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

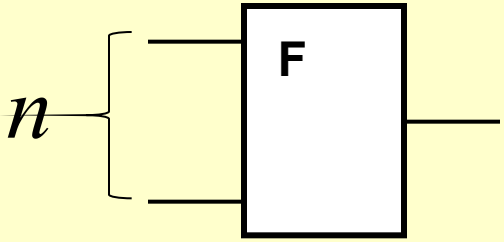
- Направленным отношением *R* арности (*n*,*m*) на носителе *D* называется множество упорядоченных пар кортежей элементов *D* длины *n* и *m*, соответственно.
- НО $R^{(n,m)}$ может моделироваться (*n*+*m*)-арным отношением R’ на D:

$$< a_1,...,a_n,a_{n+1},..,a_{n+m} > \in R' \approx < a_1,...,a_n,a_{n+1},...,a_m > \in R.$$

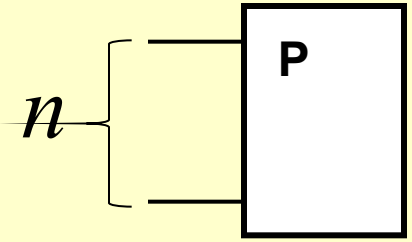
- Естественной и наглядной формой определения НО и основой эффективной организации их вычисления является представление в форме сетевых языков.
- НО характеризуются наличием фундаментальных свойств – *тотальности* и *функциональности* прямых и обратных НО.

Семантические объекты

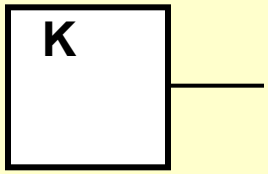
- Функция $F^{(n,1)}$



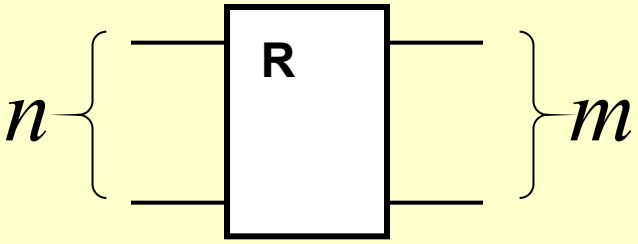
- Предикат $P^{(n,0)}$



- Константа $K^{(0,1)}$



- НО общего вида $R^{(n,m)}$



Варианты определения НО

Пример: определение натурального числа 3

- В форме графика:

```
Nat3= {:Succ(Succ(Succ(Null)))}
```

- В форме композиции:

```
Nat3= Null·Succ·Succ·Succ
```

- При помощи свертки:

```
Nat3= Null·(· I=1..3)Succ
```

Определение НО в виде графика

- В форме графика НО задается в виде

  R = **{** Терм1 **:** Терм2 **?** Формула **}** , где

  Терм1 – входной терм (образец аргументов вызова),

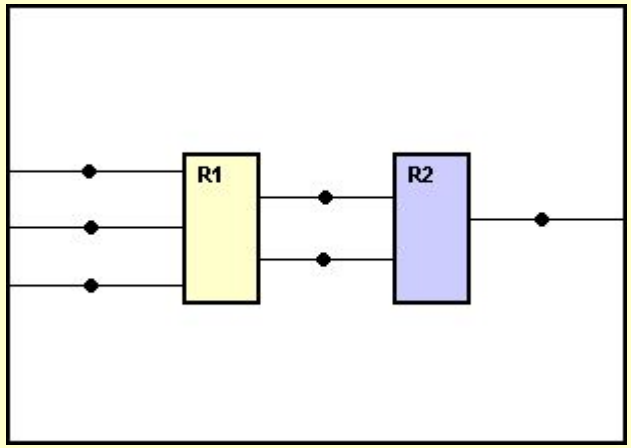
  Терм2 – выходной терм (результат вызова),

