Понятийный анализ и контекстная технология моделирования

Выхованец В.С.
Институт проблем управления РАН
http://valery.vykhovanets.ru

Содержание

6 слайдов

14 слайдов

11 слайдов

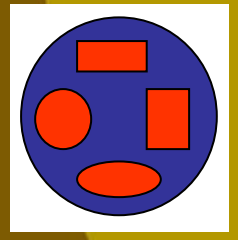
5 слайдов

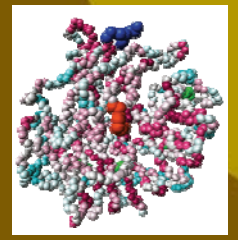
8 слайдов

1 слайд

План доклада

- Введение
- Основная проблема
- Суть подхода
- Демонстрационные примеры
- Контекстная технология
- Содержательные выводы





Парадигмы познания

- **Панпсихизм** (витализм, анимизм, гилозоизм) – одушевленность (необъяснимость) материального мира;
- **Редукционизм** (механицизм, атомизм, детерминизм) – сводимость сложного к простому;
- **Системный подход** – целое, которое не равно сумме своих частей.
- **Корпусной подход** – многоаспектность и мультисистемность целого

Корпусной подход

Что выделяется и познается?

Объект – фрагмент объективной реальности, рассматриваемый с точки зрения решения нескольких прикладных проблем.

Как представ-ляется?

Предмет – предметная область как совокупность всех накопленных о ней знаний.

Каким образом исследуется?

Метод – многоаспектный анализ знаний и их формализация в виде нескольких взаимодополняющих друг друга теорий (корпус теорий).

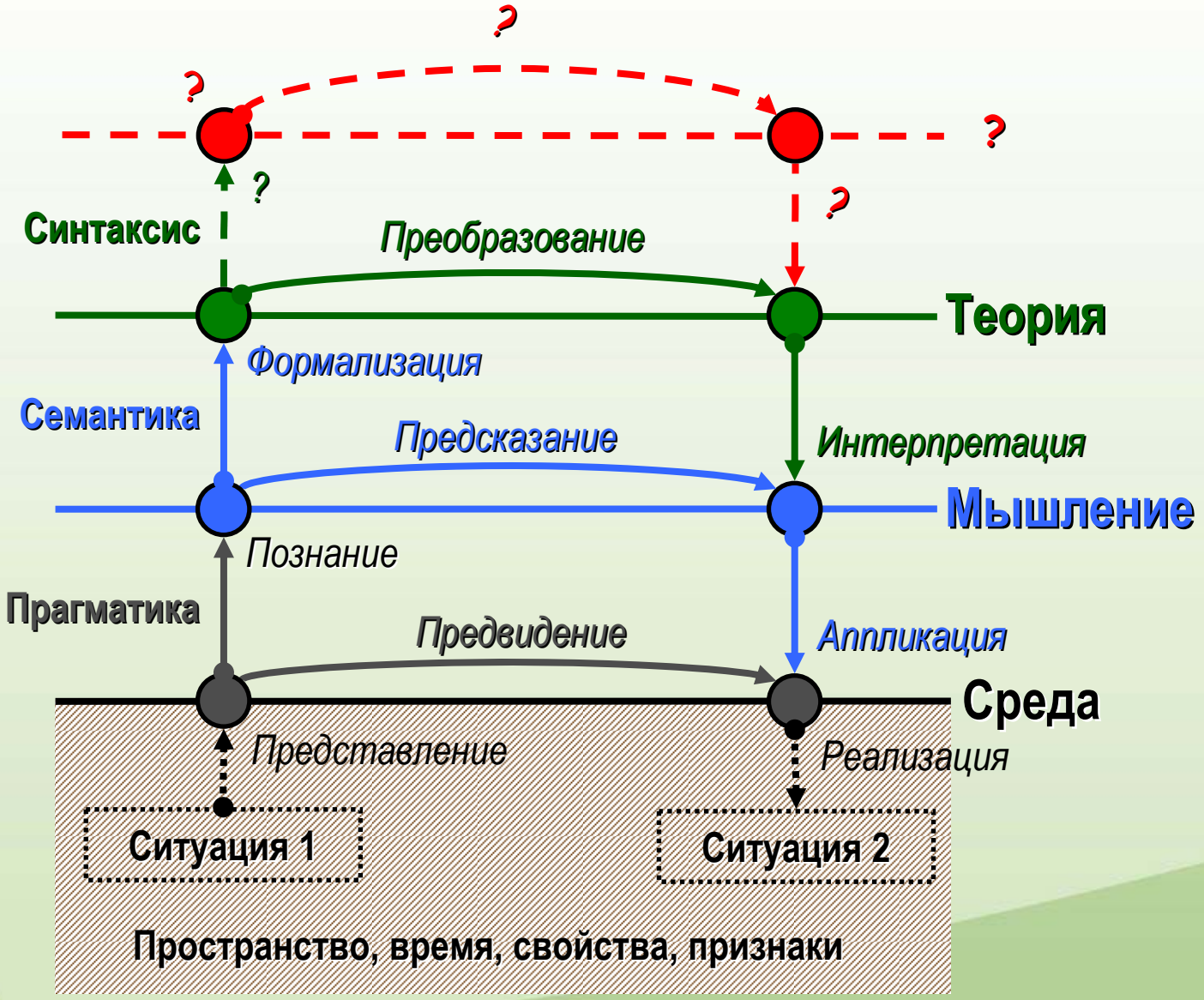
Формальный вывод

Абтрактное мышление

Наглядно-образное мышление

Наглядно-действенное мышление

Феномен формализации



Предметный анализ

- **Анализ** – формализация накопленных знаний относительно предметной области в точные данные и правила их интерпретации.

- **Методология**:
  - приемы выявления значимых сущностей;
  - формы представления знаний;
  - способы преобразования данных.

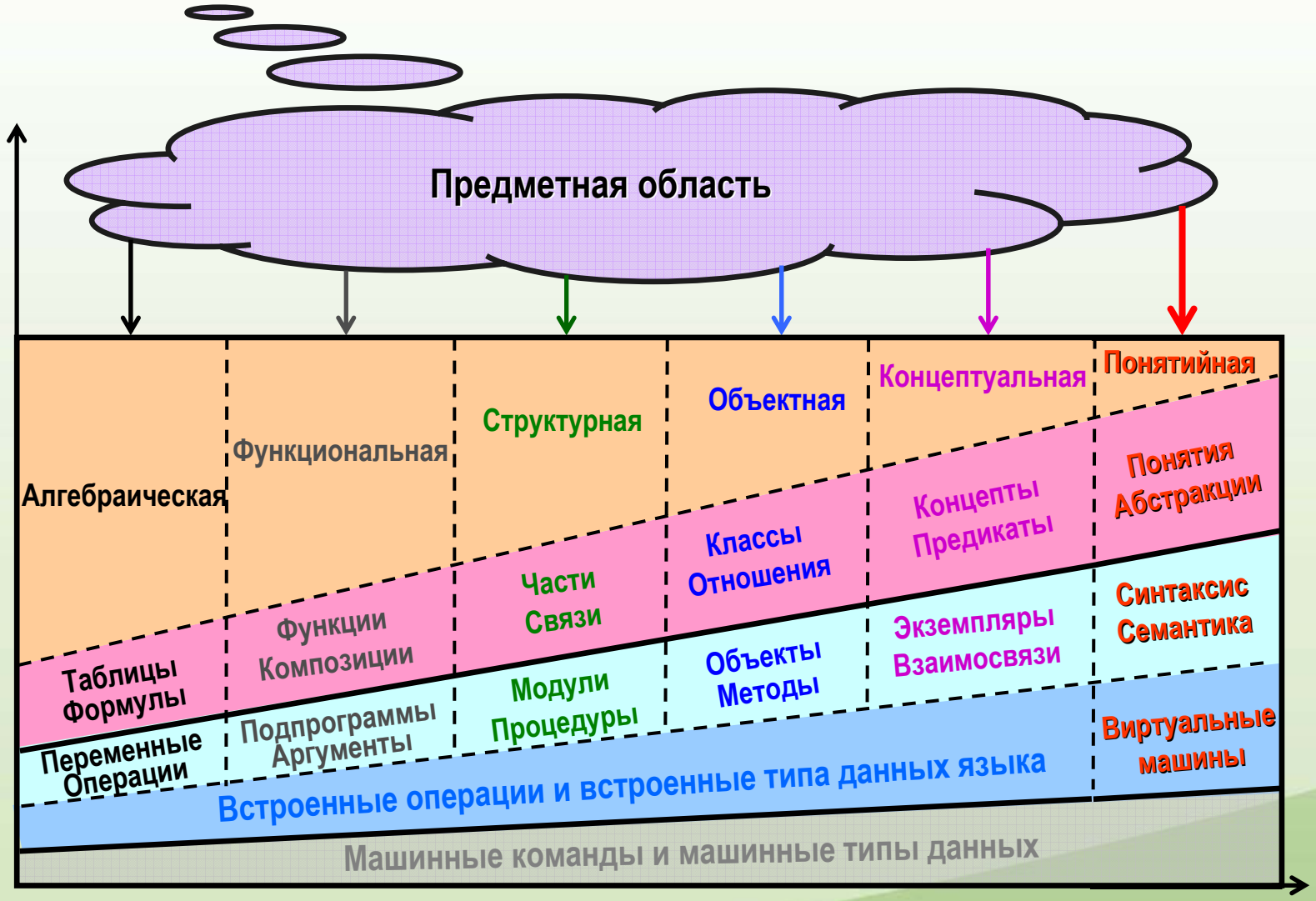
Методологии анализа

Семантичес-
кий разрыв

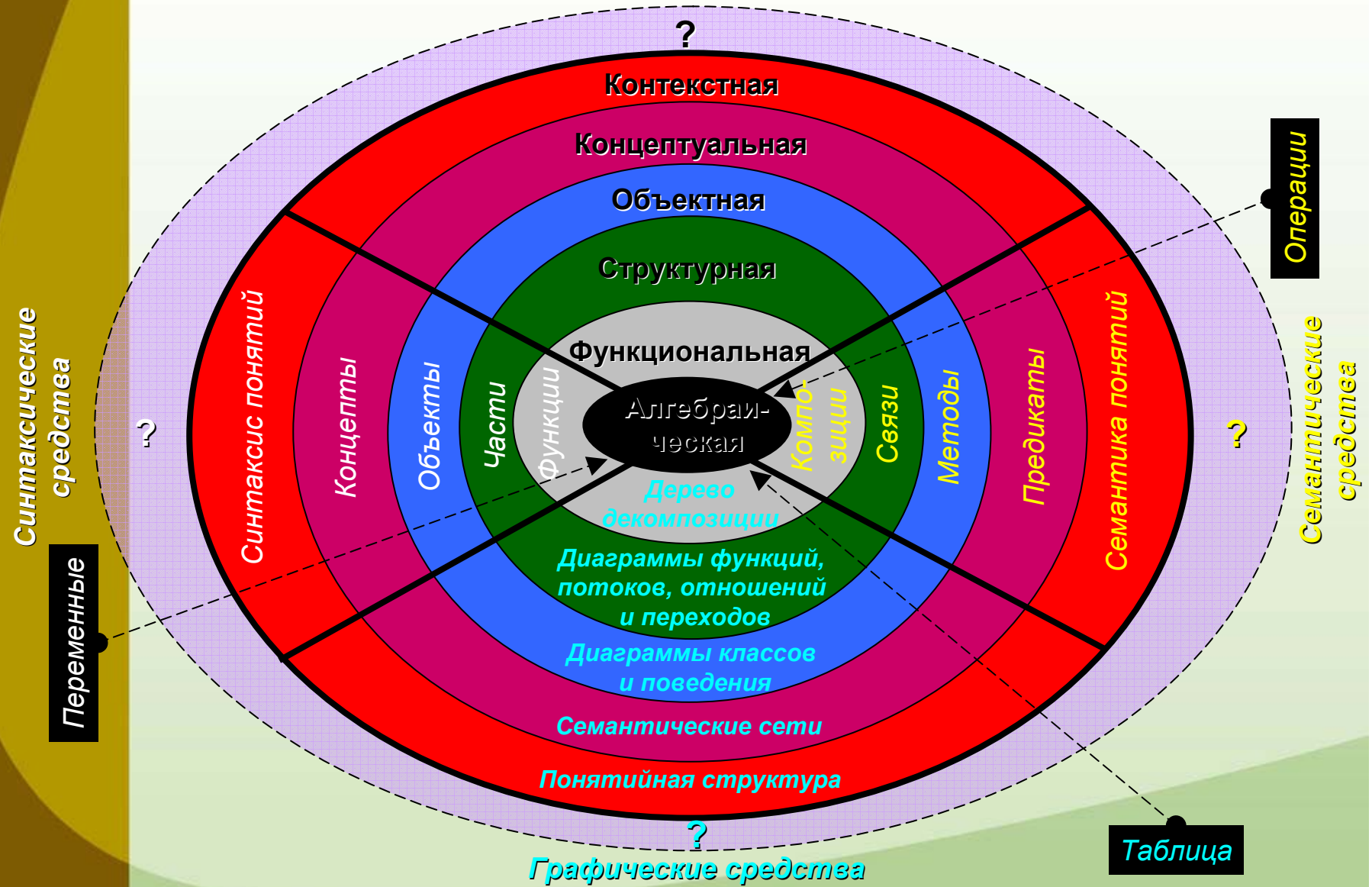
Методология

Технология

Платформа



Технологии моделирования



Не все задачи решаются

Решаются «любые» задачи

Возможно расширение языка

Создаются другие языки

Языки формализации

- **Специализированные**:
  - предметно-ориентированные (SQL, Postscript);
  - проблемно-ориентированные (GPSS, OKKAM),
- **Универсальные**:
  - императивные (C, C++, Java, Perl, Ruby);
  - декларативные (Prolog, Рефал),
- **Метаязыки**:
  - регулярные (Форт);
  - бесконтекстные (Рефал);
  - контекстные (разнесенный разбор),
- **Протоязыки**:
  - дедуктивные (YACC, Bison);
  - индуктивные (Protolanguage).

