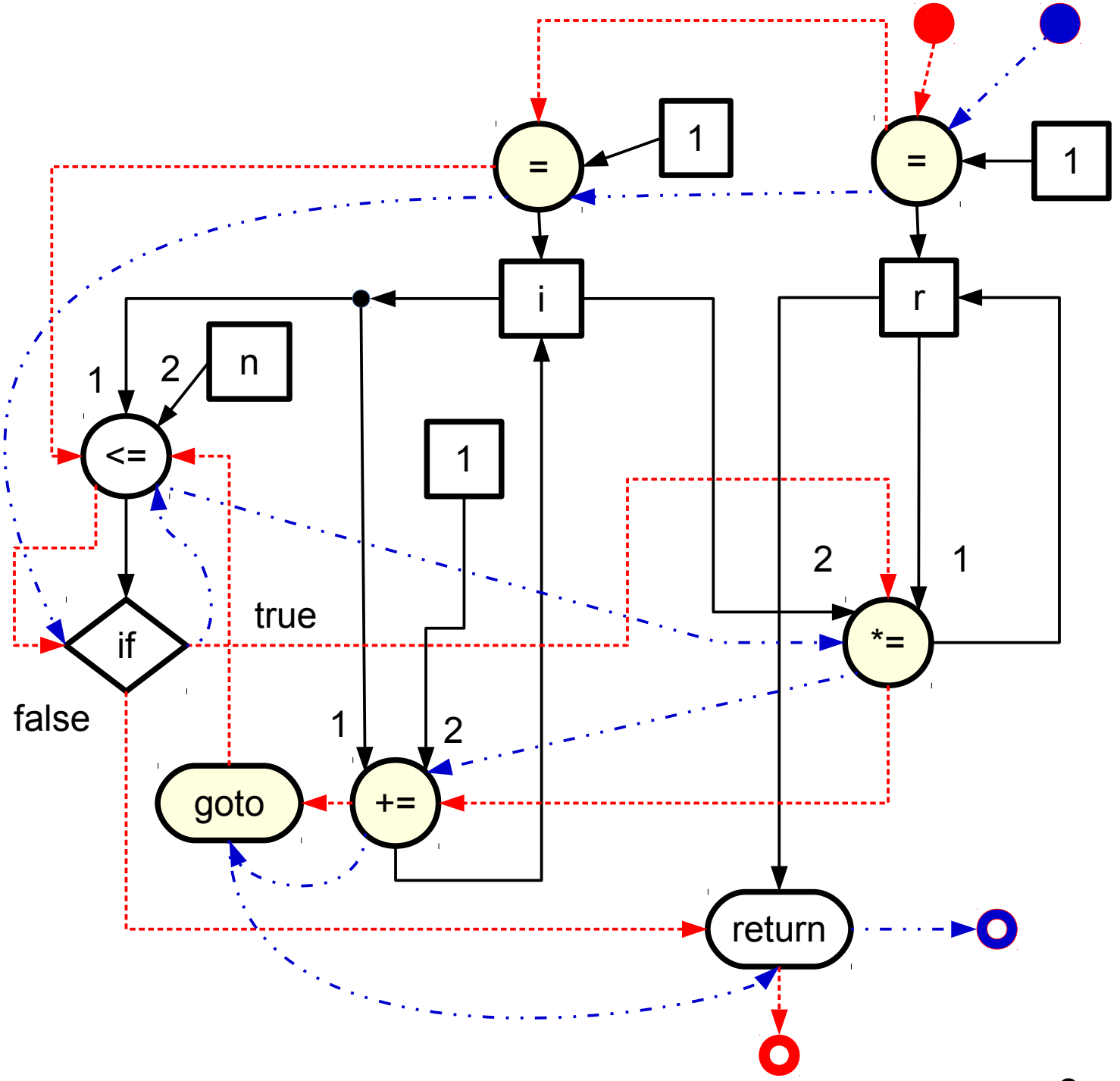
Изменение стратегий управления вычислениями при архитектурно-независимом параллельном программировании

Александр Легалов, Владимир Васильев, Иван Матковский

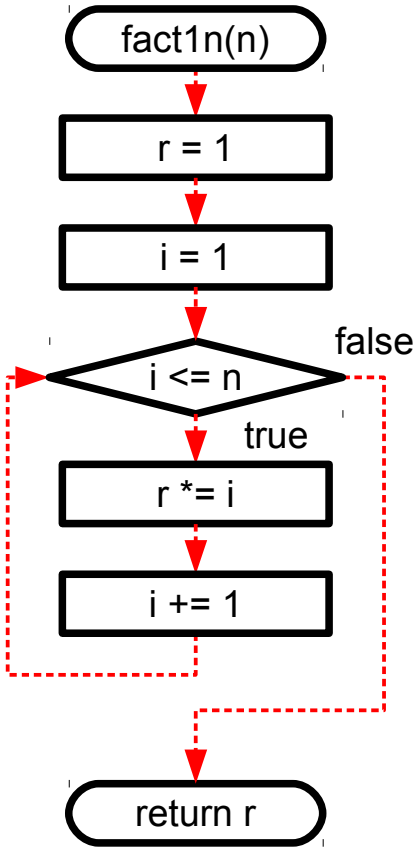
Сибирский федеральный университет

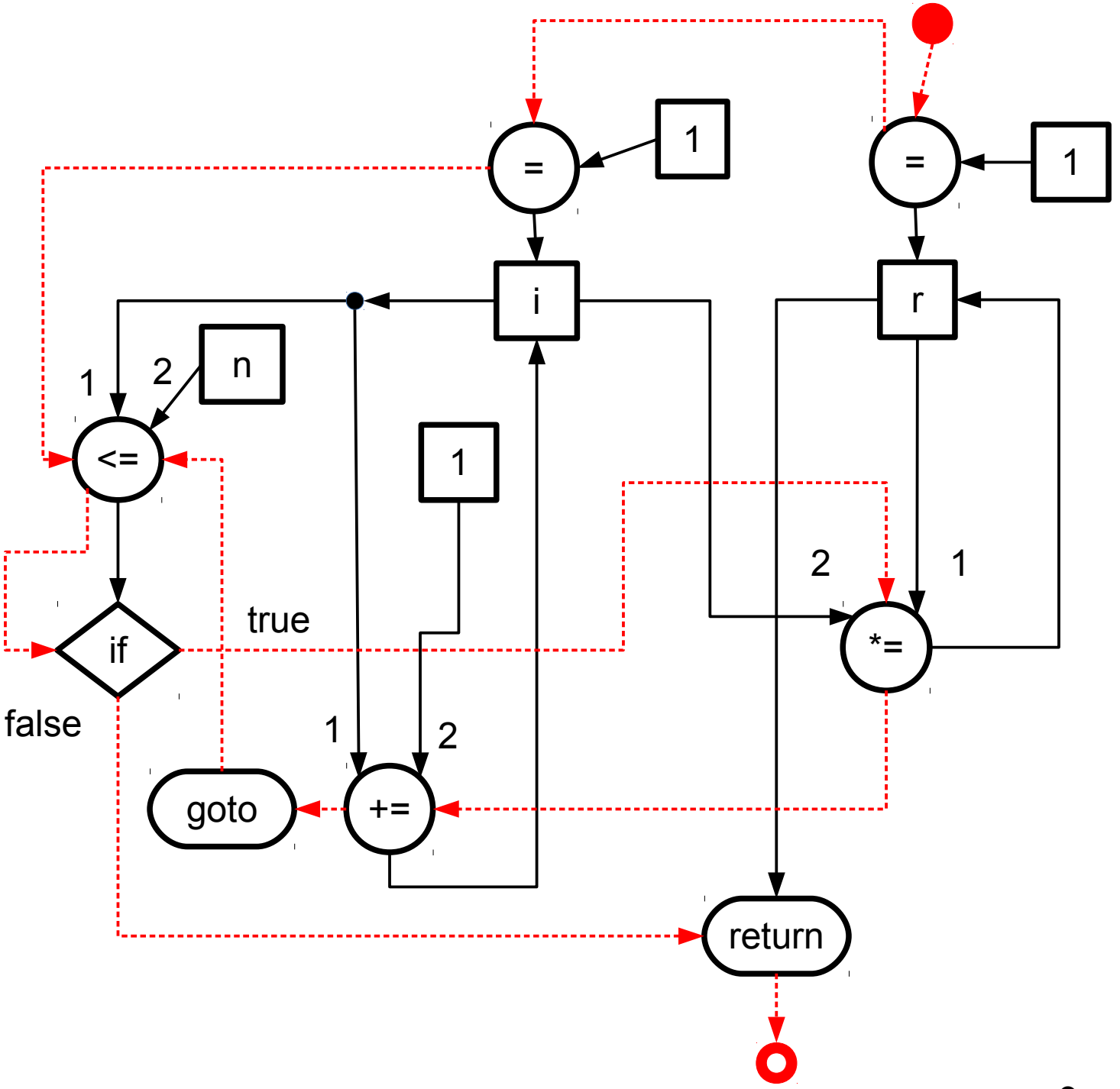
Исследование выполняется при финансовой поддержке РФФИ в рамках научного проекта № 17-07-00288

```
// Факториал от 1 до n
int fact1n(int n)
{
    int r = 1;
    int i = 1;
loop:
    if(i <= n) {
        r *= i;
        i++;
        goto loop;
    }
    return r;
}
```

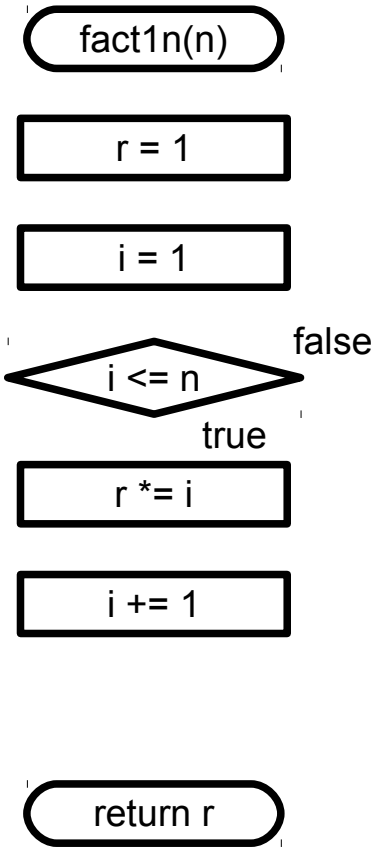


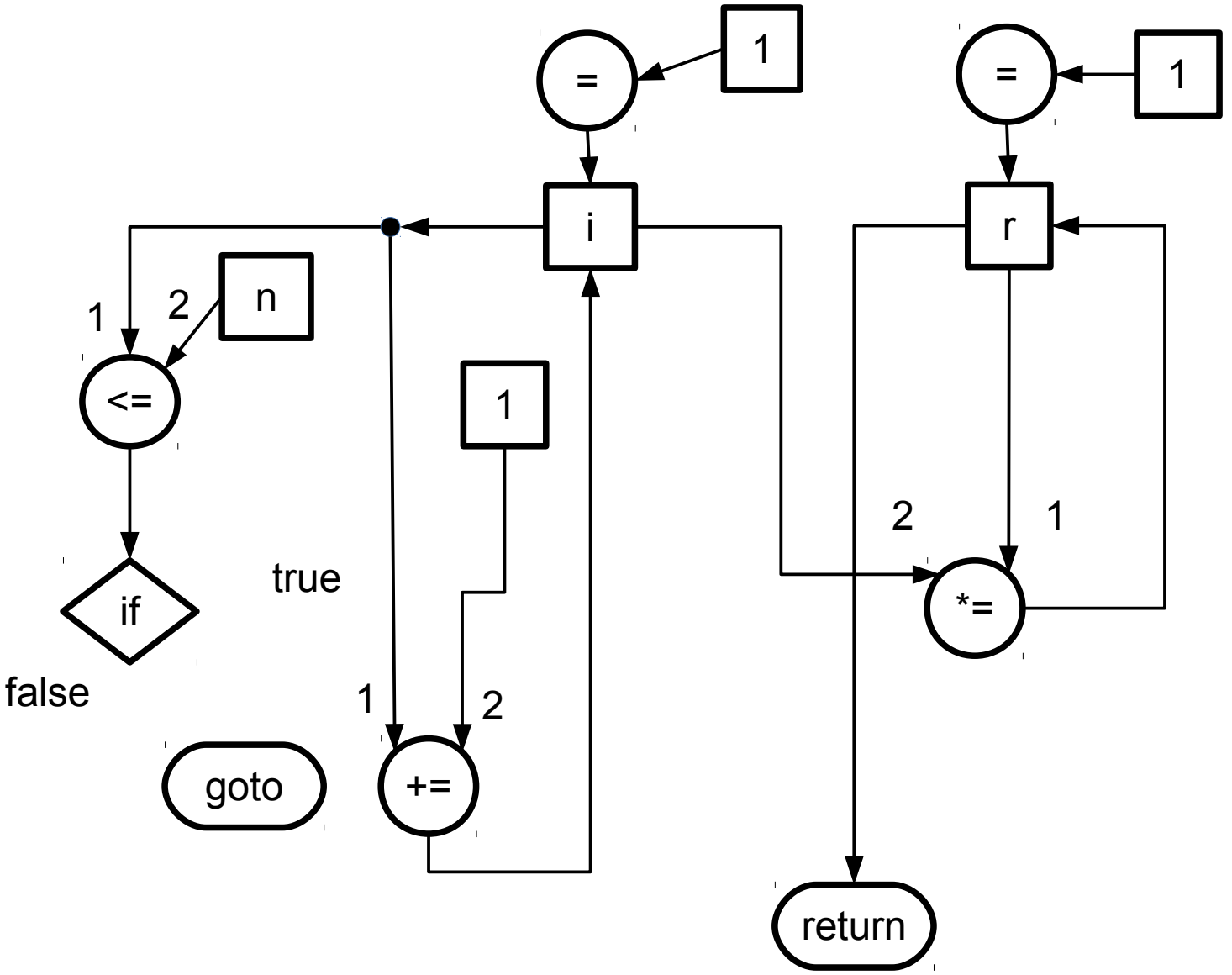
```c
// Факториал от 1 до n
int fact1n(int n)
{
    int r = 1;
    int i = 1;
loop:
    if(i <= n) {
        r *= i;
        i++;
        goto loop;
    }
    return r;
}
```



