FLIDE

FLOGOL Integrated Development Environment

Система функционально-логического программирования на языке S-FLOGOL

Московский институт радиотехники, электроники и автоматики (МИРЭА ТУ)

Бебчик Антон Михайлович

Разработка и исследование подсистемы исполнения запросов и графического редактора 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), основанной на формализме направленных отношений (НО) и обладающей развитыми интерфейсными средствами построения и отладки программ.

Основные задачи:

- разработка и исследование моделей вычислений НО,
- разработка технологии графического построения программ,
- программная реализация подсистемы исполнения запросов,
- программная реализация графического редактора СФЛП.

Направленные отношения.

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:

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

x'	x''	y
0	0	0
0	1	1
1	1	2
1	2	3
...	...	...

Пример графика НО факториала:

$$Fact^{(1,1)} = \{(x, y) \mid y = x!\}$$

x	y
0	1
1	1
2	2
3	6
...	...

Свойства НО и представимые семантические объекты.

1. НО R называется *функциональным* ($\mathbf{F}(R)$), если

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

2. НО R называется *тотальным* ($\mathbf{T}(R)$), если

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

3. НО R^{-1} называется *обратным* для R, если

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

Пример представления некоторых семантических объектов логического и функционального программирования:

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

Сетевое представление НО

Базисом сети B называется тройка (X, μ', μ''), где X - множество сортов элементов, а $\mu', \mu'' : X \to 0..$ задают арность элементов каждого сорта.

Размеченной сетью S арности (n', n''), $n', n'' \geq 0$, в базисе элементов B называется шестерка $< P, I, O, E, U, \sigma_0 >$, где

P – множество точек сети;

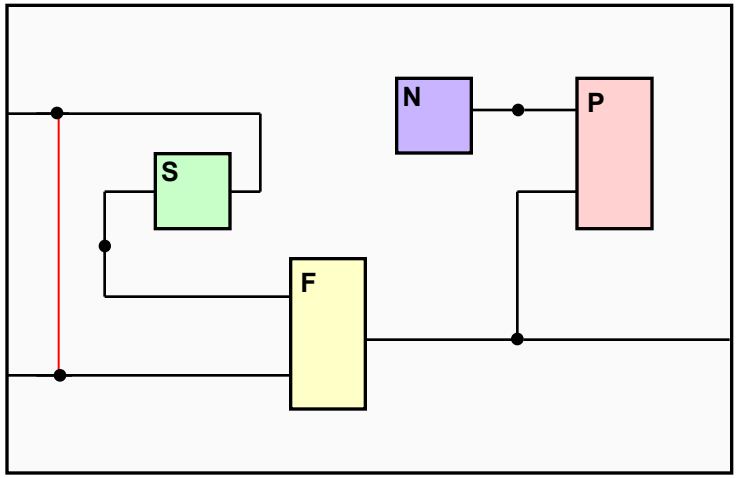
I, O – входной и выходной кортежи, $|I| = n'$, $|O| = n''$;

E – множество элементов сети;

U – граф семантического различия;

σ_0 – начальная разметка сети.

Графическая нотация СПНО:



Интерпретация Ψ сети S есть

$$\Psi_{\varphi}\,\square S\,\square \equiv \{(\alpha', \alpha'') \mid (\exists \sigma : P \to D)((\forall p \in dom(\sigma_0))(\sigma(p) = \sigma_0(p)) \,\&\, \alpha' = \sigma(I) \,\&\, \alpha'' = \sigma(O) \,\&$$

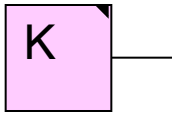
$$(\forall \{p', p''\} \in U)(\sigma(p') \neq \sigma(p'')) \,\&\, (\forall e \equiv < x, I_e, O_e > \in E)(< \sigma(I_e), \sigma(O_e) > \in \varphi\,\square x\,\square)\},$$

где $\varphi\,\square x\,\square$ - НО арности $\mu'(x), \mu''(x)$ на носителе D.

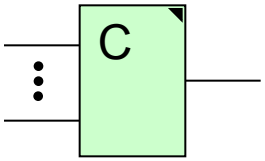
Сетевое представление семантических объектов

Семантические примитивы

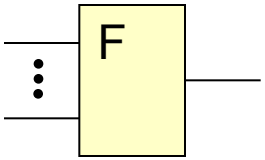
· Константа



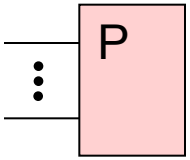
· Конструктор



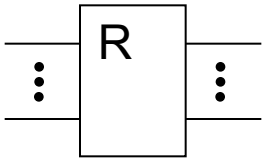
· Функция



· Предикат

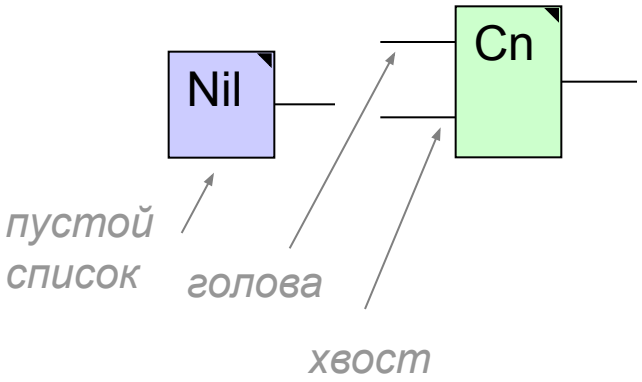


· НО общего типа

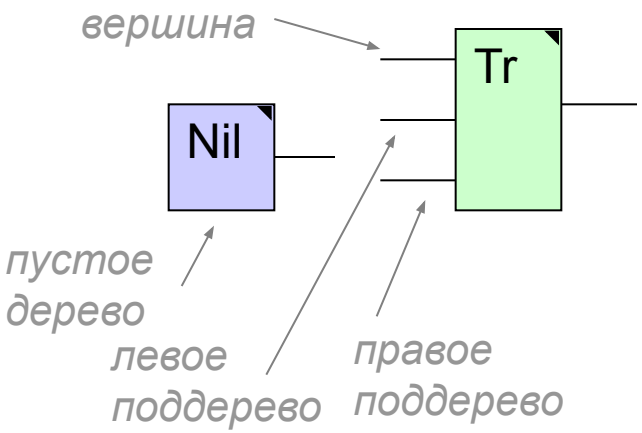


Составные объекты

· Список



· Дерево



Программа

Программа задается контекстно-свободной сетевой грамматикой (КССГ), определяемой как четверка $< B_m, B_н, \alpha, R >$, где

B_m – терминальный базис;

$B_н$ – нетерминальный базис $(B_m \cap B_н = \varnothing)$;

$\alpha \in B_н$ – аксиома;

R – множество правил вида $b \to S$, где $b \in B_н$, S – сеть в базисе $(B_m \cup B_н)$.

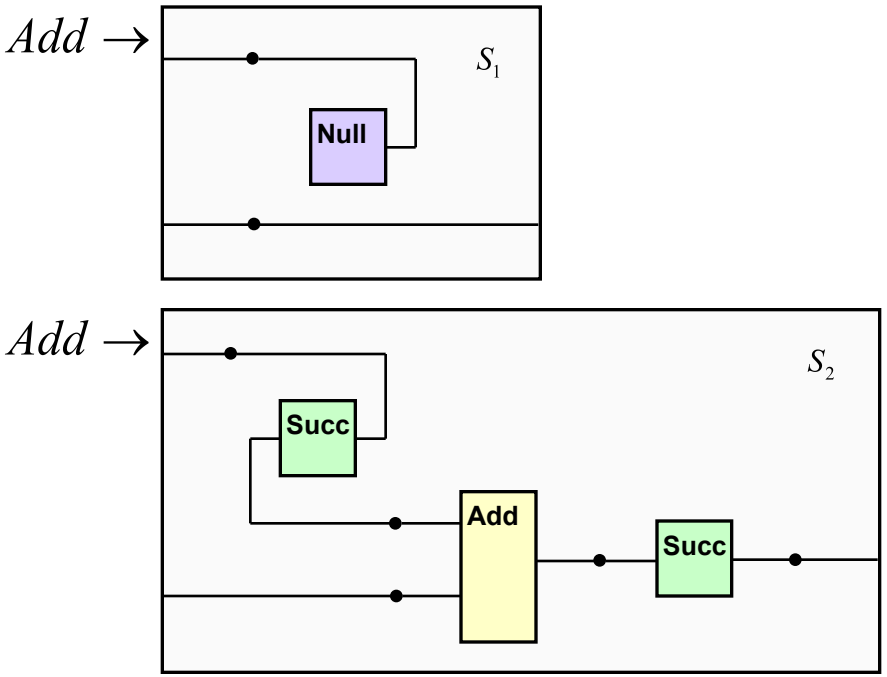
Пример: НО сложения Add.

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

$B_н = \{Add^{(2,1)}\},$

$\alpha = Add,$

$R = \{Add \to S_1, Add \to S_2\}.$



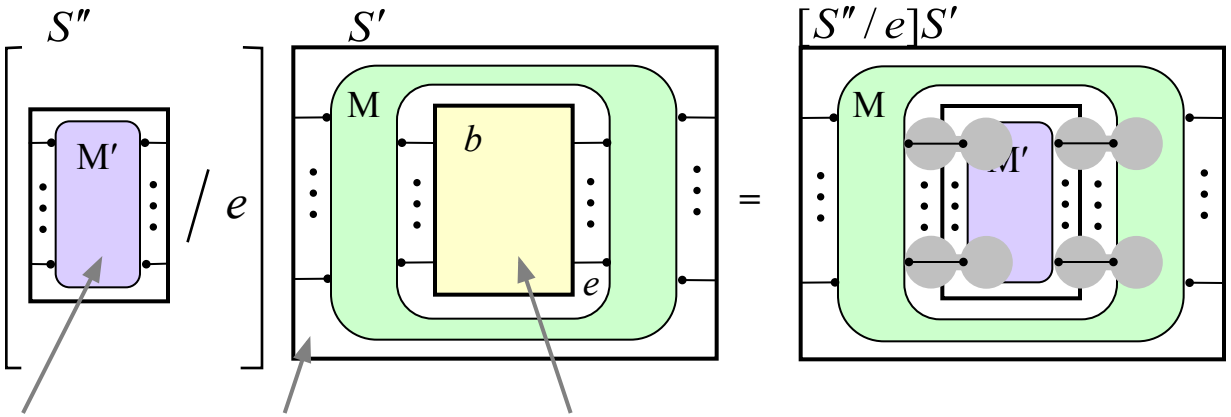
Вычисление

1. Вычисление программы производится путем порождения сетевого языка по заданной КССГ.

2. Сетевой язык определяется как $L_G \cong \{S \mid a \xrightarrow{R} S \,\&\, (\forall (x, I_e, O_e) \in E(x \in B_н))\}$ где $\alpha \xrightarrow{R} S$ обозначает выводимость сети S из аксиомы α:

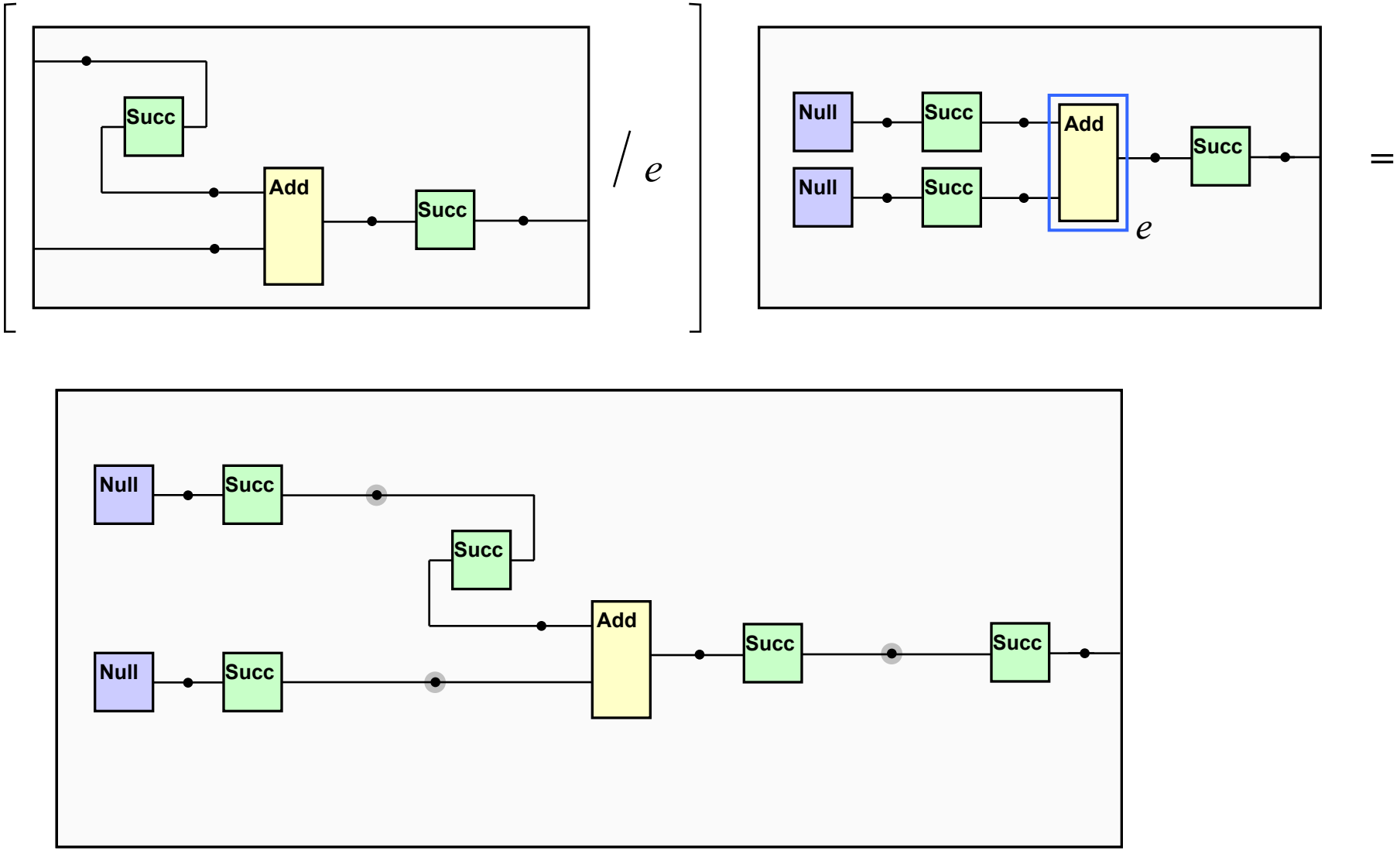
$$\alpha \xrightarrow{R} S \cong (\alpha \to S) \in R \vee \exists S_1 (\alpha \xrightarrow{R} S_1 \,\&\, (\exists r \in R)(S_1 \xrightarrow{r} S))$$

3. Шаг вывода (применение правила КССГ к сети) производится путем выполнения подстановки сети S'' правила $b \to S''$ вместо элемента e сорта b сети S' (обозначается $[S''/b]S'$).