Синтаксис – отношения между знаками языка (Ч. Моррис)

Семантика – отношения между знаками и «мыслимым миром» (Р. Монтегю)

Синтаксис и семантика

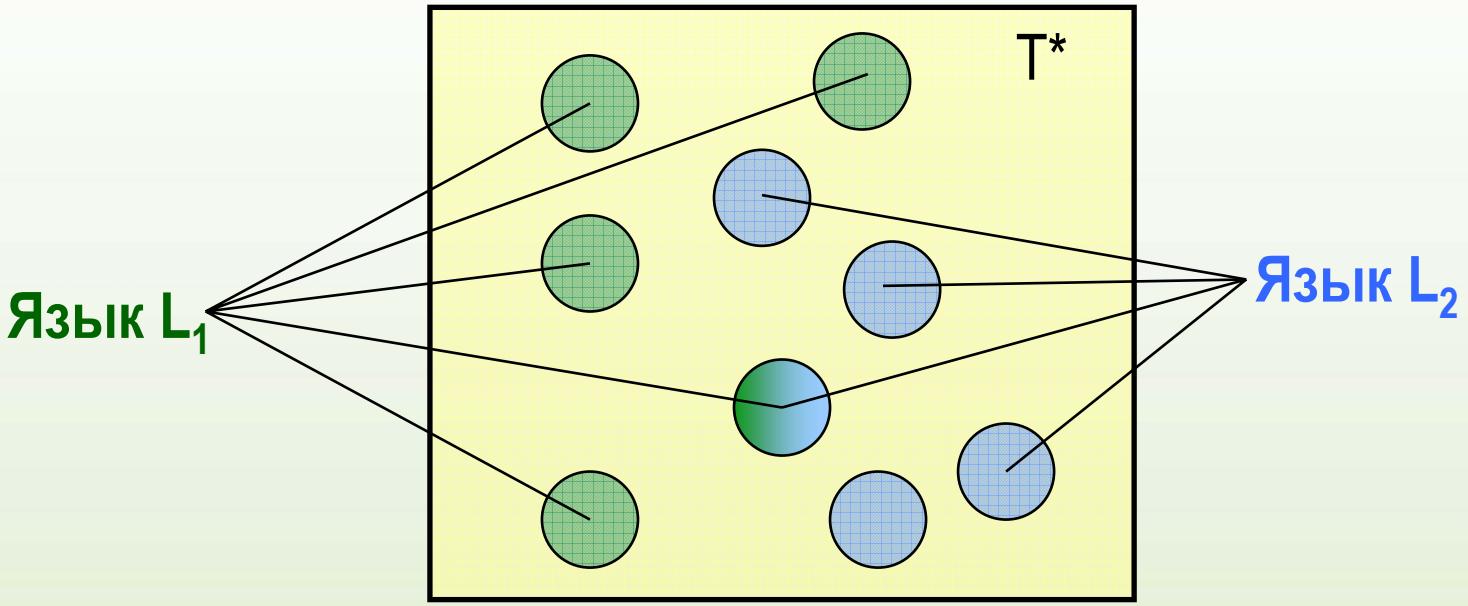
- **Проблема** – сложность описания семантики формальных языков:
  - на этапе формализации задачи;
  - при интерпретации полученных результатов.

Формальный язык – подмножество универсального множества строк T*

Грамматика – формальное средство для задания формальных языков

Грамматики:
– регулярные,
– контекстно-свободные,
– контекстные,
– произвольные.

Формальный язык



$$G(L) =< T, N, P, I >$$

$$G(L) =< \{0,1\}, \{I, J\}, P, I >,$$
$$P = \{I \to 0 \mid J, J \to 1 \mid J0 \mid J1\}$$

$$I \Rightarrow 0, \quad I \Rightarrow J \Rightarrow 1, \quad I \Rightarrow J \Rightarrow J0 \Rightarrow 10, \ \ldots$$

Эффективно разрешимые

Практически разрешимые

Теоретически разрешимые

Неразреши-мые, перечис-лимые

Неразреши-мые, непере-числимые

Классификация Хомского

- Типа 3 – Регулярные (конечно-автоматные)

$$P = \{A \to a \mid aB \ \| \ a \in T, \ A, B \in N\}$$

- Типа 2 – Контекстно-свободные (бесконтекстные)

$$P = \{A \to \alpha \ \| \ \alpha \in (T \cup N)^*\}$$

- Типа 1 – Контекстно-зависимые (контекстные)

$$P = \{\alpha\Omega\beta \to \alpha\omega\beta \ \| \ \Omega \in N, \ \alpha, \beta, \omega \in (T \cup N)^*\}$$

- Типа 0 – Произвольные (трансформационные)

$$P = \{\alpha \to \beta \ \| \ \alpha, \beta \in (T \cup N)^*\}$$

- Типа ? – Неперечислимые

$$P = \{?\}$$

Исчисление высказыва-ний – разрешимо, перечислимо

Пример: логические исчисления

- Тип 2
$$\alpha \to (\alpha \vee \beta)$$
$$\beta \to (\alpha \vee \beta)$$
$$\alpha \to (\beta \to \alpha)$$
$$\alpha \to (\beta \to (\alpha \wedge \beta))$$
- Тип 1
$$(\alpha \to \beta) \to ((\alpha \to \neg\beta) \to \neg\alpha)$$
$$(\alpha \to \beta) \to ((\alpha \to (\beta \to \gamma)) \to (\beta \to \gamma))$$
$$(\alpha \to \gamma) \to ((\beta \to \gamma) \to ((\alpha \vee \beta) \to \gamma))$$
- Тип 0
$$\neg\neg\alpha \to \alpha$$
$$(\alpha \wedge \beta) \to \alpha$$
$$(\alpha \wedge \beta) \to \beta$$

$$\alpha \to (\beta \to \alpha)$$
$$(\alpha \to (\beta \to \gamma)) \to ((\alpha \to \beta) \to (\alpha \to \gamma))$$
$$(\neg\alpha \to \neg\gamma) \to ((\neg\alpha \to \beta) \to \alpha)$$

Исчисление предикатов – неразрешимо, перечислимо

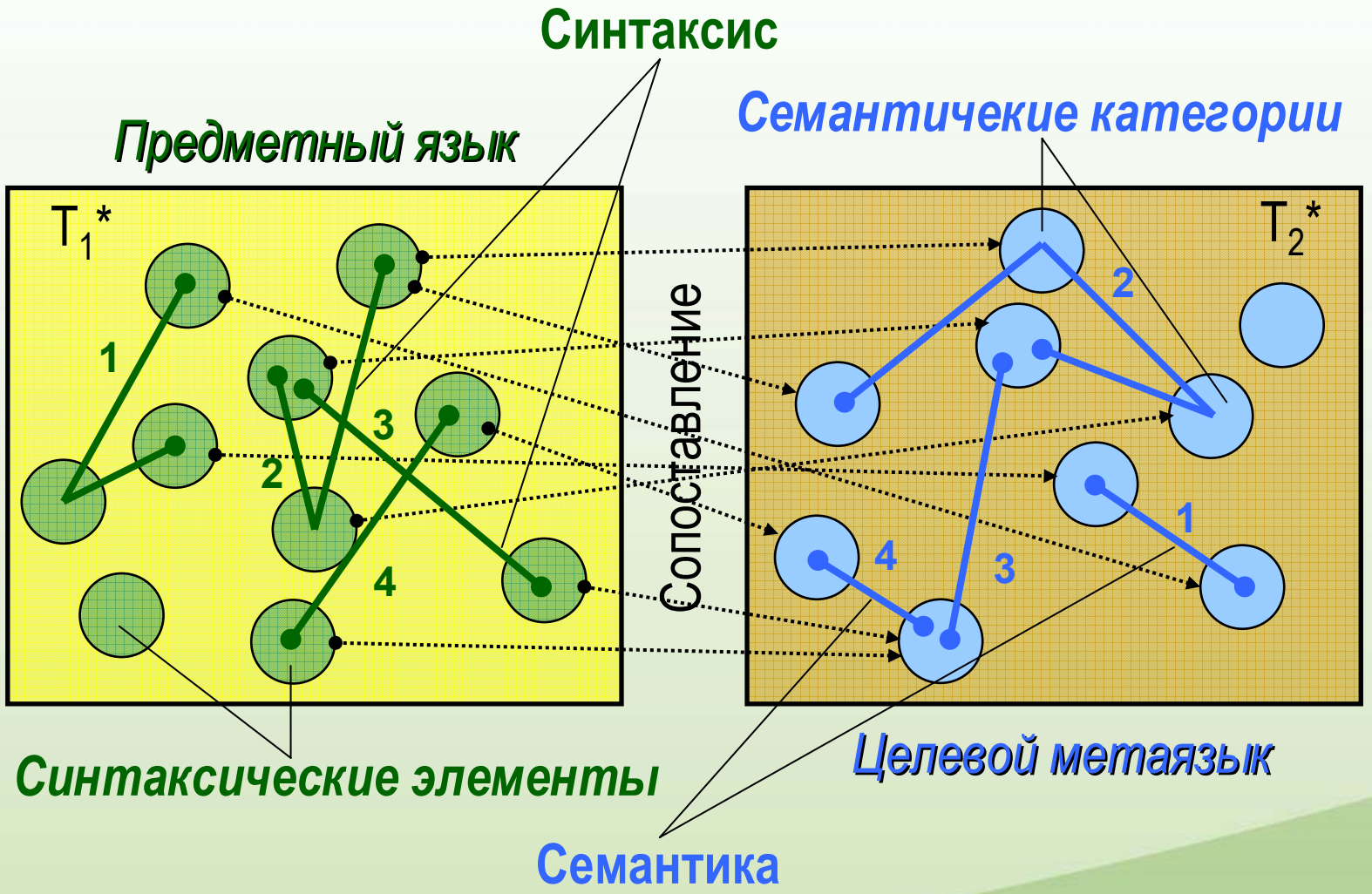
- Тип 1 $\varphi(A) \to \exists B(\varphi(B))$
- Тип 0 $\forall A(\varphi(A)) \to \varphi(B)$

Принцип композицион-ности: семантика целого строится из семантики частей (Г. Фреге)

Синтаксичес-кие элементы предметного языка – неделимые единицы синтаксиса

Семантичес-кие категории метаязыка – неделимые единицы смысла

Предметный язык и метаязык



Пример 1
Пример 2
Пример 3
Пример 4
Пример 5
Продолжение

Методы описания семантики

- **Непосредственный** – прямое сопоставление строк двух формальных языков
- **Аксиоматический** – сопоставление правилам вывода формул аксиоматической теории, задаваемой аксиомами и правилами вывода
- **Алгебраический** – сопоставление правилам вывода формул алгебраической теории, задаваемой множеством и операциями на нем
- **Денотационный** – сопоставление правилам вывода композиций функциональных денотатов
- **Операционный** – сопоставление правилам вывода последовательности команд интерпретатора
- **Индуктивный** – сопоставление правилам вывода строк самого определяемого языка

Непосредственная семантика

$$N = \{0,1,2,3,\ldots\}$$

0	$\{S \leftarrow 0\}$
1	$\{S \leftarrow 1\}$
10	$\{S \leftarrow 2\}$
11	$\{S \leftarrow 3\}$
100	$\{S \leftarrow 4\}$
101	$\{S \leftarrow 5\}$
...	...

Метод описания семантики

Метаязык – натуральные числа

Предметный язык – двоичные числа

101 — $S \leftarrow 5$

$S \leftarrow 5$

Аксиоматическая семантика

$\exists 0; \quad \forall a \exists a'; \quad \forall a(\neg(a' = 0)); \quad \forall a(a + 0 = a); \quad \ldots$

$I \to 0$	$\{S_I \leftarrow 0\}$
$I \to J$	$\{S_I \leftarrow S_J\}$
$J \to 1$	$\{S_J \leftarrow 0'\}$
$J \to J0$	$\{S_J \leftarrow (S_J)*(0'+0')\}$
$J \to J1$	$\{S_J \leftarrow (S_J)*(0'+0')+0'\}$

$I \to J \to J1 \to J0 \to 1$

$I \Rightarrow J \Rightarrow J1 \Rightarrow J01 \Rightarrow 101$

S_J	$\leftarrow$	$0'$
S_J	$\leftarrow$	$(S_J)*(0'+0')$
S_J	$\leftarrow$	$(S_J)*(0'+0')+0'$
S_I	$\leftarrow$	S_J

$$S_I \leftarrow ((0')*(0'+0'))*(0'+0'))+0'$$

Метод описания семантики

Метаязык – алгебра натуральных чисел

Предметный язык – двоичные числа

Алгебраическая семантика

$$A =< N,+,* >$$

$I \to 0$	$\{S_I \leftarrow 0\}$
$I \to J$	$\{S_I \leftarrow S_J\}$
$J \to 1$	$\{S_J \leftarrow 1\}$
$J \to J0$	$\{S_J \leftarrow (S_J)*2\}$
$J \to J1$	$\{S_J \leftarrow (S_J)*2+1\}$

$I \to J \xrightarrow{J \to J1} J \to J0 \xrightarrow{J \to 1}$

$I \Rightarrow J \Rightarrow J1 \Rightarrow J01 \Rightarrow 101$

S_J	$\leftarrow$	1
S_J	$\leftarrow$	$(1)*2$
S_J	$\leftarrow$	$((1)*2)*2+1$
S_I	$\to$	S_J

$$S_I \leftarrow ((1)*2)*2)+1$$

Денотационная семантика

$$F =< 0, 1, f, g >, \quad gx = x * 2, \quad fx = x + 1$$

$I \to 0$	$\{S_I \leftarrow 0\}$
$I \to J$	$\{S_I \leftarrow S_J\}$
$J \to 1$	$\{S_J \leftarrow 1\}$
$J \to J0$	$\{S_J \leftarrow (\lambda x.gx)S_J\}$
$J \to J1$	$\{S_J \leftarrow (\lambda x.fgx)S_J\}$

$I \to J \overset{J \to J1}{\to} J \to J0 \overset{J \to 1}{\to}$

$I \Rightarrow J \Rightarrow J1 \Rightarrow J01 \Rightarrow 101$

S_J	$\leftarrow$	1
S_J	$\leftarrow$	$(\lambda x.gx)1$
S_J	$\leftarrow$	$(\lambda x.fgx)(\lambda x.gx)1$
S_I	$\leftarrow$	S_J

$$S_I \leftarrow (\lambda x.fgx)(\lambda x.gx)1$$

Операционная семантика

$$K = \{mov\ A, B;\ shl\ A, B;\ inc\ A\}$$

$I \to 0$	$\{S_I \leftarrow mov\ V,0;\}$
$I \to J$	$\{S_I \leftarrow S_J\}$
$J \to 1$	$\{S_J \leftarrow mov\ V,1;\}$
$J \to J0$	$\{S_J \leftarrow S_J\ shl\ V,1;\}$
$J \to J1$	$\{S_J \leftarrow S_J\ shl\ V,1;\ inc\ V;\}$

$I \to J \quad J \to J1 \quad J \to J0 \quad J \to 1$

$I \Rightarrow J \Rightarrow J1 \Rightarrow J01 \Rightarrow 101$

S_J	$\leftarrow$	$mov\ V,1;$
S_J	$\leftarrow$	$S_J\ shl\ V,1;$
S_J	$\leftarrow$	$S_J\ shl\ V,1;\ inc\ V;$
S_I	$\leftarrow$	S_J

$$S_I \leftarrow mov\ V,1;\ shl\ V,1;\ shl\ V,1;\ inc\ V;$$

Особенности описаний

- Семантика формального языка – это интерпретация его строк с помощью другого языка, возможно формального
- Используется принцип композиционности Фреге, выражаемый в применении аппарата формальных грамматик
- Фиксируется конечное множество базовых семантических категорий, через которые описывается семантика языка
- Описание семантики осуществляется конструктивными средствами
- Определяется только однозначная семантика