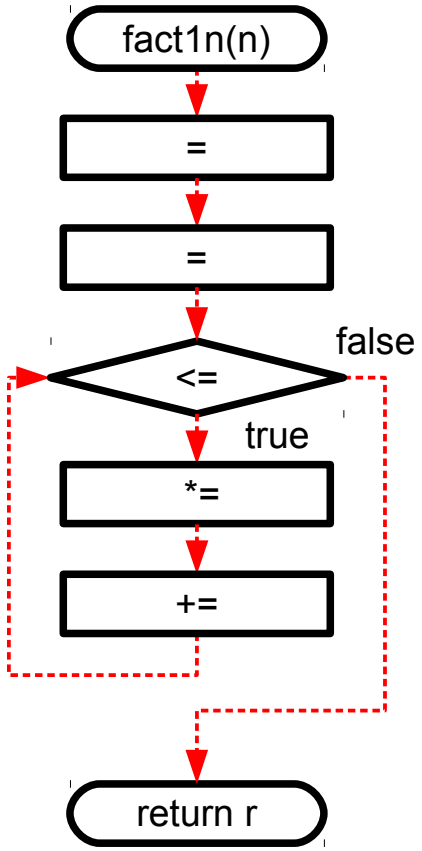


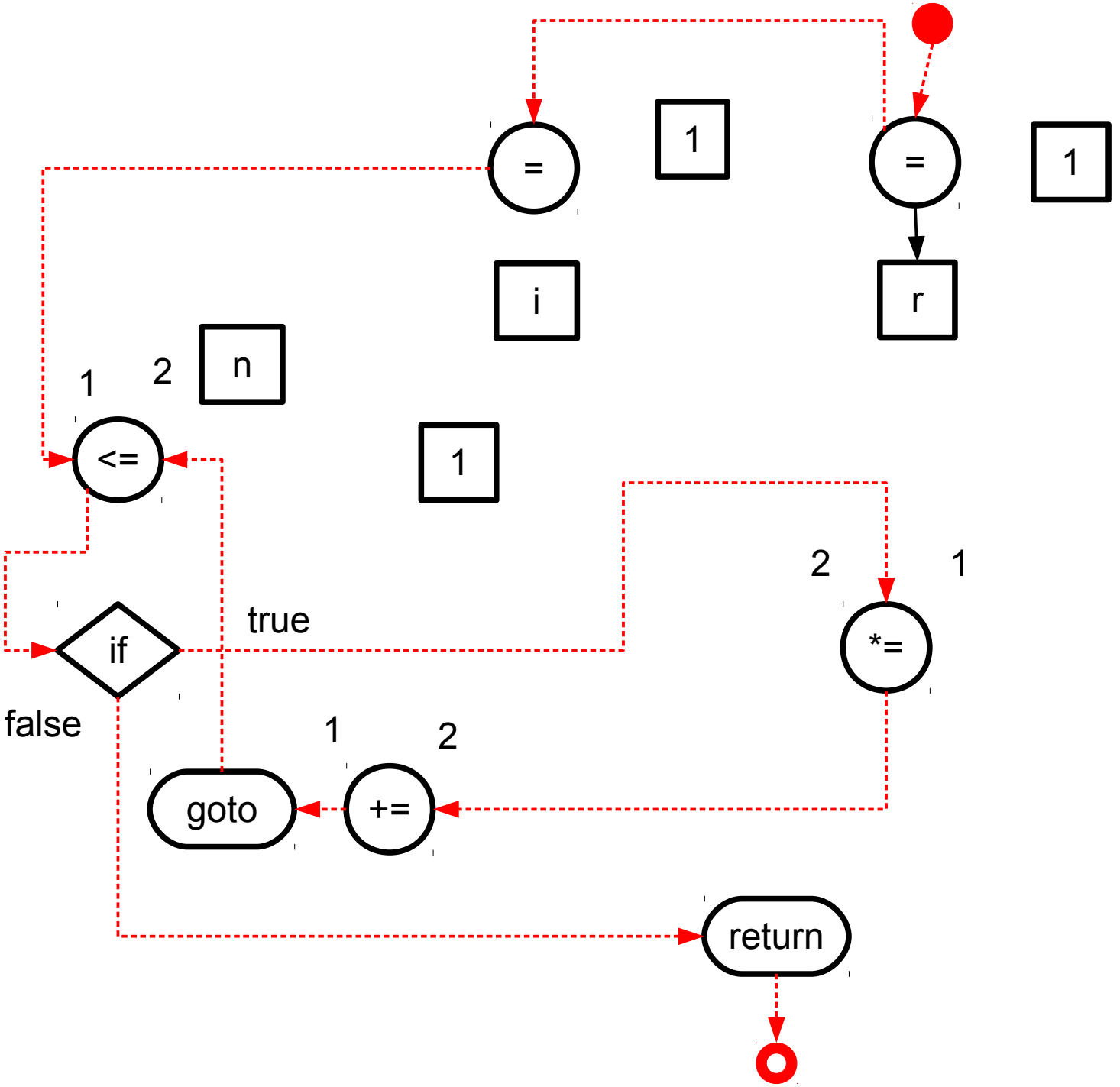
```
// Факториал от 1 до n
int fact1n(int n)
{
    int r = 1;
    int i = 1;
loop:
    if(i <= n) {
        r *= i;
        i++;
        goto loop;
    }
    return r;
}
```



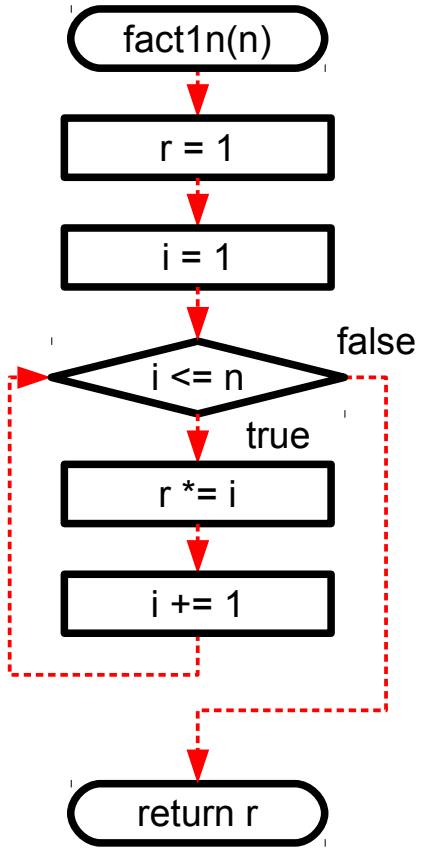


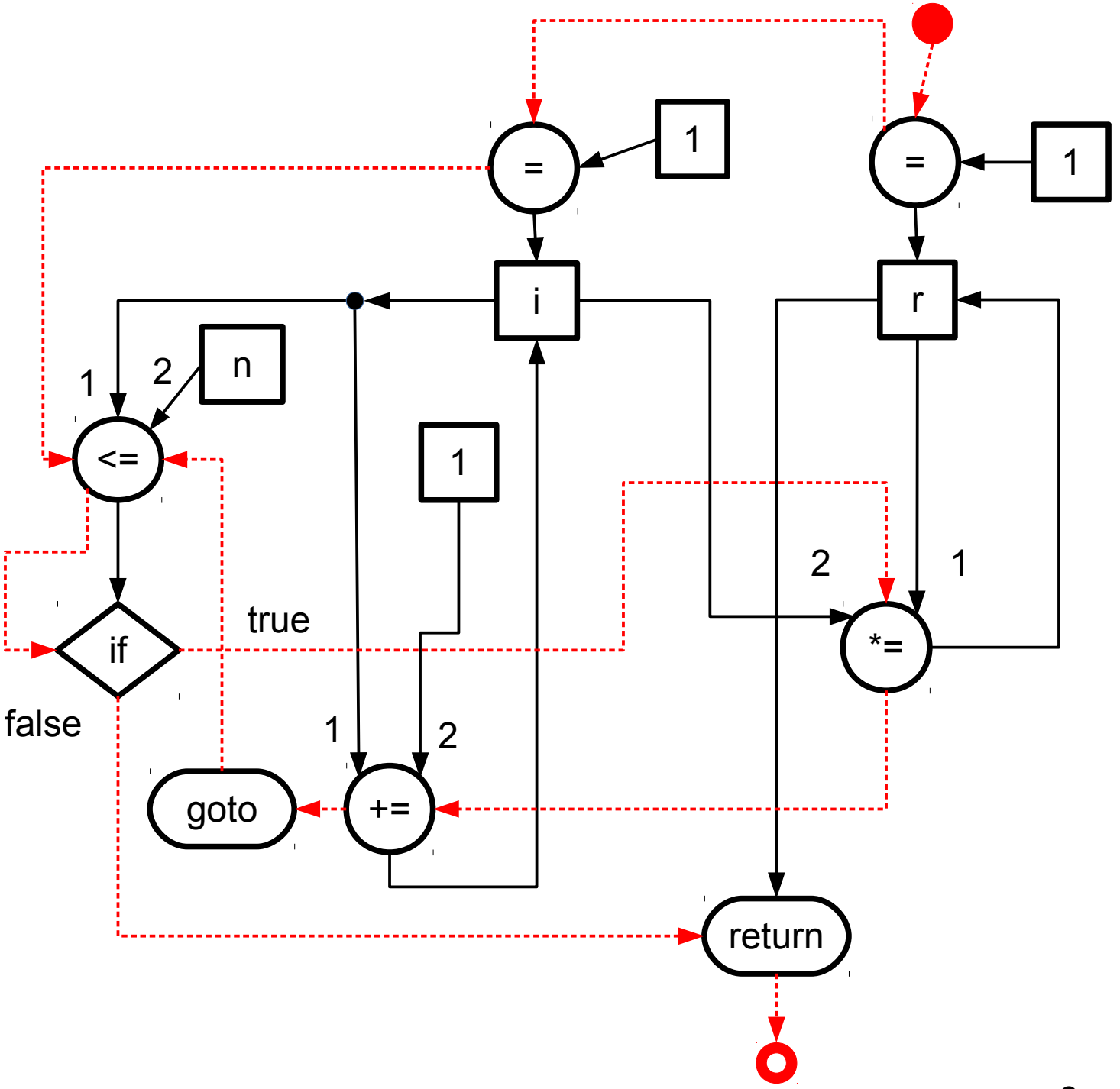
```
// Факториал от 1 до n
int fact1n(int n)
{
    int r = 1;
    int i = 1;
loop:
    if(i <= n) {
        r *= i;
        i++;
        goto loop;
    }
    return r;
}
```