4. Интерпретация программы $\Psi_\varphi \square \sum \square = \boxtimes_{S \in \Sigma} \Psi_\varphi \square S \square$

Подстановка

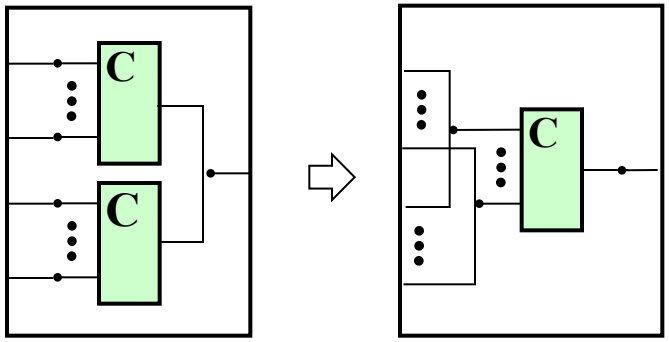


Редукция сетей

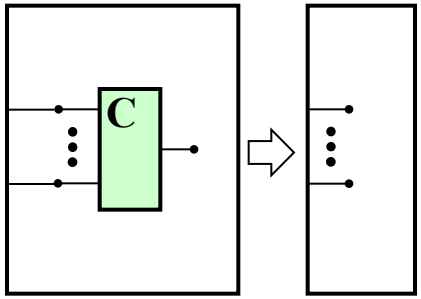
Редукция предназначена для трансформации сетей на основе знаний о свойствах интерпретации их элементов.

Редукция по структуре

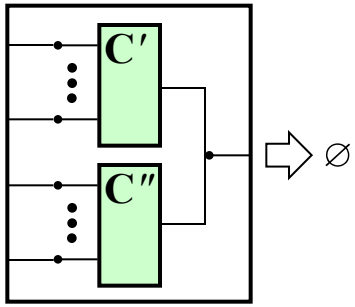
функциональность деструкторов



тотальность конструкторов

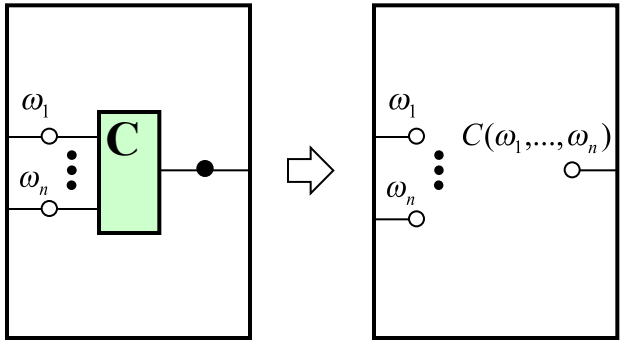


ортогональность конструкторов

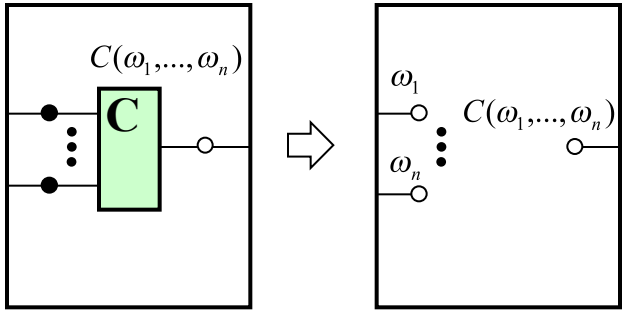


Редукция по разметке

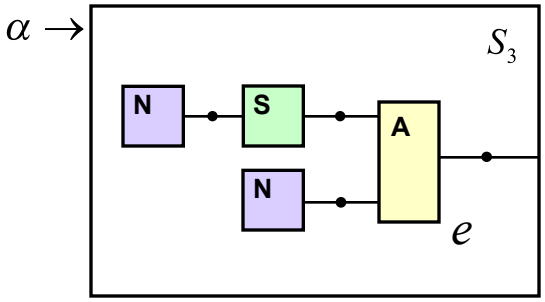
прямое распространение



обратное распространение



Вычисление в базисе конструкторов



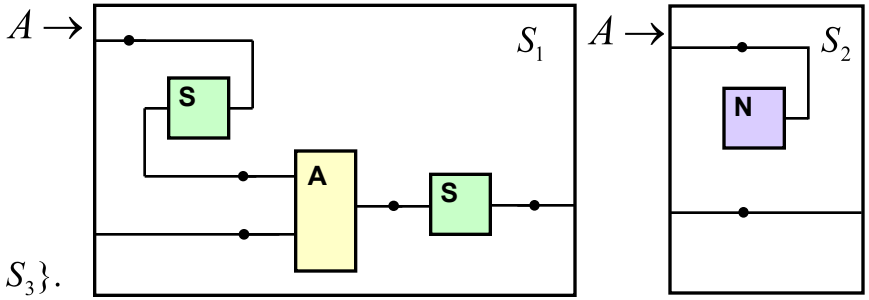
$G =< B_m, B_н, \alpha, R >$, где

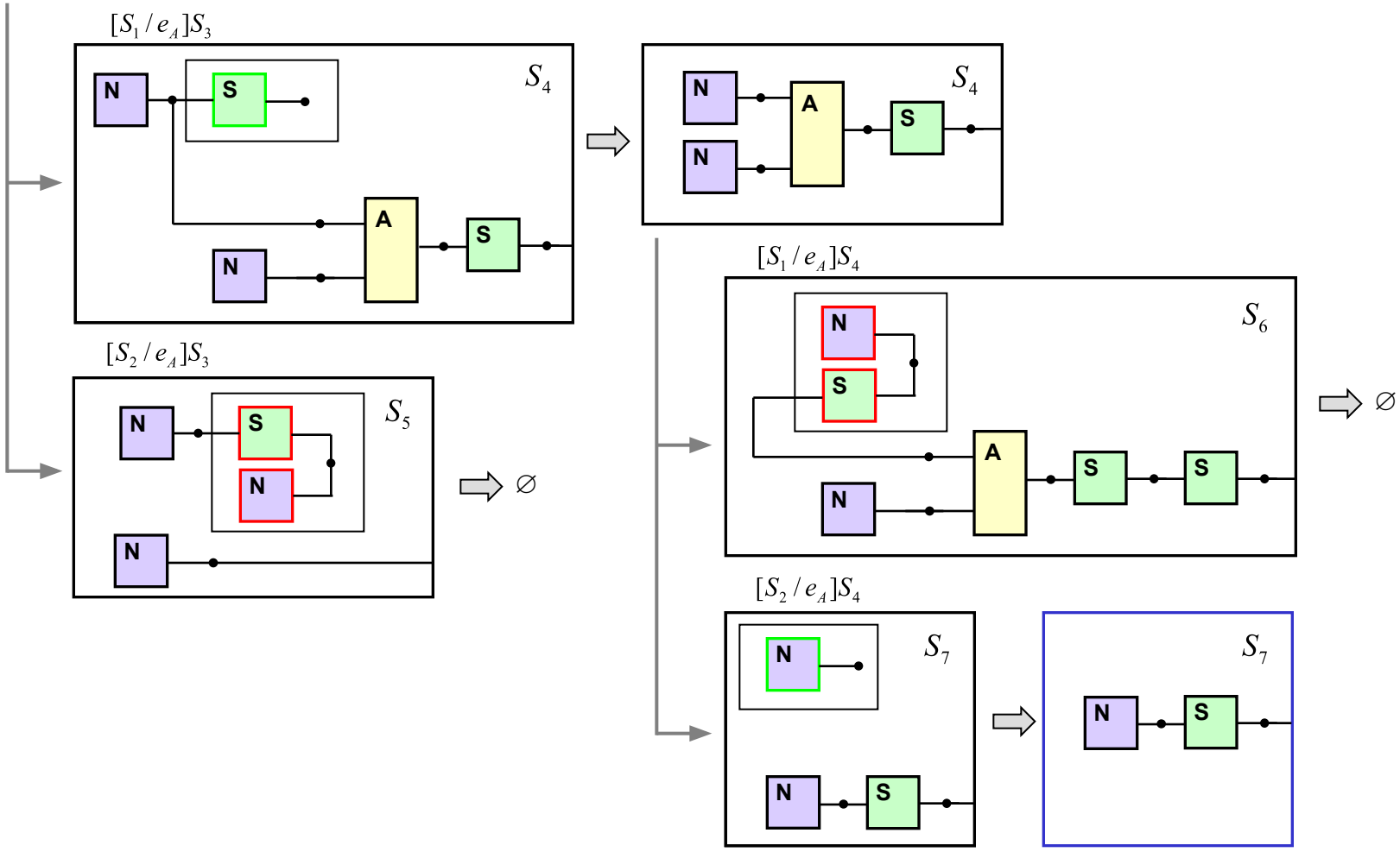
$B_m = \{N^{(0,1)}, S^{(1,1)}\},$

$B_н = \{A^{(2,1)}, Q\},$

$\alpha = Q,$

$R = \{A \to S_1, A \to S_2, Q \to S_3\}.$





Вычисление с разметкой

Введенное понятие размеченной сети и предложенные правила редукции по разметке положены в основу режима вычисления с разметкой.

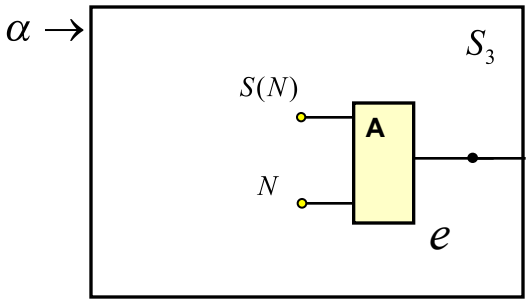
Использование разметки позволяет:

- упростить структуру сети,
- снизить требования к объему доступной оперативной памяти,
- упростить взаимодействие с внешними процедурами вычисления,
- повысить скорость вычисления за счет простой обработки разметки,
- использовать системные типы данных и системные НО,
- реализовать Data - Flow подход,
- повысить читаемость результатов вычисления.

Недостатки использования разметки:

- отсутствие поддержки неполных структур данных,
- необходимость использования смешанной редукции.