  Формула – набор ограничений вида *Терм = Терм*, *Терм <> Терм.*

Композиционное определение НО

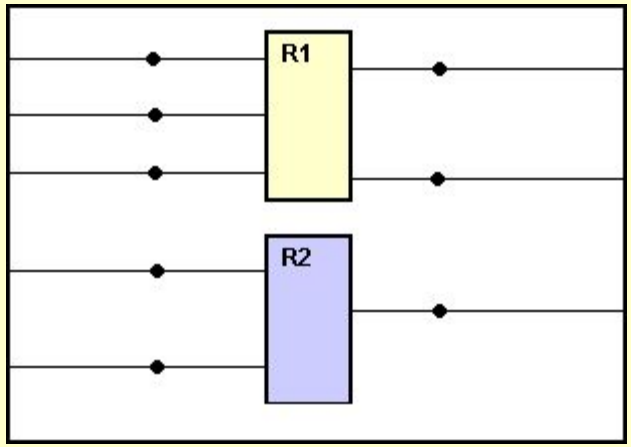
- Последовательная композиция

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



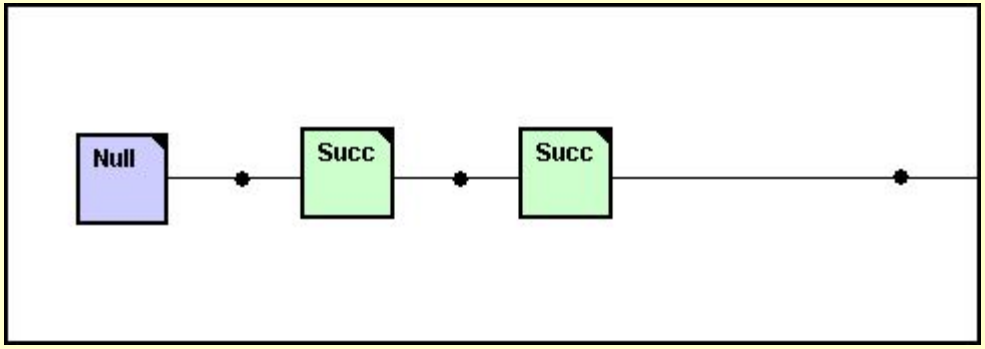
- Параллельная композиция

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

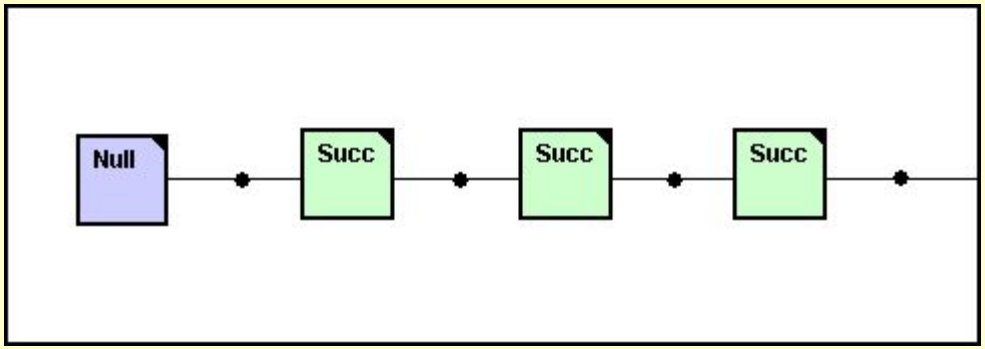


Представление натуральных чисел

Число 2 (сеть):



Число 3 (сеть):

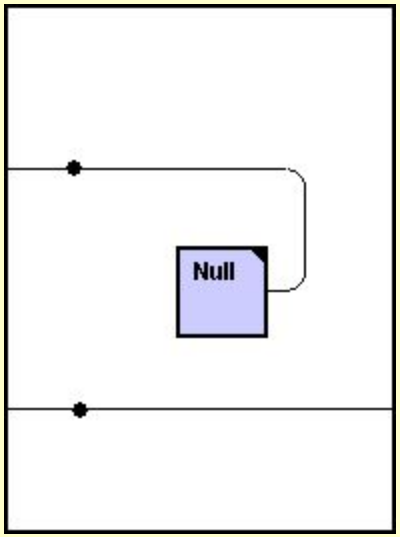


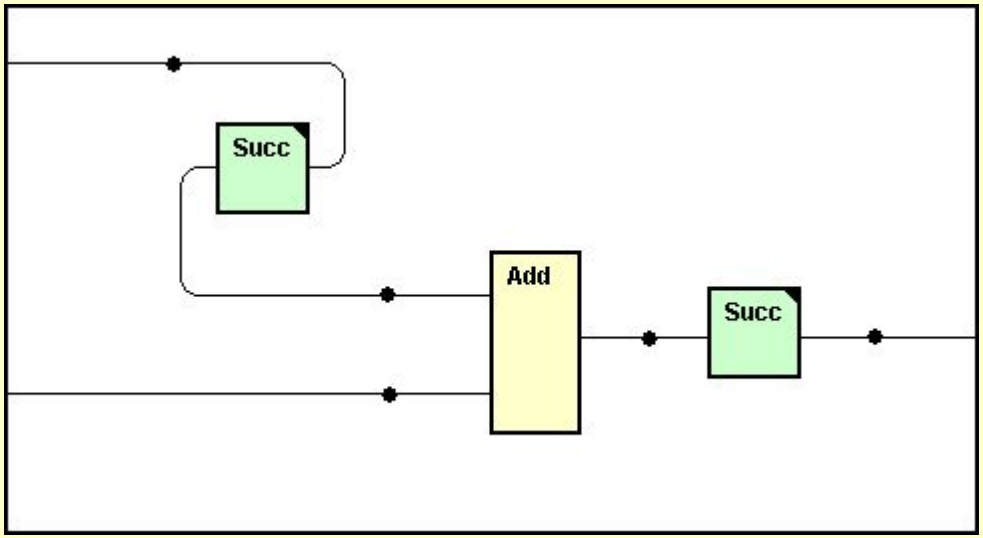
Числа 2,3 (текст):

