КОНТЕКСТНАЯ ТЕХНОЛОГИЯ ПРОГРАММИРОВАНИЯ

В.С. Выхованец

Институт проблем управления РАН

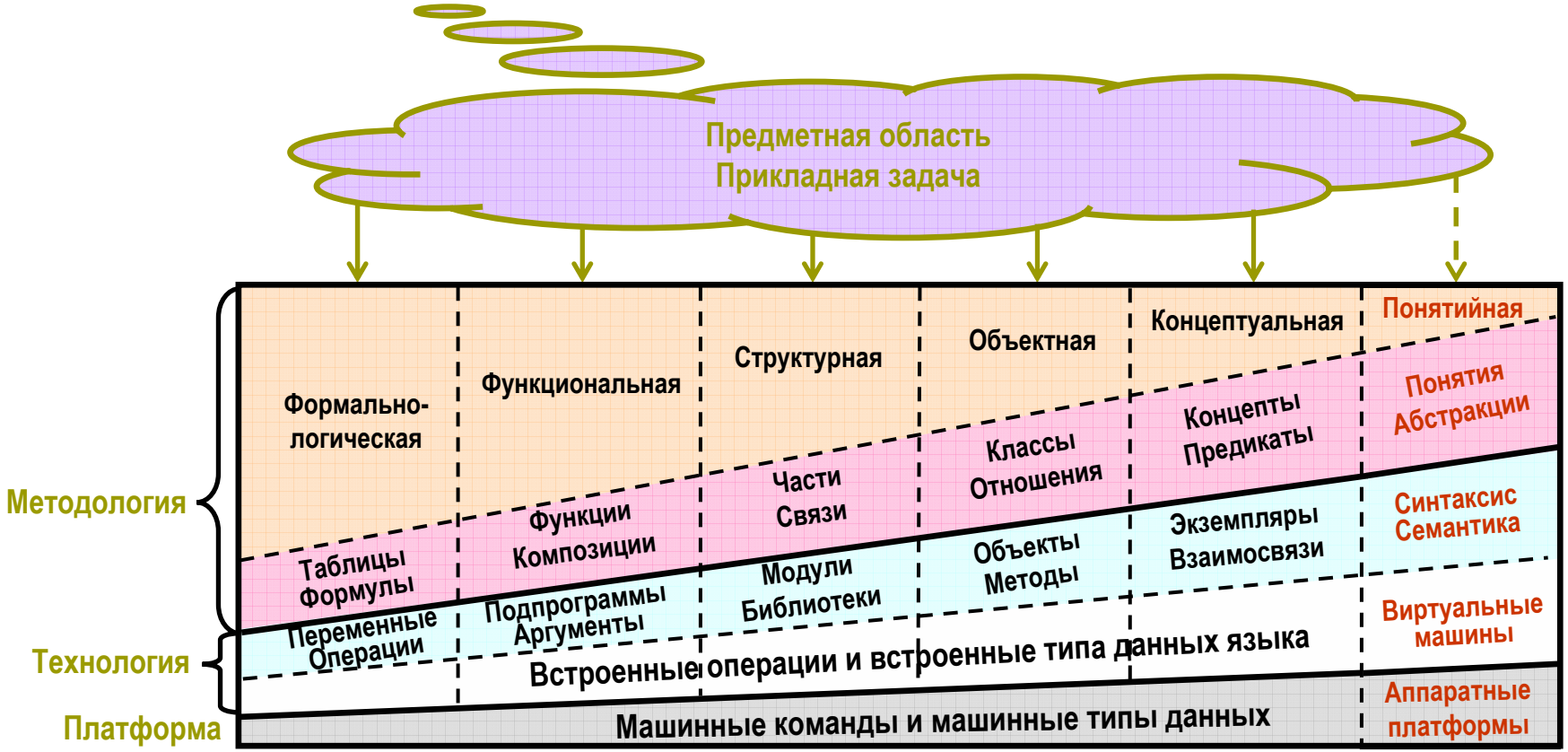
http://valery.vykhovanets.ru

Проблематика

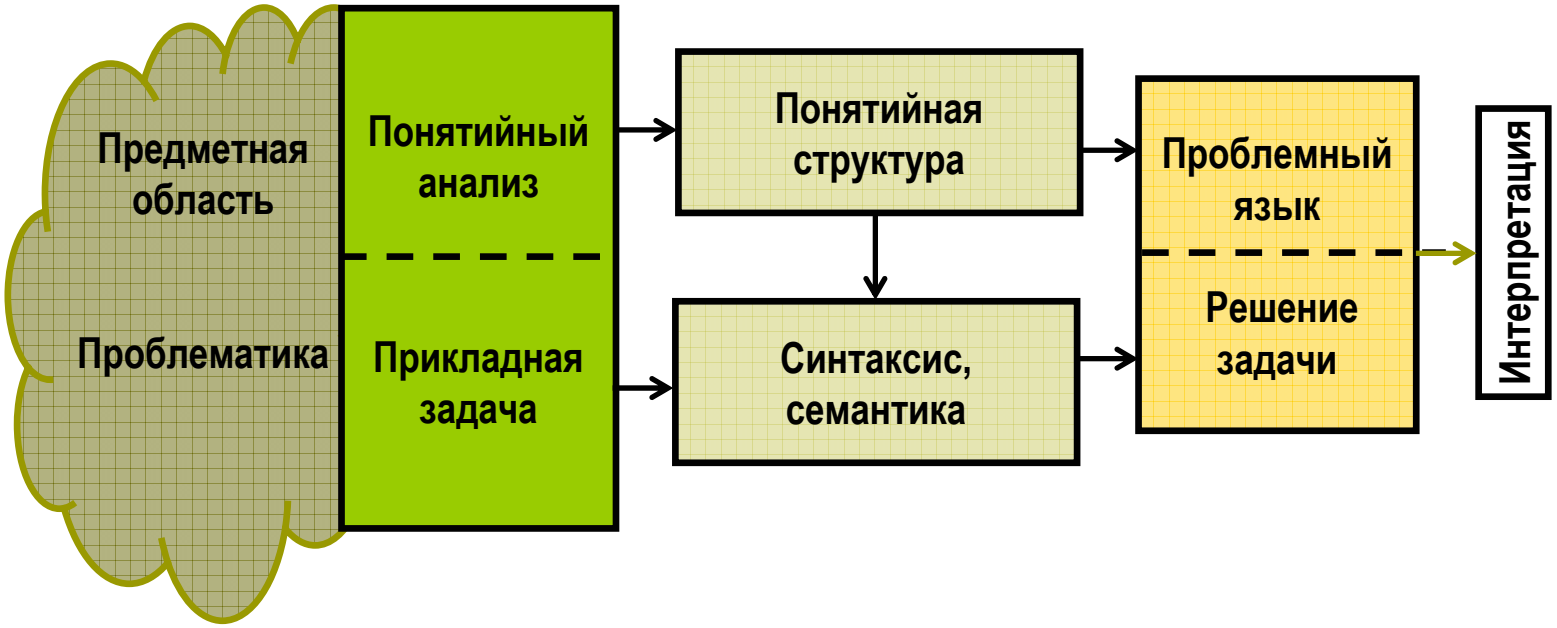
- Высокая сложность разработки
- Низкая надежность программных средств
- Недостаточное качество программ

