

FLIDE

FLOGOL Integrated Development
Environment

Система

функционально-логического
программирования
на языке S-FLOGOL

Язык S-FLOGOL

- Основан на теории направленных отношений (НО) (Фальк В.Н., Кутепов В.П.).
- Имеет развитые средства схемного описания НО.
- Допускает использование:
 - индексированных имен НО,
 - параметризованных НО.
- Поддерживает объектно-ориентированный стиль программирования.
- Позволяет строить многомодульные программы.
- Обладает средствами ограничения области видимости определяемых отношений (Private-домены).

Направленные отношения

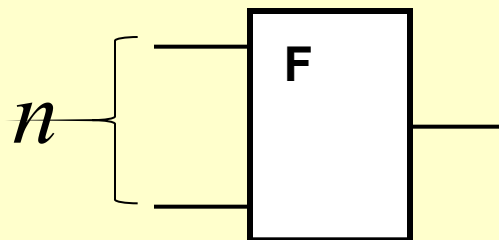
- Направленным отношением R арности (n, m) на носителе D называется множество упорядоченных пар кортежей элементов D длины n и m , соответственно.
- НО $R^{(n, m)}$ может моделироваться $(n+m)$ -арным отношением R' на D :

$$\langle a_1, \dots, a_n, a_{n+1}, \dots, a_{n+m} \rangle \in R' \approx \langle a_1, \dots, a_n, a_{n+1}, \dots, a_m \rangle \in R.$$

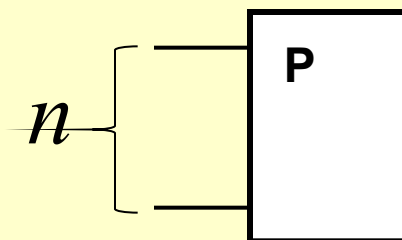
- Естественной и наглядной формой определения НО и основой эффективной организации их вычисления является представление в форме сетевых языков.
- НО характеризуются наличием фундаментальных свойств – *тотальности* и *функциональности* прямых и обратных НО.

Семантические объекты

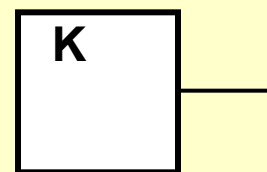
- Функция $F^{(n,1)}$



- Предикат $P^{(n,0)}$

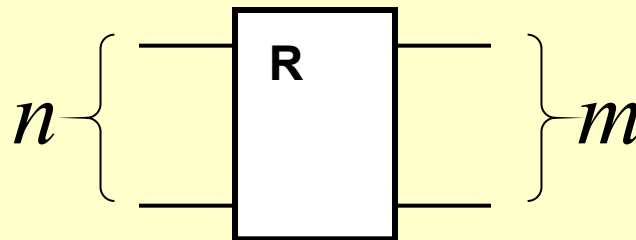


- Константа $K^{(0,1)}$



- НО общего вида

$$R^{(n,m)}$$



Варианты определения НО

Пример: определение натурального числа 3

- В форме графика:

$$\text{Nat3} = \{:\text{Succ}(\text{Succ}(\text{Succ}(\text{Null}))))\}$$

- В форме композиции:

$$\text{Nat3} = \text{Null} \cdot \text{Succ} \cdot \text{Succ} \cdot \text{Succ}$$

- При помощи свертки:

$$\text{Nat3} = \text{Null} \cdot (\cdot \text{I}=1..3)\text{Succ}$$

Определение НО в виде графика

- В форме графика НО задается в виде

$R = \{ \text{Терм1} : \text{Терм2} ? \text{Формула} \}$, где

Терм1 – входной терм (образец аргументов вызова),

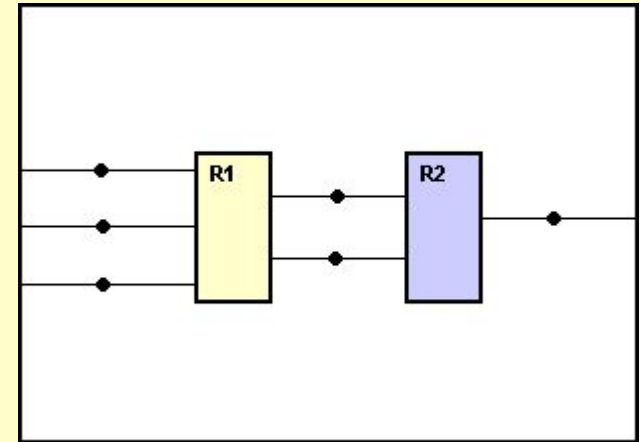
Терм2 – выходной терм (результат вызова),

Формула – набор ограничений вида $\text{Терм} = \text{Терм}$, $\text{Терм} <> \text{Терм}$.