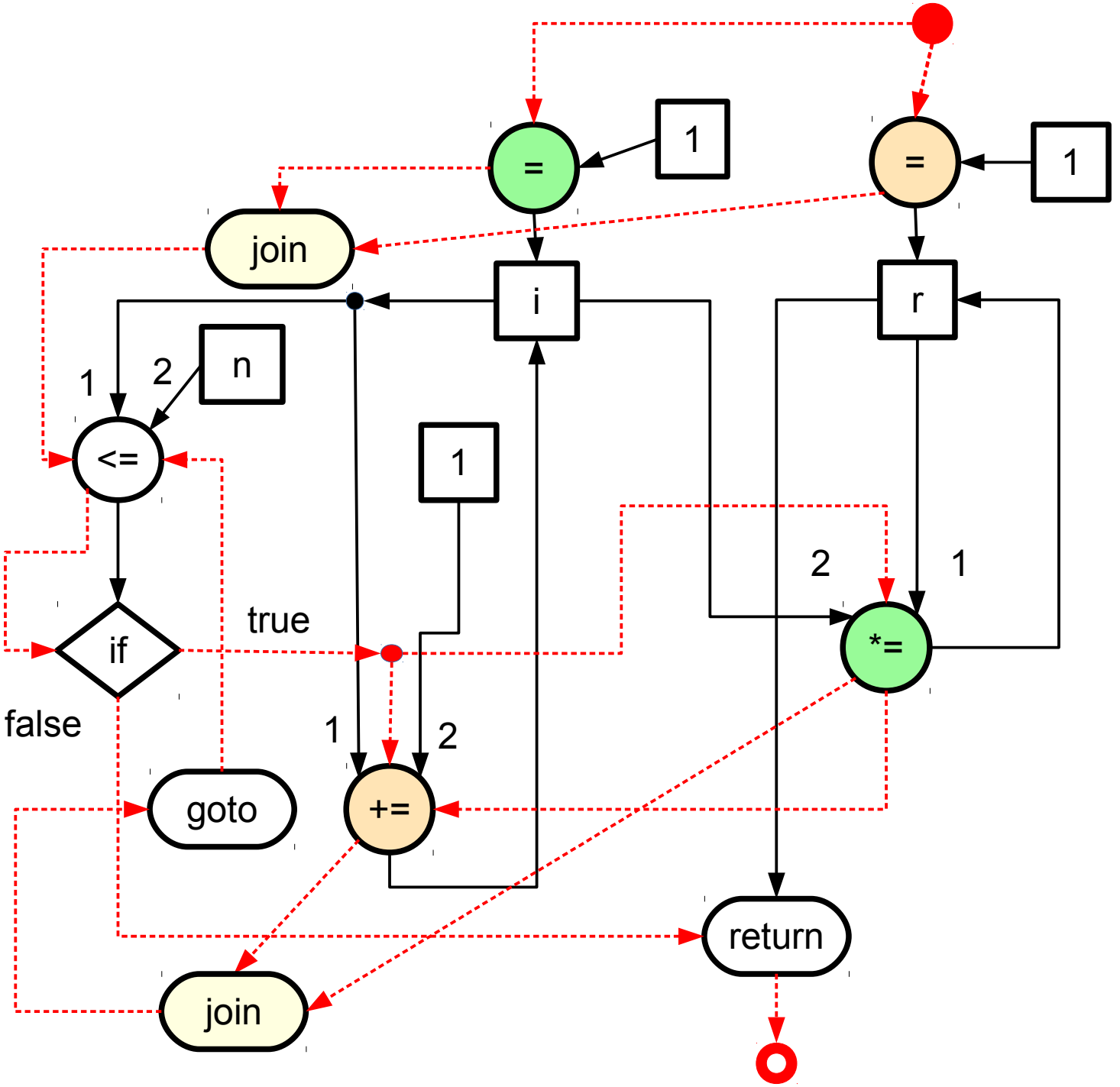


```c
// Факториал от 1 до n
int fact1n(int n)
{
    int r = 1;
    int i = 1;
loop:
    if(i <= n) {
        r *= i;
        i++;
        goto loop;
    }
    return r;
}
```





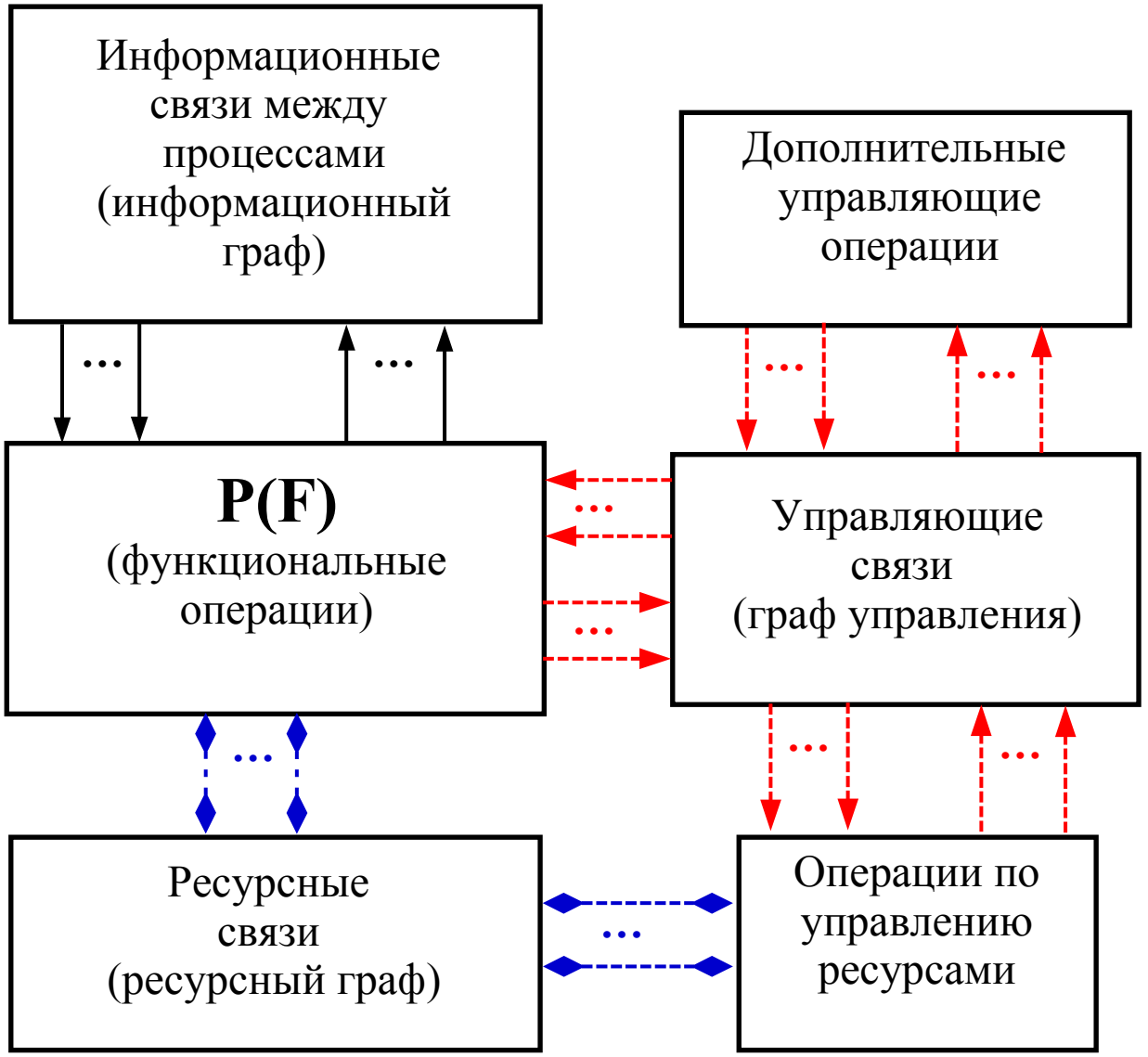
```
// Факториал от 1 до n
int fact1n(int n)
{
  fork
    int r = 1;
    int i = 1;
  join
loop:
    if(i <= n) {
      fork
        r *= i;
        i++;
      join
      goto loop;
    }
    return r;
}
```



Легалов А.И. Об управлении вычислениями в параллельных системах и языках программирования – Научный вестник НГТУ, № 3 (18), 2004. С. 63-72.

Легалов А. И. Стратегии управления в вычислительных системах и языках программирования - 2002 г. Электронный ресурс: http://softcraft.ru/parallel/strat/

ICR-сеть.
Описание поведения процессов в ресурсах вычислительной системы



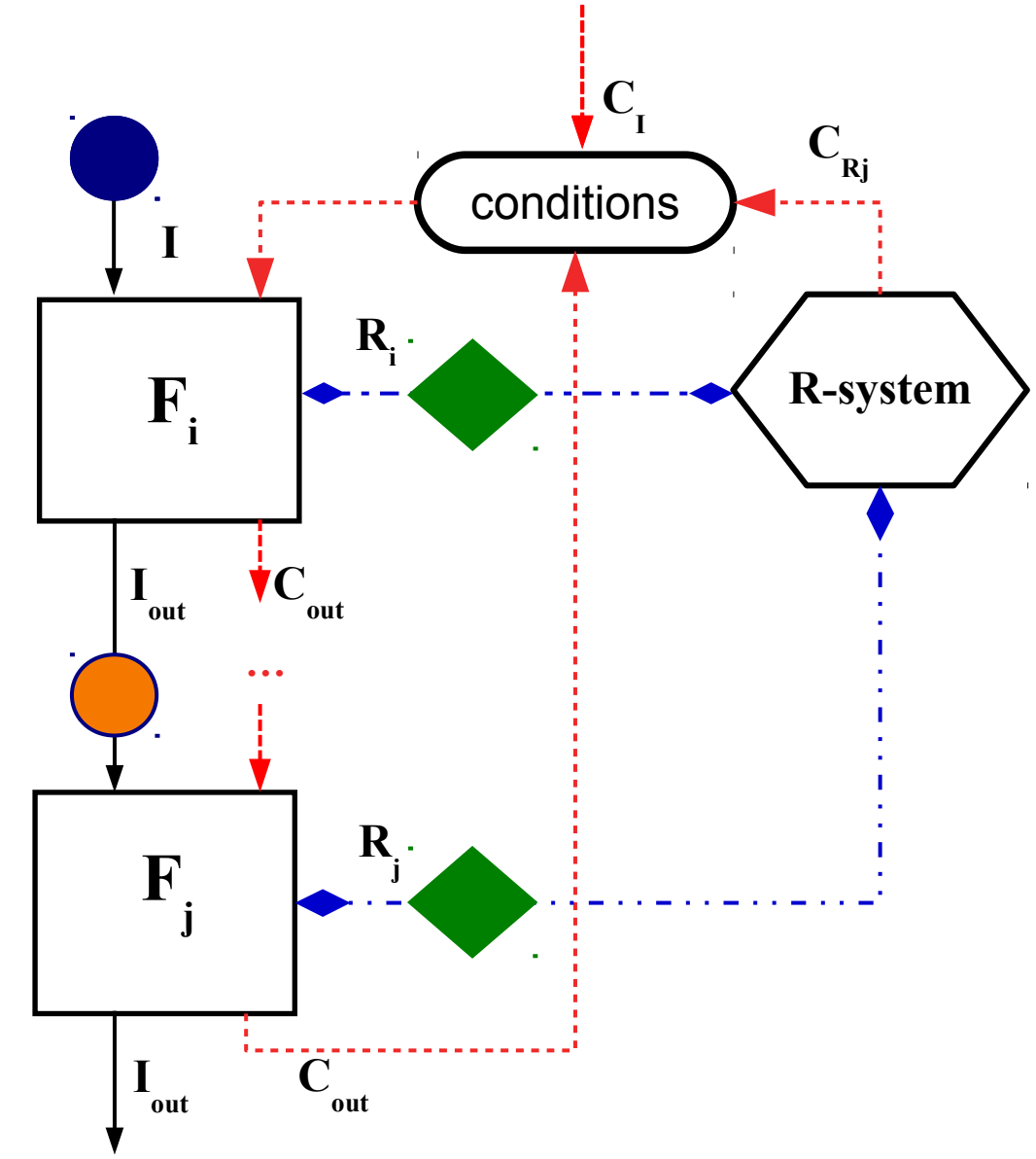
Стратегии управления в вычислительных системах

Условия готовности вычислений

Условие готовности данных
(Information, I-условие).
Перед запуском процесса на его информационных входах должны присутствовать все необходимые аргументы и функции, составляющие операционный пакет. Отсутствие каких-либо данных к моменту запуска процесса ведет к получению неправильного результата.

Условие выделения ресурсов
(Resource, R-условие).
Выполнение процесса протекает в соответствующих ресурсах, поэтому, перед его запуском, эти ресурсы должны быть определены и предоставлены.

Условие подтверждения
(Acknowledge, A- условие).
Ресурсы, занимаемые процессом, могут освобождаться или повторно использоваться только после того, как результаты вычислений будут использованы всеми процессами, являющихся приемниками этих результатов в качестве аргументов.

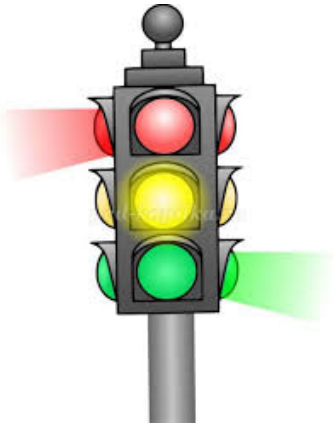


Стратегии управления в вычислительных системах

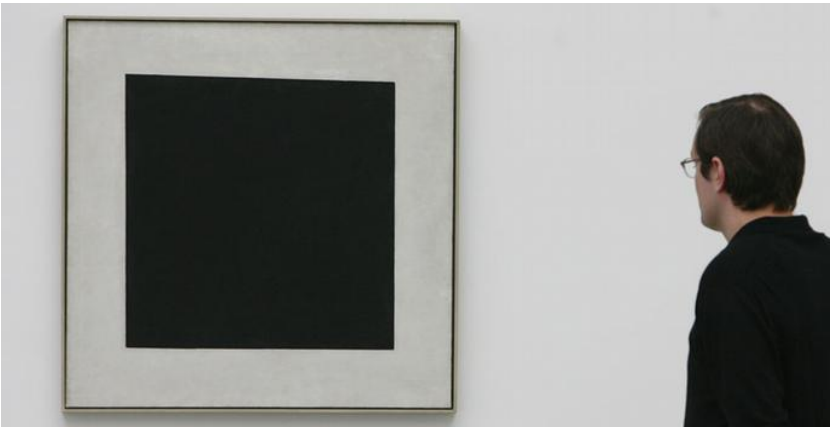
Способы задания управления

Явный (субъективный) (**Subjective**)
В ходе написания программы разработчик сам создает логику порождения и проверки условий готовности

Неявный
Управление тем или иным условием готовности не задается из расчета, что процессы и так будут выполняться корректно.



- *Автоматический* (**Automatic**)
Ресурсы ВС организованы таким образом, что всегда поддерживают выполнение заданного условия.



- *Пустой* (**Empty**)
Описание взаимодействия процессов может быть сделано так, что, несмотря на отсутствие в ВС механизма автоматической поддержки, условие готовности всегда истинно, поэтому его и не нужно проверять.