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

Отношения над числами

Пример: НО $Add^{(2,1)}$

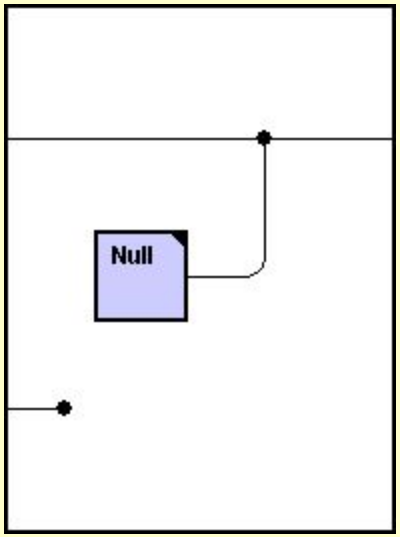


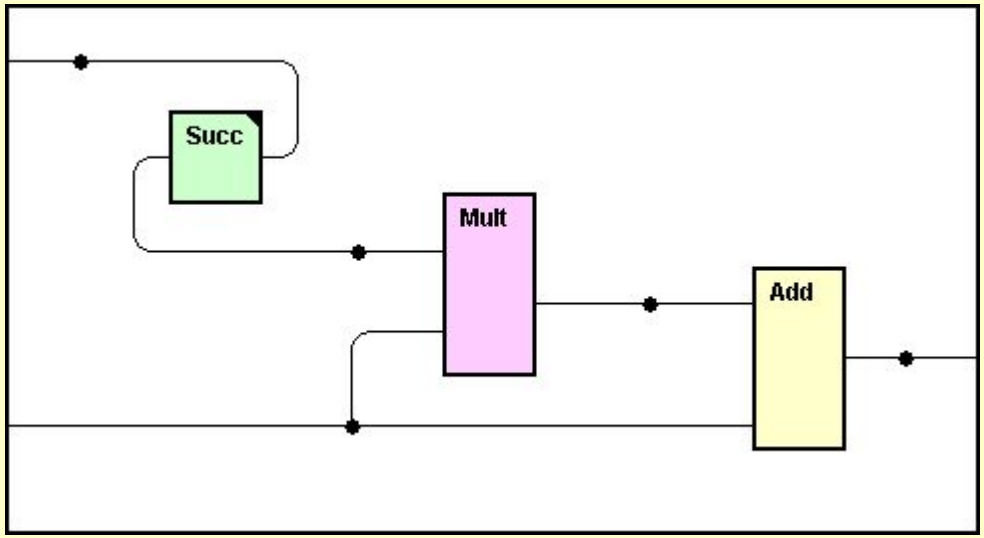


(0:1)Null;
(1:1)Succ;
(2:1)Add={Null,x:x};
(2:1)Add={Succ(x),y:Succ(@(x,y))}
или
(2:1)Add={Null,x:x}U{Succ(x),y:Succ(@(x,y))}

Отношения над числами

Пример: НО $Mult^{(2,1)}$

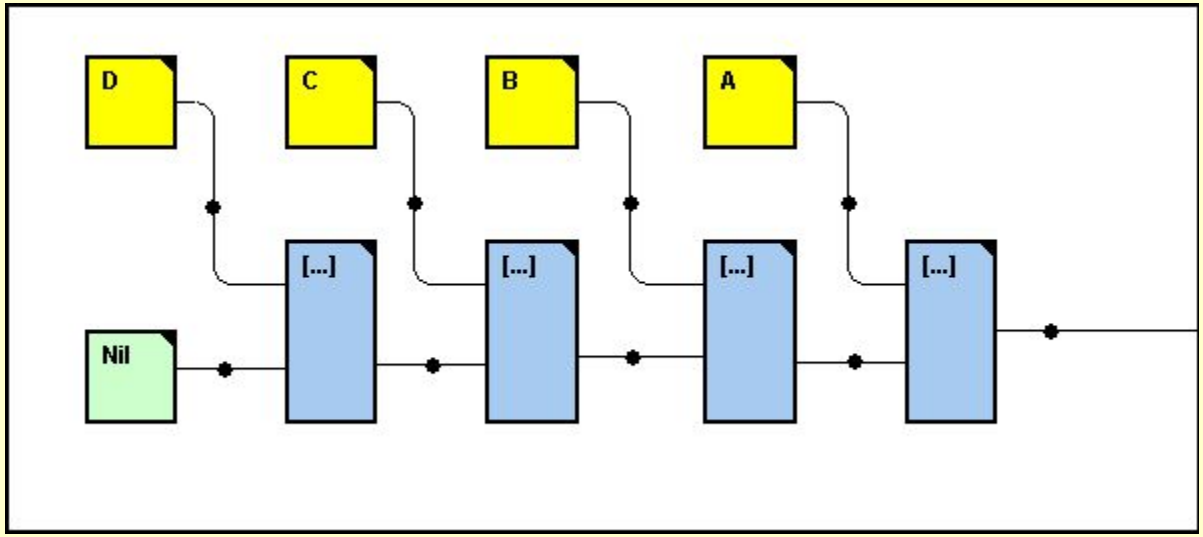




(0:1)Null;
(1:1)Succ;
(2:1)Mult={Null,x:Null};
(2:1)Mult={Succ(x),y:Add(@(x,y),y)}
или
(2:1)Mult={Null,x:Null}U{Succ(x),y:Add(@(x,y),y)}

