

FLIDE

FLOGOL Integrated Development
Environment

Система

функционально-логического
программирования
на языке S-FLOGOL

Бибчик Алексей Михайлович

Разработка и исследование структурно-ориентированного редактора и компилятора запросов системы функционально-логического программирования

Специальность 05.13.11 – Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

Цель работы:

создание системы функционально-логического программирования (СФЛП) на основе формализма направленных отношений (НО), предназначенной для решения задач искусственного интеллекта и для учебных целей .

Основные задачи:

- исследование языка FLOGOL и формальное описание его семантики,
- разработка основных принципов и метода компиляции FLOGOL-запросов,
- разработка специальной технологии и интерфейсных средств ввода программ,
- программная реализация и интеграция в СФЛП компилятора запросов и структурно-ориентированного редактора FLOGOL-программ.

Московский институт радиотехники, электроники и автоматики (МИРЭА ТУ)

Бибчик Алексей Михайлович

Разработка и исследование структурно-ориентированного редактора и компилятора запросов системы функционально-логического программирования

Специальность 05.13.11 – Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

Научный руководитель:
д.т.н., проф. Фальк Вадим Николаевич

Цель, задачи и структура работы

Цель работы:

создание системы функционально-логического программирования (СФЛП), основанной на формализме направленных отношений (НО) и обладающей развитыми интерфейсными средствами построения и отладки программ.

Основные задачи:

- исследование языка FLOGOL и формальное описание его семантики;
- разработка основных принципов и метода компиляции FLOGOL-запросов;
- разработка специальной технологии ввода программ и соответствующих интерфейсных средств;
- программная реализация и интеграция в СФЛП компилятора запросов и структурно-ориентированного редактора, основанного на разработанной технологии ввода;
- тестирование созданной СФЛП на наборе типовых задач декларативного программирования.

Структура диссертации:

Глава 1: Языки и системы декларативного программирования.

Глава 2: Теория направленных отношений и язык функционально-логического программирования S-FLOGOL.

Глава 3: Компилятор запросов программ на языке S-FLOGOL.

Глава 4: Структурно-ориентированный редактор.

Научная новизна и практическая значимость

Научная новизна:

1. Предложена система ограничений языка FLOGOL, позволяющая выполнить компиляцию программы в конечную контекстно-свободную сетевую грамматику (КССГ) и значительно упрощающая реализацию языка без существенного снижения его основных выразительных возможностей.
2. На основе предложенной системы ограничений разработан язык S-FLOGOL, формально описаны его синтаксис и семантика.
3. Сформулированы основные принципы и разработан метод компиляции запросов программ на языке S-FLOGOL, опирающийся на формальное описание семантики.
4. Предложена оригинальная технология дедуктивного ввода программ, позволяющая минимизировать ручной ввод, полностью исключить синтаксические ошибки и значительно снизить количество семантических ошибок при вводе программы.

Практическая значимость работы заключается в возможности использования созданной совместно с Бебчиком Ан. М. СФЛП для проведения исследовательских работ с применением формализма НО, а также для обучения студентов основам декларативного программирования.

Практическая значимость работы подтверждается использованием созданной СФЛП в учебном процессе МИРЭА и МАИ.

Язык S-FLOGOL (Small FLOGOL)

- Основан на теории направленных отношений (НО).
- Поддерживает аппликативный и композиционный стили программирования.
- Обладает развитой схемной надстройкой, включающей:
 - индексированные имена объектов,
 - условные конструкции,
 - свертки.
- Допускает динамическую типизацию объектов.
- Поддерживает параметризацию определений НО.
- Позволяет строить многомодульные программы.
- Обладает средствами ограничения области видимости НО.

Основные ограничения по сравнению с языком FLOGOL:

- Упрощенная модульная структура.
- Исключение пользовательских связей.
- Упрощенный механизм параметризации (компиляция).
- Отсутствие открытых интервалов в свертках (компиляция).

Формализм направленных отношений

Направленным отношением (НО) R арности (n', n'') на носителе D называется множество упорядоченных пар кортежей элементов D длины n' и n'' , соответственно.

Пример графика НО сложения:

$$Add^{(2,1)} = \{(x'x'', y) \mid y = x' + x''\}$$

Обратное отношение:

$$\forall \alpha \forall \beta ((\alpha, \beta) \in R \approx (\beta, \alpha) \in R^{-1})$$

Пример графика обратного НО для НО $Add^{(2,1)}$:

входной кортеж		выходной кортеж
x'	x''	y
0	0	0
0	1	1
1	1	2
...

входной кортеж	выходной кортеж	
y	x'	x''
0	0	0
0	1	1
1	1	2
...

Свойства НО и представление семантических объектов

- НО R называется *функциональным* ($F(R)$), если

$$\forall \alpha \forall \beta' \forall \beta'' ((\alpha, \beta') \in R \ \& \ (\alpha, \beta'') \in R \supset \beta' = \beta'')$$

- НО R называется *тотальным* ($T(R)$), если

$$\forall \alpha \exists \beta ((\alpha, \beta) \in R)$$

Пример представления некоторых семантических объектов:

НО	Арность	$F(R)$	$T(R)$	$F(R^{-1})$	$T(R^{-1})$
Утверждение	(0,0)	+		+	
Константа	(0,1)	+	+	+	
Функция (тотальная)	(n ,1)	+	+		
Частичная функция	(n ,1)	+			
Предикат	(n ,0)	+			
Отображение	(m , n)				

Функционально-логический язык S-FLOGOL

FLOGOL – декларативный язык высокого уровня, основанный на формализме НО. Автор – Фальк В.Н.

Форма реализации – смешанная: компиляция в промежуточное представление и его последующая интерпретация вычислительной подсистемой.

Промежуточное представление – контекстно-свободная сетевая грамматика (КССГ), задаваемая четверкой $\langle B_T, B_H, \alpha, R \rangle$:

B_T – терминальный базис сортов,

B_H – нетерминальный базис сортов,

$\alpha \in B_H$ – аксиома КССГ,

R – множество правил вида $b \rightarrow S$, где $b \in B_H$, S – **сеть** в базисе $B_T \cup B_H$.

Сорт – имя, входная и выходная арности, а также наличие свойств функциональности и тотальности НО.

Терминальный базис – НО, определенные вне КССГ,

Нетерминальный базис – НО, определенные правилами КССГ,

Аксиома – НО, заданное КССГ.

Сетевое представление НО

Сеть задается пятеркой $\langle P, I, O, E, U \rangle$, где

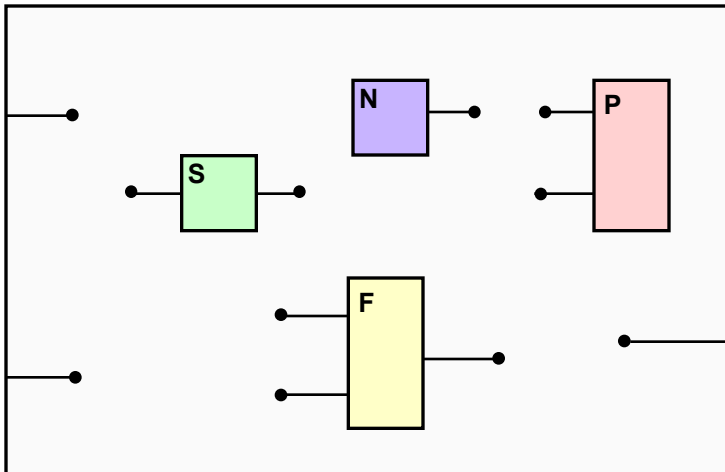
P – множество точек сети,

I, O – входной и выходной кортежи точек,

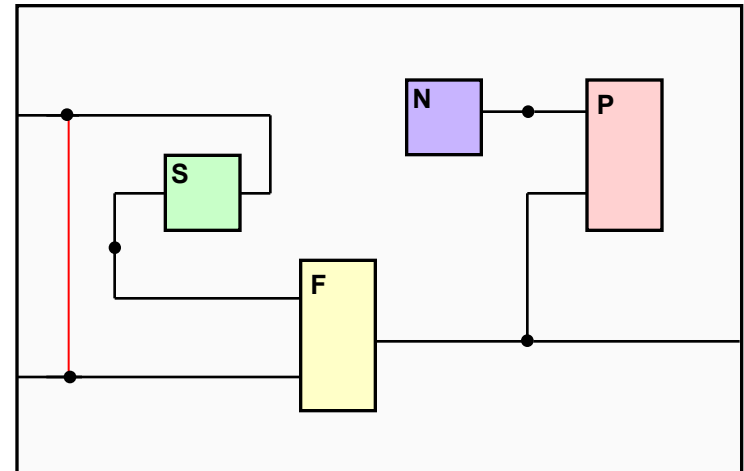
E – множество элементов сети,

U – граф семантического различия.

Сеть без ограничения на интерпретацию точек:



Сеть с отождествленными точками и ребром dif-графа:



Реляционная интерпретация сети – НО, заданное разметкой точек ее входного и выходного кортежей.

Система ограничений. Язык S-FLOGOL (Small FLOGOL)

Проблема:

компиляция FLOGOL-программ в промежуточное представление в виде КССГ с конечным множеством правил невозможна.

Решение:

введение системы ограничений и определение на ее основе языка S-FLOGOL.

Основные ограничения:

- 1) исключение открытых интервалов сверток,
- 2) запрет транзитной передачи отдельных параметров.

Дополнительные ограничения (упрощение реализации):

- 1) упрощенная внутренняя организация модуля,
- 2) исключение виртуальных определений и наследования,
- 3) упрощенная форма параметризации,
- 4) отсутствие пользовательских связей и внешних определений.

Расширения:

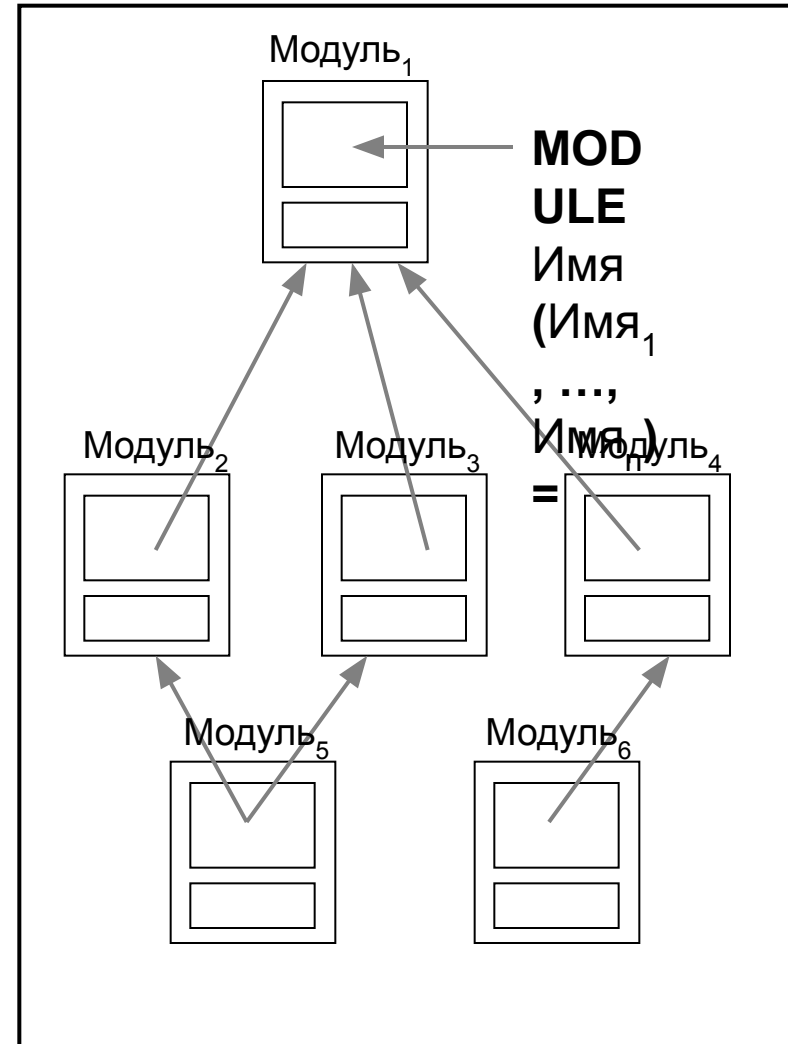
- 1) транзитная передача всех параметров вызова,
- 2) Поддержка системных типов данных и отношений.