Стратегии управления в вычислительных системах

Множество условий готовности:

$$X = (Information, Resource, Acknowledge)$$

Множество способов управления, обеспечивающих соблюдение условий готовности:

$$Y = \{Subjective, Automatic, Empty\}$$

S_1={Information×Y} – множество методов управления данными;
S_2={Resource×Y} – множество методов управления ресурсами;
S_3={Acknowledge×Y} – множество методов управления подтверждениями.

Множество стратегий управления в ВС:

$$S=\{S_1 \times S_2 \times S_3\} = 27$$

Стратегии управления в языках программирования

Множество условий готовности:

X = (Information, Resource, Acknowledge)

Множество способов управления, обеспечивающих соблюдение условий готовности:

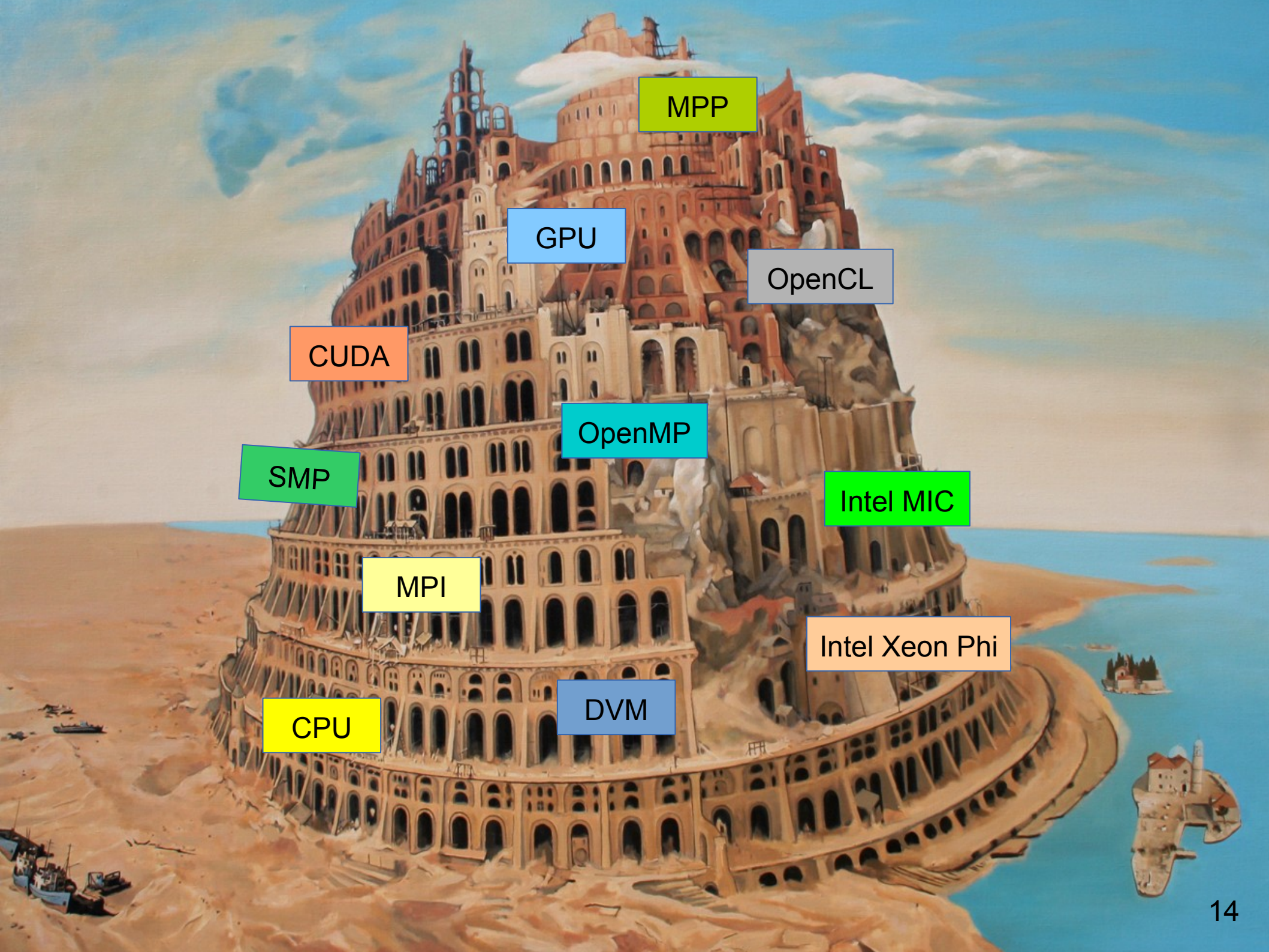
Automatic + Empty = Unevident - неявный

Z = {Subjective, Unevident}

S_1={Information×Z} – множество методов управления данными;
S_2={Resource×Z} – множество методов управления ресурсами;
S_3={Acknowledge×Z} – множество методов управления подтверждениями.

Множество стратегий управления в ВС:

$$S=\{S_1\times S_2\times S_3\} = 8$$


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

Архитектурно-независимое параллельное программирование

Общая стратегия управления: Unevident Information, Unevident Resource, Unevident Acknowledge

(Empty Information, Empty Resource, Empty Acknowledge)-машина

Виртуальный программируемый вычислитель должен:
- обладать неограниченными вычислительными ресурсами;
- обеспечивать поддержку автоматического управления данными и неограниченными ресурсами (при этом предполагается, что ресурсы могут только предоставляться, но не забираться обратно);
- не поддерживать управление подтверждениями, обеспечивая, вместо этого, предоставление новых ресурсов каждому независимому набору данных.

То есть, поддерживать стратегию для систем типа:

Automatic Information, Automatic Resource, Empty Acknowledge

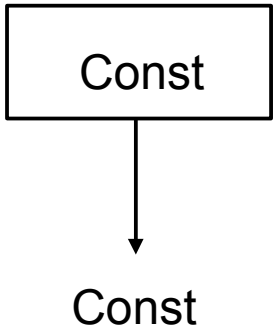
Не искажать информационный граф программы!

Модель функционально-потоковых параллельных вычислений

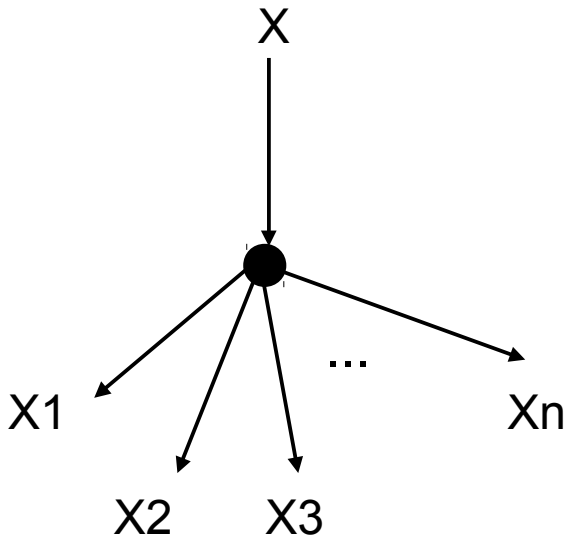
- Ориентация на бесконечные вычислительные ресурсы (принцип единственного использования вычислительных ресурсов).
- Запуск операции по готовности данных. Определяется в соответствии с аксиоматикой языка и его алгеброй преобразований (неявное управление вычислениями).
- Программа не имеет циклов, а значит 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.