Вычисление с разметкой



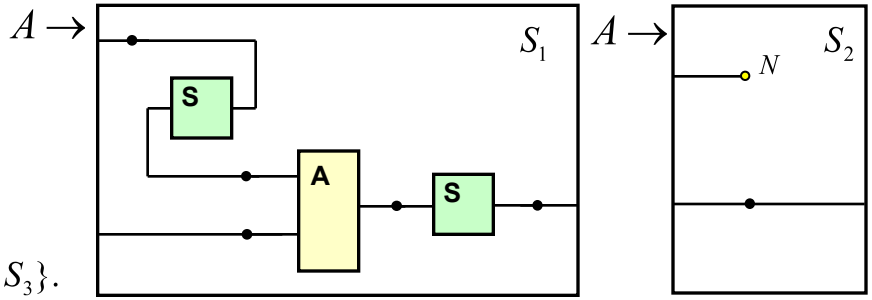
$G =< B_m, B_н, \alpha, R >$, где

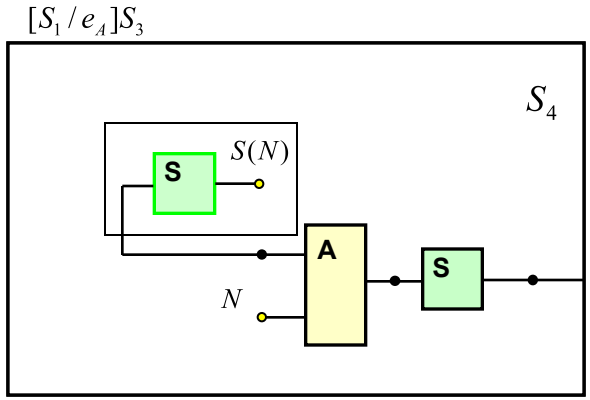
$B_m = \{N^{(0,1)}, S^{(1,1)}\},$

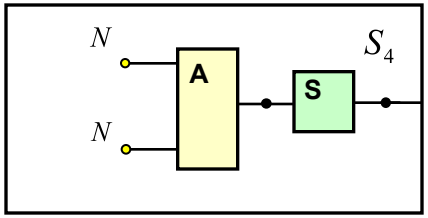
$B_н = \{A^{(2,1)}, Q\},$

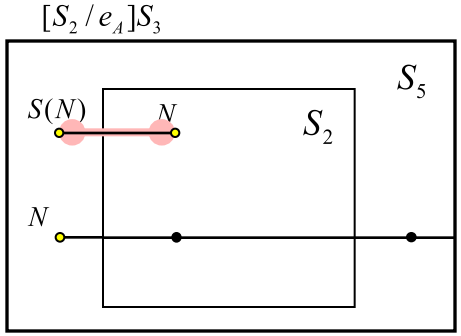
$\alpha = Q,$

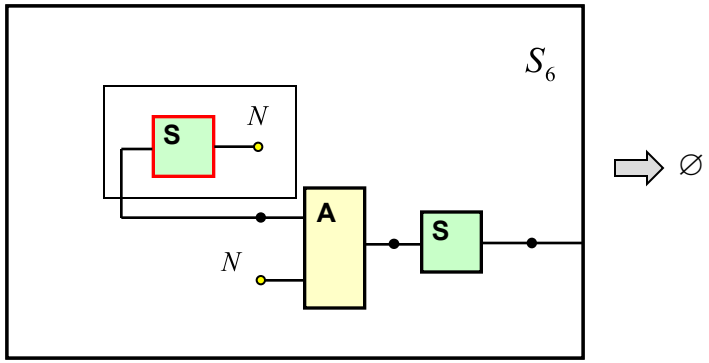
$R = \{A \to S_1, A \to S_2, Q \to S_3\}.$

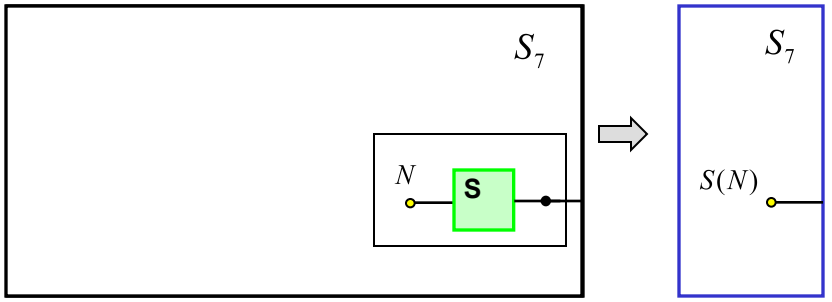












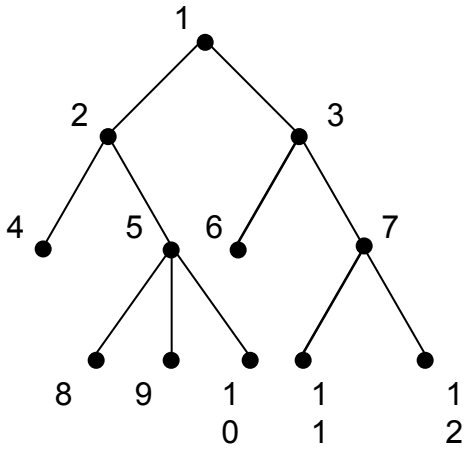
Стратегии вычисления

В СФЛП реализованы следующие стратегии:

· поиск в ширину,

+ гарантированное получение результата

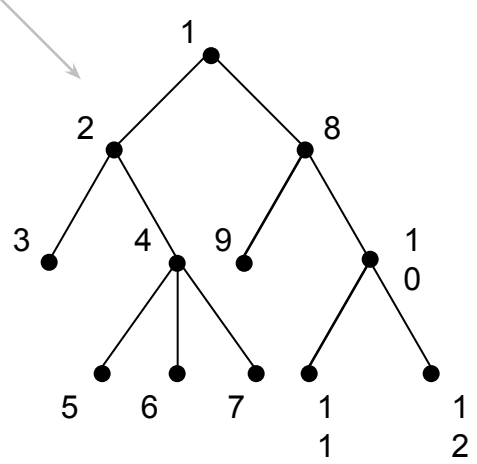
- высокие требования к вычислительным ресурсам



· поиск в глубину,

+ высокая эффективность выполнения программ

- возможность «зацикливания»



· смешанная стратегия.

+ сохранение преимуществ чистых стратегий и ослабление их недостатков

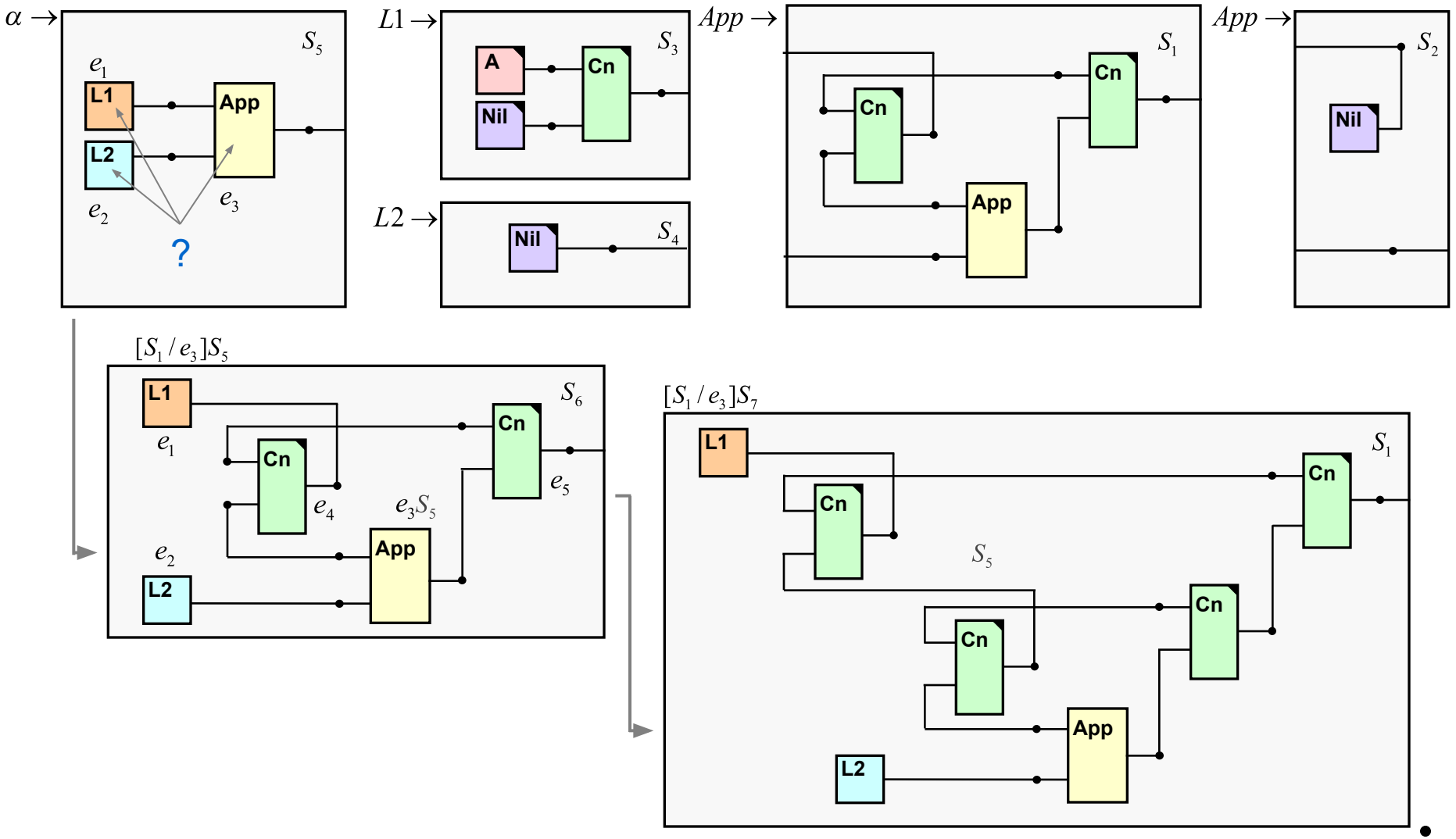
- необходимость учета специфики задачи

Реализация смешанной стратегии основана на использовании ***масок вычислимости***.

Маски вычислимости

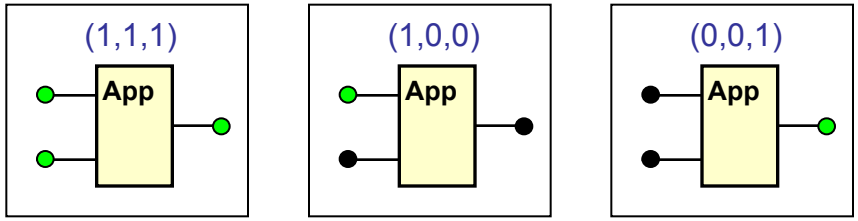
Маски вычислимости позволяют задавать требования к наличию разметки точек входов и выходов элементов для выполнения подстановки.

Пример: Вычисление <u>без использования масок</u>.

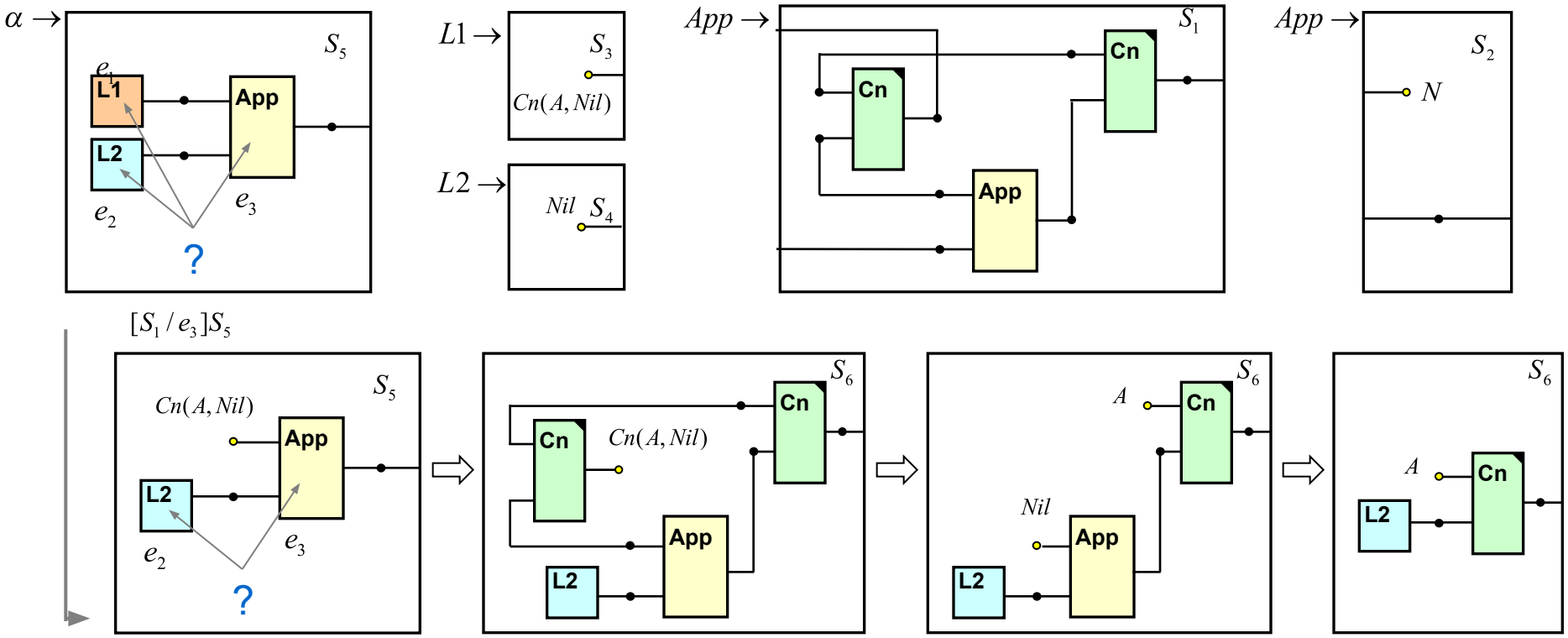


Маски вычислимости (продолжение)

Пусть для сорта App заданы маски {(1,1,1), (1,0,0), (0,0,1)}, тогда элемент сорта App может быть выбран для подстановки в следующих случаях:



Пример: Вычисление НО App с использованием масок.