Списки

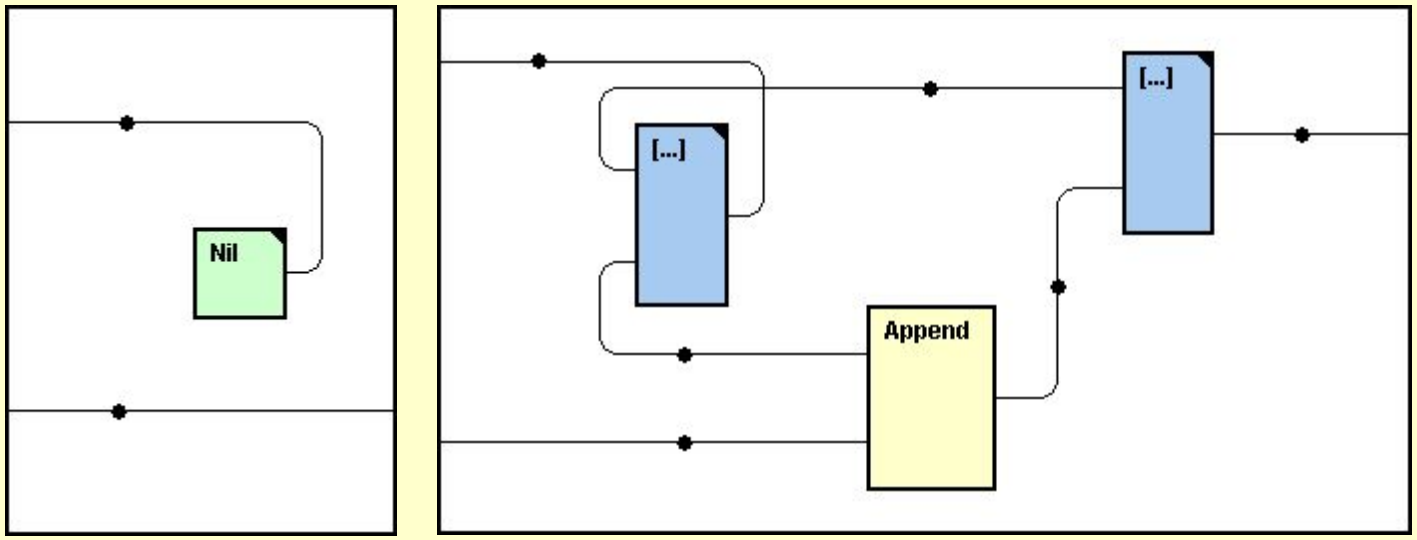
Пример: НО $List^{(0,1)}$ [A,B,C,D]




```
(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

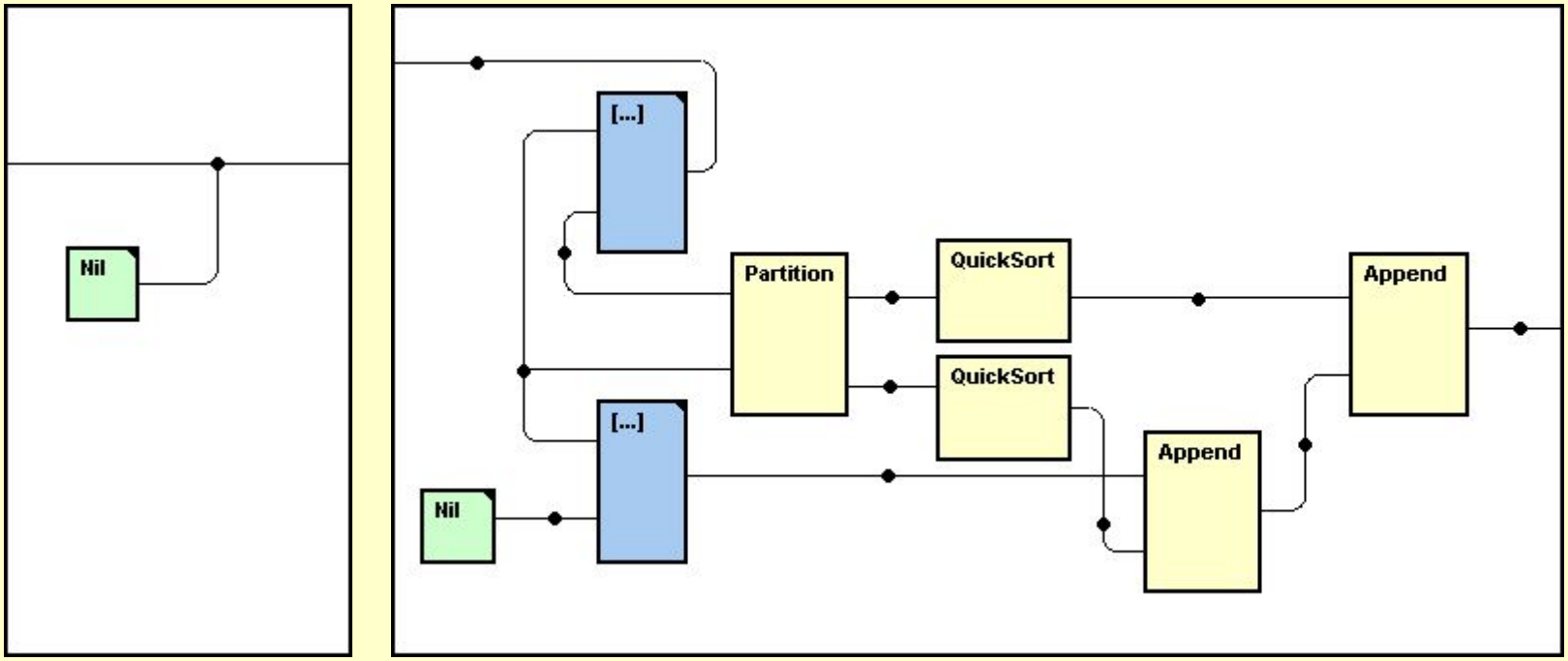
Пример: НО $Append^{(2,1)}$