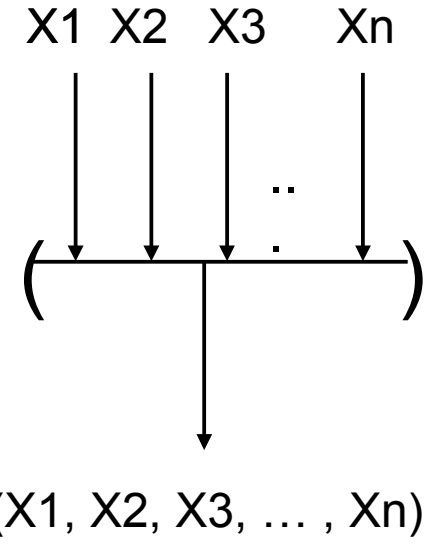
Основные операторы модели вычислений



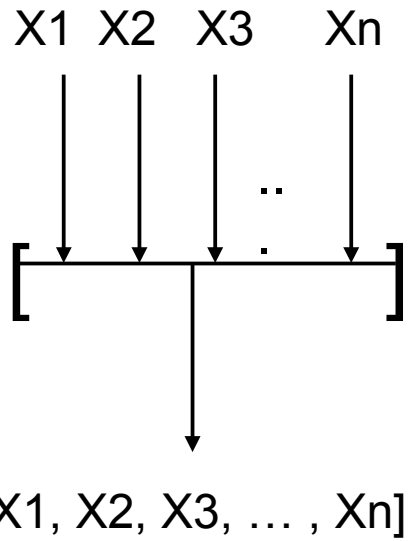
Константный оператор



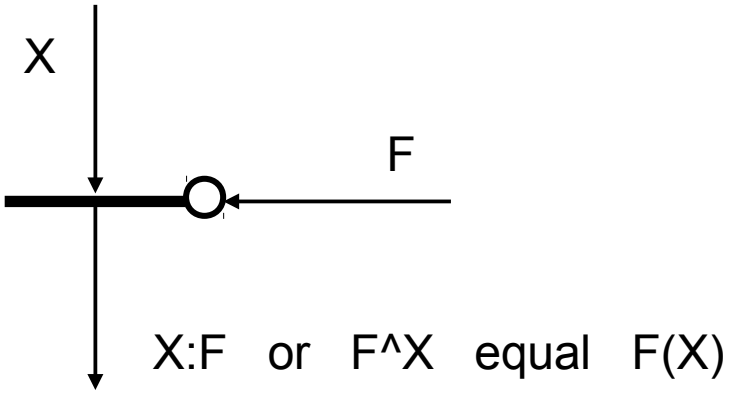
Копирование данных



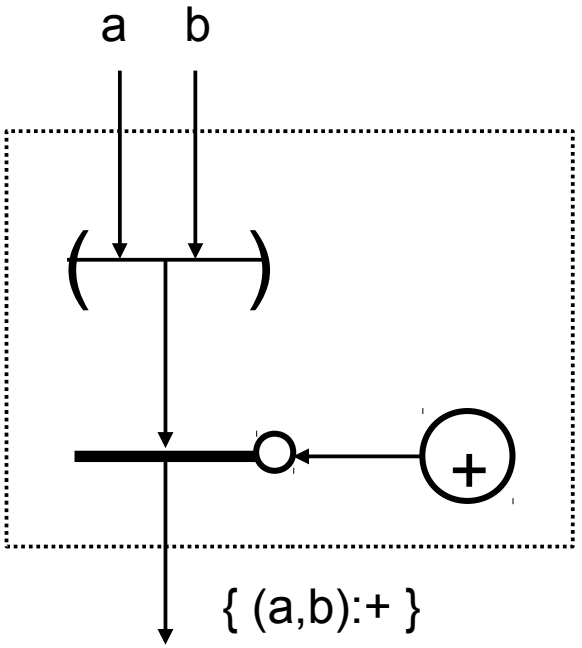
Список данных



Параллельный список

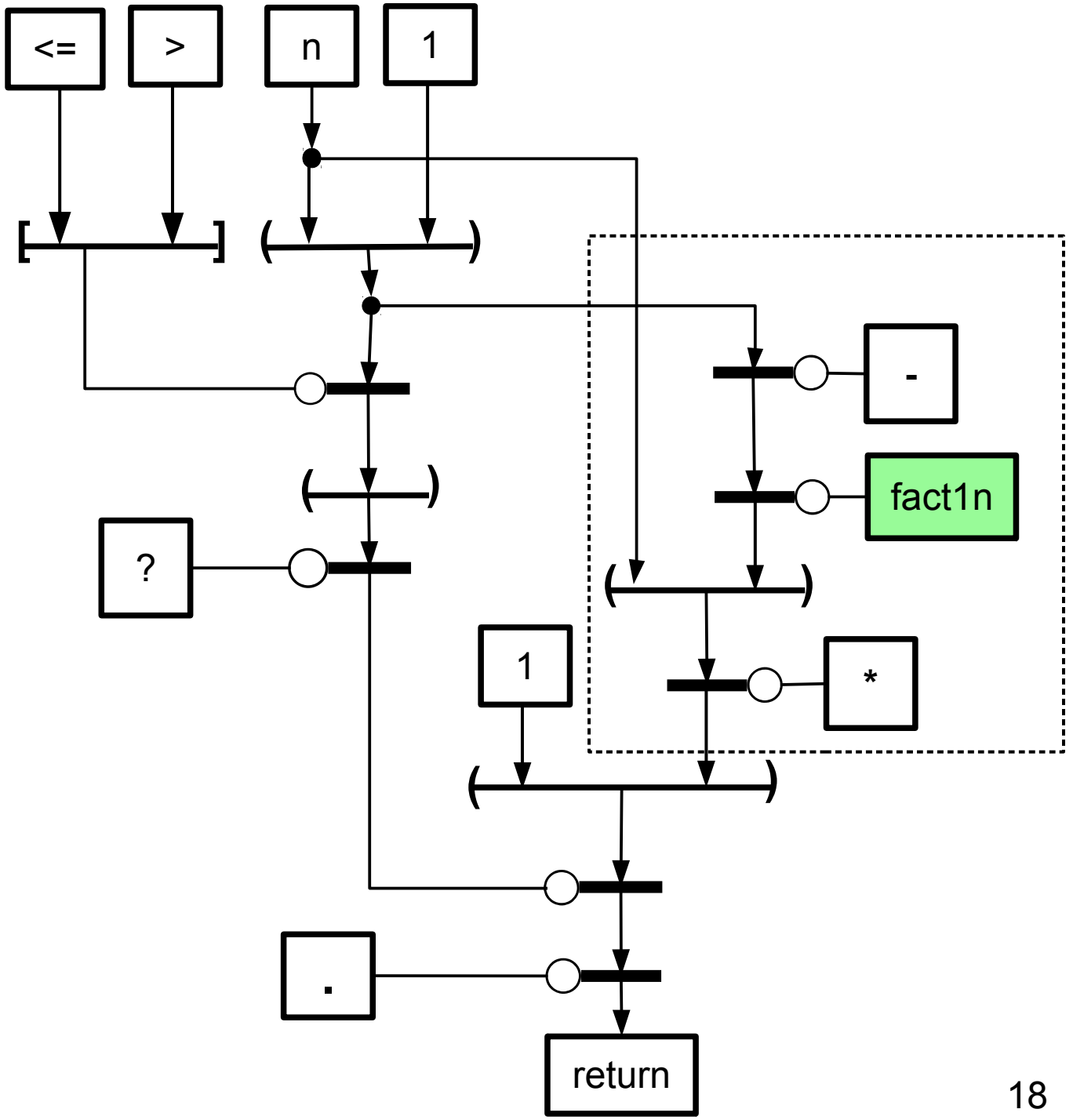


Оператор интерпретации

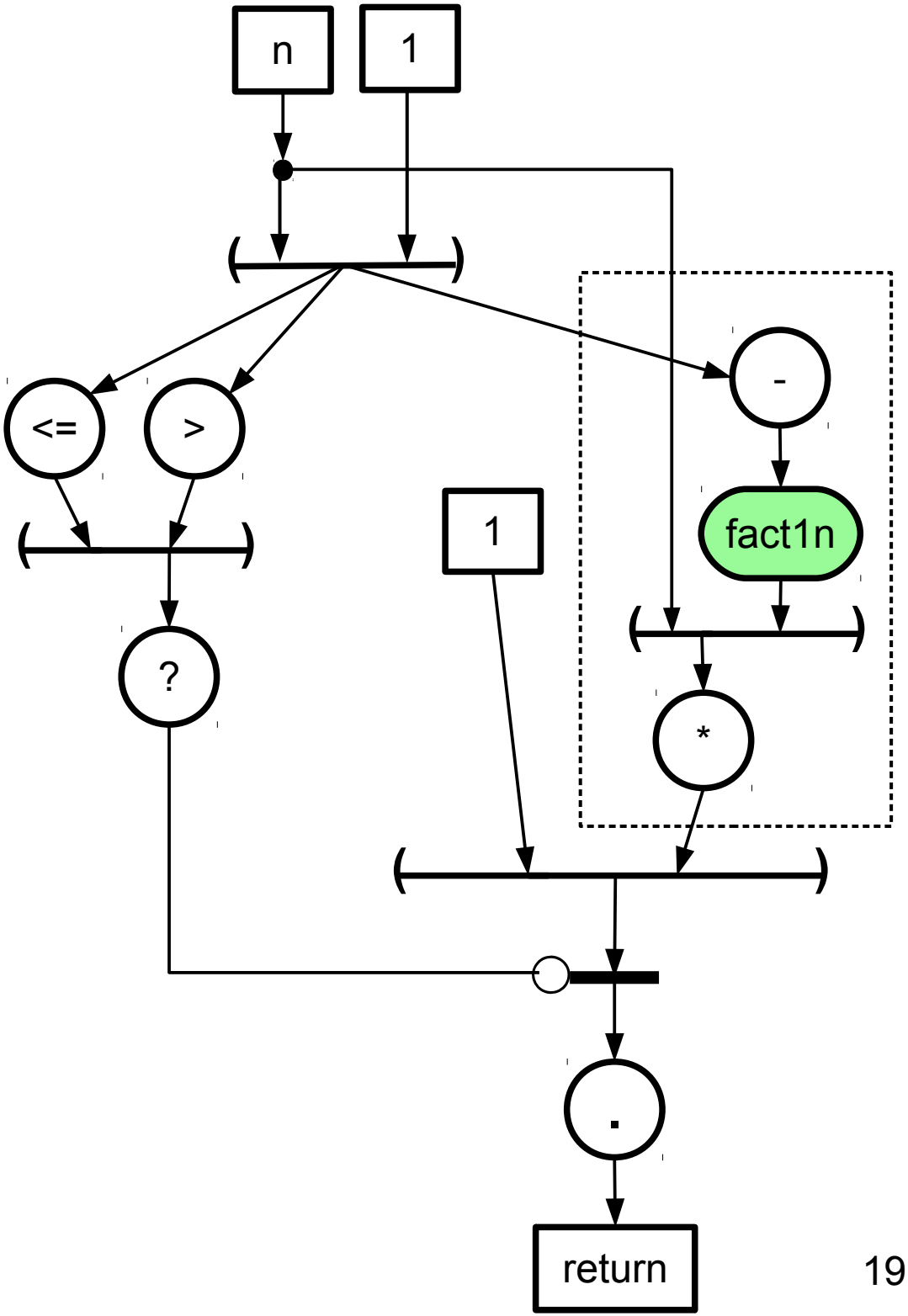


Задержанный список

```
fact1n << funcdef n {
    n1<< (n,1);
    [(n1:[<=,>]):?]^
    (
        1,
        {(n,
n1:-:fact1n):*}
    ):.
>>return
}
```



```
fact1n << funcdef n {
    n1<< (n,1);
    [(n1:[<=,>]):?]^
    (
        1,
        {(n,
n1:-:fact1n):*}
    ):.
>>return
}
```



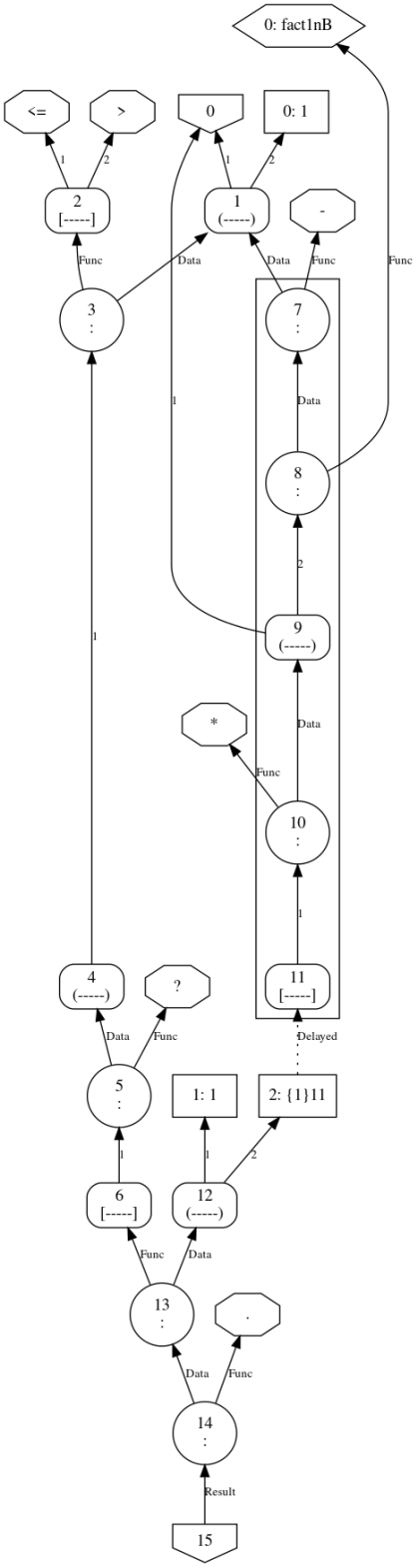
External

0	fact1nB	

Local

0	1	
1	1	
2	{1}11	

id	delay	operation	links
0	0	arg	
1	0	(---)	0 loc:0
2	0	[---]	<= >
3	0	:	1 2
4	0	(---)	3
5	0	:	4 ?
6	0	[---]	5
7	1	:	1 -
8	1	:	7 ext:0
9	1	(---)	0 8
10	1	:	9 *
11	1	[---]	10
12	0	(---)	loc:1 loc:2
13	0	:	12 6
14	0	:	13 .
15	0	return	14



fact1nB

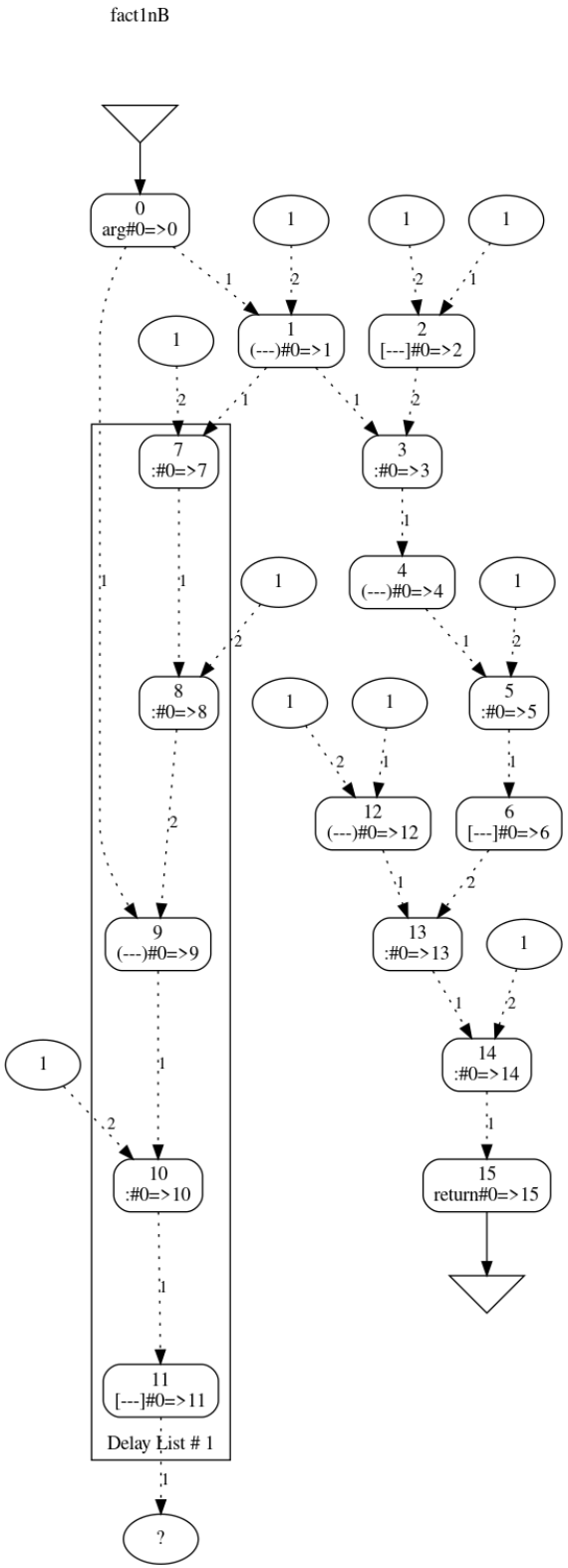
id	delay	automat	inode	links
0	0	arg,0	0	links:1,1;9,1;
1	0	(---),0	1	links:3,1;7,1;
2	0	[---],0	2	links:3,2;
3	0	:,0	3	links:4,1;
4	0	(---),0	4	links:5,1;
5	0	:,0	5	links:6,1;
6	0	[---],0	6	links:13,2;
7	1	:,0	7	links:8,1;
8	1	:,0	8	links:9,2;
9	1	(---),0	9	links:10,1;
10	1	:,0	10	links:11,1;
11	1	[---],0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