Логический вывод

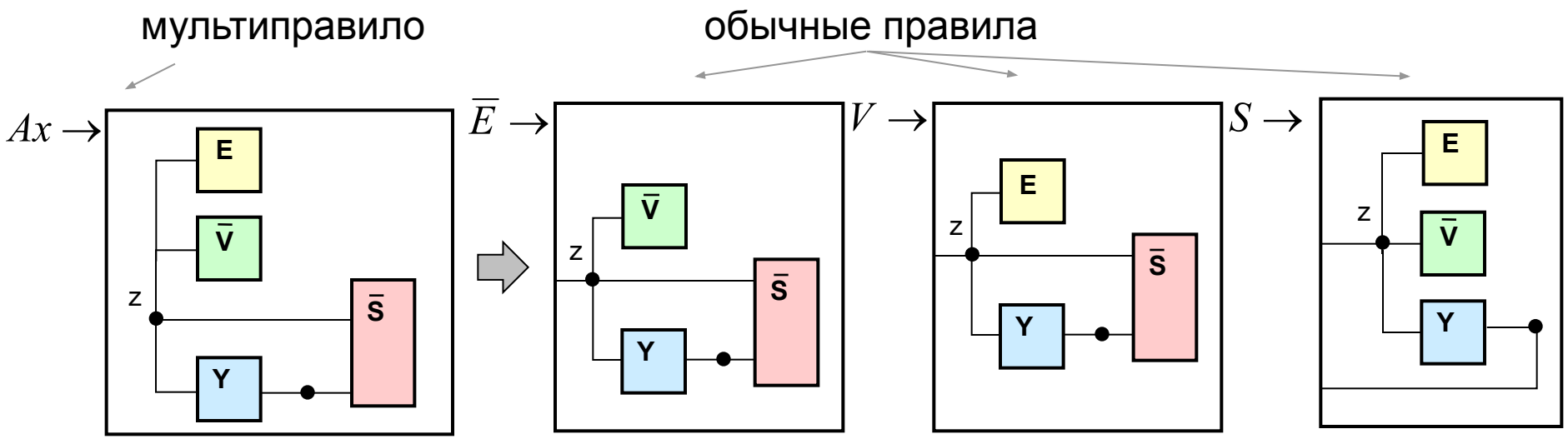
Теория НО позволяет производить доказательство общезначимости формул логики предикатов первого порядка.

Особенности процедуры доказательства:

- логические формулы представляются сетями арности $(0, 0)$,
- проводится прямое доказательство общезначимости,
- используется двойственная форма сколемизации,
- используется сетевая резолюция (реализована через подстановку).

Для повышения удобства построения программ и увеличения эффективности вычисления введено понятие *мультиправила*.

Пример: зададим формулу $\exists z \forall y (E(z) \& \neg V(z) \& \neg S(z, y))$



Реализация

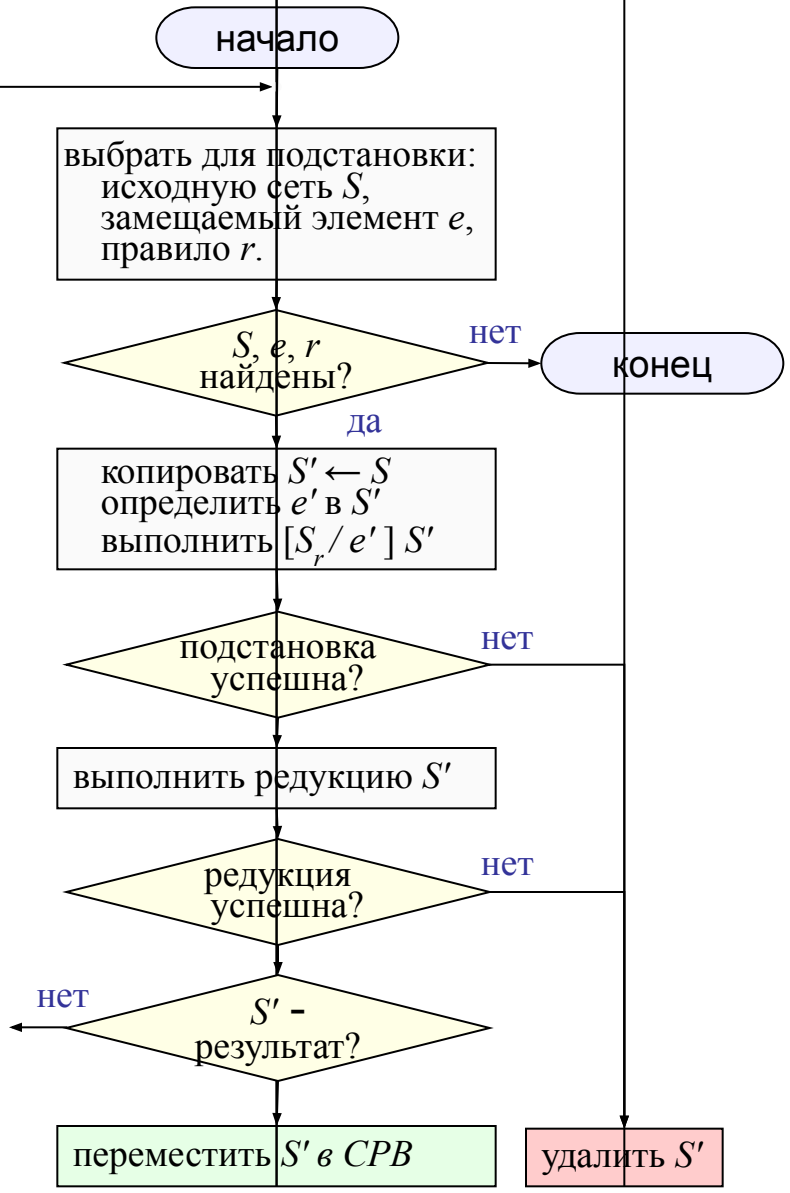
Подсистема исполнения запросов (ПСИЗ)

Абстрактная машина вычисления НО

Реализация ПСИЗ СФЛП основана на предложенной абстрактной машине вычисления НО (АМНО).

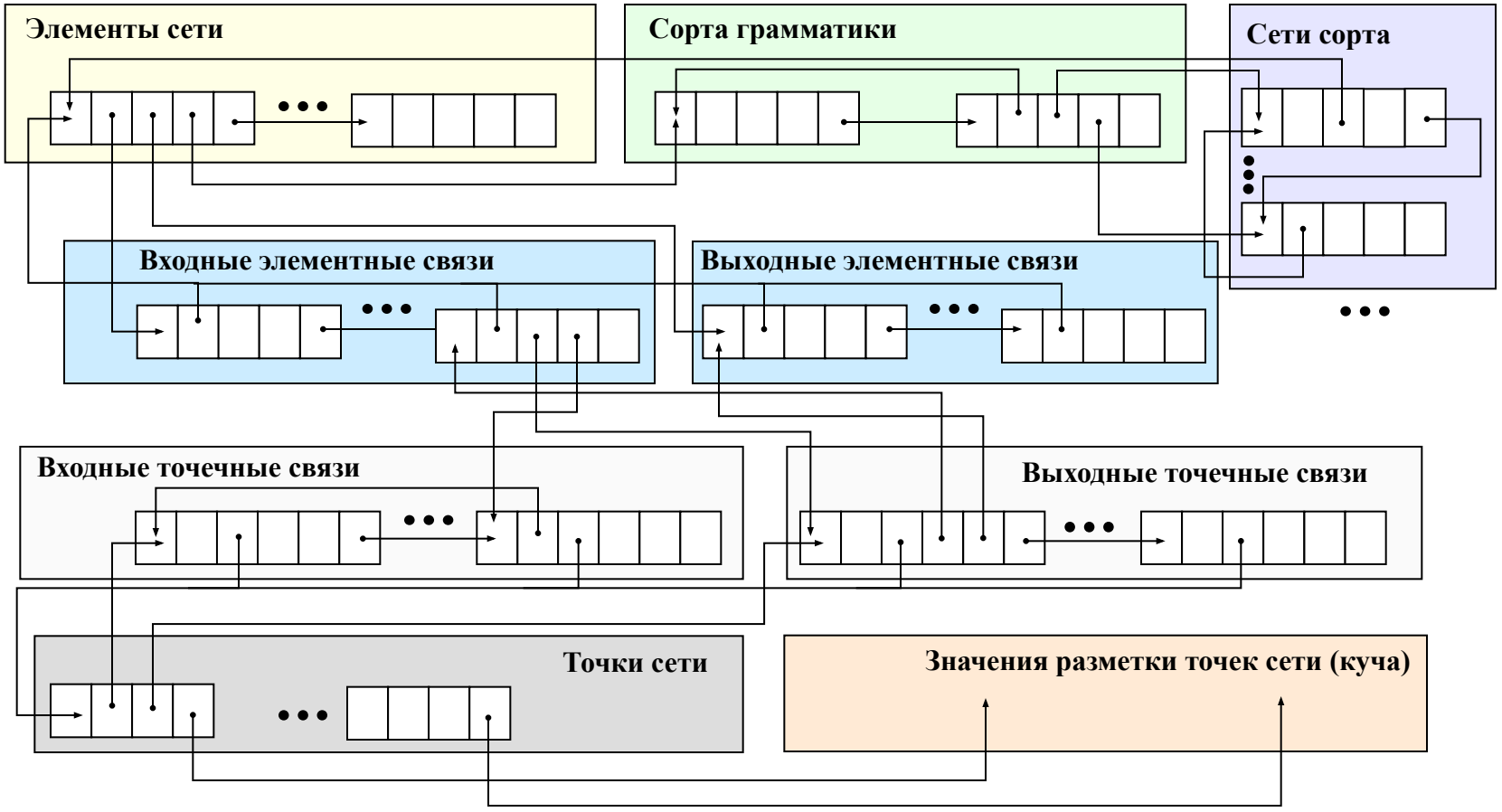
Реализация ПСИЗ СФЛП основана на предложенной абстрактной машине вычисления НО (АМНО).

- параметры подстановки
  - **FindNet** – поиск исходной сети
  - **FindElm** – поиск замещаемого элем.
  - **FindRule** – поиск правила
- подстановка
  - **Subst** – выполнение подстановки
- редукция
  - **Normalize** – редукция сети
- вспомогательные
  - **DelNet** – удаление сети
  - **CopyNet** – копирование сети
  - **MoveNet** – перемещение сети
  - **IsResult** – проверка на результат



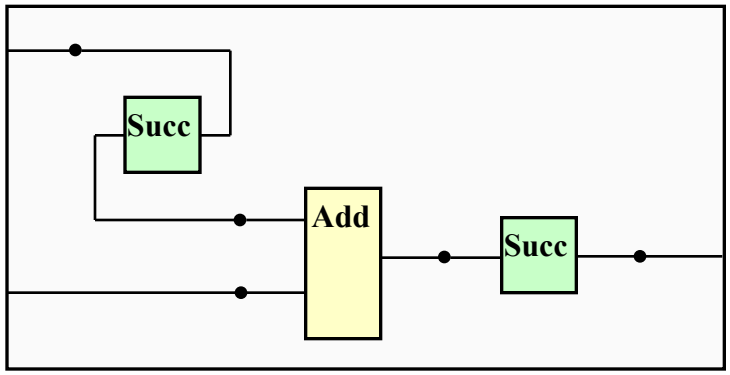
Внутреннее представление КССГ

Структура внутреннего представления сети является избыточной, но позволяет реализовать эффективные алгоритмы выполнения основных вычислительных операций.

