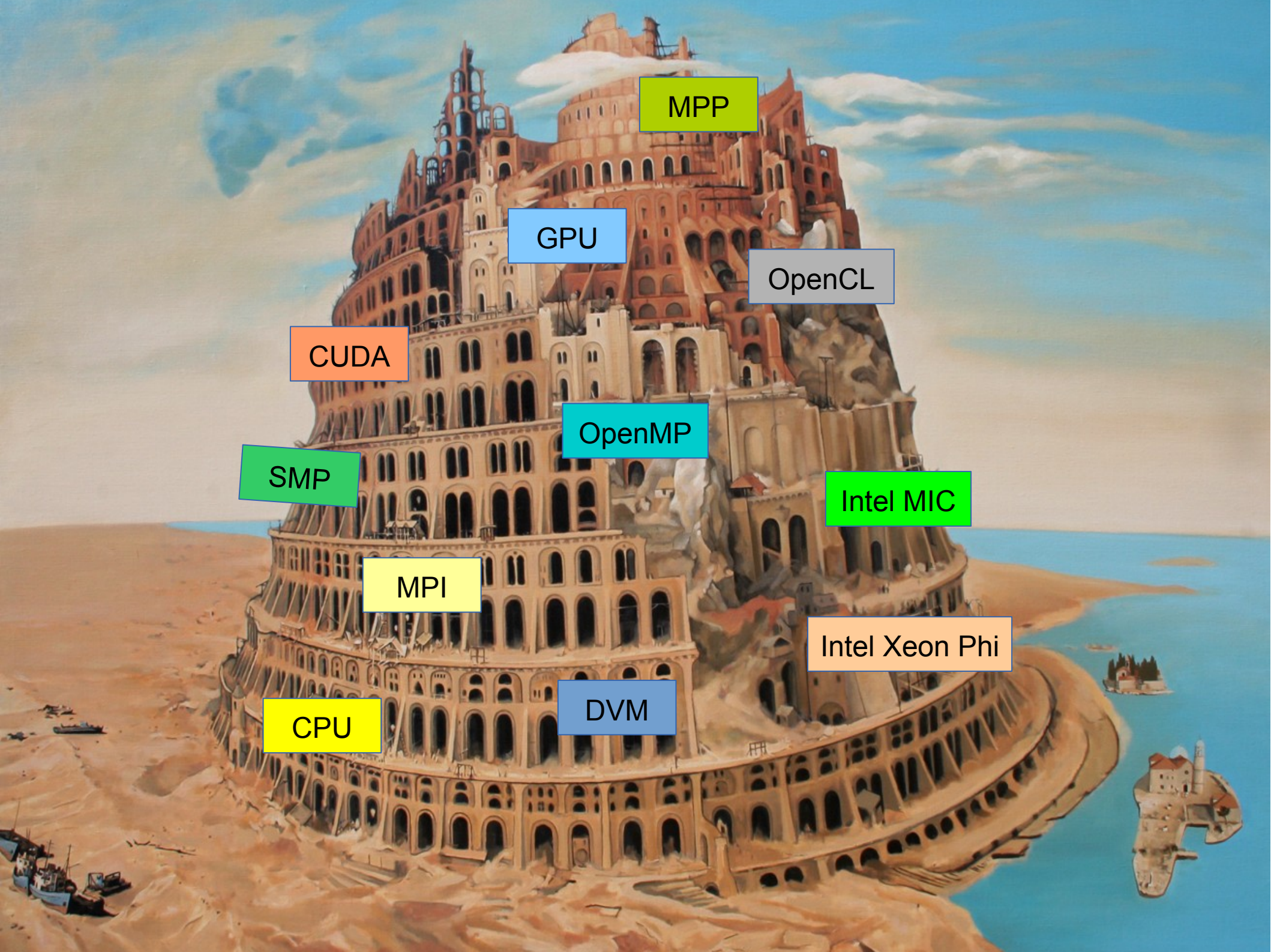
PLC-2017

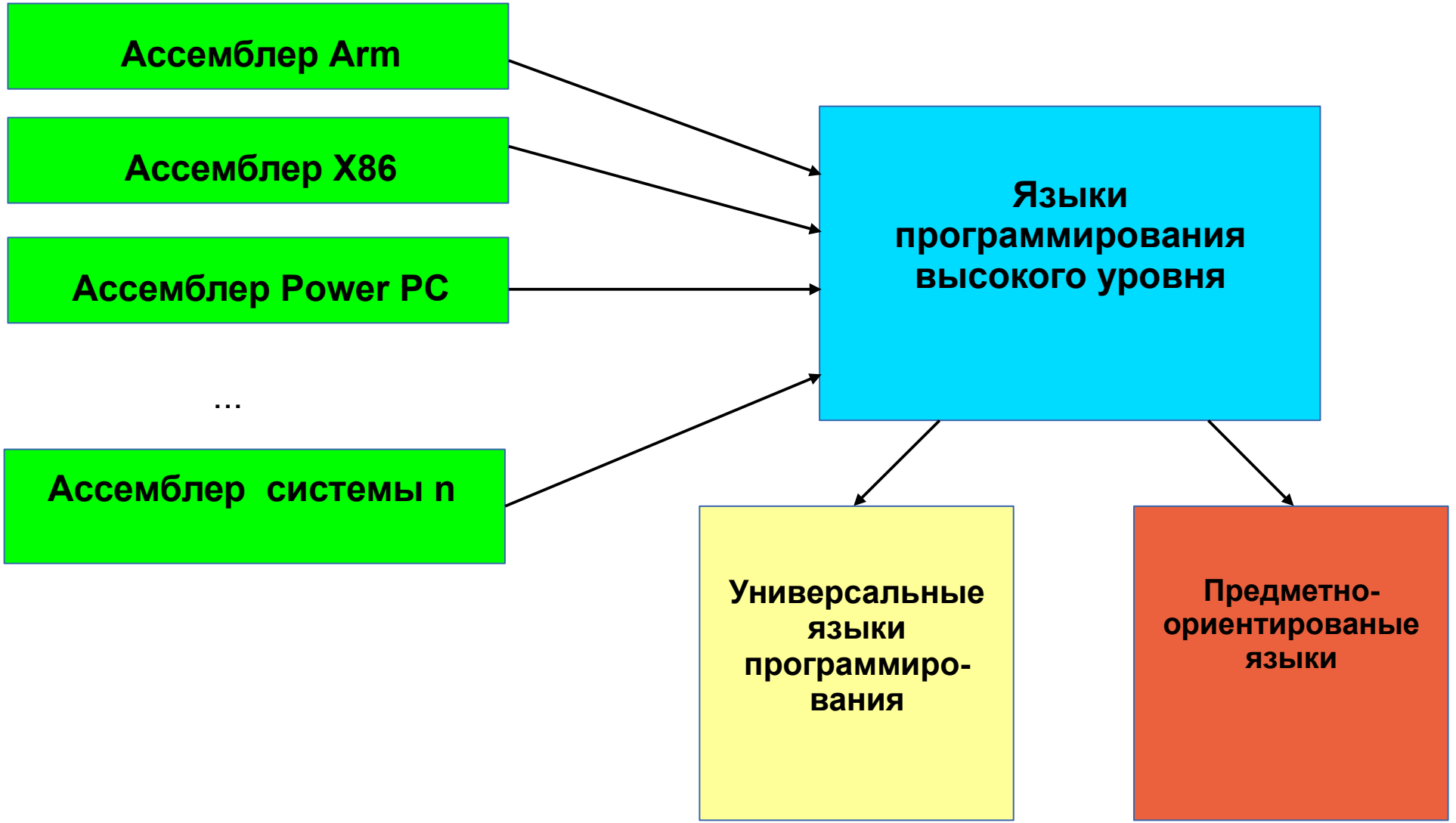
СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ
SIBERIAN FEDERAL UNIVERSITY

Языковая поддержка архитектурно-независимого параллельного программирования

Легалов А.И. (legalov@mail.ru)


MPP
GPU
OpenCL
CUDA
OpenMP
SMP
Intel MIC
MPI
Intel Xeon Phi
DVM
CPU

Последовательные вычислительные системы



Архитектуры параллельных ВС

Общие идеи

http://softcraft.ru/fppp/

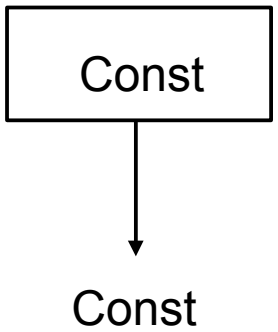
- Вместо распараллеливания последовательных программ – сжатие параллелизма программы, написанной в максимально параллельном стиле.
- Использование максимально параллельной программы для анализа, оптимизации, тестирования, верификации, отладки.
- Последующее сжатие параллелизма с применением формальных методов в соответствии с используемыми вычислительными ресурсами.