- Увеличенные сроки разработки
- Превышение бюджета проектов
- Неудовлетворенные ожидания заказчика
- Прямой экономический ущерб

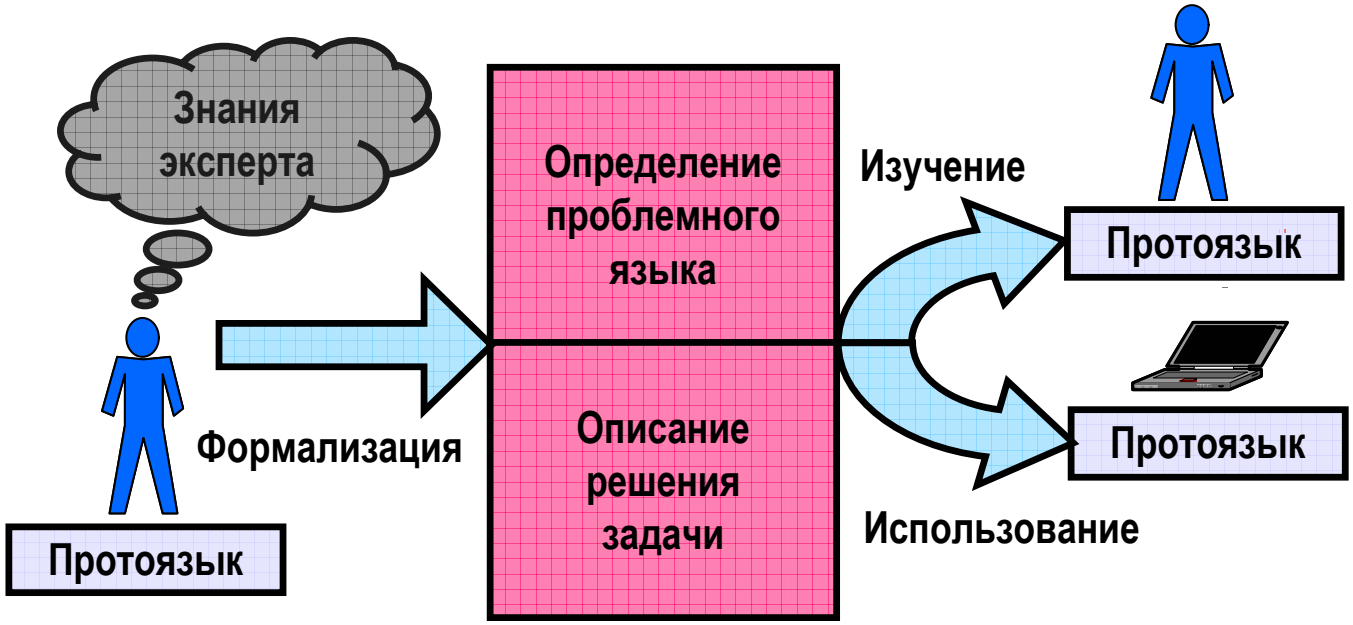
Семантический разрыв



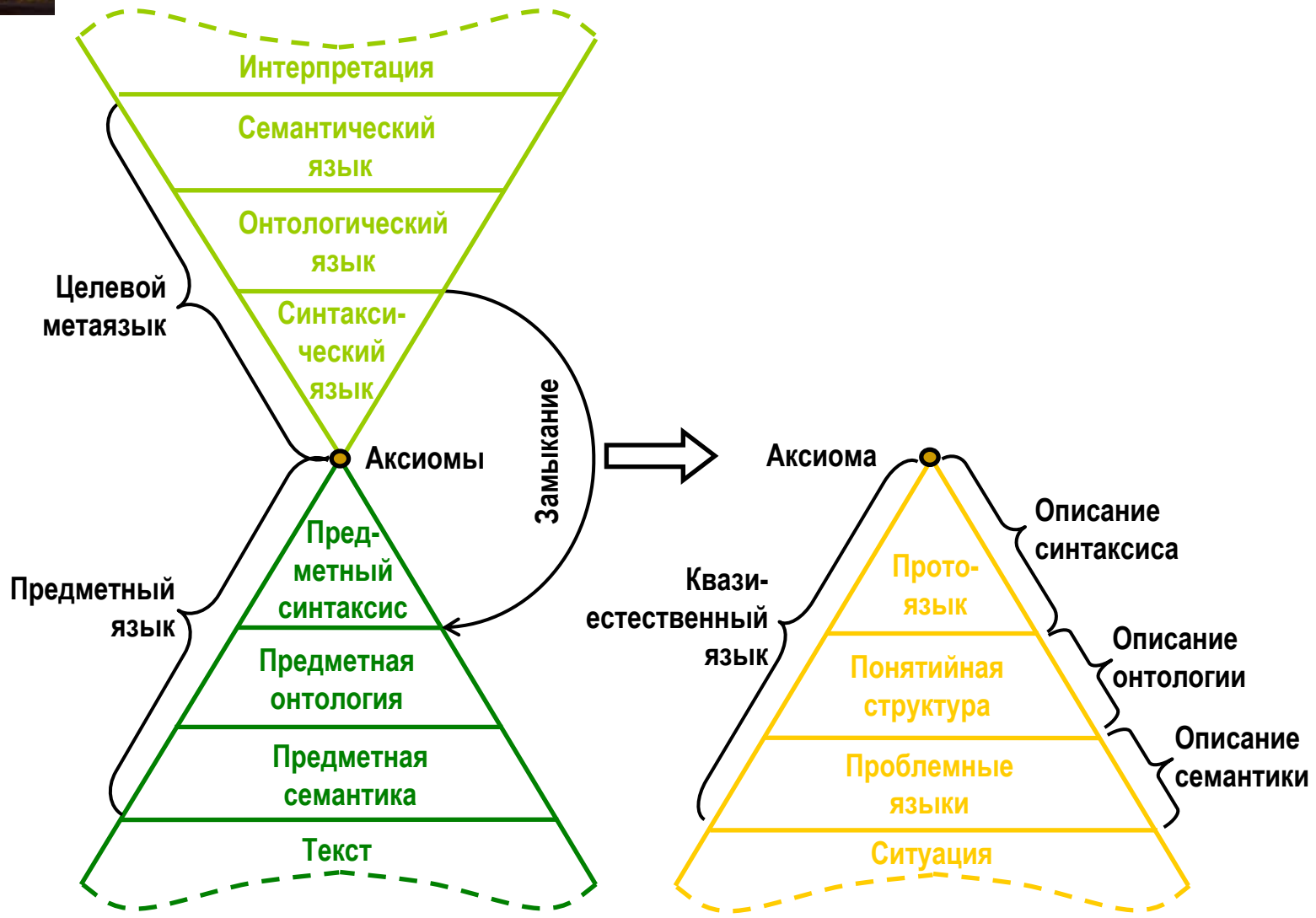
Методология



Технология


Знания эксперта
Протоязык
Формализация
Определение проблемного языка