Программа на языке S-FLOGOL

Программный модуль:



Программа:



Пример программы (модуль “Arithm”)

MODULE Arithm =

//Конструкторы

(+0+:+1)Null;

(+1+:+1)Succ;

//Аппликативные определения

Nat0={:Null};

Nat1={:Succ(Null)};

Add={Null,x:x};

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

//Композиционные определения

--- = {x:x};

[0]Nat=Null;

(K=1..100)

[K]Nat=Null·(·J=1..K)Succ;

AddC= (~Succ # ---)·@·Succ U (~Null # ---);

//Запросы к системе

QAdd={:Add(Nat0,Nat1)};

QAddC=([1]Nat#[2]Nat)·AddC;

END

имя модуля

комментарии

индексированные
имена НО

свертка

описания НО

Сетевая интерпретация программ

Программа:

OBJECT Null;

CONSTRUCTOR(1)Succ;

Nat1=Null·Succ;

Nat2=Null·Succ·Succ;

Add={Null,x:x};

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

QUERY=(Nat1#Nat2)·Add



Базис КССГ:

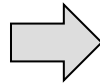
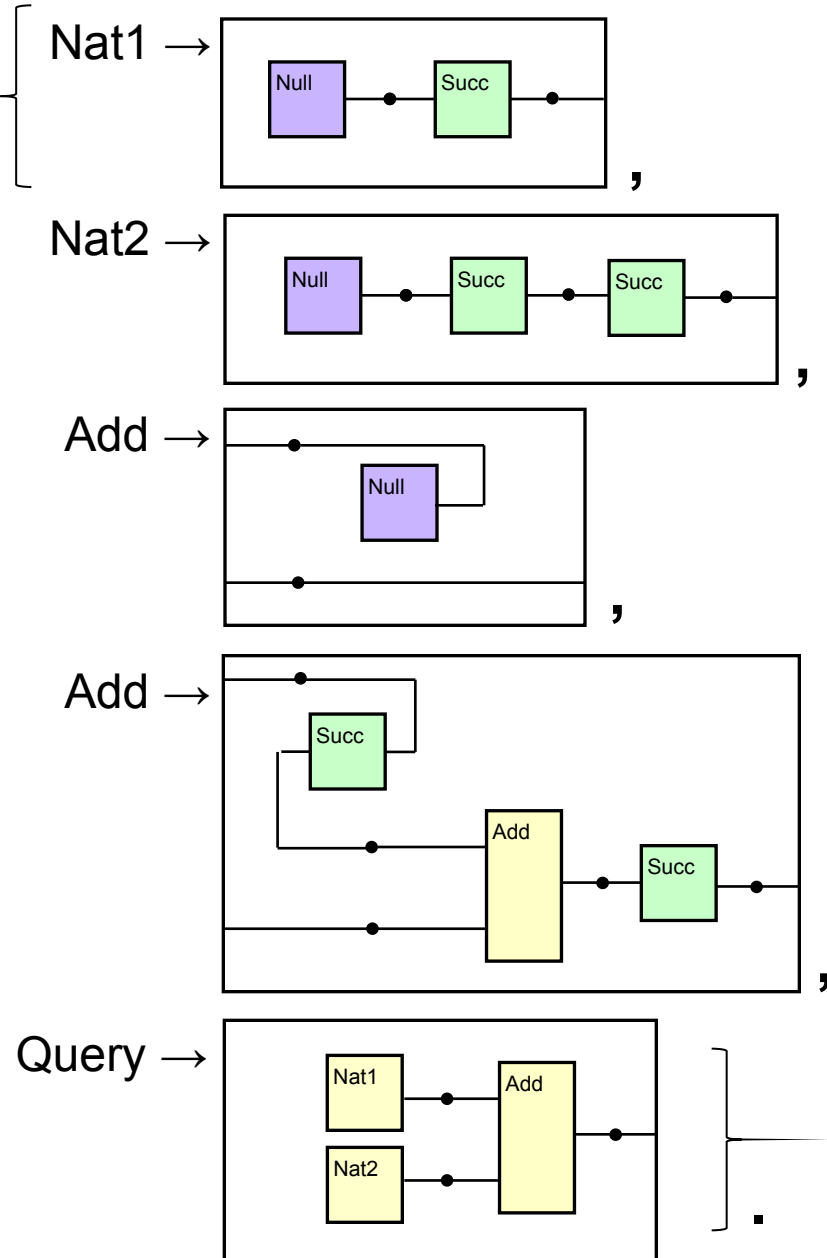
$$B_m = \{\text{Null}^{(0,1)}, \text{Succ}^{(1,1)}\},$$

$$B_H = \{\text{Nat1}^{(0,1)}, \text{Nat2}^{(0,1)}, \text{Add}^{(2,1)}, \text{QUERY}^{(0,1)}\}.$$

Аксиома КССГ:

$$\alpha = \text{QUERY}^{(0,1)}.$$

Правила КССГ $R =$



Описания НО

Описания НО задаются доменными выражениями (Дом):

- Объявление НО:

Спец Имя

(+0+:+1) Null
(+1+:+1) Succ

Ограничение

Исключена внешняя
пользовательская
интерпретация.

- Определение НО:

Спец Имя = Рел

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

- Объединение описаний НО: Дом ; Дом

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

Определение НО

Определение НО задается реляционным выражением (Рел):

· Вызов: **ИмяНО** Пример: $R1 = R2$

· График: $\{\text{Терм} : \text{Терм} ? \text{Формула}\}$

Входной терм задает входной кортеж НО

Выходной терм задает выходной кортеж НО

Логическая формула описывает дополнительные ограничения на интерпретацию переменных термов

Пример: $\text{Add} = \{ \text{Succ}(x), y : \text{Succ}(\text{Add}(x, y)) \}$

· Терм:

- переменная
- вызов НО
- соединение термов

Переменная

Примеры:

x

ИмяНО(Терм)

$\text{Succ}(\text{Null})$

Терм , Терм

x , y

Succ(Терм)

Add(Терм)

Терм , Терм

x y

Примеры аппликативных описаний НО

- Конструкторы натуральных чисел:

OBJECT Null;

CONSTRUCTOR(1) Succ

- Сложение натуральных чисел:

Add = {Null,x:x};

Add = {Succ(x),y:Succ(Add(x,y))}

- Удаление элемента из списка:

Del = {x,Cons(x,y) : y};

Del = {x,Cons(y,ys) : Cons(y,Del(x,ys)) ? x<>y }

↑
формула

- Вычисление длины списка:

Length = {Nil : Null};

Length = {Cons(x,xs) : Succ(Length(xs))}

· Терм:

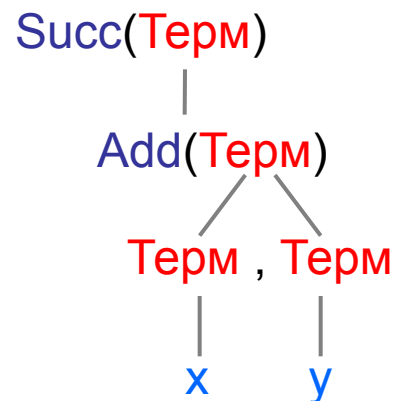
Пример: $\text{Succ}(\text{Add}(x,y))$

Переменная

Имя(Терм)

Терм , Терм

Терм | Терм



· Композиция НО: $\text{Рел} \otimes \text{Рел}$, где

$\otimes \in \{., \#, \rightarrow, \nabla, *, \cup, \cap\}$

· Инверсия отношения: $\sim \text{Рел}$

· Комбинаторные константы:

$\text{---} = \{x:x\}$

$\text{-/-} = \{x:y?x<>y\}$

$\text{<--} = \{x\}$

$\text{-->} = \{x:\}$

$\text{--<} = \{x:x,x\}$

$\text{>--} = \{x,x:x\}$

Композиционное описание НО

Спецификация НО

Общая форма: Спец \rightarrow (+ Ar + : + Ar +)

функциональность

входная арность

тотальность

обратная тотальность

выходную арность

обратная функциональность

Ключевая спецификация:

Ключевое слово	Спецификация
OBJECT	(+0+:+1)
CONSTRUCTOR (n)	(+n+:+1)
OPERATOR	(1+:+1)
PREDICATE (n)	(+n:0)
FUNCTION (n)	(+n+:1)

Примеры:

(+0+:+1)Null

OBJECT Null

(+1+:+1)Succ

CONSTRUCTOR(1)Succ

Полиморфизм: (2:1)Add

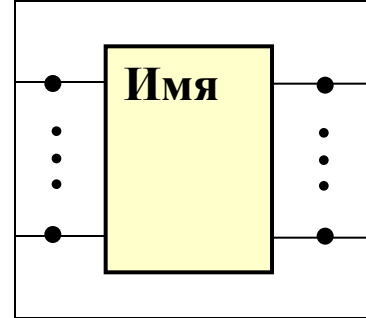
(3:0)Add

Неявная спецификация определений: (2:1)Add={Succ(x),y:Succ(Add(x,y))} U {Nil,x:x}

Определение НО

Определение НО задается реляционным выражением (**Рел**):

- Вызов по имени: **Рел** → **Имя**
- Рекурсивный вызов: **Рел** → @
- Вызов параметра: **Рел** → <<Ар>>



- График (аппликативная форма):

Рел → {**Терм** : **Терм** ? **Формула**}

Входной терм – аргументы вызова.

Выходной терм – результат вызова.

Формула – ограничения на интерпретацию переменных графика.