Композиционное определение НО

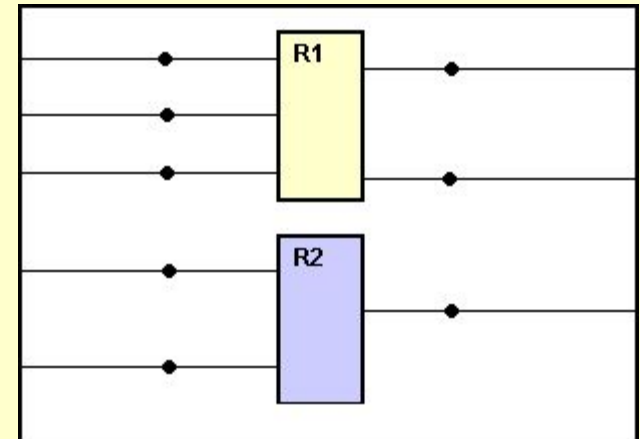
- Последовательная композиция

$$R^{(n,m)} = R1^{(n,n')} \bullet R2^{(n',m)}$$



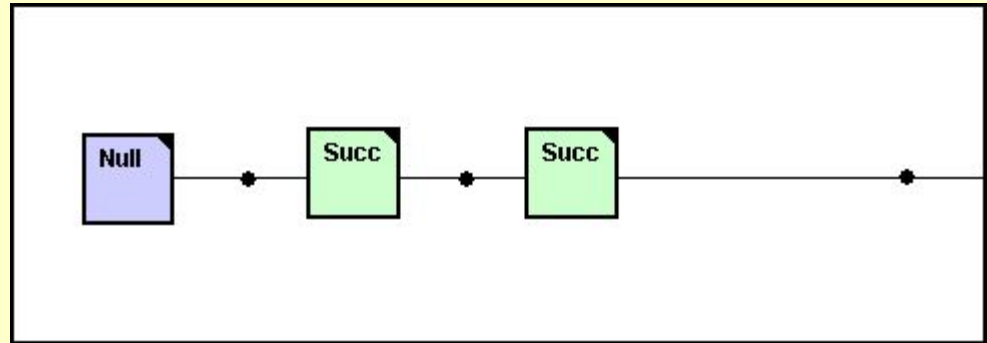
- Параллельная композиция

$$R^{(n'+n'',m'+m'')} = R1^{(n',m')} \# R2^{(n'',m'')}$$

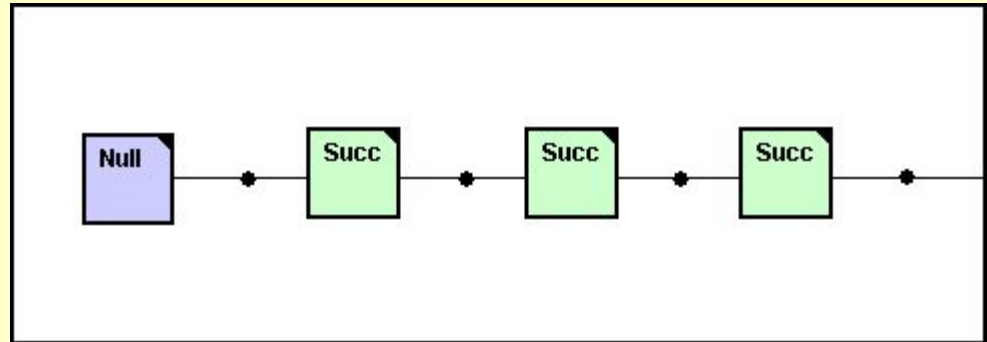


Представление натуральных чисел

Число 2 (сеть):



Число 3 (сеть):

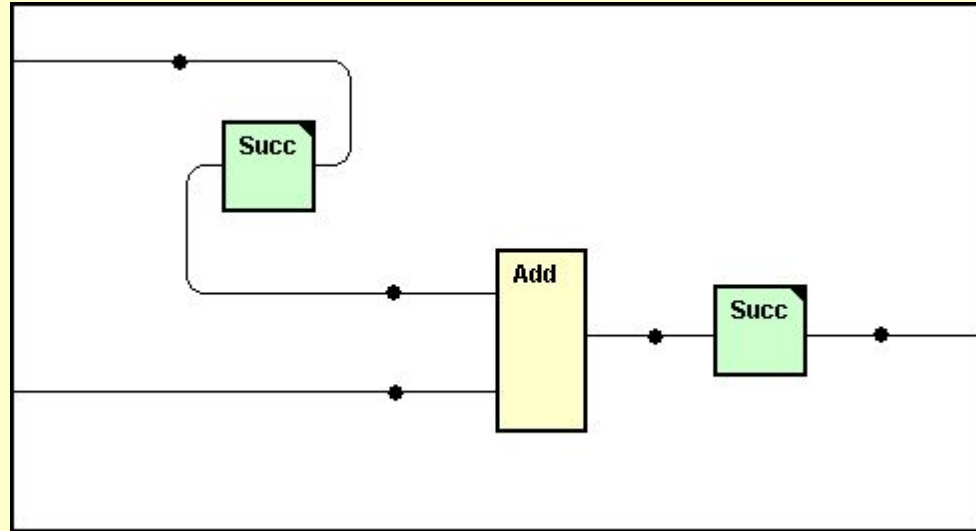
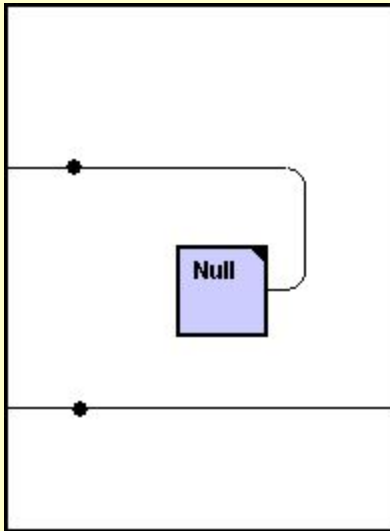


Числа 2,3 (текст):

```
(0:1)Null;  
(1:1)Succ;  
Nat2={:Succ(Succ(Null))};  
Nat3={:Succ(Succ(Succ(Null)))};
```


Отношения над числами

Пример: НО $Add^{(2,1)}$



(0:1)Null;