Описание решения задачи
Изучение
Использование
Протоязык
Протоязык

Квазиестественный язык



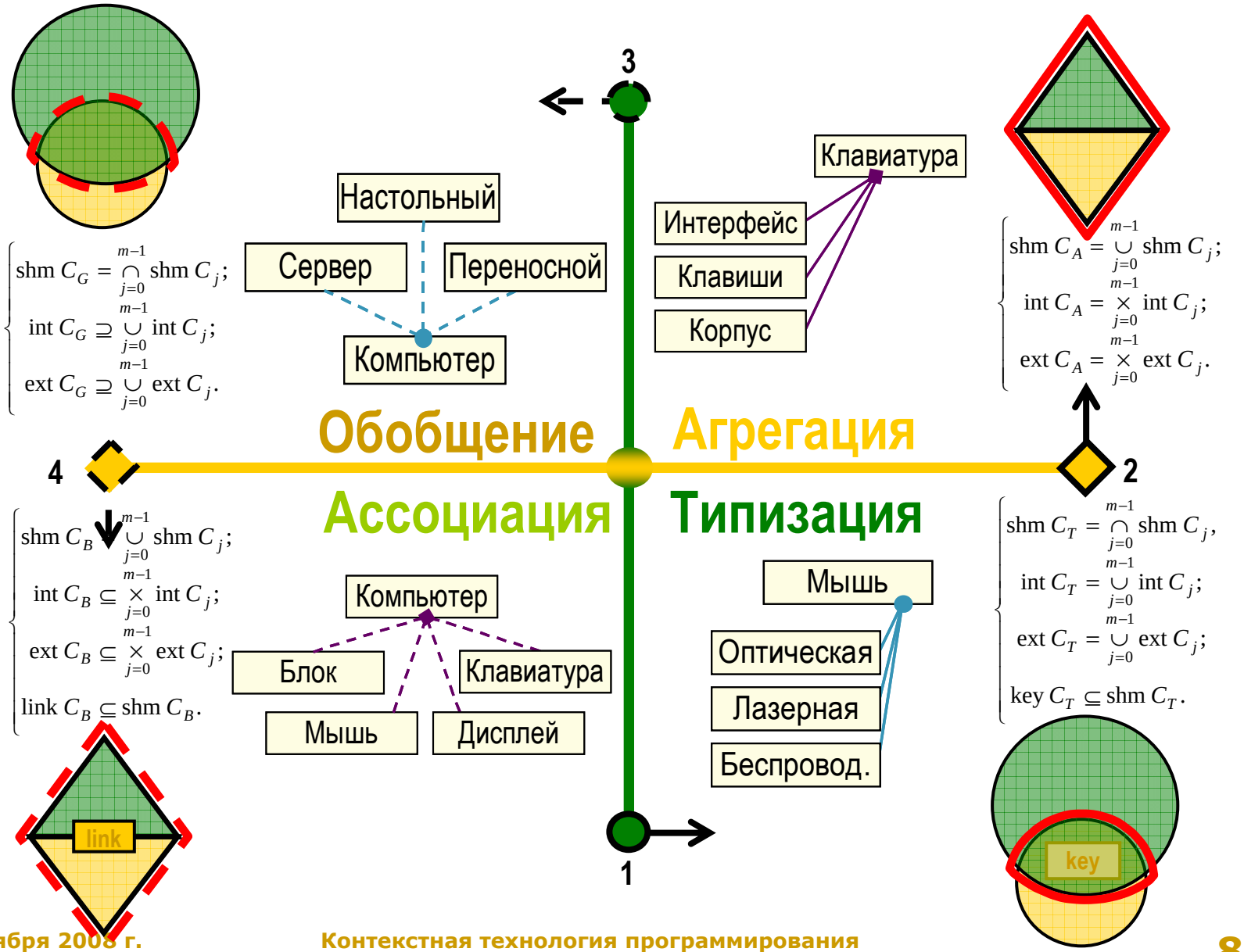
Протоязык

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

( ) *Целое* ( )
'*0*' { }
Натуральное { }
'-' **Натуральное** { }
( ) *Натуральное* ( )
"*[1-9]*" { }
Натуральное "*[0-9]*" { }