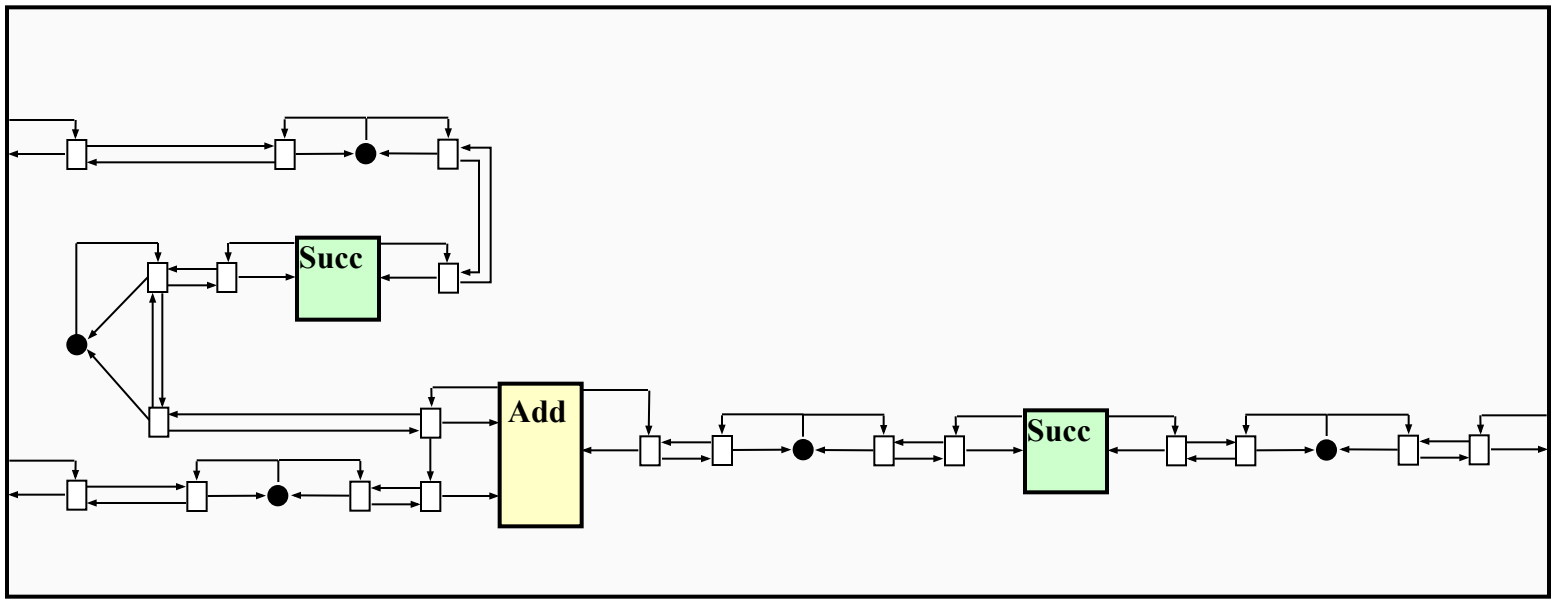


Внутреннее представление сети

· Визуальное представление



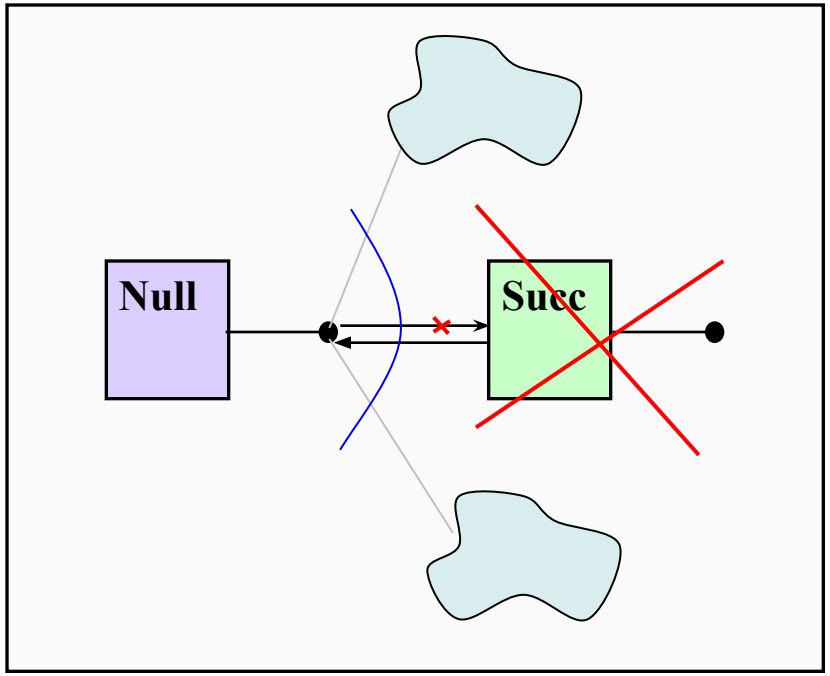
· Внутреннее представление



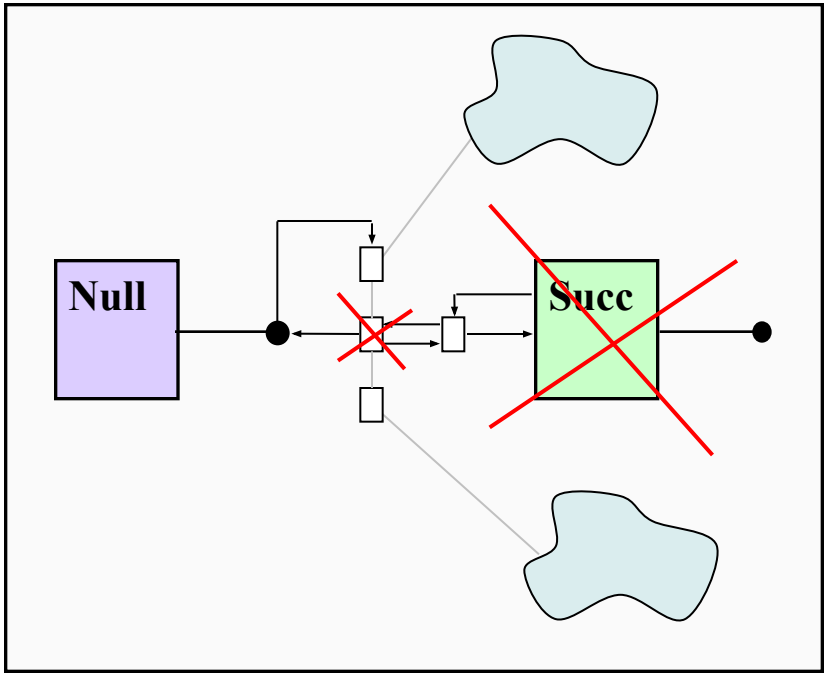
Внутреннее представление сети (продолжение)

Пример: удаление элемента сети. Требуется скорректировать ссылки точек сети на удаляемый элемент.

Простое ссылочное представление



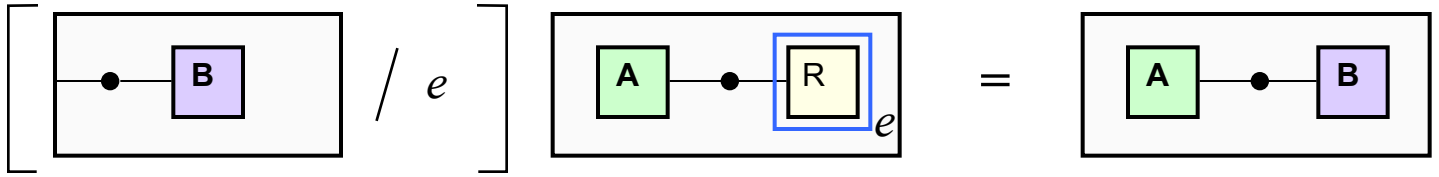
Представление ПСИЗ СФЛП



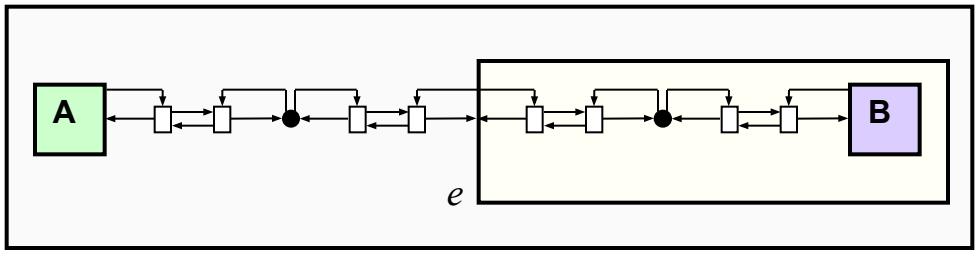
- Необходимо просматривать список связей точки с элементами сети.

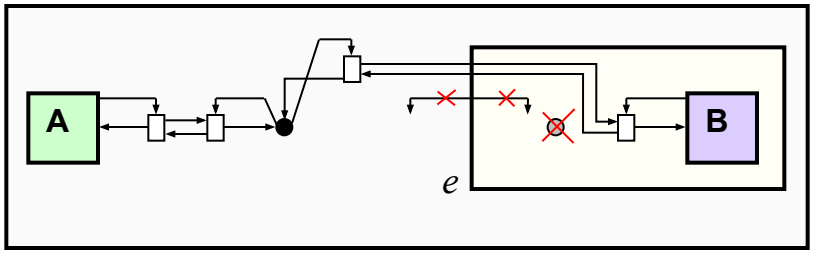
+ Точно известно, какую связь необходимо скорректировать.

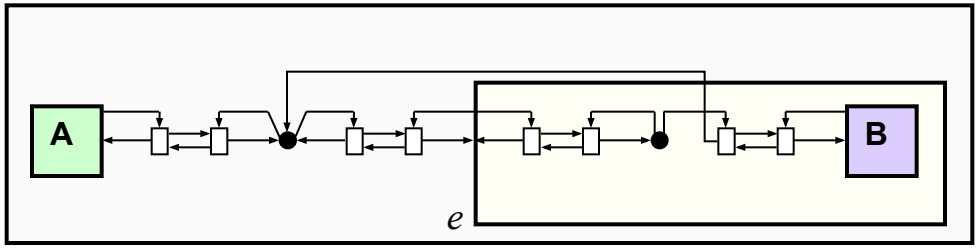
Реализация подстановки

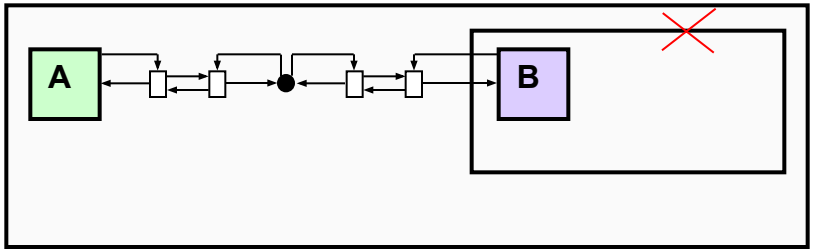


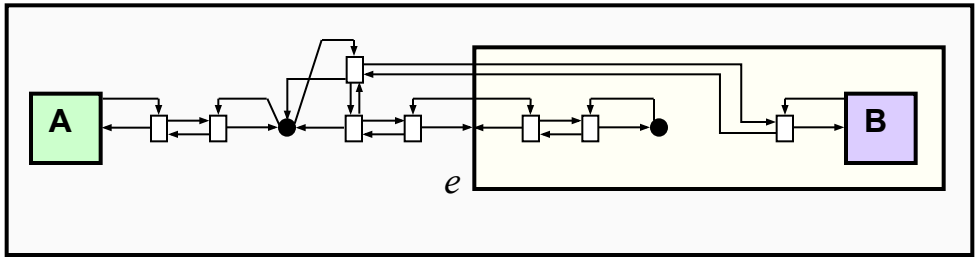
Подстановка во внутреннем представлении:



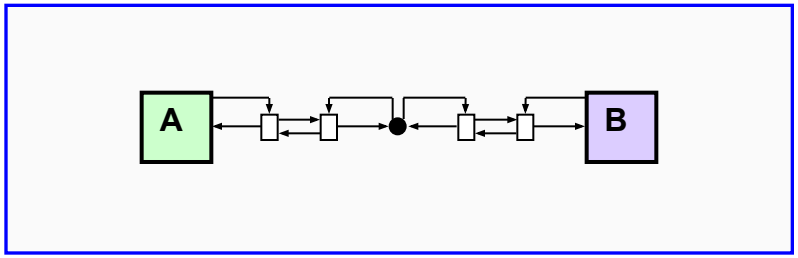








результат



Оптимизация вычислений

Реализованы следующие виды оптимизации:

- **редукция «по фронту»**

  (позволяет сократить область сети, анализируемую на применимость правил редукции − используется «ленивый» принцип анализа сети);

- **кольцевая подстановка**

  (позволяет уменьшить количество элементов сети при ее копировании для выполнения подстановки за счет реализации копирования «по необходимости»);

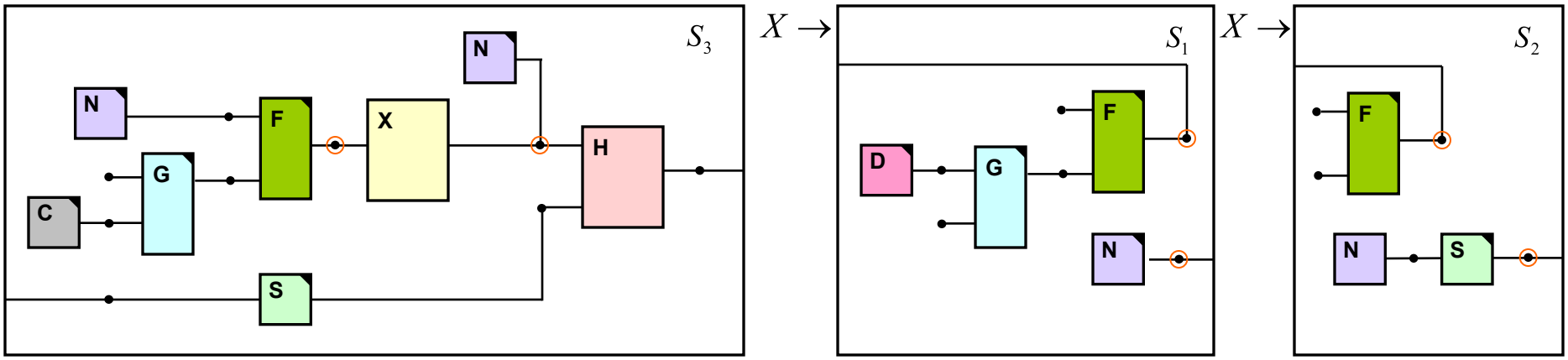
- **оптимизация последнего вызова**

  (уменьшает вычислительные затраты за счет отказа от копирования исходной сети в том случае, если для замещаемого элемента осталось применить только одно правило);

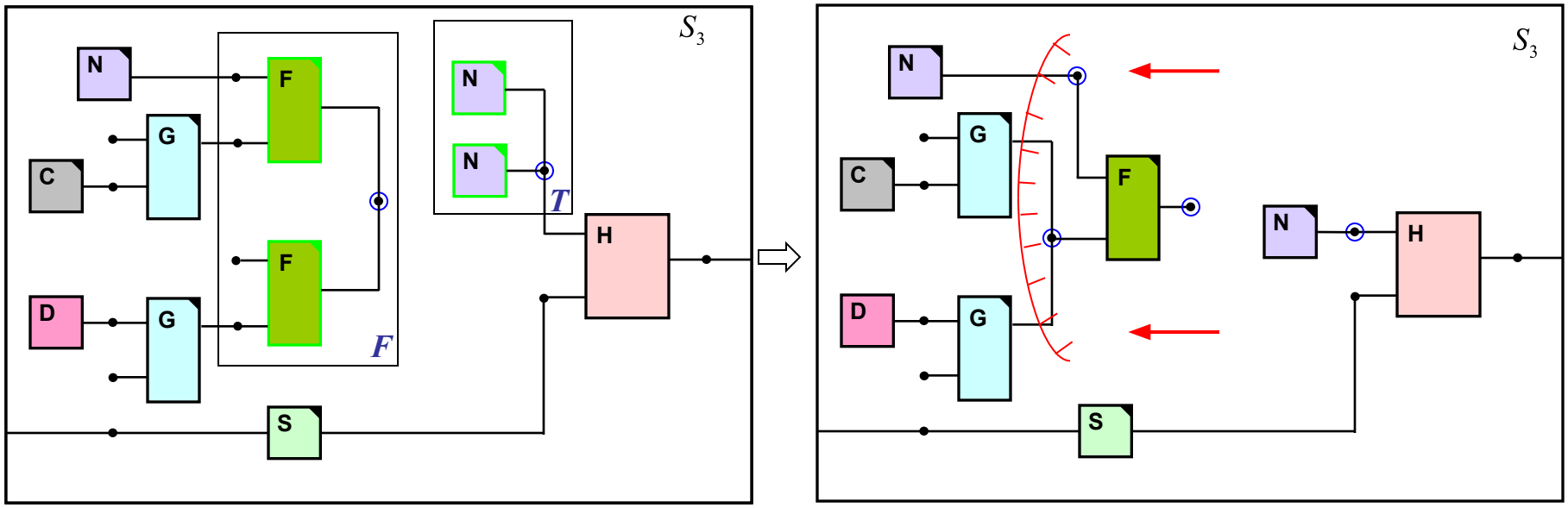
- **оптимизация порядка применения правил**