```
(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= Null·(· J=1..I)Succ
```

Эквивалентное множество определений:

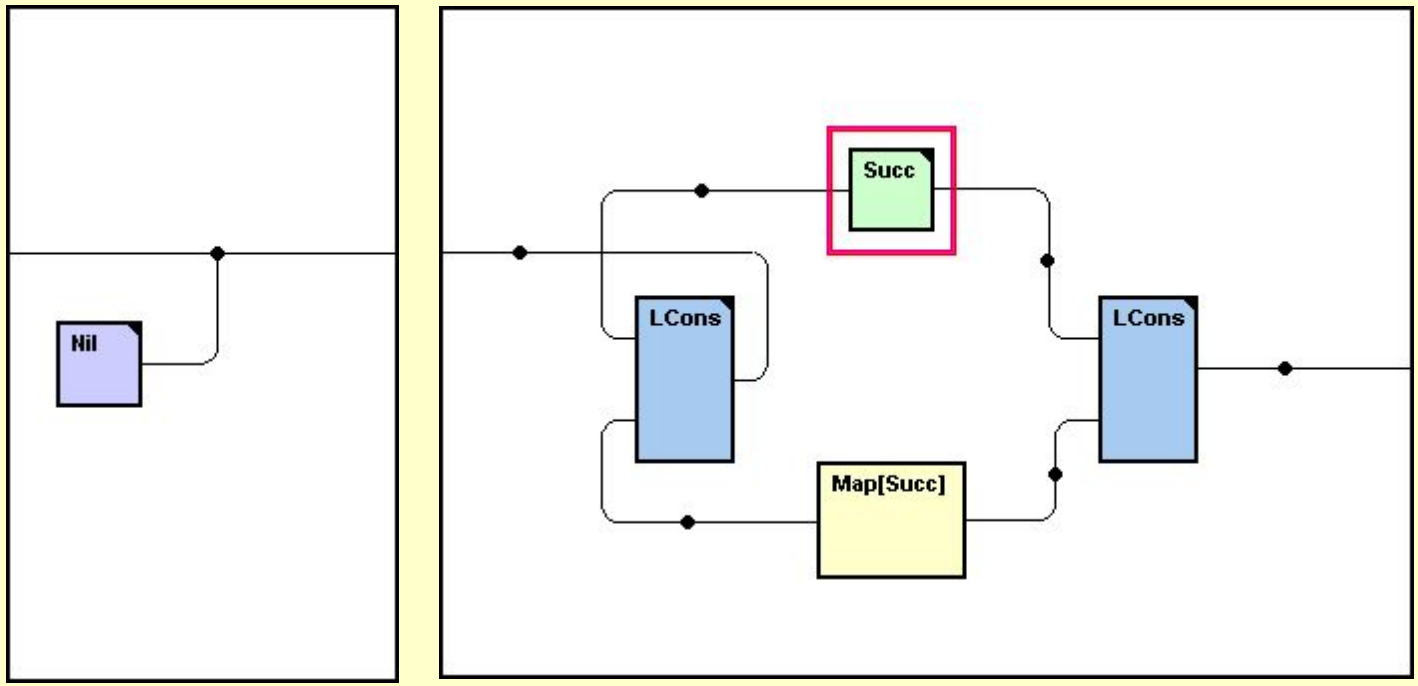
```
[0]Nat= Null;
[1]Nat= Null·Succ;
[2]Nat= Null·Succ·Succ;
[3]Nat= Null·Succ·Succ·Succ;
…
```

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 $T = (t'_1,\ldots,t'_{n'} \to t''_1,\ldots,t''_{n''})$,