(1:1)Succ;

(2:1)Add={Null,x:x};

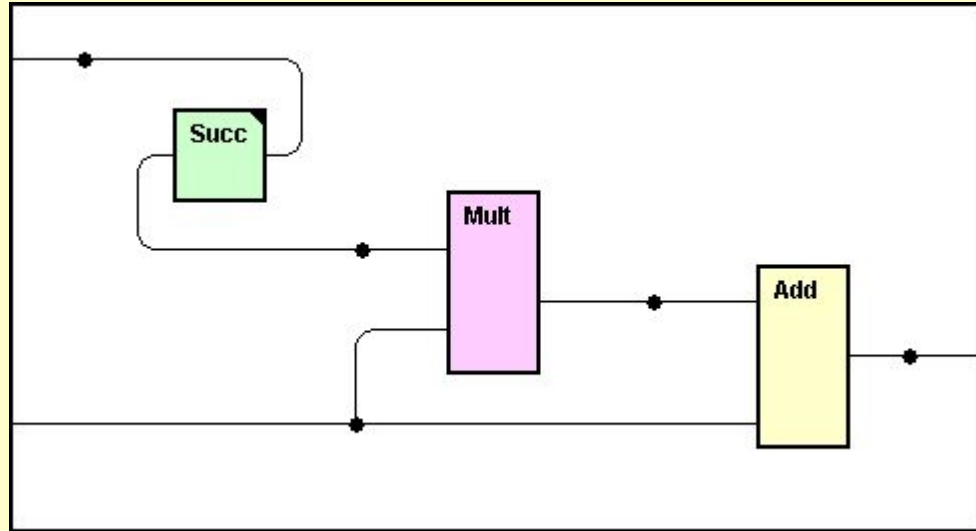
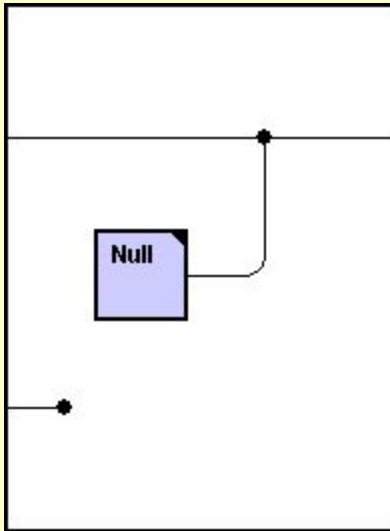
(2:1)Add={Succ(x),y:Succ(@x,y)}

или

(2:1)Add={Null,x:x}U{Succ(x),y:Succ(@x,y)}

Отношения над числами

Пример: НО $Mult^{(2,1)}$



$(0:1)Null;$

$(1:1)Succ;$

$(2:1)Mult=\{Null,x:Null\};$

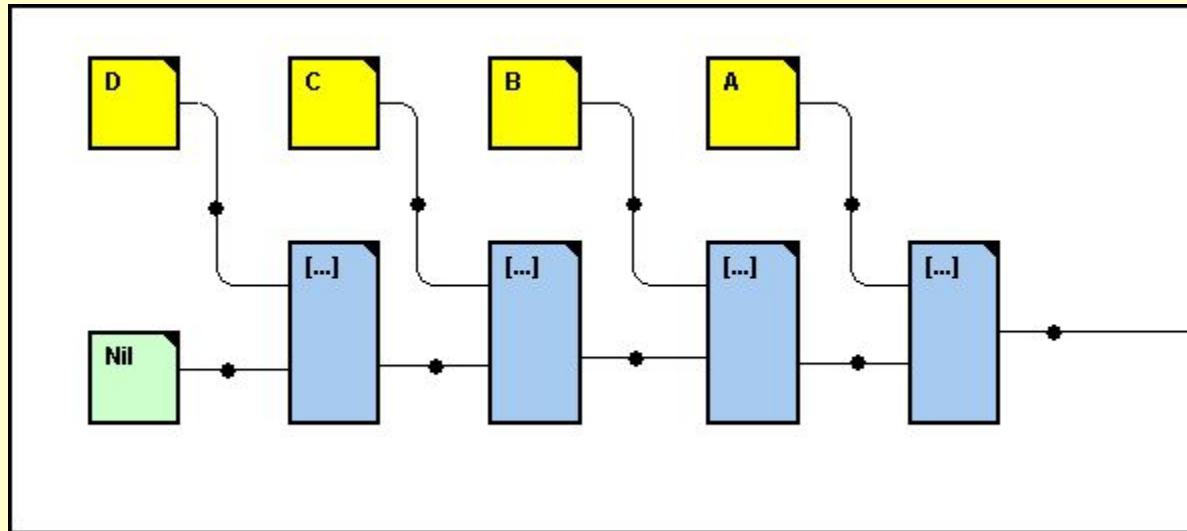
$(2:1)Mult=\{Succ(x),y:Add(@ (x,y),y)\}$

или

$(2:1)Mult=\{Null,x:Null\} \cup \{Succ(x),y:Add(@ (x,y),y)\}$

Списки

Пример: НО $List^{(0,1)}$ [A,B,C,D]



(0:1)Nil;

(1:1)LCons;

(0:1)A;

(0:1)B;