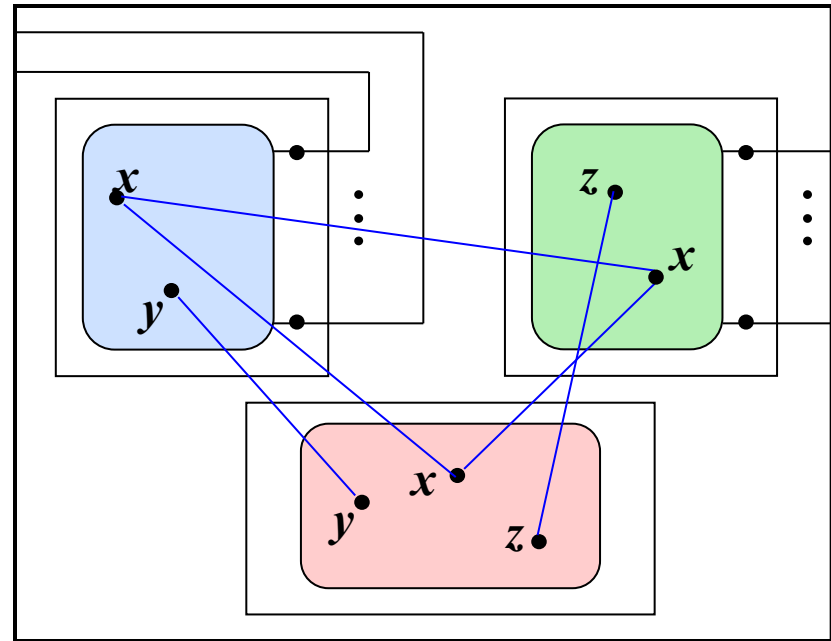


График: терм и формула

· Терм:

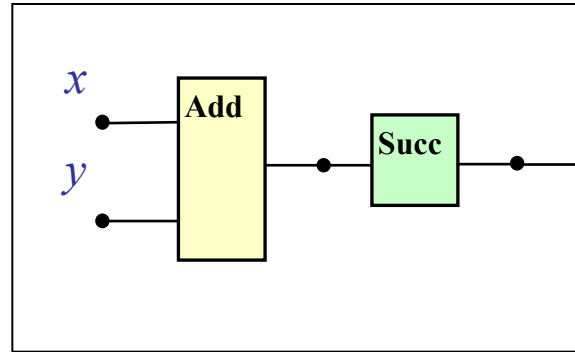
Переменная

Имя(Терм)

Терм , Терм

Терм | Терм

Пример: $Succ(Add(x,y))$



$Succ(Терм)$

$Add(Терм)$

Терм , Терм

x

y

· Формула:

Терм=Терм

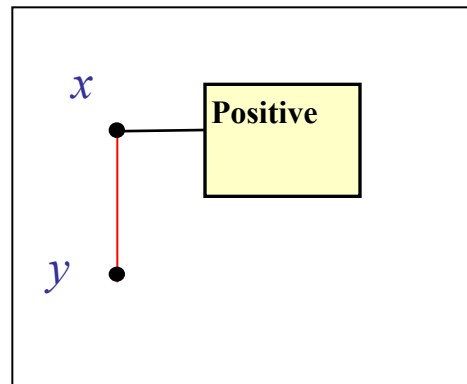
Терм<>Терм

Имя(Терм)

Формула & Формула

Формула | Формула

Пример: $Positive(x) \& x <> y$



Формула & Формула

$Positive(Терм)$

Терм<>Терм

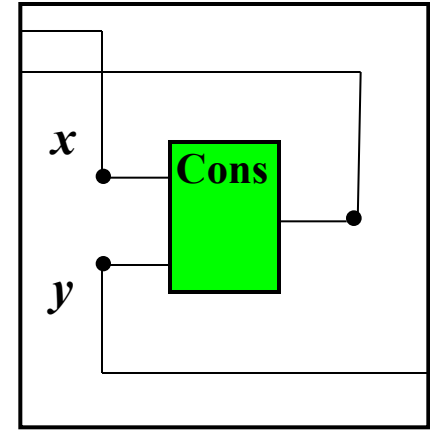
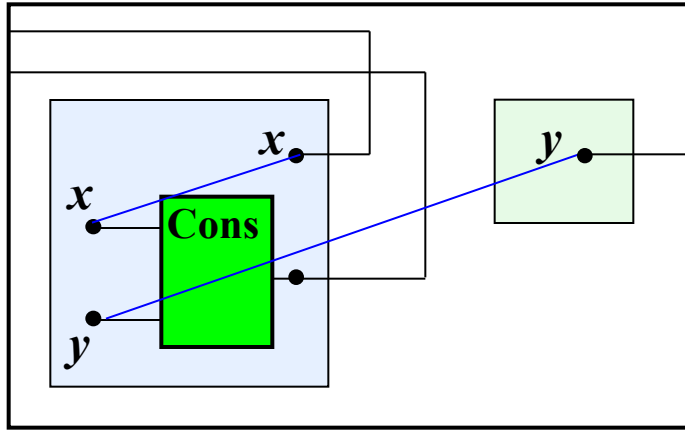
x

x

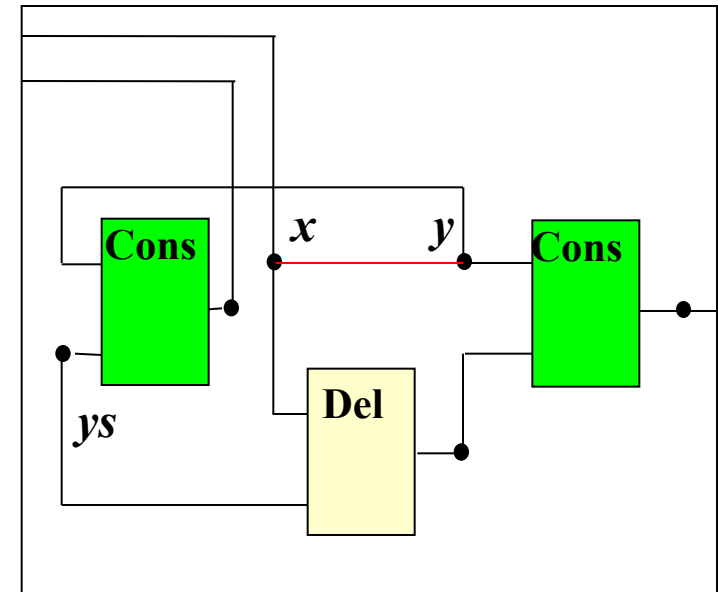
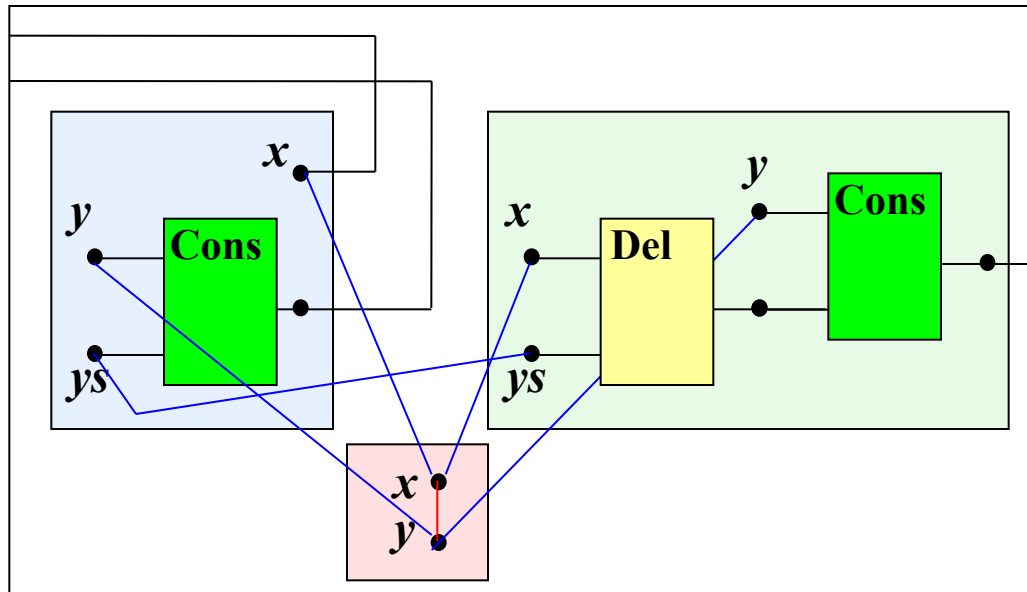
y

Пример. Определение НО «Del»

$$\text{Del} = \{x, \text{Cons}(x, y) : y\}$$

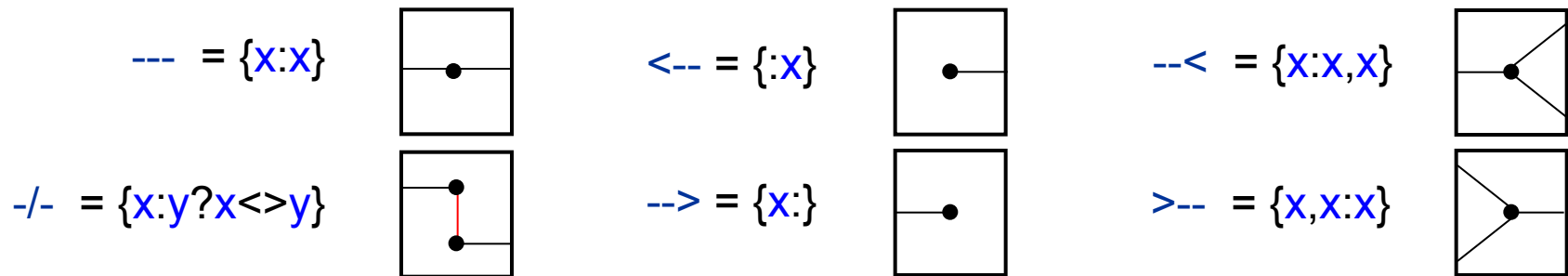


$$\text{Del} = \{x, \text{Cons}(y, \text{ys}) : \text{Cons}(y, \text{Del}(x, \text{ys})) ? x <> y\}$$



Композиционное программирование

- Композиция отношений: $\text{Рел} \rightarrow \text{Рел} \otimes \text{Рел}$, где $\otimes \in \{., \#, \rightarrow, \nabla, *, \cup, \cap\}$
- Инверсия отношения: $\text{Рел} \rightarrow \sim \text{Рел}$
- Комбинаторные константы:



Пример композиционного определения НО сложения:

ОБЪЕКТ Null ;

CONSTRUCTOR(1) Succ ;