Имеющиеся трудности

- Необходимость использования метаязыка более высокого порядка
- Необозримость получаемых описаний семантики известными методами
- Низкоуровневая интерпретируемость семантических описаний
- Потребность в пополнении множества базовых семантических категорий
- Отсутствие средств для выражения прагматических представлений
- Низкие выразительные возможности предметных языков и метаязыков

Основные последствия

Первичные

- Использование универсальных языков
- Относительная примитивность языков
- Труднопреодолимый семантический разрыв
- Высокая сложность моделирования
- Низкая надежность программных средств
- Недостаточное качество программ

Вторичные

- Увеличенные сроки разработки
- Неудовлетворенные ожидания заказчика
- Прямые экономические потери

Предметный язык

&

Целевой метаязык

↓

Квазиестест-венный язык

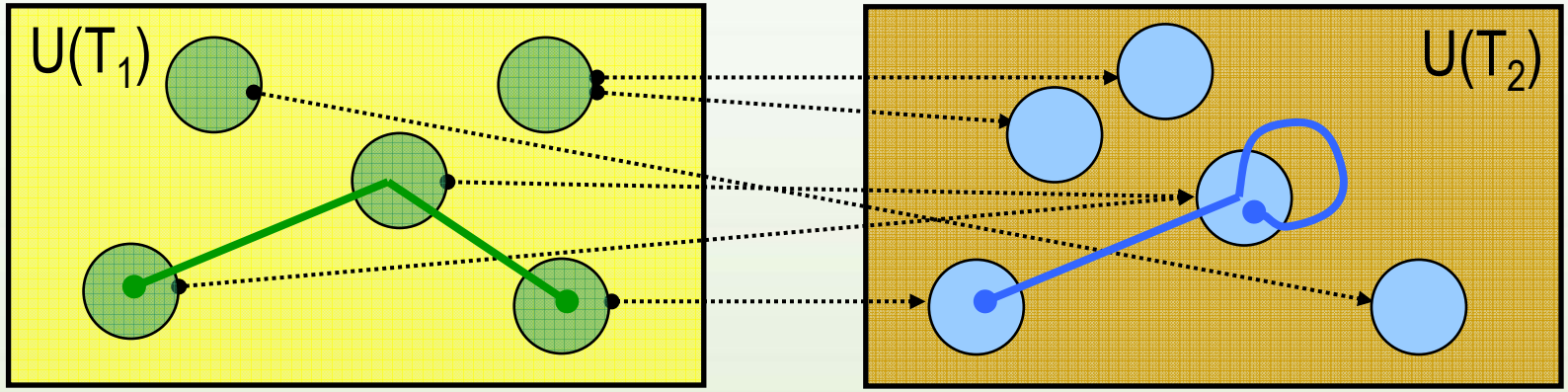
$\alpha \to \beta$
$\{\neg\alpha \vee \beta\}$

$\alpha \leftrightarrow \beta$
$\{ (\alpha \to \beta) \wedge (\beta \to \alpha) \}$

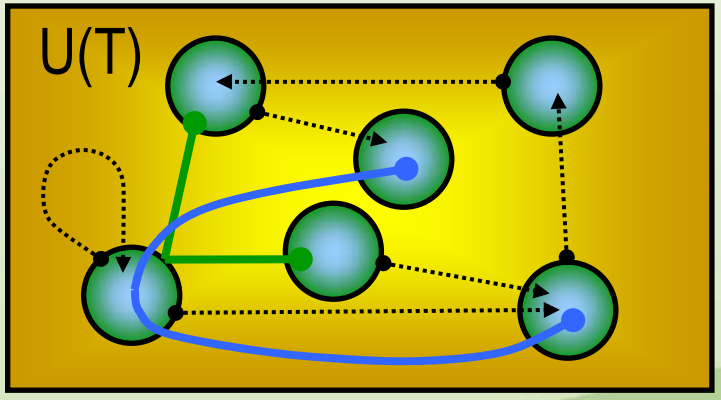
Квазиестественный язык

Предметный язык L_1

Целевой метаязык L_2

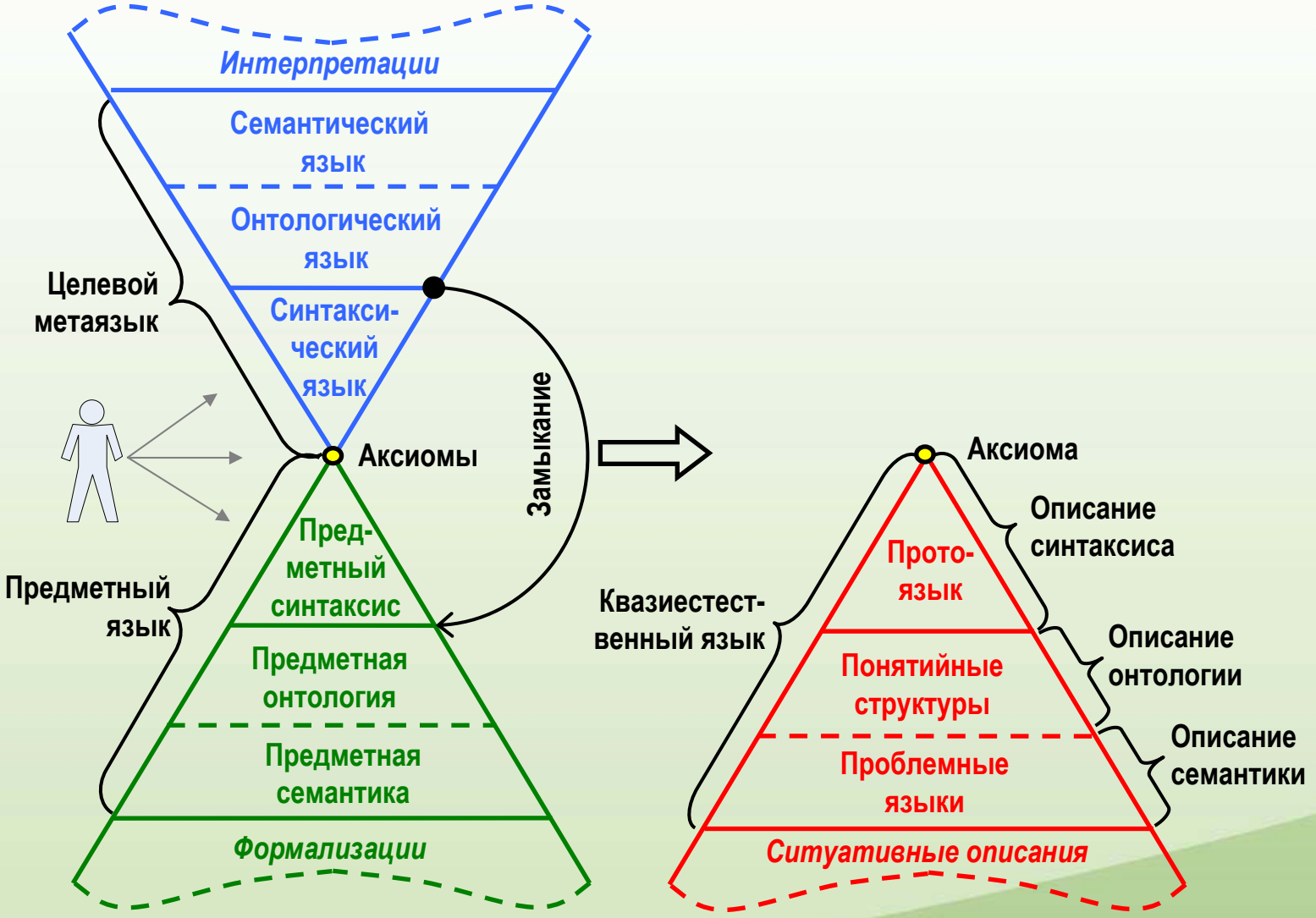


Квазиестественный язык L



- Синтаксический язык – описание синтаксиса предметных языков
- Онтологический язык – описание понятий предметных языков
- Семантический язык – описание семантики предметных языков

Семантическое замыкание



Протоязык

Протоязык:
– перечисление понятий;
– задание способов абстрагирования понятий;
– описание форм выражения понятий;
– описание синтаксиса форм выражения понятий.

model → essences [model]
essences → '(' ')' ***notion*** '(' ')' [intension]
intension → sentence [intension]
sentence → syntax '{' '}'
syntax → item [syntax]
item → **notion** | term
term → ''' [***terms***] '''

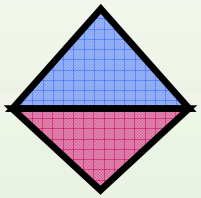
$$P = \{I \to 0 \mid J, J \to 1 \mid J0 \mid J1\}$$

() ***J*** ()
'1' { }
J '0' { }
J '1' { }
() ***I*** ()
J { }
'0' { }

Формализм понятия

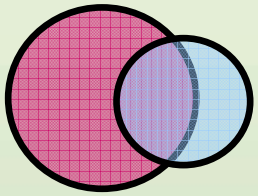
Сложный смысл

- Понятие $N = \begin{cases} \text{shm}\, N = (P^0, P^1, \ldots, P^{n-1}); \\ \text{int}\, N = \{(P_j^0, P_j^1, \ldots, P_j^{n-1}) \mid j = 0,1,\ldots\}; \\ \text{ext}\, N = \{N_0, N_1, \ldots, N_{m-1}\}. \end{cases}$



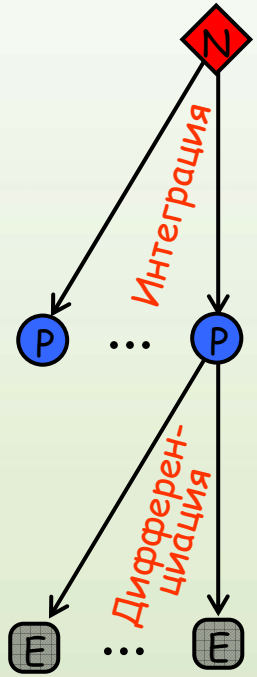
Простой смысл

- Признак $P = \begin{cases} \text{shm}\, P = (P); \\ \text{int}\, P = \{(P_0), (P_1), \ldots, (P_{u-1})\}; \\ \text{ext}\, P = \{E_0, E_1, \ldots, E_{u-1}\}. \end{cases}$



Атомарный смысл

- Сущность $E = \begin{cases} \text{shm}\, E = (E); \\ \text{int}\, E = \{(E)\}; \\ \text{ext}\, E = \{E\}. \end{cases}$



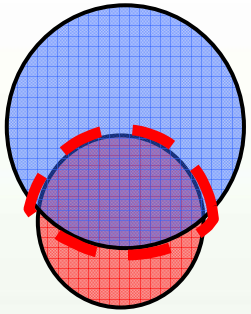
Абстракции понятий

Дифференциация понятий:
– типизация;
– обобщение.

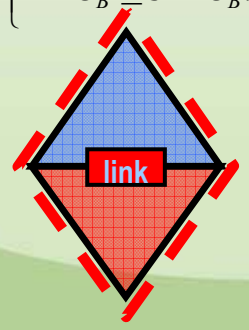
Интеграция понятий:
– агрегация;
– ассоциация.

Сильные абстракции:
– типизация;
– агрегация.

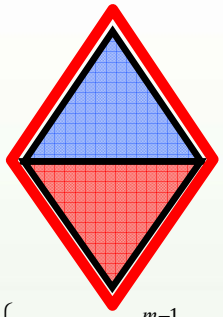
Слабые абстракции:
– обобщение;
– ассоциация.



$$\begin{cases} \operatorname{shm} C_G = \bigcap_{j=0}^{m-1} \operatorname{shm} C_j; \\ \operatorname{int} C_G \supseteq \bigcup_{j=0}^{m-1} \operatorname{int} C_j; \\ \operatorname{ext} C_G \supseteq \bigcup_{j=0}^{m-1} \operatorname{ext} C_j. \end{cases}$$

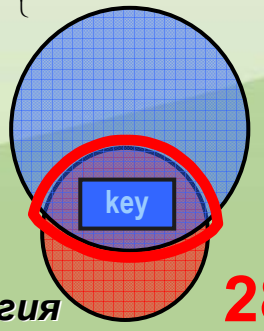


$$\begin{cases} \operatorname{shm} C_B = \bigcup_{j=0}^{m-1} \operatorname{shm} C_j; \\ \operatorname{int} C_B \subseteq \times_{j=0}^{m-1} \operatorname{int} C_j; \\ \operatorname{ext} C_B \subseteq \times_{j=0}^{m-1} \operatorname{ext} C_j; \\ \operatorname{link} C_B \subseteq \operatorname{shm} C_B. \end{cases}$$