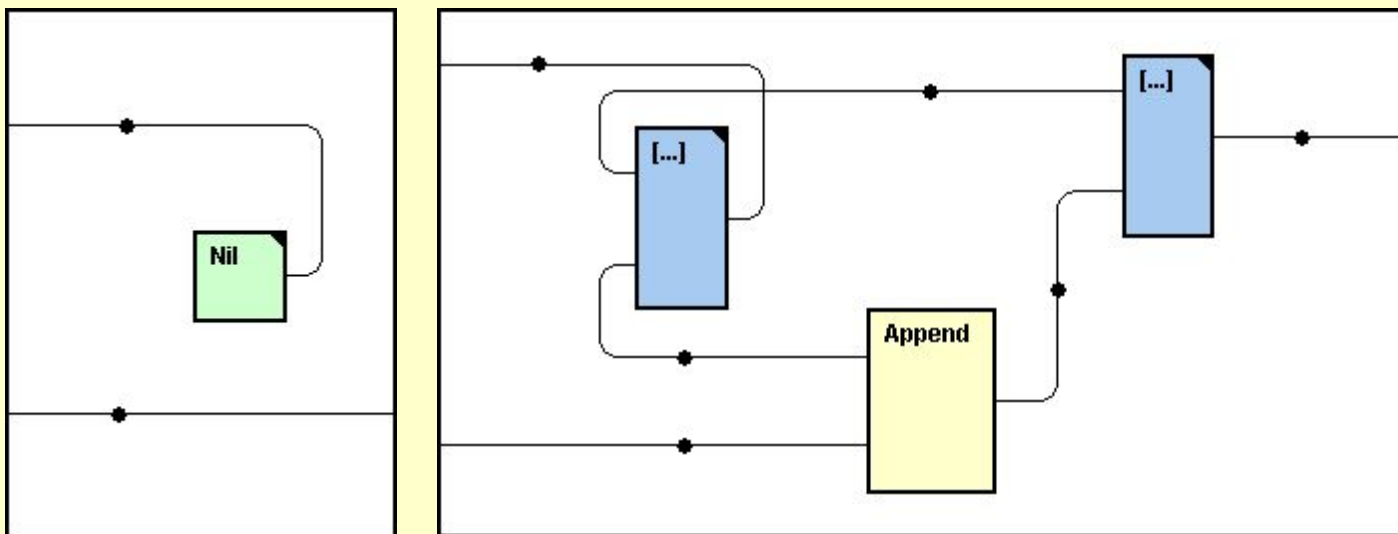
(0:1)C;

(0:1)D;

(0:1)List={:LCons(A,LCons(B,LCons(C,LCons(D,Nil))))}

Отношения над списками

Пример: НО $Append^{(2,1)}$



$(0:1)$ Nil;

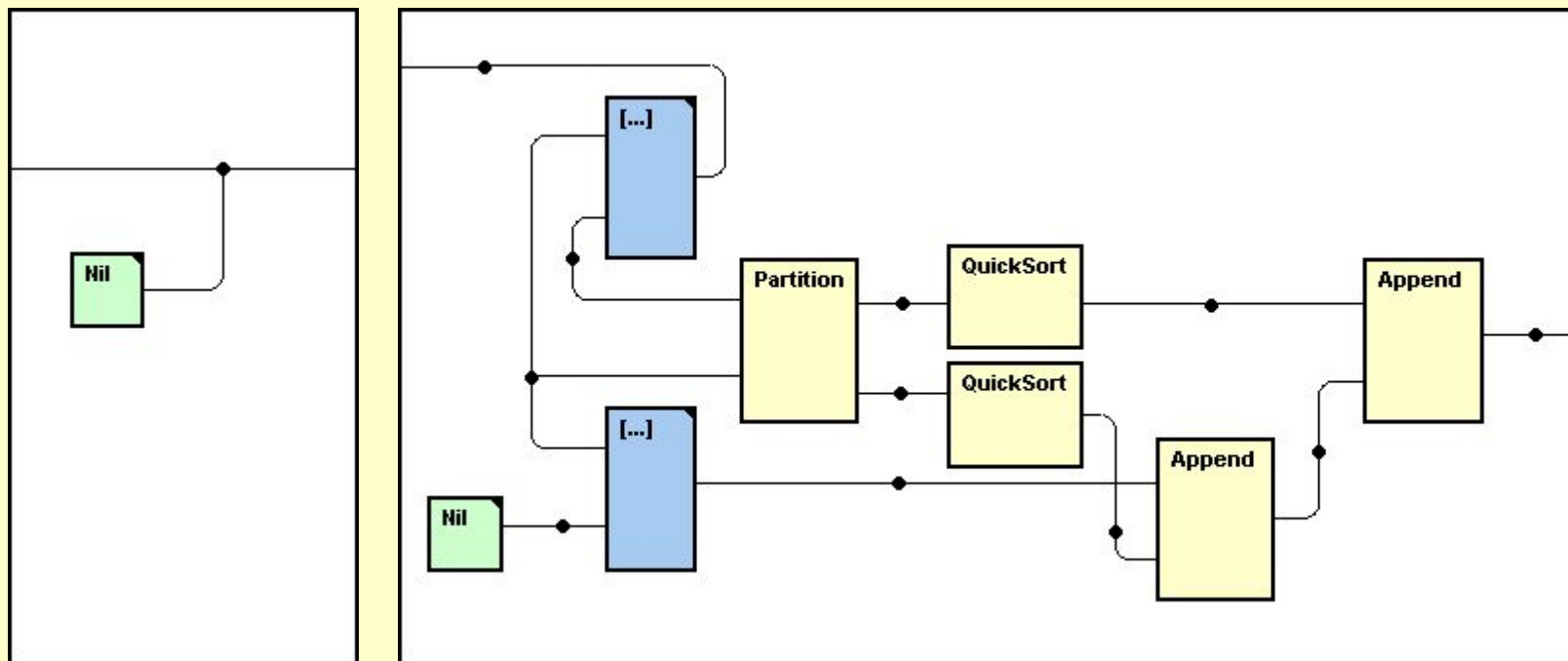
$(2:1)$ LCons;