( ) *Boolean* ( )
'*false*' { }
'*true*' { }
″[A-Za-z][A-Za-z0-9]″* { }
'*(*' **Boolean** '*)*' { }
'*not*' **Boolean** { }
Boolean '*and*' **Boolean** { }
Boolean '*or*' **Boolean** { }

Абстракции



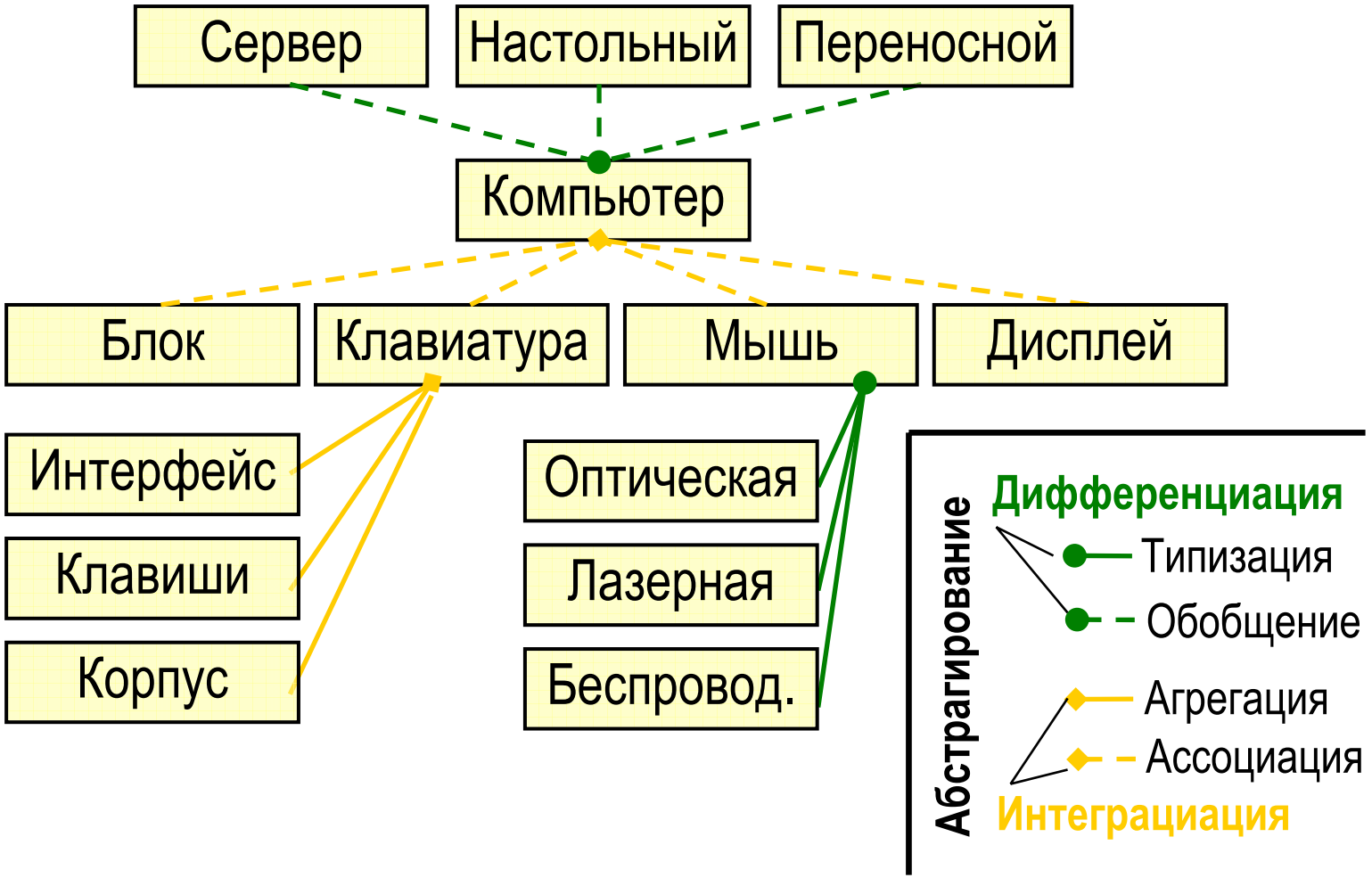
$$\begin{cases} \text{shm}\, C_G = \bigcap_{j=0}^{m-1} \text{shm}\, C_j; \\ \text{int}\, C_G \supseteq \bigcup_{j=0}^{m-1} \text{int}\, C_j; \\ \text{ext}\, C_G \supseteq \bigcup_{j=0}^{m-1} \text{ext}\, C_j. \end{cases}$$

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

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

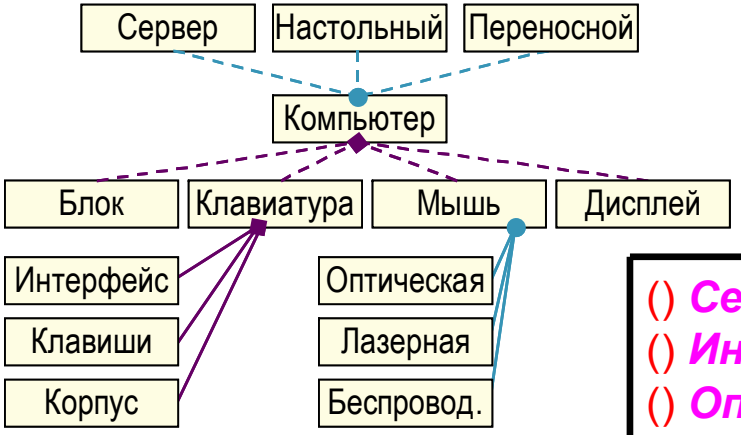
$$\begin{cases} \text{shm}\, C_T = \bigcap_{j=0}^{m-1} \text{shm}\, C_j, \\ \text{int}\, C_T = \bigcup_{j=0}^{m-1} \text{int}\, C_j; \\ \text{ext}\, C_T = \bigcup_{j=0}^{m-1} \text{ext}\, C_j; \\ \text{key}\, C_T \subseteq \text{shm}\, C_T. \end{cases}$$