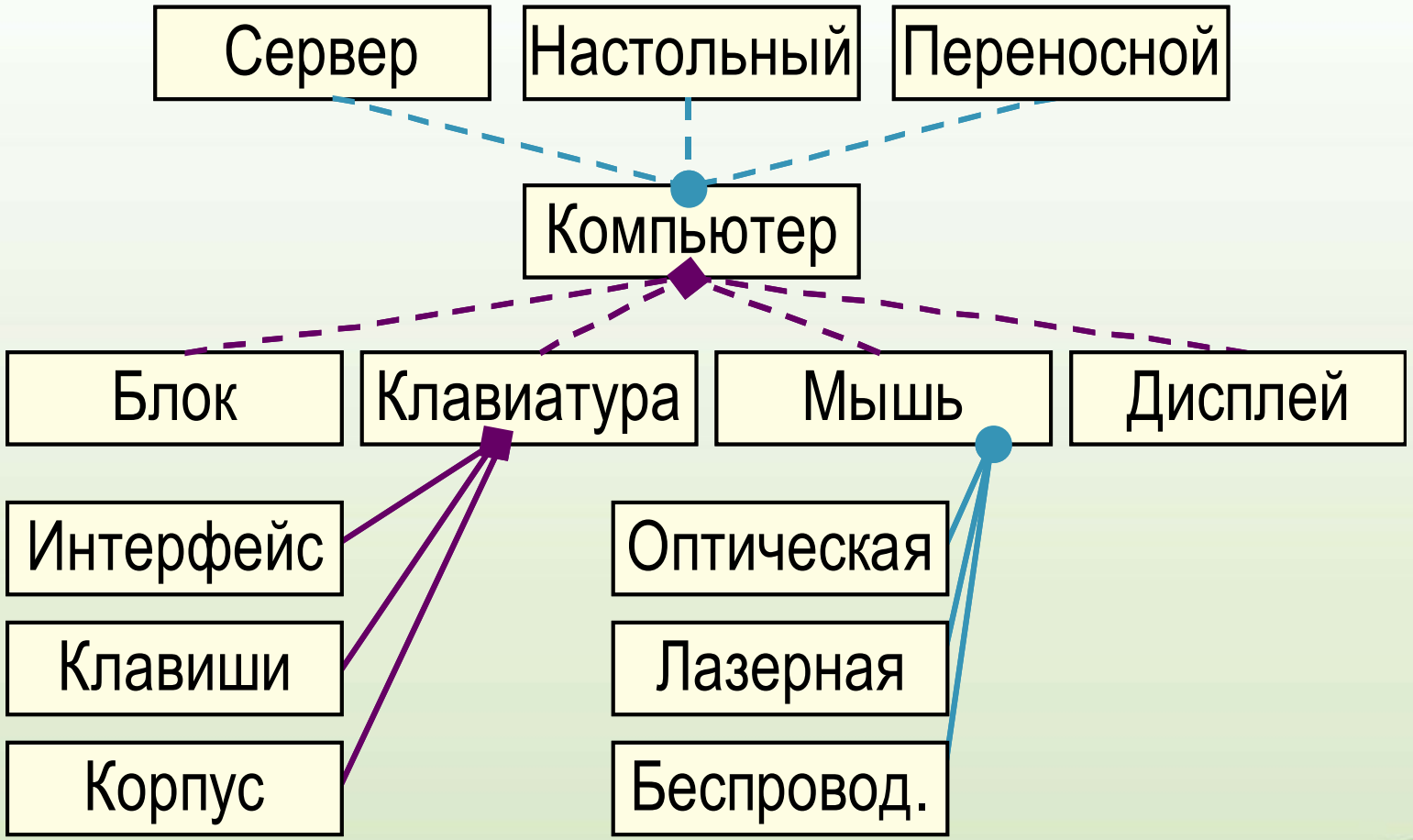
$$\begin{cases} \operatorname{shm} C_A = \bigcup_{j=0}^{m-1} \operatorname{shm} C_j; \\ \operatorname{int} C_A = \times_{j=0}^{m-1} \operatorname{int} C_j; \\ \operatorname{ext} C_A = \times_{j=0}^{m-1} \operatorname{ext} C_j. \end{cases}$$

$$\begin{cases} \operatorname{shm} C_T = \bigcap_{j=0}^{m-1} \operatorname{shm} C_j, \\ \operatorname{int} C_T = \bigcup_{j=0}^{m-1} \operatorname{int} C_j; \\ \operatorname{ext} C_T = \bigcup_{j=0}^{m-1} \operatorname{ext} C_j; \\ \operatorname{key} C_T \subseteq \operatorname{shm} C_T. \end{cases}$$



Проблематика – продажа компьютеров

Понятийная структура 1



Типизация –

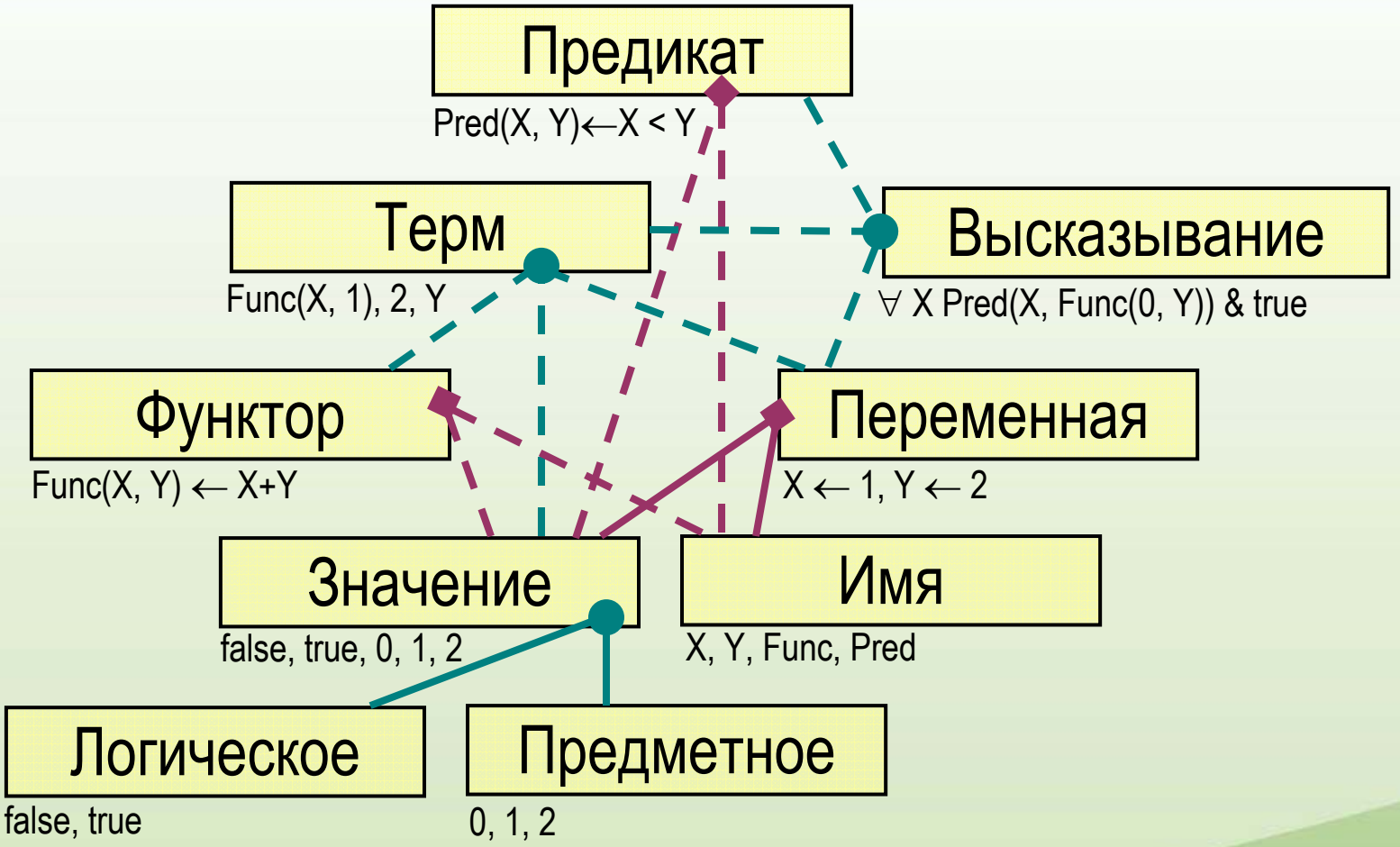
Обобщение –

Агрегация –

Ассоциация –

Проблематика – реализация логического языка.

Понятийная структура 2



Типизация –

Обобщение –

Агрегация –

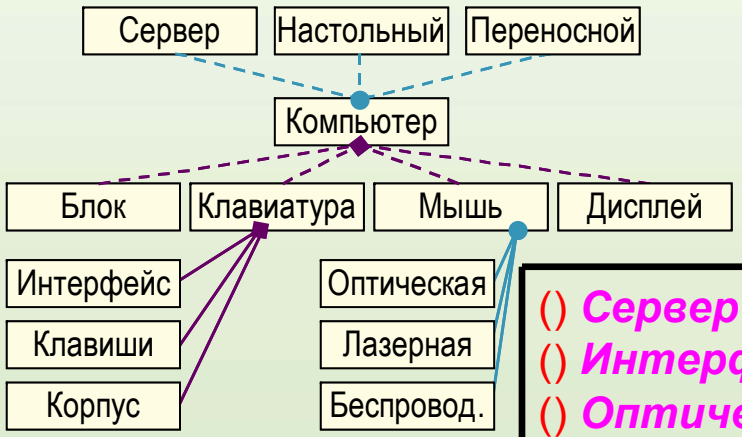
Ассоциация –

Онтология – совокупность понятий проблемной области и способы их абстрагирования.

Каждая предметная область характеризуется множеством онтологий, по числу проблемных областей.

Описание онтологии

model → essences [model]
essences → differenciation ***notion***
integration [intension]
differenciation → '(' [notions] ')'
integration → '(' [notions] ')'
notions → **notion** [notions]



() ***Сервер*** () () ***Настольный*** () () ***Переносной*** ()
() ***Интерфейс*** () () ***Корпус*** () () ***Клавиши*** ()
() ***Оптическая*** () () ***Лазерная*** () () ***Беспроводная*** ()
() ***Блок*** () () ***Дисплей*** ()
(**Оптическая Лазерная Беспроводная**) ***Мышь*** ()
() ***Клавиатура*** (**Интерфейс Клавиши Корпус**)
(**Сервер Настольный Переносной**) ***Компьютер***
(**Блок Клавитура Мышь Дисплей**)

Проблемный язык

Проблемный язык предназначен для описания семантики понятий в виде множества прагматик.

cognition → model [situation] [cognition]
model → essences [model]
essences → differenciation ***notion***
integration [intension]
intension → sentence [intension]
sentence → syntax semantic
syntax → item [syntax]
item → **notion** | '' [***terms***] ''
semantic → pragmatic [semantic]
pragmatic → [***aspect***] '**{**' [text] '**}**'
text → phrase [text]
phrase → **terms** | [**aspect**] '**{**' text '**}**'
situation → [**aspect**] '**<**' [text] '**>**'

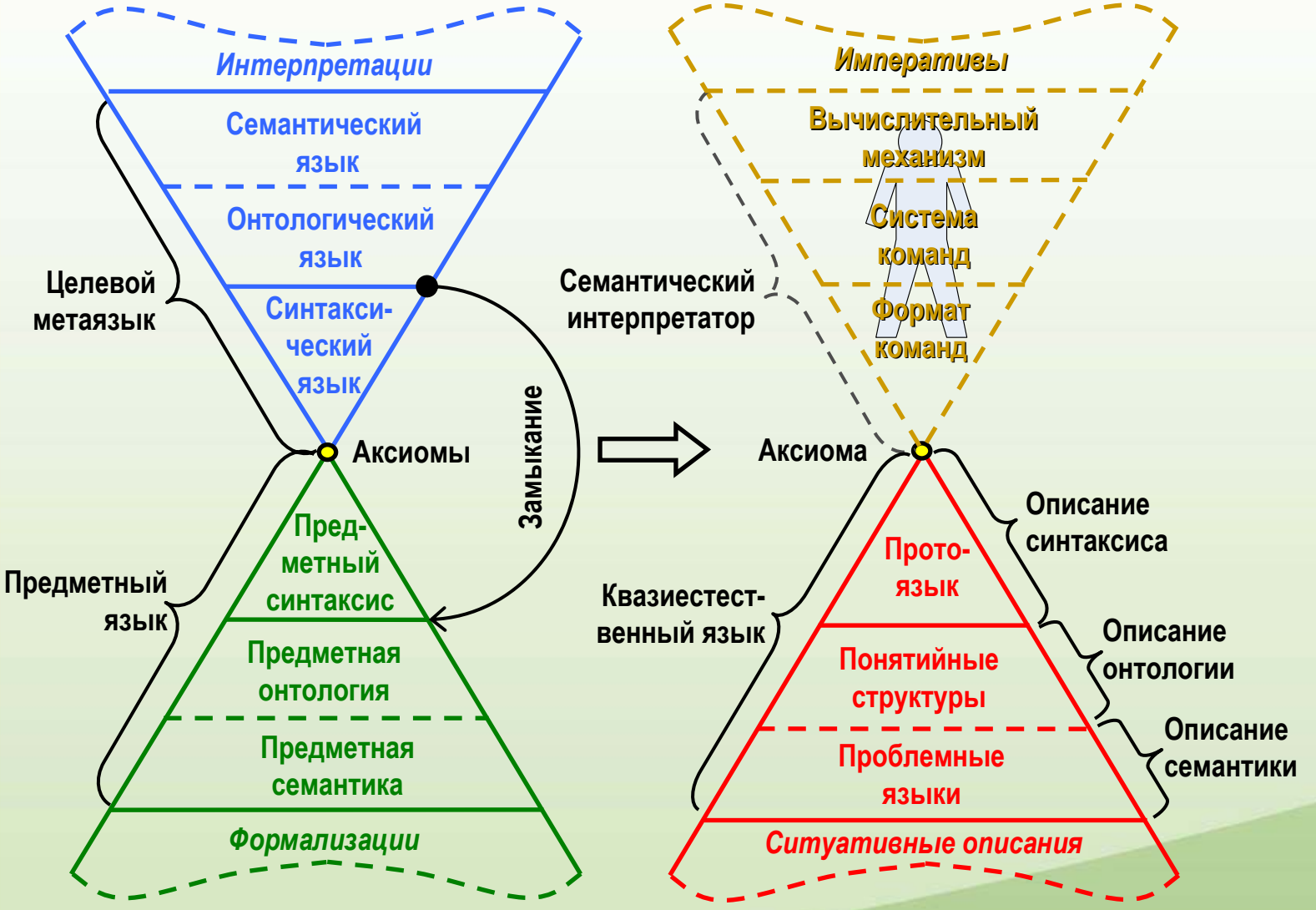
```
() Boolean ()
    'false' {…}
    'true' {…}
    '[A-Za-z][A-Za-z0-9]*' {…}
    '(' Boolean ')' { }
    'not' Boolean {…}
    Boolean 'and' Boolean {…}
    Boolean [a] 'or' Boolean [b]
        { not (not a and not b) }
< (not x or y) and z >
fuzzy < not (x or y) and z >
```

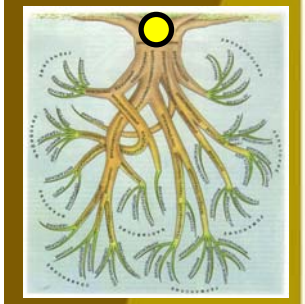
Семантический интерпретатор

Семантически замкнутый язык – язык, в котором выразима его собственная семантика (А. Тарский).

Проблема логической истины – противоречивые предложения исключаются из языка.

Естественные языки «размыкаются» в процессе обучения языку.





«Пустое» понятие – понятие, вмещающее в себя неспецифицируемые понятия проблемной области

Аксиома

«Пустые» квадратные скобки реализуют передачу семантическому интерпретатору того синтаксического элемента, после которого они применены

```
() ()
    '#' "[0-9A-F][0-9A-F]" [ ] { }
() Boolean ()
    'not' Boolean
        { #58 #F7 #D0 #50 }
    Boolean 'or' Boolean
        { #58 #5A #0B #C2 #50 }
```

```
() ()
    '`' "[0-9A-Za-z\s.,]+" [ ] '`' { }
() Boolean ()
    'not' Boolean
        { `pop eax` `not eax`
          `push eax` }
    Boolean 'or' Boolean
        { `pop eax` `pop edx`
        `or eax, edx` `push eax` }
```

Семантическая индукция – метод «размыкающего» описания семантики замкнутых языков.