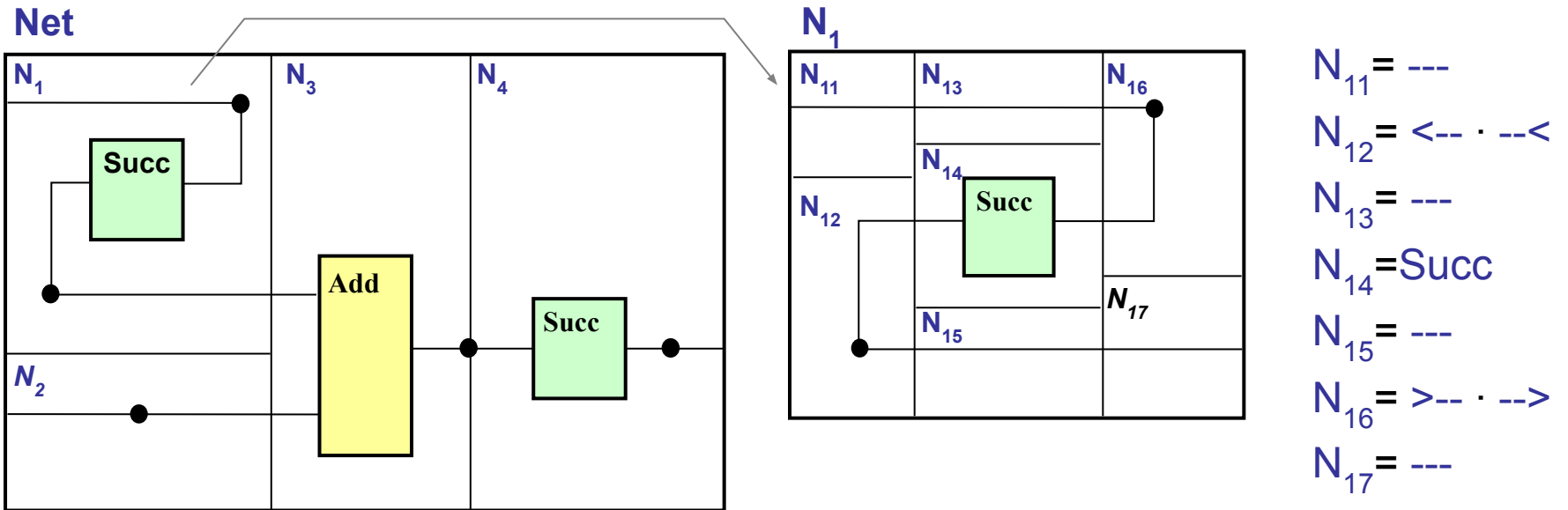
$\text{Add} = (--- \# \text{Null} \# ---) \cdot (>-- \# ---) \cdot (>-- \# ---)$;

$\text{Add} = ((--- \# <--- \cdot --<) \cdot (--- \# \text{Succ} \# ---) \cdot (>-- \cdot --> \# ---) \# ---) \cdot \text{Add} \cdot \text{Succ}$;

или $\text{Add} = (\sim \text{Succ} \# ---) \cdot \text{Add} \cdot \text{Succ} \cup (\sim \text{N} \# ---)$;

Композиционное программирование (продолжение)

Преобразование сети:



$$Net = (N_1 \# N_2) \cdot N_3 \cdot N_4$$

$$N_1 = (N_{11} \# N_{12}) \cdot (N_{13} \# N_{14} \# N_{14}) \cdot (N_{16} \# N_{17})$$

$$N_2 = \text{---}$$

$$N_3 = \text{Add}$$

$$N_4 = \text{Succ}$$

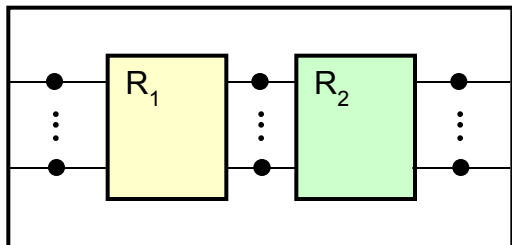


$$Net = ((\text{---} \# \text{<---} \cdot \text{---<}) \cdot (\text{---} \# \text{Succ} \# \text{---})) \cdot (\text{>---} \cdot \text{--->} \# \text{---}) \# \text{---} \cdot \text{Add} \cdot \text{Succ}$$

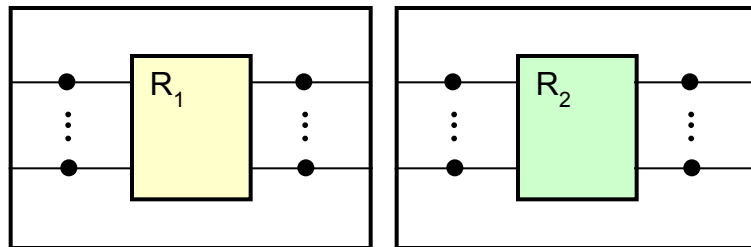
или $Net = (\sim \text{Succ} \# \text{---}) \cdot \text{Add} \cdot \text{Succ}$

Операции композиции НО

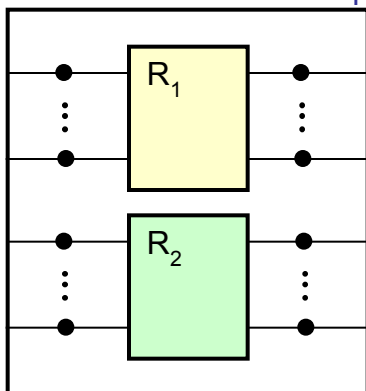
Последовательная: $R_1 \cdot R_2$



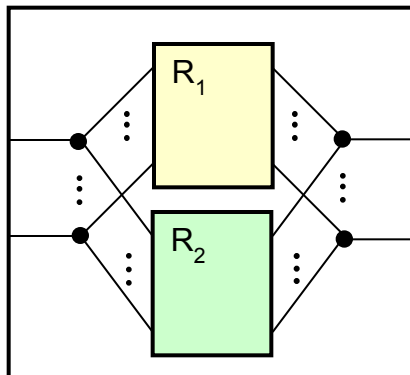
Объединение: $R_1 \cup R_2$



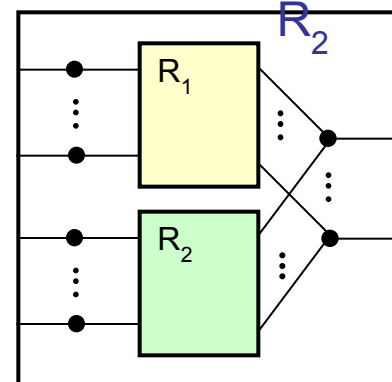
Параллельная: $R_1 \# R_2$



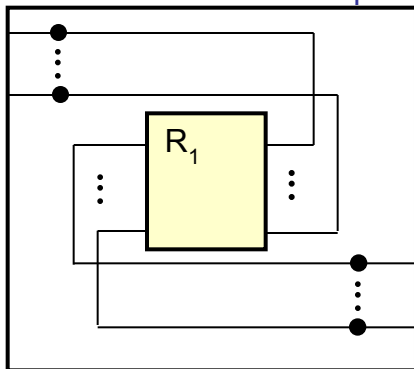
Пересечение: $R_1 \cap R_2$



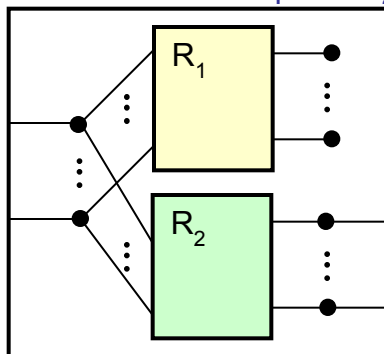
Унификация: $R_1 \nabla R_2$



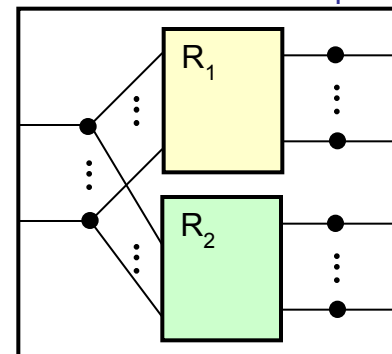
Инверсия: $\sim R_1$



Условная: $R_1 \rightarrow R_2$



Конкатенация: $R_1 * R_2$



Параметризованные описания НО

Список параметров:

- СпПар → Имя полное имя НО
- СпПар → @ собственные параметры
- СпПар → _ пропуск параметра
- СпПар → СпПар ; СпПар объединение в список

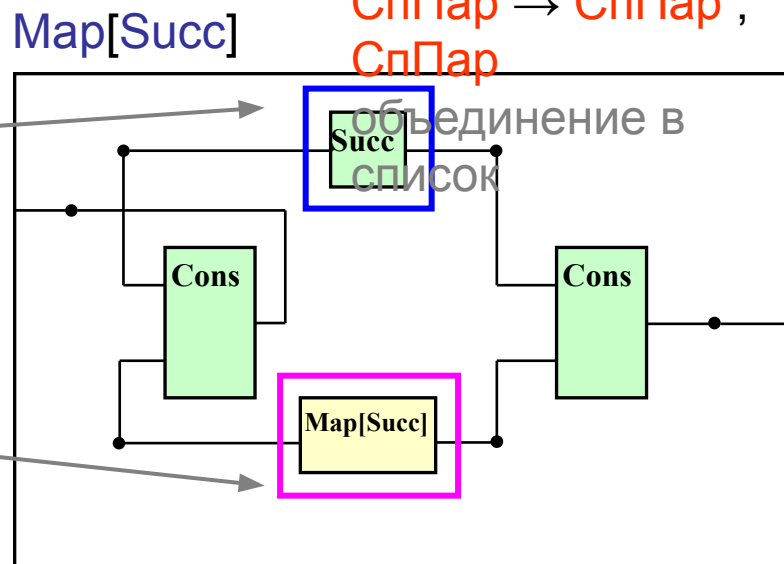
Вызов параметризованного НО: Имя[СпПар]

Вызов параметра в определении: <<Ар>>

*Значения по умолчанию: Имя[СпПар] = Рел

Пример параметризованного НО:

```
...  
--- = {x:x};  
Map[---] =  
    {Cons(x,y): Cons(<<0>>(x),@(xs))}  
Map = {Nil:Nil};  
QUERY = Map[Succ];
```



Ограничения:

1) Нет групп параметров

СпПар → Имя
 полное имя

2) Нет СпПар → График

СпПар → @
 собственные
 параметры

СпПар → _
 пропуск параметра

СпПар → СпПар ;
СпПар

Условная конструкция IF

Выр \rightarrow IF Лог THEN Выр ELSE Выр

Логическое выражение: Лог \rightarrow Ар АрСр Ар, где АрСр \in {<<0>>, <>, NOT, Лог};

Лог \rightarrow Ар ЛогСр Ар, где ЛогСр \in {AND, OR, XOR}.

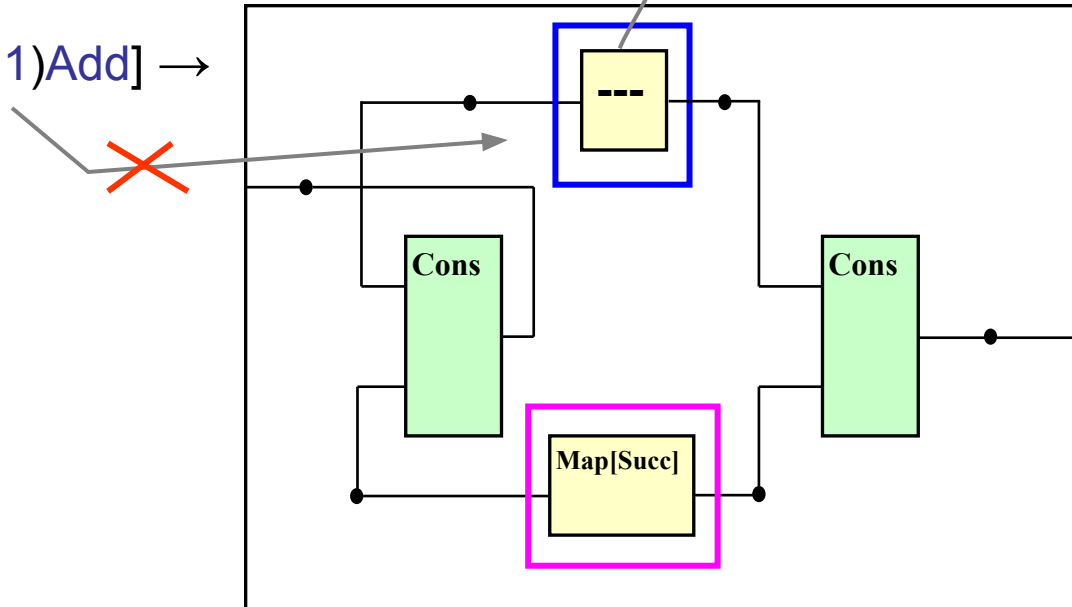
Пример (контроль арностей параметра):

...