Понятийная структура



Онтология

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**)

Проблемные языки

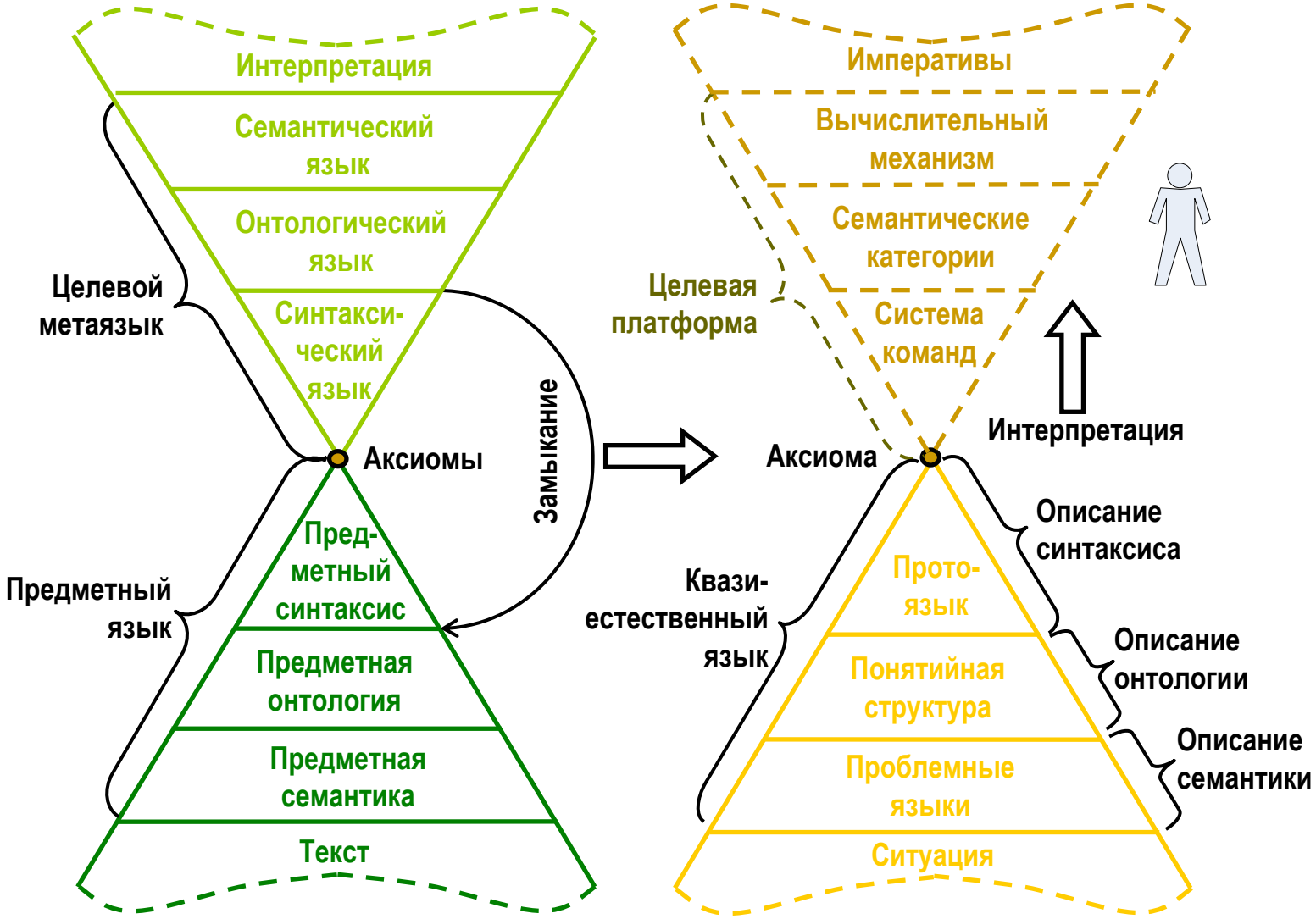
cognition → model [situation] [cognition]
model → essences [model]
essences → differenciation ***notion***
integration [intension]
intension → sentence [intension]
sentence → syntax semantic
syntax → item [syntax]
item → **notion** | '″' [***terms***] '″'
semantic → pragmatic [semantic]
pragmatic → [***aspect***] '**{**' [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 >
```

Понятийный анализ

- **Разделение** сущностей предметной области на сигнификативные и денотационные;
- **Означивание** сигнификативных сущностей и выявление существенных признаков у денотационных;
- **Сопоставление** денотационных сущностей и определение их общих и различающихся признаков;
- **Образование** новых или определение уже существующих понятий на основе интеграции и дифференциации признаков;
- **Создание** понятийной структуры предметной области путем описания отображений одних понятий на другие;
- **Уточнение** способа абстрагирования понятий (обобщения или типизации, ассоциации или агрегации);