Семантическая индукция

Определение семантических категорий по мере необходимости, в процессе определения проблемного языка и средствами этого языка:

– ***база индукции*** – первичные категории, которые непосредственно реализуются (понимаются) семантичеким интерпретатором и декларируются перед использованием;

– ***предположение индукции*** – все ранее определенные семантические категории, выраженые на проблемном языке;

– ***индуктивный переход*** – описание семантикии нового или уже существующего правила вывода на уже определенном до этого проблемном подязыке;

– ***заключение индукции*** – определение новой или доопределение существующей семантической категории.

```
'not' Boolean { #58 #F7 #D0 #50 }
Boolean 'or' Boolean { #58 #5A #0B #C2 #50 }
Boolean [a] 'imp' Boolean [b] { not a or b }
```

C′ = 0

x′ = 1

(f+g)′ = f ′ + g′

(fg)′ =f′g+fg′

Управляемый перевод

```
() String ()
        "'.*'" { … }
        String '&' String { … }
() Expression ()
        "[0-9]+" [n] { n } dif { '0' }
        'x' { 'x' } dif { '1' }
        '(' Expression ')' [e]
             { '(' & e & ')' } dif { '(' & dif { e } & ')' }
        '-' Expression [e]
             { '-' & e } dif { '-' & dif { e } }
        Expression [e1] '*' Expression [e2]
             { e1 & '*' & e2 }
             dif { '('& e1 & '*' & dif { e2 } & '+' & dif { e1 } & '*' & e2 & ')' }
        Expression [e1] "+|-" [o] Expression [e2]
             { e1 & o & e2 } dif { '(' & dif { e1 } & o & dif { e2 } & ')' }
```

---

dif < -x*x+5*(x–3) > → -(x*1+1*x)+(5*(1-0)+0*(x-3))

equ< dif{ -x*x+5*(x–3) } > → -2*x+5

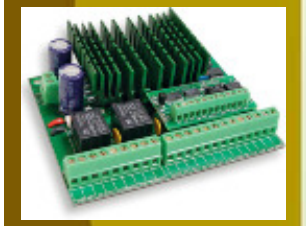
Атрибутные грамматики

Наследумые атрибуты вычисляются на текущем шаге разбора

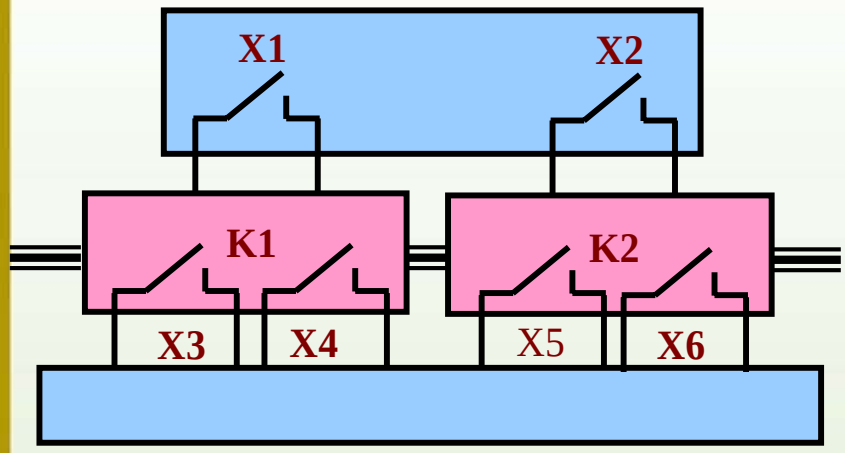
Синтезируемые атрибуты вычисляются при выполнении последующих шагов разбора

```
(Number) Integer ()
        "[0-9]" [d]
            { d }
        Integer [i] "[0-9]" [d]
            { i * 10 + d }
(Float) Fraction ()
        "[0-9]" [d]
            { d }
        "[0-9]" [d] Fraction [f]
            { d + f / 10 }
() Fixed (Float)
        Integer [i]
            { i }
        Integer [i] '.' Fraction [f]
            { i + f / 10 }
```

```
() Expr (Number)
(Number) Const ()
        "[0-9]+" [c]
            { c }
(Const) Prim ()
        '(' Expr [e] ')'
            { e }
(Prim) Mul ()
        Prim [p1] '*' Prim [p2]
            { p1 * p2 }
(Mul) Add ()
        Mul [p1] '+' Mul [p2]
            { p1 + p2 }
() Expr ()
        Add [a]
            { a }
```



Управление клапанами 1



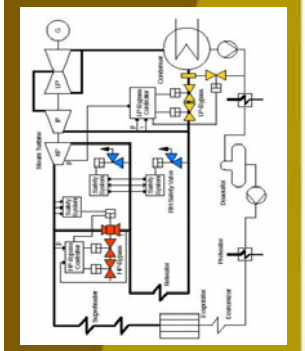
Необходимо реализовать устройство управления, которое при нажатии кнопки X1 подает управляющий сигнал на открытие клапана K1. После его открытия по сигналу открытого положения X4 снимается управляющий сигнал с клапана K1 и начинает открываться клапан K2.

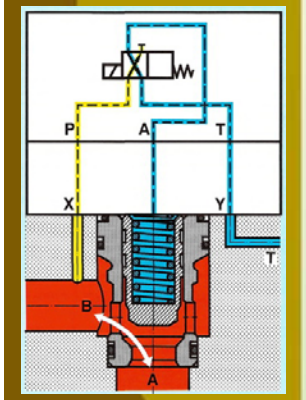
После его открытия по сигналу открытого положения X6 снимается управляющий сигнал с клапана K2, после чего устройство переходит в устойчивое состояние с открытыми клапанами.

В свою очередь, при нажатии кнопки X2 подает управляющий сигнал на закрытие клапана K2. После его закрытия по сигналу закрытого положения X5 снимается управляющий сигнал с клапана K2 и начинает закрываться клапан K1.

После его закрытия по сигналу закрытого положения X3 снимается управляющий сигнал с клапана K1, после чего устройство вновь переходит в устойчивое состояние с закрытыми клапанами.

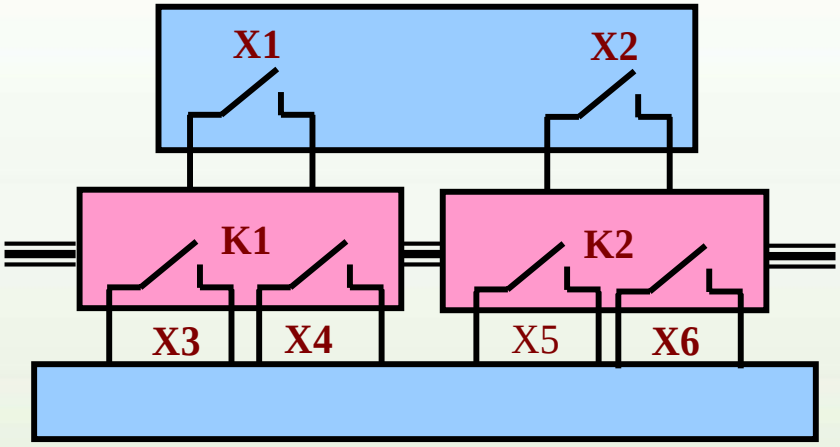
Управление клапанами 2





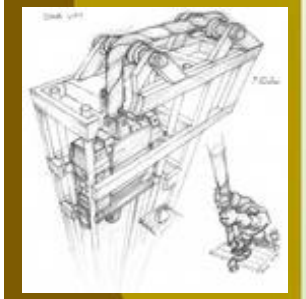

```
1   ( ) ( )
2       '`' "[^`]+" [ ] '`' {}
3   ( ) Сигнал ( )
4       'X1' [`push 0201h;`]{} 'X2' [`push 0202h;`]{}      'X3' [`push 0203h;`]{}
5       'X4' [`push 0204h;`]{} 'X5' [`push 0205h;`]{}      'X6' [`push 0206h;`]{}
6   ( ) Команда ( )
7       'Открытие K1' [`push 0301h;`]{}   'Закрытие K1' [`push 0302h;`]{}
8       'Открытие K2' [`push 0303h;`]{}   'Закрытие K2' [`push 0304h;`]{}
9   ( ) ( )
10      ', ' {`mov eax, 100; L1: inc eax; jnz L1;`}          '. ' { , , }
11      'Выдать' Команда {`pop ebx; mov [ebx], 1;`}
12      'Снять' Команда {`pop ebx; mov [ebx], 0;`}
13      'Ожидать' Сигнал {`pop ebx; L2: mov eax, [ebx]; test eax, 0; je L2;`}
14      'Начало' [`L3: lea eax, L3; push eax;`] '' 'Повторить' [`pop eax; jmp eax;`] {}
```

Управление клапанами 3



< Начало.
Ожидать X1, Выдать Открытие K1.
Ожидать X4, Снять Открытие K1, Выдать Открытие K2.
Ожидать X6, Снять Открытие K2.
Ожидать X2, Выдать Закрытие K2.
Ожидать X5, Снять Закрытие K2, Выдать Закрытие K1.
Ожидать X3, Снять Закрытие K1.
Повторить. >

Управление лифтом 1



1 Ожидание вызова

Если вызовов нет, то ожидать вызов (1). Если вызов со второго этажа, то перейти к открытию дверей (2), иначе – на принятие решения о движении (4).

2 Открытие дверей

Открыть дверь. Если дверь открылась, то перейти на принятие решения о движении (4), иначе повторить открытие двери (2).

3 Закрытие дверей

Закрыть дверь. Если дверь не закрылась, то повторить закрытие двери (3), иначе перейти к определению направления движения (4).

4 Направление движения

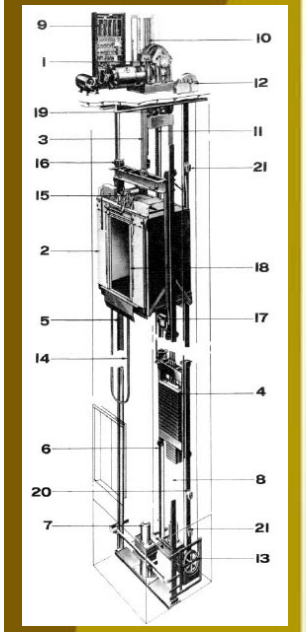
Продолжение движения. Если лифт двигался вверх (вниз) и имеются вызовы на движение вверх (вниз), то начать движение вверх (5) (вниз 6). ***Изменение направления***. Если вызовов на движение вверх (вниз) нет, но есть вызовы на движение вниз (вверх), то начать движение вниз (6) (вверх 5). ***Возврат в начало***. Если вызовов нет и при этом лифт выше (ниже) второго этажа, то начать движение вниз (6) (вверх 5), иначе перейти в состояние ожидания (1).

5 Движение вверх

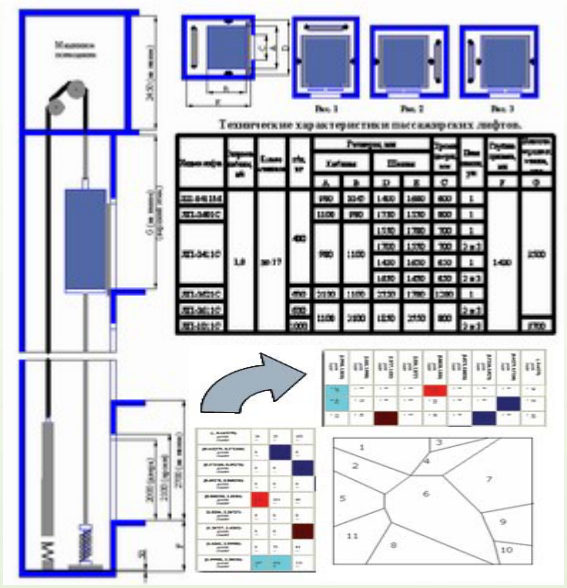
Начать движение вверх. Если при проходе этажа имеется вызов на движение вверх, то остановить лифт и открыть дверь (2), иначе продолжить движение вверх (5).

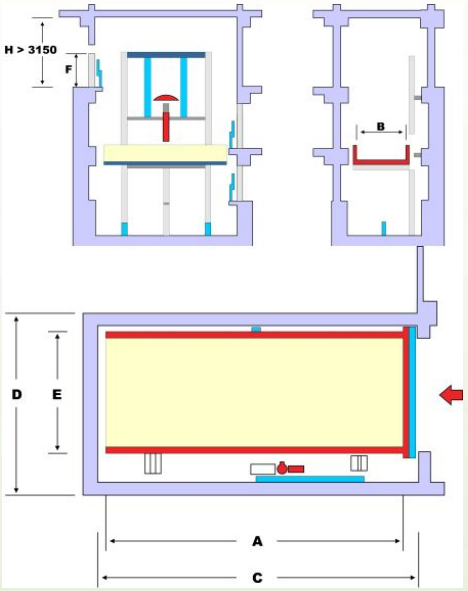
6 Движение вниз

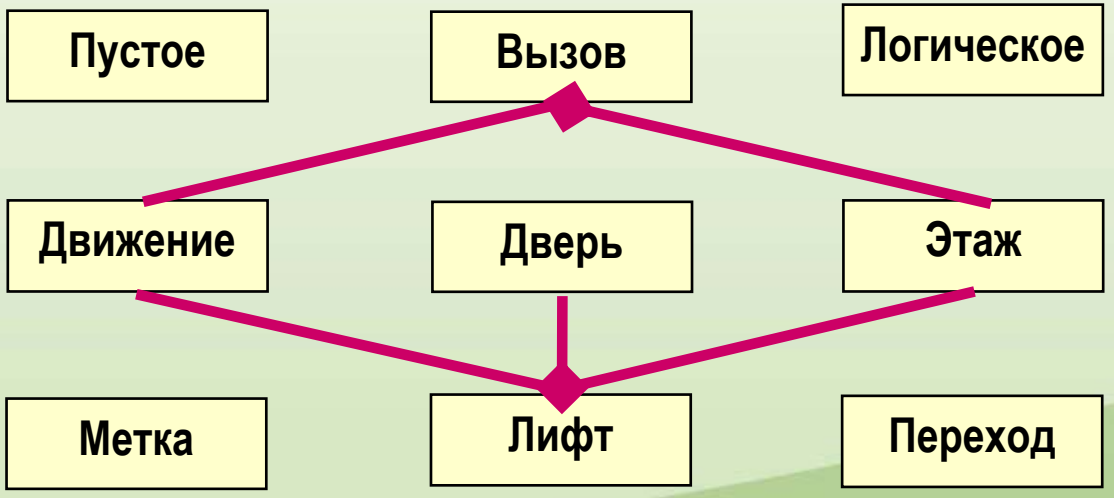
Начать движение вниз. Если при проходе этажа имеется вызов для движения вниз, то остановить лифт и открыть дверь (2), иначе продолжить движение вниз (6).