$(2:1)$ Append = { Nil, $x:x$ };

$(2:1)$ Append = { LCons(x, xs), ys : LCons($x, @(xs, ys)$) }

Отношения над списками

Пример: НО $QuickSort^{(1,1)}$



QuickSort=

{LCons(x,xs):Append(@(ls),Append(LCons(x,Nil),@(bs)))

?Partition(xs,x)=ls,bs};

QuickSort={Nil:Nil};

Индексированные имена НО

Пример: множество натуральных чисел

$$(I=1..)[I]Nat = \text{Null} \cdot (\cdot J=1..I) \text{Succ}$$

Эквивалентное множество определений:

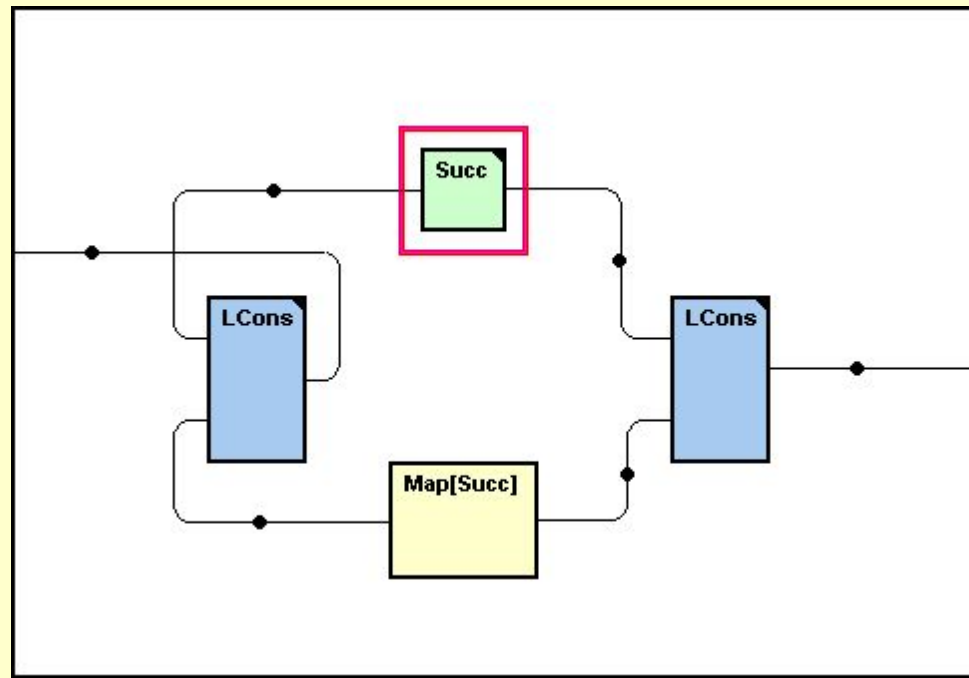
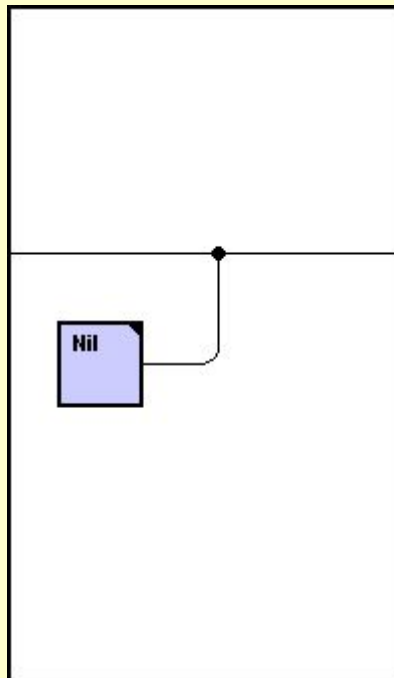
```
[0]Nat = Null;  
[1]Nat = Null · Succ;  
[2]Nat = Null · Succ · Succ;  
[3]Nat = Null · Succ · Succ · Succ;  
...
```

Параметризованные НО

Пример:НО $SuccList^{(1,1)}$

(инкрементация элементов списка)

```
Map={LCons(x,xs):LCons(«0»(x),@(xs))};  
Map={Nil:Nil};  
SuccList = Map[Succ]
```