- **Вычисление** схем понятий и задание ключей – для типизации, связей – для ассоциации.

Интерпретация



Аксиома

«Пустые» квадратные скобки реализуют запись в область промежуточного кода того значения, которое задается текущим элементом предложения

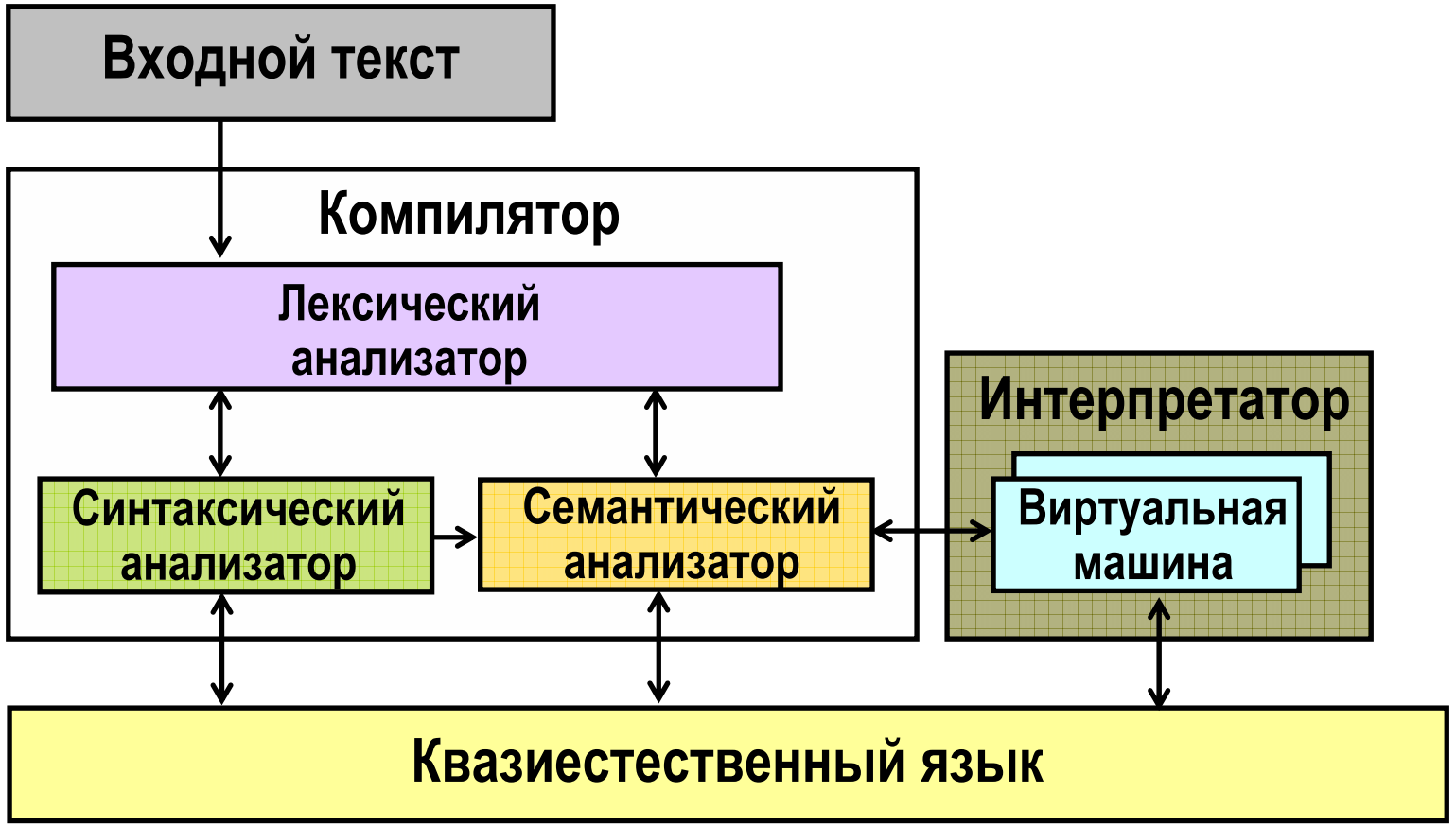
```
() ()
    '#' "[0-9A-F][0-9A-F]" [ ] { }
() Boolean ()
    'not' Boolean
        { #58 #F7 #D0 #50 }
    Boolean 'or' Boolean
        { #58 #5A #0B #C2 #50 }
    Boolean [a] 'imp' [b] Boolean
        { not a or b }
```

Низкоуровневая семантика

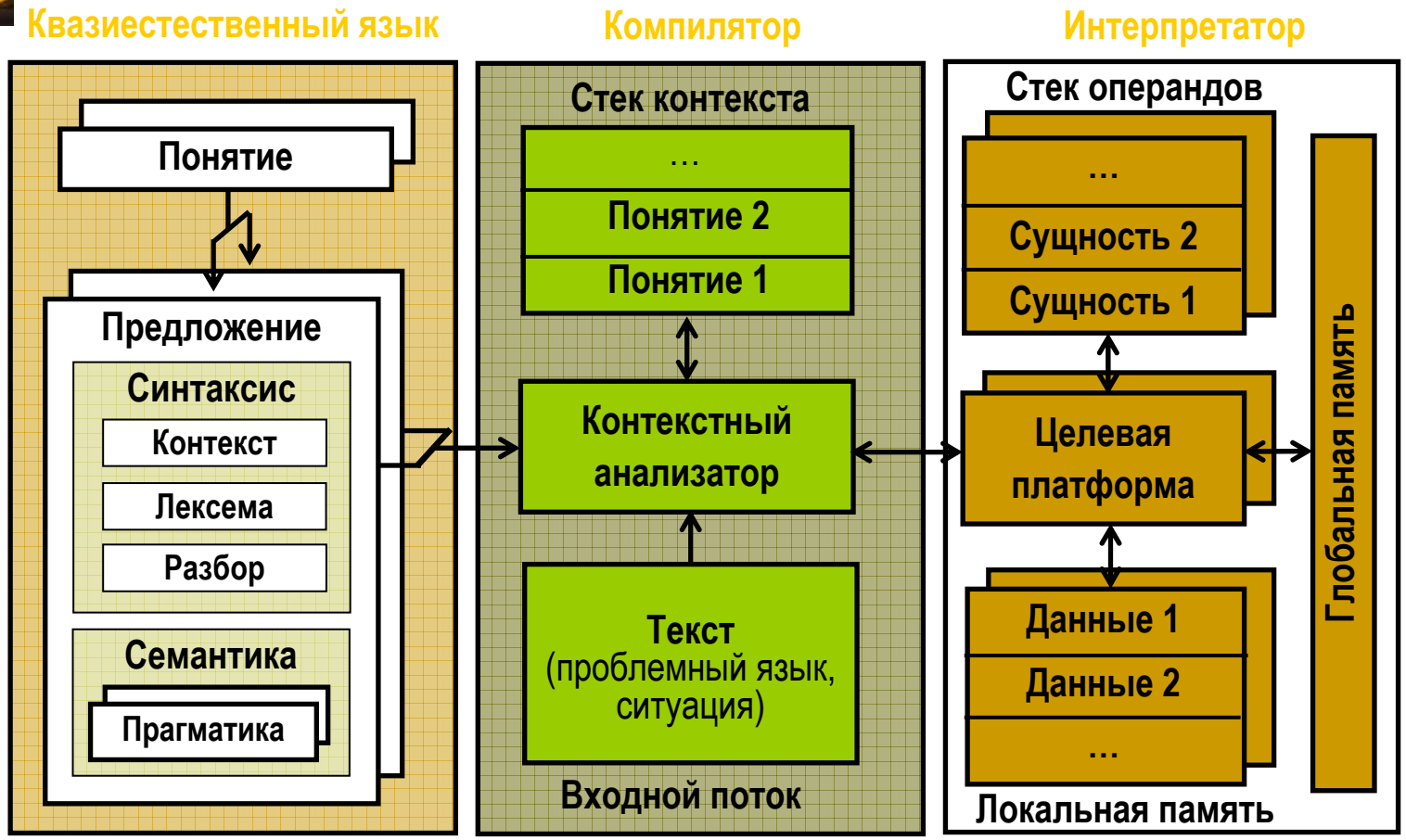
```
() ()
    '`' "[0-9A-Za-z\s.,]+" [ ] '`' { }
() Boolean ()
    'not' Boolean
        { `pop eax` `not eax`
          `push eax` }
    Boolean 'or' Boolean
        { `pop eax` `pop edx`
          `or eax, edx` `push eax` }
    Boolean [a] 'imp' [b] Boolean
        { not a or b }
```

Высокоуровневая семантика

Система


Входной текст
Компилятор
Лексический анализатор
Синтаксический анализатор
Семантический анализатор
Интерпретатор
Виртуальная машина
Квазиестественный язык

Разнесенный разбор



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

Контекст — Лексема — Разбор — Прагматика

Декларативный стиль