Модель функционально-потоковых параллельных вычислений

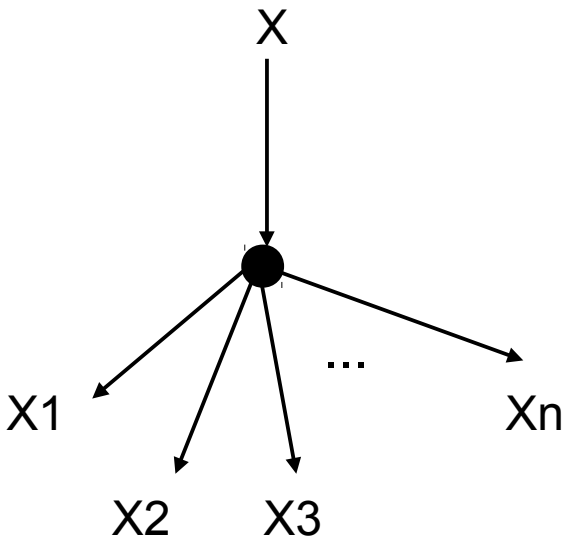
- Ориентация на бесконечные вычислительные ресурсы (принцип единственного использования вычислительных ресурсов).
- Запуск операции по готовности данных. Определяется в соответствии с аксиоматикой языка и его алгеброй преобразований (неявное управление вычислениями).
- Программа не имеет циклов, а значит и механизмов описания повторного использования ресурсов (все итеративные вычисления задаются через рекурсию).
- Для повышения эффективности при описании параллелизма используются специальные программо-формирующие структуры данных, определяемые как списки различного вида.
- Параллелизм на уровне элементарных операций.
- Программа – информационный граф.
- Выбор операций и аксиом, определяющих примитивы языка, ориентирован на наглядное текстовое представление информационного графа программы.

Ключевые идеи навеяны статьей Дж. Бэкуса:
Backus J. Can programming be liberated from von Neuman style? A functional stile and its algebra of programs // CACM. 1978. Vol. 21, N 8. P. 613–641.

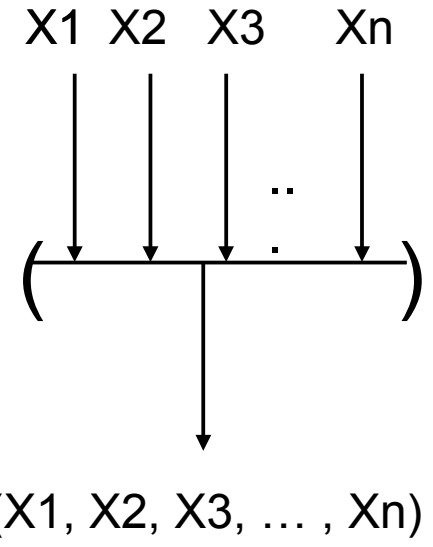
Основные операторы модели вычислений



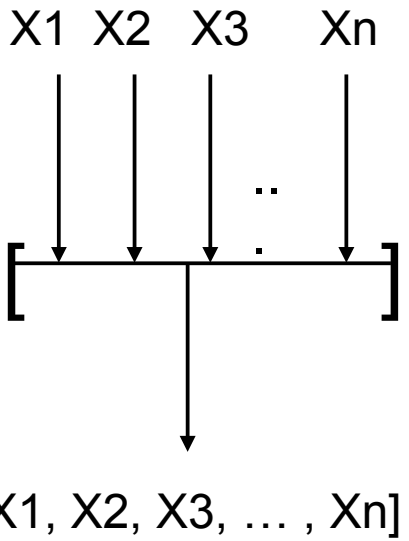
Константный оператор



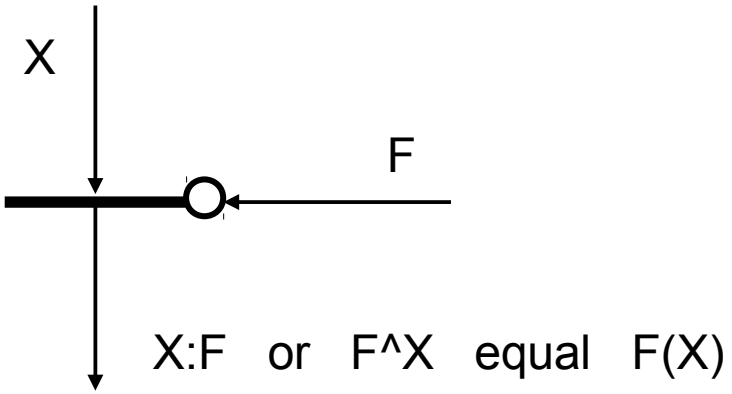
Копирование данных



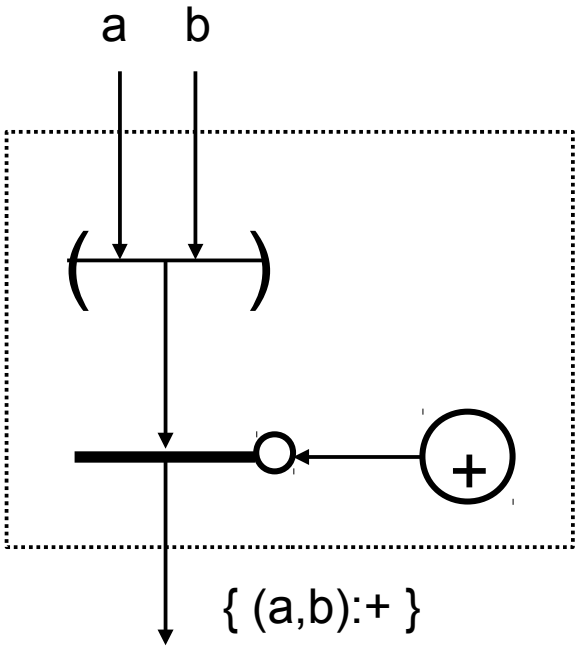
Список данных



Параллельный список



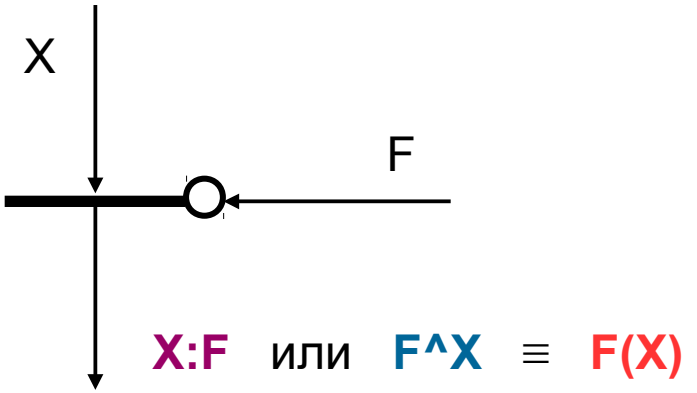
Оператор интерпретации



Задержанный список

Имеется только одна функция: оператор интерпретации
Обеспечивает преобразование входного набора данных X, выступающего в качестве аргумента, в выходной набор Y, который является результатом, используя при этом входной набор F в качестве функции, определяющей алгоритм преобразования.

