности, если параметров оценки решений два или больше?
  - Как эффективно осуществлять вывод в байесовских сетях/диаграммах влияния?

# Рекомендуемая литература

- 1) Рассел С., Норвиг П. Искусственный интеллект. Современный подход. 2-е издание (Гл. 13, 14) – байесовские сети: построение, алгоритмы вывода
- 2) Clemen R. Making Hard Decisions: An Introduction to Decision Analysis. 2nd edition. Duxbury, 1997 – деревья решений, диаграммы влияния (influence diagrams), общие вопросы
- 3) Курс проф. Д. Кёллер на Coursera (Probabilistic Graphical Models 1: Representation)
- 4) Таха Х. Введение в исследование операций. 7е издание. – Вильямс, 2007.
- 5) Bertsekas D. Dynamic Programming and Optimal Control.

## Библиотеки и инструменты:

- Деревья решений:
  - Silver Decisions - <http://silverdecisions.pl>
- Байесовские сети и диаграммы влияния:
  - 1) Samlam (Sensitivity Analysis, Modeling, Inference and More)  
<http://reasoning.cs.ucla.edu/samiam/>
  - 2) Python:
    - PyMC3
    - PgmPy
    - PyAgrum