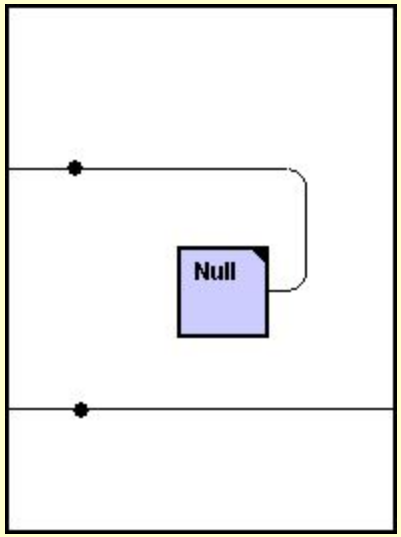
где $t'_i(t''_i)$ – сорта входных(выходных) данных, определяется в сетевой форме как:

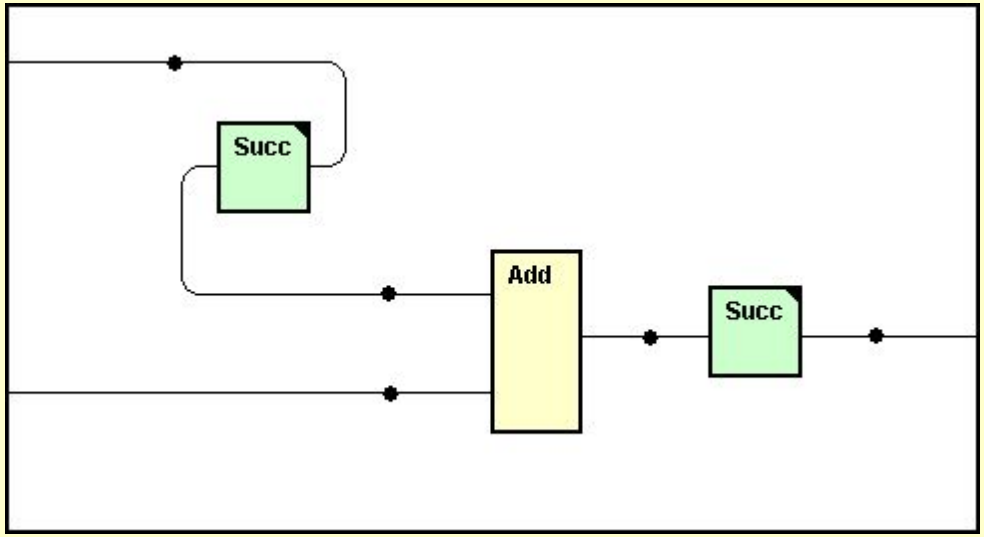
$$TypeT = \{G_{t'_1},\ldots,G_{t'_{n'}} : G_{t''_1},\ldots,G_{t''_{n''}}\} \cap «0»,$$

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'').

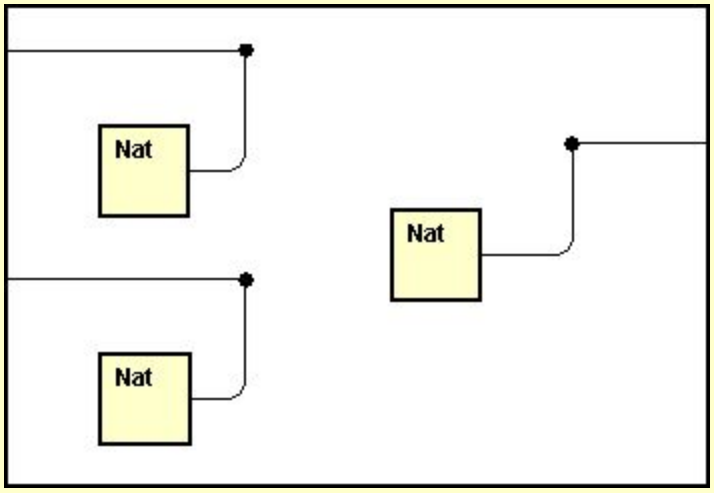
Типизация HO

Пример: HO $Add^{(2,1)}$