Постфиксная запись: **X:F ⇒ Y**
Префиксная запись: **F^X ⇒ Y**

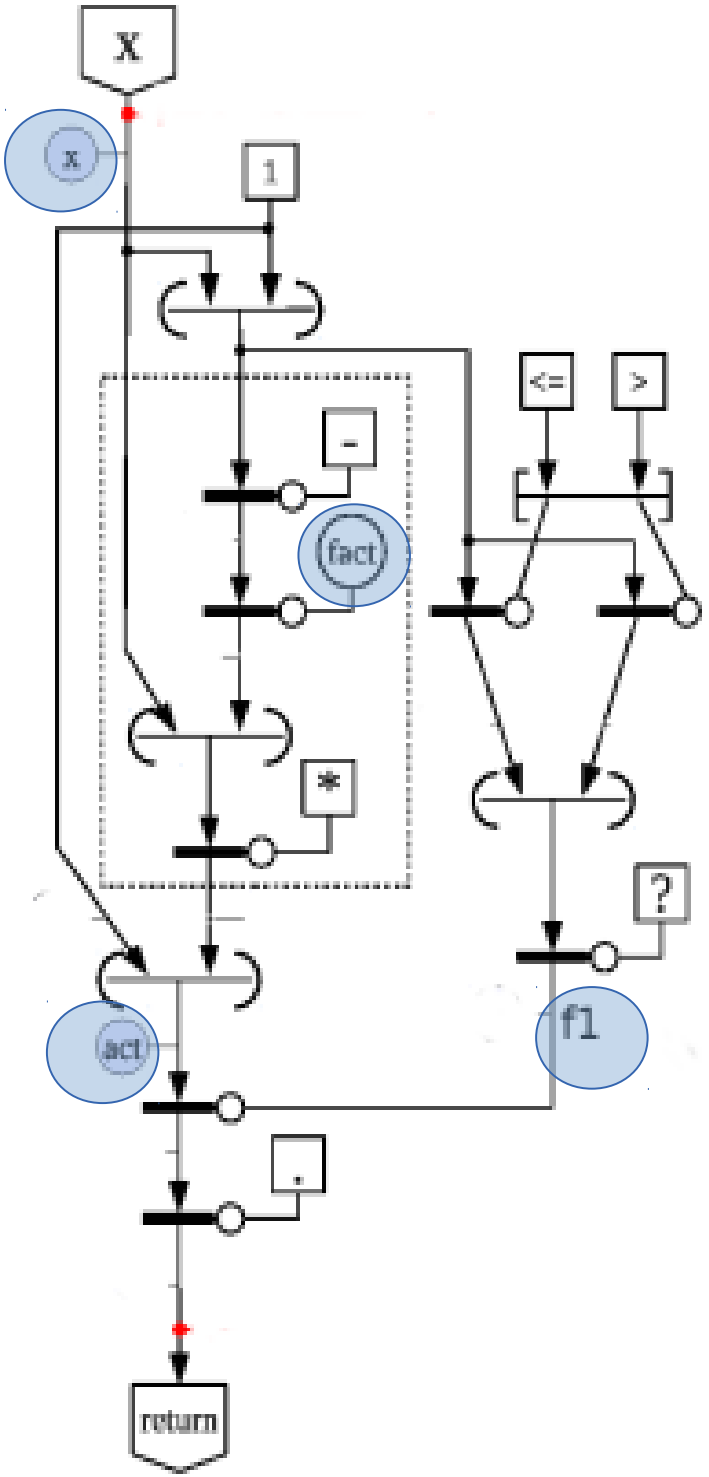



```
(a,b):+  или   +^(a,b)   ≡   a+b

x: sin   или   sin ^ x   ≡   sin(x)

((b,b):*,((4, a):*,c):*):- ≡
                    b * b - 4 * a * c
```

```
// Factorial definition
fact << funcdef x {
   // selector of variant
   fl << ((x,1):[<=,>]):?;
   // two possible variants
   act << (
      1,
      { (x, (x,1):-:fact ):* }
   );
   return << act:fl:.;
}
```



Преобразования параллельных списков

[(x1, x2), (x3, x4),(x5,x6)]:+
=> [(x1, x2):+, (x3, x4):+,(x5,x6):+]

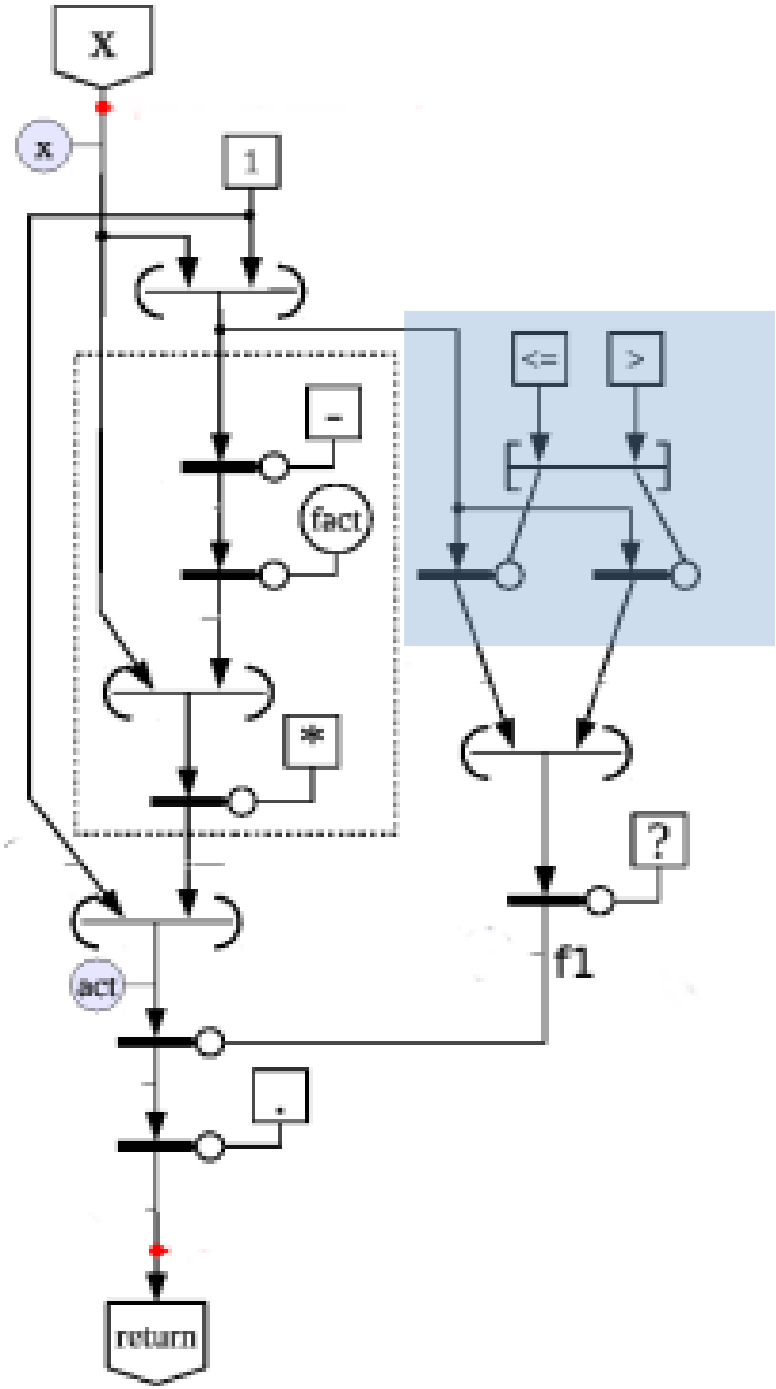
(x1,x2):[+,-,*,/]
=> [(x1,x2):+,(x1,x2):-,(x1,x2):*,(x1,x2):/]

Общий случай:

[x,y,z]:[f,g,h]
=> [x:[f,g,h],y:[f,g,h],z:[f,g,h]]
=> [[x:f,x:g,x:h],[y:f,y:g,y:h],[z:f,z:g,z:h]]

В программе:

(x,1):[<=,>]
=> [(x,1):<=,(x,1):>]