  (позволяет повысить эффективность вычисления рекурсивных НО за счет применения сначала не рекурсивного, а затем рекурсивного правила, как правило, с оптимизацией последнего вызова);

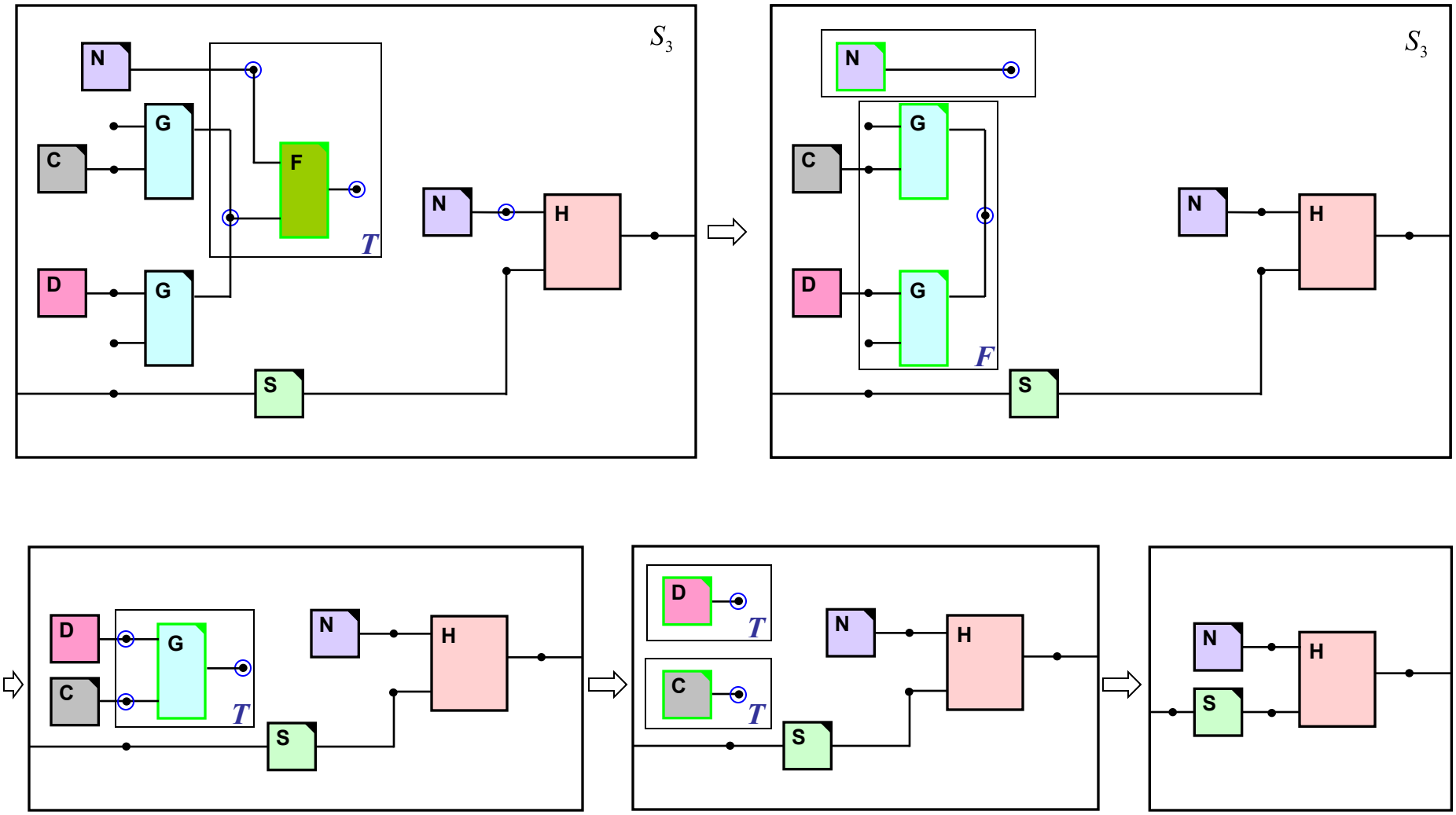
Редукция «по фронту»



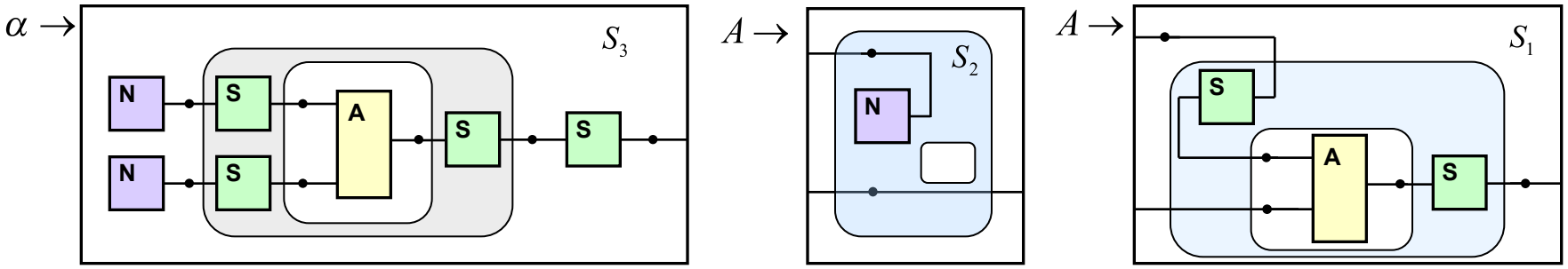
$[S_1 / e_X]S_3$



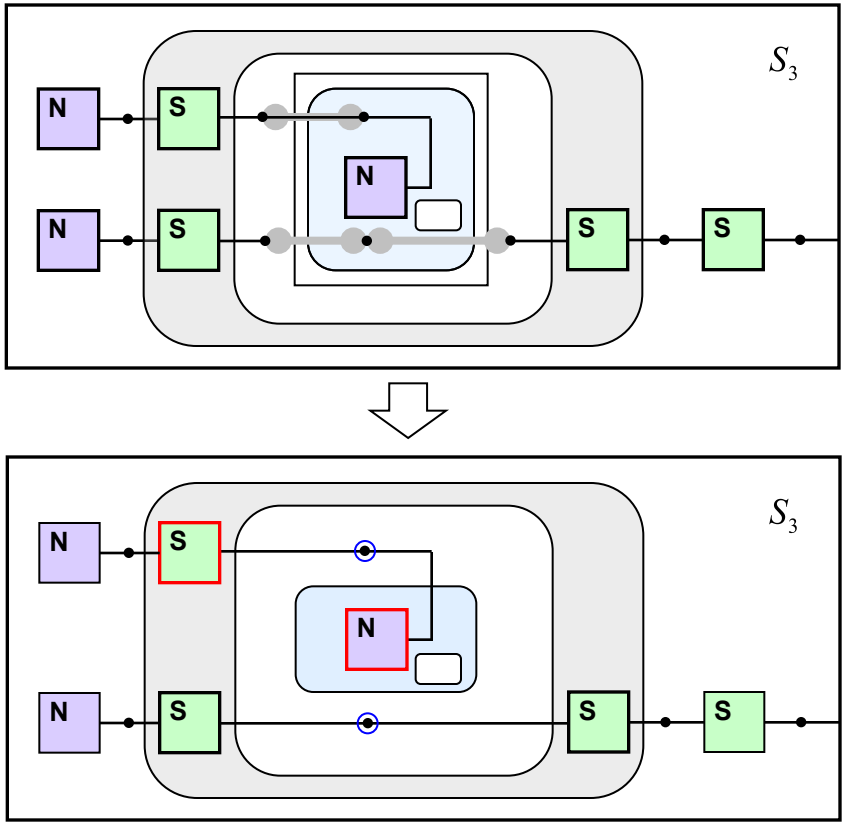
Редукция «по фронту» (продолжение)



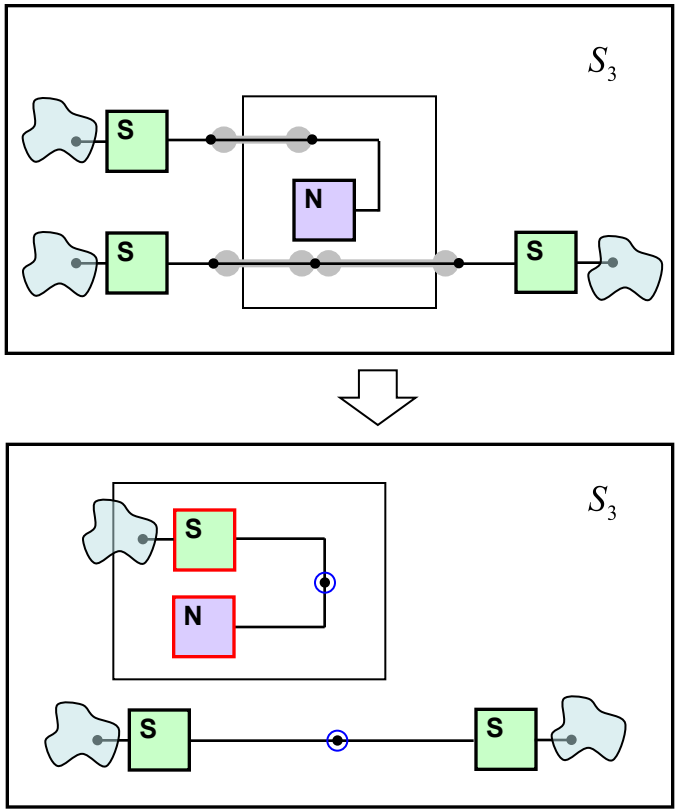
Кольцевая подстановка



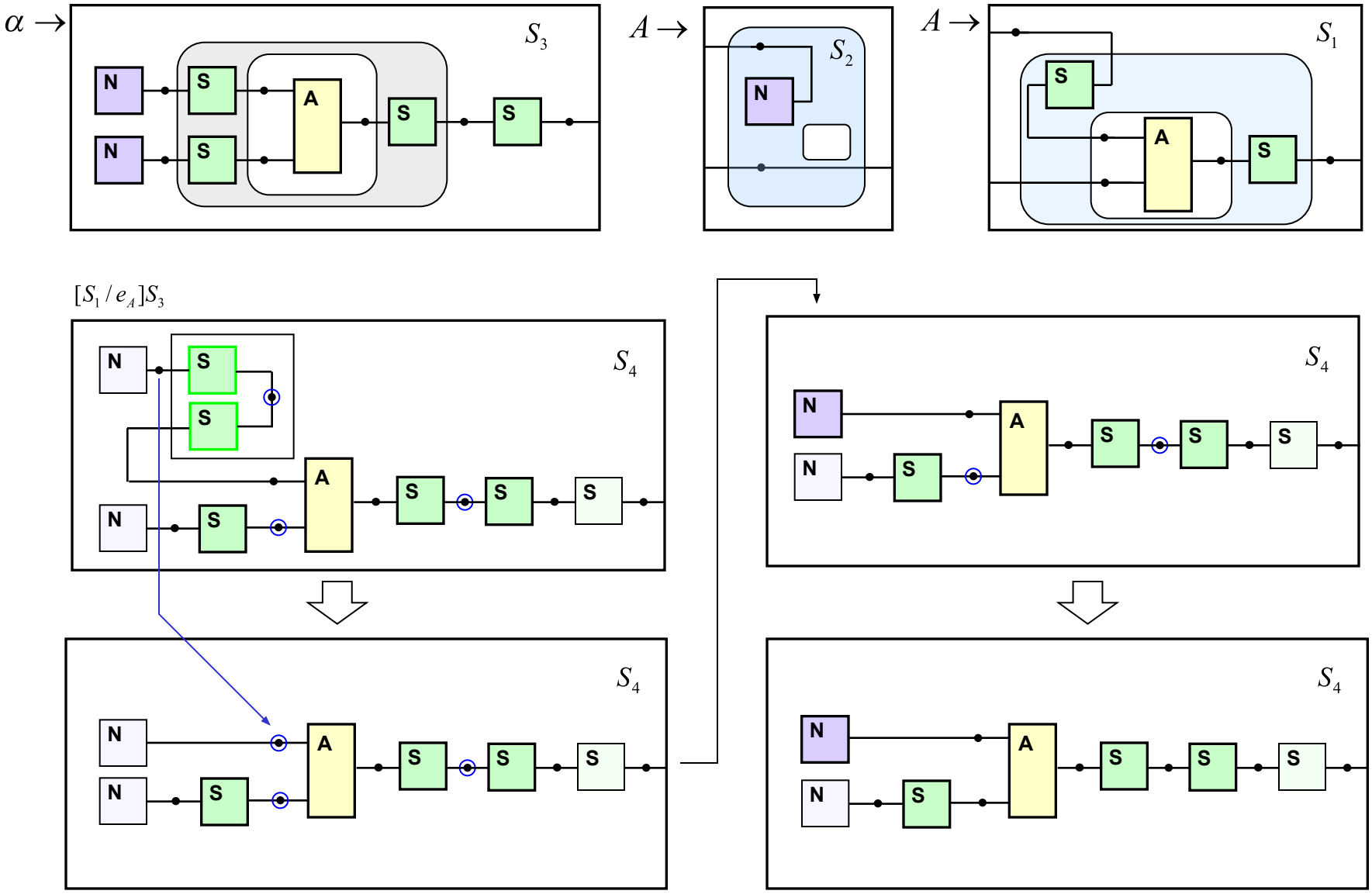
Обычная подстановка:



Кольцевая подстановка:



Кольцевая подстановка (докопирование контекста)



Системные типы данных и системные НО

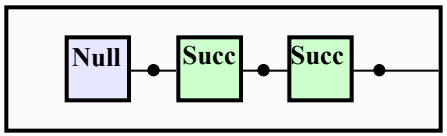
Для повышения производительности СФЛП и удобства ее использования введены системные типы данных и системные НО* для их обработки.