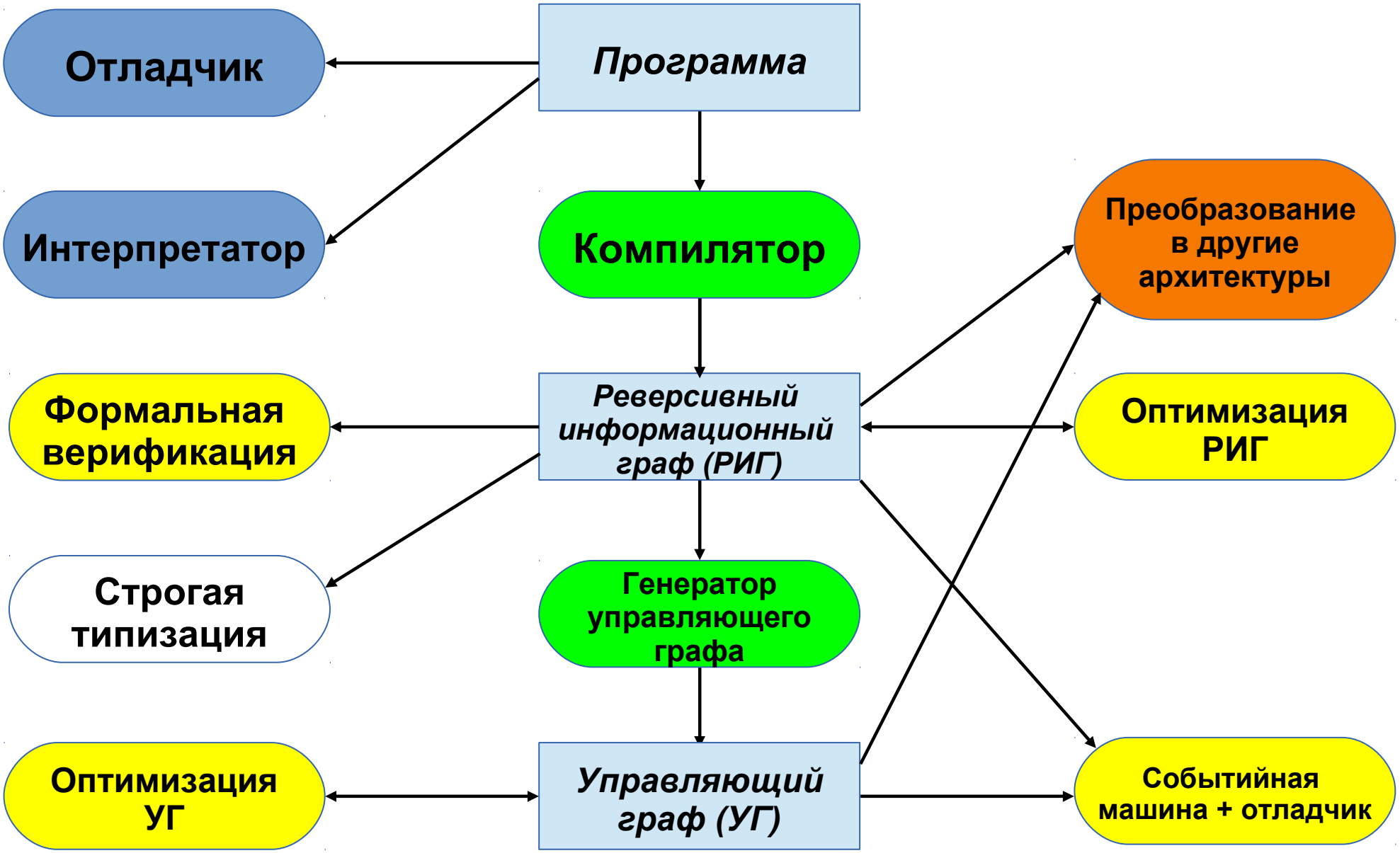
Signals: (Number, Node, Input)

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

Dynamic links: (Number, Delay list, Node)

0	1	11




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

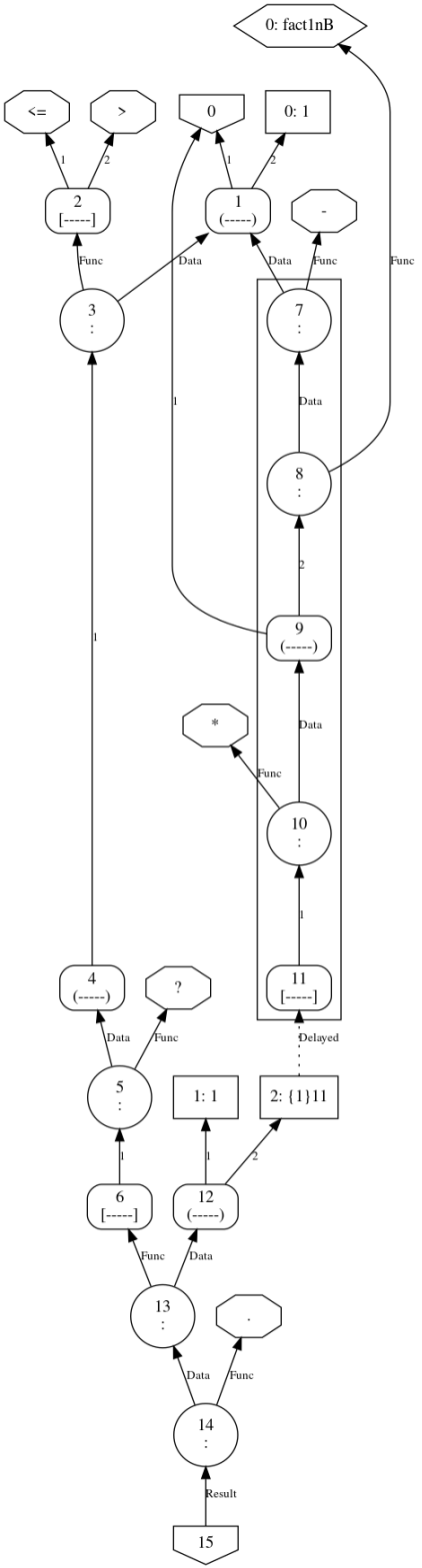
External

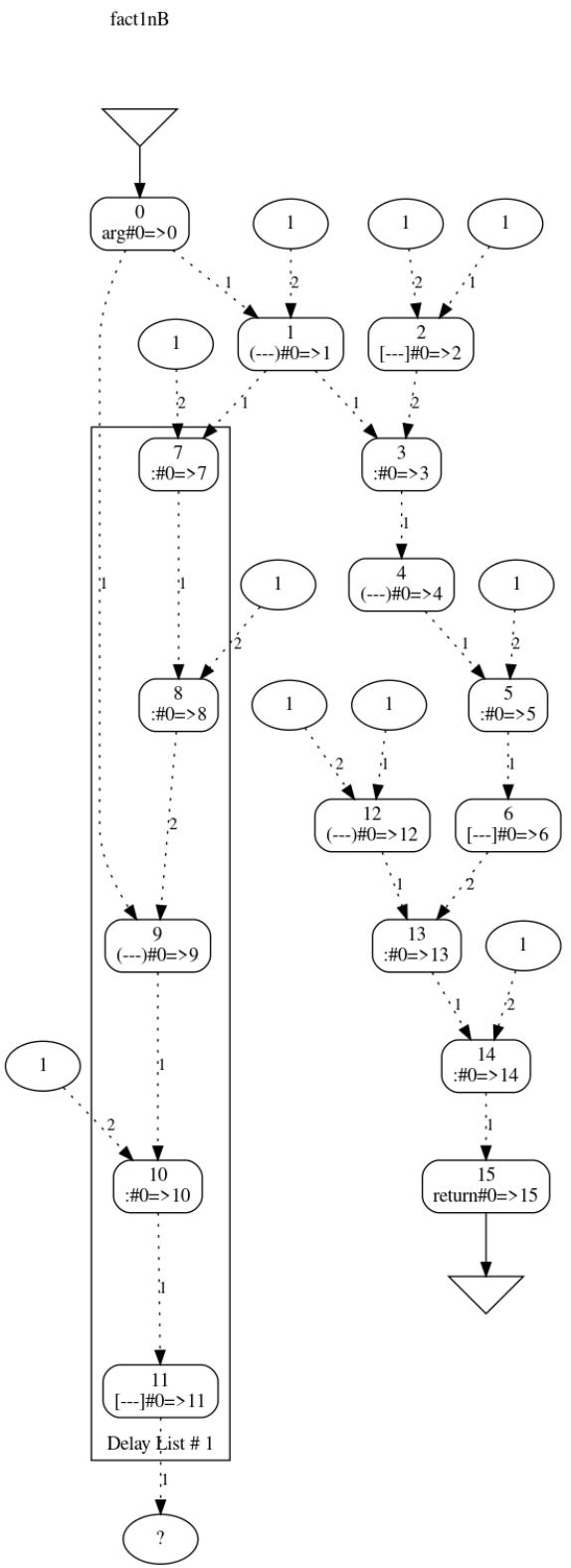
0	fact1nB

Local

0	1
1	1
2	{1}11

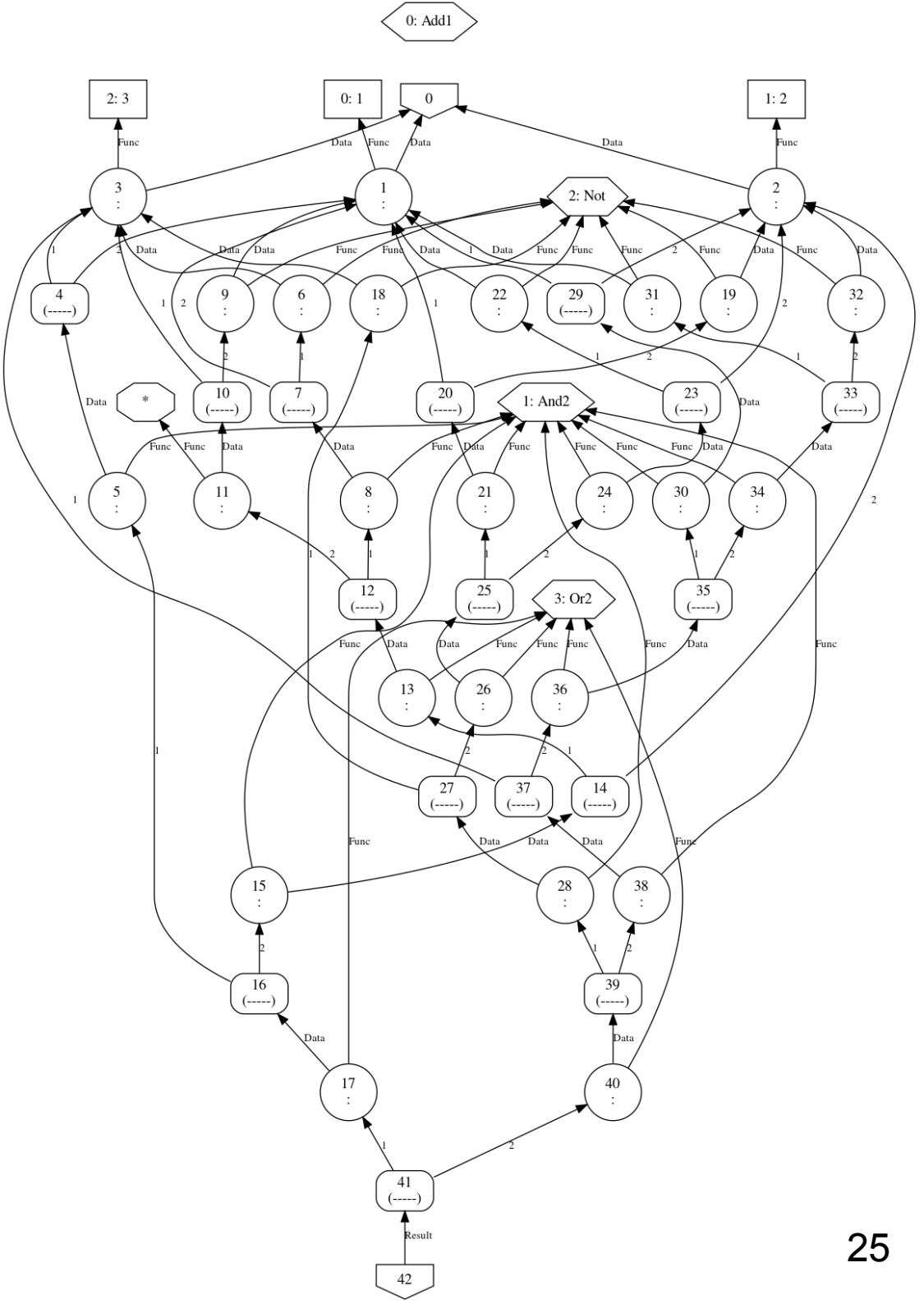
id	delay	operation	links
0	0	arg	
1	0	(---)	0 loc:0
2	0	[---]	<= >
3	0	:	1 2
4	0	(---)	3
5	0	:	4 ?
6	0	[---]	5
7	1	:	1 -
8	1	:	7 ext:0
9	1	(---)	0 8
10	1	:	9 *
11	1	[---]	10
12	0	(---)	loc:1 loc:2
13	0	:	12 6
14	0	:	13 .
15	0	return	14