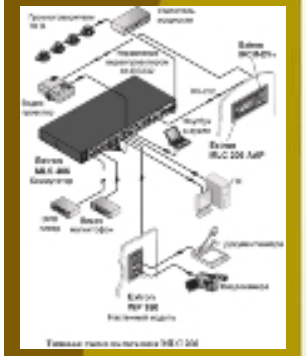
Управление лифтом 2







Управление лифтом 3



```
( ) Движение ( )
    "[Дд]вижение" {…}
( ) Дверь ( )
    "[Дд]верь" {…}
( ) Этаж ( )
    "этажу?" "[0-9]" {…}
    "этажу?" 'вызова' "|вверх|вниз" {…}
    "этажу?" 'лифта' {…}
( ) Вызов (Этаж Движение)
    "вызова?" {…}
( ) Лифт (Этаж Движение Дверь)
    'лифт' {…}
( ) Метка ( )
    "[А-Яа-я][А-Яа-я0-9]*" {…}
( ) Переход ( )
    Метка {…}
( ) Логическое ( )
    Этаж "выше|ниже|равен" Этаж {…
    Вызов "вверх|вниз|нет" {…}
    Дверь "открыта|закрыта" {…}
    'не ' Логическое { }
( ) ( )
    '. ' {…}
    Метка ': ' {…}
    Движение "вверх|вниз|останов" {…}
    Дверь "открыть|закрыть" {…}
    'Если ' Логическое ', ' 'то ' Переход {…}
    'Если ' Логическое ', ' 'то ' Переход ', '
        'иначе ' Переход {…}
```

Ожидание:
- Если вызова нет, то Ожидание.
- Если этаж вызова равен этажу 2, то Открыть, иначе Решение.

Открыть:
- Дверь открыть.
- Если дверь открыта, то Решение, иначе Открыть.

Закрыть:
- Дверь закрыть.
- Если дверь закрыта, то Решение, иначе Закрыть.

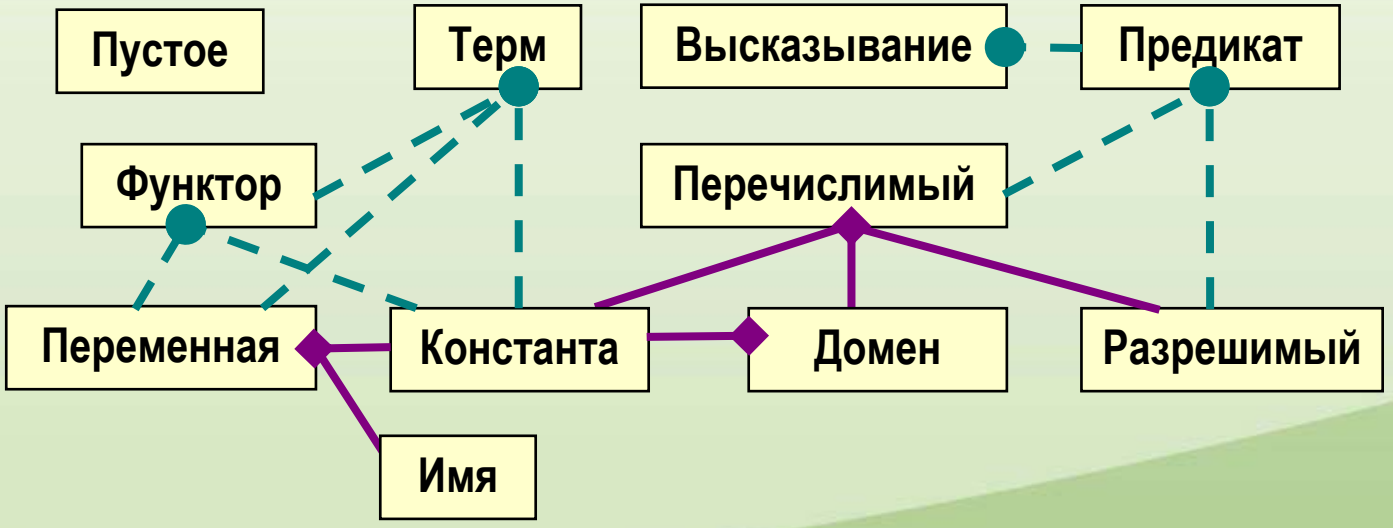
Решение:
- Если лифт вверх и вызов вверх, то Вверх.
- Если лифт вниз и Вызов вниз, то Вниз.
- Если не вызов вверх и вызов вниз, то Вниз.
- Если не вызов вниз и вызов вверх, то Вверх.
- Если вызова нет и этаж лифта выше этажа 2, то Вниз.
- Если вызова нет и этаж лифта ниже этажа 2, то Вверх, иначе Ожидание.

Вверх:
- Движение вверх.
- Если этаж лифта равен этажу вызова вверх, то Открыть, иначе Вверх.

Вниз:
- Движение вниз.
- Если этаж лифта равен этажу вызова вниз, то Открыть, иначе Вниз.

Логический язык 1

1. $\exists x S(x) \;\&\; \exists x P(x) \;\&\; \exists x M(x)$ - пос.
2. $\exists x \neg S(x) \;\&\; \exists x \neg P(x) \;\&\; \exists x \neg M(x)$ - пос.
3. $\forall x (M(x) \supset \neg P(x))$ - пос.
4. $\forall x (S(x) \supset \neg\neg M(x))$ - пос.
5. $\exists x S(x)$ - $\&_и$, 1
6. $S(x)$ - $\exists_и$, 5, x -абс. огр.
7. $S(x) \supset \neg\neg M(x)$ - $\forall_и$, 4
8. $\neg\neg M(x)$ - $\supset_и$, 7, 6
9. $M(x)$ - $\neg_и$, 8
10. $M(x) \supset \neg P(x)$ - $\forall_и$, 3
11. $\neg P(x)$ - $\supset_и$, 10, 9
12. $S(x) \;\&\; \neg P(x)$ - $\&_в$, 6, 11
13. $\exists x (S(x) \supset \neg P(x))$ - $\exists_в$, 12



Онтология проблемной области и синтаксис логического языка

Логический язык 2

```
( ) ( )
        ';'
        'if' Proposition 'then' '' 'else' '' 'end‘ { }
        'while' Proposition 'do' '' 'end‘ { }
        'do' '' 'while' Proposition 'end‘ { }
        'for' Variable '=' Term ',' Term '' 'end‘ { }
        'Variable' Name '=' Term ',‘ { }
        Variable '=' Term ',‘ { }
        Domain '=' Term ',‘ { }
        'Predicate' "[1-9][0-9]" Name ',‘ { }
        Enumerable '(' Term ',' ')' '←' Proposition { }
        '∆' Name '(' Name ',' ')' '←' Proposition { }
        '∇' Variable ',' '∈' Enumerable '' '.‘ { }
( ) Name ( )
        "[A-ZA-Я][a-za-я0-9]*" { }
( ) Constant ( )
        "-?[1-9][0-9]*" { }
        Domain '(' ')' { }
        Domain '(' Term ')' { }
        Domain '[' ']' { }
        'Arity' Enumerable { }
( ) Variable (Name Constant)
        Domain '[' Term ']' { }
(Constant Variable) Functor ( )
        '(' Functor ')' { }
        '|' Functor '|' { }
        '-' Functor { }
        Functor '%' Functor { }
        Functor '/' Functor { }
        Functor '*' Functor { }
        Functor '-' Functor { }
        Functor '+' Functor { }
        Functor '?' Functor ':' Functor { }
(Constant Variable Functor) Term ( )
( ) Solvable ( )
        'false' { }
        'true' { }
        Term '=' Term { }
        Term '>' Term { }
        Term '<' Term { }
        Enumerable '(' Term ',' ')' { }
( ) Domain (Constant)
        'Domain' Enumerable { }
() Enumerable (Domain Constant Solvable)
(Solvable Enumerable) Predicate ( )
(Predicate) Proposition ( )
        '(' Proposition ')' { }
        '¬' Proposition { }
        Proposition '&' Proposition { }
        Proposition '∨' Proposition { }
        Proposition '→' Proposition { }
        Proposition '↔' Proposition { }
        "∃|∀" Variable ',' '∈' Enumerable Proposition { }
```

Логический язык 3

Определение выполнимости предиката

(…) – описание абстракции

{ … } – описание интерпретации

[…] – опиание формализации

< … > – описание ситуации

```
( Solvable Enumerable ) Predicate ( )
…
    Enumerable `enum` '(' Term `ters` ', ' [ … #\ Обработка списка термов \# ] ')'
    {
    #\ Объявление временных переменных
        Variable kor, ter
    #\ Вычислить максимальный номер кортежа
        kor = Domain enum () / ters
    #\ Просмотреть все кортежи предиката
        while
            kor > 0
        do
        #\ Сравнить поэлементно список термов и текущий кортеж предиката
            ter = ters
            while
                ter > 0
            do
                if
                    operand( ter ) = Domain enum ( kor * ters – ter )
                then
                    ter = ter – 1
                else
                    ter = -1
                end
            end
            kor = ( ter = -1 ) ? 0 : kor – 1
        end
    #\ Возврат разрешимого предиката как логического значения
        ( ter = -1 )
    } …
< … >
```

Задача: пересечение прямой с областью

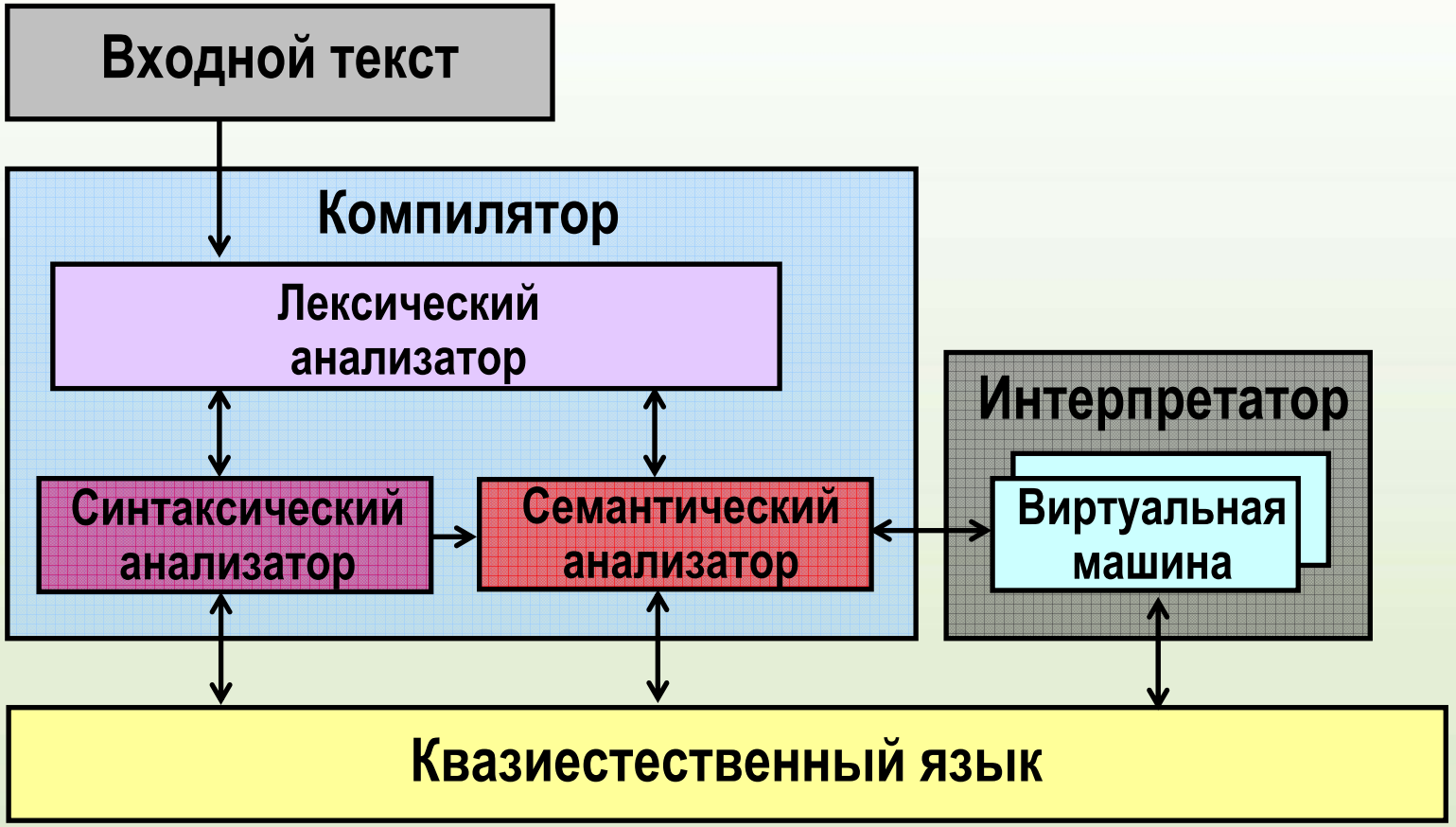
Решение: P(2, 2) P(3, 3)

Ситуация: прямая ax + y = 0 при a=–1 пересекает заданную область R выше гиперболы, определенной уравнением x * y = b при b=2, в точках (2, 2) и (3, 3).