Ветвления

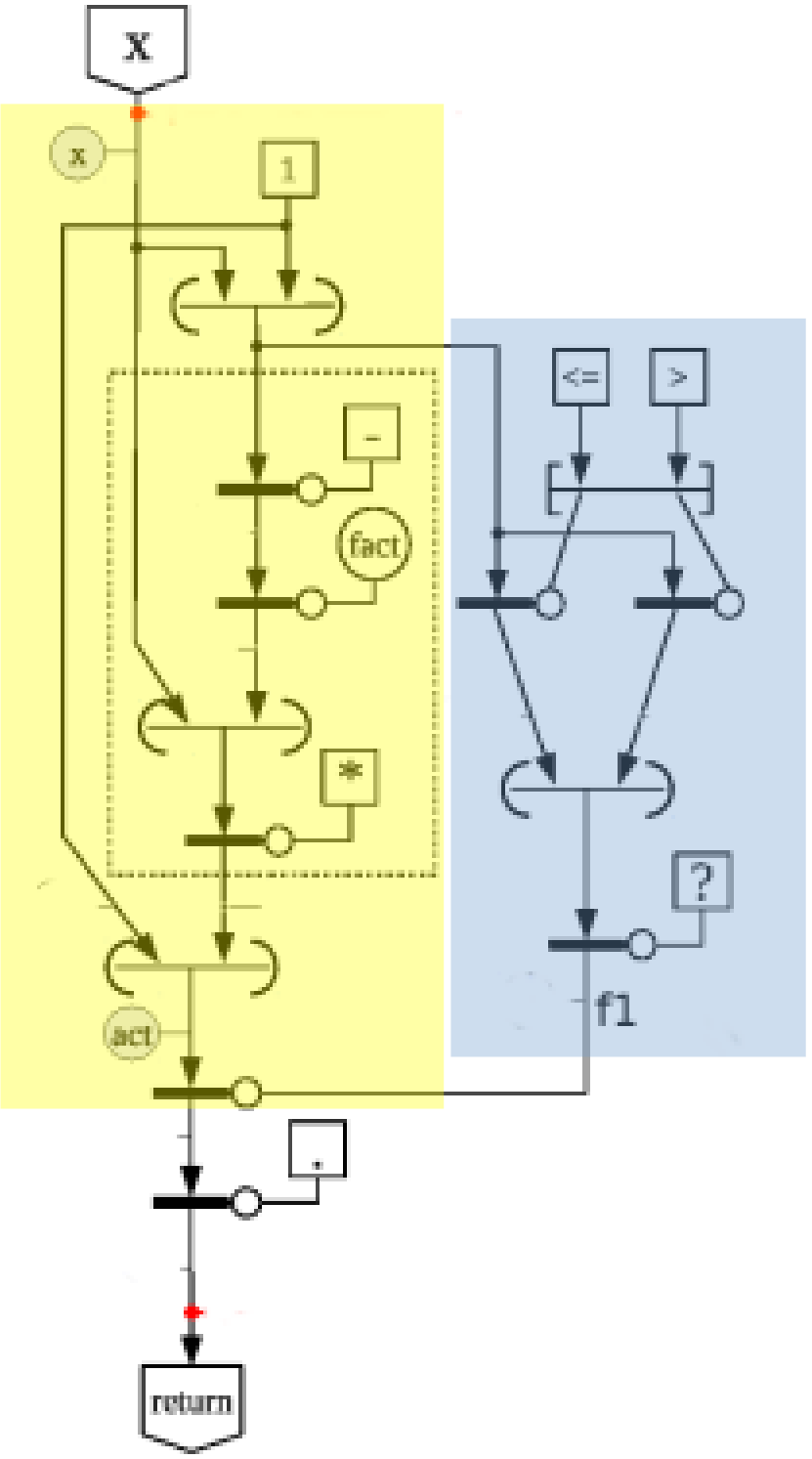
Селектор:

```
(3,7,2,6):3 => 2
(x,1):[<=,>]
                => [true, false] or [false, true]
(true,true,false,false,true,false):?
    ==> [1,2,5]
```

В программе:

```
((x,1):[<=,>]):?
     => ([true, false]):? or ([false, true])?
     => (true, false):? or (false, true):?
     => [1] or [2] == 1 or 2 >> f1
```

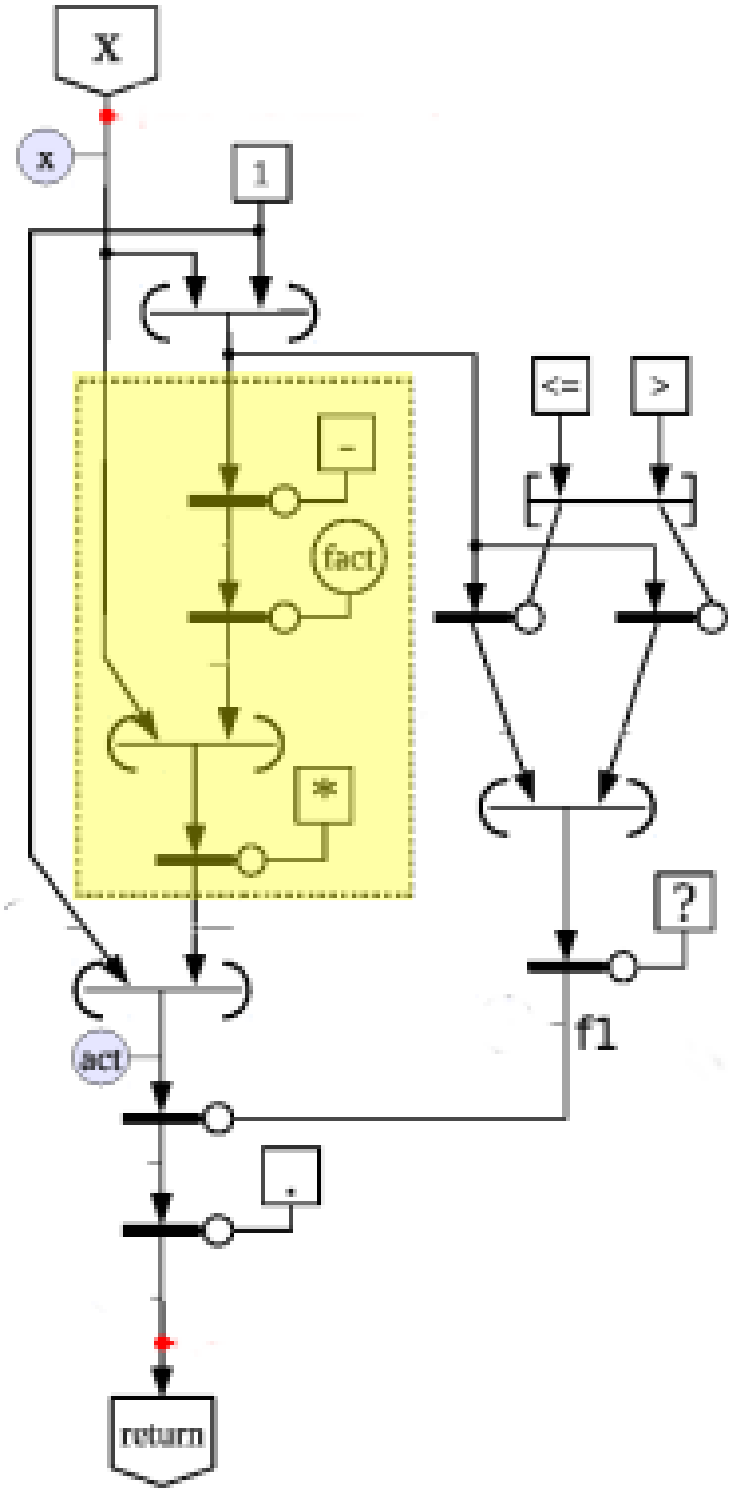
```
( 1,  { (x, (x,1):-:fact ):* }  ) : 1
     => 1          // 0! or 1!
( 1,  { (x, (x,1):-:fact ):* }  ) : 2
     => { (x, (x,1):-:fact ):* }
                   // x*(x-1)!
```