Типовые НО

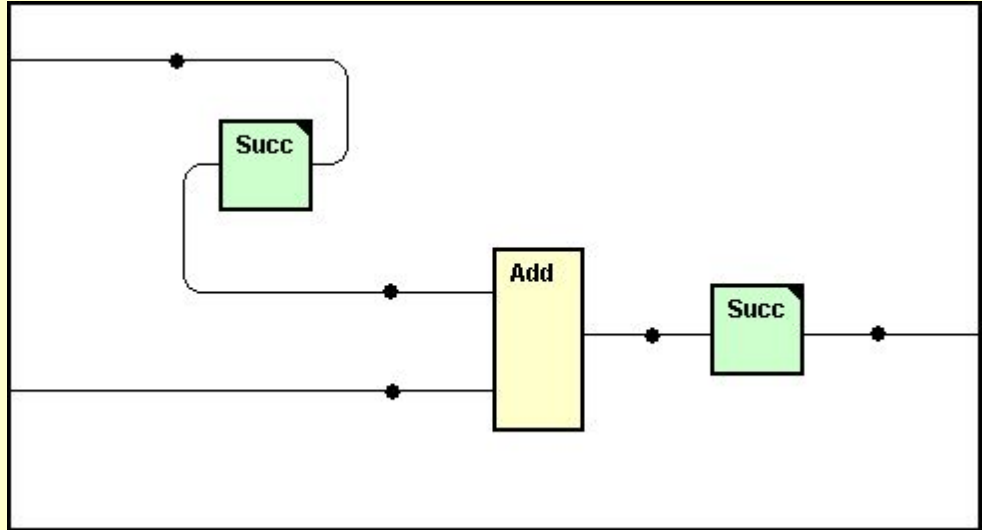
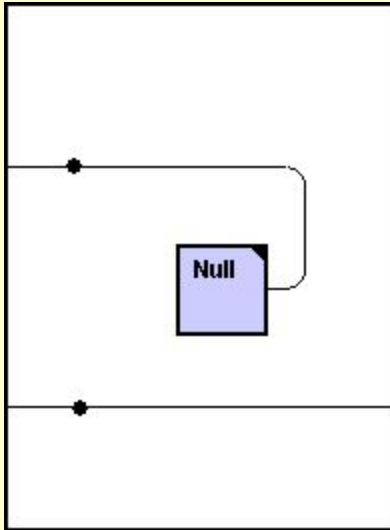
Типовое отношение для типа $T = (t'_1, \dots, t'_{n'} \rightarrow t''_1, \dots, t''_{n''})$,
где $t'_i(t''_i)$ – сорта входных(выходных) данных,
определяется в сетевой форме как:

$$TypeT = \{G_{t'_1}, \boxtimes, G_{t'_{n'}} : G_{t''_1}, \boxtimes, G_{t''_{n''}}\} \cap \langle\langle 0 \rangle\rangle,$$

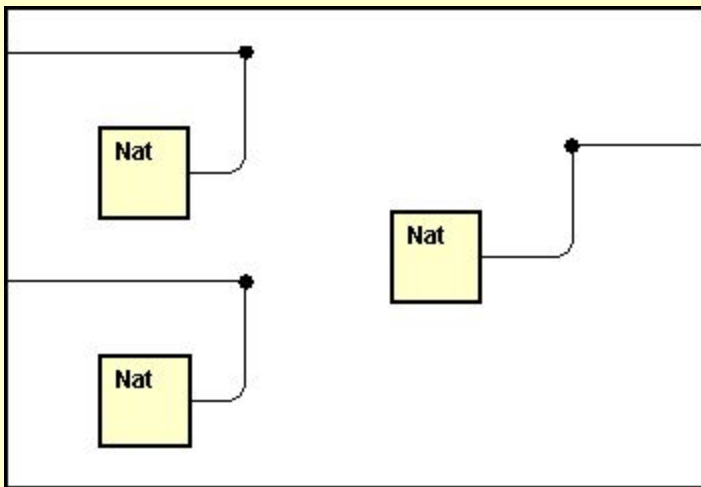
где G_t – отношение арности $(0,1)$, генерирующее
данные сорта t . Описание отношения R типа T
будет иметь вид $R = TypeT[e]$, где e – реляционное
выражение, определяющее направленное
отношение арности (n', n'') .

Типизация НО

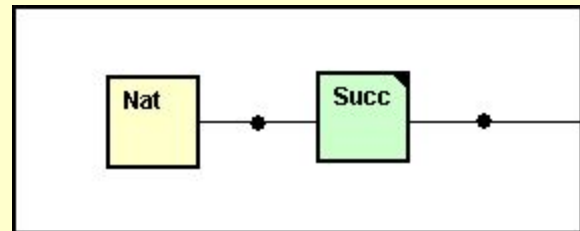
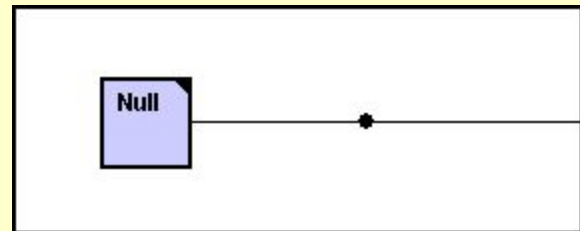
Пример: НО $Add^{(2,1)}$