Логический язык 4

```
< #\ Декларация предметных констант
Constant a = -1, b = 2
#\ Объявление предметных переменных
Variable x, y
#\ Объявление местности и имен предикатов
Predicate 2 P, 2 R, 3 Q
#\ Перечислимый предикат (ввод фактов)
R(6, 0) ←;
R(5, 0) ←; R(5, 1) ←;
R(4, 0) ←; R(4, 1) ←; R(4, 2) ←;
R(3, 0) ←; R(3, 1) ←; R(3, 2) ←; R(3, 3) ←;
R(4, 0) ←; R(2, 1) ←; R(2, 2) ←; R(2, 3) ←; R(2, 4) ←;
          R(1, 1) ←; R(1, 2) ←; R(1, 3) ←; R(1, 4) ←; R(1, 5) ←;
                    R(0, 2) ←; R(0, 3) ←; R(0, 4) ←; R(0, 5) ←; R(0, 6) ←;
#\ Разрешимый предикат (правило вывода логического языка)
Δ Q(u, v, w) ← w * u + v = 0
#\ Нахождение перечислимого предиката (задание логической цели)
∇ x, y ∈ R   P(x, y) ← x * y > b & Q(x, y, a).
#\ Вывод решения
for x, y ∈ P
    message 'P(' & x & ', ' & y & ')'
end >
```

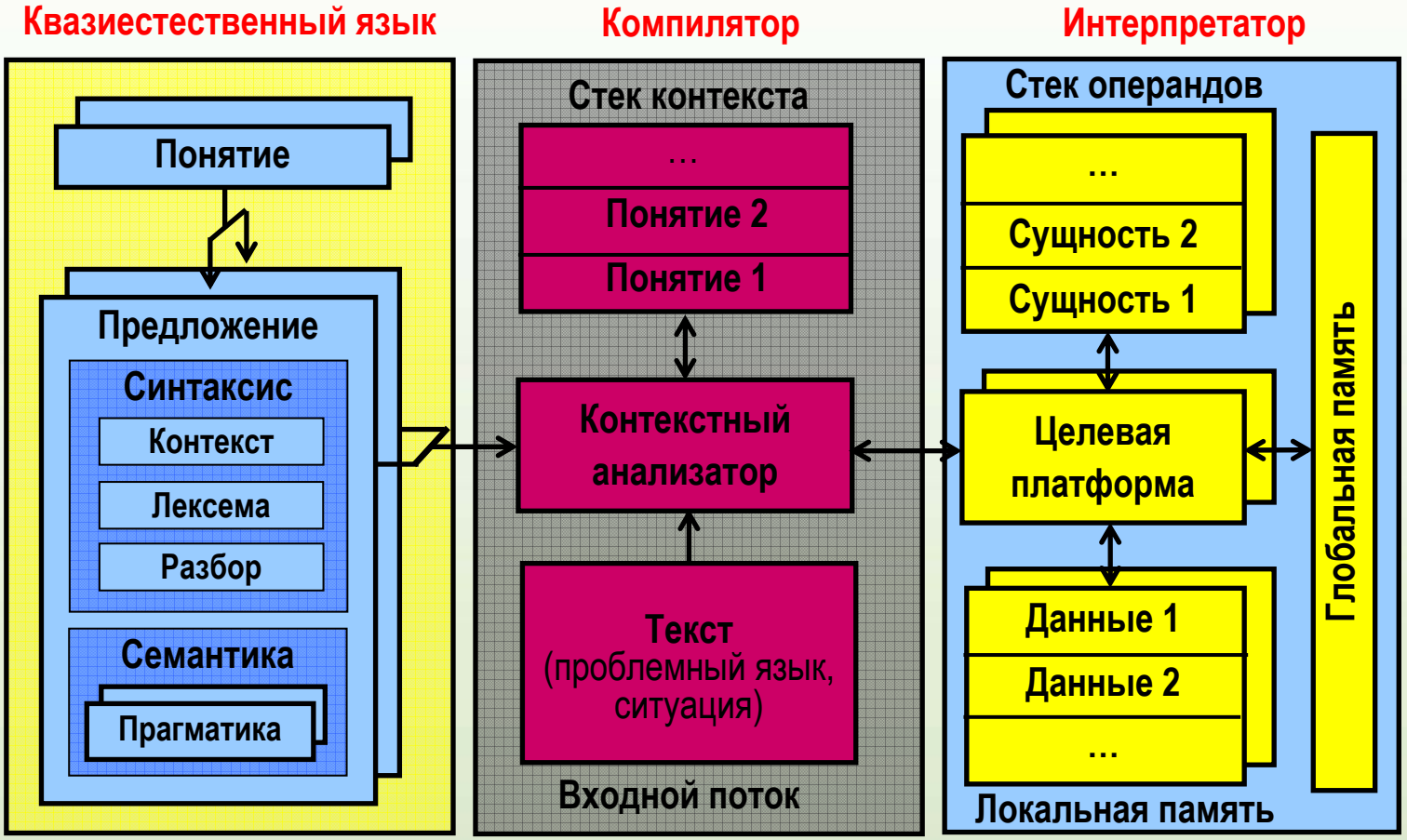
Контекстная технология



Проблемные языки:
- определяемые;
- дополняемые;
- библиотечные;
- общедоступные.

Интерпретаторы:
- контроллеры, процессоры;
- абстрактные машины;
- операционные системы;
- системы программирования.

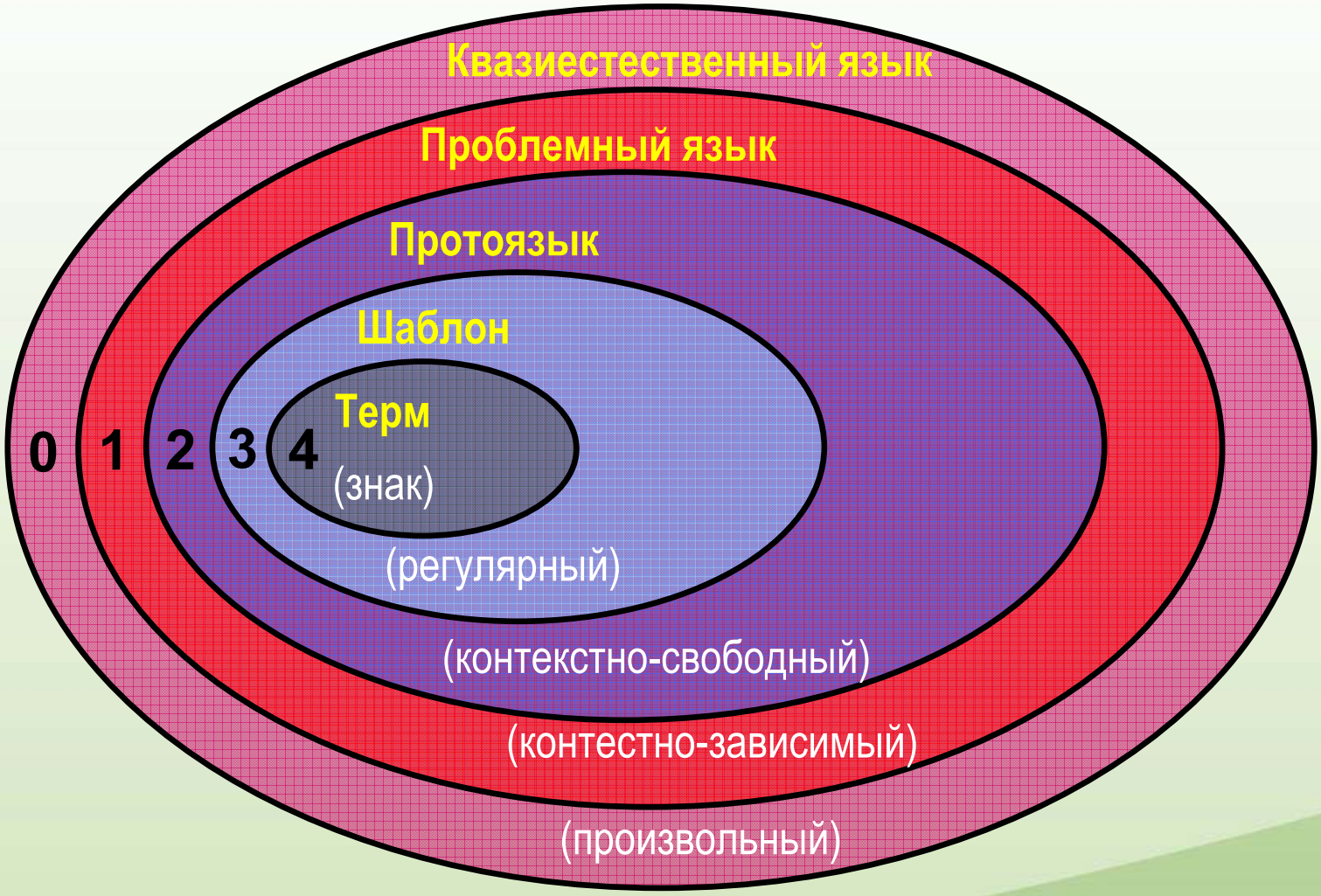
Разнесенный разбор



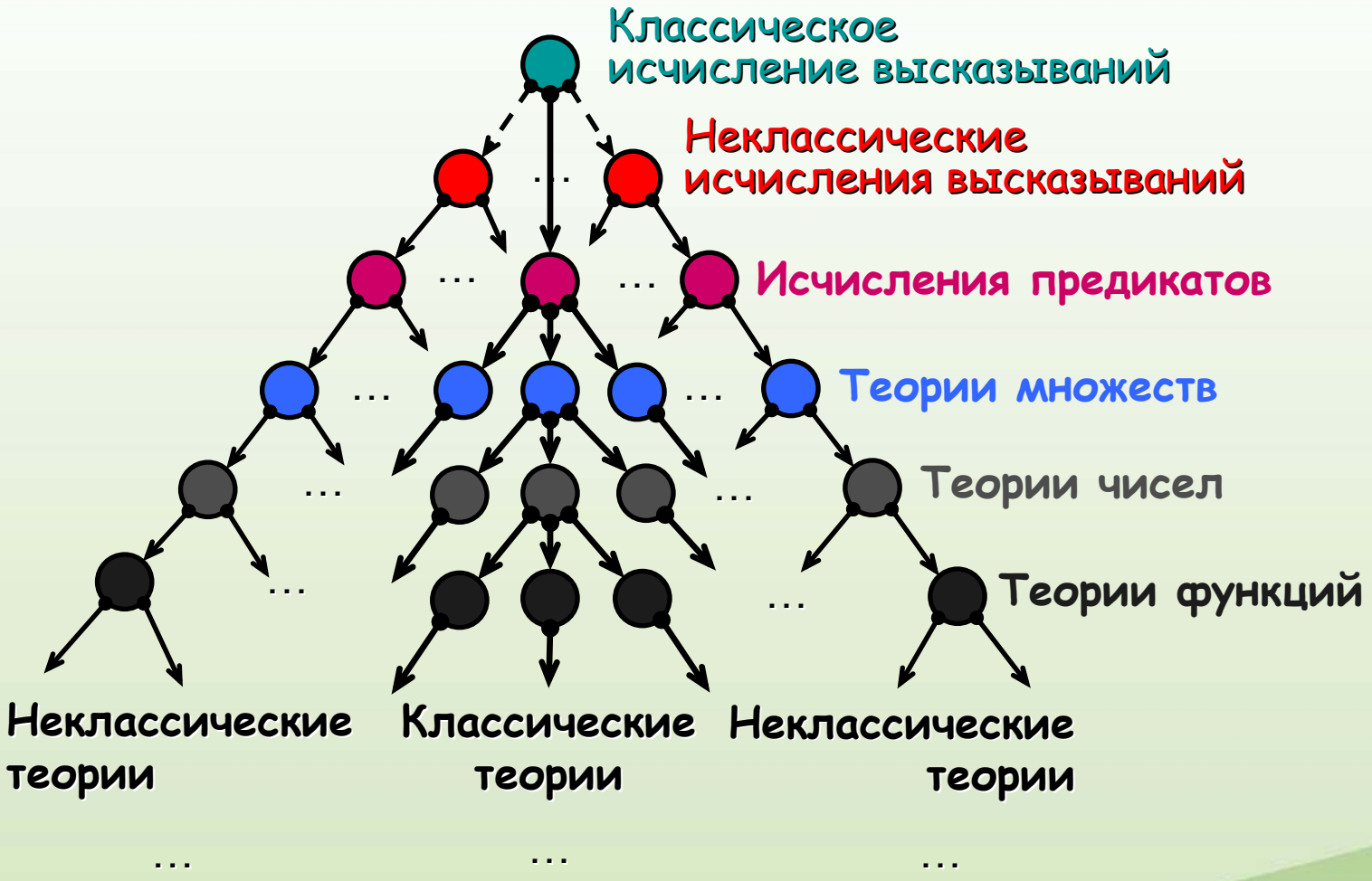
Boolean [a] '*imp*' [b] **Boolean** { not a or b }

Контекст | Лексема | Разбор | Прагматика

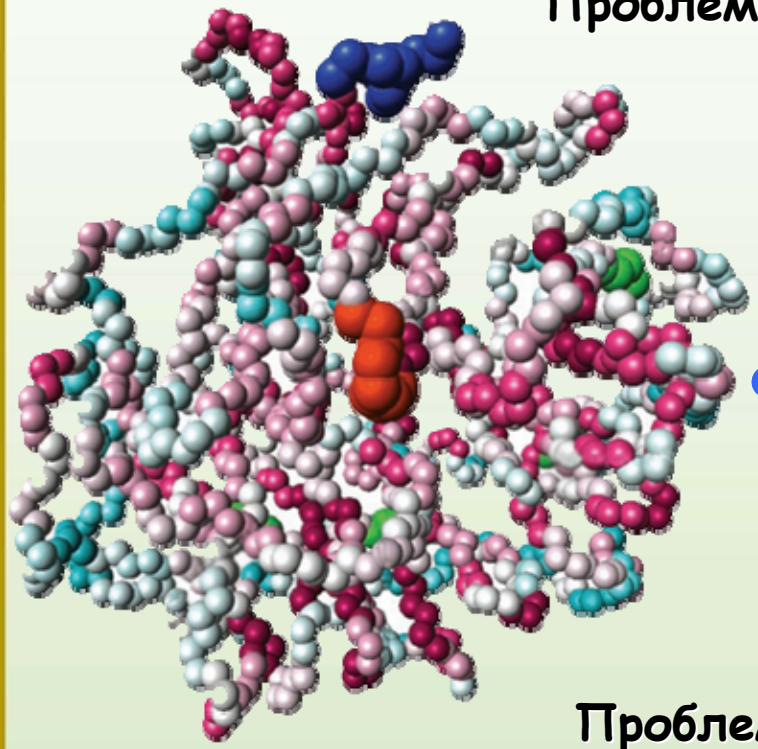
Иерархия языков



Решетка теорий



Корпус теорий

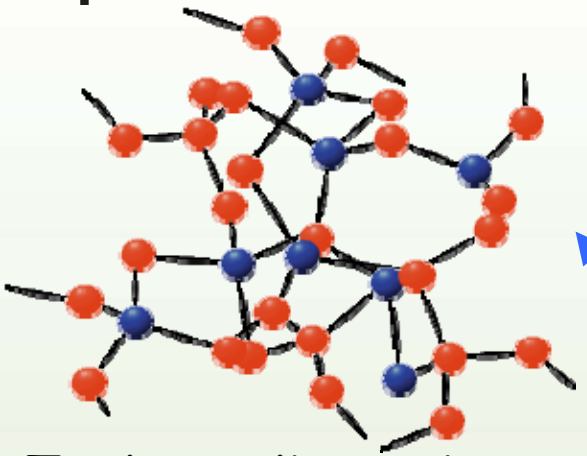


Естественный язык

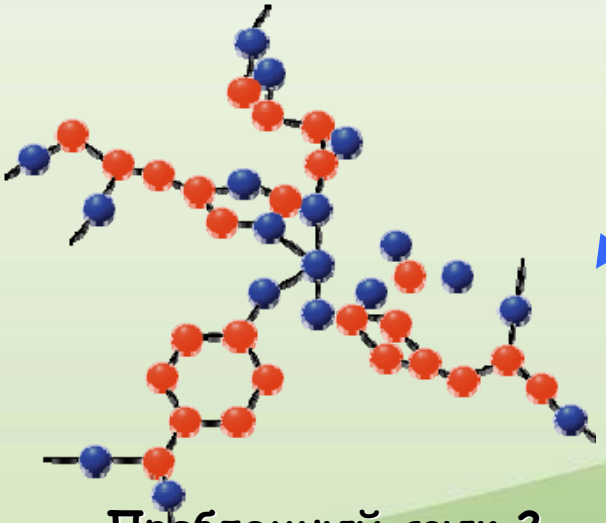
(противоречивый, алогичный, паралогичный)