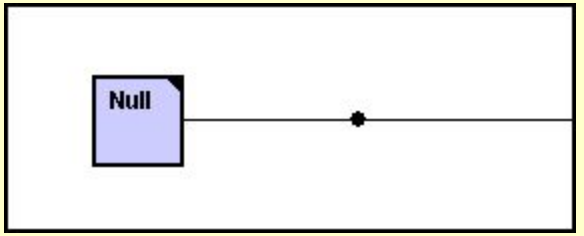


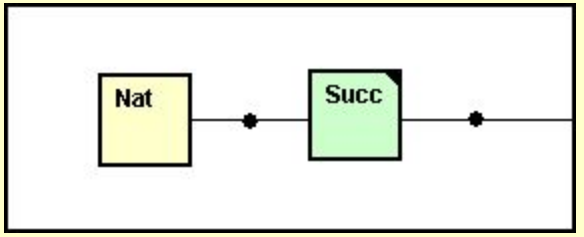


$TNatAdd^{(2,1)}$



$Nat^{(0,1)}$

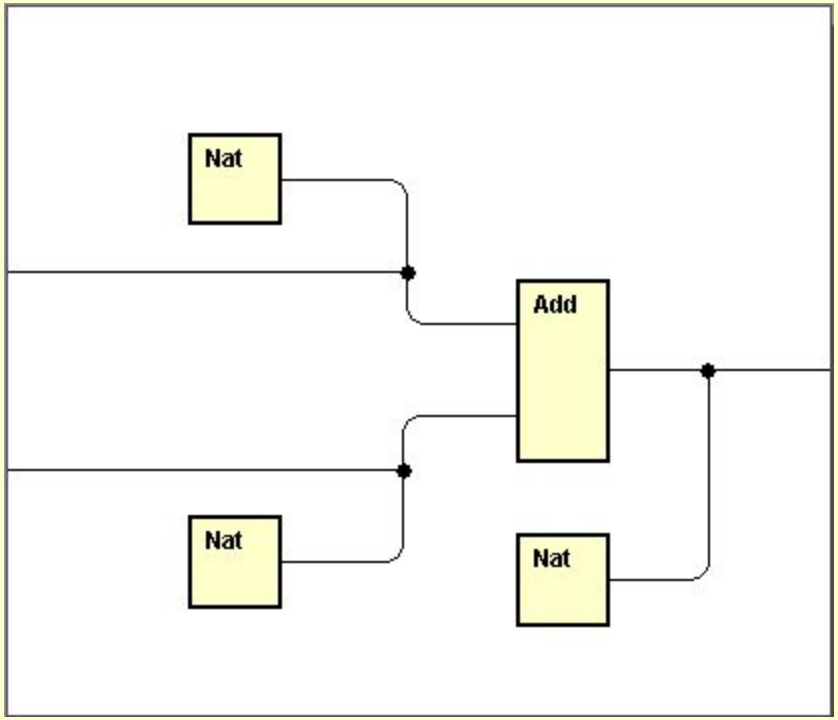




Типизация НО

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

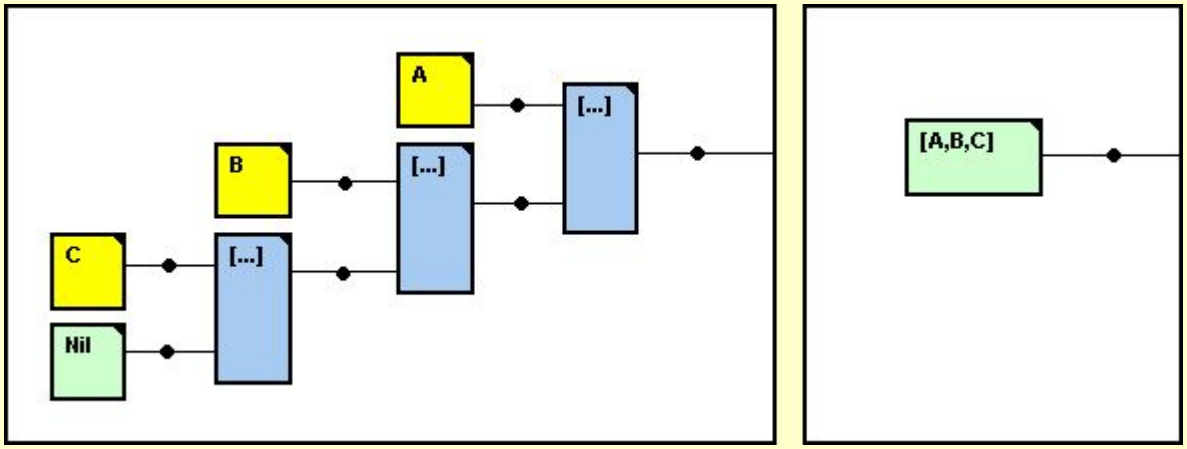
$$TypedAdd = TNatAdd[Add]$$



Системные типы данных

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

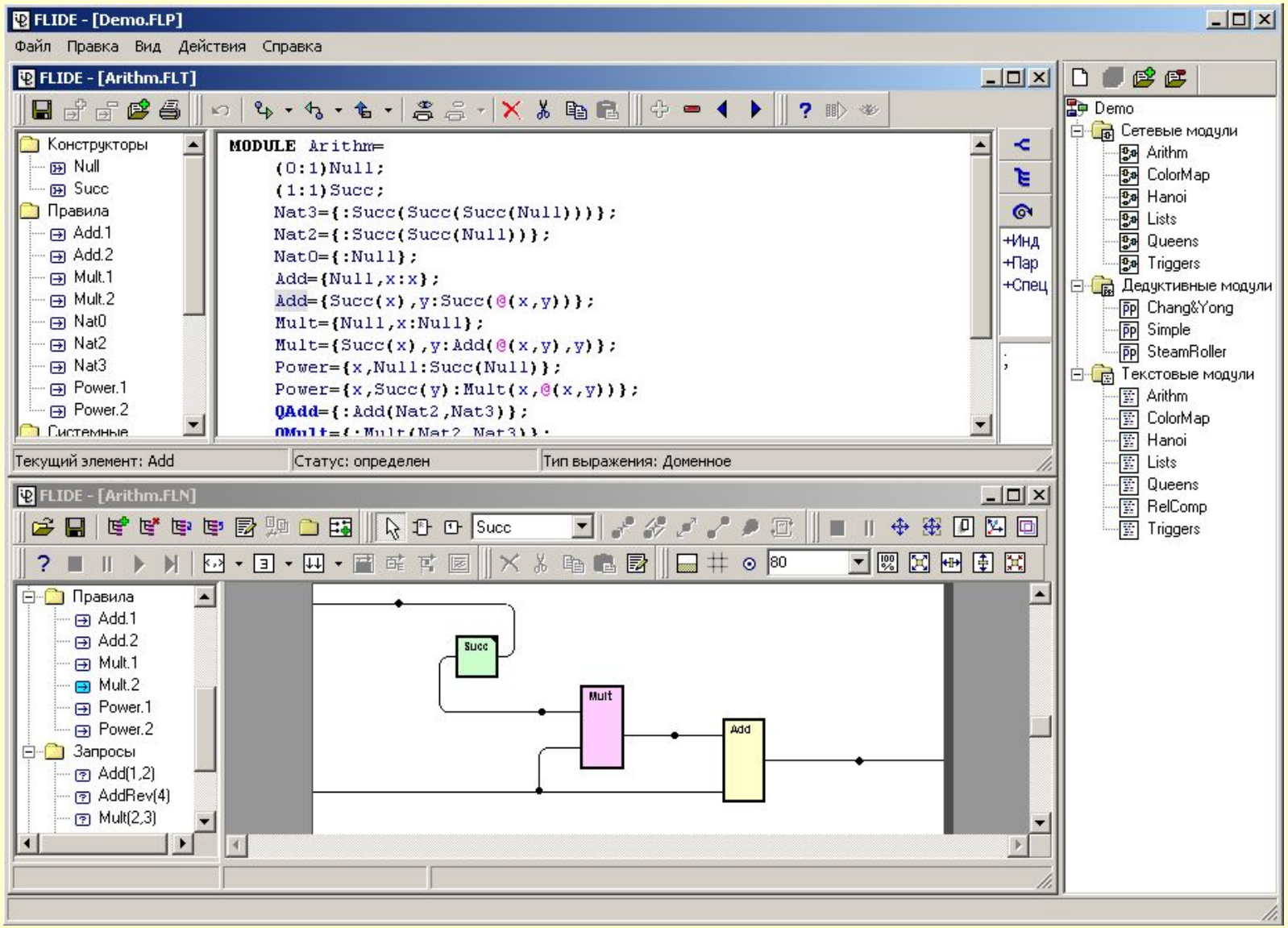