$TNatAdd^{(2,1)}$



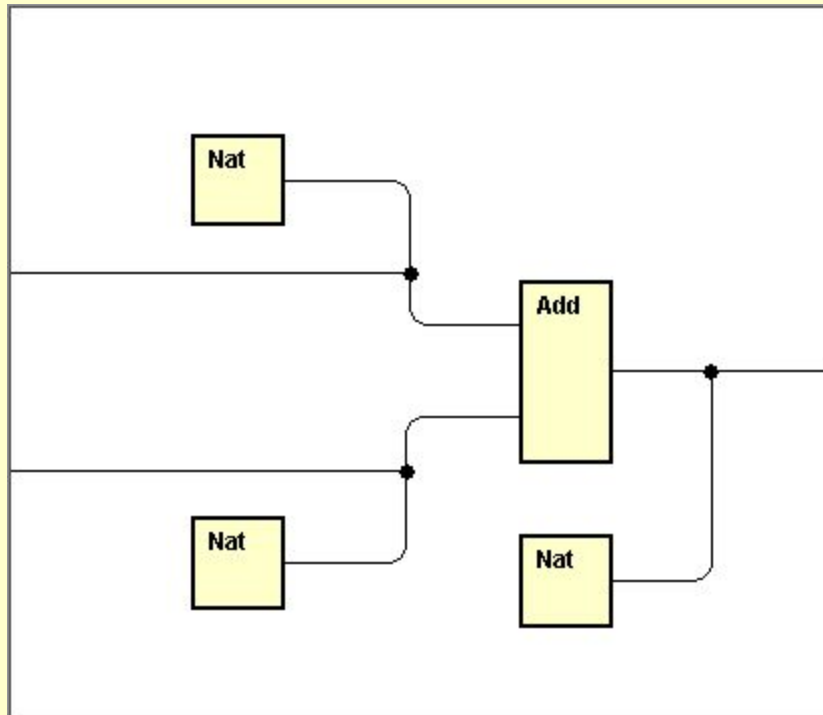
$Nat^{(0,1)}$



Типизация НО

Пример: типизированное НО $TypedAdd^{(2,1)}$

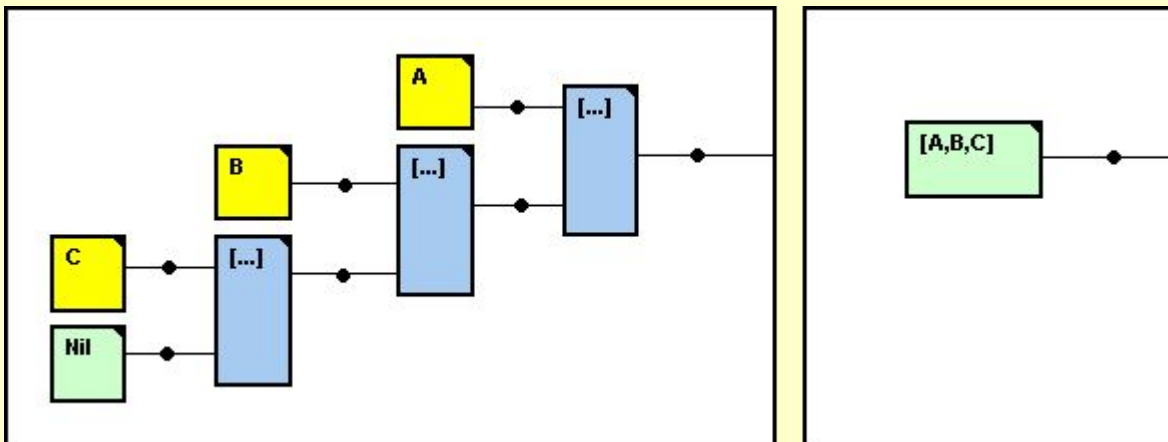
$TypedAdd = TNatAdd[Add]$



Системные типы данных

- Натуральные числа (0,1,2,...).
- Списки ([], [A(B),[D]]).
- Строки ('Some text').

Пример записи списка [A,B,C] в сетевой форме при помощи конструктора списка и системного генератора списков:



Среда программирования

- Проектный подход к организации работы с комплексами программ.
- Оригинальные технологии графического и структурированного текстового построения программ.
- Интегрированные средства разработки и отладки.
- Собственное сетевое ядро вычислений и компилятор запросов на языке S-FLOGOL.
- Возможность ограниченного импорта программ, написанных на языке Пролог.
- Реализация в среде C++ Builder 5.0.

Общий вид интерфейса системы

The screenshot displays the FLIDE software interface, which is used for developing and testing logic programs. It is divided into several main sections:

- Top Panel:** Contains the menu bar (Файл, Правка, Вид, Действия, Справка) and the title bar for the active window, "FLIDE - [Arithm.FLT]".
- Left Panel:** A tree view showing a hierarchy of "Конструкторы" (Constructors) and "Правила" (Rules). Under "Правила", there are sub-categories like "Add.1", "Add.2", "Mult.1", "Mult.2", "Nat0", "Nat2", "Nat3", "Power.1", and "Power.2".
- Center Panel:** A text editor displaying the source code for a module named "Arithm". The code defines various constructors and rules, such as:

```
MODULE Arithm=  
  (0:1)Null;  
  (1:1)Succ;  
  Nat3={:Succ(Succ(Succ(Null)))};  
  Nat2={:Succ(Succ(Null))};  
  Nat0={:Null};  
  Add={Null,x:x};  
  Add={Succ(x),y:Succ(@x,y)};  
  Mult={Null,x:Null};  
  Mult={Succ(x),y:Add(@x,y)};  
  Power={x,Null:Succ(Null)};  
  Power={x,Succ(y):Mult(x,@(x,y))};  
  QAdd={:Add(Nat2,Nat3)};  
  Mult={:Mult(Nat2,Nat3)}.
```
- Right Panel:** A project browser showing a "Demo" folder with sub-folders for "Сетевые модули" (Network modules), "Дедуктивные модули" (Deductive modules), and "Текстовые модули" (Textual modules). It lists various modules like "Arithm", "ColorMap", "Hanoi", "Lists", "Queens", "Triggers", "Chang&Yong", "Simple", "SteamRoller", "RelComp", etc.
- Bottom Panel:** A diagram editor showing a flow graph. The graph consists of several nodes connected by lines. The nodes are labeled "Succ" (green), "Mult" (purple), and "Add" (yellow). The flow starts from a source node, goes through "Succ", then "Mult", and finally "Add".