--- = {x:x};

Map[---] = IF <<0>> <> 1 OR <<0>> <> 1 THEN ---
ELSE {Cons(x,y): Cons(<<0>>(x), @(xs))} U {Nil:Nil}

QUERY = Map[(2:1)Add]; Map[(2:1)Add] \rightarrow



Индексированные имена

Имя отношения:

Имя \rightarrow Спец [СпИнд] Ид СпПар

Имя переменной терма:

Переменная \rightarrow [СпИнд] Ид

Список индексов:

СпИнд \rightarrow Ар

СпИнд \rightarrow СпИнд, СпИнд

Натуральные числа:

[0]Nat= Null;

[1]Nat= Null·Succ;

[2]Nat= Null·Succ·Succ;

Add={Succ([1]x),[2]x:Succ(@([1]x,[2]x))};

QUERY=[1]Nat # [2]Nat·Succ

Массивы объектов:

...

[1,1]Cell= [1]Nat;

[1,2]Cell= [4]Nat;

[2,1]Cell= [2]Nat;

[2,2]Cell= [10]Nat;

QUERY= ([1,1]Cell#[1,2]Cell)·Add

Свертка

$\text{Выр} \rightarrow (\text{Инф ИдСв} = \text{СпЗнач}) \text{Выр}$

Ид – операторная переменная свертки.

Инф – инфиксная связка.

СпЗнач – список значений переменной.

Выр – выражение-операнд свертки.

Ограничение

Нет пользовательских связей для реляционного выражения.

$(\otimes_{J=1..10})\text{Выр} \Rightarrow \otimes([\frac{1}{J}]\text{Выр}, \otimes([\frac{2}{J}]\text{Выр}, \dots \otimes([\frac{9}{J}]\text{Выр}, [\frac{10}{J}]\text{Выр}) \dots))$

$[\frac{x}{J}]\text{Выр}$ – результат подстановки x вместо J в выражение **Выр**.

Примеры:

1) $[3]\text{Nat} = \text{Null} \cdot (\cdot_{J=1..3})\text{Succ} \Rightarrow [3]\text{Nat} = \text{Null} \cdot \text{Succ} \cdot \text{Succ} \cdot \text{Succ}$

2) $[0]\text{Nat} = \text{Null};$
 $(\cdot_{J=1..100})[J]\text{Nat} = \text{Null} \cdot (\cdot_{K=1..J})\text{Succ};$
 \Rightarrow
 $[0]\text{Nat} = \text{Null};$
 $[1]\text{Nat} = \text{Null} \cdot \text{Succ};$
 $[2]\text{Nat} = \text{Null} \cdot \text{Succ} \cdot \text{Succ};$
...
 $[100]\text{Nat} = \text{Null} \cdot \text{Succ} \cdot \dots \cdot \text{Succ};$

Свертка: моделирование связей

Операции над массивами объектов:

...

[1,1]Cell = [1]Nat;

[1,2]Cell = [4]Nat;

[2,1]Cell = [2]Nat;

[2,2]Cell = [10]Nat;

F = (<<0>># ---)·<<1>>;

SummCells = Null·(·I = 1..2)(·J = 1..2)F[[I,J]Cell;Add];

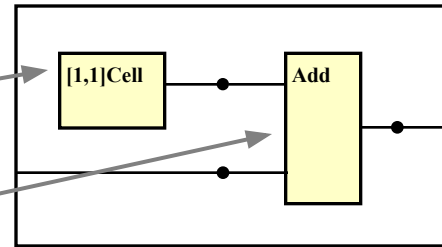
FLOGOL:

SummCells = (ADD I = 1..2)(ADD J = 1..2)[I,J]Cell

ПОЛЬЗОВАТЕЛЬСКАЯ СВЯЗКА

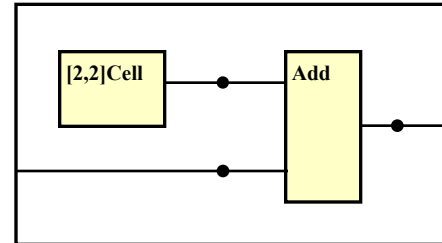
комбинирующая форма →

F[[1,1]Cell;Add]

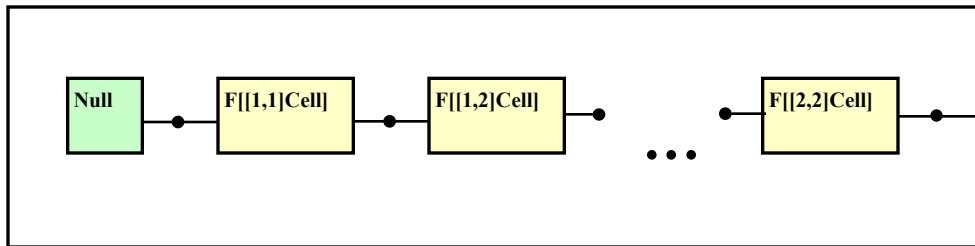


...

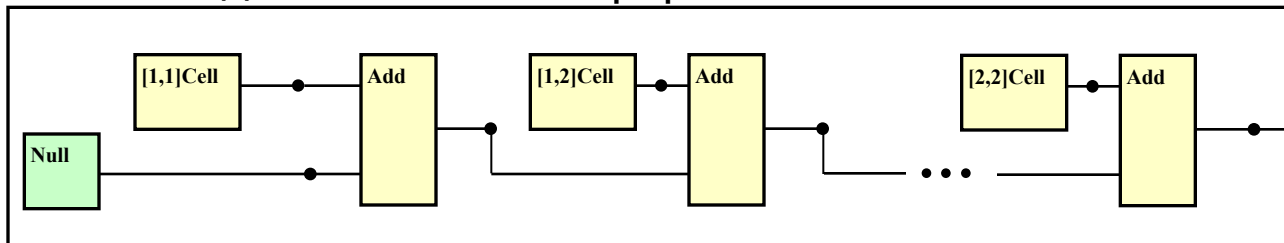
F[[2,2]Cell;Add]



SummCells



после подстановки комб. форм



Типизация

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 как:

$$\text{Type}T = \{G_{t'_1}, \boxtimes, G_{t'_{n'}} : G_{t''_1}, \boxtimes, G_{t''_{n''}}\} \cap \ll 0 \gg,$$

где G_t – НО арности $(0,1)$, генерирующее данные сорта t .

Пример (типизация НО сложения):

//Генератор натуральных чисел

`Nat = Null U @·Succ;`

//Отношение сложения

`Add = {Null, x:x};`

`Add = {Succ(x), y:Succ(@ (x,y))};`

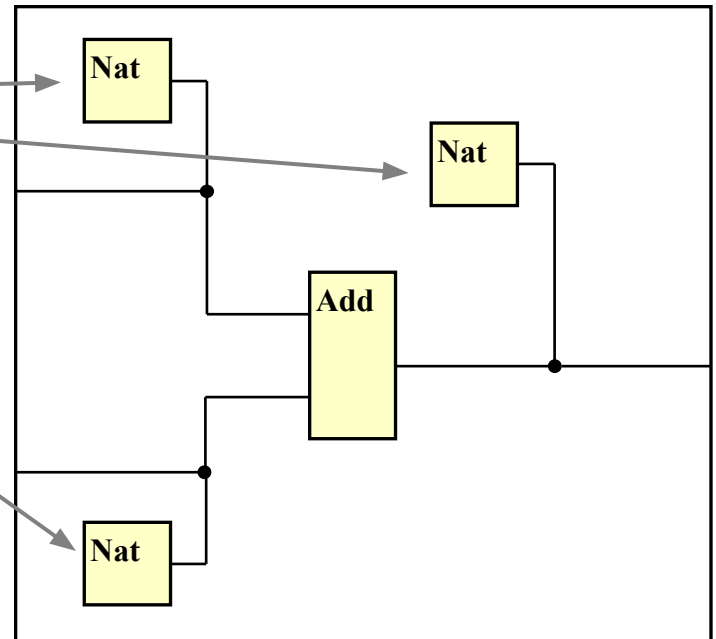
//Типовое отношение арности (2,1)

`(2:1)TypeNat = {Nat, Nat:Nat} \cap \ll 0 \gg;`

//Типизированное отношение сложения

`TypedAdd = TypeNat[Add];`

TypeNat[Add]



Описание семантики языка

Семантика языка описана с точностью до преобразования в КССГ, индуцированную выделенным в программе запросом к системе.

Принципы описания:

- Φ_φ – интерпретация синтаксической конструкции языка при интерпретации φ контекста ее переменных.
- $\varphi = \langle \xi, Name \rangle$ – контекст вызова НО, где
 - ξ – контекст сверки,
 - $Name$ – имя вызываемого НО.
- $\xi = v_1 v_2 \dots v_n, n \geq 0$ – значения операторных переменных свертки (v_i) в области действия которых находится описываемая конструкция.
- $Name = \langle Spec, IndList, Id, Cons, Prms \rangle$ – спецификатор, список индексов, идентификатор, признак НО – конструктора, список параметров вызова.
- Контекст и имя вызова записываются в виде списков.
 - $elm(Список, Номер)$ – выборка элемента списка с указанным номером,
 - $append(Список, Список)$ – конкатенация списков,
 - $[]$ – пустой список.

Семантика основных конструкций

Область интерпретации выражений:

Тип	Область интерпретации	Обозначение	По умолчанию
Дом	Множество правил	R	\emptyset
Рел	Множество сетей, Множество правил	S, R	\emptyset, \emptyset
СпПар	Список имен НО	$Prms$	$[\]$
Терм	Множество сетей, Множество правил	S, R	$\{S0^{(0,0)}\}, \emptyset$
Формула	Множество сетей, Множество правил	S, R	$\{S0^{(0,0)}\}, \emptyset$
Ар	Множество натуральных чисел	Nat	0
Лог	$\{True, False\}$	$Bool$	$False$
СпИнд	$[\]$	$IntList$	$[\]$
СпЗнач	$[\]$	$ValList$	$[\]$

Семантика определения НО в доменном выражении:

$\Phi_\varphi \sqsubseteq Cne\varphi e_1 \text{ Ид } e_2 = e_3 \sqsubseteq = R$, где $e_1 \in e(CnИнд)$, $e_2 \in e(CnПар)$, $e_3 \in e(Рел)$,

$Name_0 = elm(\varphi, 2)$,

имя вызываемого НО

$Name_{call} = [elm(Name_0, 1), elm(Name_0, 2), elm(Name_0, 3)]$,

имя вызываемого НО без параметров

$Name_{dom} = [\Phi_\varphi \sqsubseteq Cne\varphi \sqsubseteq , \Phi_\varphi \sqsubseteq e_1 \sqsubseteq , k(Ид)]$,

имя определяемого НО

$Name = unify(Name_{call}, Name_{dom})$,

унификация имен вызываемого и определяемого НО

$Prms = joinPrms(elm(\varphi, 5))$,

подстановка значений по умолчанию в параметры вызова

$[S, R_1] = \Phi_{\varphi'} \sqsubseteq e_3 \sqsubseteq \varphi' = [elm(\varphi, 1), append(Name, Prms)]$,

формирование сети вызываемого НО и индуцированных сетевых определений

$R_2 = makeR(Name, S)$,

формирование сетевого определения

$R = R_1 \cup R_2$.