Проблема 1

Проблема 2



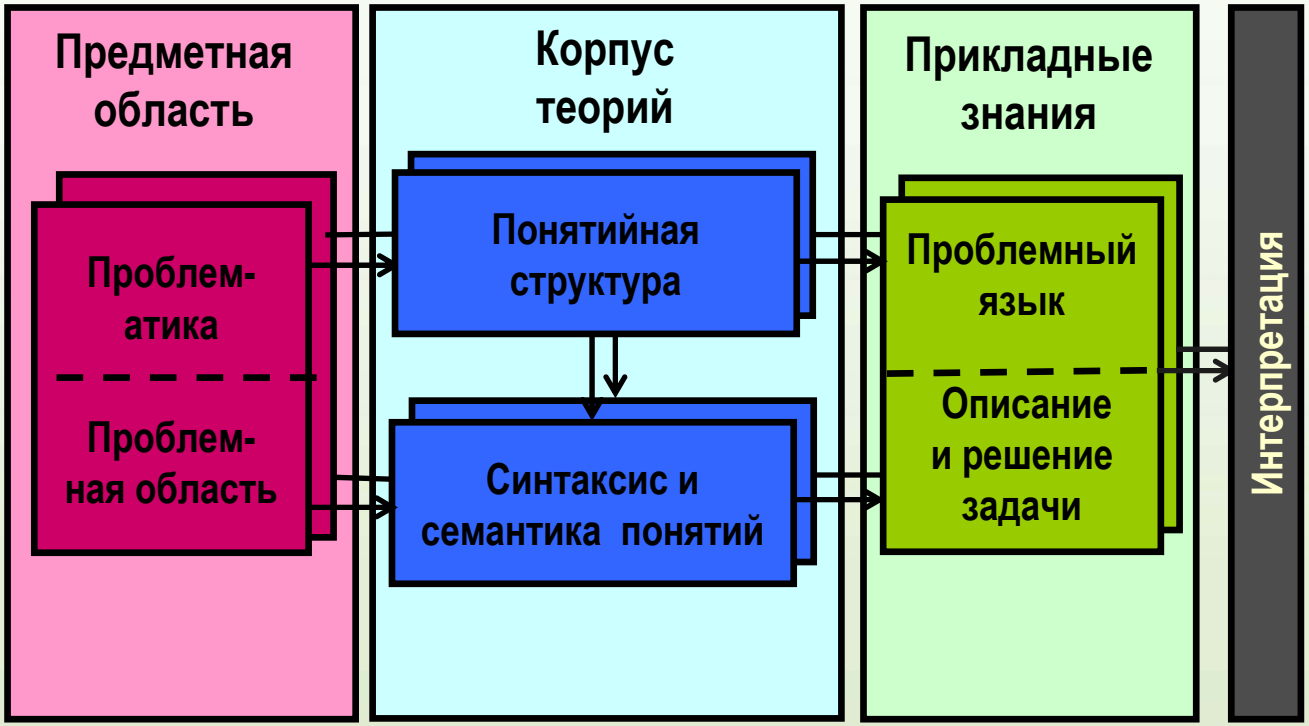
Проблемный язык 1
(непротиворечивый, логичный)



Проблемный язык 2
(непротиворечивый, логичный)

Противоречивость, паралогичность

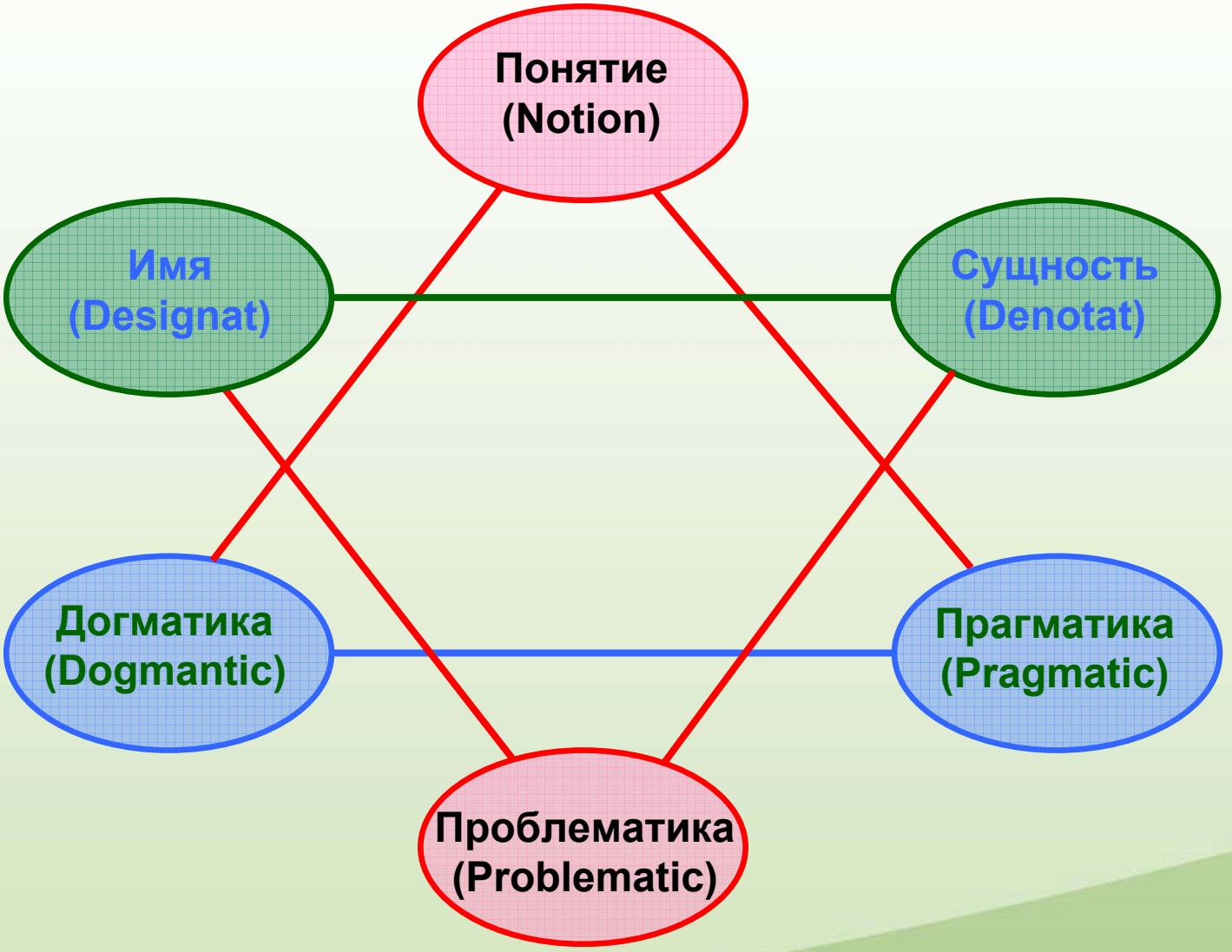
Содержательное моделирование



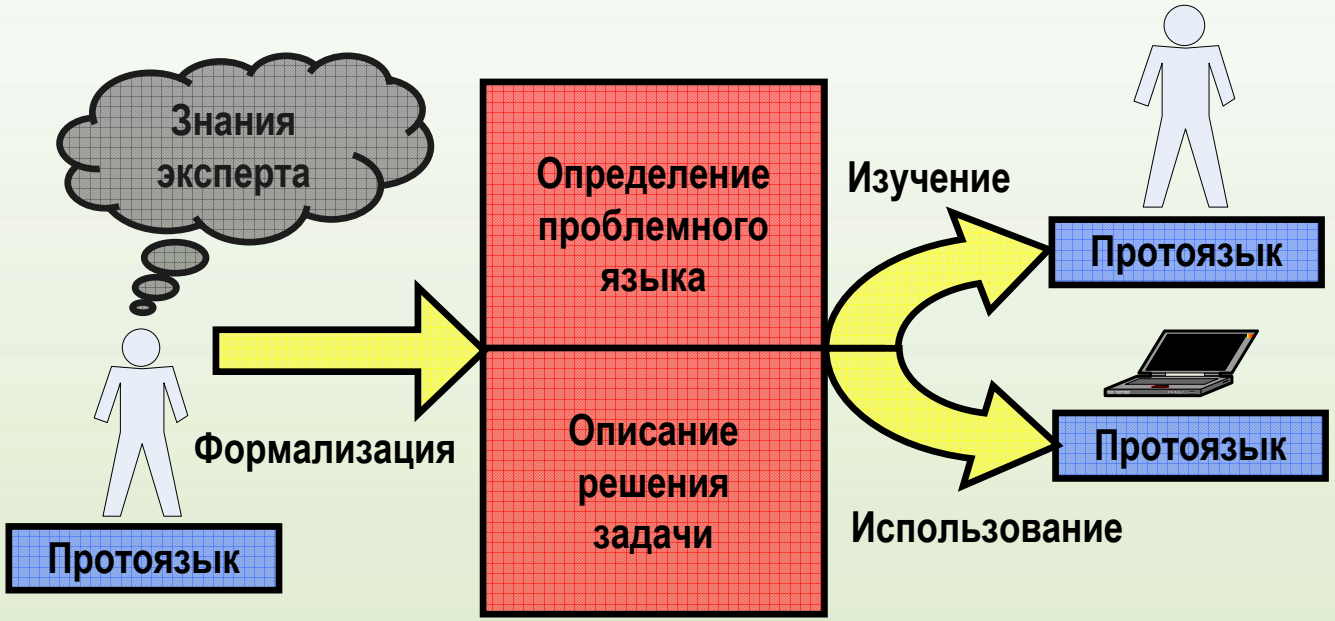
Синтаксис

Семантика

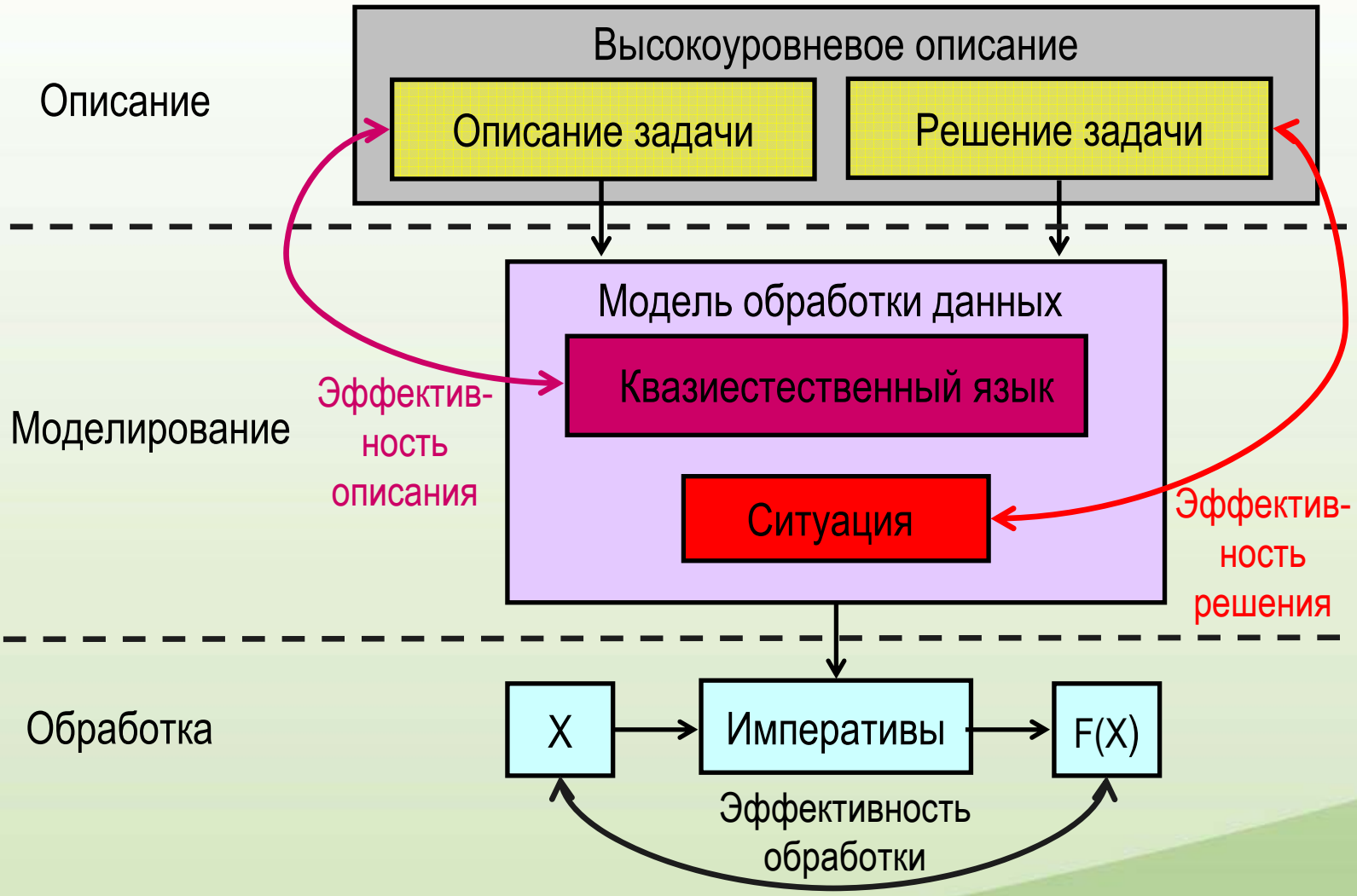
Структура понятия

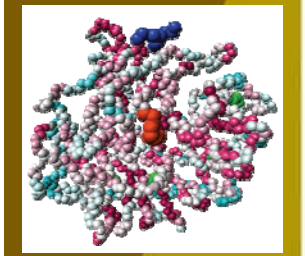


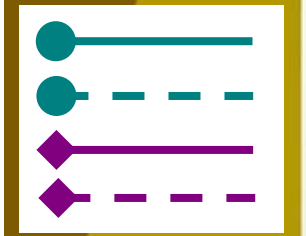
Применение технологии

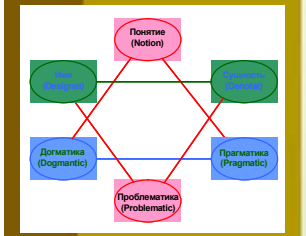


Эффективность подхода









План доклада

Содержательные выводы

- Обоснована парадигма корпусного подхода, позволяющая ставить и решать многоаспектные задачи формального моделирования
- Предложена методология понятийного анализа, основанная на фиксации 4-х видов связей между понятиями.
- Разработана технология представления и обработки знаний, сокращающая семантический разрыв между предметной областью и средствами ее формальной спецификации.
- Решена задача индуктивного определения семантики формальных языков.