***Системные НО поддерживаются только в режиме вычисления с разметкой.**
Системные НО можно рассматривать как НО с внешней интерпретацией.

Системные типы данных:

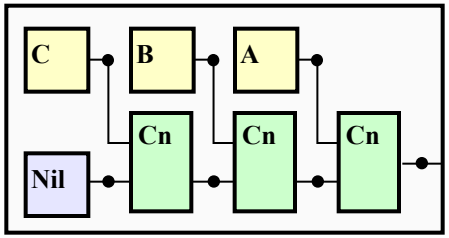
· натуральные числа,

Пример: **2**



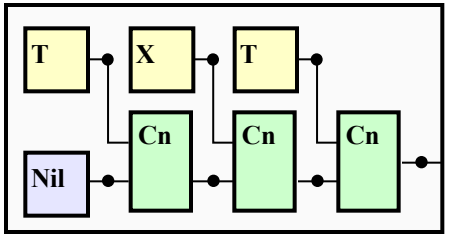
· списки,

Пример: **[A,B,C]**



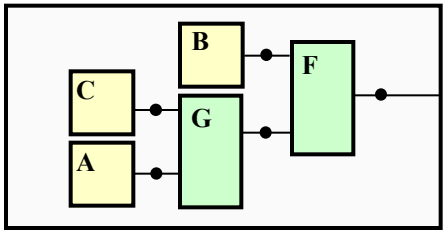
· строки,

Пример: **"TXT"**



· термы.

Пример: **F(B,G(C,A))**



Системные НО:

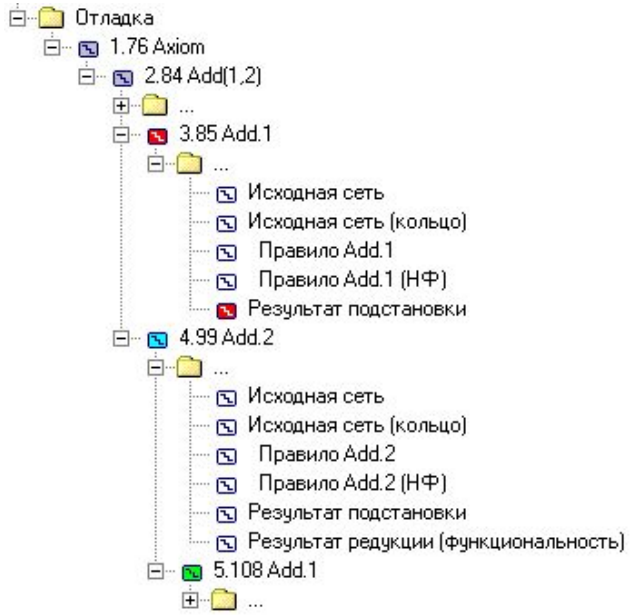
Имя НО	Комментарий
Натуральные числа	
123	Генератор числа 123
Add	Сложение
Sub	Вычитание
Mult	Умножение
Div	Деление
>	Оператор «больше»
>=	Оператор «больше либо равно»
<	Оператор «меньше»
<=	Оператор «меньше либо равно»
==	Оператор «равно»
=/=	Оператор «не равно»
Особые НО	
=	Унификация термов
\\=	Термы не унифицируемы
is	Предикат is Пролога
write	Выдача текстового сообщения
flogol_spec	Оптимизация компиляции

Отладка программ

Для отладки программ в СФЛП предусмотрены следующие средства:

- **Дерево отладки**

  (отображает построение дерева вывода в процессе вычисления запроса),



- **Системное НО $Write**

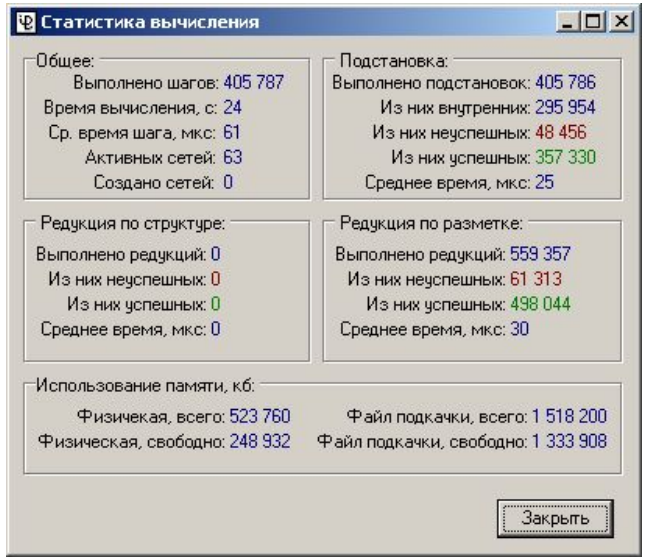
  (отображает разметку входных точек элемента сети сорта $Write в специализированном окне редактора),

- **Системное НО $Show**

  (отображает сеть в момент начала вычисления элемента сети сорта $Show),

- **Окно статистики**

  (отображает основные параметры процесса вычисления).



Импорт программ языка Пролог

Импорт программ, написанных на языке логического программирования Пролог повышает универсальность СФЛП и обеспечивает поддержку компиляции входного языка СФЛП S-FLOGOL.

Ограничения:

· отсутствие синтаксического анализа,

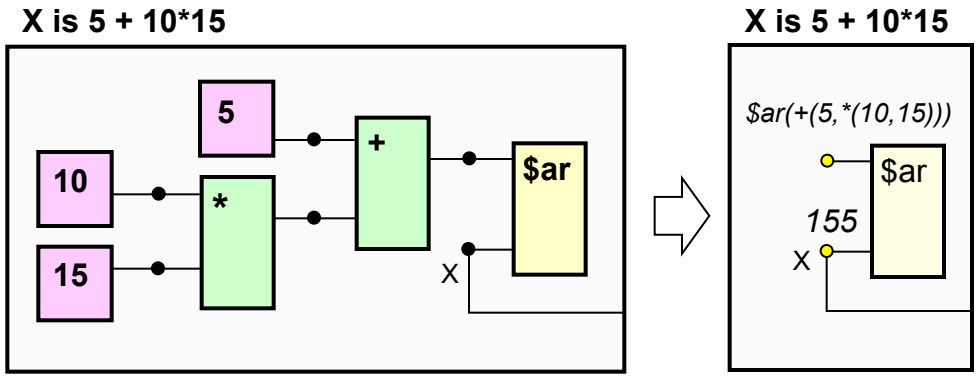
· отсутствие поддержки отсечения и отрицания,

· отсутствие поддержки полиморфизма предикатов,

• из системных предикатов поддерживаются:

=\=, = =, >=, **is**, **write**, **fail**, **[|]**, = и \=.

Пример преобразования предиката **is**:



Пример конструкции **[|]**:

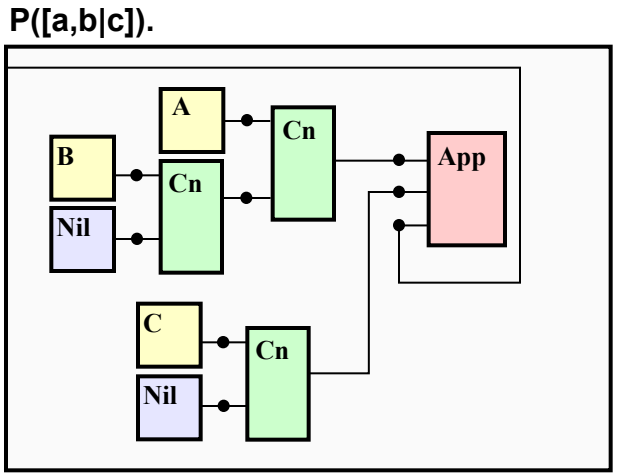
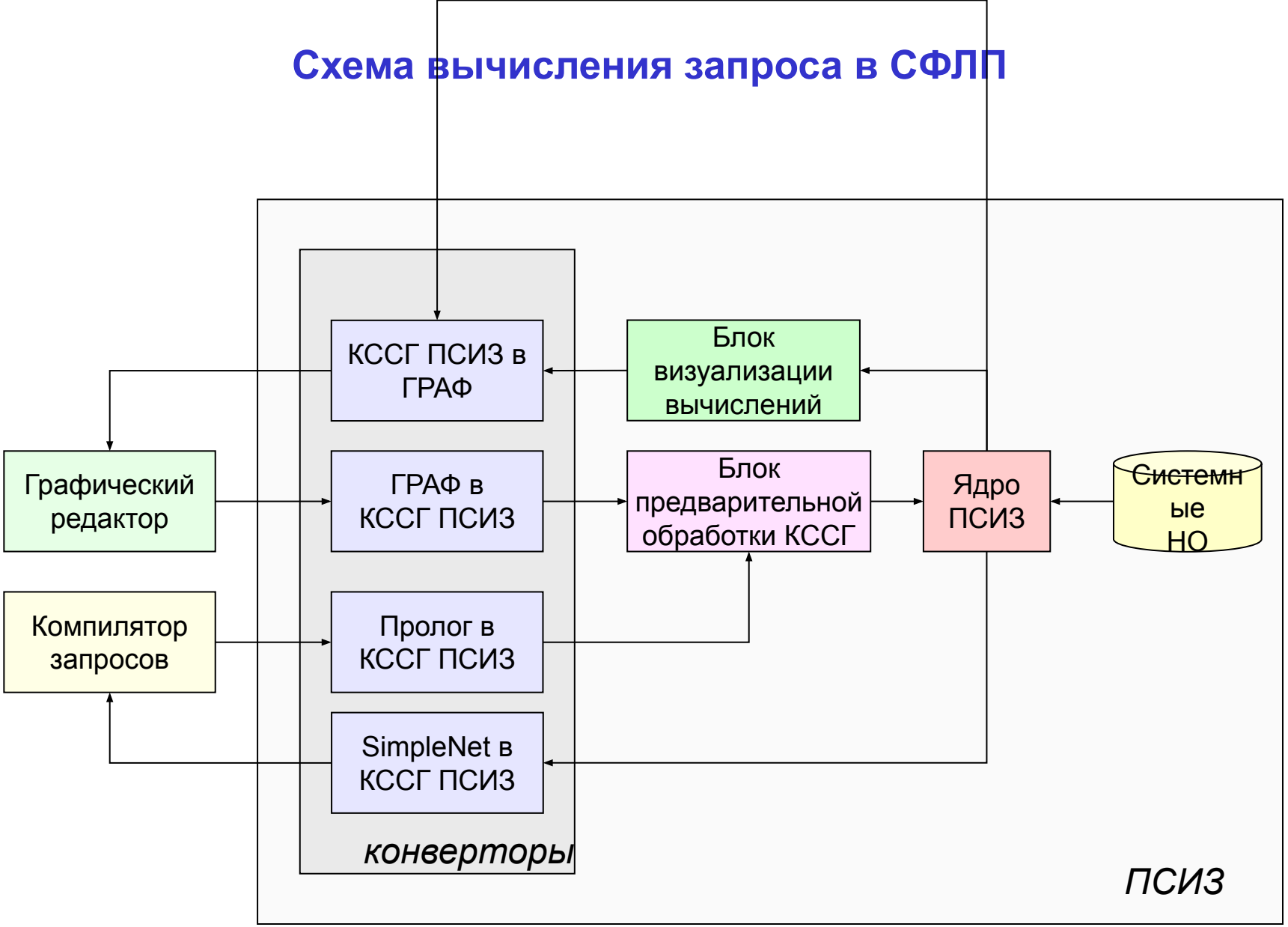


Схема вычисления запроса в СФЛП



Интерфейсные средства

Графический редактор

Технология графического построения программ (ТГПП)

ТГПП предназначена для обеспечения корректности формируемой программы на каждом шаге ее построения.