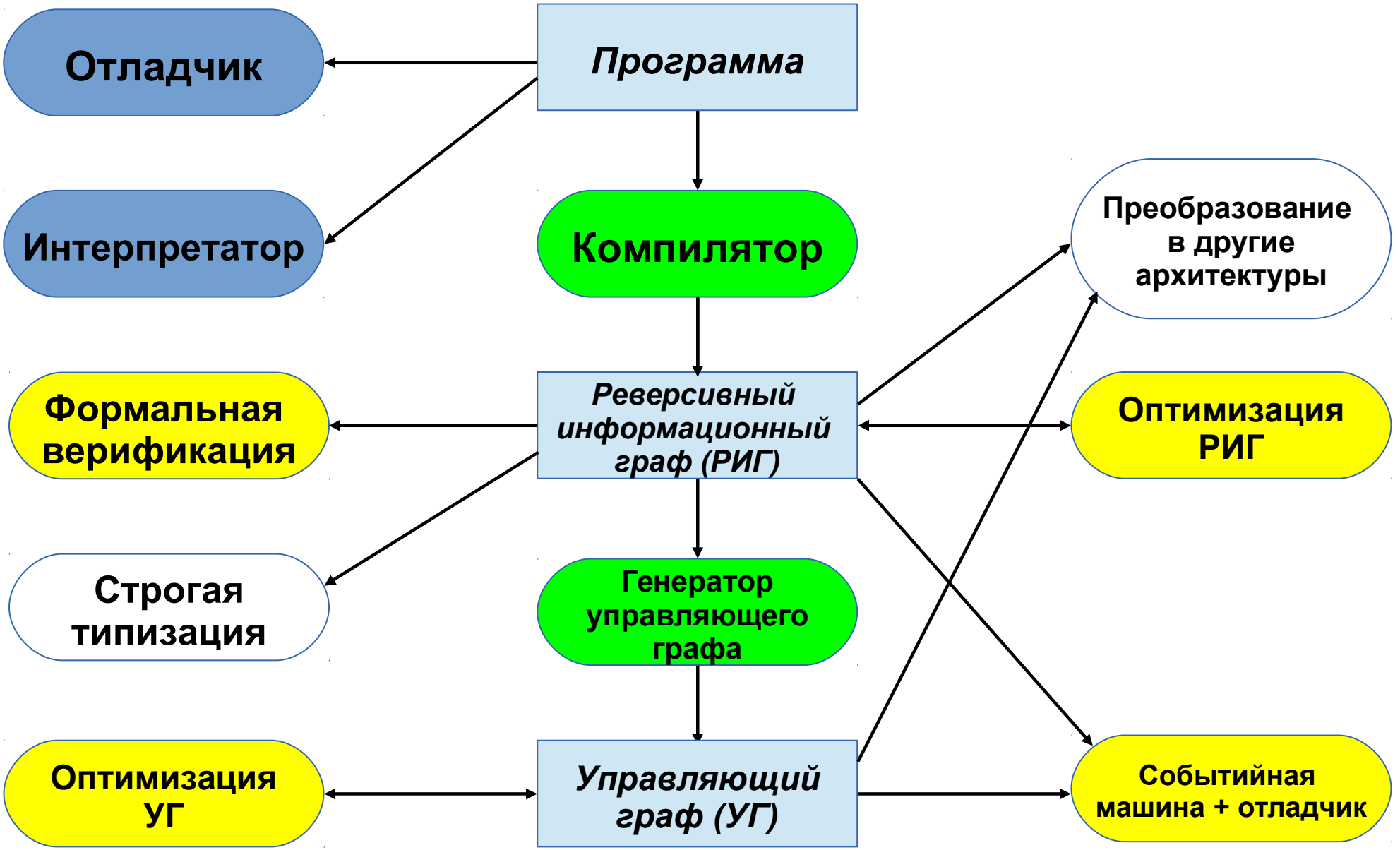


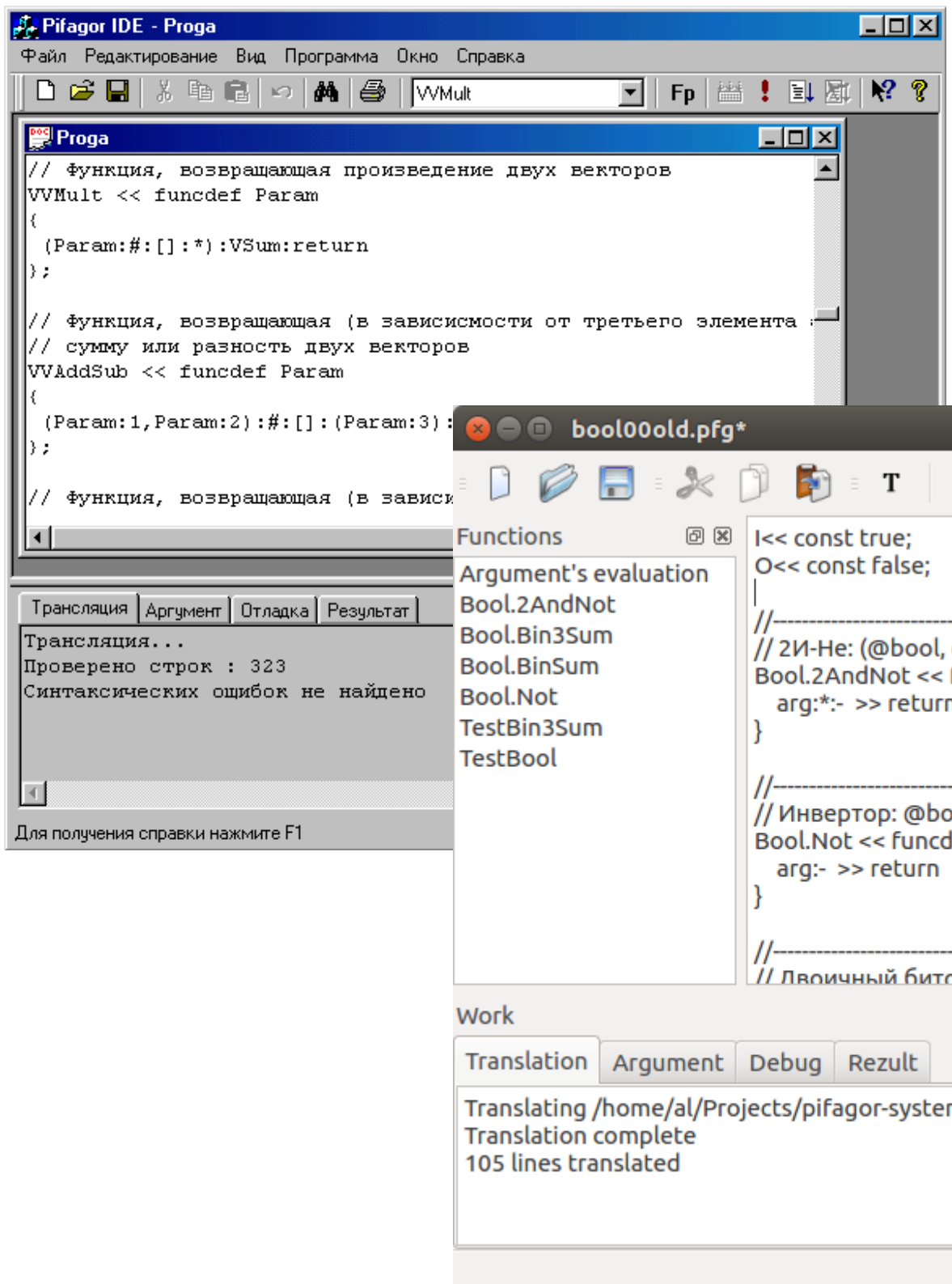
Задержанный список

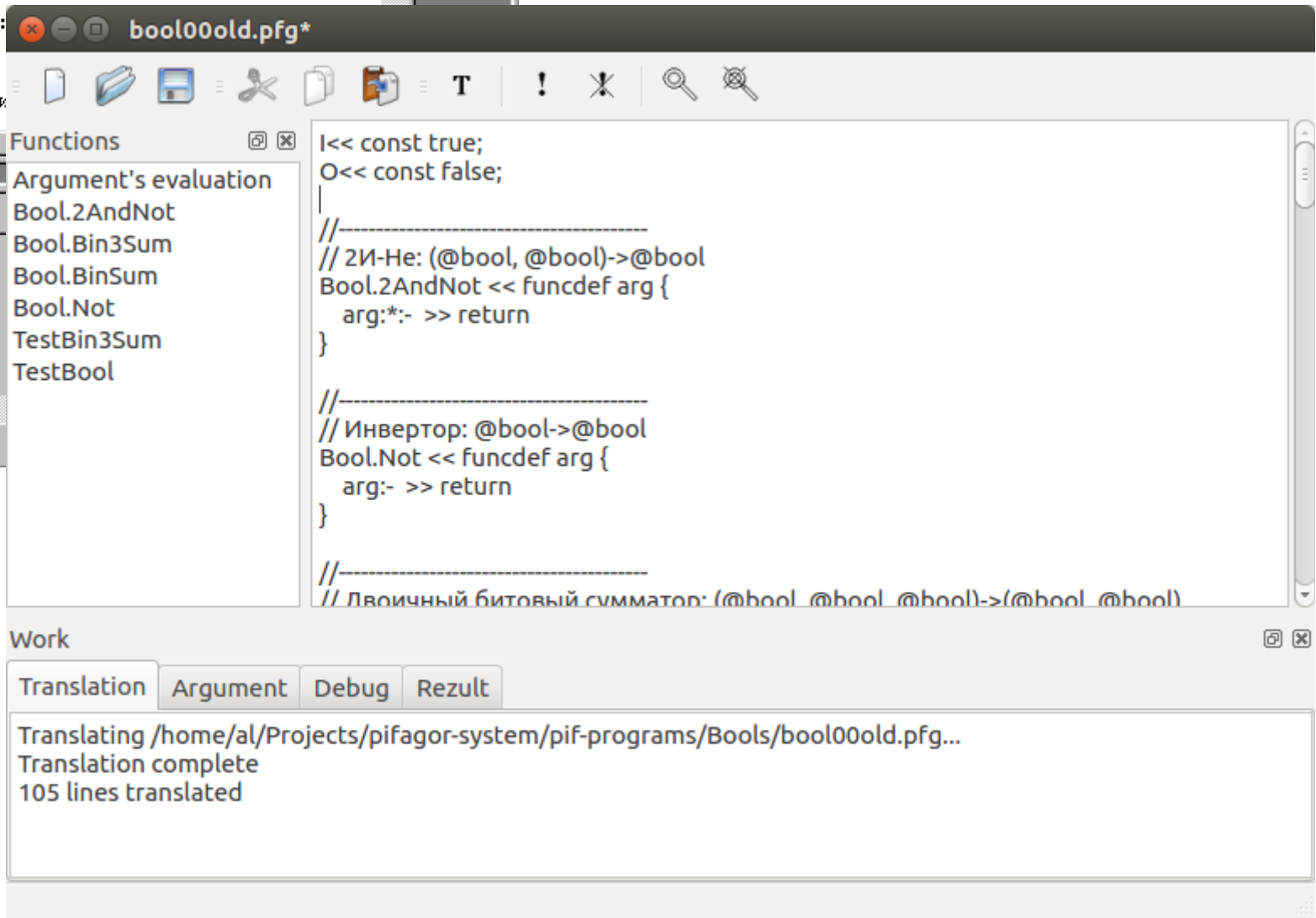
В программе:

```
{ (x, (x,1):-:fact ):* }:.
    => [ (x, (x,1):-:fact ):* ]:.
    => … == x*(x-1)!
```



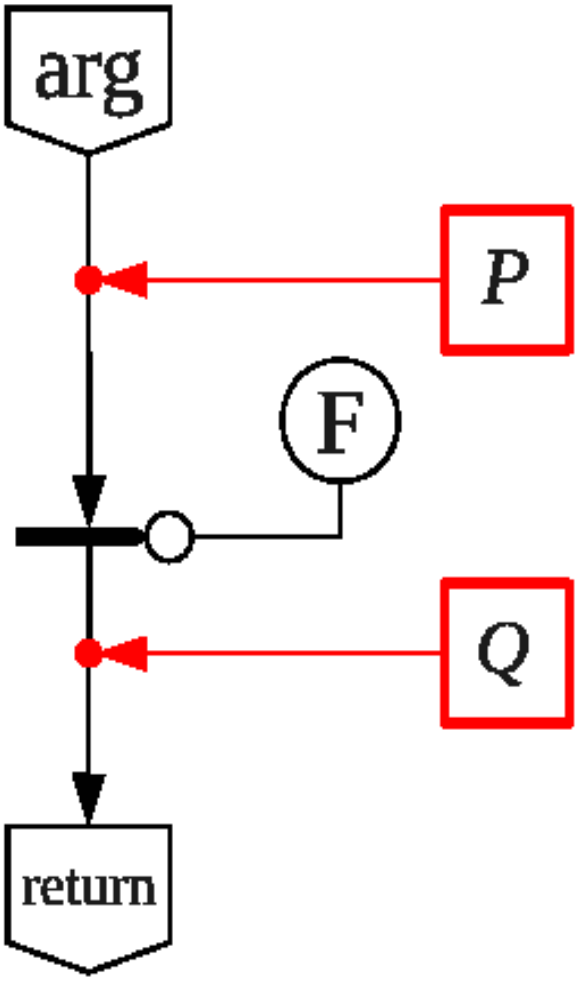

Отладчик
Программа
Интерпретатор
Компилятор
Преобразование в другие архитектуры
Формальная верификация
Реверсивный информационный граф (РИГ)
Оптимизация РИГ
Строгая типизация
Генератор управляющего графа
Оптимизация УГ
Управляющий граф (УГ)
Событийная машина + отладчик


Pifagor IDE - Proga
Файл Редактирование Вид Программа Окно Справка
VVMult
Fp
Proga
// Функция, возвращающая произведение двух векторов
VVMult << funcdef Param
{
(Param:#:[]:*):VSum:return
};
// Функция, возвращающая (в зависисмости от третьего элемента
// сумму или разность двух векторов
VVAddSub << funcdef Param
{