формирование результата

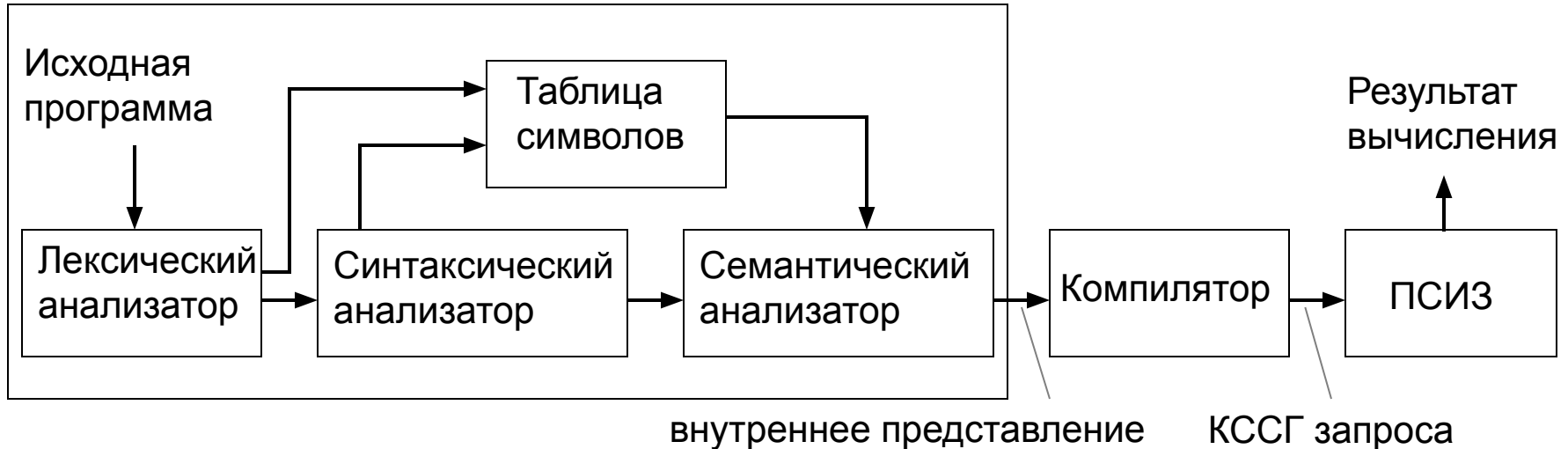
Реализация языка S-FLOGOL

Компилятор запросов

Реализация языка S-FLOGOL

Смешанная форма = компиляция + интерпретация

Структурно-ориентированный редактор



Компиляция – генерация контекстно-свободной сетевой грамматики (КССГ), эквивалентной сетевой интерпретации S-FLOGOL-программы с выделенным запросом к системе.

Интерпретация – вычисление КССГ подсистемой исполнения запросов (ПСИЗ).

Оригинальная технология ввода позволяет:

- исключить лексический и синтаксический анализ,
- обеспечить синтаксическую корректность и однозначность разбора,
- выполнить первичный семантический анализ.

Стадии компиляции

- **Предварительная стадия**

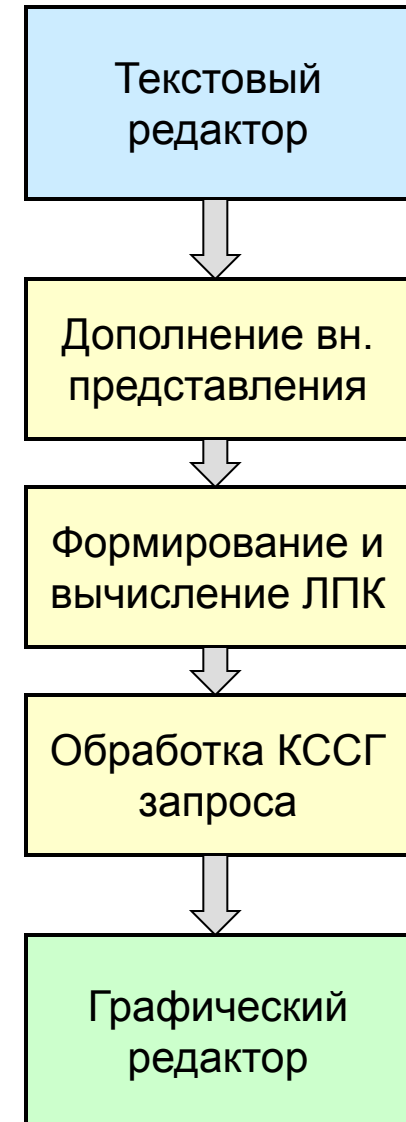
Завершение формирования и дополнительная обработка внутреннего представления программы

- **Основная стадия**

Формирование и вычисление логической программы-компилятора (ЛПК), результатом которого является КССГ запроса

- **Заключительная стадия**

Обработка полученной КССГ запроса, запись КССГ в сетевой модуль СФЛП и передача в графический редактор



Предварительная стадия компиляции

- Дополнение не полностью введенных элементов программы соответствующими значениями по умолчанию:

CONSTRUCTOR(Ap) Succ

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

Add = ИмяОтн

CONSTRUCTOR(0) Succ

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

Add = EmptyRel

- Первичный семантический анализ:

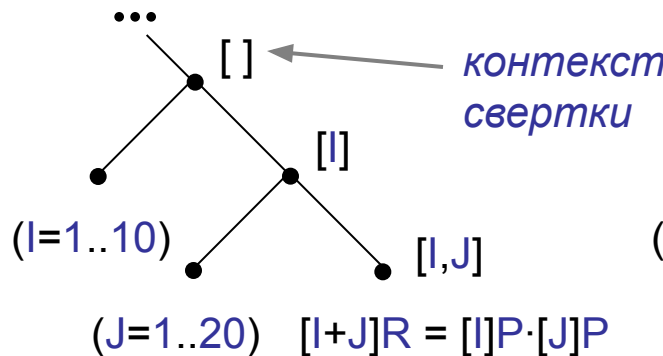
- контроль вызова не объявленных НО,
- чистка грамматики (исключение не используемых НО).

- Индексирование вхождений операторных переменных сверток:

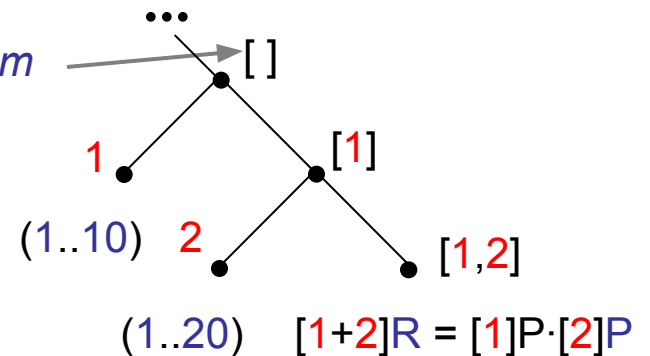
Программа

...
 (I=1..10)
 (J=1..20)
 [I+J]R = [I]P·[J]P
 ...

Дерево

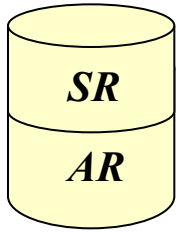


Индексированное дерево



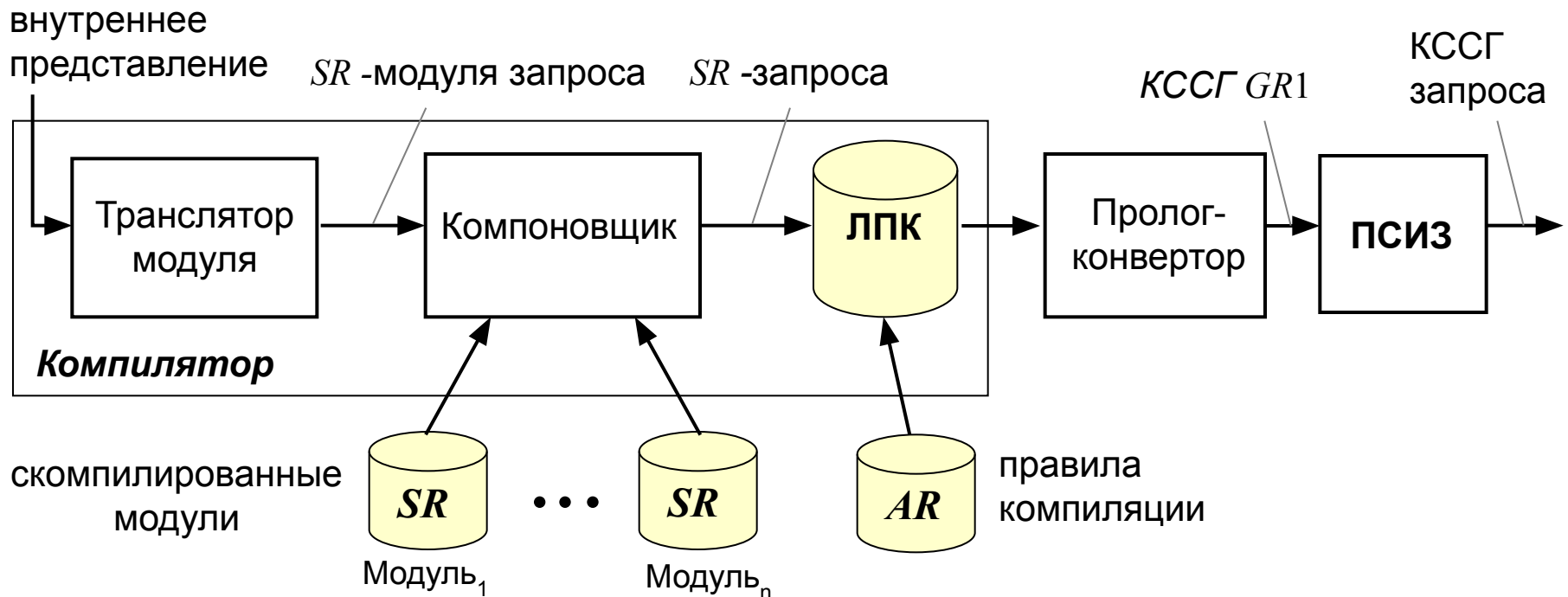
Основная стадия компиляции

Формирование КССГ выполняется логической программой-компилятором (ЛПК), база данных которой содержит:



- *SR* -правила (*SpecialRules*) - база описаний НО.
- *AR* -правила (*AdditionalRules*) - управляющие и вспомогательные правила компиляции.