Формирование КССГ:

- добавление элемента базиса
- определение свойств элемента базиса
- удаление элемента базиса
- добавление правила грамматики
- удаление правила грамматики
- выделение запроса к системе

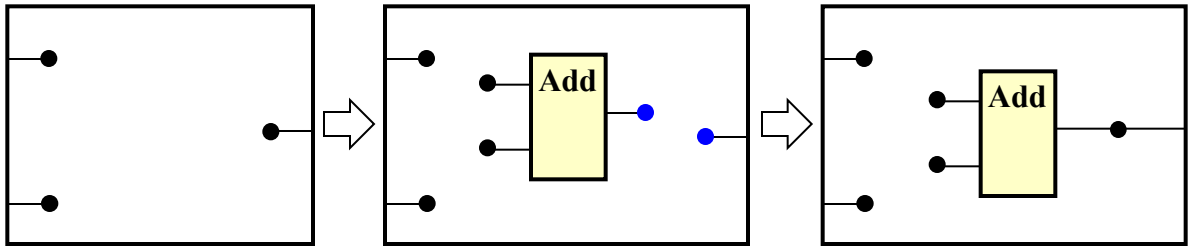
Формирование сетей:

- добавление элемента
- удаление элемента
- отождествление точек
- «расщепление» точек
- добавление ребра *dif*-графа
- удаление ребра *dif*-графа

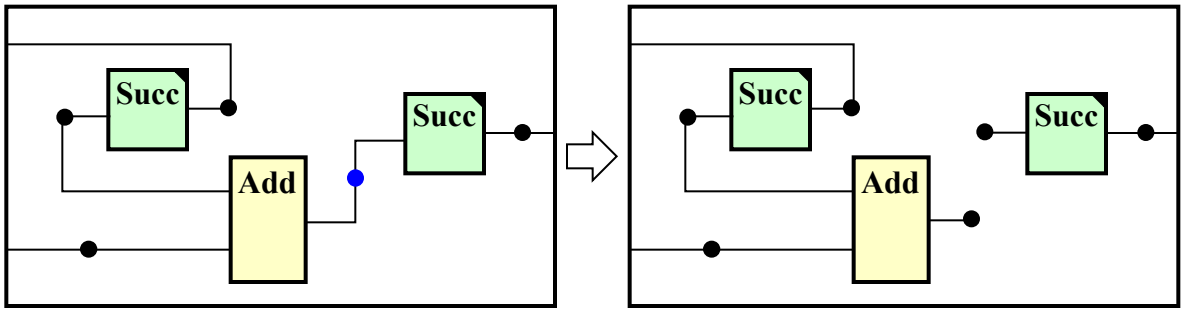
Для каждой технологической операции формально определено изменение сети после ее выполнения.

Формирование сетей

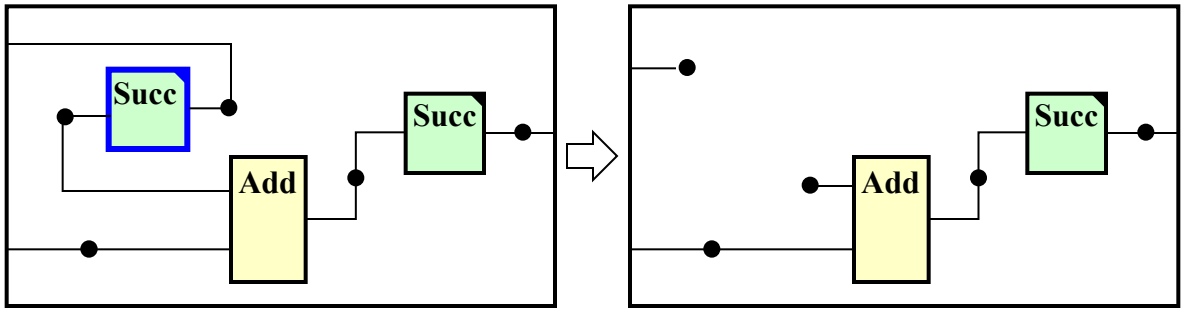
Пример добавления элемента сети и отождествления точек:



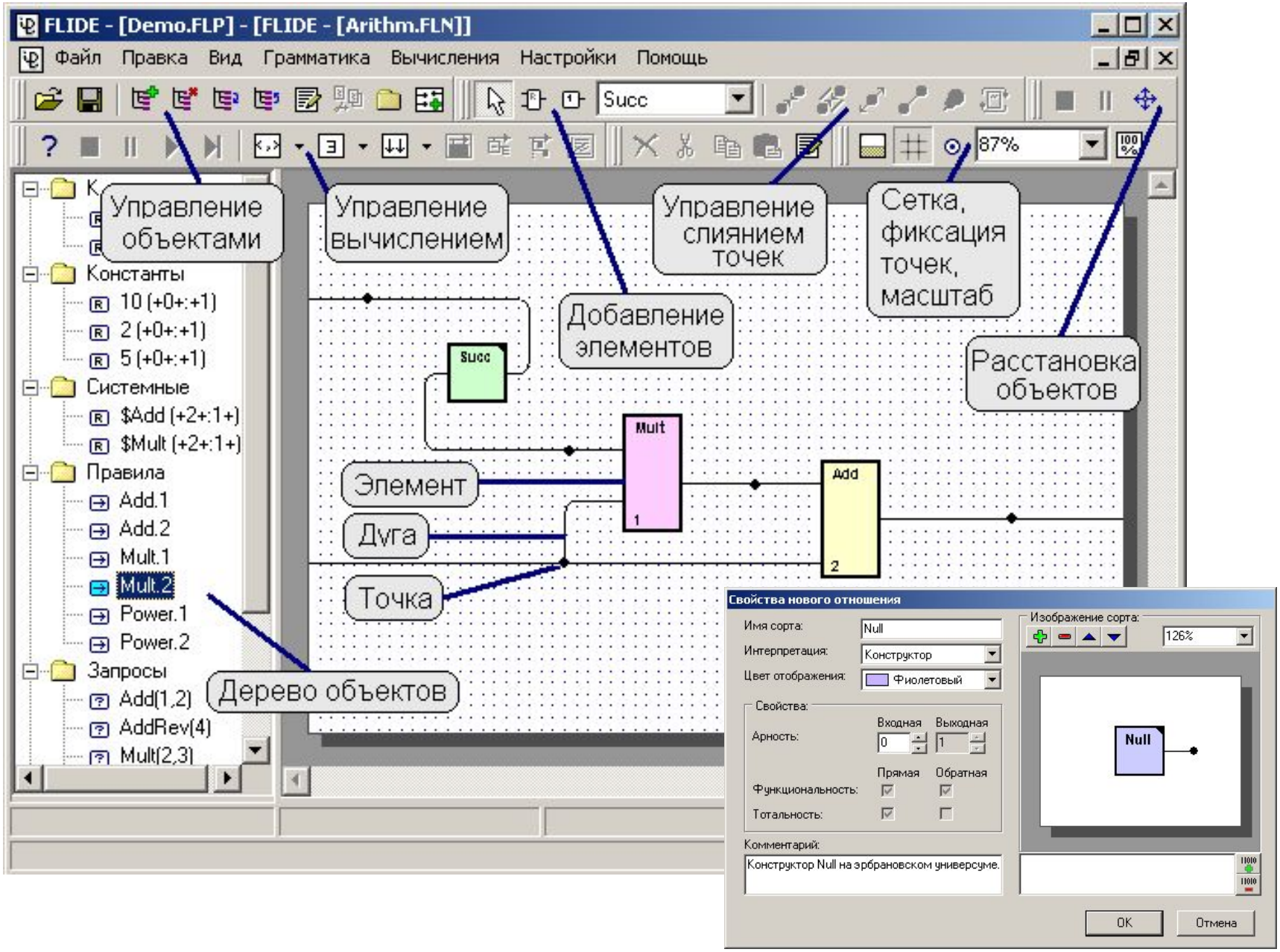
Пример разрушения связей («расщепления») точки сети:



Пример удаления элемента сети:



Вид окна графического редактора



Архитектура графического редактора

Автоматическая расстановка сетей

Для отображения результатов вычисления в сетевом виде требуется по внутреннему представлению сетей строить их графическое отображение.

Этапы построения изображения:

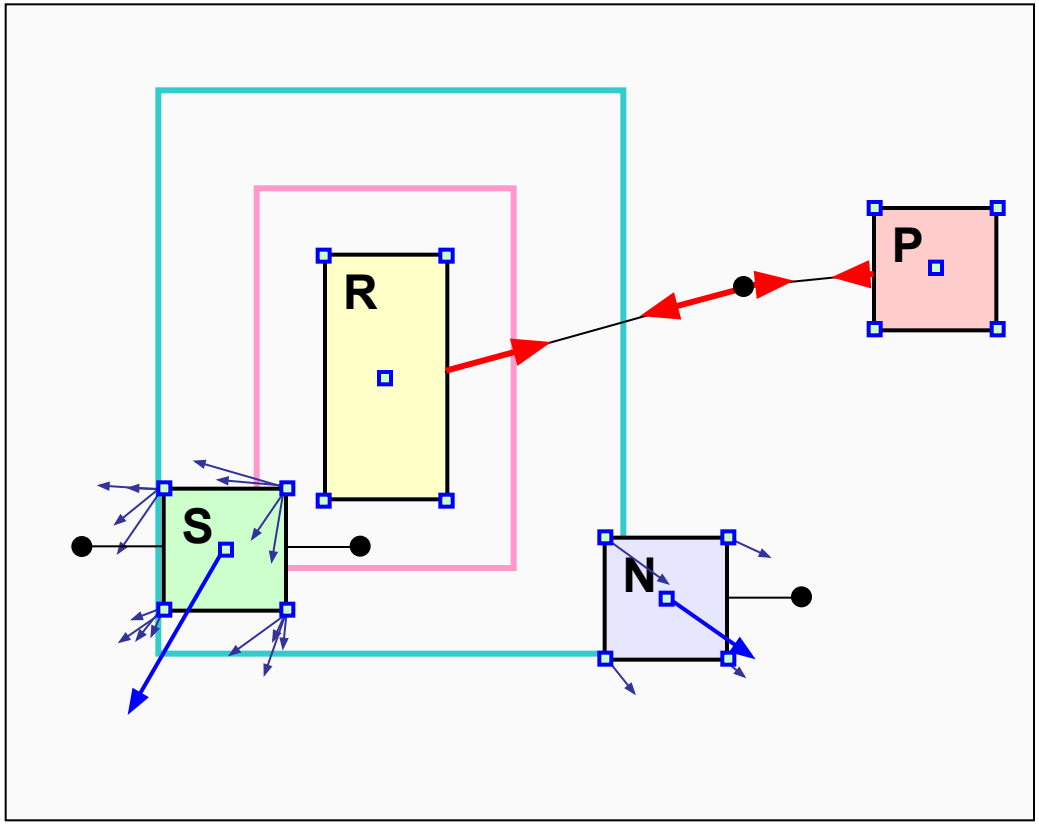
- предварительная расстановка элементов
- основная расстановка методом силовых воздействий
- обработка полученного изображения

Особенности реализации основного этапа:

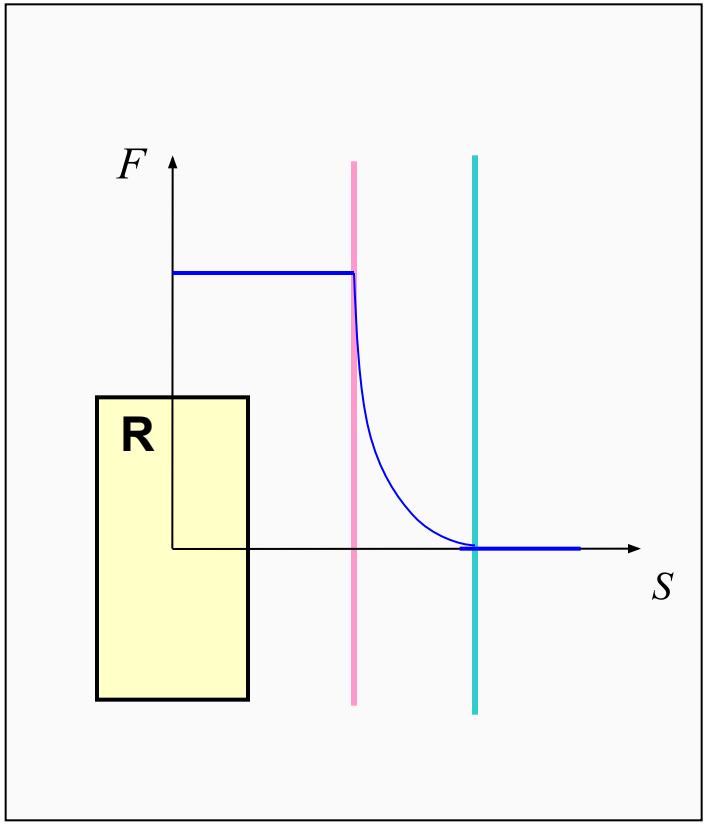
- учет геометрических размеров объектов сети
- использование метода оболочек для ускорения расстановки
- особое распределение сил отталкивания
- обнаружение и гашение циклических колебаний
- моделирование вязкости среды

Автоматическая расстановка сетей

Рамки, силы отталкивания и притяжения:



Распределение сил отталкивания:



Основные результаты работы

1. Введено понятие размеченной сети. Предложены правила редукции для размеченных сетей.
2. Предложена абстрактная машина вычисления НО. Разработаны алгоритмы линейной сложности для выполнения подстановки и редукции.
3. Предложены и реализованы методы повышения эффективности вычисления НО – редукция «по фронту», кольцевая подстановка, оптимизации последнего вызова, маски вычислимости и мультиправила.
4. На основе предложенных методов и алгоритмов выполнена программная реализация подсистемы исполнения запросов СФЛП. Реализован импорт программ на языке Пролог.
5. Разработана технология графического построения программ (ТГПП), гарантирующая корректность формируемой программы.
6. Предложены оптимизационные модификации и реализован силовой метод автоматического формирования изображения сетей.
7. Разработана архитектура и выполнена программная реализация графического редактора СФЛП, поддерживающего ТГПП.
8. Подсистема исполнения запросов и графический редактор интегрированы в реализованную совместно с Бебчиком Ал.М. СФЛП.

Спасибо за внимание!