(Param:1,Param:2):#:[]:(Param:3):
};
// Функция, возвращающая (в зависи
Трансляция
Аргумент
Отладка
Результат
Трансляция...
Проверено строк : 323
Синтаксических ошибок не найдено
Для получения справки нажмите F1


bool00old.pfg*
Functions
Argument's evaluation
Bool.2AndNot
Bool.Bin3Sum
Bool.BinSum
Bool.Not
TestBin3Sum
TestBool
I<< const true;
O<< const false;
//---------------------------------------
// 2И-Не: (@bool, @bool)->@bool
Bool.2AndNot << funcdef arg {
arg:*:- >> return
}
//---------------------------------------
// Инвертор: @bool->@bool
Bool.Not << funcdef arg {
arg:- >> return
}
//---------------------------------------
Work
Translation
Argument
Debug
Rezult
Translating /home/al/Projects/pifagor-system/pif-programs/Bools/bool00old.pfg...
Translation complete
105 lines translated

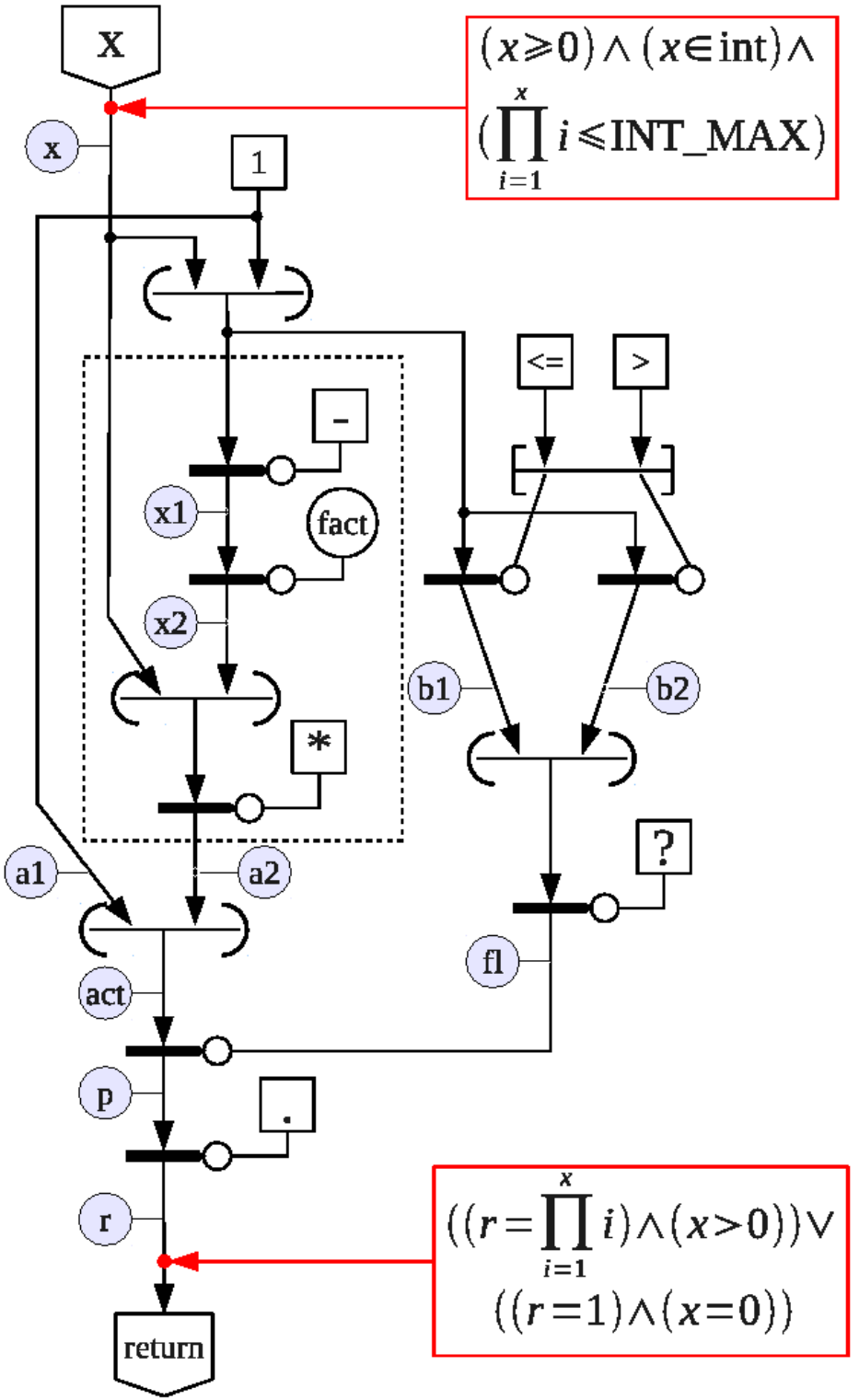
```
Fun << funcdef arg {
    arg:F >> return
}
```