Пример записи списка [A,B,C] в сетевой форме при помощи конструктора списка и системного генератора списков:



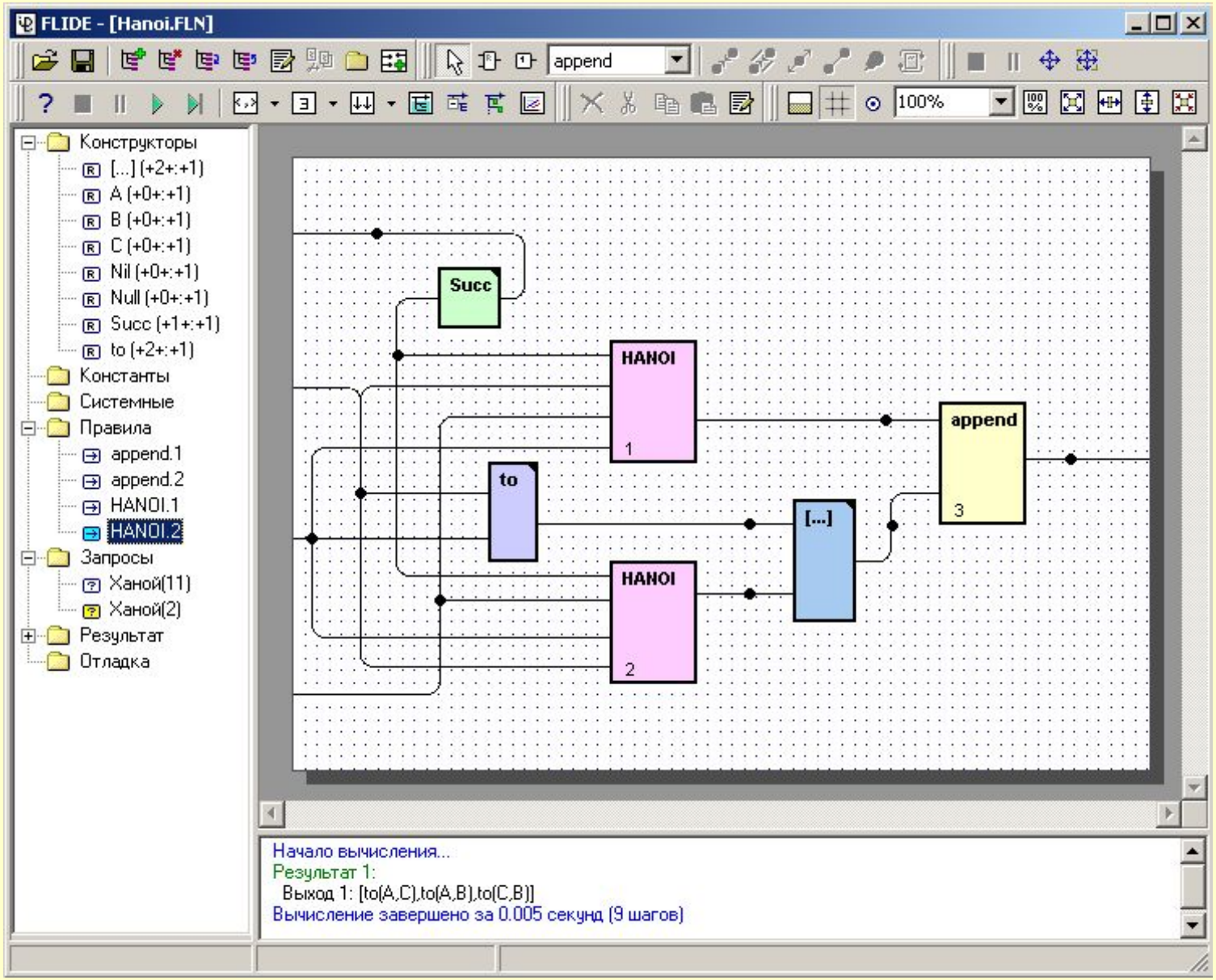
Среда программирования

- 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.

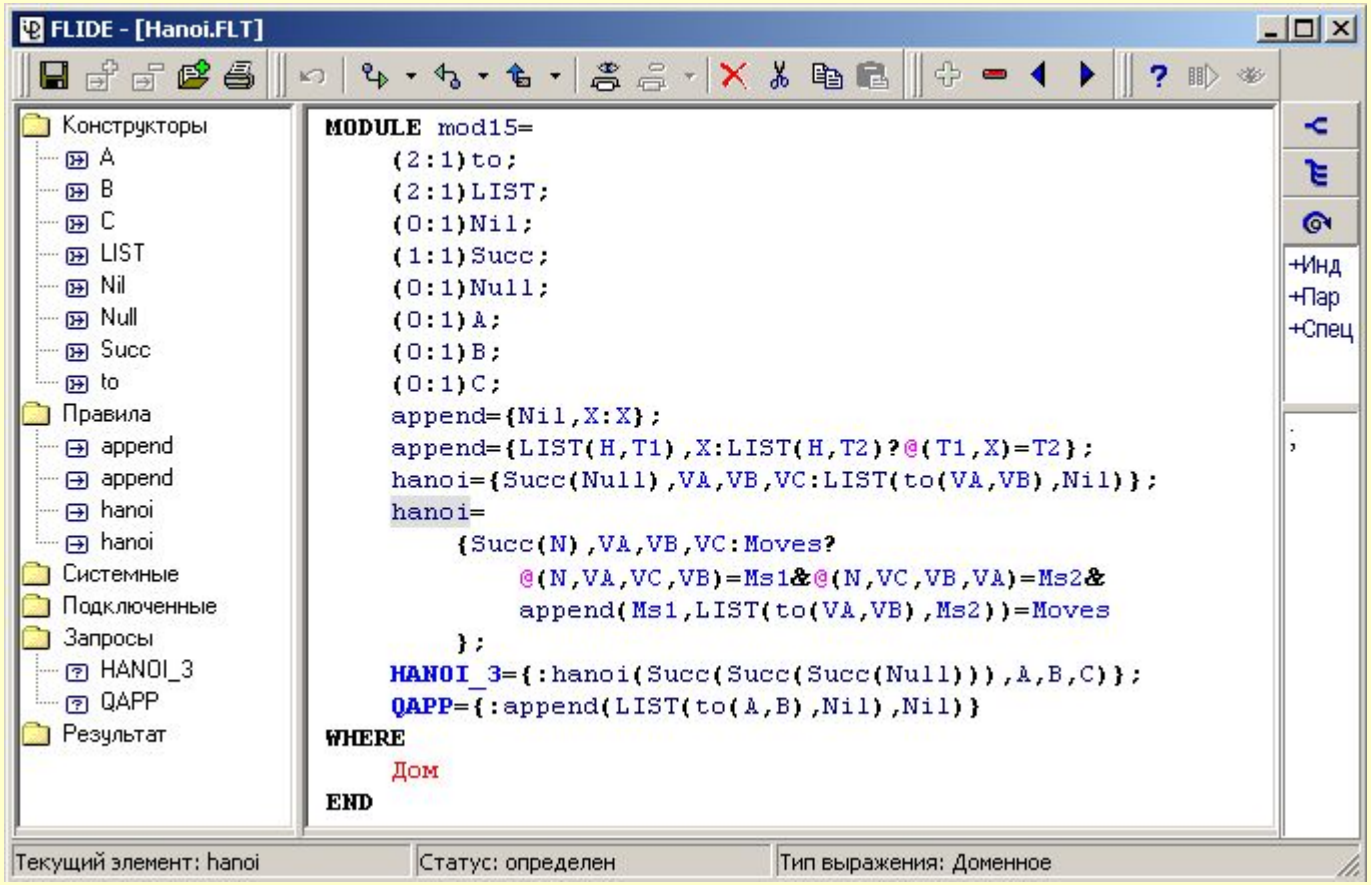
Общий вид интерфейса системы



Графический редактор



Текстовый редактор



Спасибо за внимание!