```
() 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
```

Императивный стиль

```
( ) Движение ( )
        ″[Дд]вижение″ {…}
( ) Дверь ( )
        ″[Дд]верь″ {…}
( ) Этаж ( )
        'этаж' ″[0-9]″ {…}
        'этаж' 'вызова' ″|вверх|вниз″ {…}
        'этаж' 'лифта' {…}
( ) Вызов (Этаж Движение)
        'вызов' {…}
( ) 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 вызов нет, то Ожидание.
    Если этаж вызова равен этаж 2,
        то Открыть, иначе Решение.

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

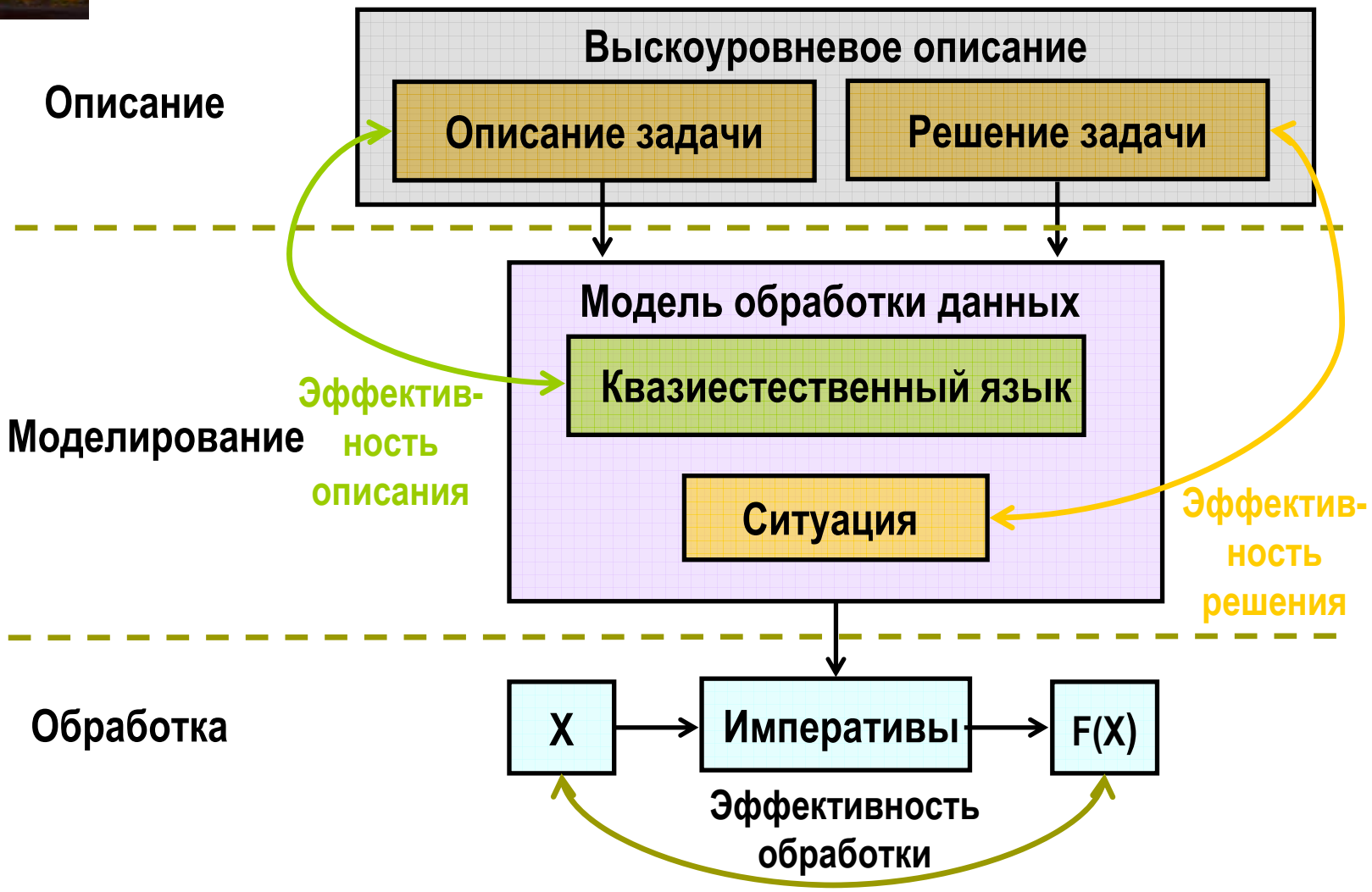
Закрыть:
    Дверь закрыть.
    Если дверь закрыта, то Решение,
        иначе Закрыть.