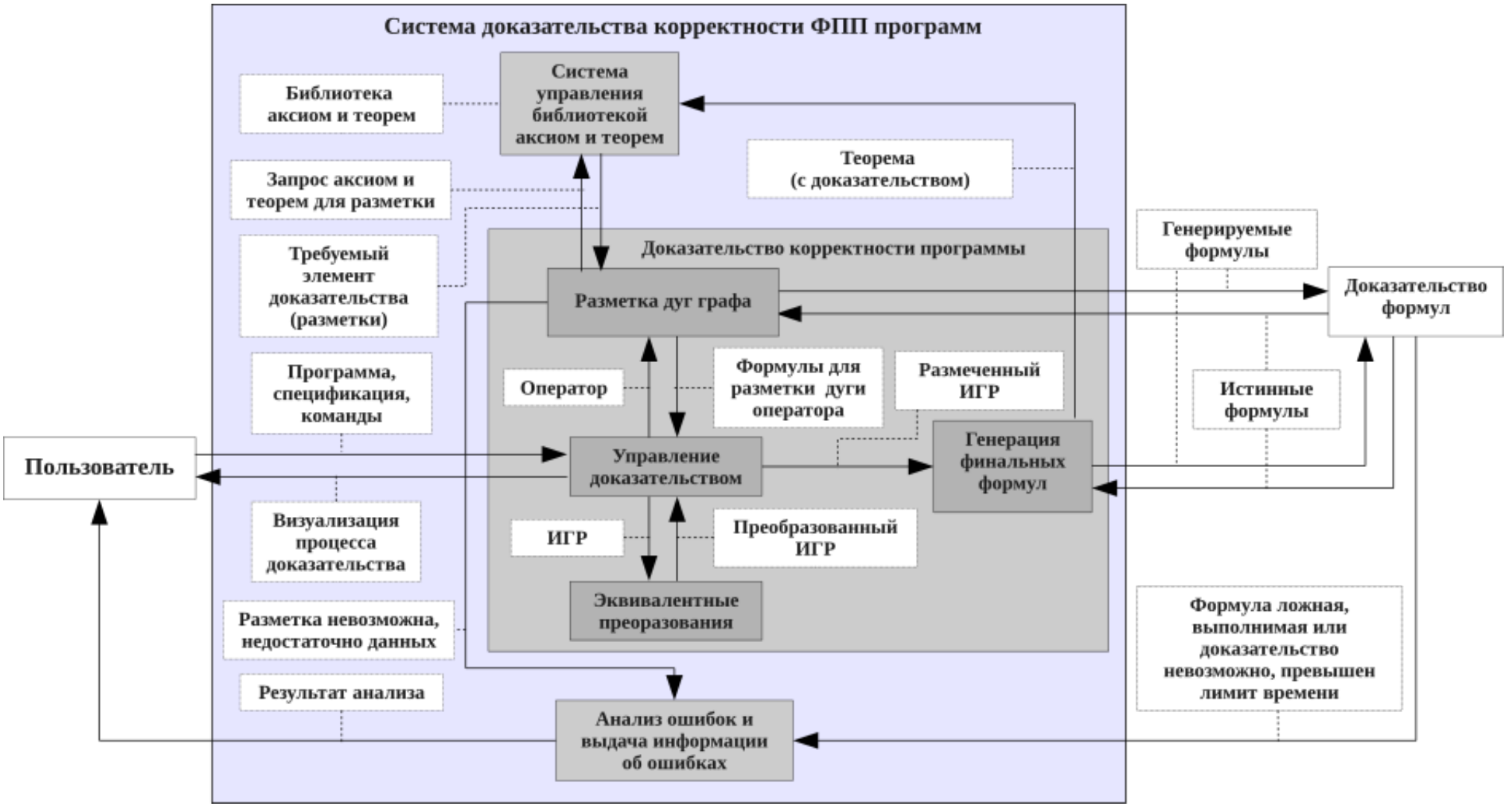
$$\boxed{P(arg)}\ \texttt{arg:F}\rightarrow r\ \boxed{Q(r)}$$



```
fact << funcdef x {
  fl << ((x,1):[<=,>]):?;
  Act << (
    1,
    { (x, (x,1):-:fact ):* }
  );
  return << act:fl:.;
}
```

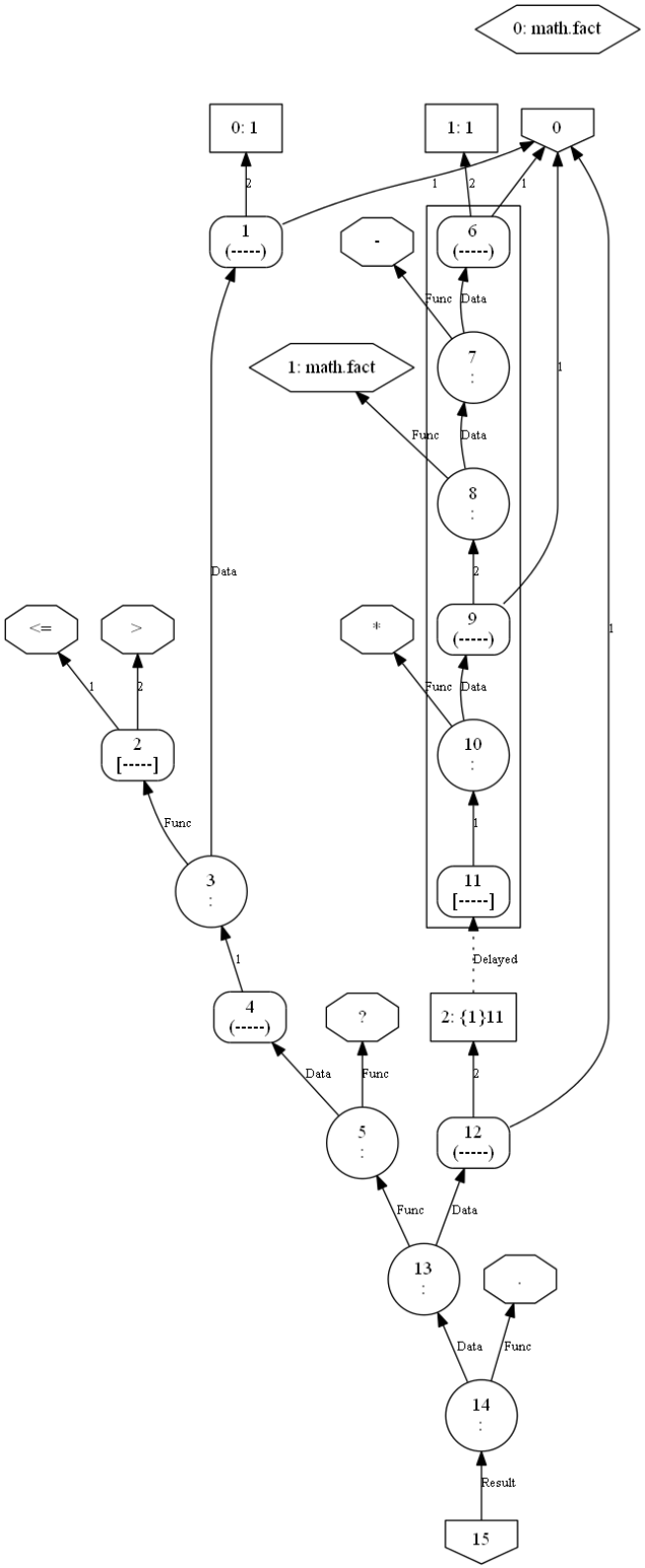
$$(x \in int) \wedge (x \geqslant 0) \wedge \prod_{i=1}^{x} i \leqslant \texttt{INT\_MAX} \quad \texttt{x:fact} \rightarrow r \quad ((r = \prod_{i=1}^{x} i) \wedge (x > 0)) \vee ((r = 1) \wedge (x = 0))$$