Схема выполнения основной стадии:



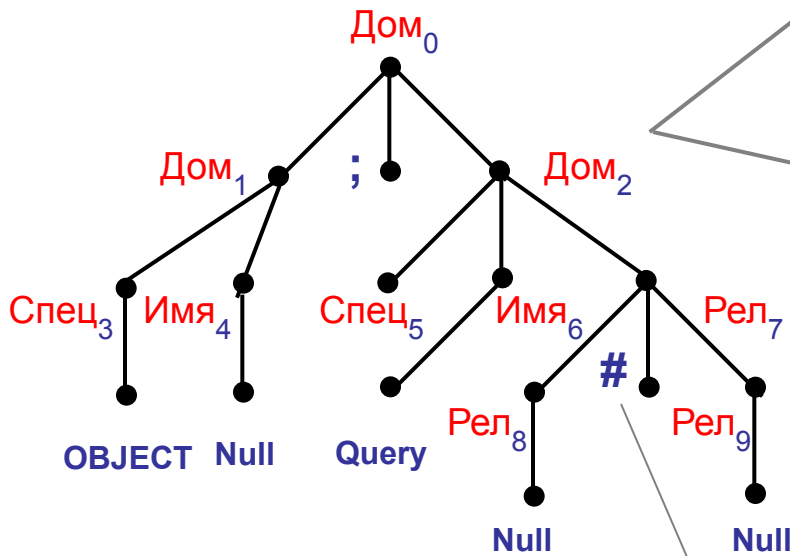
База описаний НО (SR-правила)

Программа:

OBJECT Null;

QUERY = Null#Null

Внутреннее представление:



SR-БД:

дом_0(Контекст, Правило):-
дом_1(Контекст, Правило).

дом_0(Контекст, Правило):-
дом_2(Контекст, Правило).

дом_1(Контекст, Правило):-
спец_3(Контекст, Спец),
одноэлСеть(Спец, 'Null', Сеть),
Правило = ['Конс', 'Query', Сеть].

спец_3(Контекст, сСпец(+,0,+,+,1,-)).

дом_2(Контекст, Правило):-
спец_5(Контекст,Спец),
рел_7(Контекст, Сеть)
Правило = ['Прав', 'Query', Сеть].

рел_7(Контекст, Сеть):-
рел_8(Контекст, Сеть1),
рел_9(Контекст, Сеть2),
послКомп(Сеть1,Сеть2,Сеть).

рел_8(Контекст, Сеть):-
одноэлСеть(Спец, 'Null', Сеть).

рел_9(Контекст, Сеть):-
одноэлСеть(Спец, 'Null', Сеть).

Контекст вызова:

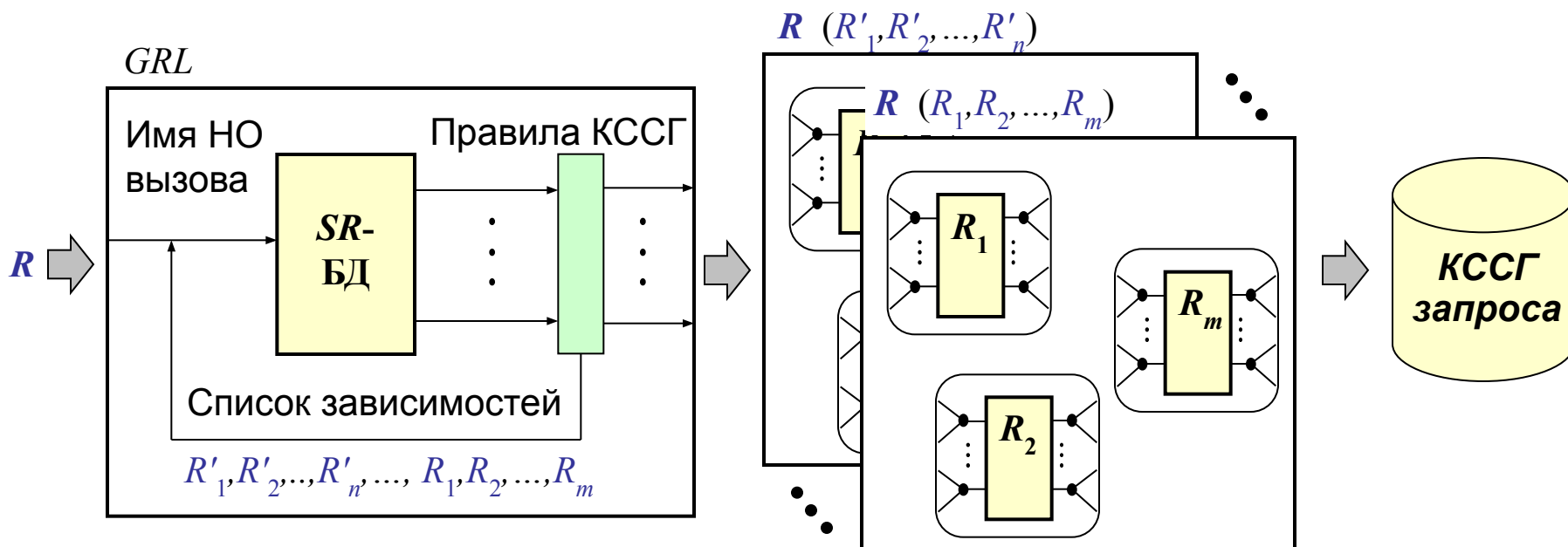
- список значений переменных свертки;
- имя вызываемого отношения.

Управление компиляцией (AR-правила)

Конструктивное построение КССГ:

- 1) получить сетевые определения НО с указанным именем,
- 2) выполнить шаг 1) для всех имен НО, индуцированных этим определением.

Схема формирования КССГ (AR-правило *GRL*):



Формирование КССГ:

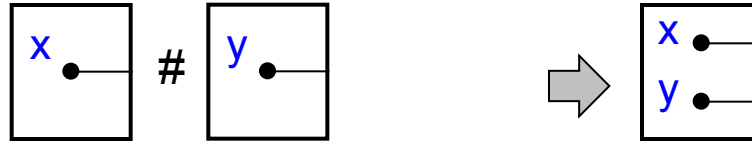
- начинается с шага 1) для НО запроса,
- завершается, когда список зависимостей пуст.

Формирование сети

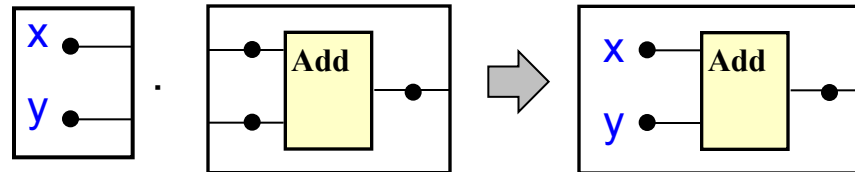
· Формирование графика

Пример: $\{x, y: \text{Add}(x,y)?x \langle \rangle y\}$

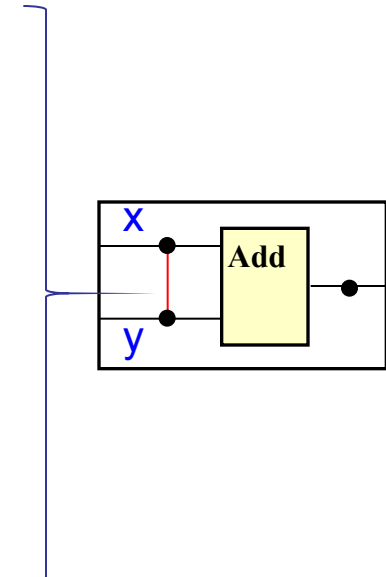
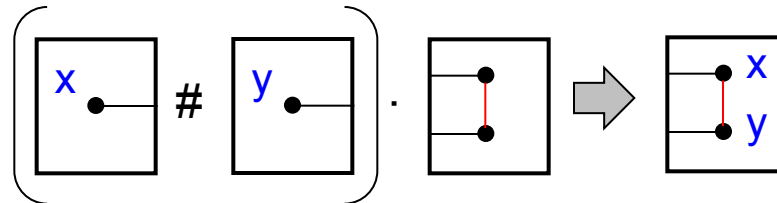
Входной терм = x, y



Выходной терм = $\text{Add}(x,y)$



Формула = $x \langle \rangle y$



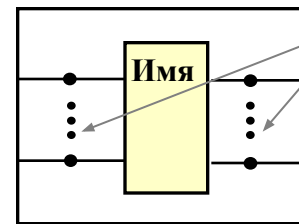
· Формирование вызова НО

Проблема:

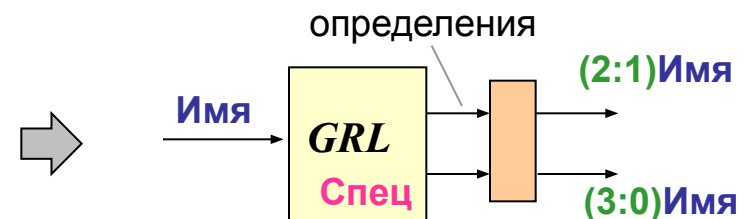
имя вызываемого и определяемого НО, как правило, не специфицировано.

Решение:

получить определение вызываемого НО, извлечь информацию о его спецификации.



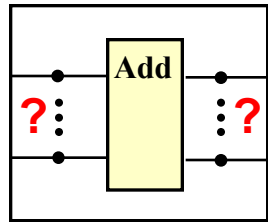
? необходимо знать арность



Оптимизация получения спецификаторов

- Исключение дублирующих спецификаторов.

Вызов НО Add:

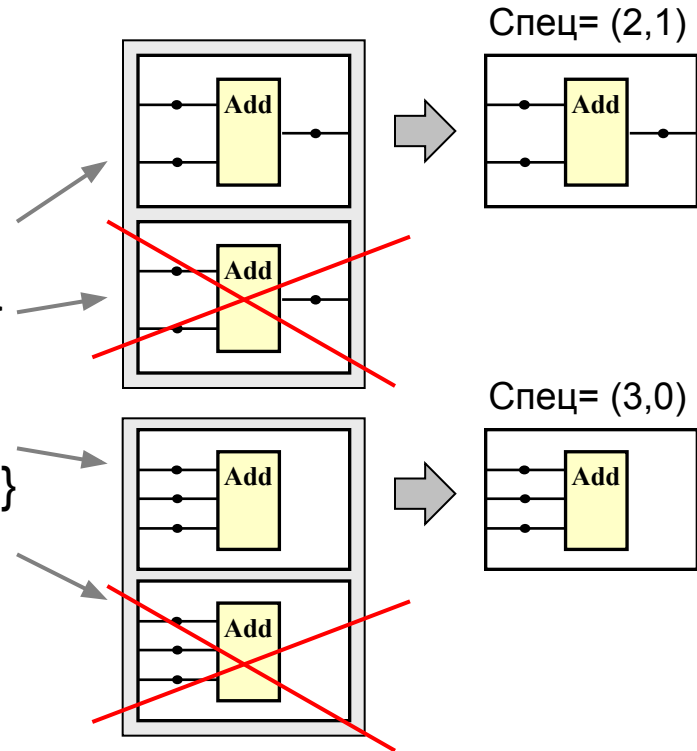


Add={Null,x:x}

Add={Succ(x),y:Succ(Add(x,y))}

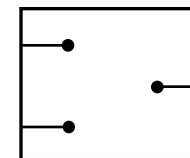
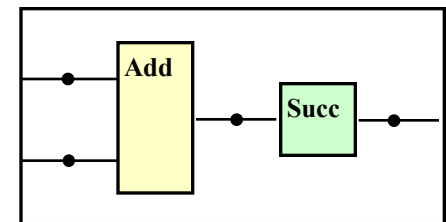
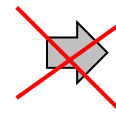
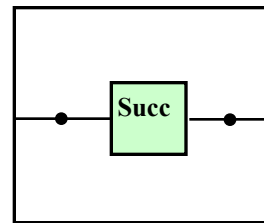
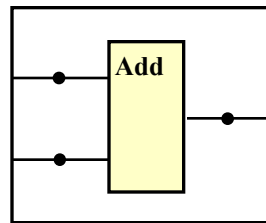
Add={Null,x,x:}

Add={Succ(x),y,Succ(Add(x,y)):}



- Частичное выполнение композиций НО.