Графический редактор

The screenshot displays the FLIDE graphical editor interface. The title bar reads "FLIDE - [Hanoi.FLN]". The toolbar contains various icons for file operations, editing, and execution. The left sidebar shows a project tree with the following structure:

- Конструкторы
 - [...] (+2+:+1)
 - A (+0+:+1)
 - B (+0+:+1)
 - C (+0+:+1)
 - Nil (+0+:+1)
 - Null (+0+:+1)
 - Succ (+1+:+1)
 - to (+2+:+1)
- Константы
- Системные
- Правила
 - append.1
 - append.2
 - HANOI.1
 - HANOI.2**
- Запросы
 - Ханой(11)
 - Ханой(2)
- Результат
- Отладка

The main workspace shows a flowchart on a grid background. The flowchart consists of the following elements:

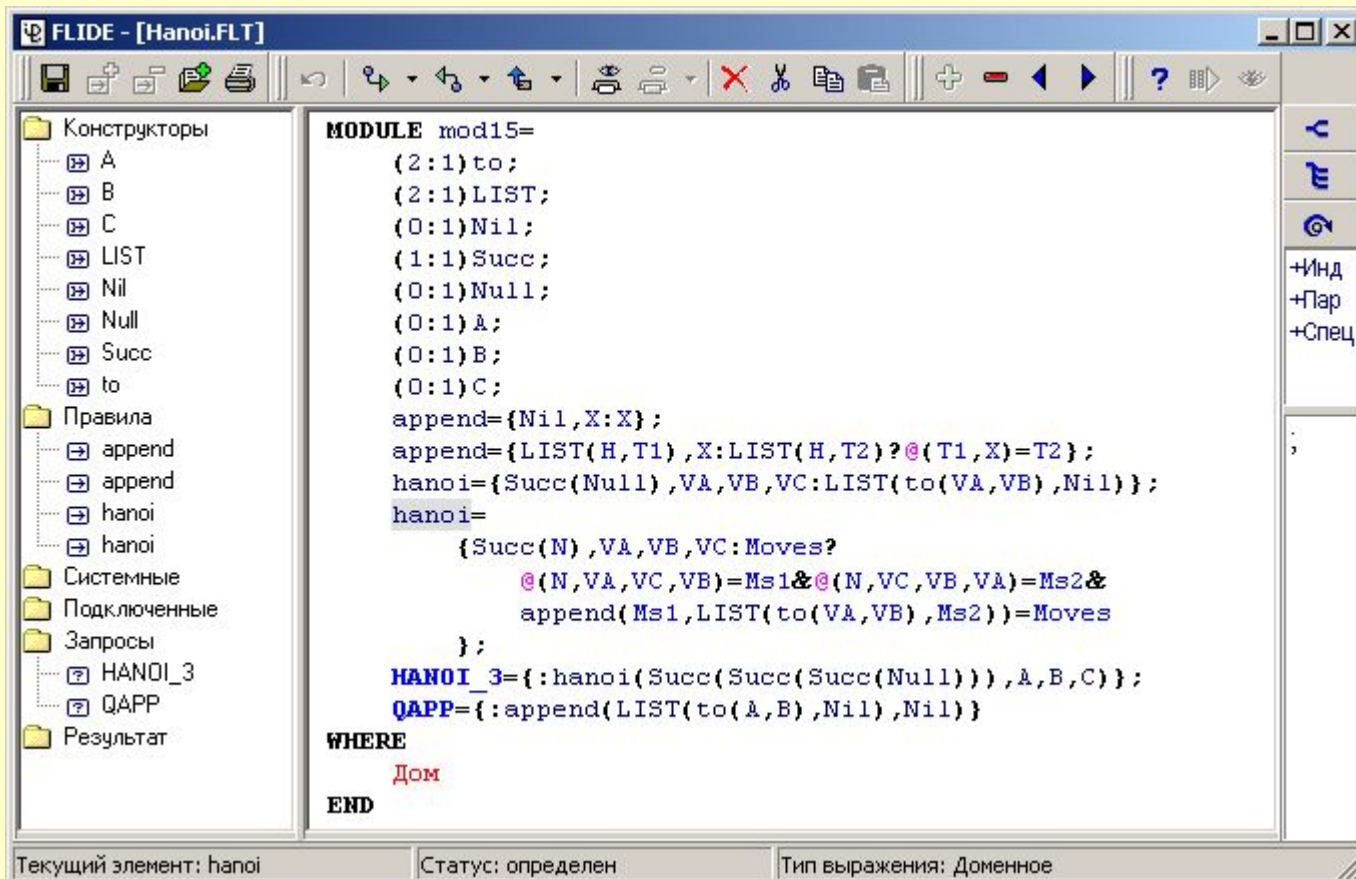
- A green box labeled "Succ" at the top left.
- Two pink boxes labeled "HANOI" with sub-labels "1" and "2" in the center.
- A blue box labeled "to" on the left side.
- A blue box labeled "[...]" on the right side.
- A yellow box labeled "append" with the number "3" below it on the far right.

Wires connect these boxes to form a complex network. The "Succ" box is connected to the top of the "HANOI 1" box. The "to" box is connected to the left side of the "HANOI 1" box. The "HANOI 1" box is connected to the top of the "HANOI 2" box. The "HANOI 2" box is connected to the left side of the "[...]" box. The "[...]" box is connected to the top of the "append" box. The "append" box is connected to the right side of the "[...]" box.

The bottom status bar displays the following text:

Начало вычисления...
Результат 1:
Выход 1: [to(A,C),to(A,B),to(C,B)]
Вычисление завершено за 0.005 секунд (9 шагов)

Текстовый редактор



Спасибо за внимание!