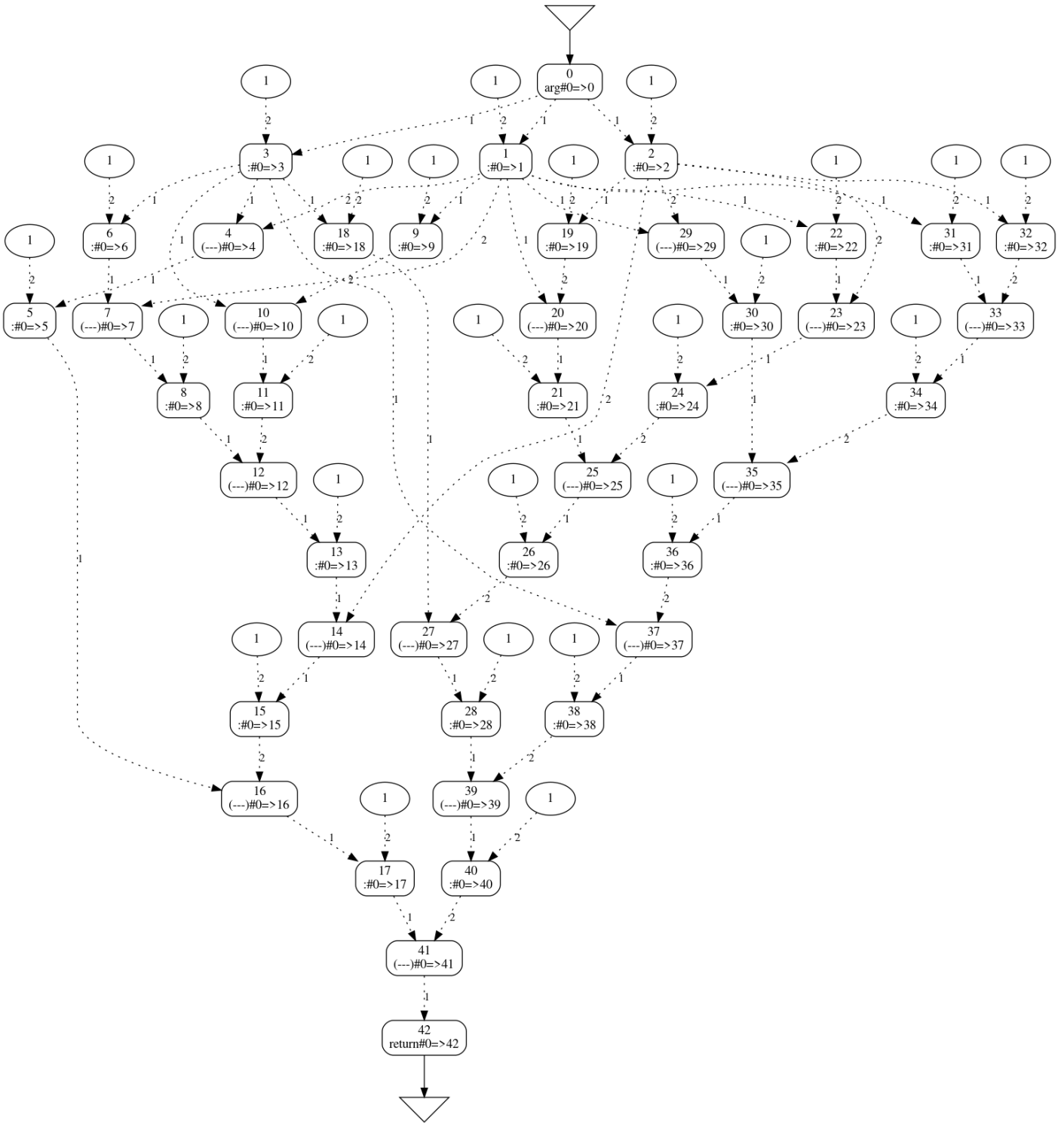
```
// Одноразрядный двоичный сумматор
// Аргумент = (0|1, 0|1, 0|1) - (1-е слагаемое, 2-е слагаемое, перенос)
Add1 << funcdef x1x2p {
    x1 << x1x2p:1;   // первое слагаемое
    x2 << x1x2p:2;   // второе слагаемое
    p  << x1x2p:3;   // заем из предыдущего разряда

    // Формирование переноса
    //C = (P && X1) || (((!P && X1) || (P && !X1)) && X2);
    c << ((p,x1):And2,(((p:Not,x1):And2,(p,x1:Not):*):Or2,x2):And2):Or2;

    // Формирование результата
    //Y = (!P && ((X1 && !X2) || (!X1 && X2))) || (P && ((X1 && X2) || (!X1 && !X2)));
    y << ((p:Not,((x1,x2:Not):And2,(x1:Not,x2):And2):Or2):And2,
         (p,((x1,x2):And2,(x1:Not,x2:Not):And2):Or2):And2):Or2;
    (c,y) >>return;
}
```

```
External
      0     Add1
      1     And2
      2     Not
      3     Or2
Local
      0     1
      1     2
      2     3
id    delay operation    links        positions
0     0     arg                       pos 1 17 1 22
1     0     :            0 loc:0             pos 2 16 2 17
2     0     :            0 loc:1             pos 3 16 3 17
3     0     :            0 loc:2             pos 4 16 4 17
4     0     (---)        3 1          pos 6 11 6 17
5     0     :            4 ext:1             pos 6 17 6 18
6     0     :            3 ext:2             pos 6 27 6 28
7     0     (---)        6 1          pos 6 25 6 35
8     0     :            7 ext:1             pos 6 35 6 36
9     0     :            1 ext:2             pos 6 46 6 47
10    0     (---)        3 9          pos 6 41 6 51
11    0     :            10 *         pos 6 51 6 52
12    0     (---)        8 11         pos 6 24 6 54
13    0     :            12 ext:3            pos 6 54 6 55
14    0     (---)        13 2         pos 6 23 6 62
15    0     :            14 ext:1            pos 6 62 6 63
16    0     (---)        5 15         pos 6 10 6 68
17    0     :            16 ext:3            pos 6 68 6 69
18    0     :            3 ext:2             pos 8 13 8 14
19    0     :            2 ext:2             pos 8 25 8 26
20    0     (---)        1 19         pos 8 19 8 30
21    0     :            20 ext:1            pos 8 30 8 31
22    0     :            1 ext:2             pos 8 39 8 40
23    0     (---)        22 2         pos 8 36 8 47
24    0     :            23 ext:1            pos 8 47 8 48
25    0     (---)        21 24        pos 8 18 8 53
26    0     :            25 ext:3            pos 8 53 8 54
27    0     (---)        18 26        pos 8 11 8 58
28    0     :            27 ext:1            pos 8 58 8 59
29    0     (---)        1 2          pos 8 69 8 76
30    0     :            29 ext:1            pos 8 76 8 77
31    0     :            1 ext:2             pos 8 85 8 86
32    0     :            2 ext:2             pos 8 92 8 93
33    0     (---)        31 32        pos 8 82 8 97
34    0     :            33 ext:1            pos 8 97 8 98
35    0     (---)        30 34        pos 8 68 8 103
36    0     :            35 ext:3            pos 8 103 8 104
37    0     (---)        3 36         pos 8 65 8 108
38    0     :            37 ext:1            pos 8 108 8 109
39    0     (---)        28 38        pos 8 10 8 114
40    0     :            39 ext:3            pos 8 114 8 115
41    0     (---)        17 40        pos 9 5 9 10
42    0     return       41           pos 9 13 9 19
```

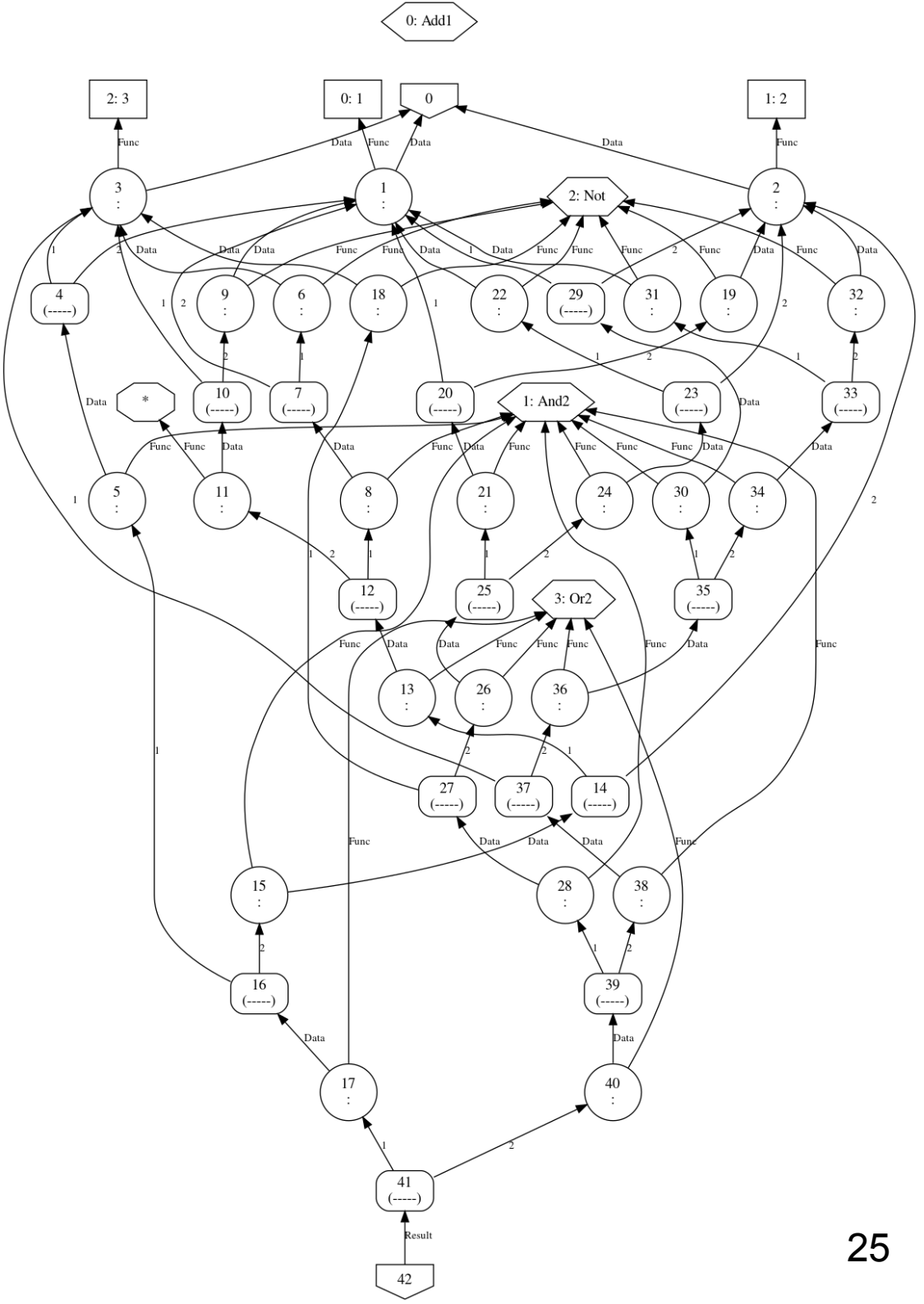


```
Add1
id    delay automat          inode links
0     0     arg,0       0    links:1,1;2,1;3,1;
1     0     :,0         1    links:4,2;7,2;9,1;20,1;22,1;29,1;31,1;
2     0     :,0         2    links:14,2;19,1;23,2;29,2;32,1;
3     0     :,0         3    links:4,1;6,1;10,1;18,1;37,1;
4     0     (---),0          4    links:5,1;
5     0     :,0         5    links:16,1;
6     0     :,0         6    links:7,1;
7     0     (---),0          7    links:8,1;
8     0     :,0         8    links:12,1;
9     0     :,0         9    links:10,2;
10    0     (---),0          10   links:11,1;
11    0     :,0         11   links:12,2;
12    0     (---),0          12   links:13,1;
13    0     :,0         13   links:14,1;
14    0     (---),0          14   links:15,1;
15    0     :,0         15   links:16,2;
16    0     (---),0          16   links:17,1;
17    0     :,0         17   links:41,1;
18    0     :,0         18   links:27,1;
19    0     :,0         19   links:20,2;
20    0     (---),0          20   links:21,1;
21    0     :,0         21   links:25,1;
22    0     :,0         22   links:23,1;
23    0     (---),0          23   links:24,1;
24    0     :,0         24   links:25,2;
25    0     (---),0          25   links:26,1;
26    0     :,0         26   links:27,2;
27    0     (---),0          27   links:28,1;
28    0     :,0         28   links:39,1;
29    0     (---),0          29   links:30,1;
30    0     :,0         30   links:35,1;
31    0     :,0         31   links:33,1;
32    0     :,0         32   links:33,2;
33    0     (---),0          33   links:34,1;
34    0     :,0         34   links:35,2;
35    0     (---),0          35   links:36,1;
36    0     :,0         36   links:37,2;
37    0     (---),0          37   links:38,1;
38    0     :,0         38   links:39,2;
39    0     (---),0          39   links:40,1;
40    0     :,0         40   links:41,2;
41    0     (---),0          41   links:42,1;
42    0     return,0         42
```

```
Signals: (Number, Node, Input)
0     1     2
1     2     2
2     3     2
3     5     2
4     6     2
5     8     2
6     9     2
7     11    2
8     13    2
9     15    2
10    17    2
11    18    2
12    19    2
13    21    2
14    22    2
15    24    2
16    26    2
17    28    2
18    30    2
19    31    2
20    32    2
21    34    2
22    36    2
23    38    2
24    40    2
Dynamic links: (Number, Delay list, Node)
```