Спец? (Add·Succ):



Спец= (2,1)

Заключительная стадия компиляции

- Преобразование КССГ из *SN*-формата в формат КССГ (выполняется в ПСИЗ).
- Редукция имен – исключение не значимых для КССГ элементов имени.

имя НО после компиляции:

(-,0,-,-,1,-),Nil,Null,Nil,Конс

(-,2,-,-,1,-),Nil,Add,Nil,Прав

(-,3,-,-,0,-),Nil,Add,Nil,Прав

(-,1,-,-,1,-),Nil,Map,Succ,Прав

· Редукция имен –
исключение не
значимых для КССГ
элементов имени.

(-,3,-,-,0,-)Add

Map[Succ]

- Установка интерпретации системных типов данных и отношений.

(-,3,-,-,0,-),Nil,124,Nil,Конс → системный тип – натуральное число

(-,3,-,-,0,-),Nil,\$\$Add,Nil,Конс → системное НО – сложение нат. чисел

- Сохранение полученной КССГ и ее отображение пользователю.

КССГ сохраняется в сетевой модуль текущего проекта и открывается графическим редактором для:

- просмотра и возможной корректировки результата компиляции,
- выполнения запроса.

Интерфейсные средства

**Структурно-ориентированный
текстовый редактор**

Технология дедуктивного ввода программ

Технология позволяет:

- Минимизировать ручной ввод (идентификаторы, константы).
- Исключить лексический и синтаксический анализ программы.
- Снизить число семантических ошибок ввода.
- Повысить читаемость текста программы.
- Использовать математические и др. специальные символы.
- Обеспечить переносимость на различные естественные языки.

Недостатки:

- Необходимость введения в грамматику дополнительных атрибутов.
- Несколько сниженная скорость ввода программ.

Возможные специальные области применения:

- Учебные приложения и скриптовые языки.
- Блокнотные КПК, не обладающие полноразмерной клавиатурой.
- Приложения для людей с ограниченными возможностями.

Основа технологии

Программа – выводимая из грамматики цепочка символов.

Шаг ввода – применение правила к нетерминальному символу цепочки.

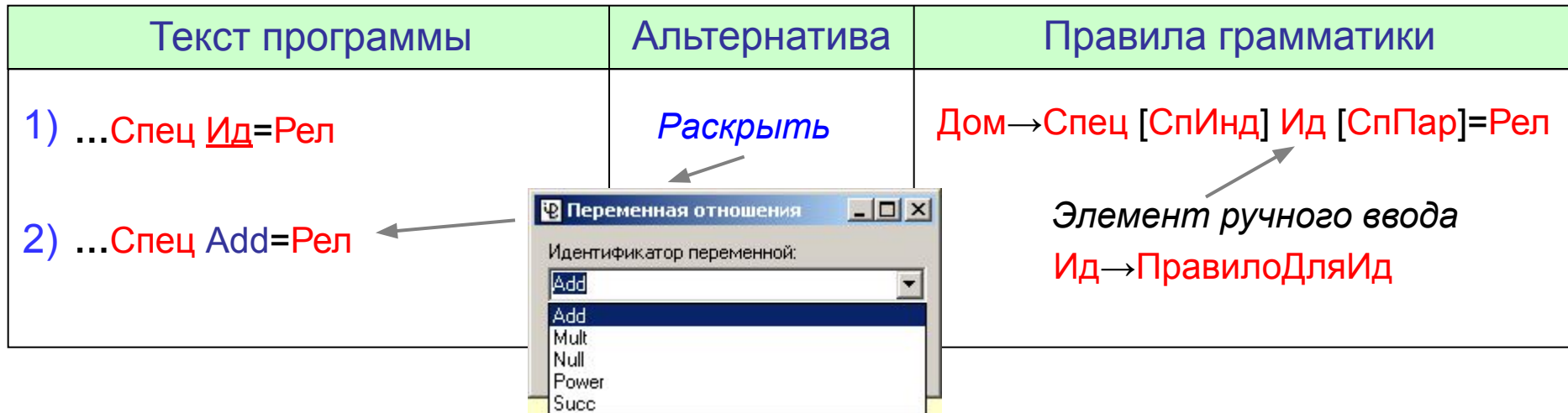
Исходное состояние – начальный нетерминальный символ грамматики.

Текст программы	Альтернатива	Правила грамматики
1) MODULE Common= <u>Дом</u> END	<i>Определение</i> Конструктор Объединение	$\text{Дом} \rightarrow \text{Спец} [\text{СпИнд}] \text{Ид} [\text{СпПар}] = \text{Рел}$ $\text{Дом} \rightarrow \text{Спец} [\text{СпИнд}] \text{Ид}$ $\text{Дом} \rightarrow \text{Дом} ; \text{Дом}$
2) MODULE Common= Спец [СпИнд]Ид[СпПар]= <u>Рел</u> END	Вызов <i>График</i> Композиция	$\text{Рел} \rightarrow \text{ИмяОтн}$ $\text{Рел} \rightarrow \{ \text{Терм} : \text{Терм} ? \text{Формула} \}$ $\text{Рел} \rightarrow \text{Рел} \text{ РелИнф} \text{ Рел}$
3) MODULE Common= Спец [СпИнд]Ид[СпПар]= { <u>Терм</u> :Терм?Формула} END	Переменная Вызов Соединение	$\text{Терм} \rightarrow [\text{СпИнд}] \text{ИдТ}$ $\text{Терм} \rightarrow \text{ИмяОтн}(\text{Терм})$ $\text{Терм} \rightarrow \text{Терм} , \text{Терм}$

Специализированный ввод отдельных конструкций

Фрагмент текста программы:	Операция:	Правила грамматики:
Выражения		
1) ... [1]Nat= <u>Null</u> ← 2) ... [1]Nat=Null· <u>Рел</u>	·, *, ∇, →, #, U, П, ~	$\text{Ид} \rightarrow \text{Рел} \text{ РелИнф} \text{ Рел}$ $\text{РелИнф} \rightarrow \{ \cdot, *, \nabla, \rightarrow, \#, U, \text{П}, \sim \}$
Опциональные элементы		
1) ... (2:1) [<u>СпИнд</u>]Add[СпПар]=Рел 2) ... (2:1) <u>Add</u> [СпПар]=Рел 3) ... (2:1) [<u>СпИнд</u>]Add[СпПар]=Рел	скрыть отобразить опции восстановить	$\text{Дом} \rightarrow \text{Спец} \text{ [СпИнд]Ид[СпПар]=Рел}$  <p style="text-align: center;">опциональные элементы</p>
Обработка списков		
1) MODULE Cmn(<u>System</u>)= ... 2) MODULE Cmn(System, <u>ИмяМод</u>)= ...	добавить удалить в начало в конец	$\text{Модуль} \rightarrow \text{ИмяМод}(\text{ИмяМод}) = \dots$  <p style="text-align: center;">список с разделителем ‘;’</p>

Ввод идентификаторов и числовых констант



Также разработаны и реализованы:

- Алгоритм автоматического структурирования текста программы.
- Схема стилистического оформления текста.
- Управление детализацией отображения текста.
- Интерфейсные средства поддержки ТДВП.
- Выполнение операций по горячим клавишам.
- Типизированные буферы обмена.
- Многоуровневый откат.

```

MODULE RelComp=
(0:1)Null;
(1:1)Succ;
(2:1)F;
SecComp=
{~<<0>>(( ,I=1..<<0>>) [I] x) :
<<1>>(( ,I=1..<<0>>) [I] x)
};
ParComp=
{( ,I=1..><<0>>) [I] x,( ,J=1..><<1>>) [J] y:
<<0>>(( ,I=1..<<0>>) [I] x) ,
<<1>>(( ,J=1..<<1>>) [J] y)
};
ConcComp=
{( ,I=1..><<0>>) [I] x:
<<0>>(( ,I=1..><<0>>) [I] x) ,
<<1>>(( ,I=1..><<1>>) [I] x)
};
    
```

Общий вид окна редактора

FLIDE - [Demo.FLP] - [FLIDE - [*Arithm.FLT]]

Файл Правка Вид Действия Справка

Конструкторы
Null
Succ
Правила
Add.1
Add.2
Mult.1
Mult.2
Nat0
Nat2
Nat3
Power.1
Power.2
Системные
Подключенные
Запросы
Nat10
QAdd
QMult
QPower
Результат

```
MODULE Arithm=  
  //Конструкторы натуральных чисел  
  (0:1)Null;  
  (1:1)Succ;  
  //Натуральные числа  
  Nat3={:Succ(Succ(Succ(Null)))};  
  Nat2=  
  Nat0=  
  //Отношения сложения  
  Add={Null,x:x};  
  Add={Succ(x),y:Succ(@(x,y))};  
  //Отношение умножения  
  Mult={Null,x:Null};  
  Power={x,Null:Succ(Null)};  
  Power={x,Succ(y):Mult(x,@(x,y))};  
  //Запросы  
  QAdd={:Add(Nat2,Nat3)};  
  QMult={:Mult(Nat2,Nat3)};  
  QPower={:Power(Nat2,Nat3)};  
  Nat10=Null*(•I=1..10)Succ  
END
```

Управление списками

Управление опциями

IF, CASE, Свертка

Управление компиляцией

Область редактирования

Список операций

Дерево объектов

Текущий элемент: Null Тип выражения: Реляционное Пробел: Изменить

Результаты работы

1. Предложена система ограничений языка FLOGOL, позволяющая выполнять компиляцию в конечную КССГ и значительно упрощающая реализацию.
2. На основе системы ограничений разработан язык S-FLOGOL, формально описаны его синтаксис и семантика.
3. Разработан метод компиляции запросов S-FLOGOL-программ. На основании формального описания семантики определены функции трансляции для всех синтаксических конструкций языка.
4. Предложены оптимизационные модификации, позволившие существенно сократить время компиляции запросов.
5. Предложена оригинальная технология дедуктивного ввода программ (ТДВП), позволяющая полностью исключить синтаксические ошибки и значительно сократить количество семантических ошибок при вводе программ.
6. Разработаны архитектура и интерфейсные средства структурно-ориентированного редактора, реализующего ТДВП.
7. Выполнена программная реализация структурно-ориентированного редактора и компилятора запросов и их интеграция в СФЛП.

По теме диссертации опубликовано 9 печатных работ.

Спасибо за внимание!