X
$(x \geqslant 0) \wedge (x \in \text{int}) \wedge$
$(\prod_{i=1}^{x} i \leqslant \text{INT\_MAX})$
x
1
<=
>
-
x1
fact
x2
*
b1
b2
?
a1
a2
fl
act
p
.
r
return
$((r = \prod_{i=1}^{x} i) \wedge (x > 0)) \vee$
$((r = 1) \wedge (x = 0))$


Система доказательства корректности ФПП программ
Библиотека аксиом и теорем
Система управления библиотекой аксиом и теорем
Теорема (с доказательством)
Запрос аксиом и теорем для разметки
Требуемый элемент доказательства (разметки)
Доказательство корректности программы
Разметка дуг графа
Генерируемые формулы
Доказательство формул
Программа, спецификация, команды
Оператор
Формулы для разметки дуги оператора
Размеченный ИГР
Истинные формулы
Пользователь
Управление доказательством
Генерация финальных формул
Визуализация процесса доказательства
ИГР
Преобразованный ИГР
Эквивалентные преоразования
Разметка невозможна, недостаточно данных
Формула ложная, выполнимая или доказательство невозможно, превышен лимит времени
Результат анализа
Анализ ошибок и выдача информации об ошибках

```
math.fact<<funcdef X
{
fl << ((X,1):[<=,>]):?;
act << (X,{ (X, (X,1):-:math.fact ):*  } );
return << act:fl:.;
}
```

```
External
    0   math.fact
    1   math.fact
Local
    0   1
    1   1
    2   {1}11
id  delay   operation  links      positions
0   0   arg            pos 1 20 1 21
1   0   (---)      0 loc:0    pos 3 8 3 13
2   0   [---]      <= >       pos 3 14 3 20
3   0   :       1 2     pos 3 13 3 14
4   0   (---)       3       pos 3 7 3 21
5   0   :       4 ?     pos 3 21 3 22
6   1   (---)       0 loc:1    pos 4 17 4 22
7   1   :       6 -     pos 4 22 4 23
8   1   :       7 ext:1     pos 4 24 4 25
9   1   (---)       0 8     pos 4 13 4 36
10  1   :       9 *     pos 4 36 4 37
11  1   [---]       10      pos 4 11 4 41
12  0   (---)       0 loc:2    pos 4 8 4 43
13  0   :       12 5        pos 5 14 5 15
14  0   :       13 .        pos 5 17 5 18
15  0   return      14      pos 5 1 5 7
```



math.fact

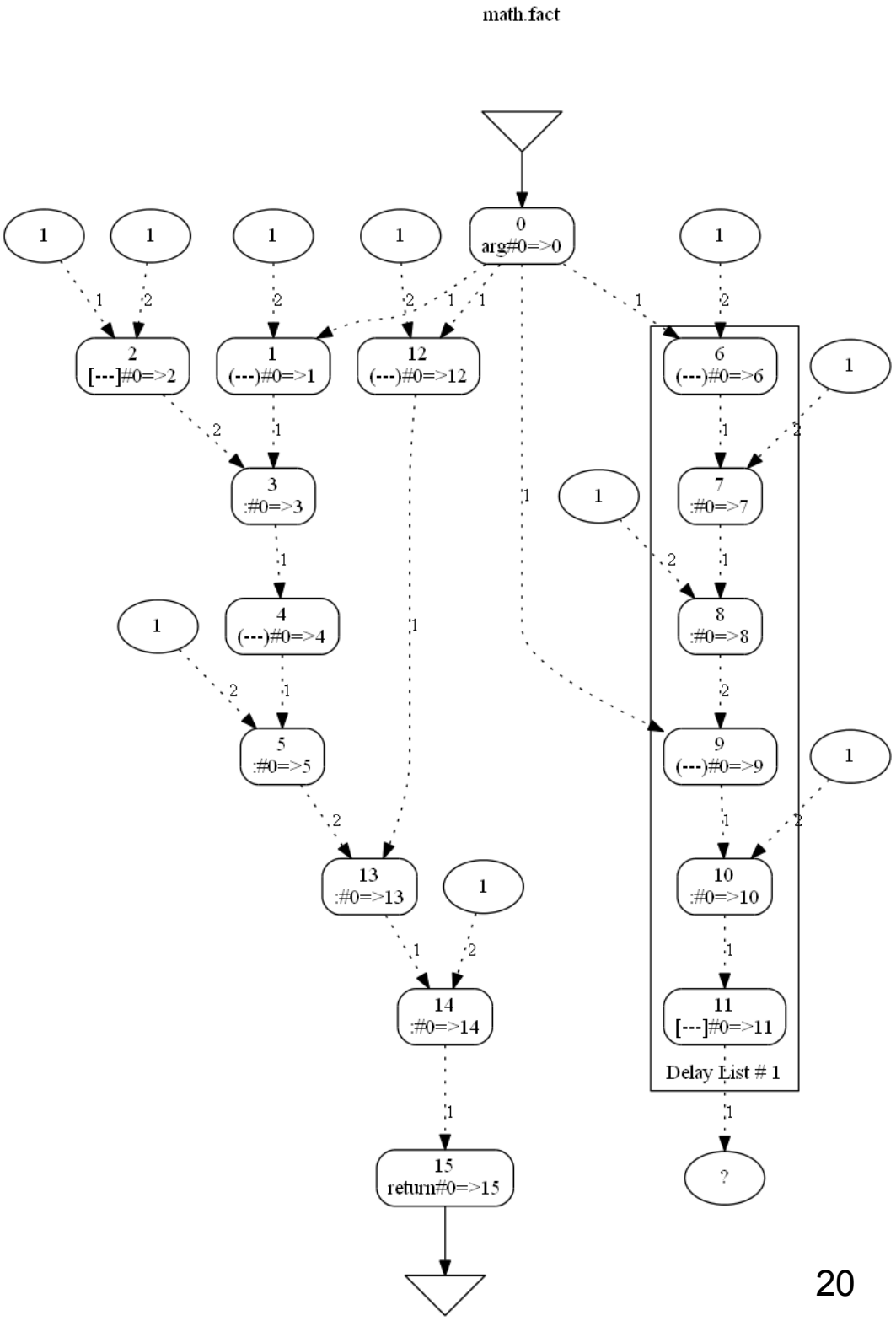
id	delay	automat	inode	links
0	0	arg,0	0	links:1,1;6,1;9,1;12,1;
1	0	(---),0	1	links:3,1;
2	0	[---],0	2	links:3,2;
3	0	:,0	3	links:4,1;
4	0	(---),0	4	links:5,1;
5	0	:,0	5	links:13,2;
6	1	(---),0	6	links:7,1;
7	1	:,0	7	links:8,1;
8	1	:,0	8	links:9,2;
9	1	(---),0	9	links:10,1;
10	1	:,0	10	links:11,1;
11	1	[---],0	11	
12	0	(---),0	12	links:13,1;
13	0	:,0	13	links:14,1;
14	0	:,0	14	links:15,1;
15	0	return,0	15	

Signals: (Number, Node, Input)

0	1	2
1	2	1
2	2	2
3	5	2
4	6	2
5	7	2
6	8	2
7	10	2
8	12	2
9	14	2

Dynamic links: (Number, Delay list, Node)

0	1	11



Благодарю за внимание!