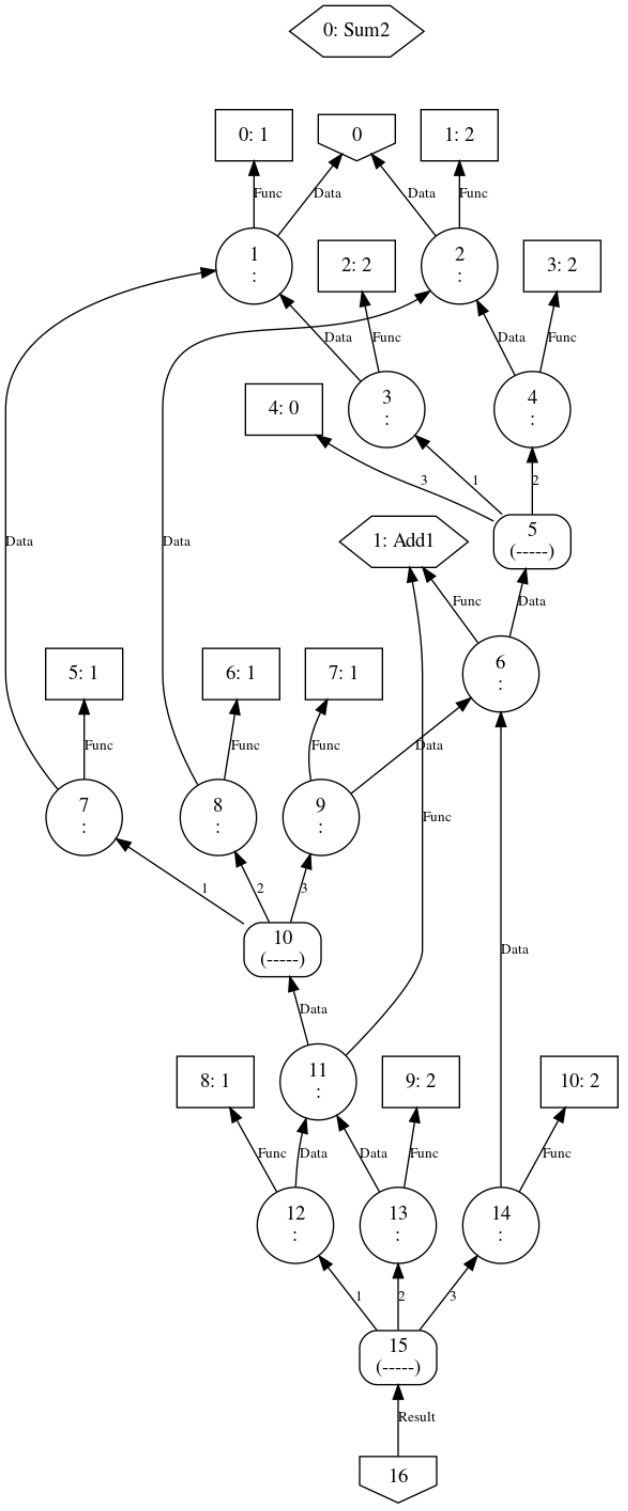

Add1
0
arg#0=>0
1
:#0=>1
2
:#0=>2
3
:#0=>3
4
(---)#0=>4
5
:#0=>5
6
:#0=>6
7
(---)#0=>7
8
:#0=>8
9
:#0=>9
10
(---)#0=>10
11
:#0=>11
12
(---)#0=>12
13
:#0=>13
14
(---)#0=>14
15
:#0=>15
16
(---)#0=>16
17
:#0=>17
18
:#0=>18
19
:#0=>19
20
(---)#0=>20
21
:#0=>21
22
:#0=>22
23
(---)#0=>23
24
:#0=>24
25
(---)#0=>25
26
:#0=>26
27
(---)#0=>27
28
:#0=>28
29
(---)#0=>29
30
:#0=>30
31
:#0=>31
32
:#0=>32
33
(---)#0=>33
34
:#0=>34
35
(---)#0=>35
36
:#0=>36
37
(---)#0=>37
38
:#0=>38
39
(---)#0=>39
40
:#0=>40
41
(---)#0=>41
42
return#0=>42

```
External
      0     Add1
      1     And2
      2     Not
      3     Or2
Local
      0     1
      1     2
      2     3
id    delay operation    links        positions
0     0     arg                       pos 1 17 1 22
1     0     :            0 loc:0             pos 2 16 2 17
2     0     :            0 loc:1             pos 3 16 3 17
3     0     :            0 loc:2             pos 4 16 4 17
4     0     (---)        3 1          pos 6 11 6 17
5     0     :            4 ext:1             pos 6 17 6 18
6     0     :            3 ext:2             pos 6 27 6 28
7     0     (---)        6 1          pos 6 25 6 35
8     0     :            7 ext:1             pos 6 35 6 36
9     0     :            1 ext:2             pos 6 46 6 47
10    0     (---)        3 9          pos 6 41 6 51
11    0     :            10 *         pos 6 51 6 52
12    0     (---)        8 11         pos 6 24 6 54
13    0     :            12 ext:3            pos 6 54 6 55
14    0     (---)        13 2         pos 6 23 6 62
15    0     :            14 ext:1            pos 6 62 6 63
16    0     (---)        5 15         pos 6 10 6 68
17    0     :            16 ext:3            pos 6 68 6 69
18    0     :            3 ext:2             pos 8 13 8 14
19    0     :            2 ext:2             pos 8 25 8 26
20    0     (---)        1 19         pos 8 19 8 30
21    0     :            20 ext:1            pos 8 30 8 31
22    0     :            1 ext:2             pos 8 39 8 40
23    0     (---)        22 2         pos 8 36 8 47
24    0     :            23 ext:1            pos 8 47 8 48
25    0     (---)        21 24        pos 8 18 8 53
26    0     :            25 ext:3            pos 8 53 8 54
27    0     (---)        18 26        pos 8 11 8 58
28    0     :            27 ext:1            pos 8 58 8 59
29    0     (---)        1 2          pos 8 69 8 76
30    0     :            29 ext:1            pos 8 76 8 77
31    0     :            1 ext:2             pos 8 85 8 86
32    0     :            2 ext:2             pos 8 92 8 93
33    0     (---)        31 32        pos 8 82 8 97
34    0     :            33 ext:1            pos 8 97 8 98
35    0     (---)        30 34        pos 8 68 8 103
36    0     :            35 ext:3            pos 8 103 8 104
37    0     (---)        3 36         pos 8 65 8 108
38    0     :            37 ext:1            pos 8 108 8 109
39    0     (---)        28 38        pos 8 10 8 114
40    0     :            39 ext:3            pos 8 114 8 115
41    0     (---)        17 40        pos 9 5 9 10
42    0     return       41           pos 9 13 9 19
```



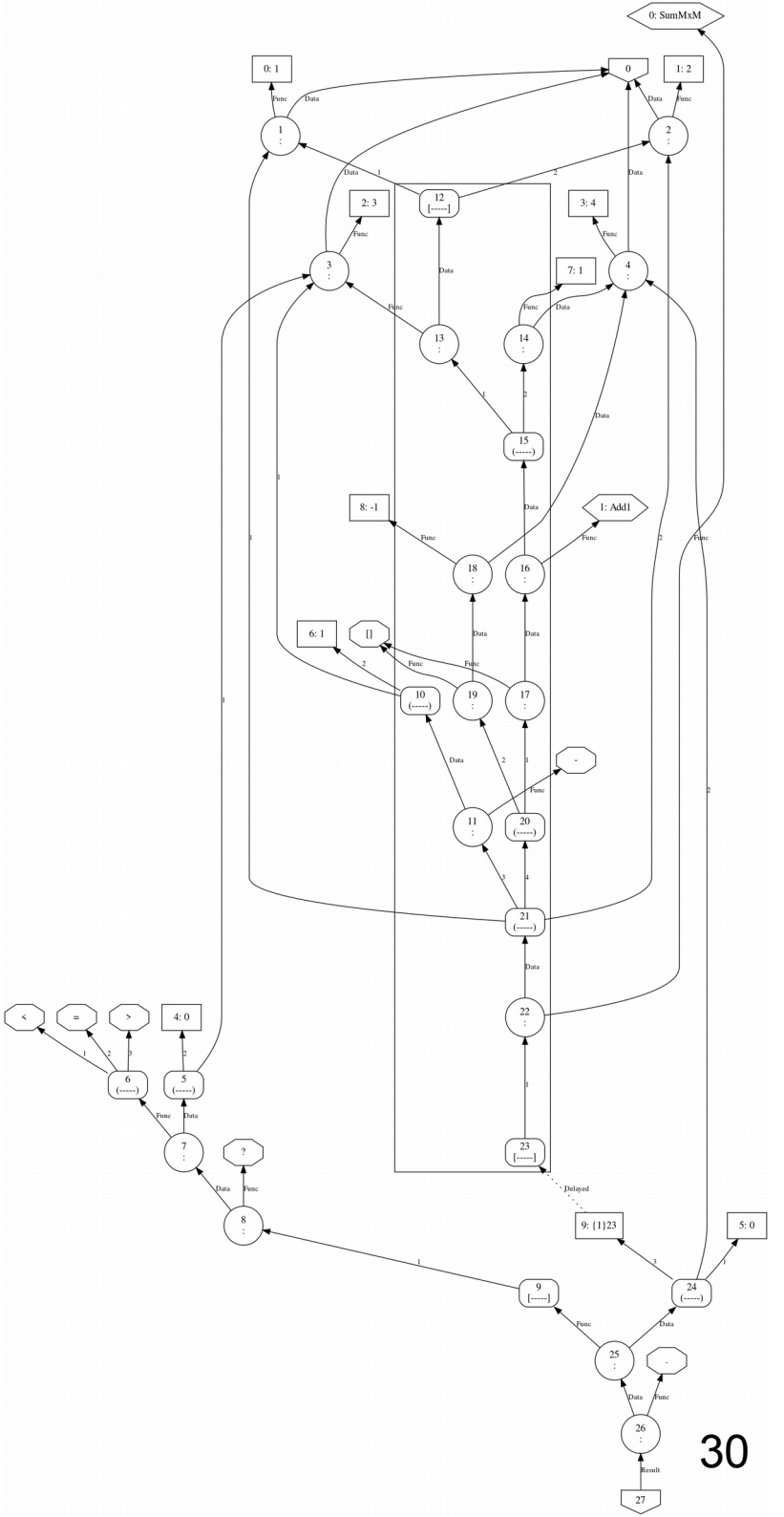
```
// Двухразрядный двоичный сумматор,
// построенный на основе одноразрядных
// Аргумент = (1-е слагаемое, 2-е слагаемое)
// Результат = (перенос, старший разряд, младший
разряд)
Sum2 << funcdef s1s2 {
     s1<< s1s2:1;
     s2<< s1s2:2;
     y0<< (s1:2,s2:2,0):Add1;
     y1<< (s1:1,s2:1,y0:1):Add1;
     (y1:1,y1:2,y0:2) >>return
}
```

```
External
      0     Sum2
      1     Add1
Local
      0     1
      1     2
      2     2
      3     2
      4     0
      5     1
      6     1
      7     1
      8     1
      9     2
      10    2
id    delay operation   links       positions
0     0     arg                     pos 1 17 1 21
1     0     :           0 loc:0           pos 2 14 2 15
2     0     :           0 loc:1           pos 3 14 3 15
3     0     :           1 loc:2           pos 4 13 4 14
4     0     :           2 loc:3           pos 4 18 4 19
5     0     (---)       3 4 loc:4         pos 4 10 4 23
6     0     :           5 ext:1           pos 4 23 4 24
7     0     :           1 loc:5           pos 5 13 5 14
8     0     :           2 loc:6           pos 5 18 5 19
9     0     :           6 loc:7           pos 5 23 5 24
10    0     (---)       7 8 9       pos 5 10 5 26
11    0     :           10 ext:1          pos 5 26 5 27
12    0     :           11 loc:8          pos 6 8 6 9
13    0     :           11 loc:9          pos 6 13 6 14
14    0     :           6 loc:10          pos 6 18 6 19
15    0     (---)       12 13 14          pos 6 5 6 21
16    0     return            15          pos 6 24 6 30
```



```
SumMxM << funcdef s1s2 {
    s1<< s1s2:1;
    s2<< s1s2:2;
    count<< s1s2:3;
    sum<< s1s2:4;
    // Проверка равенства длин слагаемых
    [(count,0):(<,=,>):?]^
    (
        0,    // некорректная длина
        sum,  // сумма накоплена
        {     // правая рекурсия с накоплением
            (s1,s2,(count,1):-,
            (([s1,s2]:count,sum:1):Add1:[],sum:-1:[])):SumMxM
        }
    ):. >>return
}
```

```
External
      0     SumMxM
      1     Add1
Local
      0     1
      1     2
      2     3
      3     4
      4     0
      5     0
      6     1
      7     1
      8     -1
      9     {1}23
id    delay operation  links       positions
0     0     arg                    pos 1 19 1 23
1     0     :          0 loc:0          pos 2 14 2 15
2     0     :          0 loc:1          pos 3 14 3 15
3     0     :          0 loc:2          pos 4 17 4 18
4     0     :          0 loc:3          pos 5 15 5 16
5     0     (---)      3 loc:4          pos 6 6 6 15
6     0     (---)      < = >       pos 6 16 6 23
7     0     :          5 6         pos 6 15 6 16
8     0     :          7 ?         pos 6 23 6 24
9     0     [---]      8           pos 6 5 6 26
10    1     (---)      3 loc:6          pos 11 22 11 31
11    1     :          10 -        pos 11 31 11 32
12    1     [---]      1 2         pos 11 36 11 43
13    1     :          12 3        pos 11 43 11 44
14    1     :          4 loc:7          pos 11 53 11 54
15    1     (---)      13 14       pos 11 35 11 56
16    1     :          15 ext:1         pos 11 56 11 57
17    1     :          16 []       pos 11 61 11 62
18    1     :          4 loc:8          pos 11 68 11 69
19    1     :          18 []       pos 11 71 11 72
20    1     (---)      17 19       pos 11 34 11 75
21    1     (---)      1 2 11 20        pos 11 15 11 76
22    1     :          21 ext:0         pos 11 76 11 77
23    1     [---]      22          pos 10 10 12 11
24    0     (---)      loc:5 4 loc:9          pos 7 5 13 6
25    0     :          24 9        pos 6 26 6 27
26    0     :          25 .        pos 13 6 13 7
27    0     return           26          pos 13 11 13 17
```



Благодарю за внимание!