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 вызов нет и этаж лифта выше этаж 2,
        то Вниз.
    Если вызов нет и этаж лифта ниже этаж 2,
        то Вверх, иначе Ожидание.

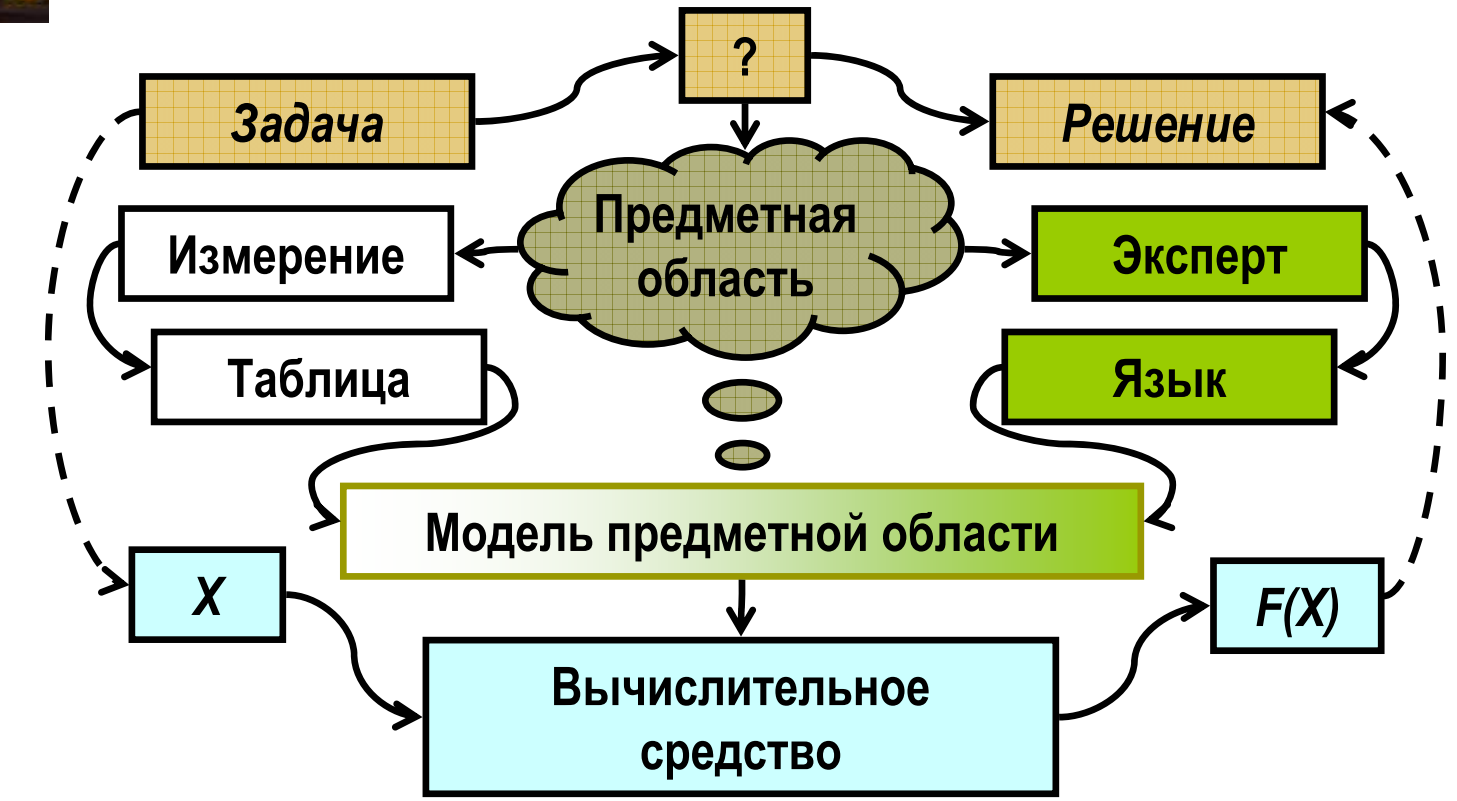
Вверх:
    Движение вверх.
    Если этаж лифта равен этаж вызова вверх,
        то Открыть, иначе Вверх.

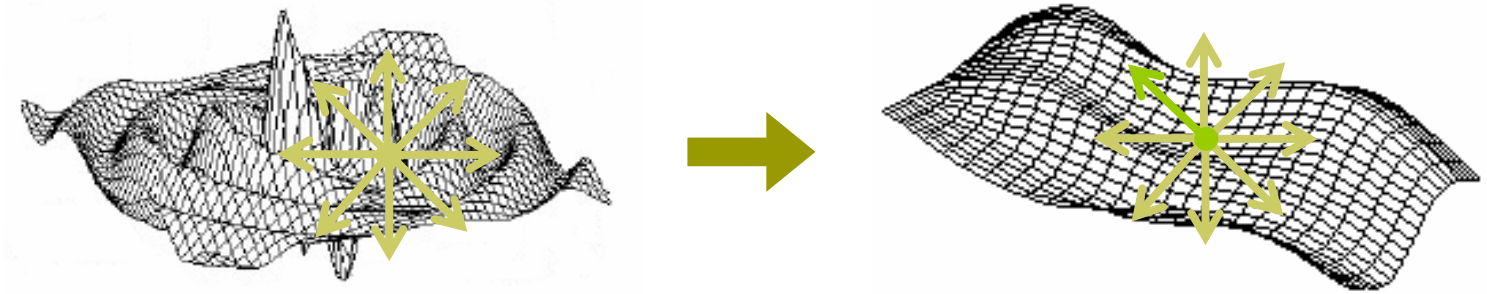
Вниз:
    Движение вниз.
    Если этаж лифта равен этаж вызова вниз,
        то Открыть, иначе Вниз.

Эффективность



Область применимости





Выводы

- Описание предметной области и решаемой задачи на языке, наиболее близком содержательным представлениям
- Повышение надежности и качества программных средств, легкость доработки и развития
- Возможность адекватным образом выразить и реализовать естественный параллелизм задачи